

Manifold-Based Robotic Workspace Formulation: Path Planning and Obstacle Avoidance

by

Sindhu Radhakrishnan

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

© Sindhu Radhakrishnan, Ottawa, Canada, 2022

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Autonomous robots navigate unknown and known environments. Regions of the environment that are not suitable for navigation may be in the form of stationary obstacles, limitations of the robot, unfavourable terrain/structure of the environment and sudden appearance of unaccounted obstacles. In the context of robotics for known environments such as an automated industrial environment or a warehouse, the environment is known apriori. That is, locations of regions not favourable for path planning, called static constraints, are known. However, there is still a possibility of encountering obstacles that are not part of the known environment, called dynamic constraints. They could be human beings, other robots (either part of them or as a whole), components belonging to the environment (boxes, cables, tools, manufactured products) and anticipated dangers (spills, compromised structures). So, path planning in such an environment consists of the following general two steps. First, a path between the desired source and target representation is generated. Second, segments of the path are evaluated for any encounter with constraints.

The two general steps are accomplished differently by different algorithms, each with merits and demerits. The differing success of approaches used, depends on how the environment is represented. In methods that aim to save memory, the map is generated by sampling; so, the map is only as good as the sampling method. Then, the produced path has to be periodically checked for whether a segment of the path is truly constraint-free (static and dynamic). Sometimes, the method may stagnate at a non-optimal path, or may even not be able to complete the process of finding one. Alternatively, in approaches that store a detailed grid based map, changes in terrain and structure are expensive memory costs. The problem thus remains open, with the aim of representing the map with only constraint-free, navigable regions and generating paths as a reaction to, or in anticipation of, encountering new constraints.

To solve this open problem, the Constraint Free Discretized Manifold based Path Planner is proposed. The algorithm divides the problem into two parts: the first focuses on maximizing knowledge of the map using manifolds, and the second uses homology and homotopy classes to compute paths. The first step is instrumental in constructing a complete representation of the navigable space as a manifold, free of constraints known apriori. Using topological tools, this representation is shown to have favourable properties, such that any path generated on it is guaranteed to be constraint-free. So, on this constraint-free manifold, no segment of the path has to be explicitly evaluated for a collision with a known constraint. It is shown that alternative spaces associated with the environment also share the same properties under certain conditions. Thus, one can transform the constraint-free path to other equivalent spaces.

The second step deals with new knowledge of constraints that render the originally produced path as invalid. Using homology and homotopy, paths on the original manifold can be recognized and avoided by tuning a parameter, thus resulting in an alternative constraint free path. By operating on the discretized constraint free manifold, path classes characterize uniqueness of paths around constraints. This designation provides the ability to avoid a specific path class, should that not be desirable in light of newly encountered constraints. Then, the algorithm can be queried for a new path class free of constraints, without any explicit

modification of the original map created and even when there is no physical indication of constraints. Tuning may be performed to produce more than one alternative path to be on the manifold.

The proposed algorithm is seen to produce paths on the manifold with an average percentage path length deviation of 29.6%, which is over 70% less than those produced by sampling algorithms. The proposed algorithm also provides an increase in retention of usable samples by a margin of at least 30%, when compared with sampling algorithms. This is while maintaining on-par run times at worst, and better run times in most cases, when evaluated against other algorithms. These general trends hold true even when the proposed algorithm is utilized to generate alternative paths. Any deviation in path length related trend is seen only when a query is made to generate an alternative path that avoids the shortest path previously generated; a feature not present in sampling algorithms.

Acknowledgements

To Dr. Wail Gueaieb,

I am indebted to you for being just the kind of mentor I needed. The gentle nudge in the right directions, the steadfast confidence in my abilities and teaching me to hold myself accountable to high standards are some of what I cherish your mentorship for. You set a phenomenal example of someone who thinks and communicates clearly, especially when you dissect a problem into its components. From the first robotics class that I was your student in, to the last one I was your teaching assistant for, you have never stopped teaching me. I truly aspire to learn the way you do, and here's to hoping that we always share more such discussions. I not only enjoyed your technical tutelage, but I truly found a guide in you. Thank you, sir.

To Dr. Lili Tabrizi,

You were the first person to introduce control systems to me, and it was mesmerizing. Your singular comment about my senior design presentation is what started my journey towards graduate school in engineering. At every point after, you continued to be a beacon of light for me, including the doctoral program. Thank you for continuing to be an amazing role model.

To my patient and rock-solid family: Amma, Appa and Moo,

I have to offer an eon's worth of gratitude to all of you. You continue to show me what unconditional love is, and I am forever grateful to you. You patiently continue to love me like you always did, despite how much I've changed. It is miraculous for one family to have this much patience towards one's antics and tantrums; sorry about that! I hope to become the person you once knew. This journey of changing back to my old self shouldn't be as tumultuous; so, sit back, relax and enjoy the ride. And, I don't think I said this as much as I should have- I love you!

To Kripa,

As my funny, giggle-inducing sister, thank you for always making me feel like we're sneaking a giggle in the middle of a quiet class in the seventh grade! You have always been by my side irrespective of where we were geographically. From early on in school, through homesick days in my freshman year at university to now, you're always there for me, and I'm always thankful for you. So really, what I'm trying to say is, and I love kripsy because kripsy rox!

To Rama mami and Sundar mama,

Thank you for morphing into the role of my parents when I needed it. You continue to treat me as your daughter, and I could not have come this far without you. I am always grateful for the nurturing love you showed me; love you!

Dedication

This thesis is dedicated to my 12 year old self, who thought that working towards a Ph.D was the coolest thing to do in life.

To Sindhu- If you're reading this and are somehow 12 years old (time machine/ time turner), just say yes to Hogwarts. At some point, Hogwarts and the Ph.D will seem equally fictional.

Table of Contents

List of Tables	x
List of Figures	xi
Abbreviations	xxv
List of Latin Symbols	xxix
List of Greek Symbols	xxxi
1 Introduction	1
1.1 Background	1
1.2 Motivation	2
1.3 Objectives and Contributions	3
1.4 Thesis Organization	4
1.5 Scholarly Outcome	4
2 Literature Review	5
2.1 Set/Space of Points	6
2.1.1 Sampling-Based Methods	7
2.1.2 Optimization-Based Methods	17
2.1.2.1 Artificial Potential Fields	17
2.1.2.2 Model Predictive Control	22
2.1.2.3 Evolutionary Algorithms	23
2.1.2.4 Navigation Functions	24
2.2 Manifolds	28
2.2.1 Parametrizing Manifolds in Path-Planning Space	29

2.2.2	Charting of Manifolds	44
2.3	Path Planning with Dynamic Constraints	46
2.3.1	Model Based Methods	47
2.3.2	Homotopy Based Methods	60
2.4	Summary	85
3	Methodology	90
3.1	Notations and Definitions	90
3.2	Methodology for Algorithm Formulation: Constraint-Free Discretized Manifolds-based Path Planner (CFDMPP)	116
3.2.1	Formulation of a Constraint-Free Manifold (CFM)	117
3.2.1.1	Primary and Secondary Path Planning Spaces (PPSs)	117
3.2.1.2	Dimensions of Path Planning Spaces (PPSs)	117
3.2.1.3	Bounds of Path Planning Spaces (PPSs)	118
3.2.1.4	Mapping Between Path Planning Spaces (PPSs)	118
3.2.1.5	Information About Static Constraints (SCs)	119
3.2.1.6	Potential Dynamic Constraints (DCs)	125
3.2.1.7	Choosing the Appropriate Space Where to Commence Path Planning	126
3.2.1.8	Complete Definition of Chosen Path Planning Space (PPS)	128
3.2.1.9	Constructing a Constraint-Free Manifold (CFM) in the Chosen Path Planning Space (PPS)	129
3.2.2	Discretizing the Constraint-Free Manifold (CFM) and Path Planning	141
3.2.3	Path Transformation Between Path Planning Spaces (PPSs)	145
3.2.4	Summary	146
4	Results and Analysis	147
4.1	Stage 1: Path planning on the CFDM	148
4.1.1	Setup	148
4.1.2	Results	150
4.1.2.1	Case 1	150
4.1.2.2	Case 2	152
4.1.2.3	Cases 3 and 4	152
4.1.3	Analysis	154

4.2	Stage 2: Path planning with DCs on the CFDM	157
4.2.1	Setup	157
4.2.2	Results and analysis	163
4.2.2.1	Case 1	163
4.2.2.2	Case 2	178
4.2.2.3	Cases 3 and 4	192
4.3	Stage 3: Case Studies From Simulations in a Warehouse Environment	219
4.3.1	Setup	219
4.3.2	Results and Analysis	221
4.4	Conclusion	225
5	Conclusion and Future work	234
5.1	Future Work	237
	References	238
	APPENDICES	253
A	Stage 2: Case 1	254
B	Stage 2: Case 2	275
C	Stage 2: Case 3	296
D	Stage 2: Case 4	317

List of Tables

2.1	Comparison of path planning algorithms in the context of SCs	88
2.2	Comparison of path planning algorithms in the context of DCs	89
4.1	CFDMPP Parameters	162
4.2	Case 1 : Percentage of Path Length Deviation	171
4.3	Case 1 Ratio of observed path length to ideal length	171
4.4	Case 1 Percentage of usable samples	172
4.5	Case 1 Percentage Rejected Samples	172
4.6	Case 1 Run time	173
4.7	Case 2 : Percentage of Path Length Deviation	185
4.8	Case 2 Ratio of observed path length to ideal length	185
4.9	Case 1 Percentage of usable samples	186
4.10	Case 1 Percentage Rejected Samples	186
4.11	Case 2 : Run time	187
4.12	Case 3 Percentage of Path Length Deviation	200
4.13	Case 3 Ratio of observed path length to ideal length	200
4.14	Case 3 Percentage of usable samples	201
4.15	Case 3 Percentage Rejected Samples	201
4.16	Case 3 Run time	202
4.17	Case 4 Percentage of Path Length Deviation	212
4.18	Case 4 Ratio of observed path length to ideal length	212
4.19	Case 4 Percentage of usable samples	213
4.20	Case 4 Percentage Rejected Samples	213
4.21	Case 4 Run time	214
4.22	CFDMPP Parameters	221

List of Figures

2.1	Flowchart of Rapidly exploring Random Tree (RRT)	8
2.2	Paths taken and available for a robot using Rapidly exploring Random Tree (RRT) [112]	9
2.3	Flowchart of Probabilistic Road Map (PRM)	10
2.4	Transition-based RRT (T-RRT) on a cost-map [66]	13
2.5	Illustration of uniform and Goal-Oriented sampling in different sampling environments; uniform sampling is seen in (a) & (c) and Goal-Oriented sampling in (b) & (d) [5]	14
2.6	Illustration of uniform and goal-oriented sampling in the configuration space of a manipulator [72]	15
2.7	Illustration of uniform and goal-oriented sampling in the workspace of a manipulator [72]	16
2.8	Paraboloidal and Conical attractive potentials [154]	18
2.9	Repulsive potential [146]	19
2.10	Equipotential contours of a polygonal obstacle [154]	19
2.11	Repulsive potential added to an attractive potential [166]	20
2.12	Isopotential contours changing from elliptical repulsion around rectangular obstacle to circular attraction away from it. [166]	21
2.13	Isopotential contours changing from superquadric repulsion around triangular obstacle to circular attraction away from it. [79]	22
2.14	Superquadric touching a triangular obstacle's edges [79]	22
2.15	Equi-potential contours computed by the numerical Navigation Function (NF) as seen in [10]; note the contour value of 0 at the goal on the top-left of the figure.	27
2.16	A globally Euclidean space such as a plane where the shortest path between any two points is always a straight line.	28
2.17	A globally Non-Euclidean space such as a Genus-1 Torus (donut) where the shortest path between any two points is a geodesic.	29

2.18	Breakdown of the general Path Planning Space (PPS) $\mathcal{P}$ into $\mathcal{P}^{\text{constraint}}$ and $\mathcal{P}^{\text{free}}$	30
2.19	Forward and inverse mapping between $\mathcal{C}$ and $\mathcal{W}$ of robots such as manipulators	32
2.20	Equivalence of $\mathcal{W}^{\text{free}}$ cannot be directly and simply assumed from $\mathcal{C}^{\text{free}}$ and vice-versa. Care must be taken to choose the PPS as a space that can be characterized mathematically.	32
2.21	Equivalence of the associated spaces of the PPS using the $\mathbf{f}^{\text{fwd}}$ and $\mathbf{f}^{\text{inv}}$ maps.	33
2.22	The configuration space $\mathcal{C}$ of a 2-Link Planar Manipulator (2LPM)	34
2.23	A 2D work space example of a 2-Link Planar Manipulator (2LPM) [154] . .	34
2.24	Modeling the obstacle as a spherical manifold and computing a path around it [121].	36
2.25	(a) Homotopic or deformable paths (b) Paths are not homotopic due to the visibility of an obstacle [68].	37
2.26	An initial calculated geodesic between the source and target locations on a hypersurface [177].	41
2.27	Left: Homotopic paths τ_1 and τ_2 , but not τ_3 , with either of the other paths. Right: Homologous paths τ_1 and $-\tau_2$ enclose shaded area; τ_3 is not homologous with either of the other paths. [15].	42
2.28	Example of several charts (blue polygons) covering/charting the spherical manifold; the union of charts forms an atlas [127]	45
2.29	Charting $\mathcal{C}^{\text{free}}$ which was formulated as an implicitly defined manifold [22] . .	46
2.30	Trulla algorithm	49
2.31	An example of the FSH from [108]	52
2.32	Illustration of obstacle avoidance viewed in three dimensions [12]	53
2.33	Illustration of obstacle avoidance on the horizontal and vertical planes [12] .	54
2.34	An example of a tangent graph with the target location T [149]	56
2.35	Top Image: Exit tangent point can be taken since the robot heading is appropriately aligned. Bottom Image: Exit tangent point cannot used since the robot heading is not aligned with the exit tangent direction. [149]	57
2.36	The comparison between APF and DAPF as illustrated in [33]	58
2.37	Figures (a), (b) and (c) show regular, random and free maps respectively, where black and colored points represent static and dynamic obstacles respectively. [169]	58
2.38	The time augmented and time layered environments as explained in [168] . .	61
2.39	Homotopic and homologous paths [19]	62
2.40	Paths in different homotopy classes as seen in [19]	62

2.41	The integral of F depends on the enclosed points [18]	63
2.42	The graph on the left side as opposed to the LAG on the right side [18]	64
2.43	Impact of LAG and A* in non-blocked, some blocked and all blocked homotopy classes [18]	65
2.44	Paths in different homotopic classes [18]	65
2.45	Paths in more cluttered environments in different homotopy classes [18]	65
2.46	Surfaces admitting and disallowing multiple homotopy classes [90]	66
2.47	10 distinct homotopy classes in 3-dimensions [90]	67
2.48	Homotopic path planning in a dynamic environment [90]	67
2.49	Word based homotopic path planning [82]	67
2.50	Evolution of sub-optimal paths in grey to the optimal path in black [82]	68
2.51	Topological path in the context of reference frame on the left, and in terms of the topological graph on the right [56]	69
2.52	Paths generated with HA* (red), HRRT (cyan) and HBug (green) algorithms for the two homotopy classes [56]	69
2.53	HBug and homotopy classes [56]	70
2.54	Different path solutions to the NAES problem [165]	72
2.55	Illustration of curve jumping [165]	72
2.56	Paths seen in 3D using NAES [165]	73
2.57	Impact of auxiliary lines [44]	74
2.58	Impact of increased auxiliary lines [44]	75
2.59	Different homotopy classes based on different cells [124]	76
2.60	Paths to avoid dynamic obstacles [124]	76
2.61	Impact of Recursively Embedded Palindrome strings in single and bi-directional RRT [180]	76
2.62	Paths in six different homotopy classes obtained by HARRT* [180]	77
2.63	Generation of vertices and edges [70]	77
2.64	Different environments and their road maps [70]	77
2.65	Change of Dubin's paths directed towards different locations on an annulus [179]	78
2.66	Diagonal path guides steering away from constraints [53]	79
2.67	Deadlock free paths with multiple robots in different settings [53]	80
2.68	Illustration of Planning Online by Switching Homotopies (POSH) [86]	80
2.69	An example graph construction with three chains and interconnections [86]	80

2.70	Different desired levels of inter-connectivity [86]	81
2.71	In (a), the gray circle and green circle are the robot and goal states. In (b), A* is used to find a low-cost trajectory (red) on the graph. In (c), edges that are no-longer viable (dashed blue lines) after the first movement are pruned. These can include the collision-free segment currently chosen as the direction of movement and segments that currently or potentially intersect with obstacles. Finally in (d), as the robot executes its chosen direction of movement, the environment changes with the lower square moving further along the positive x-axis. Post re-optimization, POSH finds a trajectory in a new homotopy class (red). [86]	81
2.72	Another illustration of POSH [86]	82
2.73	Path as a result of sampling and pruning candidate paths [104]	82
2.74	Additional pruning through HIP in conjunction with RRT* [104]	83
2.75	The two distinct homology groups of a torus	84
2.76	Illustration of admissible variation based path deformation in 3-dimensions [21]	85
2.77	Illustration of admissible variation based path deformation in 2-dimensions [21]	85
3.1	Example of a 1-manifold: a. The Euclidean space $\mathbb{R}^1$ is covered by a single chart $(\mathbb{R}^1, I_{\mathbb{R}^1})$, where $I_{\mathbb{R}^1} : \mathbb{R}^1 \rightarrow \mathbb{R}^1$ is the identity map. b. A curve and a circle are examples of a 1-manifold.	94
3.2	Examples of 2-manifolds [128]	94
3.3	Illustration of Definition 3.8 shows that the transition function $\psi \circ \phi^{-1}$ is defined on $\phi(\mathcal{U} \cap \mathcal{V})$	96
3.4	Open subsets of the circle, a 1-manifold from whom charts	96
3.5	Illustration of a smooth map between manifolds as seen in Definition 3.15	99
3.6	Common examples of surfaces	101
3.7	Examples of surfaces [97]	101
3.8	Potential paths for different start and goal pairs. The trivial case with one missing region is on the left hand side, and the non-trivial case with three missing regions is on the right hand side.	102
3.9	Path homotopy with the unit plane on the left hand side and the resulting homotopic paths on the right hand side. The divisions of the unit plane are mapped to X as homotopic paths, with the same initial and final points [115]	104
3.10	If $F : f \simeq_p f'$ and $F' : f' \simeq_p f''$, then $G : f \simeq_p f''$	105
3.11	Straight line homotopy. Left hand side: Path homotopy between f and g . Right hand side: Path homotopy between g and f , but not between f and h ; [115].	106
3.12	The product operation of two path-homotopy classes [115].	107

3.13	Connection between homology and homotopy	108
3.14	Non-simple loop: Winding number of its partitions [117]	109
3.15	Non-simple loop winding around one pole [117]	110
3.16	Riemann Sum and Winding number [117]	111
3.17	Contours surrounding one pole	112
3.18	Contour surrounding several poles [117]	113
3.19	Decomposition of a contour enclosing two singularities [117]	115
3.20	Each uniquely defined static constraint may lie in any of the PPSs identified for a robotic system; therefore, that PPS becomes the native space of the constraint, and the constraint has no meaning in another PPS.	120
3.21	In order for the original constraint to have an equivalence in another PPS, a mapping is required. Only after mapping the original constraints in their native space to the alternative PPS, is equivalence of constraints in both spaces established.	121
3.22	The 2LPM example as seen in step 3. The native PPS for static constraints present in the PPP is seen here. Note that, the constraints in their originally defined form do not have an equivalence in the alternative PPS.	122
3.23	Homeomorphism h_{R-1}^{R-2} from R-1 to R-2	134
3.24	Homeomorphism h_{R-2}^{R-3} from R-2 to R-3	137
3.25	Homeomorphism h_{R-3}^{R-4} from R-3 to R-4	138
3.26	Flowchart of the CFDMPP during the class marker value computation	143
4.1	R-1 space	150
4.2	R-1 and R-4 of the warehouse example with a glass wall, shelving unit and a box atop the shelving unit	151
4.3	Case 1: Generated path. (a) Path on R-4, starting at the OCE, bordering the ICE and terminating at the OCE. (b) Equivalent path in the robot workspace.	152
4.4	Case 2: Generated path. (a) Path on R-4 traversing through the interior. (b) Equivalent path in the robot workspace.	153
4.5	Case 3: Generated path. (a) Path on R-4 traveling purely on the ICE. (b) Equivalent path in the robot workspace.	154
4.6	Case 4: Generated path. (a) Path on R-4 traveling purely on the OCE. (b) Equivalent path in the robot workspace.	155
4.7	Geodesic on the Constraint-Free Discretized Manifold (CFDM) of a torus . .	156
4.8	Connected sum of two genus-1 tori [111]	156
4.9	Geodesic on the Constraint-Free Discretized Manifold (CFDM) of a sphere .	157

4.10	Simulation environment for stage 2	158
4.11	CFM of the environment denoted by R-4	160
4.12	Case 1: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.	166
4.13	Case 1: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.	167
4.14	Case 1: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.	168
4.15	Case 1: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and re- sulting path in R-1.	169
4.16	Case 1: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 0.45$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	170
4.17	Case 1: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	174
4.18	Case 1: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	175
4.19	Case 1: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	176
4.20	Alternative paths produced by CFDMPP on the CFDM	177
4.21	Case 2: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.	180
4.22	Case 2: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.	181
4.23	Case 2: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.	182
4.24	Case 2: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs $\epsilon_3 = 0.85$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	183
4.25	Case 1: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 1$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	184
4.26	Case 2: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	188

4.27	Case 2: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	189
4.28	Case 2: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	190
4.29	Alternative paths produced by CFDMPP on the CFDM	191
4.30	Case 3: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.	195
4.31	Case 3: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.	196
4.32	Case 3: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.	197
4.33	Case 3: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	198
4.34	Case 3: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 0.45$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	199
4.35	Case 3: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	203
4.36	Case 3: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	204
4.37	Case 3: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	205
4.38	Alternative paths produced by CFDMPP on the CFDM	206
4.39	Case 4: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.	207
4.40	Case 4: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.	208
4.41	Case 4: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.	209
4.42	Case 4: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	210
4.43	Case 4: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 0.45$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	211

4.44	Case 4: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	215
4.45	Case 4: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	216
4.46	Case 4: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.	217
4.47	Alternative paths produced by CFDMPP on the CFDM	218
4.48	Simulation environment based on the area of an average Costco Warehouse .	219
4.49	Case 1: CFDMPP with and without blocking in Costco style warehouse . . .	227
4.50	Case 2: CFDMPP with and without blocking in Costco style warehouse . . .	228
4.51	Case 3: CFDMPP with and without blocking in Costco style warehouse . . .	229
4.52	Case 4: CFDMPP with and without blocking in Costco style warehouse . . .	230
4.53	Case 5: CFDMPP with and without blocking in Costco style warehouse . . .	231
4.54	Case 1: PRM with and without blocking in Costco style warehouse	232
4.55	Case 1: RRT with and without blocking in Costco style warehouse	233
A.1	Case 1: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.	255
A.2	Case 1: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.	256
A.3	Case 1: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.	257
A.4	Case 1: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.	258
A.5	Case 1: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.	259
A.6	Case 1: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.	260
A.7	Case 1: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.	261
A.8	Case 1: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.	262
A.9	Case 1: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.	263
A.10	Case 1: CFDMPP in the case of 1 pole (first group of DCs). Only 1 path-class exists, so blocking is not possible. (a) Path in R-4. (b) Path in R-1.	264

A.11	Case 1: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1.4). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	265
A.12	Case 1: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	266
A.13	Case 1: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	267
A.14	Case 1: CFDMPP in the case of 16 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	268
A.15	Case 1: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	269
A.16	Case 1: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	270
A.17	Case 1: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	271
A.18	Case 1: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	272
A.19	Case 1: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.	273
A.20	Case 1: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.	274
B.1	Case 2: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.	276
B.2	Case 2: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.	277

B.3	Case 2: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.	278
B.4	Case 2: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.	279
B.5	Case 2: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.	280
B.6	Case 2: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.	281
B.7	Case 2: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.	282
B.8	Case 2: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.	283
B.9	Case 2: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.	284
B.10	Case 2: CFDMPP in the case of 1 poles (first group of DCs). Due to the pole's location, only 1 path class is possible. (a) Path in R-4. (b) Path in R-1.	285
B.11	Case 2: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.425). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	286
B.12	Case 2: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.85). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	287
B.13	Case 2: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.85). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	288
B.14	Case 2: CFDMPP in the case of 16 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	289
B.15	Case 2: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1.15). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	290
B.16	Case 2: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	291

B.17	Case 2: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	292
B.18	Case 2: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	293
B.19	Case 2: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.	294
B.20	Case 2: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.	295
C.1	Case 3: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.	297
C.2	Case 3: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.	298
C.3	Case 3: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.	299
C.4	Case 3: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.	300
C.5	Case 3: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.	301
C.6	Case 3: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.	302
C.7	Case 3: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.	303
C.8	Case 3: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.	304
C.9	Case 3: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.	305
C.10	Case 3: CFDMPP in the case of 1 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1.15). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	306
C.11	Case 3: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.8). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	307

C.12	Case 3: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.4). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	308
C.13	Case 3: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.4). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	309
C.14	Case 3: CFDMPP in the case of 16 poles (first group of DCs); unable to tune for an alternative path class. (a) Path in R-4. (b) Path in R-1.	310
C.15	Case 3: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	311
C.16	Case 3: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.	312
C.17	Case 3: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.75). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	313
C.18	Case 3: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.75). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	314
C.19	Case 3: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.	315
C.20	Case 3: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.	316
D.1	Case 4: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.	318
D.2	Case 4: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.	319
D.3	Case 4: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.	320
D.4	Case 4: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.	321
D.5	Case 4: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.	322

D.6	Case 4: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.	323
D.7	Case 4: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.	324
D.8	Case 4: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.	325
D.9	Case 4: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.	326
D.10	Case 4: CFDMPP in the case of 1 poles (first group of DCs). Only 1 path class is possible, so no blocking is seen. (a) Path in R-4. (b) Path in R-1.	327
D.11	Case 4: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.325). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	328
D.12	Case 4: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.5). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	329
D.13	Case 4: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.518). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	330
D.14	Case 4: CFDMPP in the case of 16 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	331
D.15	Case 4: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	332
D.16	Case 4: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	333
D.17	Case 4: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.35). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	334

D.18 Case 4: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.3). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.	335
D.19 Case 4: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.	336
D.20 Case 4: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.	337

Abbreviations

2LPM 2-Link Planar Manipulator [31](#), [33](#), [91](#), [117–120](#), [122](#), [124–129](#)

ACO Ant Colony Optimization [23](#), [24](#)

APF Artificial Potential Field [13](#), [14](#), [17](#), [19–21](#), [23–26](#), [29](#), [30](#), [35](#), [37](#), [38](#), [41](#), [48](#), [50–53](#), [57–60](#), [86](#), [87](#)

BFP Best First Planner [27](#)

CCM Closed Chain Manipulator [33](#), [34](#), [117](#)

CDGT Cell Decomposition and Graph Traversal [7](#), [9](#), [11](#), [60](#), [86](#), [87](#)

CFDM Constraint-Free Discretized Manifold [141–147](#), [149](#), [150](#), [159–161](#), [220](#), [221](#), [223–225](#), [234–237](#)

CFDMPP Constraint-Free Discretized Manifolds-based Path Planner [3](#), [4](#), [87–90](#), [101](#), [115–117](#), [125](#), [126](#), [143](#), [146](#), [147](#), [151–154](#), [157](#), [160–168](#), [171–176](#), [178](#), [180–182](#), [185–190](#), [192–197](#), [200–205](#), [207–209](#), [212–217](#), [219](#), [221–226](#), [234–237](#), [254–263](#), [273–284](#), [294–305](#), [315–326](#)

CFM Constraint-Free Manifold [116](#), [117](#), [121](#), [126](#), [129](#), [141](#), [146](#), [149](#), [151](#), [157](#), [159](#), [161](#), [163–165](#), [192](#), [193](#), [220](#), [225](#), [226](#), [234](#), [237](#)

CS Constraint Set [123](#), [124](#), [129](#)

CV Certainty Value [52](#)

DAPF Dubin’s APF [57](#)

DC Dynamic Constraint [47](#), [53](#), [60](#), [61](#), [86](#), [87](#), [116](#), [117](#), [125–127](#), [142–144](#), [146](#), [148](#), [157](#), [159](#), [160](#), [164](#), [179](#), [193](#), [220](#), [221](#), [223](#), [235](#)

DCS Disjoint Constraint Set [123–125](#), [127–133](#), [136](#), [138](#), [140](#), [142](#), [146–149](#), [154](#), [156](#), [159](#), [220](#)

DMP Dynamic Movement Primitives [60](#)

DOF Degrees Of Freedom [27](#), [37](#), [91](#)

DTD Dynamic Topology Detector [39](#)

EA Evolutionary Algorithm [17](#), [22–24](#), [86](#)

EE End-Effector [6](#), [31](#), [33](#), [91](#)

FSF Free Space Force [52](#)

FSH Free Space Histogram [52](#)

GA Genetic Algorithm [23](#), [24](#)

GVD Generalized Voronoi Diagram [35](#), [38](#), [39](#)

HA* Homotopic A* [69](#), [71](#)

HB Homotopic Bug [70–72](#)

HBM Homotopy Based Method [60](#), [80](#), [87](#), [89](#)

HCM Homotopy Continuation Methods [72–74](#)

HIMM Histogram In Motion Mapping [52](#)

HRRT Homotopic RRT [70](#), [71](#)

HSFM Headed Social Force Model [58](#)

ICE Inner Constraint Edge [140](#), [149–151](#), [153](#), [159](#), [160](#), [163](#), [165](#), [178](#), [192](#), [193](#), [220](#)

IFTM Inverse Function Theorem for Manifolds [99](#)

LAG L-Augmented Graph [64](#)

LPRM Lazy PRM [10](#), [15](#), [16](#)

LM-RRT Machine Learning-based Multi-RRT [14](#), [15](#)

MA Memetic Algorithm [23](#), [24](#)

MBM Model Based Methods [47](#), [87](#), [89](#)

MPC Model Predictive Control [17](#), [22–24](#), [26](#), [47](#), [49](#), [51](#), [55](#), [87](#)

MR Mobile Robot [117](#)

NAES Non-linear Algebraic Equation System [72–76](#)

NF Navigation Function [17](#), [22](#), [25](#), [26](#), [29](#), [30](#)

NLOC Non-Linear Optimal Control [23](#)

NOCP Non-Linear Optimal Control Problem [23](#)

NOP Non-Linear Optimization Problem 23

OA Optimization Algorithm 17, 25, 41, 46–48, 56, 60, 69, 86–88, 90, 151, 152

OCE Outer Constraint Edge 140, 149–151, 153, 159, 163, 165, 178, 220

OCM Open Chain Manipulator 33, 117

P-RRT* Potential Functions-based RRT* 14

PDR Path Deformation Roadmap 37

PGD-RRT* Potential Guided Directionalized-RRT* 13, 14

PPM Path Planning Manifold 74, 86, 90, 93, 100, 101, 115, 117, 126, 127, 147, 155, 157

PPP Path Planning Problem 8, 10, 14, 17, 22, 25, 31, 32, 47, 55, 56, 63, 72, 73, 85–87, 90, 93, 100, 115, 117–119, 123, 126–129, 140, 142, 148, 154, 156

PPS Path Planning Space 6, 7, 16, 17, 24–27, 31, 33, 36, 39, 41–44, 46–48, 50, 51, 53, 54, 60, 62, 63, 65, 68, 74, 76, 78, 83, 85, 86, 88, 90–93, 100, 102, 103, 106, 107, 115–135, 140, 141, 145–149, 151, 154–159, 163, 164, 219, 220, 224, 225

PPPS Primary Path Planning Space 117–122, 124–128, 148, 157–159, 219, 220

PRM Probabilistic Road Map 7–13, 15, 16, 36, 37, 78, 86, 87, 147, 160–164, 166–168, 171–176, 178, 180–182, 185–190, 192, 194–197, 200–205, 207–209, 212–217, 224–226, 234, 235, 254–263, 273–284, 294–305, 315–326

PRM* Probabilistic Road Map* 12, 13

PSM Product Smooth Manifold 98

PSO Particle Swarm Optimization 23, 24

RPP Randomized Path Planner 27

RRG Rapidly exploring Random Graph 12, 13

RRT Rapidly exploring Random Tree 7–16, 37, 46, 71, 76, 83, 86, 87, 147, 160, 162–164, 166–168, 171–176, 178, 180–182, 185–190, 192, 194–197, 200–205, 207–209, 212–217, 224–226, 234, 235, 254–263, 273–284, 294–305, 315–326

RRT* Rapidly exploring Random Tree* 12–15

RRT*-AB RRT*-Adjustable Bounds 15

SA Sampling Algorithm 7, 9–11, 14, 16, 17, 37, 41, 46, 47, 56, 60, 69, 76, 86–88, 90, 151, 152, 162–164, 178, 179, 192, 193, 224, 225, 235–237

SC Static Constraint 46, 47, 61, 86, 87, 116, 119–126, 142, 145, 147–149, 151, 153, 156, 158, 219, 234

SFLA Shuffled Frog Leaping Algorithm [23](#), [24](#)

SHIO Single Homotopy Inducing Obstacle [66](#)

SLPRM Semi-Lazy PRM [15](#)

SM Smooth Manifold [98–100](#)

SO Special Orthogonal Group [33](#), [34](#), [43](#)

SPPS Secondary Path Planning Space [117–122](#), [124–127](#)

SR Stationary Robot [117](#), [118](#)

S-RRT Smoothly-RRT [16](#)

TG Tangent Graph [55](#), [56](#)

TM Topological Manifold [93–95](#), [98](#), [100](#)

T-RRT Transition-based RRT [12](#)

UGV Unmanned Ground Vehicle [23](#), [117–121](#), [124](#), [125](#), [127–129](#), [148](#), [150](#), [156–158](#), [220](#), [221](#), [223](#)

UAV Unmanned Aerial Vehicle [1](#), [117](#), [155](#), [156](#)

UUV Unmanned Underwater Vehicle [117](#)

VG Visibility Graph [35](#)

VOS Velocity Obstacle Sets [53–55](#), [87](#)

VFF Virtual Force Fields [52](#)

List of Latin Symbols

- $\mathcal{C}$ Configuration space [6](#), [13](#), [17](#), [18](#), [20](#), [21](#), [27](#), [31–38](#), [42](#), [43](#), [45](#), [91](#), [98](#), [115](#), [117–120](#), [125](#), [127–129](#)
- $\mathcal{CO}$ Convex obstacle region in configuration space [17–19](#), [123](#)
- $\mathcal{C}^{\text{constraint}}$ Configuration space corresponding to constraints [31](#), [32](#)
- $\mathcal{C}^{\text{free}}$ Configuration space free of constraints/obstacles [12](#), [13](#), [17](#), [26](#), [27](#), [31](#), [32](#), [45](#), [88](#)
- CS_r Constraint set r [123](#)
- DCS_r Disjoint constraint set r [124](#), [125](#), [130](#)
- DCSs Set of all disjoint constraint sets of all path planning spaces [124](#), [125](#)
- $\text{DCSs}^{(\cdot)}$ Set of all disjoint constraint sets of space $(\cdot)$ [124](#), [125](#), [127–130](#)
- $\mathbf{f}^{\text{fwd}}$ Mapping defining forward kinematics [34](#), [128](#), [129](#)
- $\mathbf{f}^{\text{inv}}$ Mapping defining inverse kinematics [34](#), [35](#)
- H -signature** Unique measure associated to a homotopic class [61–64](#), [66–68](#), [82](#)
- $\mathcal{LC}$ Level Curve [101](#)
- n_{ML} Number of SCs classified as mechanical limits (ML) [119](#)
- n_{NG} Number of SCs classified as no-go zones (NG) [119](#), [148](#), [158](#), [220](#)
- n_{OB} Number of SCs classified as obstacles (OB) [119](#), [148](#)
- n_{SG} Number of SCs classified as singularities (SG) [119](#)
- n_{SC} Total number of SCs [119](#), [123](#), [148](#), [158](#), [220](#)
- $\text{node}^{\text{start}}$ Starting point in CFDM [143](#), [144](#), [150](#), [152](#), [153](#), [193](#)
- $\text{node}^{\text{goal}}$ Target point in CFDM [150](#), [153](#), [193](#)
- curr** Current node in CFDM [143–145](#)

nbr Neighbour node in CFDM 143, 144

$\mathcal{P}$ Path planning space 6–8, 10, 91–93, 102, 118

$\mathcal{P}^{\text{mechanical limits}} \subseteq \mathcal{P}$ Space corresponding to the mechanical limits of the robot (e.g., joint limits) 93

$\mathcal{P}^{\text{no-go zones}} \subseteq \mathcal{P}$ Space corresponding to the no-go zones 93

$\mathcal{P}^{\text{obstacles}} \subseteq \mathcal{P}$ Space corresponding to the obstacles 93

$\mathcal{P}^{\text{singularities}} \subseteq \mathcal{P}$ Space corresponding to the singularities of the robot 93

$\mathcal{P}^{\text{constraint}} \equiv \mathcal{P}^{\text{mechanical limits}} \cup \mathcal{P}^{\text{no-go-zones}} \cup \mathcal{P}^{\text{obstacles}} \cup \mathcal{P}^{\text{singularities}}$ Space corresponding to all constraints 7, 8, 10, 16, 17, 27, 29, 86, 92, 93, 115

$\mathcal{P}^{\text{free}} \equiv \mathcal{P} \setminus \mathcal{P}^{\text{constraint}}$ Constraint-free Space 7, 10, 14, 16, 17, 29, 30, 33, 90, 92, 93, 102, 103

$\mathbf{p} \in \mathcal{P}$ A point in $\mathcal{P}$ 6, 91, 131, 133–138, 141, 149, 159, 220

$\mathbf{p}^{\text{start}} \in \mathcal{P}$ Starting point in $\mathcal{P}$ 7–9, 11, 61, 86, 102, 103, 115, 116, 143–145, 150, 152, 153, 163, 178, 192, 193

$\mathbf{p}^{\text{goal}} \in \mathcal{P}$ Destination point in $\mathcal{P}$ 7–9, 11, 61, 86, 102, 103, 115, 116, 143–145, 150, 152, 153, 163, 178, 192, 193

$\mathbf{p}_{\text{path}} \in \mathcal{P}$ Path connecting $\mathbf{p}^{\text{start}}$ to $\mathbf{p}^{\text{goal}}$ in $\mathcal{P}$ 145, 150

$\mathbf{q} \in \mathcal{C}$ A point in the configuration space 17–20, 91

$\mathbf{q}^{\text{start}} \in \mathcal{C}$ Starting point in configuration space 35

$\mathbf{q}^{\text{goal}} \in \mathcal{C}$ Destination point in configuration space 17–19, 35

$\mathcal{S}_{\text{surface}}$ A Surface 100, 101

$\mathbf{SC}_k$ Static constraint k 123, 124

$\mathbf{T}_p\mathcal{S}$ Tangent Space at a point p on a surface 101

$\mathcal{TS}$ Topological space of set $\mathcal{S}$ and topology $\mathcal{T}$ 130–136, 149, 153, 159

$\mathcal{W}$ Workspace 6, 20, 31–35, 91, 98, 117–119, 121, 122, 124, 125, 127–129, 148, 157, 219

$\mathcal{W}^{\text{constraint}} \subseteq \mathcal{W}$ Space corresponding to all constraints 31, 32

$\mathcal{W}^{\text{free}} \equiv \mathcal{W} \setminus \mathcal{W}^{\text{constraint}}$ Constraint-free space 26, 27, 31, 32, 88

$\mathbf{x} \in \mathcal{W}$ A point in the work space 20, 91

$\mathbf{x}^{\text{start}} \in \mathcal{W}$ Starting point in the work space 26, 42

$\mathbf{x}^{\text{goal}} \in \mathcal{W}$ Destination point in the work space 20, 26, 42

List of Greek Symbols

η_0 The range of influence [18](#), [19](#)

Δ_{CS} Global proximity threshold between constraints [123–125](#), [129](#), [148](#), [159](#), [220](#)

Δ_{CS}^{ij} Measured proximity between constraints i and j [123](#), [124](#)

δ Boundary. For instance, δx denotes the boundary of x [123](#)

Chapter 1

Introduction

1.1 Background

Robotics in its many forms has intrigued the common man and has kept the engineering community on its feet. The effort to build and use stationary and mobile robots that make life more comfortable has been apparent with the extent of robot design and use for several applications. A few such applications include automated manufacturing, cleaning robots, robots for assisted living, autonomous vehicles for autonomous driving, extra-terrestrial explorations, search and rescue missions, surveillance drones, terrain/ hazardous environment exploration and mapping, and delivery/retrieval robots, to name a few.

Current generation autonomous robots are entities capable of receiving information, processing it, and making their own decisions to respond on the fly [113]. Such autonomous robots and artificial intelligence are steadily integrating into daily lives of consumers and that of the automation and supply chain industry. Thus, a large and increasing proportion of economic activity becomes robotized [63]. Even as far back as 2016, there was 35% annual growth in robot orders from 2010–2015. Canada leads the North American rise in demand for robots, with strong growth in the automotive, [Unmanned Aerial Vehicle \(UAV\)](#) and space segments. Canada has been at the forefront of this global robotics investment cycle. Foreign investment in the robotics sector has seen a steady upswing in Canada, with global incumbents such as ABB, Amazon and General Electric making sizable investments for their robotics divisions in this country, paralleling the efforts of local Canadian innovators, such as MDA, Clearpath Robotics, Bionik Labs, D&D Automation, Kinova Robotics and Provectus Robotics [29].

The increased economic investment in robotics is clearly understood in the case of an automation chain in a factory or a semi/fully-autonomous warehouse. Here, working robots have to be aware of their surroundings and must meet real-time constraints to interact with people while avoiding collisions with humans and other obstacles, such as other autonomous robots [113]. Endangering humans or encountering a collision with another automated machine, static or dynamic, is an expensive consequence that must be avoided at all costs. Investment in warehouse robotics technology startups clocked in at \$381 million in the first

quarter of 2020, up 57% from the same period in 2019. Warehouse robotics investment exploded during the global pandemic period of Covid-19, which started in early 2019. These post-Covid funding rounds reflect a growing interest in touchless automation technology to replace or supplement human workers, make warehouses safer and make sure operations are not disrupted [50].

1.2 Motivation

The information thus far gives a brief insight of warehouse robotics' increasing dominance in the robotic market and the investments that support its growth. The focus of this research is on autonomous path planning, especially in the context of warehouse robotics. Henceforth, a brief overview of the problem in hand and general solutions for the problem is provided.

Typical autonomous robots can be used either for exploring unknown environments or navigating known environments. In the former, autonomy occurs in the form of Simultaneous Localization and Mapping (SLAM) [138, 140]. We shall focus on the latter, whose environment may be broken down into stationary and moving obstacles, and obstacle-free navigable space. Regions that prevent navigation are generally called constraints. Static constraints include stationary obstacles, no-go zones, mechanical limits and singularities; whereas, dynamic constraints could be the sudden appearance of an obstacle, human or another robot. The knowledge of dynamic constraints is not known before hand, unlike that of static constraints, so such an environment is considered partially known. Path planning in an environment with a known map and potential to encounter dynamic constraints is the main focus of this thesis. The motivation stems from path planning needs in a warehouse/factory floor setting, for instance, where the map with static constraints is known and must be avoided, and the generated path must be free of dynamic constraints as well.

Common algorithms in the literature handle path planning in such environments by accomplishing two main steps, either sequentially or in parallel. First, a path between the source and target locations is generated. Second, segments of the path are evaluated for any constraint violation, such as crossing collision zones. The two main path planning approaches are based on sampling and optimization methods. Sampling based algorithms generate paths by sampling the space, such that only samples that do not violate constraints are retained. Samples that are found to interact with constraints are discarded. Until a sequence of such constraint-free samples that connect the source to the target location is generated, the periodic evaluation of samples is necessary. This process requires that such algorithms always evaluate segments of the path for constraint violation, in order to generate a path. Optimization approaches on the other hand, use cost functions to minimize the distance (or other cost) towards the target location and maximize the distance towards constraints. Apart from facing the same issue of periodic evaluation of every point as part of the cost function, cost functions must be re-formulated every time the source and/or target location is changed. Additionally, since the fundamental operation is that of minimization, there is a risk of stagnating at local minima and not being able to generate a complete path. Therefore, the path planning problem remains open, with the aim of developing a

path planner that can avoid the aforementioned pitfalls, while also generating a path in the navigable constraint-free region.

1.3 Objectives and Contributions

The aim of the thesis is to plan paths in an environment whose static constraints are known apriori and where knowledge of new constraints (static or dynamic) renders part of the space non-navigable. We achieve this objective by proposing the [Constraint-Free Discretized Manifolds-based Path Planner \(CFDMPP\)](#), which is a path planner based on topology and differential geometry. The objective of the thesis is then to develop a path planner that accomplishes the following:

- For static constraints whose knowledge is known during map (manifold) construction, explicit periodic verification of constraint violation needs to be avoided.
- No ambiguity of path solution must be encountered, such as stagnation at local minima.
- Should the desired path be rendered invalid, an alternative path, if exists, must be produced.

To achieve such objectives, the [CFDMPP](#) makes the following general assumptions:

- The expanse of the path planning space and the boundaries of static constraints are known. The latter may be obtained offline via the environment map.
- The existence of a sensing system that monitors the presence or absence of any constraints that were not present during the initial path planning. This sensing system relays such information to the path planner.
- The working environment is 2D.

The main contributions of the thesis via the [CFDMPP](#) are:

- The formulation of all or a subset of the navigable static constraint-free space as a topological space, and eventually as a smooth constraint-free manifold. Compactness ensures discretization of the manifold and generation of the path on the discretized constraint-free surface. Path connectedness ensures that any path on that constraint-free manifold is always constraint-free, without any explicit constraint-evaluation or minimization process.
- The use of homology and homotopy classes on the constructed constraint-free manifold to generate alternative paths. This allows the constraint-free manifold/map to remain unmodified while an alternative path is queried. The knowledge of the map thus remains intact, leaving only the path to be changed.

- The identification of [CFDMPP](#)'s tuning parameters and their potential impacts on the quality of alternative paths obtained. This, in conjunction with a systematic explanation of the algorithm, provides the foundation for a body of future research in the area.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter [2](#) provides a comprehensive literature review about the path planning problem.
- Chapter [3](#) provides the theoretical background required to demonstrate the [CFDMPP](#), and details the steps to apply it.
- Chapter [4](#) illustrates results of the [CFDMPP](#) and the corresponding analysis via three stages: proof of concept, comparison with other commonly used algorithms for path planning, and finally, [CFDMPP](#) as a path planner for warehouse robotics.
- Chapter [5](#) draws conclusions and suggests possible areas of future work.

1.5 Scholarly Outcome

The research presented in the thesis has led to the following papers (either published or in preparation):

- [\[137\]](#): S. Radhakrishnan and W. Gueaieb. Genetic algorithm based direction finder on the manifold for singularity free paths. In *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 151–156, 2016
- [\[139\]](#): S. Radhakrishnan and W. Gueaieb. Constraint-free discretized manifolds for robotic path planning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1878–1884. IEEE, October 2020
- [\[143\]](#): Sindhu Radhakrishnan and Wail Gueaieb. A state-of-the-art review on topological geometry-based robotic path planning. Part I: Static constraints. *Robotics and Autonomous Systems*, 2022. (To be submitted)
- [\[144\]](#): Sindhu Radhakrishnan and Wail Gueaieb. A state-of-the-art review on topological geometry-based robotic path planning. Part II: Dynamic constraints. *Robotics and Autonomous Systems*, 2022. (To be submitted)
- [\[142\]](#): Sindhu Radhakrishnan and Wail Gueaieb. Manifold-based homotopic path planning. *International Journal of Advanced Robotics Systems*, 2022. (To be submitted)
- [\[141\]](#): Sindhu Radhakrishnan and Wail Gueaieb. Analysis of parameters guiding manifold-based homotopic path planning. *Journal of Robotics*, 2022. (To be submitted)

Chapter 2

Literature Review

The autonomy of robots in their environments is primarily dependent on their autonomy of perceiving their environment, planning the required task while accounting for constraints and then executing that chosen task. In the case of stationary robots such as industrial manipulators, the required task may be a pick-and-place operation. The task involves picking an object, planning the end-effector's path from a source to a target location while avoiding singularities and obstacles, travelling that path and placing the object. Alternatively, for a mobile robot in an automated warehouse, the task of interest may be surveillance and inventory/stock monitoring at a specific location of the warehouse. Such a task can be a combination of parsing the known map of the environment, sensing constraints such as obstacles, no-go zones and other robots, and planning a path that is free of such constraints. Irrespective of the stationary/mobile nature of robots used, a common necessity of autonomy in robotics is path planning.

Autonomous path planning is broadly broken into two phases, where the first is deliberative and the second is reactive [93]. The deliberative phase is task oriented, i.e focuses purely on generating a path to satisfy certain path metrics or characteristics. This phase assumes that all constraints that need to be accounted for are known, thus implying that knowledge of the environment is available apriori. Therefore the deliberative phase is synonymous with planning a path in an environment with constraints which are known and stationary in time. Examples include an industrial manipulator whose singularities and no-go zones are known, and a mobile robot in a room with an obstacle, whose map is available. On the other hand, the reactive phase serves the purpose of reacting in case of unexpected situations or situations whose knowledge was not available apriori. This could include cases where constraints such as obstacles, other robots or humans are suddenly encountered, and the planner must react in time and modify the original path locally to avoid collisions with such constraints. Examples include unplanned human presence in the workspace of an industrial manipulator and in the case of mobile robots in a warehouse, other robots, humans and objects falling from storage spaces constitute a change in constraint knowledge. Thus reactive planning is a tool used for dynamic environments.

A desirable path may be evaluated by several metrics such as path length, proximity of the path to the desired source and target locations, and characteristics of the path such as

path continuity, smoothness, clearance from constraints and other application based rubrics. Similarly, path planning algorithms may also be evaluated against rubrics of computation time, computational complexity, completeness of solution, capability of being used for online versus offline planning and for a static or dynamic environment. The choice of path planning algorithms can affect different metrics of the path differently, so care must be taken to choose a path planner. This is especially true when the nature of the space in which the path is planned can vary. Algorithms for path planning can be broadly segmented into two main categories depending on the formulation of the [Path Planning Space \(PPS\)](#) without explicit mathematical structure (as a set of points), or with mathematical structure of smoothness (manifolds). In this chapter, a comprehensive literature review pertaining to deliberative path planning algorithms is provided.

This chapter is divided into three main sections. The first two sections given by Section [2.1](#) and Section [2.2](#), represent path planning via differences in representation of the space. The former uses approaches that represent the space as a set of points, and the latter, via manifolds. These sections establish the difference between the approaches for path planning in general. Section [2.3](#) then provides an overview of techniques that handle dynamic constraints, by differentiating between model based methods and homotopy based methods.

2.1 Set/Space of Points

The most fundamental representation of a path planning environment is by representing it as a union of all possible points that constitute the [PPS](#). A [PPS](#) can be any space used in the path search, such as the configuration space $\mathcal{C}$, the workspace $\mathcal{W}$, joint velocity space, [End-Effector \(EE\)](#) velocity space, joint torque space, and the [EE](#) force space, to name a few. For the purposes of generality, a general notation of $\mathcal{P}$ is chosen to represent the [PPS](#). If the path planning environment is a Euclidean space, i.e., $\mathcal{P} \subset \mathcal{R}^n$, then the representative point $\mathbf{p}$ is n-dimensional, i.e., $\mathbf{p} \in \mathcal{R}^n$.

$$\mathbf{p} = [p_1 \ p_2 \ \cdots \ p_n]^T, \text{ where } p_i \in [p_i^{\min}, p_i^{\max}] = p_i^{\text{span}} \quad (2.1)$$

Naturally, since it is impossible to store every single point that makes up the space, the bounds of the space as seen in [\(2.1\)](#), and a chosen discrete interval allows for points located at certain distance from each other to represent the space. The union of all such points thus characterizes the [PPS](#), which in this case is $\mathcal{P}$, as

$$\mathcal{P} = \mathbf{p}^{\text{all}} = \prod_{i=1}^n p_i^{\text{span}} \quad (2.2)$$

A path planning problem needs to represent the region occupied by the robot, the constraint region consisting of obstacles, no-go zones and singularities, and the constraint-free region. The point-set representation as seen in [\(2.2\)](#) therefore allows two kinds of representations for each of the three aforementioned regions. The first involves denoting the region as simply a point. For instance, the robot may be denoted by its current location accounted

from its centre of mass or possibly any other suitable choice of reference. Similarly, constraint regions can also be denoted as points, for example, singularities by their specific singular configurations and obstacles/ no-go zones as a set of discrete points. Note here, that the point-set representation does not attribute any boundaries unless otherwise specified by the explicit bounds as seen in (2.1). Even so, there is no explicit partitioning of the space, since partitioning would imply that points in a set satisfy relationship properties, that are different from those satisfied in other sets. This introduces the notion of unpartitioned and partitioned space.

Unpartitioned space such as point-sets, facilitate sampling and optimization based algorithms with great ease, since all regions of interest are denoted by points. Partitioned space on the other hand, enables region characterization as polytopes. That is, they are still a union of discrete points, however, connectivity relationships are satisfied at least for points on the boundaries of polytopes. Partitioned space facilitates a variety of approaches, including cell decompositions, graph traversal and sampling based optimization techniques. It is in fact, common to notice an interchange/combination of such approaches and such spaces for needs that are specifically suited to the application. This section therefore, is now divided into the main approaches used for point-set characterization of the path planning environment. As such, section 2.1.1 focuses on sampling based methods, followed by optimization based techniques in 2.1.2.

2.1.1 Sampling-Based Methods

Sampling Algorithms (SAs) exploit the structure of point-set representations of environments. This automatically makes them an easy choice when sensors producing point-cloud approximations of environments are involved. Hence, they form the foundation for a substantial amount of path planning literature. As the name suggests, the principle of **SAs** lies in sampling or picking a point from the **PPS**, generically denoted by $\mathcal{P}$. The sampling procedure in its most basic form randomly generates a point based on the bounds of the sampling space, a given sampling interval, a source $\mathbf{p}^{\text{start}}$ and a target $\mathbf{p}^{\text{goal}}$. The sampled point is then processed to determine where it lies in the following three regions of interest: $\mathcal{P}^{\text{free}} = \mathcal{P} \setminus \mathcal{P}^{\text{constraint}}$, $\mathcal{P}^{\text{constraint}}$ or the region occupied by the robot. If the point is not in $\mathcal{P}^{\text{free}}$, then it is discarded and another point must be generated. Once a compliant point is generated, further processing involves finding other points in the space that lead to the target location in the space. Such algorithms employ tree-based structures to store and update information about the generated points. In other words, the connectivity of the space is stored via parent-child node relationships. Depending on the kind of **SAs** used, the model either simultaneously builds a tree and finds a path between the desired $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, or finds a path after building a representation of the space. The primary advantage of **SAs** is their versatility, as they are independent of robot models and do not require an explicit description of the constraint region or the robot itself. To this end, **SAs** can be broadly classified into three main groups: Probabilistic Road-Maps (**Probabilistic Road Map (PRM)**), Rapidly exploring Random Trees (**Rapidly exploring Random Tree (RRT)**), and **Cell Decomposition and Graph Traversals (CDGTs)**. A succinct description of each of the above

groups of planners is provided below. Algorithms for [PRM](#) and [RRT](#) are presented here as introduced in [\[76, 112\]](#).

The basic [RRT](#) algorithm is a single-query planner, i.e., aimed at single query applications [\[73, 112\]](#). Single query problems usually consider the [Path Planning Problem \(PPP\)](#) as defining a specific $\mathbf{p}^{\text{start}}$, $\mathbf{p}^{\text{goal}}$ and finding a path to connect them. No new $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ may be defined, as this is a different query and may change the problem definition, thus changing the solution from the perspective of [RRT](#). A tree is incrementally grown from the source to the target, or vice versa. Simply stated, a point is randomly selected in the path planning space $\mathcal{P}$. If the point lies in the free space, a connection is attempted to the nearest vertex in the tree with additional details of its flowchart shown in [Figure 2.1](#). The [RRT](#) was shown to be probabilistically complete [\[95\]](#). An example of typical motions experienced by a non-holonomic robot using [RRT](#) are seen in [Figure 2.2](#) as originally presented in [\[112\]](#).

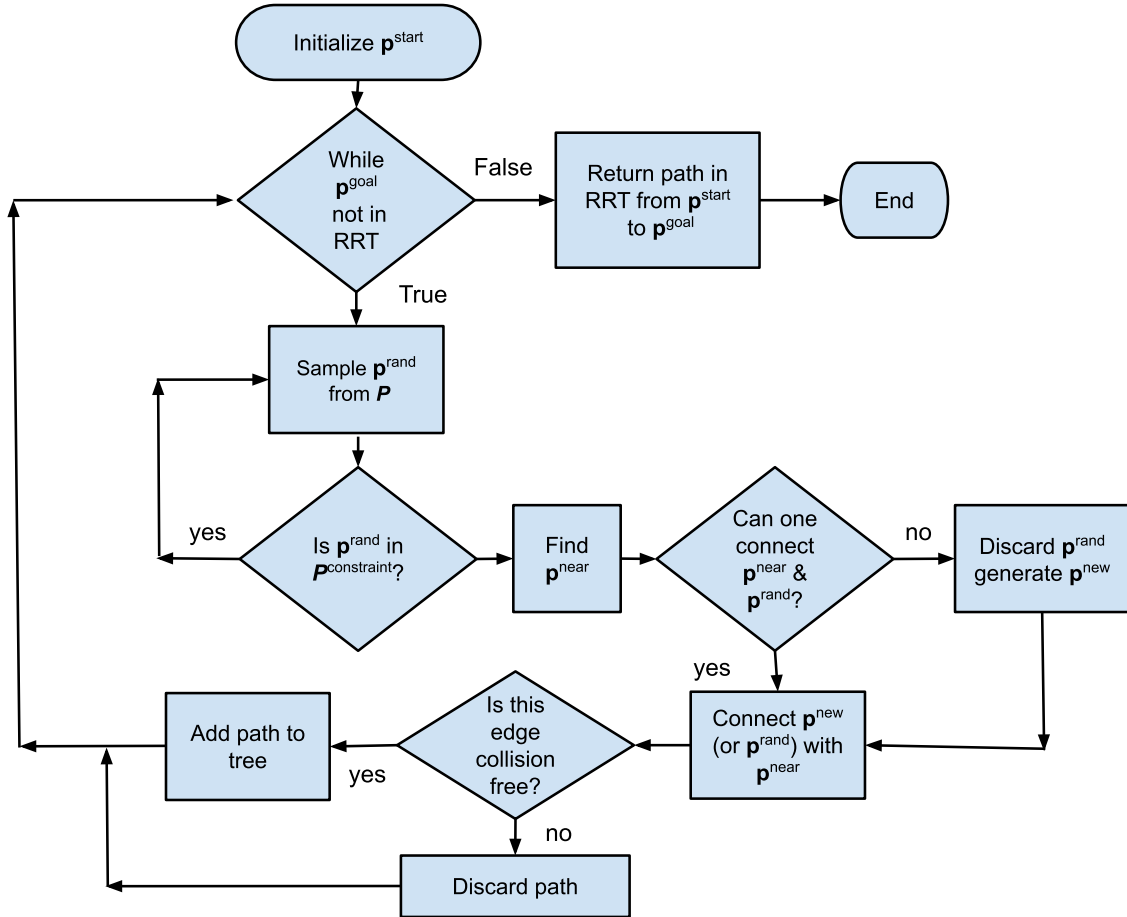


Figure 2.1: Flowchart of Rapidly exploring Random Tree (RRT)

The [PRM](#) algorithm on the other hand, is a tool that focuses on multi-query applications [\[45, 73, 76\]](#). In its basic version, it consists of two phases, which are the pre-processing phase and the search phase. The pre-processing phase, commonly referred to as the learning phase is when a road-map of $\mathcal{P}$ is constructed. It is accomplished by attempting connections among n randomly sampled points in $\mathcal{P}$, which are discarded if located in $\mathcal{P}^{\text{constraint}}$. Once

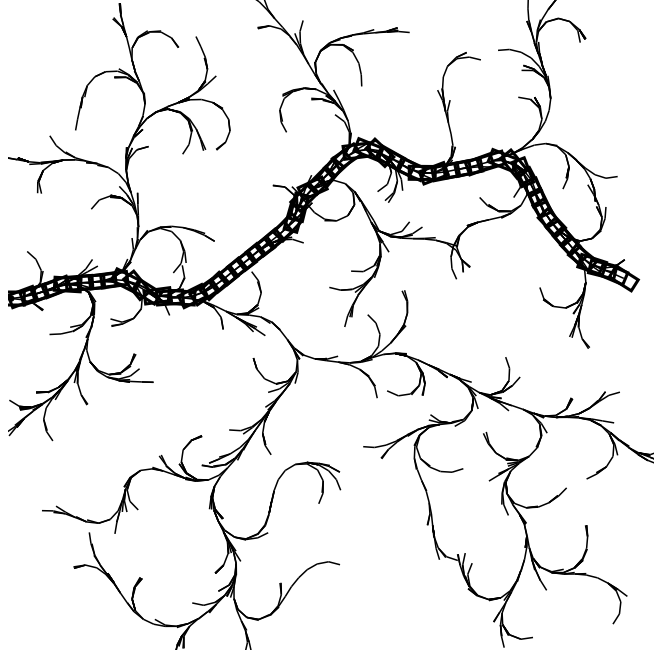


Figure 2.2: Paths taken and available for a robot using Rapidly exploring Random Tree (RRT) [112]

the learning phase is complete and a road map is constructed, the query or processing phase can commence. In this phase, $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ are appended to the constructed PRM, and any graph traversal algorithm may be used to seek paths connecting initial and final conditions [45, 73]. The flowchart detailing the learning and processing phases is shown in Figure 2.3. One may note that for single query problems, RRT is faster compared to PRM, as it completely avoids the learning phase seen in PRM [95].

SAs provide efficient solutions for path planning problems which may be challenging for deterministic algorithms. However, solutions generated by SAs are widely regarded as sub-optimal [45]. Sampling based planners are not complete, i.e., they are not guaranteed to find a solution if one exists. They do, however, ensure a weaker notion of completeness that is probabilistic completeness [45] as seen with RRT and PRM. That is, the probability of finding a solution converges to 1 as the running time approaches infinity. Sampling based planners can be optimized for path quality by modifying some of their main steps such as sampling strategies, exploration guidance, metrics, collision checking, heuristics, etc. These modifications may appear as a combination of attributes from PRM, RRT and CDGT methods, to achieve overall improvement in the path planning process, as seen in a detailed fashion in [45]. For example, the original PRM and RRT algorithms were proposed with uniform sampling schemes [45]; however, this has the disadvantage of sampling from a small region with just as much likelihood as that of sampling from a wider region. This may not capture the connectivity of the environment all that well. Therefore, modifications in the sampling procedure range from combining Voronoi Diagrams' medial axes as the region to sample, to forcing sampling from boundary regions, such as Goal Biasing (also seen later as Goal-Oriented sampling in Figure 2.5), Gaussian sampling, Bridge-test, hybrid sampling

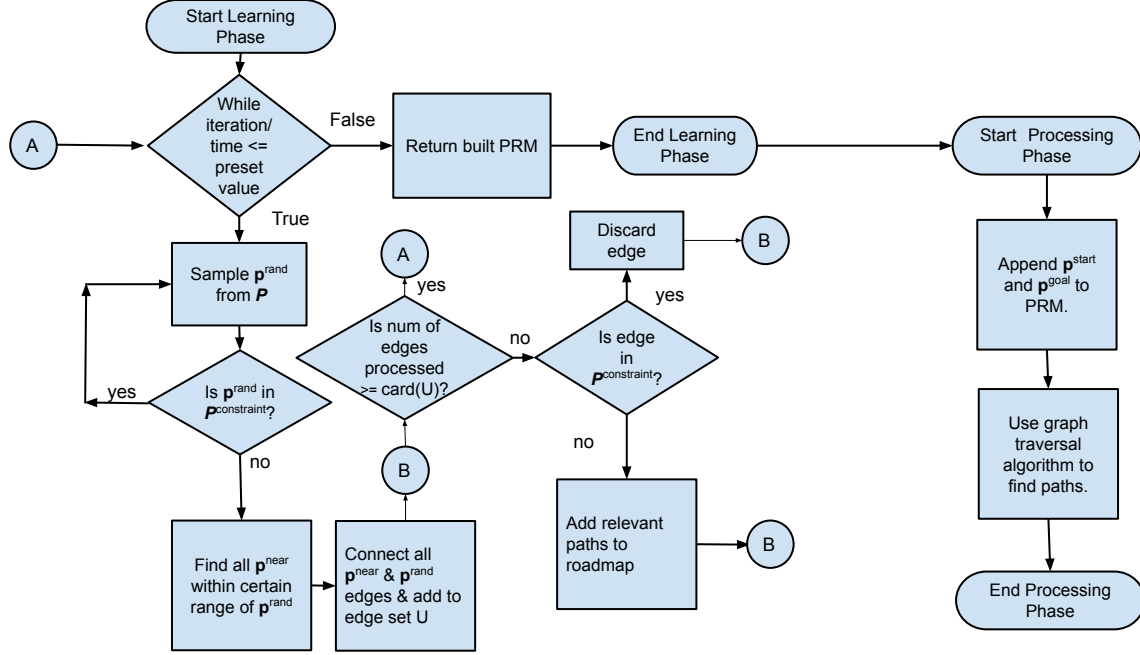


Figure 2.3: Flowchart of Probabilistic Road Map (PRM)

and Visibility PRM. An effect of changing the sampling strategy is the change in guiding the exploration either away from the obstacle or towards the destination. Some modifications, such as the Utility-RRT, change the direction and length of movement, while the Bidirectional-RRT expands the tree from both the source and the destination nodes. Yet another important attribute is that of the chosen metric, which is the value (time/energy/cost) of moving from one node to another. Metrics can and should be chosen for the specific PPP and robot at hand. A globally Euclidean-based distance metric is no match for a globally curved surface, as is assuming a uniform cost even at the boundary of the constraint region. Adaptive criteria for metrics have evolved several modifications of SAs as seen below. The most pertinent aspect of SAs remains to be collision checking. While it is advantageous in SAs to not have to explicitly define the constraint regions, it implies that every node must be evaluated for satisfying the constraint-free conditions to enable further processing. Most connections are collision free, but some experiments show that more than 90% of planning time is spent processing collision checking queries [62]. Lazy planning algorithms such as the Lazy PRM (LPRM) [5, 24] delay collision checking until it is necessary, typically doing so only if a path is found. In that case, the collision segment is removed and the path planning continues. The effect of sampling on the performance of SAs remains an open question in that there is no single sampling strategy that outperforms the others in every scenario [45]. For the purposes of this work, the focus lies mainly on how well SAs can account for the underlying topology of the sampled space and the fact that the node's compliance with the constraint-free region must be explicitly evaluated at every juncture.

It is important to note that true origin of path planning algorithms lies in classical grid based algorithms. Grid-based search is considered by many to be the most straightforward form of path planning [94]. The simplest example of partitioning $\mathcal{P}$ into $\mathcal{P}^{\text{constraint}}$ and $\mathcal{P}^{\text{free}}$

can be seen in grid based representations of the path planning space. In its barest form, every grid cell in the discretized grid map is evaluated for collisions, and the status of the region being occupied or unobstructed is updated in the map. The process is deterministic and the grid map becomes more extensive as the number of dimensions of the space increases, or is not Euclidean. Even its simplest form, the partitioned space needed graph infrastructure to store relationships amongst grid cells, i.e., connectivity of the underlying space. Consequently, graph traversal algorithms, such as A^* , were used to find paths. An example of such algorithms can be seen in [100]. Thus, a given query is quantized, and the grid-map (otherwise known as a bitmap) can then be treated as a graph which is searched using algorithms such as dynamic programming, A^* , D^* [157] best-first, or bidirectional search, to connect $\mathbf{p}^{\text{start}}$ to $\mathbf{p}^{\text{goal}}$. In fact, the bitmap can also be searched using recent path planning methods that are based on incremental search, such as randomized potential fields [94] (will be seen in section 2.1.2). Therefore, it is easy to see that classical grid based representations of the environments, such as cell decompositions, Voronoi Diagrams and occupancy grids [140], work in conjunction with other graph traversal algorithms. Additionally, most CDGT methods are fundamentally SAs by nature, as they rely on sampling to create grid-based representations. Therefore, CDGT are used either as helpful qualifiers, heuristics or metrics in conjunction with the modifications seen with PRMs and RRTs, as seen in the literature below.

Kavraki et al. in [76] explains PRM in two phases: learning and query, as seen in Figure 2.3. In the learning phase, a probabilistic road-map is constructed and stored as a graph whose nodes correspond to collision-free configurations and whose edges correspond to feasible paths between these configurations. In the query phase, the road-map is then searched for a path joining these two nodes corresponding to the start and goal nodes. The learning and the query phases can happen simultaneously. This seminal work in [76] paved the way for further development. There has been a steady introduction of randomized motion planners since then (for example [24, 34, 35, 94]). These algorithms allowed effectively facing problems in high-dimensional spaces, that were not always solvable by formerly developed planners [30].

Consider the work described in [35], where optimal, constraint-free paths are planned using a clustering algorithm and Grassman’s line geometry. The singularity points are determined inside the workspace then grouped into several clusters, by the clustering algorithm. These clusters are representations of obstacles. Subsequently, a road-map is employed to find an optimal path around said obstacle clusters. Should a singularity point lie on or very close to the path, the path is restructured to avoid that point by a local routing method based on Grassmann’s line geometry. The work poses an approach to demarcate valid sampling regions from those that should not be sampled, since the faster constraint free regions get sampled, the better the probability of a path being found. Improvements to sampling strategies will be seen later, in conjunction with either PRMs or RRTs.

A different approach can be seen in [30], where a new motion planner is developed based on random walks. Carpin et al. of [30] note that random walks have been sporadically used and only in particular situations; for example, to escape local minima in potential field methods [10]. Many randomized planners which have been developed so far draw independent samples from an apriori fixed probability density function whose support is the entire configuration space. The aim is then to build a graph or a tree which captures its

connectivity [76, 89]. In [30] however, a new sample is generated in the neighborhood of the last point, in accordance with a Gaussian distribution. If the segment connecting them lies entirely in the free configuration space $\mathcal{C}^{\text{free}}$, the point becomes the last point in the walk, otherwise it is discarded. A bidirectional search is employed to speed the process. When the algorithm terminates, a path connecting the source and target points is generated. A smoothing procedure is then used to smooth out unnecessary intermediate motions that affect the optimality of the path. Since a Gaussian distribution is used for sampling (rather than a uniform distribution), as samples are generated, the distribution also adapts based on accumulated samples. The covariance of the distribution is updated based on a specific number of samples, in order to enable better sampling. So, when the variance of the samples through which the robot is currently moving is too large, it permits movements to fall outside the free configuration space. In such a case, the algorithm is able to automatically reduce the variance of its steps, thus augmenting the probability of exploring a narrow space, e.g., a corridor, in a more effective and rapid way. The time taken to generate a sample does not depend on the size of the previously generated sample set, provided that the update of the covariance matrix can be done in constant time [30]. This can be done by processing a fixed number of samples at every point, unlike most formerly developed algorithms, such as PRMs and RRTs whose performance is worse than linear in the number of generated samples. Nonetheless, [9] proposes a version of the RRT algorithm where the time needed to generate a new sample is logarithmic in the number of already generated samples.

Low-dimensional spaces saw a combination of discretizing the space into grids and using RRTs as path planners [37, 48, 89, 96], as mentioned in [66]. Authors of [31] even used pre-defined way points to better suit the local path planning around static obstacles. As an alternative method to evaluate the candidacy of the newly generated points, authors of [66] present a general algorithm called Transition-RRT (Transition-based RRT (T-RRT)). It combines the exploratory strength of the RRT with an effective test that determines if a new sample is acceptable or not. The test used is called the transition test and is used in stochastic optimization methods to accept or reject new potential states. The filtering of transition tests relies on the gradient of the cost function along the local motion to connect a given state to the RRT. This results in an expansion biased to follow the valleys and the saddle points of the configuration-space cost-map as seen in Figure 2.4.

Karaman and Frazzoli of [73] focus more on the optimality of PRM and RRT, and propose improvements to both algorithms on account of better optimality. They prove that the standard PRM and RRT algorithms are not asymptotically optimal, and that the simplified-PRM as seen in [75] algorithm is asymptotically optimal, but computationally expensive. It is also shown that the k-nearest variant of the simplified-PRM algorithm is not necessarily probabilistically complete, and is not asymptotically optimal for cases. To address this problem, Karaman and Frazzoli of [73] propose new algorithms, Probabilistic Road Map* (PRM*), Rapidly exploring Random Graph (RRG), and Rapidly exploring Random Tree* (RRT*), and prove them to be probabilistically complete, asymptotically optimal, and computationally efficient. The PRM* is a batch variable-radius PRM, applicable to multiple-query problems, in which the radius is scaled with the number of samples in a way that provably ensures both asymptotic optimality and computational efficiency. The RRG is an incremental algorithm that builds a connected road-map, providing similar performance to

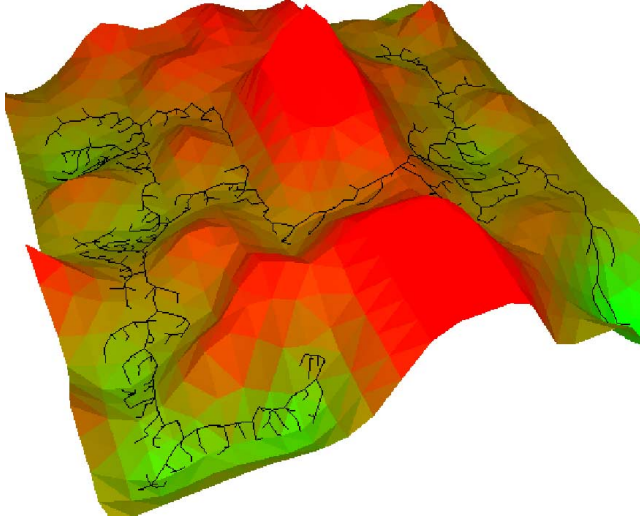


Figure 2.4: Transition-based RRT (T-RRT) on a cost-map [66]

PRM* in a single-query setting, and in an anytime fashion (i.e., a first solution is provided quickly, and monotonically improved if more computation time is available). The **RRT*** algorithm is a variant of the **RRG** that incrementally builds a tree. It provides anytime solutions and provably converges to an optimal solution, with minimal computational and memory requirements.

Most sampling-based planning algorithms, such as **PRMs** [24, 76] or **RRTs** [13, 41, 67, 73, 89] rely on randomly sampling $\mathcal{C}$ and building a graph connecting those configurations. Validity of local paths is usually established by invoking collision checking routines. While graphs resulting from **PRMs** and **RRTs** produce collision-free paths, no volumetric information about $\mathcal{C}$ is available. A simple solution is to incorporate so-called bubbles of free $\mathcal{C}$ [132], i.e., $\mathcal{C}^{\text{free}}$. The bubble is a collision-free local volume around a given configuration [131, 132]. The bubble is computed using a minimum distance between the robot and the set of obstacles in the workspace. This distance may be observed as a "safety" margin that provides rough information about how far the robot can move from a specific configuration without colliding with any of the obstacles. The authors of [2] describe a simple algorithm that uses bubbles of $\mathcal{C}^{\text{free}}$, to compute collision free paths for robotic manipulators. The algorithm mimics a classical RRT-Connect from [89], yet uses modified procedures for extending and connecting the trees via bubbles.

It was seen in [73] that **RRT*** was proposed as an extension of the **RRT** algorithm that provides a collision-free asymptotically optimal path regardless of obstacle geometry in a given environment. However, one of the limitations in the **RRT*** algorithm is slow convergence to optimal path solution [136]. To overcome the limitations due to speed, authors of [136] propose the potential function-based-**RRT*** that incorporates the **Artificial Potential Field (APF)** algorithm in **RRT***. Biasing the probabilistic planner towards the direction of the goal while accommodating the workspace terrain was seen in [66], and more efforts in that direction are also seen subsequently. Directionalizing random samples through **APF** was initially introduced as **Potential Guided Directionalized-RRT*** (**PGD-RRT***) [133,

134]. While PGD-RRT^* finds an initial path faster than RRT^* , it may fail to converge to an optimal path. The proposed $\text{Potential Functions-based RRT}^*$ (P-RRT^*) in [136] is a variant and an extension of the previously proposed PGD-RRT^* [133, 134]. The use of APF in P-RRT^* directs random samples in the direction of decreasing potential to provide a quick, optimal solution. Moreover, this also results in lesser dispersion of samples in the configuration space and provides a more memory-efficient solution operating at a much faster rate compared to RRT^* [136]. As seen before with Gaussian distribution based sampling in [30], the aforementioned works focus on manipulating the sampling to take place in the appropriate region of the PPP . This steady change in sampling strategy showcases SAs ' main disadvantage, which is the inability to sample from the true PPP , which should be $\mathcal{P}^{\text{free}}$.

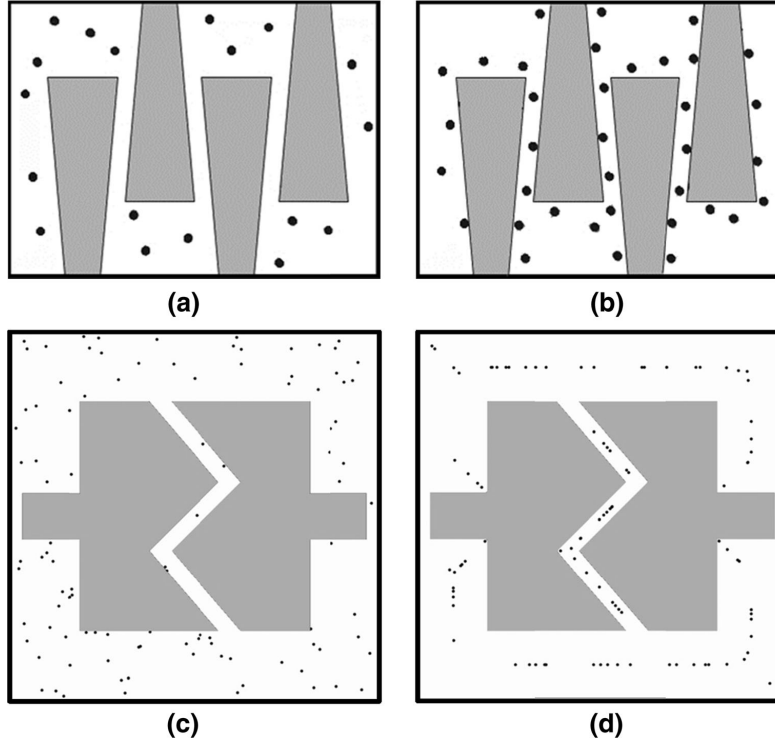


Figure 2.5: Illustration of uniform and Goal-Oriented sampling in different sampling environments; uniform sampling is seen in (a) & (c) and Goal-Oriented sampling in (b) & (d) [5]

The advantage of SAs is easily seen, in the form of reliability in a wide range of applications in robotics, including Unmanned Aerial Vehicle (UAV) collision avoidance. Authors of [102] propose a method based on Closed-Loop Rapidly-exploring Random Tree (Closed-Loop RRT) [91]. Improvements to node connections and adding more candidate paths is explored by utilizing information about the set of allowed states for the vehicle and obstacle. RRTs are greatly influenced by how a tree is connected or trimmed, which influences the efficiency of path planning. Authors of [172] propose a novel $\text{Machine Learning-based Multi-RRT}$ (LM-RRT) for robot path planning in narrow passages. The LM-RRT approach models the tree selection process as a multi-armed bandit problem and uses a reinforcement learning algorithm that learns action values and selects actions with an improved greedy

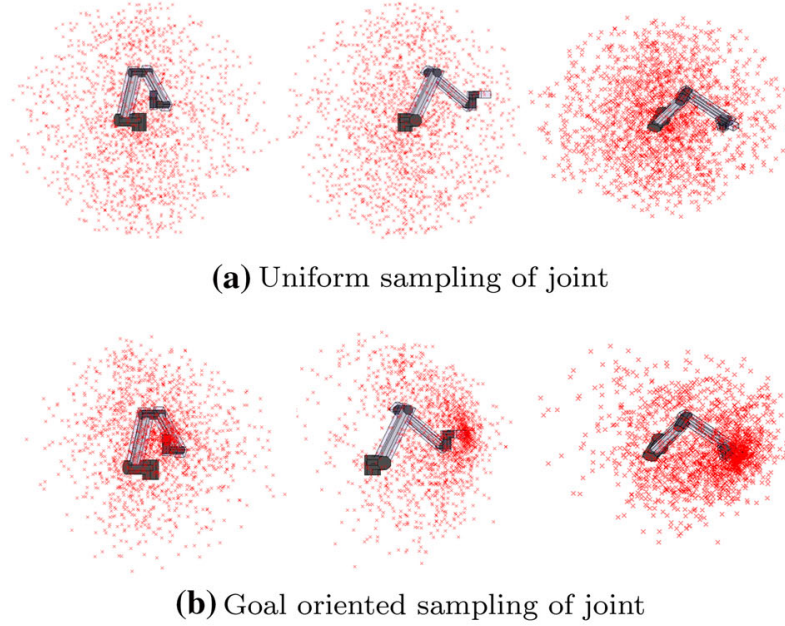


Figure 2.6: Illustration of uniform and goal-oriented sampling in the configuration space of a manipulator [72]

strategy. Compared to previous [RRT](#) algorithms, the [LM-RRT](#) can not only enhance the local space exploration ability of each tree, but also guarantees the efficiency of global path planning [172]. Yet another improvement to the [RRT](#) is presented in [159], with a focus on improving the need to sample extensively before extending a tree locally. Authors of [159] introduce an efficient nonmyopic (plan ahead for possible observations that can be made in the future) path-planning algorithm by combining [RRT*](#) and a stochastic optimization method, called cross entropy. The proposed method constructs two [RRTs](#), where the first tree is a standard [RRT*](#) tree. It determines the nearest node in the tree to be extended to a randomly chosen point. The second tree contains the first tree with additional long extensions. By maintaining two separate trees, one can grow the search tree nonmyopically to improve the efficiency of the algorithm while ensuring the asymptotic optimality of [RRT*](#) [159].

As seen before, [RRT*](#) was proposed in [73] as a variant of the [RRT](#), which generates an optimal or near optimal path, as the number of iterations tends to infinity [73, 135]. However, slow convergence of [RRT*](#) has led to further analysis of its limiting constraints. Noreen et al. [120] explicitly address slow convergence towards the path by improving early rejection of beneficial samples, avoidance of adding far-away samples, and reducing the number of nodes in the tree. An off-line algorithm called [RRT*-Adjustable Bounds \(RRT*-AB\)](#) is proposed to overcome the above issues. [RRT*-AB](#) performs informed exploration in a known environment and generates a highly converged tree populated with useful nodes.

As [RRT](#)'s main competing technique, [PRM](#) also underwent its share of improvements. Akbaripour et al. [5] propose a new variation of sampling-based methods called [Semi-Lazy PRM \(SLPRM\)](#), which boasts of advantages from [PRM](#) and [LPRM](#) methods. Demonstrated with industrial manipulators, the [SLPRM](#) collision-checks random configurations for only a

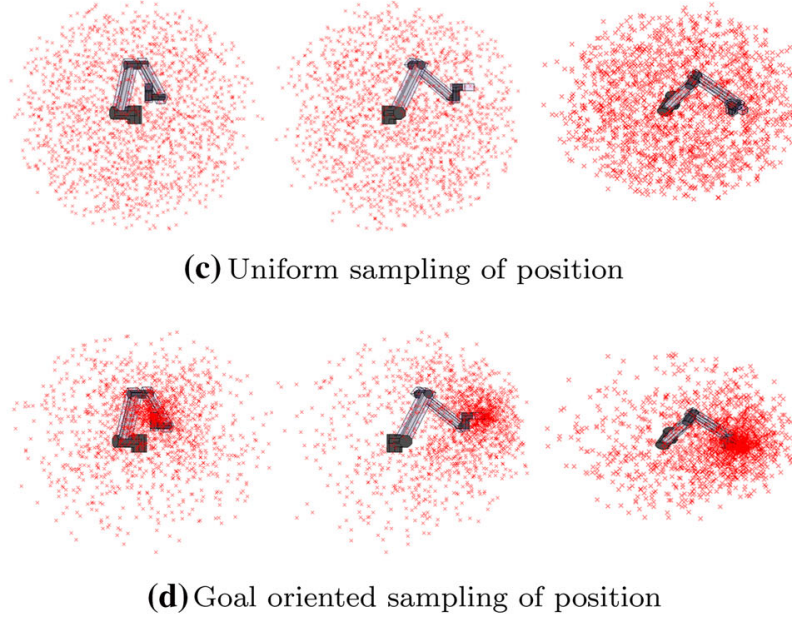


Figure 2.7: Illustration of uniform and goal-oriented sampling in the workspace of a manipulator [72]

subset of terminal links of the manipulator in the road-map construction phase. This is opposed to the exhaustive and zero collision-checking policies implemented respectively in PRM and LPRM. This change in sampling a subset of manipulator parameters, results in a reduction of road-map construction time compared to the PRM and in query time compared to the LPRM.

Sampling methods have the advantage of being extended to less structured environments, however in their most basic forms, may not be competent to do so [173]. Thus, the authors of [173] propose an obstacle avoidance algorithm which is an improvement to the basic RRT, called Smoothly-RRT (S-RRT). This method targets a directional node, and can extend and increase the sampling speed and efficiency of RRT dramatically. The maximum curvature constraint is presented as a path optimization strategy to generate a smooth, curved and continuous path, that is executable for a manipulator. Targeting the RRT towards the goal is explored in [72], by introducing the Goal-Oriented (GO) sampling method in conjunction with the RRT. The area over which the sample points are distributed is restricted and the probability distributions of sample points near the goal configuration are modified. This biasing ensures that the initial path is found quickly, where the authors of [72] focus on using the reduction of computation time as the primary rubric. The difference between uniform and GO sampling can be seen in Figure 2.5; illustration of the concepts in the context of manipulator joint configuration and EE pose can be seen in Figure 2.6 and Figure 2.7 respectively. Further enhancements in the sampling procedure and in turn, in the construction of the PPS is seen in [109]. Authors of [109], use the concept of reachable volumes (set of reachable positions around the origin of a joint), to design a better sampler, that aims to sample the constrained space for better path planning. Thus far, SA have been shown to be versatile algorithms that seldom need a distinct characterization of $\mathcal{P}^{\text{free}}$ and $\mathcal{P}^{\text{constraint}}$ and

are probabilistically complete. However, most improvements to fundamental [SAs](#) seemingly encourage [SAs](#) to make the distinction between $\mathcal{P}^{\text{free}}$ and $\mathcal{P}^{\text{constraint}}$, innate. They aim to make the sampling process better, by forcing the sampling to take place in $\mathcal{P}^{\text{free}}$ as much as possible and by directing the sampling towards the goal (path convergence speed).

2.1.2 Optimization-Based Methods

[Optimization Algorithms \(OAs\)](#) lend themselves to an intuitive perception of the robotic [PPP](#). Planning a path aims to produce the shortest path, i.e., minimize the path length/time and to be as far away from constraints as possible, i.e., maximizing distance from constraints. Such minimization and maximization processes have led to several variants of [OAs](#), where the variance is a result of one or a combination of the following parameters. First, the intrinsic purpose of the [PPP](#) is an important factor in determining the variant of the [OA](#) used. They can involve any of the following: planning the shortest path from a source to a target configuration; finding paths that avoid obstacles; or determining path feasibility across multiple locations on the [PPS](#). Second, the nature of the environment influences the choice of the [OA](#), i.e., the planar or non-planar nature of the [PPS](#). Third, the size of the [PPS](#) must be considered, since in that case, [OAs](#) are combined with [SAs](#) for an ample representation of the environment. Fourth, the number of robots used in the application directly influences the optimization approach of choice, since the [OA](#) must account for multiple optimization processes occurring at once. Naturally, combinations of variants are introduced to abate demerits that would occur otherwise. A general list of algorithms aiming to minimize a cost function for robotic path planning are [APFs](#), [Evolutionary Algorithms \(EAs\)](#), [Model Predictive Control \(MPC\)](#) and [Navigation Functions \(NFs\)](#). Seen either on their own or as a combination, all of the aforementioned methods are fundamentally iterative optimization based approaches. [EAs](#) can be viewed as a combination of sampling and optimization, and have further evolved into a multitude of variants. In an effort to reduce the sampling space, [MPC](#) algorithms were explored. [MPC](#) algorithms combine knowledge of the robot model and then use a variety of optimization techniques in the space of model allowed motions. [NFs](#) have been the most direct solutions to combat [APFs](#)' disadvantage of local minima.

2.1.2.1 Artificial Potential Fields

[APFs](#) are some of the most frequented approaches to path planning with obstacle avoidance. Pioneered by Khatib in [78], artificial potentials for obstacle-free path planning can be described as the following. Assume a point representation for the robot as $\mathbf{q}$ in the configuration space $\mathcal{C}$. Then, the robot at $\mathbf{q}$ is subject to the influence of a potential function $U_{tot}(\mathbf{q})$, where

$$U_{tot}(\mathbf{q}) = U_{att}(\mathbf{q}) + U_{rep}(\mathbf{q}) \quad (2.3)$$

$U_{tot}(\mathbf{q})$ is the superposition of $U_{att}(\mathbf{q})$ and $U_{rep}(\mathbf{q})$. $U_{att}(\mathbf{q})$ is a potential whose influence is attractive towards the goal $\mathbf{q}^{\text{goal}} \in \mathcal{C}^{\text{free}}$. $U_{rep}(\mathbf{q})$ is a potential whose influence is repulsive, away from the [CO](#) region [122, 154]. The influence or force $\mathbf{f}(\mathbf{q})$ of any potential function

$U(\mathbf{q})$ is given by its negative gradient, such that

$$\mathbf{f}(\mathbf{q}) = -\nabla U(\mathbf{q}) \quad (2.4)$$

From (2.3), consider $U_{att}(\mathbf{q})$ first. Its potential is calculated using (2.4):

$$\mathbf{f}_{att}(\mathbf{q}) = -\nabla U_{att}(\mathbf{q})$$

For $U_{att}(\mathbf{q})$ to produce an attractive influence $\mathbf{f}_{att}(\mathbf{q})$ on $\mathbf{q}$ towards $\mathbf{q}^{goal}$, $U_{att}(\mathbf{q})$ is modeled as a positive semi-definite function. Its unique minimum value of 0 must be located at $\mathbf{q} = \mathbf{q}^{goal}$. Then, one can construct $U_{att}(\mathbf{q})$ as a positive semi-definite function of the error vector $\mathbf{e}(\mathbf{q})$, defined as

$$\mathbf{e}(\mathbf{q}) = \mathbf{q}^{goal} - \mathbf{q}$$

That is, the general form of $U_{att}(\mathbf{q})$ is given by

$$U_{att}(\mathbf{q}) = U_{att}(\mathbf{e}(\mathbf{q})) \quad (2.5)$$

It can then be seen that $\forall \mathbf{q} \in \mathcal{C}$, the minimum value of $\mathbf{e}(\mathbf{q}) = 0$ is reached under the influence of $\mathbf{f}_{att}(\mathbf{q})$, when $\mathbf{q} = \mathbf{q}^{goal}$. Examples of such attractive potentials include paraboloidal and conical profiles, as shown in Figure 2.8. A similar analogy is used for $U_{rep}(\mathbf{q})$ where $U_{rep}(\mathbf{q})$ must produce a repulsive influence $\mathbf{f}_{rep}(\mathbf{q})$ away from $\mathcal{CO}$. The potential of $U_{rep}(\mathbf{q})$ is calculated using (2.4) and is given by

$$\mathbf{f}_{rep}(\mathbf{q}) = -\nabla U_{rep}(\mathbf{q})$$

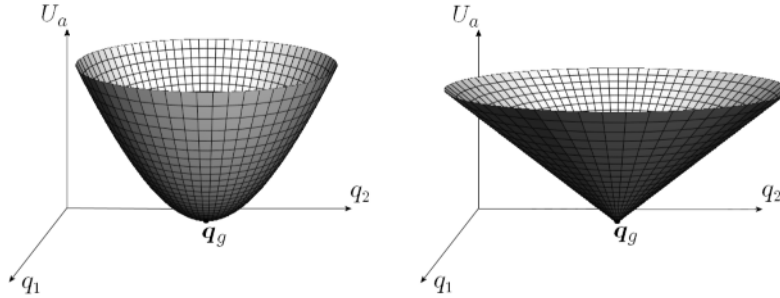


Figure 2.8: Paraboloidal and Conical attractive potentials [154]

Assume that the obstacle-region is convex and/or non-convex obstacles have been partitioned into convex components. For illustration purposes, assume one convex obstacle denoted by $\mathcal{CO} \in \mathcal{C}$. Then, $U_{rep}(\mathbf{q})$ is modeled such that $\mathbf{f}_{rep}(\mathbf{q})$ behaves as a barrier in the vicinity of $\mathcal{CO}$, and has no influence outside that vicinity. Denote the vicinity of $\mathcal{CO}$ as η_0 , formally referred to as the range of influence [154]. Further, denote the minimum point-set distance between the robot configuration $\mathbf{q}$ and $\mathcal{CO}$ as $\eta(\mathbf{q}) = \min_{\mathbf{q}' \in \mathcal{CO}} \|\mathbf{q} - \mathbf{q}'\|$. For $U_{rep}(\mathbf{q})$ to produce a repulsive influence in the region of influence η_0 and no influence outside it, the function's value must be 0 outside η_0 and very high inside it. Then, $U_{rep}(\mathbf{q})$ takes the following general form:

$$U_{rep}(\mathbf{q}) = \begin{cases} u(\eta_0, \eta(\mathbf{q})), & \text{if } \eta(\mathbf{q}) \leq \eta_0 \\ 0, & \text{if } \eta(\mathbf{q}) > \eta_0 \end{cases} \quad (2.6)$$

where $u(\eta_0, \eta(\mathbf{q}))$ is any function such that as:

$$\lim_{(\eta(\mathbf{q}) - \eta_0) \rightarrow 0} u(\eta_0, \eta(\mathbf{q})) \rightarrow \infty$$

This is clearly illustrated in Figure 2.9, where a repulsive field around an obstacle is computed [146]. The resulting equi-potential contours around a polygonal obstacle can be seen in Figure 2.10.

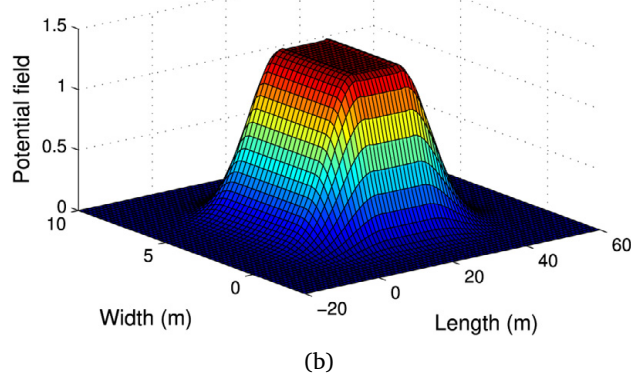


Figure 2.9: Repulsive potential [146]

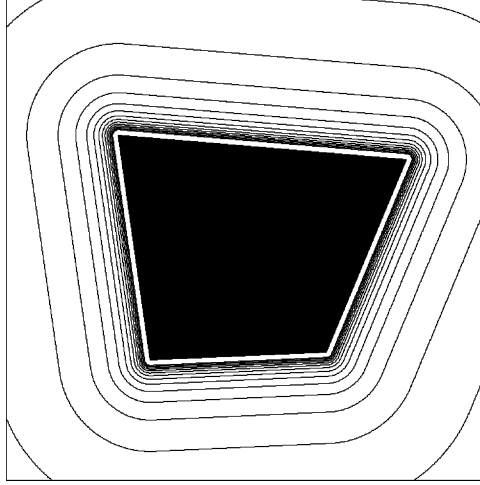


Figure 2.10: Equipotential contours of a polygonal obstacle [154]

Thus, the resulting potential corresponding to $U_{tot}(\mathbf{q})$ can be expressed as

$$\mathbf{f}_{tot}(\mathbf{q}) = \mathbf{f}_{att}(\mathbf{q}) + \mathbf{f}_{rep}(\mathbf{q}) \quad (2.7)$$

where $\mathbf{f}_{rep}(\mathbf{q})$ acts on $\mathbf{q}$ to move away from $\mathcal{CO}$ and $\mathbf{f}_{att}(\mathbf{q})$ influences $\mathbf{q}$ to attract it towards $\mathbf{q}^{goal}$. An example of a repulsive potential added to an attractive well is illustrated in Figure 2.11.

The general form of the APF allows freedom to define $U_{rep}(\mathbf{q})$ and $U_{att}(\mathbf{q})$, while still respecting the definitions of the potential fields. Such flexibility can benefit different applications and sometimes combat disadvantages. Typical definitions of the attractive potential

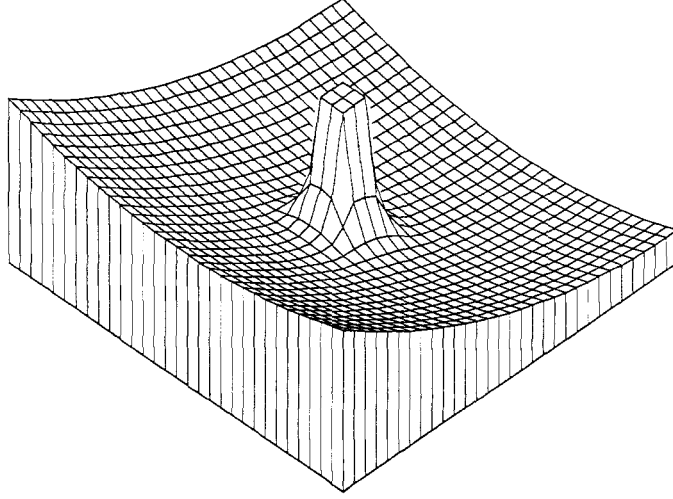


Figure 2.11: Repulsive potential added to an attractive potential [166]

resemble $U_{att}(\mathbf{q}) = \frac{1}{2}k_a\|\mathbf{e}(\mathbf{q})\|^2$, if defined in $\mathcal{C}$, for some scalar $k_a > 0$ [154]. Khatib [78] uses APFs in the workspace $\mathcal{W}$ as opposed to the configuration space $\mathcal{C}$. The attractive potential is defined as $U_{\mathbf{x}^{goal}}(\mathbf{x}) = \frac{1}{2}k_p(\mathbf{x} - \mathbf{x}^{goal})^2$, for some scalar $k_p > 0$, which is similar to (2.5). U_{rep} is defined in $\mathcal{W}$ for obstacles characterized as n-ellipsoids. Khatib shows the effective use of APFs to avoid both obstacles and joint constraints; however, local minima may occur. This happens when the repulsive potential has equipotential contours with lower curvature than the attractive potential in the same area [154].

Even as early as 1987, solutions were being explored to avoid the local minima problem. Authors of [166] modeled obstacles using modified elliptical potential functions, while avoiding local minima by using a spherically symmetric attractive wells. The elliptical avoidance potentials surround a rectangular object to produce isopotential contours that tend to infinity at the boundaries of the obstacle surface, and morph into the symmetrical, circular attractive wells away from the obstacles. Therefore, the transition from an elliptical repulsive function to an attractive circular well enables better navigation towards the goal; this transition is seen in Figure 2.12. Faverjon and Tournassoud in [47] argue that APFs can work properly when only few obstacles are present in the environment, as opposed to complex environments. Since terms of different nature are arbitrarily added in a single function in APFs, problems appear in more complex environments. These include oscillations between opposite obstacles, arbitrary higher repulsion for two adjacent obstacles than for a unique one and impossibility to get close to an obstacle. In [47]’s approach, the task is described by a minimization problem and eventually, some geometric constraints. As opposed to APFs, the anti-collision aspect is translated into geometric constraints in $\mathcal{C}$ and is not included in the function to minimize. While attempts were being made to modify the APF as a whole or in parts, Koditschek [84] was one of the first authors to investigate topological properties of the obstacle filled environment in light of APFs. Using Morse theory (theory about directions where the function increases in value), he shows that there is some fundamental topological obstruction preventing the construction of a gradient vector field with a single

minimum. Using this knowledge, Kodistchek defines an “almost global” navigation algorithm as a smooth real valued function on $\mathcal{C}$ for total energy η if the following is satisfied. The function exceeds the value η and is exterior directed on the boundary of the space, and contains only a single local minimum coinciding with the desired target point. The algorithm multiplies a parabolic valley centered at the desired destination with a sum of m Gaussian distributions placed over the center of each of the m obstacles, represented as disjoint discs of finite radius (less than their distance to the goal). An algorithm based upon the gradient of such a function carries the guarantee that all trajectories avoid the obstacles, and excepting a set of zero measure initial conditions, all other initial conditions arrive at the destination. While the difficulty of constructing such a function is prevalent, it is expected that solutions can be found.

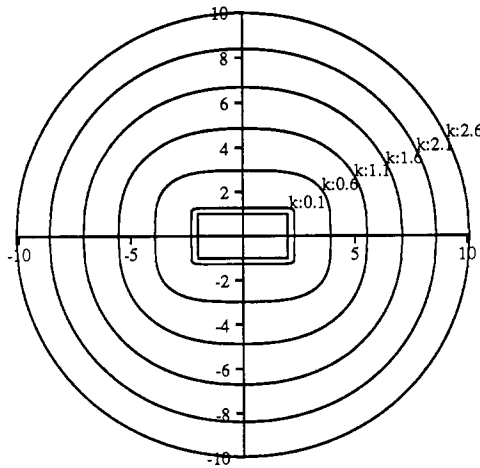


Figure 2.12: Isopotential contours changing from elliptical repulsion around rectangular obstacle to circular attraction away from it. [166]

Volpe and Khosla of [79] improve on their previous work in [166], by introducing superquadratic formulation (similar to quadratics, but instead formulated with arbitrary powers as opposed to squares) for safe approach towards objects. This is a generalization of the elliptical potential function method seen in [166], and therefore viable for a much larger class of object shapes, as opposed to just rectangular obstacles. The superquadratic repulsive field is then added to the attractive well. Similar to [166], since the superquadratic avoidance potential only becomes a circle asymptotically as seen in Figures 2.13 and 2.14, a spurious minimum may be present. However, this minimum can effectively be removed by making the depression associated with it smaller than the resolution of the system or by adjusting parameters that define the quadratic well. As an improvement, Volpe and Khosla presented two compatible forms of the superquadratic potential function; one for obstacle avoidance, and another for obstacle approach in [167]. APFs have also been extended to path planning for multiple robots [99]. To avoid local minima while using APFs for path planning, authors of [4] propose a new repulsive potential developed using the potential density along the geometry of the obstacle. A typical path obtained by traditional APF gradients, are geodesics (locally, paths of the shortest lengths) projected on to the plane. However, such paths may run into local minima. So, the authors of [4] instead propose the quasi-geodesic

method, where the remaining shortest-distance path from a given point of the curve, to the target, is approximated with a straight line. Optimization for PPP has seen different kinds of solutions to tackle the local minima problem, which can broadly be classified into MPC, NFs, EAs, and discretized combinations of the above. A brief overview of the literature in the area is provided below.

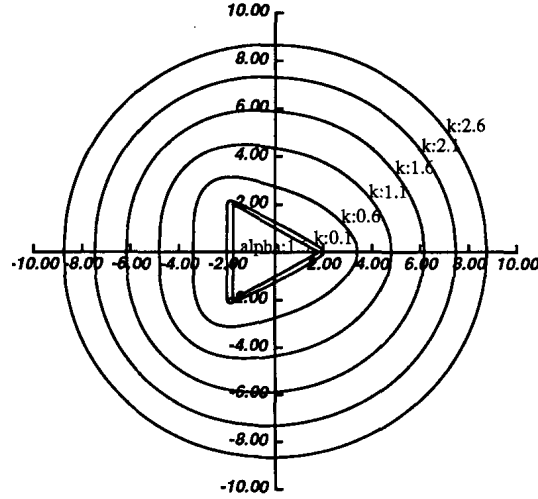


Figure 2.13: Isopotential contours changing from superquadric repulsion around triangular obstacle to circular attraction away from it. [79]

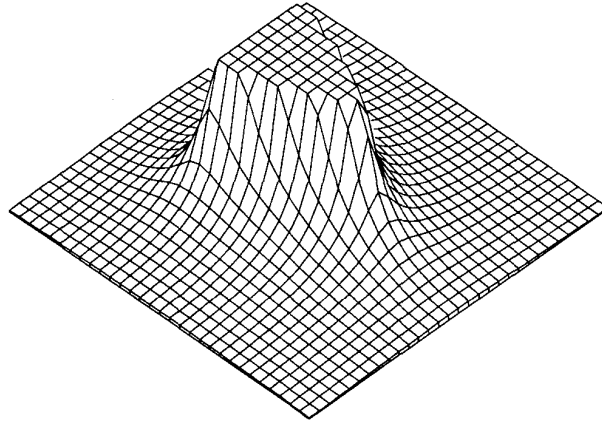


Figure 2.14: Superquadric touching a triangular obstacle's edges [79]

2.1.2.2 Model Predictive Control

In an attempt to produce an optimized path specific to a robot model, the authors of [123] apply MPC. The work uses a robot model with a certain look-ahead horizon and is provided with a pre-defined set of way points from the source to the target. An optimization function with performance indices to be minimized include the deviation from the pre-planned path,

control input and an additional cost term that reflects local obstacle threat. This optimization limits the immediate directions of motion and attempts to optimize locally within the look-ahead horizon. The nature of the optimization function and the MPC together lead to less local minima than the APF [123]. A similar approach of optimizing a path towards a pre-defined path is also seen in [7]. Yoon et al. from [181] also use MPC for an optimal collision-free trajectory formulated in terms of cost minimization. With APFs, rapid computation for trajectory regeneration is possible, but collision with obstacles can happen due to local minima or difficulty to incorporate constrained dynamics over a longer time duration. The results with MPC approaches show that the application of distance-based methods may require excessive computation especially in complex environments. Modifications aimed towards better computational efficiency in MPC is seen in [181], where the modified local path generator called Modified-Parallax MPC (MP-MPC) effectively utilizes information about a Unmanned Ground Vehicle (UGV), shows high computational efficiency and better characteristics in path length and control efforts. Thus, the proposed MP-MPC approach can be a good candidate for real-time obstacle avoidance of autonomous UGVs in complex environments [181]; however it is model specific. More examples of potential field-based MPC for autonomous path planning can be seen in [145, 146, 152]. MPC methods and other Non-Linear Optimal Control (NLOC) techniques use information about the system model and subsequently minimize cost functions with performance indices. Some examples of MPC and NLOC used for autonomous driving while avoiding obstacles can be seen in [1, 51]. Solving Non-Linear Optimal Control Problem (NOCP) yields a Non-Linear Optimization Problem (NOP). The resulting NOP is generally solved using iterative methods, that are a combination of sampling and processing the sampled data. EAs are some of the most popular algorithms that sample data and process them with an aim of optimizing a certain feature while respecting constraints. While EAs can be used on their own, path planning has seen their usage in conjunction with other methods, such as APFs, so as to improve the avoidance of local minima [114].

2.1.2.3 Evolutionary Algorithms

To overcome problems of local optima and solving NP-hard problems with a large number of variables, researchers have propounded EAs for searching near optimal solutions to such problems [3]. EAs typically follow three main steps: sampling a point, evaluating the strength of that point, and finally applying operators on the point to generate a new one. This cycle is repeated until the strength or the corresponding solution of the point is deemed satisfactory.

Aghababa from [3] has used NLOC to find optimal paths for under water vehicle path planning with static obstacles. As part of the optimization procedure, five EAs have been applied to solve the non-linear optimal control problem. The EAs compared include Genetic Algorithm (GA), Memetic Algorithm (MA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Shuffled Frog Leaping Algorithm (SFLA) whose brief descriptions are as follows. To use a GA for problem solving, a cost function must be defined for evaluating potential solutions, together with a suitable representation (real, binary, etc.) for them, and with crossover and mutation operators which must manipulate solutions in the chosen representation. Both operators manipulate the existing solution in the direction of

optimized cost or maximum compliance with the solution’s fitness. The **MA** differs from the **GA** in that it is population-based. That is, first, an initial population is created at random. Then, a local search is performed on each population member to improve its experience and thus obtain a population of local optimum solutions. Subsequently, crossover and mutation operators are applied, similar to **GAs** to produce offspring which are then subjected to the local search so that local optimality is always maintained. **PSO** is yet another population based algorithm, but is also stochastic in nature. A swarm of particles is optimized locally and globally in the search space. Each particle in the swarm has the freedom to locally search its space and communicate its movement with the other particles. Particles in the swarm move toward that particle in the swarm with the best reward/energy. The final optimized solution is influenced by the particle’s local search and by other particles in the swarm, i.e. global search. **ACO** is similar to **PSO** in that it involves socializing behaviour, but this time, with reinforcements for positive behaviour and lack thereof, otherwise. The solution is updated until it has met the needs of the constrained optimization problem. **SFLA** combines the advantages of genetic-based **MA** and the socially driven **PSO** algorithms. In **SFLA**, the population composes of a set of frogs (solutions) further divided into subsets, called memplexes. The different memplexes are considered as different groups of frogs, each performing a local search. Within each memplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of **MA**. After a defined number of memetic evolution steps, ideas are passed among memplexes in a shuffling process. The local search and the shuffling processes continue until defined convergence criteria are satisfied. While all **EAs** found paths in an environment with obstacles, **PSO** produces the most optimal path in the fastest time [3].

Use of **PSO** for static path-planning can also be seen in [170]. An interesting solution to keep track of local minima is seen in [81] where the obstacle free space is discretized into cells and the intrinsic cost of every cell is pre-calculated. This enables the robot to always find a path while avoiding high costs representative of constraints such as obstacles. The disadvantage of course, is the cost of recalculating the cost for the grid every time a target is changed. Others have combined an improved **APF** with other collision prediction models as seen in [171]. The merits of using **EAs** for last mile optimization in path planning algorithms offers merits of flexibility to static/dynamic environments, vehicle model and possibly noise. Akin to tune-able constants and parameters used in **APFs**, all **EAs** have parameters that must first be determined by trial and error. Furthermore, there is no guarantee that the tuned parameters will work for new environments. As seen in [3], solutions reach the target position with a 10% margin of error. Additionally, some **EAs** consistently perform updates to an entire population for an iteration which can be an excessive computational cost.

2.1.2.4 Navigation Functions

All optimization methods focus on representing the problem space as different levels of cost or energy, and points in the **PPS** that share the same cost or energy are located on equipotential contours. Therefore, most path planners used on optimization problems focus on some variant of gradient descent procedure. Paths generated using **APF** and **MPC** as such may sometimes be longer than necessary due to the random nature of sampling, but are

still dependent on gradient descent. Optimization for [PPP](#) can also involve optimizing an existing path as seen in [\[28\]](#). Authors of [\[28\]](#) optimize typical paths resulting from sampling based algorithms which are a combination of straight lines into less noisy paths. Randomness of sampling results in detours of paths; such unnecessary path deviations are aimed to be avoided based on standard numerical tools linearly constrained quadratic program (LCQP). The authors do so by proposing a method to shorten the path length after every path planning step. Note that they do not address path planning and take the result of a probabilistic motion planner as the input to their path optimization method. The algorithm claims to require only collision checking, and not any time-consuming obstacle distance computation nor geometry simplification since they constrain only part of the configuration variables that may cause a collision, and not the entire configuration vector. Thus, parasite motions that are not useful for the problem resolution are reduced [\[28\]](#). The work shown here does not plan an optimized path from start, but optimizes a given path for optimal length.

Variants of [OAs](#) that optimize path lengths indicate that [OAs](#) can produce paths that are close to optimality, but not the most optimal solutions. Since [OAs](#) depend on sampling, the solutions are optimal in terms of sampling frequency and effectiveness. So, apart from navigating the local minima problem, one must also be conscious of the true optimality of the path produced using sampling and optimization based methods.

On the note of avoiding local minima, one of the earliest solutions proposed was the ideal [APF](#), containing a single global minimum, called the Navigation Function ([NF](#)), as proposed by Koditschek and Rimon [\[85\]](#). The pioneering work focuses on a non-Euclidean representation of the [PPS](#), and thus will be focused on in greater detail in [Section 2.2](#). The analytical approach to finding a suitable navigation function is to use or derive special functions that can remove or lessen the effects that a local minima has on the potential field. Koditschek and Rimon of [\[85\]](#) define a proper navigation function as one that satisfies the following criteria:

1. Continuous and differentiable (smooth) on the path to the goal in the connected set, uniformly maximal on the boundaries.
2. Has a unique and global minimum at the goal and must be Morse (defines directions where the function increases in value).
3. The resulting navigation function (in a generalized sphere world), eliminates the local minimum problem but only with proper tuning of a sharpening parameter defined as part of the function [\[130\]](#).

The navigation function therefore is posed to look ideal in that it is of maximum value at the boundary of the obstacle, directed away from the boundary and towards the workspace with a minimum at the goal. However, constructing such a function is challenging to do analytically.

As an alternative, authors of [\[130\]](#) attempt to construct the navigation function numerically. Using wavefront expansion, the metric based on which the [NF](#) is constructed is the control effort. Traditional metrics involved distance-based contour computation, as opposed

to a metric based on the control effort of the system to form the contour levels in the potential field. Since the metric is control effort-based, they are able to construct the **NF** by making use of the solution to the minimum control effort problem given a fixed initial and final state of a linear system. Then, the second method will be to take an inverse dynamics approach from a reachable state (as considered for a nonlinear rover model) [130]. In this case, **NF** is combined with **MPC** as have been done in [87, 88] to produce trajectories and then track them. Yet another slightly different approach towards **APF**, is the use of R-functions, as seen in [161]. Similar in properties to an **APF**, an R-function is a real-valued function whose properties are fully determined by corresponding attributes of their parameters, which is usually applied to express a geometrical object.

In [161], a signed distance function based on R-functions is created to represent two pieces of information. First, it indicates whether two objects intervene and second, it provides information about their level of intervention or separation. As the signed function is continuous and differentiable, the gradient information of the objective function guides a moving part to avoid its obstacles and to approach its target position rapidly. The function enables an implicit description of the obstacle and its boundaries using the convex hull approach.

Perhaps the most detailed approach to numerical **NFs** can be seen in [10]. Authors Barraquand and Latombe of [10] approach it systematically with three main steps: first, construct the discretized **PPS**, second compute numerical navigation functions in the discretized space, and finally, propose two path planners, with one being gradient-based and the other being Brownian motion-based. The first step creates a multi-resolution bit-map/grid representation, i.e., there exist more than one layer of grid-based representation for the same space. Each layer assumes a pre-defined resolution and thus ranges from coarse to fine when representing the environment, from which a skeleton of the **PPS** is extracted.

The second step involves computing numerical potential functions that need to satisfy the following two conditions of a single minimum at the goal and the ability to use a gradient based planner. The former condition of the numerical potential/navigation function requiring a single minimum at the goal, makes this a major heuristic step towards minimizing spurious local minima. The latter requirement of being able to use a gradient based planner implies that the algorithm travels in the direction of lowest energy or cost. The **NF** is computed for every point in the **PPS** skeleton, starting at $\mathbf{x}^{\text{goal}}$ to whom the **NF** assigns the lowest cost of 0. Other points located at the same distance away from $\mathbf{x}^{\text{goal}}$ lie on equi-potential contours, and increasing distance from the goal increases the potential. The process is repeated until the connected subset of $\mathcal{W}^{\text{free}}$ containing $\mathbf{x}^{\text{goal}}$ is exploited, as can be seen in Figure 2.15. The complexity of the numerical navigation function is linear in the number of points in the bitmap and constant in the number and shape of the obstacles. It can therefore compute a path from any $\mathbf{x}^{\text{start}}$ in the connected component of $\mathcal{W}^{\text{free}}$ to the $\mathbf{x}^{\text{goal}}$ following the negated gradient, which is guaranteed to be the shortest path at the resolution of the bitmap. Notice that the computation occurs only in the connected subset of $\mathcal{W}^{\text{free}}$ that contains the $\mathbf{x}^{\text{goal}}$. Hence, if $\mathbf{x}^{\text{start}}$ is not in the same connected component as $\mathbf{x}^{\text{goal}}$, the value of the potential is not computed at this point, and one can immediately infer that there is no collision-free path, without even relying on the path planner. Additional numerical metrics can be included to provide contours that don't graze the obstacle boundaries as such. The authors also create a multi-resolution grid equivalent of $\mathcal{C}^{\text{free}}$, since while obstacles are included, singularities are

not. That is, several configurations in $\mathcal{C}$ may permit the same desired location in $\mathcal{W}^{\text{free}}$. In order to minimize the number of solutions, the authors pre-select a desired region of control in $\mathcal{C}$. One must keep this in mind, since path planners must ideally be able to avoid all constraints in $\mathcal{P}^{\text{constraint}}$, including singularities, failing which $\mathcal{C}^{\text{free}}$ may truly not be free of constraints.

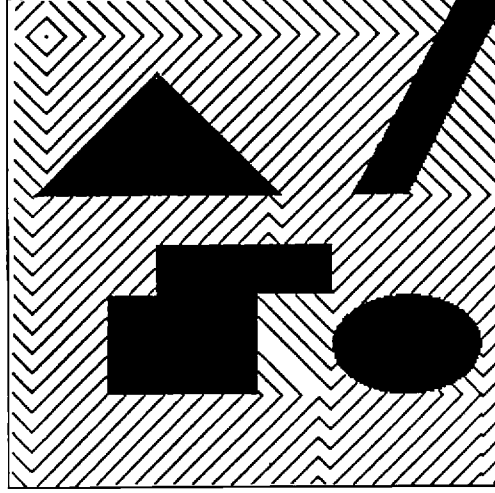


Figure 2.15: Equi-potential contours computed by the numerical Navigation Function (NF) as seen in [10]; note the contour value of 0 at the goal on the top-left of the figure.

The third and final step proposes two path planners: the **Best First Planner (BFP)** and the **Randomized Path Planner (RPP)**. The two versions differ mainly in the way the local minima of the configuration space potential function are escaped. The **BFP** iteratively considers each level of resolution in the **PPS** bit-map/grid pyramid, from the coarsest to the finest. It terminates with success as soon a path is generated. It terminates with failure if, after having considered the finest bitmap, it still has not generated a path. The **BFP** works for robots with fewer than 5 **Degrees Of Freedom (DOF)**. It is simpler and deterministically resolution complete. With the **RPP** on the other hand, rather than filling up a local minimum, it applies a Monte Carlo procedure that consists of generating Brownian motions until the minimum is escaped. The resulting planning algorithm is probabilistically (rather than deterministically) resolution complete [10]. This work is essential in introducing the numerical aspects of potential/navigation functions and the essence of gradient-based optimization methods. The numerical formulation accommodates several features that would otherwise be impossible. Those features include grid map representation, multi-layered map, numerical assignment of potential and choosing a subset of control inputs to avoid singularities. The discretized set of points allows for such concessions, but lack in the connectivity of the space that can only be imposed mathematically, as shall be seen in Section 2.2.

2.2 Manifolds

This section focuses on approaches that plan paths in spaces with an underlying structure. Such spaces contain points whose relationship to each other is defined by imposed characteristics. They include satisfying a system of equations, obeying laws of smoothness and adhering to a certain metric. One of the most obvious, intuitive characteristics of such spaces is that they are only locally Euclidean, and not globally so (they can also be both). Such spaces are called manifolds and will be formally defined in Chapter 3. The advantage of formulating paths on manifolds is in utilizing the natural structure of the robotic workspace. This is especially of great use when the resulting manifold of the configuration or work space is free of constraints. Then, the path planning can proceed on the natural structure of manifold, without having to repeatedly evaluate points in the space for constraints. Path planning on globally Euclidean spaces (such as planes) and globally non-Euclidean surfaces (such as manifolds) still aim to find the shortest path in their respective spaces. The shortest path in the former is a straight line as seen in Figure 2.16, whereas its equivalent in the latter is called a geodesic seen in Figure 2.17. Informally, geodesics are the straight lines of manifolds, but appear non-straight, since they adhere to the underlying curvature of the surface. Path planning between a source and target configuration on curved surfaces therefore involves determining the geodesic, parts of it, or discrete approximations of either.

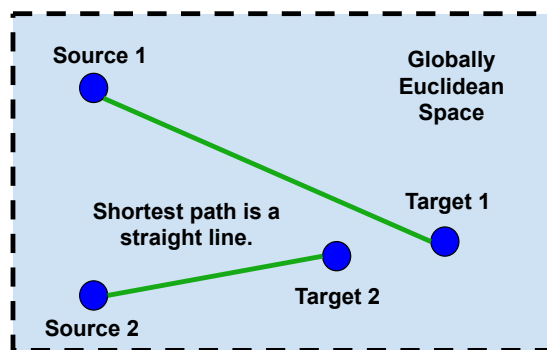


Figure 2.16: A globally Euclidean space such as a plane where the shortest path between any two points is always a straight line.

Constructing geodesics requires knowledge of the manifold's representation. The difficulty lies in representing or characterizing the manifold. While some variants or shapes of manifolds exist that are represented using global parametrizations, complex manifolds are seldom easily characterized by a global parametrization. The approaches to path planning therefore bifurcate into two main groups. The first method focuses on adapting methods used in the point-set space (such as sampling, potential fields and graph based approaches) to familiar manifolds. The second group involves incrementally building charts or local representations of the unknown manifold, to eventually build a global representation. Some planners simultaneously attempt to search the built surface for a path, while others find paths after the manifold construction process. Sections 2.2.1 and 2.2.2 discuss the first and second methods, respectively.

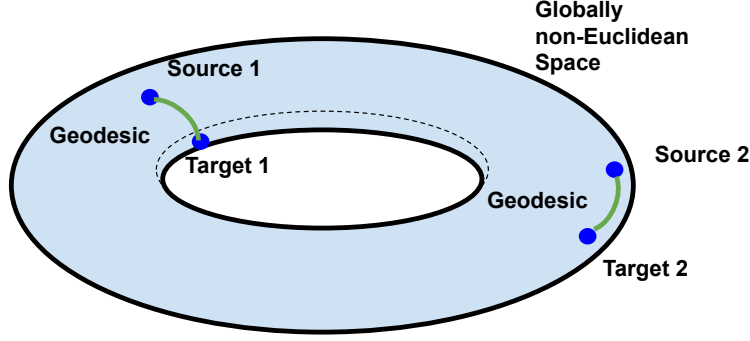


Figure 2.17: A globally Non-Euclidean space such as a Genus-1 Torus (donut) where the shortest path between any two points is a geodesic.

2.2.1 Parametrizing Manifolds in Path-Planning Space

Point-sets in the workspace are often a union of points representing the constraints and those that represent the constraint free space, as seen in Figure 2.18. It is from these point-sets that the surface denoting the constraint space $\mathcal{P}^{\text{constraint}}$ or the constraint-free space $\mathcal{P}^{\text{free}}$ is extracted mathematically. That is, the point-set is partitioned into constraint and constraint-free regions by employing systems of equations to construct surfaces from those regions. This can be especially challenging when both sets may not be compatible mathematically. Due to the challenging nature of the problem, several early efforts use approaches analogous to those used in Euclidean spaces. The most fundamental approach that attempts this can be seen as early as 1983, where authors of [25] explored mapping the constraint space into the admissible configuration space. Although not explicitly defined as manifolds or geodesics, this paper is probably the earliest work on projection of the constraint space into the admissible configuration space. The authors of [25] then find upper and lower bounds for the projected constraint space, and find paths in so called freeways. These freeways are areas that are free of constraints, and are represented by primary motions, such as a translation along a fixed axis.

APFs are powerful concepts as attractors toward the goal configurations, however cannot be directly adapted to curved surfaces, because of the fundamental difference in the nature of curved and planar surfaces. As an adaptation of APFs, Koditschek and Rimon of [85] pioneered path planning by formulating an equivalent global attractor function on manifolds. Their work was based on analytic manifolds with a boundary and the attractor functions were referred to as NFs. NFs are an extension of artificial potential fields in point sets to manifolds, specifically in a sphere world. A sphere world is a compact connected subset whose boundary is formed by the disjoint union of a finite number of spheres. It follows that there is one large sphere which bounds the workspace, and smaller spheres which bound the obstacles. The free space $\mathcal{P}^{\text{free}}$ is obtained after removing all the smaller sphere-bounded obstacles from the bigger encompassing sphere-bounded space. Koditschek and Rimon show that using non-degenerate vector fields (isomorphic or structure preserving vector fields) which are transverse on the boundary of the free workspace does not provide a globally attracting equilibrium state. This is because a smooth vector field on any sphere world which has a unique attractor, must have at least as many saddles as there are obstacles.

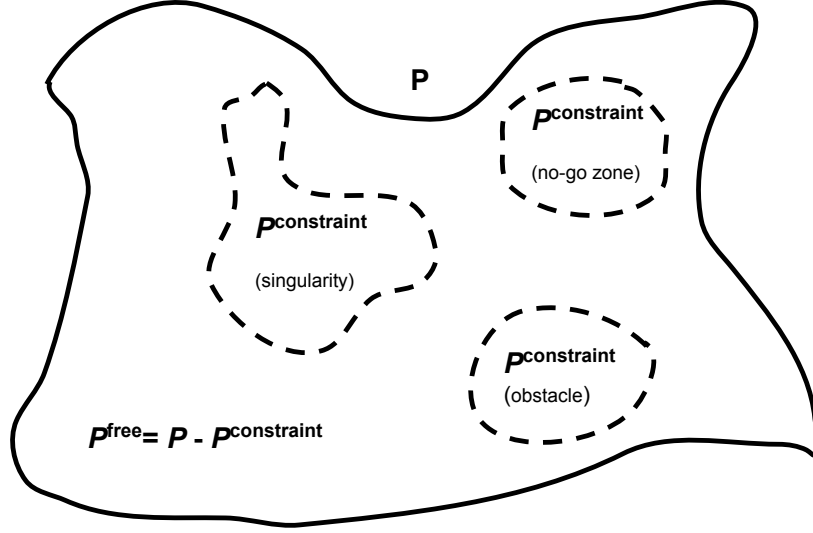


Figure 2.18: Breakdown of the general Path Planning Space (PPS) $\mathcal{P}$ into $\mathcal{P}^{\text{constraint}}$ and $\mathcal{P}^{\text{free}}$

Thus, a globally attracting equilibrium state is topologically impossible [85]. The presence of such saddles or local minima is typical of what is experienced in APFs in point sets. As a solution instead, Koditschek and Rimon of [85] impose further restrictions on the obstacle free workspace. They do so by defining it as a compact connected analytic manifold $\mathcal{P}^{\text{free}}$, that admits a NF. The NF is Morse and analytic on $\mathcal{P}^{\text{free}}$, and is polar, i.e., has a unique minimum in $\mathcal{P}^{\text{free}}$. With these conditions in place, if the gradient is transverse and directed away from the interior of $\mathcal{P}^{\text{free}}$ on the boundary, then the negative gradient is a positive invariant set. The method by itself focuses on formulating the navigation function, which is a composition of three functions, where the first function is polar, almost everywhere Morse and analytic; the second function is a diffeomorphism to limit the output range, resulting in a polar, admissible, and analytic function which is non-degenerate everywhere on $\mathcal{P}^{\text{free}}$ except the target point in $\mathcal{P}^{\text{free}}$; and the third function addresses the target point. The authors point out that analytic NFs are harder to construct, but once defined, yield a provably correct control algorithm. Unquestionably, real world scenes do not often admit even a smooth, much less an analytic representation, and extending the approach beyond the class of a ball of obstacles may require relaxation of some constraints [85]. Tanner and Kumar of [160] also use NFs in the context of multi-agent robotics to avoid local minima.

Control theorists were familiar with the concepts of systems of equations defining surfaces, i.e., vector fields representing curvature and thus directions of motion leading to integral curves. Integral curves could then be viewed as feasible paths that the system could take. In the field of robotic path planning however, that was not the case. Authors of [42] were some of the early researchers who introduced the same concepts for path planning. They use kinematic constraints to come up with the structure of the manifold and the subsequent variety of paths resulting from such a structure. Contrary to Koditschek and Rimon of [85], where point-set based methods were adapted to manifolds, authors of [42] focus on formulating the structure of the surface and move away from point-set based methods. This

step is the precursor to constructing geodesics. By showing that vector fields already account for kinematic constraints, they show that the surface being operated on is free of kinematic constraints. Paths on this kinematic constraint-free surface, then simply need to be checked for intersection with the obstacle region’s boundary. Thus, feasible paths incorporate both kinematic and obstacle constraints [42]. Since there may exist multiple feasible paths, a cost or metric is chosen for optimality. Authors of [42] choose the Riemannian metric, where-from one can compute the geodesics of that metric. As stated previously, geodesics are paths corresponding to “straight lines” in the metric space. However, unlike straight lines, geodesics are not always feasible paths. In order to be so, they need to be in the total flow of the vector fields of control. In general, and especially for non-holonomic problems, no such situation exists. In that case, the geodesics are approximated by pieces of flow lines of the field [42]. The discretization procedure of geodesics computation was accomplished by using the cost wave propagation algorithm [42].

As seen above, using constraints to define the surface is indeed possible, but is made challenging by two factors. First, availability of information about constraints influences how the surface can be characterized. Second, there sometimes exists a choice in how the PPS can be characterized. Both factors are now explained in detail.

First, information about constraints must be available, since constraints help define the constraint-free space. Typically, constraints could be any combination of the following: mechanical and/or kinematic constraints, singularities, and obstacles/no-go zones, as seen in Figure 2.18. Note that all constraints of a PPP may not co-exist in the same space, i.e., they need not lie entirely in $\mathcal{C}$ or $\mathcal{W}$, as this is application dependent. Some applications may have constraints purely in $\mathcal{C}$ or alternatively, purely in $\mathcal{W}$. Other applications may have some constraints that can feature in $\mathcal{C}$, while other constraints can lie in $\mathcal{W}$. This constitutes an ambiguous and/or incomplete definition of the constraint space. It is so because of the following. Constructing a constraint-free space based on the constraints lying exclusively in $\mathcal{C}$ would result in a constraint-free space, say $\mathcal{C}^{\text{free}}_{\mathcal{C}\text{constraints}}$. Equivalently, the constraint-free space resulting from constraints lying exclusively in $\mathcal{W}$ is denoted by $\mathcal{W}^{\text{free}}_{\mathcal{W}\text{constraints}}$. The problem with this definition is that while both $\mathcal{C}^{\text{free}}_{\mathcal{C}\text{constraints}}$ and $\mathcal{W}^{\text{free}}_{\mathcal{W}\text{constraints}}$ are constraint-free in their own right, they are not equivalent to each other. $\mathcal{W}^{\text{free}}_{\mathcal{W}\text{constraints}}$ does not represent the same constraints as represented by those in $\mathcal{C}^{\text{free}}_{\mathcal{C}\text{constraints}}$. In other words, there exists no equivalence mapping from $\mathcal{C}^{\text{free}}_{\mathcal{C}\text{constraints}}$ to $\mathcal{W}^{\text{free}}_{\mathcal{W}\text{constraints}}$, because they do not represent mapped images of the same constraints in the native or original space. Therefore, knowledge of $\mathcal{C}^{\text{constraint}}$ does not automatically imply that its equivalent is $\mathcal{W}^{\text{constraint}}$, unless such a mapping has been deliberately accounted for; see Figures 2.19 to 2.21. Thus, until all constraints are accounted for in at least one space, i.e., either $\mathcal{C}$ or $\mathcal{W}$, one cannot conclude defining the constraint space. Such a difference in characterizing the constraints in different spaces can be seen with the 2-Link Planar Manipulator (2LPM) and a mobile robot. For a 2LPM, constraints such as singularities, operational joint limits and no-go zones in $\mathcal{C}$ are collectively defined in $\mathcal{C}$. However, constraints in the form of obstacles and no-go zones in the EE’s workspace are defined in $\mathcal{W}$. For a constraint-free surface to be defined in either $\mathcal{C}$ or alternatively in $\mathcal{W}$, all constraints must be defined in either $\mathcal{C}$ or alternatively in $\mathcal{W}$, respectively. Consequently, the constraint-free space can be defined in $\mathcal{C}$ or alternatively in $\mathcal{W}$, respectively. In contrast, the PPS of mobile robots are

usually defined in $\mathcal{W}$, where the robot traverses the environment. In this case, constraints are usually entirely defined in $\mathcal{W}$, as is the constraint-free space.

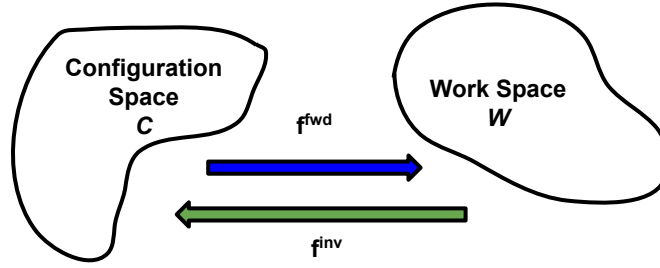


Figure 2.19: Forward and inverse mapping between $\mathcal{C}$ and $\mathcal{W}$ of robots such as manipulators

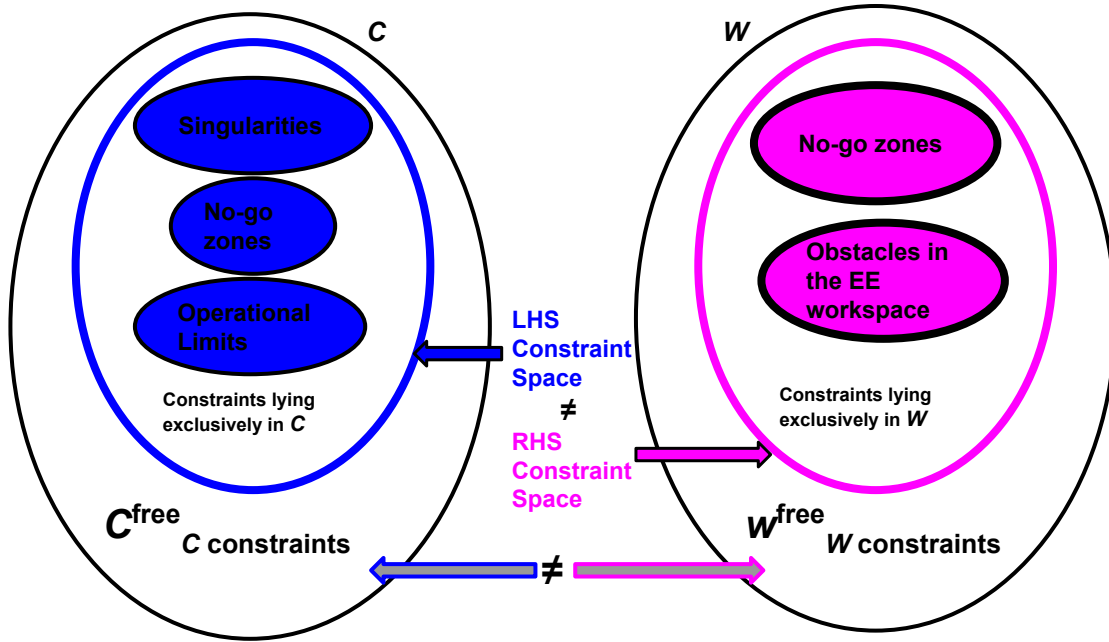


Figure 2.20: Equivalence of $\mathcal{W}^{\text{free}}$ cannot be directly and simply assumed from $\mathcal{C}^{\text{free}}$ and vice-versa. Care must be taken to choose the PPS as a space that can be characterized mathematically.

The second factor concerns the different ways a constraint-free surface can be defined. It was seen that all constraints (original and mapped) can be defined in $\mathcal{C}$ or $\mathcal{W}$, i.e. as $\mathcal{C}^{\text{constraint}}$ or as $\mathcal{W}^{\text{constraint}}$. Since $\mathcal{C}$ and $\mathcal{W}$ are fundamentally different sets of spaces by definition, $\mathcal{C}^{\text{constraint}}$ and $\mathcal{W}^{\text{constraint}}$ will also appear as different subsets in their respective parent spaces, $\mathcal{C}$ and $\mathcal{W}$. Then naturally, the removal of the constraint spaces will also result in differently characterizable constraint-free spaces, $\mathcal{C}^{\text{free}}$ and $\mathcal{W}^{\text{free}}$. This difference enables defining the surfaces that can represent $\mathcal{C}^{\text{free}}$ and/or $\mathcal{W}^{\text{free}}$ differently. This is especially true if the all the constraints pertaining to the PPP are native to exclusively one space; for example, assume all constraints lie in $\mathcal{C}$. Thus, the constraint space in the native space

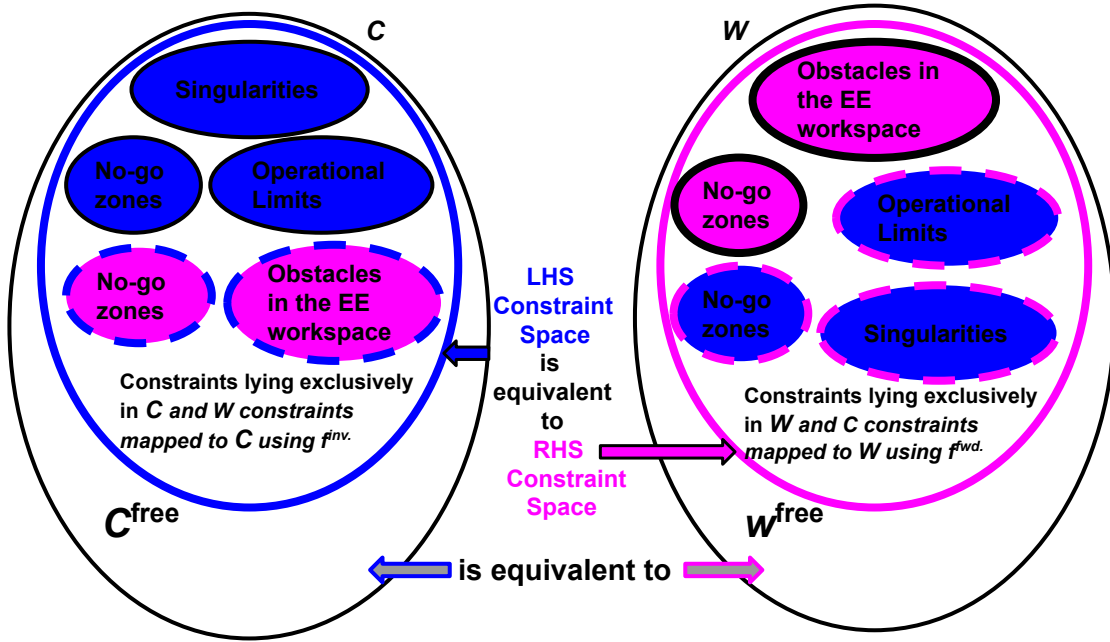


Figure 2.21: Equivalence of the associated spaces of the PPS using the f^{fwd} and f^{inv} maps.

$\mathcal{C}$ would not consist of any mapped constraints from another space ($\mathcal{W}$) and would be the disjoint union of all the constraint sets in $\mathcal{C}$. Then, the constraint-free space in the native space $\mathcal{C}$ is easily defined by the absence or removal of the constraint-space. However, if this constraint-free space were mapped to the equivalent space $\mathcal{W}$, the mapped constraint-free space will now be defined as a stand alone surface as a result of the mapping applied to the constraint-free space in native $\mathcal{C}$.

One can see an example of the fundamental difference between the two possible spaces $\mathcal{C}$ and $\mathcal{W}$, where constraints can lie, for a simple 2LPM in Figures 2.22 and 2.23. Even without constraints, the two spaces appear different; one is Euclidean, while the other is not easily recognizable as a subset of the Euclidean space. Thus, one cannot immediately assume that the path-planning space devoid of constraints automatically results in $\mathcal{P}^{\text{free}}$, because it depends on the ability to characterize the constraints, the PPS and whether removal of constraints results in a characterizable space. Care must be taken to map the surfaces/spaces into the equivalent $\mathcal{W}$ or $\mathcal{C}$, before path planning can commence.

Some authors therefore, shifted their focus towards explicitly defining manifolds that describe the singular sets and singularity spaces as well. Trinkle and Milgram of [162] tackle path planning on the singularity-free workspace for a planar closed chain spherical manipulator. They accomplish the complex task of determining $\mathcal{W}$, $\mathcal{C}$, and the singular sets in either space, by approximating the Closed Chain Manipulator (CCM) as an equivalent Open Chain Manipulator (OCM). The OCM's fixed joint EE is equivalent to the CCM's first and last fixed joints. The following steps are taken to determine $\mathcal{W}$, $\mathcal{C}$, and the singular sets. First, it can be seen that the $\mathcal{C}$ of the OCM is a product of Special Orthogonal Groups (SOs) and the EE of this open chain is free to map anywhere inside the annular workspace $\mathcal{W}$. Second, they determine the complete $\mathcal{C}$ corresponding to the forward kinematics' image space

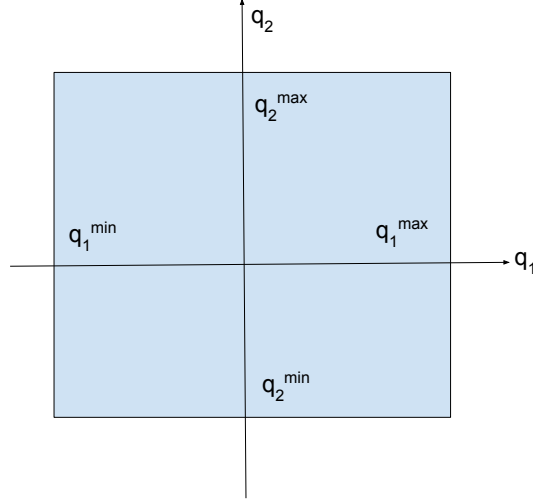


Figure 2.22: The configuration space $\mathcal{C}$ of a 2-Link Planar Manipulator (2LPM)

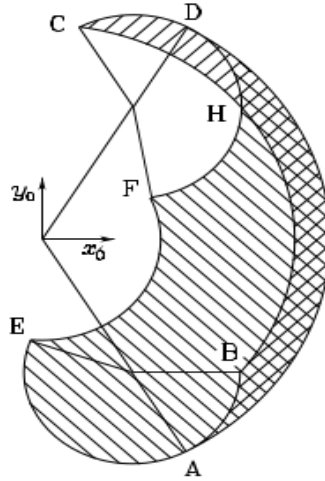


Figure 2.23: A 2D work space example of a 2-Link Planar Manipulator (2LPM) [154]

$\text{Im}\{\mathbf{f}^{\text{fwd}}\} = V$. This image V takes the form of either a closed ball or an annulus, and lies in $\mathcal{W}$. Subsequently, the inverse location of the removed joint e is obtained using $\mathbf{f}^{\text{inv}}(e)$, as a special case. In this case, the singular points of $\mathbf{f}^{\text{fwd}}$ are disjoint spheres in $\mathcal{C}$. This completes defining the singular sets of the CCM and the singularity-free space of the CCM in $\mathcal{C}$, all of which are some variants of SOs, thus resulting in manifolds. Next, the image of the singular set decomposes $V \in \mathcal{W}$ into connected regions, which are open annuli centered at the base of the open chain. Since the singular set in $\mathcal{C}$ consists of disjoint spheres (i.e., compact), the map $\mathbf{f}^{\text{fwd}}$ restricted to the inverse image of each open region in the annuli in $\mathcal{W}$, is a closed compact manifold (sphere) in $\mathcal{C}$. Consequently, the inverse images for any two points in one of these regions are at least locally diffeomorphic. Now, the problem reduces to two basic steps: first, find the inverse image in one of these regions; and, second, understand how the image changes when passed through the image of a singular point. Standard methods of differential topology and algebraic geometry aid in determining how the global structure of

the space $\mathbf{f}^{\text{inv}}$ changes as one passes through the image of the singular points. The structure and the number of connected components in the inverse image provides the solution to the path existence problem. That is, a path from $\mathbf{q}^{\text{start}}$ to $\mathbf{q}^{\text{goal}}$ exists only if $\mathbf{q}^{\text{start}}$ and $\mathbf{q}^{\text{goal}}$ are in the same path component in $\mathbf{f}^{\text{inv}}$. Further information about the global structure of $\mathbf{f}^{\text{inv}}$ gives insight on how to construct efficient paths consisting of long geodesic segments. They use accordion moves to generate such paths.

Trinkle and Milgram of [162] utilized properties of homeomorphism and diffeomorphism (both of which rely on continuity) to effectively build the path planning manifold. So clearly, continuity is an important aspect of the complexity of path planning, and more so on manifolds. In [46], Farber illustrates precisely this, i.e., the notion of topological complexity of the motion planning problem. The topological complexity is a number that measures discontinuity of the motion planning process in the configuration space. Farber establishes an upper and lower bound for the topological complexity as a function of the dimension of the configuration space. Examples such as spheres, two-dimensional surfaces and products of spheres are demonstrated, which arguably form the most commonly seen manifolds in a systematic $\mathcal{W}$ or $\mathcal{C}$ derivation. A specific extension to robotics is seen in the example where the topological complexity of the motion planning problem for a robot arm in a configuration space without obstacles is computed. The idea presented in [46] is of importance, because it is able to present a mathematical measure of what surfaces are viable for motion planning. Such a measure is useful if and when there exists a choice between the kind of surfaces on which paths must be planned on.

As the topological nature of workspaces and configuration spaces were becoming more apparent, efforts to combine the benefits of topology while not being mathematically intensive began to appear. Take for example [107], where the authors recognized the need to account for environmental topology such as terrain curvature. However, they leverage tools such as **Generalized Voronoi Diagrams (GVDs)**, **Visibility Graphs (VGs)** and **APFs** that are typical of discrete data sets representing the environment. The study in [107] is an important example that shows the gradual transition from planning paths in point set representative environments, to requiring the imposition of structure to define the environment, albeit using small steps. Masehian and Amin-Naseri of [107] demonstrate the effectiveness of combining **GVDs**, **VGs**, **APFs** and maximal inscribed discs to produce an obstacle-free configuration space. The assumptions include a point robot in a two-dimensional space, where the environment with static obstacles is known apriori. Olmstead and Yang of [121] take an alternative view to using geodesics for robotic path planning. Unlike the approach that uses manifolds to model the environment as seen in [42, 46, 85, 107, 162], the authors of [121] model the obstacle as a manifold instead. Typically, the shortest path from the source to the target configurations in a planar environment is a straight line connecting the two configurations. However, with the presence of an obstacle, the straight line path may not be feasible anymore. In an effort to minimize the deviation of the original path, the segment of the path around the obstacle must be such that it is, at least locally, the shortest path. In other words, it must be a geodesic. To compute a geodesic around the obstacle, the obstacle must be a surface that is complete, convex and edge-free. But since obstacles need not comply with such requirements on their own, authors of [121] model an ellipsoid around the obstacle (interestingly similar to elliptical potential functions used to model repulsive **APFs** around

obstacles by [166]). An illustration of such a path around a spherical manifold enveloping an obstacle is seen in Figure 2.24. Subsequently, geodesics on the surface of the ellipsoid model of the obstacle can be computed mathematically. The authors compute the discrete geodesic around the obstacle and append it to the remaining points that make up the path. Mathematical computation is involved since geodesics require some constants (Christoffel Symbols) to be computed.

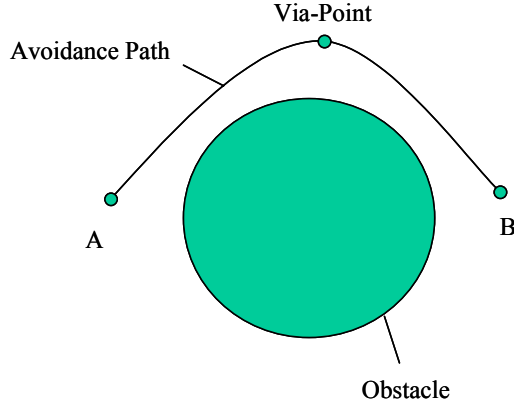


Figure 2.24: Modeling the obstacle as a spherical manifold and computing a path around it [121].

One can now begin to see the relationship amongst global and explicit representation/characterization/parametrization of the PPS, the shortest paths and the properties that the PPS must satisfy. As an extension to the work done in [162], Shvalb et al. [153] examine the global connectivity of the planar star-shaped manipulators, for efficient path planning. The authors determine the topological complexity of the respective $\mathcal{C}$ similar to [162]. Through the analysis of the critical set of $\mathcal{C}$, the authors derive necessary and sufficient conditions for the global connectivity of $\mathcal{C}$, and those for path existence. Based on these results, a complete polynomial algorithm is devised for motion planning [153].

Connectivity is a property of manifolds that is inherent to its definition. It is so, since if globally parameterized, every point on the manifold is related to each other by virtue of the global parametrization. Alternatively if locally parameterized, every neighbourhood on the manifold must be able to transition to its neighbour using a local map. This connectivity allows the manifold to be well defined. Thus, for the purposes of path planning, it can be queried for a path multiple times with different source and target configurations. Multiple motion planning queries in a static environment have enjoyed the benefits of sampling-based techniques, such as PRMs. Since PRMs have been successful in solving problems in configuration spaces of dimensions approaching 100, many researchers have worked to make the method more efficient. For example, modifications have been made to solve more challenging types of problems, such as those with closed kinematic loops, non-holonomic constraints, dynamics, and intermittent contact motion plans, which are difficult to solve using traditional PRMs [153]. In such systems that model real-life applications however, $\mathcal{C}$ is often most naturally viewed as a lower-dimensional space embedded in an ambient space (typically the joint space), where embeddings are special maps between manifolds.

The embedding results from equality constraints corresponding to kinematic loop closure or contact constraints. Thus it is challenging to obtain an explicit description of $\mathcal{C}$ with minimal number of parameters and a suitable metric to guide sample generation. These problems make it difficult to construct a roadmap with the requisite properties, and hence difficult to solve motion planning problems for systems with kinematic loops using PRMs (or any SA). Unless SAs like PRMs, RRTs and APFs account for the global structure of $\mathcal{C}$, they may fail to sample globally relevant regions [153].

Jaillet and Simeon capitalize exactly on that issue in [68] by developing a new variant of PRM that inherently encodes the connectedness of the configuration space. The encoding is inside small yet representative graphs which capture the different varieties of free paths well. The graphs are produced using the proposed Path Deformation Roadmaps (PDRs). PDRs depend on whether paths can be continuously deformed into another existing path, and use a deformation criteria that is simpler than permitted for homotopic paths. Informally defined, functions are homotopic if they can be continuously deformed from one another. It implies that the functions are equivalent topologically under the homotopy class. This is of use in constructing paths for robotic applications, since, should one path not be viable, a deformation may be applied to result in an alternative, equivalent path. A simple illustration is seen in Figure 2.25. The relaxed deformation in [68] allows the use of the Visibility-PRM technique to construct road-maps that encode better suited information than representative paths of homotopy classes. As an added advantage, PDRs contain additional useful cycles between paths in the same homotopy class that otherwise would be hardly deformed into each other [68]. This approach is important in recognizing multiple aspects of the path planning problem such as global and connected nature of the space, characterizing paths into classes and planning for alternative paths within every class. Alternatives in the form of deformable paths are of utmost importance for contingent environment changes, such as the appearance of new static and dynamic obstacles. It is also important to know that while the PDR is still a variant of the PRM, it strongly utilizes the topological basis of the space, rather than assuming the relationship, as often seen in pure sampling based methods.

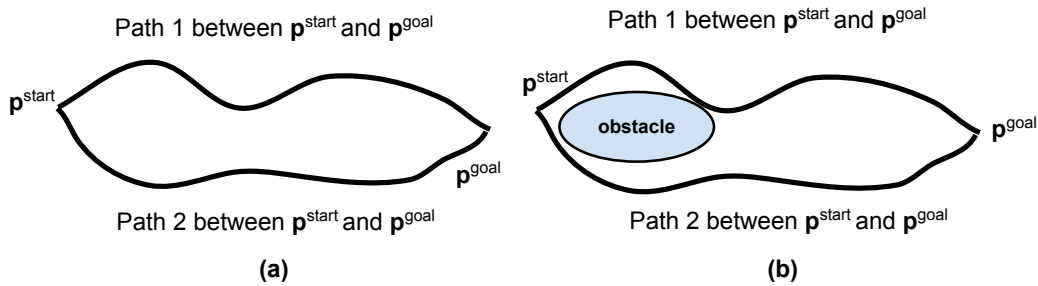


Figure 2.25: (a) Homotopic or deformable paths (b) Paths are not homotopic due to the visibility of an obstacle [68].

Capturing global connectedness is important, as seen in [68]. However, it is a complex task. In an attempt to simplify the solution, Roy of [147], formulates a subset of the configuration manifold by breaking the $\mathcal{C}$ of a high DOF robot into smaller parts. Every joint's $\mathcal{C}$ is treated as a plane, and so are obstacles. The intersection of the joint space planes and obstacles result in sets (points, lines, circles, etc.), for which a $\mathcal{C}$ map is computed. The final

set of the collision areas are the union of all those sets of computed $\mathcal{C}$ maps. A visibility map-based heuristic algorithm is used to generate near-optimal safe path in the formulated subset of the manifold. Similar to potential or energy based optimization techniques, an equivalent concept is seen in [148], where path planning is based on topology and thermal conduction. That is, by identifying the source and target configurations of the robot as the heat source and sink of a conducting plate, respectively, the path planning problem can be formulated as a topology optimization problem that minimizes thermal compliance. Obstacles can be modeled as regions of zero-conductivity or varying non-uniform levels depending on the application. Analogous to an APFs, this technique uses artificial mass. The formulation of the problem as such is intended to make obstacle regions less conducive to move towards and may be viewed as an indirect way of extracting the manifold (without any formal characterization) of the obstacle-free space. Yet, another attempt to capture the constraint-free space is seen in [109], where the concept of reachable volumes is derived by computing the union of points that result from the reachable confines of the joints' extremities.

Using an entirely different perspective, authors of [54] also attempt to create a representation of the path planning manifold by learning the underlying structure of the manifold from existing trajectories. The authors argue that their approach is model-free. That is, an explicit and complete analytical specification of the task which is difficult to obtain for some realistic robots, need not be assumed. Instead, they focus on maneuver sets, which are sets containing the solutions/trajectories to optimal path planning problems by accounting for constraints and variations in trajectories/skills. As noted before in this section, these sets typically admit a description in the form of manifolds embedded in the configuration space of the system being controlled. The focus of [54] is on learning such solution sets from data in the form of a manifold, generalizing from a sample set of trajectories that share the underlying qualitative structure, and utilizing this manifold as the basis for subsequent motion synthesis, by a process of constrained geodesic trajectory-generation [54]. The skill manifold is learned in an offline phase directly from demonstrated data. Such data can be either the result of a computationally expensive optimization procedure (which is feasible offline) or even examples of trajectories as demonstrated by a human being. The learnt skill manifold can then be used by the robot online and autonomously, accommodating novel constraints and goals. The disadvantages of combining learning techniques with complex, higher-dimensional problems are as follows. First, providing an appropriate number of and appropriately admissible solution sets/trajectories that are amply representative of the true manifold is seldom easy. This may result in the degradation of the algorithm's ability to correctly learn the characteristics that define the manifold, and additionally risk never being able to converge to a valid solution. Second, the learning algorithm runs into risks of learning the underlying structure incompletely, as learning is proportional to the training set provided and the amount of training permitted. Finally, such learning based approaches are not robust to parameter changes; for example, changes in robot dimension, placement of the obstacle, definition of the environment, and even the definition of the task. With even one such change, training must commence from the start.

Similar to [107], the work in [129] also relies on grid-based GVD to find paths. Since incrementally constructed GVDs in the scope of robotics are used to extract the underlying metric skeletons of grids, the result is typically a graph structure with vertices and edges

representing the connectivity of the space. However, the resulting structure contains no topological information about the environment, thus making it difficult to combine with path planners that need more heuristic information, such as terrain change and elevation about the path planning space. Authors of [129] introduce a new algorithm called [Dynamic Topology Detector \(DTD\)](#) that combines the [GVD](#) with discrete grid-based occupancy maps. The [DTD](#) then modifies the [GVD](#) stages to account for topological information as such. For instance, when a cell is marked as occupied instead of free, the authors introduce operations that allow for elevated or depressed scanning. That is, the occupancy grid along with the voronoi cells now store information that is specific to the elevation. Along with retaining connectivity amongst nodes, the [DTD](#)'s topological information is useful as obstacle and terrain structures can now be incrementally updated. Since [DTD](#) works incrementally, it is possible to use it in dynamic environments. One must note that works, such as [107, 129], attempt to represent a subset of the path planning space. Since it is not a comprehensive solution, the risks of incompletely representing the environment and possibly misinterpreting the true topological nature based on simply what was sensed are high. Therefore, the use of mathematical (topology and differential geometry) tools may provide a better comprehensive foundation towards a topologically representative solution of a path in its environment, as seen in much of the work in this section. This approach may be restrictive across parameters, such as the kinds of surfaces used, assumptions about apriori knowledge of the map, and computational restrictions. However, it is such restrictions that allow a foundational analysis of the problem. Of course, there are several tools that can provide valid approximation procedures that transcend such restrictions, and will be introduced during the implementation stage.

Take for instance [178], where the authors investigate path planning algorithms that are based on level set methods, which reflect the underlying mathematical structure of the space. These are for applications in which the environment is static, but where an apriori map is inaccurate and the environment is sensed in real-time. As the authors mention, the principal contribution is not a new path planning algorithm, but rather a formal analysis of path planning algorithms based on level set methods [178]. Approaches of this kind, i.e., formal and mathematically complete analysis as seen in [20, 178], even in seemingly simple environments is what will aim to provide a representation as close to the actual environment as possible. Computational costs when planning paths with level set methods are associated with the creation of the level set function. Once the level set function is computed, the optimal path is obtained by performing gradient descent down the level set function. The approach in [178] rests on the formal analysis of how the value of the level set function varies, when the changes in the environment are detected. The authors show that in many practical cases, only a small domain of the level set function needs to be re-computed when the environment changes [178]. Although the focus of [17] is optimizing multi-robot coverage of a region, the approach that the authors develop is for a non-Euclidean space. Specifically, they adapt Lloyd's algorithm (discrete-time cost-functional minimizing) algorithm originally meant for Euclidean surfaces, to a typical [PPS](#). That is, the [PPS](#) considered is a manifold punctured (absence of region) where obstacles would be located. They are able to demonstrate uniform coverage of the path planning manifold with a hole by multiple robots by reducing it to a discrete graph. The algorithm was shown to be effective in the case where multiple

robots achieved uniform coverage on a two-sphere and in an indoor environment with walls and obstacles. Such work is essential in showing that mathematically intensive formulations can be approximated for computation purposes, while not compromising on the topological representation of the environment.

Complete reliance on geodesics for 3D robotic path planning is seen in [176]. Wu et al. [176] compute paths on a non-flat surface, such as a parameterized regular surface, using geodesics. As noted before, one can see the implication of having to use a surface restricted to having a parameterized characterization. The authors assume a simple parameterized surface for simulation purposes and compute discretized geodesics to show its efficiency for path planning purposes. Since the computed geodesic from the source to the target configuration has to be discretized for computational purposes, there exists a number of discretization intervals to choose from. In order to evaluate the choice of discrete geodesics with appropriate intervals, the authors of [176] use the criteria of minimized cost of geodesics. Geodesics locally minimize the arc length [176] between points that are sufficiently close. Therefore, a geodesic connecting source and target configurations that are not sufficiently close will be comprised of geodesic segments between intermediate configurations. As the number of intermediate configurations tends to infinity, the discretized geodesic will approach the true geodesic between the start and target points. For a given source configuration therefore, the authors use potential fields to determine the next suitable robot configuration such that it complies with the geodesic equation and possesses the minimum cost in terms of the deviation from the true geodesic.

Authors of [177] also model their path planning terrain as a manifold and then compute geodesics as paths. Since geodesics imbibe the curvature of the underlying space, they represent the shortest paths on the surface. The work in [177] differs from that in [176], with respect to the presence of obstacles. While the terrain assumes no obstacles in [176], it is not the case in [177]. Authors of [177] assume an initial calculated geodesic between the desired source and target configurations, for example as seen in Figure 2.26. An encountered obstacle is characterized by a circle with a certain radius. This circle is the projection of the bounding volume (sphere) around the obstacle onto the manifold. The original geodesic is now evaluated for any intersection with the characterized obstacle. Intersections are handled as follows: At the point of intersection, the algorithm computes the tangent plane (restricted to a certain local range to preserve the structure of the manifold) and picks a direction with magnitude on the tangent plane as the next movement. This chosen line, along with its end points are projected on the manifold to be connected as an intermediate segment to the original geodesic. However, should this new segment still intersect with the obstacle, the algorithm can pick yet another direction of motion from the tangent plane to proceed with. One may be concerned about the computational resource/time used to correctly find the direction of motion away from the obstacle, since there exists an infinite number of directions on the tangent plane to choose from. To assuage that, the authors of [177] use the Gauss-Bonnet theorem and the geodesic triangle to show that choosing either of the basis vectors of the tangent plane at the point of intersection is sufficient. This is because, one of the two tangent vectors is guaranteed to be collision-free; additionally, it is guaranteed to conform to the curvature of the manifold. The authors show such an example on an ellipsoid representation of their path planning space.

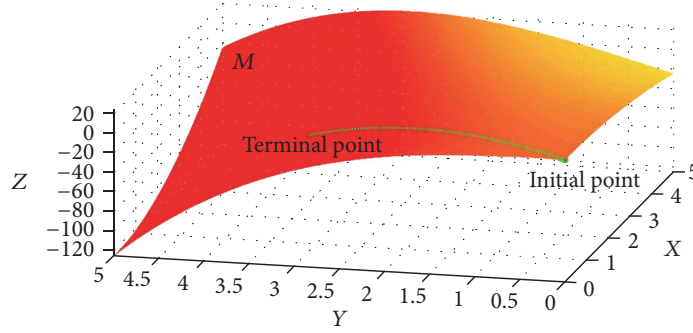


Figure 2.26: An initial calculated geodesic between the source and target locations on a hypersurface [177].

Methods such as those used in [182] revert to attempt creating an approximate, discrete topological map of the environment. They combine probabilistic path planning and APF. The authors of [182] compute probabilities of reliability to visible points and use the Floyd-Warshall algorithm to compute paths. The algorithm uses APFs to model obstacles and correspondingly update the probabilities of the region and edges (connecting vertices) that constitute the path. Regions with obstacles have their vertices updated with a low reliability score. Simulations and real-world experiments show the generation of a reliable path. However, the environment is planar and capturing the topology of the environment is not clearly elaborated. While different methods from OA and SA can be combined to determine the topological nature of the path planning space, they make heavy use of approximations of the space.

Alternatively, one could approach the problem of designing paths around obstacles by both extracting and imposing topological information on the path planning algorithm. Authors of [26] show that the topological complexity, as previously seen in [46], can be extended to smooth compact Riemannian manifolds (manifolds equipped with a smooth metric). This accounts for motion planners that aim to construct paths with the lowest possible lengths. It also shows that the topological complexity of motion planners on supposedly more restrictive Riemannian manifolds only differs by a factor of 1. This is important to note since Riemannian manifolds admit smooth paths and their metric allows definitions of angles at intersections, curvature, area and volume, etc., which are useful in path planning applications. The PPS, as considered in [26], can then be considered as a space of continuous paths or motions that the robotic system can take. It is then natural to ask if paths in this space can be collectively defined by virtue of sharing some equivalence relationship, such that when one path is deemed unfit, another equivalent path may be chosen. Bhattacharya et al. [15] answer exactly this question and take it further by designing classes of paths around obstacles using concepts of homotopy and homology. Note that such an effort to produce homotopic paths was undertaken earlier in [68] as illustrated in Figure 2.25.

Bhattacharya et al. [15] show that their proposed representation allows them to identify/distinguish trajectories in different classes and compute least-cost paths in non-trivial configuration spaces with topological constraints using graph search-based planning algorithms. The two classes of paths used are homotopy and homology, respectively. Informally

defined as homotopy earlier, homotopic trajectories can be described in the context of path planning as follows. Two trajectories τ_1 and τ_2 connecting the same start and end configurations $\mathbf{x}^{\text{start}}$ and $\mathbf{x}^{\text{goal}}$, respectively, are homotopic only if one can be continuously deformed into the other without intersecting any obstacle. As an alternative equivalence class in homology, the same pair of trajectories above (but this time τ_2 possessing an opposite orientation) forms the complete boundary of a 2-dimensional manifold embedded in $\mathcal{C}$ not containing/intersecting any of the obstacles. This definition stems from the fact that homology informally defines the empty space or holes (genus) that characterizes the manifold. Paths in the homology and homotopy classes are seen in Figure 2.27. The authors in [15] use this strong mathematical feature to stay clear of obstacles. They note that in a typical path planning environment that has been discretized into a graph, the common cost function is that of the distance assigned to a graph node/vertex. Such a function is a differential one-form (mapping from several dimensions to a single dimension/real value) but only accounts for distance and do not capture topological information. The authors therefore use properties of differential one-forms such as the H-signature (integral of the one-form) to design a homology class that is invariant for 2-dimensional $\mathcal{C}$. They use representative points to mark the interior of obstacles and thus regions around obstacles form distinct homotopy classes (boundaries of obstacles would form homology classes). Subsequently, an obstacle marker function is used, that allows trajectories to stop deforming once they reach the obstacle's boundaries. Using this imposed structure on an obstacle filled environment, they successfully generate obstacle avoiding trajectories, and are able to compute alternative paths within the classes. Successful examples include up to 20 different homotopy classes are seen in the 2D case.

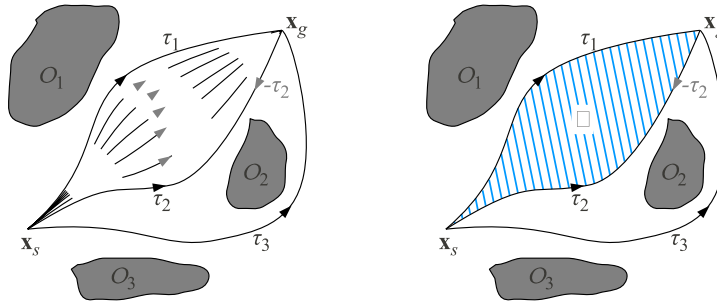


Figure 2.27: Left: Homotopic paths τ_1 and τ_2 , but not τ_3 , with either of the other paths. Right: Homologous paths τ_1 and $-\tau_2$ enclose shaded area; τ_3 is not homologous with either of the other paths. [15].

Representing a PPS using manifolds is challenging in that global and explicit parametrizations are difficult to generate. Even if methods can generate such representations for environments that approximate the robot's environment, it is so only in cases where the environment is known apriori. Authors of [77] explore the possibility of capturing topological information in an environment where the robot can sense obstacles as point clouds. The authors propose a method to define convex corridors that specify free space and provide linear constraints required for finding the optimal trajectory for polygonal robots in cluttered environments. Obstacles sensed are represented by point clouds, and by using a point-wise Minkowski sum,

the occupied regions are defined for every orientation of the robot. An initial seed path is generated using a graph/sampling based planner. Subsequently, the algorithm uses obstacle points to define hyperplanes, which in turn define polytopes that enclose the chords of the seed path. Since the polytopes that enclose the chords of the path can overlap, the inherent overlap produces a chain of overlapping polytopes, and hence convex regions. These convex regions now form an approximated representation of the obstacle free space. The initial seed path can then be optimized for velocity using quadratic programming, resulting in smooth, short trajectories, or any higher desired derivative the selection of which is driven by robot dynamics, kinematics, or power constraints. This approach lends itself to the advantage of constructing adaptable polytopes to maximize the free workspace representation. Additionally, it can be scaled to several obstacles; however it is not without its disadvantages. Without prior knowledge of the arrangement of obstacles and the seed path, choosing obstacle points for polytope creation could be challenging. A naive approach is to rank the obstacle points for each chord. This can become computationally intensive depending on the number of obstacles points [77].

Bhattacharya of [15,77] approaches the problem of path planning in non-Euclidean spaces differently in [16] and proves to be an interesting transition from manifolds as seen so far, towards gradually discretizing them. The work proposes the tasks of effectively building a non-Euclidean space and planning a path on it using the Rips complex of graph based representations. Using the Rips complex allows tremendous flexibility in defining connections between vertices and also across faces, i.e., connections are no longer simply between neighbouring vertices. The removal of such a restrictive connection and the imposition the Rips complex implies that maximal simplices are now in use for traveling along the graph representing the manifold. To build and estimate cost at the same time, The study in [16] proposes an algorithm called S^* , which is a descendant of the Dijkstra algorithm. S^* works like A^* , however, the difference is that it operates on maximal simplices and constructs local embeddings of the identified maximal simplices to create a more geometric representation of the manifold. The work in [16] is a crucial marker of how discrete mathematical tools are essential to find a suitable approximation to the complex problem of representing the PPS accurately while still adhering to their fundamentally mathematical nature. The work in [103], which is similar to that of [162], still adheres to restrictions of manifolds without discretizing them for computational ease. Instead, it uses some fundamental properties specific to planar serial manipulators to minimize chart construction, where the union of charts represents the manifold. Usually, several charts are required to accurately map the manifold completely, however the authors of [103] prove that only two coordinate charts to parameterize the configuration space are required. Assuming that the planar serial manipulator has m joints and $(m - 1)$ links, each such chart is embedded in an $(m - 3)$ dimensional torus (since every joint is an SO group, as seen in [162]). The resulting formulated $\mathcal{C}$ allows obstacles to be defined by their convex hulls and their boundaries be defined as varieties (solutions to set of polynomials). This forms two structural sets: one using the boundary varieties to define the obstacles, and the other defining charts to represent the manifold. They then sample across the boundary varieties and charts to obtain a path planning solution. Compared to traditional algorithms, this technique not only avoids massive time-consuming projections from the ambient space to the path planning space, but also is able to solve problems for

which the [PPS](#) contains bifurcations and narrow passages [\[103\]](#).

2.2.2 Charting of Manifolds

This section provides a brief overview of the alternative approach to formally parametrizing the manifold. This approach may involve any or a combination of the following. One may define the surface using a system of implicitly defined equations, and create parametrizations approximating the surface locally, called charts. Some authors construct all charts that cover the surface (called the atlas of a manifold), following which suitable charts (points located there) are picked by the path planner. Alternatively, there may exist situations where the bounds of the parameters defining the surface may not be defined, i.e., resulting surface is without a boundary. In such a situation, one may want to construct charts selectively in the direction of the target point to save time and computational resources. Several variants of chart construction methods for manifolds or their discretization exist, although mostly in the form of mathematical tools that can later be applied in the context of robotics. Some others visualize manifolds as triangulated surfaces either from point clouds representing the surface or from closed-form parametrizations. The following therefore focuses on how mathematical tools and path planning/graph traversal algorithms can be used to represent and compute paths on parts of/the entire manifold describing the path planning space.

Sethian first pioneered the Fast Marching Method in 1995, which was later published in [\[150\]](#) as a tool to compute the position of propagating fronts. That is, topological changes, corner and cusp development, and accurate determination of geometric properties, such as curvature and normal direction are naturally obtained in this setting. Sethian and Kimmel [\[83\]](#) formally introduce the Fast marching method to propagate level set functions and build a surface. As an application, they provide an optimal time algorithm for computing the geodesic distances and thereby extracting the shortest paths on triangulated manifolds. Mathematical tools, such as Fast Marching methods [\[83, 150, 151\]](#), enable the robotics community to choose from a wide variety of methods as solutions to visualize complex surfaces and subsequently compute a geodesic distance. The aforementioned work predominantly solved mathematical problems, which made their way into computer vision and with time, slowly forayed into more robotics-related applications. It is interesting to note that such techniques were precursors and alternatives to the formal notions of defining metrics (like the Rips Complex) on a Riemannian manifold, as seen in [\[16\]](#). The onus of constructing a minimal number of charts to represent the manifold is always one of importance [\[103\]](#). As an equivalent venue in the discrete world, several authors explore constructing charts in a discrete fashion, that enable to pave the manifold. For example, Henderson [\[55\]](#) presents a new continuation method for computing implicitly-defined manifolds. The manifold is represented as a set of overlapping neighborhoods, and extended by an added neighborhood of a boundary point. The boundary point is found using a boundary expression in terms of the vertices of a set of finite convex polyhedra. The resulting algorithm allows adaptive spacing of the computed points, and deals with the problems of local and global overlap in a natural way [Figure 2.28](#). The algorithm pioneered the technique to chart a manifold incrementally, based on implicit equations defining a surface, and has spawned several other works based on extending the idea a bit further [\[22, 23, 65, 127\]](#).

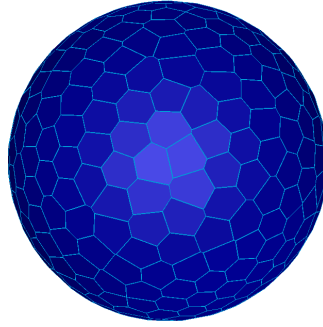


Figure 2.28: Example of several charts (blue polygons) covering/charting the spherical manifold; the union of charts forms an atlas [127]

Bohigas et al. [22, 23, 127] use the higher dimensional continuation method developed by Henderson in [55] to build a parametrization of the $\mathcal{C}$ space. The approach in the aforementioned works extract from the constraint-filled configuration space, a system of equations that formulate $\mathcal{C}^{\text{free}}$ as an implicitly defined manifold. Subsequently, higher-dimensional continuation techniques are used to progressively construct an atlas of the manifold that contains the start configuration, until the goal configuration is reached, or path non-existence is proved at the chosen resolution of the atlas. An example of a charted implicitly defined manifold of $\mathcal{C}^{\text{free}}$ as constructed in [22] is seen in Figure 2.29. In this case, the path planner used on the atlas is A^* , which uses the centres of charts as nodes for path planning. Chart creation, as adopted from [55], is customized so that only those charts that comply with both A^* and the manifold curvature are admitted. The approaches described thus far and throughout this section, use incremental constructions of manifolds. They are vastly different from the methods seen in Section 2.2.1, since no explicit and global parametrization of the path planning manifold is required. Complex configuration spaces are seldom easily explicitly characterized, and more so in the presence of constraints, such as singularities and obstacles. This method however avoids the need of representing the singularity locus explicitly as an obstacle, i.e., no explicit representation of the constraint region is required. Constraints are also represented implicitly as the region that does not comply with the system of equations defining the original desired manifold. The solution manifold of this system can then be freely navigated without fear of encountering any constraints, which in [22, 23] were manipulator singularities. With the ability to both progressively and exhaustively generate the atlas from a reachable configuration, the method lends itself to multi-query planning and visualization. The resolution completeness of the approach is at the expense of a computational cost that scales exponentially with the dimension of $\mathcal{C}$. To deal with higher-dimensional problems, however, one could adapt the approach in [68], which trades off resolution completeness with efficiency and probabilistic completeness. Alternatively, one could adopt the approach in [72] which, additionally, guarantees asymptotic optimality [23]. Another advantage of using higher dimensional continuation methods to represent the manifold is the almost discrete nature of the intermediate results, i.e., that of the charts. The discrete nature, which is the individualized representation for each chart or patch, allows them to be viewed as nodes in a graph. Subsequently, any graph traversal algorithm can be used to pick directions to expand

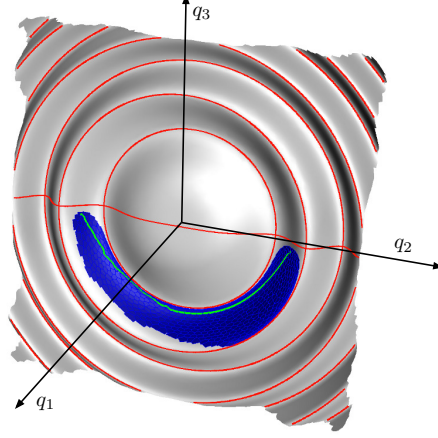


Figure 2.29: Charting $\mathcal{C}^{\text{free}}$ which was formulated as an implicitly defined manifold [22]

the chart generation or to plan paths. As seen in [22, 23, 127], A* was the planner of choice. But with such flexibility, improvements in the form of other graph traversal/tree growing methods can be seen in conjunction with higher dimensional continuation. For example, Jaillet and Porta of [65] combine RRTs with the higher dimensional continuation method to propose a new method called AtlasRRT. The AtlasRRT uses an atlas to efficiently explore a configuration space manifold implicitly defined by kinematic constraints.

To combat the expensive process of defining the full atlas for a given manifold, the AtlasRRT algorithm intertwines the construction of the atlas and the RRT. The partially constructed atlas is used to sample new configurations and to generate new branches for the RRT, with the latter used to determine directions of expansion for the atlas. This approach retains the exploration bias typical of RRT approaches, i.e., the tree is strongly pushed towards yet unexplored regions of the configuration space manifold. Jaillet and Porta of [65] suggest that experimentally, the AtlasRRT is generally more efficient for highly constrained spaces. As opposed to highly constrained regions, mildly constrained spaces imply that the embedded PPS is very similar to the ambient space.

2.3 Path Planning with Dynamic Constraints

This section provides an overview of techniques that perform path planning that avoid dynamic constraints. Dynamic constraints can mean one of the following things. It can be a sudden appearance of a static constraint that was not known apriori, such as a falling object. Alternatively, it can be the appearance of a moving constraint into the region of the path, where the dynamic constraint's location at the time of path planning did not interact with the planned path. Such an example can be seen with the motion of another robot meandering into the planned path, or with the encounter of a human being whose presence was not accounted for previously. We saw that techniques to plan paths while avoiding Static Constraints (SCs) could be broadly classified into methods based on how the PPS was represented. If represented as a set of points, then SAs and OAs may be used. As an alternative, manifold based approaches could be used when the PPS is represented as a manifold.

Path planning techniques that avoid [Dynamic Constraints \(DCs\)](#) can broadly be classified into two main approaches. The first group of methods assumes knowledge of the robot's model and/or of the dynamic constraints, and performs path planning as a local response. This group of methods is referred to as model based algorithms in this thesis. The second group assumes a higher level understanding of the problem and plans/modifies the path as a response to the global path planning problem, without requiring any knowledge of the specific robot model at hand. Typically, this group focuses on path deformation by borrowing concepts from manifold theory, called homotopy deformation; therefore, being referred to as homotopy based algorithms. It can be seen that model based methods typically encompass the methods seen in [SAs](#) and [OAs](#) (discussed in Sections 2.1.1 and 2.1.2), which were used when the path planning environment was modeled as a set of points. Similarly, homotopy based methods are applicable when the [PPS](#) has an underlying topological representation. Model based algorithms and Homotopy based algorithms are detailed below.

2.3.1 Model Based Methods

[Model Based Methodss \(MBMs\)](#) rely on assumed models of the robots, constraints to be avoided and the environment, to plan a path free of said constraints. They may do so either by solving the problem globally or as a reactive mechanism. Global methods that avoid obstacles in motion are usually adaptations of [SAs](#) and [OAs](#). Their being global typically implies that knowledge of the number, potential size, initial locations and maximum velocities of [DCs](#) are available or may be inferred. Then, paths are planned to avoid the assumed/predicted worst case scenario, which is that of a collision. On the other hand, local planners are reactive in nature. That is, without assuming too much knowledge of the constraints to avoid, these planners only react when a collision is indicated to happen at a user/algorithm defined proximity. These locally reactive algorithms, such as model based control algorithms, graph based solvers, reachability and velocity sets of obstacles, and methods seen as part of [OA](#) and [SA](#), are heavily reliant on the quality of sensor data. They also usually resort to some variant of obstacle boundary following, until a clear line of sight towards the target is visible.

Naturally, global and local reactive planners may both be reliant on models of the robot used, constraints expected and the environment, thus giving rise to a variety of model based algorithms. The primary advantage of using model based approaches is that paths generated are suited to be executed for exactly the kind of robot that it was designed for. So the next stage of trajectory tracking becomes easier. The obvious disadvantage however, is that the solutions are restricted to the models defined. Through the literature review, one also notices that, similar to [MPC](#) methods described to avoid [SCs](#), model based methods are better suited to generate trajectories rather than paths. Trackable trajectories sometimes compromise metrics that are of importance in the [PPP](#), such as path length, clear avoidance of [DCs](#) and [SCs](#), smoothness and respecting the underlying curvature of the [PPS](#). In this thesis, generating a path that is free of [SCs](#) and [DCs](#), without repetitive evaluation, takes precedence over generating a trackable trajectory, which may compromise the aforementioned metrics.

In [158], the authors propose a so called fast reactive path planner for two and three dimensional spaces. Their goal is to compute the fastest path from the source to the target location, while reacting to unexpected obstacles. To do this, the authors of [158] propose a diffusion algorithm in a multi-layer spatial grid map, consistently updated with sensor data. The two principal cases for occurrence of an obstacle are as follows. The first possibility is that there is no global information and the robot does not recognize the obstacle until its sensors detect it. The alternate possibility is that, while the robot follows the given path, a message about the potential obstacle is broadcast as global information during motion. The algorithm simulates a physical diffusion process in a workspace that is assumed to have a multi-layer map, where every layer has different kinds of information, such as data about mobile or stationary robots. The target location in the PPS is treated as the source of the diffusion process of a material. The diffusion equation enables the diffusive material to emanate out of its source with a constant concentration and spreads across the diffusion landscape. After a certain relaxation time, the diffusion process becomes stationary, i.e., the concentration of diffusive material will be constant in time at all spatial points [158]. Then, the steepest ascent path from the start point to the destination point can be computed, which will then be followed. The boundary and constraint conditions establish the source of the diffusion process and the avoidance of collisions respectively. A numerical solution is used as an approximation to the original analytical solution, where the landscape’s discretization interval impacts the granularity of the path. Should the multi-layer map see an update, the boundary conditions are changed and the algorithm restarts, by assuming the current position as the new start point. The authors of [158] suggest alternatives to the path initially generated by the diffusion algorithm, should it not satisfy some criterion of optimality. While the robot follows the initially generated path, one alternative looks for narrow passes in the initially found path. The narrow passes will be closed and the diffusion process is started again to produce a newer path. Alternatively, a remedial solution involves closing up those narrow passages by increasing the size of the obstacles, and then restarting the diffusion process.

It is easy to view the diffusion process in the same note as that of OAs, where the general change of energy is expected to stabilize. And in fact, several of the upcoming works that are described, utilize a combination or variant of OAs. On such a note, authors of [116] use the Trulla algorithm that was originally proposed in [64], which is similar to the discretized APF at a high level. In a grid based map, it shows the optimal direction of motion for every cell towards the goal. The algorithm assumes a grid based PPS, and information about each cell is propagated to its neighbouring cells. Every grid cell may potentially form a node in the final path, and therefore its candidacy as a viable sub-goal is evaluated and propagated. Each cell contains weights associated with it, which describe the degree to which that cell can allow movement, and the directions in which movement can be generated. When a new cell is evaluated for propagation, a set of rules determine if a previous sub-goal is accessible or if a new sub-goal must be created. A comparison algorithm also determines the optimal path for each location, rejecting any apparently shorter paths which inadvertently cross a region of higher weight. As the authors of [116] point out, the Trulla algorithm does require the PPS to be weighted apriori, to differentiate between occupied and unoccupied regions. Additionally, since Trulla’s output is essentially a potential field, reaction to dynamic events

merely displace the robot from its original optimal path to a new optimal path [116]. The authors show in Figure 2.30 from [116], an unmodelled obstacle in an environment, the directions of potential motion in every cell, the expected path (solid line), the actual path with Trulla (long dashed line), and the purely reactive path (short dashed line). One can note the similarity between the output of this algorithm and that of discrete navigation functions from [10].

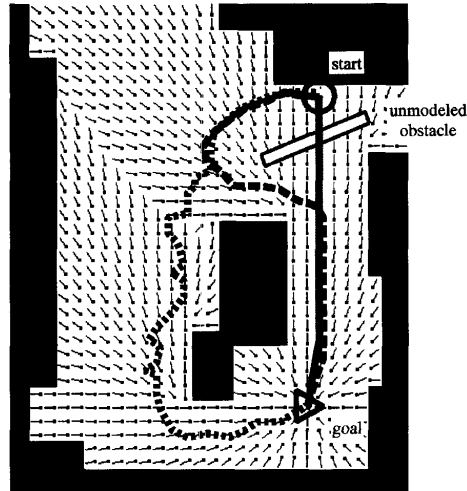


Figure 2.30: Trulla algorithm [116]. The figure shows an unmodelled obstacle in an environment, the directions of potential motion in every cell, the expected path (solid line), the actual path with Trulla (long dashed line), and the purely reactive path (short dashed line).

Authors of [110] propose path planning by exploiting knowledge of the robot model. They propose a path planning approach based on what they call motion strategies. Motion strategies are ways of moving a robot that are designed for specific kinds of tasks, by restricting the motion of a robot to a subset of its operating space. The authors break the path planning into two phases, containing offline and online processes. Using a manipulator for demonstration, the offline phase involves choosing a subset of its configuration space based on the kind of configurations required for commonly performed tasks. Since there may exist various combinations of sub-spaces that allow the same motions, the authors group similar motion problems together, i.e., several kinds of motion operations that can be solved using one strategy are grouped together. Following this, simulations are used to provide an estimate of the probability of faults to enable appropriate association of the motion strategy to the motion problem. Then, the offline stage results in a specific motion strategy for a chosen motion problem. Once the strategies have been chosen, they are used for on-line motion using a local and reactive planning algorithm. This brings the algorithm to its online phase, which attempts to accomplish three tasks- movement towards the goal, movement around the obstacle and changing local direction of movement. These three specific changes in motion are accomplished by commanding changes in the configuration of the robot. This approach may be viewed as similar to either standard bug algorithms where boundary following is a solution to abate collisions; or, it may be likened to a MPC approach, where a control signal is commanded to produce movement in a specific direction. Such similarities can consistently be seen in the literature, because in the absence of other information,

immediate collision-free movements often require boundary following or a blatant turning away from the obstacle. Another observation is the reliance on grid based maps to store information about the changing environment, which is often the easiest way to update the status of whether a small area of the PPS is occupied or not.

Even in [40], authors use a grid based map to eventually perform deliberative online path planning. Here, the grid based occupancy map is the result of data fusion resulting from multiple sensors. Each sensor’s contribution is weighted differently, and thus the final grid based map is expected to reflect a more accurate representation of the known environment. Subsequently, the distance-transform is applied to the grid map, to obtain a measure of proximity to obstacles. The distance transform is an operator that computes a measure of proximity to a certain area of interest. In [40], distance transform computation is equivalent to a wave front expansion, where the values of all the free cells take on the lowest value of the neighbour cells increased by one. The algorithm visits each cell once and works on a window of cells around the original cell. The expansion completes when all of the control points of the global path have been visited. The distance transform accounts for the distance to the goal, so the resulting map is synonymous to a discretized APF. However, as the authors point out, since the target location uniquely defines the values in the grid map, the resulting map is akin to that of a discretized navigation function, with a singular minimum at the target location. Then, the path is extracted by concatenation of cells from the goal to the origin following decreasing values. This results in paths that may be too close to obstacles; therefore, a modification is proposed in the form of a local path improvement. An anytime algorithm is proposed to reduce the risk of collision for the path previously computed. The algorithm computes two values: maximum risk and total risk, which are associated to cells belonging to the path. The maximum risk associated to a cell is the largest of the risk values of the cells on the path, inside a specific sized neighbourhood centred on the cell of interest. In a similar way, the total risk of a cell is the sum of the risk of every cell of the path inside the chosen neighbourhood, centred on the cell of interest. By accounting for all possible values obtained by choosing different cells as preceding or subsequent nodes of the centred cell in the neighbourhood, an evaluation of the risk parameters can be made. Should a better alternative be found for the intermediate nodes in the path, then the original neighbourhood with the original sequence of nodes is now replaced by the newly chosen neighbourhood. The anytime property comes from the fact that only one cell is modified for each procedure invocation. So, after each invocation of the procedure, a better path is achieved. Each invocation of the routine takes the time needed to substitute one single cell on the path [40]. The results demonstrate the direct consequence of the distance transform and the reactive local path improvement approach, which is that of boundary following of the obstacle.

The use of APF is seen yet again in [92], as a part of the local reactive path planning segment. The authors of [92] exploit an existing global motion planning solution called Move3D from [155] to compute an initial path in the given map of the environment. During the motion of the robot, if the on board sensors sense an obstacle that collide with the initially generated path, then the local path deformation algorithm is called. The authors of [92] specifically focus on non-holonomic robots, so their path deformation solution aims to produce such paths that comply with the appropriate directions of motion. The reactive path deformation thus occurs in two stages. First, they show that the perturbations of

the input affect the deformation of the path by modelling it as a linear dynamic system. They do so by modelling the path as the linear dynamic system's state that varies in time; i.e., a change in path results in the deformation of the path. Then, the change in path may be viewed as the directions of motion that allows the original path to deviate from the original path. To drive such a change in the path, appropriate directions must be identified. That is, the direction of path deformation must avoid obstacles and still travel toward the goal. So, input perturbations that drive such behaviour must be chosen. In order to do that, the authors use [APF](#). The potential value of the path is defined by the integration along the path of a potential field over the potential value over the configuration space (the chosen [PPS](#)) [\[92\]](#). Thus, paths going close to obstacles have a high potential value and paths staying far from obstacles have a low potential value. However, this may result in an inadmissible path for the non-holonomic model, since the kinematic model allows specific directions of motion only. This therefore necessitates modification of the newly generated collision free path such that it is composed of purely admissible directions of motion as dictated by the kinematic model of the non-holonomic system. The second stage of the reactive path planning therefore proposes an algorithm that iteratively computes a sequence of admissible paths for a discretized time interval, given an initial path. Naturally then, at each iteration of the algorithm, the admissible path segment is generated based on the a direction of deformation in the original potential field. This work is an interesting example of how optimization based methods can work in conjunction with model specific control algorithms.

On the contrary, [\[164\]](#) approaches path planning amidst moving obstacles, using the concept of growing disks. The authors assume that the maximum velocities of the robot and the obstacles in the [PPS](#) are known beforehand. Then, the authors characterize each moving obstacle as an area of potential collision, which grows larger in time. This region of collision is characterized by the disk on the [PPS](#) initially. To show the change in region of collision over time, the authors introduce time as the third coordinate of the [PPS](#). Therefore, the growing region of collision can be visualized as a cone that grows in diameter as time increases. That is, the cone now gives an estimate of where and how big the region of the collision may lie and at what time instant that can occur. The motive of [\[164\]](#) is to determine the minimal time path among such growing regions of potential collisions. In situations where there exist several moving obstacles, there exist several such growing cones. The authors determine paths by determining bi-tangents of adjacent cones. As the authors mention, paths can be found that are guaranteed to be collision-free in the anticipated time slot, regardless of the behaviour of the moving obstacles. However, a clear limitation of the approach remains that the robot is required to have a higher maximum speed than any of the moving obstacles. In fact, the algorithm requires that the maximum velocities of all its constraints be known apriori. With such knowledge of the robot and the other moving obstacles, one could argue that perhaps a state estimator may be used to estimate the potential behaviour of the obstacles and simply use a [MPC](#) approach to guide the robot to its target location. This alternative suggestion shows that there are several ways to tackle the same sub-problem, and it is essential to solve the global problem in such a way that assumptions about the robot model do not entirely affect the quality of path planning performed.

Methods that rely on [MPC](#) clearly require knowledge of the robot model used, and under

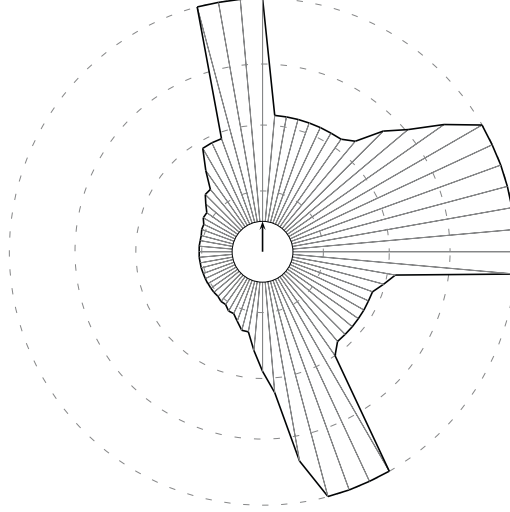


Figure 2.31: An example of the FSH from [108]

this restriction, use of model independent APF based methods may seem appealing. However, it is known that oscillatory motion and local minima pose as disadvantages. Therefore, in an attempt to utilize the merits of APF, authors of [108] propose the Agoraphilic algorithm as an optimistic approach to reactive path planning. The algorithm is a combination of Free Space Histogram (FSH), Free Space Force (FSF), Histogram In Motion Mapping (HIMM) and Virtual Force Fields (VFF) whose contributions are explained in a brief manner as follows. The FSH is a polar map representing the distance profile of the surrounding environment, as seen in Figure 2.31. The HIMM is a method in which an occupancy grid in the form of a polar histogram around the robot, is constructed. The VFF is a modified version of the APF, where virtual forces are constructed to guide the robot towards a desired direction and/or away from another. The Agoraphilic algorithm employs the aforementioned methods in the following sequence. First, the FSH is created by acquiring sensor data around the robot and then perform sensor fusion. Then, for every discretized polar interval or sector around the robot, the depth of the sector indicates the maximum distance of navigable free space in that sector. The depth is established by computing a Certainty Value (CV) based on data from all sensors available; the CV is indicative of how certain the occupancy is at the depth as indicated in the sector. Once the FSH has been created, the next and second step is to generate the FSF corresponding to the FSH that was just computed, to move in the most appropriate direction. The FSF mainly generates only one kind of force, which is attractive in nature towards the general direction of the goal in the collective free space, as opposed to a specific location in the free space. The Agoraphilic algorithm generates force vectors, one for each sector in the FSH. No explicit goal force is used; however a repulsive force is explicitly exerted by occupied locations in sectors. The third step is the generation of the force shaping coefficients, so that the fourth step, which is the weighted summation of the forces is applied. The weighting for each sector is generated by the fuzzy controller and is similar to the importance factor employed by the fuzzy-scaled potential field [108]. The algorithm moves the robot to a point such that the total force acting on the robot is zero. At every point of time, applying different scale factors to the FSFs shifts the equilibrium

point, thus forcing the robot to move. This method is slightly different from that of the traditional APF in that the attractive forces toward the free general direction of the free space and the repulsive force away from the occupied regions are influenced by the weights of the fuzzy system. The effect is to produce the same behaviour as that of APFs, but without producing a local minimum. The authors of [108] aim to utilize the strengths of traditional APF in conjunction with fuzzy logic based methods, and perform tests using a mobile robot in a corridor with hallway openings. While the algorithm's response is sensitive to just one occupied cell, the tests do not reflect its sensitivity to a dynamic environment.

Another approach to handle DCs is by the use of relative velocity methods which are similar to Velocity Obstacle Sets (VOS), which characterize constraints in motion with the help of their absolute and relative velocities with respect to the robot in motion. By augmenting the estimated velocity of such constraints as a function of time, the PPS can be seen to contain regions that are prone to collision at given intervals of time. These regions in the generic sense are called VOS, since such collision prone regions are characterized by spaces which are occupied moving constraints in the due course of time. The authors of [11] propose a variant of such a method for reactive path planning in a environment with DCs. The solution combines linear navigation laws with the notion of the virtual plane and velocity, and orientation windows. The virtual plane is obtained by transforming the reference frame to the relative velocity space. Then using the collision cone method (which is the VOS grown as a function of time in the vertical axis), they determine if collisions occur. The parameters in the linear navigation laws aid in circumventing such collision zones. The method is reactive in the sense that only if constraints in the field of view of the robot will preempt a deviation from the original path. The author of [11] extends the work on linear navigation laws for obstacle avoidance in [12]. By considering the reactive path planning in three dimensions, the authors of [12] segment the parameters of navigation control laws by separating the PPS into the horizontal and vertical planes. More specifically, the robot's heading angle and flight path angle are linear functions of the line of sight angles in the horizontal and vertical planes, respectively [12]. The parameters that affect the control law can be modified to generate non-straight paths to the goal, such that the path may evade the obstacle on the horizontal or vertical plane, as seen in Figures 2.32 and 2.33.

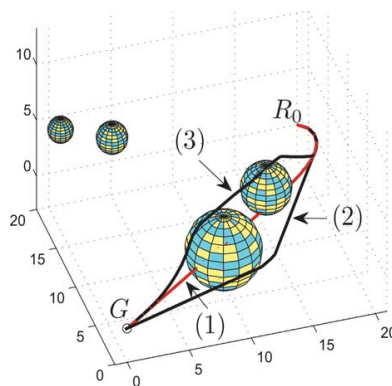


Figure 2.32: Illustration of obstacle avoidance viewed in three dimensions [12]

Reactive path planning does aim to account for changes in the environment that may not

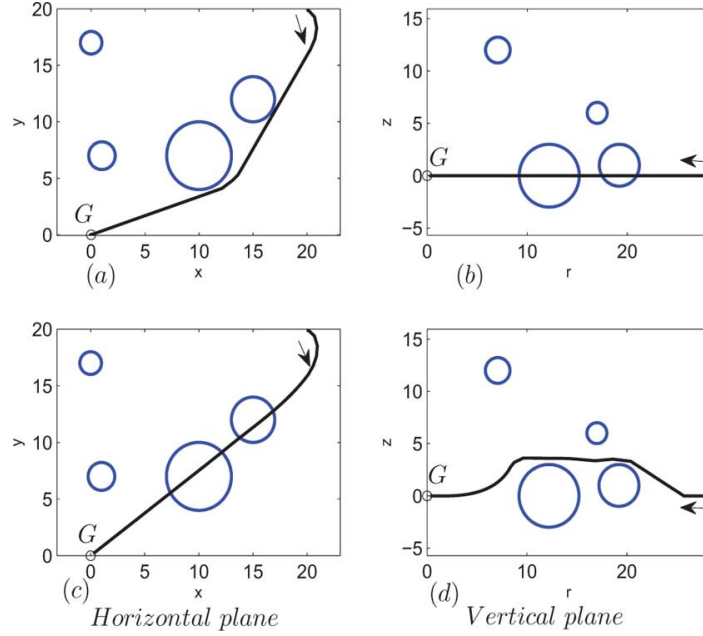


Figure 2.33: Illustration of obstacle avoidance on the horizontal and vertical planes [12]

have been encountered before. However, it may be disadvantageous to only consider such field of view based reactions. That is, ideal reactive path planning methods should be able to respond to changes in the environment as they are sensed and the knowledge of potential collisions, without yet sensing the actual change in the environment. The advantage lies in the fact that reactive path planning methods are required to avoid collisions and change maneuvers quickly, so the path quality and the path objective may be compromised. That is, the smoothness, length or any other metric chosen may be impacted due to the sudden nature with which the collision avoiding maneuver is generated. Additionally, most reactive path planning methods focus on avoiding the collision by following the boundary of the obstacle and or turning away in a direction that is perpendicular to the obstacle, and then pursuing the target location. If the knowledge that a deviation in path is required to be planned earlier is available, then the path re-planning may be performed with a greater focus on reaching the target location using a new path that need not have sudden collision evasion segments. This topic will be revisited in the latter part of the chapter.

In [175], one can again see the use of VOS to determine the ideal velocity of the robot for collision-free safe state. This work brings up an interesting and essential point about invariant safe states, which will be discussed after a brief summary of the approach in [175]. As the first step, the authors of [175] compute the reachability sets in the physical PPS, which are boundaries of those regions that result in collisions as a function of time. These regions are characterized by the constraint's upper bounded constant speed and path curvature [32]. These collision regions are grown further, to account for the robot's shape/size (assumed to be circular) and some space around the said collision regions. This final modified collision region is called the simplified collision region. As a second step, the authors convert the simplified collision regions in the physical space to the velocity space. The work in [49] introduces the

concept of velocity obstacles as follows. Given a collision region in the physical space that must not be entered at a given time period, then there exists a set of (constant) velocities corresponding to this collision region that would drive the robot to arrive at the collision region. This set of velocities to avoid, associated with collisions occurring at a specific future time, is an instantaneous velocity obstacle [175]. Any velocity not belonging to the velocity obstacle, would drive the robot away from that collision region. To account for multiple such obstacles, the authors of [175] determine the union of all such velocity obstacles in the velocity space, resulting in the final VOS. The third step involves determining the appropriate velocities that lie outside the VOS, to drive the robot towards a safe state. This formulation allows them to guarantee a collision-free path, as long as an appropriate time window is chosen for the VOS. Then, in their formulation, even with the choice of an infinite time window, the validity of the VOS is seen to lead to invariant safe states. That is, if a VOS must be avoided at all times, then it may restrict the allowed movements of the robot and possibly prevent it from reaching the goal. However, if the priority of the PPP is to first avoid collisions and only then, to move towards the goal, then several definitions of an evasion maneuver can be used. For example, safe invariant sets/states can be used to represent the target location, an alternative collision-free path or simply no motion at all. These are escape states with speed greater than the velocities of the obstacles, that eventually leave all the obstacles behind the vehicle [175]. The authors make an important observation that if the VOS was to be avoided for all time, a safe state could mean any of the aforementioned alternatives. However, as long as the initial conditions of the robot can be solved for at least one escape velocity away from the VOS, then a collision-free path is guaranteed to exist. The authors explicitly mention that this approach is not complete. Velocity obstacle sets only return single-velocity escape trajectories. There may exist a curved trajectory that extracts the robot from a “surrounded” situation or brings it to the goal, but this method will not find it. However, this method does guarantee that it will never put the robot in such a situation, if the initial condition is not so.

Graph based approaches, such as A*, Dynamic A*, Theta* and other such variants seen in [36] and [43] amongst other works, boast of their ability to be applied to any kind of system. However the generality of the path may sometimes be seen as a disadvantage. This is so, since the path may not entirely consist of segments that can be executed by the specific robot used. MPC algorithms solve that exact problem. However, they are unable to use the advantage of graph based methods to compute the shortest paths. A combination of both such advantages is what is seen with the use of the Tangent Graph (TG), proposed in [149]. The TG roughly explained consists of defining specific tangents that fall in the direction of the robot’s kinematic or admissible directions of motion. A collection of such predefined, admissible directions of motion that are also tangent to the regions of interest such as the initial path and obstacles in the path planning environment, form the TG. Before defining a restricted set of such allowable directions of motion, the assumptions used by the authors of [149] are mentioned. They assume a non-holonomic robot with a hard constraint on angular velocity. The robot is equipped with a vision sensor that is able to detect obstacles, tangents and the target so long as they are in line-of-sight. A tangent segment can only be a segment of the path if it is approached by the robot in the correct direction. The tangent lines may take any of the following forms: simultaneously tangent to two obstacle

boundaries, tangent to the same obstacle boundary at two different points, tangent to an obstacle boundary and an initial circular path, tangent to an obstacle boundary and crosses through the target and tangent to an initial circular path and crosses the target. The individual segments of the completed **TG** can be seen in Figure 2.34. Such a graph is seen to have a finite number of vertices, and thus the **PPP** is reduced to the problem of finding the shortest target-reaching path to a search among all paths in the tangent graph. When the line of sight does detect the target, there is a strategy of exit, depending on where the robot is currently located. When the robot moves along the boundary of an obstacle or an initial circle and reaches a tangent point, it can leave the boundary and move along the corresponding straight line edge of the tangent graph if its heading is equal to the direction of this edge, as seen in the top image in Figure 2.35. In this case, the robot has reached an exit tangent point. Contrarily, a case when the robot cannot leave the boundary at a tangent point is shown in the bottom image in Figure 2.35. The choice that determines what kind of exit point to take (target pursuit or collision avoidance) is characterized systematically in [149]. In addition to exit-point selection strategies, the authors of [149] also group their primary control strategies into three desired behaviours. The first is that of following an initial path, the second deals with pursuit navigation (goal tracking) and the third behaviour is that of boundary following. The important take away from this work is that of the need for exit points from boundary following behaviour. Although not the same as the escape trajectories seen in [175], the idea of requiring intermediate safe states in the form of boundary following or simply staying still is commonly seen. The challenge in such a situation is to always have, as a reliable invariant safe state, the shortest collision-free path.

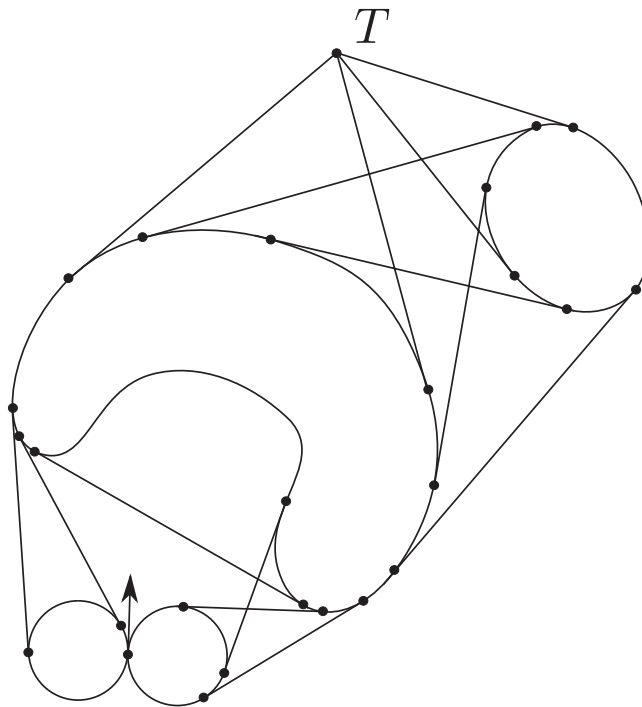


Figure 2.34: An example of a tangent graph with the target location T [149]

OA and **SA** aim to produce optimal paths, a combination of which is seen in [61]. The

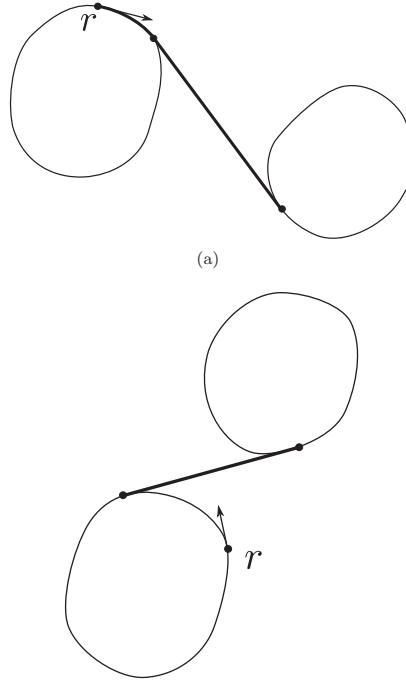


Figure 2.35: Top Image: Exit tangent point can be taken since the robot heading is appropriately aligned. Bottom Image: Exit tangent point cannot be used since the robot heading is not aligned with the exit tangent direction. [149]

algorithm used in [61] generates virtual particles around the robot, from which the ideal direction of motion is chosen. The best direction of movement is determined by using a distance-to-target function and an associated Gaussian repellent cost function. Depending on the resulting cost error, the best particle is determined for further movement. Should there be no obstacle, then of course the particle is chosen on its ability to traverse to the target in the shortest amount of time. The best particle at every iteration is appended and the path is a concatenated sequence of particles. The simulation handles low to medium density obstacles, and produces paths of reasonably average path lengths. This approach is generic and not robot specific. However, it approaches constraint avoidance with a repellent function in a manner similar to that of APF. The authors of [33] propose the **Dubin's APF (DAPF)** to solve the path planning problem for docking in unknown environments based on combination of the Dubin's curves and the APF approach. The Dubin's curve primarily consists of a straight line segment and a circular segment, where the radius of the circular segment is equal to the minimal turning radius of the robot. The authors of [33] aim to generate such Dubin's curves in 3D, a challenging problem since the path segments need to be co-planar as opposed to the case in 2D. Additionally, they also aim to avoid obstacles, which are both known and unknown. The authors use two steps to generate paths comprising of Dubin's curves, that also avoid obstacles. First, they propose a deterministic method formulated as a least squares problem solved with the Levenberg-Marquardt algorithm, to determine the parameters of the desired Dubin's curve in 3D. Although the Dubin's curves are feasible paths, it is difficult to determine the collision free nature of the path and the safety

of the robot without further processing. To overcome this shortcoming, the authors combine the Dubin's curves with the APF approach to avoid collisions with obstacles. The potential field enables determining the feasible intermediate configuration for obstacle avoidance while considering both the positions and velocities of the obstacles [33]; an illustration of its effectiveness is seen in Figure 2.36. While the authors of [33] modify the potential field function to produce intermediate movements better suited for Dubin's curves, the generality of APF based approaches is still seen to be used for different models of robots. For example, the authors of [60] first connect the source and target locations using a sequence of parabolas whose sequence is generated by A*. However, the general APF is used to handle any unseen obstacles.

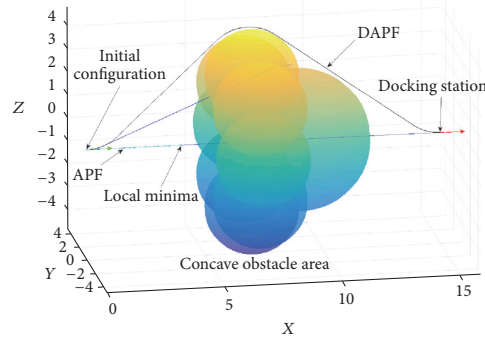


Figure 2.36: The comparison between APF and DAPF as illustrated in [33]

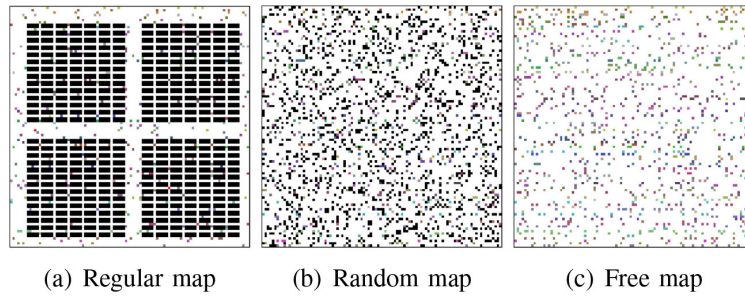


Figure 2.37: Figures (a), (b) and (c) show regular, random and free maps respectively, where black and colored points represent static and dynamic obstacles respectively. [169]

In [14], a method called **Headed Social Force Model (HSFM)** is used for assistive robots. Although not necessarily the same as fully autonomous robots, it is useful to see the parallels of obstacle avoidance techniques in the field of assistive robotics, where collisions are seldom catastrophic. The **HSFM** assumes that the agent using the assistive robot tend to move in the environment under the influence of forces. It is seen that the movement of such agents is usually in the direction of heading, much like non-holonomic systems. Thus, the adoption of such a model gives a nice interpretation of the mechanism that leads to the formation of human trajectories, i.e., the minimization of jerk, which is the derivative of path curvature [14]. The authors of [14] propose a reactive planner using the **HSFM** to predict

the possible future motion of obstacles (humans, in this case). The planner minimizes the deviation from the originally planned path as the parameter to minimize. The probability of collisions is computed analytically and the trajectory generated takes the form of clothoid curves. When an obstacle is detected, the algorithm generates a number of possible clothoid curves, each associated with a certain velocity profile and target location. Each curve is associated with the probability of use, which in turn derives from the likelihood of its target and velocity profile. By considering the probability associated with each of these curves, the planner computes the total probability of a collision and identifies the trajectory with an acceptable collision probability that minimizes the deviation from the original path. Clearly, this application does not require a specific target location to be reached. However the general idea of alternative safe paths that minimize the deviation from the collision ridden path is a metric that will remain a common objective for several approaches throughout the literature.

A different variant of the [APF](#) is seen in [8], where the idea of an artificial current is used instead of a potential. Seeking inspiration from the laws of electromagnetism where a particle close to a current carrying conductor experiences a magnetic field, the authors of [8] draw a parallel to robotics. The artificial current flowing on the obstacle surface located at a certain distance from the robot is designed in such a way that a generated force guides the robot away from a head-on collision and follows the obstacle boundary instead. The authors of [8] tackle the problem of unknown obstacle geometry by assuming that the sensed obstacle is perpendicular to that of the robot, in which case the desired motion is perpendicular to the line connecting the robot and the obstacle. As such, the final motion that is produced encourages the robot to travel parallel to that of the obstacle boundary, thus performing boundary following at least momentarily. Similar to the local minima of the [APF](#), this algorithm may encounter a situation of zero robot speed, or even local oscillations. Yet another similar approach is seen in [101], where the path planning problem of a mobile robot is treated similar to that of heat conduction on a surface. The objective of [101] is similar to that of [8], where path re-planning is the main focus. It is widely known that such behaviour is associated with such attractive/repellent force exerting methods, and learned behaviour may aid in avoiding such pitfalls. Authors of [169] use reinforcement learning to compute path deviations from the originally planned global path. As inputs to the network, the authors use the local segment of the global path, the current local observation and a history of local observations until that time segment. The output is an action at every time step that enables the robot to make the next move. Note that the authors use A^* to plan the global path initially and use the reinforcement learning only for path re-computation in the case of collisions. Training on environments such as regular, random and free maps as shown in Figure 2.37 helps emulate an environment such as a warehouse. This warehouse example may contain a regular density of static and dynamic obstacles (regular map), random density of obstacles (random map) or a fixed density (free map). The performance metrics used are moving cost, detour percentage and computing time at every time step during the testing phase. The moving cost indicates the ratio of actual moving steps with respect to the ideal number of moving steps, without considering any obstacles. The detour percentage is calculated with A^* by only considering the static obstacles. This metric indicates the percentage of detour with respect to the shortest the shortest path length. Performance in unknown environments is seen to be fair, however, a learning based algorithm is only as good

as its training data.

Authors of [52] use a different learning based approach called the **Dynamic Movement Primitives (DMP)** in conjunction with **APF**. According to the authors of [52], **DMPs** learn a trajectory from just one demonstration. They encode the trajectory in a system of second-order linear ordinary differential equation (ODE), where a forcing term is learned as a linear combination of predefined time-dependent functions. Some of the advantages of **DMP** are the flexibility in learning trajectories, convergence of executed trajectory to the target and the execution of trajectories at different speeds by changing only one parameter. For the purposes of obstacle avoidance, the authors design a new potential function called the dynamic potential, such that it depends on the distance between the robot and the obstacle, the angular and linear velocities of the robot and the heading of the robot with respect to the obstacle. While the proposed dynamic potential formulation generates fewer oscillations in the proximity of obstacles, the major drawback is that they do not guarantee convergence to the goal, which is a typical issue with potential-based formulations [52]. With [168], the representation of the relevant parts of the **PPS** are considered. The authors use two kinds of spaces called the time layered and time augmented spaces. The former is a collection of the path planning environment at different intervals of time, such that every interval contains information about potential positions of obstacles. The time augmented path planning space consisted of the time layered space, with just those time slices which would necessitate path re-planning, as seen in Figure 2.38. State estimators, learning methods, optimization and sampling algorithms are all equally opportunistic approaches to the problem of path re-planning, and yet, all their individual demerits cannot be counteracted all at once. Sometimes, authors such as those of [6, 74], even propose a general but deeper investigation into the high level sequence of obstacle avoidance, such that any subsequent choice of algorithm may implement the path planning in a better and advantageous manner.

2.3.2 Homotopy Based Methods

Methods seen in Section 2.3.1 are adaptations of **SAs**, **OAs** and **CDGTs** methods to the problem of **DC** avoidance during path planning. An alternative class of methods based on concepts of homotopy and homology exists, which will be referred to as **Homotopy Based Methods (HBMs)**. **HBMs** rely on the fact that paths planned in a space with constraints follow a certain structure. Roughly explained, the structure shows how the path crosses the region between obstacles, i.e., the sequence in which it occurs and the kinds of turns the path takes while traversing the environment. Knowing the structure of the path helps generate other paths with the same structure or alternatively, can help choose a different structure of paths. This structure that defines the nature or deformability of the path is called the homotopy class of the path. With a basic understanding of the premise of using homotopy classes, one can see that the literature in the area focuses on two main concepts. First, they recognize and compute homotopy classes in the **PPS**, and second, they attempt to augment existing **SAs**, **OAs** and **CDGTs** with homotopy classes.

The problem of testing for homotopy paths on the plane has existed as a problem in computational geometry [27], and has only recently started moving into the world of robotics

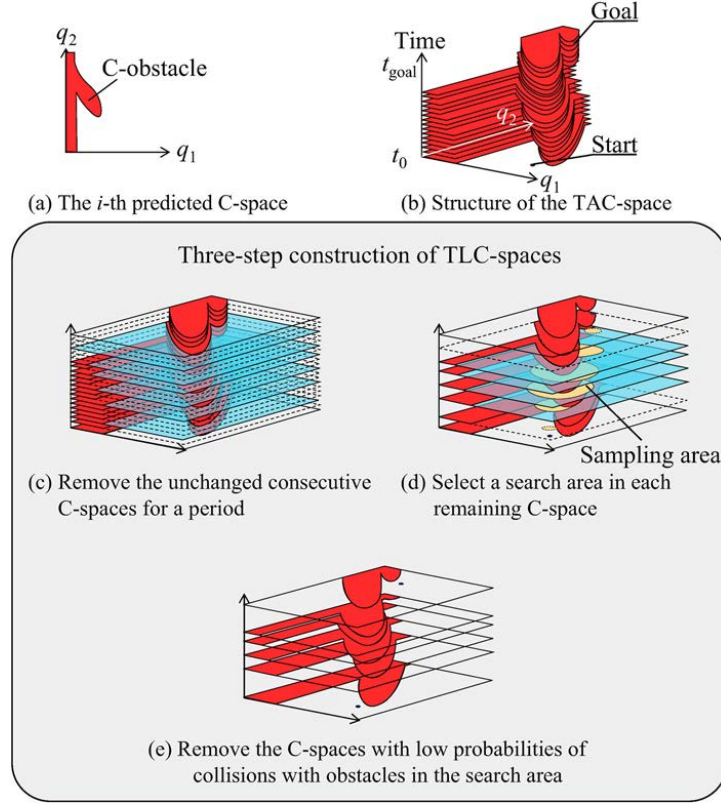


Figure 2.38: The time augmented and time layered environments as explained in [168]

for path planning. Works such as [18, 19, 82, 90] predominantly focus on identifying and representing homotopy classes for path planning. Authors of [19] succinctly define homotopic trajectories as the following. Two trajectories τ_1 and τ_2 connecting the same $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, respectively, are called homotopic if and only if one can be continuously deformed into the other without intersecting any obstacle, as seen in Figure 2.39. Then, sets of homotopic trajectories form homotopy classes. The different homotopy classes are seen in Figure 2.40, where the difference in the classes correspond to how they navigate through obstacles. So, homotopy classes can have different implications when being evaluated for path planning. That is, path planners may simply desire to generate a path that is constraint-free, in which case any of the classes may be picked from. Alternatively, a path planner may also require to avoid certain classes due to some heuristic measure, such as the probability of encountering an increased number of DCs, general path length and the class distance from SCs that it encounters along the way. Clearly, there is a benefit to using homotopy classes. One may check if two arbitrary paths are homotopic, or equivalently, verify what homotopic class the path falls into. This however is a challenging problem. One needs to construct such a measure of deformability of every homotopy class, such that paths in the same class bear the same measure, as opposed to paths from other classes.

The authors of [19] propose constructing a functional (linear one-form) for a trajectory, such that its value uniquely identifies the homotopy class. This functional of the form $\mathcal{H}(\tau)$ is called the H -signature of the trajectory τ . It is beneficial that the H -signature take the

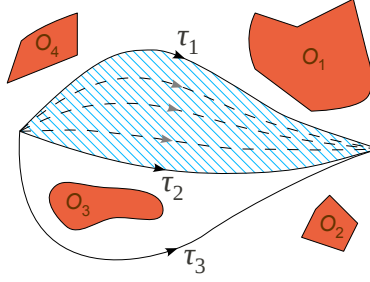


Figure 2.39: Homotopic and homologous paths [19]

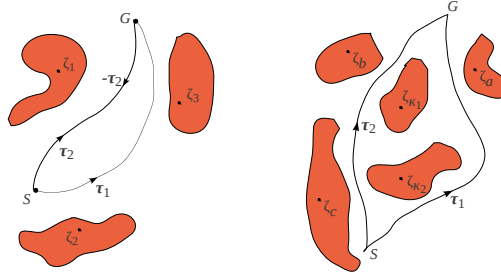


Figure 2.40: Paths in different homotopy classes as seen in [19]

form of an integration as such, $\mathcal{H}(\tau) = \int_{\tau} dh$, where dh is also a linear one-form. This way, the chosen one-form may be a heuristic, such as path length, and may be integrated along the path. The authors of [19] compute one such *H-signature* in their earlier work [18] using complex analysis. In fact, one may rely on the work from [18] to comprehend the use of homotopy classes in path planning, and an excerpt of the work is explained. This explanation is only meant to serve as a high level interpretation of homotopy classes and their relevance to constraint avoidance, and not as a formal mathematical proof. Readers are directed to [18, 19, 82, 90] for any formal statements of the concepts explained here.

Across all their works, the authors of [18, 19, 82, 90] focus on the following three goals. First, the homotopy and/or homology classes in a PPS are required to be detected. Second, specific mathematical tools are required to represent constraints that need to be avoided and a measure that indicates the homotopic class in which the test path currently lies. Third, this homotopic measure of a path needs to be integrated with path planning for the path planner's needs. In order to accomplish the first goal, the authors of [18] use holomorphic functions (analytic functions in the complex plane) to formulate the path and the PPS in the complex domain. In the complex plane $\mathbb{C}$, assume a simply connected region $\mathcal{R}$ within which there is a closed and oriented contour γ that encloses a point z_0 and a holomorphic function $f : \mathbb{C} \rightarrow \mathbb{C}$. Then, the Cauchy Integral theorem affords the following:

$$\oint_{\gamma} f(z) dz = 0 \qquad \oint_{\gamma} \frac{f(z)}{z - z_0} dz = 2\pi i f(z_0)$$

That is, the effect of the holomorphic function f on every point of the path γ , around a point z_0 , can be represented as a holomorphic function (acting on the path) with a pole at

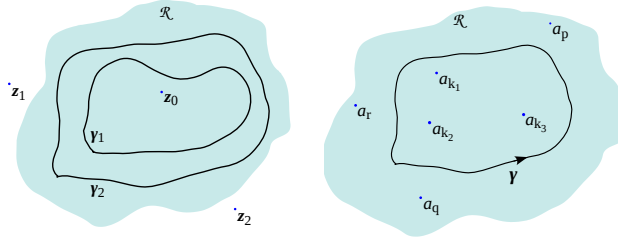


Figure 2.41: The integral of F depends on the enclosed points [18]

z_0 . This pole indicates that the function is holomorphic everywhere except at the pole. The Residue theorem provides an expression to quantify the effect of a holomorphic function F with M poles, on a contour γ , enclosing only $m < M$ poles.

$$\oint_{\gamma} F(z)dz = 2\pi i \sum_{l=1}^m \lim_{z \rightarrow a_l} (z - a_l) F(z)$$

This shows that the value of the integral of F is solely dependent on the points that the path γ encloses, as seen in Figure 2.41 [18]. This is of consequence in the context of the PPP, since obstacles/constraints can be represented as such poles. The homotopy class within which the path falls, is entirely dependent on the poles which the path encloses. Recall that homotopic trajectories are defined as continuously deformable paths, without intersecting obstacles. So, if one can define the obstacles as the poles of $F(z)$, then the homotopic classes can be defined. The authors of [18] do so by defining an obstacle marker function as follows:

$$F(z) = \frac{f_0(z)}{(z - \xi_1)(z - \xi_2) \dots (z - \xi_N)} \quad (2.8)$$

where f_0 is analytic over $\mathcal{C}$, and each of the N poles is a representative of each obstacle or constraint to be avoided. The intuitive understanding of the obstacle marker function can be seen with the function

$$f_i(z) = \frac{f_0(z)}{(z - \xi_1) \dots (z - \xi_{i-1})(z - \xi_{i+1}) \dots (z - \xi_N)} \quad (2.9)$$

defined for each of the N poles. Using $f_i(z)$ in (2.8) results in $f_i(z) = (z - \xi_i)F(z)$, which is analytic inside regions that do not contain the poles of (2.9). This, in conjunction with the integral being influenced only by the poles that the path encompasses, provides a way to identify the different homotopic classes of paths in the PPS. The authors of [18] define the *H-signature* of the path as

$$L(\tau) = \int_{\tau} F(z)dz \quad (2.10)$$

for a trajectory $\tau \in \mathcal{C}$. In this specific case, the authors refer to the *H-signature* as the L-value of a trajectory. They define (2.10) as the unique descriptor of a homotopy class, since the *H-signature* of trajectories (connecting the same two points) in same homotopy class are

equal, while the *H-signature* of trajectories in different homotopy classes are different. Thus far, the authors of [18] accomplish detecting homotopy classes and representing constraints, initially described as the first two goals. The last goal was to combine such information with path planning methods, for which the authors rely on a graph based representation. They construct an exploratory graph, whose edges and vertices are augmented with the L-value. The L-value of every edge e connecting vertices z_1 and z_2 in the *L-Augmented Graph (LAG)* is computed either numerically or analytically (as seen in [18]).

As the *LAG* explores new nodes, the new edges are augmented with L-values, which is illustrated in Figure 2.42. This *LAG* can be searched for a path using any standard graph traversal technique. Since every homotopy class has a distinct complex number associated with it, the *LAG* may be equipped with homotopy classes to either specifically navigate in, or avoid. Figure 2.43, shows the impact of using A* on an exploratory *LAG* with three cases: without specifying a homotopy class, choosing a specific class and avoiding all classes. In the second case, blocking a homotopy class prompts the search for a solution in another homotopy class. As the authors point out, the exploration of the homotopy classes can be performed in a single run of graph search. Finally, in the third case, where both homotopy classes are blocked, the search is biased towards finding paths in different homotopy classes in the order of their path costs. Such paths may include non-Jordan (self-intersecting) curves. More such examples are seen in Figures 2.44 and 2.45. It is important to note that the *H-signature* computed in the *LAG* are complete invariants for homology classes as opposed to homotopy classes [19]. For the purposes of path planning, homology and homotopy may be considered to provide the same information. Two paths τ_1 and τ_2 (with an orientation opposing that of τ_1) are homologous if τ_1 and τ_2 together form the boundary of a 2-manifold free of constraints. Referring to Figure 2.42, one can see that homotopic trajectories are always homologous, and non-homotopic paths are non-homologous. While the converse may not always be true, the authors of [19] note that homology serves as a fair analog to homotopy in robotic path planning.

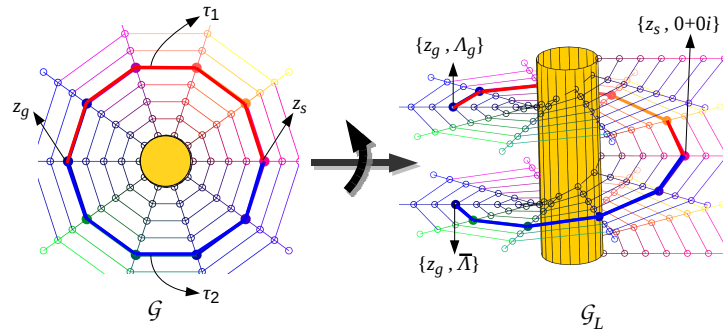


Figure 2.42: The graph on the left side as opposed to the *LAG* on the right side [18]

The authors of [18] extend their work in 2-dimensions to 3-dimensions in [90]. While the work in 2-dimensions using complex analysis is a compact way of generating homotopy classes that is independent of the geometry of constraints and robust to sensor noise [90], the technique applies only to 2-dimensions. The extension predominantly deals with representing obstacles or constraints as surfaces with genus-1 or more [19]. This is of importance, since

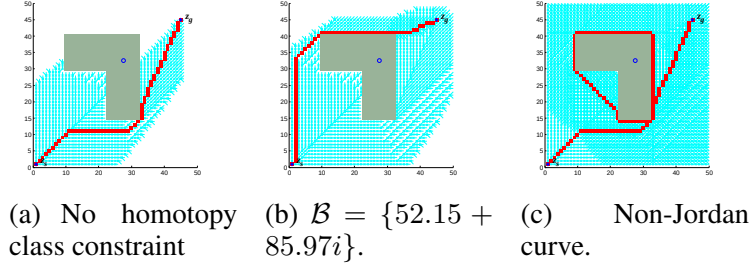


Figure 2.43: Impact of LAG and A^* in non-blocked, some blocked and all blocked homotopy classes [18]

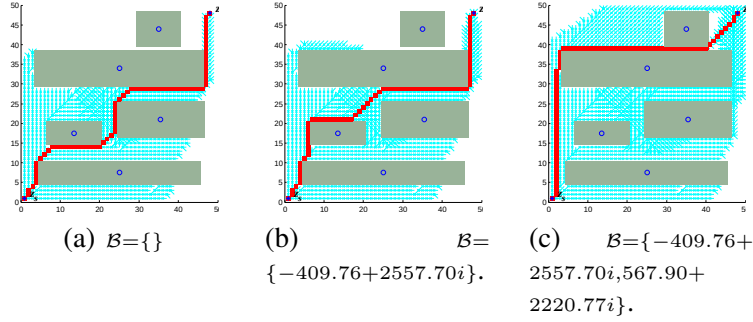


Figure 2.44: Paths in different homotopic classes [18]

in a 3-dimensional PPS, constraint regions with genus-1 or more, can induce more than one homotopy class. Figure 2.46 shows examples of such constraints that permit multiple homotopy classes. This is opposed to the behavior of genus-0 surfaces, such as a sphere or a solid cube, which cannot induce the same behaviour. Then, one can no longer use a representative point for the constraint or obstacle in the obstacle marker function. The authors of [90] propose representing such a constraint by its skeleton. They do so by relying on Biot-Savart's Law and Ampere's Law to define a differential 1-form, the integration of which along trajectories allow them to distinguish between different classes of trajectories. By using skeletons of constraints, the authors of [90] are able to exploit the same objectives

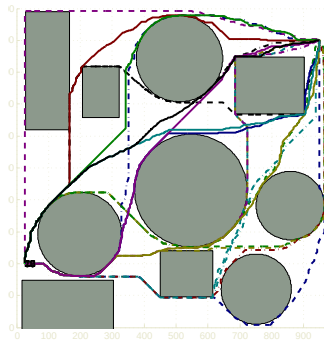


Figure 2.45: Paths in more cluttered environments in different homotopy classes [18]

from the 2-dimensional case but now in 3-dimensions, which are identification of different homotopy classes and combining them with graph traversal algorithms to compute least-cost paths. The resulting algorithm is still independent of geometry and sensor noise, and can be integrated with graph traversal algorithms. A brief summary of their approach is now explained.

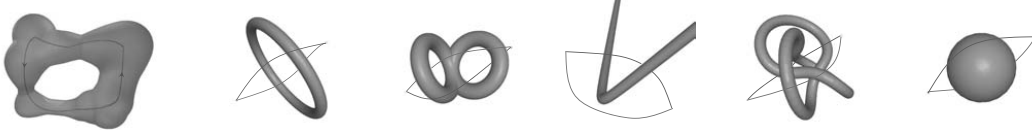


Figure 2.46: Surfaces admitting and disallowing multiple homotopy classes [90]

In 2-dimensions, a finite sized obstacle can theoretically induce multiple homotopy classes for trajectories joining two points. However, the notion of homotopy classes in three dimensions can only be induced by obstacles with genus-1 or more, or with obstacles stretching to infinity in two directions. This can be explained with Figure 2.47 from [90]. Note that a path around surfaces without any genus, such as a sphere or a solid cube, can simply be shrunk to a point beyond the size of the surface itself. That is, every path around the surface may be deformed to a point, therefore multiple homotopy classes cannot exist. It is not the case for surfaces with genus-1 or more, since there can exist paths that cannot be deformed into one another. To simplify the concept of homotopy classes and obstacle surfaces in 3-dimensional space, the authors of [90] perform two main tasks. First, they generate a representation in the form of a skeleton for a surface that induces only one homotopy class, called the skeleton of a **Single Homotopy Inducing Obstacle (SHIO)**. A **SHIO** is defined as a bounded obstacle of genus-1, such as a torus. Subsequently, as a second step, they convert generic obstacles with genus- k into a collection of k **SHIOs**, each with genus-1. Since the skeleton of the **SHIO** is used as an equivalent representative point for the obstacle marker in 3-dimensions, one must ensure that the skeleton is a representation that does not alter the topology of the surface. So, the skeleton of a surface is defined as follows. A 1-dimensional manifold, S , is called a skeleton of a **SHIO**, $\mathcal{O}$, if and only if S is homeomorphic to $\mathbb{S}^1$ (a circle), S is completely contained inside $\mathcal{O}$, and if S and $\mathcal{O}$ are homotopy equivalent. That is, if the obstacle $\mathcal{O}$ is replaced by an equivalent obstacle S , then the homotopy equivalence between two arbitrary trajectories, τ_1 and τ_2 , connecting every pair of fixed points in the environment, will remain unchanged [90]. The conversion of genus- k obstacles into k **SHIOs**, with overlap and each being genus-1, allows k skeletons to be defined for the original obstacle of genus- k . This simplifies the computations of **H-signature** of paths.

The general principle of **H-signature** remains the same irrespective of dimensions. The only difference lies in the principle used to compute it. The authors used complex analysis in the 2-dimensional case. They relied on laws of electromagnetism in the 3-dimensional case. They model the skeleton S of **SHIO** as a current carrying manifold. This can be performed for every genus- k obstacle, by converting it to k skeletons of **SHIO** obstacles of genus-1. Next, given every skeleton S_i corresponding to **SHIO** $_i$, they define a virtual magnetic field vector generated at a point in space, as a result of the current carrying conductor (skeleton). Then, the **H-signature** for one skeleton is defined as the integral of the virtual magnetic field,

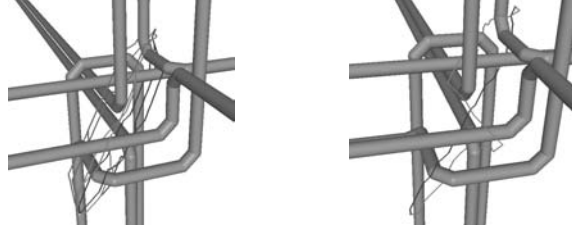


Figure 2.47: 10 distinct homotopy classes in 3-dimensions [90]

along the path τ . Should there be M skeletons, then the *H-signature* is a vector consisting of M individual *H-signatures*. Post this step, the approach is similar to that of [18], where an *H-signature* augmented graph is computed. This augmented graph contains allowed and blocked homotopy classes, and any graph traversal algorithm may be used to find a path. The results of the algorithm may be seen in Figure 2.47 and as applied to a dynamic environment in Figure 2.48. The environment is 2-dimensional, with time as the third dimension. The dynamic obstacles are assumed to move in an oscillatory manner, and their skeletons are traced by the movement of their central representative points. Figure 2.48 shows that the trajectories in the non-trivial homotopy classes go behind the obstacles, a region that would otherwise not be visited by the least cost path without any homotopy class consideration [90].

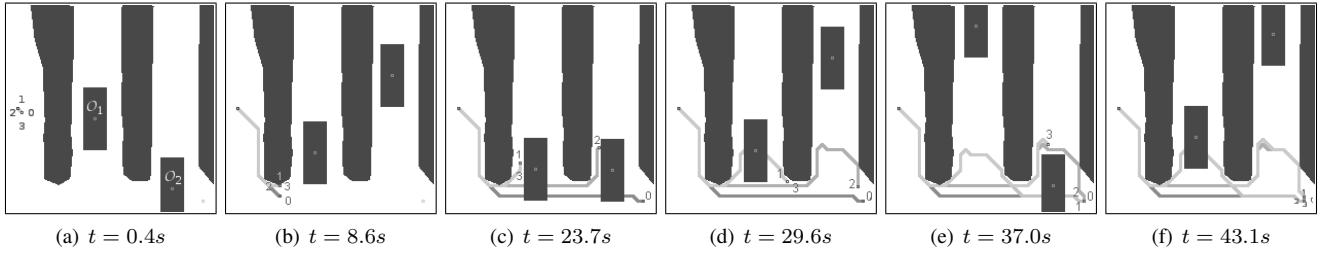


Figure 2.48: Homotopic path planning in a dynamic environment [90]

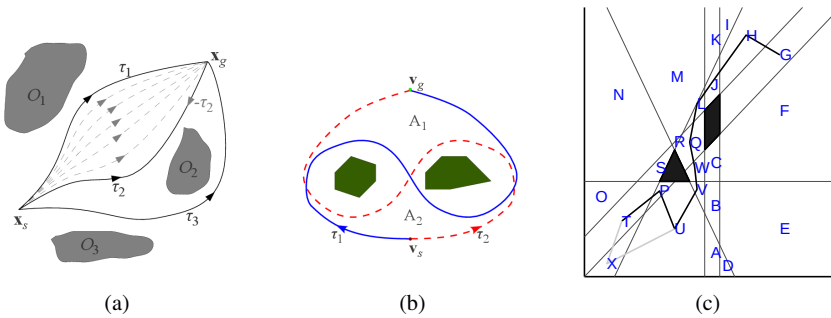


Figure 2.49: Word based homotopic path planning [82]

The shift towards discretizing the identification of homotopy classes for path planning was beginning to be seen in the works of [18, 19, 90]. Their work in [82] is one of the first that discretizes the *H-signature* of a homotopy class and then uses mixed integer linear

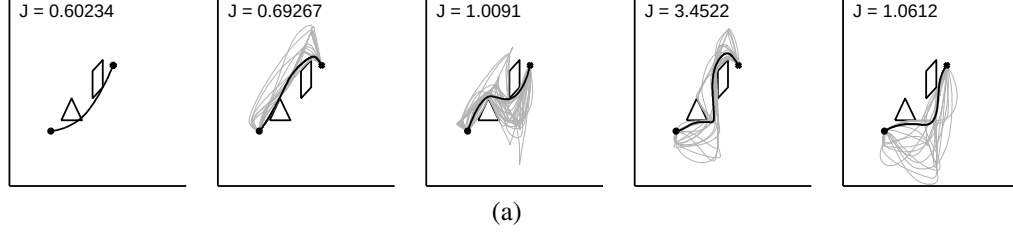


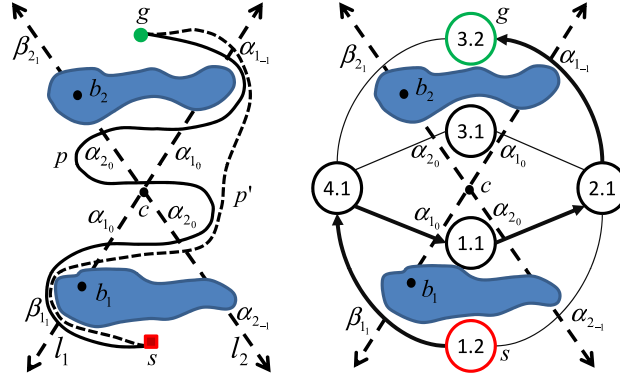
Figure 2.50: Evolution of sub-optimal paths in grey to the optimal path in black [82]

programming to find the optimal path in the chosen homotopy class. They approach the problem of trajectory generation in 2-dimensions, and divide it into two main parts, as follows. The first part involves describing the homotopy class using a “word”, which is a coarse representation of the trajectory, and in turn the homotopy class to which it belongs. In the second part, they determine the most optimal path corresponding to the word describing the homotopy class. Now, each of the aforementioned parts is explained in detail. To accomplish the first part, the authors of [82] partition the PPS, and represent each partition by a letter, as seen in Figure 2.49. It is easy to see that any path can be represented by a sequence of letters indicating the parts of the environment that the path crosses, and in turn the constraints/obstacles that the path encloses. Naturally, there can be more than one word associated with a homotopy class, since more than the minimum number of partitions in the same homotopy class can result in several non-unique words that describe that class. The *H-signature* of the trajectory τ with respect to a representative point of obstacle j is

$$H_j(\tau) = \int \frac{1}{z - z_j} dz = \int_{t_0}^{t_f} \frac{1}{\tau_x(t) + i\tau_y(t) - z_j} (\dot{\tau}_x(t) + i\dot{\tau}_y(t)) dt \quad (2.11)$$

Note that (2.11) considers time to evaluate the *H-signature*, as opposed to (2.10), which did not do so. The *H-signature* for a trajectory encompassing n obstacles is given by $H(\tau) = [H_1(\tau), H_2(\tau), \dots, H_n(\tau)]^T$, similar to what is seen in the authors’ original works in [18, 19, 90]. Before the *H-signature* can be used to determine the optimal path, one must describe the homotopy class of interest in detail. Given a source and target location, the resulting path falls into a certain homotopy class, called the desired homotopy class. This initial trajectory helps detect the homotopy class of interest, using which an optimal path in the class can be determined. To find an optimal trajectory satisfying the given homotopy class constraint, they first construct the set of words in the same homotopy class as the required one; this set is denoted by W_h . Then, as part of the second step, every word in W_h undergoes an optimization process that minimizes the time spent by the path in each letter or partition of the PPS. In due course, the most optimal path corresponding to each word is returned such that the path is made up of segments with the least amount of time in every partition. The optimization process imposes the *H-signature* as seen in (2.11) as a constraint to be adhered to; additionally, the time spent in each partition may also be imposed as a constraint. The resulting problem is non-linear, so the global minimum for the optimized trajectory may not be guaranteed. However, the iterative nature of the algorithm does produce a trajectory whose cost is further minimized, as opposed to that of the initial solution. So, it is considered as an anytime algorithm that provides better solutions with time. Consider Figure 2.50,

which results from simulations in a planar environment. The authors use four homotopy classes and generate optimal trajectories in these classes. The figure shows the impact of not using a homotopy constraint on the left most subplot, whereas the remaining four subplots show the imposition of four different homotopy constraints. The optimal trajectory is in black in each homotopy class, and the gray lines depict the sub-optimal, iterated over trajectories in that homotopy class. With this work, the authors of [82] have successfully introduced systematic optimization of paths being generated.



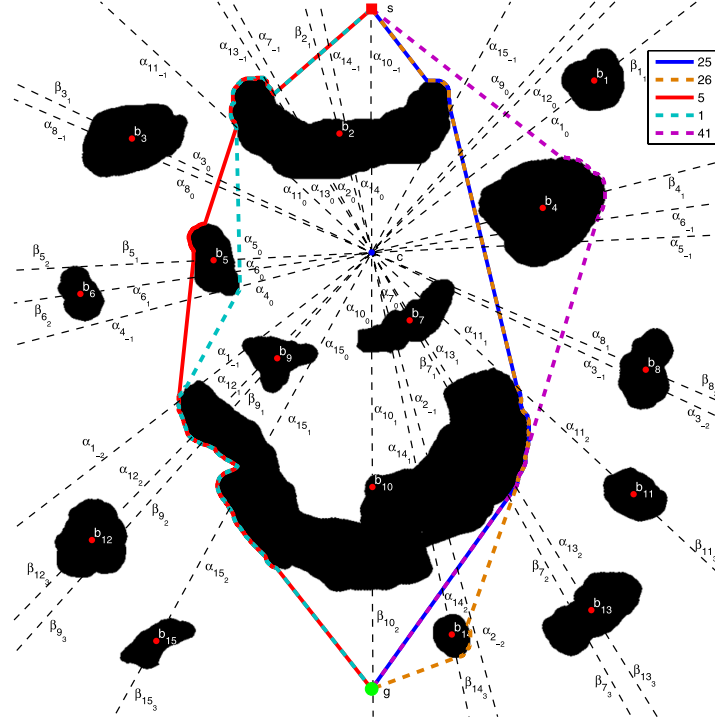


Figure 2.53: HBug and homotopy classes [56]

A* (HA*) [57], the Homotopic RRT (HRRT) [58] and the Homotopic Bug (HB) [59]. They ensure completeness, because in case the goal is not reachable, no homotopy class exists and, consequently, no paths will be generated. The homotopy class of the global optimal path is guaranteed to be generated by the algorithm [56]. A prime advantage of driving exploration in the regions favoured by homotopy classes is that it allows generating good solutions at a fraction of the computation time taken by exhaustive search methods [56]. The authors generate homotopy classes by following the approach outlined in [69] for any 2-dimensional workspace with obstacles by turning the environment into a topological graph, which allows the systematic generation of homotopy classes using graph-search algorithms. The method first builds a reference frame which determines the topological relationships between obstacles in the workspace and is used to name the homotopy classes. The reference frame is then used to build the topological graph which allows the computation of homotopy classes systematically. Similar to the word generation seen in [82], this approach involves determining a canonical sequence of operations or transformations that correspond to every partition of the environment. Then, paths in different homotopy classes can be seen to have a different sequence of operations with respect to the origin of the reference frame. Such a representative path is called the canonical sequence. The canonical sequence is the simplest representation of a path without changing its topology, and only one canonical sequence exists for each homotopy class, but each canonical sequence can be represented by infinite trajectories in the workspace [56]. Figure 2.51 represents one such canonical sequence. Having understood the canonical sequence, the aim of the work is to systematically determine canonical sequences for every homotopy class. For this, the authors construct the topological

graph. In the original reference frame, a path is defined according to the segments it crosses, whereas in the graph, it turns into traversing the graph from the source to target nodes. The graph on the right in Figure 2.51 depicts the same canonical sequence seen on the left side in Figure 2.51, in the topological graph. Following the construction of the topological graph, a breadth-first algorithm is used to traverse the graph. Starting from the source node, it explores every neighbouring node until the goal is found. The difference between the standard breadth-first algorithm and the one used here is that the former stops when all vertices have been visited; the latter however, continues until there are no more homotopy class candidates to explore or the length of the last homotopy class candidate is larger than a given threshold [56].

The number of homotopy classes generated by the algorithm highly depends on the number of the nodes in the topological graph. To set up a ranking system for the classes, the authors use a modified version of the funnel algorithm. It computes a quantitative measure for each homotopy class estimating its quality, by estimating the lower bound of the optimal path in the region of the selected homotopy class [56]. Once the sorting of homotopy classes is complete, three path planning algorithms are used to compute a path in the homotopy class of choice- A^* , RRT and Bug. This process implies that the topological graph is converted into a metric path, with the reference frame being the key and only link between the graph and the original environment. The HA^* is a graph-based search algorithm based on the A^* which only explores the zones in the workspace that satisfy a given homotopy class, as opposed to the entire space. It does so by checking the intersections with the reference frame before taking into consideration the node as a candidate to be explored. The advantage of HA^* is that it is complete. The second variant used is that of the HRRT is based on the RRT. HRRT allows a constrained growing of the tree only in those directions that satisfy a given homotopy class. Similar to HA^* , adding a new node into the tree is done only if the topological path traversed belongs to the homotopy class. It can be seen that the HRRT is probabilistically complete for a given homotopy class.

The final algorithm used is that of the HB, based on Bug 2 from [106]. The opportunistic algorithm exploits the already computed lower bound path generated by the funnel algorithm, which was used to determine the preference of homotopy classes based on their lower bound path estimates. The segments in the reference frame constrain the regions the paths can go through, but do not take into account the shape of the obstacles. So should a collision be imminent, the HB performs boundary following according to the homotopy class until the lower bound path leaves the obstacle. The authors note that the HB is complete; however, no explicit assumptions about the obstacles are made. In that case, it is safe to assume that their conclusion may be based on convex obstacle regions only. The efficiency and scalability of the proposed methods have been tested in synthetic scenarios mimicking a cluttered environment. Figure 2.52 depicts the paths of two homotopy classes and a clearer example of the HB is seen in Figure 2.53. When the authors made an explicit comparison between the homotopic set of three algorithms, i.e., HA^* , HRRT and HB, with their original counterparts, the following was observed. The homotopic variants returned an optimal solution typically faster than their original variants. Additionally, they generated an optimal path in all of the computed homotopy classes, and each variant took a certain fraction of time that it took to compute the optimal path. While this time is longer than the original counterparts'

performance for optimal paths, note that the original counterparts cannot accomplish this task at all. The [HB](#) offered the best performance among the homotopic variants, with a mere cost of 1.05 times the optimal path cost.

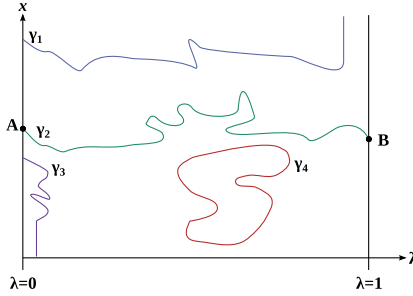


Figure 2.54: Different path solutions to the NAES problem [165]

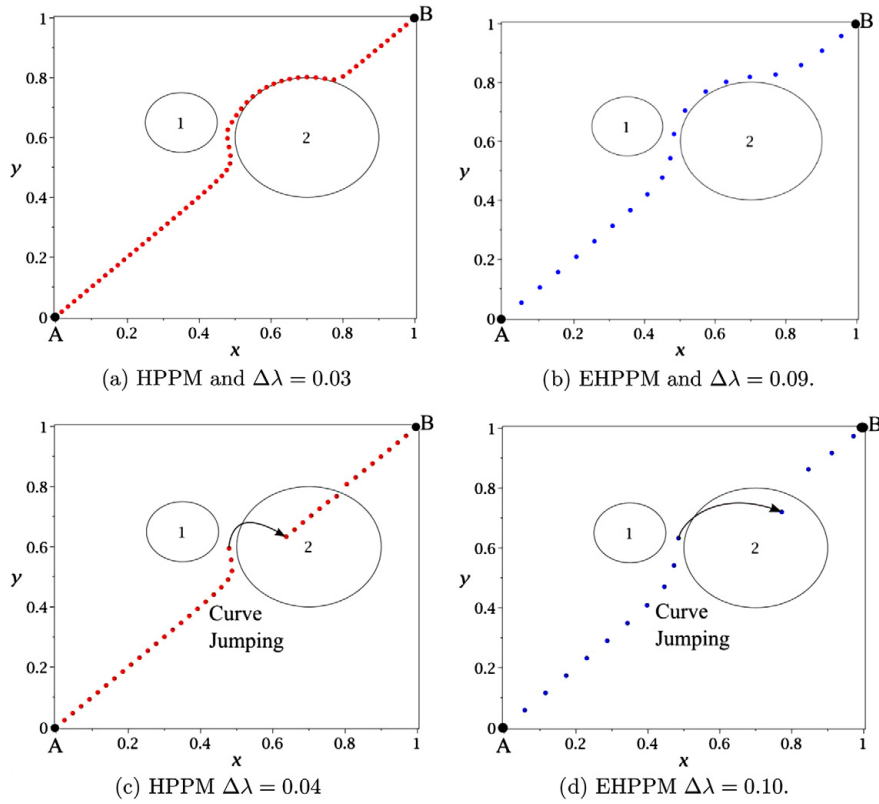


Figure 2.55: Illustration of curve jumping [165]

Yet another approach to discretize the homotopic [PPP](#) is by using [Non-linear Algebraic Equation System \(NAES\)](#), which can be solved iteratively for a valid path. Works such as [38, 39, 44, 165] use different variants of [NAES](#) to model the ability to determine a path using homotopic constraints. Since the environment is not discretized in the same manner as was seen with the usage of words in a homotopy class, techniques that rely on [NAES](#) are often referred to as [Homotopy Continuation Methods \(HCM\)](#). As the authors of [165]

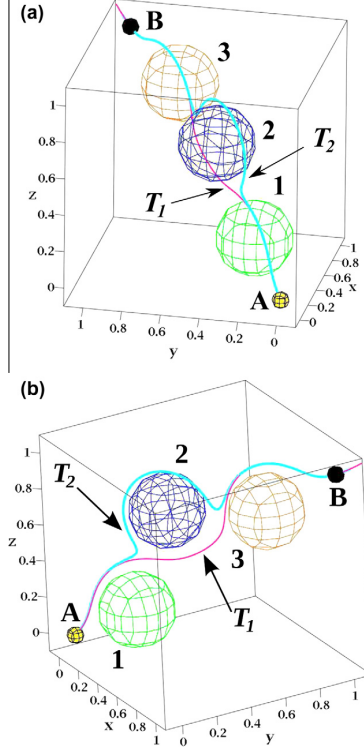


Figure 2.56: Paths seen in 3D using NAES [165]

explain, **HCMs** are a continuous transformation from one trivial problem (simple to solve) to the problem at hand (hard to solve). The general process of the **HCM** is briefly described first, before explaining how the **PPP** can be adapted to it. In the general **HCM** problem, the first step to formulate a homotopy is to establish a nonlinear equation that models the problem to be solved. It is defined as $f(x) = 0$, where $f : \mathcal{R}^n \rightarrow \mathcal{R}^n$, consisting of n total variables of the problem, denoted by x . A homotopy map may be written in this form as $H(f(x), \lambda) = 0$, where $H : \mathcal{R}^{n+1} \rightarrow \mathcal{R}^n$, and λ is the homotopy parameter. In this homotopy formulation, the value of the homotopy parameter λ determines whether a solution to the **NAES** has been found. When $\lambda = 0$, the solution to $H^{-1}(0)$, is either known or can be found using numerical methods, and when $\lambda = 1$, it implies that the solution/s to $H(f(x), 1) = f(x)$, can be found. The homotopy map therefore is a continuous function of $0 \leq \lambda \leq 1$. The resulting solution set of $H^{-1}(0)$, forms an initial homotopy path, that can then be traced by numerical continuation techniques. The next step involves setting a specific homotopy denoted by $g(x)$, which is combined with $H(f(x), \lambda)$ to augment the existing **NAES**. This new set of **NAES** can be solved by continuously varying λ , resulting in paths, as seen in Figure 2.54. Details about the numerical continuation method for **NAES** are provided in [165]. It is interesting to note that in Figure 2.54, trajectory γ_2 is a successful case from the starting point A ($\lambda = 0$) to solution B ($\lambda = 1$), while the remaining paths are cases of failure for the **HCM**. This matter of incorrect solutions, did not occur so far in the original and discretized forms of homotopic path planning, and must be noted for their reliability in path planning in dynamic environments.

Having obtained a perfunctory understanding of HCM, its application to path planning is now seen. The authors of [165] formulate all parts of the PPS as algebraic equations. Obstacles modeled as circular or rectangular structures are modeled as their respective algebraic counterparts. The PPS is planar and normalized so that the source and target locations can also be represented algebraically. The homotopic constraint is chosen in the form of a repellent behaviour, and an attraction towards two specific initial solutions. As the authors themselves note, this represents some sort of force fields around the real obstacles. The homotopy path cannot cross those fields, however it can pass close to them. This, along with the possibility of multiple solutions must be considered, in the efficacy of using NAES for homotopic path planning. Though the authors show that this method returns paths in static 2-d and 3-d environments, there exists a problem of curve jumping, when parameters of the NAES are non-optimal. An example of the paths generated in 2-d and an example of curve jumping is seen in Figure 2.55. Paths in 3-d are seen in Figure 2.56. Works, such as [38, 39, 44], also rely on NAES to solve the homotopic Path Planning Manifold (PPM). The individual algorithms vary either based on how the robot is represented and the nature of the initial paths used as initial best guesses. The authors of [44] use guiding parameters, such as auxiliary lines, that help keep the path deformation under control. The effect of auxiliary lines essentially adjusts the repulsion parameter, and thus adjusts the deformation of the path, as seen in Figure 2.57. The more auxiliary lines there are, the better the deformation of the path and its resulting quality, as seen in Figures 2.57 and 2.58. The caveat with such methods however, is that the algebraic equations representing the PPS and the robot appear to lie on a plane. In such a case then, there is not much of a vast advantage over original homotopic methods using complex analysis or discretized methods using words to represent homotopy classes.

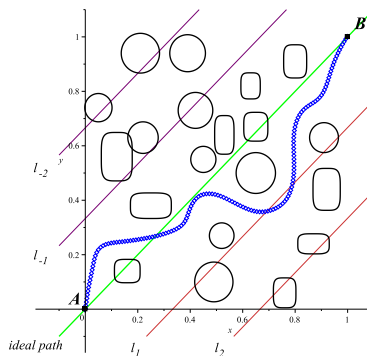


Figure 2.57: Impact of auxiliary lines [44]

One such successful example of discretized homotopic path planning in dynamic environments in a 2-d polygonal space is presented in [124]. The authors of [124] use a combination of cell decomposition and graph theory to effectively create representations of homotopy classes. Similar to earlier works in the field, once the homotopy classes have been found, mixed integer quadratic programming is used to determine an optimized trajectory within a preferred cell sequence associated to a specific homotopy class. The algorithm is seen to comprise the following general steps. First, the PPS is decomposed into convex polygons using well known convex decomposition algorithms. This is of use eventually, since adja-

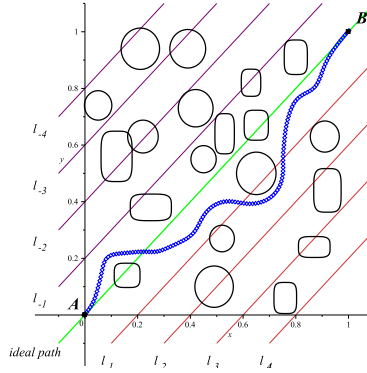


Figure 2.58: Impact of increased auxiliary lines [44]

ency relationships between decomposed polygons can be employed to represent adjacencies as a graph. Eventually, graph based methods can be used to search the adjacency graph for an optimal path connecting the source and target locations. Adjacencies in the form of adjacent cells or polygons, contain valuable information related to homotopy classes. In the second step, they refine the construction of convex polygons in the interest of creating a minimal vertex graph. This step creates partitions of the space in the form of vertices of convex polygons, such that new vertices are created only if they represent new homotopic information. The third step sees the construction of the adjacency graph, where adjacency of cells is recognized by virtue of shared edges, as opposed to vertices. This is useful to determine sequences of cells, that the path takes and thus determine the homotopy class. The fourth step involves determining what sequences of cells signify a homotopy and how that information can be extracted from the graph. The authors exploit the loop properties of the adjacency graph to determine homotopy classes, similar to the creation of words as seen in previous discretized homotopy algorithms.

Once they partition the infinite trajectories into finite sets of trajectories, they show that there exists a one-to-one correspondence of loop-less cell sequences with homotopy classes. In other words, a feasible trajectory is homotopic with any other feasible trajectory corresponding to the same loop-less sequence of cells on the graph, and not homotopic with all feasible trajectories in different sequences of cells in the graph. Figure 2.59 illustrates these properties. The most important and defining statement of the work is seen with the correlation between the cell sequence and homotopy class, which is as follows. If two feasible trajectories correspond to two different cell sequences, which are loop-less in the adjacency graph constructed through a convex decomposition with the minimal vertex set, they are not homotopic. With this information, the algorithm now proceeds to the final step, which now has the information about which homotopy class is required to be used as a constraint. The algorithm then uses mixed integer programming to optimize the trajectory segment in each convex cell of the homotopy class. The successful use of this algorithm in simulated dynamic environments on a model with non-linear vehicle dynamics can be seen in Figure 2.60.

The clear advantage of such discretized graph based approaches of homotopic path planning over NAES is the one-to-one correspondence between the cell sequence and the homotopy class. The issue of incorrect or multiple solutions across homotopy classes, experienced

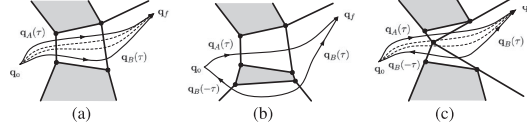


Figure 2.59: Different homotopy classes based on different cells [124]

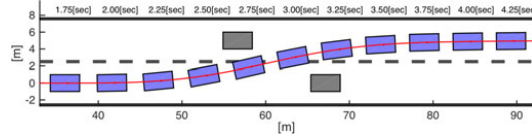


Figure 2.60: Paths to avoid dynamic obstacles [124]

as curve jumping in [NAES](#) is not encountered here. Authors of [\[180\]](#) also use a variant of such discrete representations of homotopy classes to guide their [SA](#) of choice, which in this case is the [RRT](#). They propose an algorithm called the Homotopic Aware [RRT*](#), or [HARRT*](#). As seen in previous similar works, the authors of [\[180\]](#) partition their [PPS](#) based on variants of partitioning algorithms used previously, such as those based on reference frames, as seen in [\[56\]](#). Consequently, they use the canonical representation for every homotopy class, similar to previously described works. The main contribution of this study is the use of homotopic groups to drive a biased exploration using the [RRT*](#). The authors propose an algorithm called the Recursively Embedded Palindrome (REP) to trim newly generated strings or paths, to their minimum representative lengths or sequence. This allowed the authors to identify homotopic equivalence of path segments or branches generated thus far, and allow the expansion to take place in restricted homotopy classes. The authors also show its impact on both the single and bi-directional [RRT](#), as seen in [Figure 2.61](#). The authors do recognize the challenge of combining [RRT](#) with the REP trim algorithm, since there is a possibility that potentially optimal paths in a homotopy class may be prematurely discouraged from being explored. While it was resolved by using a bi-directional [RRT](#), the authors point out that there may be other configurations of environments that potentially still have the same problem. [Figure 2.62](#) shows the optimal solutions returned by [HARRT*](#) for six different homotopy classes in another 2-dimensional environment with planar obstacles. Consistent improvements to graph construction and representation of homotopy classes can be seen through works, such as [\[70, 71\]](#). The study in [\[71\]](#) shows the direction of improvement

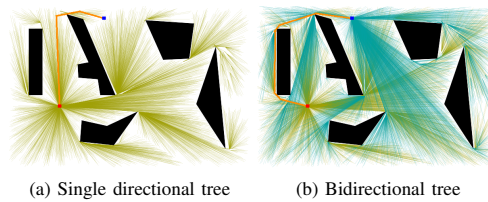


Figure 2.61: Impact of Recursively Embedded Palindrome strings in single and bi-directional [RRT](#) [180]

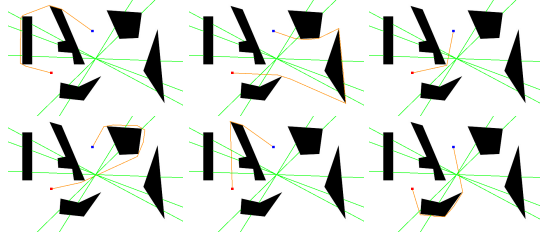


Figure 2.62: Paths in six different homotopy classes obtained by HARRT* [180]

catering to dynamic environments, where road map construction and path generation need better efficiency, in terms of the next sampled point or direction of growth. In environments with narrow corridors, discovering all such narrow spaces and redundantly connecting them with the road map plays an important role towards the optimality and completeness of the solution. The objective is that the road map must consist of the least number of vertices and edges (for minimizing road map construction and query time), while leading to discovery of all possible homotopic groups of paths between all possible sources and goals. Discovery of all homotopic groups guarantees a path for every pair of valid source and goal locations, or in simple words, the approach is complete [71].

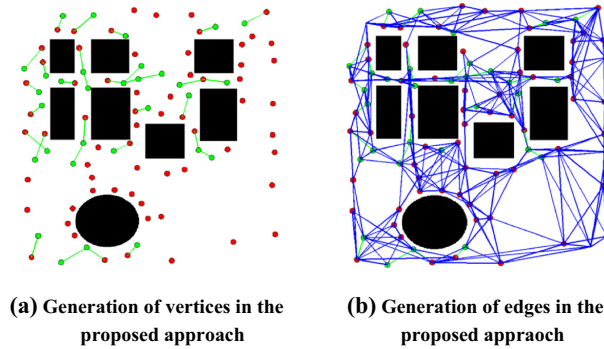


Figure 2.63: Generation of vertices and edges [70]

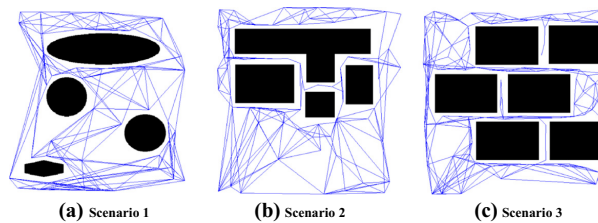


Figure 2.64: Different environments and their road maps [70]

To achieve the objective, [71] thus focuses on different ways to manipulate graphs to reduce redundant cycles and yet, provide better paths in narrow areas. The author of [71] therefore proposes a better sampling strategy to enable a homotopy conscious road map to be

constructed. An extension of this work is seen in [70], which better outlines the contributions in the area of homotopy conscious road maps.

The work in [70] focuses on devising a road map generation method using PRM, such that most homotopic groups are represented in small computation times. The key difference between this approach and other homotopic variants of sampling techniques, is that this approach changes the sampling strategy to detect homotopy classes in an unstructured environment, and possibly unknown as well. Additionally, the work also proposes a difference in the kind of local road maps constructed depending on the nature of the region being sampled. The local road map facilitates redundant connectivity to the rest of the road map, which is an advantage, because it can simultaneously cater to narrow corridors, congested environments and otherwise difficult areas. An example of such a local roadmap can be seen in Figure 2.63. In order to connect these sampled points, an edge connection strategy is formulated. This strategy discounts the connectivity of an edge based on the sizes of the road maps that it is directly or indirectly connected to. It saves computational time while simultaneously maintaining nearly the same connectivity of the road map as a k -connected PRM.

The design of the algorithm aims to suit a variety of cases including narrow corridors, open spaces, multiple pathways between obstacles of multiple widths and narrow corridors amidst wide open spaces. Figure 2.63 shows the different environments and their associated road maps. Yet another work that attempts to combine efficient sampling with homotopy is [80]. A combination of modified sample generation and discard is used to generate points of the Volumetric Tree* (VT*). Hyperspheres around the robot are used to generate new samples, from which vertices of VT* are created. The authors of [80] claim that the points generated in the hypersphere are homotopic to other points in the same hypersphere, and use this as a justification of generating sufficient points in the VT*. They use the Dynamic Shortest Path Tree to compute paths and also perform homotopy exploration. One must note that [80] does not establish homotopy classes as established in previous works on the subject. Since random sampling and dropout techniques do not explicitly lead to identification of homotopy classes in the entirety of the PPS, previously described works may serve as better references on the matter.

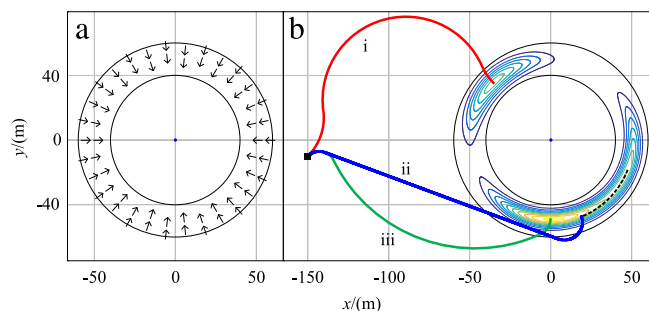


Figure 2.65: Change of Dubin's paths directed towards different locations on an annulus [179]

Authors of [179] show that integrating homotopy classes with path planning is indeed a structured process. They propose Structured Homotopy (SH) based path planning for a

Dubin's vehicle model, which is assumed to have an upper bound on the linear and angular velocities. Such bounds force the Dubin's model to produce paths that are a combination of straight lines and/or circular segments, each of whose length and curvature is bounded by the linear and angular velocities. Yet another work that combines homotopy and Dubin's curves can be seen in [105]. These curves are referred to as the Dubin's curves, and are the kind of paths being used and generated in [179]. Using these curves, a homotopy class called the curvature bounded homotopy class is constructed. This homotopy class constitutes an envelope of the path planning region, where segments of the Dubin's curves can be continuously deformed into other curvature bounded Dubin's path segments. The aim of this is to have a dedicated region in which the initial guess of the Dubin's curve path can be optimized for path length, while still being homotopically the same. In order to do so, the authors choose a specific set of Dubin's curves to work with, for the following reason. The path deformation of such curves when discretized, is a recursive function of the intermediate segments' end points. Called the Structured Homotopy based path planning, this is what finds the solution to a new path in the homotopy class with desired length characteristics. To demonstrate the effectiveness of SH based planning with Dubin's curves, the authors simulate a case where the target locations lie on a manifold, which in this case is that of the annulus. This is of use, since an annulus' vector bundle induces different kinds of Dubin's curves (due to change in curvature), as opposed to a plane. This change of paths is clearly seen in Figure 2.65 with the different colored paths indicating different target locations on the annulus.

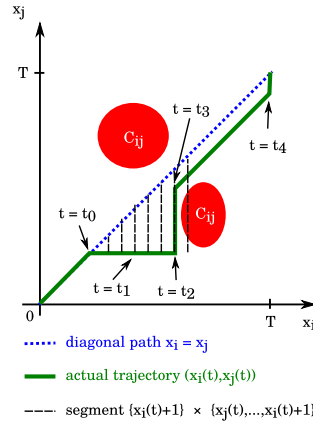


Figure 2.66: Diagonal path guides steering away from constraints [53]

In [53], homotopy is used for multi-robot path planning, such that no deadlock is encountered when the multiple robots track their paths. They do so by accomplishing two goals. First they show that coordinated trajectories computed by an arbitrary multi-robot motion planner can be used to derive a maximal set of homotopic solutions forming a homotopy class. This was obtained by translating the problem into the coordination space. This set of solutions is composed of all paths in the coordination space that are homotopic to an initial path called the diagonal path. This diagonal path is a path in the coordination space that steers clear of any constraints, as seen in Figure 2.66. Viewed from a different perspective, the diagonal path serves as a different method to partition the coordination

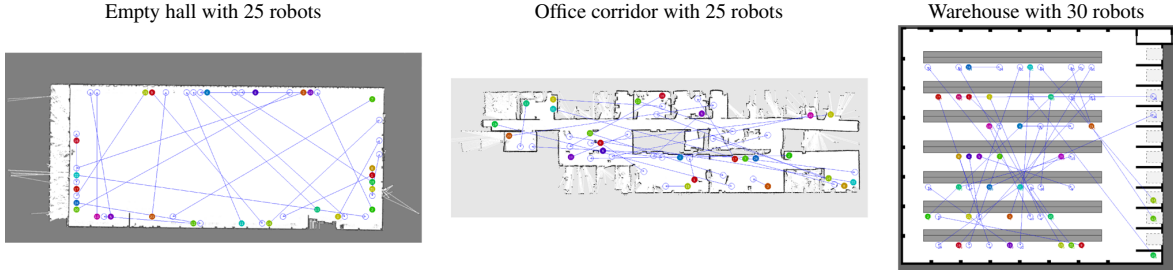


Figure 2.67: Deadlock free paths with multiple robots in different settings [53]

space. Then, the authors derive a collision and deadlock free region in the coordination space, which is also comprised of the maximal set of solutions in the homotopy class. The second contribution involves devising a controller that the system remains in the homotopy class once it has entered it. That is, the homotopy class is formulated as an invariant set. Successful results from their experimental validation are seen with several robots, as seen in Figure 2.66. Though multi-robot coordination is out of the scope of this thesis, this work is yet another example of how homotopy classes can be discretized, and that they may be robust to perturbations or noise. Original HBMs and their discretized variants may be seen to have an advantage over this method, due to their robustness towards 3-d spaces. The method outlined here, is catered to planar 2-d environments, which are realistic for office like environments, but may not be truly robust as a generic homotopic path planner.

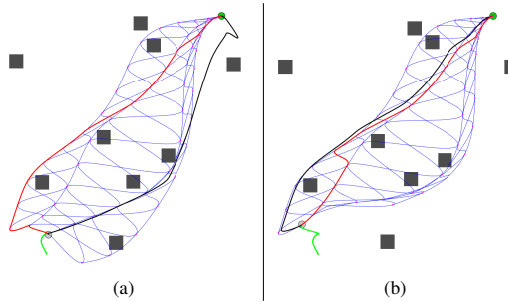


Figure 2.68: Illustration of Planning Online by Switching Homotopies (POSH) [86]

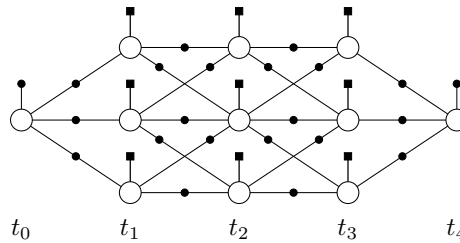


Figure 2.69: An example graph construction with three chains and interconnections [86]

The authors of [86] offer an interesting perspective on homotopy based path planning,

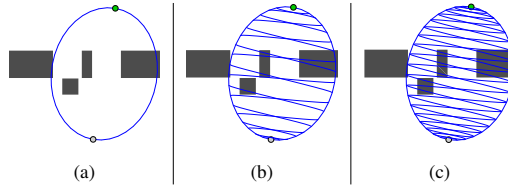


Figure 2.70: Different desired levels of inter-connectivity [86]

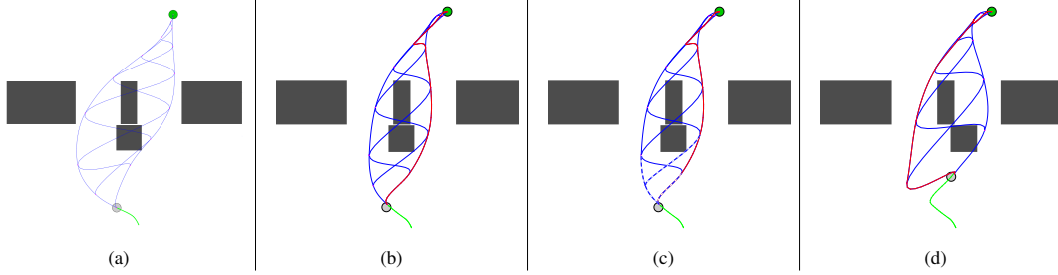


Figure 2.71: In (a), the gray circle and green circle are the robot and goal states. In (b), A* is used to find a low-cost trajectory (red) on the graph. In (c), edges that are no-longer viable (dashed blue lines) after the first movement are pruned. These can include the collision-free segment currently chosen as the direction of movement and segments that currently or potentially intersect with obstacles. Finally in (d), as the robot executes its chosen direction of movement, the environment changes with the lower square moving further along the positive x-axis. Post re-optimization, POSH finds a trajectory in a new homotopy class (red). [86]

that one could consider as an application for dynamic environments. The main contribution of the paper is a switch from one homotopy class to another, should one class no longer be favorable to traverse in. They do so by proposing an algorithm called the Planning Online by Switching Homotopies (POSH). An illustration of this is seen in Figure 2.68. POSH builds on inference based trajectory optimization techniques to plan online and dynamically switch homotopy classes. POSH can extract information about multiple homotopy classes, and in light of updated information of the dynamic environment, the appropriate homotopy class can be selected as the new optimal class. This prevents the algorithm from stagnating at a class that recently became sub-optimal due to a new constraint. The authors use Gaussian Process Motion Planning (GPMP) to enable a probabilistic update to the representation of the environment in the form of a graph. Instead of retaining the same optimized trajectory from the initial time step for execution as does GPMP, POSH maintains and updates the entire graph at every time step. Specifically, at any time step the graph is pruned to remove unreachable (in time) states. The graph is then re-optimized with the changes in the environment, and is now equipped to find the new optimal trajectory. The details of GPMP may be seen in [86]. They are summarized briefly here. GPMP constructs a factor graph with multiple initial trajectories (chains), each with its own set of path segments, called support states. Adjacent chains have interconnected support states with a prior factor and

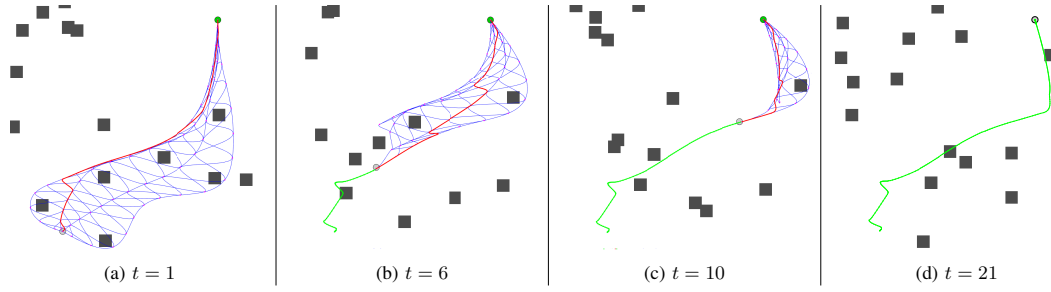


Figure 2.72: Another illustration of POSH [86]

interpolated factors between two states from two different chains. This allows the robot to switch between chains. An example graph construction with three chains and interconnections is shown in Figure 2.69. Note that such interconnecting support states and factors are what represent the multiple paths in the same homotopy class and in different classes, without which deliberate class switching is not possible. The *H-signature* is used as the unique identifier for the homotopy class and is applied to identify the frequency at which the robot changes the homotopy class of its trajectory. The POSH algorithm consists of the following steps. First, using the signed distance field, the obstacle pose is initialized. Next, by computing a factor graph, one can determine the extent to which the nodes are required to be interconnected for better switching or faster solution times; refer to Figure 2.70. In the third step, A* is employed to generate the initial best cost path, which will be followed until new information about the environment changes the graph. The fourth step executes the motion planned thus far, and performs an update of the support nodes using the signed distance field. This will help prune useless support states or nodes that are no longer valid. On this graph, A* is then used again to produce the next best path, and the algorithm is recursive, until the goal is reached. Figure 2.71 illustrates the algorithm pictorially, while Figure 2.72 shows the results. The latter shows the response of the algorithm in the dynamic forest environment consisting of numerous objects exhibiting random 2D motion. The dynamic forest shown in Figure 2.68 is employed to benchmark the ability of the algorithms to contend with a challenging re-planning problem [86].

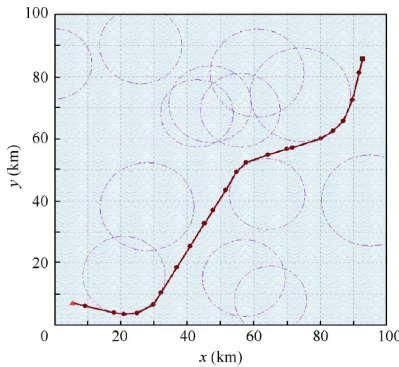


Figure 2.73: Path as a result of sampling and pruning candidate paths [104]

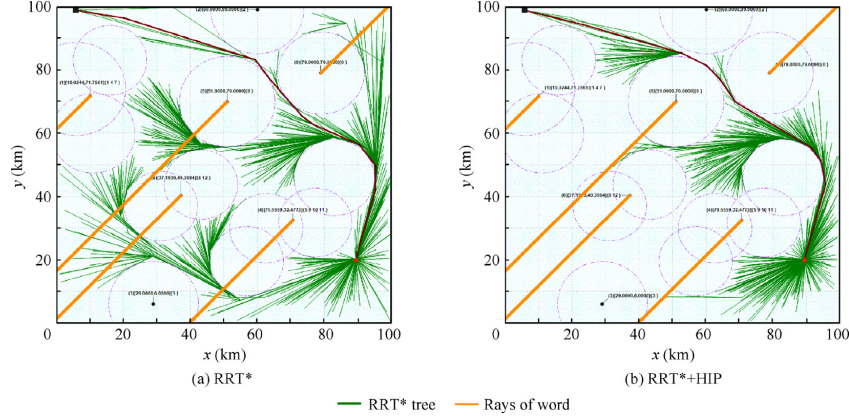


Figure 2.74: Additional pruning through HIP in conjunction with RRT* [104]

The authors of [104] also use word based homotopic path planning. They propose the Homotopy Informed Preprocessor (HIP) algorithm to find the subset of the PPS containing the optimal path and prune those path candidates that fall outside the subset. The three steps of the HIP are as follows. The first step is to use homotopic invariants to obtain disjoint homotopy classes as a partition. The second step uses the Multiple Jump Point Search (Multiple JPS) algorithm to obtain all shortest paths locally in the discrete grid representation of the PPS. From this, they are able to correlate those homotopy classes that produce optimal paths, and these classes are called the optimal collection. The remaining non-optimal classes are filtered during the pruning process. The third step constructs words of the homotopy classes of the optimal collection. The word requirement forms an additional pruning criterion in rejecting those path candidates that do not satisfy the word requirement. This work’s primary contribution is in the qualifier used for pruning and sampling. Figure 2.73 illustrates results from this algorithm. Figure 2.74 shows the use of the additional pruning through HIP in conjunction with RRT*.

One of the most recent works in the field of path planning using manifold theory is seen in [21]. The authors of [21] solve the problem of obstacle avoidance by interpolating points on the PPS, which is assumed to be either a Riemannian manifold or a space described by Lie groups. While the method does not explicitly deal with generating paths in homotopy classes, it focuses on admitting only such variations to path segments that comply with the local tangent space. Therefore, the deformations agree with the underlying manifold, and are seen to be $\mathcal{C}^1$, which is not just continuous as required for homotopic classes, but also once continuously differentiable. In the absence of any obstacles, the resulting path is shown to be a geodesic, which satisfies the requirement of the shortest path on a manifold. The quality of deformation of paths is comparable to that of deformations experienced by paths in a homotopic class, since the resulting path still respects the topology of the underlying manifold. Recall from [90] that a homotopy group was induced in the ambient space of the PPS; however, that is not the only homotopy class that can be deduced from the situation. Homology groups that are innate to the surface may also be considered, and these are irrespective of the ambient space, as seen in Figure 2.75. This implies that paths or cycles that lie in these groups may be transformed to other non-canonical cycles in the same

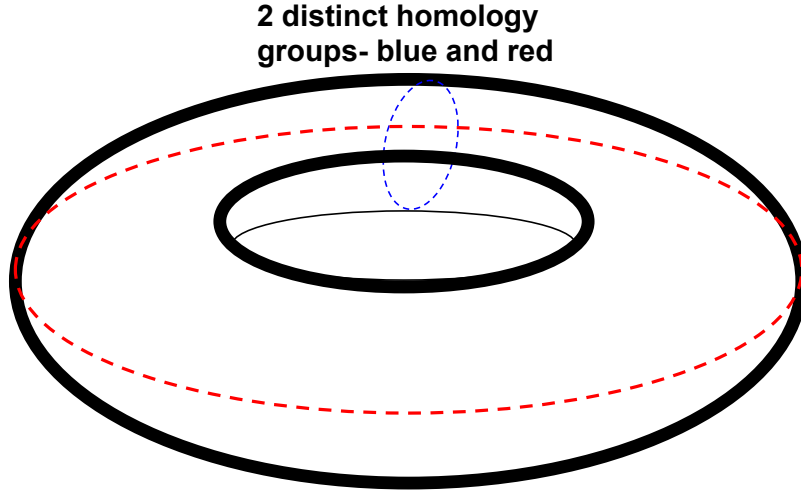


Figure 2.75: The two distinct homology groups of a torus

groups. The resulting cycles or paths still lie on the surface of interest. This digression from the content in [21], is to show that the deformation of path segments therein ensures that the resulting path segment lies on the tangent space of the manifold, and thus is as important to consider as are other works on homotopic classes of paths. The work in [21] is now briefly explained. The premise of the general problem being solved here, is the dynamic interpolation of path segments on the tangent space of a Riemannian manifold. Therefore, it is assumed that a standard metric on the tangent space is defined, and the path segments conform to being $\mathcal{C}^1$ piecewise smooth curves. The path segments obeying the boundary conditions on the tangent space of the manifold are identified as a class of curves, which collectively admits a manifold structure, called the admissible manifold ω . The deformation or interpolation of these path segments are explained with the help of the curvature tensor defined as follows. On a given Riemannian manifold, the influence of a vector field on a curve is given by the covariant derivative of that vector field on the curve. When there are several vector fields acting on the curve, the combined influence of all these vector fields on the curve is given by the curvature tensor, which is a function of all the covariant derivatives of the vector fields and their lie brackets.

For the class of curves in ω , the authors of [21] introduce the notion of a $\mathcal{C}^1$ piecewise smooth one-parameter admissible variation α of a curve $x \in \omega$. The admissible variation α is characterized infinitesimally by the $\mathcal{C}^1$ piecewise smooth vector field along the curve x , called the variational vector field. Due to the boundary conditions imposed, the variational vector field belongs to the tangent space of the curves, as originally defined. This one-parameter variation satisfies a relationship involving the curvature tensor. Then, the authors define a functional J to be minimized to solve the problem defined. The functional J is given by a weighted combination of the velocity and covariant acceleration of the curve x regulated by a parameter ρ , together with an artificial potential function V used to avoid the obstacle. The obstacle is described by a region in the manifold bounded by S , a regular level set of some

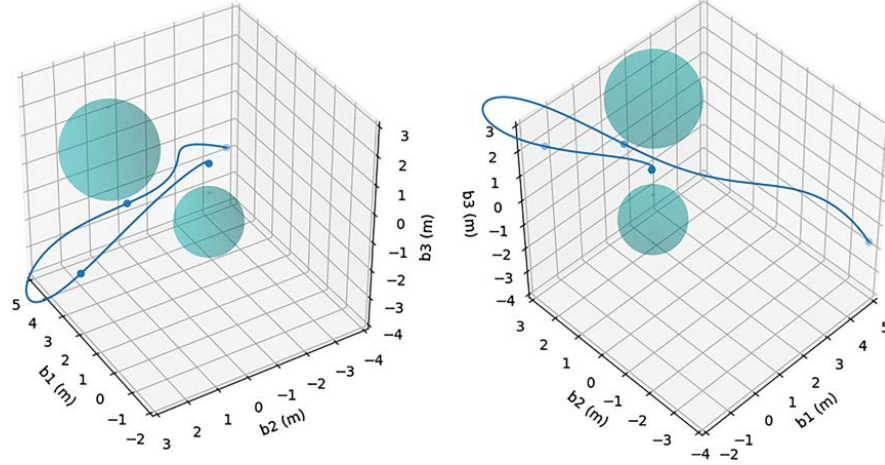


Figure 2.76: Illustration of admissible variation based path deformation in 3-dimensions [21]

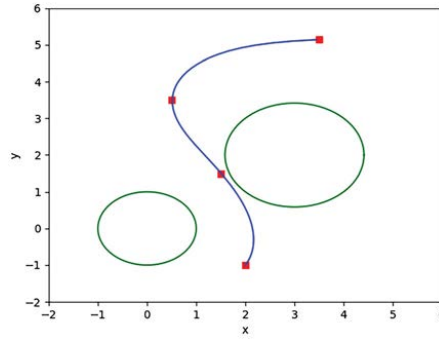


Figure 2.77: Illustration of admissible variation based path deformation in 2-dimensions [21]

scalar valued smooth function f . Examples are circles, spheres and ellipsoids. The potential function V is an artificial smooth (or at least $\mathcal{C}^2$) function aimed to produce a fictitious force inducing a repulsion from S . Now that all aspects of the functional J are defined, it is minimized on the admissible manifold ω to produce path segments. Illustrations of the method are seen in Figure 2.76 and Figure 2.77. The interesting aspect of this work is to observe the nature of the path, if the notion of an obstacle was eliminated from the functional J . That is, when the repulsive force V becomes 0, the resulting path is a geodesic. This shows that using structured approaches to path modifications guarantee a return to definitions of the underlying space, which is of consequence when the PPS is non-Euclidean.

2.4 Summary

The PPP is diverse, both in the kind of environments encountered and how they can be represented. This chapter addressed these two diversities by focusing on static versus dynamic environments and, the different algorithms/mathematical tools used to represent and manipulate the PPS. A brief summary of the chapter is provided, where the PPP is first

discussed in the perspective of SCs, followed by that of DCs.

A PPS can be viewed either as a set of points or a characterized surface. The former facilitates the use of SAs, like PRMs, RRTs and CDGT methods. Alternatively or in conjunction, OAs such as APFs and EAs, can be used to solve the path planning problem. Such methods are robust to change in robot models and do not require an explicit description of $\mathcal{P}^{\text{constraint}}$. Since sampling enables collection of data that represents the PPS, the frequency at which data is collected needs to be analyzed. A higher sampling frequency is assumed to be better suited for environments with many distinct or dynamic features. However, it is computationally expensive for PPSs that do not have such features. Conversely, an environment that requires a higher sampling rates may suffer from infrequent sampling. Therefore, sampling frequency is an important, but volatile parameter whose change affects the true representation of the PPS in SAs and OAs. Map resolution or the total number of points required to represent a smallest area of the map can be an independent parameter or a direct consequence/factor of sampling frequency.

In either case, map resolution behaves similarly to sampling frequency. That is, cluttered areas may require higher map resolution as opposed to regularly spaced areas. However, map resolution is typically fixed, and a dynamic change in resolution to account for a changing environment is a complex predict-behave process, that needs tuning. On the note of cluttered environments, the sampling strategy also plays a major role in how well the environment is perceived. For example uniform sampling in a PPS with several narrow corridors, may completely avoid sampling the entirety of the narrow space, thus building an inaccurate representation of the cluttered space. Therefore, a non-uniform sampling strategy needs to be adopted as the PPS changes to a cluttered setting. Additionally, other parameters that govern the constraint avoidance scheme may require tuning, especially when avoiding mishaps native to the optimization algorithm, such as local minima. The consequences listed above are viewed as disadvantages, since their ability to represent the environment with constraints needs intervention in the form of parameter tuning.

These disadvantages are characteristic to the absence of a structured representation of the PPS, which was handled in the second form of representation, i.e., as a manifold. Global representations of realistic PPSs, i.e., representing PPSs as PPMs, are seldom easy to create. However, the properties that such formulations offer include continuity, smoothness, neighbourhood relationships, metrics and equivalence classes. Such properties can help determine paths that follow the underlying curvature of the space and aid in producing better quality paths that closely resemble the true nature of the path in the original space. Using this as a foundation, one may then progress towards discretizing the representation of the PPS into a PPM, and formulating the constraint-free path as a geodesic on the discretized PPM. Thus, the definition of the PPP involves finding a constraint-free path (geodesic or a segment of a geodesic) from $\mathbf{p}^{\text{start}}$ to $\mathbf{p}^{\text{goal}}$ on the PPM.

Viewing the PPP in the context of avoiding DCs in the PPS, showed that there are three main factors to consider, which are as follows: defining DCs, characterizing the response of the algorithm to DCs and the kinds of mathematical tools used to define the chosen response. DCs could be moving objects or the sudden presence/absence of stationary objects. This means that the knowledge of such constraints may be evanescent, thus characterizing

the nature of the response provided by the algorithm. Fleeting information about dynamically appearing constraints may require the algorithm to reactively provide a local, remedial solution. This is opposed to global, deliberative approaches, that preemptively steer clear of constraints (assuming their knowledge is at least partially known apriori) and prioritize navigating towards the destination. Global, deliberative approaches and local, reactive algorithms typically induce slightly different path variations and maneuvers. The former aims to steadily and seamlessly modify the path such that the potential constraint is not encountered at close quarters, whereas the latter aims to generate immediate collision-free movements, such as a blatant turning away or termination of movement. While both kinds of algorithms modify the path either as a whole or in part, the operation involved is that of constructing a deviation in the original, pre-constraint path. This deviation is what divides the approaches of the PPP with DCs, into MBMs and HBMs.

In order to produce the desired modification of the path, MBMs may rely on all or some of the following: models of the robot, environment and expected constraints. Some examples of MBMs include a combination and/or variants of MPC (Dubin's curves, VOSs), OAs (diffusion, conductivity and APF based methods), SAs (reactive RRTs, PRMs) and CDGT (tangent graphs). The primary advantage of producing path deviations especially suited to the robot model, also means that the algorithm is not model independent. Robustness to different environments and robot models is seen as a desirable feature, since there is minimal modification required to use the algorithm in a new PPP. The other demerit experienced by MBMs is that they inherit the same disadvantages as were originally experienced in the context of SCs. So, issues of sampling affecting the representation of the environment and encounters with non-optimal solutions in the form of oscillations and local minima, still arise. The alternative therefore, is a generic solution that minimizes the pitfalls seen with MBMs, and that group of solutions is called HBMs. HBMs are able to distinguish categories of paths based on how many constraints the path encloses. The knowledge not only allows identifying desired homotopy classes within which ideal paths should lie, but also the deformations that will allow the path to fall into the desired class. When combined with graph traversal based algorithms or used on their own, the original and discretized HBMs are seen to effortlessly identify homotopy classes in environments with several obstacles. Their robustness to noise (although still a function of sensor data), independence of robot model and avoidance of ambiguous solutions (such as local minima or oscillations) serve as important advantages over MBMs.

A high-level comparison of the different types of path planning algorithms, including the CFDMPP algorithm proposed in this thesis, is summarized in Table 2.1 and Table 2.2. The former compares them in the perspective of SCs and the latter focuses on DCs. More details about the CFDMPP algorithm are covered in the following chapters.

Table 2.1: Comparison of path planning algorithms in the context of SCs

Characteristic	Algorithms			
	SA	OA	Manifold-Based	CFDMPP
Requires environment map	No	No	Yes	Yes
Limitations on dof	No	No	Yes	Yes
Single/multiple query	Both	Single	Both	Both
PPS representation affected by sampling	Yes	Yes	No	No
Path quality affected by sampling strategy	Yes	Yes	No	No
Robust to stationary and mobile robots	Yes	Yes	Yes	Yes
Completeness	Probabilistically complete	Probabilistically complete	Complete	Complete (resolution-complete after discretization)
Explicit path evaluation for violation of constraints	Required iteratively	Required iteratively	Not required	Not required
Vulnerable to local minima	No	Yes	No	No
Shortest path guarantee	No. A smoothing function is needed to shorten generated path.	No. A smoothing function is needed to shorten generated path.	Yes if PPS is parametrized, or if it is charted and an atlas exists.	Yes (shortest discretized path)
Flexibility to work in subsets of $\mathcal{C}^{\text{free}}$ or $\mathcal{W}^{\text{free}}$	No (works only in the sampled space and its subsets)	No (works only in the sampled space and its subsets)	Yes	Yes
Equivalence of working in $\mathcal{C}^{\text{free}}$ and $\mathcal{W}^{\text{free}}$	No	No	Yes	Yes

Table 2.2: Comparison of path planning algorithms in the context of DCs

Characteristic	Algorithms		
	MBM	HBM	CFDMPP
Requires robot/constraint models	Yes	No	No
Single/multiple query	Both	Both	Both
Anytime algorithm	Possibly; depends on the variant chosen.	Yes	Yes
Robust to noise	No	Yes	Yes
Robust to stationary and mobile robots	Yes	Yes	Yes
Completeness	Probabilistically complete	Complete (resolution-complete after discretization)	Complete (resolution-complete after discretization)
Vulnerable to local minima	Yes	No	No
Multiple possible alternative solutions	No	Yes	Yes
Manifold based path planning	No	No	Yes

Chapter 3

Methodology

Chapter 2 provided three main approaches to solve the general PPP which are SA, OA and the use of manifold theory. This chapter lays down the foundation of the proposed CFDMPP method to solve the PPP. The CFDMPP provides a solution to the generalized PPP, i.e. with static and dynamic constraints, in a single framework that encompasses manifold theory. The working mechanism of CFDMPP can be briefly explained as follows. Once a PPS is chosen for the application at hand, a subset of the PPS is formulated into $\mathcal{P}^{\text{free}}$ using a global parametrization. Following this step, a discretization procedure is performed and the discretized constraint-free manifold is queried for paths. The algorithm performs multi-query, constraint-free path planning. It combines global parametrization from Section 2.2.1 and the creation of discrete charts, from Section 2.2.2, to pave the manifold. Exploiting manifold theory in robotic path planning is motivated by the advantages that it offers. The main advantages inherited by CFDMPP include the following:

1. A path generated using CFDMPP is guaranteed to be constraint-free. As such, the path never has to be evaluated for constraint violation against constraints modelled out of the Constraint-Free Manifold (CFM).
2. CFDMPP natively accommodates both static and dynamic constraints via homotopic path planning. The constraints can include static and dynamic obstacles, singularities and artificial constraint regions, such as joint limits or workspace no-go zones.

Having understood a brief breakdown of CFDMPP and its primary motivation, the remaining sections establish the notation and definitions required to formulate the problem and lay down the proposed CFDMPP algorithm in Section 3.1 and Section 3.2, respectively.

3.1 Notations and Definitions

This section provides an introduction to concepts used in topology and elementary differential geometry that will be used to define the proposed solution CFDMPP, in section 3.2. CFDMPP uses the manifold approach to solve the PPP. To help define the points that make up the PPS and eventually the PPM, one uses a tuple.

Definition 3.1 (Tuple). An n -tuple is an ordered set of n elements, where $n \geq 0$. It is represented as such: $\mathbf{z} = \{z_1, z_2, \dots, z_n\}$, where z_i is the i^{th} element of the n -tuple.

Tuples are the fundamental forms of representation of points and vectors in space. Vectors in the generalized n -dimensional PPS thus take the form of an n -tuple. Spaces can then be defined as union of all such tuples, as can be seen below for $\mathcal{C}$, $\mathcal{W}$ and $\mathcal{P}$.

Definition 3.2 (Configuration Space). A system's generalized coordinates is an n -tuple, with every element representing a DOF that can be actuated or controlled. This n -dimensional configuration vector $\mathbf{q}$ of a system is represented as

$$\mathbf{q} = [q_1 \quad q_2 \quad \cdots \quad q_n]^T; q_i \in [q_i^{\min}, q_i^{\max}] = q_i^{\text{span}}$$

Then, the n -dimensional configuration space $\mathcal{C}$ of that system is the set of all configurations that the system can assume. It can be represented as

$$\mathcal{C} = \mathbf{q}^{\text{all}} = \prod_{i=1:n} q_i^{\text{span}} \quad (3.1)$$

Definition 3.3 (Workspace). A system's workspace $\mathcal{W}$ represents that portion of the environment, that the system's EE can access [154]. Consider an m -tuple that represents a point in $\mathcal{W}$ as

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_m]^T; x_i \in [x_i^{\min}, x_i^{\max}] = x_i^{\text{span}}$$

Then, the m -dimensional workspace $\mathcal{W}$ of that system is given by

$$\mathcal{W} = \mathbf{x}^{\text{all}} = \prod_{i=1}^m x_i^{\text{span}} \quad (3.2)$$

An example of $\mathcal{C}$ and $\mathcal{W}$ of the 2LPM are seen in Figures 2.22 and 2.23, respectively. Generally, the mapping from $\mathbf{q}$ to $\mathbf{x}$, and thus from $\mathcal{C}$ to $\mathcal{W}$, is given by (3.3) and (3.4), respectively. The inverse mappings (if exist) are defined in (3.5) and (3.6).

$$\mathbf{x} = f^{\text{wd}}(\mathbf{q}) \quad (3.3)$$

$$\mathcal{W} = \mathbf{x}^{\text{all}} = f^{\text{wd}}(\mathbf{q}^{\text{all}}) \quad (3.4)$$

$$\mathbf{q} = f^{\text{inv}}(\mathbf{x}) = f^{\text{wd}^{-1}}(\mathbf{x}) \quad (3.5)$$

$$\mathcal{C} = \mathbf{q}^{\text{all}} = f^{\text{inv}}(\mathbf{x}^{\text{all}}) = f^{\text{wd}^{-1}}(\mathbf{x}^{\text{all}}) \quad (3.6)$$

In general, the PPS can take several forms, such as the configuration space $\mathcal{C}$, the workspace $\mathcal{W}$, the joint velocity space, the force space, and so forth. To that end, we denote the set of all points in the PPS by $\mathcal{P}$. A point $\mathbf{p}$ in l -dimensional $\mathcal{P}$ is denoted by an l -tuple

$$\mathbf{p} = [p_1 \quad p_2 \quad \cdots \quad p_l]^T \quad p_i \in [p_i^{\min}, p_i^{\max}] = p_i^{\text{span}} \quad (3.7)$$

Therefore any PPS $\mathcal{P}$ can be expressed as:

$$\mathcal{P} = \mathbf{p}^{all} = \prod_{i=1}^l p_i^{span} \quad (3.8)$$

Equation (3.8) can also be used to express any other region associated with path planning.

Having understood the relationship between tuples and the space that their union can represent, one is now interested in assessing the relationship between and amongst such points in the space. Such relationships can be observed by applying certain rules of compatibility in that space, and a topological space is one such example. Informally, a topological space is a set of points on which rules of compatibility are imposed. Typically, rules are applied to subsets of the original space. For the purposes of this work, an intuitive understanding of topology aids in defining a topological manifold, which will be used later to define the PPS.

Definition 3.4 (Topological space). *A topology on a set $\mathcal{S}$ is a collection $\mathcal{T}$ of subsets containing both the empty set $\emptyset$ and the set $\mathcal{S}$ such that $\mathcal{T}$ is closed under arbitrary unions and finite intersections. In other words, if $\mathcal{U}_\alpha \in \mathcal{T}$ for all α in an index set $\mathcal{A}$, then $\bigcup_{\alpha \in \mathcal{A}} \mathcal{U}_\alpha \in \mathcal{T}$ and if $\mathcal{U}_1, \dots, \mathcal{U}_n \in \mathcal{T}$, then $\bigcap_{i=1}^n \mathcal{U}_i \in \mathcal{T}$. The elements of $\mathcal{T}$ are called open sets and the pair $(\mathcal{S}, \mathcal{T})$ is called a topological space [163].*

The topology (set of compatibility rules) as described in Definition 3.4 is the general definition of topology. It imposes such a relationship that an arbitrary union and intersection of subsets must still belong in the original space. Topology can be defined using different tools, such as neighbourhoods, open sets and closed sets, thus resulting in different kinds of topologies. The following example aims to provide a simple understanding of a topological space, before proceeding to understand the topology that will be used henceforth, called the standard topology.

Example 3.1 (Topological space). *Consider the set $\mathcal{S} = \{1, 2, 3\}$ and suppose that the chosen topology $\mathcal{T} = \{\emptyset, \{1, 2, 3\}\}$. Then, all unions and intersections are closed in $\mathcal{T}$, thus making $(\mathcal{S}, \mathcal{T})$ a topological space. Contrarily, $\mathcal{T} = \{\emptyset, \{1, 2, 3\}, \{1, 3\}, \{1, 2\}\}$ is not a topological space, since $\{1, 2\} \cap \{1, 3\} = \{1\} \notin \mathcal{T}$, for example.*

While several variants of the topology to be imposed exist, the most commonly used one is the standard topology. In this topology, a set $\mathcal{U}$ is open in $\mathbb{R}^n$ if and only if $\forall p \in \mathcal{U}$, there is an open ball $B(p, \epsilon)$, with center p and radius ϵ , that is contained in $\mathcal{U}$. Generally, point-set topology is concerned with properties that remain invariant under homeomorphisms (continuous maps having continuous inverses) [163]. This invariance in the PPS is eventually what will help in planning constraint-free paths in a space whose constraint-free nature remains invariant when transformed to another, similarly equipped topological space.

As a topological space, one can see that $\mathcal{P}$ is a union of constrained and constraint-free subsets

$$\mathcal{P} = \mathcal{P}^{free} \cup \mathcal{P}^{constraint} \quad (3.9)$$

By defining the PPS $\mathcal{P}$ as a topological space, one can define its associated subsets as well.

Definition 3.5 (Constraints). *A constraint can be defined as a set of all points in the topological PPS $\mathcal{P}$ that have to be avoided by the robot during path planning. The different categories of constraints are robot's mechanical limits, singularities, no-go zones and obstacles. The constraints of a robotic system may be defined as the union of the above categories.*

$$\mathcal{P}^{\text{constraint}} = \mathcal{P}^{\text{mechanical limits}} \cup \mathcal{P}^{\text{singularities}} \cup \mathcal{P}^{\text{no-go zones}} \cup \mathcal{P}^{\text{obstacles}}$$

Definition 3.6 (Constraint-free Space). *The constraint-free space is the subset of the topological PPS $\mathcal{P}$ which the robot is capable to access. It can be formally written as*

$$\mathcal{P}^{\text{free}} = \mathcal{P} \setminus \mathcal{P}^{\text{constraint}}$$

The associated spaces can be seen in Figure 2.18. Thus far, all spaces relevant to the PPP have been defined as topological spaces. While the topology enables intrinsic relationships amongst members of the space, it does not aid in defining the entire space or its subsets using more than just the imposed rules. That is, one must be able to characterize either the entire topological space or its subsets by an explicit or implicit description. The example of a sphere, which is a **Topological Manifold (TM)**, can be used to differentiate the explicit and implicit representations. An explicit description describes the nature of the space as viewed from outside the space, such as the equation of a sphere, which is a global representation from the perspective of the ambient space in which the sphere lives. The implicit description on the other hand, describes the nature of the space as viewed by being on the space, i.e., by not having an idea of what the entire space looks like globally. This is done by describing how one subset of the space can be related to or stitched to its neighbouring subset; for example, by describing the relationship between the Northern and Southern hemispheres of a sphere.

While topological spaces can be globally Euclidean in nature, like $\mathbb{R}^n$, most topological spaces are not so, as seen with the sphere. Therefore, the main binding agent that allows globally Euclidean and globally non-Euclidean topological spaces to be locally and globally defined, is their locally Euclidean nature. **TMs** describe such topological spaces either globally or locally. That is, they can either be described by a characterization of the entire space or by maps that relate its subsets. Since the PPS will eventually be defined as a PPM, the **TM** will be formally defined. Defining a **TM** involves showing that every subset of the space is locally Euclidean or that there exists a local relationship between the subset of m -dimensional tuples and $\mathbb{R}^m$. A homeomorphic map is used to describe that relationship. A homeomorphic map f is bijective, continuous and has a continuous inverse.

Definition 3.7 (Topological Manifold). *A topological manifold is a topological space $\mathcal{M}$ that is locally Euclidean of dimension m if every point p in $\mathcal{M}$ has a neighborhood $\mathcal{U}$ such that there exists a homeomorphism γ from $\mathcal{U}$ to an open subset of $\mathbb{R}^m$. The pair $(\mathcal{U}, \phi : \mathcal{U} \rightarrow \mathbb{R}^m)$ is a chart, $\mathcal{U}$ is a coordinate neighborhood or a coordinate open set, and ϕ is a coordinate map or a coordinate system on $\mathcal{U}$. [163]*

Some examples of manifolds are: a point, which is a 0-manifold, a line or circle, which is a 1-manifold; and a surface such as sphere, which is a 2-manifold, etc. The notation

n -manifold implies that the manifold is locally homeomorphic to $\mathbb{R}^n$. Illustrations of a 1- and 2-manifolds are seen in Figures 3.1 and 3.2. Definition 3.7 addresses the need to show that the TM is indeed locally mapped to $\mathbb{R}^m$, and by virtue of using a homeomorphic map, the inverse mapping back to the TM also exists. The inverse mapping establishes the local similarity to the Euclidean space.

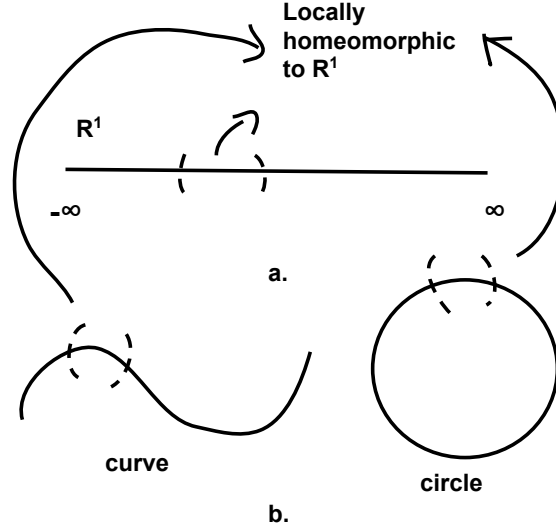


Figure 3.1: Example of a 1-manifold: a. The Euclidean space $\mathbb{R}^1$ is covered by a single chart $(\mathbb{R}^1, I_{\mathbb{R}^1})$, where $I_{\mathbb{R}^1} : \mathbb{R}^1 \rightarrow \mathbb{R}^1$ is the identity map. b. A curve and a circle are examples of a 1-manifold.

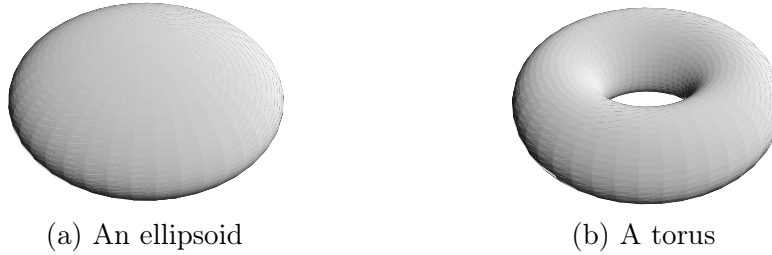


Figure 3.2: Examples of 2-manifolds [128]

In order to define the implicit representation of a TM, the relationship between its subsets must be established. Expressed by a map, this relationship imposes two qualities on the subsets. The first property imposes that the subsets lie on a TM, i.e., they are locally Euclidean. The second property imposes that the subsets lie on the “same” TM. This means that the subsets of the TM share some points via an intersection, and these shared points are also locally Euclidean. Establishing that the subsets indeed neighbour each other requires a diffeomorphic map. This is a stronger requirement than that of a homeomorphic map.

Diffeomorphisms are defined with the help of smooth maps. A map or function f that is once continuously differentiable is denoted as a $\mathcal{C}^1$ map. Subsequently, n -times continuously differentiable maps are $\mathcal{C}^n$ maps. $\mathcal{C}^\infty$ maps are infinitely continuously differentiable and are

called smooth maps. A diffeomorphic map f is a homeomorphism, whose inverse f^{-1} is at least C^1 . Smooth diffeomorphisms have inverses that are C^∞ .

Proposition 3.1 (Composition of diffeomorphisms). *Every composition of diffeomorphisms is a diffeomorphism. [97, Proposition 2.15(a)].*

Proposition 3.1 shows that diffeomorphic maps applied to diffeomorphic maps result in diffeomorphisms. That is, irrespective of the number of diffeomorphic transformations applied, they may always be reversed, and one can always go back to the first pre-image/domain of the map. This quality is essential in determining if neighbouring subsets do indeed lie on the same TM. As seen in Definition 3.7, subsets of a TM are represented by their mapping to the Euclidean space, which are otherwise called charts. Thus, the adjacency relationship shared by subsets of a TM can also be defined using the respective charts of each of the subsets. Such charts are called compatible charts.

To explain compatible charts, one must first check that the charts satisfy the underlying topology of the TM. Suppose $(\mathcal{U}, \phi : \mathcal{U} \rightarrow \mathbb{R}^n)$ and $(\mathcal{V}, \psi : \mathcal{V} \rightarrow \mathbb{R}^n)$ are two charts of a topological manifold. Since $\mathcal{U} \cap \mathcal{V}$ is open in $\mathcal{U}$ and $\phi : \mathcal{U} \rightarrow \mathbb{R}^n$ is a homeomorphism onto an open subset of $\mathbb{R}^n$, the image $\phi(\mathcal{U} \cap \mathcal{V})$ will also be an open subset of $\mathbb{R}^n$. Similarly, $\psi(\mathcal{U} \cap \mathcal{V})$ is an open subset of $\mathbb{R}^n$ [163]. This shows that the charts respect the underlying standard topology, as seen in Definition 3.4. Compatible charts can now be formally defined.

Definition 3.8 (Compatible Charts). *Two charts $(\mathcal{U}, \phi : \mathcal{U} \rightarrow \mathbb{R}^n)$, $(\mathcal{V}, \psi : \mathcal{V} \rightarrow \mathbb{R}^n)$ of a topological manifold are C^∞ -compatible if the two maps*

$$\begin{aligned}\phi \circ \psi^{-1} &: \psi(\mathcal{U} \cap \mathcal{V}) \rightarrow \phi(\mathcal{U} \cap \mathcal{V}) \\ \psi \circ \phi^{-1} &: \phi(\mathcal{U} \cap \mathcal{V}) \rightarrow \psi(\mathcal{U} \cap \mathcal{V})\end{aligned}$$

are C^∞ . These two maps are called the transition functions between the charts [163].

Definition 3.8 is illustrated in Figure 3.3 and will be explained later in an example. The compatibility of two charts and their operating sets means two things. First, both charts must be able to act on the intersection of the operating sets, since the intersection is what connects the two sets. Second, results of the charts' operation on the intersection should be quantifiable using a map. The second step is what defines the transition step. Note that such C^∞ -compatibility is automatic when the intersection of sets $(\mathcal{U}, \mathcal{V})$ is empty, since $\emptyset$ is part of the standard topology. From here on, since only C^∞ -compatible charts are focused on, in an abuse of notation, the label " C^∞ " will no longer be explicitly mentioned and C^∞ -compatible charts will be referred to simply as charts. These charts help identify a specific subset of the manifold, and by being compatible with another chart on the manifold, they help relate one subset of the manifold to another (usually neighbouring) subset of the manifold. If one were to find all such relationships in the TM, then one has a set of relationships that can enable traversing from one subset to another in the TM. In other words, the implicit description of the TM is accomplished by a set of compatible charts. Compatible charts impose restrictions on what kind of maps to and from the manifold are smooth and eventually enable defining continuity and smoothness, i.e., the fundamentals of calculus, on the entire manifold.

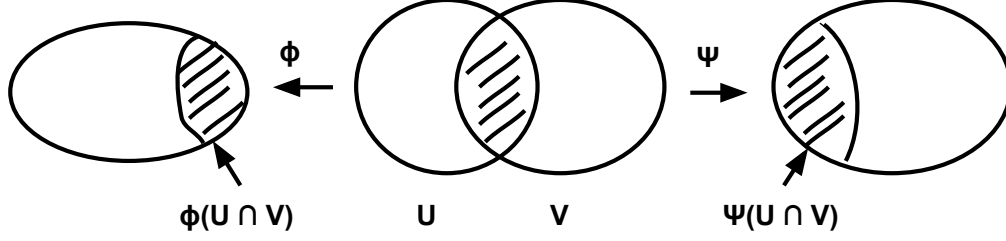


Figure 3.3: Illustration of Definition 3.8 shows that the transition function $\psi \circ \phi^{-1}$ is defined on $\phi(\mathcal{U} \cap \mathcal{V})$.

Definition 3.9 (Atlas). A C^∞ atlas, or simply an atlas, on a locally Euclidean space $\mathcal{M}$ is a collection $\mathcal{U} = \{(\mathcal{U}_\alpha, \phi_\alpha)\}$ of pairwise C^∞ -compatible charts that cover $\mathcal{M}$. That is, $\mathcal{M} = \bigcup_\alpha \mathcal{U}_\alpha$ [163].

Thus, a set of compatible charts, together as an atlas, pave the manifold. The following example illustrates some of the concepts introduced above.

Example 3.2 (Compatible Charts). Consider a unit circle $\mathbb{S}^1$, as seen in Figure 3.4. The

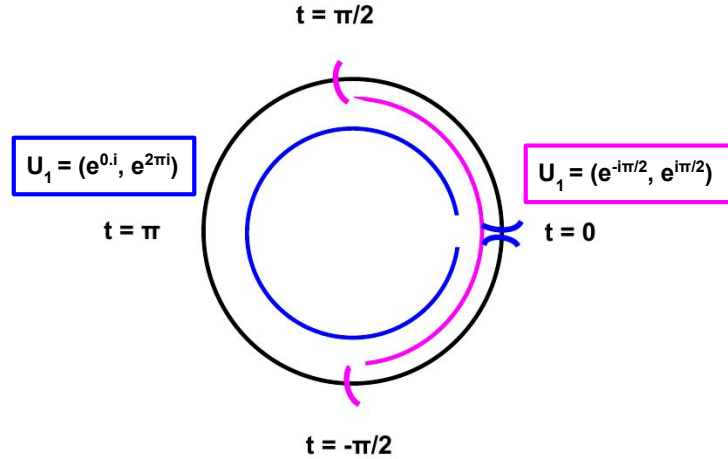


Figure 3.4: Open subsets of the circle, a 1-manifold from whom charts

set of points that make up the unit circle is given by $\mathcal{U} = \{e^{it} \in \mathbb{C} | 0 \leq t \leq 2\pi\}$. In order to create an atlas of this 1-manifold, one can consider two open sets $\mathcal{U}_1$ and $\mathcal{U}_2$ such that $\mathcal{U}_1 = \{e^{it} \in \mathbb{C} | 0 < t < 2\pi\}$ and $\mathcal{U}_2 = \{e^{it} \in \mathbb{C} | -\frac{\pi}{2} < t < \frac{\pi}{2}\}$. To show that these points are indeed locally homeomorphic to $\mathbb{R}^1$, one must find charts that map the open sets to $\mathbb{R}^1$. The following charts ϕ_1 and ϕ_2 of the form:

$$\phi_1 : \begin{cases} \mathcal{U}_1 \rightarrow \mathbb{R}^1 \\ (u = e^{it}) \mapsto t = \frac{1}{i} \log(u) \end{cases} \quad \phi_2 : \begin{cases} \mathcal{U}_2 \rightarrow \mathbb{R}^1 \\ (u = e^{it}) \mapsto t = \frac{1}{i} \log(u) \end{cases}$$

Both charts $(\mathcal{U}_1, \phi_1)$ and $(\mathcal{U}_2, \phi_2)$ are homeomorphisms. To check if they form compatible charts, they must satisfy Definition 3.8. First, define $\mathcal{U}_1 \cap \mathcal{U}_2 = (0, \frac{\pi}{2}) \sqcup (-\frac{\pi}{2}, 0)$, where

$\sqcup$ represents the disjoint union. Note from Definition 3.8 and Figure 3.3, that transition functions are only defined on the region of set intersection $\mathcal{U}_1 \cap \mathcal{U}_2$. The transition function is found if a map from $\phi_1(\mathcal{U}_1 \cap \mathcal{U}_2) \rightarrow \phi_2(\mathcal{U}_1 \cap \mathcal{U}_2)$ is found (along with its inverse). $\phi_1(\mathcal{U}_1 \cap \mathcal{U}_2)$ can be computed as follows:

$$\begin{aligned}\phi_1(\mathcal{U}_1 \cap \mathcal{U}_2) &= \phi_1\left(\left(e^{i \cdot 0}, e^{i \cdot 2\pi}\right) \cap \left(e^{i \frac{-\pi}{2}}, e^{i \frac{\pi}{2}}\right)\right) = \phi_1\left(\left(e^{i \cdot 0}, e^{i \frac{\pi}{2}}\right) \sqcup \left(e^{i \frac{-\pi}{2}}, e^{i \cdot 0}\right)\right) \\ &= \phi_1\left(e^{i \cdot 0}, e^{i \frac{\pi}{2}}\right) \sqcup \phi_1\left(e^{i \frac{-\pi}{2}}, e^{i \cdot 0}\right) = \left(0, \frac{\pi}{2}\right) \sqcup \left(\frac{3\pi}{2}, 0\right) \quad (3.10)\end{aligned}$$

Similarly, define $\phi_2(\mathcal{U}_1 \cap \mathcal{U}_2)$:

$$\begin{aligned}\phi_2(\mathcal{U}_1 \cap \mathcal{U}_2) &= \phi_2\left(\left(e^{i \cdot 0}, e^{i \cdot 2\pi}\right) \cap \left(e^{i \frac{-\pi}{2}}, e^{i \frac{\pi}{2}}\right)\right) = \phi_2\left(\left(e^{i \cdot 0}, e^{i \frac{\pi}{2}}\right) \sqcup \left(e^{i \frac{-\pi}{2}}, e^{i \cdot 0}\right)\right) \\ &= \phi_2\left(e^{i \cdot 0}, e^{i \frac{\pi}{2}}\right) \sqcup \phi_2\left(e^{i \frac{-\pi}{2}}, e^{i \cdot 0}\right) = \left(0, \frac{\pi}{2}\right) \sqcup \left(\frac{-\pi}{2}, 0\right) \quad (3.11)\end{aligned}$$

Now, one finds a map $f_{\text{transition}}$ that relates (3.10) and (3.11), i.e., $f_{\text{transition}} : \phi_1(\mathcal{U}_1 \cap \mathcal{U}_2) \rightarrow \phi_2(\mathcal{U}_1 \cap \mathcal{U}_2)$ or $f_{\text{transition}} \equiv \phi_2 \circ \phi_1^{-1}(\mathcal{U}_1 \cap \mathcal{U}_2)$. For $f_{\text{transition}}$ to exist,

$$\begin{aligned}f_{\text{transition}} &\equiv \phi_2 \circ \phi_1^{-1}(\mathcal{U}_1 \cap \mathcal{U}_2) : \\ \phi_1(\mathcal{U}_1 \cap \mathcal{U}_2) &= \left(\left(0, \frac{\pi}{2}\right) \sqcup \left(\frac{3\pi}{2}, 0\right)\right) \rightarrow \phi_2(\mathcal{U}_1 \cap \mathcal{U}_2) = \left(\left(0, \frac{\pi}{2}\right) \sqcup \left(\frac{-\pi}{2}, 0\right)\right)\end{aligned}$$

Thus, $f_{\text{transition}}$ is found to be

$$f_{\text{transition}} = \begin{cases} t - 2\pi, & \text{for } t \in \left(\frac{3\pi}{2}, 0\right) \\ t, & \text{for } t \in \left(0, \frac{3\pi}{2}\right) \end{cases}$$

Similarly, $f_{\text{transition}}^{-1} : \phi_2(\mathcal{U}_1 \cap \mathcal{U}_2) \rightarrow \phi_1(\mathcal{U}_1 \cap \mathcal{U}_2)$ is found as follows:

$$\begin{aligned}f_{\text{transition}}^{-1} &\equiv \phi_1 \circ \phi_2^{-1}(\mathcal{U}_1 \cap \mathcal{U}_2) : \\ \phi_2(\mathcal{U}_1 \cap \mathcal{U}_2) &= \left(\left(0, \frac{\pi}{2}\right) \sqcup \left(\frac{-\pi}{2}, 0\right)\right) \rightarrow \phi_1(\mathcal{U}_1 \cap \mathcal{U}_2) = \left(\left(0, \frac{\pi}{2}\right) \sqcup \left(\frac{3\pi}{2}, 0\right)\right)\end{aligned}$$

Thus, $f_{\text{transition}}^{-1}$ is found to be

$$f_{\text{transition}}^{-1} = \begin{cases} t + 2\pi, & \text{for } t \in \left(\frac{-\pi}{2}, 0\right) \\ t, & \text{for } t \in \left(0, \frac{\pi}{2}\right) \end{cases}$$

Since both $f_{\text{transition}}$ and $f_{\text{transition}}^{-1}$ exist, the charts ϕ_1 and ϕ_2 are compatible over $\mathcal{U}_1$ and $\mathcal{U}_2$. Since $f_{\text{transition}}$ and $f_{\text{transition}}^{-1}$ are smooth, $(\mathcal{U}_1, \phi_1)$ and $(\mathcal{U}_2, \phi_2)$ form $\mathcal{C}^\infty$ compatible charts. $\mathcal{U}_1 \cup \mathcal{U}_2$ results in the entire manifold, and since both $\mathcal{U}_1$ and $\mathcal{U}_2$ have compatible charts, $(\mathcal{U}_1, \phi_1)$ and $(\mathcal{U}_2, \phi_2)$ form a $\mathcal{C}^\infty$ atlas.

Definition 3.10 (Maximal Atlas). *An atlas $\mathcal{M}$ on a locally Euclidean space is said to be maximal if it is not contained in a larger atlas. In other words, if $\mathcal{M}$ is maximal and $\mathcal{U}$ is an atlas containing $\mathcal{M}$, then $\mathcal{U} = \mathcal{M}$ [163].*

Thus far, definitions of the manifold (space locally homeomorphic to $\mathbb{R}^n$), compatible charts and the union of all compatible charts that cover a topological space have been seen as stand alone concepts. Used together, the compatible charts of the maximal atlas introduce additional structure to the **TM** that allows properties of continuity, smoothness and invertibility invariant on the **TM**. It is referred to as a **Smooth Manifold (SM)**.

Definition 3.11 (Smooth Manifold). *A C^∞ manifold, or simply **SM**, M is a **TM** together with a maximal atlas. The maximal atlas is also called a differentiable structure on M . [163].*

Definition 3.12 (Manifold Dimension). *A manifold is said to have dimension n if all of its connected components are of dimension n [163].*

Thus to summarize, a **TM** is a **SM** if all transition maps in the atlas are C^∞ diffeomorphisms, i.e., all partial derivatives exist and are continuous. In such a case, the atlas is a smooth atlas [118, 119, 174]. For more details, the reader is advised to refer to [163, Proposition 5.10]. Stated simply, a smooth atlas has at least two charts that are smoothly compatible with each other [97]. Having defined a manifold and its atlas, one now define interactions between manifolds, such as creating a new manifold from two or more manifolds and mapping between two or more manifolds. Product manifolds and maps between manifolds will eventually help define the spaces and the relationship between the manifolds in **C** and **W** as manifolds.

Definition 3.13 (Product Manifold). *Suppose $M_1, \dots, M_k$ are **TMs** of dimensions $n_1, \dots, n_k$, respectively. The product space $M_1 \times \dots \times M_k$ is a **TM** of dimension $n_1 + \dots + n_k$. [97]*

The n -torus is an example of a topological n -manifold. For a positive integer n , the n -torus is the product space $\mathbb{T}^n = \mathbb{S}^1 \times \dots \times \mathbb{S}^1$ [97]; see Figure 3.2(b) for a genus-1, 2-torus which is a 2-manifold. A similar extension can be afforded to smooth product manifolds.

Definition 3.14 (Smooth Product Manifold). *If $M_1, \dots, M_k$ are **SMs** of dimensions $n_1, \dots, n_k$, respectively, then the product space $M_1 \times \dots \times M_k$ is a **TM** of dimension $n_1 + \dots + n_k$, with charts of the form $(\mathcal{U}_1 \times \dots \times \mathcal{U}_k, \phi_1 \times \dots \times \phi_k)$. Any two such charts are smoothly compatible. This defines a natural smooth manifold structure on the product, called the **Product Smooth Manifold (PSM)** structure. [97]*

This implies that if $\{\mathcal{U}_\alpha, \phi_\alpha\}$ and $\{\mathcal{V}_i, \psi_i\}$ are C^∞ atlases for the manifolds M and N of dimensions m and n , respectively, then the collection of charts

$$\{(\mathcal{U}_\alpha \times \mathcal{V}_i, \phi_\alpha \times \psi_i : \mathcal{U}_\alpha \times \mathcal{V}_i \rightarrow \mathbb{R}^m \times \mathbb{R}^n)\}$$

is a C^∞ atlas on $M \times N$. Therefore, $M \times N$ is a C^∞ manifold of dimension $m + n$ [163]. Using the example of the n -torus, this yields a **SM** structure on the n -torus product manifold defined by $\mathbb{T}^n = \mathbb{S}^1 \times \dots \times \mathbb{S}^1$ [97].

So far, one has seen how a new manifold with a larger dimension can be constructed from two smaller dimensional manifolds. The next step is to be able to traverse from one SM to another using a smooth map and subsequently using a diffeomorphism. A smooth map between manifolds is defined in Definition 3.15 and illustrated in Figure 3.5.

Definition 3.15 (Smooth Maps between Manifolds). *Let M and N be SMs. Then, the map $F : M \rightarrow N$ is a smooth map, if for every $p \in M$, there exist smooth charts $(\mathcal{U}, \phi)$ containing p and $(\mathcal{V}, \psi)$ containing $F(p)$, such that $F(\mathcal{U}) \subseteq \mathcal{V}$ and the composite map $\psi \circ F \circ \phi^{-1}$ is smooth from $\phi(\mathcal{U})$ to $\psi(\mathcal{V})$ [97].*

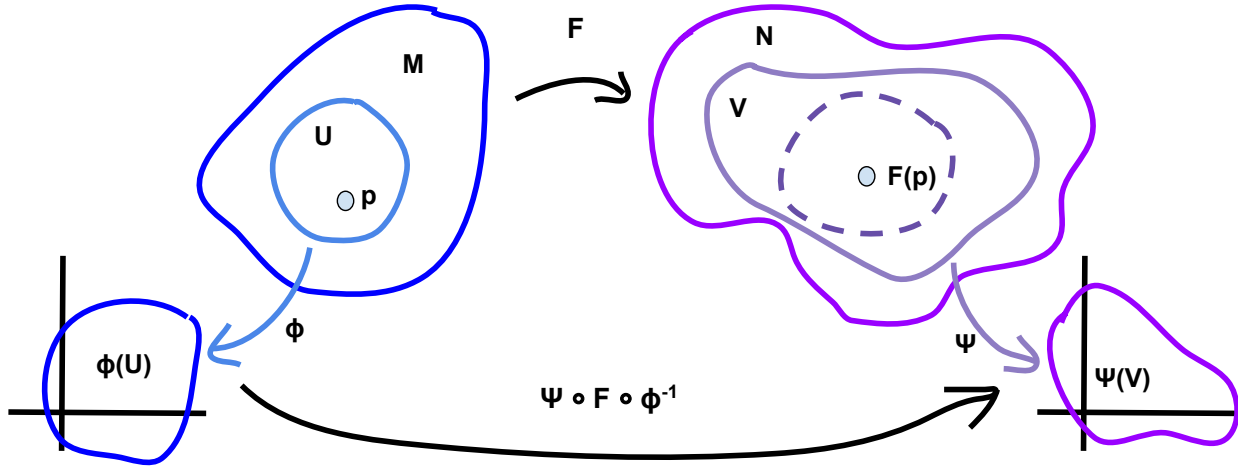


Figure 3.5: Illustration of a smooth map between manifolds as seen in Definition 3.15

Having defined a smooth map between two manifolds using charts, a diffeomorphic map between two manifolds can be understood. If M and N are SMs, a diffeomorphism from M to N is a smooth bijective map $F : M \rightarrow N$ that has a smooth inverse. That is, M and N are diffeomorphic if there exists a diffeomorphism between them. It is symbolized by $M \approx N$. The following example explains this concept.

Example 3.3 (Diffeomorphic SMs). *Consider the two maps $F : X \rightarrow Y$ and $G : Y \rightarrow X$, such that $F(x) = x / \sqrt{1 - |x|^2}$ and $G(y) = y / \sqrt{1 + |y|^2}$. Both F and G are smooth. Since they are inverses of each other, they are both smooth inverses. Thus, they are diffeomorphic.*

Since manifolds are usually defined by charts and open sets, it is natural to check for a local diffeomorphism between manifolds by using the chart representation. Theorem 3.1 helps determine when a local diffeomorphism exists between two manifolds.

Theorem 3.1 (Inverse Function Theorem for Manifolds (IFTM)). *Let $F : M \rightarrow N$ be a C^∞ map between two manifolds of the same dimension, and $\mathbf{p} \in M$. Suppose for some charts $(\mathcal{U}, \phi) = (\mathcal{U}, x^1, \dots, x^n)$ about $\mathbf{p} \in M$ and $(\mathcal{V}, \psi) = (\mathcal{V}, y^1, \dots, y^n)$ about $F(\mathbf{p}) \in N$, $F(\mathcal{U}) \subset \mathcal{V}$. Set $F^i = y^i \circ F$. Then, F is locally invertible at $\mathbf{p}$ if and only if its Jacobian determinant $\det(\delta F^i / \delta x^j(\mathbf{p}))$ is nonzero [163].*

An interesting consequence of diffeomorphisms between manifolds is that both manifolds must be of the same dimension. This is made explicit in Theorem 3.2.

Theorem 3.2 (Diffeomorphism Invariance of Dimension). *A nonempty SM of dimension m cannot be diffeomorphic to an n -dimensional SM unless $m = n$ [97].*

This dimensional invariance that is imposed using diffeomorphisms later allows for clear definition of mapping to and from constraint-free manifolds.

It was noted earlier that TMs are topological spaces with additional structure that could be represented implicitly and explicitly. As seen for TMs and SMs , charts and the associated atlas accomplish the implicit definition of manifolds. The global or explicit representation of the manifold is seen from its ambient space. As seen previously, manifolds exist in many dimensions, such as the 0-manifold (e.g., a point), 1-manifold (e.g., a line or circle), and 2-manifold (e.g., surfaces such as a sphere, cylinder, torus, etc.). Since the global representation is from the perspective of the ambient space, it must be noted that global representations are possible only if the dimension of the ambient space within which the manifold lives is greater than the dimension of the manifold. So, the aforementioned examples can be globally represented provided that an n -manifold lives in an $(n + 1)$ or higher-dimensional space. For example, a 2-manifold, such as a sphere or cylinder, lies in $\mathbb{R}^n$, where $n \geq 3$. Thus, it is easy to see that any manifold of greater dimension is not easy for humans to perceive. Hence, only manifolds of dimensions 0, 1 and 2 can be visualized, and thus perceived for the PPP . For the purpose of this thesis, only dimensions of 2 or less will be considered to represent the manifolds associated with the PPP . The PPS will be defined as a 2-manifold whenever possible and the path generated will be defined as a 1-manifold. Thus, the focus for global and explicit representation will be defined for a 2-manifold, from the perspective of a surface in space, followed by the parametrization of a 1-manifold.

Definition 3.16 (Surface). *A subset $\mathcal{S}_{surface}$ of $\mathbb{R}^3$ is a surface if, for every point $\mathbf{p} \in \mathcal{S}_{surface}$, there is an open set $\mathcal{U}$ in $\mathbb{R}^2$ and an open set $\mathcal{W}$ in $\mathbb{R}^3$ containing $\mathbf{p}$, such that $\mathcal{S}_{surface} \cap \mathcal{W}$ is homeomorphic to $\mathcal{U}$. A subset of surface $\mathcal{S}_{surface}$ of the form $\mathcal{S}_{surface} \cap \mathcal{W}_{subset}$, where $\mathcal{W}_{subset}$ is an open subset of $\mathbb{R}^3$, is called an open subset of $\mathcal{S}_{surface}$. A homeomorphism $\rho : \mathcal{U} \rightarrow \mathcal{S}_{surface} \cap \mathcal{W}_{subset}$ as in this definition, is called a surface patch or parametrization of the open subset $\mathcal{S}_{surface} \cap \mathcal{W}_{subset}$ of $\mathcal{S}_{surface}$. A collection of such surface patches whose images cover the whole of $\mathcal{S}_{surface}$ is called an atlas of $\mathcal{S}_{surface}$ [128].*

Examples of a surface include a plane, disc, cylinder, annulus, sphere, ellipsoid, and torus, to name a few. Figures 3.2, 3.6 and 3.7 show a few such examples. The hole-like region of a torus is referred to as a genus. A surface with n hole-like regions is generally called an n -genus surface.

To define a path in the PPM , we formally define a 1-manifold or any variant of a curve using the following definition.

Definition 3.17 (Parametrized curve). *A parametrized curve in $\mathbb{R}^n$ is a map $\gamma : (\alpha, \beta) \rightarrow \mathbb{R}^n$, for some α, β , such that $-\infty \leq \alpha \leq \beta \leq \infty$ [128].*

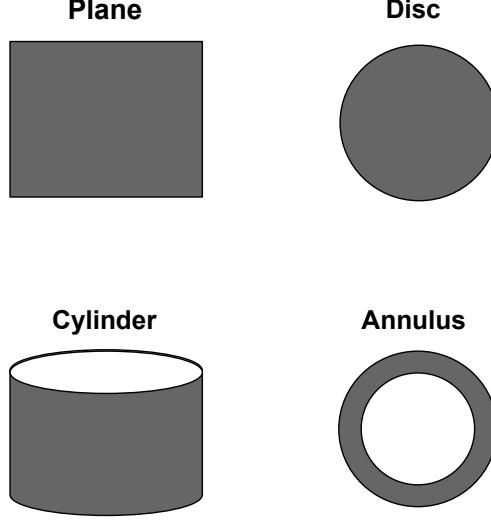


Figure 3.6: Common examples of surfaces

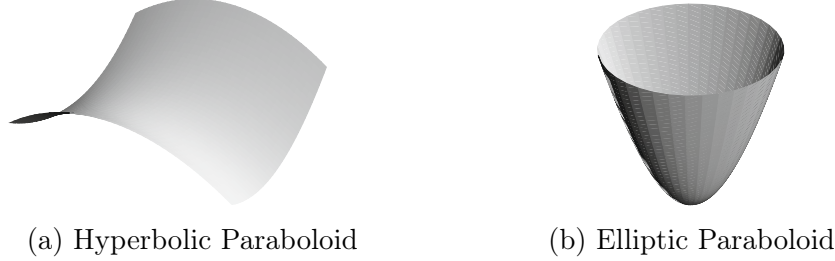


Figure 3.7: Examples of surfaces [97]

The symbol (α, β) denotes the open interval $(\alpha, \beta) = \{t \in \mathbb{R} \mid \alpha < t < \beta\}$. An example is that of the unit circle, where $\gamma(t) = (\gamma_1(t), \gamma_2(t)) : t \rightarrow (\cos(t), \sin(t))$ for $t \in (0, 2\pi + \delta)$, with δ being an arbitrarily small number. A parametrized curve of 1-manifold, whose image is contained in a level curve $\mathcal{LC}$, is called a parametrization of (part of) $\mathcal{LC}$ [128]. The $\mathcal{LC}$ representation is analogous to the surface $\mathcal{S}_{surface}$ represented by a level set function. The goal of the proposed CFDMPP is to generate the shortest constraint-free path on the PPM. The shortest path on a manifold is called a geodesic. In order to understand a geodesic on a surface/manifold, one requires the definition of the tangent space on a surface.

Definition 3.18 (Tangent Space). *A tangent vector to a surface $\mathcal{S}_{surface}$ at a point $\mathbf{p} \in \mathcal{S}_{surface}$ is the tangent vector at $\mathbf{p}$ of a curve in $\mathcal{S}$ passing through $\mathbf{p}$. The tangent space $\mathbf{T}_p\mathcal{S}$ of $\mathcal{S}_{surface}$ at $\mathbf{p}$ is the set of all tangent vectors to $\mathcal{S}_{surface}$ at $\mathbf{p}$ [128].*

When the $\mathbf{T}_p\mathcal{S}$ is a plane, it is referred to as the tangent plane.

Definition 3.19 (Geodesic). *A curve $\gamma(t)$ on a surface $\mathcal{S}_{surface}$ is called a geodesic if for every value t_0 of the parameter t , $\ddot{\gamma}(t_0)$ is zero or perpendicular to the tangent plane of the surface at that point $\gamma(t_0)$, i.e., parallel to its unit normal [128].*

The proposed CFDMPP is based on generating geodesics as constraint-free paths. That

is, the aim is to generate the shortest path possible on the $\mathcal{PPS}$, while considering its underlying curvature. Some examples of geodesics are shown in Figure 2.17. However, modifications to the generated shortest paths may be necessary to overcome dynamic constraints. To this end, concepts known as homotopy and homology, are applied. Before providing formal definitions associated with both concepts, let us look at how they can be linked to the path planning problem.

Topological spaces lay the foundation for defining metrics, which in turn formalize notions of continuity and differentiability. At this stage, with the help of more definitions, one can define a surface, manifold or in its most basic form, a space. In the path planning context, it helps define the $\mathcal{PPS}$. A path between two points in that $\mathcal{PPS}$ is then defined as the image of a map, whose pre-image lies in some space, for example, the control input space. In a $\mathcal{PPS}$ devoid of constraints, i.e. $\mathcal{P} = \mathcal{P}^{\text{free}}$, the desired path between $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, is typically the shortest path. This assumes that the metric to evaluate path quality is that of path length. Then, geodesics allow us to generate the shortest path on a surface, as defined by the metric on that surface. Therefore, the metric of minimizing path length can be held as a reliable metric across $\mathcal{PPS}$ s.

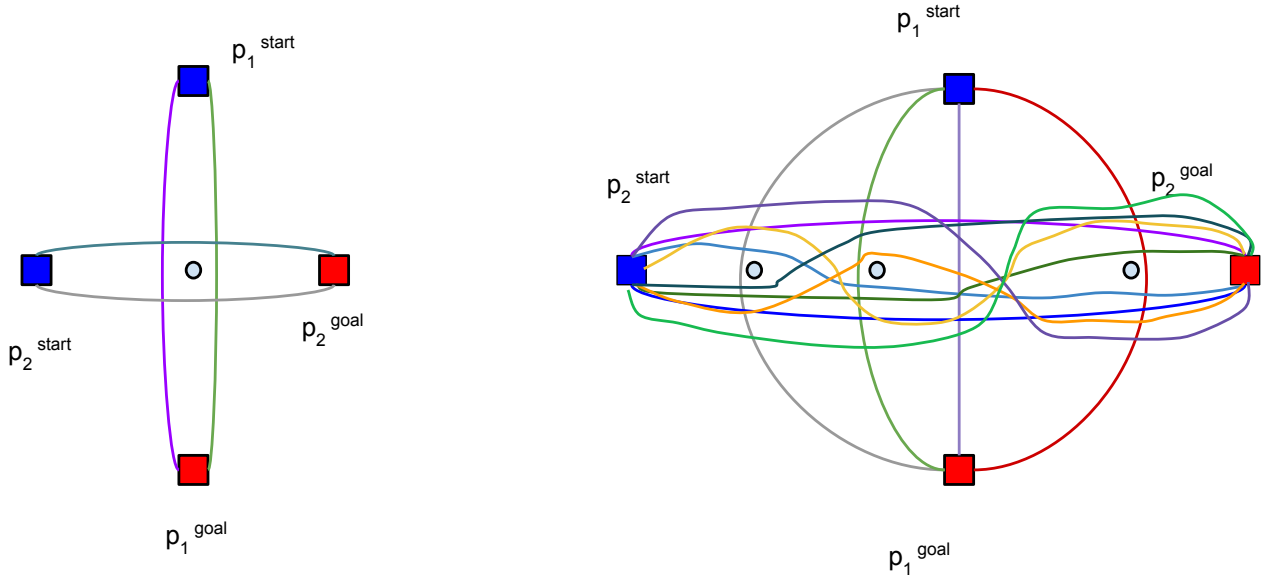


Figure 3.8: Potential paths for different start and goal pairs. The trivial case with one missing region is on the left hand side, and the non-trivial case with three missing regions is on the right hand side.

If however, the $\mathcal{PPS}$ is no longer pristine, that is, $\mathcal{P} \neq \mathcal{P}^{\text{free}}$, then some mathematical element has disturbed the $\mathcal{PPS}$. An exclusion of a region (as small as a point) from the $\mathcal{PPS}$ has occurred, such that the space can no longer be defined as it was before. Naturally, in path planning, an intrusion by a constraint renders a region of the $\mathcal{PPS}$ inaccessible for path planning. This exclusion of free space changes the original definition of the $\mathcal{PPS}$, to now reflect a missing region. Note that the term missing region is not simply a visual marker, but

eventually forms the basis of understanding homotopy and homology. This causes a change in the topological property of connectedness of the space, which is formally defined later. Roughly speaking however, a space is simply connected if every closed curve in the space can be shrunk to a point [115]. In the example considered thus far, points that previously could have been connected with a line segment, may no longer be able to satisfy that. So in the path planning parallel, the shortest path may no longer be mathematically correct, if it appears to cut across the missing region. In such a case, a new shortest path must be planned. But, there may now exist alternative paths, each satisfying the metric of path length.

This can occur as a consequence of the path having to divert itself around the missing region, so as to still lay on the new **PPS** and also reaching its destination. One missing region can result in two possible diversions, and thus result in two alternative paths. Determining the optimal path first involves recognizing that there exist two alternative paths. Doing so is trivial in the case of one missing region. However, it is non-trivial when there exist more than one constraint or missing region. Figure 3.8 illustrates the trivial case with one missing region and a non-trivial case with three missing regions for two different start-goal pairs. The differently coloured lines represent unique path types between their respective start (blue square) and goal (red square) locations.

The number of possible paths in a space with several missing regions, depends on two attributes: the locations of $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, and the number of missing regions, as seen in Figure 3.8. Figure 3.8 shows that even for the trivial case, the resulting paths are different for different start-goal pairs. It also shows that for the non-trivial case, the number and type of possible paths can be dramatically different, with a change in start-goal pair and the number of missing regions. Clearly, the number of possible paths is directly proportional to the number of missing regions in the space. So, path planning in a $\mathcal{P}^{\text{free}}$ which is littered with missing regions, can now present multiple constraint free paths between the requested $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$. In order to evaluate each one for its compliance with the chosen metric, a list of possible paths needs to be created. The list requires distinguishing paths that are different from one another. It also requires that we identify paths that are simply a variant (for example, longer or shorter version) of another path, so that they are grouped together. This important requirement of being able to group paths into one, or separate them into different groups, is what leads us to homotopy and homology.

Roughly speaking, homotopy and homology give us a tool to uniquely characterize a path, by determining how many missing regions the path encloses. Consequentially, paths that are of different lengths but still enclose the same missing regions can be characterized as being the same kind of paths. Inversely, paths of the same path lengths that enclose different missing regions will be classified as being different kinds of paths. The combined use of homology and homotopy endows us the capability of making the same judgements on paths, even when they do not fully enclose the missing region. This is typical of paths in the path planning context, when shortest paths are contours located between missing regions, but not necessarily winding around the entirety of the missing region.

In such a case, homology equips us with another trait that further aids in uniquely characterizing the path. The trait marks the direction that the path takes when it encloses

some or all of the missing region. The direction of enclosure, in conjunction with the number of missing regions enclosed, now provides a complete measure of the uniqueness of a path kind. Using this as an indicator of path type, a generated path may now be classified as belonging to or being different from, a path type. Paths belonging to the same type may be evaluated for a metric so that the best path can be used as an ideal representative for that path type. Representative paths across different path types may then finally be evaluated for a metric, so that the best path type may be chosen as the ideal path type. This process then returns best path across path types, and within the chosen path type. Now that we intuitively understand the importance of homology and homotopy in the context of path planning, we shall progress towards their associated definitions.

Definition 3.20 (Homotopy). *If f and f' are continuous maps of from spaces $X \rightarrow Y$, f is homotopic to f' if there is a continuous map $F : X \times I \rightarrow Y$ such that*

$$F(x, 0) = f(x) \text{ and } F(x, 1) = f'(x)$$

where $I = [0, 1]$ is the unit-interval. The map F is called a homotopy between f and f' and the relationship between the two maps are written as $f \simeq f'$ [115].

In other words, homotopy can be thought of as a one-parameter family of maps from $X \rightarrow Y$. If that single parameter was interpreted as time, for instance, then homotopy can be viewed as follows. As time is varied from 0 to 1, the homotopy F correspondingly deforms the map from f to f' .

Homotopy as defined above applies to maps in general. We are interested in homotopy as specifically applied to paths in a space, so we shall first define a path. A path in X from x_0 to x_1 is a continuous map $f : I \rightarrow X$ such that $f(0) = x_0$ and $f(1) = x_1$, where $I = [0, 1]$ is the unit interval [98]. Then, x_0 and x_1 are termed as the initial and final points, respectively.

We now define homotopy specifically for paths, called path-homotopy, which is a stronger relation than mere homotopy [115].

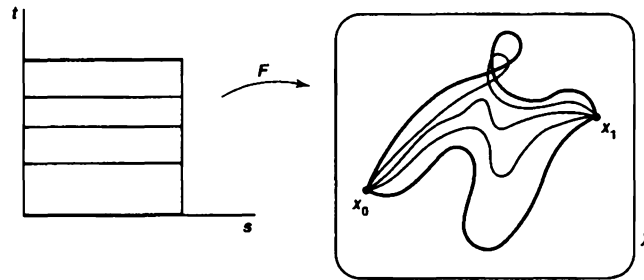


Figure 3.9: Path homotopy with the unit plane on the left hand side and the resulting homotopic paths on the right hand side. The divisions of the unit plane are mapped to X as homotopic paths, with the same initial and final points [115]

Definition 3.21 (Path Homotopy). *Two paths f and f' from $I \rightarrow X$, where $I = [0, 1]$, are said to be path homotopic, if they have the same initial point x_0 and the same final point x_1*

and if there is a continuous map $F : I \times I \rightarrow X$ such that:

$$F(s, 0) = f(s) \text{ and } F(s, 1) = f'(s) \quad (3.12)$$

$$F(0, t) = x_0 \text{ and } F(1, t) = x_1 \quad (3.13)$$

$\forall s \in I$ and $\forall t \in I$. Then, F is called the path homotopy between f and f' and the homotopic paths are expressed as $f \simeq_p f'$. [115].

The two conditions in the definition can be explained as follows. The first condition states that F is a homotopy between f and f' as per Definition 3.20. The second condition now imposes that the above deformation take place, all while keeping the initial and final points constant. Figure 3.9 illustrates exactly this. Path homotopy is of consequence for the path planning context, essentially for the second condition. This is so, because in the event of encountering a constraint, the start and goal locations still remain the same, but an alteration of the path is necessary.

Lemma 3.1. *The relations $\simeq$ (homotopy) and $\simeq_p$ (path homotopy) are equivalence relations. [115].*

This lemma aids us with a significant tool for path planning, because we are now able to categorize different paths as equivalent, if they are homotopic. Formally, if f is a path, its path-homotopy class is denoted by $[f]$. The formal proof of the lemma is seen in [115]. We shall informally explore the three requirements for path homotopy to be an equivalence relation. In the process, its implications in the context of path planning are seen.

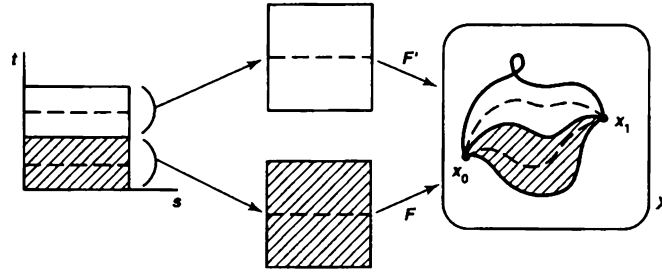


Figure 51.2

Figure 3.10: If $F : f \simeq_p f'$ and $F' : f' \simeq_p f''$, then $G : f \simeq_p f''$.

The first requirement of showing that $f \simeq_p f$ is trivial. The second shows that $f \simeq_p f'$ implies that $f' \simeq_p f$. This is important because it establishes that irrespective of the sequence in which paths of a class are computed, homotopy between them can be established. The next requirement imposes that if $f \simeq_p f'$ and $f' \simeq_p f''$, then $f \simeq_p f''$. This now shows that irrespective of how many paths are computed, one can show their equivalence by relating all paths to one representative path. Such a situation can occur when path planning algorithms produce numerous tentative paths based on their current point of exploration. Figure 3.11 illustrates straight line homotopy, where the homotopy between paths is carried across through straight line segments. The illustration on the right hand side has a missing

region at the origin, shows three different paths. Paths g and f are (straight line) path homotopic, but h is not homotopic to f . It is intuitively clear that there exists no homotopy in that case, one cannot deform f past the missing region at the origin. Otherwise stated, one cannot continuously shrink f to a point in that space, because the smallest shrunk contour will wind around the origin, and not at it. One now starts to see how the concept of a missing region relates to path homotopy, and eventually, different homotopic classes. In the context of path planning, this aids in representing constraints as missing regions around which path classes are defined.

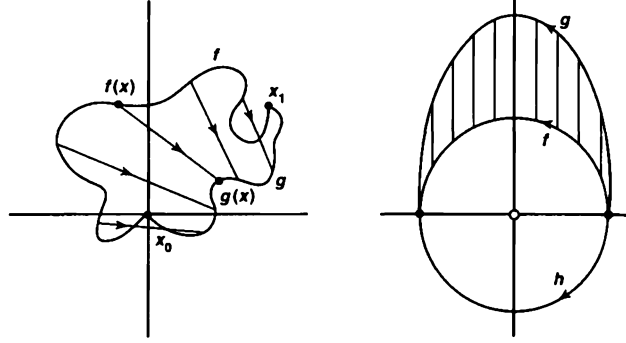


Figure 3.11: Straight line homotopy. Left hand side: Path homotopy between f and g . Right hand side: Path homotopy between g and f , but not between f and h ; [115].

Subsequent definitions find important use for path planning in a discretized [PPS](#).

Definition 3.22 (Product of paths). *If f is a path in X from x_0 to x_1 , and if g is a path in X from x_1 to x_2 , the product of f and g is the path h . It is denoted by $h = f * g$ and is given by the equations([115]):*

$$h(s) = \begin{cases} f(2s) & \text{for } s \in [0, \frac{1}{2}] \\ g(2s - 1) & \text{for } s \in [\frac{1}{2}, 1] \end{cases}$$

The function h is continuous and well defined on the unit interval, and it represents a path in X from x_0 to x_2 . The path h initially travels from x_0 to x_1 via path f , and then from x_1 to x_2 via g . Both segments of the final path are now being traversed at twice the "speed" of the parameter $s \in I$, so that the final path still satisfies the definition of being a path from $I \rightarrow X$. Definition 3.22 equips us with the ability to deal with incremental path planning. That is when a path is planned, it is typically generated incrementally, segment wise. Path homotopy was defined for a path between an initial and final point; however in path planning situations, that path itself is a composition of path segments. So, it is of great use to know that the composition of path segments can still undergo the same analysis as opposed to the global, unified representation of the path. Armed with the product operation on paths, one can now appreciate the induced product operation on path-homotopy classes as defined in [115]:

$$[f] * [g] = [f * g] \quad (3.14)$$

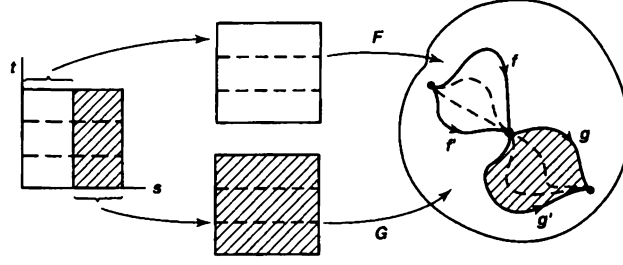


Figure 3.12: The product operation of two path-homotopy classes [115].

Figure 3.12 shows the significance of the product operation on path-homotopy classes. The figure illustrates that the product of two path-homotopy classes is the path-homotopy class of the product of paths. The operation $*$ satisfies properties of associativity, right and left identities and the inverse, referred to as the groupoid properties. One notes that these properties are satisfied only when $f(1) = g(0)$, i.e. when the product operation is defined on the paths themselves. A remarkable consequence of the groupoid properties, shows that the product operation on path-homotopy classes is not restricted to simply two classes as seen in equation (3.14). Intuitively, the result shows that associativity of path-homotopy classes holds for any finite product of classes. Which in turn means that, the path-homotopy class of the resulting path remains the same, irrespective of how many intermediate path segments, the product is applied to. Stated formally as a theorem as seen in [115]:

Theorem 3.3. *Let f be a path in X and let $a_0, \dots, a_n$ be numbers, such that $0 = a_0 < a_1 < \dots, a_n = 1$. Let $f_i : I \rightarrow X$ be the path that equals the positive linear map of $I = [0, 1]$ onto $[a_{i-1}, a_i]$ followed by f . Then, $[f] = [f_1] * \dots * [f_n]$.*

Theorem 3.3 is of prime significance for the purposes of path planning, and specifically so, in the context of discretized path planning. This thesis proposes discretizing the PPS and then commencing path planning. That is, path segments are generated incrementally and their composition is what needs to be evaluated for the path-homotopy class. The number of path segments generated is proportional to the map resolution. So with the help of theorem 3.3, one is confident that theoretically, discretized path segments and the map resolution has no detrimental effect on the path-homotopy class. Therefore, generated path segments can be evaluated to see if they belong to the same or different classes.

One now understands the significance of homotopy classes with respect to possible paths navigating an environment with constraints. It is easy to see that the constraints are what give rise to the different homotopy classes in the first place. That is, the number and placement of constraints/missing regions have an impact on the number of type of homotopy classes. With this in consideration, one now approaches the concept of homology.

Homology theory is quite detailed, and beyond the scope of this thesis. However, the important take away from homology is that the use of a specific kind of homology group (singular homology group) uses singular chains to detect holes [98]. Any chain that closes up on itself (like a closed path) but is not equal to the “boundary value” of a chain of one higher dimension must surround a hole in X [98]. These homology groups are topological invariants

and are also homotopy invariants [98]. Intuitively speaking, homology allows detection of missing regions/ holes in a topological space. This is observed with the help of cycles or closed contours on that space. The cycles are shrunk to see if they can be collapsed to a point, or their inability to be shrunk beyond a certain point. In the latter case, it implies that the shrunk cycle has reached the boundary of a missing region. In a manner similar to homotopy classes, cycles can be deformed within the same homology class. Homology has computation tools that make it easier to first compute homology classes, and then use it to determine homotopic classes. As the authors of [19] (on whose work this section of the thesis is based) state, homology serves as a fair analog of homotopy in most robotics problems. We will now explain why that is so.

Since homology depends on a closed contour or a cycle, we relate the latter to paths, and express homology as a function of paths. Given a path $f : I \rightarrow X$ with x_0 and x_1 as the initial and final points, one defines the reverse path from x_1 and x_0 as $\bar{f}(s) = f(1 - s)$ where $s \in I$ [98]. Figure 3.13 shows both such paths around a missing region. Recall that the product of paths is also a path, and such is the case for f and $\bar{f}$, resulting in a contour that surrounds the missing region. Thus, this cycle or closed contour will define the homology class surrounding the missing region. The next step is to imbue meaning to this contour in the context of path planning, so that computations of paths in different classes have a physical meaning. In the process, we will use tools from complex analysis; therefore, definitions will be introduced for clarification.

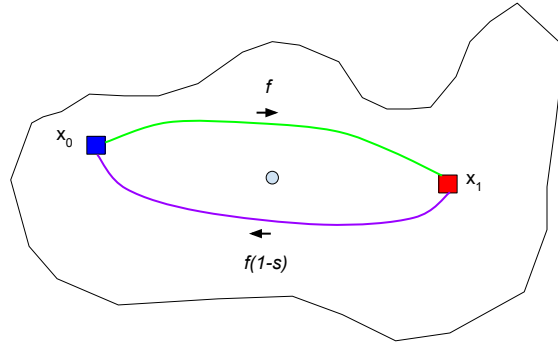


Figure 3.13: Connection between homology and homotopy

One way to characterize a contour around a missing region is by looking at its winding number around that missing region. The winding number $v(L, p)$ of a closed loop or contour about the point $p \in \mathbb{C}$ is the net number of revolutions of the direction of z as it traces out L once in its given manner [117]. A simple loop does not self intersect. For loops that are not simple, the winding number is determined by the number of partitions the sub-loops of the loop break the space into. For example, Figure 3.14 shows the partition of the space into partitions D_i where $i \in [1, 4]$. Here, each partition is assumed to have one representative point p_i , so that the winding can be counted. The winding numbers v_i as seen in the figure, indicate the number of times the loop L winds around D_i . Partitions that have a non-zero winding number v_i are defined as the inside of the loop, and vice versa. This is helpful to understand because of our interest in the presence of poles/constraints/missing regions. The

computations of winding numbers depend on where the poles are located with respect to the contour- inside or outside. The demonstration of this requires some definitions, so we proceed. The following definitions deal with functions on the complex plane denoted by $\mathbb{C}$.

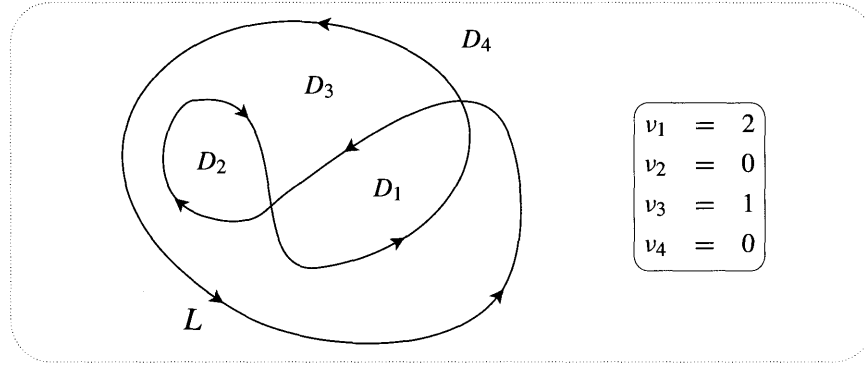


Figure 3.14: Non-simple loop: Winding number of its partitions [117]

Definition 3.23 (Holomorphic Function). *Let Ω be an open set in $\mathbb{C}$ and f a complex-valued function on Ω . The function f is holomorphic at the point $z_0 \in \Omega$ if the quotient*

$$\frac{f(z_0 + h) - f(z_0)}{h} \quad (3.15)$$

converges to a limit when $h \rightarrow 0$. In the above, $h \in \mathbb{C}$ and $h \neq 0$ but can approach 0 from any direction. Furthermore, $z_0 + h \in \Omega$, such that the quotient is well defined [156]. When the limit of the quotient or the derivative of f at z_0 exists, it is denoted by:

$$f'(z_0) = \lim_{h \rightarrow 0} \frac{f(z_0 + h) - f(z_0)}{h} \quad (3.16)$$

The function f is said to be holomorphic on Ω if f is holomorphic at every point of Ω . If D is a closed subset of $\mathbb{C}$, f is holomorphic on D if f is holomorphic in some open set containing D . If f is holomorphic in all of $\mathbb{C}$, then f is entire [156]. The term holomorphic function is synonymous with complex differentiable; however, holomorphic functions satisfy much stronger properties. Holomorphic functions are smooth (infinitely complex differentiable) and are analytic (power series expansion at every point). Holomorphic functions are of significance to path planning because their properties allow computations of integrals involving contours that enclose poles.

Consider figure 3.15, which has loop K and a singularity. Cauchy's theorem states that if a holomorphic mapping has no singularities inside a loop, its integral around a loop vanishes [117]. Using partitions as defined above, one can see that loop K does not encircle the singularity, and thus its integral around the loop should vanish. Now stated formally:

Theorem 3.4 (Cauchy's theorem for a disc). *If f is a holomorphic function in an open disc, then*

$$\int_{\gamma} f(z) = 0 \quad (3.17)$$

for any closed curve γ in that disc given by $D_r(z_0) = \{z \in \mathbb{C} : |z - z_0| < r\}$. [156].

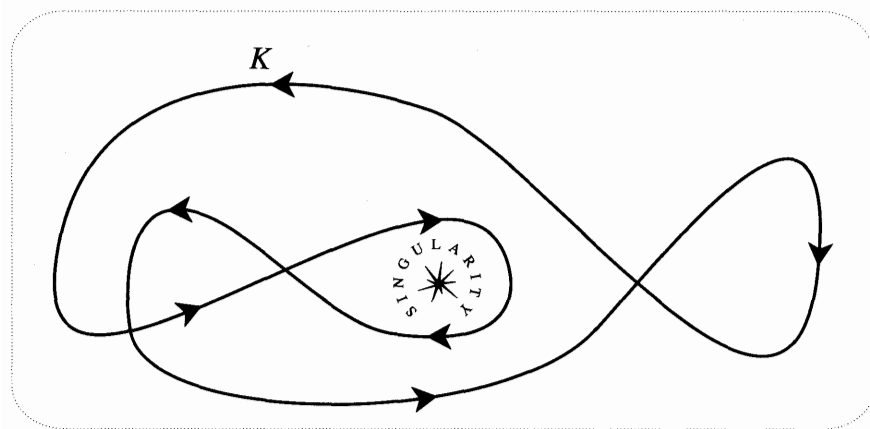


Figure 3.15: Non-simple loop winding around one pole [117]

The above definition for a disc is generalized to other spaces via open sets as follows. Cauchy's theorem states that if f is holomorphic in an open set Ω and $\Gamma \subset \Omega$ is a closed curve whose interior is also contained in Ω , then (3.17) holds true. Theorem 3.4 establishes the conditions under which the value of the integral is independent of the contour. The integrals of two mappings between x_0 and x_1 will agree, provided that the mapping is analytic (holomorphic in $\mathbb{C}$) everywhere in the region lying between the two contours [117]. This agreement of integral values can be used to determine equivalence of paths in homology and homotopy classes. That is, if two contours γ_1 and γ_2 acted on by the same holomorphic function have the same integrals along the contours, they represent the same kind of contour. For our purposes, they represent the same kind of path. The theorem as seen above, sets a baseline for when there are no missing regions, or poles on the inside of a contour. Subsequent explanations handle the situation when poles are on the inside of the contour.

Theorem 3.5 (Cauchy's Integral Formula). *Suppose f is holomorphic in an open set that contains the closure of a disc D . If C denotes the boundary circle of this disc with the positive orientation, then*

$$f(z_0) = \frac{1}{2\pi i} \int_C \frac{f(z)}{z - z_0} dz \quad (3.18)$$

for any point $z_0 \in D$ [156].

Of course, this can be extended to f being holomorphic everywhere inside a contour or loop γ , with z_0 being a point inside γ [117]. With theorem 3.5, one obtains an expression for the integral of a holomorphic mapping on a contour that surrounds a pole. The interpretation is as follows. Since $f(z)$ is holomorphic inside γ , the function $f(z)/(z - z_0)$ is also holomorphic there, except that it has a singularity at $z = z_0$. One such example is that of $f(z) = 1/z$ applied on the unit circle, i.e. $e^{i\theta}$. Since $f(z)$ has a pole at $z_0 = 0$, according to (3.18):

$$f(0) = \frac{1}{2\pi i} \int_0^{2\pi} \frac{ie^{i\theta}}{e^{i\theta}} d\theta = 1$$

That is, the net revolution of the circle around the point at the origin is 1. This is so, since the value of the integral or net direction of revolution is positive for the length of the

contour C , i.e. $2\pi i$. Since we know that the circle encloses the pole at the origin, the integral can now be interpreted as a function of the winding number. This can be done with the help of the complex Riemann Sum. Consider the curve K in $\mathbb{C}$; since the integration takes place along K as f acts on it, K will serve as the interval of integration. The complex integration $\int_K f(z)dz$ may be discretized to $R_M = \sum_i f(z_i)\Delta_i$. That is, the image of K under the mapping f is being evaluated. Every segment i has a term associated with it in the Riemann Sum, which is given by $f(z_i)\Delta_i$. This term is a complex number, connecting the beginning and end of segment i . Thus the Riemann Sum, i.e. the value of the integral is a complex number connecting the start and end points of image of K under f . It is essentially the sum of relative angular displacement of every segment, which is in line with the winding number definition that talks about the net revolution as seen in Figure 3.16. Since the result of the integral is the connecting arrow, notice that it is irrespective of the type or length of contour. It simply means that longer, convoluted contours will take a longer detour to get the same final resulting complex number of net revolutions.

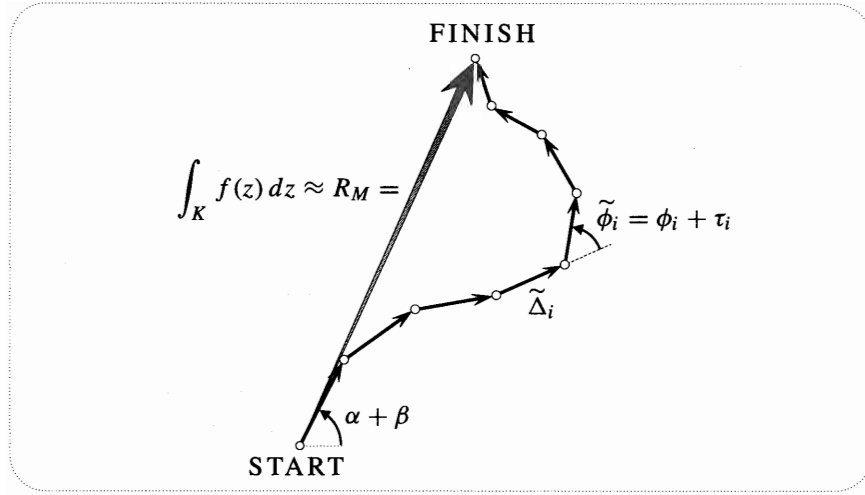


Figure 3.16: Riemann Sum and Winding number [117]

Theorem 3.6. *If a contour sweeps only through analytic points as it is deformed, the value of the integral does not change [117].*

Thus, (3.18) shows that, the value of the integral will not change if the contour γ is deformed into its interior, without crossing the pole at z_0 . Similarly, two loops may be continuously deformed into one another without ever crossing the point p , iff the two loops have the same winding number around p . Imaginably, this characteristic is what we aim to recreate for homology classes.

The Residue Formula provides a numerical method to compute the integral seen in theorem 3.5.

Theorem 3.7 (Residue). *If a holomorphic function f has a pole of order n at z_0 then*

$$Res[f(z), z_0] = res_{z_0} f = \lim_{z \rightarrow z_0} \frac{1}{(n-1)!} \left(\frac{d}{dz} \right)^{n-1} (z - z_0)^n f(z) \quad (3.19)$$

is the residue of f at z_0 [156].

Theorem 3.8 (Residue Formula). *Suppose that f is holomorphic in an open set containing a closed contour γ and its interior, except for poles at $z_1, \dots, z_n$ inside C . Then*

$$\int_{\gamma} f(z) dz = 2\pi i \sum_{k=1}^n \text{Res}[f(z), z_k] \quad (3.20)$$

for any point $z \in D$ [156].

Recall that Theorems 3.4, 3.5 and 3.8 are being examined for their relationships to winding numbers around poles. Ultimately, we would like to use some function of the winding number as the unique marker for a homology class. So, we will now show that winding numbers in conjunction with the poles, are related to the contour that encloses them via the value of the integral seen in theorem 3.4. This is explained for an example with one pole, and then with several poles. This progression shows that homology classes are applicable to the path planning problem with several constraints/obstacles.

Generally, if a closed loop does not encircle a pole, then the complex inversion mapping is holomorphic everywhere inside it. Thus, according to Cauchy's theorem, the integral vanishes. This makes sense because there is nothing to wind around. However, if the pole is encircled, then according to Cauchy's theorem, the integral no longer needs to vanish. This is because the mapping is holomorphic everywhere except at the pole, so there is something to wind around. Now, consider the Residue Formula applied to a contour K , acted on by a holomorphic function f with a pole z_0 inside the contour given by $f(z) = 1/(z - z_0)$ as demonstrated in Figure 3.17 and given by (3.21).

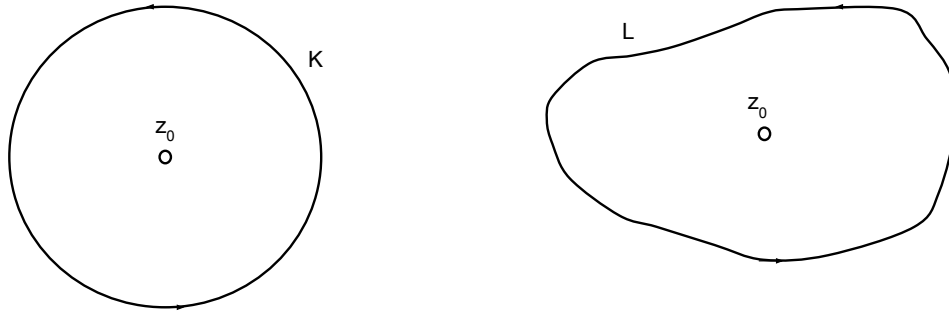


Figure 3.17: Contours surrounding one pole

$$\oint_K \frac{1}{(z - z_0)} = \oint_K f(z) = 2\pi i \text{Res}[f(z), z_0] \quad (3.21)$$

The value of the integral in (3.22) is determined by two factors. The first is the winding number of the loop itself and the second is the contribution to the winding number by the pole. This correctly assumes, that the loop winds around the pole only once, which is correct as per Figure 3.17. That is:

$$\oint_K \frac{1}{(z - z_0)} = 2\pi i v(K, z_0) \text{Res}[f(z), z_0] \quad (3.22)$$

where $v(K, z_0) = 1$.

From (3.22), the second factor is seen to be a function of the residue of f at z_0 or the amount of complex inversion contained in the decomposition of the mapping. Since this is the only part of the function that remains after the integration via the Cauchy Integral Formula, it is called the residue [117]. We have now established a connection between the pole in the Cauchy's Integral formula and the residue in the Residue Formula. Clearly, the residue occurs because of the presence of a pole and is unique to the pole (for a given holomorphic function). Different residues are thus indicative of the contour enclosing different poles. Therefore, one can now deduce that the value of the integral of f acting on K which encloses a pole at z_0 is a function of the contour's winding number (net revolution) around the pole z_0 and the residue of f at z_0 . This combination of values provides a unique marker that identifies whether a contour encloses a pole on its inside.

This combination of values provides a unique marker that identifies whether a contour encloses a pole on its inside.

This understanding holds for any contour such as loop L in Figure 3.17, since it simply means that a longer contour takes more detours before arriving at its destination. Since the winding number accounts for the net revolution around the pole, detours do not impact it. Therefore, a longer contour simply means that it meanders about its journey to its net revolution around the pole. Therefore, as seen in Theorem 3.6 what really matters is not the shape of the loop, but rather the location of the pole and the winding around it. So in light of path planning, we can now represent a constraint as a pole, and determine a marker for a class of contours around it. We will now explore the situation in which there exist multiple poles, some of which are wound around more than once. This segment of explanation focuses on how the winding number and residues are related to a contour surrounding many poles. The following explanation uses an example seen in [117]. Consider the general problem of evaluating $\oint_K f(z)dz$, where K is a general and possibly self-intersecting loop, and f possesses several singularities $s_1, s_2, \dots, s_n$ inside K .

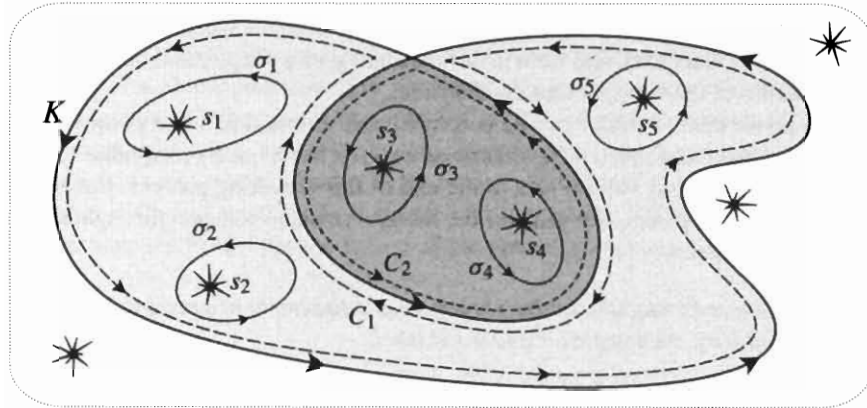


Figure 3.18: Contour surrounding several poles [117]

In this case, we know that the inside of loop K consists of two simply connected regions D_1 and D_2 . The loop K can be written as a sum of its component curves, that is: $K = \sum_j v_j C_j$. So in this case, we have $K = 1.C_1 + 2.C_2$, since K winds once round the poles in C_1 and twice around the poles in C_2 . Now, isolating the poles such that a contour σ_j was drawn surrounding the individual pole s_j , we can define the integral of that contour as:

$$I_j = \oint_{\sigma_j} f(z)dz \quad (3.23)$$

Theorem 3.6 shows that integral I_j has a **characteristic value** that does not depend on the size or shape of σ_j . Furthermore, if a simple loop contains several singularities, then the integral around that loop is the sum of I -values of the singularities that it contains. In this example:

$$\begin{aligned} \oint_{C_1} f(z)dz &= I_1 + I_2 + I_5 \\ \oint_{C_2} f(z)dz &= I_3 + I_4 \\ \Rightarrow \oint_K f(z)dz &= \oint_{C_1} f(z)dz + 2 \oint_{C_2} f(z)dz = I_1 + I_2 + I_5 + 2(I_3 + I_4) \end{aligned}$$

in which each I -value has been multiplied by the number of times K winds around the corresponding singularity. Since it is valid for arbitrary loops, it follows that this conclusion may be generalized to the following form:

$$\oint_K f(z)dz = \sum_j v(K, s_j) I_j \quad (3.24)$$

Of course, each I_j is nothing but the result of Theorem 3.8 applied to σ_j , given by $I_j = 2\pi i \text{Res}[f(z), s_j]$. Thus the **General Residue Theorem** is given by:

$$\oint_K f(z)dz = 2\pi i \sum_j v(K, s_j) \text{Res}[f(z), s_j] \quad (3.25)$$

Thus, the General Residue Theorem provides a comprehensive relationship between the integral of a non-simple contour enclosing several poles and, the winding numbers and residues of the poles. The Residue formula as seen in Theorem 3.8 is a special case of the General Residue Theorem seen in Equation (3.25), when every pole is wound around once. In the context of this thesis, we do not produce self-intersecting contours as paths. Therefore our use of the General Residue Theorem is the visualization of how several poles contribute to the integral of one simple contour acted on by f . Figure 3.19 illustrates the General Residue Theorem for two poles. This is synonymous with determining the markers of path types around constraints, which was the main motivation to delve into homology. The summary therefore, is that a contour on whose inside lies several poles, can be marked with a characteristic value as obtained via the Cauchy's Integral Formula. Alternatively, an equivalent numerical computation of the marker is obtained via the General Residue Theorem. By the

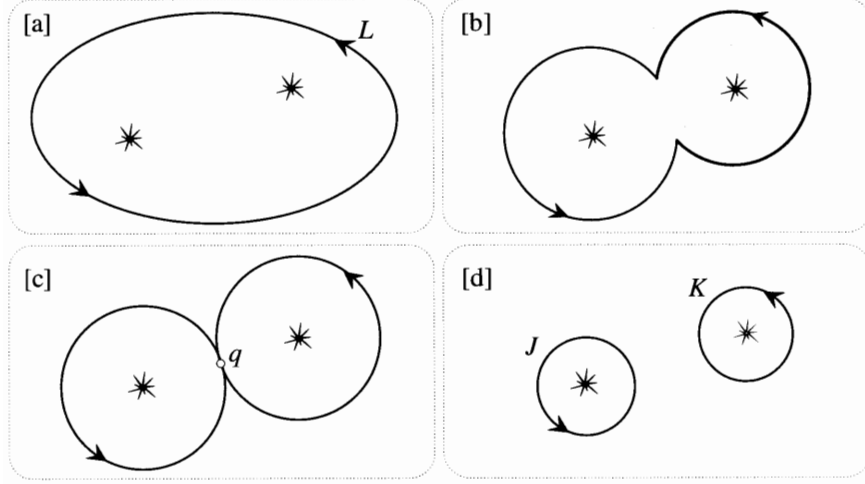


Figure 3.19: Decomposition of a contour enclosing two singularities [117]

deformation theorem, only those contours with the same value obtained via the Cauchy's Integral Formula/ General Residue Theorem may be deformed continuously. That is the deformation presumes holomorphicity in the area of a deformation, which means the deformation can never cross poles. Therefore, we can use this as a measure of homology. Since homology classes are homotopy invariants, we can use it for line segments. In light of this connection between contours and homology class markers, we shall borrow the definition of homologous paths from [15] to indicate a relationship between contours in a homotopic class and those in the homology class, around a constraint/pole or obstacle. It is important to note that the use of Complex Analysis is simply as a tool, that is specifically convenient for 2-dimensional planar surfaces or 2-manifolds. Other tools may be explored on a per-application basis, but is not the focus of the thesis.

Definition 3.24 (Homologous Paths). *Two paths τ_1 and τ_2 connecting the same start and end coordinates, $\mathbf{p}^{start}$ and $\mathbf{p}^{goal}$, respectively, are homologous if and only if τ_1 together with τ_2 (the latter with opposite orientation) forms the complete boundary of a 2-dimensional manifold embedded in $\mathcal{C}$ and not containing or intersecting the constraint space $\mathcal{P}^{constraint}$.*

Geodesics, homotopic, and homologous paths are directly reliant on the nature of the underlying topology. That is, the nature of the PPM influences the kind of geodesics produced and whether paths can be transformed using homotopy and homology. This section defined the fundamental concepts required to characterize the PPS and its associated sub-spaces, along with tools to transform between different spaces. These definitions enable formulating the shortest path on a non-Euclidean PPS, called a geodesic. Concepts of homology and homotopy further aid in analyzing an existing path and finding equivalent paths in the same PPS. Together, all the aforementioned concepts are used to propose the CFDMPP to solve the PPP in hand.

3.2 Methodology for Algorithm Formulation: Constraint-Free Discretized Manifolds-based Path Planner (CFDMPP)

This section discusses the proposed **CFDMPP**. A high level view of the planner is revealed in Algorithm 3.1. As can be seen, the algorithm takes four main steps to achieve the goal, each of which is described in detail in a designated subsection. Line 1 constructs a **Constraint-Free Manifold (CFM)** on the map of the **PPS**, by accounting for locations and bounds of the **SCs**. The **PPSs** associated with robotic systems are usually not purely Euclidean spaces. Thus, to represent the connectivity of the space as accurately as possible, a manifold representation of the **PPS** is formulated. The manifold is constructed such that it is free of **SCs**, so that the shortest path generated on it, is constraint-free. Line 2 discretizes the **CFM** from Line 1 to produce a graph based representation of the manifold. A graph traversal algorithm can then be used to compute the shortest path from $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$. Line 3 allows the generated constraint-free path to be transformed to an alternative **PPS**, if necessary.

The path following of the generated constraint-free path is out of the scope of the thesis. At Line 6, the algorithm goes into path re-planning mode to circumvent **DCs**, should any prevent the robot from following the global path computed earlier (in the offline stage). Two kinds of situations can render an existing path invalid. The first is the presence of **DCs** through moving obstacles, the sudden appearance or dislocation of an object (falling item) or human beings. The second situation involves the requirement of an alternative path, based on a human decision after assessing current conditions of the path. For example, a fluid spill on the floor that renders the original path invalid or the anticipation of human activity along the planned path. These situations constitute valid triggers for path re-planning.

Algorithm 3.1 **CFDMPP**

Input:

- Map of path planning environment
- Bounds of static constraints
- Source and target locations in **PPS**: $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$

Output:

- Shortest constraint-free path between $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$

- 1: Formulate a **CFM** on the path planning environment ▷ Section 3.2.1
 - 2: Discretize the **CFM** and generate a path from $\mathbf{p}^{\text{start}}$ to $\mathbf{p}^{\text{goal}}$ ▷ Section 3.2.2
 - 3: Transform the obtained path to another space, if necessary ▷ Section 3.2.3
 - 4: Robot starts following the computed path

 - 5: **while** Existing path is rendered inaccessible **do** ▷ Reactive (online) mode begins
 - 6: Modify path to avoid **DC** ▷ Section 3.2.2
 - 7: **end while**

 - 8: **return** followed path
-

This section is broken down into four subsections, such that each subsection reflects

the same sequence of steps seen in Algorithm 3.1. Thus, CFM formulation is detailed in Section 3.2.1, discretized path planning is described in Section 3.2.2, path transformation is seen in Section 3.2.3, and path modification due to DCs is seen in Section 3.2.2.

3.2.1 Formulation of a Constraint-Free Manifold (CFM)

The first stage of the CFDMPP algorithm is the formulation of a CFM as a viable constraint-free space on which path planning can be performed. This section formulates the CFM for a given PPP. This is necessary since the PPS in which the problem is originally defined usually contains constraint regions that must be avoided. As such, the initial PPS is converted to a PPM. To that end, one must define all the necessary parameters that can define the PPM. The following subsections explain the steps undertaken to infer information specific to the PPP at hand. The CFDMPP assumes that the following information can either be inferred or obtained via pre-processed data about the path planning environment. Along the way, Stationary Robots (SRs), such as the OCM and CCM; and Mobile Robots (MRs), such as the UGV, UAV and Unmanned Underwater Vehicle (UUV), are used as examples to clarify the steps described.

3.2.1.1 Primary and Secondary Path Planning Spaces (PPSs)

Robotic systems may have multiple PPSs, depending on what aspects of the system can be controlled to produce a desired output. When more than one potential PPS exists, one of them is designated as the Primary Path Planning Space (PPPS). The remaining alternative PPSs are understood to be the secondary equivalents of the PPPS, referred to as the Secondary Path Planning Space (SPPS). Let $|\text{PPSs}|$ and $|\text{SPPSs}|$ denote the numbers of PPSs and SPPSs, respectively. Then, $|\text{PPSs}| = |\text{SPPSs}| + 1$ (the one is for the PPPS). SPPSs are indexed as $\text{SPPS}_1, \text{SPPS}_2, \dots, \text{SPPS}_{|\text{SPPSs}|}$.

For example, in the case of a MR, one would consider its $\mathcal{W}$ as the main and only PPS. Thus, $\text{PPPS} = \mathcal{W}$, $|\text{SPPSs}| = 0$, and $|\text{PPSs}| = 1$. As an alternative example, a SR, such as a CCM or an OCM, could have two potential PPSs, with the PPPS being its $\mathcal{C}$ -space and an SPPS being $\text{SPPS}_1 = \mathcal{W}$. Hence, $\text{PPPS} = \mathcal{C}$, $|\text{SPPSs}| = 1$, and $|\text{PPSs}| = 2$.

3.2.1.2 Dimensions of Path Planning Spaces (PPSs)

As noted in Definition 3.4, the PPSs can be defined as topological spaces, and their dimensions play an essential role in the kind of maps that can be applied to them. Therefore, the dimensions of all associated PPSs need to be determined. The dimensions of the PPPS and the SPPS vary with the kind of robot and application at hand. For example, a UAV or a UUV can have a 3-dimensional $\mathcal{W}$. Contrarily, a UGV has a 2-dimensional $\mathcal{W}$. Considering it to be its primary and unique PPS, and denoting the dimension of PPS_i of by d^{PPS_i} , we get $d^{\text{PPS}_1} = d^{\text{PPPS}} = 2$. Likewise, if we consider a 2LPM, for instance, with $\mathcal{C}$ and $\mathcal{W}$ as $\text{PPS}_1 \equiv \text{PPPS}$ and $\text{PPS}_2 \equiv \text{SPPS}$, respectively, then $d^{\text{PPS}_1} = d^{\text{PPPS}} = d^{\text{PPS}_2} = d^{\text{SPPS}} = 2$.

3.2.1.3 Bounds of Path Planning Spaces (PPSs)

The operational limits of all PPSs are accounted for in this step to define the extent of each PPS available. Understanding the bounds of operable space in every PPS will later enable defining either the PPS or a subset of it as a manifold. Using (3.1), (3.2), and (3.8), one can define the bounds of any PPS $\mathcal{P}$. Generally however, the bounds may take a different form than simply satisfying an upper and lower bounds, and can require satisfying a set of constraints. This is especially so when the bounds of a PPS are determined with the help of a mapping from another PPS. The mapping may cause the resulting bounds to be no longer Euclidean, and thus bounds, as defined by (3.8), may hold insufficient information about the bounds of the PPS.

For example, the PPS of a UGV may be easily defined in the form of (3.2) as

$$\mathcal{W} = (\{x, y\} | \{x, y\} \in (x^{\min}, x^{\max}) \times (y^{\min}, y^{\max}))$$

while that of a 2LPM can be conveniently defined in the form of (3.1) as

$$\mathcal{C} = (\{q_1, q_2\} | \{q_1, q_2\} \in (q_1^{\min}, q_1^{\max}) \times (q_2^{\min}, q_2^{\max}))$$

for some given bounds

$$\begin{aligned} (x^{\min}, x^{\max}) &= (0, 10) \text{ m} & (y^{\min}, y^{\max}) &= (0, 10) \text{ m} \\ (q_1^{\min}, q_1^{\max}) &= (0, 360^\circ) & (q_2^{\min}, q_2^{\max}) &= (0, 180^\circ) \end{aligned}$$

However, it is less obvious to define the limits of the 2LPM's SPPS $\mathcal{W}$, for instance. When there exist multiple PPSs, one can typically define the limits of at least one of them based on operational limits. Nonetheless, to define the bounds of the remaining PPSs, a mapping is required. This is discussed in Section 3.2.1.4.

3.2.1.4 Mapping Between Path Planning Spaces (PPSs)

When a PPP has multiple equivalent PPSs, it implies that the PPS can be transformed to any of the other SPPSs, and vice versa, under the condition that such transformation mappings exist. This transformation is what enables transforming the bounds of the PPS to those of a SPPS. We will denote the continuous mapping (if it exists) from PPS_i to PPS_j , $i, j \in \{1, 2, \dots, |\text{PPSs}|\}$, as $f_{\text{PPS}_i}^{\text{PPS}_j} : \text{PPS}_i \rightarrow \text{PPS}_j$. Obviously, the existence of such a mapping does not guarantee the existence of its inverse $(f_{\text{PPS}_i}^{\text{PPS}_j})^{-1}$. It is also worth specifying that the mapping of a point on the boundary of a PPS may not be on the boundary of the mapped space. Consequently, the mapping of a Euclidean PPS (in the form of (3.8)) may not be Euclidean.

For instance, going back to the 2LPM example with $\mathcal{C}$ and $\mathcal{W}$ being the PPS and SPPS, respectively, then $f_{\text{PPS}}^{\text{SPPS}} \equiv f_{\mathcal{C}}^{\mathcal{W}} : \mathcal{C} \rightarrow \mathcal{W}$, which is the forward kinematic mapping. Despite the simplicity of this type of robots, one does not immediately recognize that the Euclidean $\mathcal{C}$ -space does map to a Euclidean $\mathcal{W}$ -space (which is subset of the Euclidean space), as illustrated in Figures 2.22 and 2.23. A non-planar SR will typically have a $\mathcal{W}$ -space, which may or may not be globally expressed as a subset of $\mathbb{R}^n$.

3.2.1.5 Information About Static Constraints (SCs)

The environment may contain constraints, as defined in Definition 3.5, which have to be avoided by the path planner. Thus, their information must be accounted for in the algorithm. This section deals with SCs, i.e., constraints whose presence is known apriori. Five parameters are used to process SCs and they are: classification of SCs, the total number of SCs, the PPSs native to each SC, the span of the SCs and finally, how several SCs can be grouped together into one set. These five parameters defining the constraints are derived through the following five pre-processing steps, which are performed by the user, unless otherwise indicated by a proposed algorithm. As done before, the examples of the 2LPM and UGV are used to illustrate each step.

1. Classification of SCs present in the PPP:

The SCs are classified into the categories defined in Definition 3.5, i.e., the robot's mechanical limits (ML), singularities (SG), no-go zones (NG) and obstacles (OB). Note that at this juncture, all possible static constraints are accounted for, irrespective of which of the multiple PPSs they may lie in. Common robotic systems can experience several regions that are not compatible for path planning. Depending on the specific PPP, one can classify those regions of inoperability into the aforementioned categories of constraints. the classification of constraints enables a better understanding of the kind of inoperable regions that the PPP is dealing with.

For example, in the case of the UGV, the most common kind of constraints is in the form of stationary objects blocking its free motion in the environment. Considering $PPPS = \mathcal{W}$, then the SCs in this case are classified as obstacles (OB).

In the second example of the 2LPM, assume that it contains inoperable regions only in the form of joint limits, which are obviously defined in its $PPPS = \mathcal{C}$. Additionally, assume that there exist prohibited regions of access in its $SPPS = \mathcal{W}$. In this case, the SCs are classified as the union of mechanical limits (ML) defined by the robot joint limits in the $PPPS$ and the no-go zones (NG) represented by the prohibited regions of access in the $SPPS$.

2. The total number of SCs:

After classifying the SCs into their respective categories, the total number of SCs in each category is determined. This is the cardinality of each category set, which are denoted by n_{ML} , n_{SG} , n_{NG} , and n_{OB} . We will also denote the total number of SCs by n_{SC} .

Applying this pre-processing step to the UGV, for instance, and assuming that there exist two obstacles in its $PPS = PPPS = \mathcal{W}$, then we get

$$n_{ML} = 0 \quad n_{SG} = 0 \quad n_{NG} = 0 \quad n_{OB} = 2 \quad \Rightarrow n_{SC} = 2$$

Likewise, applying the same principle on the 2LPM, for instance, with two mechanical limit regions, two singularities, and one no-go zone, yields

$$n_{ML} = 2 \quad n_{SG} = 0 \quad n_{NG} = 1 \quad n_{OB} = 0 \quad \Rightarrow n_{SC} = 3$$

3. Identifying the native PPS of each SC:

This step explicitly recognizes the PPS in which every SC is defined. Each SC is only originally defined in one particular PPS (i.e., either the PPS or any of the SPPSs), which is known as the native space of that constraint (see Figure 3.20). By default, such a constraint may have no equivalence in another PPS. For it to have one, a bidirectional mapping is required, as shown in Figure 3.21 and as shall be formally defined in Section 3.2.1.4. The mapping is bidirectional in the sense that it is invertible at least locally, on a restriction of the space. Determining the native PPS of each SC helps defining the required maps to correlate the different PPSs. Let $n_{SC}^{PPS_i}$ be the number of SCs in $PPS_i \in \{PPS_1, PPS_2, \dots, PPS_{|PPS|}\}$. Likewise, let $n_{ML}^{PPS_i}$, $n_{SG}^{PPS_i}$, $n_{NG}^{PPS_i}$, $n_{OB}^{PPS_i}$, be the (non-negative) number of SCs of their respective category in PPS_i . Therefore,

$$n_{SC} = \sum_{i=1}^{|PPSs|} n_{SC}^{PPS_i} = n_{SC}^{PPPS} + \sum_{i=1}^{|SPPSs|} n_{SC}^{SPPS_i} \quad (3.26)$$

$$n_{SC}^{PPS_i} = n_{ML}^{PPS_i} + n_{SG}^{PPS_i} + n_{NG}^{PPS_i} + n_{OB}^{PPS_i} \quad (3.27)$$

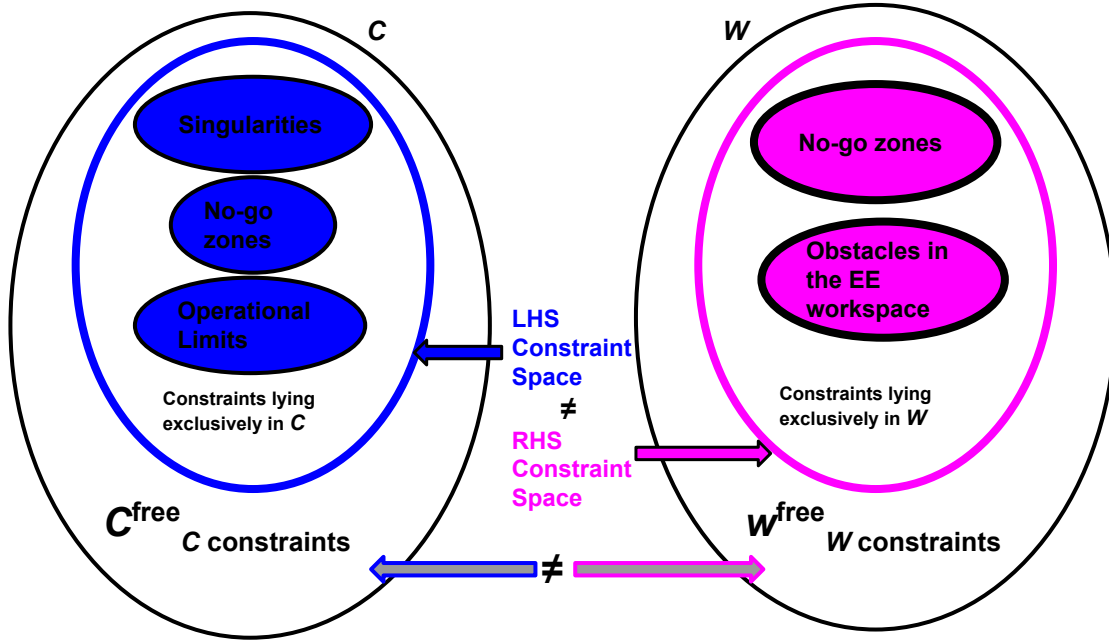


Figure 3.20: Each uniquely defined static constraint may lie in any of the PPSs identified for a robotic system; therefore, that PPS becomes the native space of the constraint, and the constraint has no meaning in another PPS.

Now, one can correlate the above, with the examples developed so far. Since the UGV has only one PPS and it is faced with only two obstacles in its workspace, it can be concluded that $n_{OB}^{PPS_1} = n_{OB}^{PPPS} = 2$ and $n_{SC} = n_{SC}^{PPS_1} = n_{SC}^{PPPS} = 2$, which is consistent with the result found in Step 2. As for the 2LPM example, the $PPPS = \mathcal{C}$ is the native PPS for the

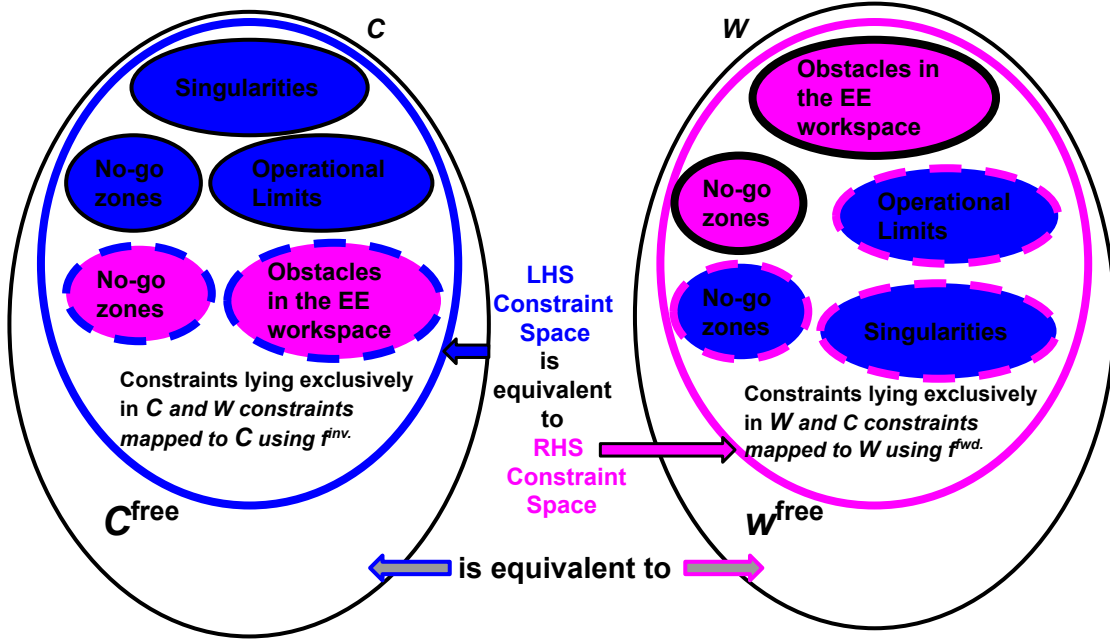


Figure 3.21: In order for the original constraint to have an equivalence in another PPS, a mapping is required. Only after mapping the original constraints in their native space to the alternative PPS, is equivalence of constraints in both spaces established.

two mechanical limits ($n_{ML} = n_{ML}^{PPPS} = 2$). As a result, $n_{SC}^{PPS_1} = n_{SC}^{PPPS} = n_{ML}^{PPPS} = 2$. In addition, the robot has one no-gone zone which lies exclusively in the $SPPS = \mathcal{W}$. Hence, $n_{NG}^{PPS_2} = n_{NG}^{SPPS} = 1$, and so $n_{SC} = n_{SC}^{PPPS} + n_{SC}^{SPPS} = 2 + 1 = 3$, which is also consistent with Step 2. Both cases are illustrated in Figure 3.22. In the example of the UGV, there exists only one PPS which is the PPPS. Therefore, all constraints are native to that space.

4. The bounds of SCs:

The limits or size of every constraint is determined in the native PPS. The SCs may be defined in a manner similar to defining the bounds of the PPSs by using (3.8). They may also be defined as an equation or simply as a point. This step of determining the expanse of each SC is essential for two reasons. First, if there exist multiple PPSs, then in order to map the constraint space to the alternative PPS, the bounds of the original definition of the constraints are required. Second, even if there exists only one PPS, then defining the bounds of the static constraints enables defining the CFM in the subsequent steps.

Continuing with the examples from previous steps, it was seen that the UGV requires its $n_{SC} = n_{SC}^{PPPS} = n_{OB}^{PPPS} = 2$ obstacles to be defined. Denote the two obstacles as OB_1 and OB_2 . Using (3.8) as a general guideline, one can define OB_1 and OB_2 as:

$$OB_1 = \{(x, y) \mid (x, y) \in [4 \text{ m}, 5 \text{ m}] \times [4 \text{ m}, 5 \text{ m}]\} \quad (3.28a)$$

$$OB_2 = \{(x, y) \mid (x, y) \in [3 \text{ m}, 6 \text{ m}] \times [3 \text{ m}, 6 \text{ m}]\} \quad (3.28b)$$

As seen in Step 3, each SC is defined with respect to its native PPS. That is, the values seen as bounds of OB_1 and OB_2 in (3.28a) and (3.28b) are defined with respect to the

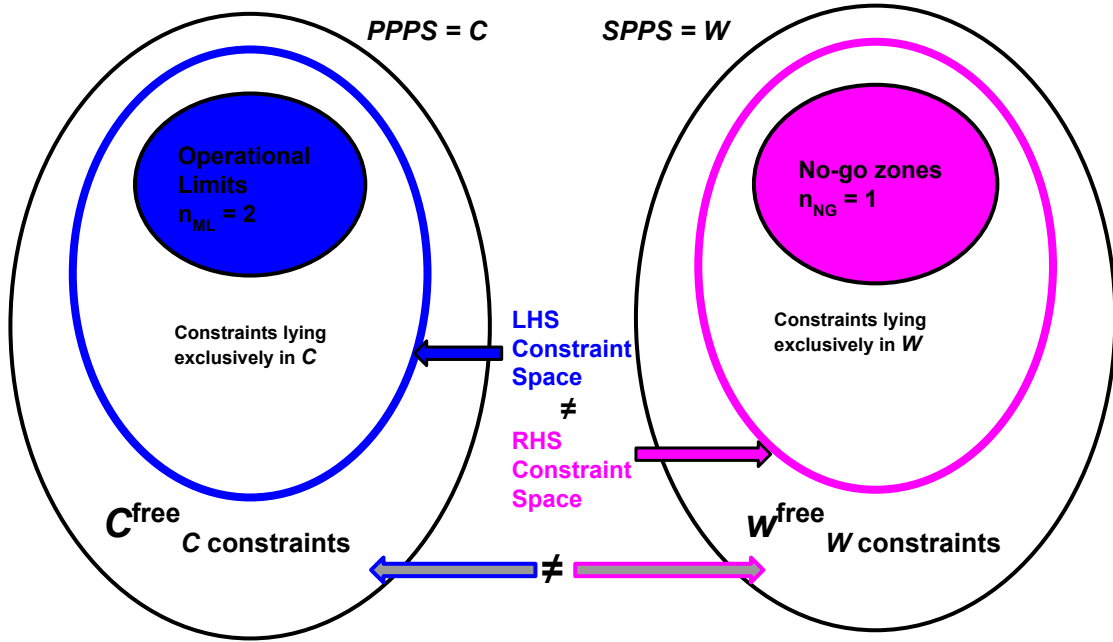


Figure 3.22: The 2LPM example as seen in step 3. The native PPS for static constraints present in the PPP is seen here. Note that, the constraints in their originally defined form do not have an equivalence in the alternative PPS.

origin of its native $PPS=W$. Naturally, they inherit the same units as the native PPS , which in this case is metres. This applies to all SCs whose bounds are defined further.

As for the 2LPM example, the bounds for its SCs are defined similarly (recall that $n_{SC} = 3$ in this case). The $n_{ML}^{PPPS} = 2$ mechanical limits, ML_1 and ML_2 , and the $n_{NG}^{SPPS} = 1$ no-go zone NG_1 (respecting the bounds of the $PPSs$ given in Section 3.2.1.3), can be expressed as

$$ML_1 = \{(q_1, q_2) \mid (q_1, q_2) \in X \times (0^\circ, 1^\circ]\} \quad (3.29a)$$

$$ML_2 = \{(q_1, q_2) \mid (q_1, q_2) \in X \times (0^\circ, 2^\circ]\} \quad (3.29b)$$

$$NG_1 = \{(x, y) \mid (x, y) \in [0 \text{ m}, 2 \text{ m}] \times [1.9 \text{ m}, 2 \text{ m}]\} \quad (3.29c)$$

The “don’t care conditions” denoted by “ X ” refers to the irrelevance of the value of the corresponding variable. For instance, the interpretation of ML_1 is that q_2 cannot be in the interval $(0^\circ, 1^\circ]$ regardless of the value of q_1 . The rest of the SC bounds can be interpreted similarly. Note that it’s perfectly fine for the SC bounds to overlap, as in the case of ML_1 and ML_2 , for example.

It is well known that the 2LPM has two singularities at

$$SG_1 = \{(q_1, q_2) \mid (q_1, q_2) \in X \times 0^\circ\}$$

$$SG_2 = \{(q_1, q_2) \mid (q_1, q_2) \in X \times 180^\circ\}$$

However, since they are not within the **PPS** bounds specified in Section 3.2.1.3, they are dropped out of the example.

5. Constructing the disjoint constraint sets:

Once the bounds for all n_{SC} **SCs** have been established, one now computes the **Disjoint Constraint Sets (DCSs)**. The term **Constraint Set (CS)** is explained first, following which the term **DCS** is explained. After the explanation of how and why the computation is performed, the algorithm to determine the **DCSs** is detailed.

A **CS**, $\text{CS}_r^{\text{PPS}_i}$, associated to a specific PPS_i , is a union of constraints satisfying two requirements. First, all constraints in the set must belong to PPS_i . Second, each constraint of the set must be located within a certain (user-defined) distance/proximity Δ_{CS} from all the other constraints in the set. To that end, one needs to define a function $f(\text{SC}_i, \text{SC}_j)$ to measure the proximity Δ_{CS}^{ij} between two **SCs** SC_i and SC_j in the same constraint set.

$$\Delta_{\text{CS}}^{ij} = f(\text{SC}_i, \text{SC}_j) \quad (3.30)$$

where $i, j \in \{1, \dots, n_{\text{SC}}^{\text{PPS}_i}\}$ and δSC_i denotes the boundary of SC_i . There are many possible ways in which such a function may be defined depending on the user preference. In this work, we use the minimum Euclidean distance between the boundaries of the two **SCs**. The distance is considered zero if the two **SCs** are touching or overlapping. Such a choice of proximity measure is quite modular. It is tailored to suit every **PPP**, its associated spaces and the bounds of the **SCs**. For example, should SC_j be a point, then it employs a point-set difference instead of computing the proximity from the boundary of both sets. In summary, a **CS**, $\text{CS}_r^{\text{PPS}_i}$, corresponding to PPS_i can be formally defined as

$$\text{CS}_r^{\text{PPS}_i} = \left\{ \text{SC}_j \mid \text{SC}_j \in \text{PPS}_i, \text{ and } \forall \text{SC}_j, \text{SC}_k \in \text{CS}_r^{\text{PPS}_i}, \Delta_{\text{CS}}^{jk} \leq \Delta_{\text{CS}} \right\}$$

Let $l_{uk}^{\min}$ and $l_{uk}^{\max}$ be the respective lower and upper limits of the k th dimension of $\text{SC}_u \in \text{CS}_r^{\text{PPS}_i}$ for an n -dimensional PPS_i . If $\text{CS}_r^{\text{PPS}_i} = \{\text{SC}_1, \text{SC}_2, \dots, \text{SC}_m\}$, then $\text{CS}_r^{\text{PPS}_i}$ is an n -dimensional Cartesian space in PPS_i , where the lower and upper limits of the k th dimension are, respectively

$$\min(l_{1k}^{\min}, l_{2k}^{\min}, \dots, l_{mk}^{\min}) \quad \max(l_{1k}^{\max}, l_{2k}^{\max}, \dots, l_{mk}^{\max})$$

Such limits, for $k = 1, \dots, n$, are what define the boundary $\delta\text{CS}_r^{\text{PPS}_i}$ of $\text{CS}_r^{\text{PPS}_i}$. The set $\text{CS}_r^{\text{PPS}_i}$ may contain **SCs** of different categories as per Definition 3.5.

A **CS** can be viewed as a union of constraints that involves the least amount of inclusion of constraint-free **PPS**. It is a building component of the **CO**-space. Let the constraint-free region that is included by virtue of the union of **SCs** be denoted by Δ_{CS} . Ideally, the value of Δ_{CS} is 0, to signify no inclusion of constraint-free space as part of the **CS**. However, non-zero user defined values can be used, to permit the inclusion of narrow, constraint-free passages within the **CS**. This flexibility allows users to decide whether narrow areas that lie in between **SCs**, are truly navigable. While such passages are constraint-free, they may fail to allow the robot to physically traverse the passage. In that case, it is counter effective to account that space as a navigable, constraint-free space by the path planning

algorithm. Additionally, different Δ_{CS} values may be adopted for different PPSs. Hence, to account for this possibility of varying needs, the user-defined Δ_{CS} is introduced as part of the definition of the CS.

A DCS, $\text{DCS}_r^{\text{PPS}_i}$, associated with PPS_i , is a complete CS in PPS_i , in the sense that it includes all the SCs of PPS_i which satisfy the proximity requirement. In other words, if $\text{SC}_k \in \text{DCS}_r^{\text{PPS}_i}$, then $\nexists \text{SC}_j \in \text{PPS}_i$, such that $\Delta_{\text{CS}}^{jk} > \Delta_{\text{CS}}$. Let $\text{DCSs}^{\text{PPS}_i}$ be the set of all DCSs corresponding to PPS_i and DCSs be the set of all DCSs of all PPSs. Then,

$$\text{DCSs} = \bigcup_{i=1}^{|\text{PPSs}|} \text{DCSs}^{\text{PPS}_i} \quad (3.31)$$

$$|\text{DCSs}| = |\text{DCSs}^{\text{PPPS}}| + \sum_{i=1}^{|\text{SPPSs}|} |\text{DCSs}^{\text{PPS}_i}| \quad (3.32)$$

Algorithm 3.2 outlines the iterative procedure to compute $\text{DCSs}^{\text{PPS}_i}$. $\alpha(j)$ denotes the j th element of the set α .

Algorithm 3.2 Computing $\text{DCSs}^{\text{PPS}_i}$

Input:

$\alpha = \{\text{SC}_1, \text{SC}_2, \dots\}$

▷ Set of all SCs in PPS_i

Δ_{CS}

▷ User-defined proximity threshold for PPS_i

Output:

$\text{DCSs}^{\text{PPS}_i}$

▷ Set of all DCSs in PPS_i

```

1:  $k \leftarrow 0$                                      ▷ Number of DCSs
2: while  $\alpha \neq \emptyset$  do
3:    $k \leftarrow k + 1$ 
4:    $\text{DCS}_k \leftarrow \alpha(1)$                          ▷ DCS gets first SC available
5:    $\alpha \leftarrow \alpha \setminus \alpha(1)$ 
6:    $\text{SC}_{\text{remove}} \leftarrow \emptyset$ 
7:   for  $j = 1$  to  $|\alpha|$  do                         ▷ Compare with remaining CSs
8:     Compute the proximity  $\Delta$  between  $\text{SC}_j$  and  $\text{DCS}_k$  using (3.30)
9:     if  $\Delta \leq \Delta_{\text{CS}}$  then
10:       $\text{DCS}_k \leftarrow \text{DCS}_k \cup \alpha(j)$ 
11:       $\text{SC}_{\text{remove}} \leftarrow \text{SC}_{\text{remove}} \cup \alpha(j)$ 
12:    end if
13:  end for
14:   $\alpha \leftarrow \alpha \setminus \text{SC}_{\text{remove}}$ 
15: end while
16: return  $\text{DCSs}^{\text{PPS}_i} = \{\text{DCS}_1, \text{DCS}_2, \dots, \text{DCS}_k\}$ 

```

We now illustrate the concept of DCSs on the UGV and 2LPM examples. Recall that the UGV has one PPS, $\text{PPS}_1 = \text{PPPS} = \mathcal{W}$, and 2 SCs OB₁ and OB₂ ($n_{\text{SC}}^{\text{PPS}_1} = 2$) whose bounds are specified in (3.28a) and (3.28b). In this example, we take $\Delta_{\text{CS}} = 1$ m,

meaning that **SCs** within 1 m or less of each other are grouped in the same **DCS**. This is an exaggerated value but simple enough for clarifying the procedure. Line 1 starts with $\alpha = \{\text{OB}_1, \text{OB}_2\}$. Then, it begins forming the first **DCS** (corresponding to **PPS**₁), **DCS**₁, by moving the first element of α to **DCS**₁, which makes

$$\text{DCS}_1 = \{\text{OB}_1\} \quad \alpha = \{\text{OB}_2\}$$

After that, the algorithm iterates over the remaining elements of α . Each element whose proximity to **DCS**₁ is less than or equal to Δ_{CS} is moved to α . As it is the case for OB_2 , the algorithm terminates with

$$\text{DCS}_1 = \{\text{OB}_1, \text{OB}_2\} \quad \alpha = \emptyset$$

Since the **UGV** has only one **PPS**, there is no need to apply the algorithm to process any more **PPSs**. Consequently, $\text{DCSs} = \text{DCSs}^{\text{PPS}_1} = \{\{\text{OB}_1, \text{OB}_2\}\}$ and $|\text{DCSs}| = |\text{DCSs}^{\text{PPS}_1}| = 1$. This means, that the total number of disjoint regions to be avoided during path planning for this example is one.

In the same fashion, Algorithm 3.2 can be applied to the **2LPM** example. As seen in (3.29), the robot has a total of three **SCs** and two **PPSs**. Hence, Algorithm 3.2 is applied to each **PPS**. Let $\Delta_{\text{CS}} = 0$ for the **PPSs**. Applying Algorithm 3.2 on the **PPS** $= \mathcal{C} = \text{PPS}_1$, which contains two **SCs**, $\{\text{ML}_1, \text{ML}_2\}$, yields one **DCS**,

$$\text{DCS}_1 = \{\text{ML}_1, \text{ML}_2\}$$

and so, $\text{DCSs}^{\text{PPS}_1} = \{\{\text{ML}_1, \text{ML}_2\}\}$ whose cardinality is 1. Now, applying the algorithm on the **SPPS** $= \mathcal{W} = \text{PPS}_2$, which contains only one **SC**, $\{\text{NG}_1\}$, leads to one **DCS**,

$$\text{DCS}_1 = \{\text{NG}_1\}$$

Therefore, $\text{DCSs}^{\text{PPS}_2} = \{\{\text{NG}_1\}\}$ whose cardinality is 1. Note that for **PPSs** with only one **SC**, the value of Δ_{CS} is irrelevant. In aggregate,

$$\text{DCSs} = \text{DCSs}^{\text{PPS}_1} \cup \text{DCSs}^{\text{PPS}_2} = \{\{\text{ML}_1, \text{ML}_2\}, \{\text{NG}_1\}\}$$

whose cardinality is 2. This means that, in total, there are 2 disjoint regions to be avoided by the path planner.

3.2.1.6 Potential Dynamic Constraints (DCs)

The **DCs** are defined to be constraints whose presence or absence in the environment is not known apriori. Examples of such **DCs** include falling objects or a sudden encounter with an unknown agent such as a human or another robot. Naturally, information about potential **DCs** is not known apriori and is obtained on the fly using an appropriate sensing algorithm. The sensing algorithm is not part of the **CFDMPP**, so some basic assumptions about the sensor and the processing algorithm are mentioned here. The **PPS** is assumed to have a sensor system, with sensors such as cameras, to detect **DCs**.

The sensors mounted to the robot typically have a limited sensing range and the processing algorithm can perform object recognition and depth estimation, producing a bounding box around the object. This provides an estimate of the size of the **DC** to avoid and its location with respect to the robot. The limited sensing range allows this stage of the path planning to be reactive, i.e., the original path is re-planned at this stage, purely as a reaction to a **DC**. The centralized nature of the sensing system also allows for global path re-planning, as opposed to simply being reactive. Obtaining any additional attributes about potential **DCs** is dependent on the sensing algorithm used.

Now, we view potential **DCs** in the context of triggering path re-planning. In this context, **DCs** comprise of those constraints that are not initially accounted for during the construction of the **CFM**. These constraints can be grouped into two kinds for the purposes of the **CFDMPP**. The first group may be any of the following. Some may take the form of **SCs** whose numbers and/or locations are unknown during **CFM** construction or are subject to change until before path planning commences. Examples include the presence of **DCs** through moving obstacles, the sudden appearance or disappearance of objects or human beings. Other examples include movable shelving units and crates, whose locations can change. Therefore, they are not accounted for as constant **SCs** to account during **CFM** phase. The second group of **DCs** involve anticipatory **DCs**, with the intention of triggering path re-planning. Examples include situations that necessitate an alternative path, such as a fluid spill on the floor that renders the original path invalid or the anticipation of human activity along the planned path.

This section thus far aids in completing the definition of the **PPS** in Section 3.2.1.7, by anticipating the **PPS** in which potential **DCs** may be located. **DCs** will be re-introduced after the construction of the **CFM** before path planning commences.

3.2.1.7 Choosing the Appropriate Space Where to Commence Path Planning

Section 3.2.1.1 shows that a **PPP** can have several **PPSs**, as seen through the **2LPM**, for example. Subsequently, Sections 3.2.1.3 to 3.2.1.6 obtain information about the **PPSs** and of constraints that lie in the **PPSs**. With such information at hand, one has the ability to choose the appropriate **PPS** where to commence path planning. The **PPSs** will later be constructed as **PPMs**, thus allowing for transformation of paths amongst the manifolds. Therefore, the choice of the **PPS** to work in, may seem unnecessary. However, it is essential and advantageous to do so, because of three reasons. But before that, further notation is defined here. We previously divided the **PPSs** into a **PPPS** and none or more **SPPSs**, where the former refers to the most commonly used **PPS**, without any previous evaluation of the **PPS** with respect to formulating a **PPM** for it. This section provides evaluative factors to choose the appropriate **PPS** for manifold formulation. By doing so, only one **PPS** is adopted as the ideal choice. To that end, we denote the **PPS** chosen for path planning as **PPS_{chosen}** and any of the rest of **PPSs** as **PPS_{alt}** (alternative **PPS**). The set containing all the alternative **PPSs** is denoted by **PPS_{alt}**. Now, the independent evaluative factors and reasons for their use are provided, followed by their order of application.

1. First Evaluative Factor:

The first factor involves the choice of the **PPS** based on the **PPS** in which the robot's physical tasks require to be accomplished. In a **PPP** with multiple **PPSs**, the robotic system typically requires to perform tasks in only one **PPS**, say $\text{PPS}_{\text{tasks}}$. Then, $\text{PPS}_{\text{tasks}}$ is a natural choice to commence path planning, since the resulting path may directly affect the performance of the robotic application at hand. Should one choose the **PPS** where the control inputs of the robot lie, the justification is that the path in the control space may be directly applied to the robot. This factor is application- and user-dependent.

Consider the example of the **UGV**. Since $\mathcal{W} = \text{PPPS}$ is the only **PPS**, there is no further evaluation to choose the appropriate **PPS** to work with. Contrarily, in the **2LPM** example, there exist two **PPSs**, which are the $\text{PPPS} = \mathcal{C}$ and $\text{SPPS} = \mathcal{W}$. The space in which the end-effector physically accomplishes tasks is $\text{SPPS} = \mathcal{W}$. Thus, based on the first evaluation factor, $\text{SPPS} = \mathcal{W}$ should be the **PPS** where to commence path planning.

2. Second Evaluative Factor:

The second factor evaluates the suitability of the **PPS** based on the number of **DCSs** it contains. This factor favors choosing the **PPS** in which the maximum number of **DCSs** is located. This is so, because of the process used to formulate the **PPM**. Once a suitable **PPS** is chosen as the $\text{PPS}_{\text{chosen}}$, it is formulated into a constraint-free **PPM**. The latter requires the $\text{PPS}_{\text{chosen}}$ to be completely defined, as will be seen in Section 3.2.1.8. The complete definition of the $\text{PPS}_{\text{chosen}}$ involves transforming the **DCSs** present in PPS_{alt} to $\text{PPS}_{\text{chosen}}$. The transformation is accomplished by applying the mapping between $\text{PPS}_{\text{chosen}}$ to PPS_{alt} , to the **DCSs** in $\text{PPS}_{\text{chosen}}$. Assume that $\text{DCSs}^{\text{PPS}_{\text{chosen}}}$ and $\text{DCSs}^{\text{PPS}_{\text{alt}}}$ are the sets of **DCSs** in $\text{PPS}_{\text{chosen}}$ and PPS_{alt} , respectively, and $|\text{DCSs}^{\text{PPS}_{\text{chosen}}}| > |\text{DCSs}^{\text{PPS}_{\text{alt}}}|$; then it is computationally less intensive to transform the **DCSs** in PPS_{alt} to $\text{PPS}_{\text{chosen}}$, rather than the other way around. It is therefore, advantageous to choose the **PPS** that requires the least transformations of **DCSs** from other **PPSs**.

For instance, since the **UGV** has only one **PPS**, this second evaluative factor does not influence it. The case of the **2LPM** however, has multiple **PPSs**, where $|\text{DCSs}^{\mathcal{C}}| = 1$ and $|\text{DCSs}^{\mathcal{W}}| = 1$. Since both **PPSs** have the same number of **DCSs**, either **PPS** would qualify as $\text{PPS}_{\text{chosen}}$.

3. Third Evaluative Factor:

The third factor evaluates the candidacy of the **PPS** based on its potential to house **DCs**. For the same reason as the second factor, transformations of potential **DCs** from their **PPS** of origin to the **PPS** of choice, now need to be accounted for. The aim is to minimize the number of transformations of **DCs** while still accounting for their presence. Choosing a **PPS** is heavily dependent on where the **DCs** may be located. Thus, it is advantageous to choose the **PPS**, which is most likely to encounter **DCs**.

For example, if the **UGV** had alternative **PPSs** apart from the $\text{PPPS} = \mathcal{W}$, then the PPPS has the most potential to house **DCs**. Applying the third factor of evaluation to the **2LPM**, results in $\text{SPPS} = \mathcal{W}$ as $\text{PPS}_{\text{chosen}}$. This is so, since it is more likely that the **2LPM** encounters dynamic obstacles in the $\mathcal{W}$, such as falling objects and interactions with other robots or humans.

In summary, one chooses the **PPS** on which to commence path planning, by evaluating the three factors described above. But used individually, they produce conflicting results and additionally may result in multiple **PPSs** as results. To resolve that conflict, the factors are ranked in the following order of precedence (strongest to weakest):

$$\text{Second Evaluative Factor} > \text{Third Evaluative Factor} > \text{First Evaluative Factor} \quad (3.33)$$

That is, the Second Evaluative Factor is the primary influencing factor of the decision and ideally, is the only factor necessary. However, should it fail to produce a unique **PPS**, then the next, weaker factor is necessary to resolve the conflict. So, only when there is a tie of results in the Second Evaluative Factor, will the Third Evaluative Factor be necessary. Similarly, unless there is a tie using the Third Evaluative Factor, the First Evaluative Factor is not necessary. Should the First Evaluative Factor also yield a tie, any **PPS** may be chosen from the last set of results.

Using this order of precedence to pick $\text{PPS}_{\text{chosen}}$, one can revisit the examples of the **UGV** and **2LPM**. Clearly, the **UGV** example needs no evaluation because of its singular **PPS**. The **2LPM** example, however, does require the use of (3.33) to pick an appropriate $\text{PPS}_{\text{chosen}}$. Applying the Second Evaluative Factor leads to a tie between the two **PPSs**, $\mathcal{C}$ and $\mathcal{W}$, since they contain one **DCS** each. Following the chain of precedence (3.33), the Third Evaluative Factor qualifies $\mathcal{W}$ to be $\text{PPS}_{\text{chosen}}$. Since a decisive result has been reached, the First Evaluative Factor need not be invoked. In aggregate, $\mathcal{W}$ is the winning candidate for $\text{PPS}_{\text{chosen}}$. This implies that any other **PPS** becomes PPS_{alt} . Only one **PPS** satisfies this condition in this case, which is $\text{PPPS} = \mathcal{C}$.

3.2.1.8 Complete Definition of Chosen Path Planning Space (PPS)

Once the appropriate $\text{PPS}_{\text{chosen}}$ is selected as the space on which to commence path planning, one must ensure that it is completely defined. This is accomplished by the following two tasks, each of which can be performed by a dedicated algorithm.

1. First, all unmapped **DCSs** from all the alternative spaces in the set PPSs_{alt} , are mapped to $\text{PPS}_{\text{chosen}}$. This can be done systematically using Algorithm 3.3.

Taking the **UGV** as an example, it contains only one **DCS** in the entire **PPP**, which is located in its only $\text{PPS} = \mathcal{W}$, which is picked as $\text{PPS}_{\text{chosen}}$. Thus, no additional processing of **DCSs** is required and the $\text{PPS}_{\text{chosen}}$ is completely defined up to this point.

In the **2LPM** example, there exist distinct $\text{PPS}_{\text{chosen}} = \mathcal{W}$ and $\text{PPS}_{\text{alt}} = \mathcal{C}$. The latter has only one **DCS**, $\text{DCSs}^{\mathcal{C}} = \{\text{ML}_1, \text{ML}_2\}$

$$\begin{aligned} \text{DCSs}^{\mathcal{C}} &= \{\text{ML}_1, \text{ML}_2\} = \text{ML}_1 \cup \text{ML}_2 \\ &= \{(q_1, q_2) \in (X \times (0^\circ, 1^\circ]) \cup (X \times (0^\circ, 2^\circ])\} \\ &= \{(q_1, q_2) \in X \times (0^\circ, 2^\circ]\} \end{aligned}$$

Next, the mapping $f_{\text{PPS}_{\text{alt}}}^{\text{PPS}_{\text{chosen}}}$ from $\text{PPS}_{\text{alt}} = \mathcal{C}$ to $\text{PPS}_{\text{chosen}} = \mathcal{W}$ must be defined. It was seen in Section 3.2.1.4 that this mapping is the robot's forward kinematics mapping $\mathbf{f}^{\text{fwd}}$.

Therefore, $f_{PPS_{alt}}^{PPS_{chosen}} = \mathbf{f}^{wd}$. The mapping of $DCSs^c$ onto $\mathcal{W}$ is

$$\mathbf{f}^{wd}(DCSs^c) = \{(x, y) = \mathbf{f}^{wd}(q_1, q_2), \forall (q_1, q_2) \in DCSs^c\}$$

With this, it is concluded that $\mathcal{W}$ contains two $DCSs$: a native one, $\{NG_1\}$, and a mapped one, $\mathbf{f}^{wd}(DCSs^c)$. It was noted earlier that an additional structure would be imposed to describe $\mathcal{W}$ as a manifold. The same topological representation can be applied to also represent the mapped ($DCSs^c$) as it is contained in $\mathcal{W}$. Even if the DCS is Euclidean while the PPS is not, one can accommodate for that. As long as the dimension of the DCS is less than or equal to the dimension of the PPS , the topological representation can be chosen to allow for compatibility. This will be detailed in Section 3.2.1.9.

2. Second, re-evaluate the mapped $DCSs$ in the PPS_{chosen} to see if they can be merged with the old or new $DCSs$. That is, in the case of any mapped $DCSs$, Algorithm 3.2 is applied on all the native and mapped $DCSs$ to ensure that all the CS are actually disjoint. This gives the user yet another chance to define a new value for the previously defined proximity measure Δ_{CS} .

Back to the UGV example, since it needed no additional mapping of $DCSs$ to PPS_{chosen} , no re-evaluation of the $DCSs$ is necessary. In the case of the $2LPM$, the re-evaluations of the $DCSs$ in PPS_{chosen} is necessary. The result depends on Δ_{CS} and $f_{PPS_{alt}}^{PPS_{chosen}}$, which are not defined here, for generality. Instead, we will assume that the process yields a single DCS .

3.2.1.9 Constructing a Constraint-Free Manifold (CFM) in the Chosen Path Planning Space (PPS)

This section focuses on redefining the PPS_{chosen} and $PPSs_{alt}$ of the PPP as topological spaces. So far, all $PPSs$ involved have accounted for the bounds of $DCSs$. In the process of defining the PPS as a topological space, the regions corresponding to $DCSs$ are considered as part of the topological space. The removal of such regions from the topological space results in a constraint-free topological representation of the PPS . Therefore, the final topological PPS on which any further structure may be imposed, is automatically constraint-free. With this basic structure in place, one can now define or even construct a manifold on this space, to allow for path planning. Following the construction of a CFM on PPS_{chosen} , it is then discretized, as explained in Section 3.2.2.

This section demonstrates three main steps required to construct a CFM . First, a topological representation of the PPS_{chosen} is formulated. Second, using information about the $DCSs$, an equivalent constraint-free topological representation of the PPS_{chosen} is constructed. Third, a constraint-free manifold on the PPS_{chosen} is constructed, such that it is suitable for discretized path planning. Equivalence between the constraint-free $PPSs$ are also shown with the help of the concepts introduced earlier in Section 3.1.

First, the topological representation of the PPS is obtained using Definition 3.4. While one can start with any PPS , it is recommended to start with PPS_{chosen} for the subsequent

Algorithm 3.3 Mapping all unmapped DCSs to $\text{PPS}_{\text{chosen}}$

Input:

$\text{PPS}_{\text{chosen}}$
 $\text{PPSs}_{\text{alt}} = \{\text{PPS}_1, \dots, \text{PPS}_{|\text{PPSs}| - 1}\}$
 $\text{DCSs}^{\text{PPS}_i}$ and $f_{\text{PPS}_i}^{\text{PPS}_{\text{chosen}}}, \forall \text{PPS}_i \in \text{PPSs}_{\text{alt}}$ $\triangleright$ DCSs and mapping information

Output:

$\text{PPS}_{\text{chosen}}$ containing the native DCSs and those mapped from PPSs_{alt}

```

1:  $\alpha \leftarrow \text{PPSs}_{\text{alt}}$   $\triangleright$  All alternative PPSs whose DCSs need processing
2:  $\text{DCS}_{\text{mapped}} \leftarrow \emptyset$ 
3: while  $\alpha \neq \emptyset$  do
4:    $\text{PPS}_{\text{curr}} \leftarrow \alpha(1)$   $\triangleright$  Alternative PPS whose DCSs are currently processed
5:   for  $i = 1$  to  $|\text{DCSs}^{\text{PPS}_{\text{curr}}}|$  do  $\triangleright$  Transform every DCS in the alternative  $\text{PPS}_{\text{curr}}$ 
6:      $\text{DCS}_a \leftarrow \text{DCSs}^{\text{PPS}_{\text{curr}}}(i)$ 
7:      $\text{DCS}_m \leftarrow f_{\text{PPS}_{\text{curr}}}^{\text{PPS}_{\text{chosen}}}(\text{DCS}_a)$ 
8:      $\text{DCS}_{\text{mapped}} \leftarrow \text{DCS}_{\text{mapped}} \cup \text{DCS}_m$   $\triangleright$  Store transformed DCS
9:   end for
10:   $\alpha \leftarrow \alpha \setminus \text{PPS}_{\text{curr}}$   $\triangleright$  Remove the PPS whose DCSs have been processed
11: end while
12: return  $\text{PPS}_{\text{chosen}}$ 

```

steps. The following explanation uses the general notation of PPS, not to compromise generality. Basic information about the PPS is given in (3.7) and (3.8). The topology assigned to the PPS, as defined by (3.8), is that of the metric topology. Generally speaking, for a metric space (M, d) , d is the metric defined on M , such that $d : M \times M \rightarrow \mathbb{R}$. One can define an open ball B of radius $r > 0$ and $r \in \mathbb{R}$ around any point m in M , i.e., $B(m, r) = \{n \in M : d(m, n) < r\}$. The set of all such open balls are subsets of M , and are open in the sense of the metric space. Therefore, this is a topology and is called the metric topology on M , or the topology generated by d [98]. A topological space is said to be metrizable if its topology happens to be the metric topology generated by some metric on the space. If the space is metrizable, then the metric that generates its topology is not uniquely determined, because many different metrics can give rise to the same topology (meaning that the same sets are open with respect to several metrics) [98]. Therefore, choosing the metric topology allows the usage of any metric and multiple metrics. One may begin by considering the PPS as a subset of $\mathbb{R}^n$ when appropriate, since subsets of $\mathbb{R}^n$ are always considered $\mathcal{TS}$ s with this topology [98]. If not a subset of $\mathbb{R}^n$, the metric topology may still be defined on the PPS. Both options are now explained.

Formally, the set of tuples in PPS is denoted by its span, i.e., $\mathcal{S}_{\text{PPS}} = \prod_{i=1}^l p_i^{\text{span}}$. Each of the l dimensions is treated as a separate topological space. The set of points in i^{th} dimension is denoted by $\mathcal{S}_i = p_i^{\text{span}}$ and is assigned the metric topology $\mathcal{T}_i$, induced by $d = \|\cdot\|_\infty$ and for some small $r > 0$. Then, $\mathcal{T}_i = \{\cup B(p, r)\}$ where, $B(p, r)$ is defined $\forall p \in \mathcal{S}_i$, such that $\exists q \in \mathcal{S}_i$ satisfying $d(p, q) = \|p, q\|_\infty < r$. Similarly, all l dimensions are equipped

with the same topology. Then, the Cartesian product of all l topological spaces results in the product topological space representing **PPS**. That is, $\mathcal{S}_{\text{PPS}} = \mathcal{S}_1 \times \cdots \mathcal{S}_i \times \cdots \mathcal{S}_l = \prod_{i=1}^l p_i^{\text{span}}$, on which the product topology is thus induced [98]. The product topology is given by $\mathcal{T}_{\text{PPS}} = \{\mathcal{T}_1 \times \cdots \mathcal{T}_i \times \cdots \mathcal{T}_l\}$. In the above deduction, if each of the l dimensions is a subset of $\mathbb{R}$, then the $\mathcal{S}_{\text{PPS}}$ is a subset of $\mathbb{R}^l$ with the metric topology as defined. Then, the topology defined consists of l -dimensional open balls of radius $< r$. Alternatively, one may directly define $\mathcal{S}_{\text{PPS}}$ as a subset of $\mathbb{R}^l$, and define a preferred metric on it. The ℓ_p -norm is used to represent the metric in a general form and is defined for an l -dimensional vector $\mathbf{p}$ as

$$\|\mathbf{p}\|_p = \left(\sum_{i=1}^l |p_i|^p \right)^{\frac{1}{p}}$$

In either case, we choose to define additional metrics on $\mathcal{S}_{\text{PPS}}$. Metrics, such as $d = \|\cdot\|_1$, $d = \|\cdot\|_2$ and $d = \|\cdot\|_\infty$ are useful tools to demonstrate homeomorphisms. Defining topologies induced by different metrics helps to view the topological space as a breakdown of smaller, open subsets (finer) as opposed to that of a few, bigger open subsets (coarse). Given two topologies $\mathcal{T}_1$ and $\mathcal{T}_2$ on a set X , $\mathcal{T}_1$ is finer than $\mathcal{T}_2$ if $\mathcal{T}_1 \supseteq \mathcal{T}_2$, and coarser than $\mathcal{T}_2$ if $\mathcal{T}_1 \subseteq \mathcal{T}_2$ [98]. Then, the topology induced by the different metrics $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$ on $\mathcal{S}_{\text{PPS}}$ may be compared and referred to as the finest, coarse and coarsest of the three topologies. Using the above, every **PPS** whose bounds are available in the form of (3.8), can be defined as a topological space. For **PPSs** whose information is unavailable, the mapping $f_{\text{PPS}_i}^{\text{PPS}_j}$ is used to determine the span of such spaces. As such, their topology is also deduced in the same manner as specified above. The topological space for PPS_i is denoted by the pair $(\mathcal{S}_{\text{PPS}_i}, \mathcal{T}_{\text{PPS}_i}) = \mathcal{T}\mathcal{S}_{\text{PPS}_i}$. This step describes the **PPS** as a topological space using its span, but without removing the **DCSs** from it. Removing the **DCSs** is explained in the next step.

In this second step, the equivalent constraint-free topological representation of the **PPSs** is now extracted. The $\text{PPS}_{\text{chosen}}$ is the first **PPS** that requires this step. Section 3.2.1.8 showed that $\text{PPS}_{\text{chosen}}$ now contains mapped and native (unmapped) **DCSs** in $\text{DCS}^{\text{PPS}_{\text{chosen}}}$, i.e., all possible regions to avoid during path planning. In the following discussion, the superscript “ $\text{PPS}_{\text{chosen}}$ ” is dropped, since only the **DCSs** lying in $\text{PPS}_{\text{chosen}}$ are tackled. The native **DCSs** are considered first, before passing on to the mapped **DCSs**. Every native **DCS** has bounds as defined in (3.7) and (3.8) with respect to $\text{PPS}_{\text{chosen}}$. That is, it is a subset of $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$. Removing it from $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$ results in a set of points that are free of constraints. Every subset of a topological space is a topological space, by virtue of the subspace topology [98]. That is, the subset can still be expressed using the topology of the original topological space. In such a case, the subset is called a subspace of $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$. Therefore, the constraint-free set of points is represented as a subspace of $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$, as follows. The **DCS** can be described as a Cartesian product, union of Cartesian products, or by using any other mechanism of defining a set of points. For simplicity purpose, and for the sake of illustration, assume that the **DCS**’s general form is $\prod_{i=1}^l p_i^{\text{DCS}_{\text{span}}}$ where $p_i^{\text{DCS}_{\text{span}}} \in (p_i^{\text{DCS}_{\text{min}}}, p_i^{\text{DCS}_{\text{max}}})$. Thus,

$\mathcal{S}_{\text{DCS}} = \prod_{i=1}^l p_i^{\text{DCS}_{\text{span}}}$. The constraint-free set of points is given by:

$$\begin{aligned} \mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}} &= \mathcal{S}_{\text{PPS}_{\text{chosen}}} \setminus \mathcal{S}_{\text{DCS}} \\ \implies \mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}} &= \prod_{i=1}^l p_i^{\text{span}} \setminus \prod_{i=1}^l p_i^{\text{DCS}_{\text{span}}} \\ \implies \mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}} &= \prod_{i=1}^l p_i^{\text{free}_{\text{span}}} ; \text{ where } p_i^{\text{free}_{\text{span}}} \in (p_i^{\text{free}_{\text{min}}}, p_i^{\text{free}_{\text{max}}}) \end{aligned}$$

The subspace topology $\mathcal{T}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$ is now that subset of $\mathcal{T}_{\text{PPS}_{\text{chosen}}}$, that is applicable to $\mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$, which is a subset of $\mathcal{S}_{\text{PPS}_{\text{chosen}}}$. The constraint-free topological space of $\text{PPS}_{\text{chosen}}$ is thus denoted as $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$. Therefore, every native DCS may be defined and removed this way to produce a constraint-free topological space. Any remaining DCS is a mapped DCS , which has been mapped to $\text{PPS}_{\text{chosen}}$ from PPS_{alt} . Note that, should the DCS 's bounds in its native PPS be non-invertible by virtue of singularities, the bounds must be slightly modified to ensure consistent, invertible mapping to $\text{PPS}_{\text{chosen}}$. That is, any or all l -dimensions may be modified by ϵ_i^{min} and ϵ_i^{max} , for $i = 1, \dots, l$, such that the new bounds become $(p_i^{\text{DCS}_{\text{min}}} - \epsilon_i^{\text{min}}, p_i^{\text{DCS}_{\text{max}}} + \epsilon_i^{\text{max}})$. In fact, every DCS can be modified as such, giving the user ample flexibility to define the PPS as necessary for the application, while still retaining topological properties. That is, this gives the user freedom in defining the clearance from the DCS , while planning a path.

Naturally then, the mapped DCS is a subspace of $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$. Thus, it can be removed from $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$ in the same manner as the unmapped DCS s. A topological representation of the PPS is completely constraint-free, if and only if all DCS s have been removed from it. A similar process can be undertaken to construct equivalent topological spaces for the PPS_{alt} , denoted by $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{alt}}}^{\text{free}}$. At this juncture, the topological representations of all PPS s are equivalent. That is, all topological spaces are free of exactly the same number of constraints and constraints have been mapped to each space, when needed. It can be noted that the original map $f_{\text{PPS}_{\text{chosen}}}^{\text{PPS}_{\text{alt}}}$, relating $\text{PPS}_{\text{chosen}}$ and PPS_{alt} , may not be invertible at singular points. But, such non-invertible points in the form of DCS , containing singularities, mechanical-limits, no-go zones and obstacles, have been removed from every space. Therefore, the map $f_{\text{PPS}_{\text{chosen}}}^{\text{PPS}_{\text{alt}}}$ restricted to the now constraint-free topological space, is invertible. If this invertibility is continuous everywhere on the constraint-free topological space, the map is a homeomorphism. Continuity of maps between topological spaces is ensured by the topology chosen, whereby pre-images of maps are open in the topology [98, Proposition 2.19]. Thus, all constraint-free topological spaces are topologically equivalent to each other, by virtue of the homeomorphisms that relate the spaces.

In the third and final step, a manifold is constructed on the constraint-free topological space. By virtue of lying in the constraint-free space, the manifold is also be constraint-free. As usual, one commences with $\text{PPS}_{\text{chosen}}$, specifically $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$. Using Definition 3.7, a manifold can be constructed on a subset of $\mathcal{T}\mathcal{S}_{\text{PPS}_{\text{chosen}}}$, where the manifold needs to satisfy two requirements. First, its shape must be such that the region representing a DCS is absent; and second, only the constraint-free region is navigable on the manifold. Utilizing

Definition 3.16, one could use a torus or an annulus as a possible choice. A torus is a closed surface without a boundary, and has one genus, which is suitable to represent one DCS. The issue with the torus however, is that it is not well suited for planar navigation. In such cases, an annulus is a better solution. It can be open or closed, i.e., one can choose to include points on the boundary. Additionally, by definition, it has an absence of region that can represent the DCS, while representing the navigable region around the DCS. Most importantly, the annulus is a planar, 2-dimensional surface and manifold in $\mathbb{R}^3$. This makes it the ideal choice as a constraint-free manifold to be constructed on $\mathcal{TS}_{PPS_{chosen}}$.

Choosing the manifold to be constructed is an essential evaluation, since a path needs to be computed on it. Section 3.2.2 computes a path by first discretizing the manifold, i.e., by creating a polygonal mesh of the manifold. Every compact surface admits a polygonal presentation [98], therefore, it is preferred that the chosen manifold be compact. It can be seen that the closed annulus (i.e., with boundary) is compact and an open annulus can be made compact. To create a representation of the environment with all its DCSs, one annulus is constructed for one DCS and the local region around it. Then, an environment with several DCSs can be constructed using the concept of connected sums of manifolds. In order to use connected sums, the manifolds must themselves be connected. The annulus can be shown to be connected. Therefore, the annulus is the manifold of choice on $\mathcal{TS}_{PPS_{chosen}}$. Now, the construction of an annulus corresponding to one DCS and its surrounding area is shown. Properties such as connectedness and compactness of the annulus are also shown along the way.

The construction of an annulus on $\mathcal{TS}_{PPS_{chosen}}^{free}$ can be accomplished through homeomorphisms. We show that the topological space of finite dimensions with the DCS removed is homeomorphic to the punctured unit plane. This punctured unit plane is shown to be homeomorphic to a punctured unit disk. The punctured unit open disc is then homeomorphic to the annulus of finite radii. Using this sequence of steps, a sequence of homeomorphisms is established. Since every composition of a homeomorphism is also a homeomorphism, one can show that the either all or a subset of $\mathcal{TS}_{PPS_{chosen}}$ can be mathematically deformed into an annulus. This annulus then, still adheres to the original topology that was attributed to $\mathcal{TS}_{PPS_{chosen}}$. This is so, since every homeomorphism is a local homeomorphism [98, Proposition 2.31(a)] and every local homeomorphism is continuous and open. So, every open set in the original topological space is mapped to an open set by virtue of the open map, i.e., the homeomorphism. Thus, the topological properties remain intact.

To demonstrate the steps, a few assumptions are made about the $\mathcal{TS}_{PPS_{chosen}}$, without loss of generality. $\mathcal{TS}_{PPS_{chosen}}$ is 2-dimensional whose span is $(-r_2, r_2) \times (-r_2, r_2)$, where $(r_2, r_2) \in \mathcal{S}_{PPS_{chosen}}$. The resulting region is an open square (boundary not included) centered at the origin of $\mathcal{TS}_{PPS_{chosen}}$, and whose sides are of length $\approx 2r_2$. This $\mathcal{TS}_{PPS_{chosen}}$ is assumed to contain one DCS, also centered at the origin, whose span is $[-r_1, r_1] \times [-r_1, r_1]$, where $[r_1, r_1] \in \mathcal{S}_{PPS_{chosen}}$ and $r_2 > r_1 > 0$. The DCS is then a closed square (boundary included) centered at the origin, whose sides are of length $2r_1$. To create $\mathcal{TS}_{PPS_{chosen}}^{free}$, when the DCS is removed from $\mathcal{TS}_{PPS_{chosen}}$, one is left with a square from whose interior, another square has been removed. This topological space can be defined with the help of previously induced metrics for the space, as such: $\mathcal{S}_{PPS_{chosen}}^{free} = \{\mathbf{p} \in \mathcal{S}_{PPS_{chosen}} : r_1 < \|\mathbf{p}\|_\infty < r_2\}$. The resulting space is devoid of internal and external boundaries by virtue of how a DCS was defined.

Boundaries may be factored in, but have been left out for simplicity. Since it is known that the same metric topology is inherited by virtue of the subspace topology, an abuse of notation will now be used. The notation $\mathcal{TS}$ will be used inter-changeably with $\mathcal{S}$ and $\mathcal{T}$. Since all homeomorphisms are operated on $\mathcal{TS}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$, the reference to $\mathcal{TS}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$ will no longer be made. Only the deformation of shapes describing the space will be referred to. The shape as defined so far, i.e., $\{\mathbf{p} \in \mathcal{S}_{\text{PPS}_{\text{chosen}}} : r_1 < \|\mathbf{p}\|_{\infty} < r_2\}$, will be referred to as representation-1 or R-1.

$$\text{R-1} \equiv \{\mathbf{p} \in \mathcal{S}_{\text{PPS}_{\text{chosen}}} : r_1 < \|\mathbf{p}\|_{\infty} < r_2\}$$

In Figure 3.23, R-1 is the area shown in gray, with dashed edges to show that boundaries are excluded.

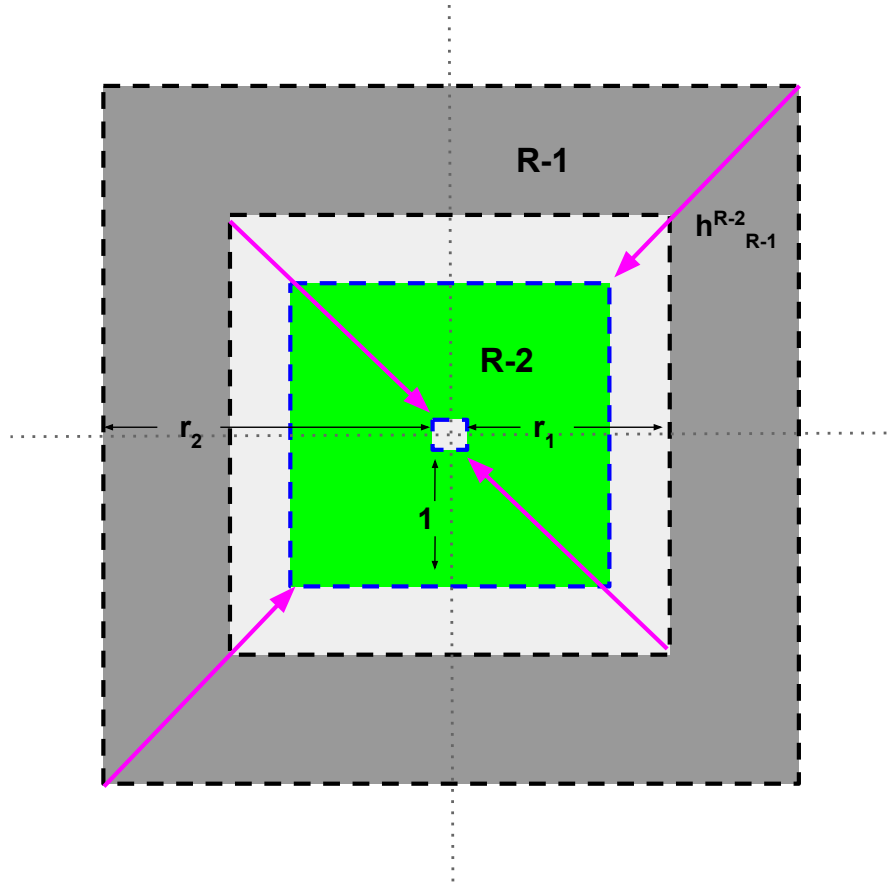


Figure 3.23: Homeomorphism $h_{\text{R-1}}^{\text{R-2}}$ from R-1 to R-2

The first homeomorphism shows that R-1 is homeomorphic to the punctured unit-plane in $\mathbb{R}^2$. The punctured unit-plane denoted by R-2 is defined as

$$\text{R-2} \equiv \{\mathbf{p} \in \mathbb{R}^2 \setminus \{0, 0\} : 0 < \|\mathbf{p}\|_{\infty} < 1\}$$

Define the homeomorphism from R-1 to R-2 as

$$h_{\text{R-1}}^{\text{R-2}}(\mathbf{p}^{\text{R-1}}) = \mathbf{p}^{\text{R-1}} \frac{\|\mathbf{p}^{\text{R-1}}\|_{\infty} - r_1}{(r_2 - r_1)\|\mathbf{p}^{\text{R-1}}\|_{\infty}} = \mathbf{p}^{\text{R-2}}$$

where $\mathbf{p}^{R-1} \in R-1$ and $\mathbf{p}^{R-2} \in R-2$. Note that the homeomorphism is taking place in the same topological space; the notation $\mathbf{p}^{R-1}$ and $\mathbf{p}^{R-2}$ are simply indicative of the domain and image of the map. The same distinguishing notation will be used for the remaining homeomorphisms as well. The inverse map, i.e., from $R-2$ to $R-1$ is defined as

$$h_{R-2}^{R-1}(\mathbf{p}^{R-2}) = \mathbf{p}^{R-2} \frac{\|\mathbf{p}^{R-2}\|_{\infty}(r_2 - r_1) + r_1}{\|\mathbf{p}^{R-2}\|_{\infty}} = \mathbf{p}^{R-1}$$

Recalling that a homeomorphism is a continuous and bijective map with a continuous inverse, one can view the homeomorphism between $R-1$ and $R-2$ in exactly the same way. Now, we show that $h_{R-1}^{R-2}(\mathbf{p}^{R-1})$ does satisfy the definition of a homeomorphism. $h_{R-1}^{R-2}(\mathbf{p}^{R-1})$ is continuous, i.e., pre-images of open sets are open in the defined topology of $\mathcal{TS}_{PPS_{chosen}}^{free}$, and is also bijective. One can verify, that $h_{R-1}^{R-2}(\mathbf{p}^{R-1})$ does indeed map $R-1$ to $R-2$ by mapping representative points within $R-1$. Denote the representative point as $(r_1 + \epsilon, r_1 + \epsilon)$. Then, $h_{R-1}^{R-2}(r_1 + \epsilon, r_1 + \epsilon) = (r_1 + \epsilon, r_1 + \epsilon) \frac{(r_1 + \epsilon) - r_1}{(r_2 - r_1)(r_1 + \epsilon)}$ takes the following values:

$$h_{R-1}^{R-2}(r_1 + \epsilon, r_1 + \epsilon) \begin{cases} \rightarrow (0, 0) \text{ as } \epsilon \rightarrow 0 \\ = (p_1^{R-2}, p_2^{R-2}) = (a, a) \text{ for } a \in (0, 1) \text{ as } \epsilon = c \text{ where } 0 < c < r_2 - r_1 \\ \rightarrow (1, 1) \text{ as } \epsilon \rightarrow r_2 - r_1 \end{cases}$$

The above verification shows a subset of the upper right half quadrant of $R-1$ being mapped to the equivalent quadrant of $R-2$. Similar checks performed on representative points for the remaining quadrants of $R-1$, such as $(-r_1 - \epsilon, r_1 + \epsilon)$, $(-r_1 - \epsilon, -r_1 - \epsilon)$ and $(r_1 + \epsilon, -r_1 - \epsilon)$, show that $R-1$ is successfully mapped onto $R-2$. The effect that h_{R-1}^{R-2} on a point $\mathbf{p}^{R-1}$ can be seen in Figure 3.23, where the gray region of $R-1$ is deformed to the green region, that is the punctured unit-plane, $R-2$. Next, one checks if the inverse mapping from $R-2$ to $R-1$ exists. Through $h_{R-2}^{R-1}(\mathbf{p}^{R-2})$, the existence of the inverse is also established. To check that $R-2$ does indeed get mapped to $R-1$, use a representative point denoted as (ϵ, ϵ) and check that $h_{R-2}^{R-1}(\epsilon, \epsilon) = (\epsilon, \epsilon) \frac{\epsilon(r_2 - r_1) + r_1}{\epsilon}$ becomes

$$h_{R-2}^{R-1}(\epsilon, \epsilon) \begin{cases} \rightarrow (r_1, r_1) \text{ as } \epsilon \rightarrow 0 \\ = (p_1^{R-1}, p_2^{R-1}) = (b, b) \text{ for } b \in (r_1, r_2) \text{ as } \epsilon \rightarrow c \text{ where } 0 < c < 1 \\ \rightarrow (r_2, r_2) \text{ as } \epsilon \rightarrow 1 \end{cases}$$

Clearly, the above exercise shows the inverse of a subset of the upper right quadrant of $R-2$ being mapped to its equivalent area in $R-1$. This is the same region whose inverse was checked using $h_{R-1}^{R-2}(r_1 + \epsilon, r_1 + \epsilon)$. Other quadrants can be verified in a similar manner as well. It can be seen that this inverse is also continuous. Therefore, $h_{R-1}^{R-2}(\mathbf{p}^{R-1})$ is indeed a homeomorphism, and consequently, $R-1$ and $R-2$ are homeomorphic or topologically equivalent. Showing that $R-1$ is homeomorphic to the punctured unit plane aids in achieving two goals. First, it is the first map that is required to construct the final homeomorphism to the annulus. Second, and more importantly, by showing that $R-1$ is topologically the same as the punctured-unit plane, properties of the latter can now be utilized. A topology (and thus its associated properties) is

precisely the information preserved by homeomorphisms [98], of which connectedness is one such property. Recall that, to create a representation of the environment with multiple DCSs, connected sums of manifolds is proposed. Since the manifold used needs to be connected, connectedness is a topological property of interest. An intuitive explanation of connectedness is as follows: a space is connected, if and only if it is not homeomorphic to a disjoint union of two or more nonempty spaces. A stronger variant of connectedness, called path-connectedness, is defined since it is easier to verify and is equivalent to connectedness for manifolds [98].

Path-connectedness of a $\mathcal{TS} X$, where $p, q \in X$, is defined as follows. Recall that a path in X from p to q is a continuous map $f : I \rightarrow X$ such that $f(0) = p$ and $f(1) = q$, where $I = [0, 1]$ is the unit interval. Then, X is path-connected if for every $p, q \in X$, there is a path in X from p to q . Path connectedness implies connectedness [98, Theorem 4.15]. Also, every continuous image of a path-connected space is path-connected [98, Proposition 4.13]. The space $\mathbb{R}^n \setminus \{\mathbf{0}\}$, for $n \geq 2$, can be shown to be path-connected, and thus connected (out of the scope of this thesis). The notation “ $\mathbf{0}$ ” denotes a vector of zeros. We showed that R-1 is homeomorphic to $\mathbb{R}^2 \setminus \{\mathbf{0}\}$, which is known to be connected. Since h_{R-2}^{R-1} is continuous, it implies that R-1, which is the image of a continuous map, is also connected. Therefore, by showing that R-1 is homeomorphic to a path-connected space, we have shown that R-1 is also path-connected. Further, homeomorphisms also preserve this property, which is known as the Topological Invariance of Connectedness, where every space that is homeomorphic to a connected space is connected.

The second homeomorphism shows that R-2 is topologically equivalent to the punctured open unit disc. Denoted by R-3, it is defined as

$$R-3 \equiv \{\mathbf{p} \in \mathbb{R}^2 \setminus \{0, 0\} : 0 < \|\mathbf{p}\|_2 < 1\}$$

It can be seen in Figure 3.24 as the area within the dashed green perimeter. The homeomorphism from R-2 to R-3 is seen as

$$h_{R-2}^{R-3}(\mathbf{p}^{R-2}) = \mathbf{p}^{R-2} \frac{\|\mathbf{p}^{R-2}\|_\infty}{\|\mathbf{p}^{R-2}\|_2} = \mathbf{p}^{R-3}$$

The inverse map, i.e., from R-3 to R-2 is defined by

$$h_{R-3}^{R-2}(\mathbf{p}^{R-3}) = \mathbf{p}^{R-3} \frac{\|\mathbf{p}^{R-3}\|_2}{\|\mathbf{p}^{R-3}\|_\infty} = \mathbf{p}^{R-2}$$

Once again, h_{R-2}^{R-3} can be seen to be continuous and bijective, with a continuous inverse in the form of h_{R-3}^{R-2} . One can confirm that R-2 is indeed mapped to R-3 using a representative point (ϵ, ϵ) , i.e., $h_{R-2}^{R-3}(\epsilon, \epsilon) = (\epsilon, \epsilon) \frac{\epsilon}{\sqrt{2\epsilon^2}}$ becomes

$$h_{R-2}^{R-3}(\epsilon, \epsilon) \begin{cases} \rightarrow (0, 0) \text{ as } \epsilon \rightarrow 0 \\ = (p_1^{R-3}, p_2^{R-3}) \in (0, 1) \times (0, 1) \text{ such that } \|\cdot\|_2 < 1 \text{ as } \epsilon = c \text{ where } 0 < c < 1 \\ \rightarrow (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \text{ as } \epsilon \rightarrow 1 \end{cases}$$

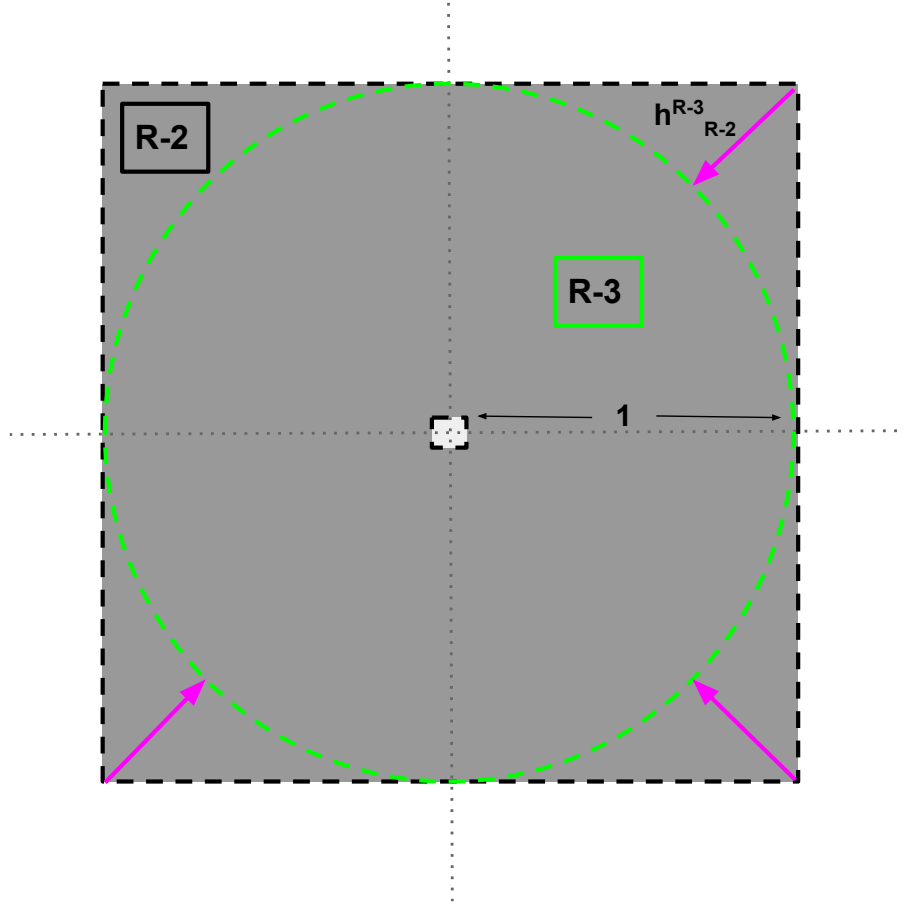


Figure 3.24: Homeomorphism h^{R-3}_{R-2} from R-2 to R-3

This homeomorphism can be seen with the help of arrows in Figure 3.24. Other quadrants yield the respective regions in R-3. A check of the inverse may also be seen using $h^{R-2}_{R-3}(\epsilon, \epsilon) = (\epsilon, \epsilon) \frac{\sqrt{2}\epsilon}{\epsilon}$, such that

$$h^{R-2}_{R-3}(\epsilon, \epsilon) \begin{cases} \rightarrow (0, 0) \text{ as } \epsilon \rightarrow 0 \\ = (p_1^{R-3}, p_2^{R-3}) \in (0, 1) \times (0, 1) \text{ such that } \|\cdot\|_\infty < 1 \text{ as } \epsilon = c \text{ where } 0 < c < \frac{1}{\sqrt{2}} \\ \rightarrow (1, 1) \text{ as } \epsilon \rightarrow \frac{1}{\sqrt{2}} \end{cases}$$

The third and final homeomorphism maps R-3 to the desired annulus. Denoted by R-4, this annulus with desired inner radius r_i and outer radius r_o , is defined as

$$R-4 \equiv \{\mathbf{p} \in \mathbb{R}^2 \setminus \{0, 0\} : r_i < \|\mathbf{p}\|_2 < r_o\} \quad (3.34)$$

The inner radius defines the expanse of the DCS that needs to be avoided. The difference between the inner and outer radii defines the expanse of the constraint-free region. R-4 can be seen in Figure 3.25 as the gray region. The homeomorphism from R-3 to R-4 is seen as

$$h_{R-3}^{R-4}(\mathbf{p}^{R-3}) = \mathbf{p}^{R-3} \frac{\|\mathbf{p}^{R-3}\|_2 (r_o - r_i) + r_i}{\|\mathbf{p}^{R-3}\|_2} = \mathbf{p}^{R-4}$$

The inverse map, i.e., from R-3 to R-2 is defined as

$$h_{R-4}^{R-3}(\mathbf{p}^{R-4}) = \mathbf{p}^{R-4} \frac{\|\mathbf{p}^{R-4}\|_2 - r_i}{(r_o - r_i) \|\mathbf{p}^{R-4}\|_2} = \mathbf{p}^{R-3}$$

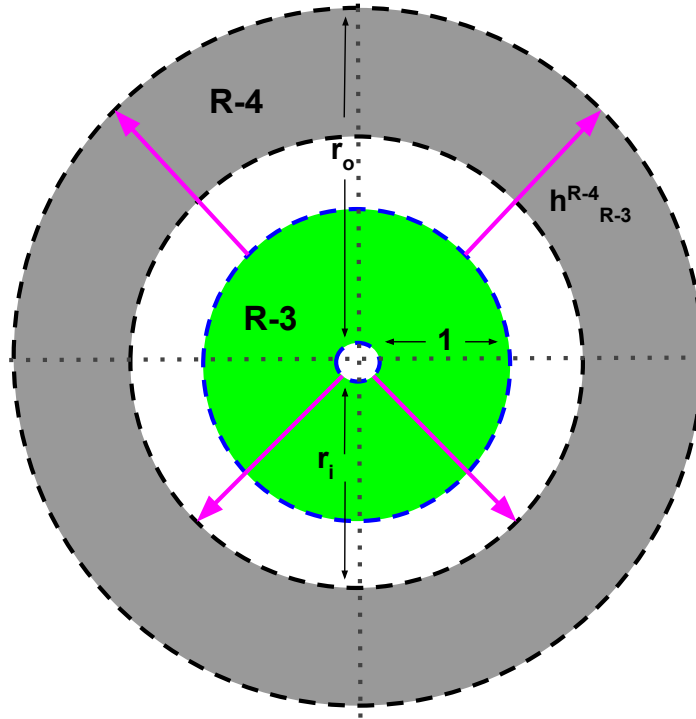


Figure 3.25: Homeomorphism h_{R-3}^{R-4} from R-3 to R-4

One can verify that h_{R-3}^{R-4} is continuous and bijective, with a continuous inverse in the form of h_{R-4}^{R-3} . Using the representative point (ϵ, ϵ) , R-3 is seen to map to R-4 by checking that $h_{R-3}^{R-4}(\epsilon, \epsilon) = (\epsilon, \epsilon) \frac{\sqrt{2}\epsilon(r_o - r_i) + r_i}{\sqrt{2}\epsilon}$ obeys the following:

$$h_{R-3}^{R-4}(\epsilon, \epsilon) \begin{cases} \rightarrow (\frac{r_i}{\sqrt{2}}, \frac{r_i}{\sqrt{2}}) \text{ as } \epsilon \rightarrow 0 \\ = (p_1^{R-3}, p_2^{R-3}) \in (r_i, r_o) \times (r_i, r_o) \text{ such that } \|\cdot\|_2 < 1 \text{ as } \epsilon = c \text{ where } 0 < c < \frac{1}{\sqrt{2}} \\ \rightarrow (\frac{r_o}{\sqrt{2}}, \frac{r_o}{\sqrt{2}}) \text{ as } \epsilon \rightarrow \frac{1}{\sqrt{2}} \end{cases}$$

Other representative points, such as $(\epsilon, 0)$, may be checked by expecting that as ϵ tends to 0, the image tends to $(\frac{r_i}{\sqrt{2}}, 0)$, and so on. The inverse may be checked similarly by

$$h_{R-4}^{R-3}(\epsilon, \epsilon) = (\epsilon, \epsilon) \frac{\sqrt{2}\epsilon - r_i}{\sqrt{2}\epsilon(r_o - r_i)}.$$

$$h_{R-4}^{R-3}(\epsilon, \epsilon) \begin{cases} \rightarrow (0, 0) \text{ as } \epsilon \rightarrow \frac{r_i}{\sqrt{2}} \\ = (p_1^{R-3}, p_2^{R-3}) \in (0, 1) \times (0, 1) \text{ such that } \|\cdot\|_2 < 1 \text{ as } \epsilon = c \text{ where } \frac{r_i}{\sqrt{2}} < c < \frac{r_o}{\sqrt{2}} \\ \rightarrow (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \text{ as } \epsilon \rightarrow \frac{r_o}{\sqrt{2}} \end{cases}$$

Naturally, other regions can be shown to be mapped as well. The composition of the three homeomorphisms, $h_{R-3}^{R-4} \circ h_{R-2}^{R-3} \circ h_{R-1}^{R-2} = h_{R-1}^{R-4}$ shows that R-1 is topologically equivalent to R-4, which is an annulus defining the constraint-free navigable space. We have shown that an annulus is constructed from the given path planning space with the original topology, by using homeomorphisms. Now, showing that this annulus is a topological manifold requires showing that the annulus is locally homeomorphic to $\mathbb{R}^2$.

Recall that a homeomorphism on a restriction of the $\mathbb{R}^2$ was used to arrive at the annulus, namely h_{R-1}^{R-4} . Then, one can use this to show that every small open ball of the annulus is homeomorphic to some open ball in $\mathbb{R}^2$. This is because every homeomorphism is also a local homeomorphism [98, Proposition 2.31]. Formally, one can define charts and an atlas, similar to Example 3.2. And in fact, since R-4 as defined is already an open subset of $\mathbb{R}^2$, we simply show this using one chart. The chart is given by $\{\mathcal{U}_\alpha, \phi_\alpha\} = \{R-4, Id\}$ is simply the identity map. That is, $Id : R-4 \rightarrow \mathbb{R}^2$ such that $\mathbf{p} = (x, y)$. It is clearly continuous, bijective and continuously invertible. Thus, it is a homeomorphism. Since $\phi_\alpha = Id$, covers the entire annulus given by R-4, one may want to conclude that the atlas on R-4 contains exactly one chart, which is $\phi_\alpha = Id$. However, before such a conclusion is reached, two factors must be considered. First, not every surface can be covered with just one chart. If one was to choose other variants of R-4, such as a torus, sphere or cylinder, as the constraint-free manifold depending on the application, then more than one chart would be required. Second, it is desirable to show that the manifold is smooth for the purpose of generating geodesics later on. Then, defining it as a surface aids in defining the atlas of the manifold. If the manifold admits a smooth structure, then concepts of derivatives are possible on the surface. It is because of this, that shortest paths or geodesics that respect the underlying curvature can be defined. Even though this work discretizes the manifold before path planning, the manifold to be discretized is shown to be smooth. We do so to show that a constraint-free space can indeed be represented by a smooth manifold of choice. Additionally, a manifold equipped with the tools to house geodesics will be guaranteed to produce discretized shortest paths (equivalents of geodesics in the discrete representation). Generally speaking, if there exists a chart for an open subset of a manifold, then the inverse of the chart is a parametrization of that subset [174]. Therefore, we can now check if the atlas of patches forms a smooth structure. If it does, it implies that one has a smooth atlas of charts on the manifold.

So, we now show that R-4 as an annulus, from the perspective of a surface and then, show that it admits a smooth structure. The annulus can be shown to be a surface using

Definition 3.16 by choosing the following parametrizations/patches. To define a surface patch, one recalls the definition of parametrization from Definition 3.17. The first patch is defined by $\gamma_\alpha(u, v) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^2 = (\gamma_1(u) \times \gamma_2(v))$ for $-r_o < u < r_o$ and $-r_o < v < r_o$.

$$\gamma_1(u) = u \tag{3.35}$$

$$\gamma_2(v) = v \tag{3.36}$$

This patch covers all of R-4. Therefore, a second patch is not needed. One now proceeds to check that the homeomorphic parametrization is C^∞ by verifying that it has partial derivatives of all orders. In this case, γ_α is C^∞ , and thus is a smooth patch. Next, one checks if the patch is indeed diffeomorphic, i.e., whether the inverse is at least C^1 . In order to do so, we invoke Definition 3.8 locally on the patches. However, it can be quite tedious if there exist several patches. So, one invokes the following corollary from [97]:

Corollary 3.1. *Suppose $U \subseteq \mathbb{R}^n$ is an open subset, and $F : U \rightarrow \mathbb{R}^n$ is a smooth function whose Jacobian determinant is non-zero at every point in U . Then,*

1. *F is an open map.*
2. *If F is injective, then $F : U \rightarrow F(U)$ is a diffeomorphism.*

Using this corollary then, one can check that the determinant of the Jacobian of γ_α is 1. This determinant value is non-zero for all values of (u, v) , as defined above. Therefore the patch is a diffeomorphism. Additionally, the inverse of the patch is also C^∞ . Therefore, the inverse is also smooth. Now, one can formally say that the atlas containing the patch, or equivalently its smooth inverse (chart), is a smooth structure on R-4, i.e., a smooth manifold, as defined in Definition 3.11. If there were more charts, then, similar to Example 3.2, one would check that the transition maps were indeed smooth.

Thus far, we showed that a PPS with one DCS can be represented as an annulus. The circumference marked by the inner radius r_i represents the expanse of the DCS to be avoided. Similarly, the circumference of the outer radius r_o can also be used to represent the bound beyond which there exist constraints as well. For this reason, the former and the latter will be referred to as the Inner Constraint Edge (ICE) and Outer Constraint Edge (OCE), respectively. When the PPP contains more than one DCS, several such R-4s may be connected to represent the entire path planning environment. To do so, one uses the concept of connected-sums of manifolds. Applying the concept of connected sums for the planning of a viable path is one of the future milestones towards the completion of the thesis.

The discretization of the constructed manifold as a step towards the path planning process is described in Section 3.2.2. Compact manifolds admit polygonal representations or discretization. Discretization algorithms may be complex enough to handle variants of compact and non-compact manifolds. However, for completeness, a brief introduction to compactness is provided here. Therefore, the compactness of $\mathcal{T}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$ in the form of R-4, the annulus, will now be briefly discussed. Compactness can be thought of as the finite-ness of open sets used to represent the topological space. Formally, it may be explained with the help of open covers of a topological space, as seen in [98]. The open cover of a space X is a

collection $\mathcal{U}$ whose union is X , and a subcover of $\mathcal{U}$ is a sub-collection of elements of $\mathcal{U}$ that still covers X . A topological space X is said to be compact if every open cover of X has a finite subcover; or in other words, if given any open cover $\mathcal{U}$ of X , there are finitely many sets $U_1, \dots, U_k \in \mathcal{U}$ such that $X = U_1 \cup \dots \cup U_k$ [98]. In other words, if a space contains its supremum and infimum, the space can be understood to be compact. To do so, one invokes the Heine-Borel theorem.

Theorem 3.9 (Heine-Borel). *The compact subsets of $\mathbb{R}^n$ are exactly the closed and bounded ones. [97].*

R-4, as defined thus far in (3.34), is not closed, by definition. That is, it does not contain its boundary points. Then by Theorem 3.9, R-4 is not compact. In order to use R-4 as a compact space, one can choose a closed subset of R-4, such as the following:

$$\{\mathbf{p} : r_i + \epsilon \leq \|\mathbf{p}\|_2 \leq r_o - \epsilon\}$$

Now, by Theorem 3.9, R-4 is closed and contains its boundary points. Therefore, it is compact. An alternative solution is to use the one-point compactification method, which populates an open set with a boundary point at ∞ , thus making the set compact. However, this technique will not be covered here.

3.2.2 Discretizing the Constraint-Free Manifold (CFM) and Path Planning

Section 3.2.1.9 demonstrates the construction of a CFM in PPS_{chosen} and shows that operating in PPS_{chosen} is equivalent to operating in PPS_{alt} , if and only if there exists a diffeomorphism between them. This section uses existing algorithms and tools for discretizing the manifold constructed in PPS_{chosen} , resulting in the **Constraint-Free Discretized Manifold (CFDM)**. The discretization process applies mesh generation procedures to create a polygonal mesh of the CFM in PPS_{chosen} . Then, graph traversal algorithms are used to perform path planning on the CFDM. The implementation of the discretization process using a mesh generator is not elaborated in this work for two reasons. First, existing mesh generating algorithms that can be used to discretize a CFM are readily available in the literature. The differences in such algorithms are due to the construction methods, efficiency, memory, resolution and other attribute-related computational geometry. Analysis of such details are out of the scope of this work. Second, the choice of a mesh generator, in general, is user-dependent. Specifically in this work, the mesh generator chosen is simply to demonstrate a proof of concept for path planning on the CFDM. Therefore, no other justification is attributed to the choice and details of the mesh generator. One may also note, that the same flexibility is true for the path planning algorithm used on the CFDM. Since mesh generators store information about the CFDM using graph theory, any graph-based path planner can be used to compute the paths. For the purposes of this work, the shortest path on the CFDM is desired, therefore A* is used.

Once discretized, the polygonal mesh representation of the CFDM is analogous to a graph. Each graph node is an n-tuple of the CFM in PPS_{chosen} , which was shown to be constraint-free. Therefore, a path on the CFDM is analogous to a path on the original CFM. Note that

this analogy is possible due to the choice of a compact, connected 2-manifold, such as an annulus, in conjunction with Proposition 3.2.

Proposition 3.2. *Every compact surface admits a polygonal presentation. [98, Proposition 6.14].*

With a CFDM at hand, one can now focus on factoring the presence of DCs as described in Section 3.2.1.6. Recall that the first group of DCs includes SCs whose locations and/or numbers are unknown up until after the CFDM has been constructed. Thus at this stage, it is assumed that the locations and number of such constraints is available. From here, the constraints in the first group will be processed to extract representative points. These points will then be defined as the poles of the chosen holomorphic function that acts on contours in the CFDM.

Let there be n SCs defined and grouped into m DCSs using (3.2.1.5). Each DCS is now assigned a representative point, to represent it as a pole. The representative point may be computed as the centroid of the DCS. Thus, representative points for all DCSs located in the first group are as follows:

$$\text{poles} = [\mathbf{p}_1 \quad \cdots \quad \mathbf{p}_i \quad \cdots \quad \mathbf{p}_m] \quad (3.37)$$

Recall, that by virtue of using complex analysis to determine homology classes, the poles are now denoted as complex numbers. At this stage, any further processing such as transformation of the poles and their associated DCSs to the CFDM is performed, need be. Note that one is free to depict these constraints in either space, since they can be transformed between R-1 and the discretized R-4. After transformation to R-4, the poles take the same form as in (3.37), simply transformed to lie on the CFDM. So far, the PPP has the CFDM that is free of SCs and other DCSs now represented as poles. So, in order to be fully equipped to determine homology classes, one needs a holomorphic function f . Note that any holomorphic function may be chosen, provided that the function is holomorphic everywhere on the CFDM. This means that the only poles of the holomorphic function chosen, should be the poles specified by the PPP seen in (3.37). That is f takes the following form:

$$f(z) = \frac{f_1(z)}{(z - \mathbf{p}_1) \cdots (z - \mathbf{p}_i) \cdots (z - \mathbf{p}_m)} \quad (3.38)$$

where $f_1(z)$ is simply a placeholder for the numerator of the function. Then for a straight lined contour between points z_1 and z_2 represented as $z = (1 - \lambda)z_1 + \lambda z_2$, one can compute the following:

$$L_e = \int_e f(z) dz = \int_0^1 f((1 - \lambda)z_1 + \lambda z_2)(z_2 - z_1) d\lambda \quad (3.39)$$

where e denotes an edge of the graph. The above expression has borrowed notation from [18], and denotes the value of the integral of the holomorphic function acting on a contour segment as $L(e)$. In this thesis, $L(e)$ referred to as the class marker value. Thus, for a path consisting of n edges, the class marker value of the path is the sum of the class marker values of the edges, i.e. $\sum_{i=1}^n L(e_i)$. Consider Figure 3.26 which provides the operational workflow of the

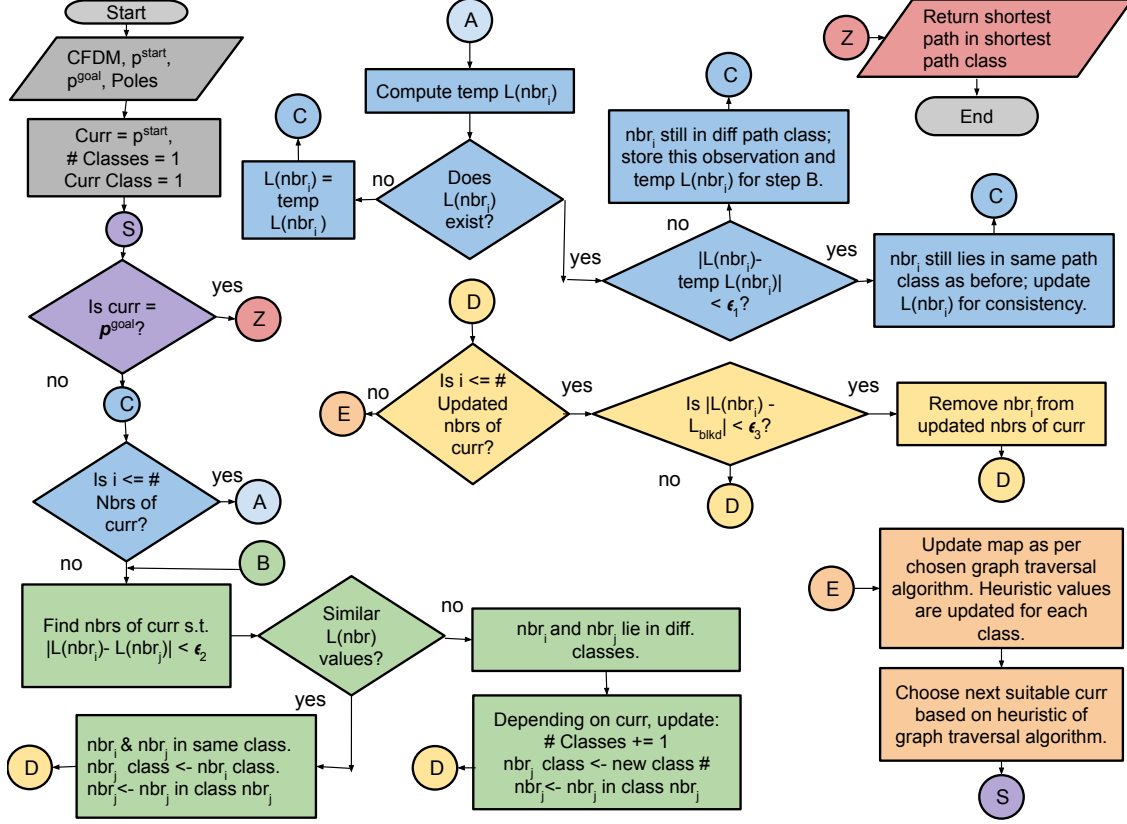


Figure 3.26: Flowchart of the CFDMPP during the class marker value computation

CFDMPP during the computation of the class marker value. The inputs to this stage of the algorithm (and thus the flowchart) are **CFDM**, $\mathbf{p}^{\text{start}}$, $\mathbf{p}^{\text{goal}}$ and the poles representing the **DCs** in the first group. The point cloud representation that describes the expanse of the latter are also passed in as inputs to the **CFDMPP** originally. However, for the purpose of computing class marker values and determining classes, they are not of consequence; the poles are. Thus, the poles are significant inputs for this stage. The flowchart aims to exhibit four important sections of this stage of the algorithm.

The first section involves computing the class marker value of the path from **node**^{start} (discretized $\mathbf{p}^{\text{start}}$) to a node that is currently being encountered via the graph traversal algorithm. The aforementioned node is encountered via its parent node, referred to as the current node and denoted by **curr**. Thus, the encountered node is the neighbour node, denoted by **nbr**. Now, the path from **node**^{start} to **nbr** is the contour being subject to evaluation. But this contour whose class marker value is being computed, has materialized by virtue of **nbr**. So, it is **nbr**'s presence that is being evaluated in different homotopy classes. To mark the via-point node of the contour and to avoid tedious notation, denote the class marker value of a path by virtue of **nbr**_{*i*} as $L(\mathbf{nbr}_i)$. A node can be in several classes at once, since paths in different classes may share a node. That is, several contours may share a physical location. In order to differentiate which class this specific encounter of the node happens in, we compare the $L(\mathbf{nbr}_i)$ to what was computed before. Recall that

the class marker values are different for different classes. In Figure 3.26, this stage initiates at the decision point denoted by "C" and its associated blocks are represented in blue. If the computed $L(\mathbf{nbr}_i)$ and its previously stored values differ by a user defined threshold ϵ_1 , then it is understood that this encounter of the node is via a different homotopy class. If the difference is below ϵ_1 , then $\mathbf{nbr}_i$ is encountered in one of the previously encountered classes. A weighted update between the previous and current values of $L(\mathbf{nbr}_i)$ may be performed to maintain consistency of the class marker value of a class. Once this stage is complete for all n $\mathbf{nbrs}$ of $\mathbf{curr}$, the processing moves to the second section.

The second section involves the different possibilities of paths that occur by virtue of $\mathbf{curr}$ having multiple $\mathbf{nbrs}$. The edge connecting $\mathbf{node}^{\text{start}}$ and $\mathbf{nbr}_i$ augments the existing contour from $\mathbf{node}^{\text{start}}$ to $\mathbf{curr}$. However, since $\mathbf{curr}$ may have several $\mathbf{nbrs}$, the existing contour may now experience n detours, caused by the respective n $\mathbf{nbrs}$. Then, each of the n detoured contours must now be evaluated to check if they are simply deformations of each other, or if they lie in different classes. Recall that the Deformation theorem states that contours may be continuously deformed only if they are holomorphic in the region of deformation. If the detour is indeed a deformation, then it will be holomorphic. If it fails to be holomorphic, then a pole is being encountered. In Figure 3.26, this stage initiates at the decision point denoted by "B" and its associated blocks are represented in green. Similar to the first stage, the class marker values of paths via $\mathbf{nbr}_i$ is denoted as $L(\mathbf{nbr}_i)$. Now, all such n class marker values corresponding to n $\mathbf{nbrs}$ are evaluated against a user-defined threshold ϵ_2 . If the values lie within the threshold, then they are simply deformations of each other. Such $\mathbf{nbrs}$ are now updated so that their class numbers are the same. Their node numbers are updated to reflect their presence in the class. Should the values exceed the threshold, it is understood that these $\mathbf{nbrs}$ lie on distinct, new classes. Their class and node numbers are also updated. With this, the path and its detours via $\mathbf{curr}$ and its $\mathbf{nbrs}$ are fully explored.

The third section involves the second group of DCs as explained in Section 3.2.1.6, which trigger path re-planning. It involves the deliberate avoidance of certain path classes, while retaining all other path planning parameters such as $\mathbf{p}^{\text{start}}$, $\mathbf{p}^{\text{goal}}$, CFDM and the poles. So this stage aims to reject path classes constructed on the CFDM without augmenting the existing constraint data. Since a certain path class is deliberately rejected, it is referred to as blocking [18]. In Figure 3.26, this stage initiates at the decision point denoted by "D" and its associated blocks are represented in yellow. The path class to be blocked is represented by its class marker value and denoted by L_{blkd} and is provided by the user. L_{blkd} is obtained by running the algorithm and keeping track of the class marker values. Then, every $\mathbf{nbr}$'s corresponding $L(\mathbf{nbr}_i)$ is compared with L_{blkd} . If the difference lies within a user-defined threshold of ϵ_3 , then the contour from $\mathbf{node}^{\text{start}}$ to $\mathbf{nbr}_i$ belongs to the blocked path class. Therefore, it must be terminated from being explored further. Once all $\mathbf{nbrs}$ are checked to see if they are homotopically blocked, the algorithm moves to section four.

The fourth section comprises of updating the map based on the most recent encounter of $\mathbf{nbrs}$ via $\mathbf{curr}$. In Figure 3.26, this stage initiates at the decision point denoted by "E" and its associated blocks are represented in orange. All corresponding class increments and respective node number changes are recorded. This section stores different path classes, i.e. different versions of the same map of CFDM. Here, nodes in path classes share the same physical coordinates, but different class and node numbers. One can imagine it as a multi-

layered map, where every layer carries node and path information solely corresponding to that path class. This representation therefore enables different path variants of the same path class to be stored, and eventually evaluated against a metric for the best path. After this multi-class map has been updated, the graph traversal algorithm (A^* in this thesis) of choice chooses the next **curr** according to its heuristic.

The algorithm progresses until the final section denoted by decision point “Z” in red is arrived at. The graph traversal algorithm, which in this case is A^* returns the shortest path denoted by $\mathbf{p}_{\text{path}}$:

$$\mathbf{p}_{\text{path}} = [\mathbf{p}^{\text{start}} \quad \dots \quad \mathbf{p}^i \quad \dots \quad \mathbf{p}^{\text{goal}}] \quad (3.40)$$

where $\mathbf{p}^i$ is the i th point in the path. By virtue of using A^* , this path is the shortest constraint-free path on the **CFDM**. Since the metric that we optimize for is path length, the representative path of every class is the shortest path in its class. Homotopically speaking, this path belongs to a class with the least residues or smallest number of detours. Therefore, the final path returned is the shortest path within the shortest path class. The path seen in (3.40) is guaranteed to be constraint-free since it lies in **CFDM**, thus inherently staying clear of all constraints.

Before we conclude, a few notes are mentioned about the thresholds used in Figure 3.26 and the above explanation. The three thresholds i.e. ϵ_1 , ϵ_2 and ϵ_3 , aid in determining classes and deformations. Ideally, these values are tuned such that they produce ideal results based on the input map and poles. That is, one aims to tune these values such that the number of path classes and the allowed deformations appear visually correct. However, tuning these thresholds is effort intensive, as they are related to each other conceptually. The thresholds are also dependent on the map resolution, number of poles and proximity to poles in a discretized map, and thus is a subject for future research. For the moment, it must be noted that the final number of path classes is dependent on these tuned parameters. Therefore, the final number of path classes generated are usually less than expected. In cases when the tuning parameters are tight, too many classes are generated, i.e. the existence of false positives in terms of a path class. In either case, the desired shortest path is still returned; it is the blocking that is affected positively or adversely. Therefore, the tuning parameters have a significant impact on the number of alternative paths generated.

3.2.3 Path Transformation Between Path Planning Spaces (PPSs)

We now describe the transformation of paths between **PPSs**. The notation of $\mathbf{p}_{\text{path}}$ with respect to its **PPS** of origin must be defined. A path located in $\text{PPS}_{\text{chosen}}$ is represented as $\mathbf{p}_{\text{path}}^{\text{PPS}_{\text{chosen}}}$. Paths in other **PPSs** follow the same notation pattern.

The path formulated in (3.40) is located in $\text{PPS}_{\text{chosen}}$. The transformation of the path to any PPS_{alt} , can be done by using the map defined by $f_{\text{PPS}_{\text{chosen}}}^{\text{PPS}_{\text{alt}}}$. That is, $\mathbf{p}_{\text{path}}^{\text{PPS}_{\text{alt}}} = f_{\text{PPS}_{\text{chosen}}}^{\text{PPS}_{\text{alt}}}(\mathbf{p}_{\text{path}}^{\text{PPS}_{\text{chosen}}})$. The mappings $f_{\text{PPS}_{\text{chosen}}}^{\text{PPS}_{\text{alt}}}$ were defined to avoid problems caused by **SCs**, especially that of invertibility. The maps must at least be homeomorphic, although diffeomorphisms are preferred, since they imply dimensional invariance of the manifolds, as

seen in Theorem 3.2. Therefore, the path obtained in any PPS can be transformed to another PPS without any loss of information.

3.2.4 Summary

The CFDMPP can be simply summarized as formulating a CFM by removing DCSs, discretizing the CFM into the CFDM, planning a path on it and transforming the path to another PPS, if necessary. The DCs include entire path classes that can be avoided. The CFDMPP, as formulated, offers advantages that can be seen as follows. First, the navigable space is guaranteed to be constraint-free by virtue of how it is formulated. Second, the user has flexibility in defining the clearance threshold away from the constraint regions. Third, either all or a subset of the constraint-free navigable space may be formulated as the CFM. Formulating it as a topological space and as a manifold allows us to utilize properties of path-connectedness, compactness and discretization abilities. This theoretical basis also enables defining the concepts of shortest path on a manifold (geodesic), and modification of paths using homotopy and homology for DCs. The fourth advantage is the equivalence of all the constraint-free topological spaces (and eventually as manifolds). This enables the user to choose the space in which to plan a path, and alternatively, transform the path to any topologically equivalent space. The next advantage is the discretization of the CFM resulting in the CFDM. With the formulation of path classes using homology and homotopy on the CFDM, one now has a clear description of the different kinds of paths that may be generated. This presents the final, tremendous advantage of generating alternative paths on CFDM without the need to augment the map with or process constraint data. Homotopic path planning on a discretized manifold combines the advantage of graph theory, user-defined underlying topology and practical use of generating alternative paths. It avoids the need to solve complex geodesic equations, while working on a polygonal representation of surface known to house geodesics; thus using the best of theoretical and discrete formulations.

Chapter 4

Results and Analysis

In this chapter, we present results from simulations conducted using the proposed [CFDMPP](#) as described in Chapter 3. The results are categorized into three incremental stages, as introduced below.

The first stage of results is seen in Section 4.1. Here, the [CFDMPP](#) is demonstrated on a simple environment containing [SCs](#), which are known apriori. The [PPM](#) is formulated to avoid such [SCs](#). Path planning is then performed on the resulting [CFDM](#). The aim of this stage of results is to establish proof of concept of the [CFDMPP](#) and demonstrate its fundamental advantages. The second stage of results is seen in Section 4.2. Here, the [PPS](#) contains [SCs](#) as factored apriori and other [SCs](#) that can appear on the [PPS](#), after the formulation of the original [PPM](#). This stage demonstrates the capabilities of the proposed [CFDMPP](#), when [DCSs](#) are encountered on the originally formulated [CFDM](#). Additionally, the performance of the [CFDMPP](#) is bench marked against those of the [PRM](#) and [RRT](#). The performance metrics used for bench marking are defined. The final stage of results is seen in Section 4.3 and is accumulated from simulations on a warehouse type of environment. The design of the [CFDMPP](#) is primarily motivated to perform in such an environment. This category demonstrates the ability of the [CFDMPP](#) to generate paths in typical autonomous warehouse settings such as surveillance, stocking and cleaning. Furthermore, the results demonstrate the ability of [CFDMPP](#) to generate alternative constraint-free paths, should the originally generated constraint-free paths either be rendered invalid, or simply not be desired as the current path.

In each result category, the simulation parameters and set-up are described. Then, the obtained results are illustrated and analyzed. Finally, an interpretation of the significance of the experiment is provided.

4.1 Stage 1: Path planning on the CFDM

4.1.1 Setup

The example considered is that of a **UGV** with an on board manipulator for retrieve-and-place operations, navigating on a planar environment, such as a warehouse floor. In this example, the **PPP** considers only a planar warehouse floor. The preliminary simulations conducted so far only deal with **SCs**. **DCs** will be dealt with after the proposal defense. Path planning is performed by following the sequence of steps detailed in Section 3.2.

With its singular **PPS** = $\mathcal{W}$ = **PPPS**, $|\mathbf{PPSs}| = 1$. Since the workspace is planar in nature, the dimension of the **PPPS** is $d^{\mathbf{PPPS}} = 2$. The **PPPS** is taken as the chosen **PPS** and is set as

$$\mathbf{PPS}_{chosen} = \{(x, y) \mid (x, y) \in (-11 \text{ m}, 10 \text{ m}) \times (-10 \text{ m}, 10 \text{ m})\} \quad (4.1)$$

There exists no mapping to any alternative **PPS**, so we proceed to analyze the information about **SCs**. The **SCs** consists of a storage box on top of a shelving unit, and a glass wall on one edge of the **PPS**. In general, shelving units are classified as obstacles rather than no-go zones. This is so, since the manipulator may still need to get close to the shelving unit for any retrieve-and-place operations, while still maintaining sufficient clearance to prevent collision. Classifying it as an obstacle therefore, implies that a less stringent clearance may be used by the user. On the other hand, the glass wall is classified as a no-go zone, implying that the user must signify a higher clearance value. Then, the total number of **SCs** in each category and the total number of **SCs** are

$$n_{\text{NG}} = 1 \quad n_{\text{OB}} = 2 \quad \Rightarrow n_{\text{SC}} = 3$$

All the **SCs** are native to the **PPPS** and their corresponding bounds are

$$\begin{aligned} OB_1 &= \{(x, y) \mid (x, y) \in [0 \text{ m}, 1 \text{ m}] \times [-1 \text{ m}, 0 \text{ m}]\} \\ OB_2 &= \{(x, y) \mid (x, y) \in [-2.414 \text{ m}, 2.414 \text{ m}] \times [-2.414 \text{ m}, 2.414 \text{ m}]\} \\ NG_1 &= \{(x, y) \mid (x, y) \in [-11 \text{ m}, -10.75 \text{ m}] \times [-10 \text{ m}, 10 \text{ m}]\} \end{aligned}$$

Applying Algorithm 3.2 with a $\Delta_{\text{CS}} = 1 \text{ m}$ leads to two **DCSs** in **PPS**_{chosen}.

$$\mathbf{DCS}_1 = \{(x, y) \mid (x, y) \in [-2.414 \text{ m}, 2.414 \text{ m}] \times [-2.414 \text{ m}, 2.414 \text{ m}]\} \quad (4.2a)$$

$$\mathbf{DCS}_2 = \{(x, y) \mid (x, y) \in [-11 \text{ m}, -10.75 \text{ m}] \times [-10 \text{ m}, 10 \text{ m}]\} \quad (4.2b)$$

If one were to account for **DCs** in this example, **PPPS** would be the native **PPS** for all such **DCs**. By default, the **PPPS** becomes the **PPS**_{chosen}. Since **PPS**_{chosen} is already completely defined, no further processing is necessary.

One can now proceed to develop the topological representation of **PPS**_{chosen}. **PPS**_{chosen} may be defined as the set $\mathcal{S}_{\mathbf{PPS}_{chosen}}$ in (4.1), with the metric topologies associated with the $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$ metrics. The set $\mathcal{S}_{\mathbf{PPS}_{chosen}}$ together with the topologies induced by the

metrics define $\mathcal{TS}_{\text{PPS}_{\text{chosen}}}$. Now, the constraint-free topological representation is extracted. Both DCS s are native to $\text{PPS}_{\text{chosen}}$, so they can be removed in any order. It is up to the user to include clearance values either as part of the bounds of the SC s, as part of the bounds of the DCS s, or even during the construction of the CFM . Here, the clearance value is added to the bounds of only one DCS originally seen in (4.2b). The new DCS s become

$$\text{DCS}_1 = \{(x, y) \mid (x, y) \in [-2.414 \text{ m}, 2.414 \text{ m}] \times [-2.414 \text{ m}, -2.414 \text{ m}]\} \quad (4.3a)$$

$$\text{DCS}_2 = \{(x, y) \mid (x, y) \in [-11 \text{ m}, -10 \text{ m}] \times [-10 \text{ m}, 10 \text{ m}]\} \quad (4.3b)$$

The bounds of DCS_1 remain unchanged, and can be changed during the construction of the CFM . The clearance value of $\epsilon_2 = 0.75 \text{ m}$ has been used to increase the clearance from the glass wall in DCS_1 , only along one dimension. Removing the sets of points constituting DCS_1 and DCS_2 , as defined in (4.3), yields

$$\mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}} = \mathcal{S}_{\text{PPS}_{\text{chosen}}} \setminus (\mathcal{S}_{\text{DCS}_1} \cup \mathcal{S}_{\text{DCS}_2})$$

Then, the resulting constraint-free set of points can be expressed as

$$\mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}} = \{\mathbf{p} = (x, y) \mid r_1 < \|\mathbf{p}\|_{\infty} < r_2\} \text{ for } r_1 = 2.414 \text{ and } r_2 = 10 \quad (4.4)$$

$\mathcal{S}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$ inherits the metric topology of $\mathcal{S}_{\text{PPS}_{\text{chosen}}}$. This topological space is denoted by $\mathcal{TS}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$. (4.4) shows that the constraint-free region takes the form of R-1 as defined in Chapter 3, as seen in Figure 4.1. Using the same set of homeomorphisms, we can formulate the annulus, which is this application's CFM of choice on $\mathcal{TS}_{\text{PPS}_{\text{chosen}}}^{\text{free}}$. Section 3.2.1.9 showed that each homeomorphism in the composition of $h_{\text{R-3}}^{\text{R-4}} \circ h_{\text{R-2}}^{\text{R-3}} \circ h_{\text{R-1}}^{\text{R-2}} = h_{\text{R-1}}^{\text{R-4}}$ transformed from R-1 to the punctured plane, then to the punctured disc and finally to the annulus of the desired size. Thus, here, all that is left to do is to simply choose r_i and r_o that define the annulus. In the context of the annulus, r_i defines the radius of DCS_1 's region to be avoided. Thus, one can introduce an additional clearance here, if necessary. In this example, the corners of DCS_1 in R-1 must be avoided, therefore r_i is chosen to account for that. Utilizing the metrics defined, r_i can be expressed as a function of r_1 . So, $r_i = r_1 \times \|(1, 1)\|_2 = 1.414r_1 = 3.414 \text{ m}$, is the minimum r_i that encompasses DCS_1 with its corners. An additional clearance may now be defined if chosen, such as $r_i + \epsilon_c$. In this example, no such additional clearance is defined. Next, choose $r_o = 10 \text{ m}$ to cover as much of the constraint-free space as possible. Then, the CFM as defined by the annulus and denoted by R-4 is given by:

$$\text{R-4} = \{\mathbf{p} = (x, y) \mid r_i < \|\mathbf{p}\|_2 < r_o\} \quad r_i = 3.414 \text{ m} \quad r_o = 10 \text{ m} \quad (4.5)$$

The ICE and OCE of this R-4 space are shown in Figure 4.2.

The annulus is now discretized into a meshed CFDM , as the one shown in Figure 4.3(a) (ignore the path in the figure for now as it is not yet computed). The mesh generator adopted in this work is DistMesh [125, 126], which comes with an open source MATLAB code. DistMesh triangulates implicitly defined geometry using distance functions and uses Delaunay triangulation to reset the topology periodically [126]. Although DistMesh is adopted here, any other mesh generator to produce graphs can be used. Once a polygonal representation of the manifold is generated, a graph based algorithm is applied to compute the path on

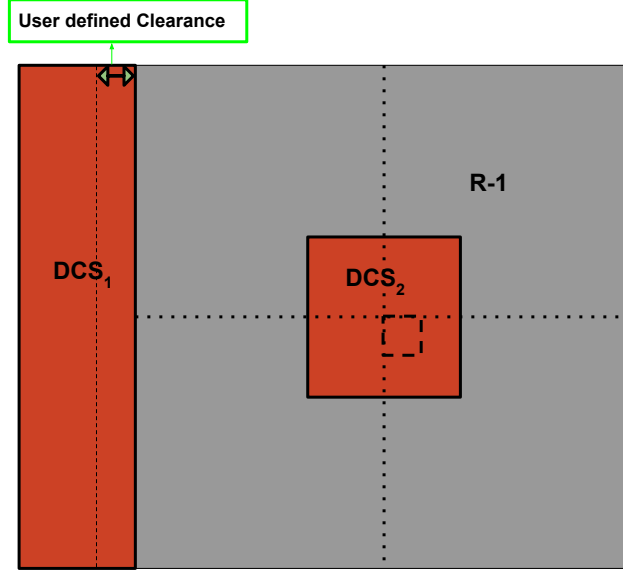


Figure 4.1: R-1 space

this discretized representation of the manifold. In this discretized manifold, the equivalent of a tuple is what represents a node on the graph. This work uses A* as the path planning algorithm to compute the shortest path between the start and goal nodes. A* is chosen for computational ease and for proof of concept of path planning on a CFDM. One may choose other alternatives to A* as graph-based path planners. The resulting path, $\mathbf{p}_{\text{path}}$, generated by A*, takes the form of an ordered sequence of nodes, as defined in (3.40).

4.1.2 Results

Four cases are used for illustration, as shown in Figures 4.3 to 4.6. All four figures mark $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ by blue and red squares, respectively. The final path is indicated in green, with green circles representing intermediate nodes. The large size of green circles are used for visibility and do not convey any physical meaning of the UGV positions or encroachment of nodes on un-meshed regions. Note that DistMesh produces different numbers of nodes and different physical locations for the respective node numbers every time it is run.

4.1.2.1 Case 1

In the first case, $\mathbf{p}^{\text{start}} = [-9.7029 \text{ m}, -2.4193 \text{ m}]$ and $\mathbf{p}^{\text{goal}} = [10.0000 \text{ m}, 0.0205 \text{ m}]$. Running DistMesh produces a total of 432 nodes, where $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ correspond to $\text{node}^{\text{start}} = 1$ and $\text{node}^{\text{goal}} = 432$, respectively (i.e., their corresponding node numbers on the graph). The resulting path is shown in Figure 4.3.

This case shows the nature of a path that crosses from one side of the OCE to the other, while traveling through the interior of R-4 and along the ICE. Such a case is important to consider because, the path originates and ends at the borders of constraint regions,

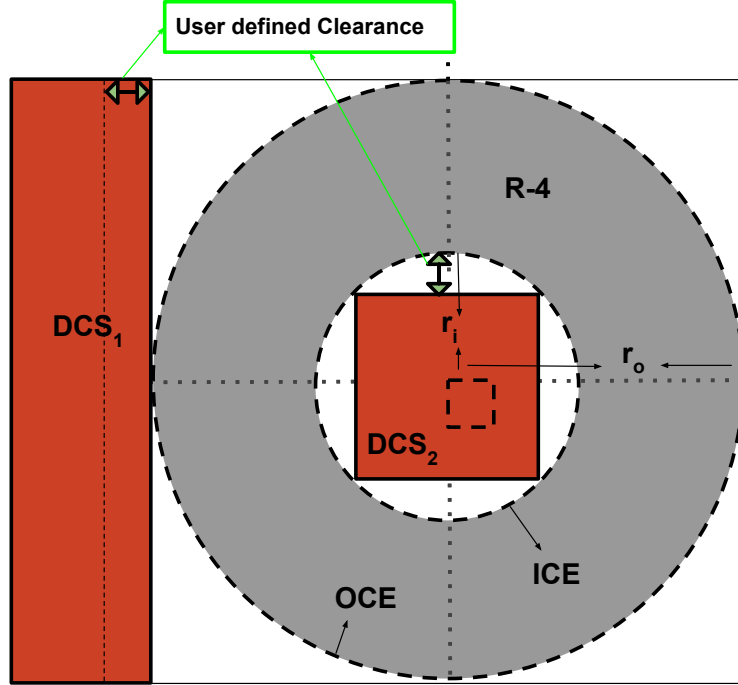


Figure 4.2: R-1 and R-4 of the warehouse example with a glass wall, shelving unit and a box atop the shelving unit

and even travels along the border for a part of the path. Boundaries of constraint regions are problematic, because of undesirable behaviour that may result from interacting with them. While **SAs** would have to periodically evaluate the sampled points for **SC** evaluation, the **CFDMPP** completely avoided that. Given that the environment to sample spans approximately 100 m^2 , the issue of periodic constraint-evaluation is a pertinent one. This is especially so, since this path borders the constraint edges and regeneration of samples is necessary. Comparing with variants of **OAs**, local minima and stagnation of paths are not encountered here. This is so, because the **CFDMPP** avoids the constraints by default, and no additional potential/navigation functions need to be formulated to avoid constraints. To accomplish a robot position that is not in the vicinity of the starting point, the shortest path as indicated by A^* involves traveling along the **ICE**, whose clearance from the actual constraint region is defined by the user. That is, the shortest path by definition involves finding nodes that may border the constraint space. However, since the **CFM** on which A^* is operating, is constraint-free by definition as seen in Chapter 3, there is no necessity to evaluate the individual nodes to check for constraints. The **OCE** and **ICE** of R-4 and its discretized counterpart may resemble and border the actual constraint spaces. But, the original definition of the space combined with the additional user-defined clearance guarantees that the path generated will always be constraint free. This is clearly seen in Figure 4.3(b), where the path generated by the **CFDMPP** is shown, as it looks in the original **PPS**, i.e. R-1. One can clearly see how this path can be visualized in a warehouse setting, where the path must border the shelving unit (at a preferred proximity) and traverse the rest of the space as necessary. Note that, the transformed path does not appear to be the shortest, since path length is a function of discretization resolution; it can be changed using parameters of the

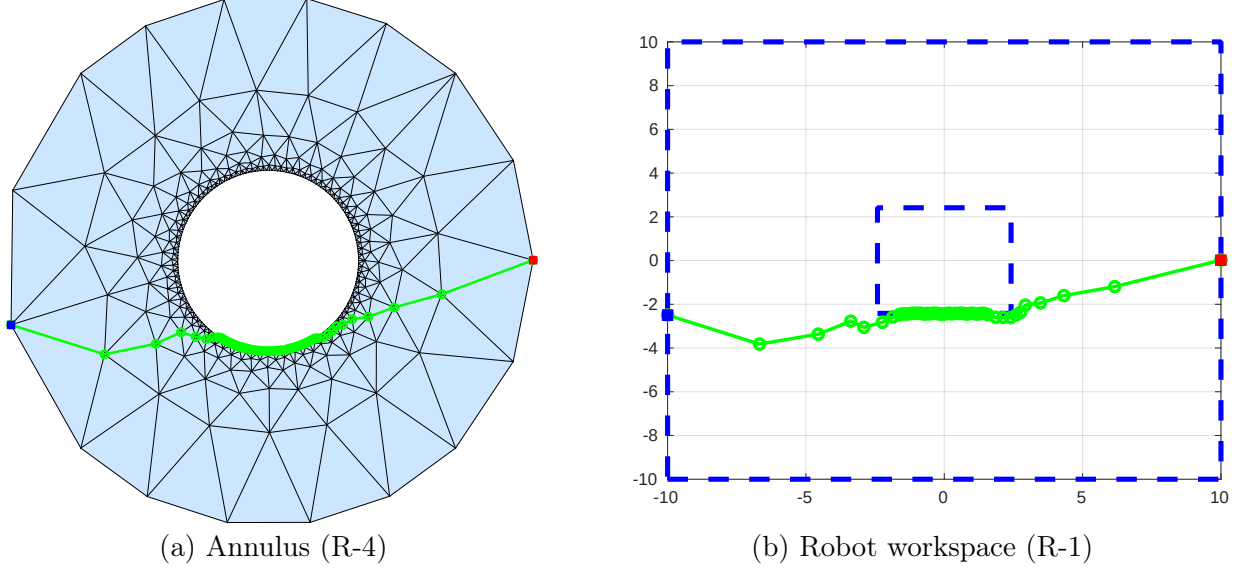


Figure 4.3: Case 1: Generated path. (a) Path on R-4, starting at the OCE, bordering the ICE and terminating at the OCE. (b) Equivalent path in the robot workspace.

meshing algorithm used. In this case, the default discretization topology has been used as suggested in DistMesh, for demonstration purposes.

4.1.2.2 Case 2

In Case 2, we take $\mathbf{p}^{\text{start}} = [3.3929 \text{ m}, 0.3789 \text{ m}]$ and $\mathbf{p}^{\text{goal}} = [9.9994 \text{ m}, -0.1080 \text{ m}]$. Executing DistMesh produces 448 nodes, where $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ correspond to $\text{node}^{\text{start}} = 400$ and $\text{node}^{\text{start}} = 448$, respectively. This case displays the ease with which one can compute the shortest path while traversing regions that stay clear of constraints. The algorithm does not need to check for constraints at every step (node). It just computes the path with no additional resources or unnecessary computational overhead. The generated path is revealed in Figure 4.4. This case shows that the CFDMPP handles multi-query, constraint-free path planning with ease. Even in this simple case, where constraint edges only house the source and target locations, variants of the OA would have re-calibrated their cost functions for the new target and source locations. Depending on the variant of SAs used, any of the following may happen. Either the tree is built again to account for the new target location or, if the learning phase does not contain points close to the source and target locations, then new points must be learnt, and finally, evaluation of constraints still occur. The CFDMPP's multi-query and non-cost based nature, does not encounter the above situations, and the resulting path from the CFDMPP transformed to R-1 is seen in Figure 4.4(b).

4.1.2.3 Cases 3 and 4

Cases 3 and 4 are presented and analyzed together, to show the effectiveness of using topological tools to manipulate spaces and the invariance of topological properties up to home-

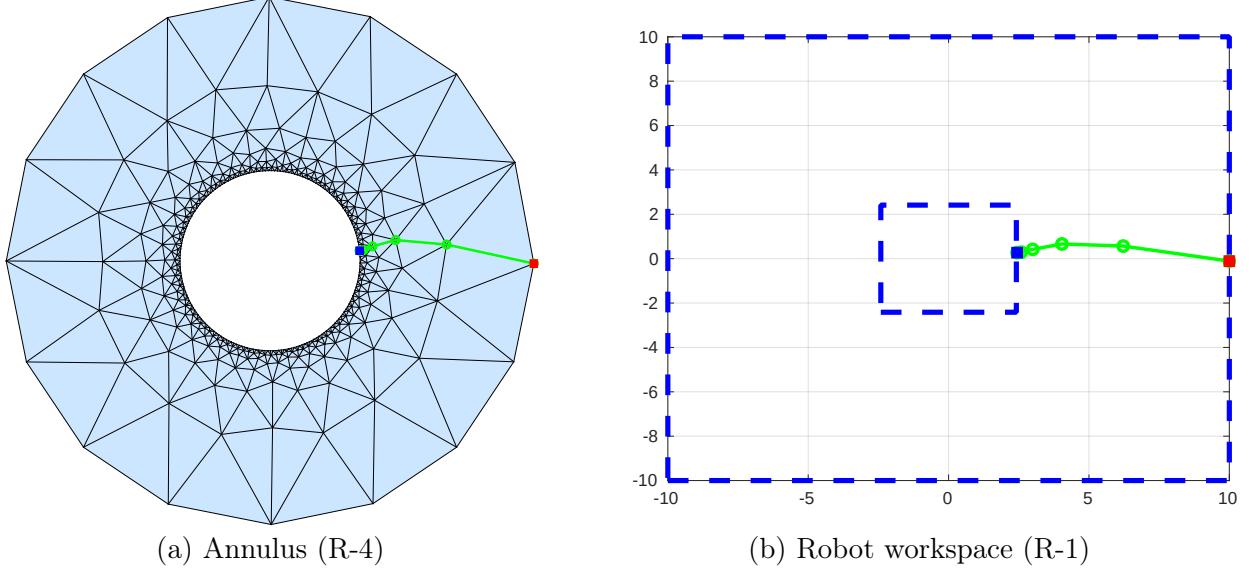


Figure 4.4: Case 2: Generated path. (a) Path on R-4 traversing through the interior. (b) Equivalent path in the robot workspace.

omorphisms. In Case 3, $\mathbf{p}^{\text{start}} = [-3.4139 \text{ m}, -0.0206 \text{ m}]$ and $\mathbf{p}^{\text{goal}} = [3.4133 \text{ m}, 0.0709 \text{ m}]$. Applying DistMesh leads to 496 nodes, with $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ being $\text{node}^{\text{start}} = 52$ and $\text{node}^{\text{goal}} = 445$, respectively. The resulting path is depicted in Figure 4.5.

Case 4 produces 459 nodes, with $\mathbf{p}^{\text{start}} = [-9.8827 \text{ m}, 1.5271 \text{ m}]$ at $\text{node}^{\text{start}} = 1$ and $\mathbf{p}^{\text{goal}} = [-8.9154 \text{ m}, -4.5294 \text{ m}]$ at $\text{node}^{\text{goal}} = 4$. The resulting path is shown in Figure 4.6.

The paths of both cases originate and end along the constraint borders. Case 3 produces a path that lines the OCE while Case 4's path lines the ICE. The topological transformation, i.e. the homeomorphism, shows that R-1 transforms to R-4 and vice versa. Path-connectedness is an important topologically invariant property under homeomorphisms, and it is clearly illustrated in these cases. A path bordering the constraint edge in R-4 is still a path bordering the equivalent constraint edge in R-1. That means, no accidental encounter with a previously accounted for SC occurs, irrespective of which space, i.e. R-1 or R-4, the path is computed on. This is of prime importance for consistent constraint avoidance and is the main strength of the CFDMPP. Using topological tools provides the underlying connectivity of the space, which, once discretized serves as a reliable source of information. Typical path planning applications for robotic systems experience discrepancies in velocities commanded and experienced, when operating close to the constraint space, and thus need to be consistently checked to see whether the robot is in such a configuration. By pre-defining a set clearance threshold from both ICE and OCE of the manifold, the two cases demonstrate the absence of computing a new cost function and the evaluation of constraints. While the paths may appear to trace the inner and outer edges, these edges are pre-determined to guarantee that even nodes on these edges stay well clear of the constraint spaces. Since the paths shown are in $\mathcal{TS}_{\text{chosen}}^{\text{free}}$, one can transform them to $\mathcal{TS}_{\text{alt}}^{\text{free}}$, as seen in Section 3.2.3, if necessary. However, such a transformation is not necessary here.

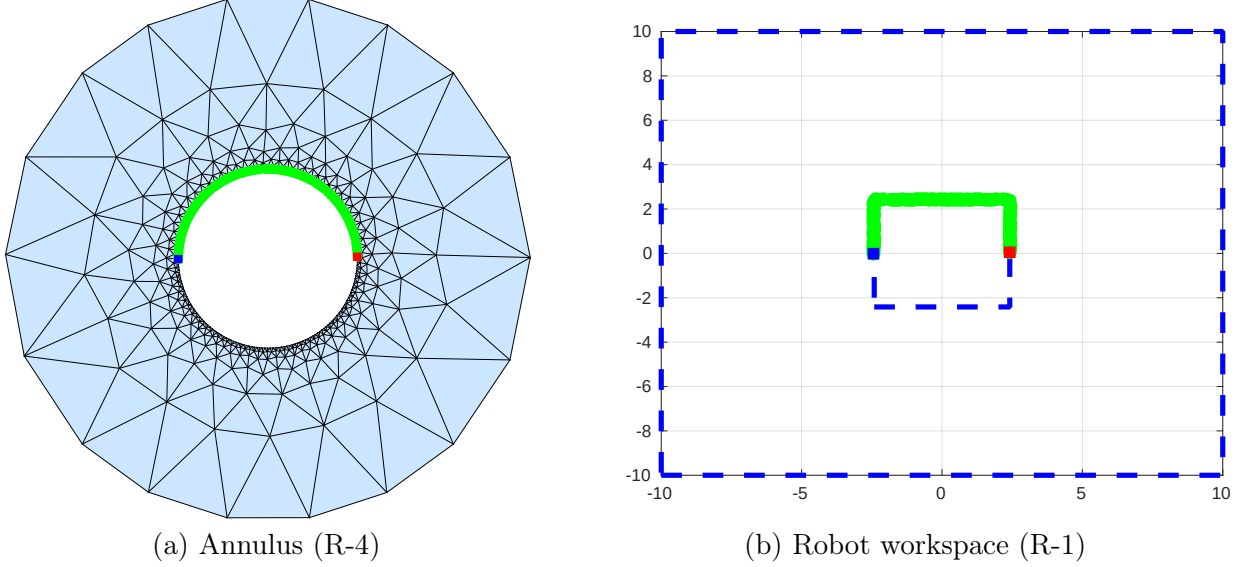


Figure 4.5: Case 3: Generated path. (a) Path on R-4 traveling purely on the ICE. (b) Equivalent path in the robot workspace.

4.1.3 Analysis

The example used shows how a realistic PPP can make use of the proposed CFDMPP. The goals of using such an example are to show the following points. First, it shows the feasibility of the CFDMPP on a subset of the Euclidean space. This is necessary, since the algorithm aims to construct a manifold in a topological space. The Euclidean space is a topological space, and all/subsets of it may be formulated as a manifold, as was done in the example. The sequence of steps that show the construction of the topological space and that of the manifold are generally the same for non-planar manifolds. So, by showing that one can indeed mathematically construct a manifold from a space and plan a path on it, we show the feasibility of the CFDMPP. As such, one becomes familiar with the tools that come with manipulating manifolds and topological spaces. The tools can help with representing the same PPS using different manifolds, representing a PPS with multiple DCSs and continuously deform paths on the manifold.

Chapter 3 and the example above used homeomorphisms to construct an annulus. The composition of homeomorphisms was a tool to represent the same PPS using different representations, i.e., punctured plane/disc and annulus. The advantage of this flexibility is not obvious because this is a planar case. So dilations/contractions and smoothing of corners may not be significant. However, it can be of great value if the surface to navigate on was a cylinder, for example. It can be shown that a cylinder is homeomorphic to an annulus, which is in turn homeomorphic to the punctured disc. Recall that a path-connectedness is invariant up to homeomorphisms. A homeomorphism maps every point from one surface to some point on the target surface. Then, any path computed on one surface can be transformed to the target surface using the homeomorphism. So now, let us revisit the example from the simulation and the cylinder, side by side. In the simulated example, the path generated on the final annulus can be transformed to any of the intermediate manifolds. In this case,

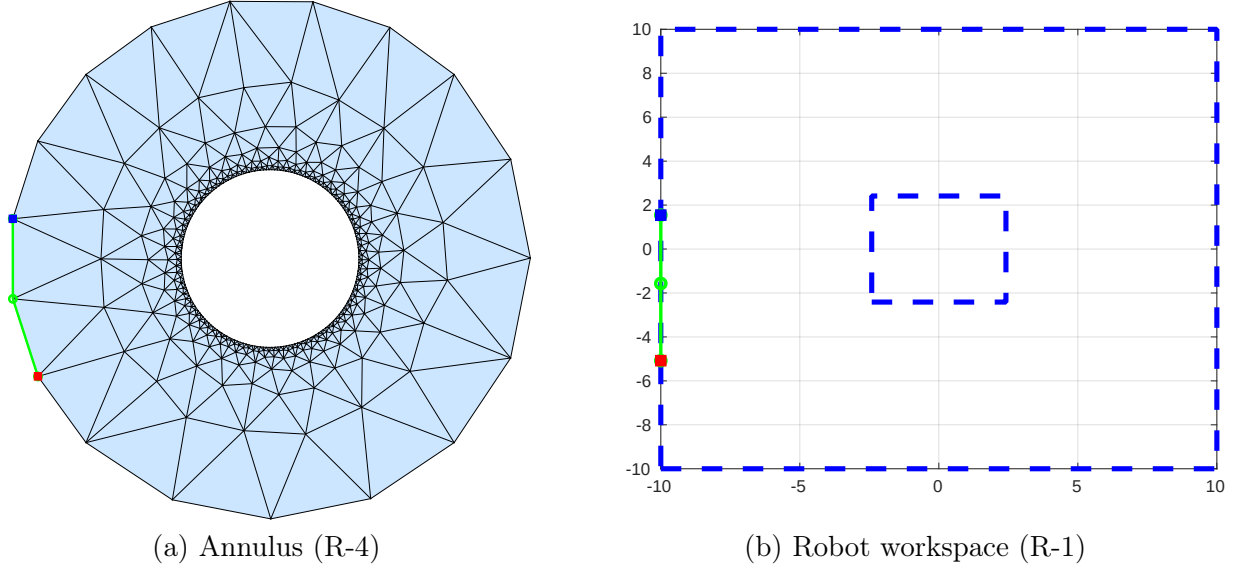


Figure 4.6: Case 4: Generated path. (a) Path on R-4 traveling purely on the OCE. (b) Equivalent path in the robot workspace.

R-1 and R-4 are what represent the [PPS](#) in its true dimensions. So, the transformation merely shows the capability to represent the same space and path differently. However, in the case of a cylinder, one can show that a path on the cylinder is now homeomorphic to a path on the annulus. This can present itself as an advantage, when the [PPM](#) needs to be represented as a cylinder around a constraint. Hence, rather than planning a path on a three-dimensional surface, one can compute the path on its equivalent homeomorphism, i.e. annulus, and transform the path back to the original [PPM](#). It is important to know that most surfaces can be represented as some function/modification of a few fundamental topological spaces. Two such fundamental spaces are $\mathbb{R}^n$ and $\mathbb{R}^n \setminus \mathbf{0}$. We have already seen that the punctured plane $\mathbb{R}^n \setminus \mathbf{0}$ may be used to construct the annulus and a cylinder. Additionally, it is also equivalent to a sphere with its North and South poles removed. So, one can see how the fundamental space can eventually, through manipulation, become topologically equivalent to another space. By demonstrating a tool that creates a manifold by manipulating another manifold on a subset of the Euclidean space, we show that there are several uses for such manipulation.

Homeomorphism is a fundamental concept. However, it is not the only tool that builds new spaces from old. Concepts of subspaces, quotient spaces and connected sums are examples of tools that enable constructing new spaces. Each comes with a variety of restrictions and advantages. Subspaces were used previously in Chapter 3 and in this chapter, and are useful in inheriting the topology for a restricted subset of the original topological space. Quotient spaces and connected sums go hand in hand. They enable connecting either parts of the same manifold or different manifolds together. Apart from requiring manipulation tools to change the surface of computation, one may also use such tools to create a complete representation of the environment. Consider a torus, which may be used for non-planar navigation, such as for a [UAV](#). One such application can be aerial surveillance and monitoring

in a warehouse with high shelving units separated by aisle space, very similar to a Costco warehouse, for instance. Then, in a manner similar to how the annulus was used to circumvent the [SC](#), the torus may also be used such that the [SC](#) slots into its genus. There exists several kinds of geodesics on a torus, allowing the [UAV](#) to take multiple smooth paths on the surface. An example of such paths computed on a discretized torus (using DistMesh) is seen in Figure 4.7. When there exist multiple [SCs](#), the connected-sum technique is used to create an equivalent representation. For example, consider Figure 4.8, where two tori have been manipulated to become one connected manifold. This new surface, can now be used to generate paths that are guaranteed to stay clear of two [SCs](#). A similar analogy for annuli may be used to represent a [PPS](#) with multiple [DCSs](#). This is part of the future work which will be included in the thesis.

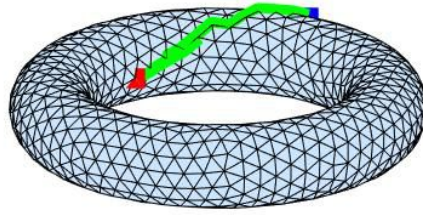


Figure 4.7: Geodesic on the Constraint-Free Discretized Manifold (CFDM) of a torus

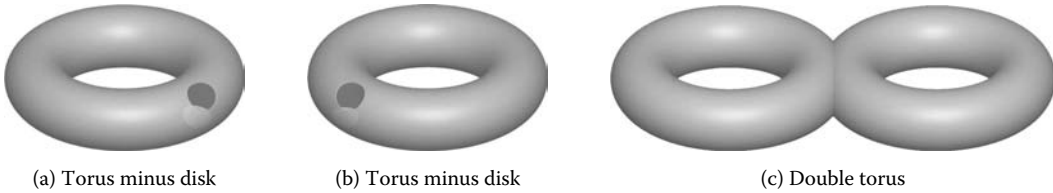


Figure 4.8: Connected sum of two genus-1 tori [\[111\]](#)

The second goal of the [UGV](#) example presented in this chapter, is that it shows the systematic sequence of steps to apply for any kind of [PPS](#). That is, one constructs the topological space with the necessary metrics, and then uses any of the aforementioned tools to construct manifolds for path planning. There exist other manifold surfaces which can be similarly utilized, such as the sphere S^2 . Consider an example, where the [PPP](#) involves a [UAV](#) equipped with a sensor to gather 3-D point cloud data-set about an object. Then, one expects the [UAV](#) to be able to traverse around the object, with a change in elevation periodically while not colliding with object itself. For that, we propose modeling a sphere enveloping the object as the manifold to traverse on. Every great circle on the sphere is a geodesic, and every segment of a geodesic is also a geodesic. While parallel circles of latitude and longitude are not geodesics, the shortest path between any two points on a sphere can be obtained by a segment of a great circle. In conclusion, smooth and length-minimizing paths can be produced for the sensor equipped [UAV](#) to collect data about the object from

all angles and elevations of interest. Another example is that of automated quality control, where unmanned systems use sensors, such as cameras, to collect data about the finished product. Paths generated on this surface are smooth and collision-free by virtue of defining the clearance, which in this case is the radius of the sphere. A path on a discretized sphere (using DistMesh) is seen in Figure 4.9.

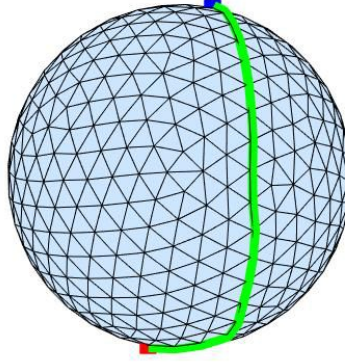


Figure 4.9: Geodesic on the Constraint-Free Discretized Manifold (CFDM) of a sphere

Finally, a systematic construction of a topological space along with tools to modify manifolds establishes the basis required for techniques used to modify paths on the surface. As previously mentioned, concepts of homology and homotopy enable deforming a specific path on a surface. In the event that a DC is encountered, the path may be deformed to avoid it. This is far less computationally intensive as opposed to re-modeling the manifold to account for the newly seen DC. This is another proposed area of future work to be added in the final thesis.

In summary, the proposed CFDMPP introduces the benefits of combining knowledge of surfaces that are known to have favorable properties, and then applying computational geometry to compute paths on such surfaces. This conglomeration has the potential to be exploited in environments where the map is known, and the priorities are path length minimization and constraint avoidance.

4.2 Stage 2: Path planning with DCs on the CFDM

4.2.1 Setup

The example considered here is similar to the planar example of Section 4.1 with a UGV. This example however, considers a PPM of a bigger scale and the presence of point representations of DCs on the formulated PPM. Therefore, formulating the CFM follows the same sequence of steps as outlined in Section 3.2, similar to what was done in Section 4.1.

With its singular $\text{PPS} = \mathcal{W} = \text{PPSs}$, $|\text{PPSs}| = 1$. Since the workspace is planar in nature, the dimension of the PPS is $d^{\text{PPS}} = 2$. The PPS is taken as the chosen PPS and

is set as

$$\text{PPS}_{\text{chosen}} = \{(x, y) \mid (x, y) \in (-102.5 \text{ m}, 102.5 \text{ m}) \times (-102.5 \text{ m}, 102.5 \text{ m})\} \quad (4.6)$$

There exists no mapping to any alternative **PPS**, so we proceed to analyze the information about **SCs**. This example is assumed to have observation chambers along all edges of the **PPS**. Additionally, there exists a freight elevator located adjacent to two wide stairwells, all of which are placed within close proximity of the central support column, as seen in Figure 4.10.

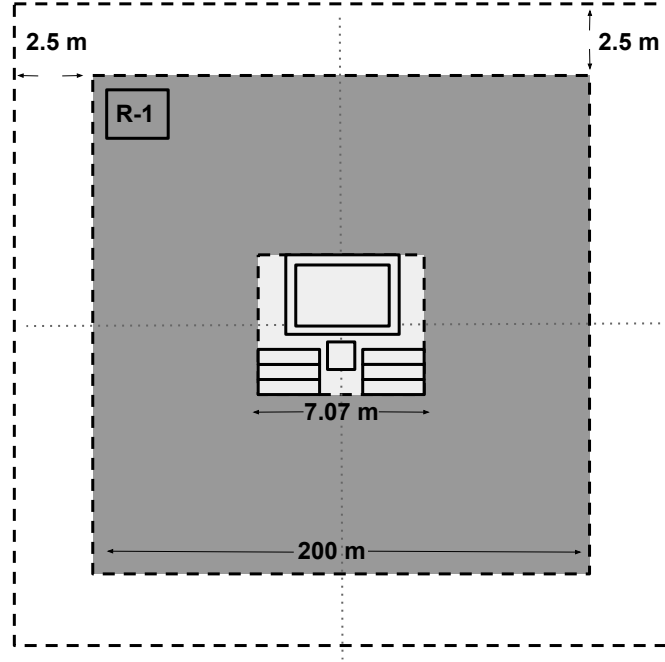


Figure 4.10: Simulation environment for stage 2

The two staircase decks are classified as no-go zones, since the **UGV** is assumed to not handle elevation changes that occur in a staircase. The elevator is considered a no-go zone, as it is restricted for human/freight access and the central column is considered as an obstacle. The observation chambers (one along each edge) are classified as no-go zones. No additional clearance has been chosen to demarcate the **SCs** from its surroundings. Then, the total number of **SCs** in each category and the total number of **SCs** are:

$$n_{\text{NG}} = 2 + 1 + 1 + 4 \quad \Rightarrow \quad n_{\text{SC}} = 8$$

All the **SCs** are native to the **PPPS** and their corresponding bounds are mentioned in the

order as introduced above.

$$\begin{aligned}
NG_1 &= \{(x, y) \mid (x, y) \in [-3.5361 \text{ m}, -1 \text{ m}] \times [-3.5361 \text{ m}, -1 \text{ m}]\} \\
NG_2 &= \{(x, y) \mid (x, y) \in [1 \text{ m}, 3.5361 \text{ m}] \times [-3.5361 \text{ m}, -1 \text{ m}]\} \\
NG_3 &= \{(x, y) \mid (x, y) \in [-2.667 \text{ m}, 2.667 \text{ m}] \times [-0.1215 \text{ m}, 3.5361 \text{ m}]\} \\
OB_1 &= \{(x, y) \mid (x, y) \in [-0.5 \text{ m}, 0.5 \text{ m}] \times [-1.25 \text{ m}, -0.25 \text{ m}]\} \\
NG_4 &= \{(x, y) \mid (x, y) \in [-102.5 \text{ m}, 102.5 \text{ m}] \times [-102.5 \text{ m}, -100 \text{ m}]\} \\
NG_5 &= \{(x, y) \mid (x, y) \in [-102.5 \text{ m}, 102.5 \text{ m}] \times [100 \text{ m}, 102.5 \text{ m}]\} \\
NG_6 &= \{(x, y) \mid (x, y) \in [-102.5 \text{ m}, -100 \text{ m}] \times [-100 \text{ m}, 100 \text{ m}]\} \\
NG_7 &= \{(x, y) \mid (x, y) \in [100 \text{ m}, 102.5 \text{ m}] \times [-100 \text{ m}, 100 \text{ m}]\}
\end{aligned}$$

Applying Algorithm 3.2 with a $\Delta_{CS} = 0.7 \text{ m}$ leads to two DCSs in PPS_{chosen} .

$$\text{DCS}_1 = \{\mathbf{p} = (x, y) \mid \|\mathbf{p}\|_\infty \leq 3.5361 \text{ m}\} \quad (4.7a)$$

$$\text{DCS}_2 = \{\mathbf{p} = (x, y) \mid 100 \text{ m} \leq \|\mathbf{p}\|_\infty \leq 102.5 \text{ m}\} \quad (4.7b)$$

Naturally, PPPS would be the native PPS for any potential DCSs. By default, the PPPS becomes the PPS_{chosen} . Since PPS_{chosen} is already completely defined, no further processing is necessary.

The topological representation of PPS_{chosen} is defined using the same sequence of steps as seen in Section 4.1 and initially in Section 3.2. Thus, the same details are spared here, to avoid redundancy. In this example, no further user defined clearance is chosen to separate the DCSs from their surroundings. Thus, the constraint-free topological space is obtained using (4.7):

$$\mathcal{S}_{\text{PPS}_{chosen}}^{free} = \mathcal{S}_{\text{PPS}_{chosen}} \setminus (\mathcal{S}_{\text{DCS}_1} \cup \mathcal{S}_{\text{DCS}_2})$$

Then, the resulting constraint-free set of points can be expressed as

$$\mathcal{S}_{\text{PPS}_{chosen}}^{free} = \{\mathbf{p} = (x, y) \mid r_1 < \|\mathbf{p}\|_\infty < r_2\} \text{ for } r_1 = 3.5361 \text{ and } r_2 = 100 \quad (4.8)$$

Equipped with the metric topology of $\mathcal{S}_{\text{PPS}_{chosen}}$, $\mathcal{S}_{\text{PPS}_{chosen}}^{free}$ is now a topological space denoted by $\mathcal{TS}_{\text{PPS}_{chosen}}^{free}$. Thus, the R-1 representation of the constraint-free region obtained in (4.8) may be transformed to a CFM of choice. Utilizing the same set of homeomorphisms, $h_{R-3}^{R-4} \circ h_{R-2}^{R-3} \circ h_{R-1}^{R-2} = h_{R-1}^{R-4}$, R-1 is transformed to an annulus denoted by R-4. The radii used in the homeomorphisms are $r_i = r_1 \times \|(1, 1)\|_2 = 1.414r_1 = 5 \text{ m}$, and $r_o = 100 \text{ m}$. The resulting CFM is:

$$\text{R-4} = \{\mathbf{p} = (x, y) \mid r_i < \|\mathbf{p}\|_2 < r_o\} \quad r_i = 5 \text{ m} \quad r_o = 100 \text{ m} \quad (4.9)$$

The ICE and OCE of this R-4 space are shown in Figure 4.11.

The CFM is discretized to form the CFDM. The environment is assumed to have a centralized sensing system with multiple sensors such as cameras for basic monitoring. Information such as the presence of DCSs is processed and then relayed to the algorithm for

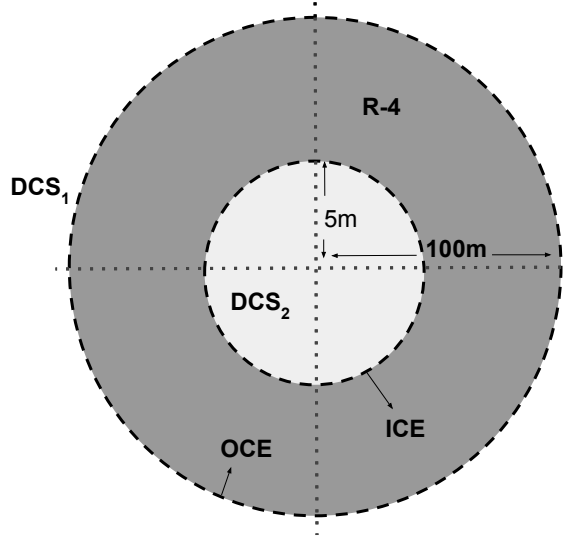


Figure 4.11: CFM of the environment denoted by R-4

path planning purposes. Recall that Section 3.2.1.6 described the potential DCs in two groups. The first group consisting of locations of constraints in the post-CFDM stage, is described as poles as seen in (3.37). In this example, assume that point representations of such constraints are received as newly updated environment information from the central sensor system; non-point representations of constraints are dealt with in Section 4.3. In this example, there exists a total of 31 such poles. Each pole is represented by the constraint's coordinates (x, y) expressed in $\mathbb{C}$, i.e. $x + iy$. Now, one chooses the holomorphic function for use. Since any polynomial is holomorphic in the entire complex plane [156], the holomorphic function chosen for simulation is that of a polynomial. As seen in (3.38), define $f_1(z) = (z(z - z_m))^a$ and thus $f(z) = f_1(z)/\text{poles}$. Here a was chosen to be a function of the number of poles in the system, in this case, half the number of poles in the system. In this case, z_m is chosen as the location on the environment that is farthest away from the ICE. This specific holomorphic function and values of its parameters are adopted from the [18]; naturally, any holomorphic function may be used.

Now, the CFDM of R-4 is complete and the performances of the PRM, RRT and CFDMPP are compared. The metrics used to evaluate the performance of the CFDMPP against that of the PRM and the RRT are defined as follows.

- Path length ratio: is the ratio of the observed path length to that of the ideal path length between the provided start and goal locations. Naturally, a value of 1 indicates that the shortest path length possible is observed, and higher values indicate an increase

in observed path lengths.

$$\text{Path length ratio} = \frac{\text{Observed path length}}{\text{Ideal path length}}$$

- Path length deviation (%): is the normalized deviation of the observed path length with respect to the ideal path length. Smaller deviation values are desired; however, their interpretation is directly related to the number and location of constraints affecting the path.

$$\text{Path length deviation (\%)} = \frac{\text{Observed path length} - \text{Ideal path length}}{\text{Ideal path length}} \times 100$$

- Rejected samples (%): is the ratio of samples discarded due to constraint compliance with respect to the total samples generated. Total samples is the sum of rejected and usable samples.

$$\text{Rejected samples (\%)} = \frac{\text{Rejected samples}}{\text{Total samples}} \times 100$$

- Usable samples (%): is the ratio of samples retained due to their constraint-free nature with respect to the total samples generated.

$$\text{Usable samples (\%)} = \frac{\text{Total samples} - \text{Rejected Samples}}{\text{Total samples}} \times 100$$

- Run time (s): The time taken by the algorithm to perform path planning in an environment provided apriori.

The simulation parameters for each of the algorithms that are compared are provided as follows.

CFDMPP is implemented using Matlab R2017b and uses DistMesh (Matlab) [125,126] to produce **CFDM** from **CFM**. The parameters chosen for simulating the **CFDMPP** across all cases in this Section 4.2 on the said **CFDM** is seen in Table 4.1. Note that parameters such as mesh generation times and inter-node computation times are a one time cost associated with a specific discretized map. The same discretized map is used for all cases in this section. Therefore, the aforementioned times are treated as overheads. The inter-node computation times involve the determination of neighbour nodes, distance between neighbouring nodes and the class marker value of line segments between neighbouring nodes. Parameters ϵ_1 and ϵ_2 maybe referenced from Figure 3.26.

The **PRM** uses an open source Matlab implementation whose github repository is located <https://github.com/petercorke/robotics-toolbox-matlab>. The maximum number of samples permitted to construct the tree is 100 (default). This implementation of the **PRM** uses an occupancy grid to represent constraints as well as the entire map. Due to this indexed representation, while the dimension of the grid is the same as what is used in **CFDMPP**,

Table 4.1: CFDMPP Parameters

Mesh generation time	Mean = 1.0740 s, Std = 0.0182 s
Inter-node computation time	Mean = 0.1640 s, Std = 0.0061 s
Mesh resolution in nodes	1451
ϵ_1	1.8
ϵ_2	1.8

the annulus is centered at a non-zero coordinate. The [RRT](#) is implemented using Matlab R2017b and uses point based representation for the map and constraints. No maximum limit on the number of samples is imposed to denote the completion of the tree. The difference in the parameters between the [PRM](#) and [RRT](#) allow for a display of the best and worst case scenarios of the [SAs](#), when compared with the [CFDMPP](#).

The [PRM](#), [RRT](#) and [CFDMPP](#) are run on the formulated R-4 for four different cases, similar to Section 4.1. Each of the four cases is further tested for a varying number of poles, in increments of three, i.e, [0,1,4,..31]. This is done to show scalability of the algorithms as the number of constraints increases. Data related to metrics are obtained by simulating the experiment 20 times. For each case, the results shown in this section include:

1. The paths produced by [PRM](#), [RRT](#) and [CFDMPP](#) in R-4 and R-1 for 0, 13 and 31 poles.
2. The paths produced by [CFDMPP](#) when the original path is homotopically blocked; displayed in R-4 and R-1 for 13 and 31 poles.
3. Table with information about a comparison metric for [PRM](#), [RRT](#) and [CFDMPP](#) (without and with blocking); shown for all five metrics for [0,1,4,..31] poles.
4. Plots displaying trends of a metric for [PRM](#), [RRT](#) and [CFDMPP](#) (without and with blocking); shown for three metrics for [0,1,4,..31] poles.

The above is subset of the results produced, and is chosen to be represented here as indicators of some trends. The remaining figures are not included here for brevity. Instead, they are shown in Appendix A, Appendix B, Appendix C and Appendix D for cases 1, 2, 3 and 4 respectively. In figures displaying the paths, the start and goal locations are shown with blue and red squares, respectively; the path is shown in green. The poles are shown with crosses. Note that sometimes, the lines representing the poles and paths may appear to overlap, mistakenly appearing like a collision. The size of the markers representing the paths and constraints are chosen for visibility, and therefore do not convey any other meaning. The paths represented are confirmed to be constraint free.

4.2.2 Results and analysis

4.2.2.1 Case 1

This case assumes that $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ are on the OCE. Figures 4.12 to 4.14 show paths produced by PRM, RRT and CFDMPP in R-4 and R-1 for 0, 13 and 31 poles, respectively. Case 1 forms the baseline of comparisons, because the nature of $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ induces an ideal path with no deviations or detours in the PPS, free of corridors or narrow passages. It also deals with the situation where the start and goal locations are on the OCE, with the expected path travelling along the ICE.

SAs are known to perform well in such non-restricted environments, and thus, this case possibly represents the best case scenario for SAs. Keeping this in mind, one can visually observe the results seen in Figure 4.12 to confirm that despite there being no poles/constraints, PRM and RRT take more detours than CFDMPP. Naturally, the sampling nature of the former two algorithms do not afford the same path quality as the latter does. However, it is important to note that both the PRM and RRT are sampling from the same CFM to for fair comparison with the CFDMPP. This is important because, all three algorithms make use of the same, known, path planning landscape. But the difference in performance lies in the how much and how, the information of the known PPS is being processed and retained for future use.

In Figure 4.12, this difference in performance may be seen as CFDMPP producing the most direct path with the least detours. Expressing this performance in terms of the metrics defined, one can choose to view it in light of either percentage of path length deviation and ratio of observed to ideal path lengths. They can be seen in Table 4.2 and Table 4.3, respectively. The percentage of path length deviation across the algorithms is better seen in Figure 4.17. The tables and figures contain a fourth entry corresponding to the performance of the CFDMPP when used under blocking circumstances; this will be discussed towards the end of this section.

One notices that CFDMPP produces a negative percentage of path length deviation in Table 4.2 (equivalently in Figure 4.17) and correspondingly, an observed ratio of less than 1 for some entries in Table 4.3. This is so, due to the choice of meshing parameters. A uniform mesh of the CFM was desired, and that choice of uniformity produced such a mesh that the ICE was coarsely approximated. That is, while the radius of the ICE is the desired radius at the vertices of the line segments approximating it, the circumference is lesser than expected. This in turn affects the length of the final path, which appears smaller in length compared to that of the ideal path. Naturally, one may simply change the meshing resolution; however, this example aids in showing the effect and flexibility of meshing the CFM.

Now consider the situations where the number of poles has increased to 13. The paths produced can be seen in Figure 4.13. The increased number of detours is immediately obvious in the case of the PRM and RRT. One may again view this in the form of percentage path length deviation and ratio of observed to ideal path lengths, but this example has a different impact on the algorithms because of the poles. SAs sample and then discard samples based on constraint violation. Therefore with the presence of constraints, one not only notices the

path detours but also the percentage of samples that are rejected, and in turn the samples that are usable.

Tables 4.4 and 4.5 show the percentage of usable and rejected samples, respectively. From Section 3.2.2, recall that the **CFDMPP** stores path classes on a given mesh, by keeping track of nodes encountered on different path classes. So, the total possible number of nodes or samples produced in the **CFDMPP** is the total number of path classes multiplied by the number of nodes in the mesh. The rejected and usable sample percentages are calculated based on this value. Table 4.5 shows that the **PRM** performs better than the **RRT**, with the **CFDMPP** producing the least percentage of rejected samples. This can be seen in Figure 4.18.

In the simulation example described, this case is indicative of needing to traverse the length of the warehouse while avoiding the first group of **DCs**. The significance of detours that take place in paths produced by the **SAs** are clear in the figures, seen in the original space, i.e. R-1. Clearly, this affects the prime metric of path length, but more importantly, it shows the disadvantage of using **SAs** for spaces whose knowledge cannot be adequately used to optimize path planning via **SAs**. The same behaviour can be observed in Figure 4.14 with respect to the **SAs**' inability to fully represent the known aspects of a known **PPS**. Finally, the run times may be seen in Table 4.6 or equivalently in Figure 4.19, where the **CFDMPP** performs well ahead of the **SAs**.

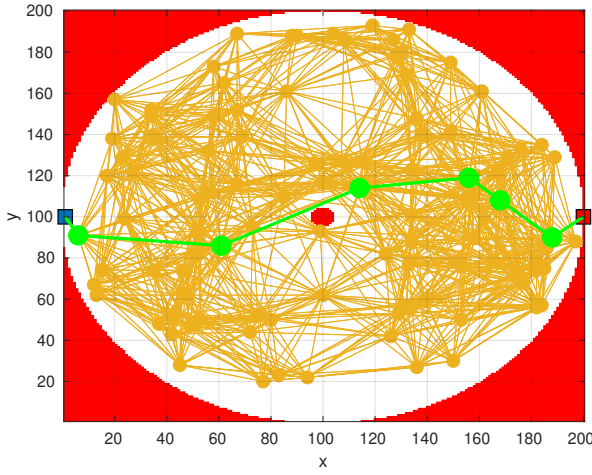
The comparison metrics seen via tables and figures so far, show the trends for a varying number of poles in the environment. When these trends are viewed in light of how the same **CFM** is being used, we recall that the method of processing known information impacts the chosen metrics. That is, in the case of **SAs**, even though the same **CFM** is being processed every time, new samples are being generated and discarded. Thus, the path generated on one run/map/tree has no future use, because that map/tree holds no fixed association with the **CFM** the next time the algorithm is called. So, one cannot rely on a previously generated path, to make an informed decision about the next desired path. Therefore, this shows that the value of a path planning algorithm is not just limited to path length metrics, memory and run-time, but also re-usability. In the context of re-usability, the focus is not only restricted to the map, but also to the representation of paths that have been generated thus far.

With **CFDMPP**, not only is the map stored, but is processed to keep track of path classes. In fact, one of the prime advantages of using **CFDMPP** is that of classes to solve the problem of the second group of constraints. Consider Figures 4.15 and 4.16, where one can see the initial path produced, and the next alternative path produced as well. This situation can occur when the original path has been rendered invalid or is chosen to be invalid or blocked. In such a case, the generated map, classes and path information are being used to avoid a specific path class, and generate an alternative path. Furthermore, the kind of alternative path generated may be chosen by tuning ϵ_3 referred to as seen in Figures 4.15 and 4.16. That is, by maximizing the knowledge of the map, we are now able to tune our required path, and deliberately avoid path. If one tried to simulate this behaviour using the **SAs**, the constraints will have to be introduced into the environment in place of the path that is not desired. This is so because the map/tree changes for each run, and there is a loss of information that would have otherwise proved to be useful for the future. Therefore, the

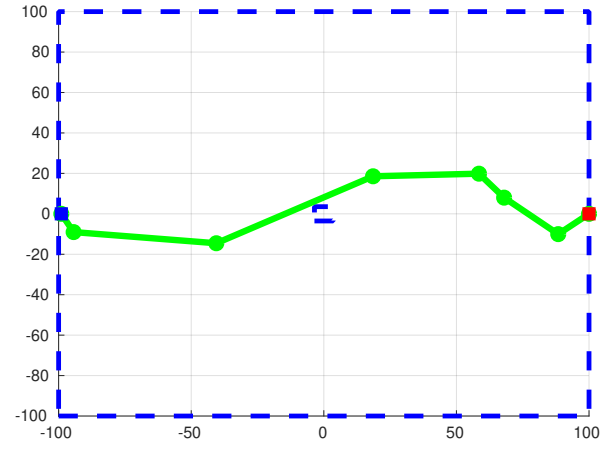
CFDMPP now maximizes the knowledge of the map and offers a tunable degree of freedom allowing the deliberate avoidance of a path class, and the generation of an alternative one.

The same metrics are used to evaluate the performance of the **CFDMPP** in blocking mode. In this mode, the tuning parameter ϵ_3 forces the algorithm to produce a path in a different path class. Whether the alternative path class is the next best (most optimal) path class in comparison to the original, blocked class is dependent on the user-defined ϵ_3 . This of course impacts the detours taken by the path, and thus the percent path length deviation. So in comparison to **CFDMPP**, Table 4.2 (or equivalently in Figure 4.17) and Table 4.3 show that blocking induces a greater deviation. But of course, this time, the deviation has been triggered, in order to avoid the previous path. Despite this deliberate triggering of re-planning an alternative path, the **CFDMPP** in blocking mode still achieves a significantly better performance with respect to percentage of rejected samples (Figure 4.18 or Table 4.5) and run time (Figure 4.19 or Table 4.6). The reader will notice some entries with a “NaN” value in the table or missing bars in the metric comparison plots. This implies that **CFDMPP** could not execute blocking because there is only one path class in the **CFM**. This may occur due to the location of poles (close together on the **ICE** or **OCE**) or due to stringent tuning parameters, both of which disallow more than one path class. In the case of our simulations, such entries are due to the location of poles.

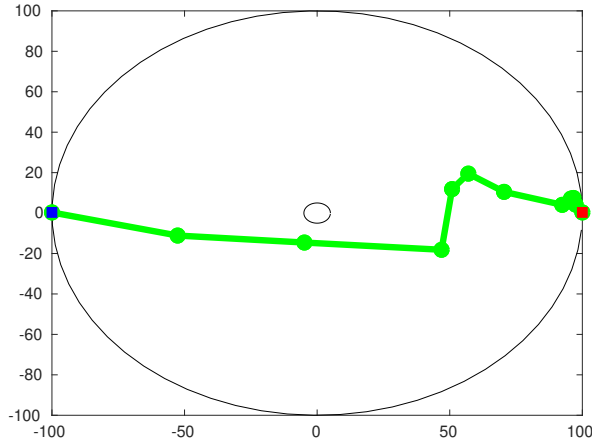
To better see the impact of this degree of freedom, refer to Figure 4.20. Each figure represents a path that is produced when the original path seen in Figure 4.14(e) is blocked using a different ϵ_3 value. ϵ_3 indicates the threshold of similarity between the class marker values of the path classes, such that a path class satisfying that that threshold is considered to be the same class as the blocked path class seen in Figure 4.20(a). So as ϵ_3 increases across Figure 4.20(b), Figure 4.20(c), Figure 4.20(d) and Figure 4.20(e), the alternative paths look increasingly less similar to the blocked path. This is well in alignment with the expected behaviour from definitions in Chapter 3, as class marker values of path classes are increasingly different due to the poles the contour encloses. It is this freedom, that we exert as a choice for tuning.



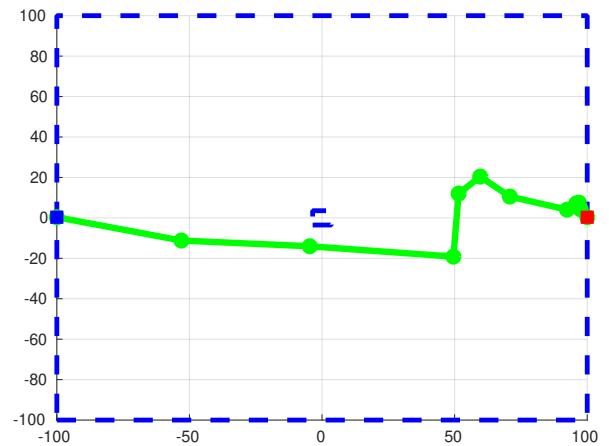
(a) PRM: Path in R-4



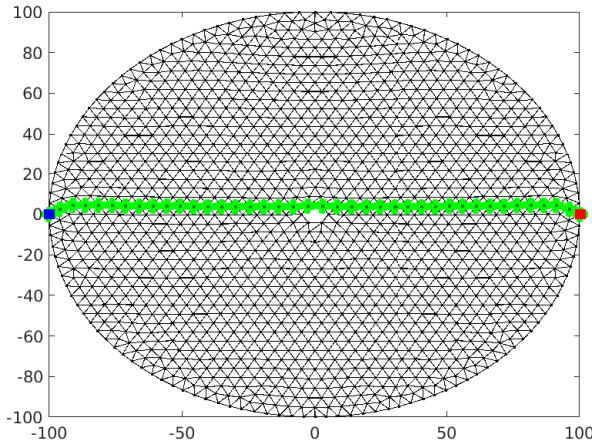
(b) PRM: Path in R-1



(c) RRT: Path in R-4



(d) RRT: Path in R-1

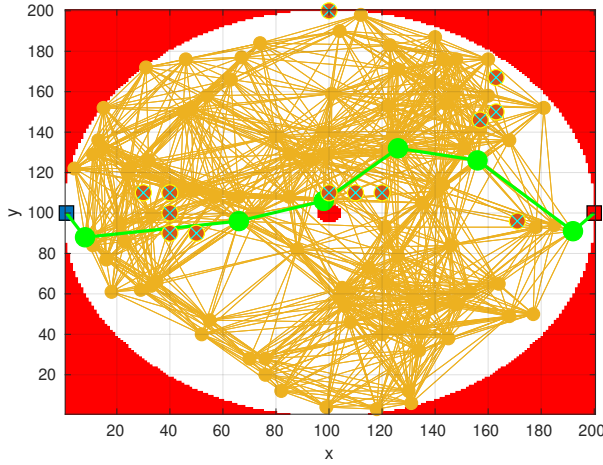


(e) CFDMPP: Path in R-4

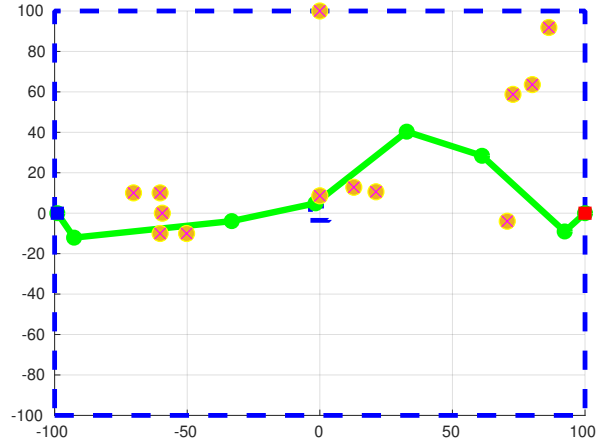


(f) CFDMPP: Path in R-1

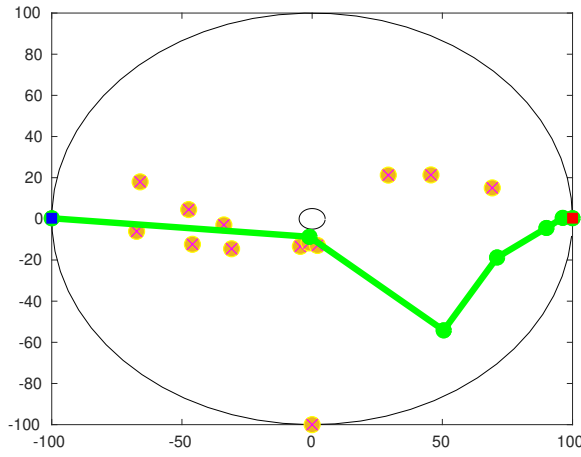
Figure 4.12: Case 1: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.



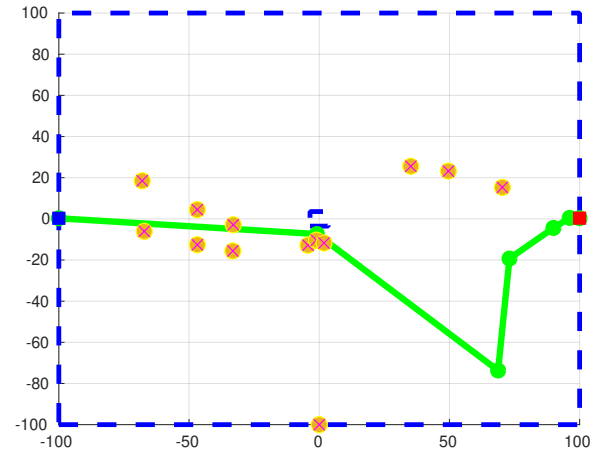
(a) PRM: Path in R-4



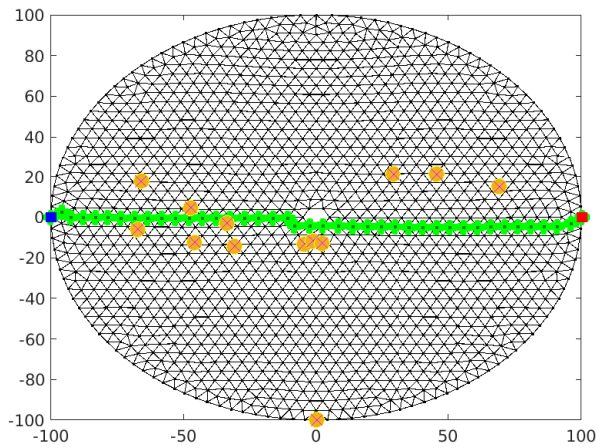
(b) PRM: Path in R-1



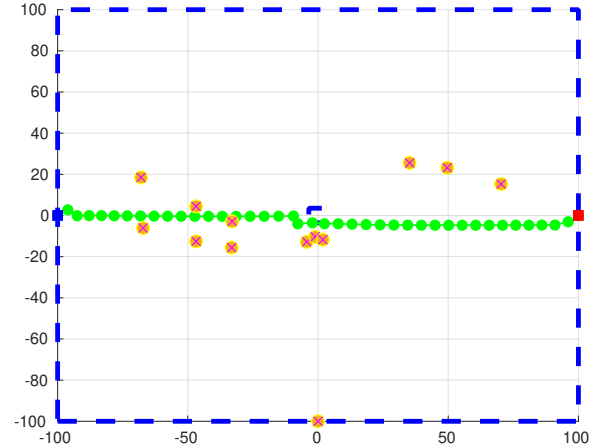
(c) RRT: Path in R-4



(d) RRT: Path in R-1

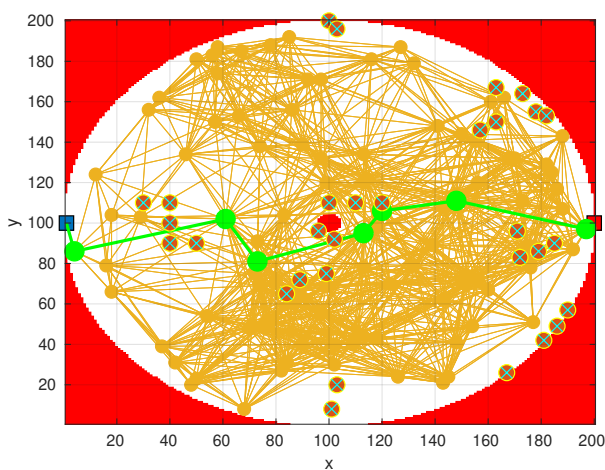


(e) CFDMPP: Path in R-4

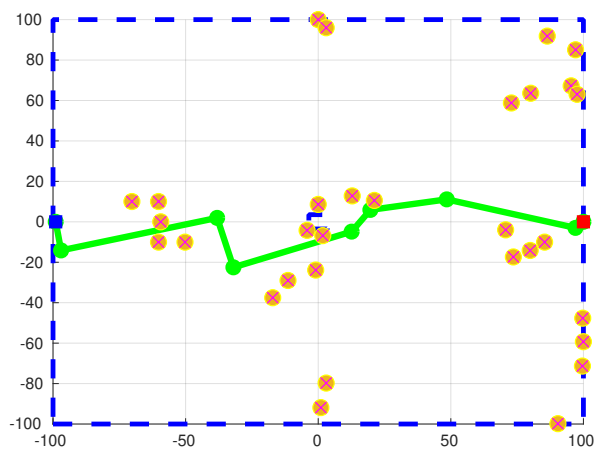


(f) CFDMPP: Path in R-1

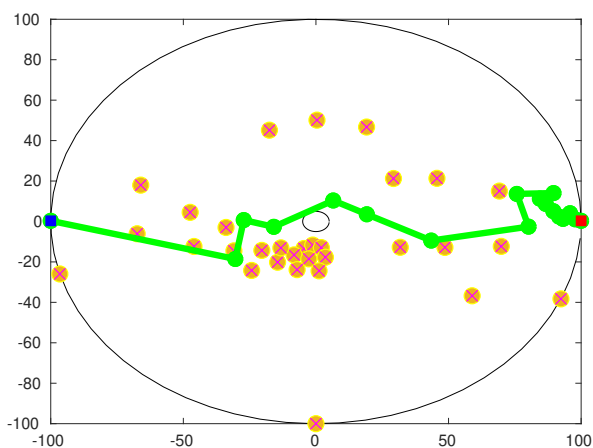
Figure 4.13: Case 1: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.



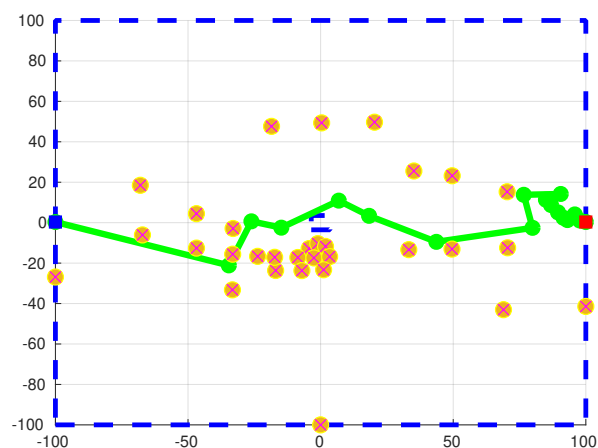
(a) PRM: Path in R-4



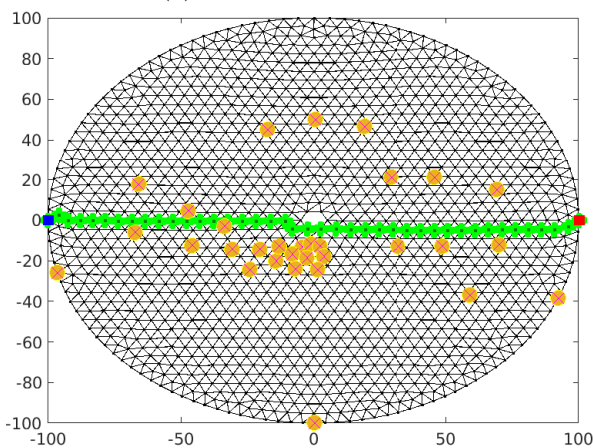
(b) PRM: Path in R-1



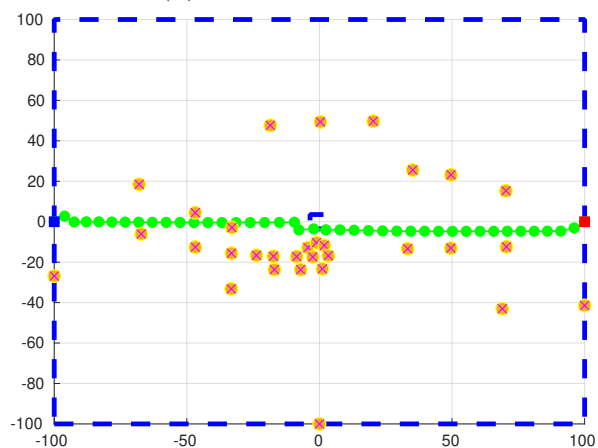
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.14: Case 1: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.

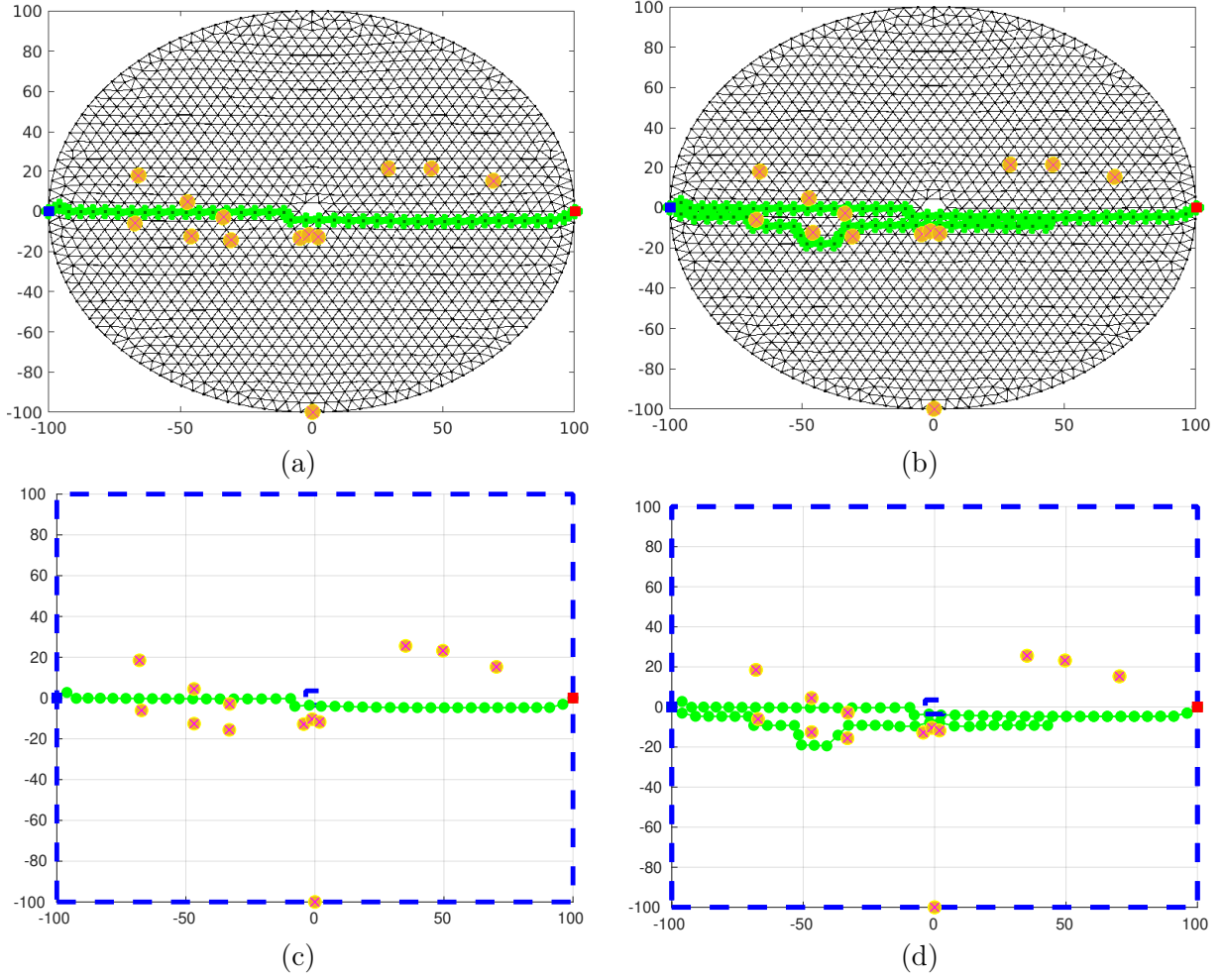


Figure 4.15: Case 1: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs HB Threshold = 0.55)). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

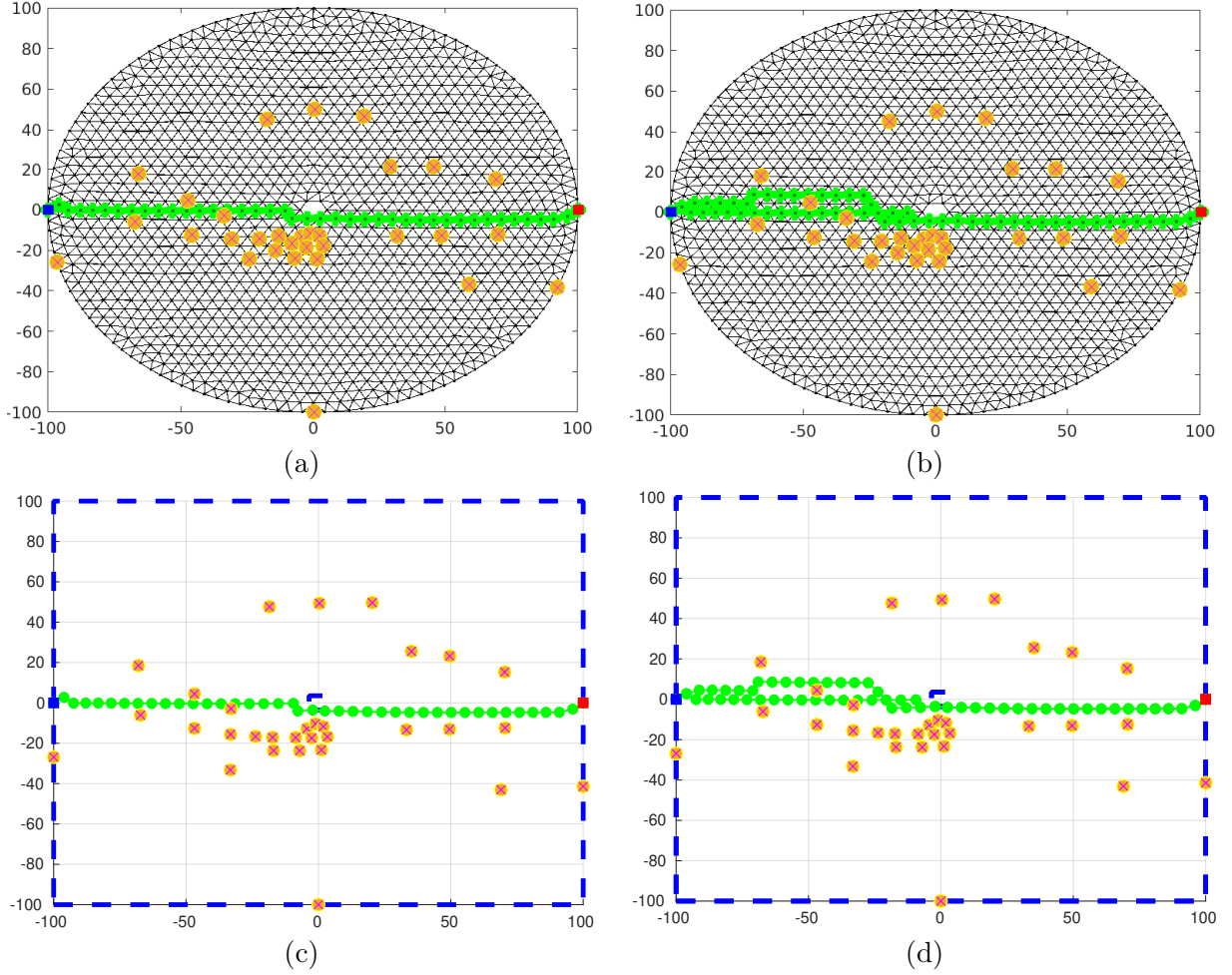


Figure 4.16: Case 1: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 0.45$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

Table 4.2: Case 1 : Percentage of Path Length Deviation

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	24.08, 13.65	52.61, 47.01	-1.60, 0.00	NaN, NaN
1	18.37, 15.14	53.08, 31.59	-1.60, 0.00	NaN, NaN
4	19.46, 10.96	58.31, 46.91	-1.60, 0.00	5.52, 0.00
7	27.94, 19.12	58.73, 34.97	-1.60, 0.00	0.67, 0.00
10	23.76, 17.68	52.47, 29.55	-0.20, 0.00	0.67, 0.00
13	15.76, 12.54	51.64, 34.33	-0.20, 0.00	5.90, 0.00
16	16.68, 13.37	54.69, 39.91	-0.20, 0.00	0.68, 0.00
19	19.01, 13.56	50.97, 28.39	-0.20, 0.00	0.95, 0.00
22	18.35, 12.18	57.77, 49.98	-0.20, 0.00	0.67, 0.00
25	18.56, 10.89	48.20, 38.06	-0.20, 0.00	2.35, 0.00
28	16.19, 10.35	27.90, 25.51	-0.20, 0.00	0.40, 0.00
31	17.75, 11.43	37.16, 23.62	-0.20, 0.00	3.07, 0.00

Table 4.3: Case 1 Ratio of observed path length to ideal length

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	1.28, 0.14	1.57, 0.48	0.98, 0.00	NaN, NaN
1	1.22, 0.16	1.57, 0.32	0.98, 0.00	NaN, NaN
4	1.23, 0.11	1.63, 0.48	0.98, 0.00	1.06, 0.00
7	1.32, 0.20	1.63, 0.36	0.98, 0.00	1.01, 0.00
10	1.28, 0.18	1.57, 0.30	1.00, 0.00	1.01, 0.00
13	1.20, 0.13	1.56, 0.35	1.00, 0.00	1.06, 0.00
16	1.20, 0.14	1.59, 0.41	1.00, 0.00	1.01, 0.00
19	1.23, 0.14	1.55, 0.29	1.00, 0.00	1.01, 0.00
22	1.22, 0.13	1.62, 0.51	1.00, 0.00	1.01, 0.00
25	1.22, 0.11	1.52, 0.39	1.00, 0.00	1.02, 0.00
28	1.20, 0.11	1.31, 0.26	1.00, 0.00	1.00, 0.00
31	1.22, 0.12	1.41, 0.24	1.00, 0.00	1.03, 0.00

Table 4.4: Case 1 Percentage of usable samples

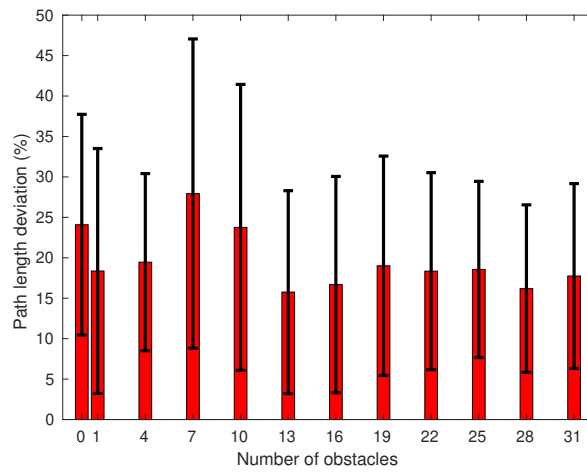
# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	78.08, 3.65	46.44, 1.40	100.00, 0.00	NaN, NaN
1	78.94, 3.38	46.52, 0.43	100.00, 0.00	NaN, NaN
4	79.63, 3.10	46.46, 0.56	99.93, 0.00	99.98, 0.00
7	78.47, 3.68	46.50, 0.90	99.95, 0.00	99.94, 0.00
10	78.49, 3.88	46.20, 0.92	99.75, 0.00	99.86, 0.00
13	77.46, 3.83	46.32, 0.43	99.80, 0.00	99.91, 0.00
16	79.15, 4.24	46.45, 0.30	99.85, 0.00	99.89, 0.00
19	79.40, 2.90	46.23, 0.61	99.84, 0.00	99.93, 0.00
22	77.37, 3.54	46.32, 0.74	99.81, 0.00	99.85, 0.00
25	78.71, 2.86	46.33, 0.77	99.78, 0.00	99.95, 0.00
28	78.29, 4.03	46.40, 1.19	99.66, 0.00	99.95, 0.00
31	78.35, 4.36	46.45, 0.55	99.80, 0.00	99.92, 0.00

Table 4.5: Case 1 Percentage Rejected Samples

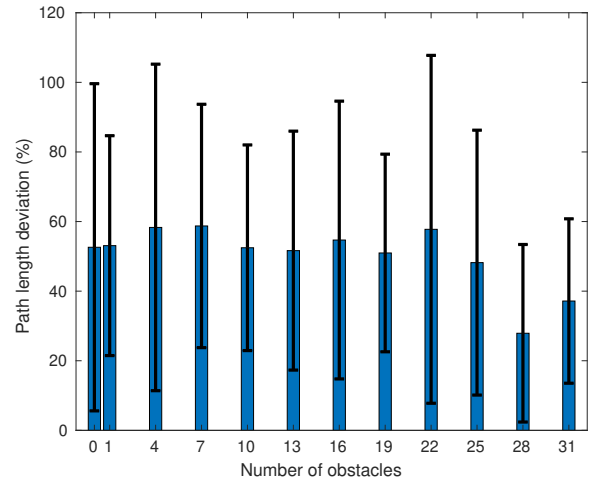
# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	21.92, 3.65	53.56, 1.40	0.00, 0.00	NaN, NaN
1	21.06, 3.38	53.48, 0.43	0.00, 0.00	NaN, NaN
4	20.37, 3.10	53.54, 0.56	0.07, 0.00	0.02, 0.00
7	21.53, 3.68	53.50, 0.90	0.05, 0.00	0.06, 0.00
10	21.51, 3.88	53.80, 0.92	0.25, 0.00	0.14, 0.00
13	22.54, 3.83	53.68, 0.43	0.20, 0.00	0.09, 0.00
16	20.85, 4.24	53.55, 0.30	0.15, 0.00	0.11, 0.00
19	20.60, 2.90	53.77, 0.61	0.16, 0.00	0.07, 0.00
22	22.63, 3.54	53.68, 0.74	0.19, 0.00	0.15, 0.00
25	21.29, 2.86	53.67, 0.77	0.22, 0.00	0.05, 0.00
28	21.71, 4.03	53.60, 1.19	0.34, 0.00	0.05, 0.00
31	21.65, 4.36	53.55, 0.55	0.20, 0.00	0.08, 0.00

Table 4.6: Case 1 Run time

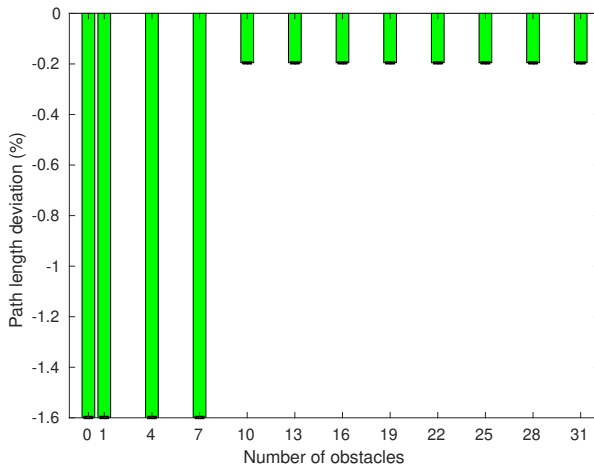
# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	2.20, 0.12	3.84, 8.84	0.22, 0.01	NaN, NaN
1	2.21, 0.10	3.59, 8.60	0.23, 0.01	NaN, NaN
4	2.22, 0.08	3.48, 5.08	0.19, 0.01	0.29, 0.02
7	2.23, 0.09	3.19, 7.06	0.20, 0.01	0.22, 0.01
10	2.18, 0.10	3.64, 5.66	0.28, 0.01	0.26, 0.01
13	2.19, 0.09	5.91, 13.56	0.29, 0.02	0.35, 0.01
16	2.18, 0.08	6.16, 19.90	0.29, 0.02	0.26, 0.02
19	2.19, 0.06	1.02, 1.23	0.29, 0.01	0.23, 0.02
22	2.21, 0.06	1.55, 2.11	0.30, 0.02	0.27, 0.01
25	2.17, 0.08	4.59, 9.19	0.30, 0.01	0.31, 0.02
28	2.19, 0.06	2.74, 4.64	0.30, 0.01	0.24, 0.01
31	2.18, 0.08	1.41, 1.84	0.30, 0.01	0.28, 0.02



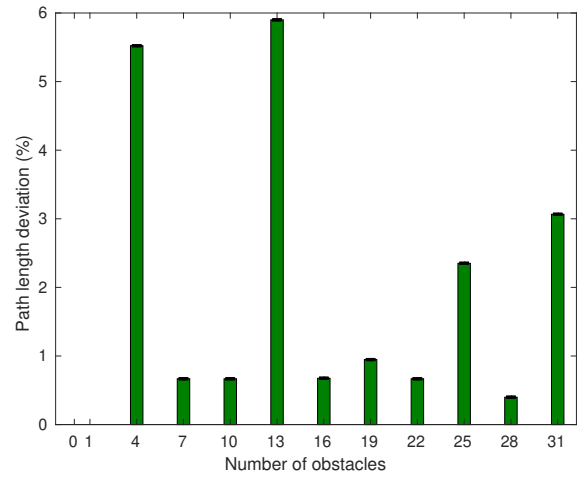
(a) PRM



(b) RRT

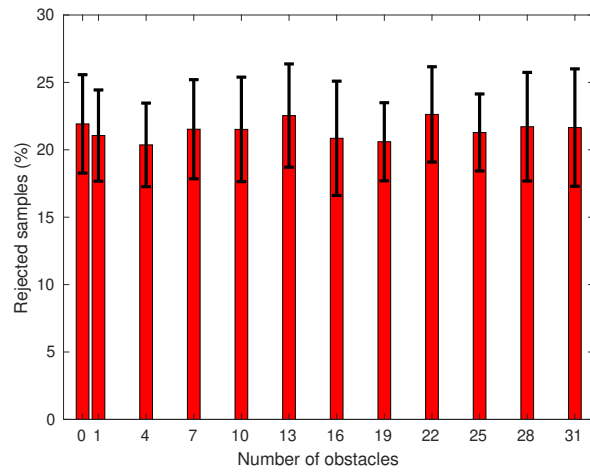


(c) CFDMPP

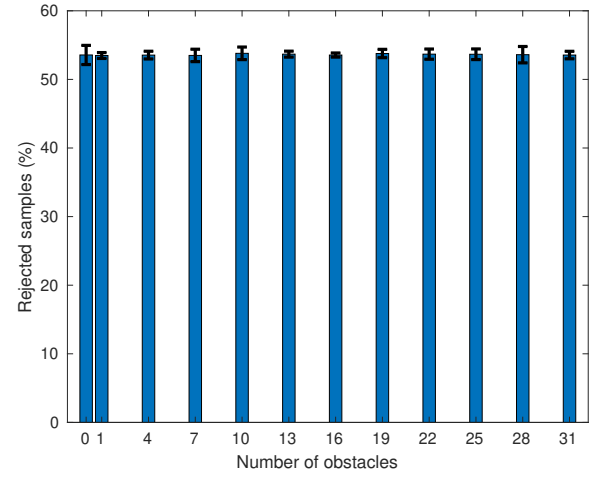


(d) CFDMPP with blocking

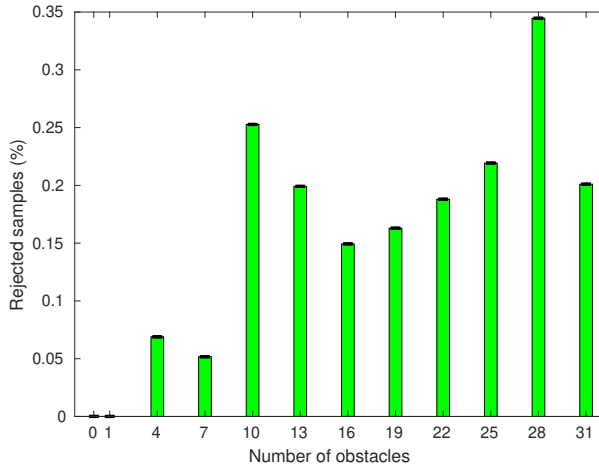
Figure 4.17: Case 1: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



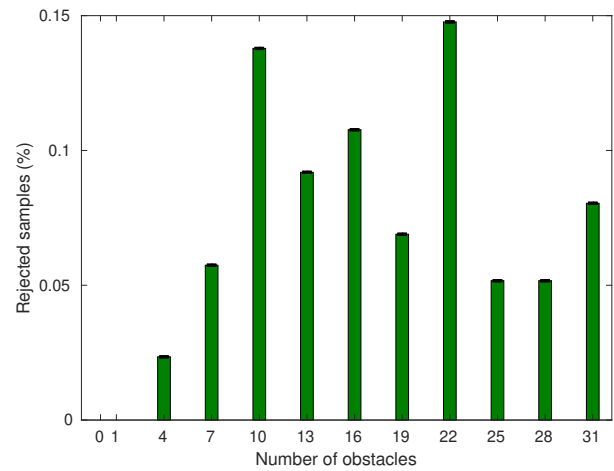
(a) PRM



(b) RRT

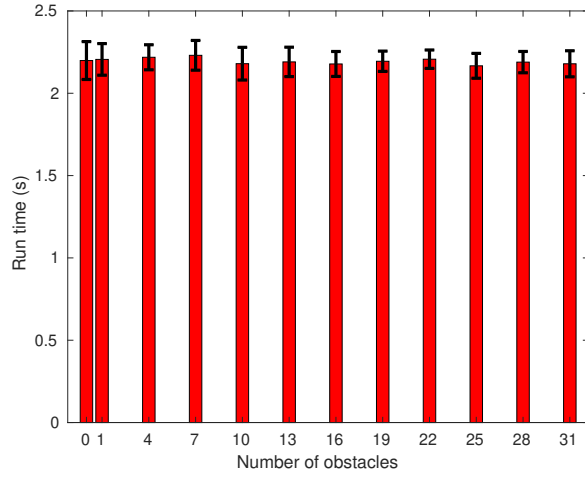


(c) CFDMPP

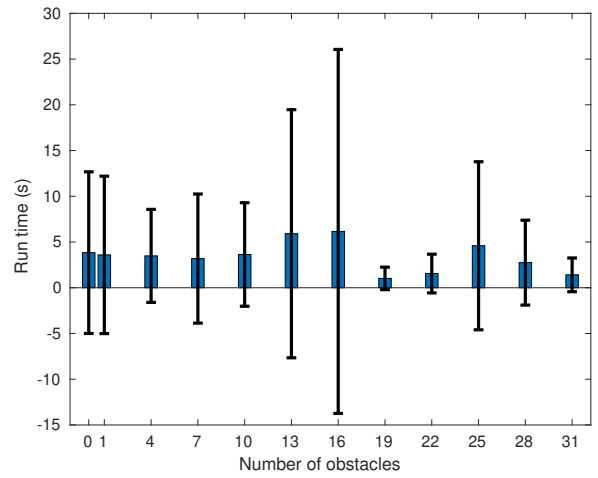


(d) CFDMPP with blocking

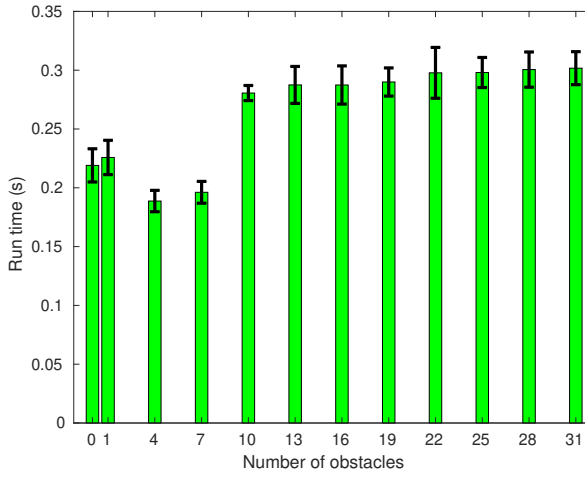
Figure 4.18: Case 1: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



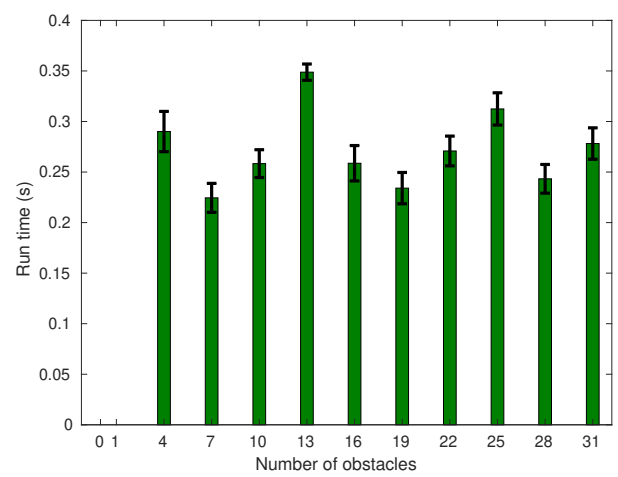
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

Figure 4.19: Case 1: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.

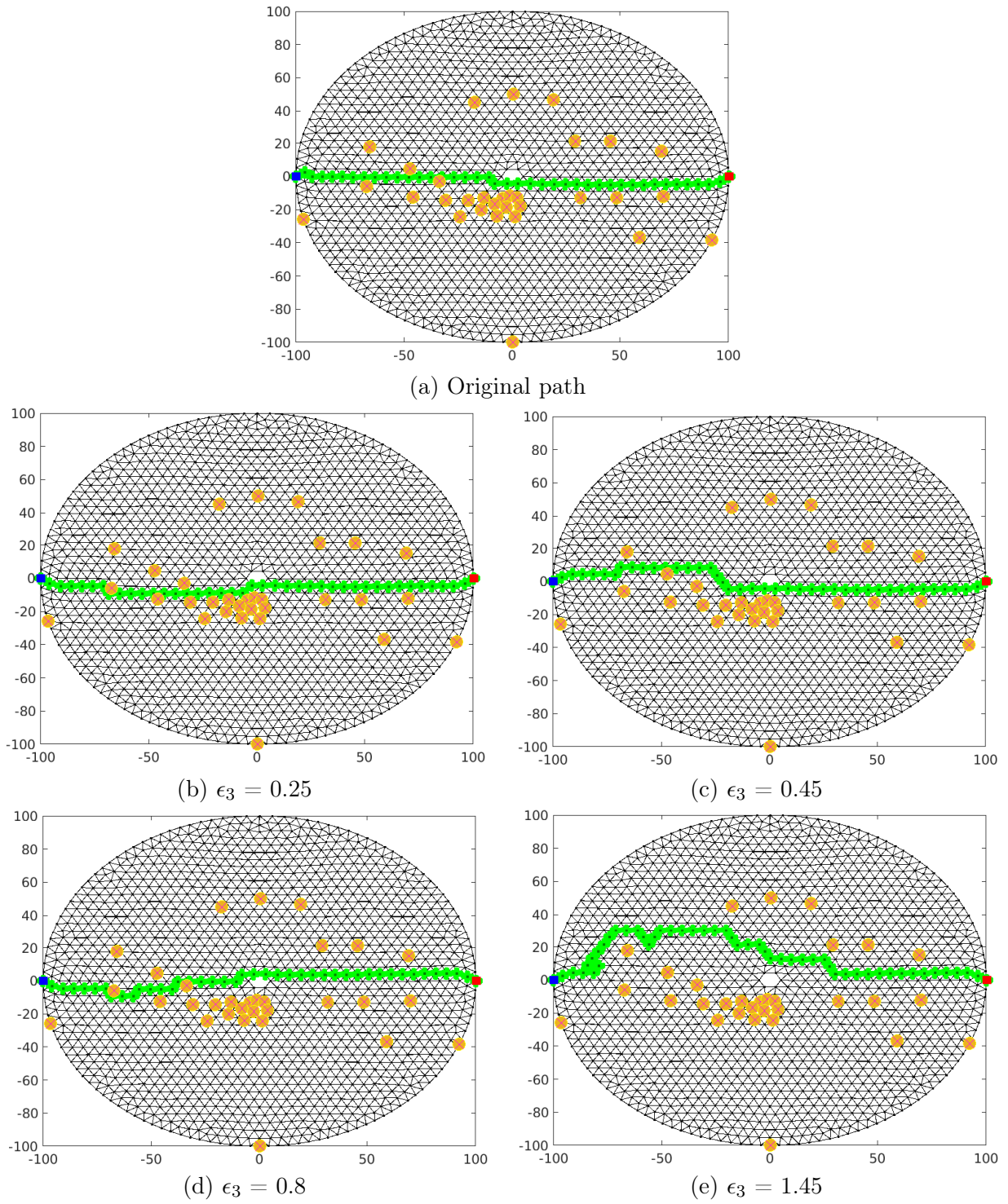


Figure 4.20: Alternative paths produced by CFDMPP on the CFDM

4.2.2.2 Case 2

Similar to case 2 in Section 4.1, this case assumes that $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ are on the OCE and ICE respectively. The aim of this case is to observe the behaviour of all three algorithms as $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ become closer in proximity. The same mesh and tuning parameters seen in case 1 is used here. The performance metrics are seen in Table 4.7 or equivalently in Figure 4.26, Table 4.8, Table 4.9, Table 4.10 or equivalently in Figure 4.27 and Table 4.11 or equivalently in Figure 4.28. The paths produced by the PRM, RRT and CFDMPP in R-4 and R-1 are seen in Figure 4.21. Since case 1 provided a detailed correlation between the results of performance metrics and the algorithms' behaviour, it is assumed that the reader can make similar inferences in the upcoming cases. From here onward, we discuss observations that are unique to each case.

It is clear that the CFDMPP yields the shortest path with the least detours, as compared to the SAs. This example is important because it starts to show a pattern of the SAs as the separation between the start and goal locations decreases. One can see from Figure 4.21 that even without poles, the SAs experience a greater amount of detours. This behaviour was also previously seen in case 1 where the start-goal pair had a larger separation. However, it becomes more prominent here as the separation from the start-goal pair decreases to approximately half of what it was in case 1. This is in alignment with the fundamental principle behind SAs, since it is more likely that longer paths are better represented, due to the likelihood of obtaining samples over a wide-spread area over time. However, when the paths become more confined, the same does not hold anymore. In the context of realistic path planning however, the separation between the start-goal pair may not be high, and algorithms are still expected to yield a path with minimum detours as possible. The CFDMPP is able to produce a path with no effect of separation between the start and goal locations.

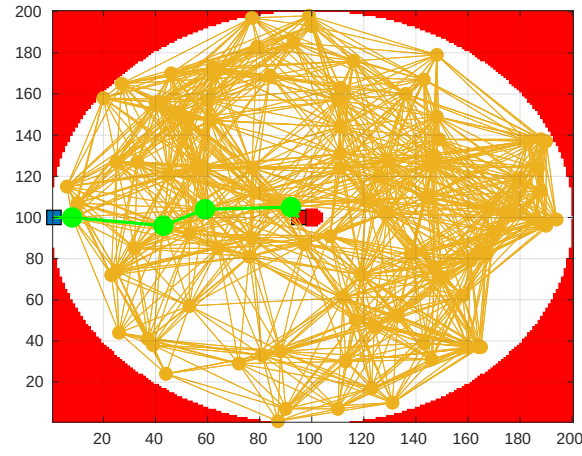
In fact, the same pattern may be observed in situations where the environment has 13 and 31 poles, whose paths are seen in Figure 4.22 and Figure 4.23 respectively. From the figures, one may note that the placement of poles appear different across the algorithms, while the number of poles is consistent. This difference is important to demonstrate the following. First, the deliberate placement of poles helps demonstrate certain features of blocking in the CFDMPP in light of tuning. Second, this helps demonstrate the difference between SAs and the CFDMPP. The former does not retain information about the map for future runs as opposed to the latter. Thus, for the purposes of comparisons, so long as the number of poles, the separation between start and goal locations are held constant, the SAs may still be evaluated against the CFDMPP using the same metrics.

We approach the topic of blocked paths and alternative paths by considering the paths produced in the presence of 13 poles seen in Figure 4.24, 31 poles seen in Figure 4.25 and exploring tuning options in Figure 4.29 in conjunction with each other. Both Figure 4.24 and Figure 4.25 produce expected alternative paths, but the interesting observation lies in Figure 4.25. Figure 4.25 shows the presence of exactly one pole that distinguishes the path class of the original path seen in Figure 4.25(a) and the alternative path class seen in Figure 4.25(b). The CFDMPP successfully yields this alternative path with considerably less detours than would a SAs. This is an example of simulating encounters with poles located in close proximity to the original path and requiring the next viable, shortest path. SAs do

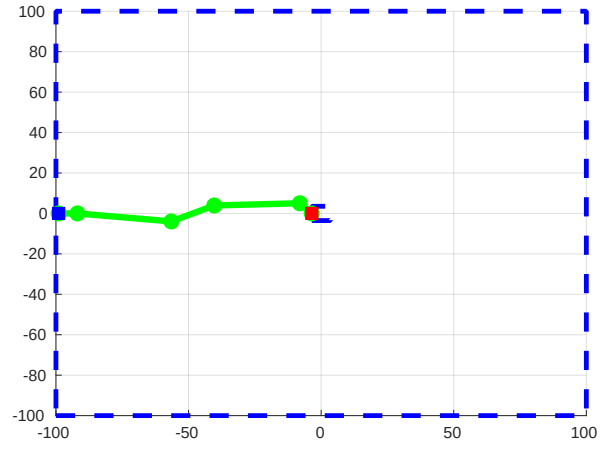
not have this capability, since they retain information about neither the map nor the paths.

We now look at the different alternative paths produced via homotopic blocking to avoid the second group of DCs. The original and resulting alternative paths are seen in Figure 4.29. This case is important in demonstrating the impacts of choosing tuning parameters and ϵ_3 , on the kinds of alternative paths produced. Notice that the paths corresponding to ϵ_3 values of 0.45 and 1 produce alternative paths as expected. Now, $\epsilon_3 = 1.15$ also results in an alternative path, but upon closer inspection, we can see that the path intuitively belongs in the same path class as that of $\epsilon_3 = 0.45$. In the case of $\epsilon_3 = 1.35$, the path produced is in a path class that is different from that of $\epsilon_3 = 0.45$ and $\epsilon_3 = 1$. But clearly, it is not optimal in path length.

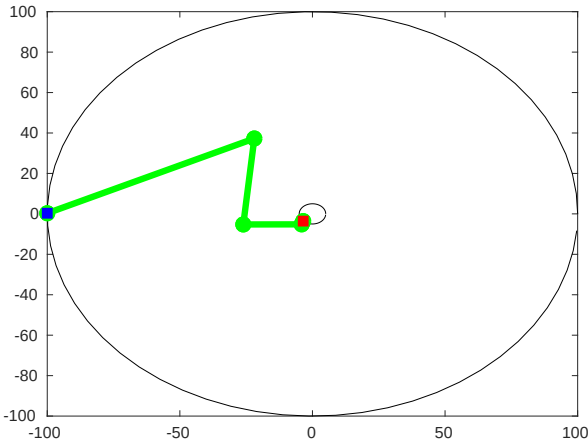
Both of the final two alternative paths are not ideal, but have been displayed here as the effects of choosing tuning parameters. Since tuning parameters ϵ_1 , ϵ_2 and ϵ_3 are inherently related, the choice of the first two eventually determines the path classes that are created. By setting ϵ_3 , we set a finite value that determines whether a contour up till a certain point should feature in one class or the next. The choice of the tuning parameters initially involves no technique. However, with observation and experience with the map, one may gradually acquire enough intuition to make more educated choices. So, in this specific case, the choice of the tuning parameters is considered to satisfy the path planning needs, since it correctly yields 3 paths—the original, and the first two alternative paths. The alternative paths produced beyond that point are not satisfactory. Naturally, the tuning parameters may be adjusted, but there will exist tradeoffs; a topic that will be discussed in detail in Section 4.3. This example is meant to serve as an introduction to the different results that we obtain as a consequence of the tuning parameters. It is important to note that despite the nuances of tuning, this case successfully yields one original path and two alternative paths, which is not something that SAs can accomplish. In fact, the ability to produce at least one alternative path will now start featuring prominently when we discuss the choice of the tuning parameters.



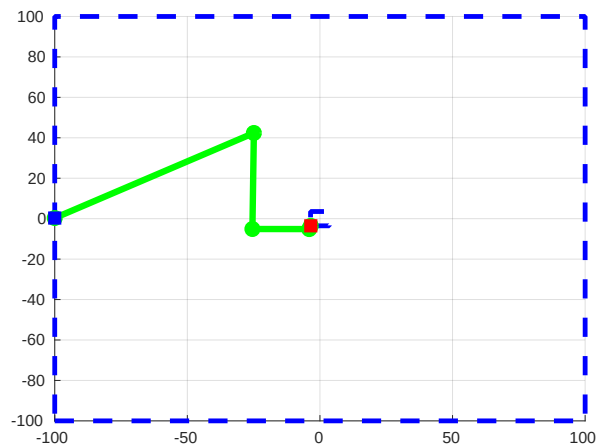
(a) PRM: Path in R-4



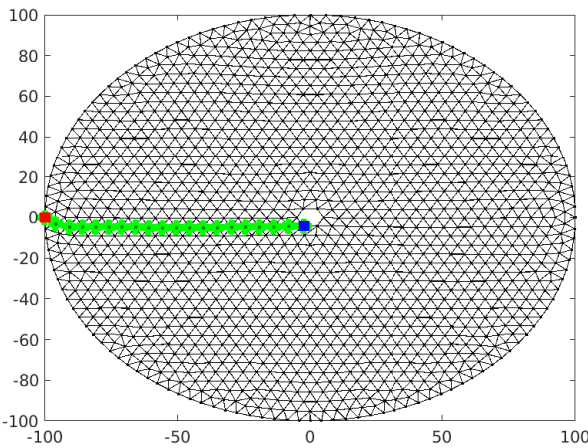
(b) PRM: Path in R-1



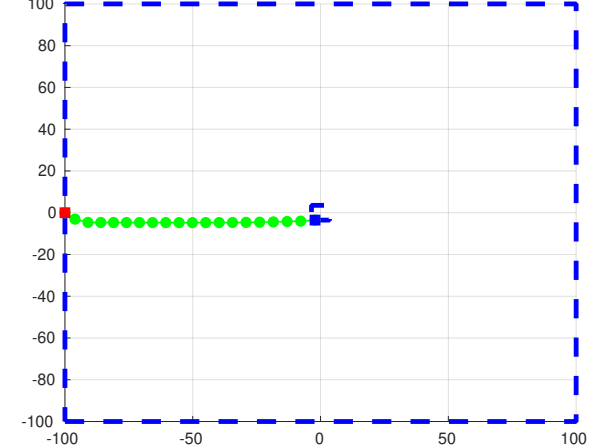
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.21: Case 2: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.

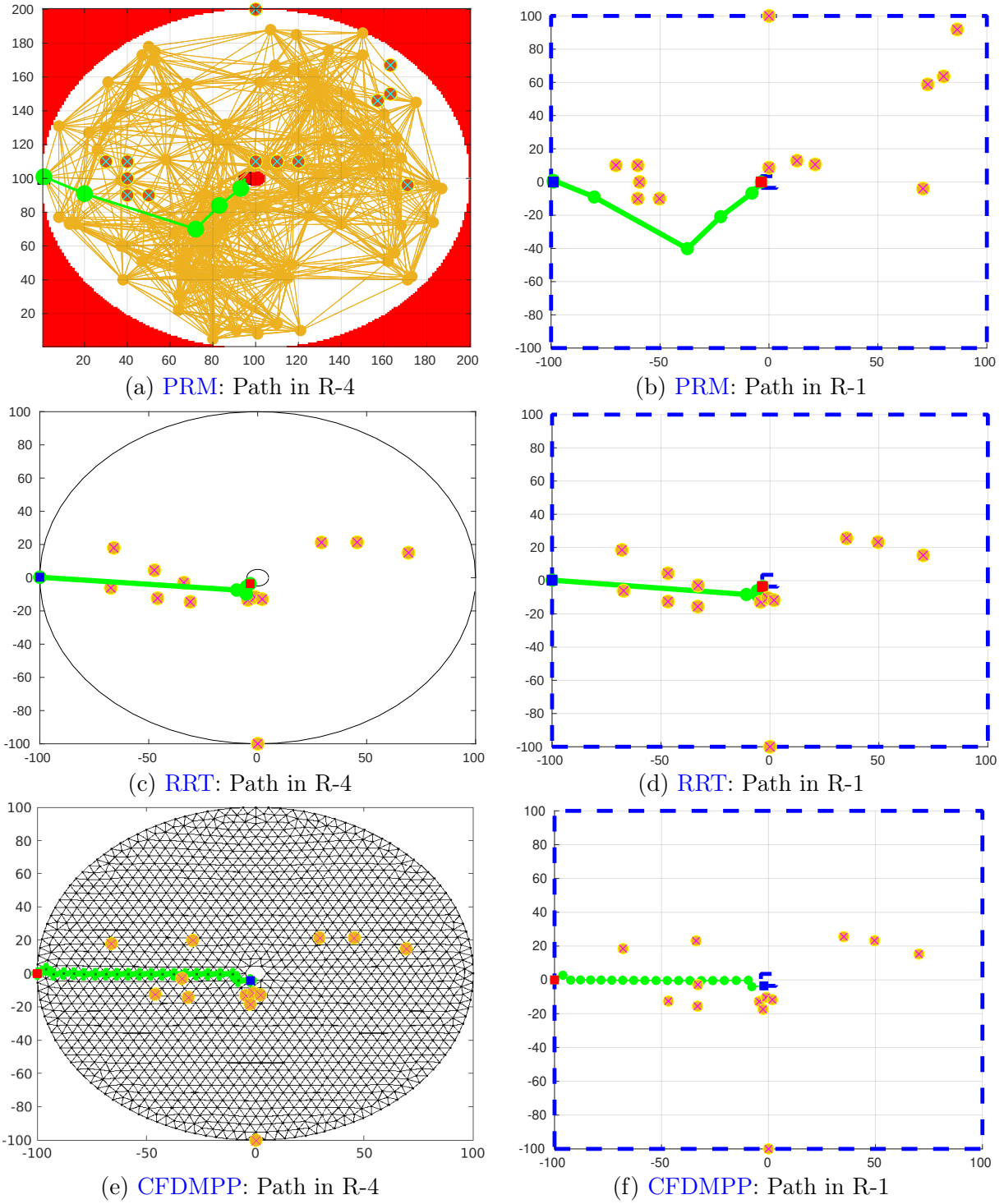
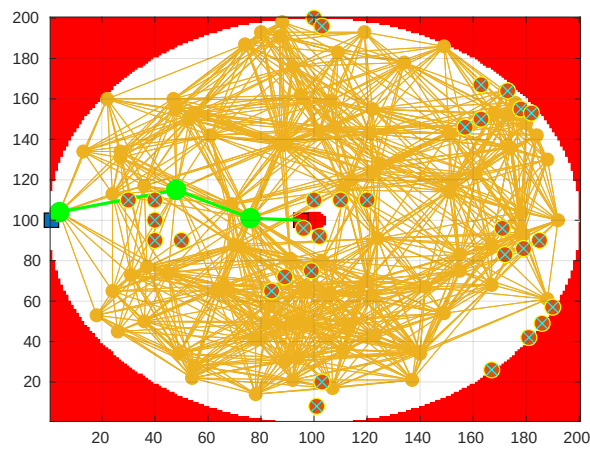
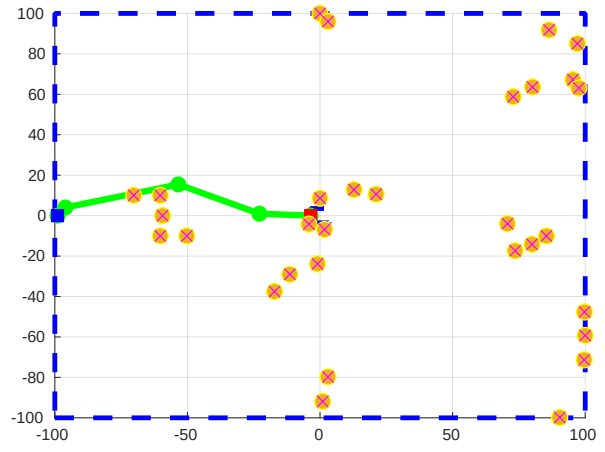


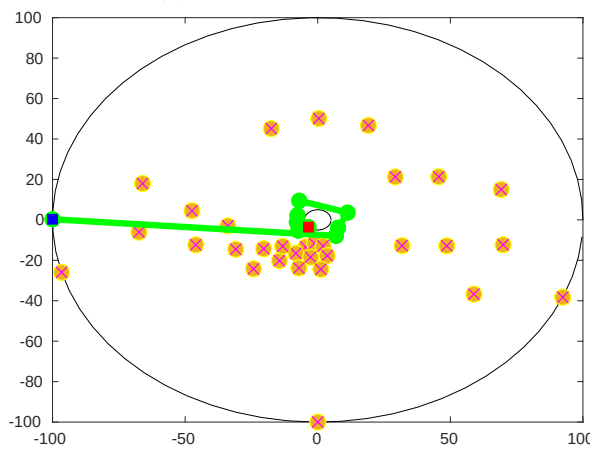
Figure 4.22: Case 2: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.



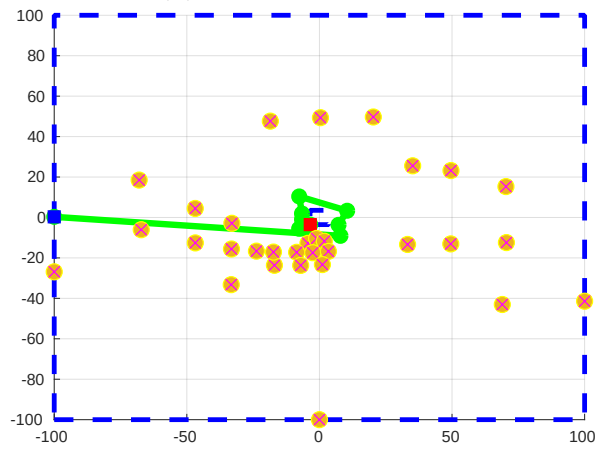
(a) PRM: Path in R-4



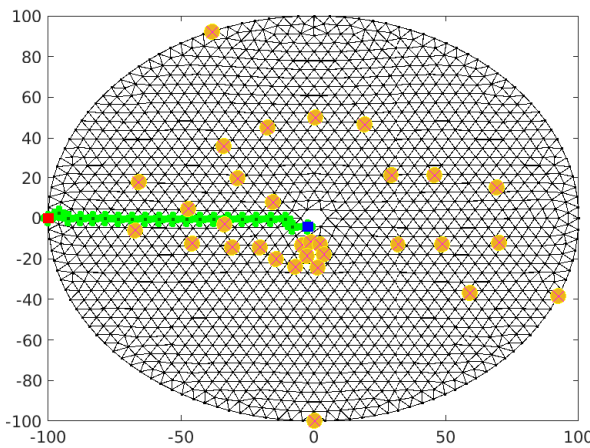
(b) PRM: Path in R-1



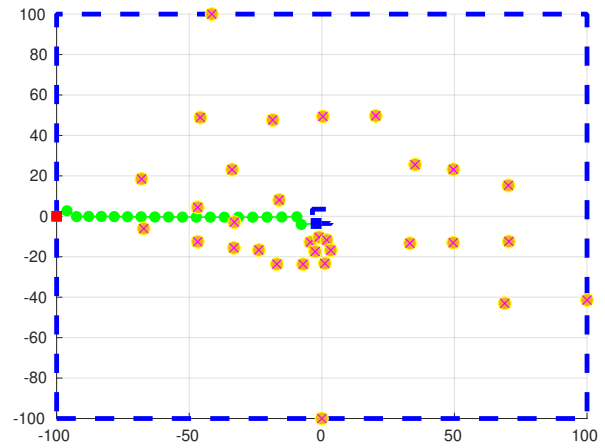
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.23: Case 2: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.

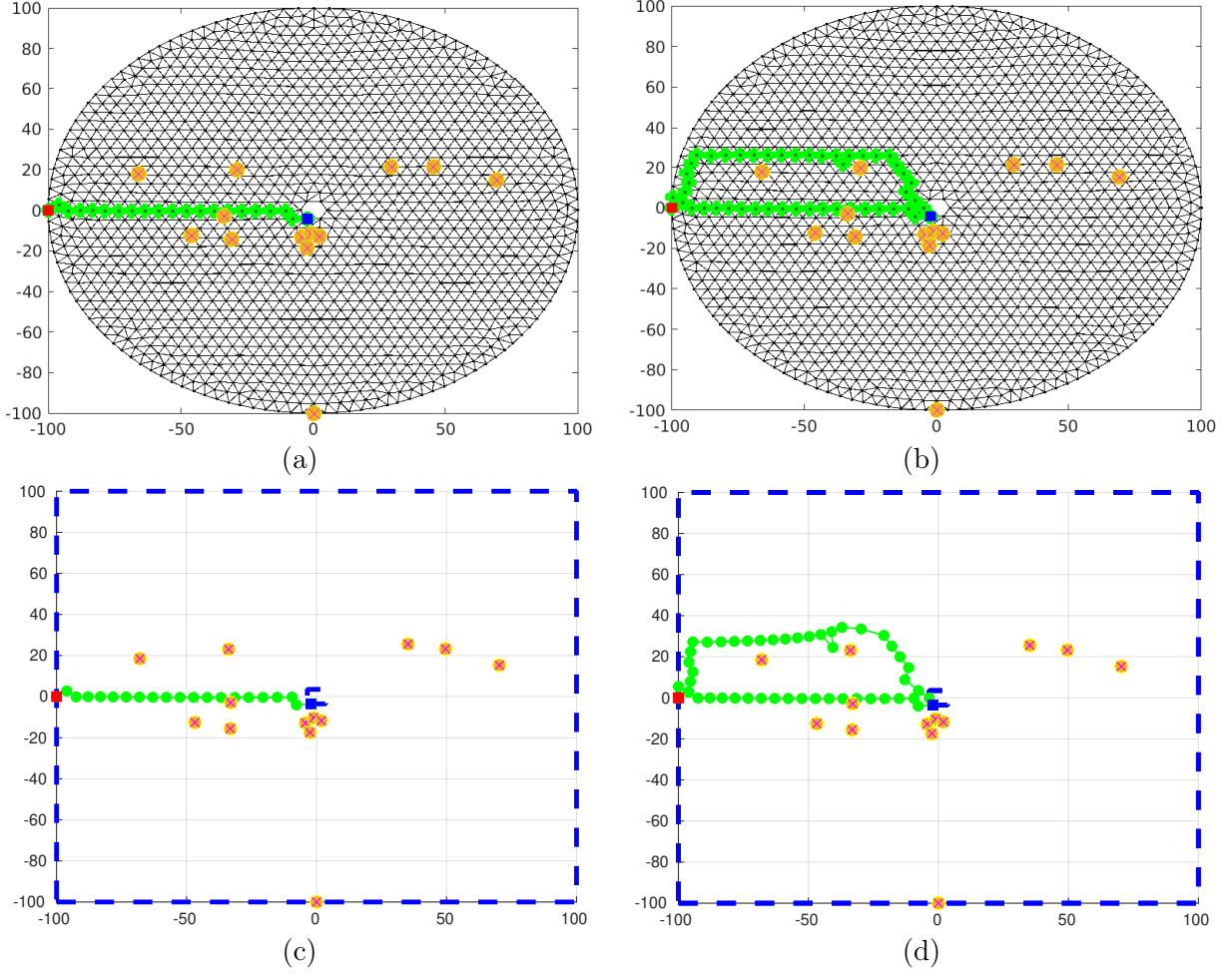


Figure 4.24: Case 2: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs $\epsilon_3 = 0.85$)). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

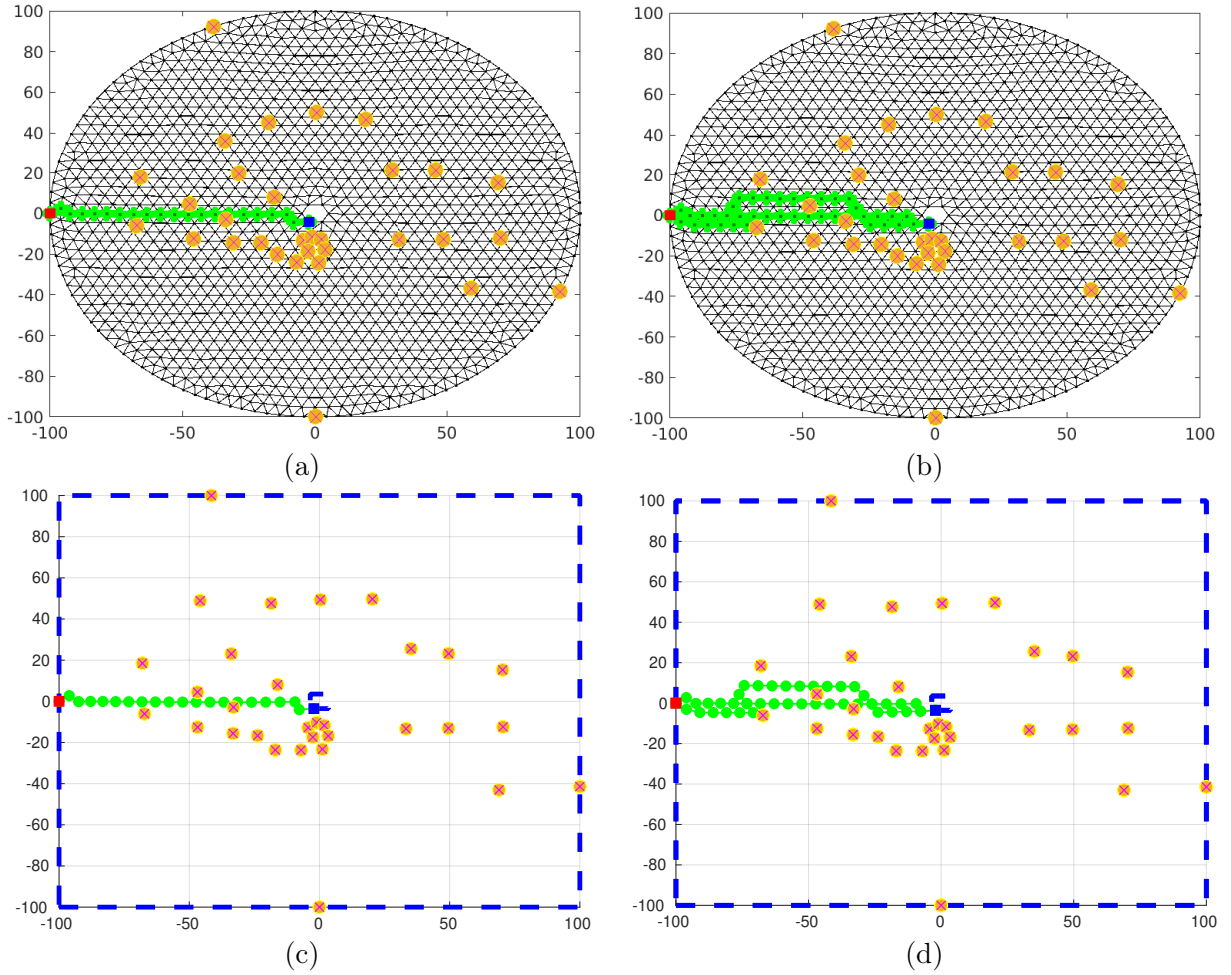


Figure 4.25: Case 1: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 1$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

Table 4.7: Case 2 : Percentage of Path Length Deviation

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	29.85, 25.69	133.18, 77.88	2.19, 0.00	NaN, NaN
1	35.00, 28.97	102.96, 86.08	2.19, 0.00	NaN, NaN
4	47.42, 41.36	148.17, 91.37	2.19, 0.00	39.73, 0.00
7	29.80, 24.37	111.34, 77.77	2.19, 0.00	29.08, 0.00
10	51.59, 26.20	117.34, 93.46	4.98, 0.00	33.22, 0.00
13	39.92, 30.12	165.72, 111.76	4.98, 0.00	53.97, 0.00
16	43.63, 34.06	119.76, 94.58	4.98, 0.00	17.40, 0.00
19	41.50, 43.98	146.71, 111.06	4.98, 0.00	28.03, 0.00
22	38.10, 33.58	82.73, 55.17	4.98, 0.00	56.83, 0.00
25	41.18, 28.56	96.21, 75.17	4.98, 0.00	32.76, 0.00
28	45.81, 34.79	118.26, 79.40	4.98, 0.00	54.78, 0.00
31	45.66, 31.45	111.57, 77.25	4.98, 0.00	28.71, 0.00

Table 4.8: Case 2 Ratio of observed path length to ideal length

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	1.30, 0.26	2.33, 0.78	1.02, 0.00	NaN, NaN
1	1.35, 0.29	2.03, 0.86	1.02, 0.00	NaN, NaN
4	1.47, 0.41	2.48, 0.91	1.02, 0.00	1.40, 0.00
7	1.30, 0.24	2.11, 0.78	1.02, 0.00	1.29, 0.00
10	1.52, 0.26	2.17, 0.93	1.05, 0.00	1.33, 0.00
13	1.40, 0.30	2.66, 1.12	1.05, 0.00	1.54, 0.00
16	1.44, 0.34	2.20, 0.95	1.05, 0.00	1.17, 0.00
19	1.41, 0.44	2.47, 1.11	1.05, 0.00	1.28, 0.00
22	1.38, 0.34	1.83, 0.55	1.05, 0.00	1.57, 0.00
25	1.41, 0.29	1.96, 0.75	1.05, 0.00	1.33, 0.00
28	1.46, 0.35	2.18, 0.79	1.05, 0.00	1.55, 0.00
31	1.46, 0.31	2.12, 0.77	1.05, 0.00	1.29, 0.00

Table 4.9: Case 1 Percentage of usable samples

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	78.72, 3.97	46.69, 0.85	100.00, 0.00	NaN, NaN
1	78.57, 4.11	46.72, 1.14	100.00, 0.00	NaN, NaN
4	76.21, 2.99	46.68, 1.65	100.00, 0.00	99.97, 0.00
7	78.68, 2.82	46.89, 1.32	100.00, 0.00	99.97, 0.00
10	78.92, 3.52	47.43, 1.42	99.77, 0.00	99.95, 0.00
13	79.40, 4.85	46.95, 1.33	99.77, 0.00	99.96, 0.00
16	79.18, 3.27	45.80, 3.35	99.79, 0.00	99.67, 0.00
19	78.12, 4.08	47.16, 1.76	99.72, 0.00	99.95, 0.00
22	77.04, 3.71	45.87, 3.04	99.79, 0.00	99.95, 0.00
25	79.50, 4.10	46.75, 2.16	99.59, 0.00	99.93, 0.00
28	78.44, 3.27	46.28, 2.29	99.79, 0.00	99.89, 0.00
31	78.86, 2.72	46.83, 2.15	99.69, 0.00	99.91, 0.00

Table 4.10: Case 1 Percentage Rejected Samples

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	21.28, 3.97	53.31, 0.85	0.00, 0.00	NaN, NaN
1	21.43, 4.11	53.28, 1.14	0.00, 0.00	NaN, NaN
4	23.79, 2.99	53.32, 1.65	0.00, 0.00	0.03, 0.00
7	21.32, 2.82	53.11, 1.32	0.00, 0.00	0.03, 0.00
10	21.08, 3.52	52.57, 1.42	0.23, 0.00	0.05, 0.00
13	20.60, 4.85	53.05, 1.33	0.23, 0.00	0.04, 0.00
16	20.82, 3.27	54.20, 3.35	0.21, 0.00	0.33, 0.00
19	21.88, 4.08	52.84, 1.76	0.28, 0.00	0.05, 0.00
22	22.96, 3.71	54.13, 3.04	0.21, 0.00	0.05, 0.00
25	20.50, 4.10	53.25, 2.16	0.41, 0.00	0.07, 0.00
28	21.56, 3.27	53.72, 2.29	0.21, 0.00	0.11, 0.00
31	21.14, 2.72	53.17, 2.15	0.31, 0.00	0.09, 0.00

Table 4.11: Case 2 : Run time

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	2.06, 0.09	6.63, 23.25	0.15, 0.01	NaN, NaN
1	2.03, 0.10	2.22, 5.56	0.15, 0.01	NaN, NaN
4	2.00, 0.08	2.67, 4.78	0.16, 0.01	0.45, 0.02
7	2.04, 0.07	3.16, 6.03	0.16, 0.01	0.32, 0.03
10	2.01, 0.08	3.73, 7.27	0.18, 0.01	0.38, 0.02
13	2.00, 0.07	5.62, 13.66	0.19, 0.01	0.37, 0.03
16	1.97, 0.06	2.31, 4.43	0.19, 0.01	0.25, 0.02
19	2.01, 0.09	5.10, 8.69	0.19, 0.01	0.25, 0.02
22	1.99, 0.07	3.04, 6.99	0.19, 0.01	0.38, 0.02
25	1.97, 0.10	2.66, 3.16	0.20, 0.02	0.31, 0.02
28	1.97, 0.10	1.30, 1.99	0.20, 0.01	0.36, 0.03
31	1.95, 0.07	2.68, 3.99	0.21, 0.01	0.30, 0.01

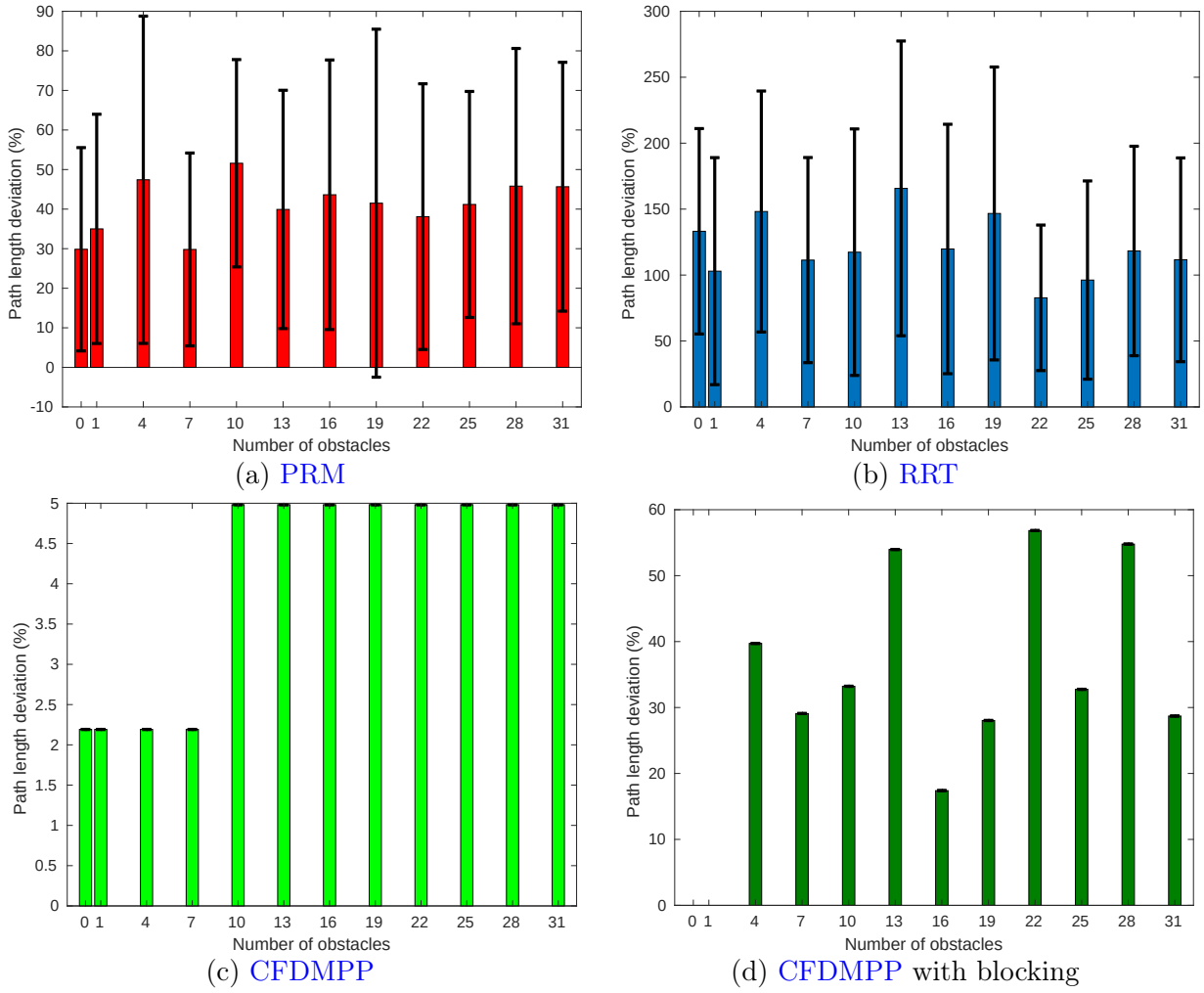
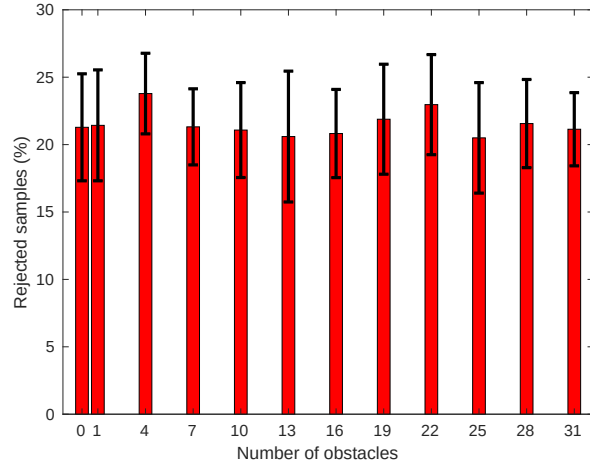
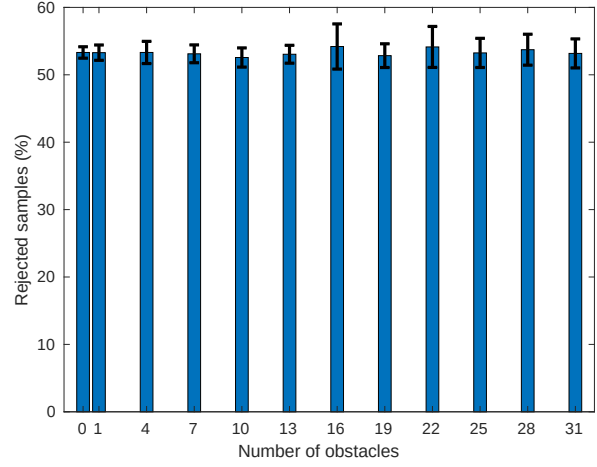


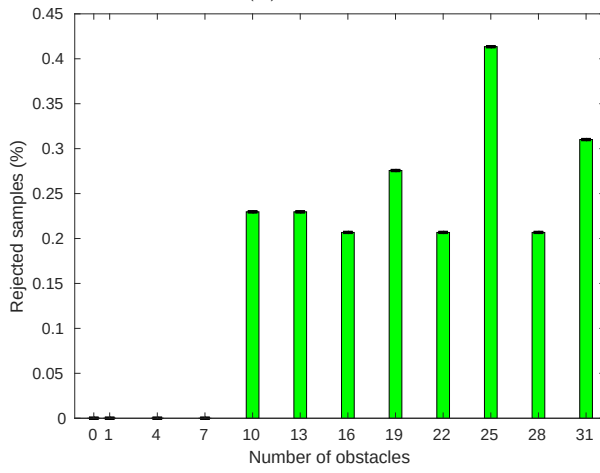
Figure 4.26: Case 2: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



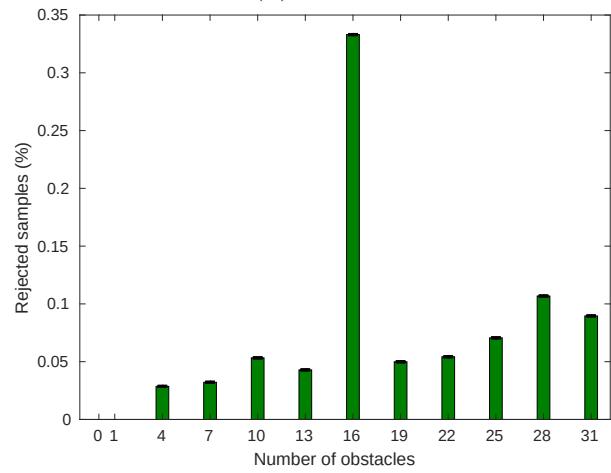
(a) PRM



(b) RRT

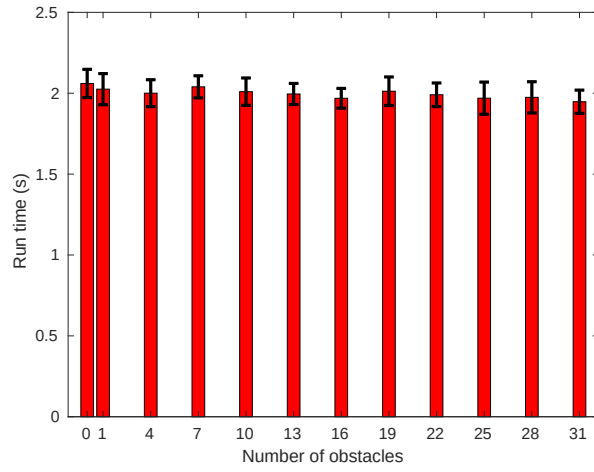


(c) CFDMPP

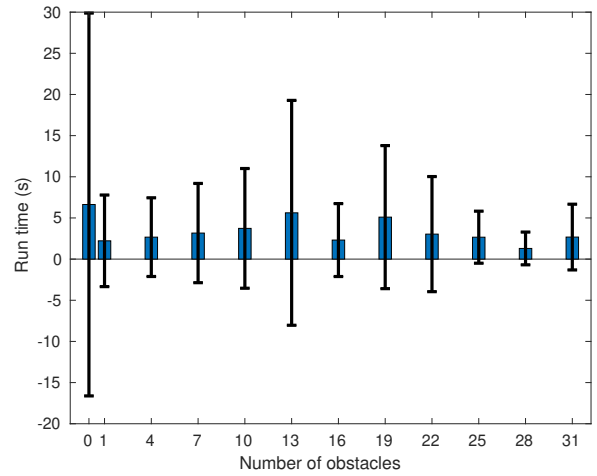


(d) CFDMPP with blocking

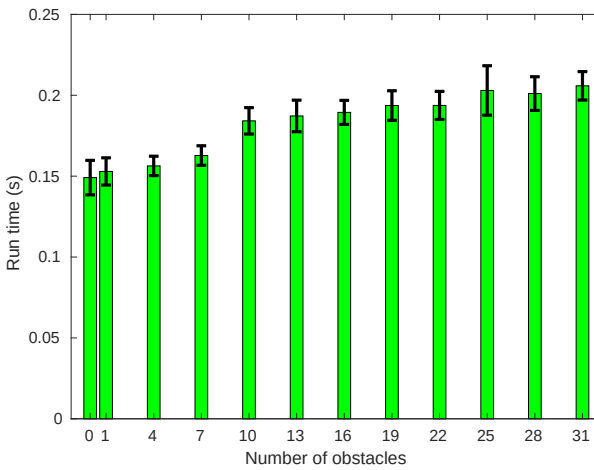
Figure 4.27: Case 2: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



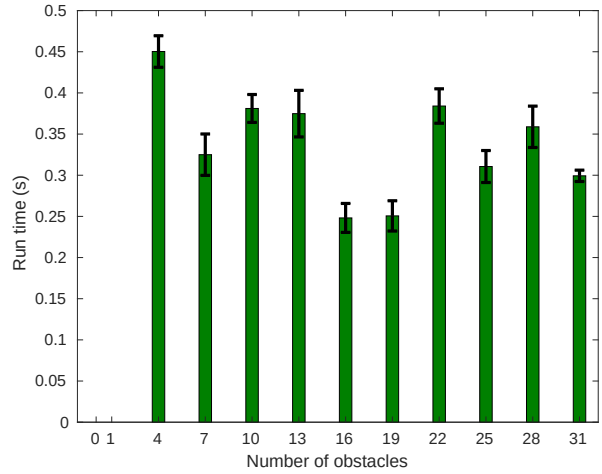
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

Figure 4.28: Case 2: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.

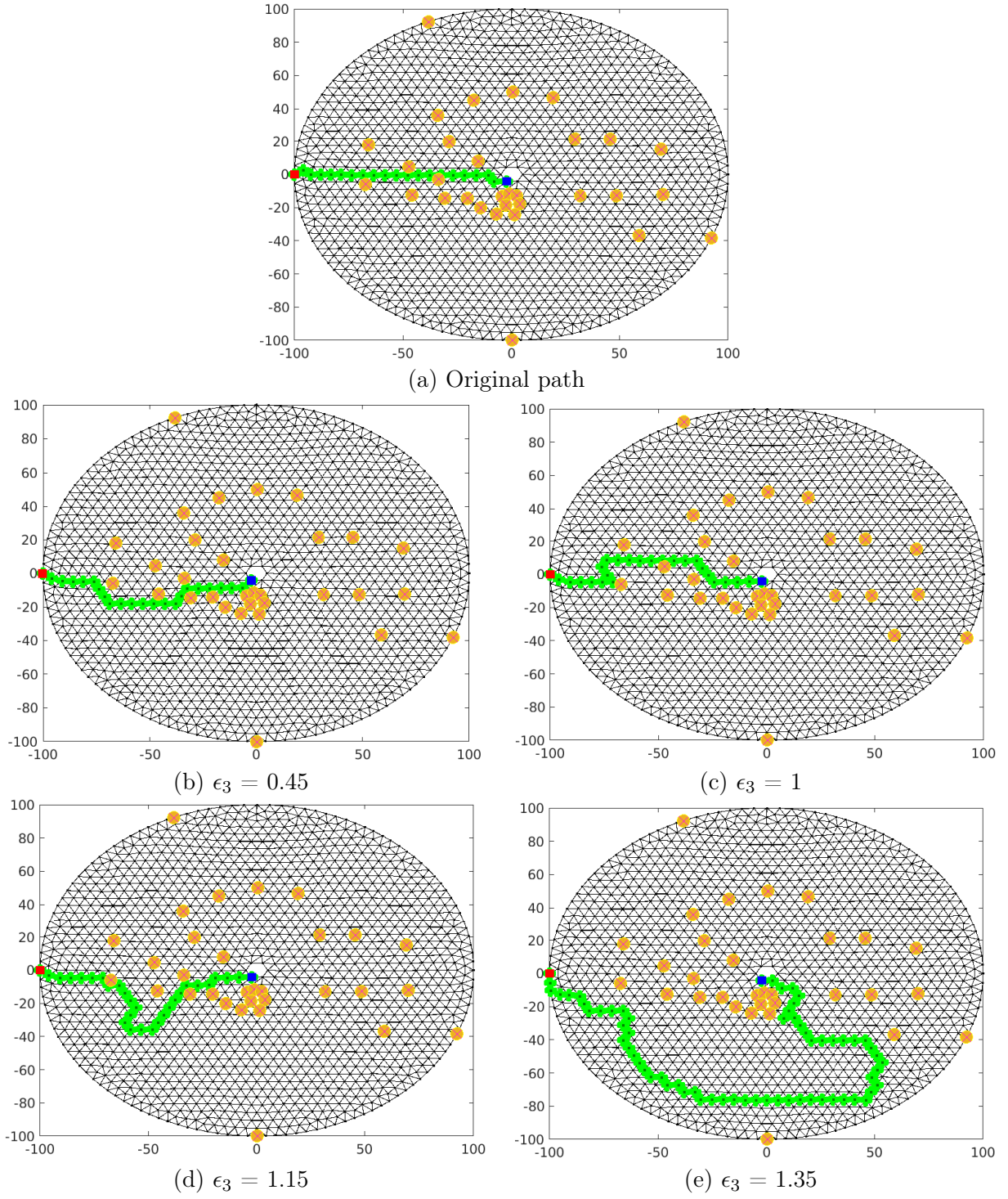


Figure 4.29: Alternative paths produced by CFDMPP on the CFDM

4.2.2.3 Cases 3 and 4

Cases 3 and 4 are similar in that both situations have $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ on the constraint edges of the CFM. Both cases exhibit similar behaviour with respect to the paths produced and the metrics, so they are grouped together for this discussion. Furthermore, case 3 demonstrates additional points for observation that case 4 does not. So, we will use case 3 as the case that is discussed in detail, while case 4 is simply referred to by its results. The performance metrics for case 3 are seen in Table 4.12 (or equivalently in Figure 4.35), Table 4.13, Table 4.14, Table 4.15 (or equivalently in Figure 4.36), and Table 4.16 (or equivalently in Figure 4.37). The paths produced by the PRM, RRT and CFDMPP in R-4 and R-1 are seen in Figure 4.30.

Case 3 consists of $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ lying on the ICE. Transitioning from case 2, one notices that in case 3, the separation between the start-goal pair has decreased even further. Therefore, this case is of paramount importance to gauge the performance of the CFDMPP with respect to the PRM and RRT. Figure 4.30 displays a tell-tale sign of the impact of decreased separation between the start-goal pair the resulting path quality. Figure 4.30 shows an obvious decline in path quality by the PRM, whereas the RRT seemingly does better. The actual decline in path quality may be seen in Table 4.12 or equivalently in Figure 4.35. Clearly, the CFDMPP yields paths that are resilient to separation between the start-goal pair. This may also be seen in Table 4.13.

Next, we observe the situations with 13 poles in Figure 4.33 and 31 poles in Figure 4.34 together. There are three observations to be made in both figures. First, the SAs produce a visibly large deviation from the required path as opposed to the CFDMPP. This is as expected. Second, the path produced by the CFDMPP when transformed to R-1 appears to cross into the constraint region. This is a consequence of the meshing parameters chosen, where the uniform mesh and its resolution result in a coarse approximation of the ICE with respect to its actual curvature. Recall that the effects of this choice were already seen in the results of case 1, where the CFDMPP produced a path length shorter than the ideal path length. Since the same mesh is used here, the same meshing parameters come into play here. The approximation of the ICE is a composition of edges with vertices. So, while the separation between the vertices and the centre of the ICE will still result in the desired radius, the sum of edge lengths may fall short of the true circumference of the ICE. The transformation between R-4 and R-1 is applied only to the vertices that constitute the path, and not to the edges connecting them. So, one may now verify in Figure 4.33 and Figure 4.34, that the vertices or points of the path are indeed correctly transformed to R-1. However since the edges from R-4 are not transformed, the edges that now connect the correctly transformed vertices in R-1 appear to infringe the constraints. To reiterate, this is a consequence of meshing parameters and not of CFDMPP's ability to produce a constraint free path. This topic is introduced here so that it can be revisited in Section 4.3 for a realistic environment.

The third and final observation is the ability of the CFDMPP to yield a path despite having a cluster of closely located poles in the region of interest. There is a deliberate difference in the placement of poles for the PRM, RRT and CFDMPP. This aids in visually demonstrating the challenges faced by the SAs, even when the poles in their environments are not dense clusters, and are sparse in comparison to the poles of the CFDMPP. That is,

despite there being an absence of narrow corridors or clustered poles, the [SAs](#)' challenge with generating a path with minimal detours seems to be related to the proximity of the start and goal locations. The performance metrics may be used to explicitly quantify the success of the [CFDMPP](#) in this case, since it is unaffected by the proximity of start-goal pair.

Observing Figure 4.35 or equivalently Table 4.12 and Table 4.13 for the [CFDMPP](#), one notices that there exists a positive percentage of path length deviation or a ratio of greater than 1, even with the coarse approximation of the [ICE](#). This in contrast to case 1, where there was a negative path length deviation. In this case, the difference in behaviour is because of the following. Since the [CFDMPP](#) performs path planning on the discretized [CFM](#), the provided $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ are mapped to their closest node equivalents in the mesh, i.e. $\text{node}^{\text{start}}$ and $\text{node}^{\text{goal}}$. The coarse approximation may yield $\text{node}^{\text{start}}$ and $\text{node}^{\text{goal}}$, that are slightly farther away from the actual $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$. Since the ideal path lengths are computed based on true locations, i.e. $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, the obtained path length may still be more than the ideal path length. Obviously, this is resolved using a finer mesh resolution, but there exist implications; they will be discussed for a realistic environment in Section 4.3.

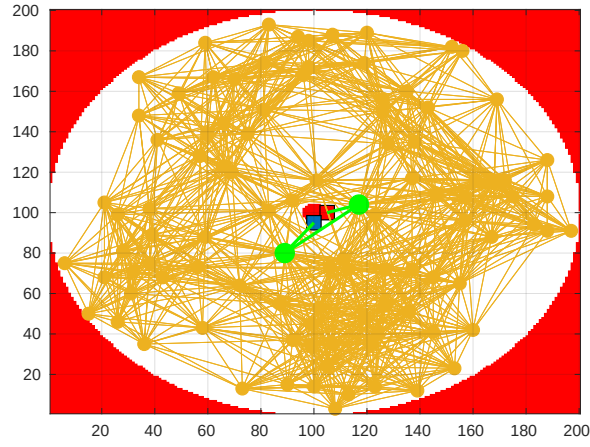
The alternative paths produced for 13 and 31 poles are seen in Figure 4.33 and Figure 4.34 respectively. They provide a clear demonstration of the strengths and benefits of the [CFDMPP](#), which are: the generation of the shortest path class and an alternative path class, devoid of start-goal proximity or pole cluster implications. This therefore leads to the [CFDMPP](#) being seen as extremely beneficial to being used in realistic environments. This is especially true when go around manoeuvres are required to avoid the second group of [DCs](#), but the next available path class also borders the constraint edge. This is precisely what is seen in Figure 4.33 and Figure 4.34, and also when observing the possible alternative paths in Figure 4.38.

We now examine the alternative path classes for the given start-goal pair and poles seen in Figure 4.38. Due to the nature of where the poles are located, there are only two path classes intuitively possible. The first is the original path class and the second is the alternative path class as seen with $\epsilon_3 = 0.45$, with the correct, mathematically shortest path in each path class. However, if we push the algorithm to produce alternative paths beyond this, we can start to see the effects of meshing and tuning parameters emerge. For example, in the figure corresponding to the $\epsilon_3 = 0.8$, the path actually belongs to the original path class. But, the tuning and meshing parameters account for it as the next class after what is seen with $\epsilon_3 = 0.45$. A similar behaviour is seen with the path produced in $\epsilon_3 = 1.05$ as well. These behaviours are important to keep track of, since they show the tolerance of the [CFDMPP](#), for a given set of tuning and meshing parameters.

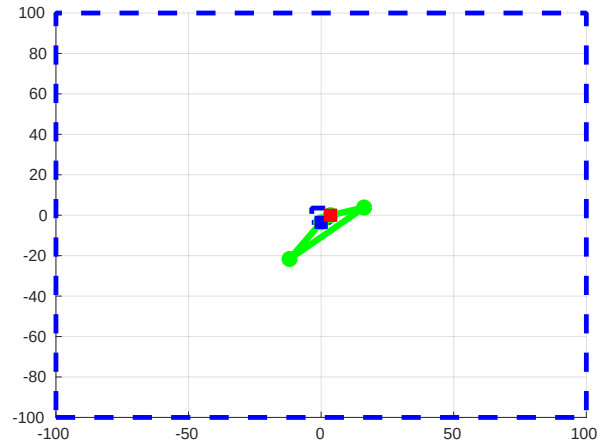
In this case, the [CFDMPP](#) successfully produced the two correct, required path classes, based on the initial estimate of the parameters. Any further paths that the algorithm is probed for, are still not ideal, but are viable, constraint-free paths. One can see from the metrics, that it is only at this point i.e. after returning at least one alternative path class (in this case, the only one) does the [CFDMPP](#) gradually return longer paths. These path lengths are still comparatively better than those produced by [SAs](#).

Case 4 is also seen to exhibit such behaviours. The figures corresponding to case 4's comparisons of paths across algorithms are seen in Figure 4.39, Figure 4.40 and Figure 4.41.

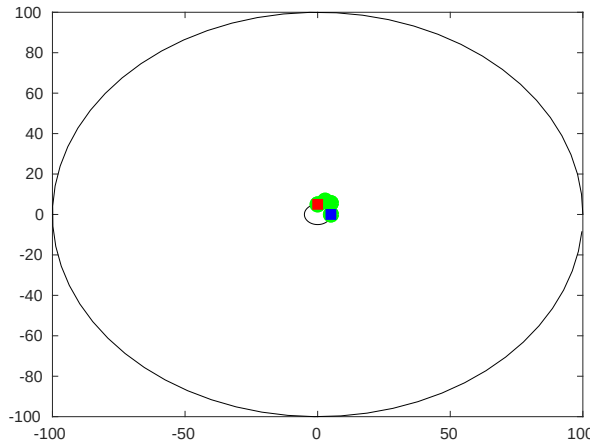
Blocked paths for 13 and 31 poles are seen in Figure 4.42 and Figure 4.43. The performance metrics for case 4 are seen in Table 4.17 (or equivalently in Figure 4.44), Table 4.18, Table 4.19, Table 4.20 (or equivalently in Figure 4.45) and Table 4.21 (or equivalently in Figure 4.46). The main observation here is the difference in the goal location seen in the environment of the PRM. Since it is grid based, the goal location is not precisely marked as it is in the RRT or CFDMPP. Note that any path length related metric is computed after having taking into account such individual variations. Alternative path classes are seen in Figure 4.47. Similar to case 3, one can see that there exists only two path classes as seen with the original path and the first alternative path class at $\epsilon_3 = 0.28$. The path corresponding to $\epsilon_3 = 0.6$ bears the impact of tuning and meshing parameters, and thus truly belongs to the original path class; however, it is still viable and constraint-free.



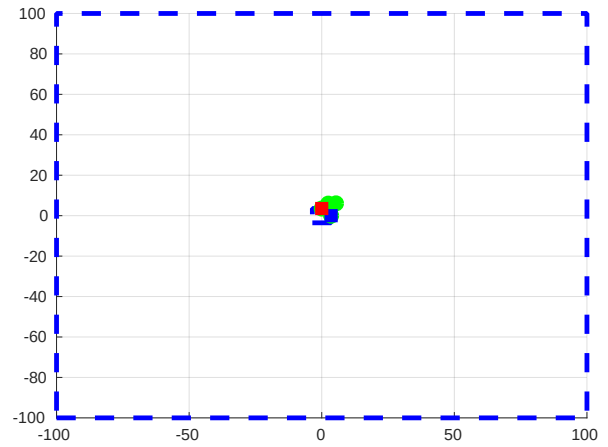
(a) PRM: Path in R-4



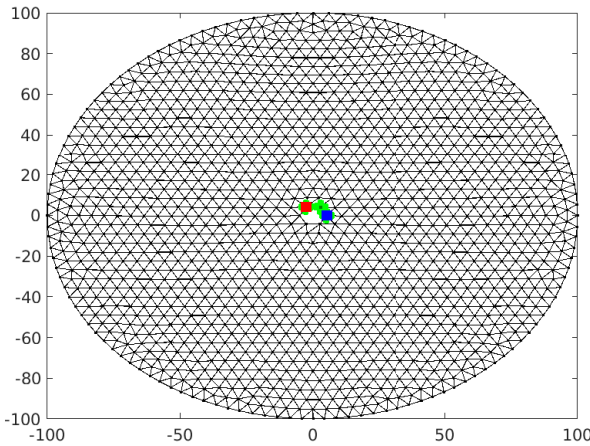
(b) PRM: Path in R-1



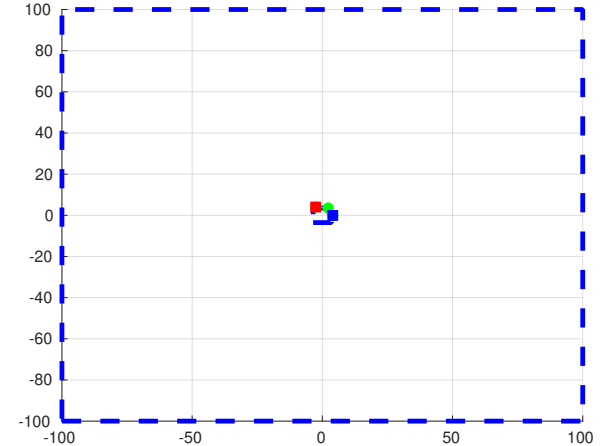
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.30: Case 3: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.

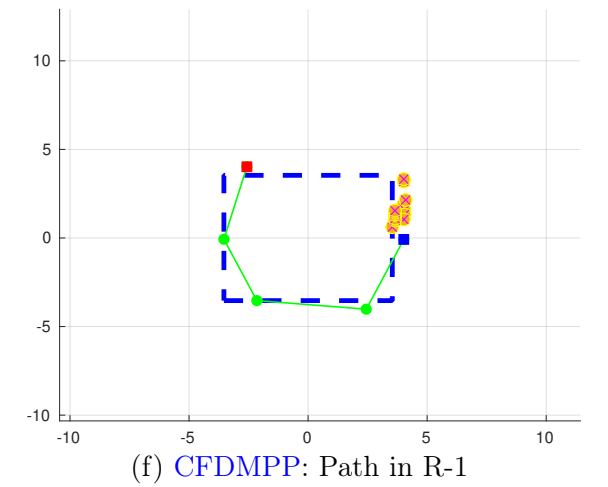
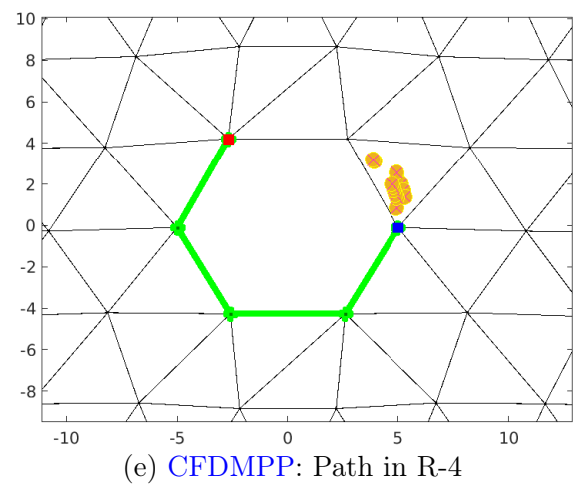
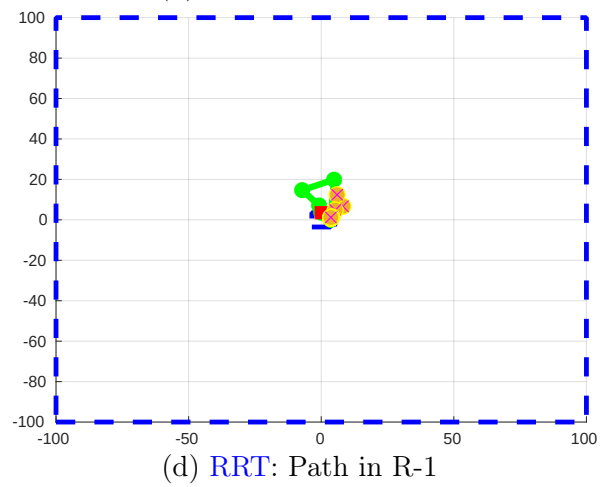
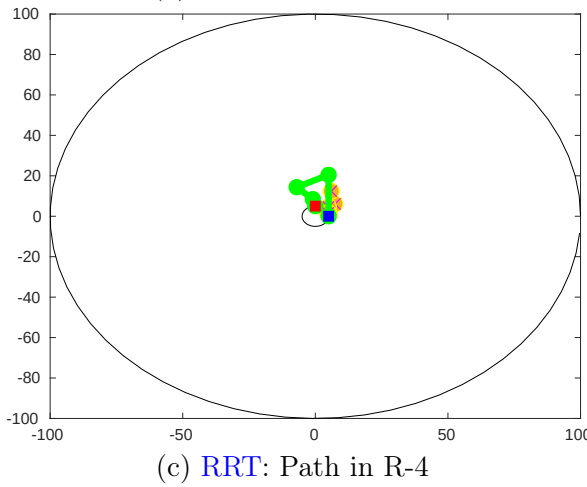
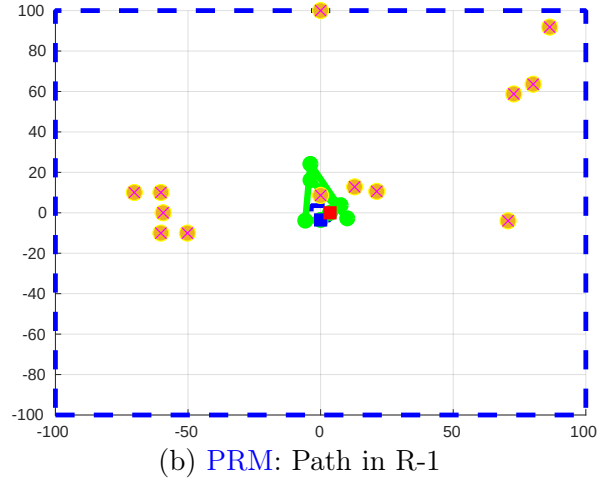
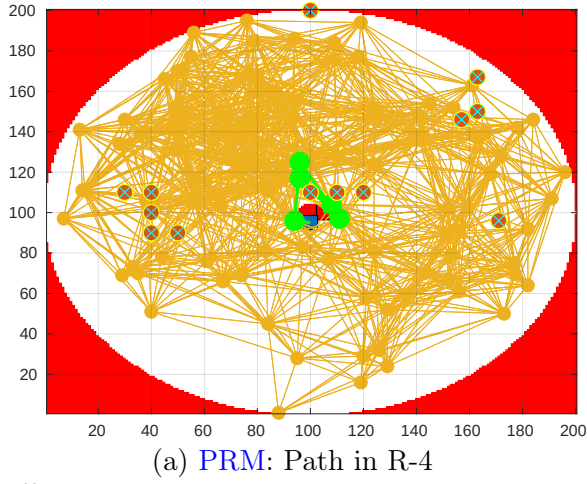
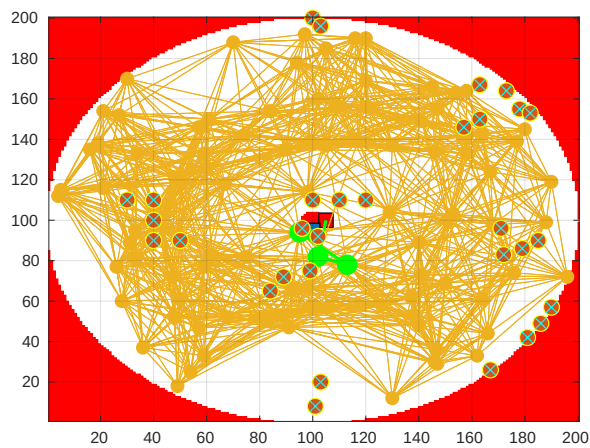
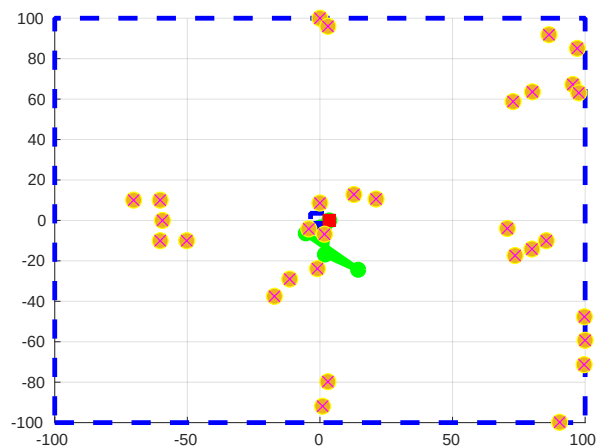


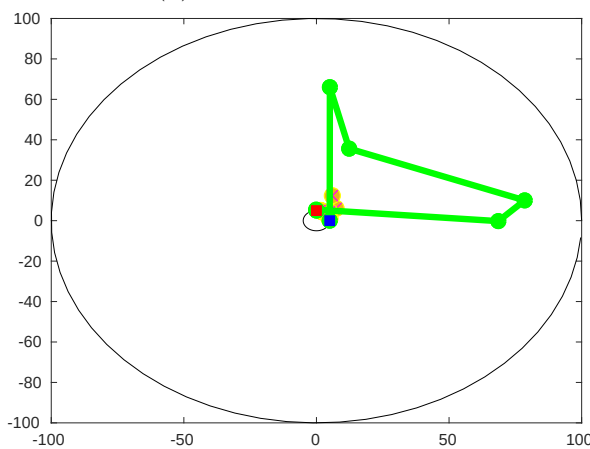
Figure 4.31: Case 3: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.



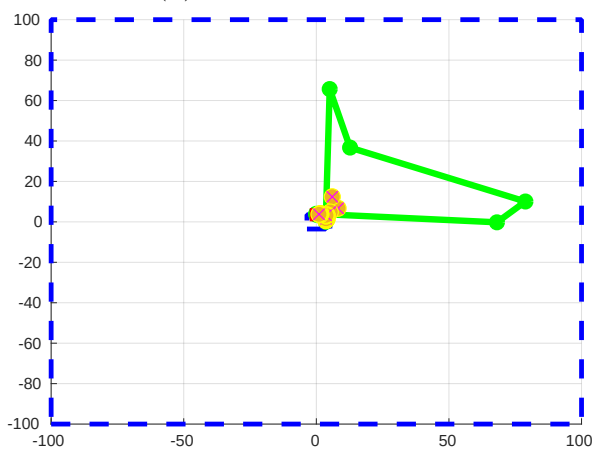
(a) PRM: Path in R-4



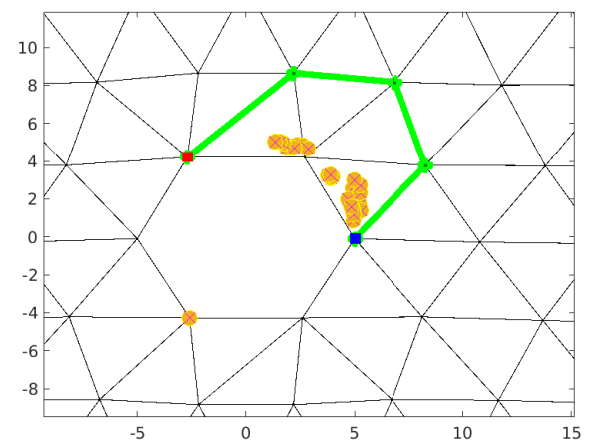
(b) PRM: Path in R-1



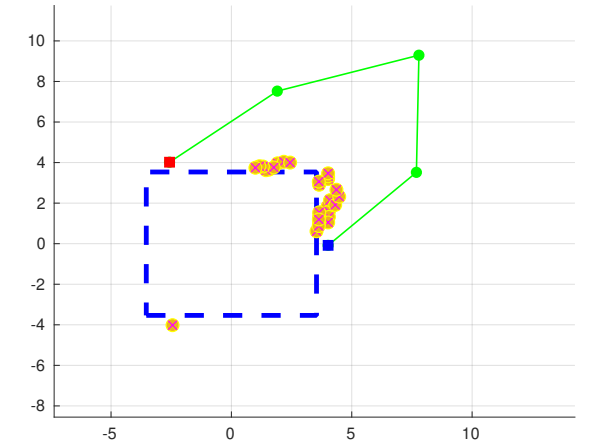
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.32: Case 3: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.

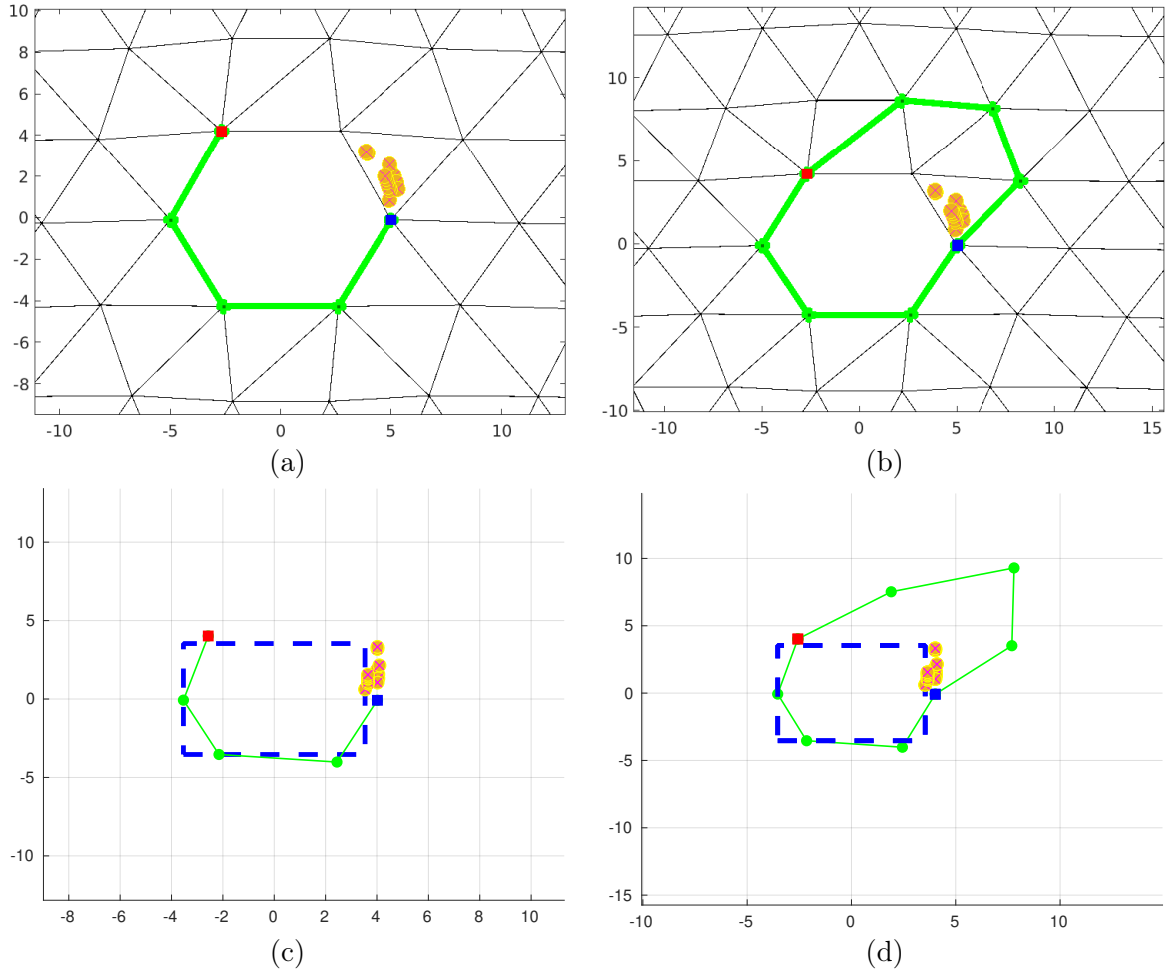


Figure 4.33: Case 3: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs HB Threshold = 0.55)). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

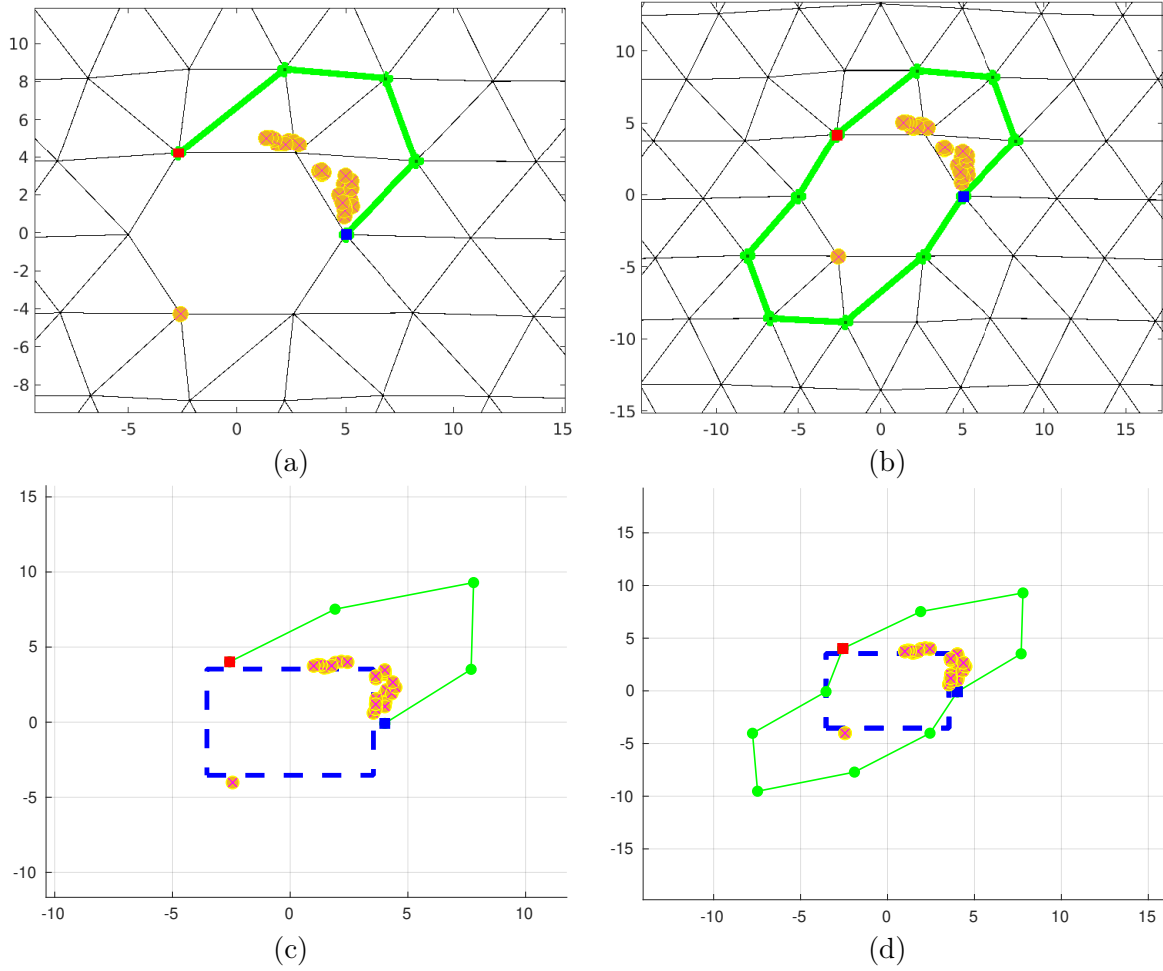


Figure 4.34: Case 3: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 0.45$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

Table 4.12: Case 3 Percentage of Path Length Deviation

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	962.34, 575.73	329.37, 457.77	30.88, 0.00	NaN, NaN
1	816.92, 527.77	339.41, 476.43	30.88, 0.00	226.10, 0.00
4	756.03, 479.49	364.44, 475.06	30.88, 0.00	150.97, 0.00
7	1034.89, 590.78	451.34, 665.83	30.88, 0.00	364.27, 0.00
10	695.80, 425.00	554.31, 812.50	30.88, 0.00	224.37, 0.00
13	632.52, 511.89	378.48, 761.70	150.97, 0.00	167.13, 0.00
16	702.73, 451.55	549.84, 816.04	150.97, 0.00	NaN, NaN
19	664.04, 428.23	713.46, 1288.30	167.13, 0.00	289.96, 0.00
22	571.07, 335.13	375.22, 674.10	167.13, 0.00	1008.76, 0.00
25	714.06, 431.72	345.47, 439.11	167.13, 0.00	1134.59, 0.00
28	878.46, 500.16	443.07, 597.51	167.13, 0.00	289.96, 0.00
31	747.32, 517.94	702.05, 1560.97	167.13, 0.00	289.96, 0.00

Table 4.13: Case 3 Ratio of observed path length to ideal length

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	10.62, 5.76	4.29, 4.58	1.31, 0.00	NaN, NaN
1	9.17, 5.28	4.39, 4.76	1.31, 0.00	3.26, 0.00
4	8.56, 4.79	4.64, 4.75	1.31, 0.00	2.51, 0.00
7	11.35, 5.91	5.51, 6.66	1.31, 0.00	4.64, 0.00
10	7.96, 4.25	6.54, 8.12	1.31, 0.00	3.24, 0.00
13	7.33, 5.12	4.78, 7.62	2.51, 0.00	2.67, 0.00
16	8.03, 4.52	6.50, 8.16	2.51, 0.00	NaN, NaN
19	7.64, 4.28	8.13, 12.88	2.67, 0.00	3.90, 0.00
22	6.71, 3.35	4.75, 6.74	2.67, 0.00	11.09, 0.00
25	8.14, 4.32	4.45, 4.39	2.67, 0.00	12.35, 0.00
28	9.78, 5.00	5.43, 5.98	2.67, 0.00	3.90, 0.00
31	8.47, 5.18	8.02, 15.61	2.67, 0.00	3.90, 0.00

Table 4.14: Case 3 Percentage of usable samples

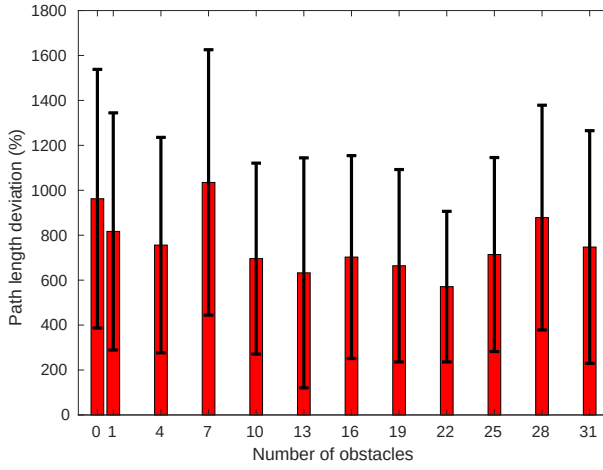
# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	78.30, 3.45	5.06, 3.44	100.00, 0.00	NaN, NaN
1	77.39, 3.69	3.76, 2.94	99.97, 0.00	99.95, 0.00
4	78.25, 4.25	6.62, 6.20	99.93, 0.00	99.93, 0.00
7	77.87, 4.01	5.04, 4.77	99.93, 0.00	99.91, 0.00
10	78.65, 3.90	3.61, 4.64	99.97, 0.00	99.89, 0.00
13	78.28, 3.16	5.89, 4.18	99.89, 0.00	99.90, 0.00
16	79.03, 2.84	3.47, 2.53	99.66, 0.00	NaN, NaN
19	78.67, 2.73	5.79, 5.84	99.38, 0.00	99.55, 0.00
22	80.19, 3.64	9.77, 9.59	99.69, 0.00	99.97, 0.00
25	77.81, 3.73	4.66, 6.12	99.69, 0.00	99.96, 0.00
28	79.02, 3.61	5.11, 4.71	99.69, 0.00	99.78, 0.00
31	76.72, 3.74	7.38, 5.27	99.75, 0.00	99.78, 0.00

Table 4.15: Case 3 Percentage Rejected Samples

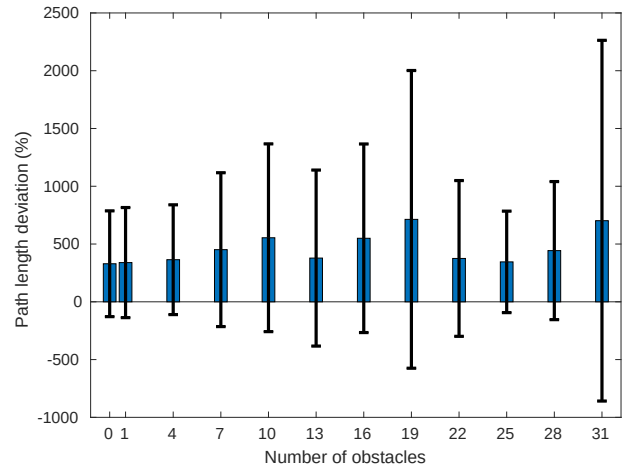
# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	21.70, 3.45	94.94, 3.44	0.00, 0.00	NaN, NaN
1	22.61, 3.69	96.24, 2.94	0.03, 0.00	0.05, 0.00
4	21.75, 4.25	93.38, 6.20	0.07, 0.00	0.07, 0.00
7	22.13, 4.01	94.96, 4.77	0.07, 0.00	0.09, 0.00
10	21.35, 3.90	96.39, 4.64	0.03, 0.00	0.11, 0.00
13	21.72, 3.16	94.11, 4.18	0.11, 0.00	0.10, 0.00
16	20.97, 2.84	96.53, 2.53	0.34, 0.00	NaN, NaN
19	21.33, 2.73	94.21, 5.84	0.62, 0.00	0.45, 0.00
22	19.81, 3.64	90.23, 9.59	0.31, 0.00	0.03, 0.00
25	22.19, 3.73	95.34, 6.12	0.31, 0.00	0.04, 0.00
28	20.98, 3.61	94.89, 4.71	0.31, 0.00	0.22, 0.00
31	23.28, 3.74	92.62, 5.27	0.25, 0.00	0.22, 0.00

Table 4.16: Case 3 Run time

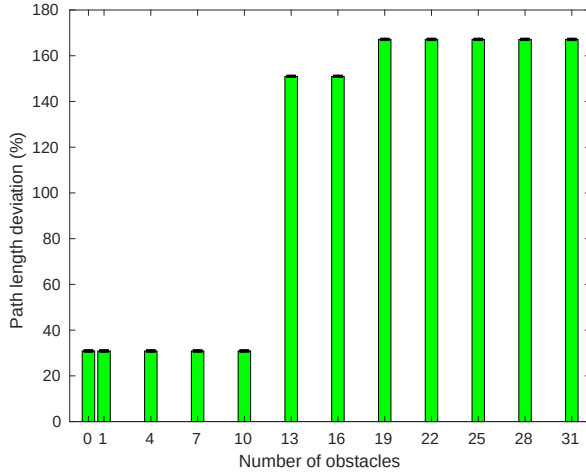
# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	2.03, 0.08	0.10, 0.13	0.14, 0.00	NaN, NaN
1	2.05, 0.07	0.32, 1.20	0.14, 0.00	0.15, 0.01
4	2.05, 0.11	0.25, 0.40	0.15, 0.00	0.15, 0.01
7	2.02, 0.09	0.10, 0.10	0.15, 0.01	0.18, 0.01
10	2.01, 0.10	0.16, 0.23	0.16, 0.01	0.17, 0.01
13	1.97, 0.09	0.11, 0.14	0.17, 0.01	0.17, 0.01
16	2.02, 0.10	0.28, 0.56	0.17, 0.00	NaN, NaN
19	2.02, 0.07	0.08, 0.09	0.17, 0.01	0.18, 0.01
22	2.00, 0.06	0.17, 0.30	0.17, 0.01	0.24, 0.01
25	1.99, 0.07	0.11, 0.15	0.18, 0.01	0.30, 0.02
28	2.02, 0.08	0.16, 0.22	0.18, 0.01	0.19, 0.01
31	2.02, 0.07	0.19, 0.35	0.19, 0.01	0.19, 0.01



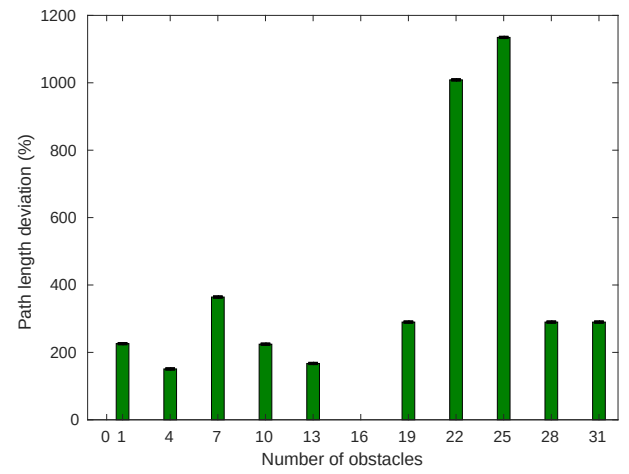
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

Figure 4.35: Case 3: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.

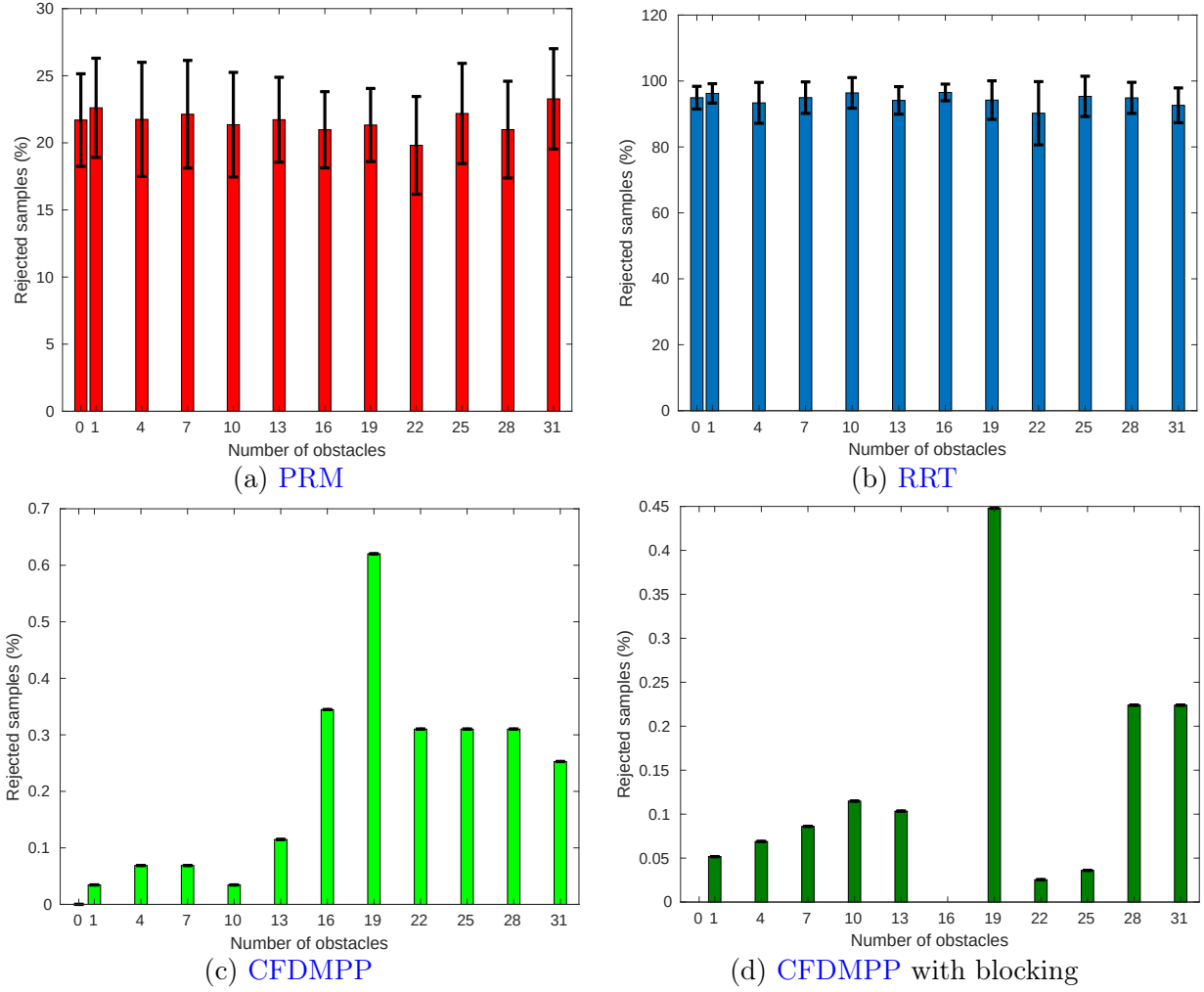
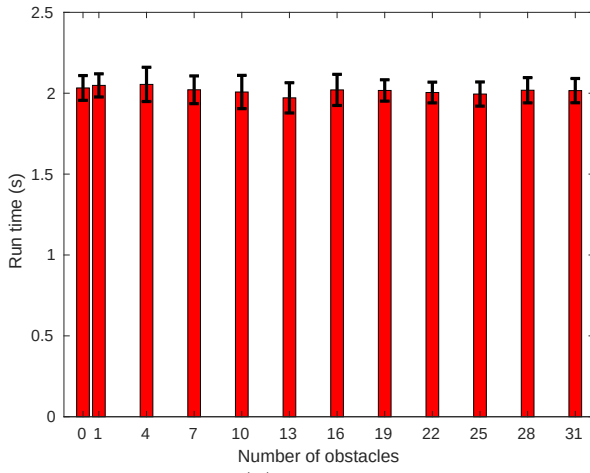
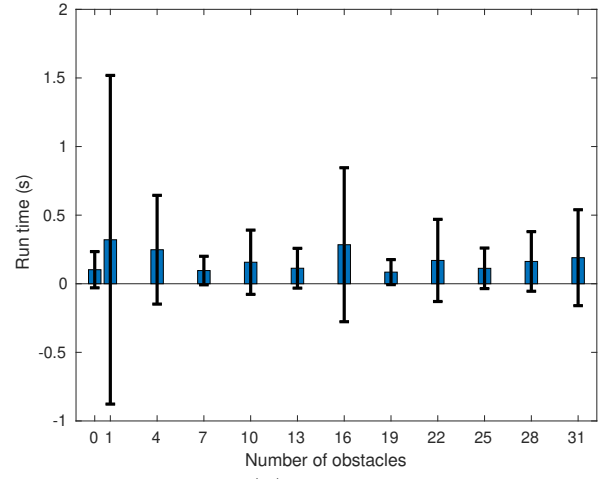


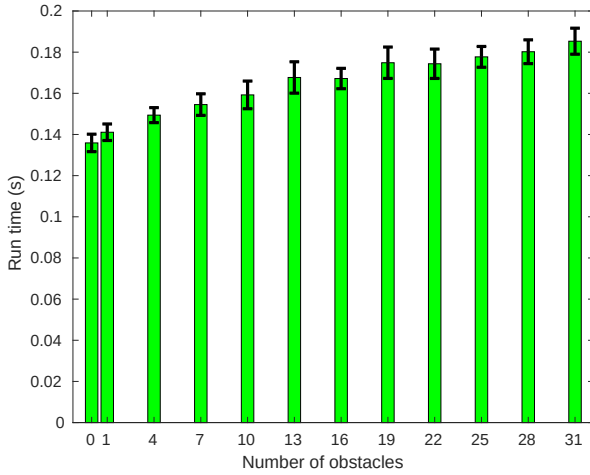
Figure 4.36: Case 3: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



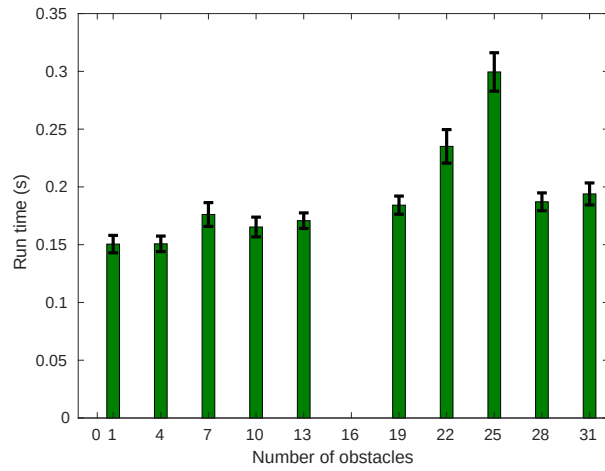
(a) PRM



(b) RRT

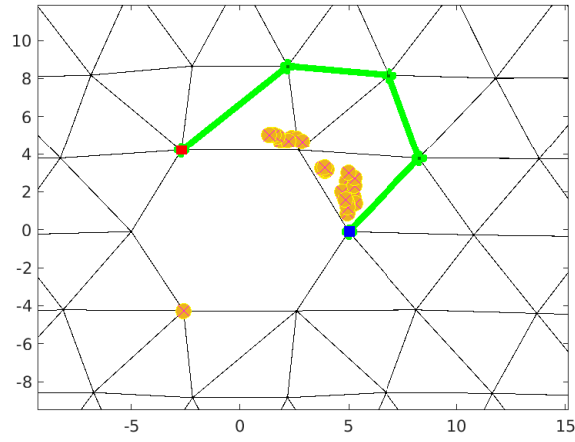


(c) CFDMPP

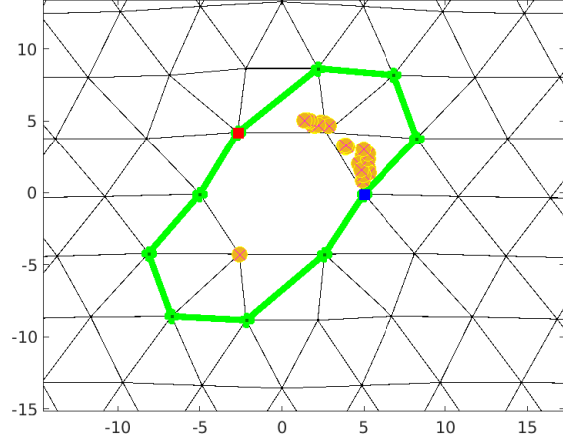


(d) CFDMPP with blocking

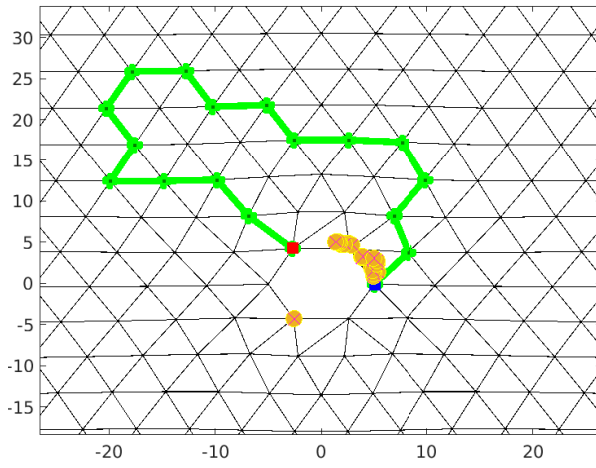
Figure 4.37: Case 3: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



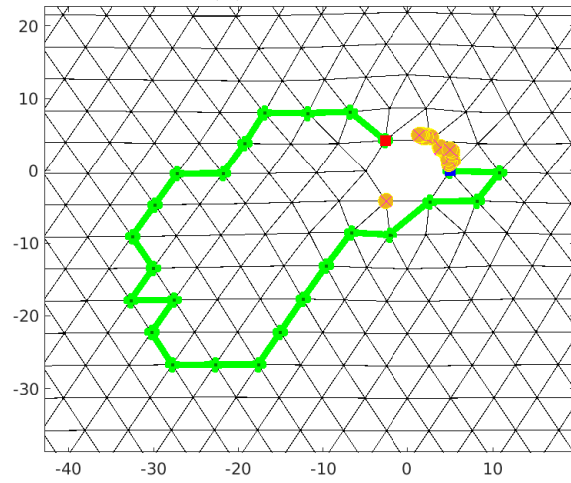
(a) Original path



(b) $\epsilon_3 = 0.45$

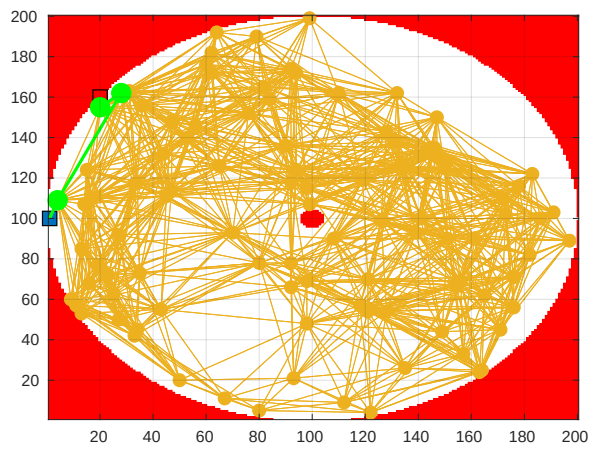


(c) $\epsilon_3 = 0.8$

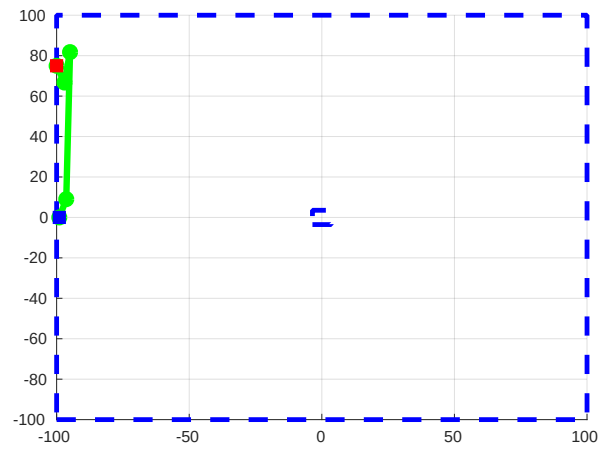


(d) $\epsilon_3 = 1.05$

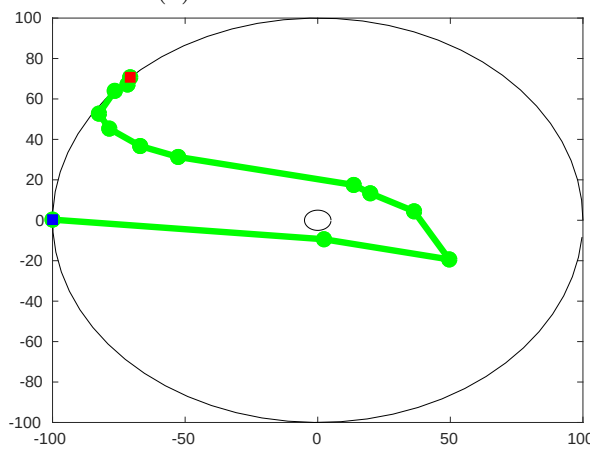
Figure 4.38: Alternative paths produced by CFDMPP on the CFDM



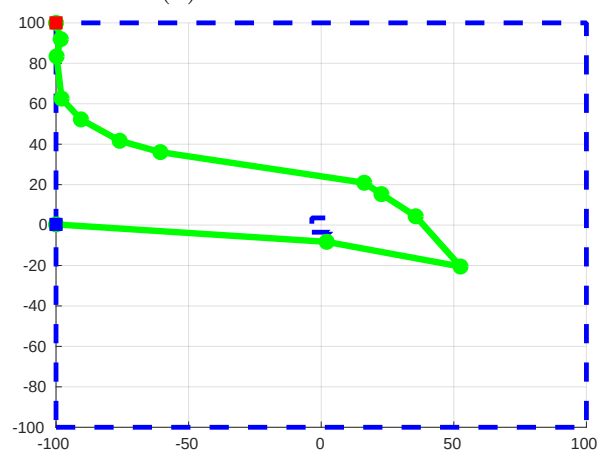
(a) PRM: Path in R-4



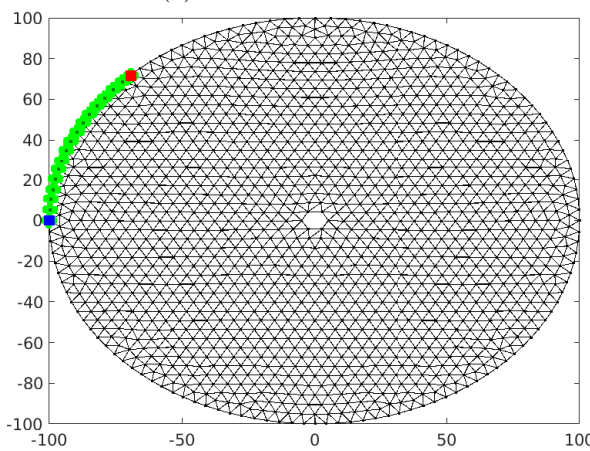
(b) PRM: Path in R-1



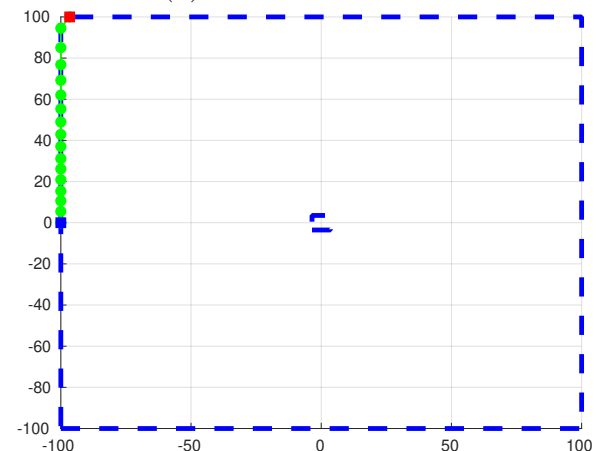
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.39: Case 4: Paths generated by PRM, RRT and CFDMPP with no obstacles/poles in the environment; paths shown in R-4 and R-1.

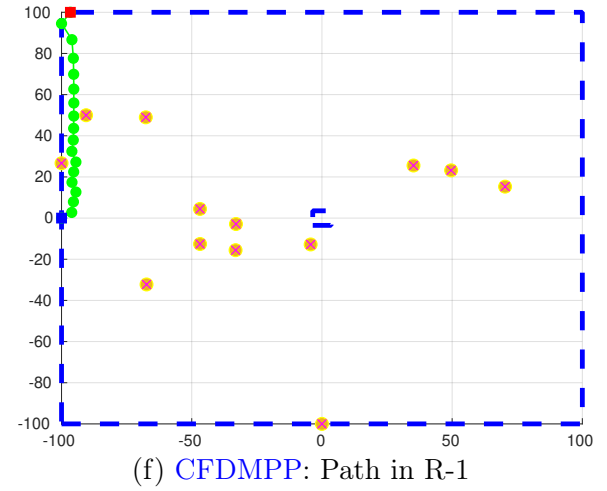
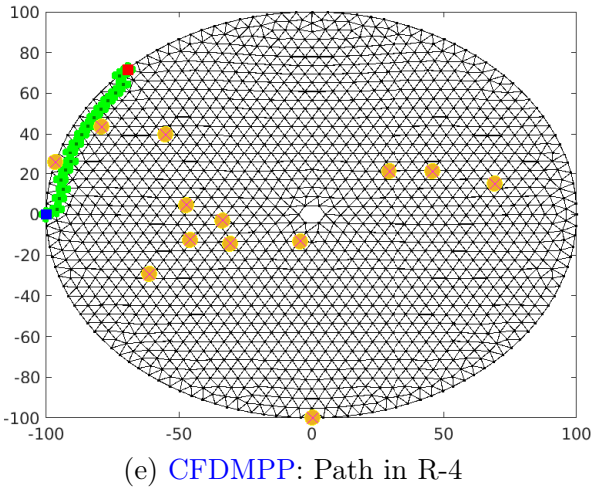
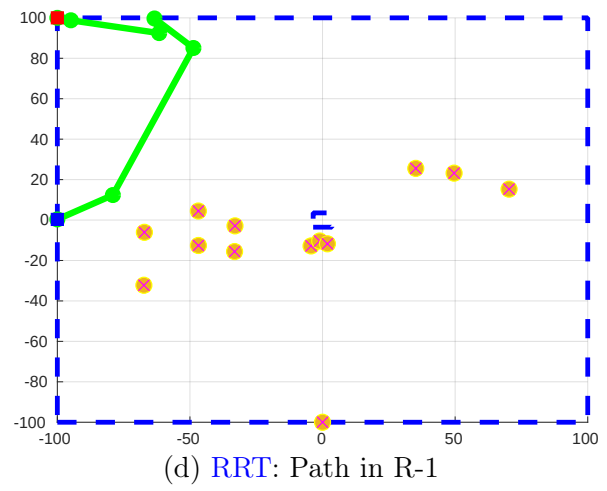
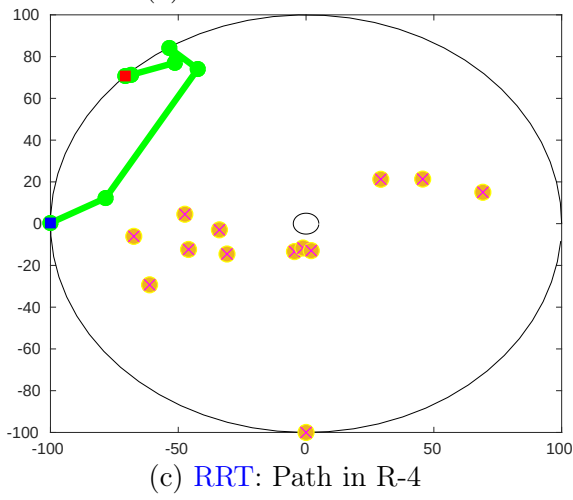
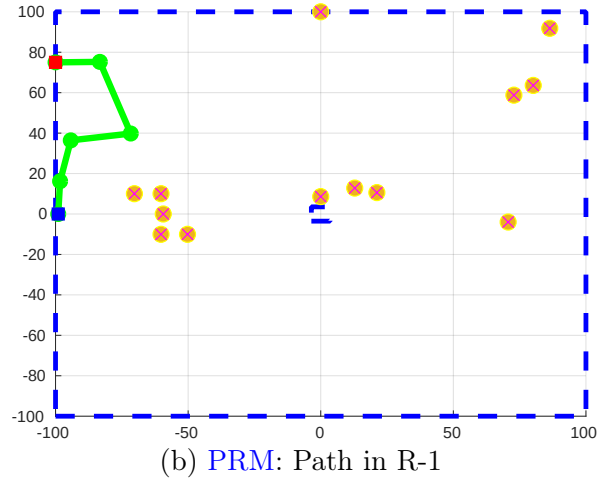
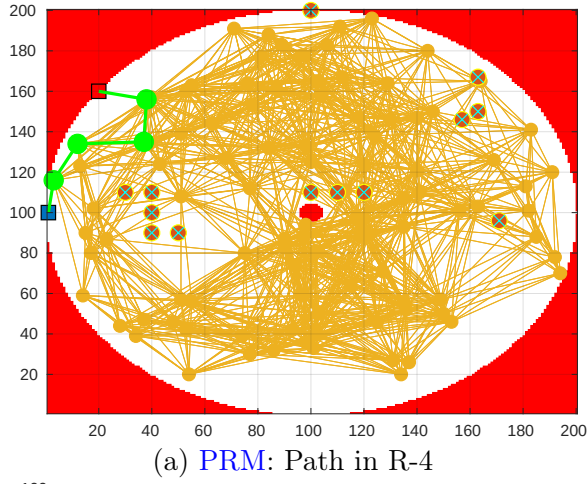
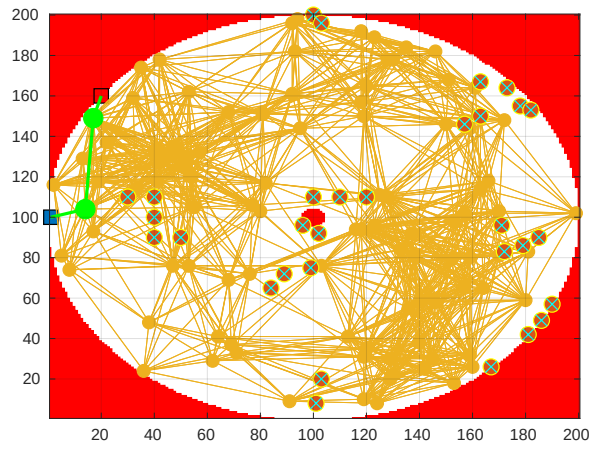
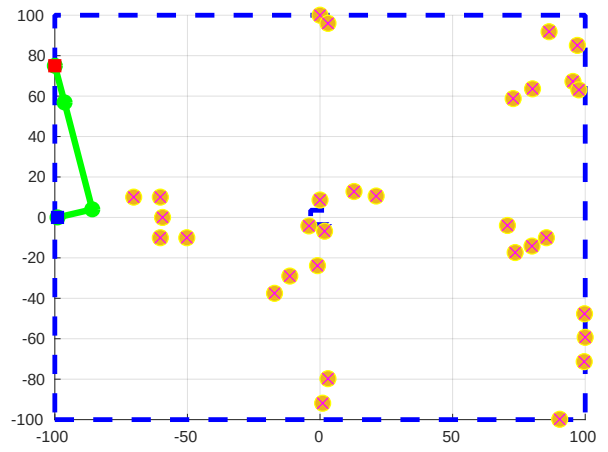


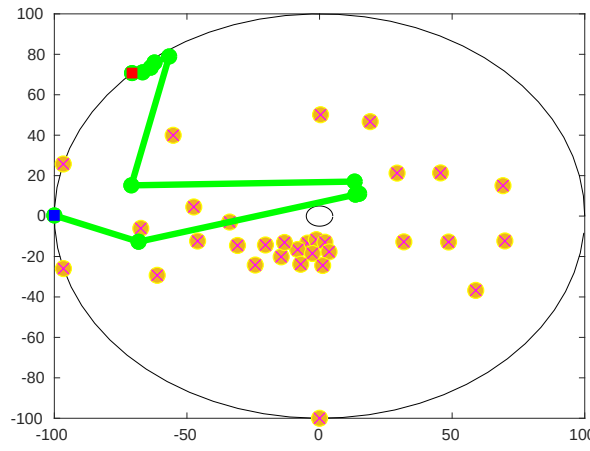
Figure 4.40: Case 4: Paths generated by PRM, RRT and CFDMPP with 13 obstacles/poles in the environment; paths shown in R-4 and R-1.



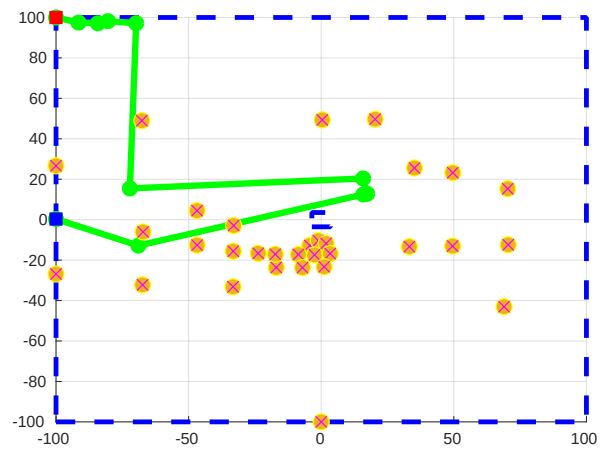
(a) PRM: Path in R-4



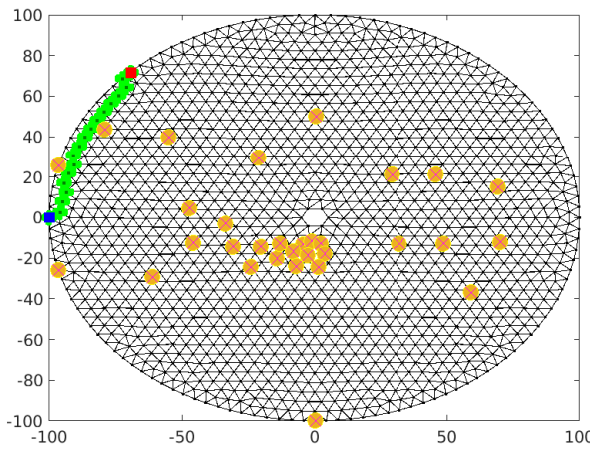
(b) PRM: Path in R-1



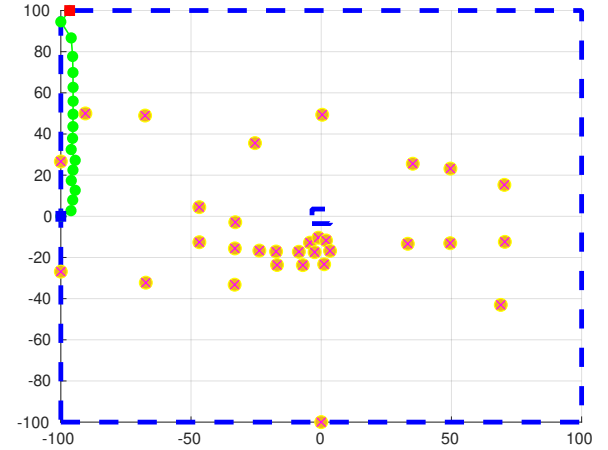
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure 4.41: Case 4: Paths generated by PRM, RRT and CFDMPP with 31 obstacles/poles in the environment; paths shown in R-4 and R-1.

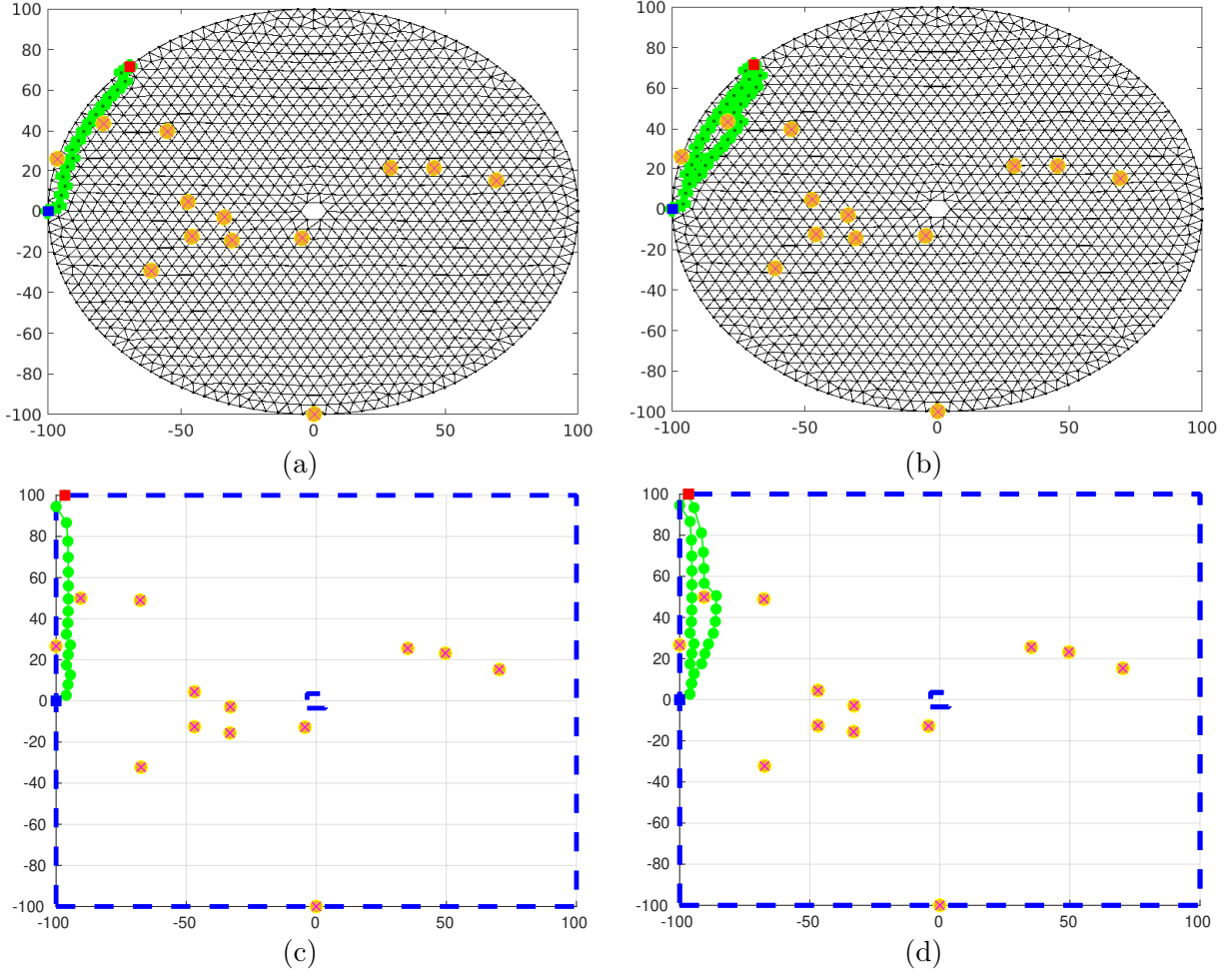


Figure 4.42: Case 4: CFDMPP in the case of 13 poles (first group of DCs and 1 blocked path (second group of DCs HB Threshold = 0.55)). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

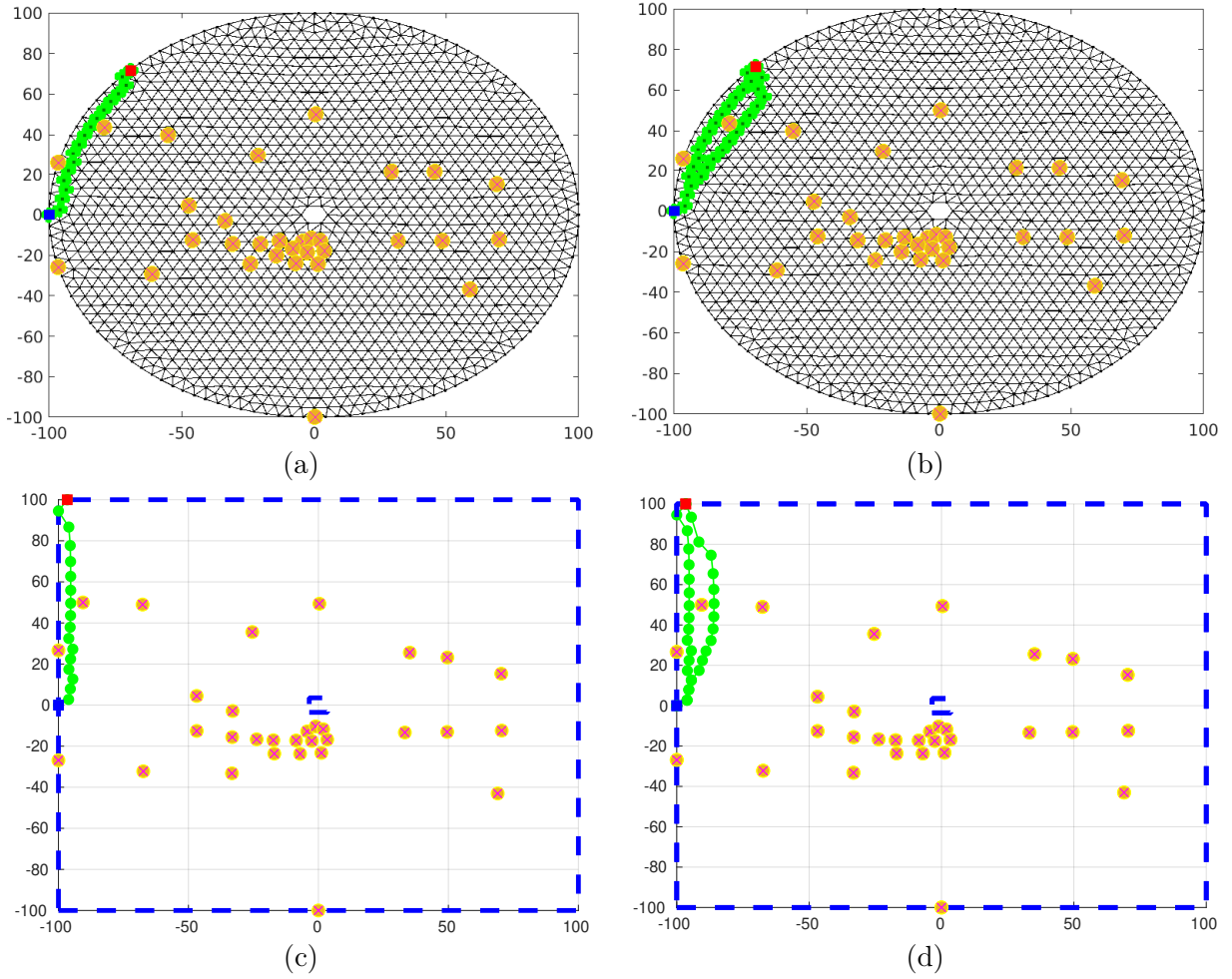


Figure 4.43: Case 4: CFDMPP in the case of 31 poles (first group of DCs and 1 blocked path (second group of DCs. $\epsilon_3 = 0.45$). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

Table 4.17: Case 4 Percentage of Path Length Deviation

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	62.50, 40.17	252.92, 134.35	5.26, 0.00	NaN, NaN
1	48.18, 40.99	259.49, 178.26	5.26, 0.00	NaN, NaN
4	71.57, 63.97	254.06, 140.58	7.56, 0.00	23.84, 0.00
7	53.98, 42.80	206.71, 153.30	7.56, 0.00	11.95, 0.00
10	63.35, 56.38	203.86, 179.76	7.56, 0.00	27.17, 0.00
13	45.39, 40.24	244.56, 186.51	7.56, 0.00	9.88, 0.00
16	70.31, 52.06	231.57, 135.27	7.56, 0.00	20.09, 0.00
19	59.93, 44.26	235.01, 150.69	7.56, 0.00	14.96, 0.00
22	68.50, 56.70	190.57, 125.40	7.56, 0.00	14.96, 0.00
25	57.49, 42.62	319.86, 218.37	7.56, 0.00	10.64, 0.00
28	62.63, 38.97	224.46, 166.89	7.56, 0.00	14.96, 0.00
31	47.53, 28.24	201.86, 141.73	7.56, 0.00	9.54, 0.00

Table 4.18: Case 4 Ratio of observed path length to ideal length

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	1.59, 0.39	3.53, 1.34	1.05, 0.00	NaN, NaN
1	1.45, 0.40	3.59, 1.78	1.05, 0.00	NaN, NaN
4	1.68, 0.63	3.54, 1.41	1.08, 0.00	1.24, 0.00
7	1.51, 0.42	3.07, 1.53	1.08, 0.00	1.12, 0.00
10	1.60, 0.55	3.04, 1.80	1.08, 0.00	1.27, 0.00
13	1.42, 0.39	3.45, 1.87	1.08, 0.00	1.10, 0.00
16	1.67, 0.51	3.32, 1.35	1.08, 0.00	1.20, 0.00
19	1.56, 0.43	3.35, 1.51	1.08, 0.00	1.15, 0.00
22	1.65, 0.55	2.91, 1.25	1.08, 0.00	1.15, 0.00
25	1.54, 0.42	4.20, 2.18	1.08, 0.00	1.11, 0.00
28	1.59, 0.38	3.24, 1.67	1.08, 0.00	1.15, 0.00
31	1.44, 0.28	3.02, 1.42	1.08, 0.00	1.10, 0.00

Table 4.19: Case 4 Percentage of usable samples

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	78.48, 3.82	46.34, 0.44	100.00, 0.00	NaN, NaN
1	79.26, 3.51	46.48, 0.51	100.00, 0.00	NaN, NaN
4	78.71, 3.59	46.13, 0.69	99.88, 0.00	99.92, 0.00
7	77.95, 3.28	46.48, 0.76	99.91, 0.00	99.92, 0.00
10	78.21, 3.09	46.46, 0.55	99.91, 0.00	99.94, 0.00
13	79.43, 4.27	46.52, 0.59	99.90, 0.00	99.92, 0.00
16	79.20, 3.49	46.45, 0.70	99.90, 0.00	99.93, 0.00
19	76.94, 3.34	46.54, 1.01	99.88, 0.00	99.93, 0.00
22	78.54, 3.78	46.61, 0.93	99.88, 0.00	99.92, 0.00
25	80.23, 2.91	46.29, 0.54	99.88, 0.00	99.90, 0.00
28	77.97, 3.80	46.42, 1.45	99.88, 0.00	99.94, 0.00
31	79.25, 3.34	46.49, 0.58	99.88, 0.00	99.91, 0.00

Table 4.20: Case 4 Percentage Rejected Samples

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	21.52, 3.82	53.66, 0.44	0.00, 0.00	NaN, NaN
1	20.74, 3.51	53.52, 0.51	0.00, 0.00	NaN, NaN
4	21.29, 3.59	53.87, 0.69	0.12, 0.00	0.08, 0.00
7	22.05, 3.28	53.52, 0.76	0.09, 0.00	0.08, 0.00
10	21.79, 3.09	53.54, 0.55	0.09, 0.00	0.06, 0.00
13	20.57, 4.27	53.48, 0.59	0.10, 0.00	0.08, 0.00
16	20.80, 3.49	53.55, 0.70	0.10, 0.00	0.07, 0.00
19	23.06, 3.34	53.46, 1.01	0.12, 0.00	0.07, 0.00
22	21.46, 3.78	53.39, 0.93	0.12, 0.00	0.08, 0.00
25	19.77, 2.91	53.71, 0.54	0.12, 0.00	0.10, 0.00
28	22.03, 3.80	53.58, 1.45	0.12, 0.00	0.06, 0.00
31	20.75, 3.34	53.51, 0.58	0.12, 0.00	0.09, 0.00

Table 4.21: Case 4 Run time

# Poles	PRM	RRT	CFDMPP	CFDMPP with blocking
	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)	(Mean, Std.Dev)
0	2.01, 0.08	3.44, 5.52	0.17, 0.01	NaN, NaN
1	2.02, 0.09	5.57, 17.25	0.18, 0.01	NaN, NaN
4	2.04, 0.08	2.43, 4.42	0.18, 0.01	0.23, 0.02
7	2.05, 0.08	1.77, 3.77	0.19, 0.01	0.21, 0.02
10	2.03, 0.08	3.44, 6.99	0.19, 0.01	0.26, 0.02
13	2.02, 0.09	1.18, 2.47	0.20, 0.01	0.20, 0.01
16	2.03, 0.09	2.37, 3.97	0.20, 0.01	0.25, 0.02
19	2.01, 0.09	2.99, 4.98	0.20, 0.01	0.23, 0.02
22	1.94, 0.07	0.77, 1.43	0.20, 0.01	0.23, 0.02
25	1.97, 0.12	2.95, 5.07	0.21, 0.01	0.22, 0.02
28	1.98, 0.07	2.96, 4.34	0.21, 0.01	0.23, 0.02
31	2.01, 0.08	2.44, 3.20	0.21, 0.01	0.22, 0.02

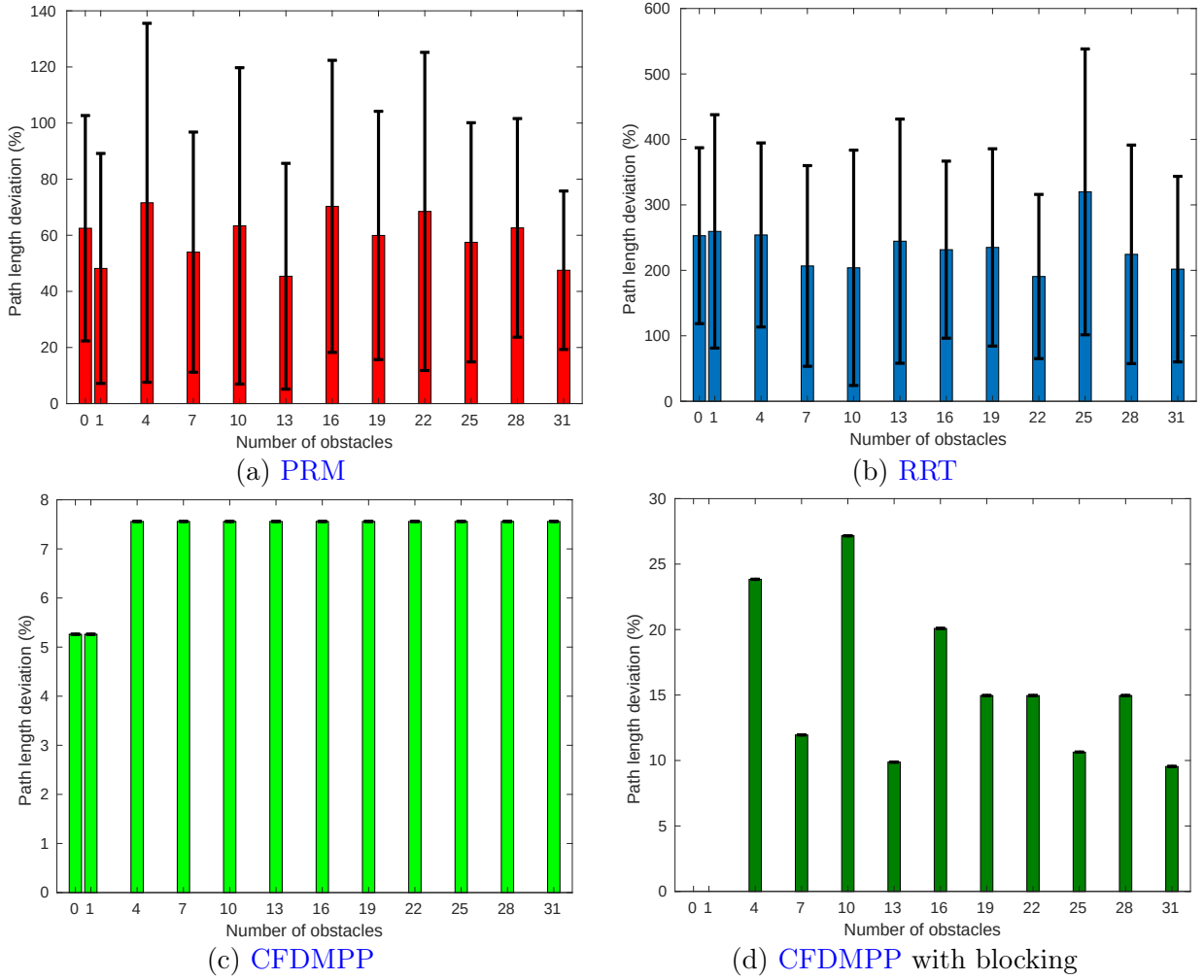
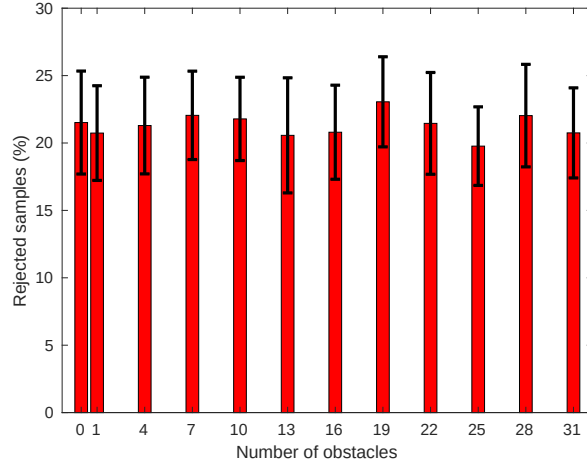
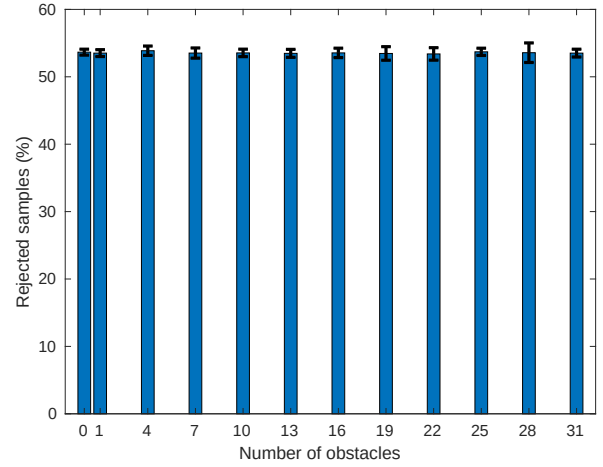


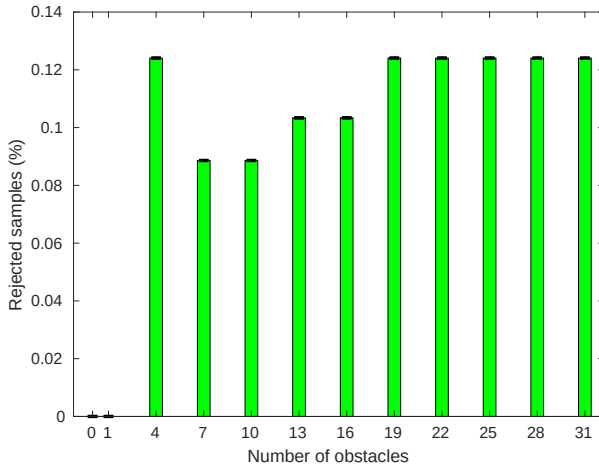
Figure 4.44: Case 4: Percent path length deviation computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



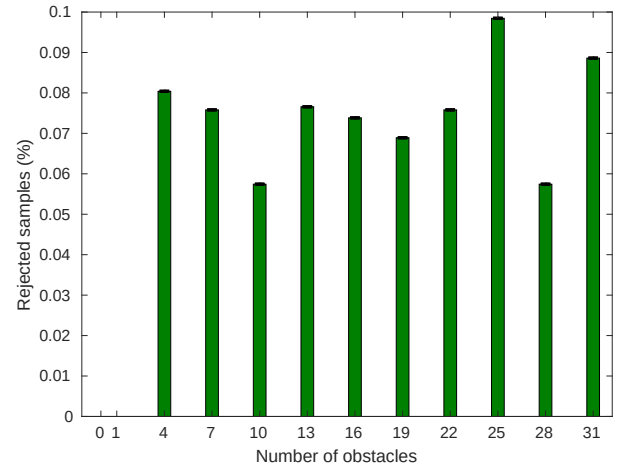
(a) PRM



(b) RRT

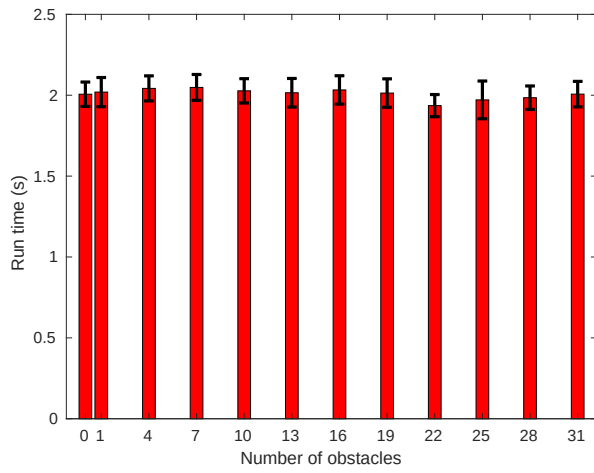


(c) CFDMPP

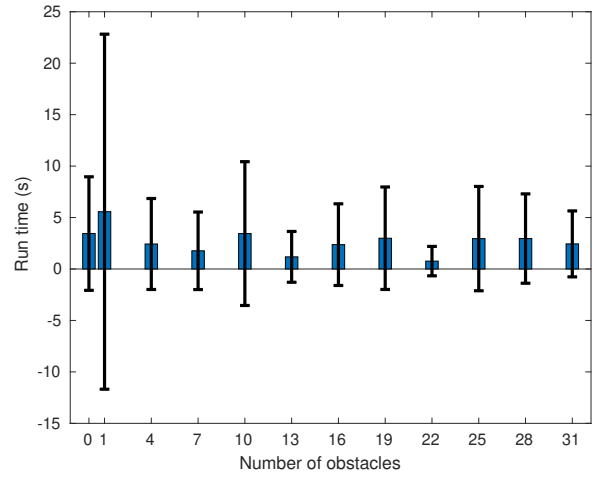


(d) CFDMPP with blocking

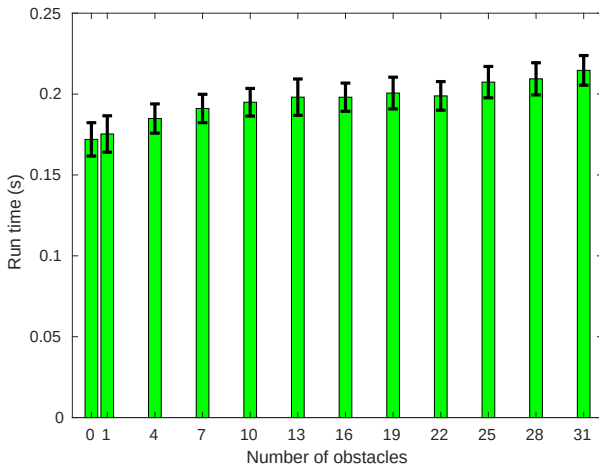
Figure 4.45: Case 4: Percentage of rejected samples computed across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.



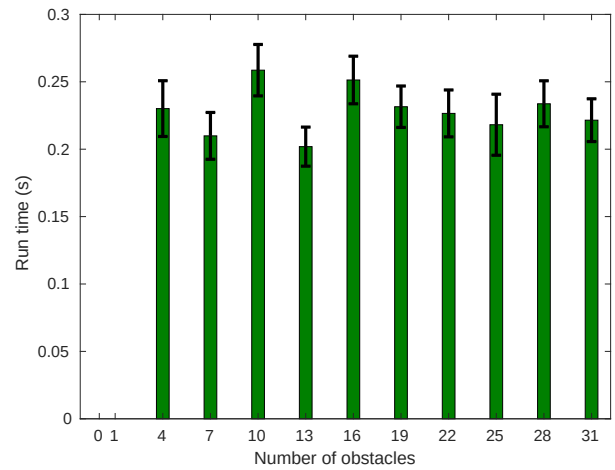
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

Figure 4.46: Case 4: Run time across 20 runs each for 0–31 obstacles/poles. Note that the y-axes span different ranges for each algorithm.

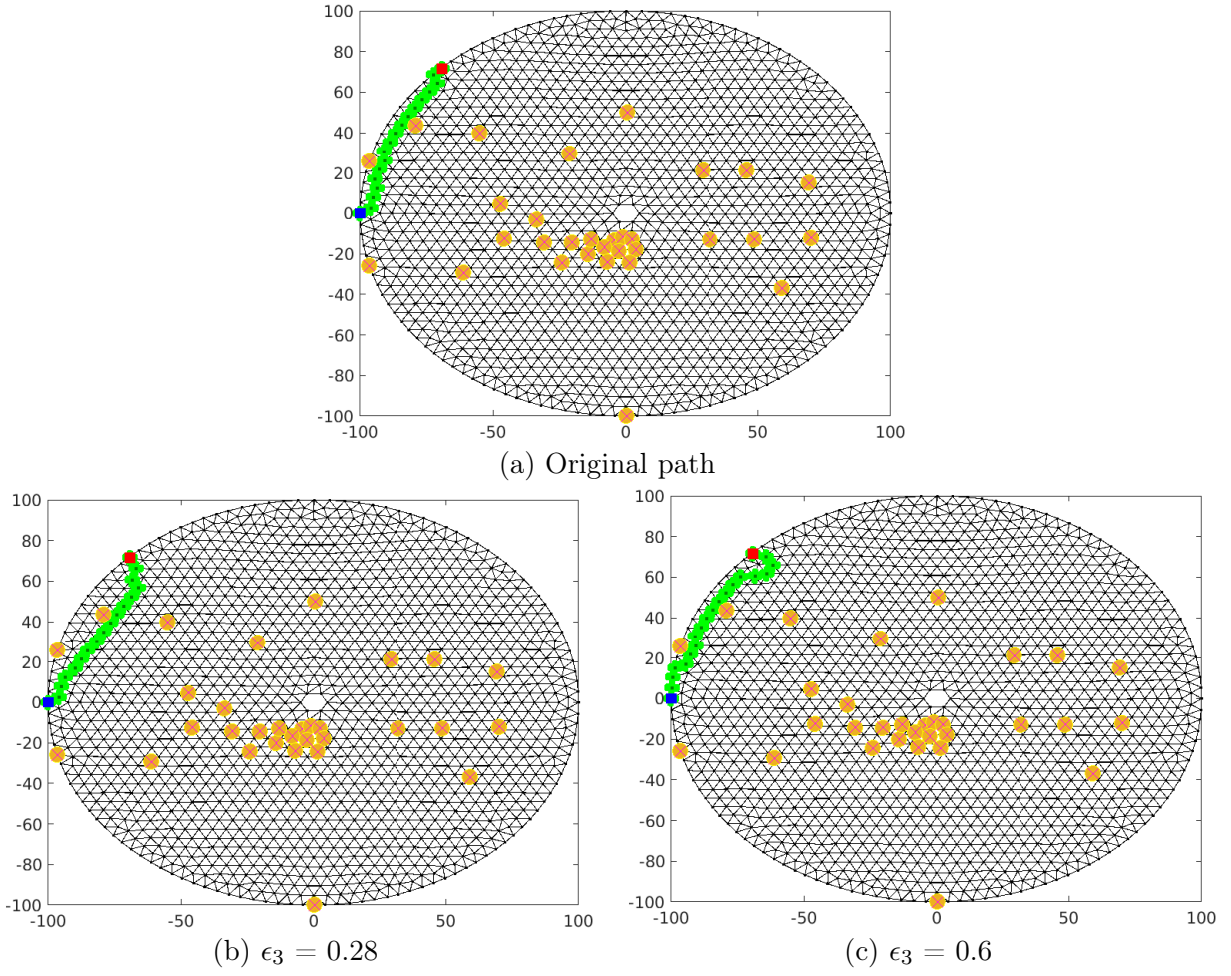


Figure 4.47: Alternative paths produced by CFDMPP on the CFDM

4.3 Stage 3: Case Studies From Simulations in a Warehouse Environment

4.3.1 Setup

In this stage, we consider the example of a real and typical, large warehouse. The aim of this stage is to demonstrate the performance of the proposed **CFDMPP** in warehouse environment replete with movable, multi-level shelving units and/or crates. The Costco warehouse is chosen as a simulation environment, since all its aisle based shelving units are fundamentally movable excepting any powered storage space such as refrigeration units. The aisle space allows the operation of vehicles that allow loading and off-loading of merchandise from shelving units, in conjunction with human activity. Therefore, this example is the closest replica of the intended target environment to demonstrate the functionality of the **CFDMPP**.

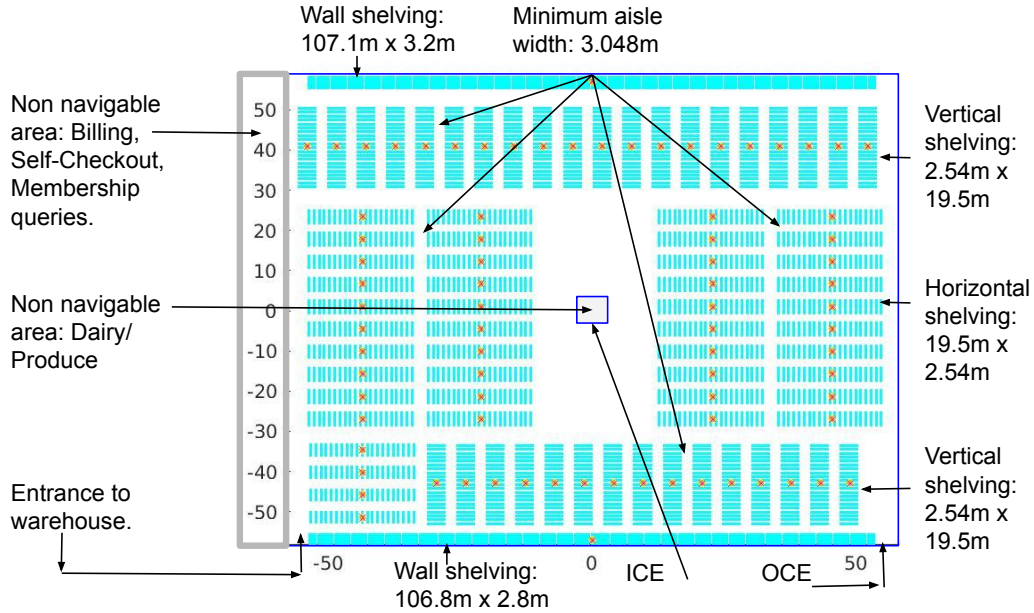


Figure 4.48: Simulation environment based on the area of an average Costco Warehouse

The average Costco warehouse spans an area of 146,000 square feet or 13564 square metres¹. Therefore, the dimensions of the environment are based on this average area. The environment is seen in Figure 4.48, which is not drawn to scale. The singular **PPS** = $\mathcal{W}$ = **PPPS** has a dimension of $d^{\text{PPPS}} = 2$. The **PPPS** i.e. the warehouse based on the average area is approximated to take the following dimensions:

$$\text{PPS}_{\text{chosen}} = \{(x, y) \mid (x, y) \in (-62.23 \text{ m}, 58.23 \text{ m}) \times (-58.23 \text{ m}, 58.23 \text{ m})\} \quad (4.10)$$

Devoid of mappings to alternative **PPSs**, the **SCs** are considered to be non-movable shelving units and areas inaccessible/undesirable for warehouse robot traversal, such as the areas with

¹<https://investor.costco.com/corporate-profile-2>

no shelving units. Therefore, as seen in Figure 4.48, the two non-navigable areas are that of billing/checkout passage and the dairy/produce section, and will be modelled as no-go zones, with no additional clearance. Then:

$$n_{NG} = 1 + 1 \quad \Rightarrow \quad n_{SC} = 2$$

$$\begin{aligned} NG_1 &= \{(x, y) \mid (x, y) \in [-3.5361 \text{ m}, 3.5361 \text{ m}] \times [-3.5361 \text{ m}, 3.5361 \text{ m}]\} \\ NG_2 &= \{(x, y) \mid (x, y) \in [-62.23 \text{ m}, -58.23 \text{ m}] \times [-58.23 \text{ m}, 58.23 \text{ m}]\} \end{aligned}$$

Applying Algorithm 3.2 with a $\Delta_{CS} = 0.5 \text{ m}$ leads to two DCSs in PPS_{chosen} which are the same as the actual constraints, i.e. $DCS_1 = NG_1$ and $DCS_2 = NG_2$. The PPS becomes the PPS_{chosen} . Since PPS_{chosen} is already completely defined, no further processing is necessary. The topological representation of PPS_{chosen} is defined using the same sequence of steps as seen in Section 4.1 and initially in Section 3.2. The constraint-free topological space is obtained using (4.7):

$$\mathcal{S}_{PPS_{chosen}}^{free} = \{\mathbf{p} = (x, y) \mid r_1 < \|\mathbf{p}\|_{\infty} < r_2\} \text{ for } r_1 = 3.5361 \text{ and } r_2 = 58.23 \quad (4.11)$$

Using the same metrics and set of homeomorphisms as seen in Section 4.1 and Section 4.2, the CFM is obtained. The R-1 representation of the constraint-free region obtained in (4.11) may be transformed to a CFM as R-4. The radii used in the homeomorphisms are $r_i = r_1 \times \|(1, 1)\|_2 = 1.414r_1 = 5 \text{ m}$, and $r_o = 58.23 \text{ m}$. The resulting CFM is:

$$R-4 = \{\mathbf{p} = (x, y) \mid r_i < \|\mathbf{p}\|_2 < r_o\} \quad r_i = 5 \text{ m} \quad r_o = 58.23 \text{ m} \quad (4.12)$$

When discretizing this environment, we choose a discretization resolution (and other tuning parameters) with the following understanding. The fundamental layout of the warehouse remains constant, i.e. the locations of walls, structural columns and pillars. This layout has a high probability of remaining constant for several years. Therefore, the discretization of the CFM representing it, should ideally have to be generated only once. That being considered, the knowledge of the CFM must be maximized. So every part of the CFM must be equally represented and the resolution should ideally be as high as possible. So we choose meshing parameters in DistMesh that produces a uniform mesh with 48,831 nodes, resulting in a resolution of approximately 0.1092 square metres per triangle in the CFDM.

The locations of aisle based shelving units are taken into account after the construction of the CFDM, because they may be subject to change before then. Therefore, they are treated as the first group of DCs, represented as poles. In this example, there is a total of 81 aisle shelving units as seen in Figure 4.48. Their dimensions and aisle widths are based on actual Costco warehouse crate, shelf and aisle widths; with some minor modifications to allow for clearance and placement in the simulation environment. Assuming a centralized sensor system similar to Section 4.2, these 81 units are represented using point cloud (x,y locations) data set and a representative location on each data set as a pole. In this case, the centroid of the point cloud is chosen as the pole. To accommodate the UGV, a clearance of approximately 0.67 m (based on Clearpath Robotics' Husky dimensions) is added to point cloud representations of the 81 shelving units. In Figure 4.48, the point cloud representation is depicted in cyan, the poles as orange crosses, and the ICE and OCE using dark blue lines.

The parameters pertaining to mesh generation, inter-node computation and tuning are provided in Table 4.22. Note that ϵ_3 is not part of the table since it is not a fixed one-time value. It can be modified to signify how different a path needs to be from its blocked counterpart, and therefore is set by the user after at least one path is found. These values are seen as part of Section 4.3.2. We emphasize that the mesh generation time and inter-node computation times are considered a one-time cost. This is especially the case, when we realize that the fundamental layout of the warehouse (devoid of poles) does not change for a given period of time. Then, it is beneficial to tune parameters after CFDM generation and obtained knowledge of poles (or in anticipation, if information is unavailable), such that it is also a one-time activity. The aim of tuning is to ensure that the values are conducive to generating sufficient path classes that allow blocking. The parameters may be re-tuned for other benefits, such as higher tolerance for detours and an increase/decrease in total path classes. In our case, however, considering the realistic nature of the application, we would like to minimize the effort associated with tuning. Therefore our tuning parameters are optimized to yield one shortest path class, and additionally, at least one alternative path class. Thus, in the context of real environments, the CFDMPP capitalizes on known information as a one-time initialization cost.

Table 4.22: CFDMPP Parameters

Mesh generation time	Mean = 77.7134 s, Std = 0.3046 s
Inter-node computation time	Mean = 30.6849 s, Std = 1.3863 s
Mesh resolution in nodes	48831
ϵ_1	1.95
ϵ_2	1.95

4.3.2 Results and Analysis

This section demonstrates a total of five cases that depict typical situations encountered in a real-world warehouse environment. These situations require a UGV to traverse the CFDM for stocking, retrieving and surveillance purposes. They involve travelling between aisles in close proximity and across the span of the warehouse. Therefore, cases 1 to 5 aim to show such situations and the effectiveness of the CFDMPP in environments with such path planning needs. Each case shows the efficiency of using poles to depict the first group of DCs. Additionally, it also shows the utilization of homotopic blocking to account for the second group of DCs. Similar to Section 4.2, the results display the path in R-4 in conjunction with the equivalent path in R-1. The start and goal locations are marked with blue and red squares, respectively, while the path is marked in green. Unlike Sections 4.1 and 4.2, all five cases will be analyzed together in this section. We do this to analyze the implications of CFDMPP as a unified solution for warehouse robotics.

The results for each case may be referred to using Figure 4.49 for case 1, Figure 4.50 for case 2, Figure 4.51 for case 3, Figure 4.52 for case 4 and Figure 4.53 for case 5. Cases

2, 4 and 5 depict different variants under the stocking and/or retrieving category; whereas case 3 is a clear example of a surveillance route across the length of the warehouse. Case 1 may be considered as a short range surveillance route and/or stocking/retrieval task route. In all cases, the path initially generated is seen first, followed by alternative paths. So Figure 4.49(a), Figure 4.50(a), Figure 4.51(a), Figure 4.52(a) and Figure 4.53(a) show paths as initially generated by the CFDMPP prior to blocking, for cases 1, 2, 3, 4 and 5, respectively.

By observing the original paths produced across all five cases, one can see an emerging pattern that differentiate cases 1 and 3, from cases 2, 4 and 5. The former two cases are required to produce paths between start and goal locations that are placed farther apart than those in the latter three cases. Let us observe the paths seen in Figure 4.49(a), Figure 4.50(a), Figure 4.51(a), Figure 4.52(a) and Figure 4.53(a), purely in the context of path length or visually directness of paths. It can be seen that the original, unblocked paths resulting in cases 2, 4 and 5 appear to be more direct than those in cases 1 and 3. That is, there exist more deviations from the ideal shortest path in cases 1 and 3, whereas in cases 2, 4 and 5, it is easily seen that there are no extraneous deviations in the paths. Recall that cases 1 and 3 require path planning over a longer range, whereas the remaining cases are for shorter distances.

This behaviour is important to notice, because it visually depicts the inherent relationship between the kinds of paths produced and the tuning parameters chosen. The meshing resolution impacts the number of possible directions of paths to explore for a given point or node in the discretized case. Naturally, with a higher resolution, it is a more exhaustive search; so it leads to a better solution. Next, the parameters ϵ_1 and ϵ_2 , impose a restriction that curbs the amount of detours that a path is allowed to take. However, note that the restriction is imposed incrementally for every addition of a newly encountered node to a path (sequence of nodes). That is, the restrictions are acting on incrementally built paths, before finally reaching the goal. As users, we set the tuning values in anticipation of some poles/detours, but not all. Over long range distances, more poles may be encountered, for which the tuning values may no longer be ideal to produce intuitively short paths with lesser detours. This is precisely what we notice in the paths of cases 1 and 3, and cases 2, 4 and 5. The tuning parameters are ideally suited for the latter, and thus the paths produced appear to be paths with minimal detours, barring those caused by discretization resolution.

Arguably, different tuning parameters may be chosen. But, recall from Section 4.3, that the choice of tuning parameters justifies three important requirements. First, the parameters require to be a one-time procedure corresponding to the map, such that even with a change in poles, at least one path class is produced. Second, for a given number of poles, they must allow different scenarios (long and short range) to produce one fundamental path class. Finally and most importantly, the tuning parameters must allow the generation of at least one alternative path class. Clearly, all five cases produce a path in the shortest path class and additional path classes. Cases 1 and 3 can be seen to produce two alternative path classes: Figure 4.49(e) and Figure 4.49(c) for case 1 and Figure 4.51(e) and Figure 4.51(c) for case 3. On the other hand, cases 2, 4 and 5 are seen to produce one alternative path class each in Figure 4.50(c), Figure 4.52(c) and Figure 4.53(c) for cases 2, 4 and 5, respectively.

So clearly, the tuning parameters chosen achieve the requirements, but there exists a

trade-off. By allowing the parameters to work well for the above requirements, there exists less of a restriction for extraneous detours as part of longer paths. That is, the parameters are not sensitive to detours that are small with respect to overall path lengths, if the paths are long. Contrarily, for shorter path segments as seen in cases 2,4 and 5, the tuning values are well suited to produce paths with minimal detours. While cases 1 and 3 produce paths with an increased number of detours, it is interesting to note that they more easily produced two alternative path classes, as opposed to just one, as seen in cases 2, 4 and 5. So, the number of alternative path classes is also affected in the trade-off. Of course, the number of path classes produced via homotopic blocking now needs to be viewed in light of yet another parameter, ϵ_3 .

The parameters ϵ_1 and ϵ_2 together give rise to path classes and the separation amongst them, but ϵ_3 is how we as users, choose an alternative path class. For example, in case 1, an ϵ_3 value of 0.475 induces the next physically available path class as compared to the original path, as seen in Figure 4.49(e). When the ϵ_3 value is increased to 0.5, a more drastic change in the path is sought, which is what is obtained in Figure 4.49(c). For the given start and goal locations, i.e. long range path planning, ϵ_3 was more easily tunable to obtain alternative path classes. A similar analogy can be seen for case 3 in Figure 4.51(e) and Figure 4.51(c), where an increase in ϵ_3 results in path classes that are increasingly different from the original path class. However, for cases 2, 4 and 5, while it was easy to tune ϵ_3 to produce at least one alternative path class, tuning for additional path classes proved challenging.

Therefore, the take-away from this analysis can be summarized as follows. Tuning of parameters for realistic path planning situations must be versatile to allow the generation of path classes for long and short range paths, as many poles as possible and at least one additional path class via homotopic blocking. Such versatility comes with trade-offs that have a low tolerance for extraneous detours over short range paths. Contrarily, they have a high tolerance for extraneous detours over long range paths, but are more easily tuned for alternative path classes. In the context of warehouse robotics, it is recommended that parameters be tuned for versatility, since alternative path classes is an effective method of avoiding the second group of DCs.

Several situations can trigger a need to generate an alternative path between the same start and goal locations. In cases 2 and 4, the trigger may be that of fallen merchandise or fluid leaks that now obstruct the originally generated path or render it unsafe for the UGV. In cases 1, 3 and 5, the user may require an alternative view of the same area for surveillance, stock monitoring or anticipation of human activity or danger. These situations are such that the data suggesting obstruction may only be available in some cases; anticipatory situations and cases such as water/fluid leaks on the floor may not be easily quantified as sensed data for path planning. Homotopic blocking is a tremendous tool in such a case, because it does not rely on a change in the map to trigger a path re-planning. It only relies on path classes based on the CFDM and tuning parameters chosen.

One can notice in the figures corresponding to all five cases, that no explicit obstacles are placed in lieu of the original, blocked paths. In other algorithms, only the explicit presence of data suggesting the presence of constraints can trigger path re-planning. In the CFDMPP however, path re-planning is still triggered because of the use of homotopy classes. Since

each path class is now succinctly represented by the class marker value, a complex number, path re-planning is triggered simply by specifying a threshold to stay away from that path class. And thus, the essence of avoiding certain regions of the navigable PPS is now reduced to choosing the proximity to this path class. So, instead of re-constructing the map with constraints on the blocked path region, one simply specifies a value. Real time information can still be processed to indicate the presence of constraints, however, in situations when it is not explicitly possible to describe it, one should not have to forcefully augment the environment with data for that purpose. With the CFDMPP, no such augmentation is necessary in those cases, and the fundamental map, i.e. the CFDM always remains the same. Therefore, this is beneficial to the overall cost of the path planning strategy.

SAs unfortunately, cannot handle such cases. We now look at how the PRM and RRT fare in simulated blocking situations. In order to simulate that, both algorithms need to be run to produce initial paths. In the case of the PRM, since the map is represented by an occupancy grid, adding clearance from shelving units prohibited the availability of free space in between shelving units. This relates to a entire grid cell being available for navigation and is a function of grid resolution. Therefore, the map was modified to no longer carry the same clearance from shelving units, to allow the algorithm to function. The situation simulated is the same as case 2 from the CFDMPP results in this section, as seen in Figure 4.50. The first path produced by the PRM in R-4 is seen in Figure 4.54(a), and its equivalent path in R-1 (zoomed in) is seen in Figure 4.54(b). The cyan coloured shelving units appear skewed because of the transformation of their equivalent occupancy grid indices from R-4 to R-1. Next, we attempt to simulate blocking.

One can truly never simulate blocking in SAs as we did in CFDMPP because of the following reasons. SAs regenerate their map for every run, via sampling. So, given a specific start and goal pair, if the algorithm is queried multiple times, different paths may be obtained. The difference may be in length and its winding around certain constraints. Blocking is used to avoid a specific kind of path, i.e., a path with a certain winding characteristic. So when the map is augmented with a simulated constraint data set in place of the original path obtained, the SAs recreate the map with additional constraints, and then return a path, which we will consider as an alternative path. Figure 4.54(c) shows the effect of adding constraints in place of the original path, and the resulting path. The R-1 equivalent of the newly generated path is shown magnified in Figure 4.54(d).

A similar exercise is attempted with the RRT, where the original path generated in R-4 is seen in Figure 4.55(a) and a magnified equivalent of the path is seen in Figure 4.55(a). The path produced via simulated blocking is seen in Figure 4.55(c) and its equivalent R-1 version in Figure 4.55(d) in magnified form. Observing the final paths in R-1 for both PRM and RRT shows that the former experiences less deviations as compared to the latter. However, the CFDMPP produces a comparatively less abrupt path and with less detours.

One may notice that the paths appear to overlap constraint regions in R-4 but not in R-1. The reasons are as follows: The use of homeomorphisms between R-1 and R-4 allows the transformation of constraint-free spaces. So, if one were to experience constraint violation of a path in one space, it is applicable to the transformed space as well. This concept was successfully demonstrated thus far in the results of the CFDMPP, because the path was

formulated on a discretized map. That is the path was a composition of sequence of non-random points that reflected the nature of the underlying space. So when the transformation was applied, the lines connecting the points also obeyed the transformation, and in turn satisfied any constraint-avoidance conditions from the original space. With [RRT](#) and [PRM](#), the path transformation between two spaces is more challenging, since the generated points may be so far apart that the line segment connecting the points may be constraint free in one space but fail in the transformed space. Ideally of course, the transformation between spaces should hold, but that can occur only if the line segments connecting the points are adequately small. For that to occur, [SAs](#) need to be tuned to generate samples that are spaced at a certain minimum distance from each other and uniformly around each other. In other words, the ideal solution requires that the map be discretized, which is what the [CFDMPP](#) makes use of.

The final observation is related to the long term response of the algorithm to different querying needs, specifically for the warehouse robotics context. For example, with the [CFDMPP](#) the same map may be used to query a path between a given start-goal pair multiple times and expect to yield the same result. Naturally, with the [PRM](#) and [RRT](#), their probabilistic nature yields different paths for the same query. While [SAs](#) are useful in unknown environments, they do not effectively capture and retain information about a known environment, since with every query, old information is discarded. This however is crucial for an application where the fundamental map remains unchanged and yet needs to account for anticipatory changes in paths. With the [CFDMPP](#), the discretized map in conjunction with the homeomorphic transformations and homotopic classes, allows for consistent definition of path classes. It is this combination that maximizes the knowledge of the [PPS](#) and provides consistent responses to the same query multiple times or multiple different queries.

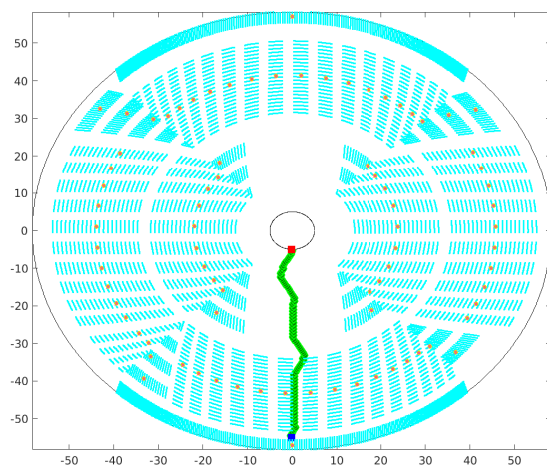
4.4 Conclusion

This chapter evaluated the performance of the proposed [CFDMPP](#) using three stages. Section 4.1 showed the benefits of performing path planning on the [CFDM](#). The [CFDM](#) was constraint-free, implying that paths did not need to be evaluated for constraint violation with constraints that were modelled out of the [CFM](#). This affords the path planner the luxury of operating on constraint edges, without any further evaluation of path quality. Section 4.2 deals with the situation when constraints that were not accounted for during the construction of the [CFM](#), are now encountered. The results of the [CFDMPP](#) are evaluated alongside those of the [PRM](#) and the [RRT](#), by use of metrics such as percentage of path length deviation, path length ratio, percentage of usable and rejected samples and, run time.

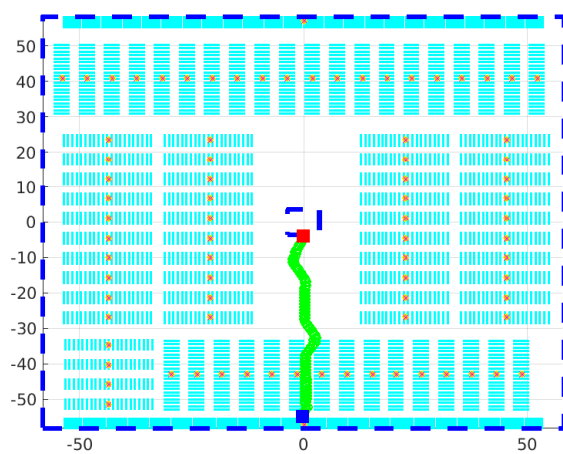
The main observation relates to the ability of the [PRM](#) and the [RRT](#) to generate paths reasonably quickly, but with a compromise in the number of usable samples and path lengths. Comparatively, the [CFDMPP](#) performed well even when the start and goal locations were close to each other and to dense clusters of obstacles, without compromising on usable samples, run time or path lengths. This is so, because the [CFDMPP](#) retains crucial map information from every run, unlike the [PRM](#) and [RRT](#), which regenerate the map every time they are queried. Since the [CFDMPP](#) retains the map in the form of the [CFDM](#), paths can

be designated and demarcated as unique path classes. This therefore, gives the [CFDMPP](#) the ability to generate several alternative paths, when the original path is either obstructed or simply becomes undesired. This feature of using path classes gives the [CFDMPP](#) the flexibility of querying for a new path even when there is no physical obstruction on the path. The [PRM](#) and [RRT](#) as is, do not have the ability to generate alternative paths and require the physical presence/data to simulate the querying of an alternative path.

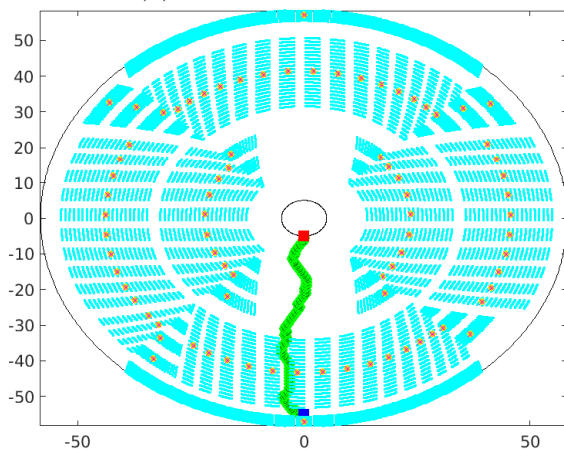
Section [4.3](#) shows the functionality of the [CFDMPP](#) for a realistic warehouse environment. The results highlighted important features related to the flexibility of the [CFDMPP](#). It was seen that the meshing parameter may be tuned for resolution depending on the environment, application and desired path coarseness. Additionally, path class related tuning parameters require appropriate initialization to trigger the generation of path classes. Meshing resolution and tuning parameters together, are seen to have an impact on the number of alternative path classes and the tolerance of a path class to detours experienced. Finally, when querying for an alternative path, it is seen that no explicit change in the map is required to be made to query this alternative path between the same start and goal locations. Therefore, the knowledge of the [CFM](#) remains intact.



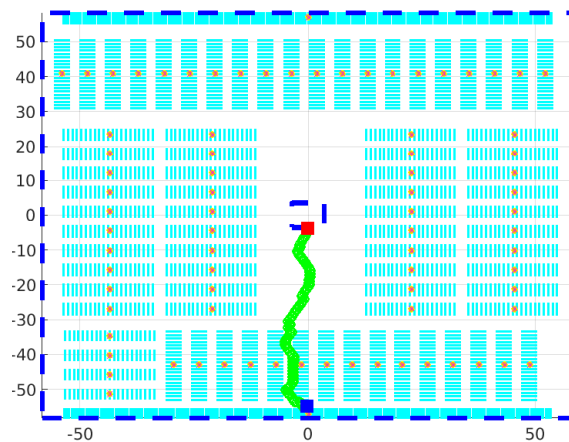
(a) Original path in R-4



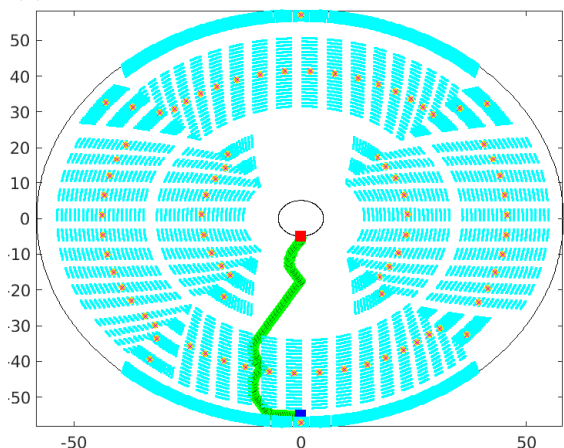
(b) Original path in R-1



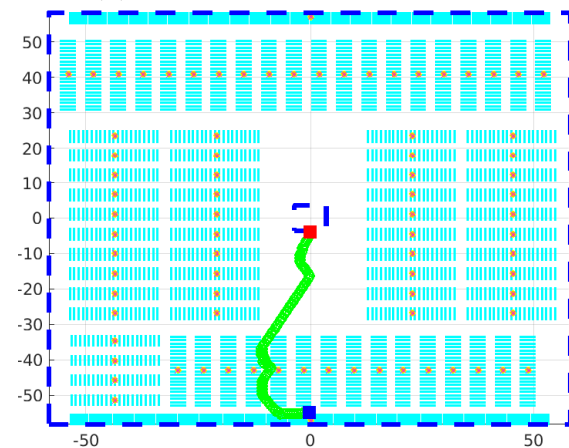
(c) First alternative path in R-4; $\epsilon_3 = 0.475$



(d) First alternative path in R-1

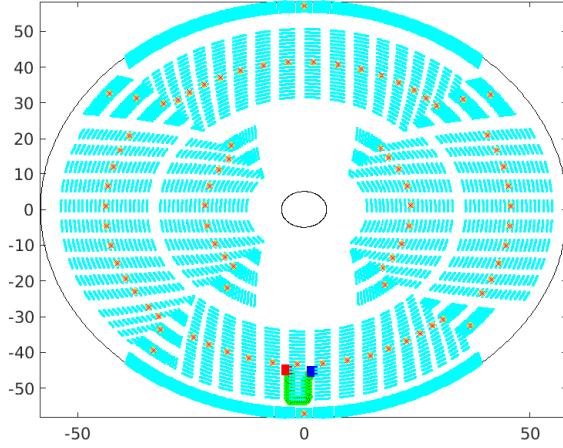


(e) Second alternative path in R-4; $\epsilon_3 = 0.5$

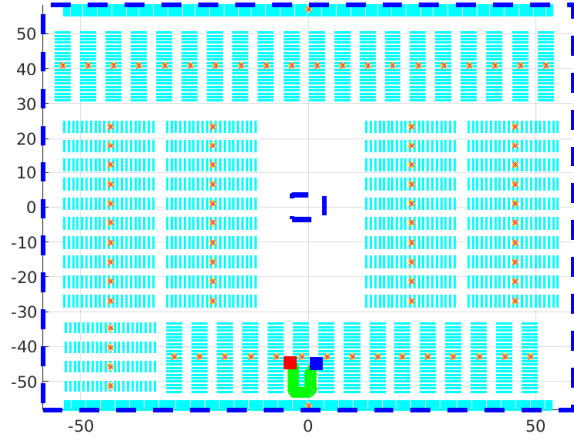


(f) Second alternative path in R-1

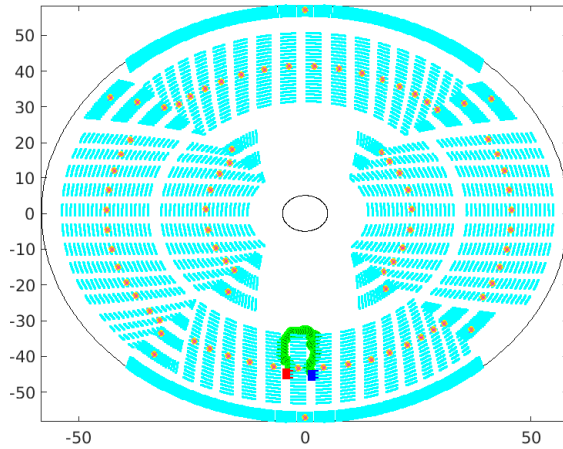
Figure 4.49: Case 1: CFDMPP with and without blocking in Costco style warehouse



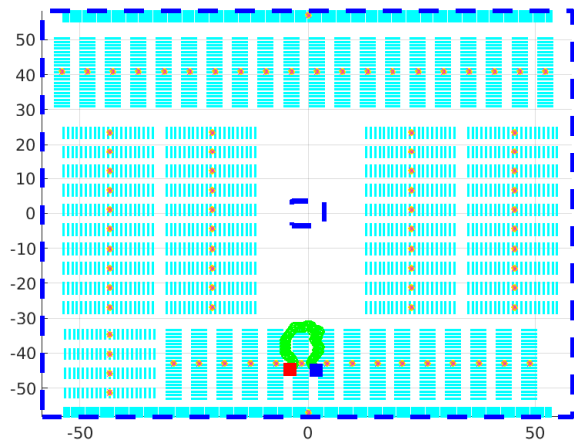
(a) Original path in R-4



(b) Original path in R-1

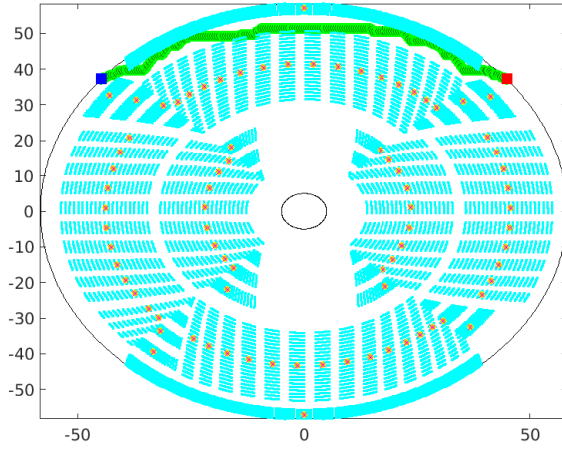


(c) First alternative path in R-4 $\epsilon_3 = 0.45$

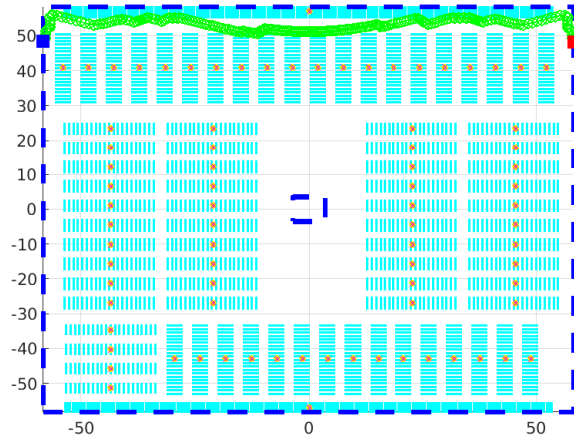


(d) First alternative path in R-1

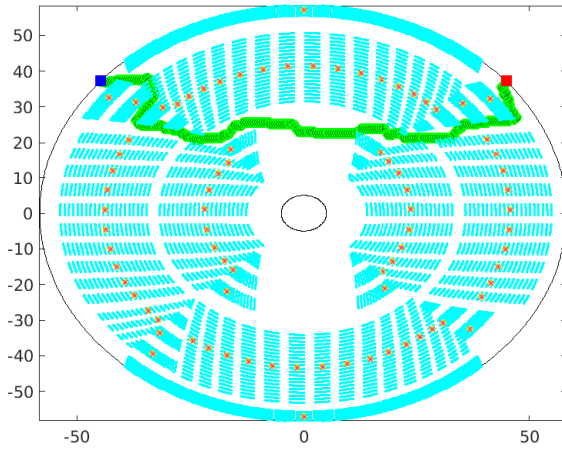
Figure 4.50: Case 2: CFDMPP with and without blocking in Costco style warehouse



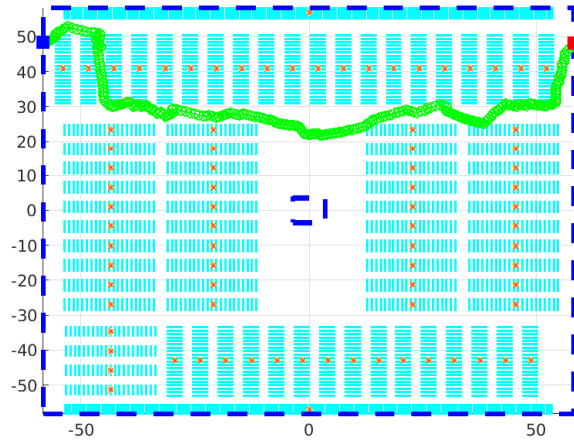
(a) Original path in R-4



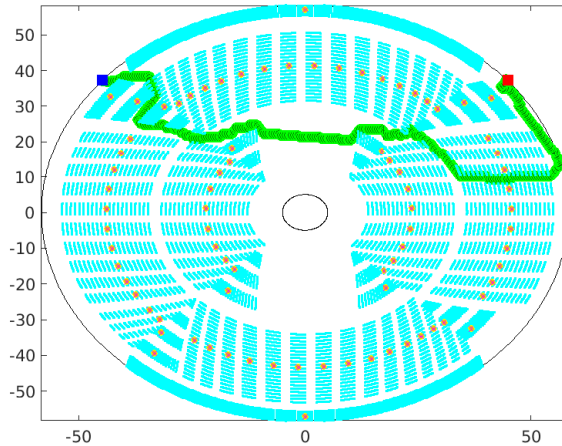
(b) Original path in R-1



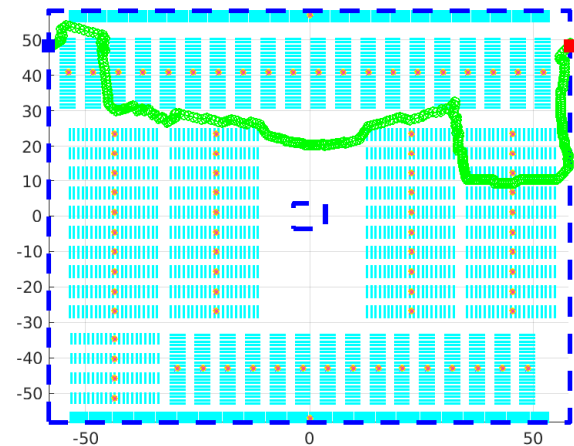
(c) First alternative path in R-4 $\epsilon_3 = 0.8$



(d) First alternative path in R-1



(e) Second alternative path in R-4 $\epsilon_3 = 1$



(f) Second alternative path in R-1

Figure 4.51: Case 3: CFDMPP with and without blocking in Costco style warehouse

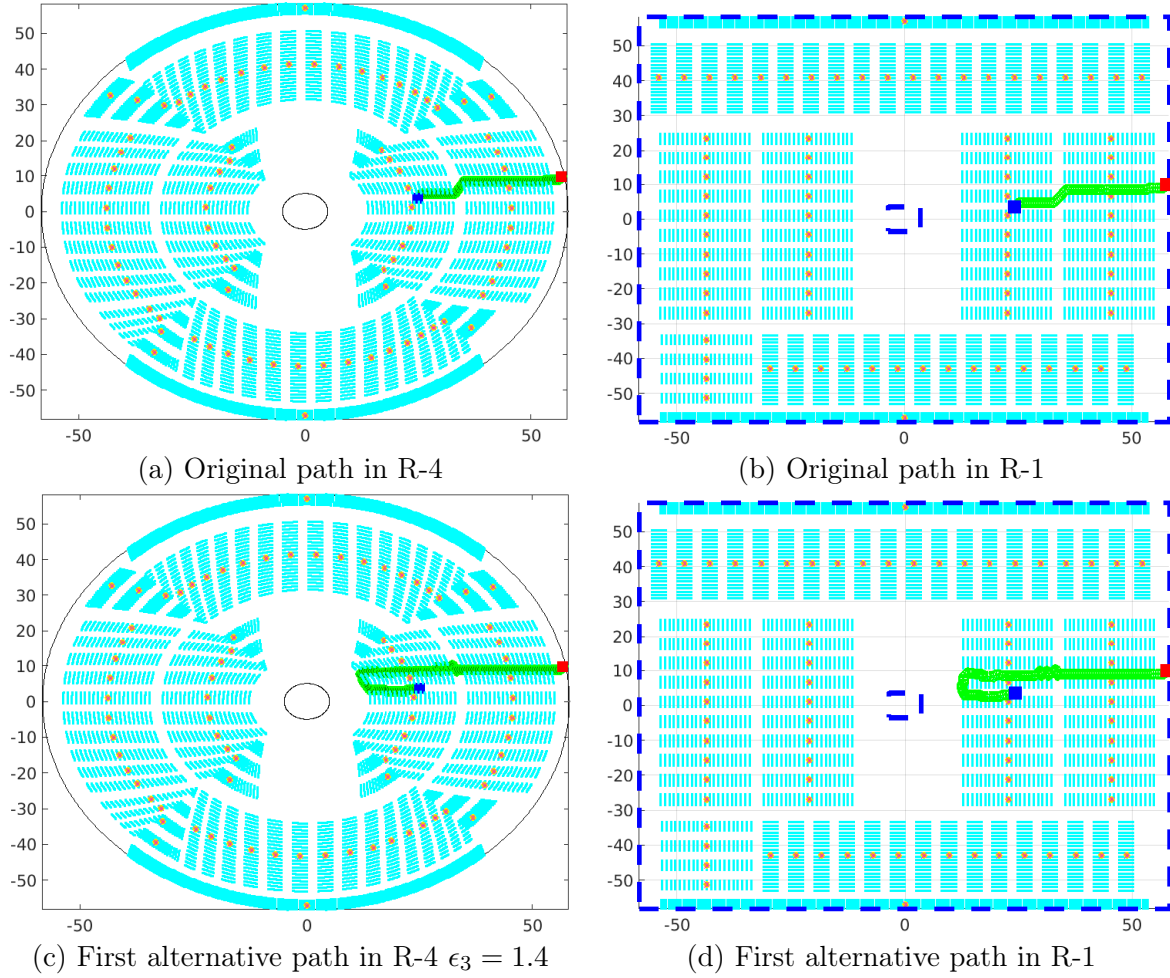
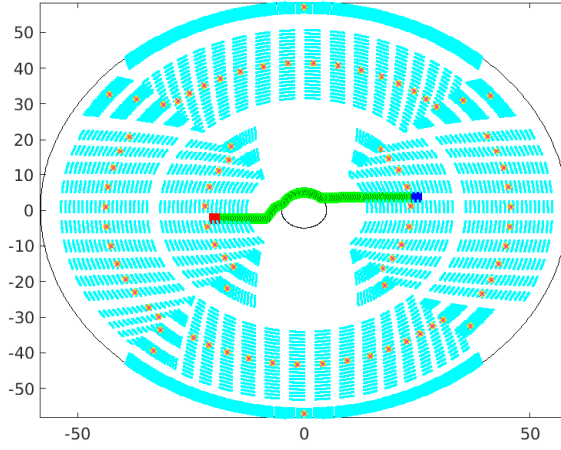
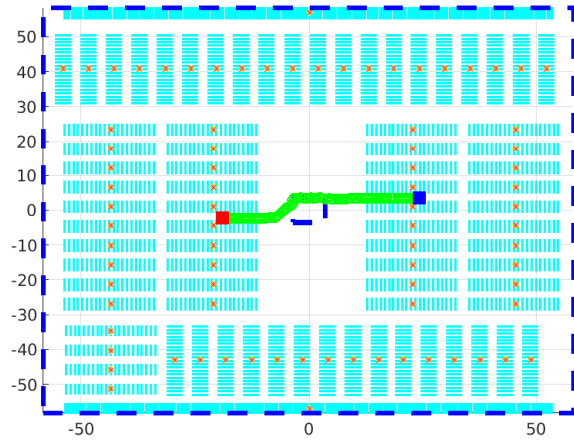


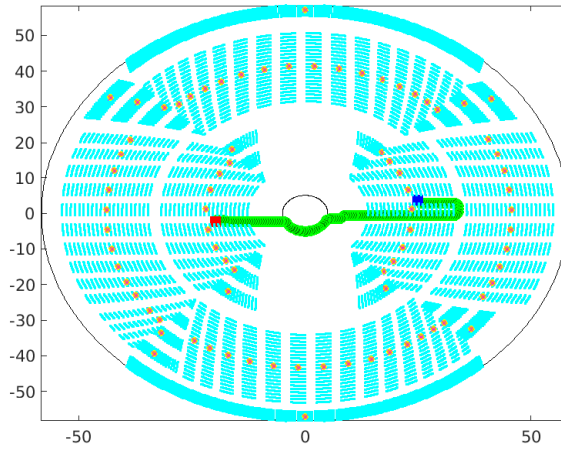
Figure 4.52: Case 4: CFDMPP with and without blocking in Costco style warehouse



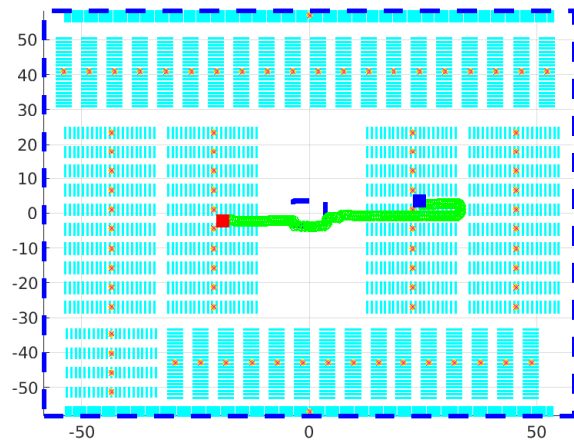
(a) Original path in R-4



(b) Original path in R-1

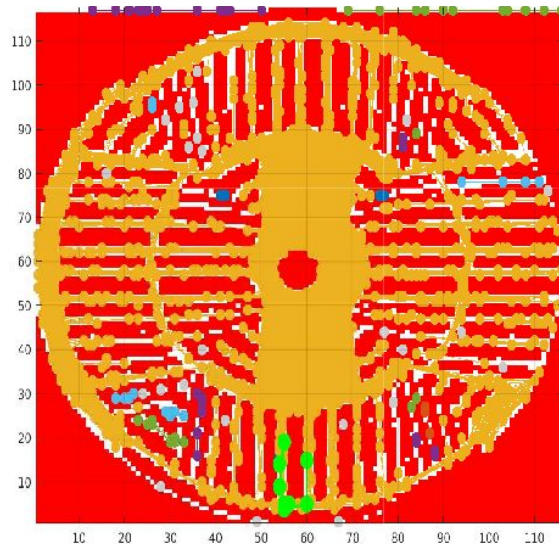


(c) First alternative path in R-4 $\epsilon_3 = 0.45$

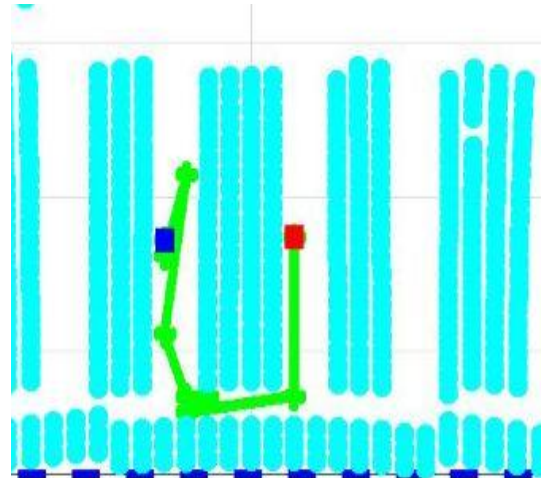


(d) First alternative path in R-1

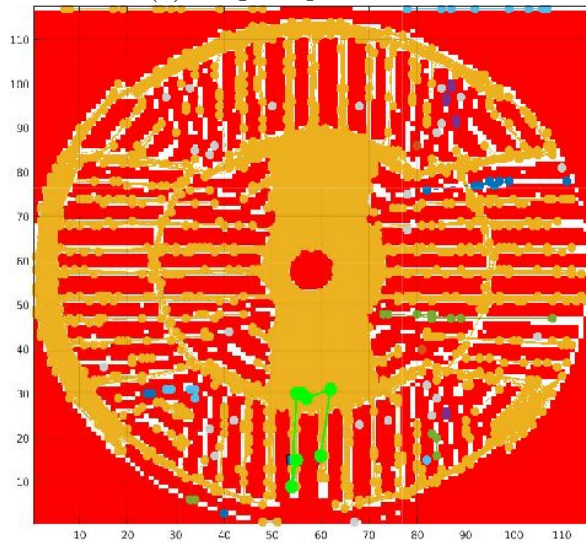
Figure 4.53: Case 5: CFDMPP with and without blocking in Costco style warehouse



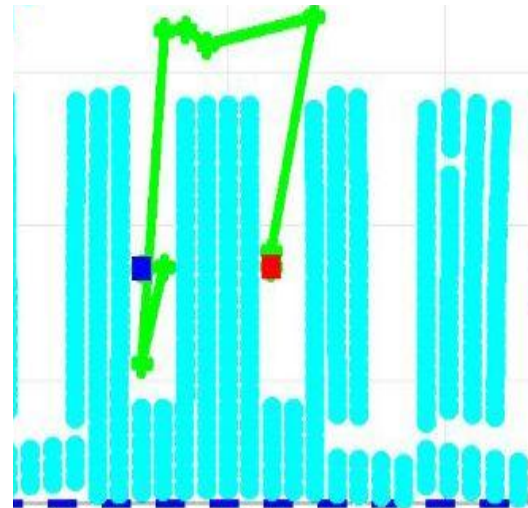
(a) Original path in R-4



(b) Original path in R-1

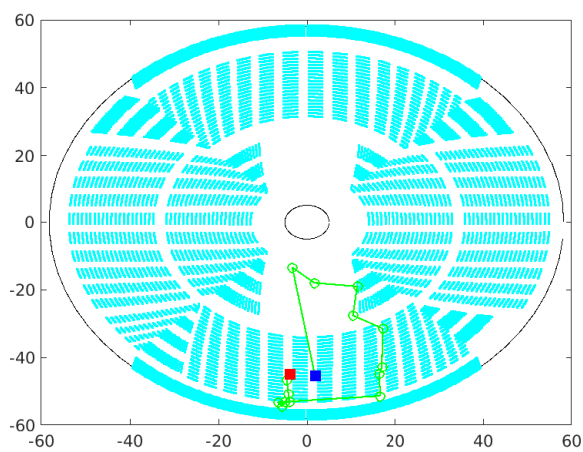


(c) Original path in R-4

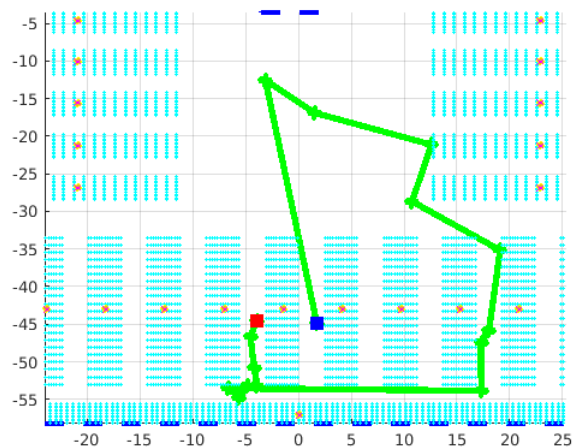


(d) Original path in R-1

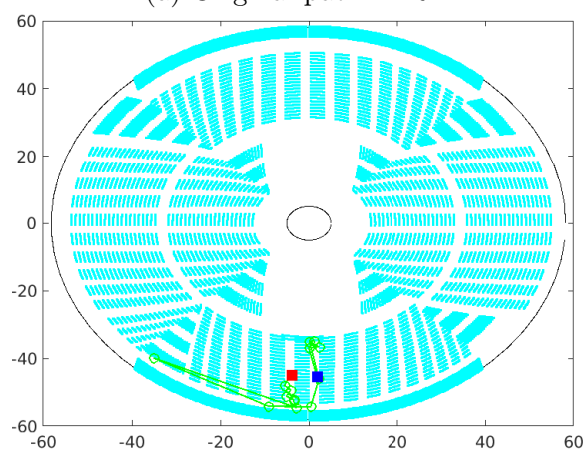
Figure 4.54: Case 1: PRM with and without blocking in Costco style warehouse



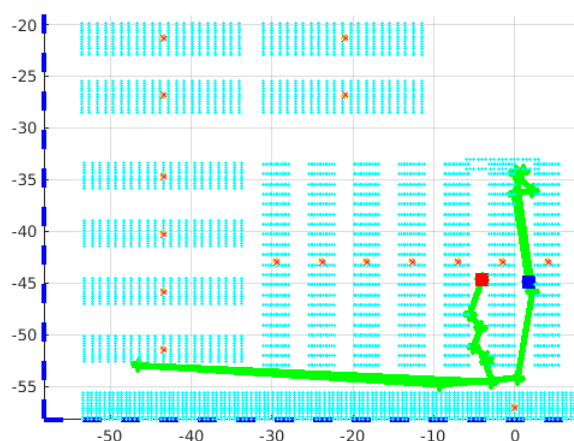
(a) Original path in R-4



(b) Original path in R-1



(c) Original path in R-4



(d) Original path in R-1

Figure 4.55: Case 1: RRT with and without blocking in Costco style warehouse

Chapter 5

Conclusion and Future work

The objectives of this thesis are to design a path planning algorithm that accomplishes the following goals: First, it avoids the explicit periodic verification of constraint violation for static constraints which are known during the map (manifold) construction. Second, the resulting paths should be free of misinterpretation, such as those caused by local minima. Finally, should a path be rendered invalid due to unforeseen constraints or deliberate choice, an alternative path must be returned with a minimal modification of the map.

The performance of the proposed algorithm (**CFDMPP**) was evaluated through the following perspectives: proof of concept, comparison with other commonly used algorithms for path planning, such as the **PRM** and **RRT**, and as a path planner for warehouse robotics. The outcome of the initial phase of research conducted was published in [137, 139]. Now, we discuss the advantages of the algorithm, followed by attributes that pose as challenges. Then, we explore a few potential avenues of future work that aim to solve those challenges.

The **CFDMPP** presents the following advantages: First, it utilizes knowledge of all known attributes of the environment to formulate the **CFM**. By maximizing knowledge of constant regions of the map, such as immovable **SCs**, the resulting **CFM** encompasses all knowledge of the constraint-free navigable space. So, when no further change in the map or constraints is encountered, the **CFM** is a complete representation of the path planning space. This creation of the **CFM** is a one-time cost, as opposed to other path planning algorithms that recreate the map every time a path planning query is made. The second advantage is a direct consequence of operating on the **CFM** under the aforementioned conditions and it is as follows: The **CFDMPP** avoids explicit verification of violation with constraints that are modelled out of the **CFM**. This not only saves resources during path planning, but also affords the luxury of operating along constraint edges without mandatory collision verification. Third, by modelling the path planning space as a manifold, properties of continuity, transformation across input/output spaces and smoothness of paths are obtained.

The fourth advantage lies in the discretization of the **CFDM**. This allows for a discrete representation of the environment, which makes the manifold accessible for path planning. As a direct consequence of discretization, the next advantage is the flexibility that comes with choosing a mesh resolution. We showed (in Chapter 4) that mesh resolution is an important attribute that can be used to generate a coarse or fine resolution of the map, and in turn

of the path. In fact, fine and coarse resolutions play the role of an increased or decreased number of nodes to represent a specific region of the map, respectively. Thus, the ability to choose meshing resolution is essential, especially when it is map/application dependent. The sixth advantage pertains to the paths produced by the [CFDMPP](#). Apart from returning the shortest constraint-free paths on the [CFDM](#), the [CFDMPP](#) uses homology and homotopic class-based path classification, avoidance and regeneration. This feature of generating an alternative path on the [CFDM](#) is one of the prime advantages of the [CFDMPP](#). The use of path classes ensures that certain paths, or [DCs](#) on a path, can be avoided by generating an alternative path, without modifying the map. That is, resources are not expended in an effort to populate the map based on constraint information whose presence may be fleeting. The presence of non-constant constraints by itself is of no consequence. They matter only when a path has been rendered invalid because of them. And it is only this path validity related information that the [CFDMPP](#) stores and utilizes for path re-planning. The final advantage is the tunable nature of parameters in the [CFDMPP](#). The parameters may be tuned for different path planning outcomes, such as the number of alternative path classes and tolerance to detours in paths. Naturally, such flexibility is of use for large environments, where one outcome may be preferred over the other. Even with such variance in tuning parameters, note that the resulting paths do not suffer from local minima. All paths produced are viable constraint-free paths.

The advantages are seen via the results of testing the [CFDMPP](#) through simulations, as seen in the three stages of Chapter 4. The first stage establishes the advantage of using manifolds to represent the path planning space as a [CFDM](#). The [CFDMPP](#) is tested on the [CFDM](#) devoid of any constraints by using four different start-goal locations, each focusing on an aspect of navigating on the [CFDM](#). Performing path planning on the [CFDM](#) showed that constraint edges can be traversed without any constraint violation checks for constraints that are modelled out of the [CFDM](#). Continuous transformations of paths between path planning spaces of interest were also demonstrated. The second stage focused on a comparison of the [CFDMPP](#) with [SAs](#) such as the [PRM](#) and [RRT](#) in an environment (200 m by 200 m) with the presence of point constraints. Each algorithm was tested for four different start-goal locations, i.e., four cases depicting long, medium and short range path queries. Furthermore, each case is tested for a varying number of constraints ranging from 0 to 31 in increments of 3.

All three algorithms are evaluated with objective metrics, such as path length ratio, path length deviation, percentage of rejected and usable samples, and run time. The [CFDMPP](#) is seen to produce paths on the manifold with an average percentage path length deviation of 29.6%, which is over 70% less than those produced by [SAs](#). The [CFDMPP](#) also provides an increase in retention of usable samples by a margin of at least 30%, when compared with [SAs](#). This is while maintaining on-par run times at worst, and better run times in most cases, when evaluated against other algorithms. For the metrics of map re-usability and generation of alternative paths on the [CFDM](#), the [CFDMPP](#) demonstrates a clear advantage over the [SAs](#), since the latter are unable to perform either task. The general trends related to metrics hold true even when the [CFDMPP](#) is utilized to generate alternative paths. Any deviation in path length related trend is seen only when a query is made to generate an alternative path that avoids the shortest path previously generated. The results show that the [CFDMPP](#)

may be queried for several alternative paths; a feature not present in general [SAs](#).

The third and final stage tests the [CFDMPP](#) in a simulated warehouse type of environment that is approximately modelled around a Costco warehouse. This environment assumes constraints to model the [CFDM](#) with, followed by a total of 81 non-point constraints such as shelving units, on the [CFDM](#). A total of five cases are used to analyze the [CFDMPP](#) as an integrated tool for warehouse robotics. The five cases are so chosen to denote path planning requirements such as pick and place, stock retrieval, autonomous cleaning, and surveillance and monitoring. Features such as mesh granularity, tuning of [CFDMPP](#) parameters and the impact of tuning on path lengths and number of alternative paths are discussed in detail. The results indicate an overall advantage of the [CFDMPP](#), which is that of map re-usability and representation of path classes. Additionally, the generation of alternative paths on the [CFDM](#) is possible when there exists no explicit presence of constraints. The results showed that some tuning parameters allow for a higher tolerance to detours in a path class and the total number of path classes produced. The five cases were instrumental in concluding that the choice of tuning parameters of the [CFDMPP](#) for warehouse robotics must satisfy the following: versatility of path planning range and at least one alternative path class.

Despite the wealth of advantages, [CFDMPP](#) does come with a few challenges. Note that the challenges presented here relate to aspects of the algorithm that are essential and inherent for the functioning of the algorithm. Features that may be added on to the algorithm but have not been implemented as part of this thesis, are not considered as challenges, as they constitute non-essential components.

Attributes of an algorithm that have tuning flexibility, also lend themselves to the challenges of tuning. The [CFDMPP](#) has two sets of tunable parameters: the first pertaining to path classes and the second being the choice of meshing resolution. Generally, tuning may be viewed as a disadvantage, because it usually requires knowledge and/or experience with the algorithm to determine appropriate values. The [CFDMPP](#) utilizes two parameters to initiate path planning and one parameter to produce alternative path classes. So at a minimum, two parameters must be tuned as part of the one-time cost of the algorithm. The third parameter that triggers the querying of an alternative path class also has a sensitivity that depends on the map at hand. As seen in Chapter 4, the implications of tuning parameters on the final results are multifaceted. Therefore, it is reasonable that tuning does require some understanding of the algorithm to avoid tedious effort. Similarly, meshing resolution of the [CFDM](#) is also user-defined and is specific to the application at hand. Since it inherently impacts the coarseness of the final path, one must make a deliberate choice of the meshing resolution. This requires knowledge of the map and the type of path planning application. We know that the [CFDMPP](#) generates the [CFDM](#) as a one-time cost, typically viewed as an overhead. However, it may be argued that this cost can be potentially high for environments that are larger or more complex than what was seen in Chapter 4. This is especially so, since the resolution also affects the number of path classes produced and the kinds of detours to which the resulting paths become tolerant. This case of tuning does indeed present a unique intertwine of meshing resolution and tuning parameters for path classes that requires deliberation before tuning.

5.1 Future Work

The advantages and challenges provide an overview of the factors of the [CFDMPP](#) that can serve as starting points for future work. In fact, some topics pertaining to potential future work include tackling some of the challenges. For example, one area of research may be focused on the optimal choice of tuning parameters. One can begin by using the analysis of parameter impact on the final results seen in Chapter 4 and an understanding of the algorithm from Chapter 3. Then, a comprehensive analysis on the relationship amongst tuning parameters, total number of path classes, proximity of the start-goal pair to poles/obstacles and the number of poles/obstacles can be performed. This work will enable the formation of a general heuristic/rubric for tuning parameters, and can potentially make the tuning process fully or at least semi systematic.

The next area of research relates the impact of meshing resolution on the resulting path classes. The area of focus would involve different kinds of meshing for different parts of the same [CFM](#). That is, some parts of the environment will be coarsely meshed as opposed to others with a finer resolution. Such a non-uniform mesh was seen in Chapter 4, but the improvement to that would involve incremental and region specific non-uniform meshing. For example, regions with potentially less path planning requirements can do with a coarse resolution, and in turn less number of path classes. There may exist situations where the converse is required. This difference in requirements on the same [CFM](#) is what triggers the segmented non-uniform mesh generation of the [CFDM](#). This effort is akin to biasing sampling from a specific place with more or less bias for path planning needs without the disadvantages of [SAs](#). The result would still offer the guarantee of path classes and a decrease in resources.

The final area of research is the exploration of using alternative manifold construction tools, such as connected sums. This would enable the definition of more complex manifolds with which to model the environment and on which the [CFDMPP](#) can be applied. One notes that this is a mathematically extensive endeavour. However, the theoretical foundations provided in Chapter 3 offer a necessary and intuitive starting point for such research, specifically in the context of robotic path planning.

References

- [1] M. A. Abbas, R. Milman, and J. M. Eklund. Obstacle avoidance in real time with non-linear model predictive control of autonomous vehicles. In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6, 2014.
- [2] Adnan Ademovic and Bakir Lacevic. Path planning for robotic manipulators using expanded bubbles of free c-space. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2016-, pages 77–82. IEEE, May 2016.
- [3] Mohammad Pourmahmood Aghababa. 3D Path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles. *Applied Ocean Research*, 38:48–62, 2012.
- [4] Josu Agirrebeitia, Rafael Avilés, Igor F de Bustos, and Goizalde Ajuria. A new APF strategy for path planning in environments with obstacles. *Mechanism and machine theory*, 40(6):645–658, 2005.
- [5] Hossein Akbaripour, Hossein Akbaripour, Ellips Masehian, and Ellips Masehian. Semi-lazy probabilistic roadmap: a parameter-tuned, resilient and robust path planning method for manipulator robots. *International Journal of Advanced Manufacturing Technology*, 89(5):1401–1430, 2017.
- [6] Javier Alonso-Mora, Jonathan A DeCastro, Vasumathi Raman, Daniela Rus, and Hadas Kress-Gazit. Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. *Autonomous Robots*, 42(4):801–824, 2018.
- [7] ASME. *Predictive Control of Autonomous Ground Vehicles With Obstacle Avoidance on Slippery Roads*, volume ASME 2010 Dynamic Systems and Control Conference, Volume 1 of *Dynamic Systems and Control Conference*, 09 2010.
- [8] Ahmad Ataka, Hak-Keung Lam, and Kaspar Althoefer. Reactive magnetic-field-inspired navigation method for robots in unknown convex 3-D environments. *IEEE Robotics and Automation Letters*, 3(4):3583–3590, 2018.
- [9] A. Atramentov and S. M. LaValle. Efficient nearest neighbor searching for motion planning. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 632–637 vol.1, 2002.

- [10] Jérôme Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, 2016.
- [11] F Belkhouche. Reactive path planning in a dynamic environment. *IEEE Transactions on Robotics*, 25(4):902–911, 2009.
- [12] F Belkhouche and B Bendjilali. Reactive path planning for 3-D autonomous vehicles. *IEEE Transactions on Control Systems Technology*, 20(1):249–256, 2012.
- [13] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.
- [14] Paolo Bevilacqua, Marco Frego, Daniele Fontanelli, and Luigi Palopoli. Reactive planning for assistive robots. *IEEE Robotics and Automation Letters*, 3(2):1276–1283, 2018.
- [15] S Bhattacharya, M Likhachev, and V Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, 2012.
- [16] Subhrajit Bhattacharya. Towards optimal path computation in a simplicial complex. *The International Journal of Robotics Research*, 38(8):981–1009, 2019.
- [17] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Multi-robot coverage and exploration on riemannian manifolds with boundaries. *The International Journal of Robotics Research*, 33(1):113–137, 2014.
- [18] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI’10, page 1230–1237. AAAI Press, 2010.
- [19] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Search-based path planning with homotopy class constraints in 3D. In *Invited paper for sub-area spotlights track on ‘Best-paper talks’, Proceedings of Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012.
- [20] Subhrajit Bhattacharya and Mihail Pivtoraiko. A classification of configuration spaces of planar robot arms for a continuous inverse kinematics problem. *Acta applicandae mathematicae*, 139(1):133–166, 2015.
- [21] Anthony Bloch, Margarida Camarinha, and Leonardo J Colombo. Dynamic interpolation for obstacle avoidance on riemannian manifolds. *International Journal of Control*, 94(3):588–600, 2021.
- [22] O Bohigas, M. E Henderson, L Ros, and J. M Porta. A singularity-free path planner for closed-chain manipulators. In *2012 IEEE International Conference on Robotics and Automation*, pages 2128–2134. IEEE, 2012.

- [23] Oriol Bohigas, Michael E Henderson, Lluís Ros, Montserrat Manubens, and Josep M Porta. Planning singularity-free paths on closed-chain manipulators. *IEEE Transactions on Robotics*, 29(4):888–898, August 2013.
- [24] R. Bohlin and L. E. Kavraki. Path planning using lazy PRM. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 521–528 vol.1, 2000.
- [25] Rodney A Brooks. Planning collision-free motions for pick-and-place operations. *The International Journal of Robotics Research*, 2(4):19–44, 1983.
- [26] Zbigniew Błaszczyk and José Gabriel Carrasquel-Vera. Topological complexity and efficiency of motion planning algorithms. *Revista Matemática Iberoamericana*, 34(4):1679–1684, 2018.
- [27] Sergio Cabello, Yuanxin Liu, Andrea Mantler, and Jack Snoeyink. Testing homotopy for paths in the plane. *Discrete & Computational Geometry*, 31(1):61–81, 2004.
- [28] Mylène Campana, Florent Lamiraud, and Jean-Paul Laumond. A gradient-based path optimization method for motion planning. *Advanced Robotics*, 30(17-18):1126–1144, 2016.
- [29] Global Affairs Canada. *Invest in Canada : Canada’s robotics industry*. [Government of Canada Publications] ; FR5-38/25-2016E-PDF. Global Affairs Canada, Ottawa [Ontario], 2016.
- [30] S Carpin and G Pillonetto. Motion planning using adaptive random walks. *IEEE Transactions on Robotics*, 21(1):129–136, 2005.
- [31] Keonyup Chu, Minchae Lee, and Myoungcho Sunwoo. Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1599–1616, 2012.
- [32] E. J Cockayne and G. W. C Hall. Plane motion of a particle subject to curvature constraints. *SIAM Journal on Control*, 13(1):197–220, 1975.
- [33] Peng Cui, Weisheng Yan, and Yintao Wang. Reactive path planning approach for docking robots in unknown environment. *Journal of Advanced Transportation*, 2017:1–11, 2017.
- [34] L. K. Dale and N. M. Amato. Probabilistic roadmaps-putting it all together. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1940–1947 vol.2, 2001.
- [35] A.K. Dash, I.-M. Chen, S.H. Yeo, and G. Yang. Singularity-free path planning of parallel manipulators using clustering algorithm and line geometry. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 761–766, 2003.

- [36] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path planning strategies for UAVs in 3D environments. *Journal of Intelligent & Robotic Systems*, 65(1):247–264, 2012.
- [37] R. Diankov and J. Kuffner. Randomized statistical path planning. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6, 2007.
- [38] G Diaz-Arango, A Sarmiento-Reyes, L Hernandez-Martinez, H Vazquez-Leal, D. D Lopez-Hernandez, and A Marin-Hernandez. Path optimization for terrestrial robots using homotopy path planning method. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2824–2827. IEEE, 2015.
- [39] Gerardo Diaz-Arango, Hector Vazquez-Leal, Luis Hernandez-Martinez, Victor Manuel Jimenez-Fernandez, Aurelio Heredia-Jimenez, Roberto C Ambrosio, Jesus Huerta-Chua, Hector De Cos-Cholula, and Sergio Hernandez-Mendez. Multiple-target homotopic quasi-complete path planning method for mobile robot using a piecewise linear approach. *Sensors (Basel, Switzerland)*, 20(11):3265, 2020.
- [40] A R Diéguez, R Sanz, and J López. Deliberative on-line local path planning for autonomous mobile robots. *Journal of Intelligent & Robotic Systems*, 37(1):1–19, 2003.
- [41] Bruce Donald, Kevin (Kevin M.) Lynch, and Daniela Rus, editors. *Algorithmic and computational robotics : new directions 2000 WAFR*. A K Peters/CRC Press, an imprint of Taylor and Francis, Boca Raton, FL, first edition, 2001.
- [42] Leo Dorst, Indur Mandhyan, and Karen Trovato. The geometrical representation of path planning problems. *Robotics and Autonomous Systems*, 7(2):181–195, 1991.
- [43] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014.
- [44] Hector Eduardo De Cos-Cholula, Gerardo Ulises Diaz-Arango, Luis Hernandez-Martinez, Hector Vazquez-Leal, Arturo Sarmiento-Reyes, Maria Teresa Sanz-Pascual, Agustin Leobardo Herrera-May, and Roberto Castaneda-Sheissa. FPGA implementation of homotopic path planning method with automatic assignment of repulsion parameter. *Energies (Basel)*, 13(10):2623, 2020.
- [45] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.
- [46] Michael Farber. Topological complexity of motion planning. *Discrete & Computational Geometry*, 29(2):211–221, 2003.
- [47] B. Faverjon and P. Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1152–1159, 1987.

- [48] D. Ferguson and A. Stentz. Anytime RRTs. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5369–5375, 2006.
- [49] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.
- [50] FreightWaves. Benzinger: Report: Warehouse robotics investment surges 572020.
- [51] Yiqi Gao, Andrew Gray, H. Eric Tseng, and Francesco Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.
- [52] Michele Ginesi, Daniele Meli, Andrea Roberti, Nicola Sansonetto, and Paolo Fiorini. Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions. *Journal of Intelligent & Robotic Systems*, 101(4), 2021.
- [53] Jean Gregoire, Michal Čáp, and Emilio Frazzoli. Locally-optimal multi-robot navigation under delaying disturbances using homotopy constraints. *Autonomous Robots*, 42(4):895–907, 2018.
- [54] Ioannis Havoutis and Subramanian Ramamoorthy. Motion planning and reactive control on learnt skill manifolds. *The International Journal of Robotics Research*, 32(9-10):1120–1150, 2013.
- [55] Michael E Henderson. Multiple parameter continuation: Computing implicitly defined k-manifolds. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, 12(3):451–476, 2002.
- [56] Emili Hernandez, Marc Carreras, and Pere Ridao. A comparison of homotopic path planning algorithms for robotic applications. *Robotics and Autonomous Systems*, 64:44–58, 2015.
- [57] Emili Hernandez, Marc Carreras, Pere Ridao, Javi Antich, and Alberto Ortiz. A search-based path planning algorithm with topological constraints. application to an AUV. *IFAC Proceedings Volumes*, 44(1):13654–13659, 2011.
- [58] Emili Hernández, Marc Carreras, and Pere Ridao. A path planning algorithm for an AUV guided with homotopy classes. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling*, volume 21, 01 2011.
- [59] Emili Hernández, Marc Carreras, and Pere Ridao. A bug-based path planner guided with homotopy classes. *ICINCO 2012 - Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics*, 2:123–131, 01 2012.
- [60] Arne-Christoph Hildebrandt, Moritz Klischat, Daniel Wahrmann, Robert Wittmann, Felix Sygulla, Philipp Seiwald, Daniel Rixen, and Thomas Buschmann. Real-time path planning in unknown environments for bipedal robots. *IEEE Robotics and Automation Letters*, 2(4):1856–1863, 2017.

- [61] Md. Arafat Hossain and Israt Ferdous. Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. *Robotics and Autonomous Systems*, 64:137–141, 2015.
- [62] D. Hsu and Zheng Sun. Adaptively combining multiple sampling strategies for probabilistic roadmap planning. In *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, volume 2, pages 774–779 vol.2, 2004.
- [63] John (John R.) Hudson. *The robot revolution : understanding the social and economic impact*. Edward Elgar Pub., Northampton, MA, 2019.
- [64] K. Hughes, A. Tokuta, and N. Ranganathan. Trulla : An algorithm for path planning among weighted regions by localized propagations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 469–476, 1992.
- [65] L Jaillet and J. M Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 29(1):105–117, 2013.
- [66] Léonard Jaillet, Juan Cortés, and T Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.
- [67] Léonard Jaillet and Josep Porta. Asymptotically-optimal path planning on manifolds. *Robotics Science and Systems VIII*, 07 2012.
- [68] Léonard Jaillet and Thierry Simeon. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. *The International Journal of Robotics Research*, 27(11-12):1175–1188, 2008.
- [69] Kevin D. Jenkins. The shortest path problem in the plane with obstacles: A graph modeling approach to producing finite search lists of homotopy classes. Master’s thesis, Naval Postgraduate School Monterey California, June 1991.
- [70] Rahul Kala. Homotopic roadmap generation for robot motion planning. *Journal of Intelligent & Robotic Systems*, 82(3):555–575, 2016.
- [71] Rahul Kala. Homotopy conscious roadmap construction by fast sampling of narrow corridors. *Applied Intelligence (Dordrecht, Netherlands)*, 45(4):1089–1102, 2016.
- [72] Gitae Kang, Yong Bum Kim, Young Hun Lee, Hyun Seok Oh, Won Suk You, and Hyouk Ryeol Choi. Sampling-based motion planning of manipulator with goal-oriented sampling. *Intelligent Service Robotics*, 12(3):265–273, 2019.
- [73] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
- [74] Ajay Kattapur and Balamuralidhar Purushotaman. Roboplanner: a pragmatic task planning framework for autonomous robots. *Cognitive Computation and Systems*, 2(1):12–22, 2020.

- [75] L. E. Kavraki, M. N. Kolountzakis, and J. . Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.
- [76] L.E Kavraki, P Svestka, J.-C Latombe, and M.H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [77] Monroe Kennedy, Dinesh Thakur, M Ani Hsieh, Subhrajit Bhattacharya, and Vijay Kumar. Optimal paths for polygonal robots in SE(2). *Journal of Mechanisms and Robotics*, 10(2), 2018.
- [78] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.
- [79] P. Khosla and R. Volpe. Superquadric artificial potentials for obstacle avoidance and approach. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1778–1784 vol.3, 1988.
- [80] Donghyuk Kim, Mincheul Kang, and Sung-Eui Yoon. Volumetric tree: Adaptive sparse graph for effective exploration of homotopy classes. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1496–1503. IEEE, 2019.
- [81] Hoyeon Kim, U. Kei Cheang, Louis W Rogowski, and Min Jun Kim. Motion planning of particle based microrobots for static obstacle avoidance. *Journal of Micro-Bio Robotics*, 14(1-2):41–49, 2018.
- [82] Soonkyum Kim, Koushil Sreenath, Subhrajit Bhattacharya, and Vijay Kumar. Trajectory planning for systems with homotopy class constraints. In *Latest Advances in Robot Kinematics (ARK)*, pages 83–90, Innsbruck, Austria, June 2012.
- [83] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences - PNAS*, 95(15):8431–8435, 1998.
- [84] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1–6, 1987.
- [85] Daniel E Koditschek and Elon Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11(4):412–442, 1990.
- [86] Keshav Kolur, Sahit Chintalapudi, Byron Boots, and Mustafa Mukadam. Online motion planning over multiple homotopy classes with gaussian process inference. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2358–2364. IEEE, 2019.
- [87] Wojciech Kowalczyk. Rapid navigation function control for two-wheeled mobile robots. *Journal of Intelligent & Robotic Systems*, 93(3-4):687–697, 2018.

- [88] Wojciech Kowalczyk, Wojciech Kowalczyk, Mateusz Przybyla, Mateusz Przybyla, Krzysztof Kozłowski, and Krzysztof Kozłowski. Set-point control of mobile robot with obstacle detection and avoidance using navigation function - experimental verification. *Journal of Intelligent & Robotic Systems*, 85(3):539–552, 2017.
- [89] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, 2000.
- [90] Subhrajit Bhattacharya Maxim Likhachev Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3D. In *Robotics*. MIT Press, 2012.
- [91] Yoshiaki Kuwata, Justin Teo, Sertac Karaman, Gaston Fiore, Emilio Frazzoli, and Jonathan How. Motion planning in complex environments using closed-loop prediction. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. AIAA, 08 2008.
- [92] F Lamiriaux, D Bonnafoous, and O Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Transactions on Robotics*, 20(6):967–977, 2004.
- [93] Christian Laugier and Raja Chatila, editors. *Autonomous Navigation in Dynamic Environments*. Springer Tracts in Advanced Robotics, 35. Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. 2007. edition, 2007.
- [94] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):673–692, 2004.
- [95] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
- [96] J. Lee, C. Pippin, and T. Balch. Cost based planning with RRT in outdoor environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 684–689, 2008.
- [97] J.M. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics ; v.218. Springer, New York, NY, second edition, 2002.
- [98] John . M. Lee. *Introduction To Topological Manifolds*. Springer, Dordrecht, second edition, 2011.
- [99] Paulo Leica, Danilo Chavez, Andres Rosales, Flavio Roberti, Juan Marcos Toibero, and Ricardo Carelli. Strategy based on multiple objectives and null space for the formation of mobile robots and dynamic obstacle avoidance. *Revista Politécnica (Quito)*, 33(1), 2014.

- [100] Jed Lengyel, Mark Reichert, Bruce R Donald, and Donald P Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. *Computer graphics (New York, N.Y.)*, 24(4):327–335, 1990.
- [101] Xiaohu Li, Gen Zhao, and Baotong Li. Generating optimal path by level set approach for a mobile robot moving in static/dynamic environments. *Applied Mathematical Modelling*, 85:210–230, 2020.
- [102] Yucong Lin and Srikanth Saripalli. Sampling-based path planning for uav collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3179–3192, 2017.
- [103] Guanfeng Liu, Jeff Trinkle, Yong Yang, and Shaoming Luo. Motion planning of planar closed chains based on structural sets. *IEEE Access*, 8:117203–117217, 2020.
- [104] Yang LIU, Zheng ZHENG, and Fangyun QIN. Homotopy based optimal configuration space reduction for anytime robotic motion planning. *Chinese Journal of Aeronautics*, 34(1):364–379, 2021.
- [105] Yongbei Liu, Naiming Qi, Weiran Yao, Jun Zhao, and Song Xu. Cooperative path planning for aerial recovery of a UAV swarm using genetic algorithm and homotopic approach. *Applied Sciences*, 10(12):4154–, 2020.
- [106] Vladimir J Lumelsky and Alexander A Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430, 1987.
- [107] Ellips Masehian and M. R Amin-Naseri. A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *Journal of Robotic Systems*, 21(6):275–300, 2004.
- [108] L McFetridge and M.Y Ibrahim. A new methodology of mobile robot navigation: The agoraphilic algorithm. *Robotics and Computer-Integrated Manufacturing*, 25(3):545–551, 2009.
- [109] Troy McMahon, Shawna Thomas, and Nancy M Amato. Sampling-based motion planning with reachable volumes for high-degree-of-freedom manipulators. *The International Journal of Robotics Research*, 37(7):779–817, 2018.
- [110] Margarita Mediavilla, José Luis González, Juan Carlos Fraile, and José Ramón Perán. Reactive approach to on-line path planning for robot manipulators in dynamic environments. *Robotica*, 20(4):375–384, 2002.
- [111] Marina Blanton Mikhail J.Atallah, editor. *Algorithms and Theory of Computation Handbook, Special Topics and Techniques*, volume 2. Chapman and Hall/CRC, 2nd edition, November 2009.
- [112] Steven M.LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, Ames, IA 50011 USA, June 1998.

- [113] Clara Pérez Molina, Rosario Gil Ortego, and Francisco Mur Pérez. *Perspectives on Technological Developments Applied to Robotics*, pages 59–86. Springer London, London, 2014.
- [114] Oscar Montiel, Ulises Orozco-Rosas, and Roberto Sepúlveda. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12):5177–5191, 2015.
- [115] James R. Munkres. *Topology*. Prentice Hall, Upper Saddle River, NJ, 2nd ed. edition, 2000.
- [116] Robin R Murphy, Ken Hughes, Alisa Marzilli, and Eva Noll. Integrating explicit path planning with reactive control of mobile robots using trulla. *Robotics and Autonomous Systems*, 27(4):225–245, 1999.
- [117] Tristan Needham. *Visual Complex Analysis*. Oxford University Press, 1997.
- [118] Liviu Nicolaescu. Homeomorphisms vs. diffeomorphisms, 01 2003. Professor Nicolaescu’s notes on the topic.
- [119] Liviu I. Nicolaescu. *Lectures on the geometry of manifolds*, volume 32. Ringgold Inc, Portland, second edition, September 2008.
- [120] Iram Noreen, Amna Khan, Hyejeong Ryu, Nakju Lett Doh, and Zulfiqar Habib. Optimal path planning in cluttered environment using RRT-AB. *Intelligent Service Robotics*, 11(1):41–52, 2017.
- [121] Jason C. Olmstead Muhs and Jingzhou Yang. A geodesics-based model for obstacle avoidance. In *2005 Digital Human Modeling for Design and Engineering Symposium*. SAE International, jun 2005.
- [122] Giuseppe Oriolo. Motion planning 3 artificial potential fields, 10 2020. Professor Oriolo’s notes for his class in Artificial Intelligence and Robotics.
- [123] J-M Park, D-W Kim, Y-S Yoon, H J Kim, and K-S Yi. Obstacle avoidance of autonomous vehicles based on model predictive control. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 223(12):1499–1516, 2009.
- [124] Junghee Park, Sisir Karumanchi, and Karl Iagnemma. Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming. *IEEE Transactions on Robotics*, 31(5):1101–1115, 2015.
- [125] Per-Olof Persson. *Mesh Generation for Implicit Geometries*. Ph.D. dissertation, Massachusetts Institute of Technology, February 2005.
- [126] Per-Olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, 2004.

- [127] Josep M Porta, Léonard Jaillet, and Oriol Bohigas. Randomized path planning on manifolds based on higher-dimensional continuation. *The International Journal of Robotics Research*, 31(2):201–215, 2012.
- [128] Andrew. Pressley. *Elementary differential geometry*. Springer undergraduate mathematics series. Springer, London, second edition, 2010.
- [129] Long Qin, Quanjun Yin, Yabing Zha, and Yong Peng. Dynamic detection of topological information from grid-based generalized voronoi diagrams. *Mathematical Problems in Engineering*, 2013:1–11, 2013.
- [130] Paul Quillen, Josué Muñoz, and Kamesh Subbarao. Path planning to a reachable state using minimum control effort based navigation functions. *The Journal of the astronautical sciences*, 66(4):554–581, 2019.
- [131] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2, 1993.
- [132] Sean Quinlan. Real-time modification of collision-free paths, 1995.
- [133] A. H. Qureshi, K. F. Iqbal, S. M. Qamar, F. Islam, Y. Ayaz, and N. Muhammad. Potential guided directional-RRT* for accelerated motion planning in cluttered environments. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 519–524, 2013.
- [134] A. H. Qureshi, S. Mumtaz, K. F. Iqbal, B. Ali, Y. Ayaz, F. Ahmed, M. S. Muhammad, O. Hasan, W. Y. Kim, and M. Ra. Adaptive potential guided directional-RRT. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1887–1892, 2013.
- [135] Ahmed Hussain Qureshi and Yasar Ayaz. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robotics and Autonomous Systems*, 68:1–11, 2015.
- [136] Ahmed Hussain Qureshi, Ahmed Hussain Qureshi, Yasar Ayaz, and Yasar Ayaz. Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, 40(6):1079–1093, 2016.
- [137] S. Radhakrishnan and W. Gueaieb. Genetic algorithm based direction finder on the manifold for singularity free paths. In *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 151–156, 2016.
- [138] S. Radhakrishnan and W. Gueaieb. Reconfigurable EKF for 2D SLAM. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6, 2016.

- [139] S. Radhakrishnan and W. Gueaieb. Constraint-free discretized manifolds for robotic path planning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1878–1884. IEEE, October 2020.
- [140] Sindhu Radhakrishnan. Observable 2D SLAM and Evidential Occupancy Grids. Master’s thesis, Carleton University, 2014.
- [141] Sindhu Radhakrishnan and Wail Gueaieb. Analysis of parameters guiding manifold-based homotopic path planning. *Journal of Robotics*, 2022. (To be submitted).
- [142] Sindhu Radhakrishnan and Wail Gueaieb. Manifold-based homotopic path planning. *International Journal of Advanced Robotics Systems*, 2022. (To be submitted).
- [143] Sindhu Radhakrishnan and Wail Gueaieb. A state-of-the-art review on topological geometry-based robotic path planning. Part I: Static constraints. *Robotics and Autonomous Systems*, 2022. (To be submitted).
- [144] Sindhu Radhakrishnan and Wail Gueaieb. A state-of-the-art review on topological geometry-based robotic path planning. Part II: Dynamic constraints. *Robotics and Autonomous Systems*, 2022. (To be submitted).
- [145] Y. Rasekhipour, A. Khajepour, S. Chen, and B. Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1255–1267, 2017.
- [146] Yadollah Rasekhipour, Iman Fadakar, and Amir Khajepour. Autonomous driving motion planning with obstacles prioritization using lexicographic optimization. *Control Engineering Practice*, 77:235–246, 2018.
- [147] Debanik Roy. Algorithmic path planning of static robots in three dimensions using configuration space metrics. *Robotica*, 29(2):295–315, 2011.
- [148] Jae Chun Ryu, Jae Chun Ryu, Frank Chongwoo Park, Frank Chongwoo Park, Yoon Young Kim, and Yoon Young Kim. Mobile robot path planning algorithm by equivalent conduction heat flow topology optimization. *Structural and multidisciplinary optimization*, 45(5):703–715, 2012.
- [149] Andrey V Savkin and Michael Hoy. Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments. *Robotica*, 31(2):323–330, 2013.
- [150] J. A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences - PNAS*, 93(4):1591–1595, 1996.
- [151] J. A Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999.
- [152] Antonio Sgorbissa. Integrated robot planning, path following, and obstacle avoidance in two and three dimensions: Wheeled robots, underwater vehicles, and multicopters. *The International journal of robotics research*, 38(7):853–876, 2019.

- [153] N Shvalb, M Shoham, G Liu, and J.C Trinkle. Motion planning for a class of planar closed-chain manipulators. *The International Journal of Robotics Research*, 26(5):457–473, 2007.
- [154] Bruno. Siciliano. *Robotics Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, London, 1st ed. 2009. edition, 2009.
- [155] T. Simeon, J.-P. Laumond, C.V. Van Geem, and J. Cortes. Computer aided motion: Move3d within molog. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1494–1499, 2001.
- [156] Elias M. Stein. *Complex analysis*. Princeton lectures in analysis ; 2. Princeton University Press, Princeton, N.J, 2003.
- [157] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, 1994.
- [158] A Stopp and T Riethmuller. Fast reactive path planning by 2d and 3d multi-layer spatial grids for mobile robot navigation. In *Proceedings of Tenth International Symposium on Intelligent Control*, pages 545–550. IEEE, 1995.
- [159] Junghun Suh, Joonsig Gong, and Songhwai Oh. Fast sampling-based cost-aware path planning with nonmyopic extensions using cross entropy. *IEEE Transactions on Robotics*, 33(6):1313–1326, 2017.
- [160] H.G Tanner and A Kumar. Towards decentralization of multi-robot navigation functions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4132–4137. IEEE, 2005.
- [161] Songqiao Tao and Juan Tan. Path planning with obstacle avoidance based on normalized r -functions. *Journal of Robotics*, 2018:1–10, 2018.
- [162] J.C Trinkle and R. James Milgram. Complete path planning for closed kinematic chains with spherical joints. *The International Journal of Robotics Research*, 21(9):773–789, 2002.
- [163] Loring W. Tu. *An introduction to manifolds*. Universitext. Springer, New York, second edition, 2010.
- [164] Jur van den Berg and Mark Overmars. Planning time-minimal safe paths amidst unpredictably moving obstacles. *The International Journal of Robotics Research*, 27(11-12):1274–1294, 2008.
- [165] H Vazquez-Leal, A Marin-Hernandez, Y Khan, A Yildirim, U Filobello-Nino, R Castaneda-Sheissa, and V.M Jimenez-Fernandez. Exploring collision-free path planning by using homotopy continuation methods. *Applied Mathematics and Computation*, 219(14):7514–7532, 2013.

- [166] R. Volpe and P. Khosla. Artificial potentials with elliptical isopotential contours for obstacle avoidance. In *26th IEEE Conference on Decision and Control*, volume 26, pages 180–185, 1987.
- [167] R. Volpe and P. Khosla. Manipulator control with superquadric artificial potential functions: theory and experiments. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1423–1436, 1990.
- [168] Hisaka Wada, Jun Kinugawa, and Kazuhiro Kosuge. Reactive motion planning using time-layered c-spaces for a collaborative robot paddy. *Advanced Robotics*, 35(8):490–503, 2021.
- [169] Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020.
- [170] C Wang, Y. S Mao, and K. J Du. Simulation on local obstacle avoidance algorithm for unmanned surface vehicle. *International Journal of Simulation Modelling*, 15(3):460–472, 2016.
- [171] Daoyong Wang, Peng Wang, Xiantao Zhang, Xiaoxian Guo, Yaqing Shu, and Xinliang Tian. An obstacle avoidance strategy for the wave glider based on the improved artificial potential field and collision prediction model. *Ocean Engineering*, 206:107356–, 2020.
- [172] Wei Wang, Lei Zuo, and Xin Xu. A learning-based multi-rrt approach for robot path planning in narrow passages. *Journal of Intelligent & Robotic Systems*, 90(1-2):81–100, 2017.
- [173] Kun Wei and Bingyin Ren. A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors (Basel, Switzerland)*, 18(2):571–, 2018.
- [174] Jenny Wilson. Manifolds, September 2012. Notes for graduate students via seminar by Dr. Jenny Wilson.
- [175] Albert Wu and Jonathan P How. Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. *Autonomous Robots*, 32(3):227–242, 2012.
- [176] K. Wu, C. Lo, Y. Lin, and J. Liu. 3D path planning based on nonlinear geodesic equation. In *11th IEEE International Conference on Control Automation (ICCA)*, pages 342–347, 2014.
- [177] Kun-Lin Wu, Ting-Jui Ho, Sean A Huang, Kuo-Hui Lin, Yueh-Chen Lin, and Jing-Sin Liu. Path planning and replanning for mobile robot navigation on 3D terrain: An approach based on geodesic. *Mathematical Problems in Engineering*, 2016:1–12, 2016.

- [178] Bin Xu, Bin Xu, Daniel J Stilwell, Daniel J Stilwell, Andrew J Kurdila, and Andrew J Kurdila. Fast path re-planning based on fast marching and level sets. *Journal of Intelligent & Robotic Systems*, 71(3):303–317, 2013.
- [179] Weiran Yao, Naiming Qi, Jun Zhao, and Neng Wan. Bounded curvature path planning with expected length for dubins vehicle entering target manifold. *Robotics and Autonomous systems*, 97:217–229, 2017.
- [180] Daqing Yi, Michael Goodrich, and Kevin Seppi. Homotopy-aware RRT: Toward human-robot topological path-planning. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, HRI '16, pages 279–286. IEEE Press, 2016.
- [181] Yongsoon Yoon, Jongho Shin, H. Jin Kim, Yongwoon Park, and Shankar Sastry. Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice*, 17(7):741–750, 2009.
- [182] Botao Zhang, Yong Liu, Qiang Lu, and Jian Wang. A path planning strategy for searching the most reliable path in uncertain environments. *International Journal of Advanced Robotic Systems*, 13(5):172988141665775, 2016.

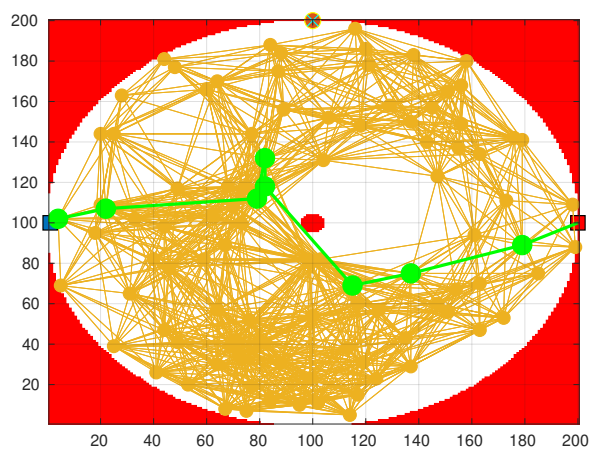
APPENDICES

Appendix A

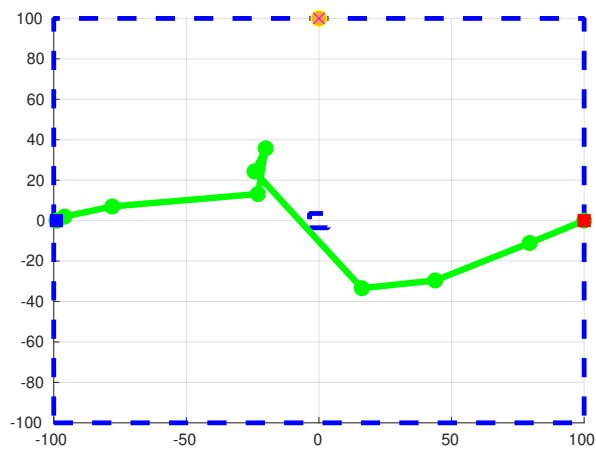
Stage 2: Case 1

This section displays those results from Section 4.2 corresponding to case 1, which were not originally shown in Section 4.2. The following results are displayed here:

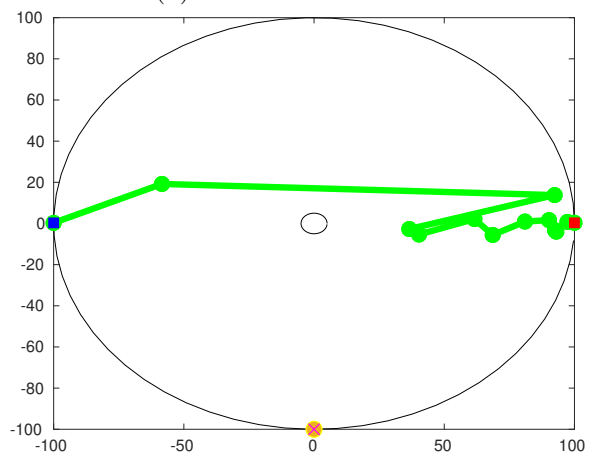
1. The paths produced by [PRM](#), [RRT](#) and [CFDMPP](#) in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
2. The paths produced by [CFDMPP](#) when the original path is homotopically blocked; displayed in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
3. Plots displaying trends of a metric for [PRM](#), [RRT](#) and [CFDMPP](#) (without and with blocking). The two metrics of path length ratio and percentage of usable samples that were not shown in Section 4.2, are shown here for [0,1,4,..31] poles.



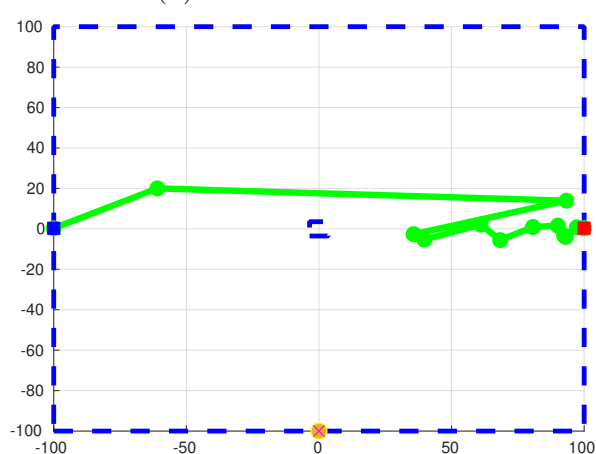
(a) PRM: Path in R-4



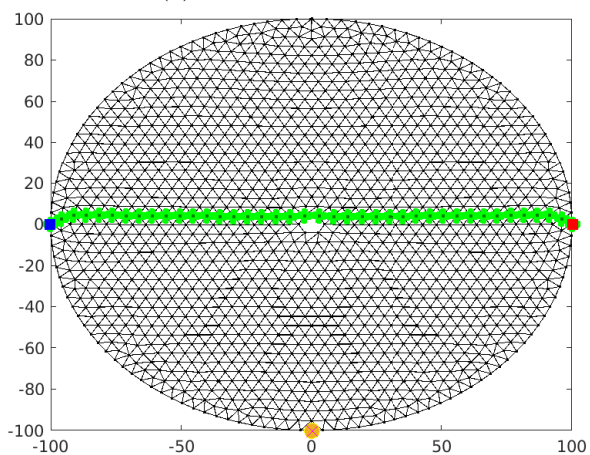
(b) PRM: Path in R-1



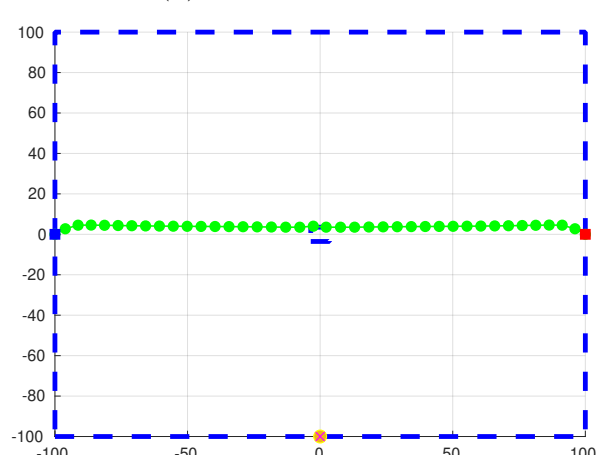
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure A.1: Case 1: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.

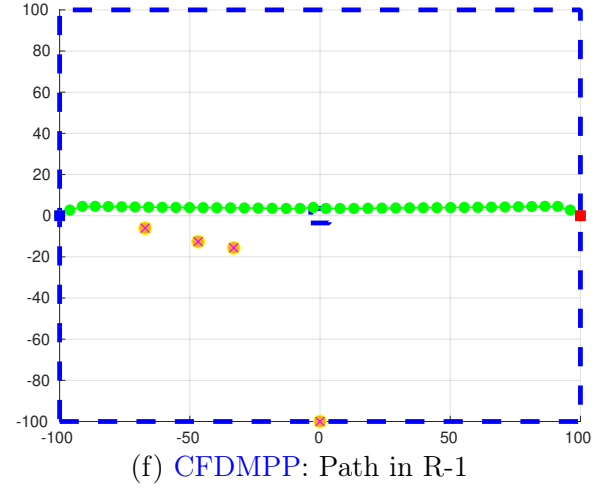
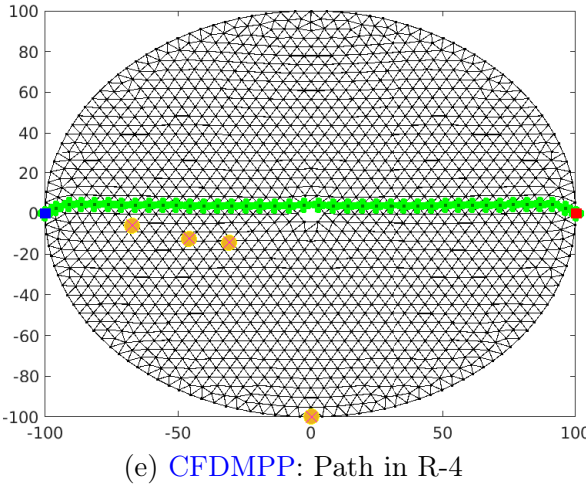
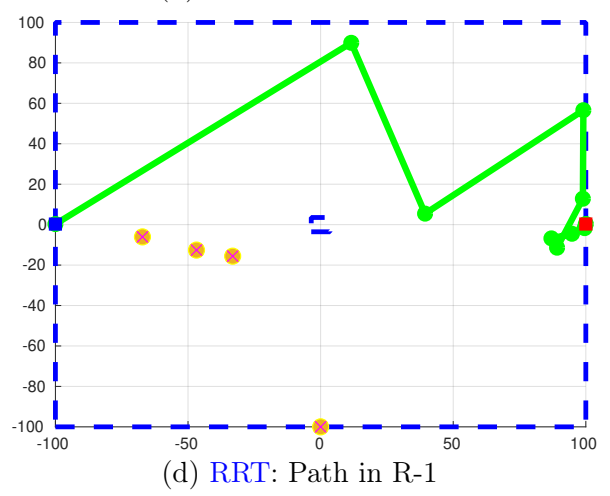
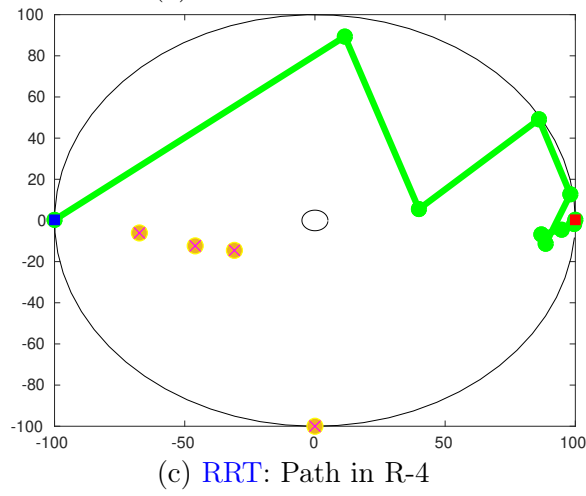
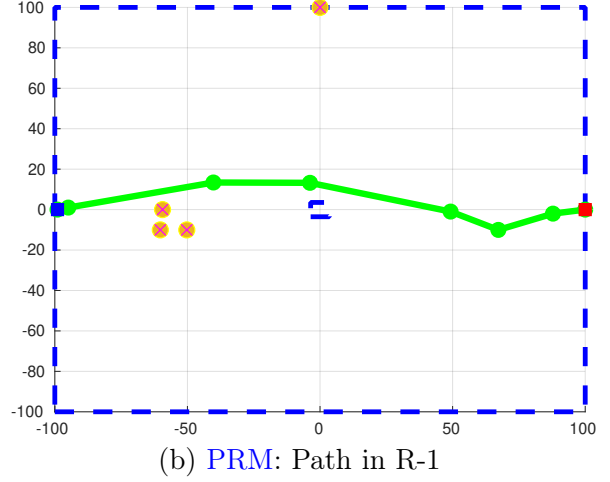
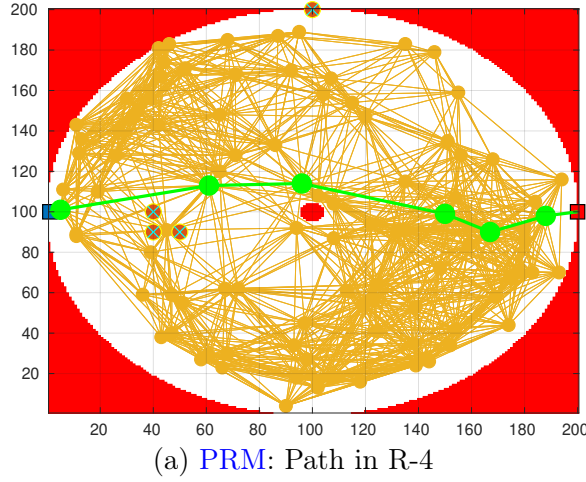
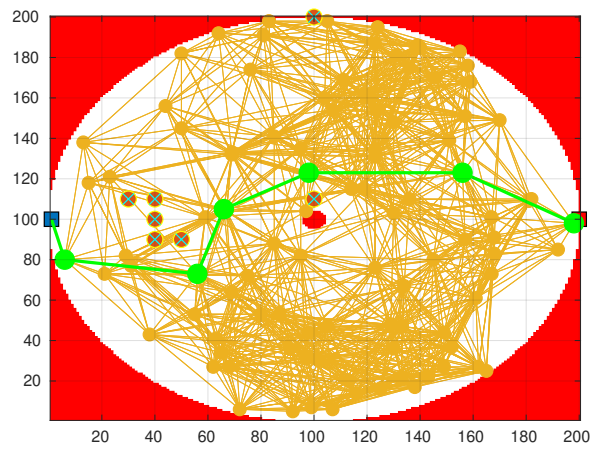
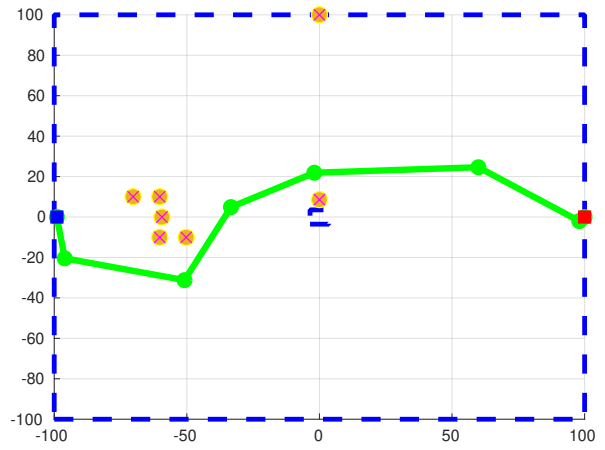


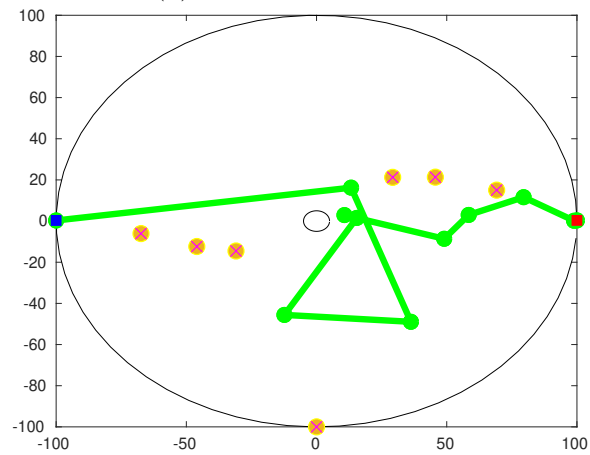
Figure A.2: Case 1: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.



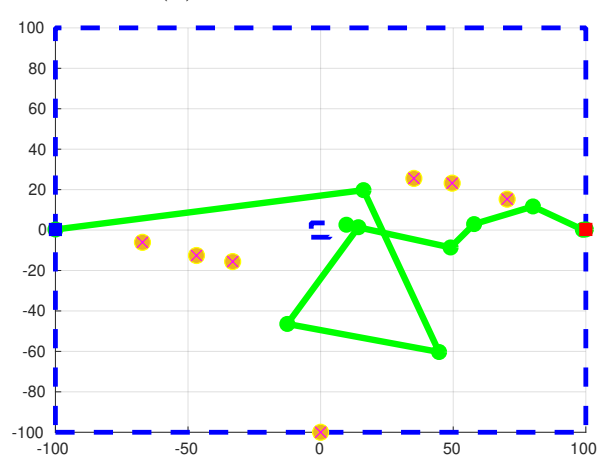
(a) PRM: Path in R-4



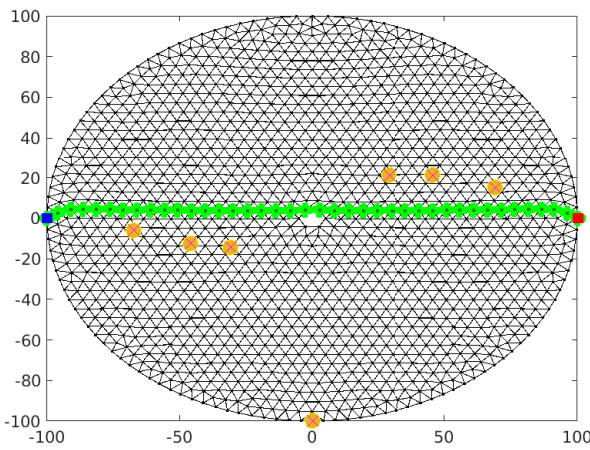
(b) PRM: Path in R-1



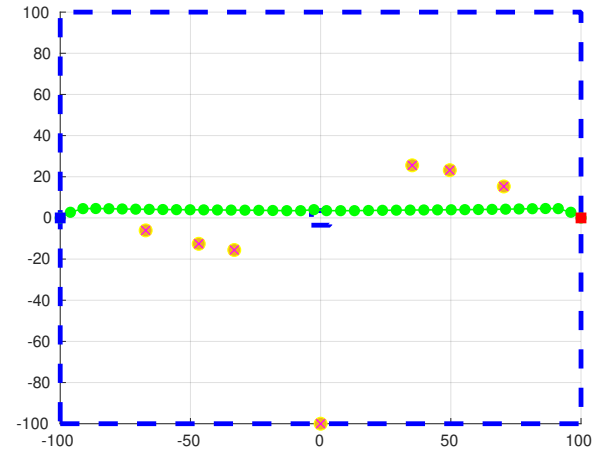
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure A.3: Case 1: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.

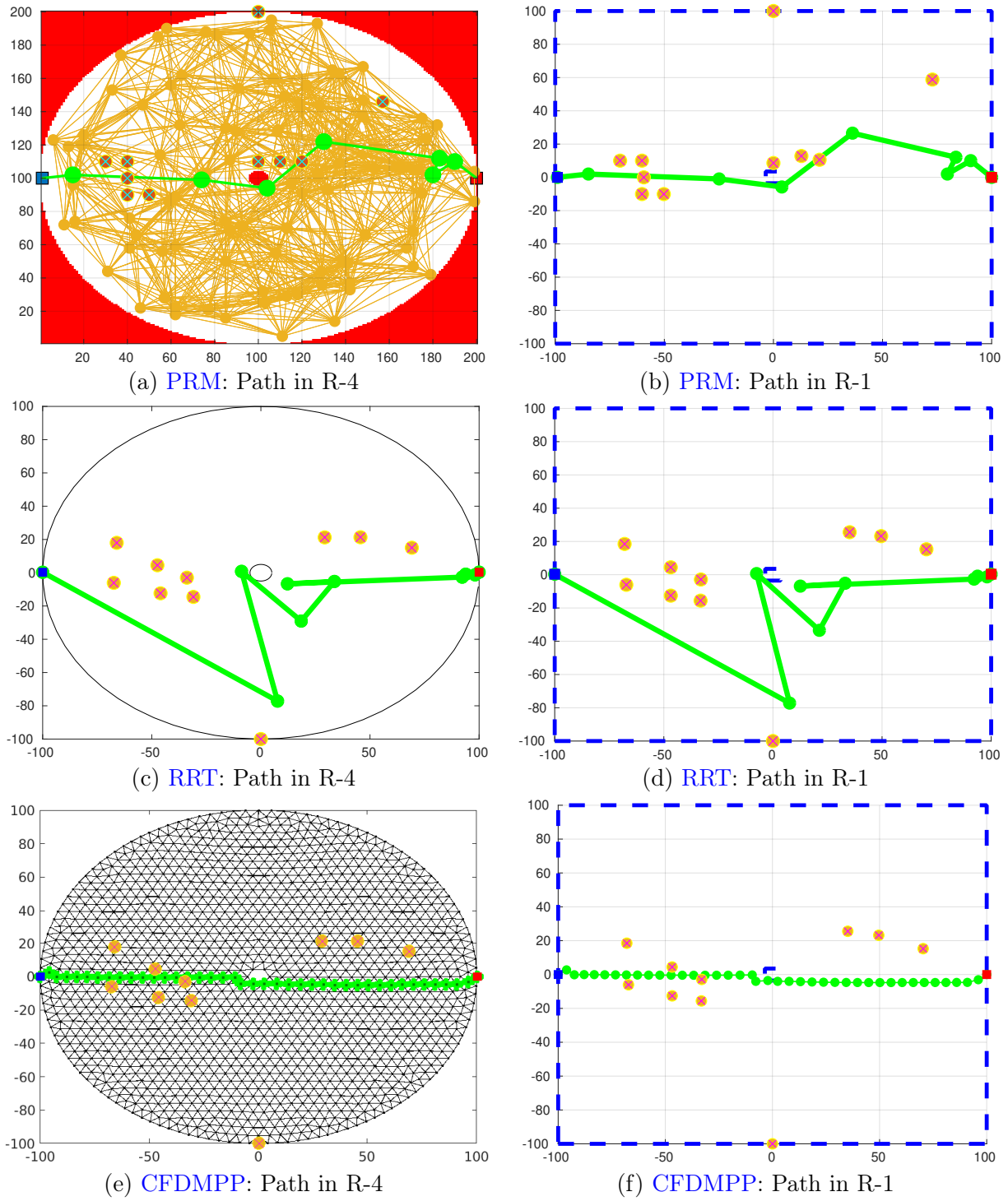
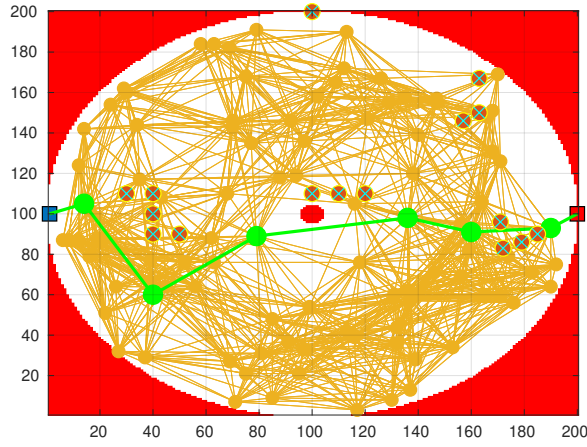
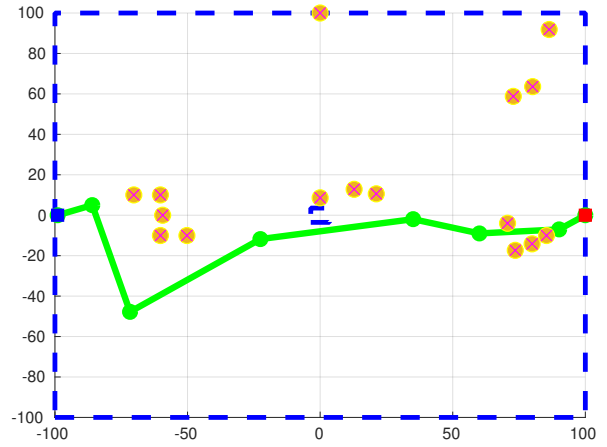


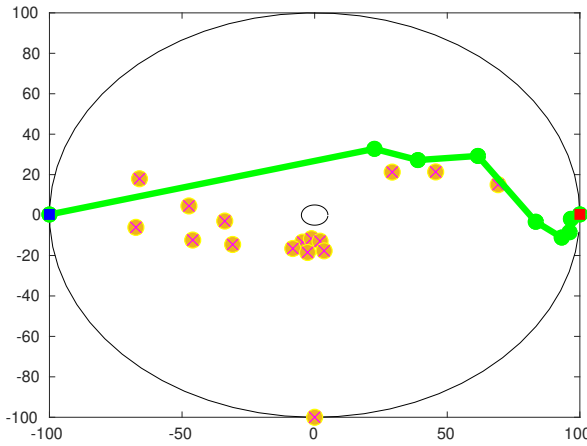
Figure A.4: Case 1: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.



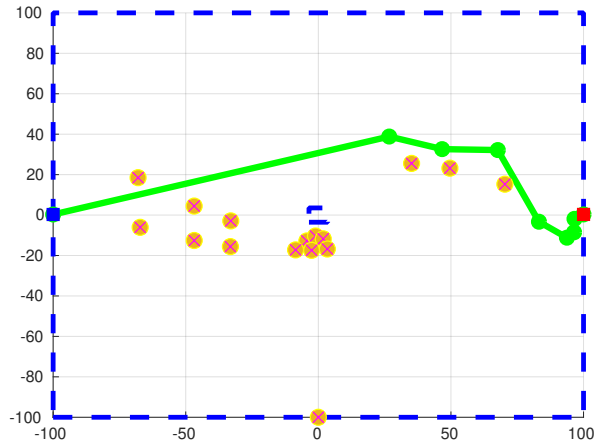
(a) PRM: Path in R-4



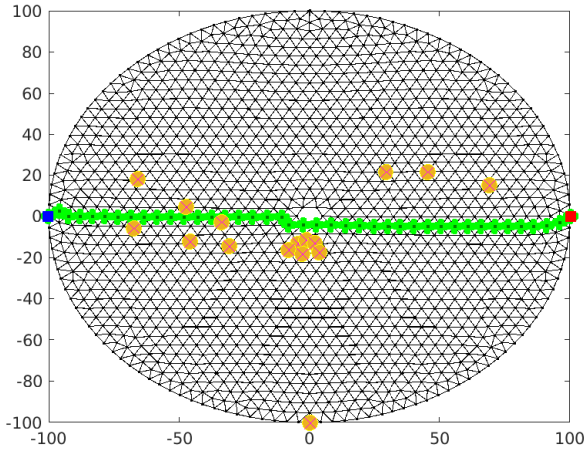
(b) PRM: Path in R-1



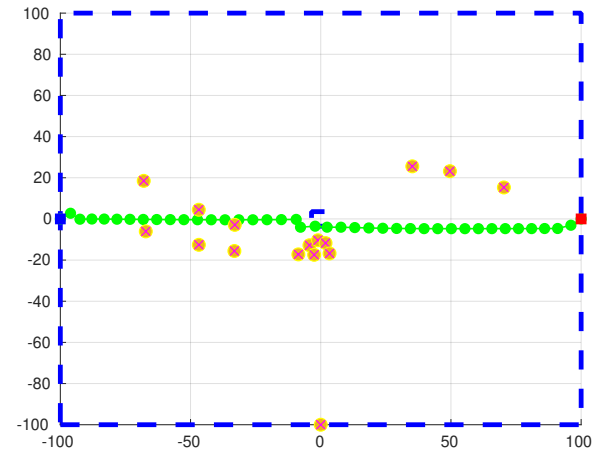
(c) RRT: Path in R-4



(d) RRT: Path in R-1

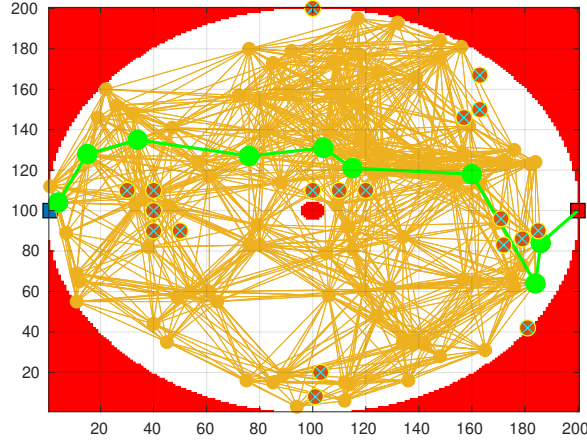


(e) CFDMPP: Path in R-4

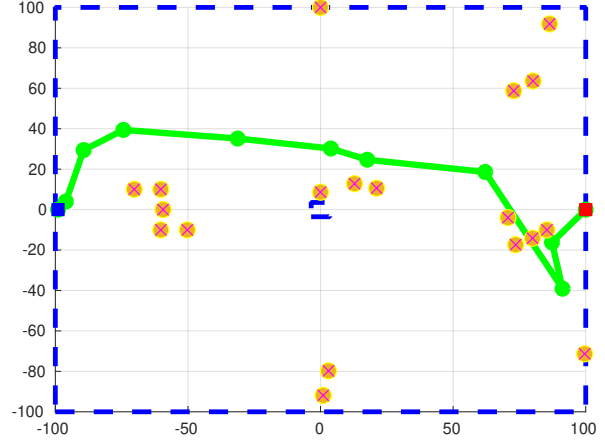


(f) CFDMPP: Path in R-1

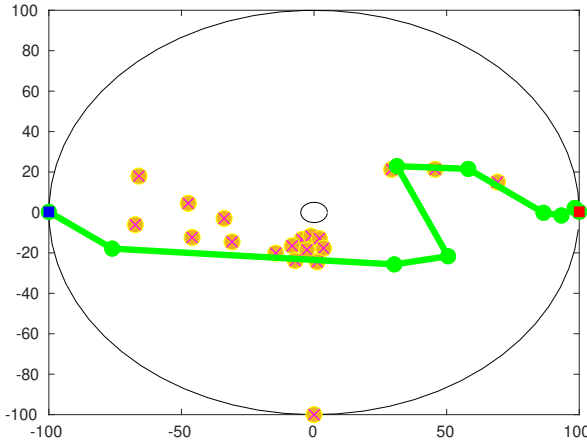
Figure A.5: Case 1: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.



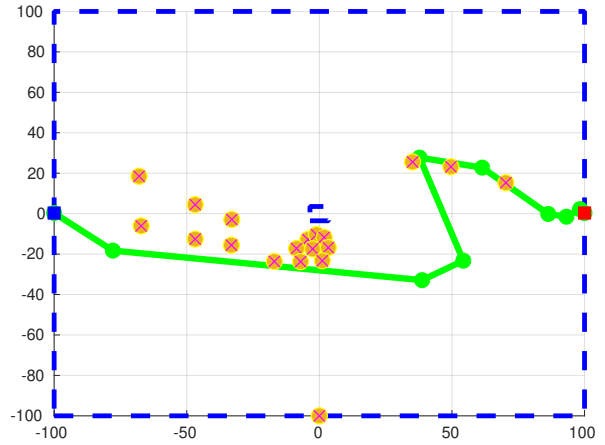
(a) PRM: Path in R-4



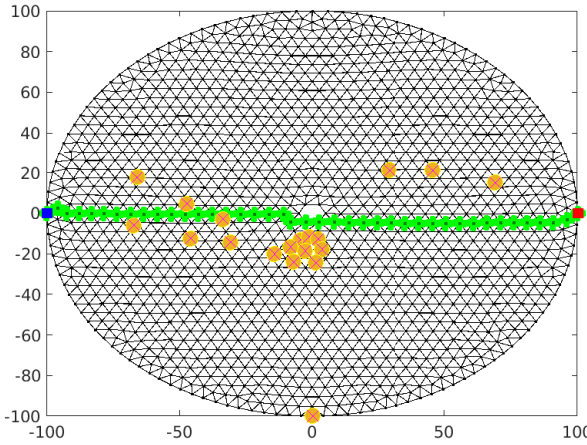
(b) PRM: Path in R-1



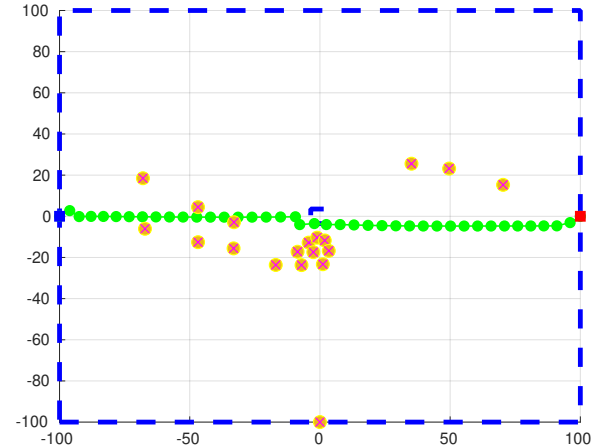
(c) RRT: Path in R-4



(d) RRT: Path in R-1

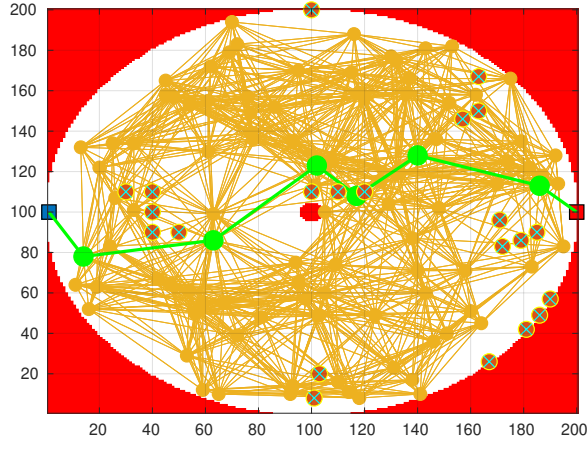


(e) CFDMPP: Path in R-4

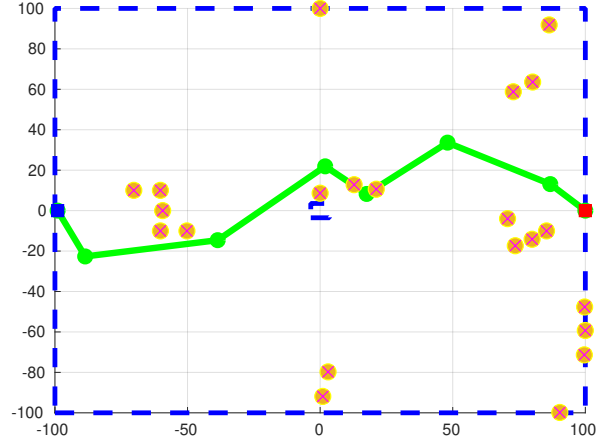


(f) CFDMPP: Path in R-1

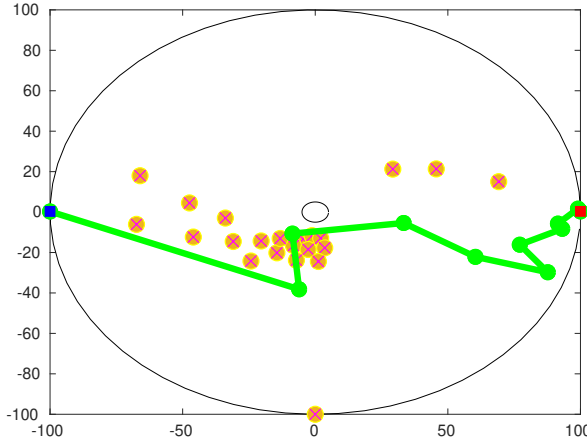
Figure A.6: Case 1: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.



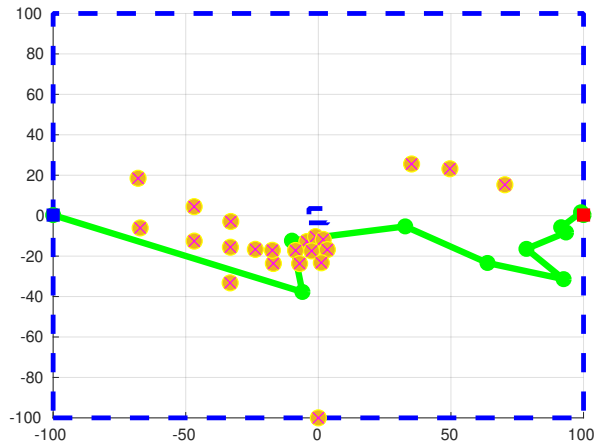
(a) PRM: Path in R-4



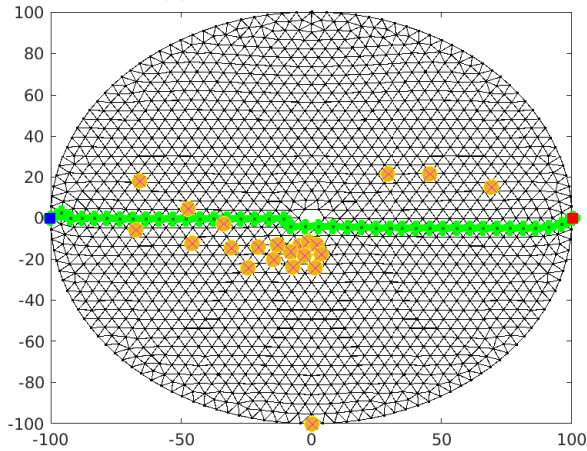
(b) PRM: Path in R-1



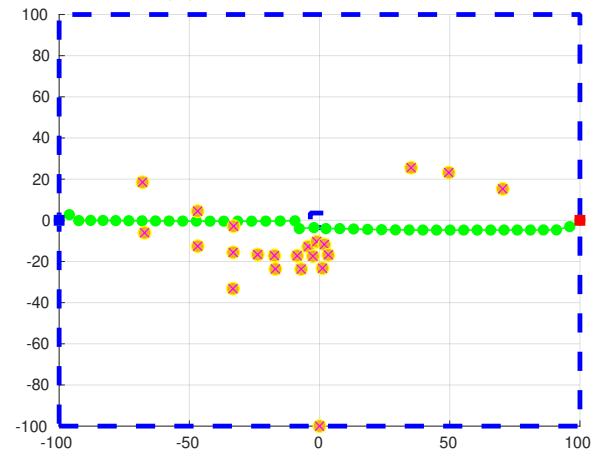
(c) RRT: Path in R-4



(d) RRT: Path in R-1

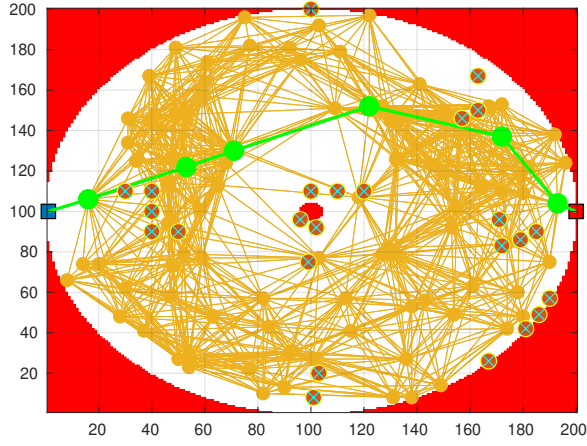


(e) CFDMPP: Path in R-4

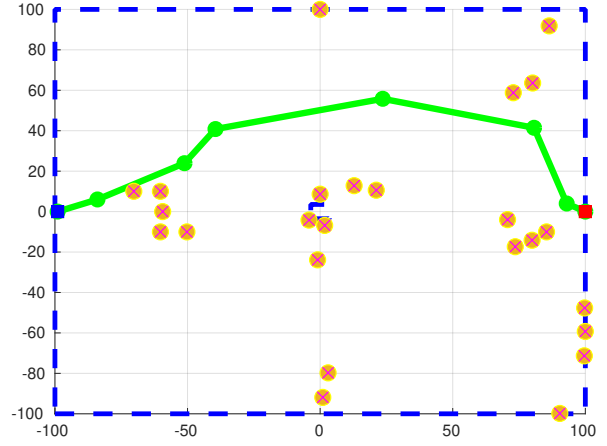


(f) CFDMPP: Path in R-1

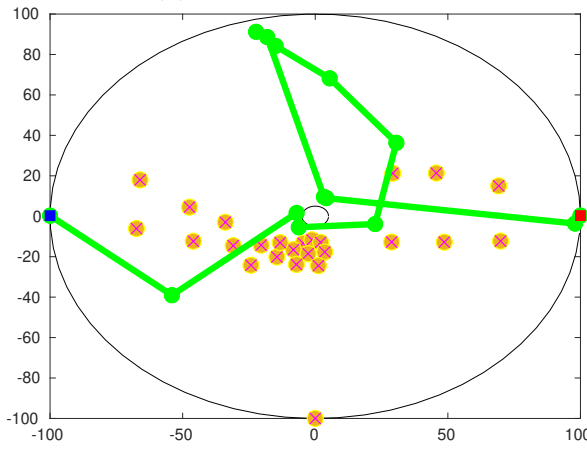
Figure A.7: Case 1: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.



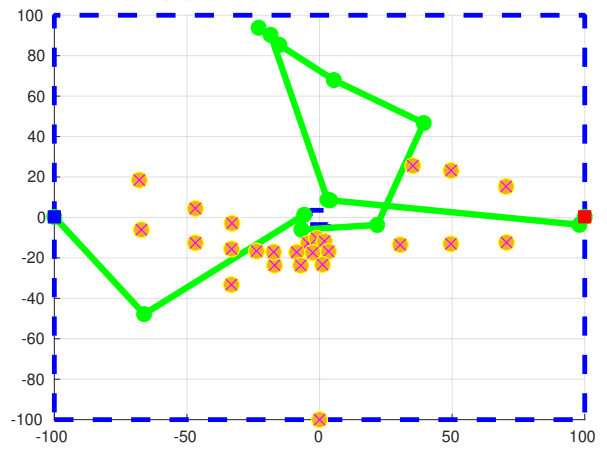
(a) PRM: Path in R-4



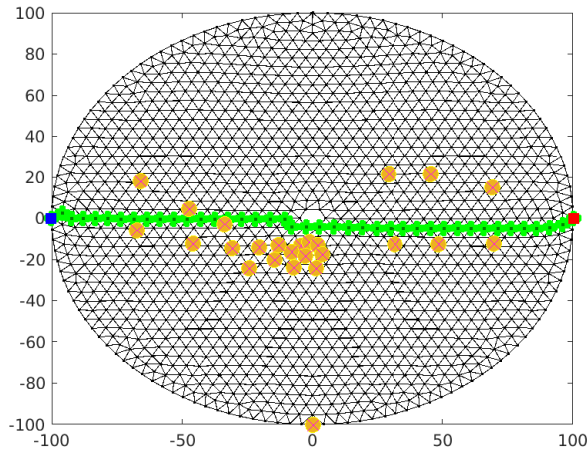
(b) PRM: Path in R-1



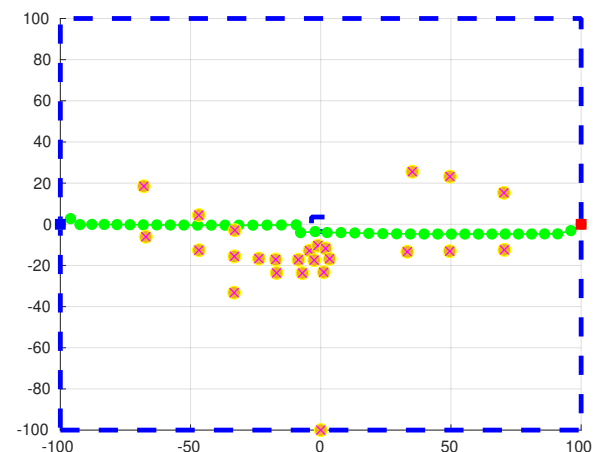
(c) RRT: Path in R-4



(d) RRT: Path in R-1

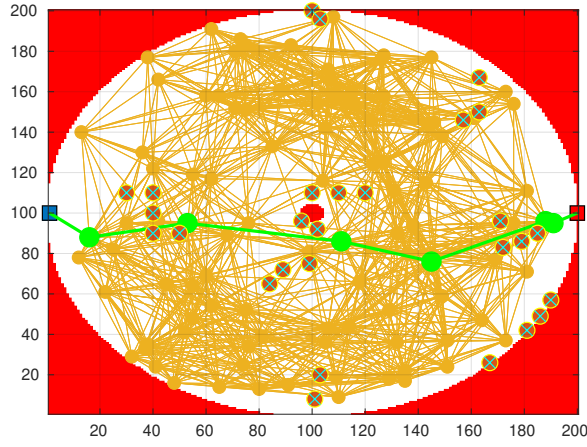


(e) CFDMPP: Path in R-4

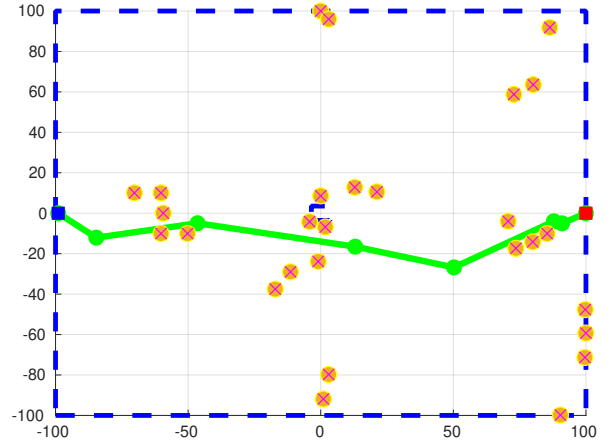


(f) CFDMPP: Path in R-1

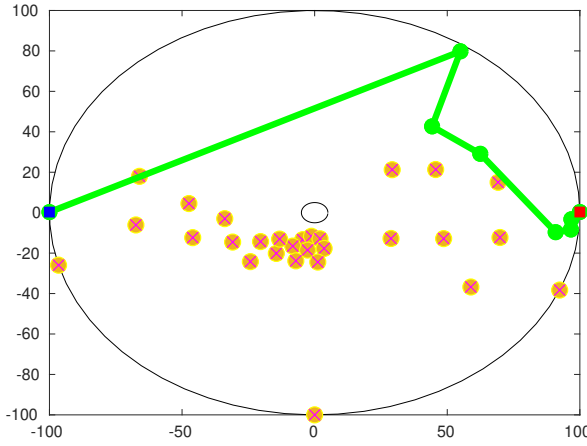
Figure A.8: Case 1: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.



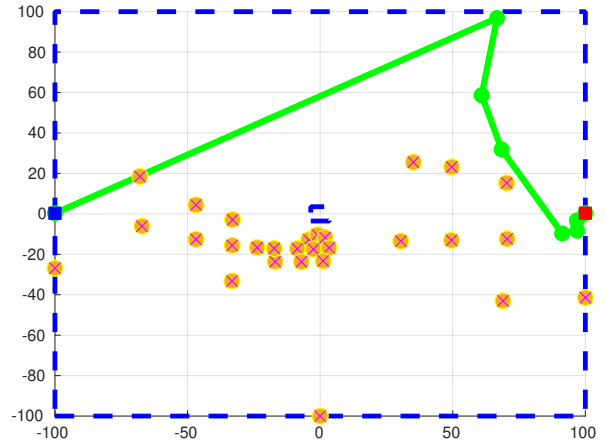
(a) PRM: Path in R-4



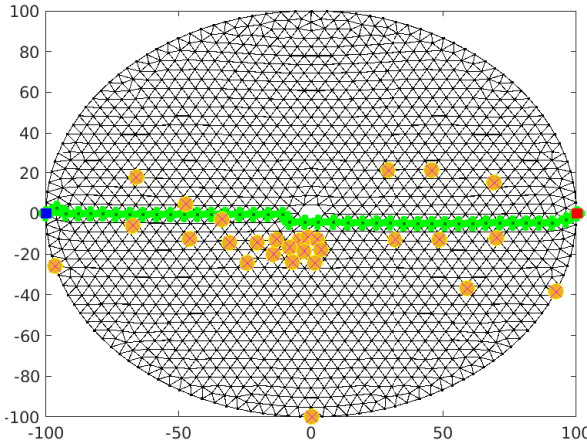
(b) PRM: Path in R-1



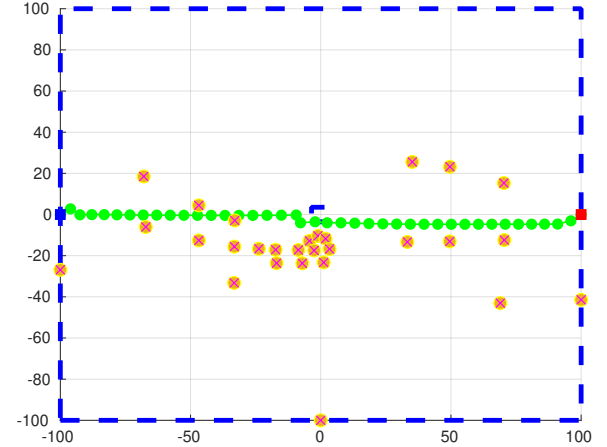
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure A.9: Case 1: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.

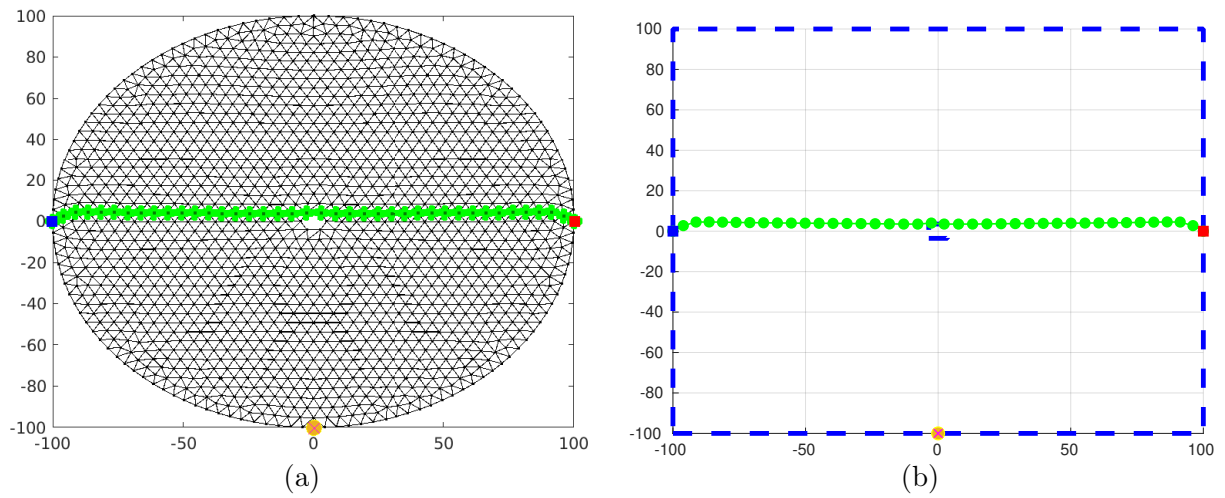


Figure A.10: Case 1: CFDMPP in the case of 1 pole (first group of DCs). Only 1 path-class exists, so blocking is not possible. (a) Path in R-4. (b) Path in R-1.

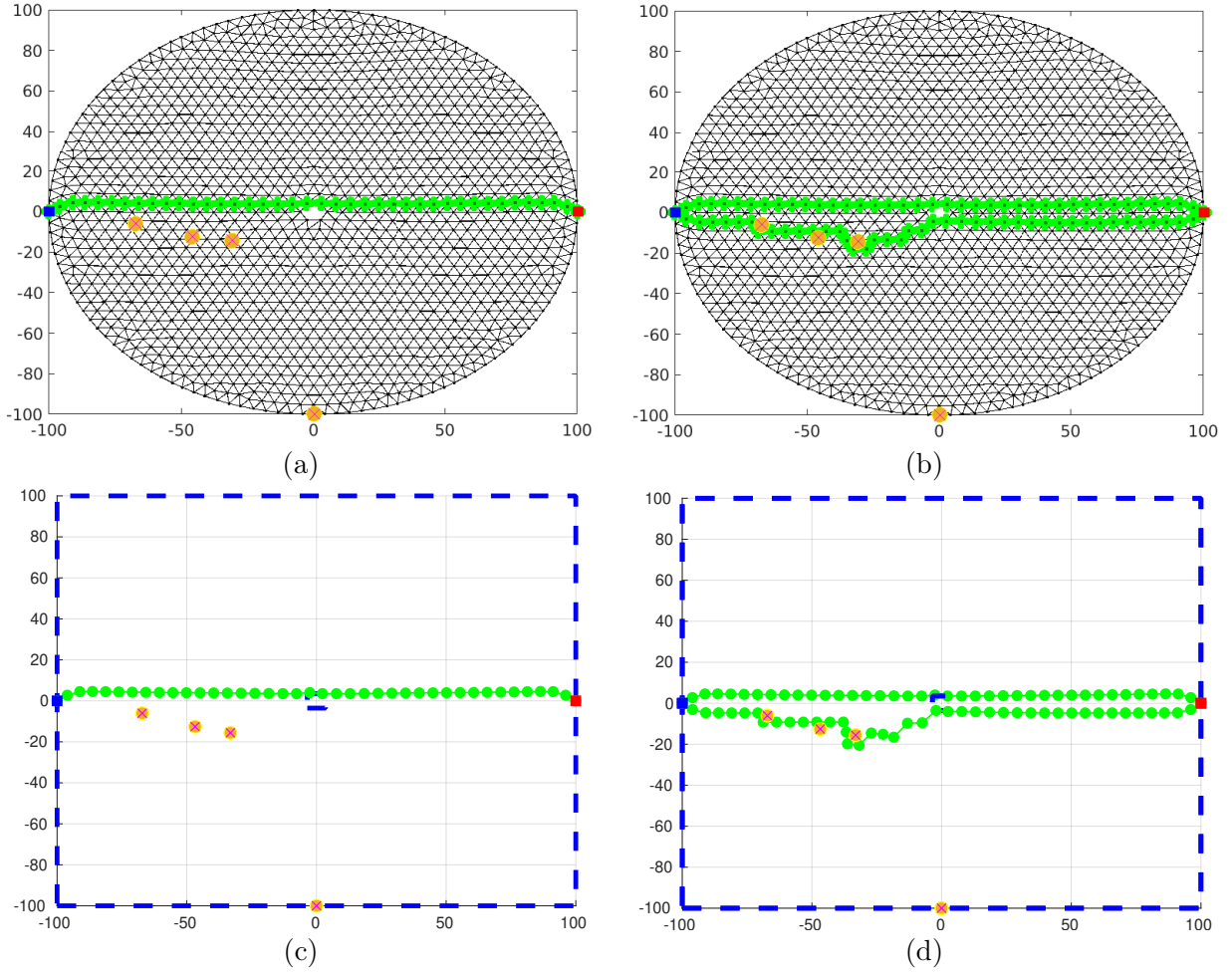


Figure A.11: Case 1: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1.4). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

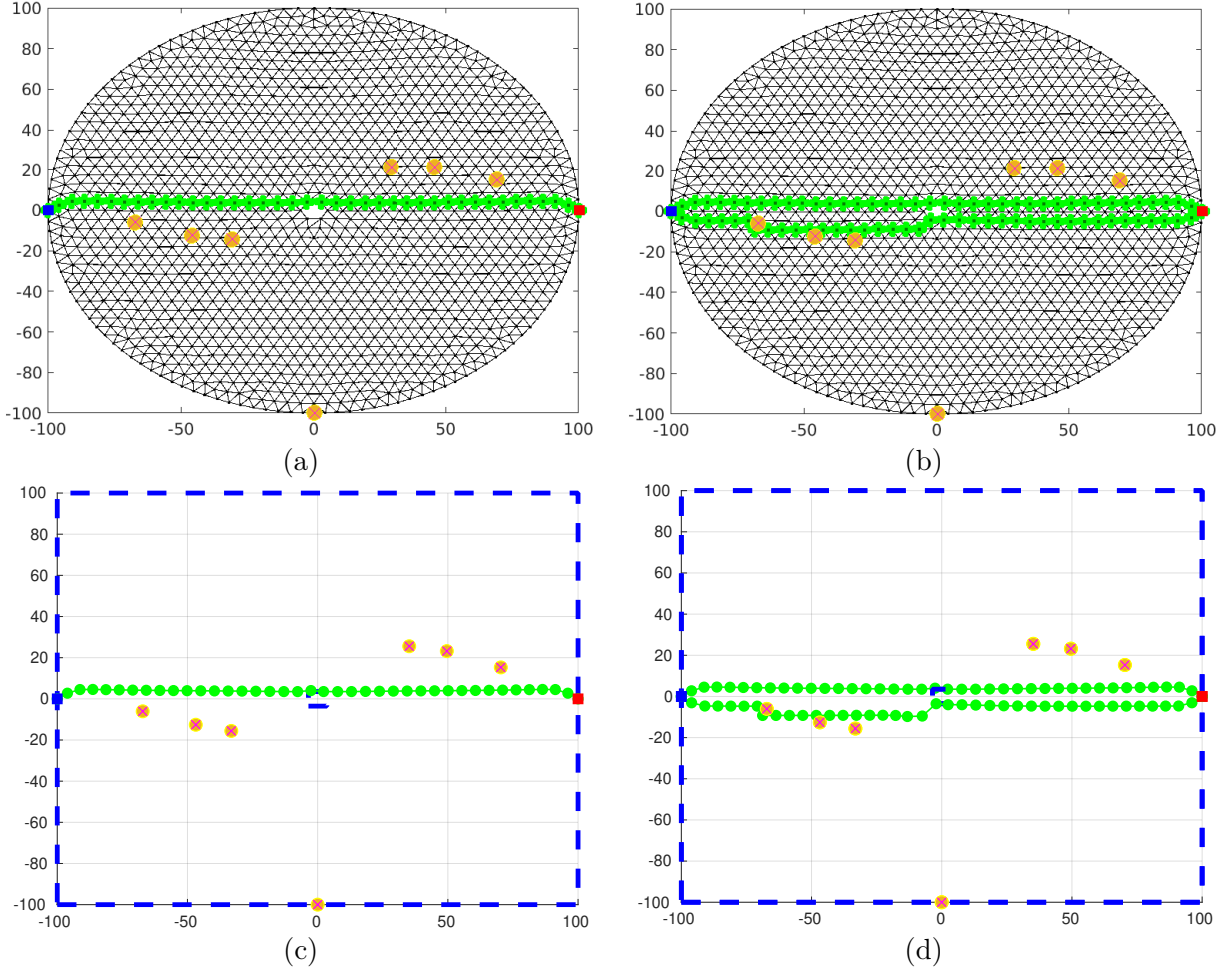


Figure A.12: Case 1: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

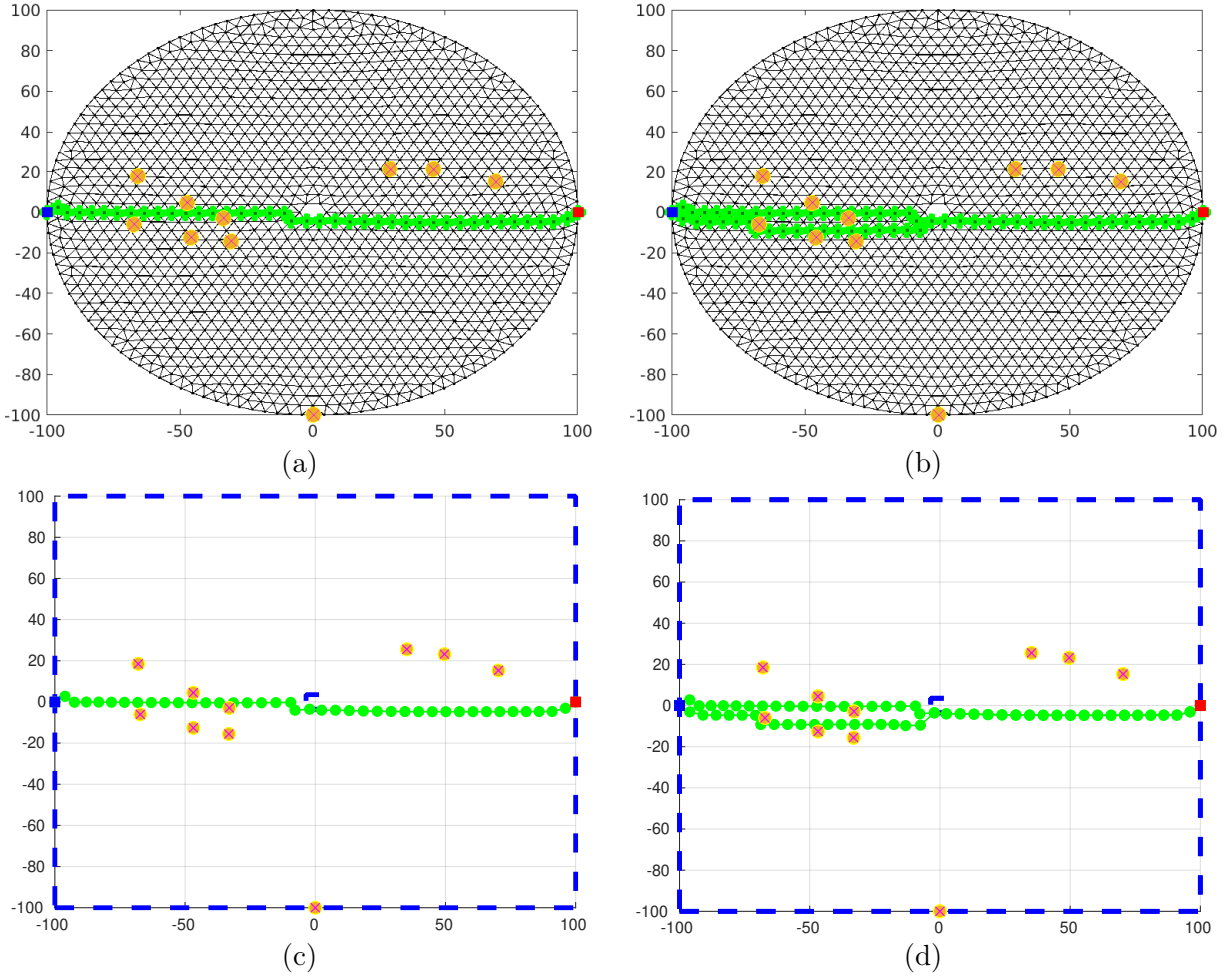


Figure A.13: Case 1: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

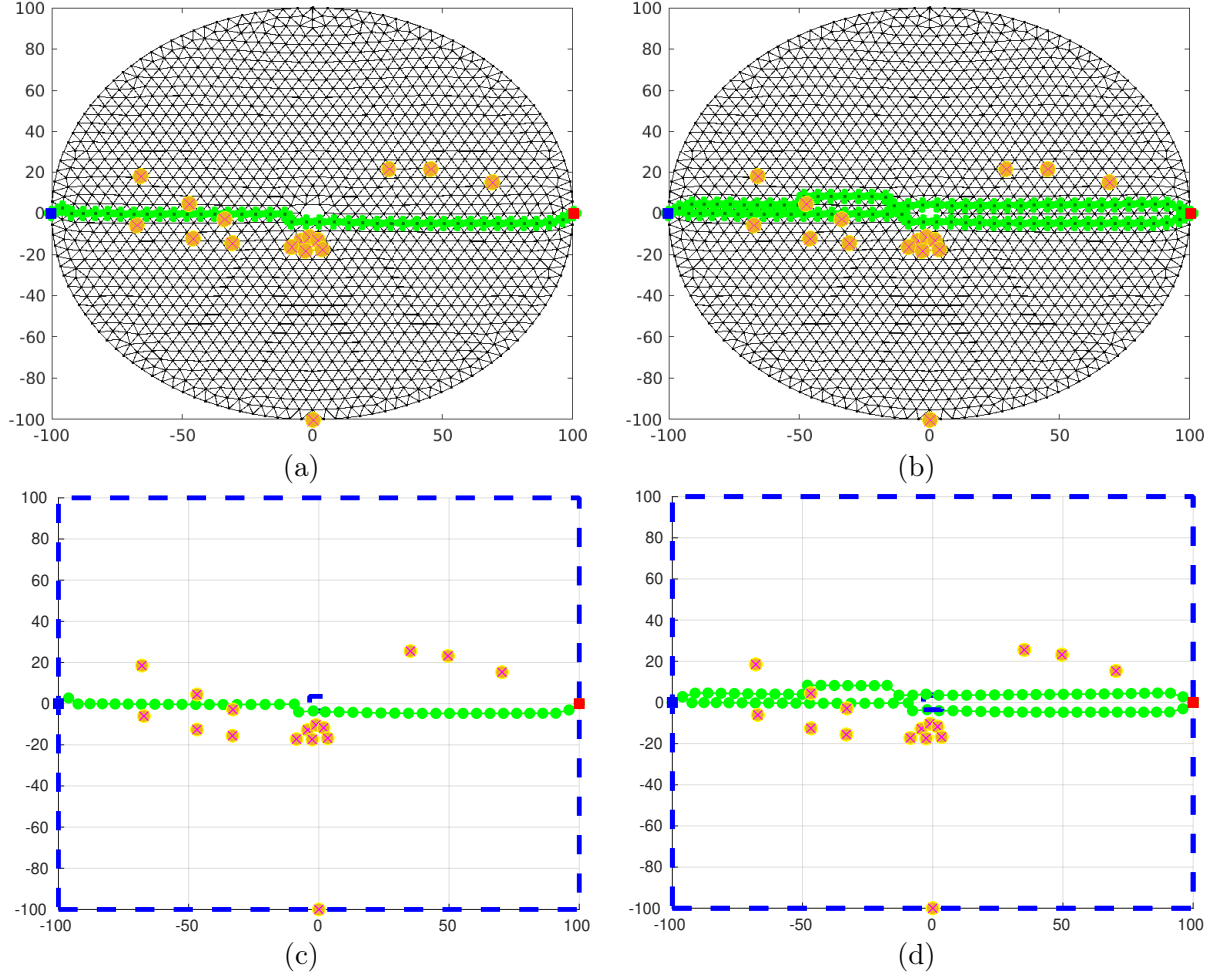


Figure A.14: Case 1: CFDMPP in the case of 16 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

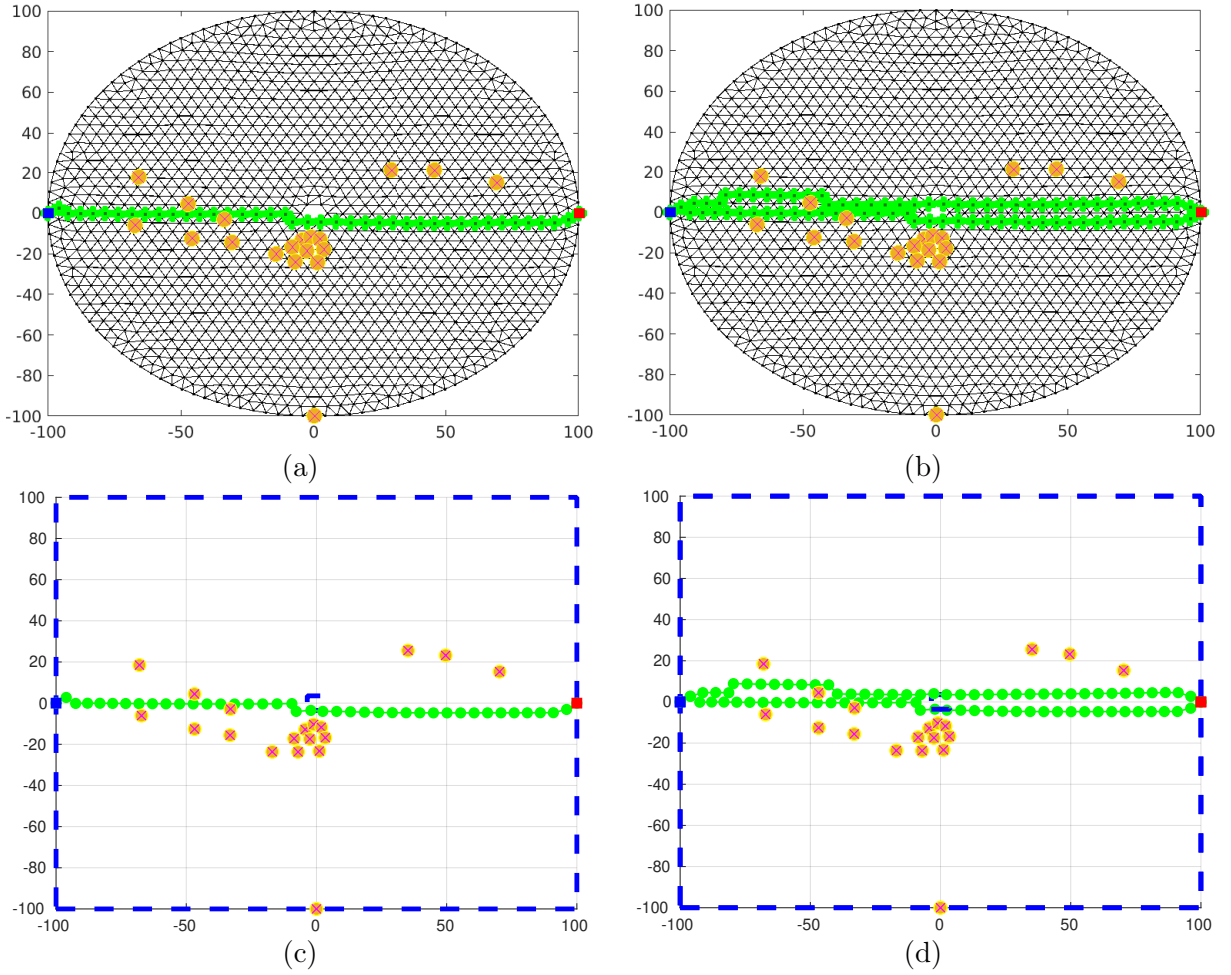


Figure A.15: Case 1: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

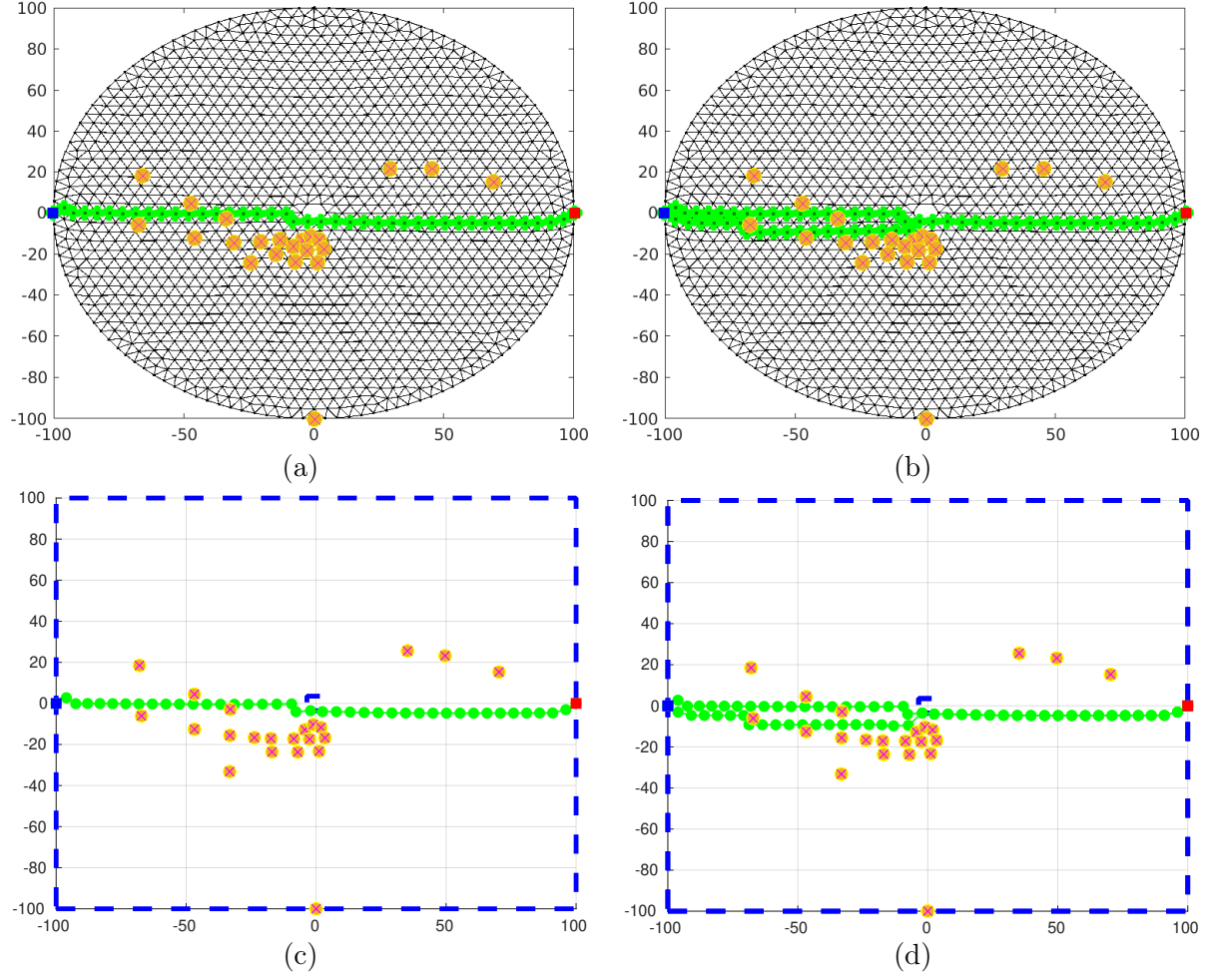


Figure A.16: Case 1: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

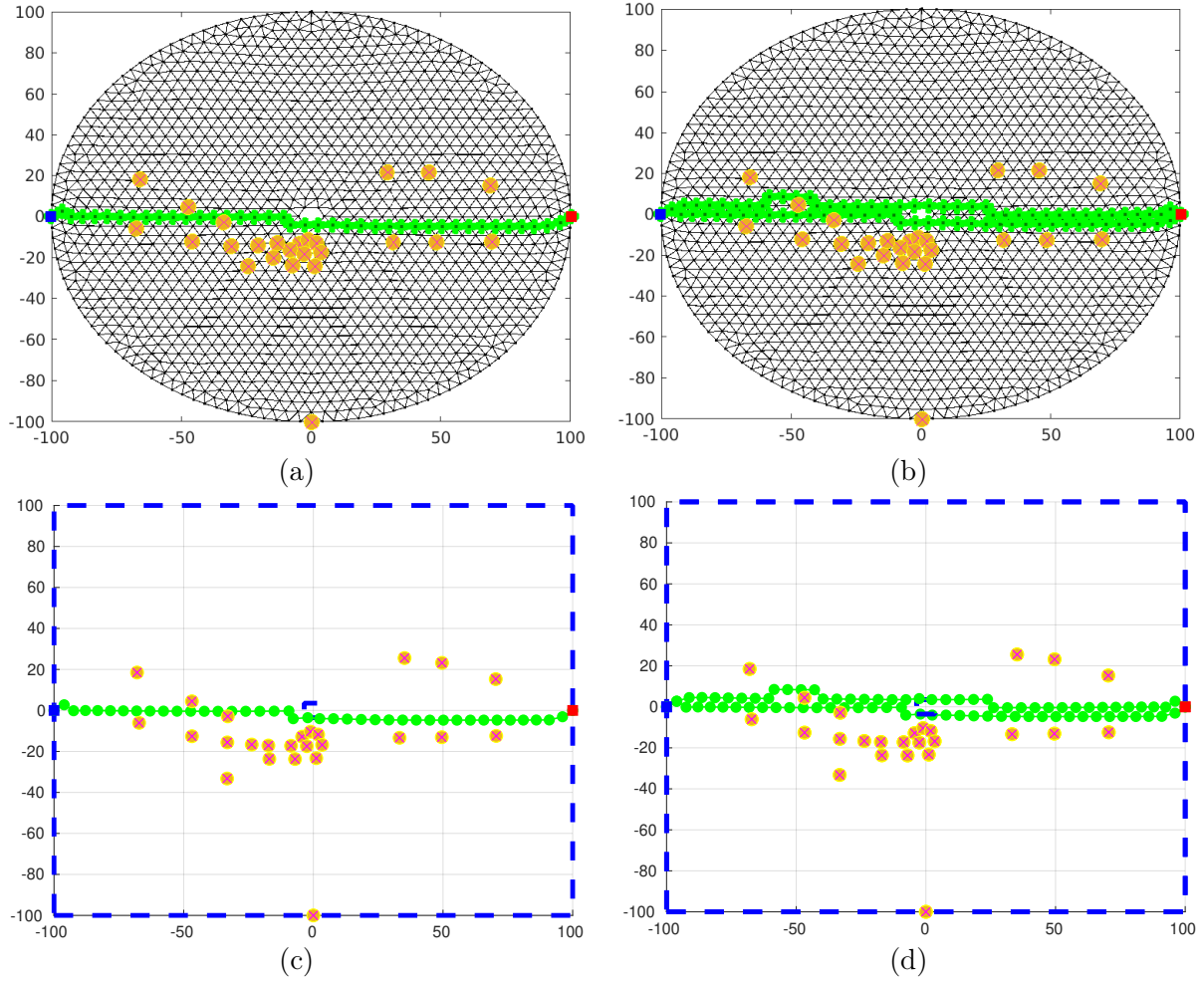


Figure A.17: Case 1: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

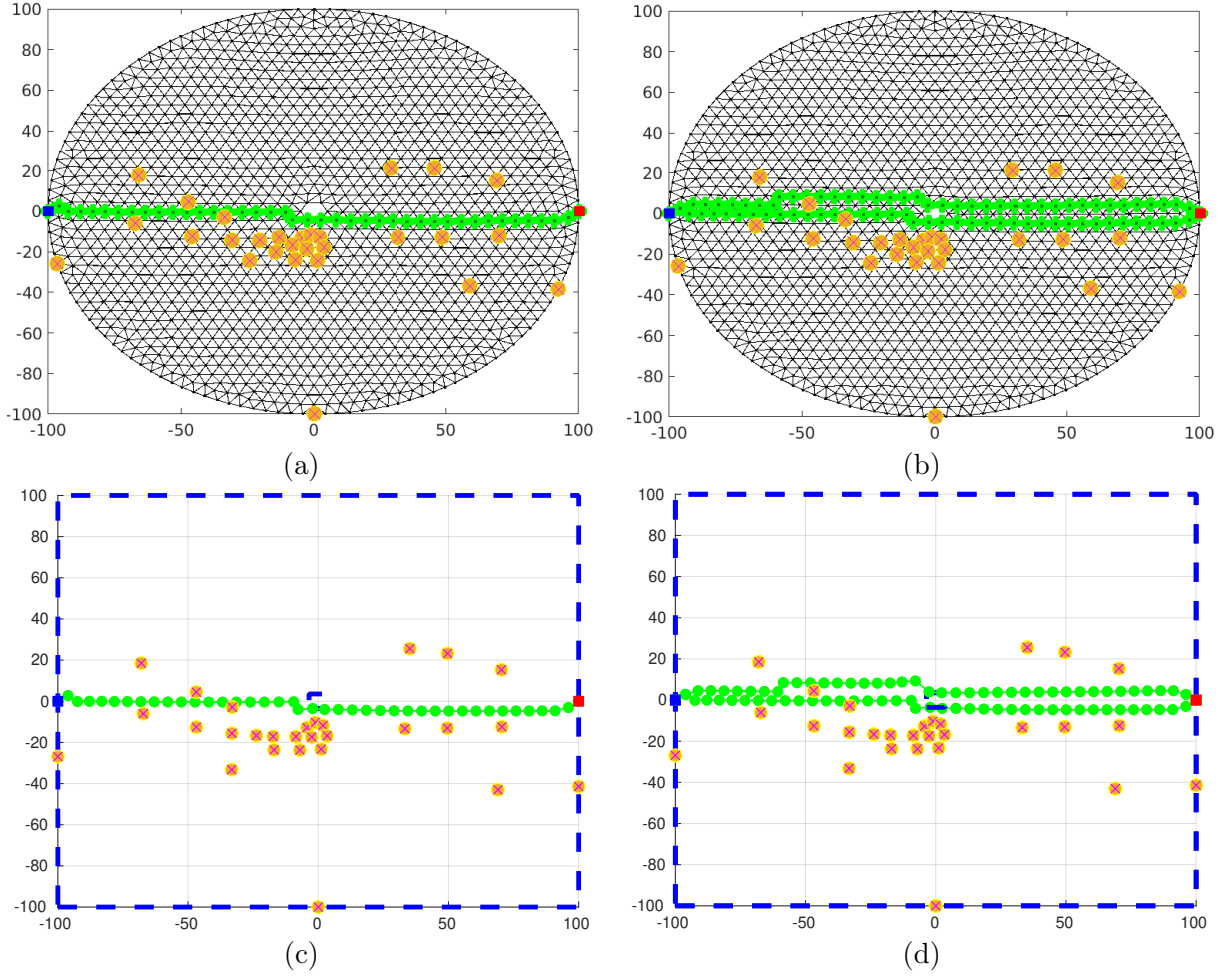


Figure A.18: Case 1: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.45). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

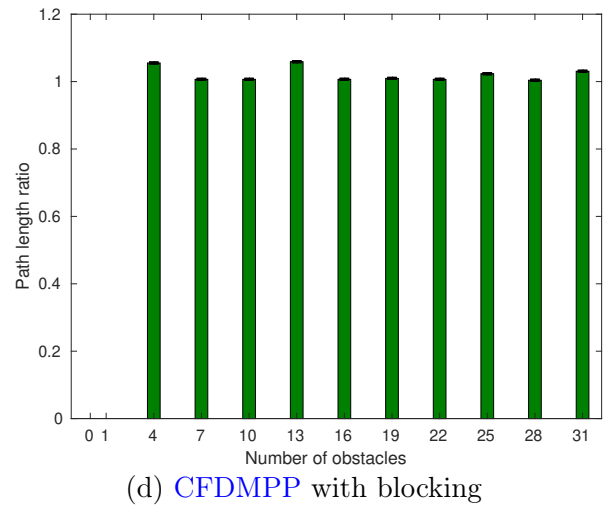
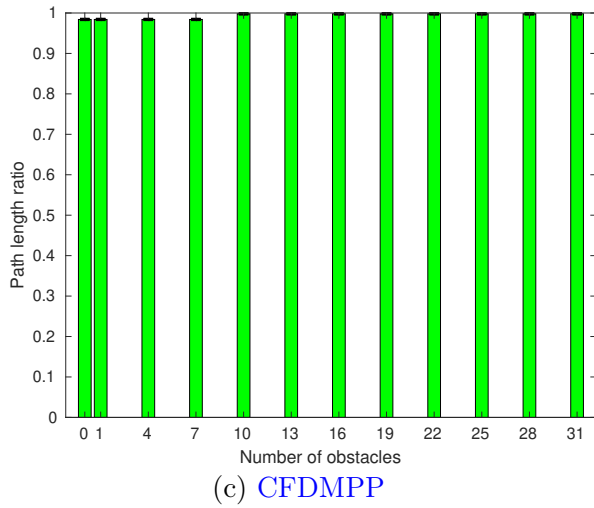
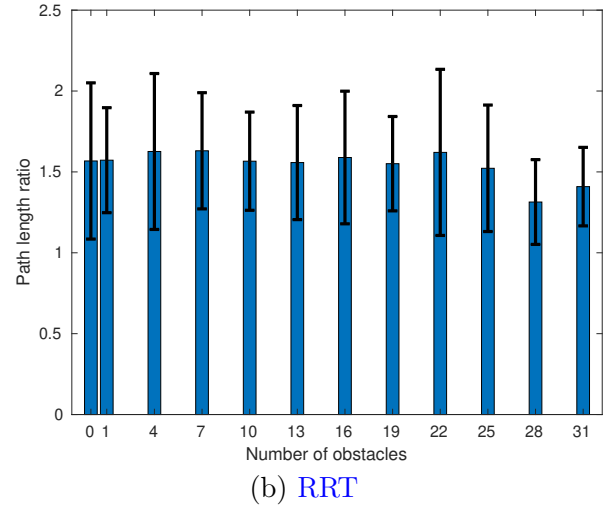
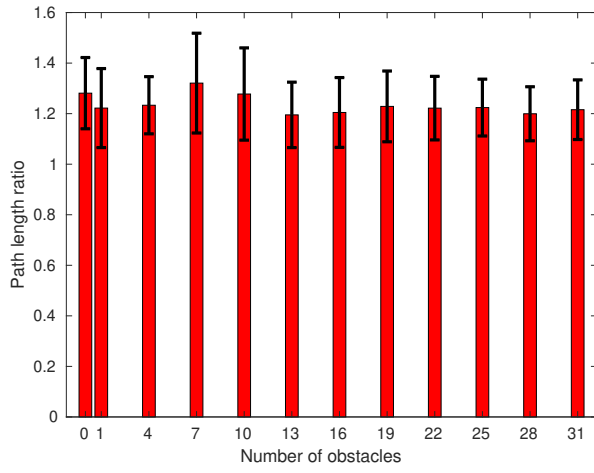
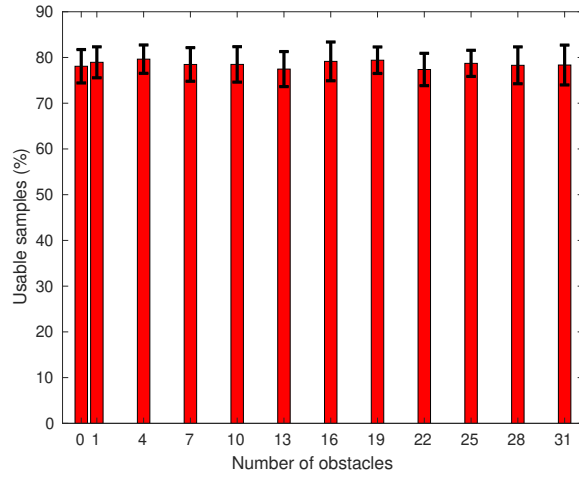
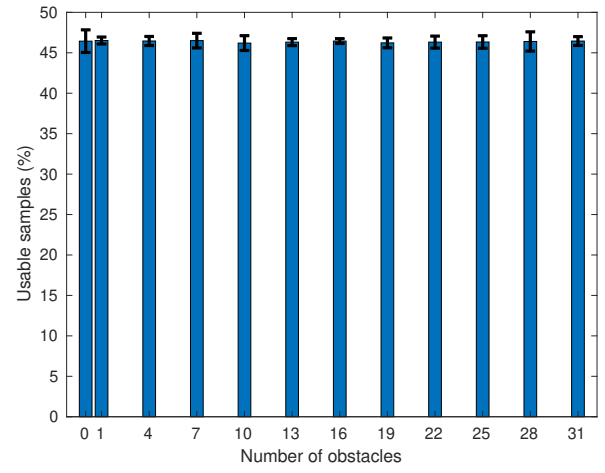


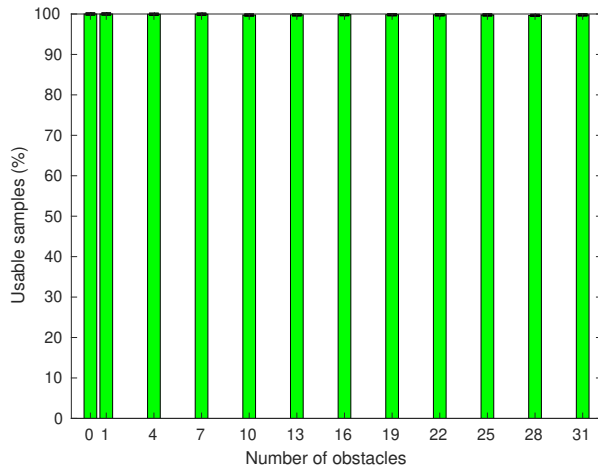
Figure A.19: Case 1: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.



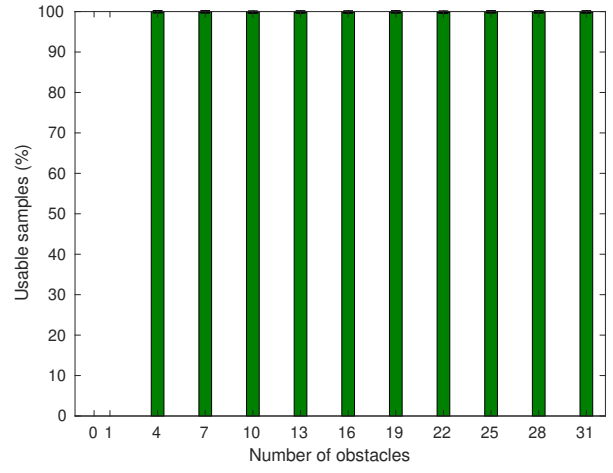
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

Figure A.20: Case 1: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.

Appendix B

Stage 2: Case 2

This section displays those results from Section 4.2 corresponding to case 2, which were not originally shown in Section 4.2. The following results are displayed here:

1. The paths produced by [PRM](#), [RRT](#) and [CFDMPP](#) in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
2. The paths produced by [CFDMPP](#) when the original path is homotopically blocked; displayed in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
3. Plots displaying trends of a metric for [PRM](#), [RRT](#) and [CFDMPP](#) (without and with blocking). The two metrics of path length ratio and percentage of usable samples that were not shown in Section 4.2, are shown here for [0,1,4,..31] poles.

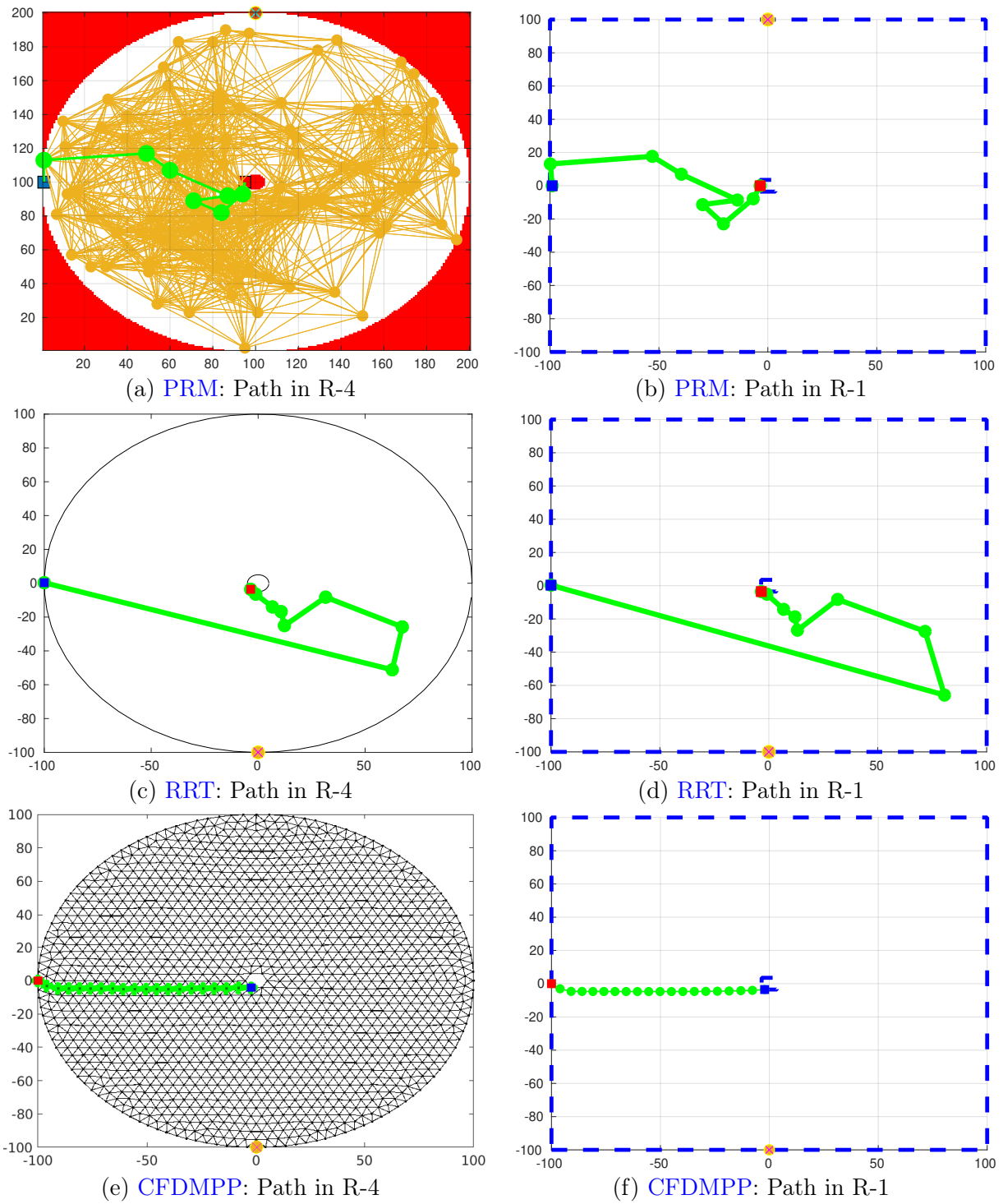


Figure B.1: Case 2: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.

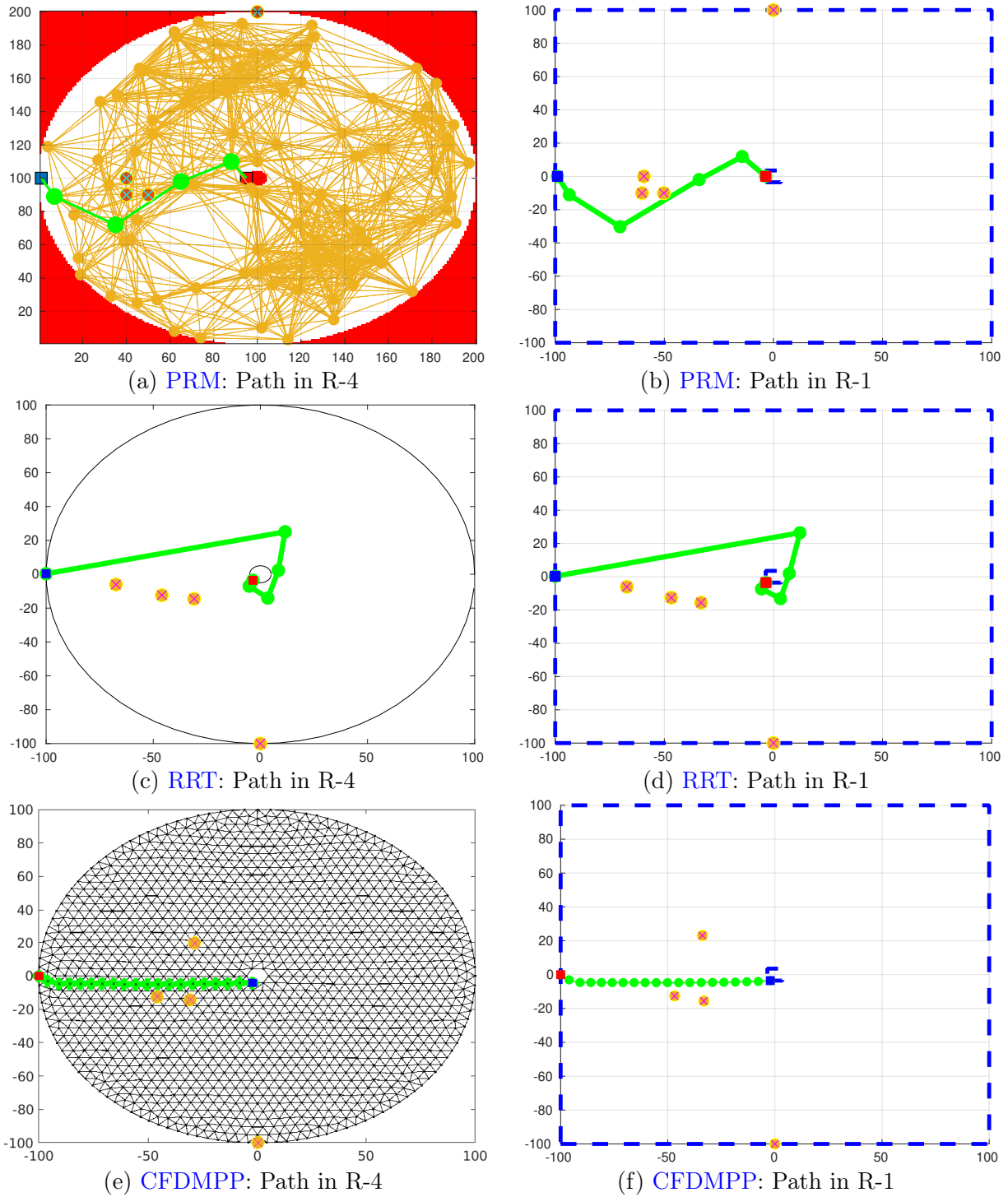
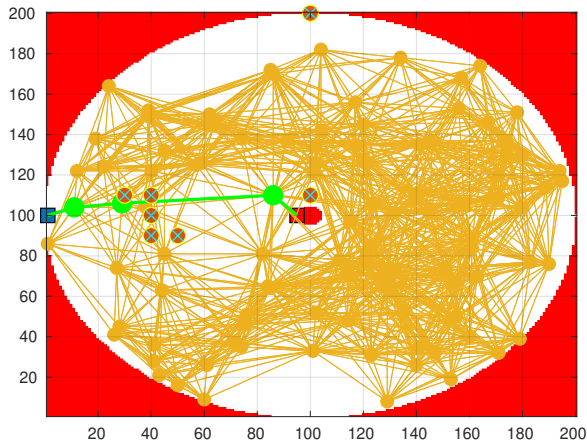
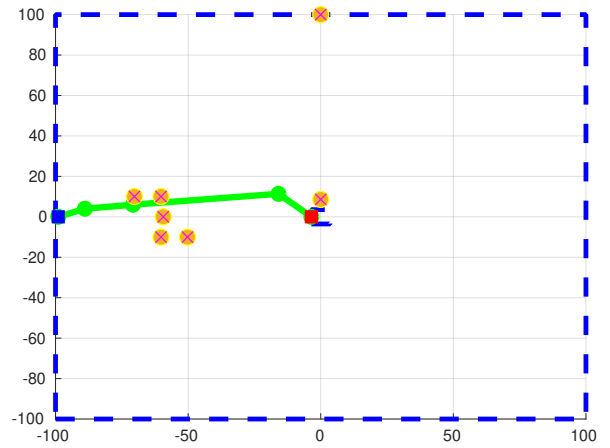


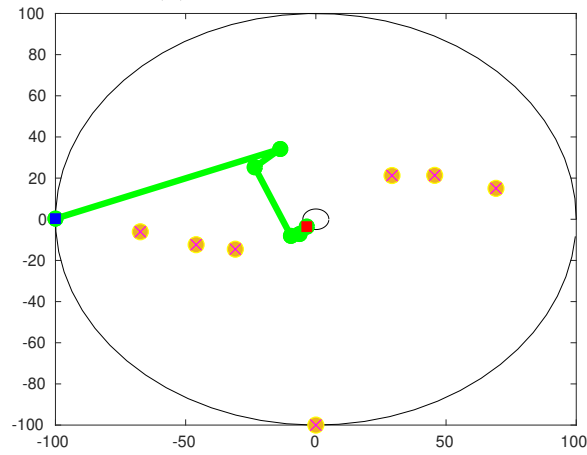
Figure B.2: Case 2: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.



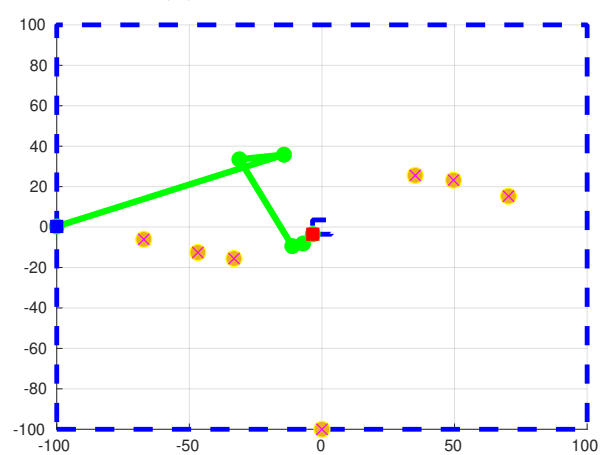
(a) PRM: Path in R-4



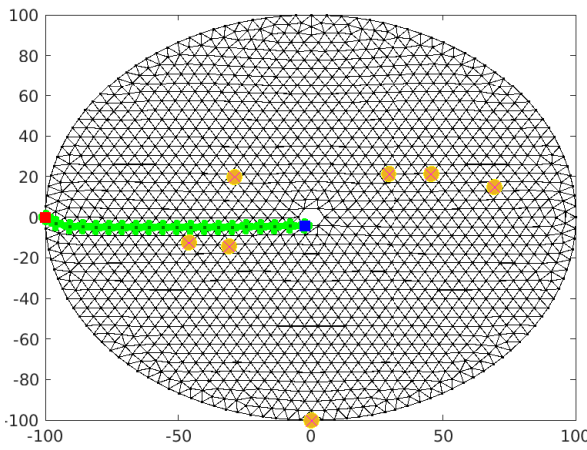
(b) PRM: Path in R-1



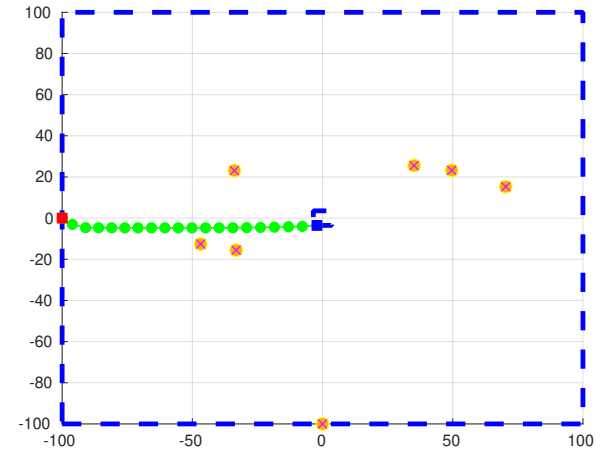
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure B.3: Case 2: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.

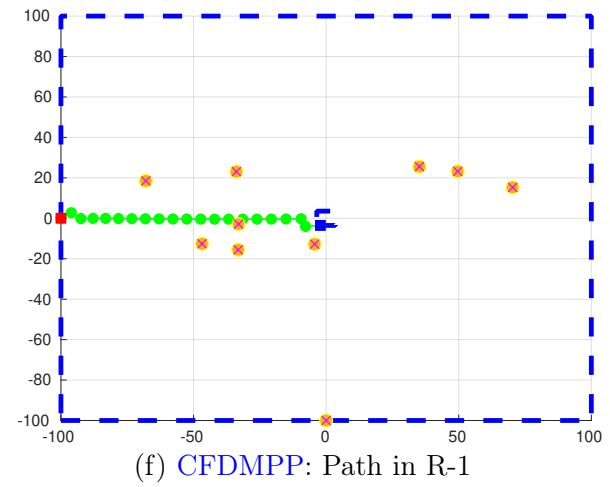
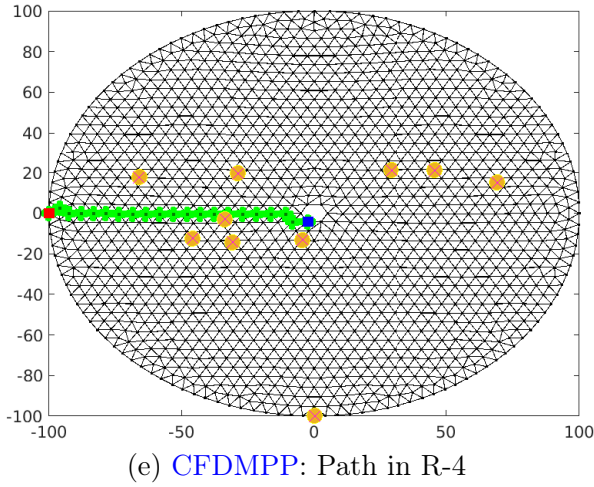
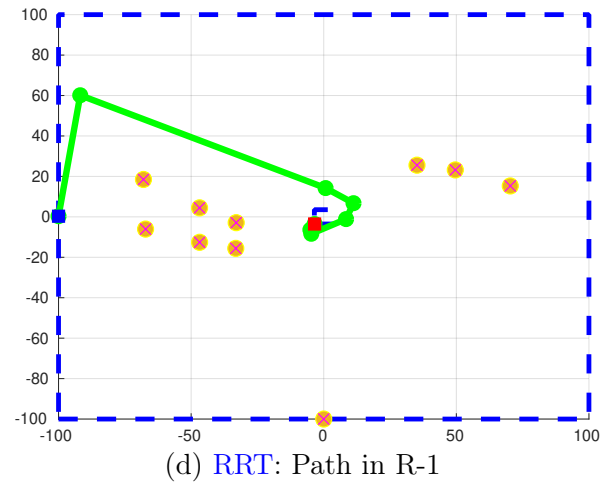
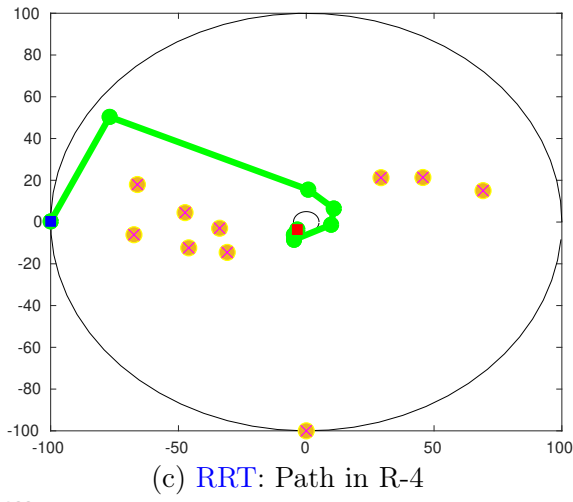
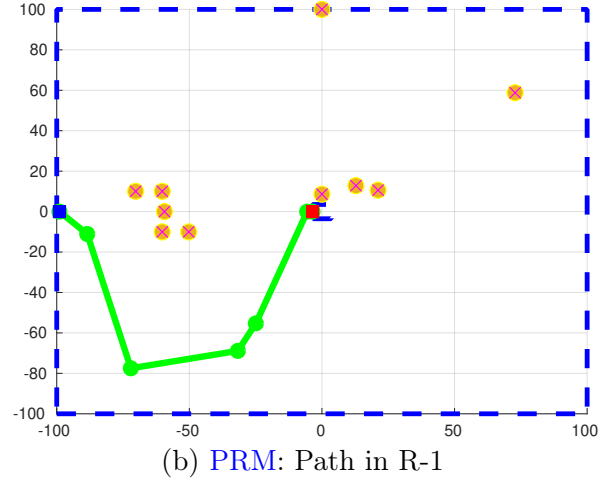
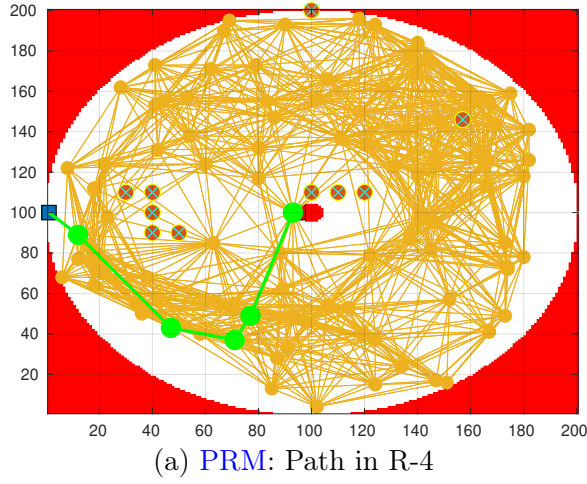
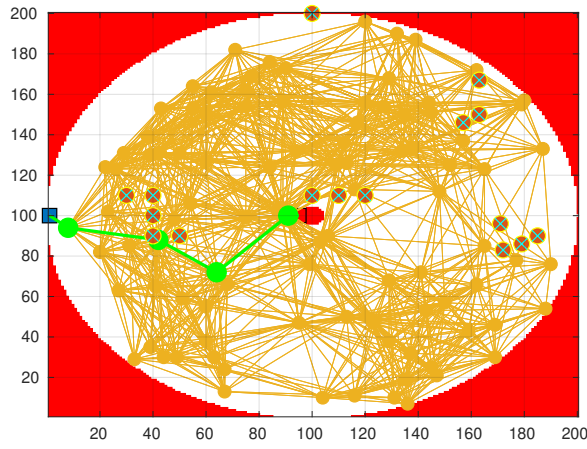


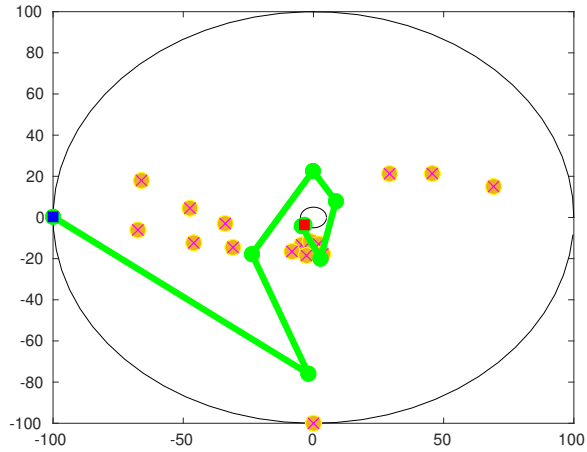
Figure B.4: Case 2: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.



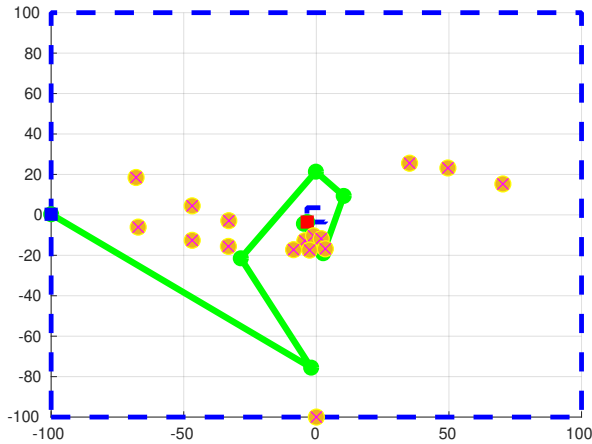
(a) PRM: Path in R-4



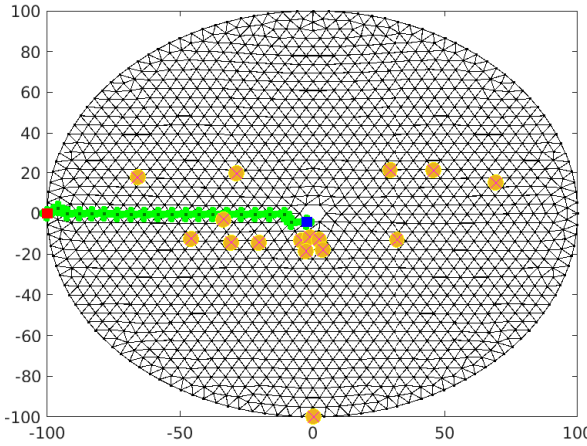
(b) PRM: Path in R-1



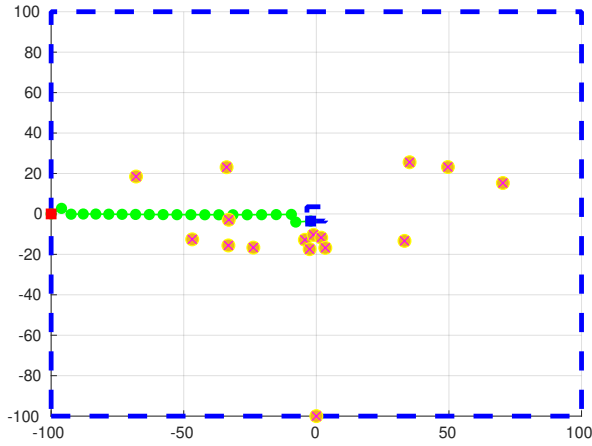
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure B.5: Case 2: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.

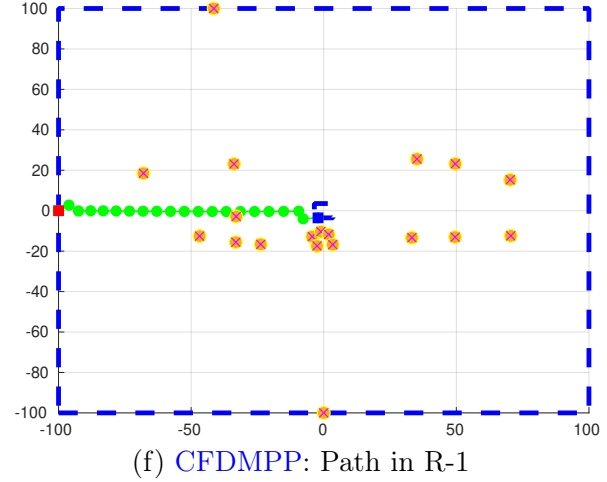
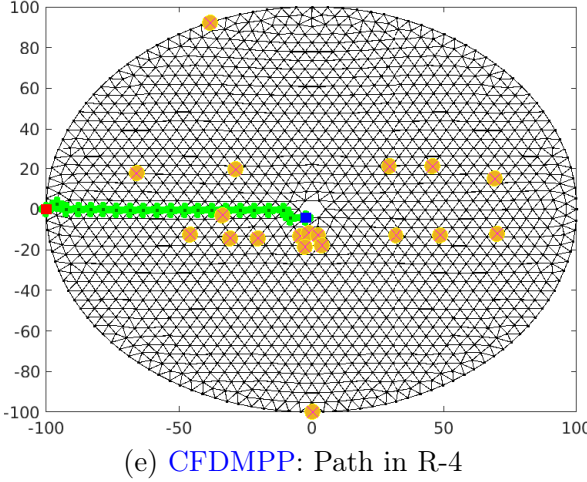
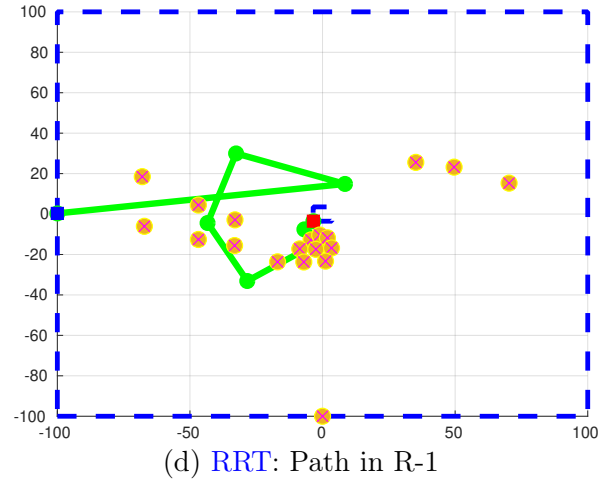
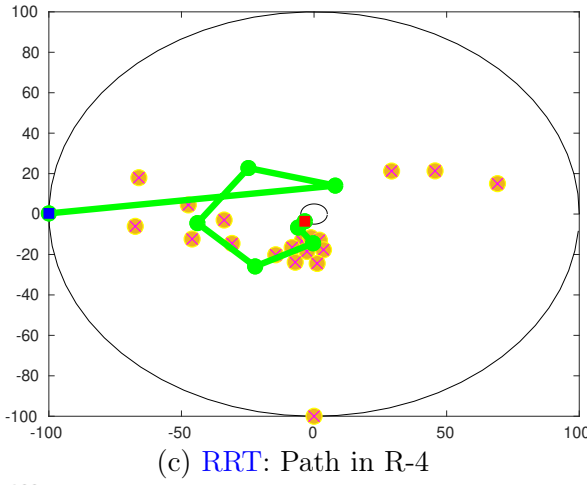
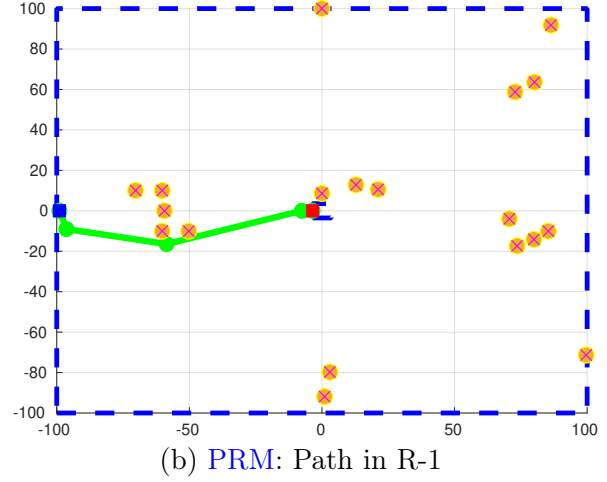
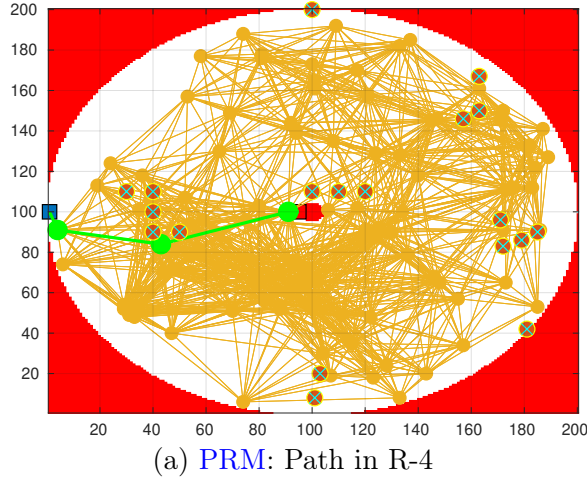
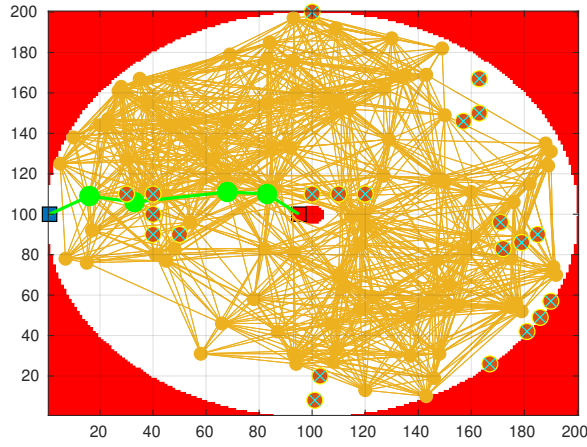
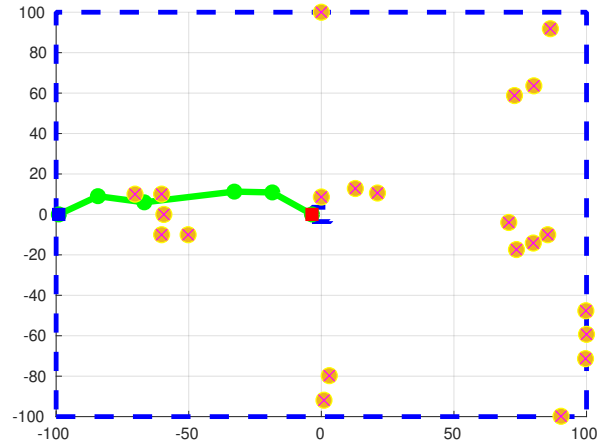


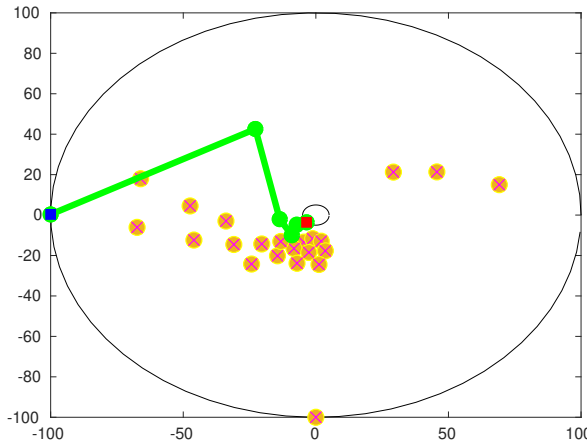
Figure B.6: Case 2: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.



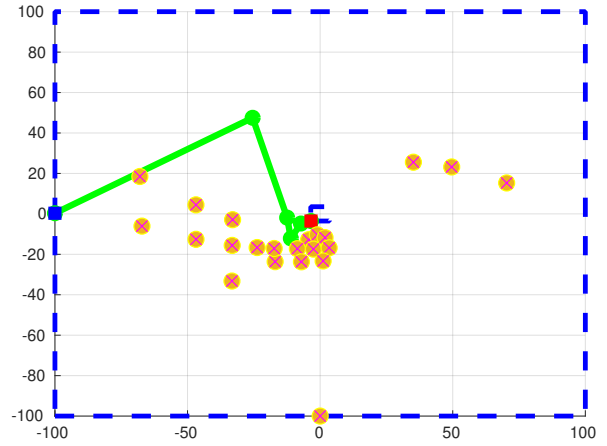
(a) PRM: Path in R-4



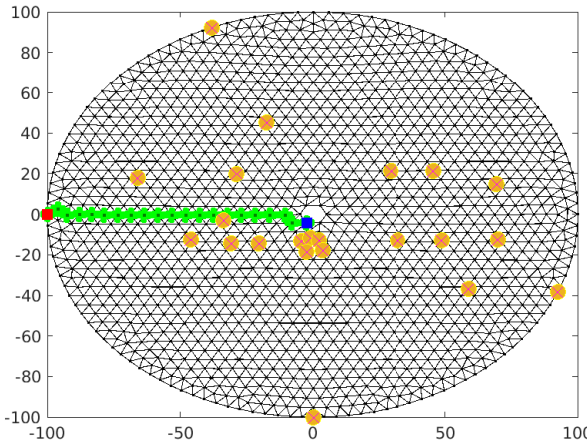
(b) PRM: Path in R-1



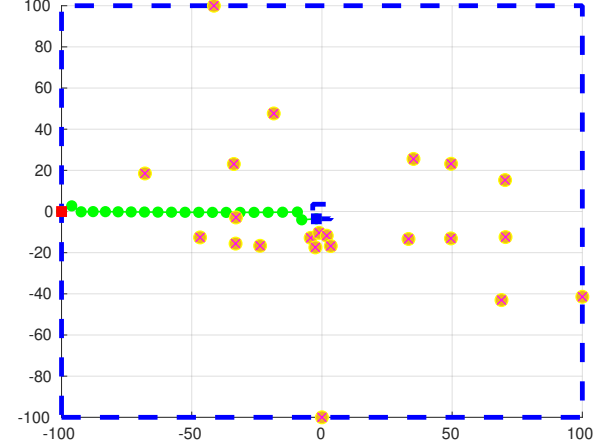
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure B.7: Case 2: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.

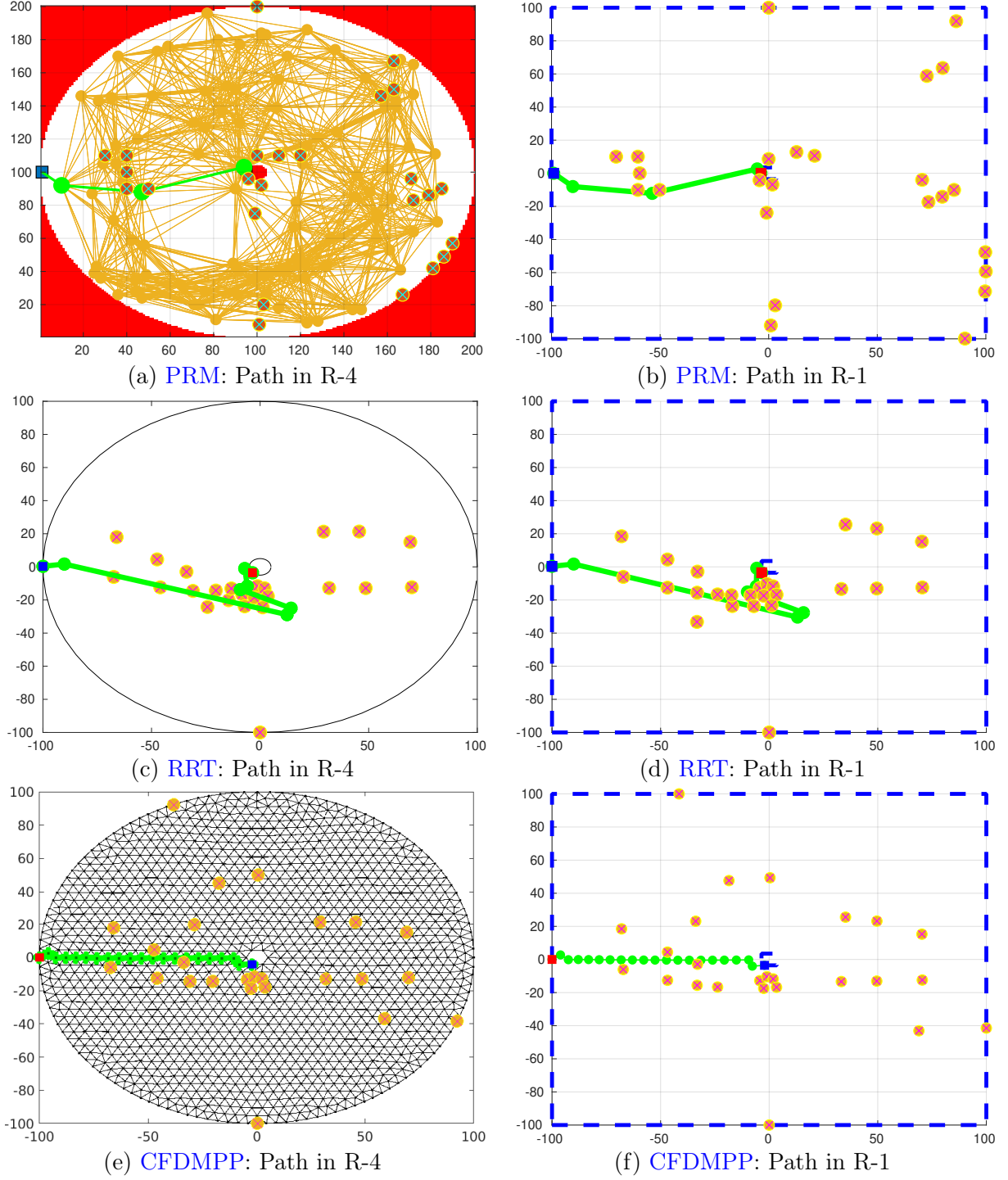
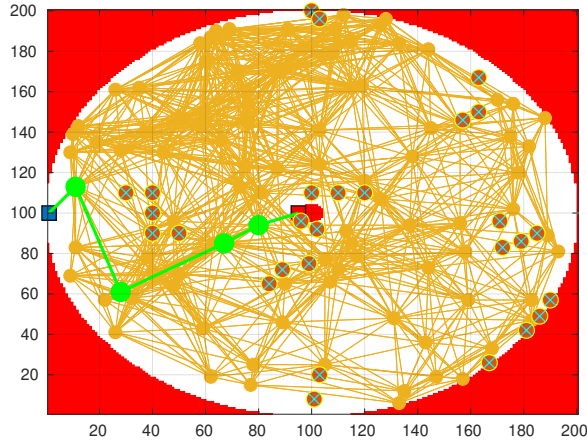
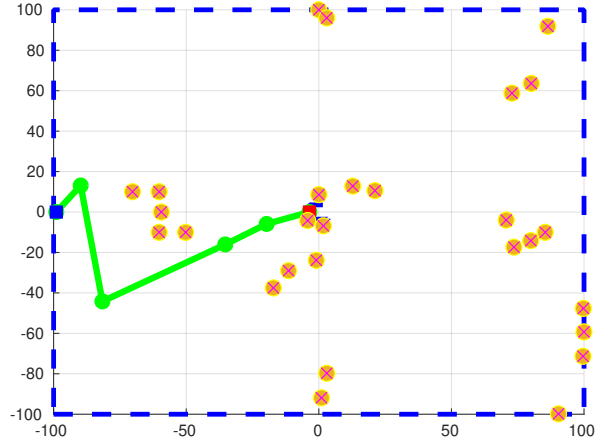


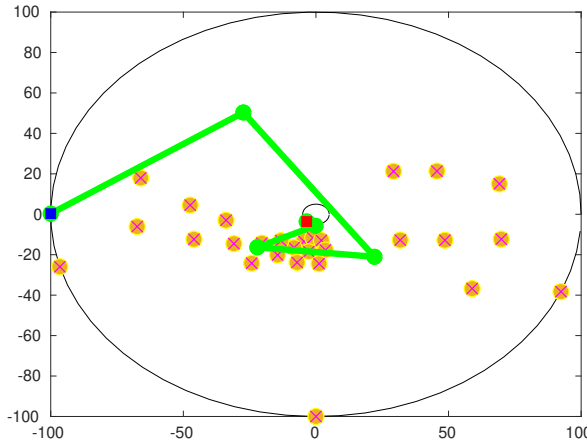
Figure B.8: Case 2: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.



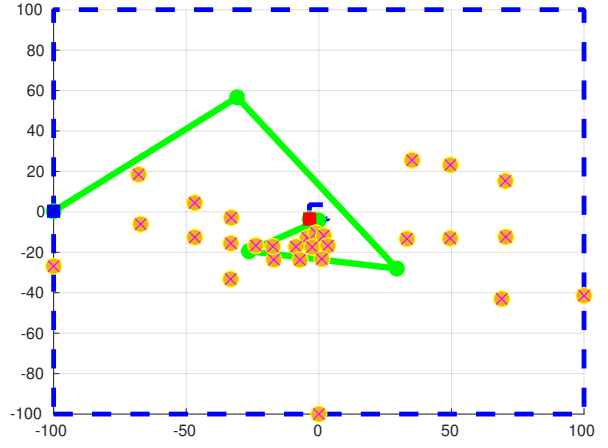
(a) PRM: Path in R-4



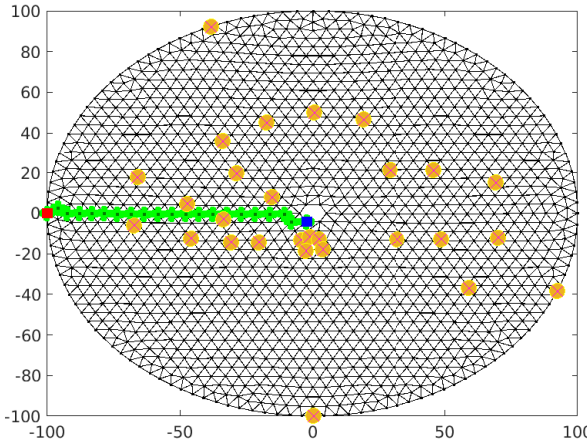
(b) PRM: Path in R-1



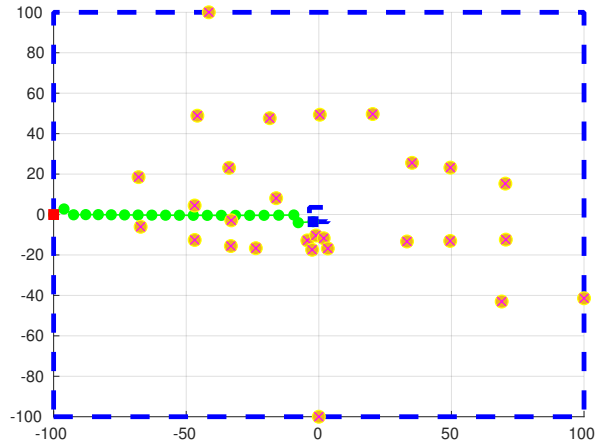
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure B.9: Case 2: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.

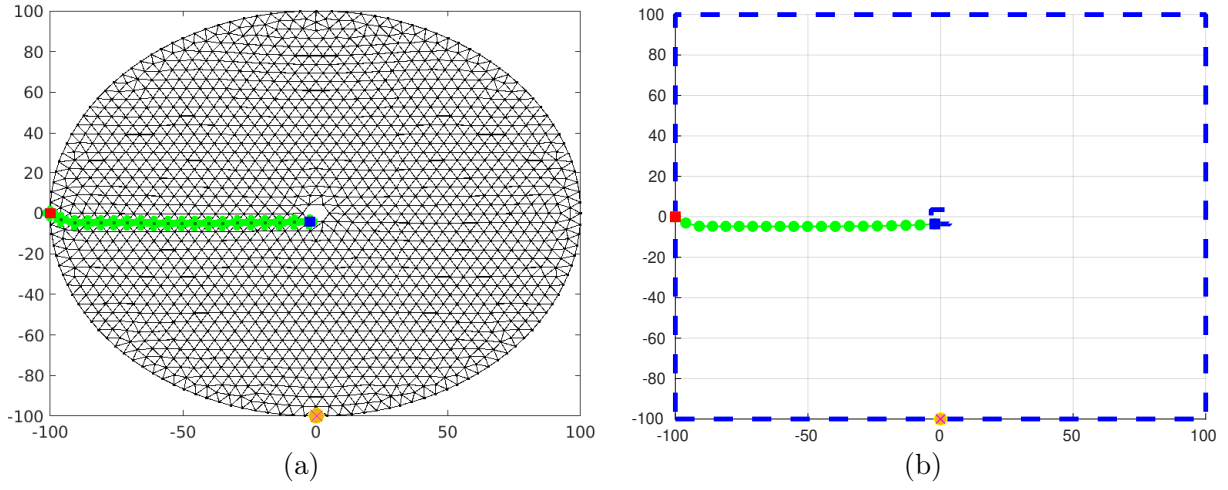


Figure B.10: Case 2: CFDMPP in the case of 1 poles (first group of DCs). Due to the pole's location, only 1 path class is possible. (a) Path in R-4. (b) Path in R-1.

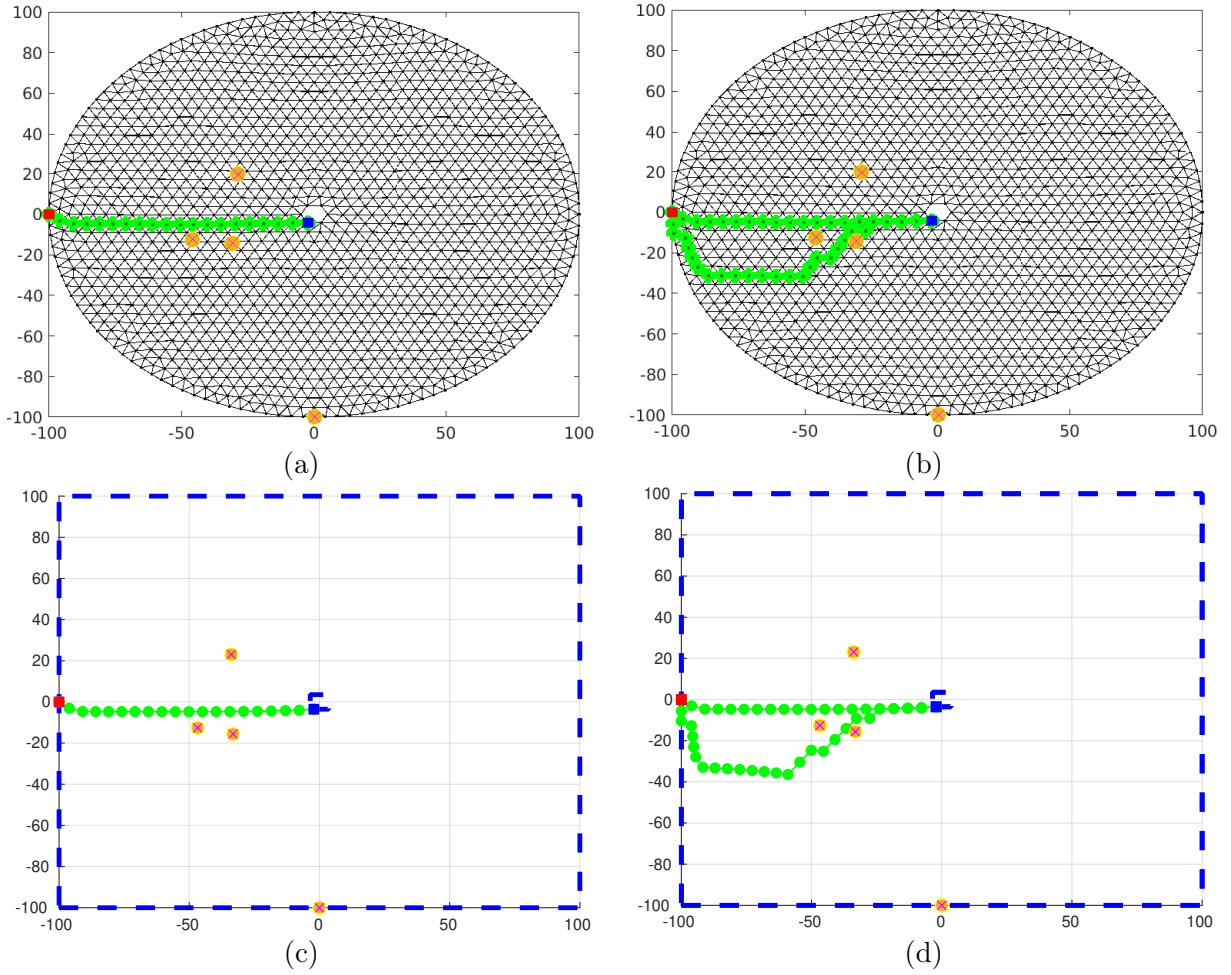


Figure B.11: Case 2: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.425). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

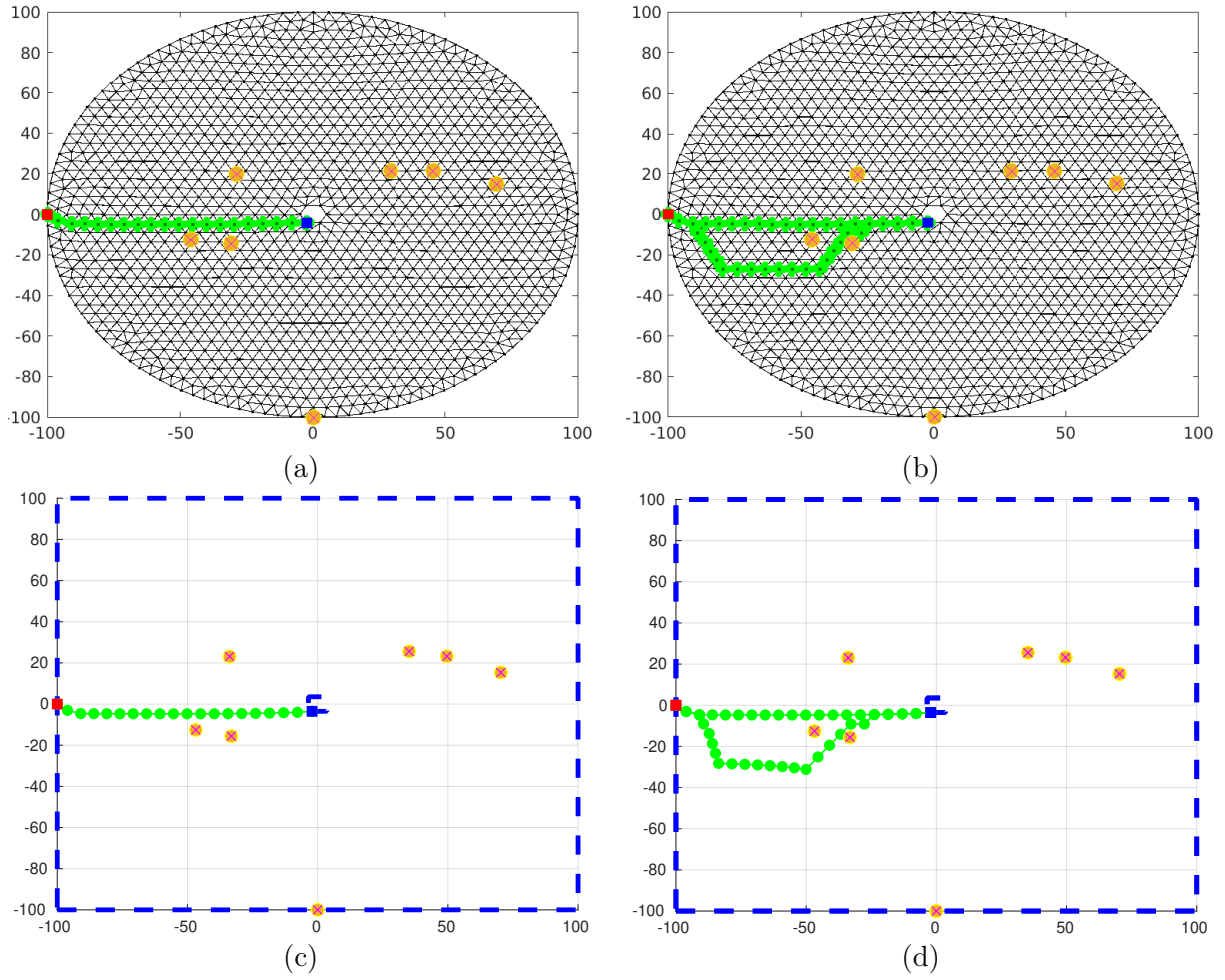


Figure B.12: Case 2: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.85). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

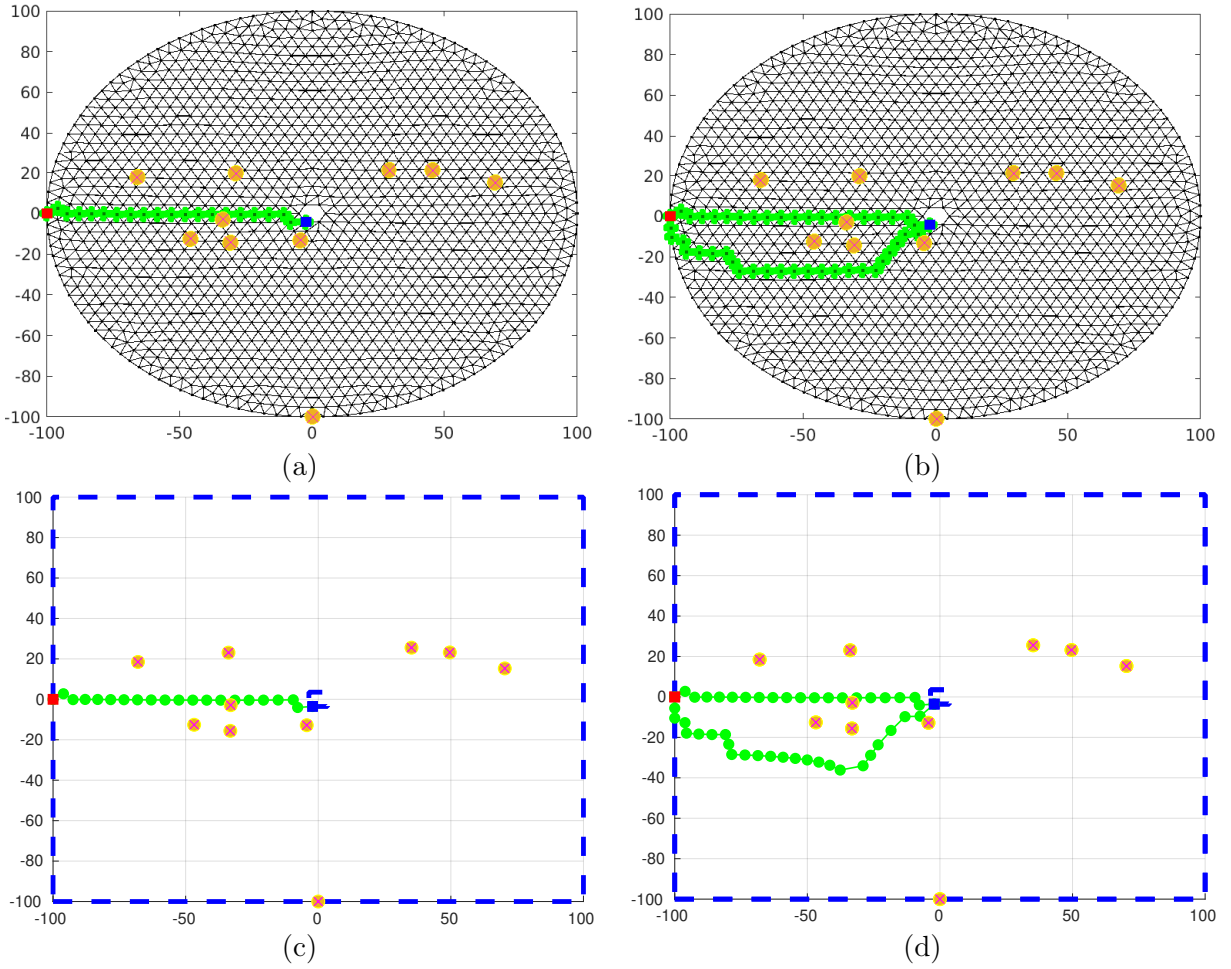


Figure B.13: Case 2: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.85). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

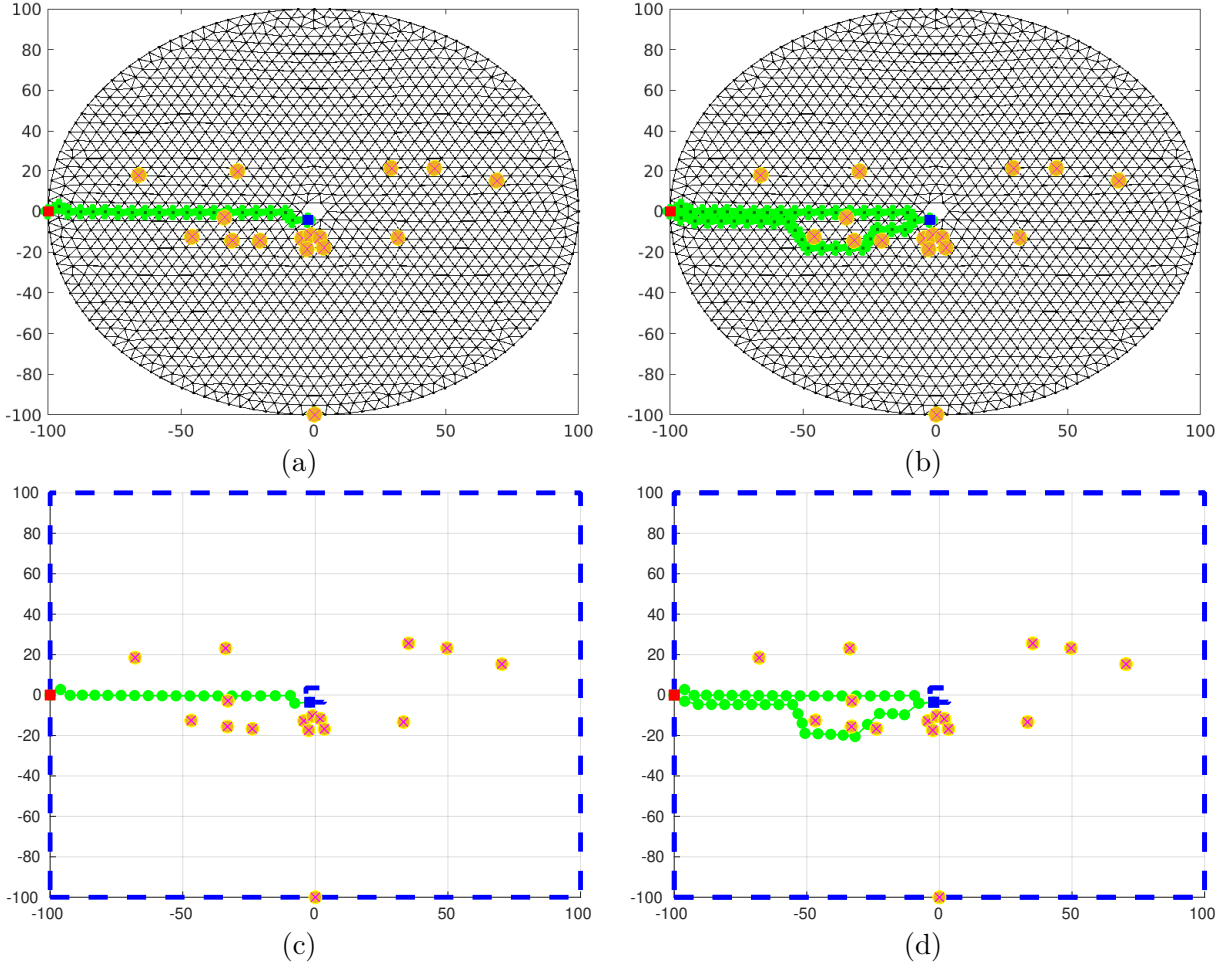


Figure B.14: Case 2: CFDMPP in the case of 16 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

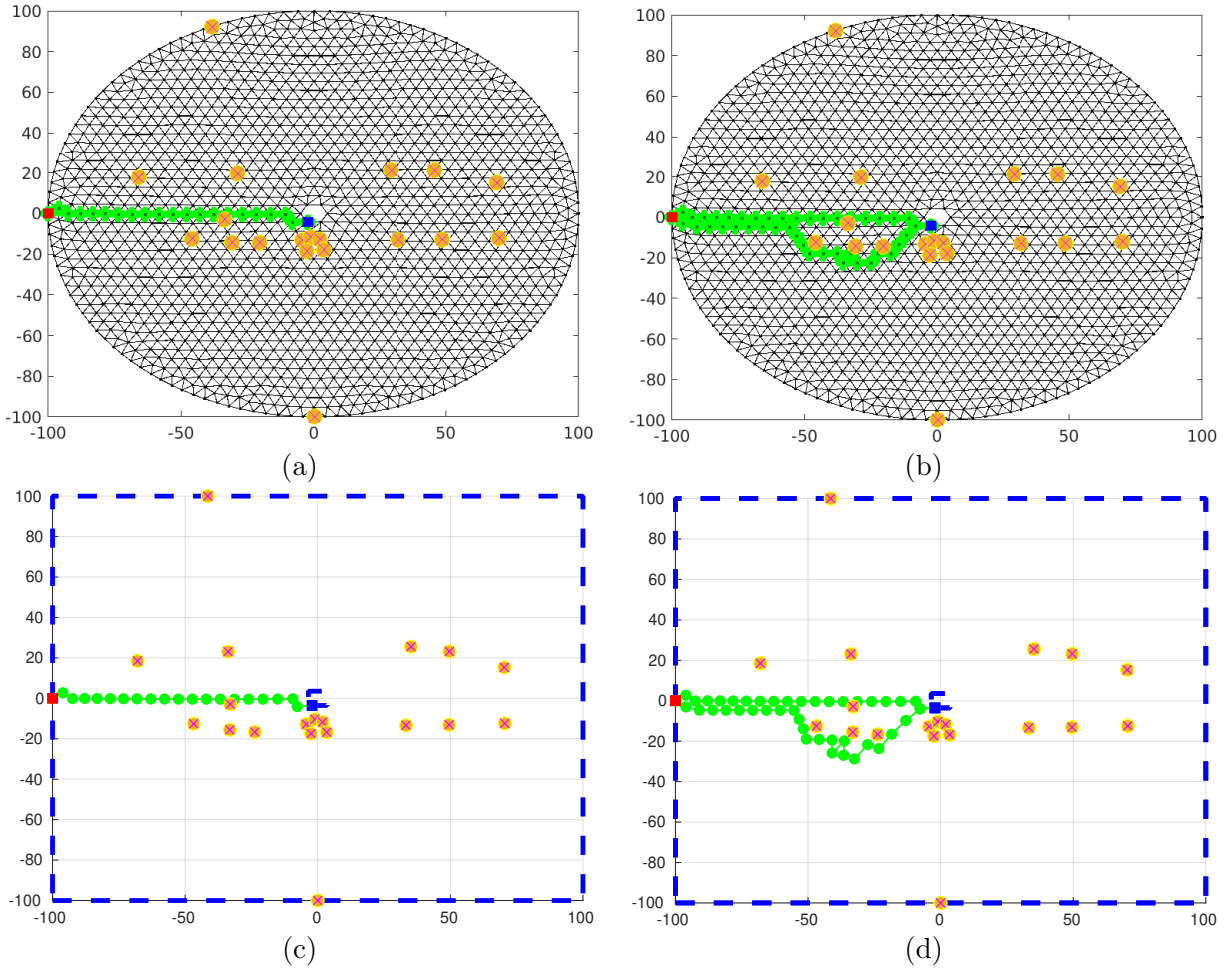


Figure B.15: Case 2: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1.15). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

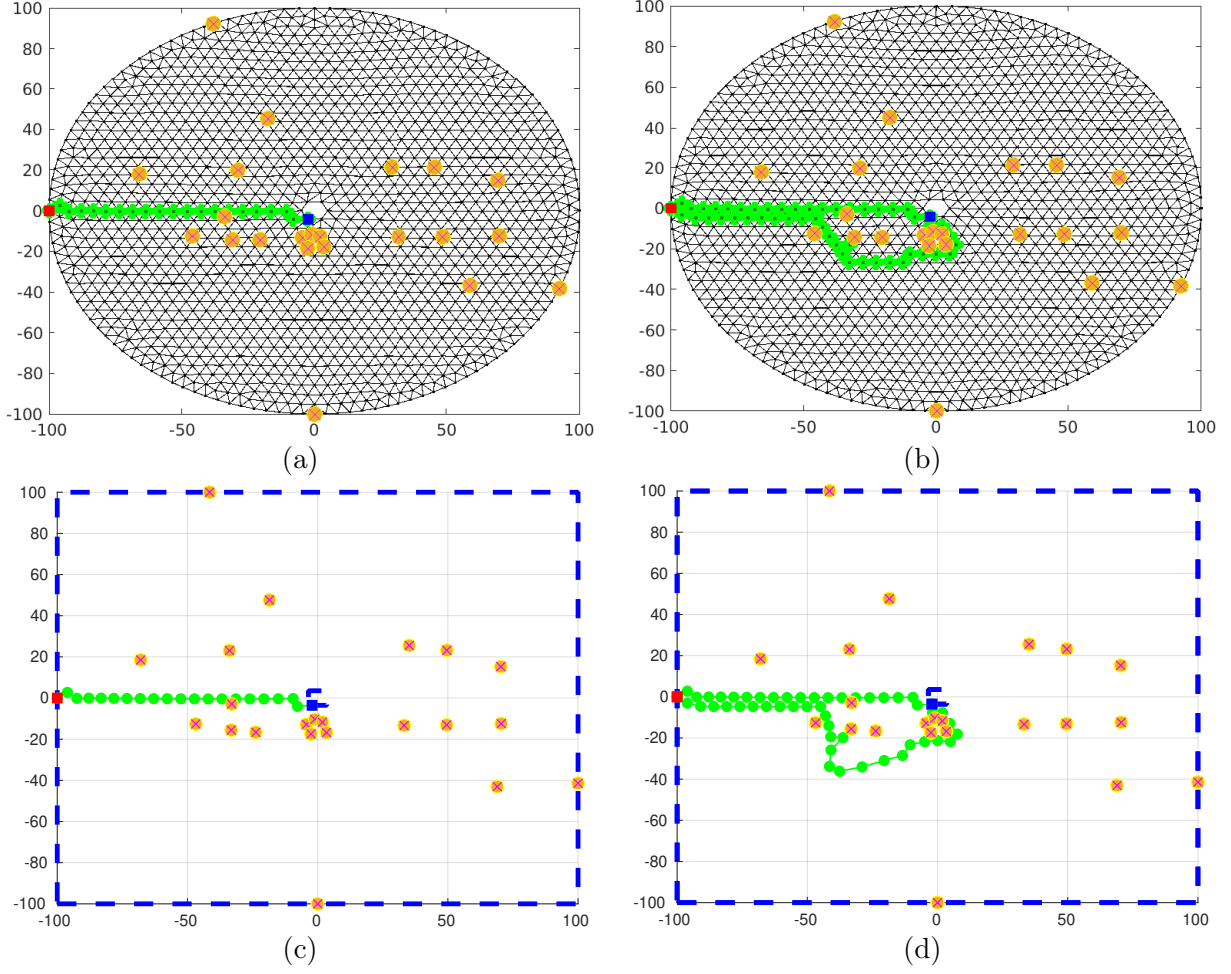


Figure B.16: Case 2: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

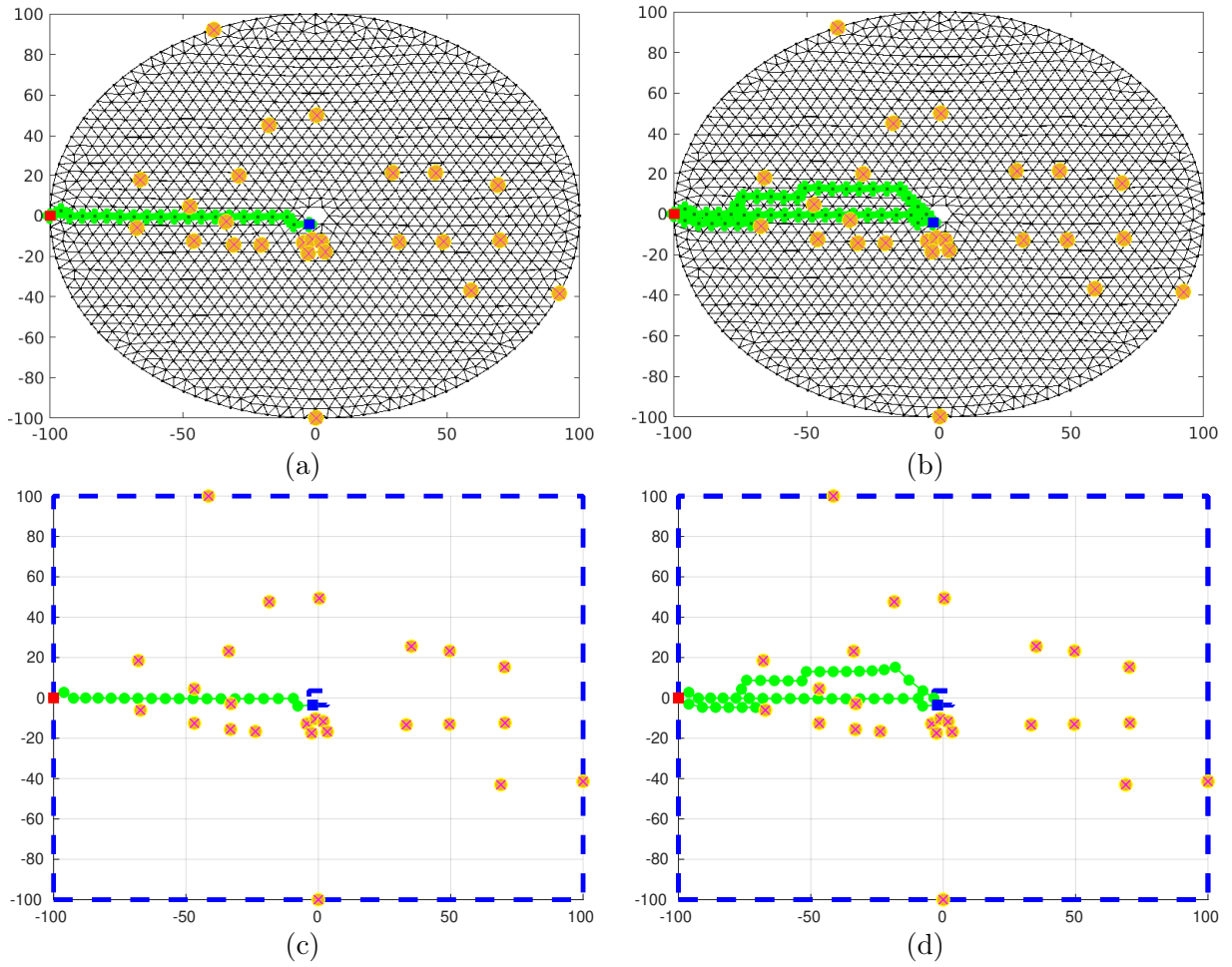


Figure B.17: Case 2: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

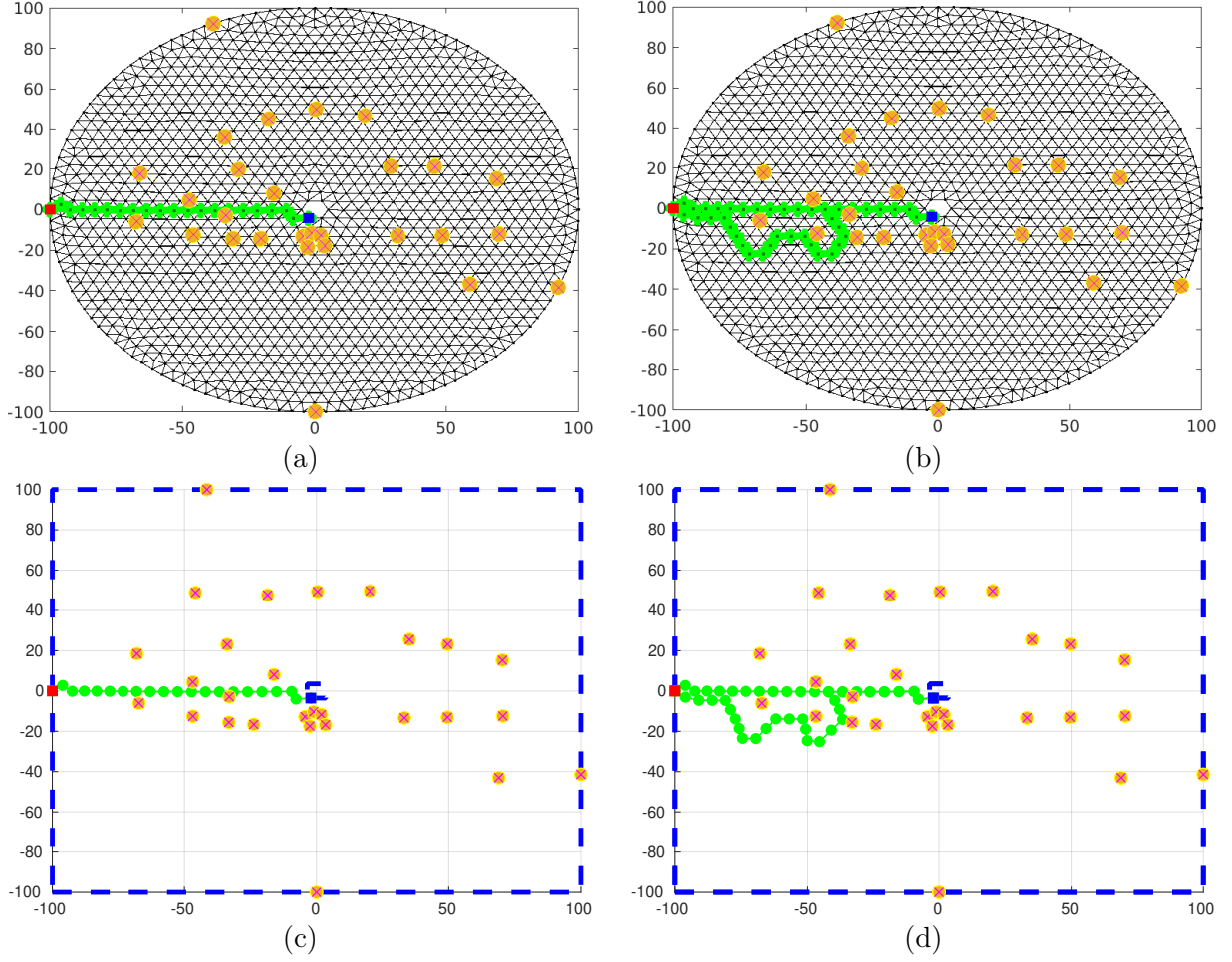


Figure B.18: Case 2: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.55). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

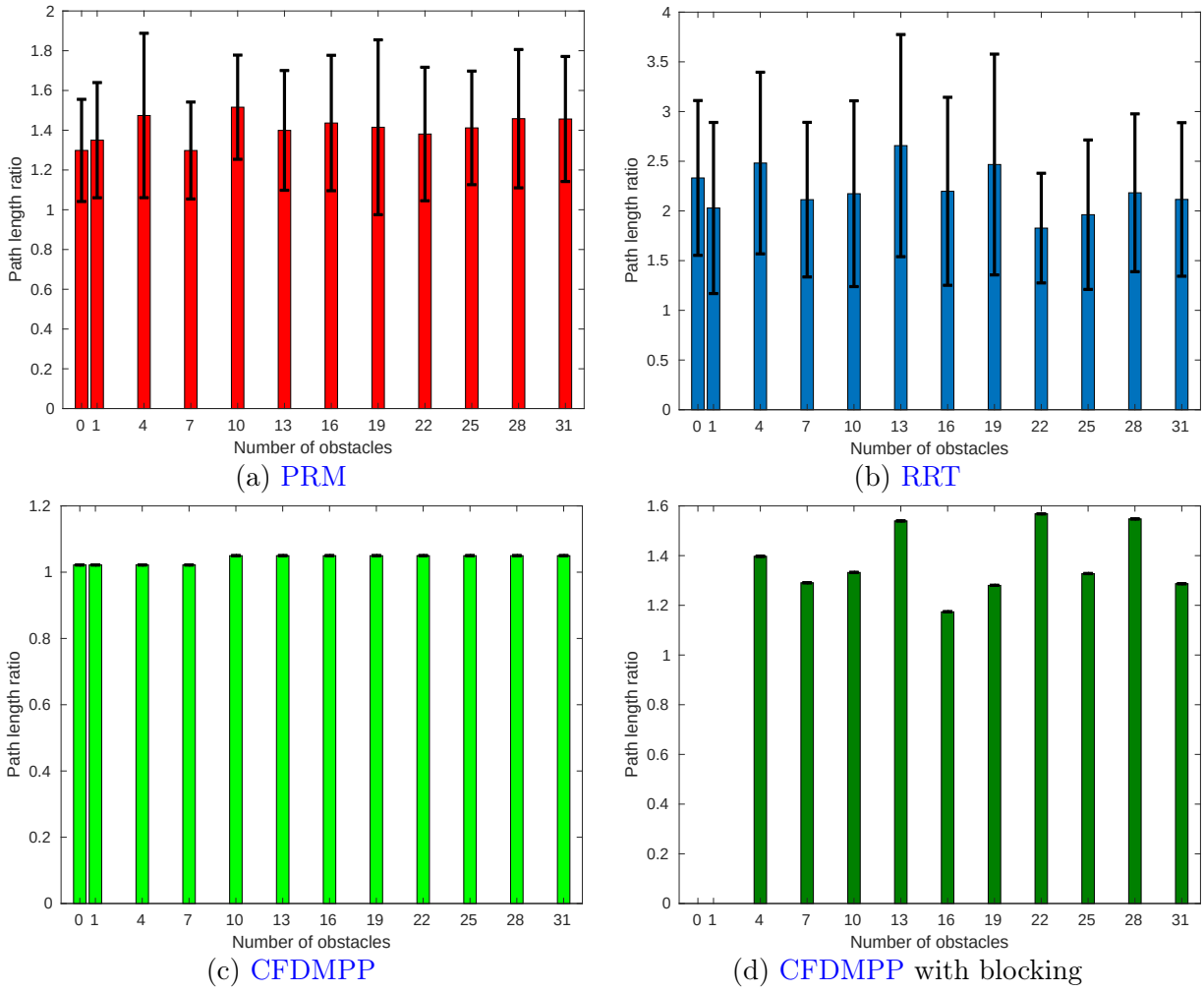
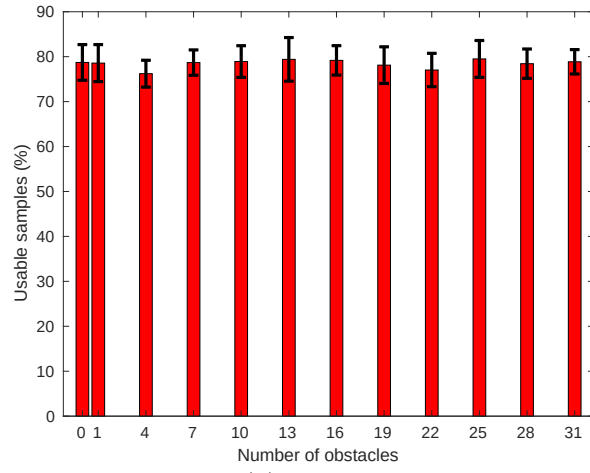
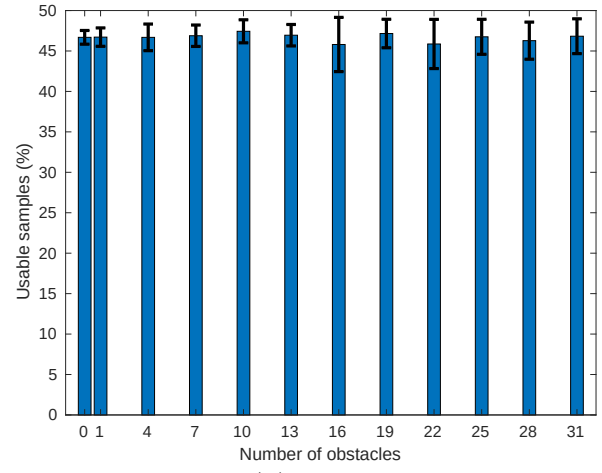


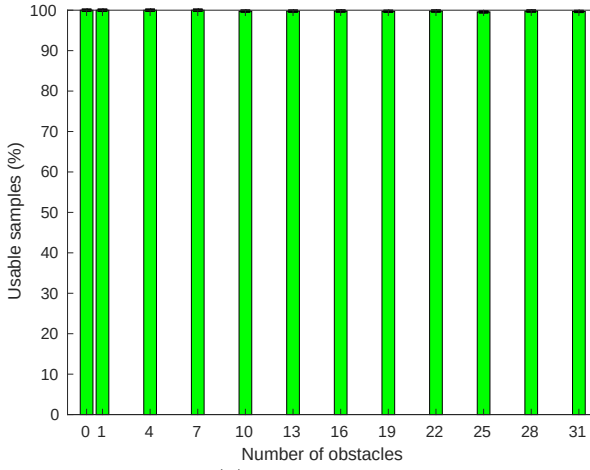
Figure B.19: Case 2: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.



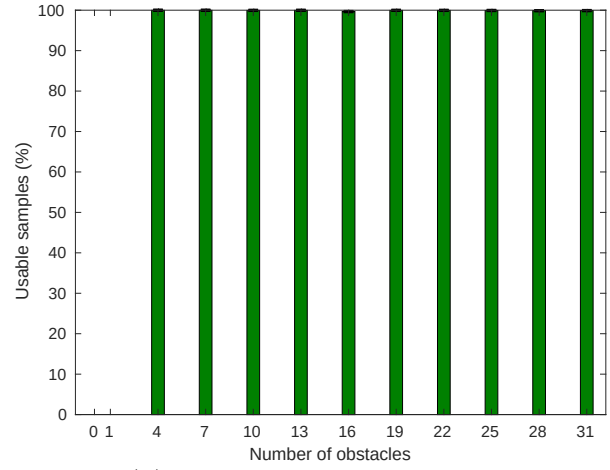
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

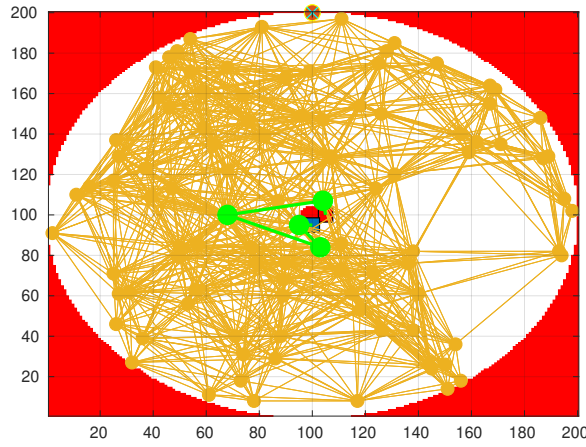
Figure B.20: Case 2: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.

Appendix C

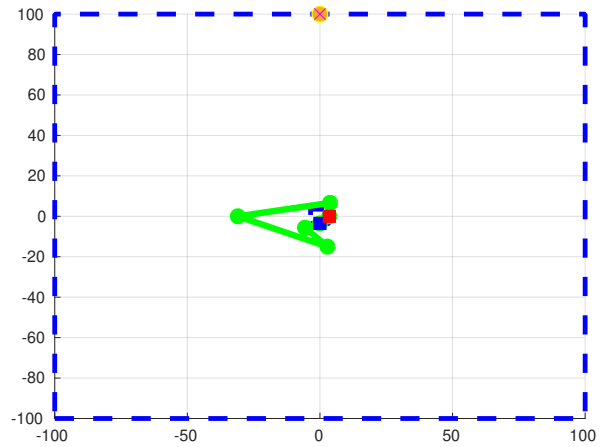
Stage 2: Case 3

This section displays those results from Section 4.2 corresponding to case 3, which were not originally shown in Section 4.2. The following results are displayed here:

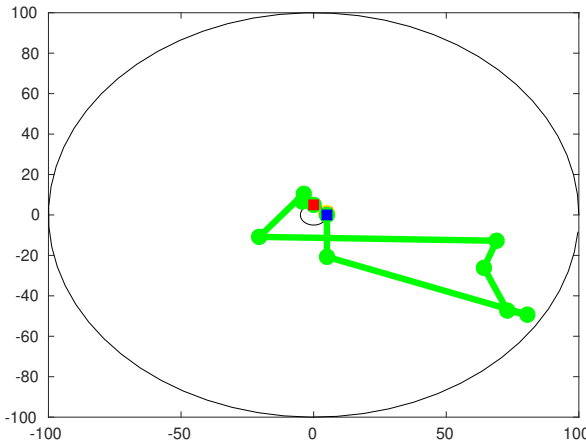
1. The paths produced by [PRM](#), [RRT](#) and [CFDMPP](#) in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
2. The paths produced by [CFDMPP](#) when the original path is homotopically blocked; displayed in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
3. Plots displaying trends of a metric for [PRM](#), [RRT](#) and [CFDMPP](#) (without and with blocking). The two metrics of path length ratio and percentage of usable samples that were not shown in Section 4.2, are shown here for [0,1,4,..31] poles.



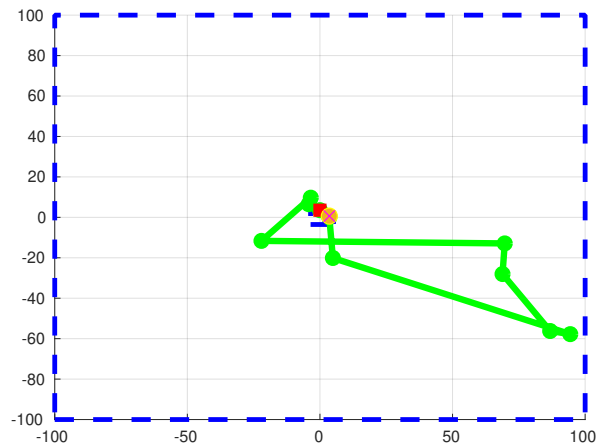
(a) PRM: Path in R-4



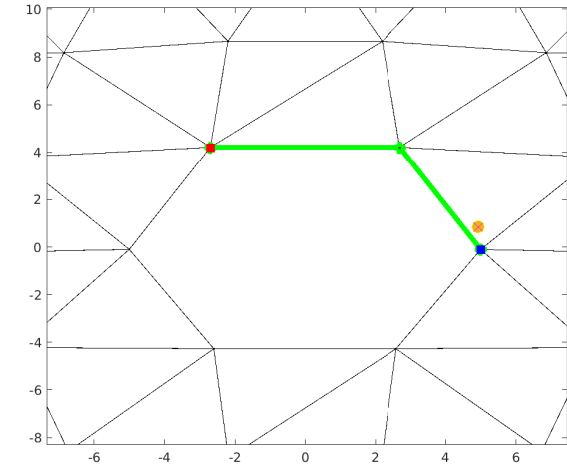
(b) PRM: Path in R-1



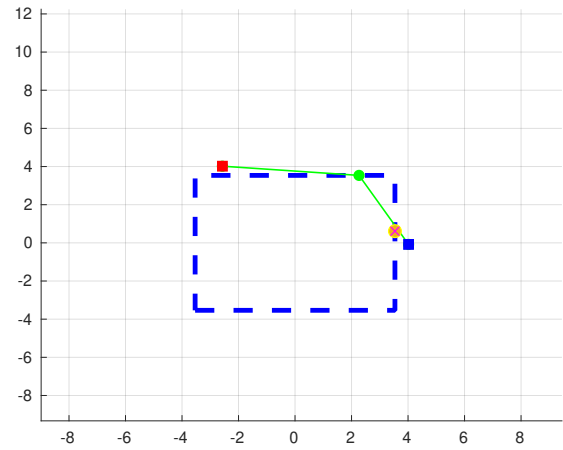
(c) RRT: Path in R-4



(d) RRT: Path in R-1

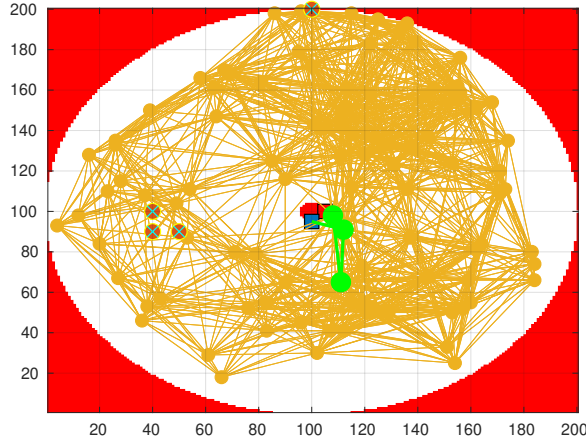


(e) CFDMPP: Path in R-4

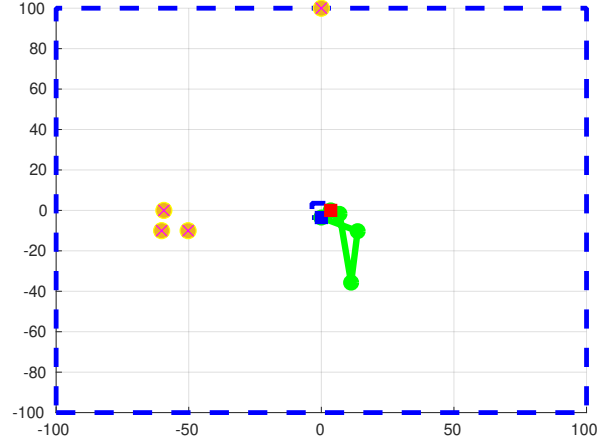


(f) CFDMPP: Path in R-1

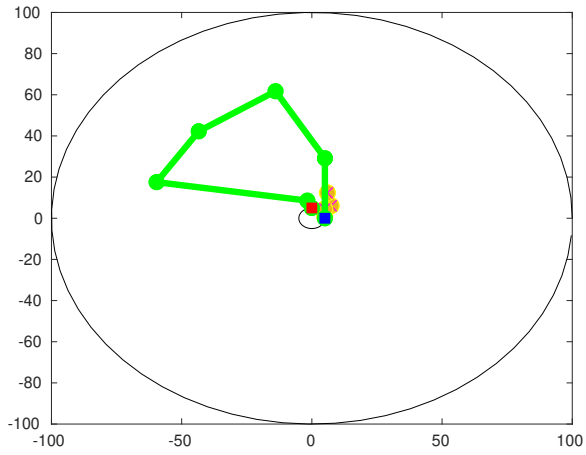
Figure C.1: Case 3: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.



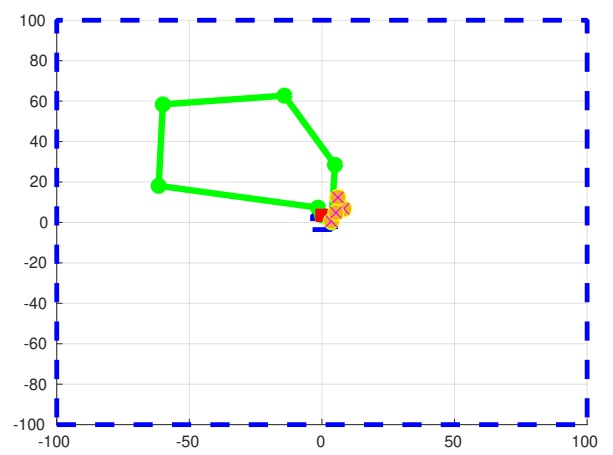
(a) PRM: Path in R-4



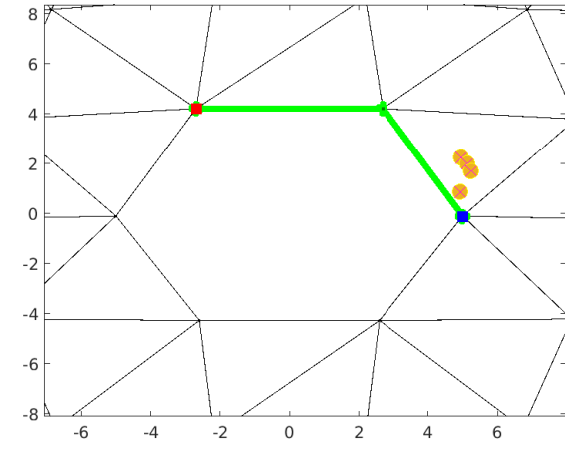
(b) PRM: Path in R-1



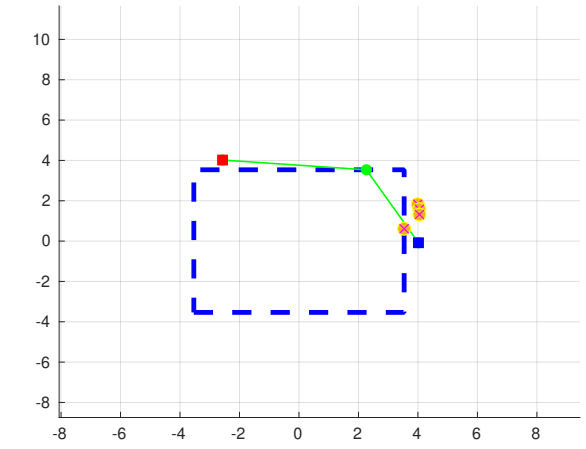
(c) RRT: Path in R-4



(d) RRT: Path in R-1

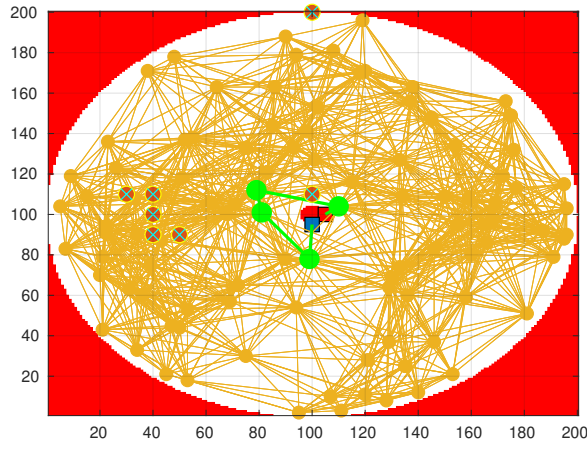


(e) CFDMPP: Path in R-4

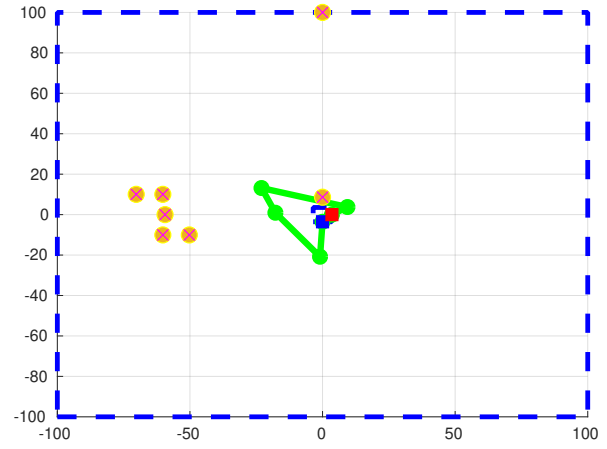


(f) CFDMPP: Path in R-1

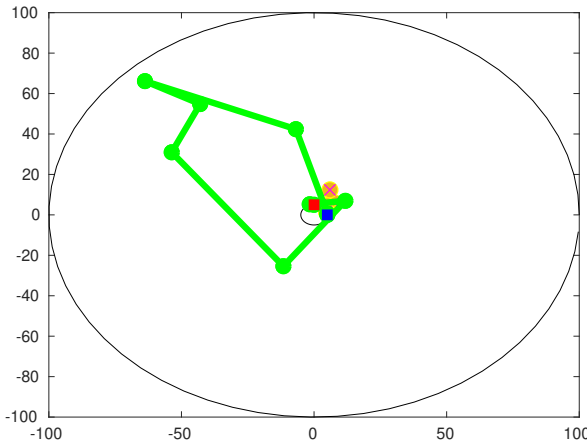
Figure C.2: Case 3: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.



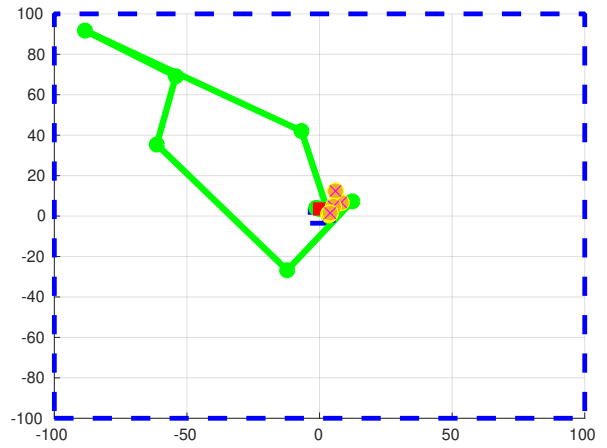
(a) PRM: Path in R-4



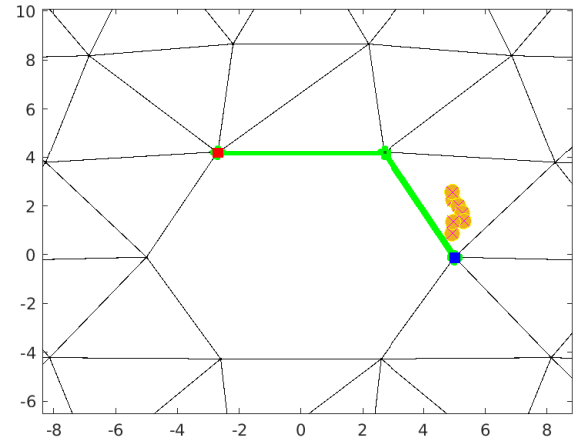
(b) PRM: Path in R-1



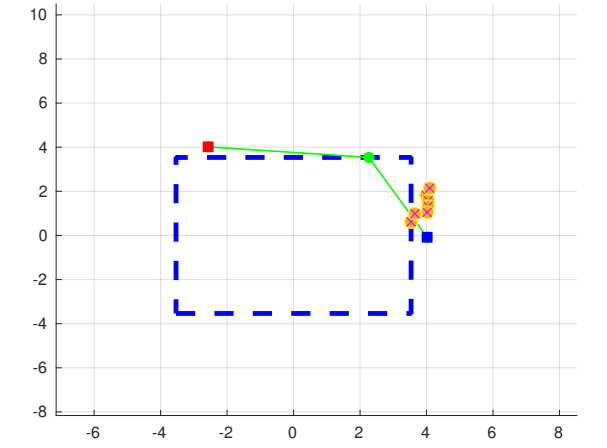
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure C.3: Case 3: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.

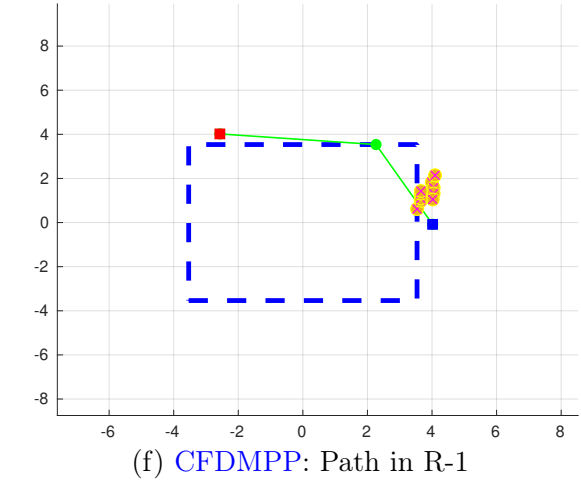
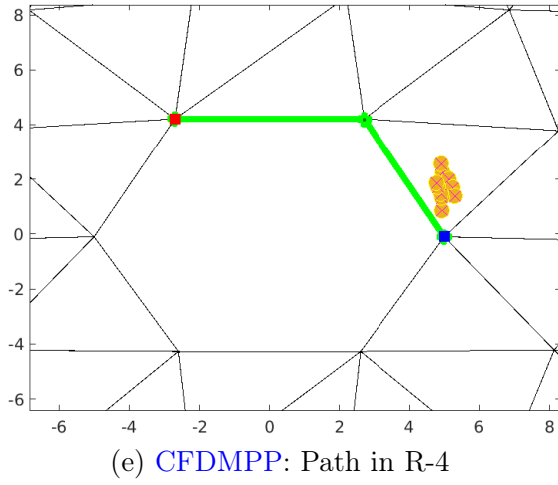
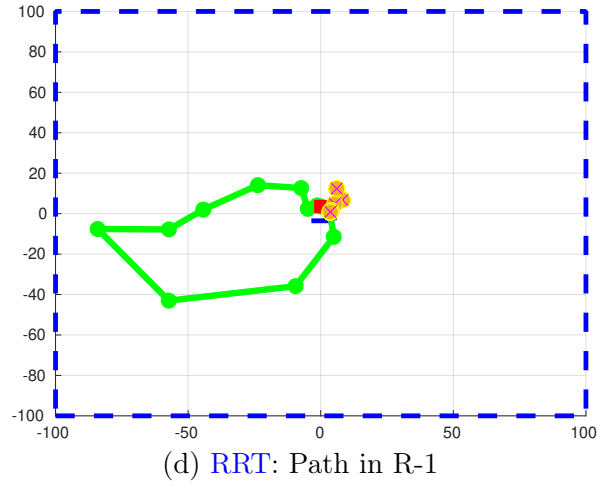
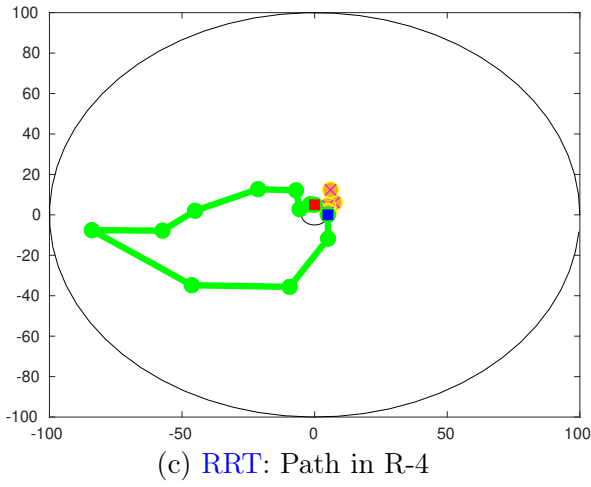
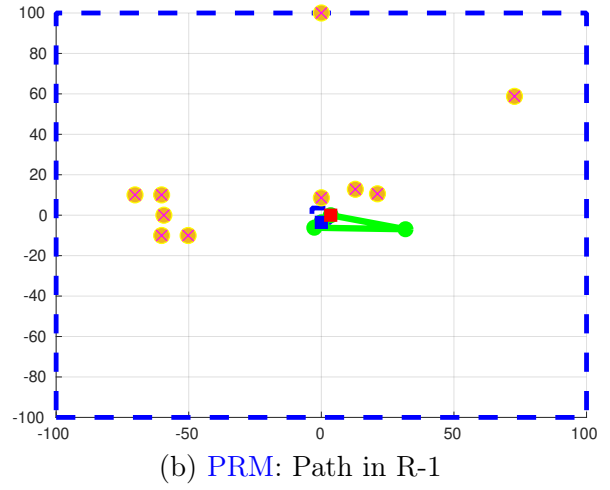
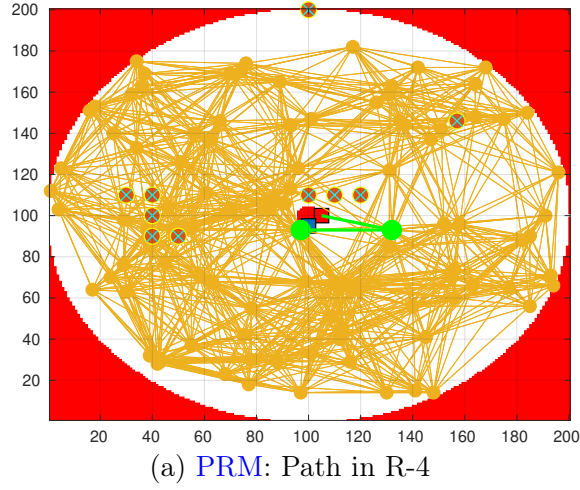
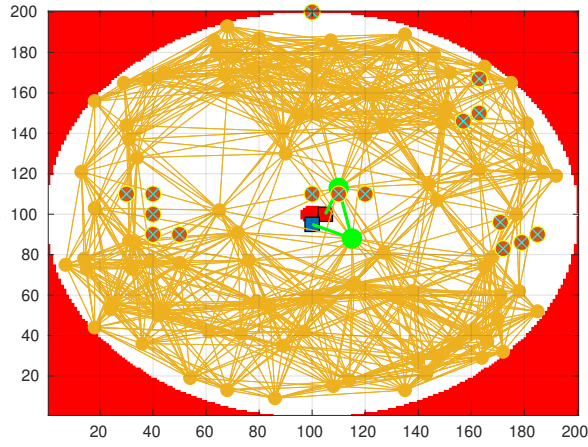
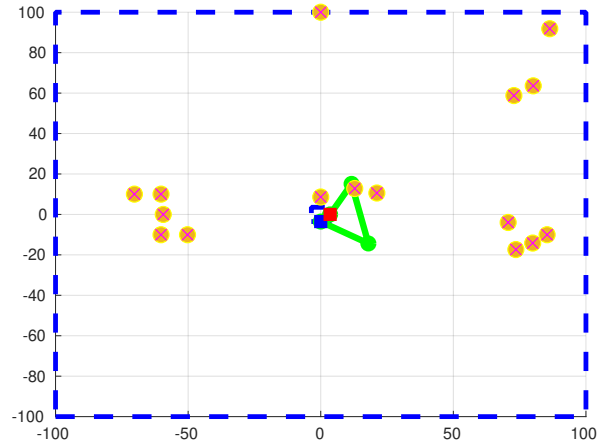


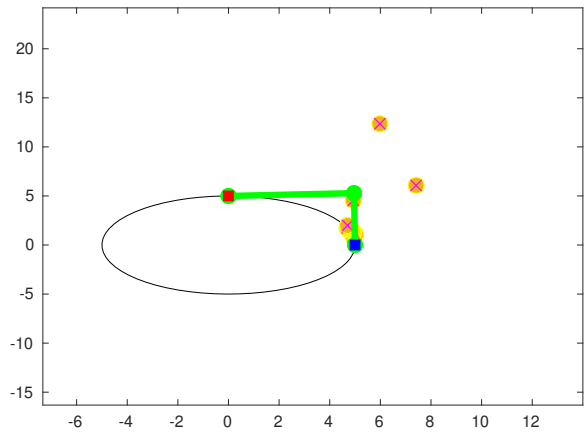
Figure C.4: Case 3: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.



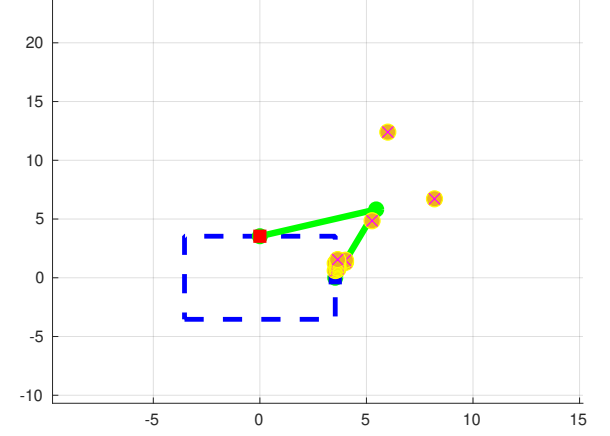
(a) PRM: Path in R-4



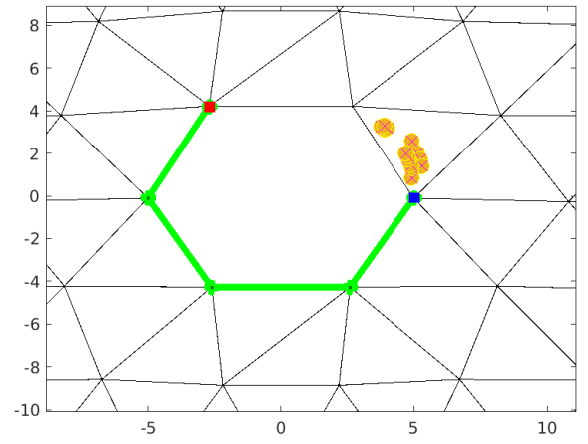
(b) PRM: Path in R-1



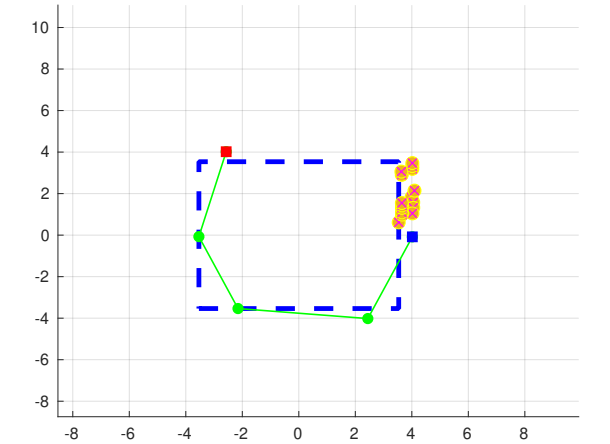
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure C.5: Case 3: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.

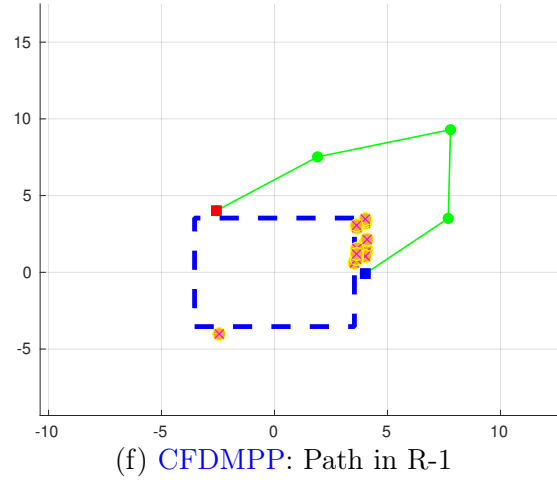
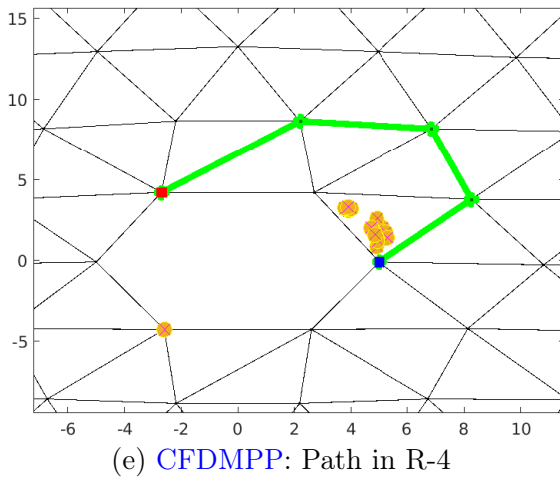
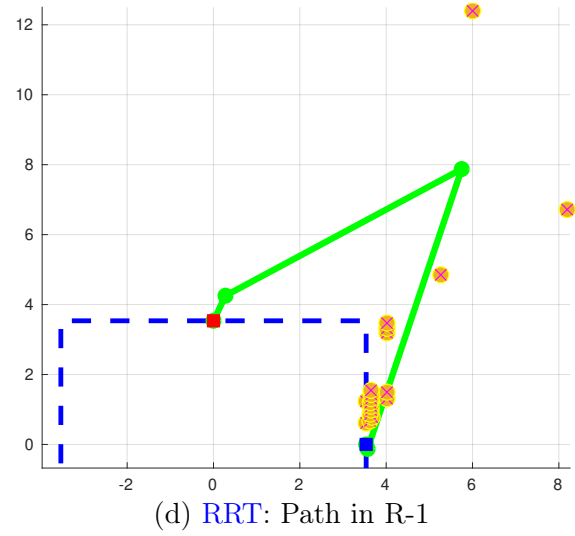
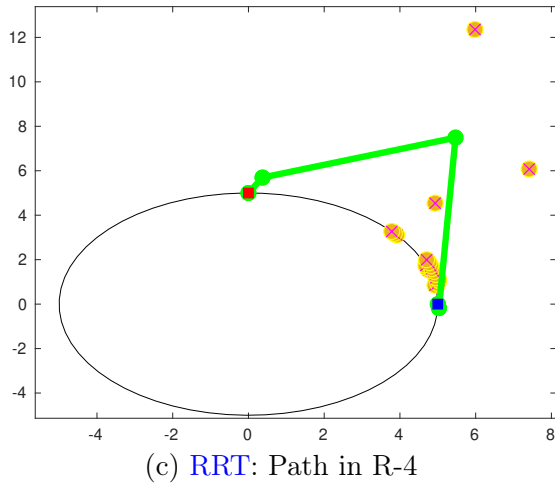
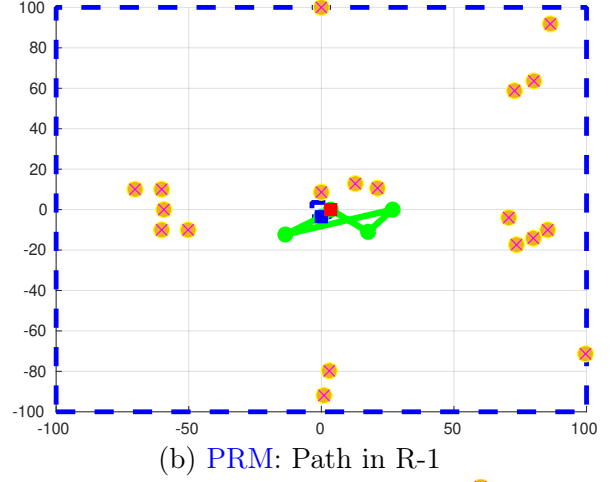
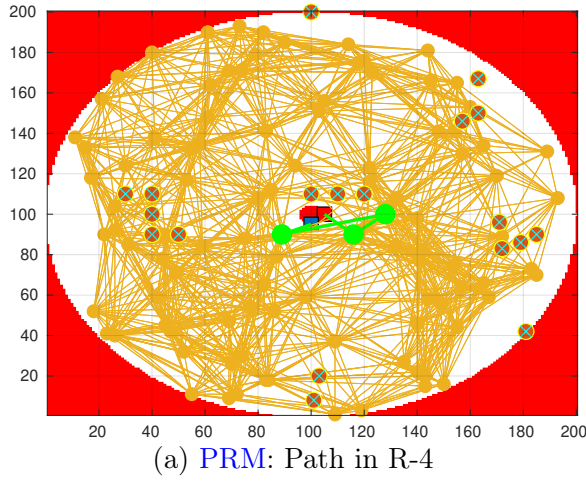
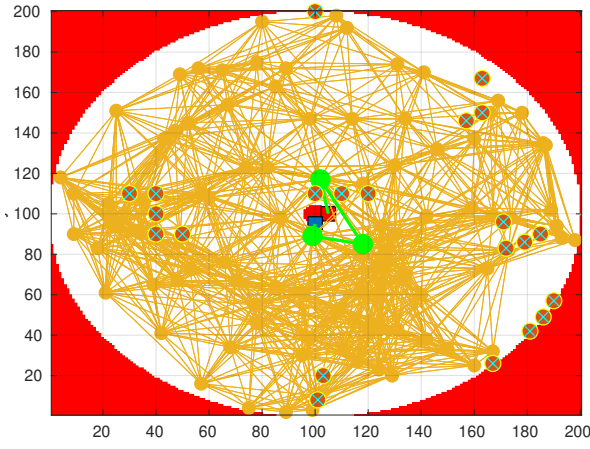
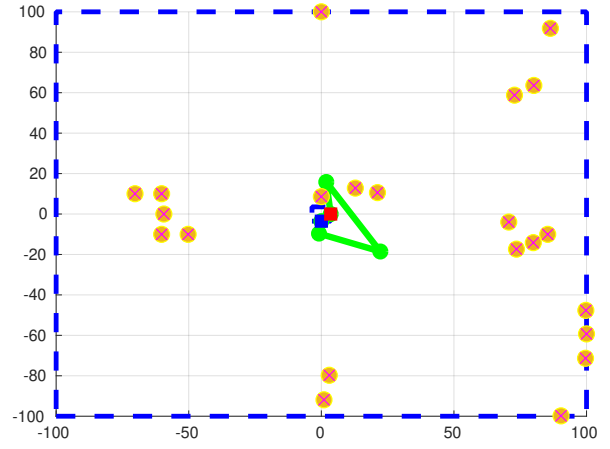


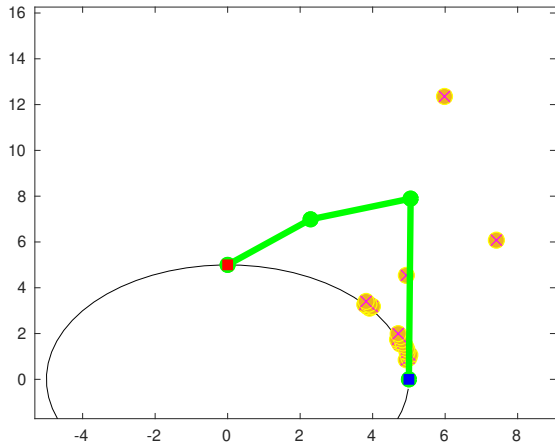
Figure C.6: Case 3: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.



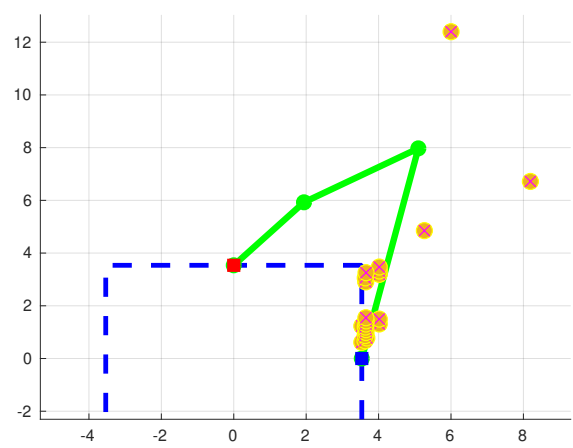
(a) PRM: Path in R-4



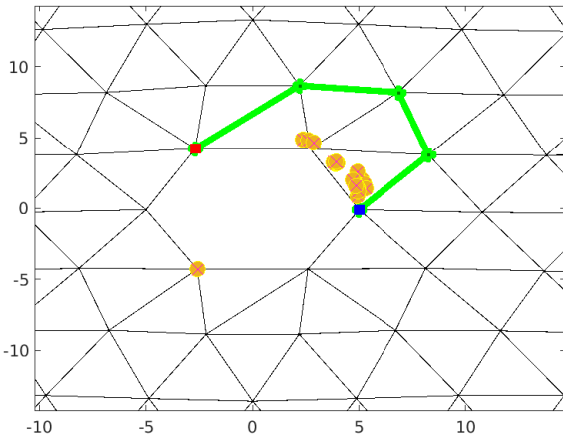
(b) PRM: Path in R-1



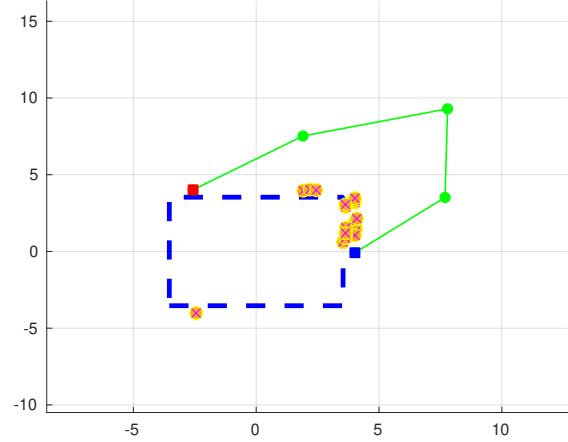
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure C.7: Case 3: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.

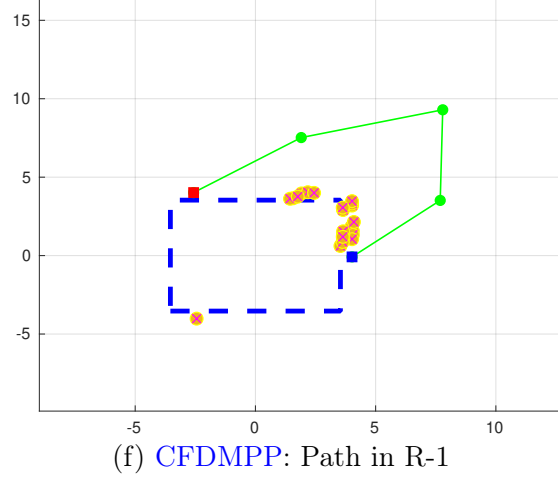
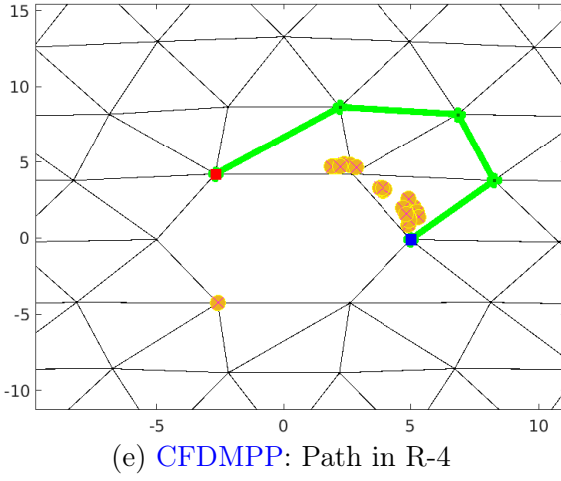
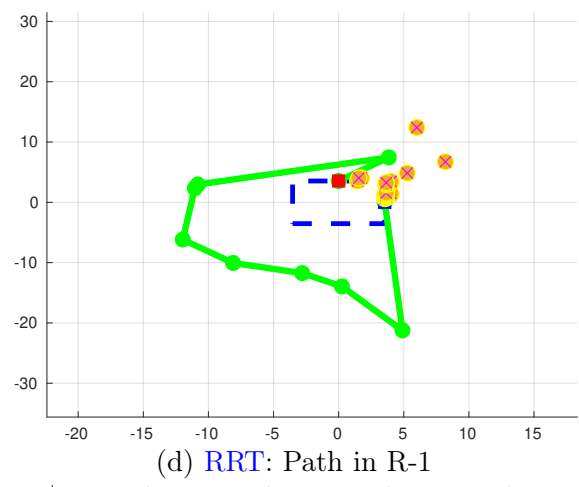
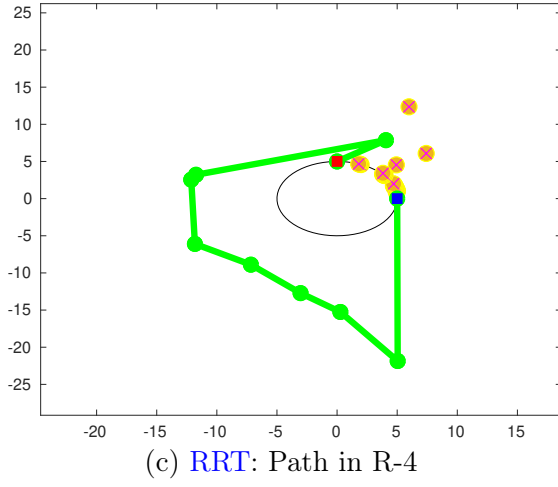
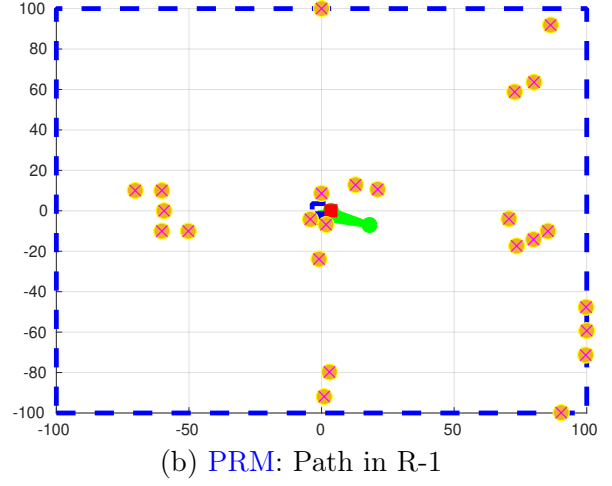
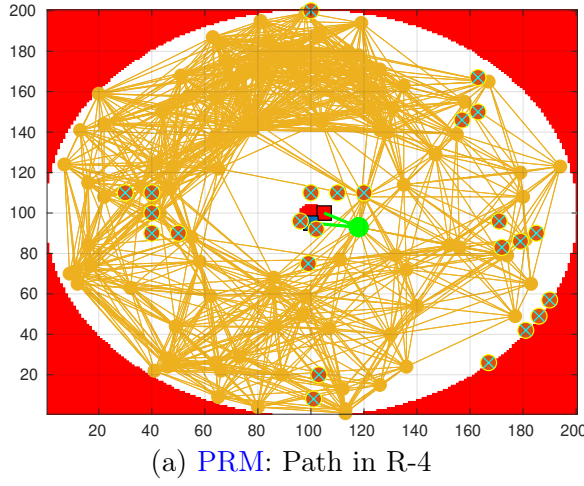
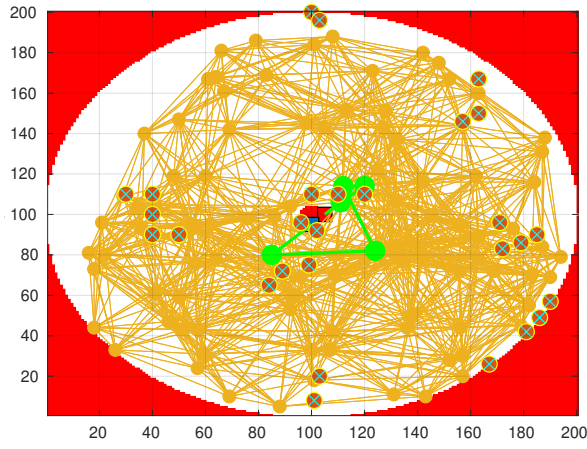
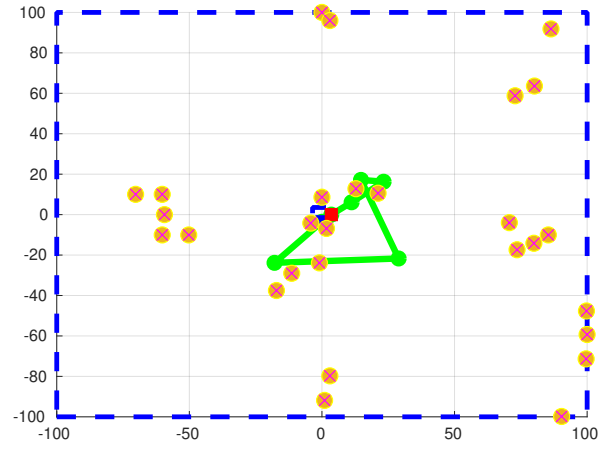


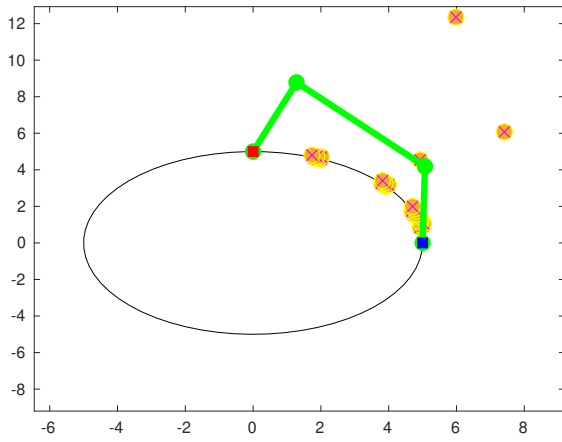
Figure C.8: Case 3: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.



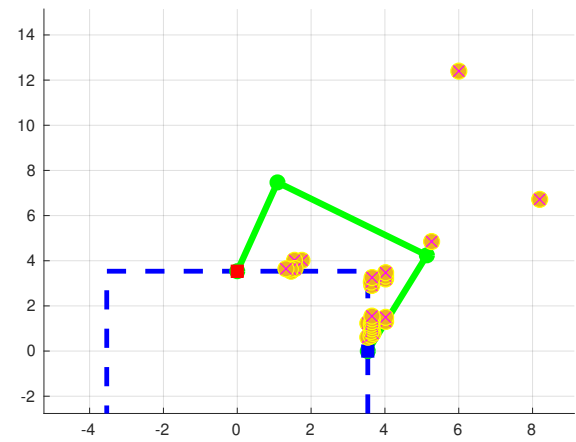
(a) PRM: Path in R-4



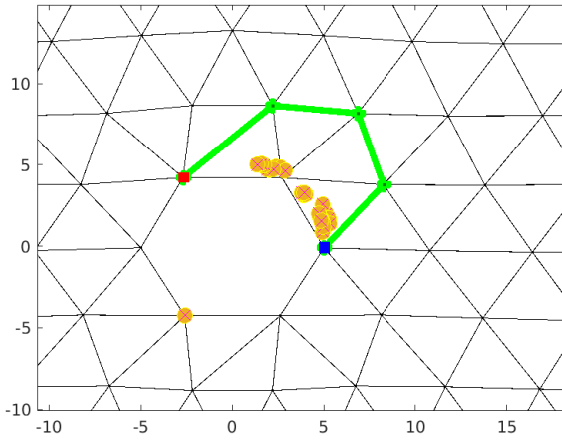
(b) PRM: Path in R-1



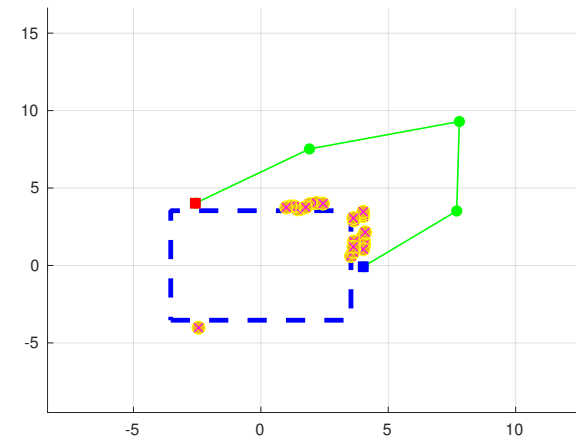
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure C.9: Case 3: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.

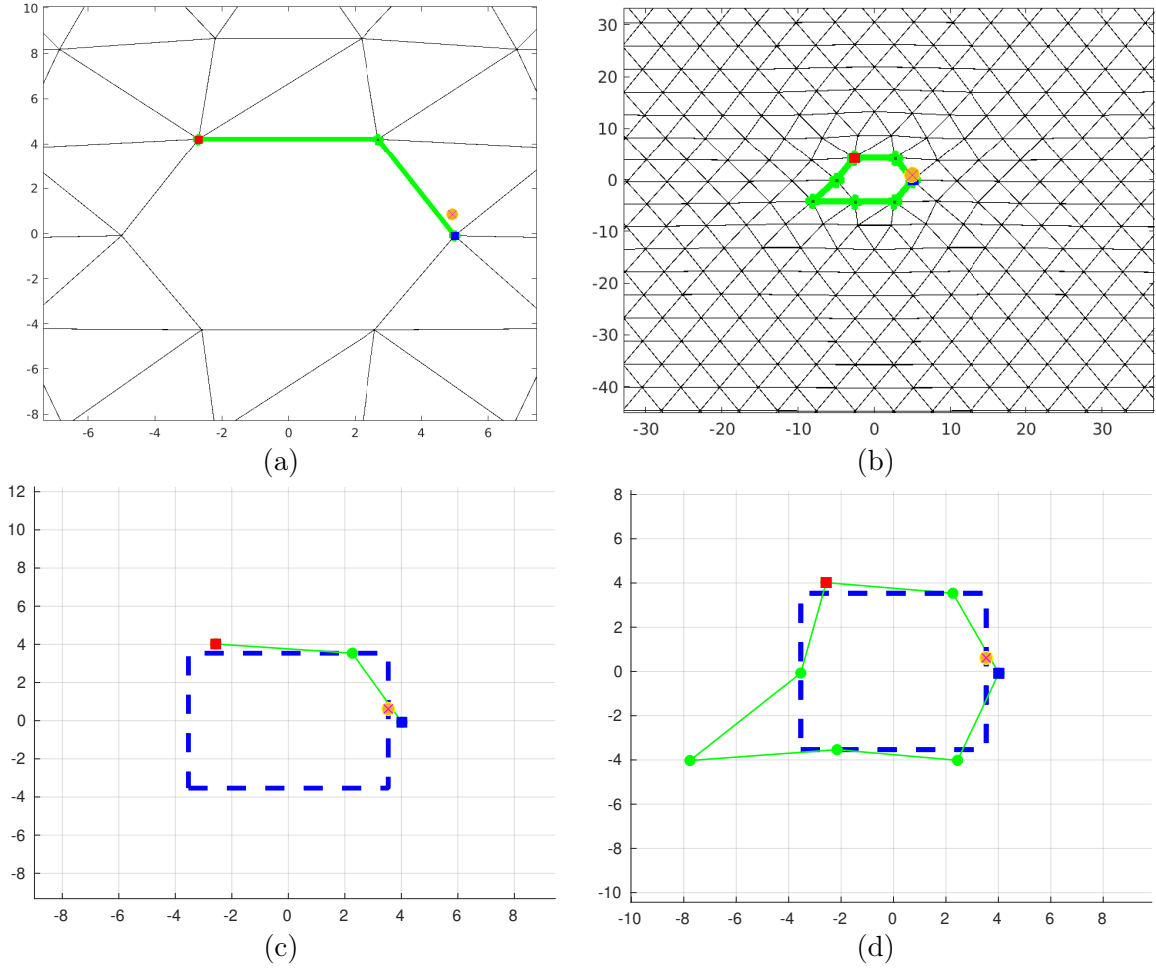


Figure C.10: Case 3: CFDMPP in the case of 1 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1.15). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

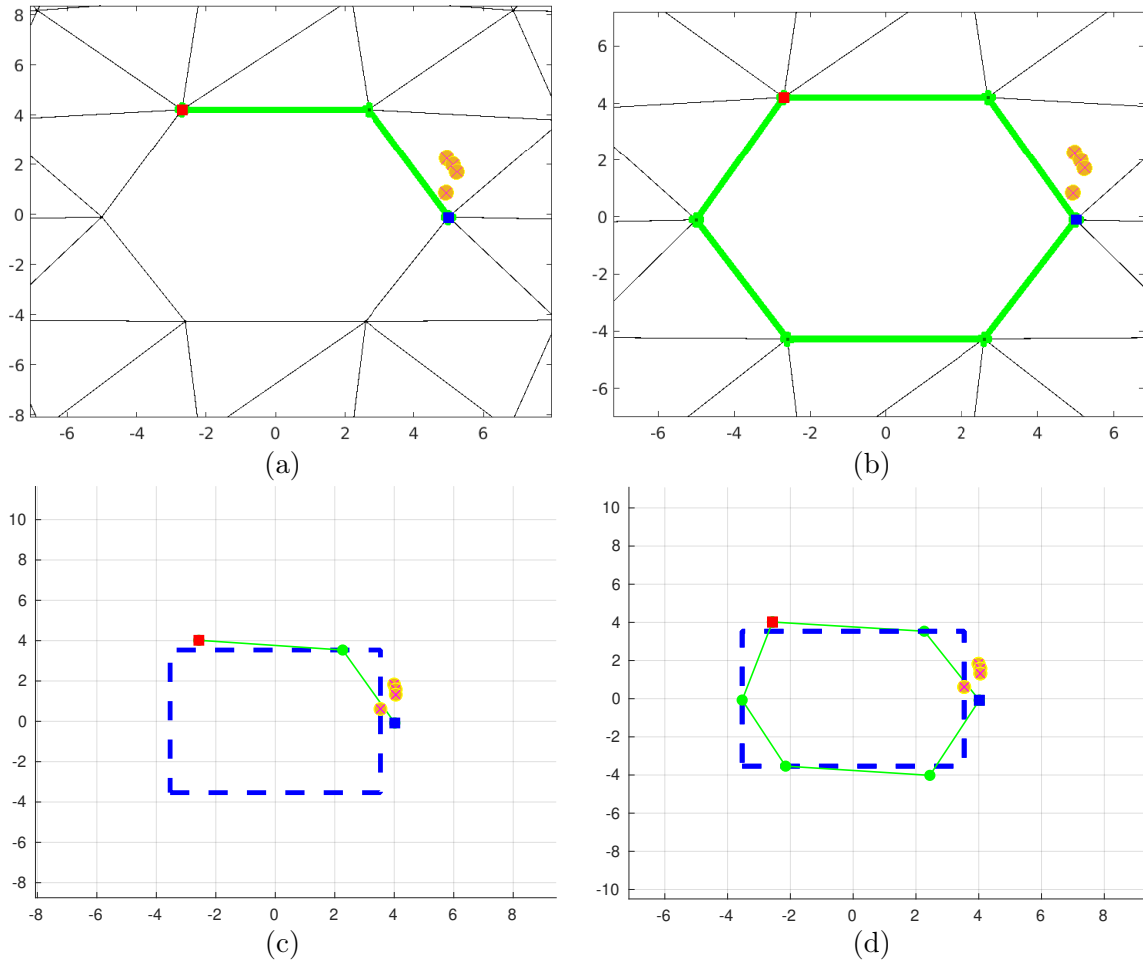


Figure C.11: Case 3: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.8). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

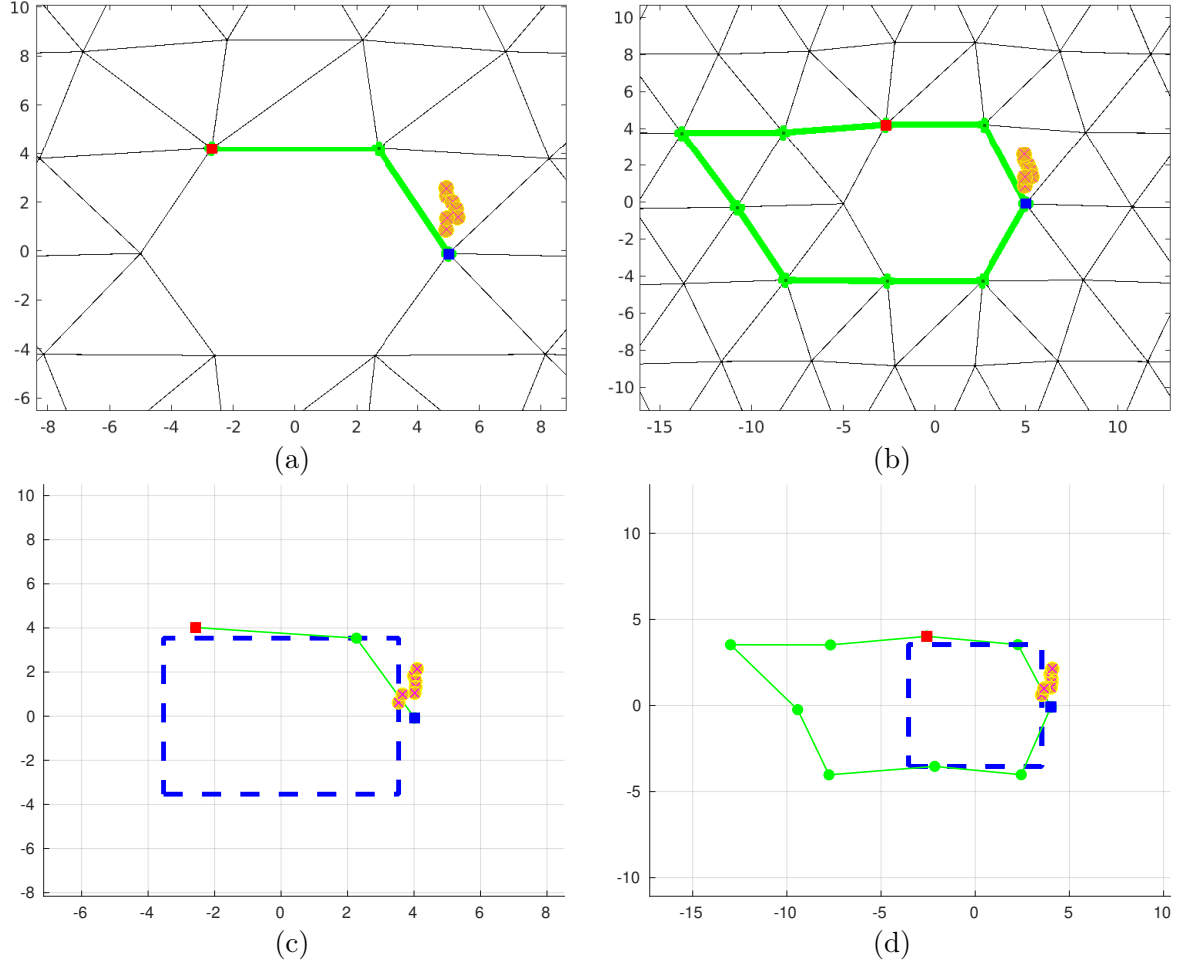


Figure C.12: Case 3: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.4). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

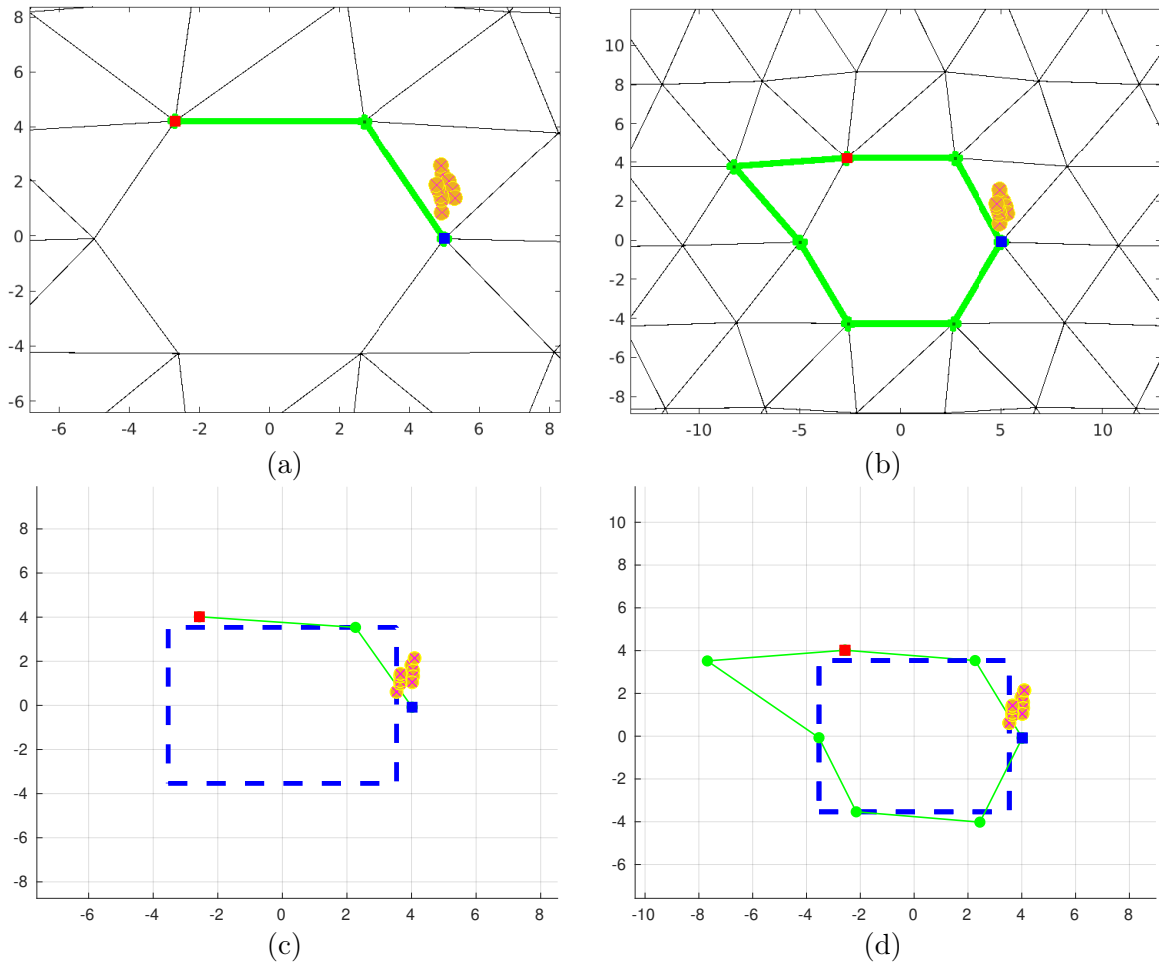


Figure C.13: Case 3: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.4). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

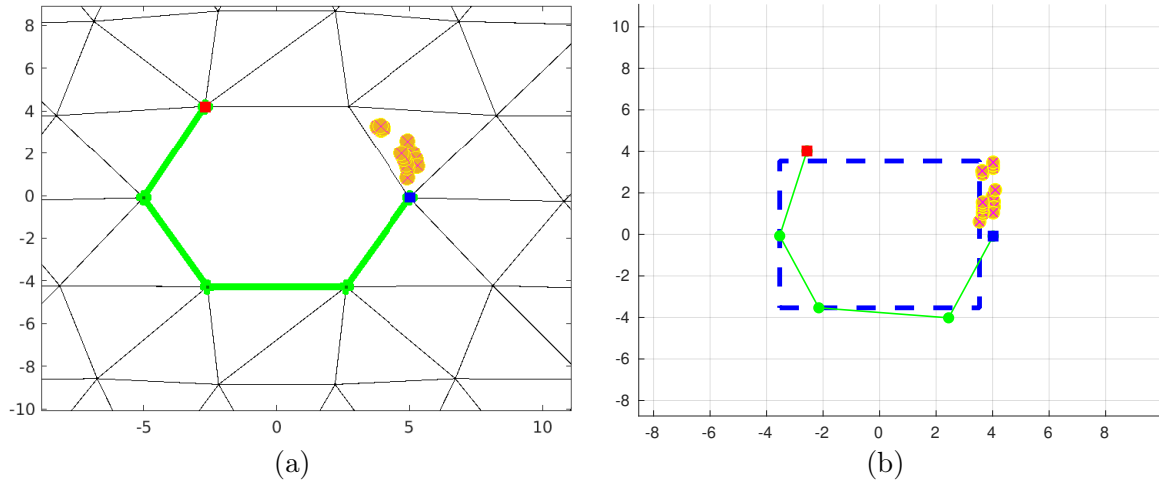


Figure C.14: Case 3: CFDMPP in the case of 16 poles (first group of DCs); unable to tune for an alternative path class. (a) Path in R-4. (b) Path in R-1.

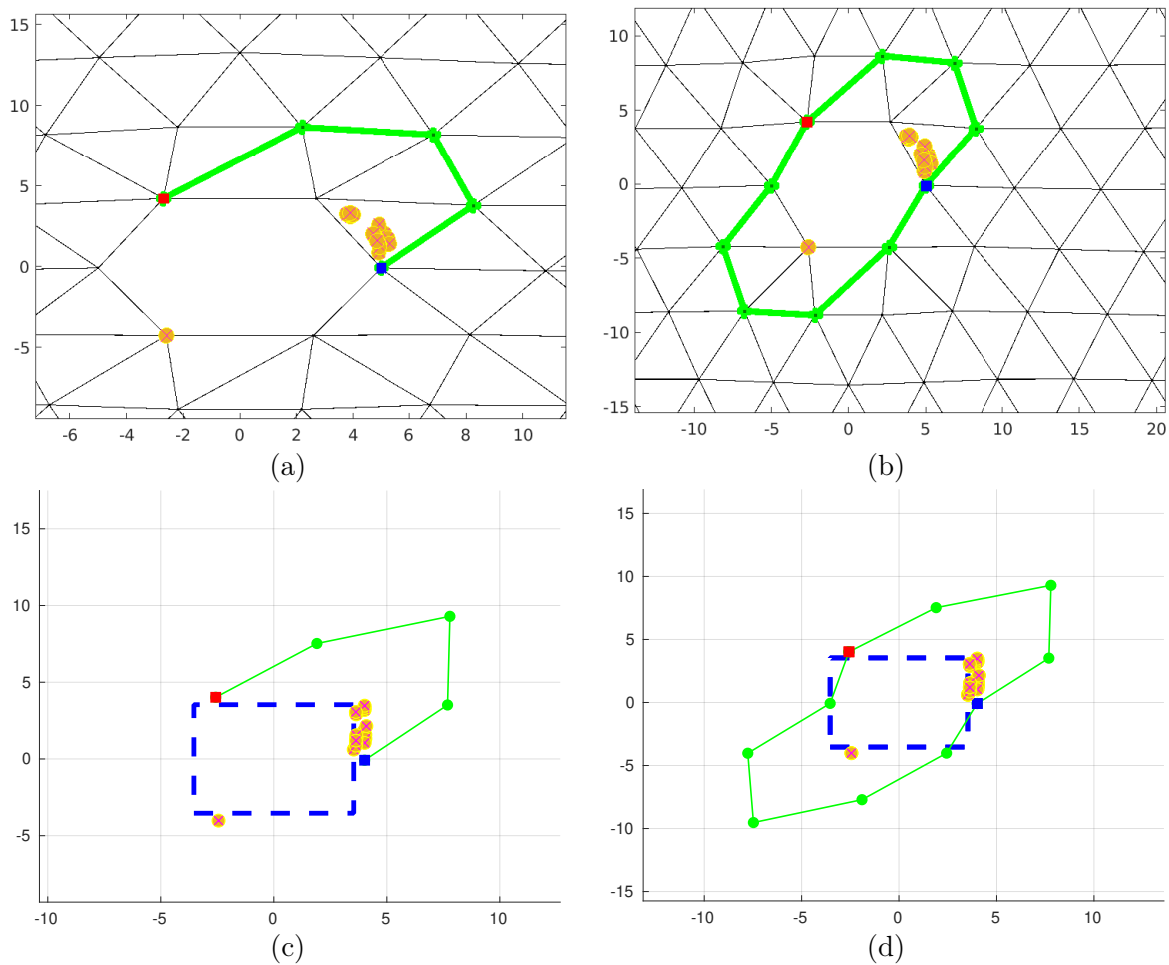


Figure C.15: Case 3: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

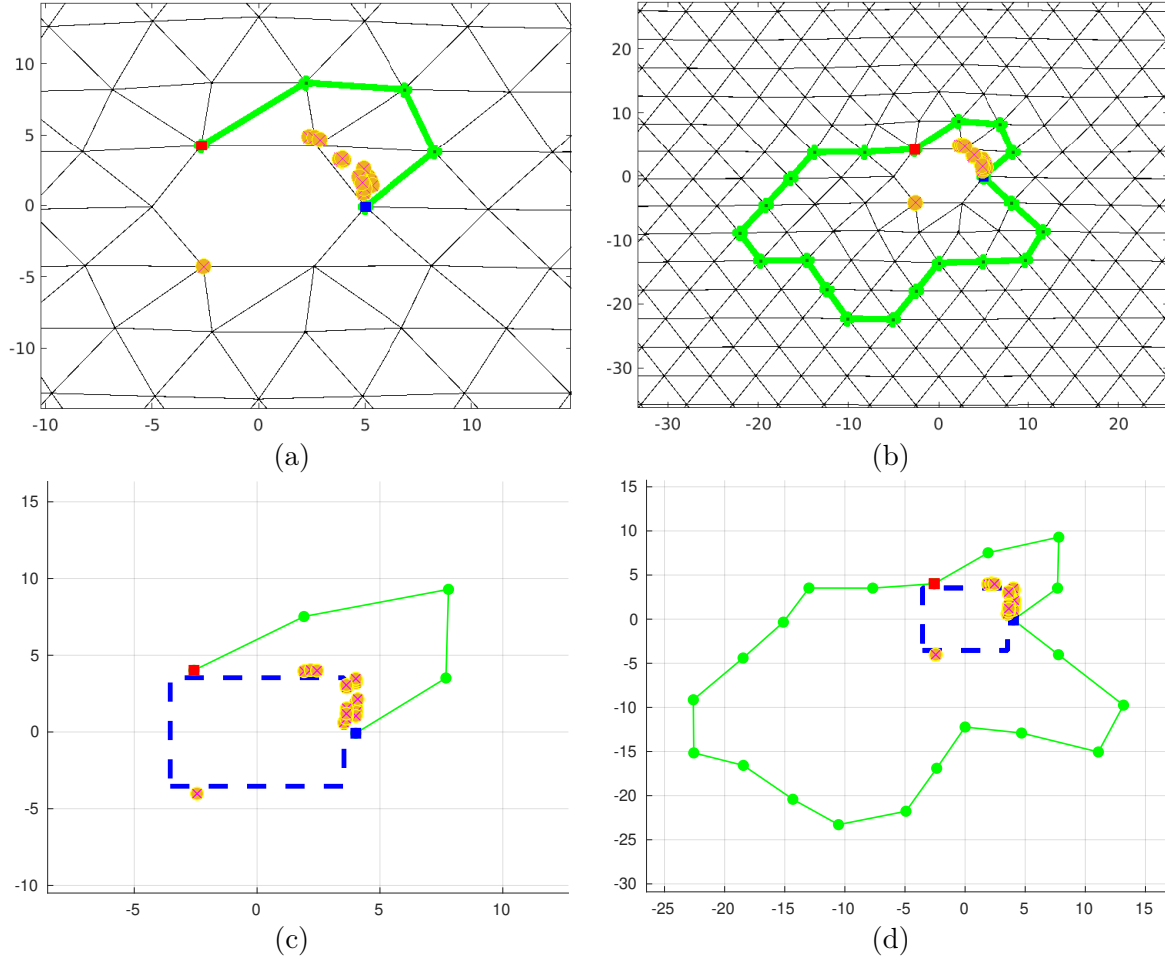


Figure C.16: Case 3: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-4. (d) Blocked path and resulting path in R-1.

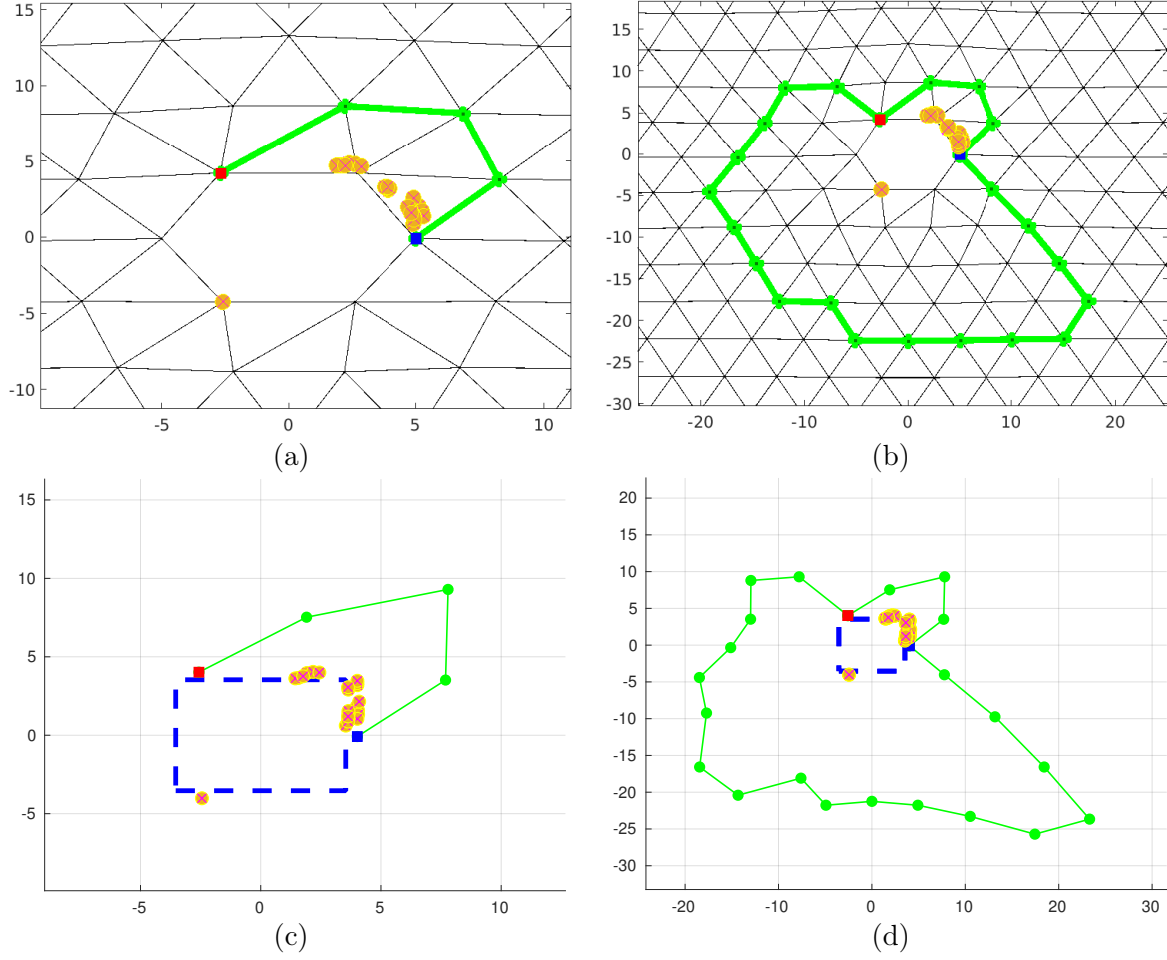


Figure C.17: Case 3: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.75). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

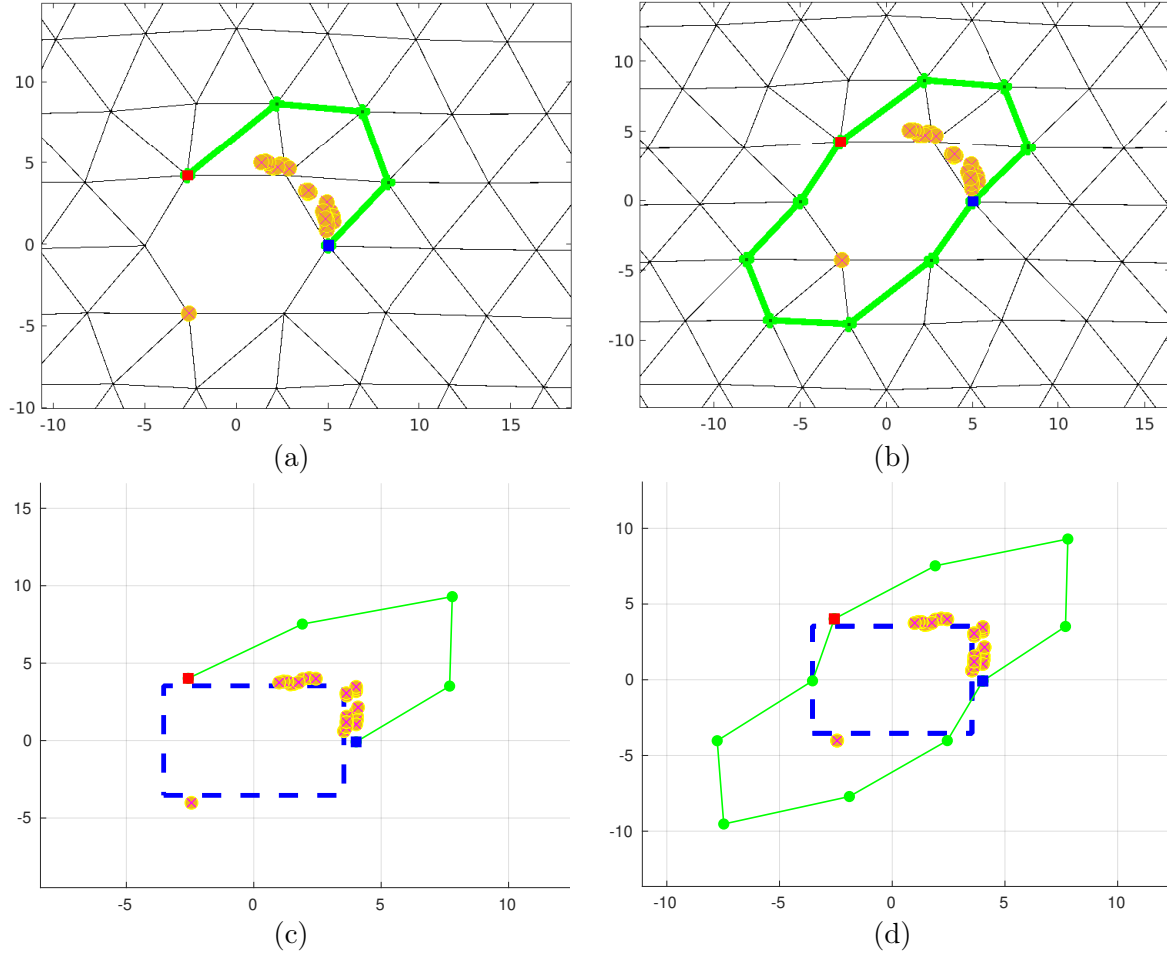


Figure C.18: Case 3: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.75). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

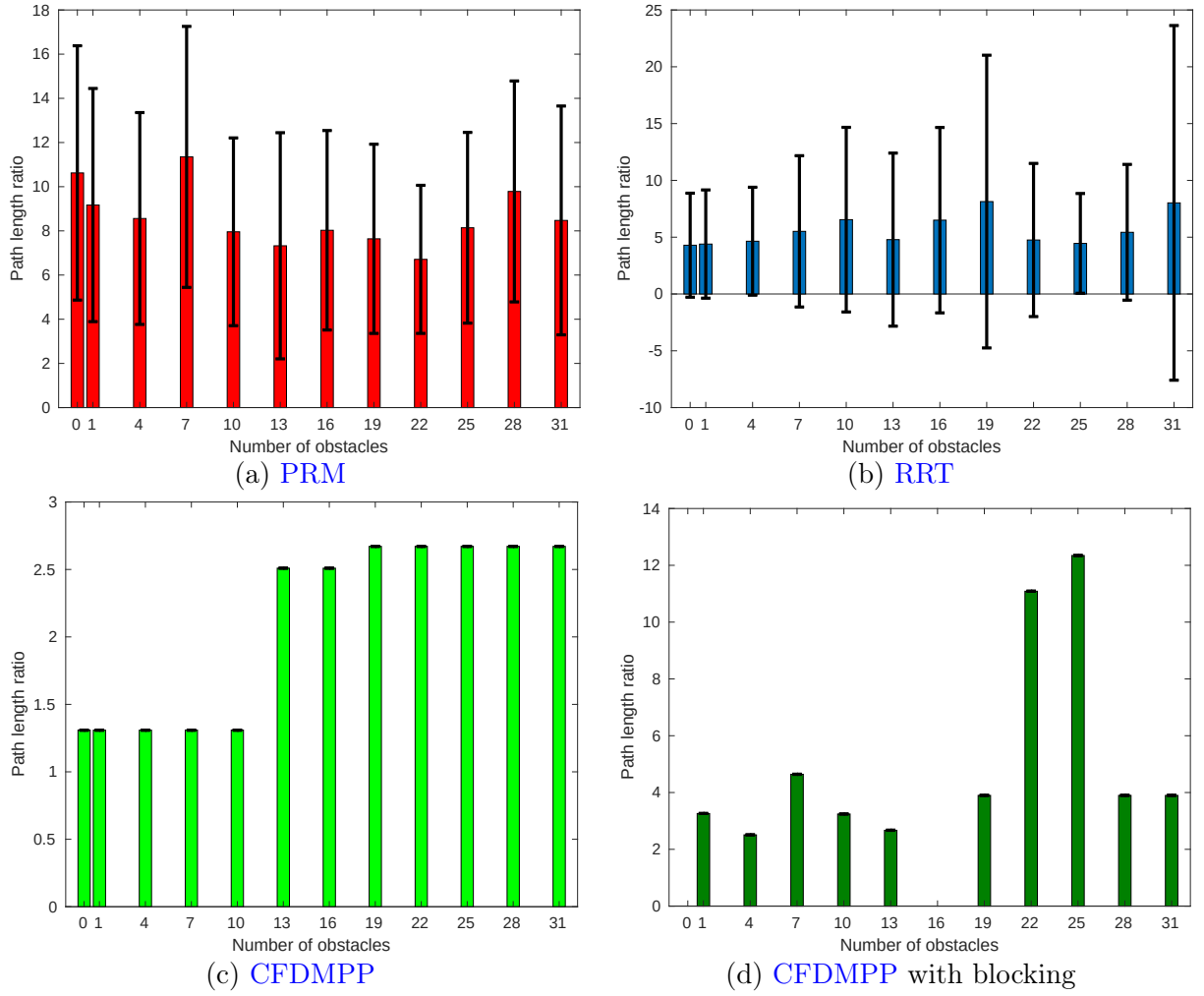


Figure C.19: Case 3: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.

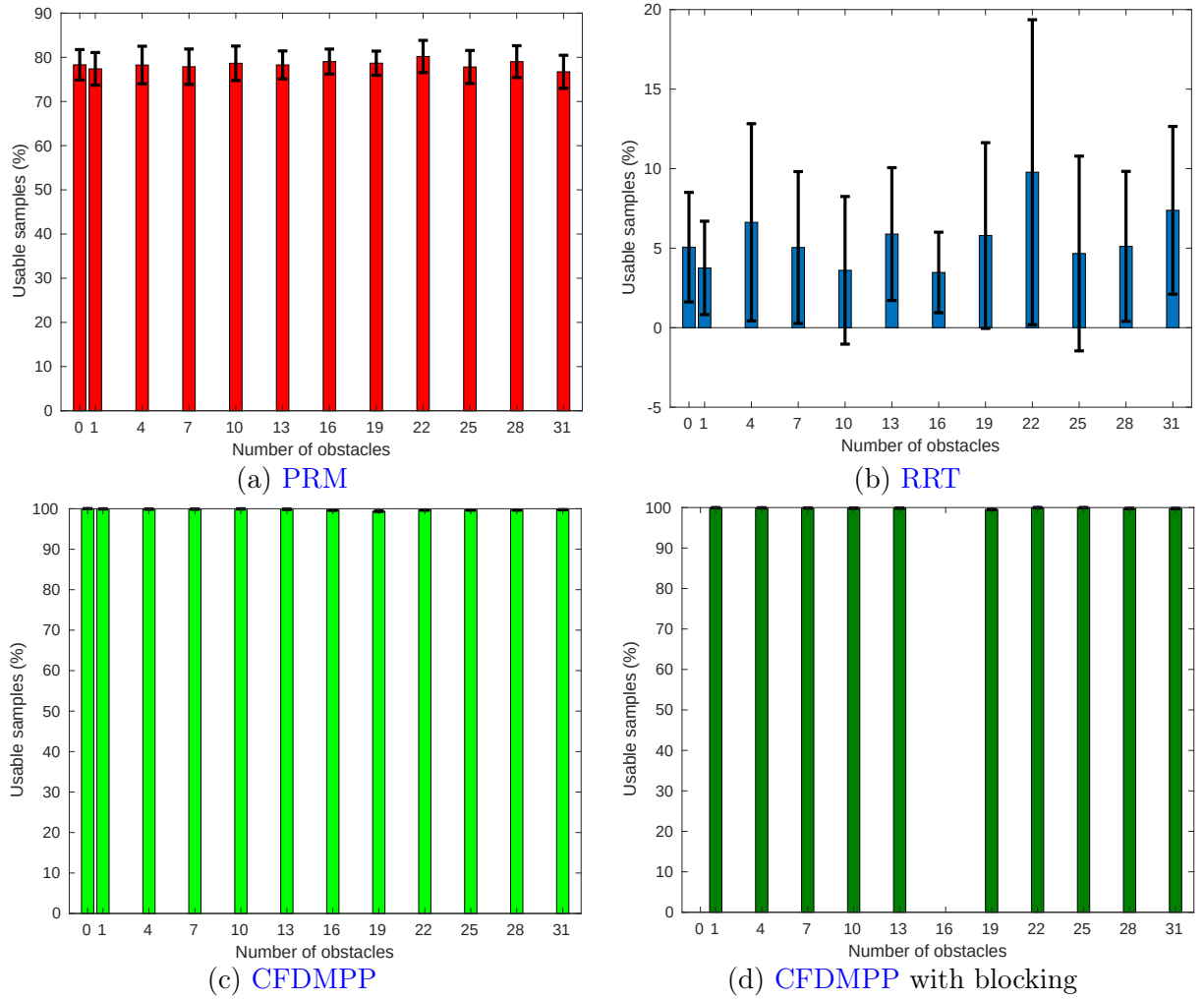


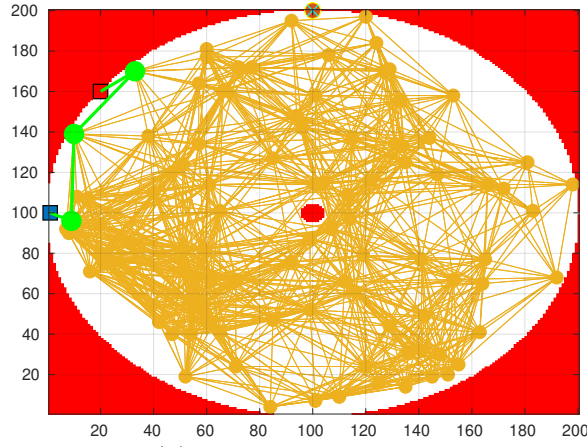
Figure C.20: Case 3: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.

Appendix D

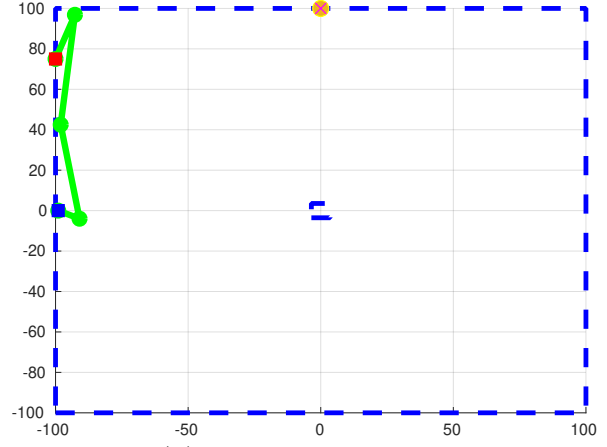
Stage 2: Case 4

This section displays those results from Section 4.2 corresponding to case 4, which were not originally shown in Section 4.2. The following results are displayed here:

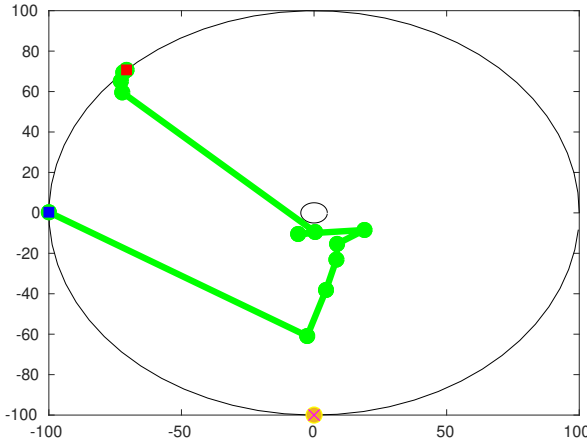
1. The paths produced by [PRM](#), [RRT](#) and [CFDMPP](#) in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
2. The paths produced by [CFDMPP](#) when the original path is homotopically blocked; displayed in R-4 and R-1 for 1, 4, 7, 10, 16, 19, 22, 25 and 28 poles.
3. Plots displaying trends of a metric for [PRM](#), [RRT](#) and [CFDMPP](#) (without and with blocking). The two metrics of path length ratio and percentage of usable samples that were not shown in Section 4.2, are shown here for [0,1,4,..31] poles.



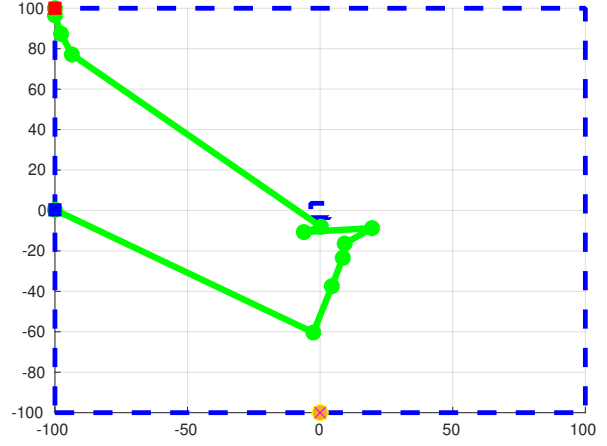
(a) PRM: Path in R-4



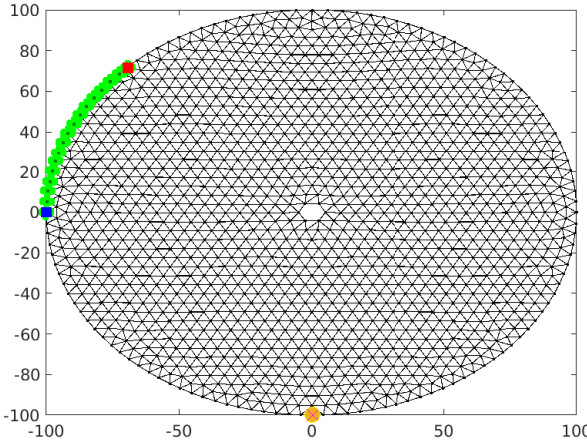
(b) PRM: Path in R-1



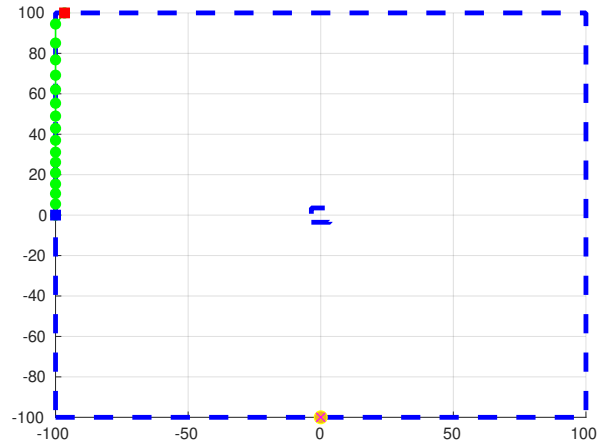
(c) RRT: Path in R-4



(d) RRT: Path in R-1

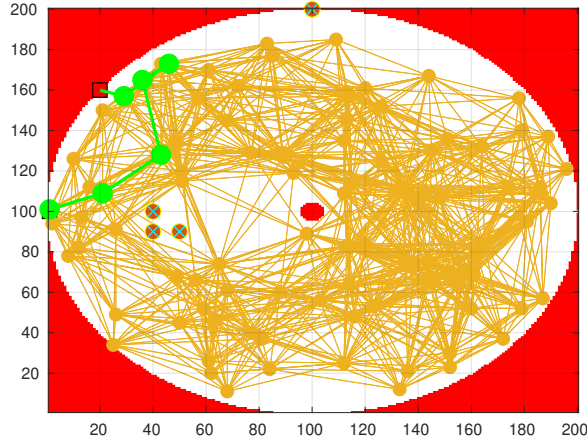


(e) CFDMPP: Path in R-4

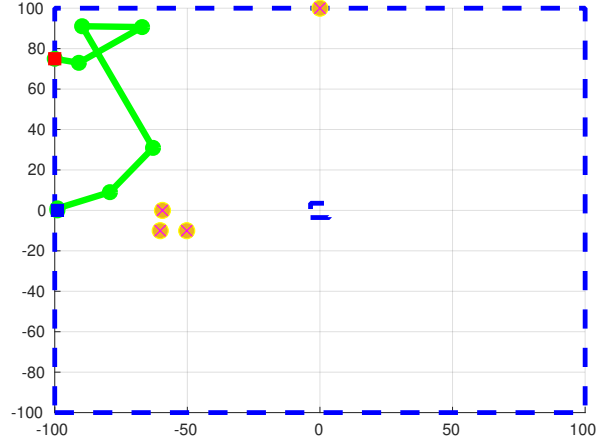


(f) CFDMPP: Path in R-1

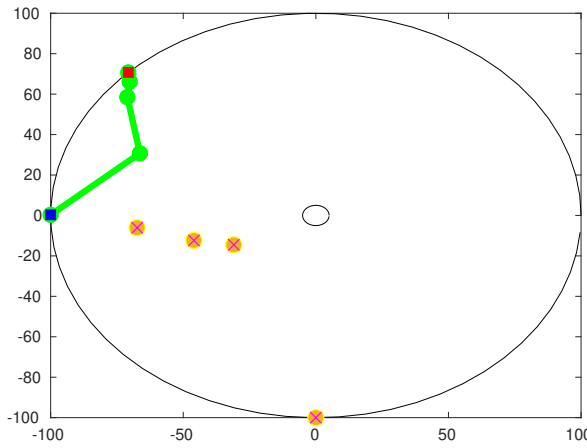
Figure D.1: Case 4: Paths generated by PRM, RRT and CFDMPP with 1 obstacle/pole in the environment; paths shown in R-4 and R-1.



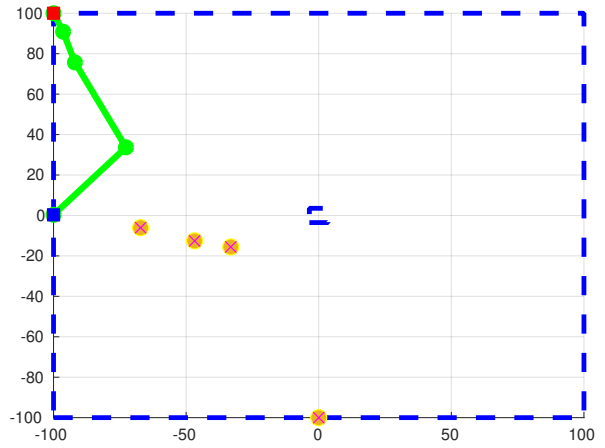
(a) PRM: Path in R-4



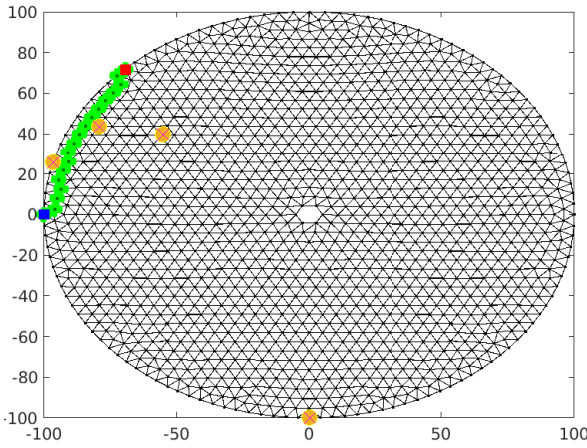
(b) PRM: Path in R-1



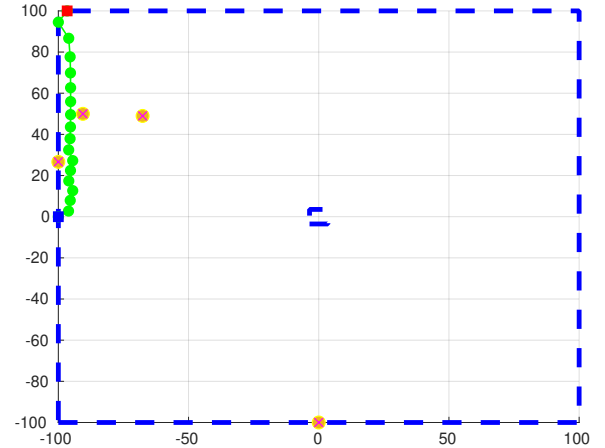
(c) RRT: Path in R-4



(d) RRT: Path in R-1

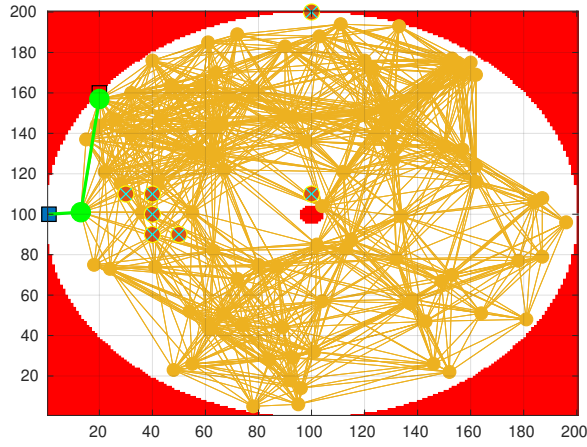


(e) CFDMPP: Path in R-4

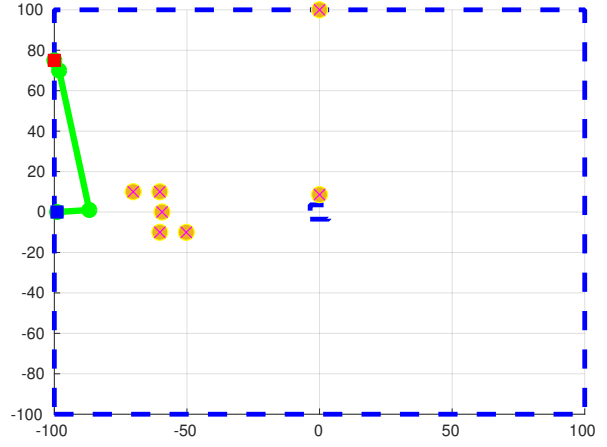


(f) CFDMPP: Path in R-1

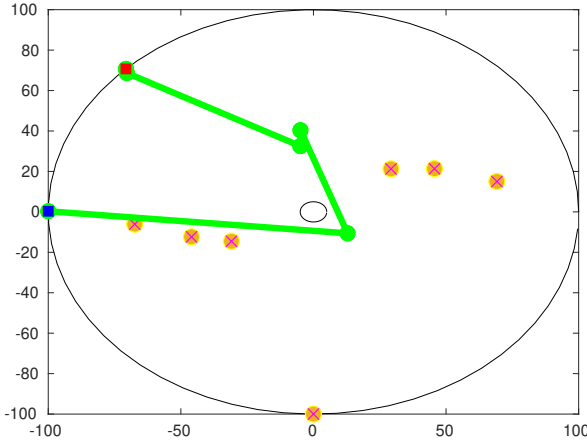
Figure D.2: Case 4: Paths generated by PRM, RRT and CFDMPP with 4 obstacles/poles in the environment; paths shown in R-4 and R-1.



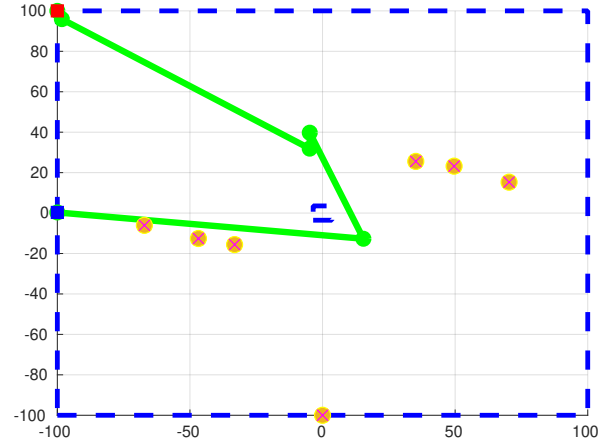
(a) PRM: Path in R-4



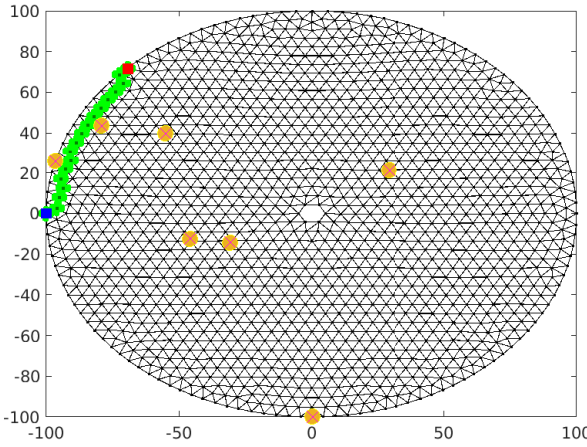
(b) PRM: Path in R-1



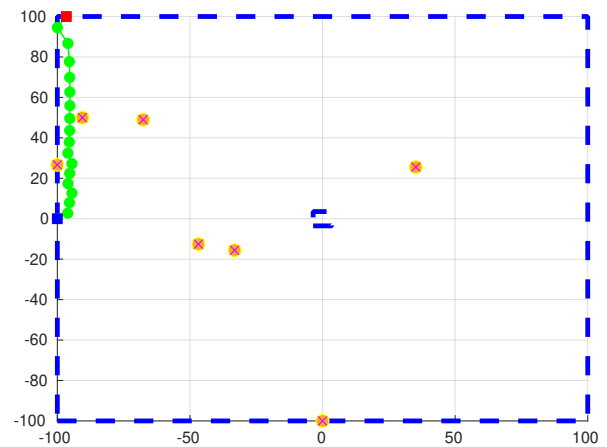
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure D.3: Case 4: Paths generated by PRM, RRT and CFDMPP with 7 obstacles/poles in the environment; paths shown in R-4 and R-1.

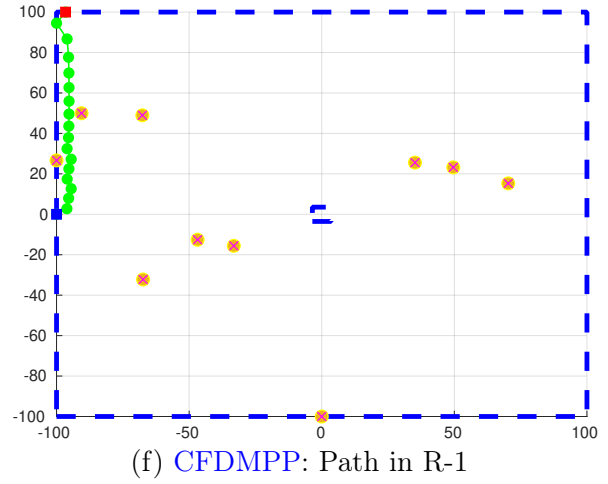
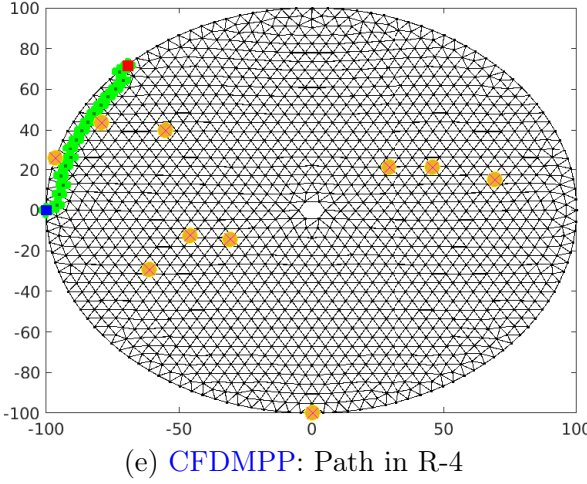
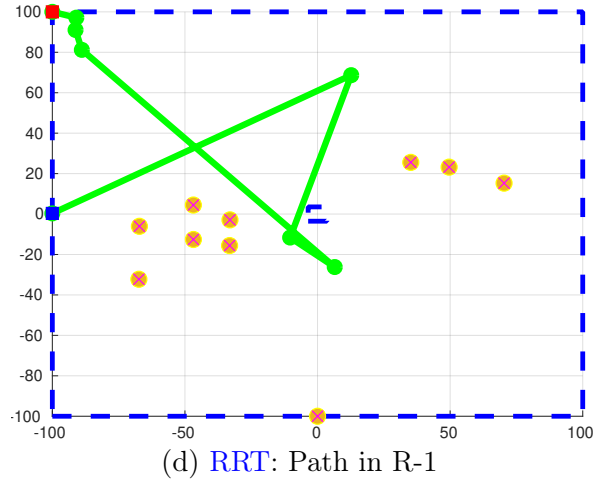
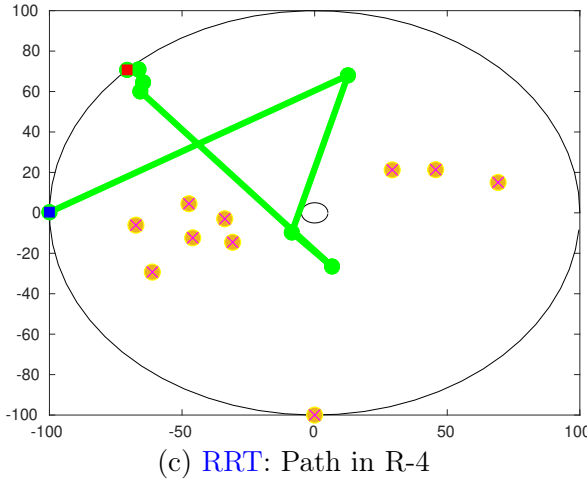
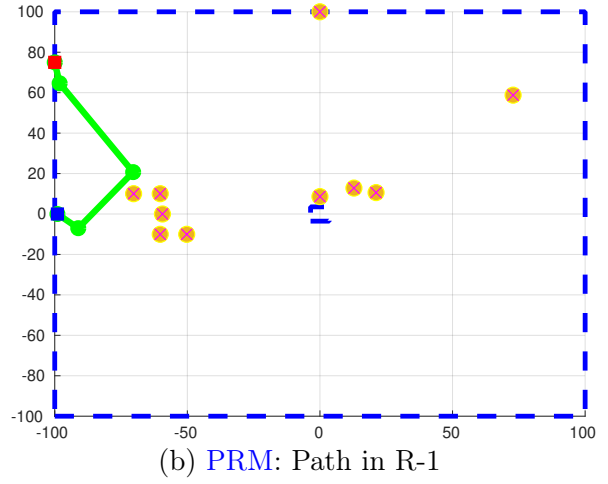
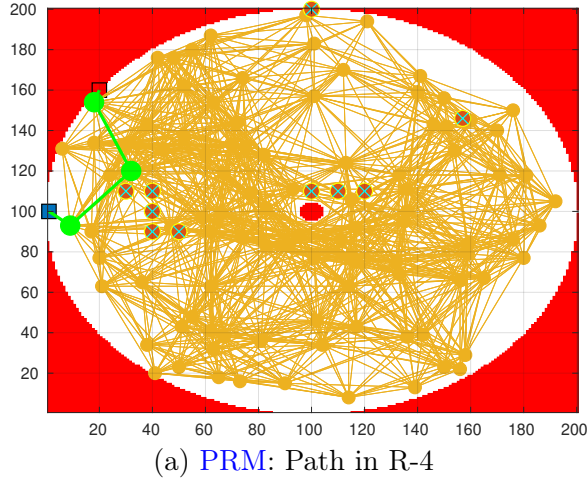


Figure D.4: Case 4: Paths generated by PRM, RRT and CFDMPP with 10 obstacles/poles in the environment; paths shown in R-4 and R-1.

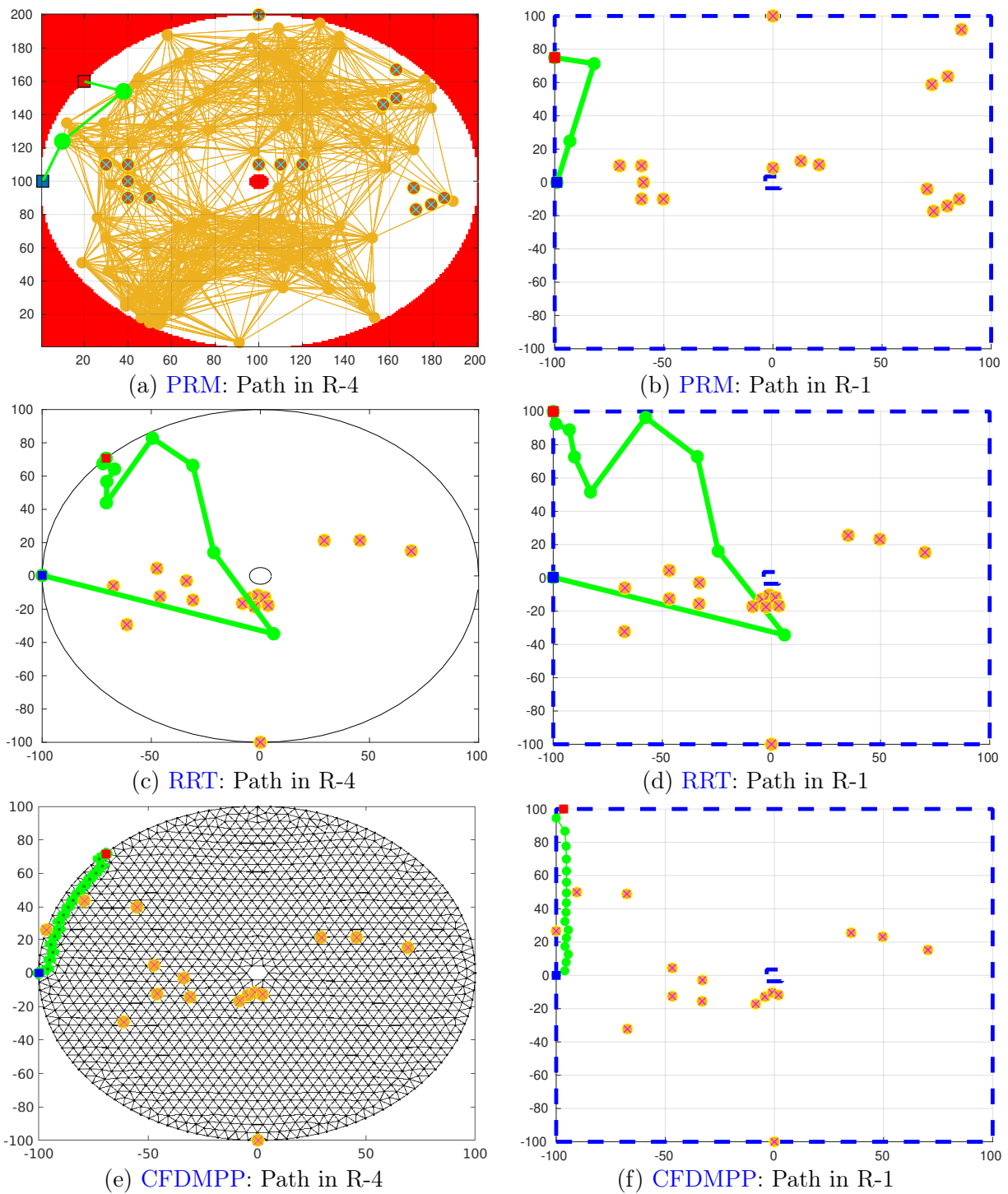


Figure D.5: Case 4: Paths generated by PRM, RRT and CFDMPP with 16 obstacles/poles in the environment; paths shown in R-4 and R-1.

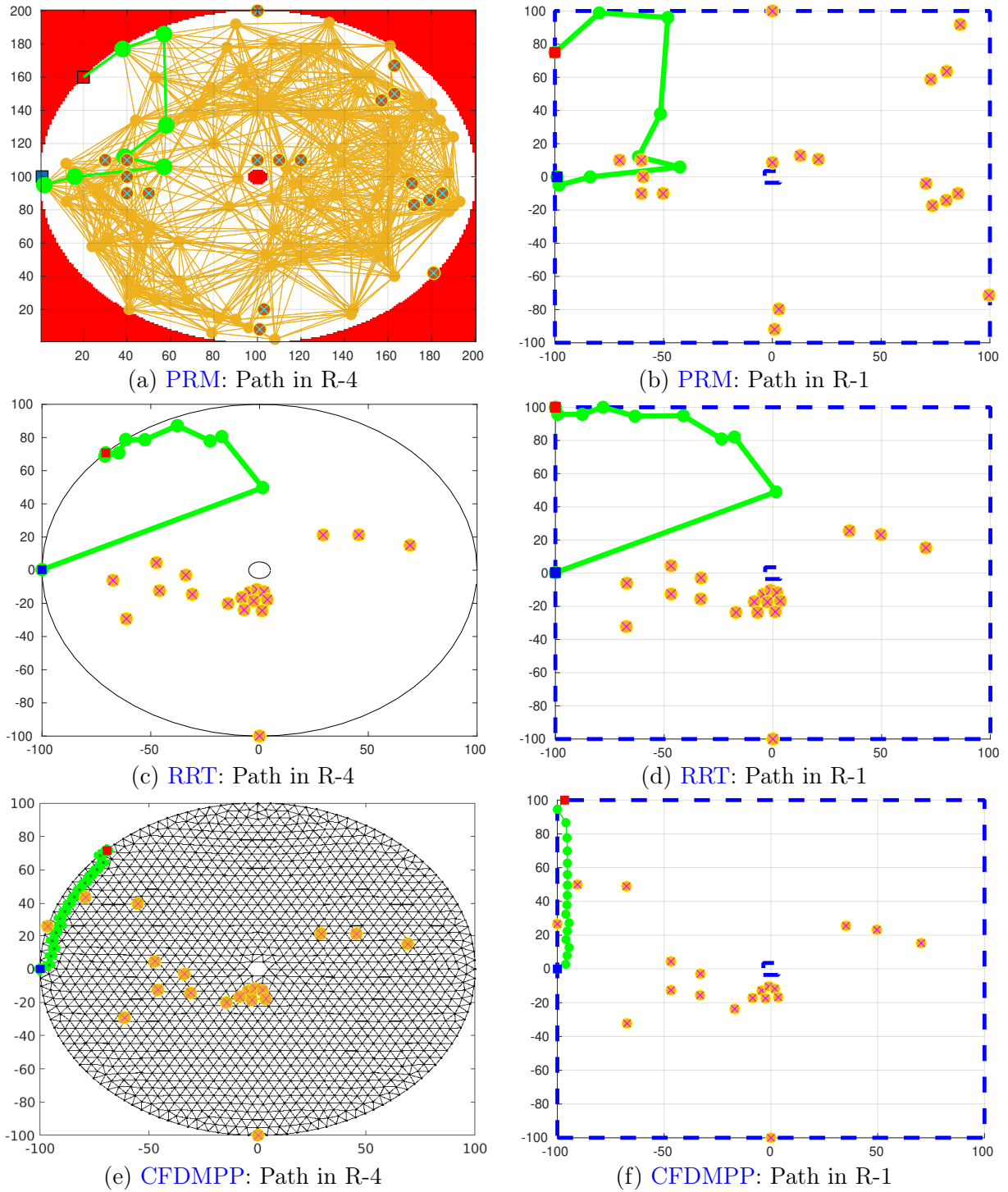
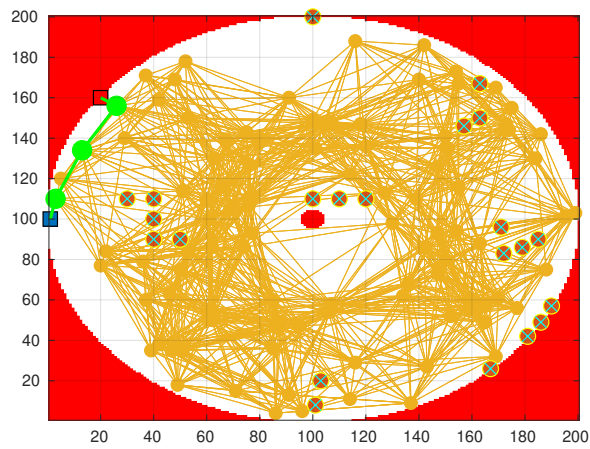
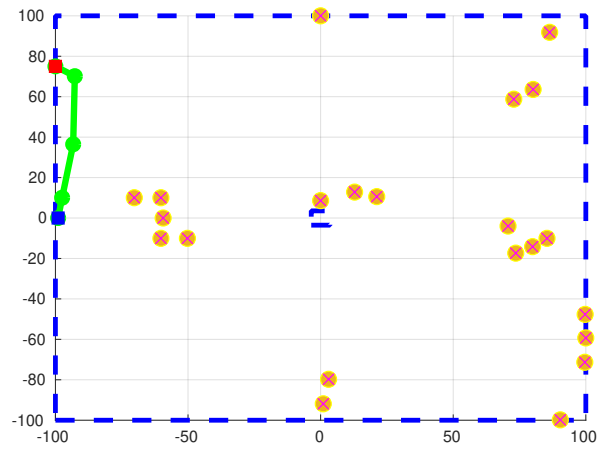


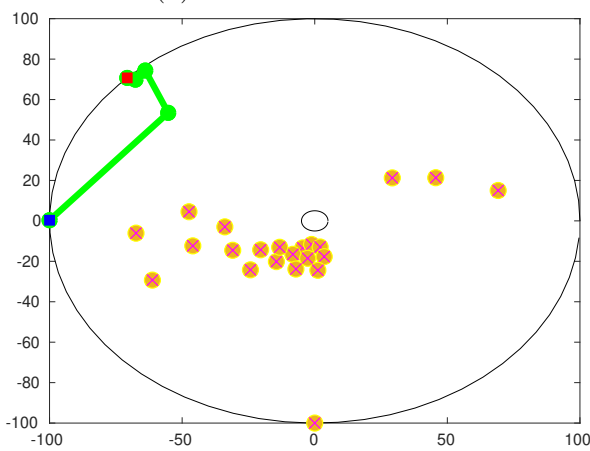
Figure D.6: Case 4: Paths generated by PRM, RRT and CFDMPP with 19 obstacles/poles in the environment; paths shown in R-4 and R-1.



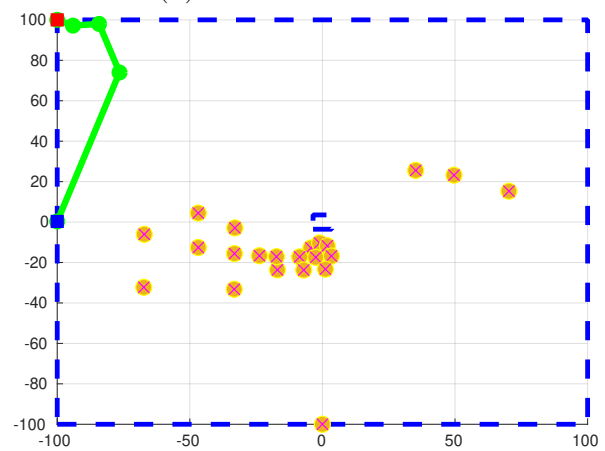
(a) PRM: Path in R-4



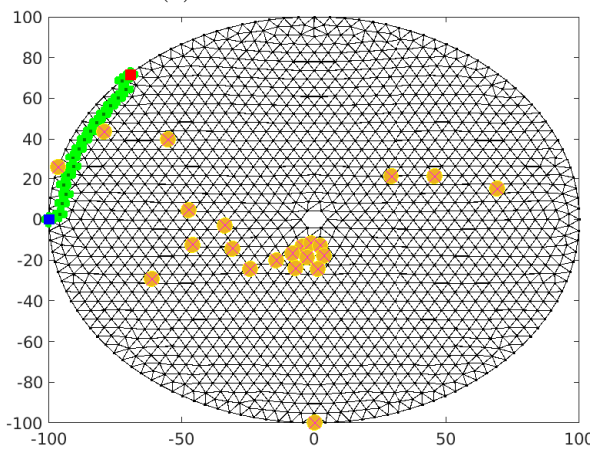
(b) PRM: Path in R-1



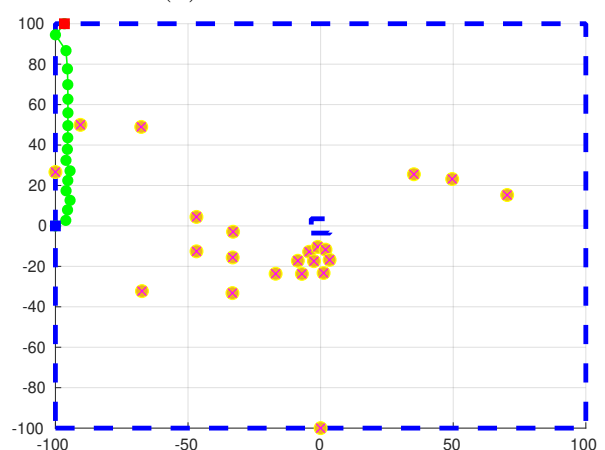
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure D.7: Case 4: Paths generated by PRM, RRT and CFDMPP with 22 obstacles/poles in the environment; paths shown in R-4 and R-1.

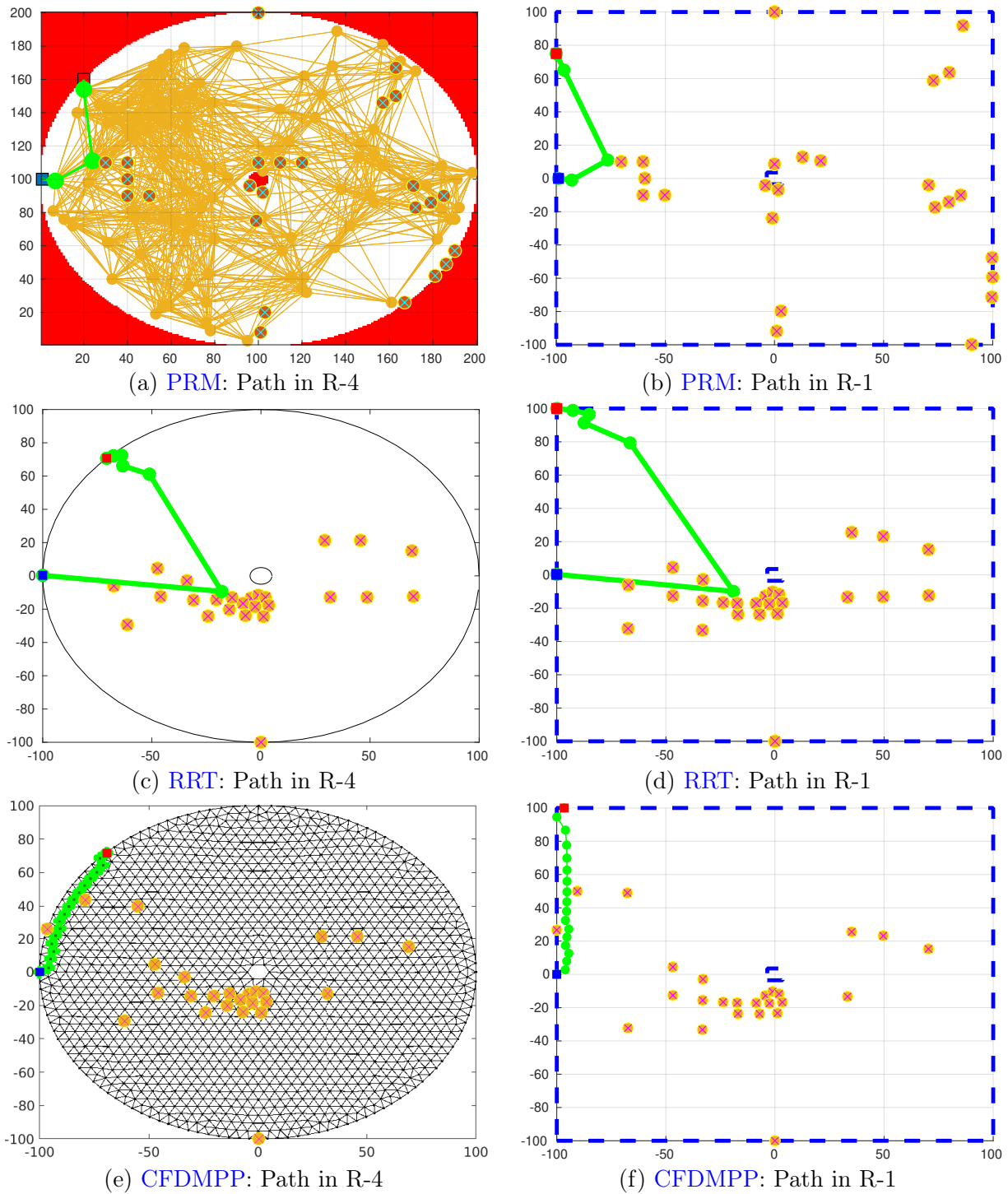
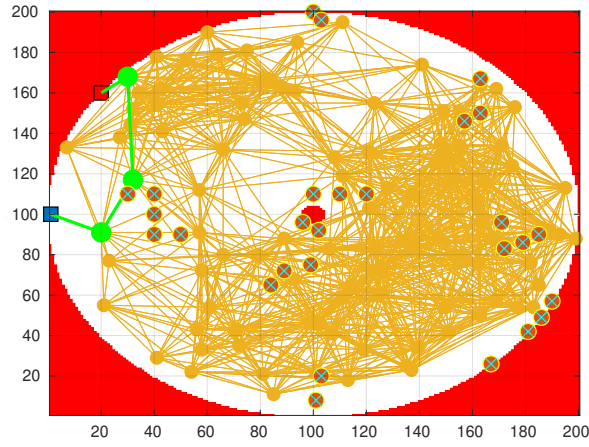
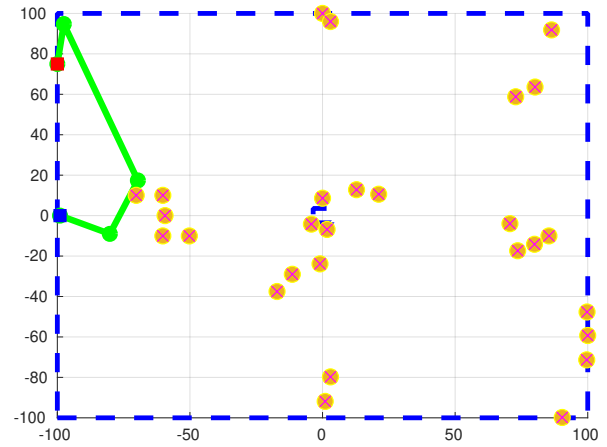


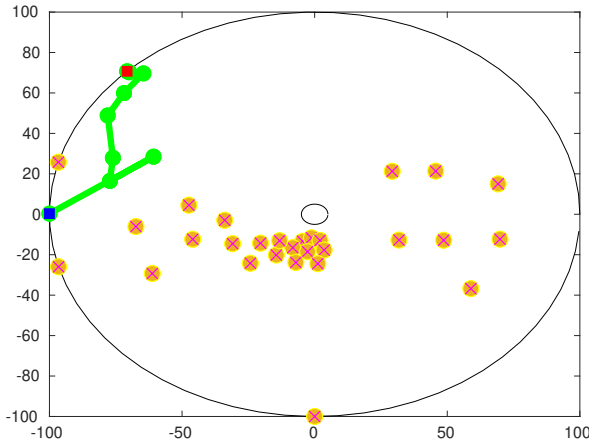
Figure D.8: Case 4: Paths generated by PRM, RRT and CFDMPP with 25 obstacles/poles in the environment; paths shown in R-4 and R-1.



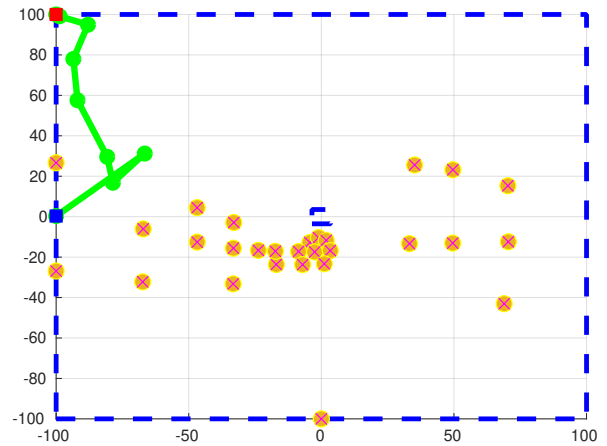
(a) PRM: Path in R-4



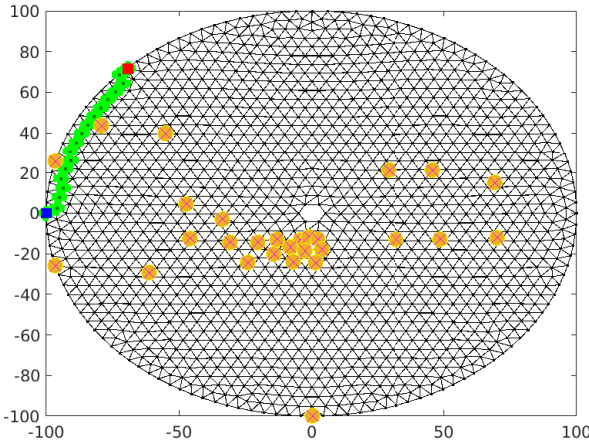
(b) PRM: Path in R-1



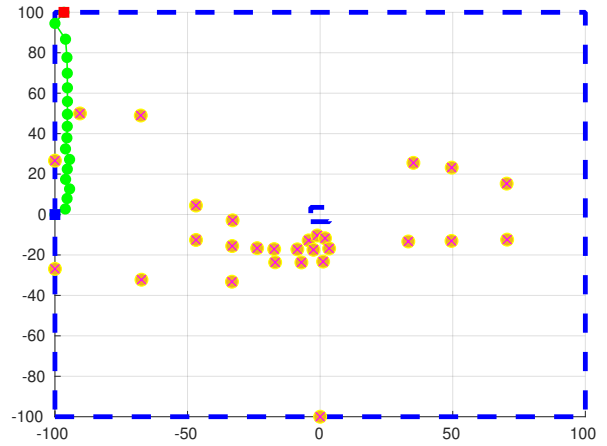
(c) RRT: Path in R-4



(d) RRT: Path in R-1



(e) CFDMPP: Path in R-4



(f) CFDMPP: Path in R-1

Figure D.9: Case 4: Paths generated by PRM, RRT and CFDMPP with 28 obstacles/poles in the environment; paths shown in R-4 and R-1.

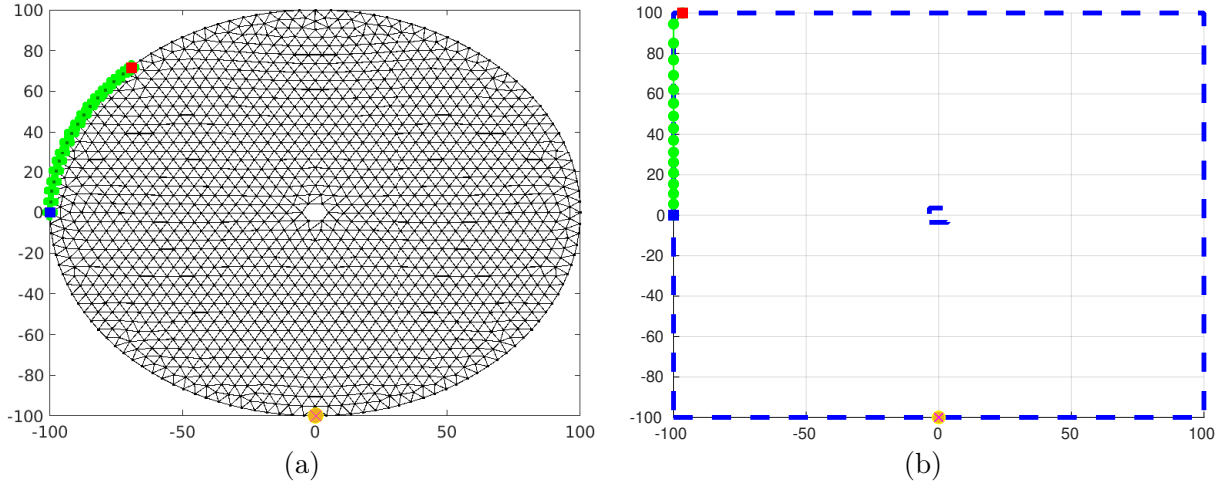


Figure D.10: Case 4: CFDMPP in the case of 1 poles (first group of DCs). Only 1 path class is possible, so no blocking is seen. (a) Path in R-4. (b) Path in R-1.

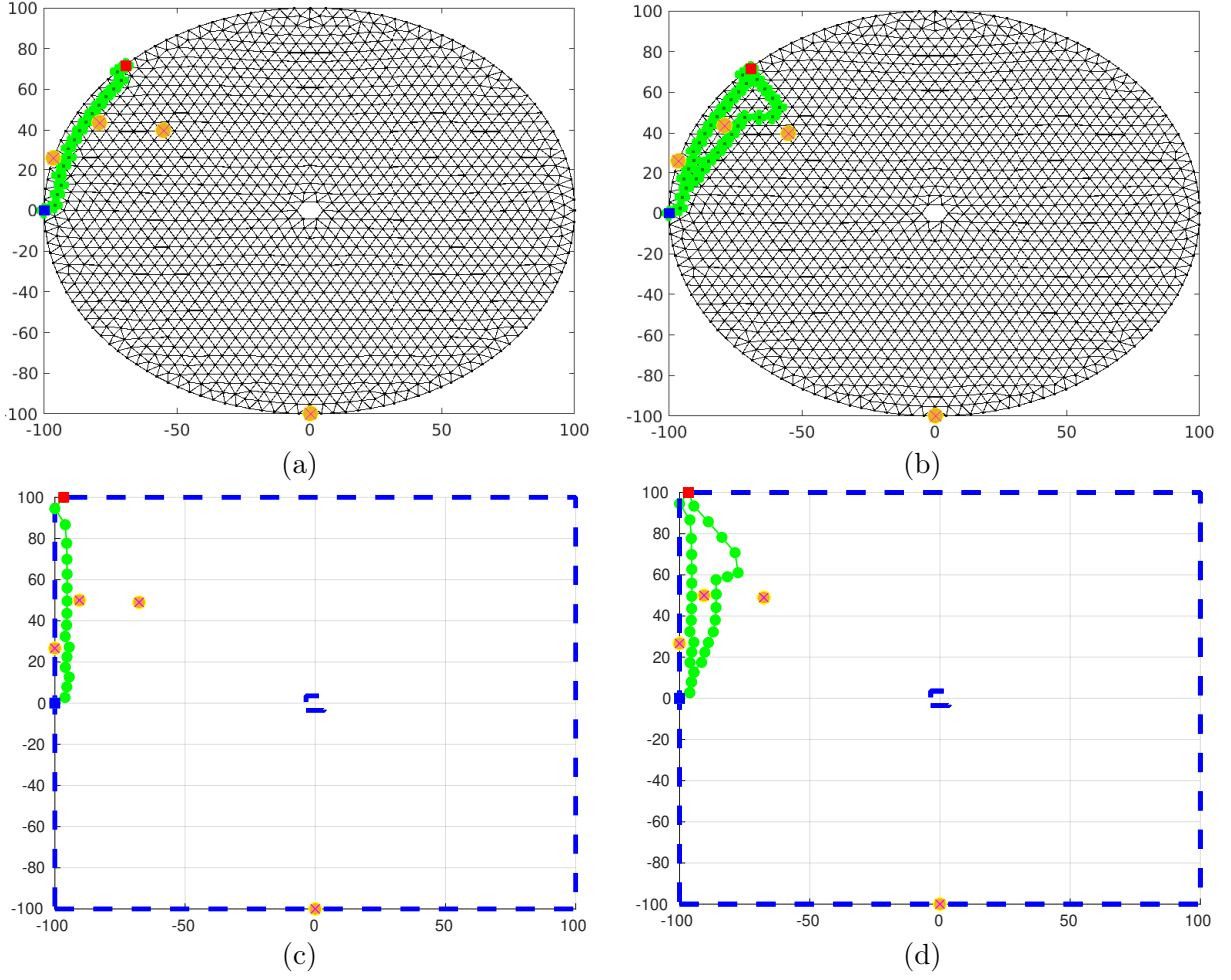


Figure D.11: Case 4: CFDMPP in the case of 4 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.325). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

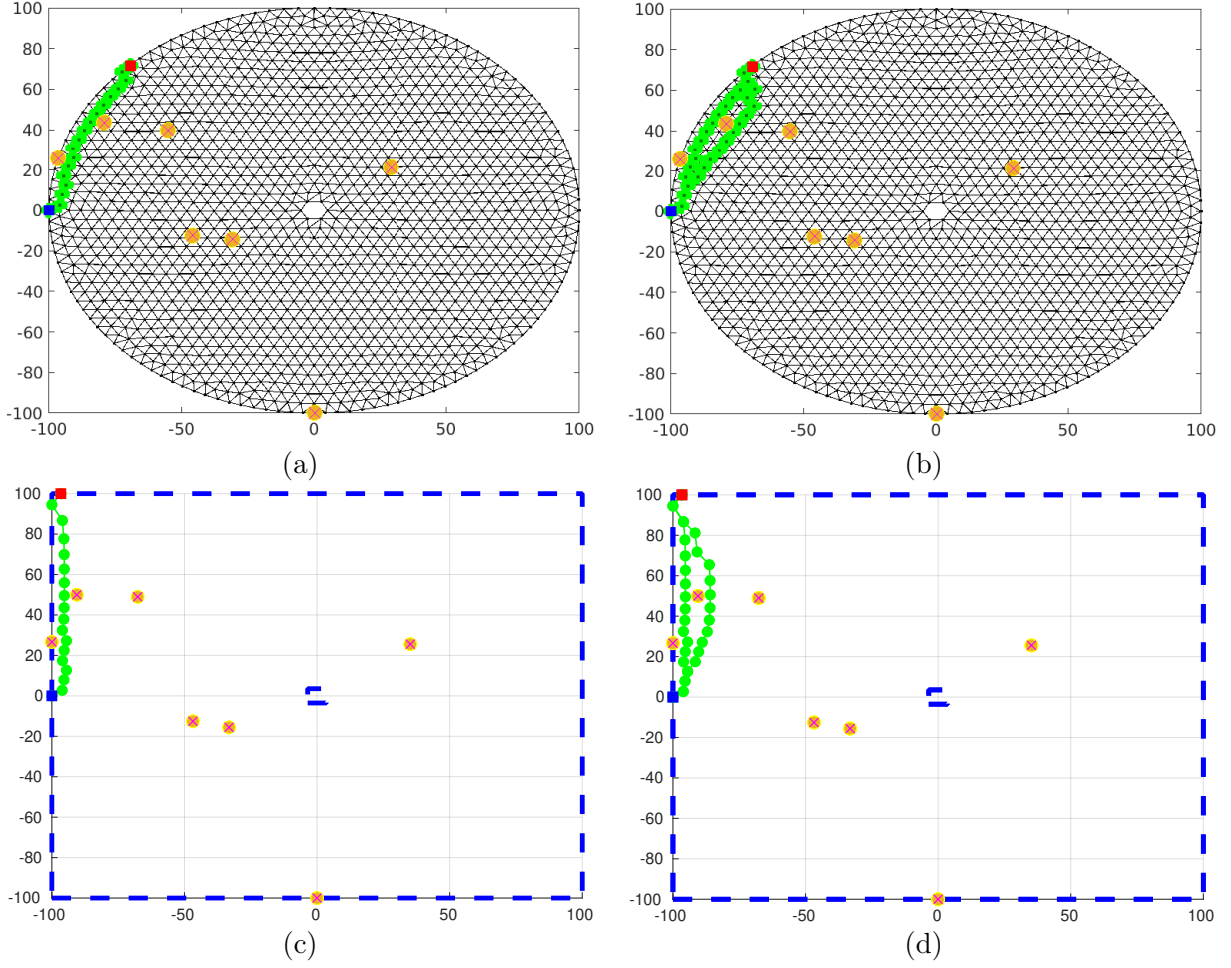


Figure D.12: Case 4: CFDMPP in the case of 7 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.5). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

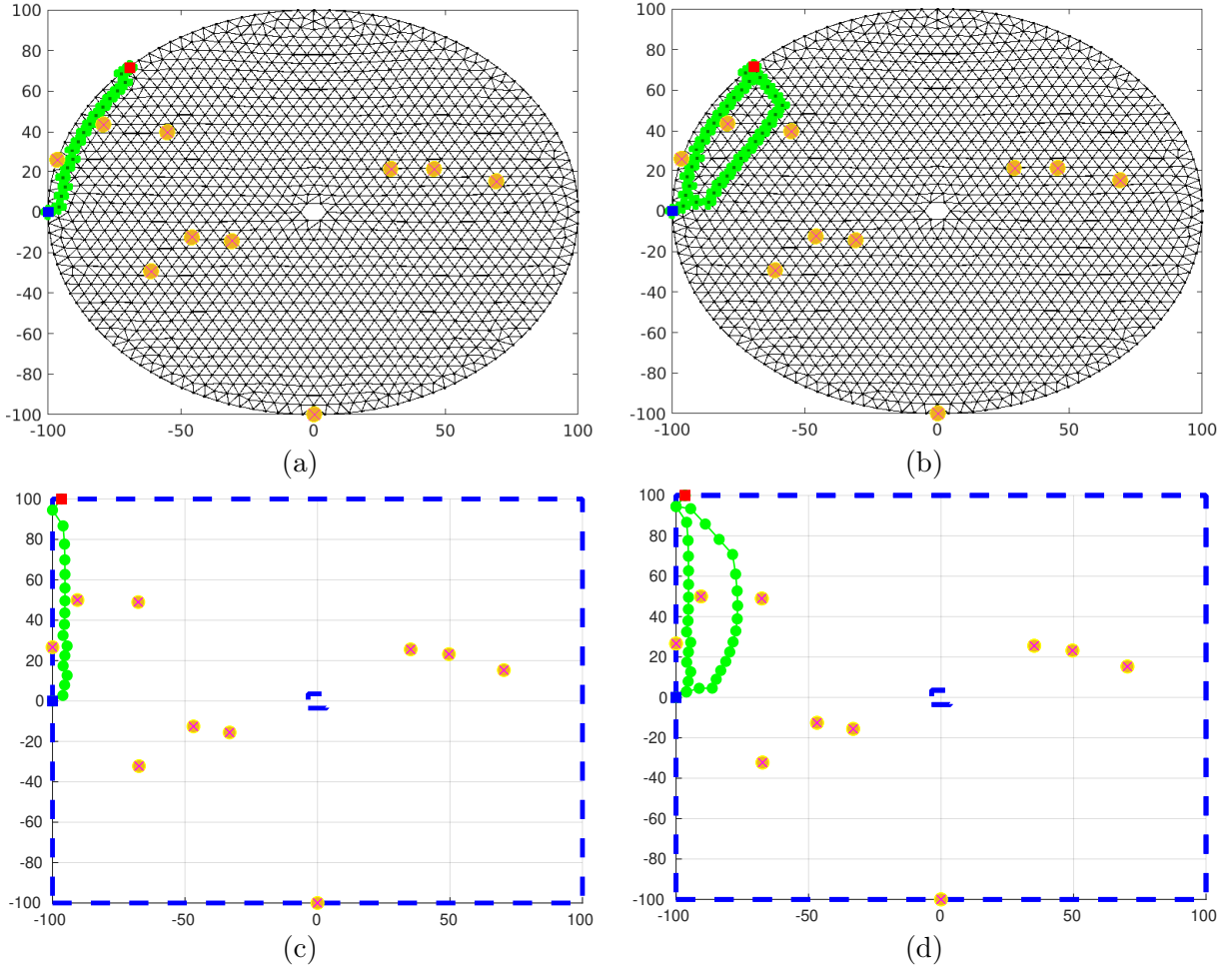


Figure D.13: Case 4: CFDMPP in the case of 10 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.518). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

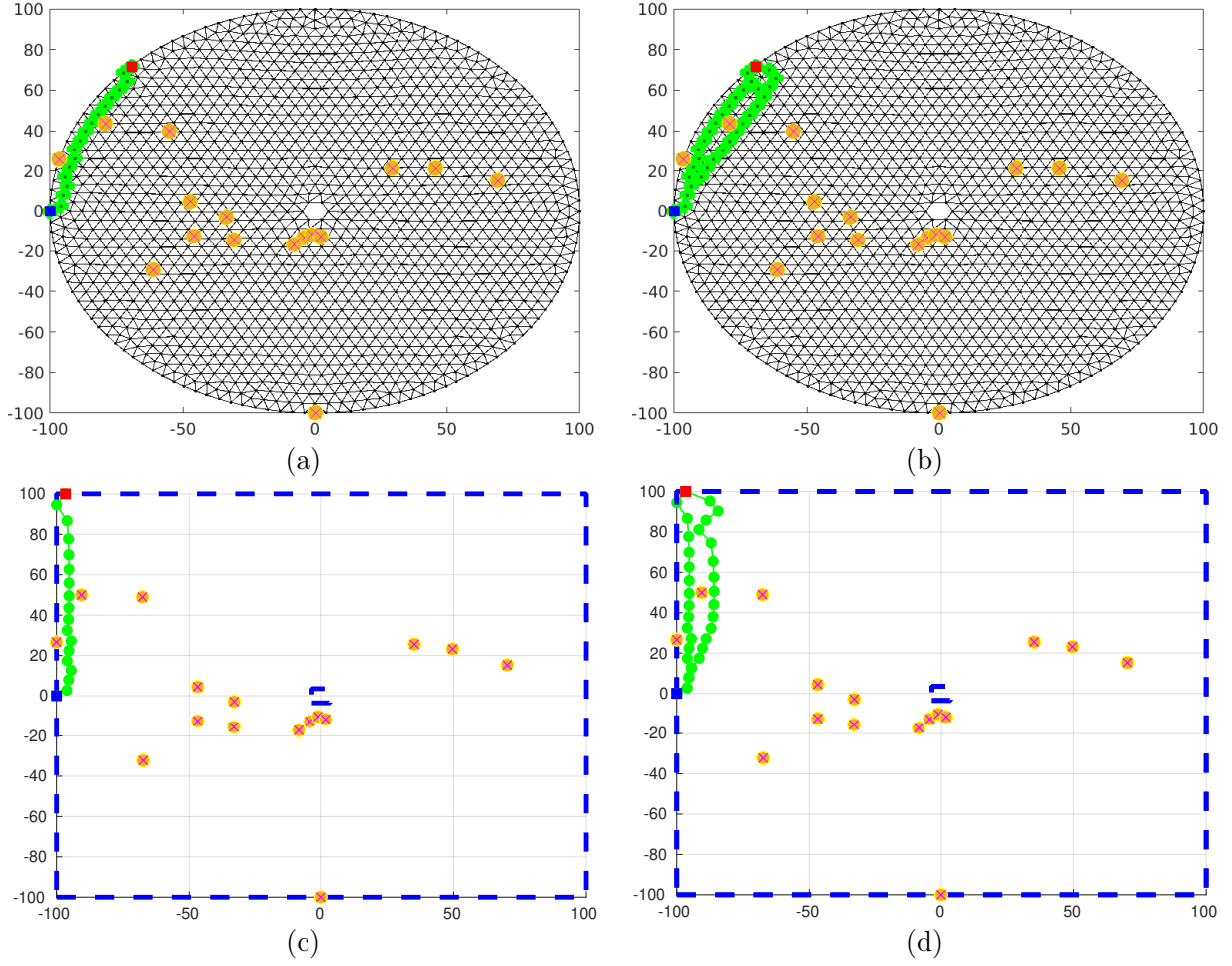


Figure D.14: Case 4: CFDMPP in the case of 16 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.2). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

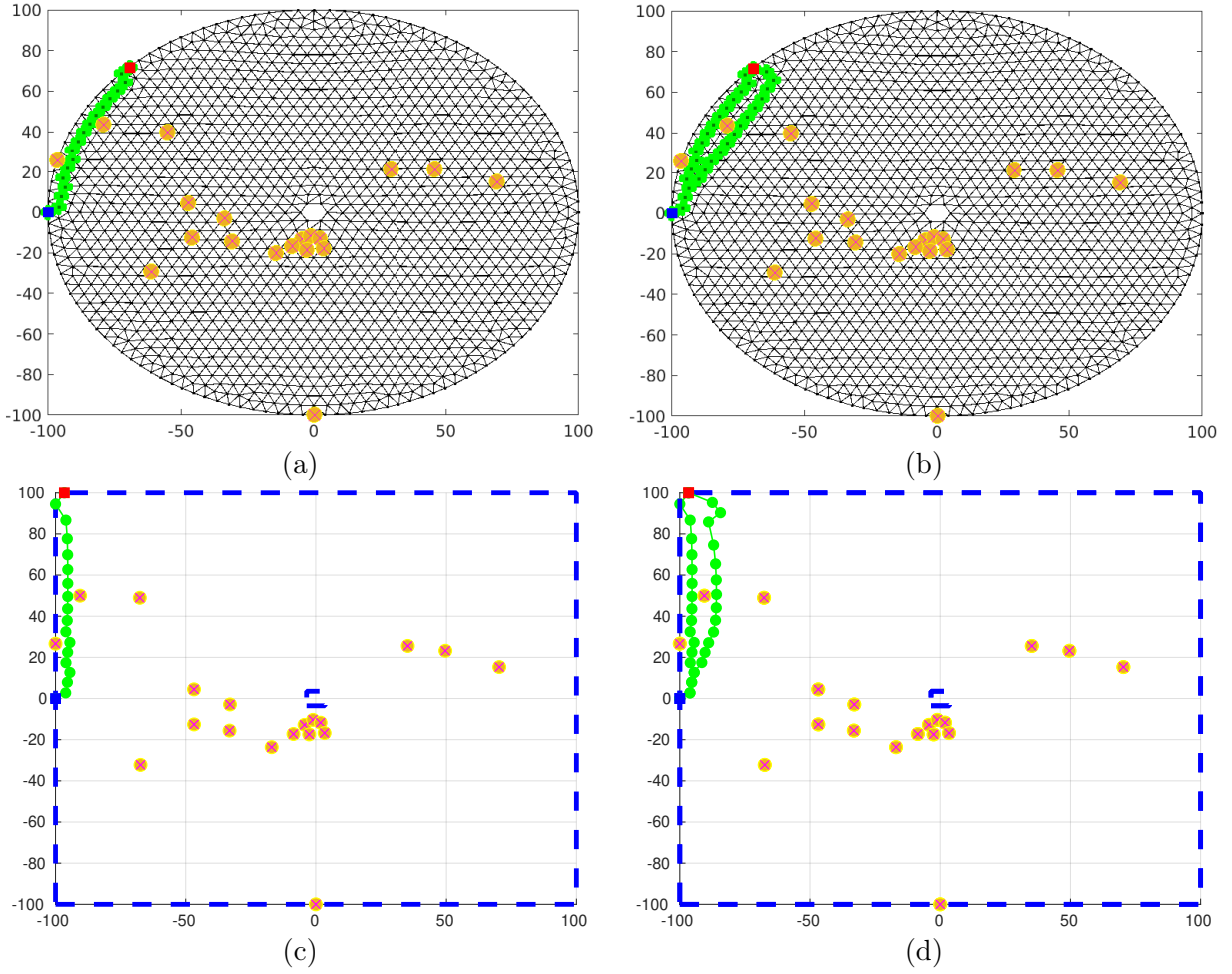


Figure D.15: Case 4: CFDMPP in the case of 19 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

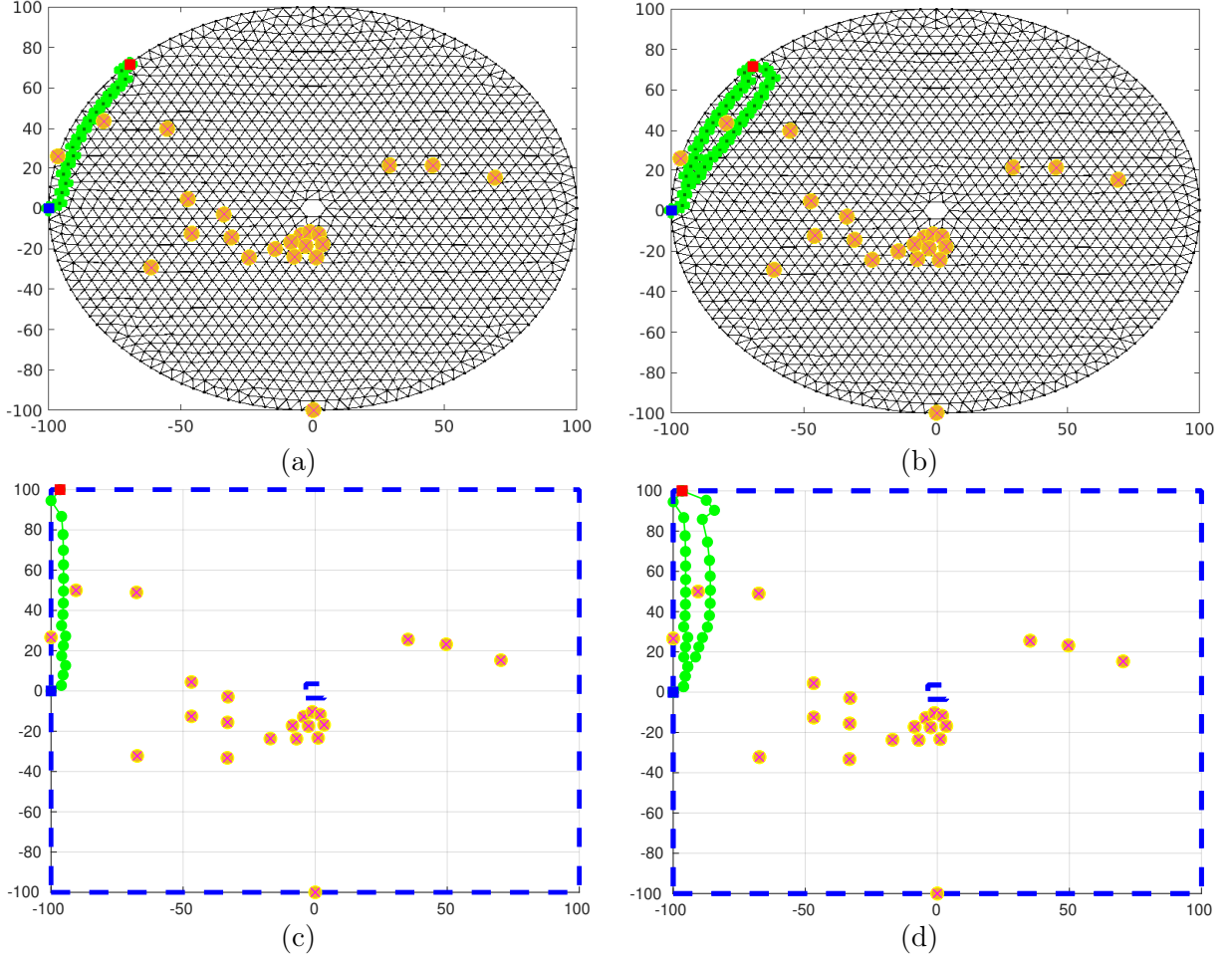


Figure D.16: Case 4: CFDMPP in the case of 22 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.1). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

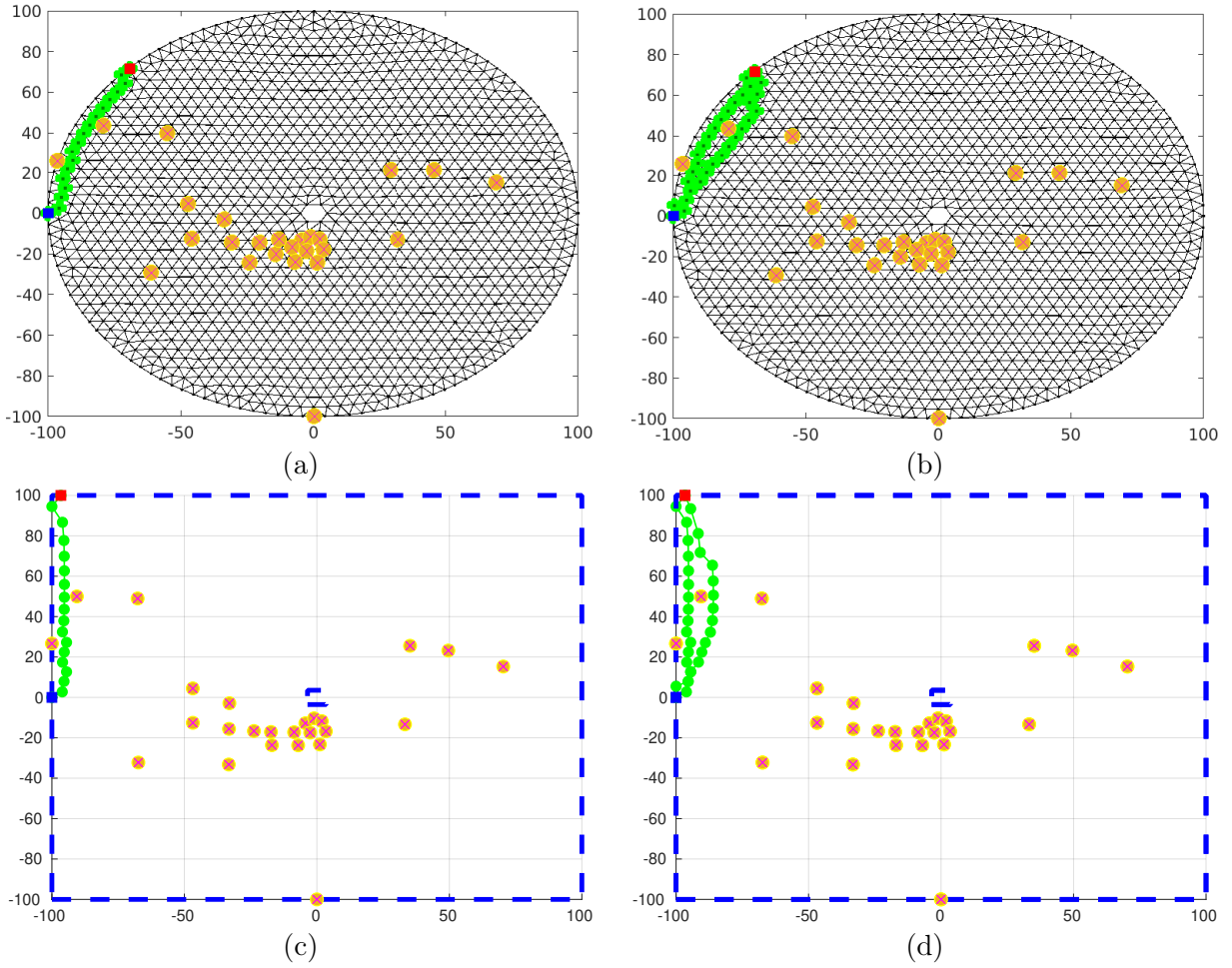


Figure D.17: Case 4: CFDMPP in the case of 25 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.35). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

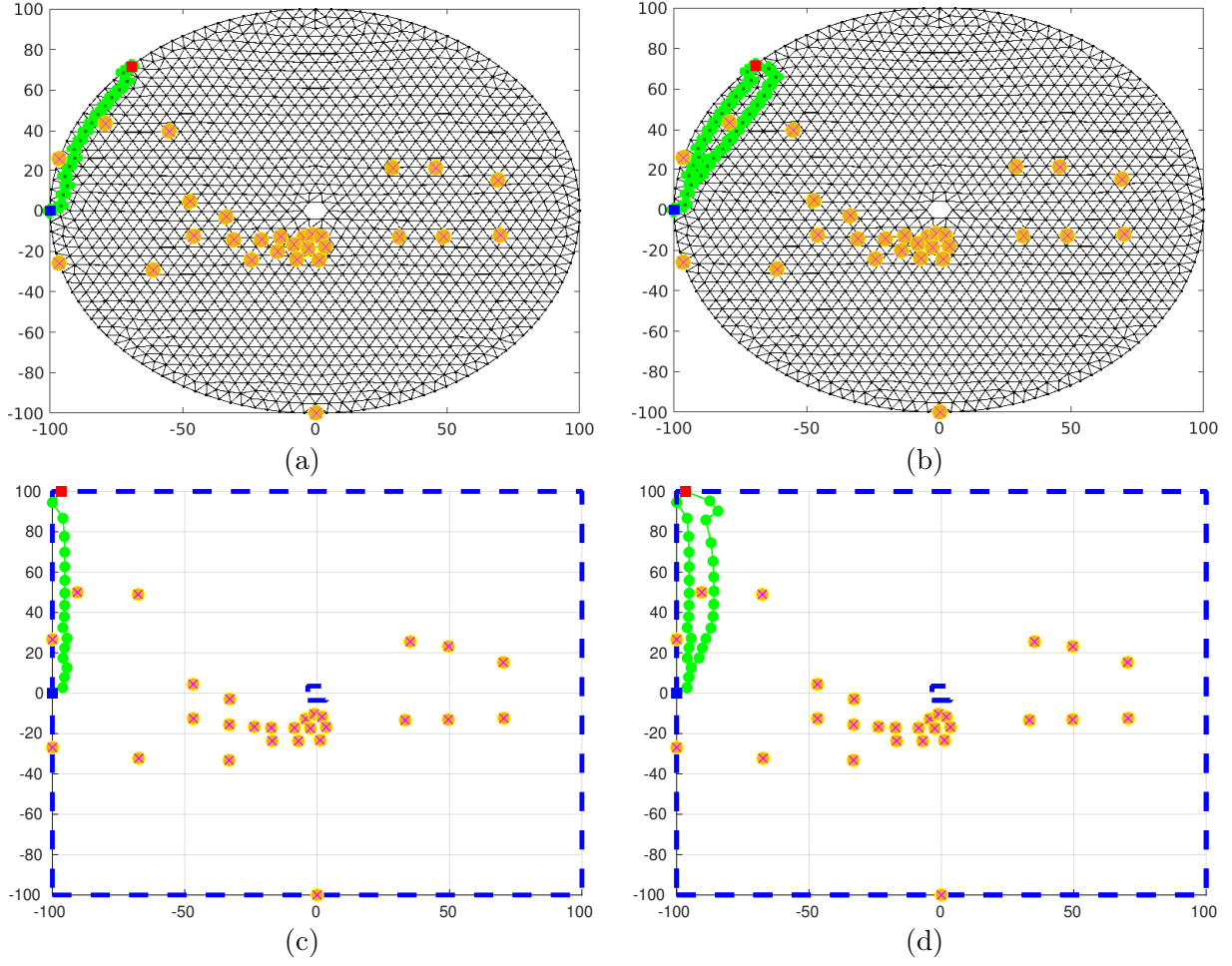


Figure D.18: Case 4: CFDMPP in the case of 28 poles (first group of DCs) and 1 blocked path (second group of DCs HB Threshold = 0.3). (a) Path in R-4. (b) Blocked path and resulting path in R-4. (c) Path in R-1. (d) Blocked path and resulting path in R-1.

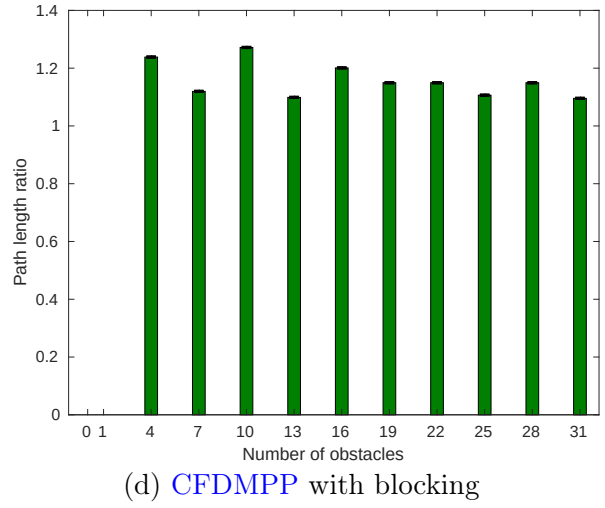
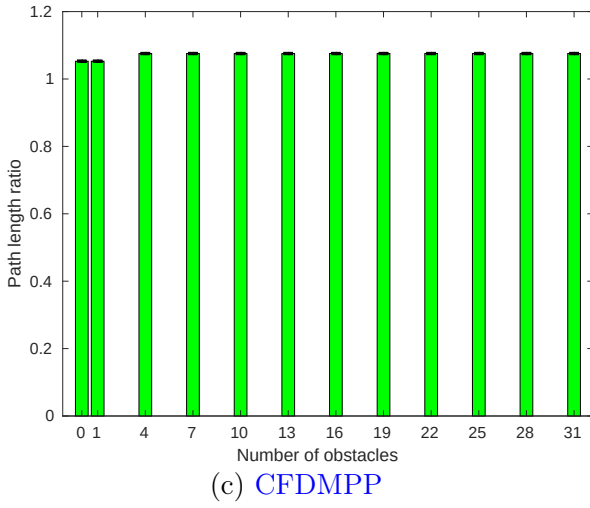
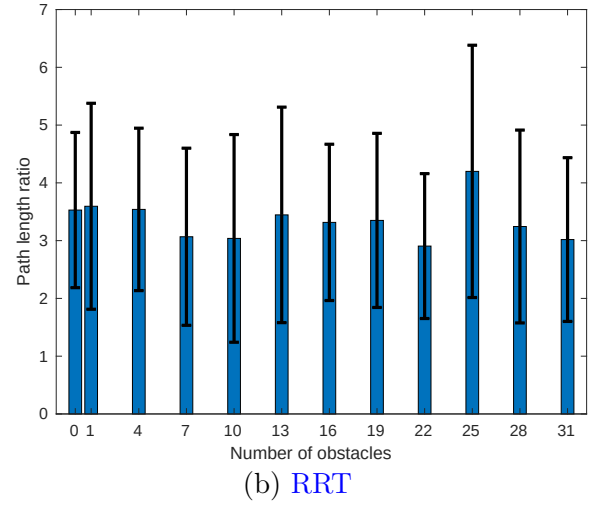
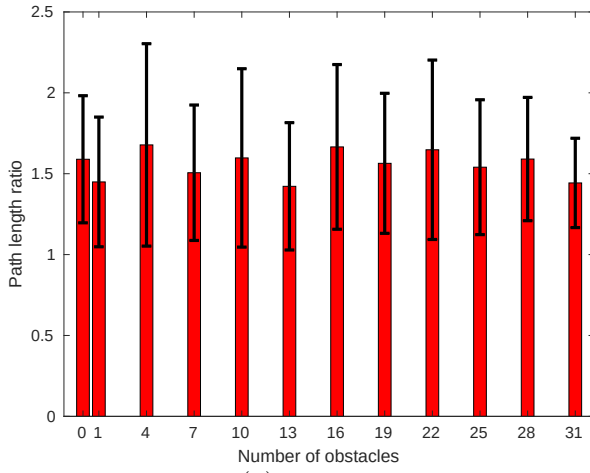
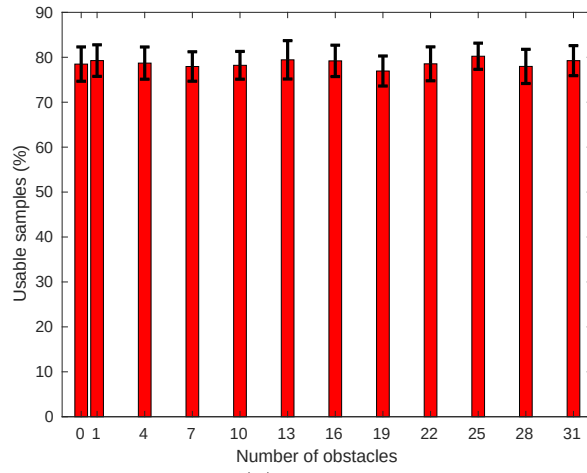
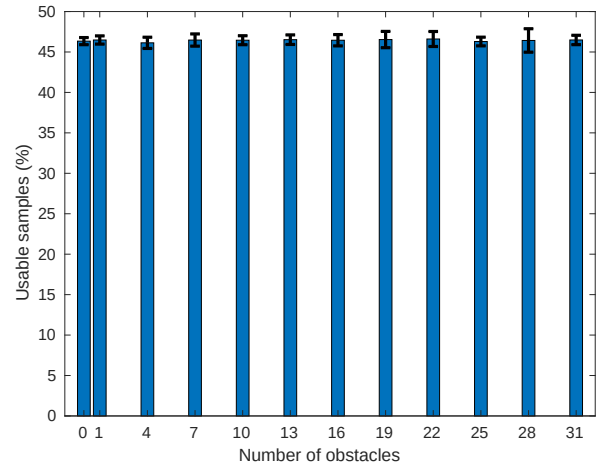


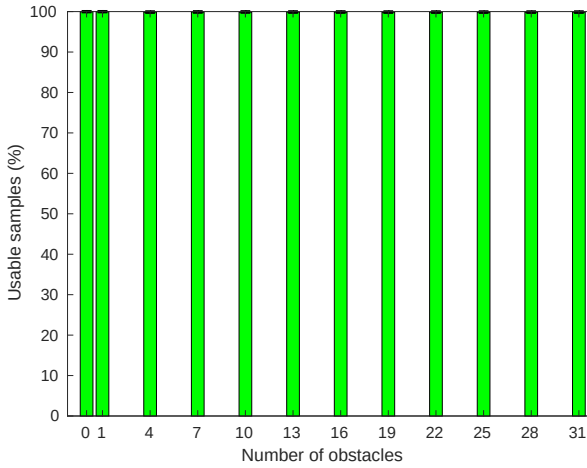
Figure D.19: Case 4: Ratio of observed to ideal path lengths computed across 20 runs each for 0–31 obstacles/poles.



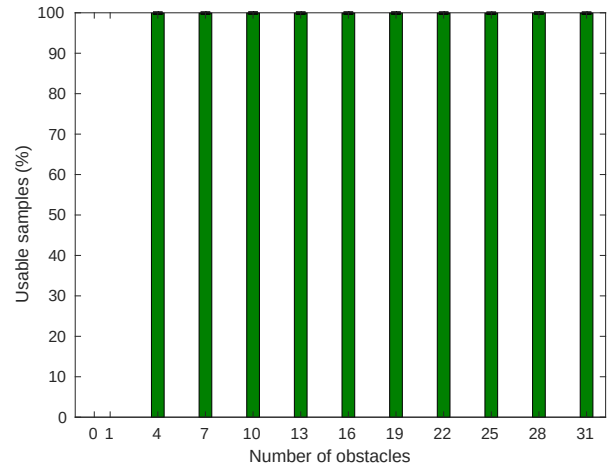
(a) PRM



(b) RRT



(c) CFDMPP



(d) CFDMPP with blocking

Figure D.20: Case 4: Percentage of usable samples computed across 20 runs each for 0–31 obstacles/poles.