The Version of Record is available online at: http://dx.doi.org/10.1007/s41315-023-00300-3

Constraint-Free Discretized Manifold-Based Path Planner

Sindhu Radhakrishnan[1] and Wail Gueaieb[1*]

[1*]School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Avenue, Ottawa, K1N 6N5, ON, Canada.

*Corresponding author(s). E-mail(s): wgueaieb@uottawa.ca;
Contributing authors: sradh006@uottawa.ca;

Abstract

Autonomous robotic path planning in partially known environments, such as warehouse robotics, deals with static and dynamic constraints. Static constraints include stationary obstacles, robotic and environmental limitations. Dynamic constraints include humans, robots and dis/appearance of anticipated dangers, such as spills. Path planning consists of two steps: First, a path between the source and target is generated. Second, path segments are evaluated for constraint violation. Sampling algorithms trade memory for maximal map representation. Optimization algorithms stagnate at non-optimal solutions. Alternatively, detailed grid-maps view terrain/structure as expensive memory costs. The open problem is thus to represent only constraint-free, navigable regions and generating anticipatory/reactive paths to combat new constraints. To solve this problem, a Constraint-Free Discretized Manifolds-based Path Planner (CFDMPP) is proposed in this paper. The algorithm's first step focuses on maximizing map knowledge using manifolds. The second uses homology and homotopy classes to compute paths. The former constructs a representation of the navigable space as a manifold, which is free of apriori known constraints. Paths on this manifold are constraint-free and do not have to be explicitly evaluated for constraint violation. The latter handles new constraint knowledge that invalidate the original path. Using homology and homotopy, path classes can be recognized and avoided by tuning a design parameter, resulting in an alternative constraint-free path. Path classes on the discretized constraint-free manifold characterize numerical uniqueness of paths around constraints. This designation is what allows path class

characterization, avoidance, and querying of a new path class (multiple classes with tuning), even when constraints are simply anticipatory.

Keywords: Path planning, Robotics, Manifolds, Topology

Abbreviations

CFDM Constraint-Free Discretized Manifold 5, 24, 47, 48, 50, 51, 53, 54, 57, 58, 63
CFDMPP Constraint-Free Discretized Manifolds-based Path Planner 5, 11, 24, 30, 31, 48, 49, 51, 53, 54, 57–64
CFM Constraint-Free Manifold 5, 24, 25, 27, 31, 32, 34, 47, 53, 63
CS Constraint Set 27–31, 33
DC Dynamic Constraint 11, 24, 25, 30, 31, 33, 47, 48, 50, 53, 54, 56, 64
DCS Disjoint Constraint Set 27, 28, 30–34, 36–39, 41, 43, 46–48, 52
EA Evolutionary Algorithm 4
EF Evaluative Factor 32, 33
HBM Homotopy Based Method 4
ICE Inner Constraint Edge 46, 53
IFTM Inverse Function Theorem for Manifolds 10
MBM Model Based Methods 4
MPC Model Predictive Control 4
NF Navigation Function 4
OA Optimization Algorithm 4
OCE Outer Constraint Edge 46, 53
PPM Path Planning Manifold 7, 11, 23, 32
PPP Path Planning Problem 5, 10–16, 24, 26, 27, 29, 32, 34, 48, 62
PPS Path Planning Space 4–7, 10–12, 15, 16, 23–40, 46, 47, 51, 52, 57, 62, 63
PPPS Primary Path Planning Space 25–27, 30, 32, 52
PRM Probabilistic Road Map 59, 61–63
PSM Product Smooth Manifold 9
RRT Rapidly exploring Random Tree 59, 61–63
SA Sampling Algorithm 4, 59, 61, 62
SC Static Constraint 24, 26–31, 47, 48, 51, 52
SM Smooth Manifold 9, 10
SPPS Secondary Path Planning Space 25–27, 30, 32
TM Topological Manifold 6–11
UGV Unmanned Ground Vehicle 53, 54, 57

List of Latin Symbols

$\mathcal{C}$ Configuration space 5, 9, 23
$\mathcal{CO}$ Convex obstacle region in configuration space 30
CS_r Constraint set r 28, 29

DCS_r Disjoint constraint set r 30, 31, 34
$DCSs$ Set of all disjoint constraint sets of all path planning spaces 30
$DCSs^{(\cdot)}$ Set of all disjoint constraint sets of space $(\cdot)$ 30–32, 34
n_{ML} Number of SCs classified as mechanical limits (ML) 27
n_{NG} Number of SCs classified as no-go zones (NG) 27, 52
n_{OB} Number of SCs classified as obstacles (OB) 27
n_{SG} Number of SCs classified as singularities (SG) 27
n_{SC} Total number of SCs 27, 52
$\boldsymbol{node}^{start}$ Starting point in CFDM 48–50
$\boldsymbol{curr}$ Current node in CFDM 48–50
$\boldsymbol{nbr}$ Neighbour node in CFDM 48–50
$\mathcal{P}$ Path planning space 5, 6, 12, 26
$\mathcal{P}^{mechanical\ limits} \subseteq \mathcal{P}$ Space corresponding to the mechanical limits of the robot (e.g., joint limits) 6
$\mathcal{P}^{no\text{-}go\ zones} \subseteq \mathcal{P}$ Space corresponding to the no-go zones 6
$\mathcal{P}^{obstacles} \subseteq \mathcal{P}$ Space corresponding to the obstacles 6
$\mathcal{P}^{singularities} \subseteq \mathcal{P}$ Space corresponding to the singularities of the robot 6
$\mathcal{P}^{constraint} \equiv \mathcal{P}^{\text{mechanical limits}} \cup \mathcal{P}^{\text{no-go-zones}} \cup \mathcal{P}^{\text{obstacles}} \cup \mathcal{P}^{\text{singularities}}$ Space corresponding to all constraints 6, 23
$\mathcal{P}^{free} \equiv \mathcal{P} \setminus \mathcal{P}^{\text{constraint}}$ Constraint-free Space 6, 12
$\boldsymbol{p} \in \mathcal{P}$ A point in $\mathcal{P}$ 5, 36, 39–43, 46, 52, 53
$\boldsymbol{p}^{start} \in \mathcal{P}$ Starting point in $\mathcal{P}$ 12–14, 23–25, 48, 50
$\boldsymbol{p}^{goal} \in \mathcal{P}$ Destination point in $\mathcal{P}$ 12–14, 23–25, 48, 50
$\boldsymbol{p}_{path} \in \mathcal{P}$ Path connecting $\mathbf{p}^{\text{start}}$ to $\mathbf{p}^{\text{goal}}$ in $\mathcal{P}$ 50, 51
$\mathcal{S}_{surface}$ A Surface 10, 11
SC_k Static constraint k 28–31
$\boldsymbol{T}_p\mathcal{S}$ Tangent Space at a point p on a surface 11
$\mathcal{TS}$ Topological space of set $\mathcal{S}$ and topology $\mathcal{T}$ 11, 35–41
$\mathcal{W}$ Workspace 5, 9, 52

List of Greek Symbols

Δ_{CS} Global proximity threshold between constraints 28–31, 34, 52
Δ_{CS}^{ij} Measured proximity between constraints i and j 28–30
δ Boundary. For instance, δx denotes the boundary of x 29

1 Introduction

Typical autonomous robots can be used either for exploring unknown environments or navigating known environments. In exploration, autonomy occurs in the form of Simultaneous Localization and Mapping (SLAM) [1, 2]. In known environments, the space is comprised of constraint-space and constraint-free navigable space. Static constraints include stationary limitations, whereas, dynamic constraints imply the sudden presence/absence of constraints. Path planning in an environment with a known map and potential to encounter unknown constraints is the main focus of this article. The motivation stems

from path planning needs in a warehouse/factory floor setting. A solution to such a path planning problem needs to have the ability to: query the same map multiple times, produce the shortest path and an alternative path, should the original path become unsuitable.

Path planning algorithms can be distinguished by whether the space is represented as a set of points, or by imposing a mathematical structure, such as topological properties. Both point-set and topology based methods generally consist of two steps. First, a path between the source and target locations is generated. Second, path segments are evaluated for constraint violation. Point-set methods that tackle static constraints consist of Sampling Algorithms (SAs) [1, 3–43], Optimization Algorithms (OAs) [44–55], Model Predictive Controls (MPCs) [47, 56–63] and Evolutionary Algorithms (EAs) [64–67] and Navigation Functions (NFs) [17, 68–73]. Point-set methods that tackle dynamic constraints consist of Model Based Methods (MBM) [74–100]. These methods combine adaptations of OAs, SAs and MPCs to reactively avoid constraints. SAs require periodic evaluation and discard of constraint-free samples. OAs and EAs are minimization procedures that risk stagnating at non-optimal local minima, while MPCs are robot specific. So with OAs, EAs and MPCs, changes in robot model, the environment or start-goal locations requires a re-creation of the map, and the solution. Since MBM use adaptations of SAs, EAs and MPCs, they inherit the same basic pitfalls of wasted samples, local minima and model specificity.

Topology based methods that avoid static constraints may be divided into approaches that parametrize the path planning surface and those that incrementally chart the surface. Parametrizing the Path Planning Space (PPS) or its subsets can be done analytically [69, 101–107], geometrically [43, 108–110], numerically [106, 111–116] or graphically [110, 111, 117–123]. Then, adaptations of SAs, OAs, EAs, MPCs and topological tools such as geodesics are used to plan paths. Charting manifolds involves incremental construction of the PPS or its subset, while simultaneously planning paths on it. Goal biased charting [42, 124–128] charts towards the goal, as opposed to unbiased charting of the PPS [128–132]. Topology based approaches for dynamic constraints typically rely on homotopy theory and are called Homotopy Based Methods (HBMs). These algorithms use the mathematical structure afforded by topology and sometimes manifold theory to impose structure to the PPS. Then, HBMs as originally seen in computational geometry [133], are used to determine potential alternative paths with some variant of graphs [134–137], SAs and OAs [138–143], numerical solutions [144–151], MPCs [152–154] and other analytical variants [155–157]. Topology based methods are more complex than point-set based methods, since imposing the desired mathematical structure confines the number of possible solutions available. 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.

As a solution, we propose the topology based Constraint-Free Discretized Manifolds-based Path Planner (CFDMPP) that achieves the following. The first goal deals with the formulation of all or a subset of the navigable constraint-free space as a Constraint-Free Manifold (CFM). Path planning on the Constraint-Free Discretized Manifold (CFDM) is always constraint-free, without any explicit constraint-evaluation or minimization. Second, the use of homology and homotopy classes on the constructed CFDM helps 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 rest of this paper is divided as follows. Section 2 handles the methodology of the CFDMPP by providing definitions in Section 2.1, declaring the problem statement in Section 3 and formulating the algorithm in Section 3.2. Details pertaining to simulations are seen in Section 4, where the setup is described in Section 4.1 and the results are analyzed in Section 4.2. Finally, Section 5 provides concluding remarks about the CFDMPP with notes on its merits and challenges.

2 Methodology

2.1 Notations and definitions

This section introduces concepts used in topology and elementary differential geometry that will be used to detail the proposed solution CFDMPP, in Section 3.2. CFDMPP uses the manifold approach to solve the Path Planning Problem (PPP). In general, the PPS can take several forms, such as: configuration space $\mathcal{C}$, workspace $\mathcal{W}$, joint velocity space, 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

$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \cdots & p_l \end{bmatrix}^T \qquad p_i \in [p_i^{min}, p_i^{max}] = p_i^{span} \tag{1}$$

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

$$\mathcal{P} = \mathbf{p}^{all} = \prod_{i=1}^{l} p_i^{span} \tag{2}$$

Notation (2) is used to express any other region associated with path planning. The relationship between tuples in space is observed by applying rules of compatibility, such as those in a topological space. Topology aids in defining a topological manifold, which will be used later to define the PPS.

Definition 1 (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 $\mathcal{U}_1, .., \mathcal{U}_n \in \mathcal{T}$, 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 [158].*

Definition 1 imposes general compatibility rules such that any union and intersection of subsets still lies in the original space. Specifically, consider the standard topology, where, a set $\mathcal{U}$ is open in $\mathbb{R}^n$ iff $\forall p \in \mathcal{U}$, there is an open ball $B(p, \epsilon)$, with centre p and radius ϵ, that is contained in $\mathcal{U}$. Such point-set topology is concerned with properties that are invariant under homeomorphisms This invariance in the PPS is what helps in planning constraint-free paths in a space whose constraint-free nature remains invariant, when transformed to another,similarly equipped topological space. Define PPS $\mathcal{P}$ as a topological space, which as a union of constrained and constraint-free subsets: $\mathcal{P} = \mathcal{P}^{\text{free}} \cup \mathcal{P}^{\text{constraint}}$.

Definition 2 (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 the 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:* $\mathcal{P}^{constraint} = \mathcal{P}^{mechanical\ limits} \cup \mathcal{P}^{singularities} \cup \mathcal{P}^{no\text{-}go\ zones} \cup \mathcal{P}^{obstacles}$.

Then $\mathcal{P}^{\text{free}} = \mathcal{P} \setminus \mathcal{P}^{\text{constraint}}$, as illustrated in Fig. 1. Topology defines inter-subset (implicit) relationships, which in turn may be developed to create explicit (global) representations of the space. Consider a sphere, which is a Topological Manifold (TM), whose explicit (viewed from ambient space) expression is the equation for a sphere. Whereas, implicitly, it can be viewed as the stitching of the Northern and Southern hemispheres/subsets using topology. While topological spaces can be globally Euclidean in nature, like $\mathbb{R}^n$,

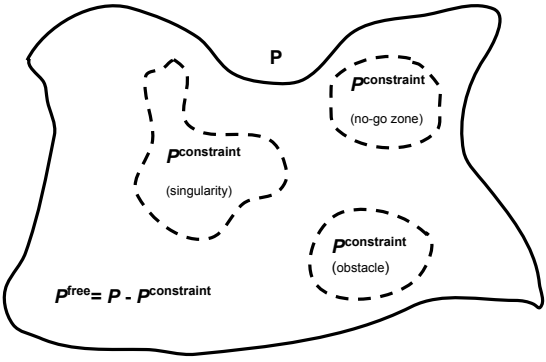


Fig. 1: Breakdown of the general Path Planning Space (PPS) $\mathcal{P}$ into $\mathcal{P}^{\text{constraint}}$ and $\mathcal{P}^{\text{free}}$

most topological spaces are not so, as seen with the sphere. Therefore, what

allows globally Euclidean and non-Euclidean topological spaces to be locally and globally defined, is their locally Euclidean nature. TMs describe such topological spaces either globally or locally. Since the PPS will eventually be defined as a Path Planning Manifold (PPM), a TM is defined.

Definition 3 (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 ϕ from $\mathcal{U}$ to an open subset of $\mathbb{R}^m$. The pair $(\mathcal{U}, \phi : \mathcal{U} \to \mathbb{R}^m)$ 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 $\mathcal{U}$.[158]*

A homeomorphic map f is bijective, continuous and has a continuous inverse [158]. n-manifold implies that the manifold is locally homeomorphic to $\mathbb{R}^n$; 1- and 2-manifolds are seen in Figs. 2 and 3. Definition 3 shows that the TM is indeed locally mapped to and from $\mathbb{R}^m$, by using a homeomorphism.

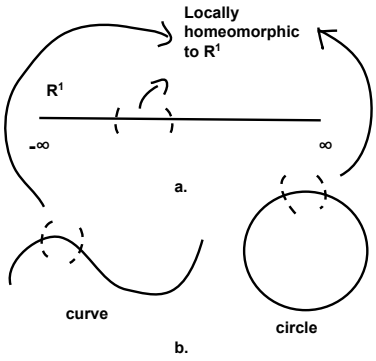


Fig. 2: Example of a 1-manifold: a. The Euclidean space $\mathbb{R}^1$ is covered by a single chart $(\mathbb{R}^1, I_{\mathbb{R}^1})$, where $I_{\mathbb{R}^1} : \mathbb{R}^1 \to \mathbb{R}^1$ is the identity map. b. A curve and a circle are examples of a 1-manifold.

(a) An ellipsoid (b) A torus

Fig. 3: Examples of 2-manifolds

Implicitly, the TM is defined by the chart, which 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 imposes that the subsets lie on the "same" TM. That is, the subsets of the TM share locally Euclidean points via an intersection. Establishing that the subsets indeed neighbour each other requires a diffeomorphism. Maps that are n-times continuously differentiable maps are denoted by $\mathcal{C}^n$. $\mathcal{C}^\infty$ maps are infinitely continuously differentiable. A diffeomorphic map f is a homeomorphism, whose inverse f^{-1} is at least $\mathcal{C}^1$. Smooth diffeomorphisms have inverses that are $\mathcal{C}^\infty$.

Proposition 1 (Composition of diffeomorphisms)**.** *Every composition of diffeomorphisms is a diffeomorphism. [159, Proposition 2.15(a)].*

That is, diffeomorphisms are reversible and can arrive at the first preimage of the map. This determines if neighbouring subsets lie on the same TM. Adjacency relationships shared by subsets of a TM are defined by respective charts of each subset pair, called compatible charts. Compatible charts must satisfy the underlying topology of the TM by using Definition 1.

Definition 4 (Compatible Charts)**.** *Two charts* $(\mathcal{U}, \phi : \mathcal{U} \to \mathbb{R}^n)$, $(\mathcal{V}, \psi : \mathcal{V} \to \mathbb{R}^n)$ *of a topological manifold are* C^∞*-compatible if the two maps*

$$\phi \circ \psi^{-1} : \psi(\mathcal{U} \cap \mathcal{V}) \to \phi(\mathcal{U} \cap \mathcal{V})$$
$$\psi \circ \phi^{-1} : \phi(\mathcal{U} \cap \mathcal{V}) \to \psi(\mathcal{U} \cap \mathcal{V})$$

are C^∞*. These two maps are called the transition functions between the charts [158].*

Fig. 4 shows that Definition 4 is twofold- actions of charts on set-intersections to indicate adjacency and their quantization using a transition map. Since only C^∞-compatible charts are focused on, in an abuse of notation,C^∞-compatible charts will be referred to simply as charts. These charts identify and relate subsets of the TM. The union of compatible charts as an atlas, implicitly pave the TM.

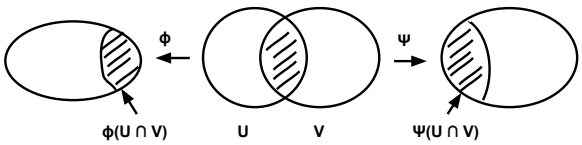


Fig. 4: Illustration of Definition 4 shows that the transition function $\psi \circ \phi^{-1}$ is defined on $\phi(\mathcal{U} \cap \mathcal{V})$.

Definition 5 (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, $\mathcal{M} = \bigcup_\alpha \mathcal{U}_\alpha$ [158].*

Definition 6 (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 $\mathcal{U} = \mathcal{M}$ [158].*

Definition 7 (Smooth Manifold). *A C^∞ manifold, or simply Smooth Manifold (SM), M is a TM together with a maximal atlas. The maximal atlas is also called a differentiable structure on M. A manifold is said to have dimension n if all of its connected components have dimension n [158].*

Thus, a TM is a SM if all transition maps in the atlas are C^∞ diffeomorphisms i.e, the atlas is a smooth atlas [160–162]. For more details, refer to [158, Proposition 5.10]. Now, we define interactions between manifolds. This aids in defining the relationship between the TMs in $\mathcal{C}$ and $\mathcal{W}$.

Definition 8 (Product Manifold). *Suppose $M_1, \ldots, M_k$ are TMs of dimensions $n_1, \ldots, n_k$, respectively. The product space $M_1 \times \cdots \times M_k$ is a TM of dimension $n_1 + \cdots + n_k$. [159]*

The n-torus is an example of Definition 8. For $n > 0$, the n-torus is $\mathbb{T}^n = \mathbb{S}^1 \times \cdots \times \mathbb{S}^1 = \prod_{i=1}^n \mathbb{S}^1$ [159]; see Fig. 3(b).

Definition 9 (Smooth Product Manifold). *If $M_1, \ldots, M_k$ are SMs of dimensions $n_1, \ldots, n_k$, respectively, then the product space $M_1 \times \cdots \times M_k$ is a TM of dimension $n_1 + \cdots + n_k$, with charts of the form $(\mathcal{U}_1 \times \cdots \times \mathcal{U}_k, \phi_1 \times \cdots \times \phi_k)$. Any two such charts are smoothly compatible. This defines a natural smooth manifold on the product, called the Product Smooth Manifold (PSM) structure. [159]*

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

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

is a C^∞ atlas on $M \times N$. Therefore, $M \times N$ is a C^∞ manifold of dimension $m + n$ [158], for example, the n-torus.

Definition 10 (Smooth Maps between Manifolds). *Let M and N be SMs. Then, the map $F : M \to N$ 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 $\psi(\mathcal{V})$ [159].*

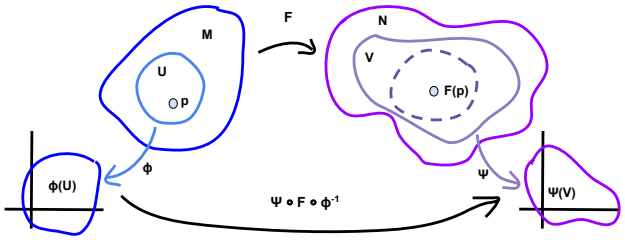


Fig. 5: Illustration of a smooth map between manifolds as seen in Definition 10

Having defined a smooth map between two manifolds using charts (see Fig. 5), 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 $F : M \to N$ that has a smooth inverse. Then, M and N are diffeomorphic, symbolized by $M \approx N$. Theorem 1 helps determine when a local, chart-wise diffeomorphism exists between two manifolds.

Theorem 1 (Inverse Function Theorem for Manifolds (IFTM)). *Let $F : M \to N$ be a C^∞ map between two manifolds of the same dimension, and $\mathbf{p} \in M$. Suppose for some charts $(\mathcal{U}, \phi) = (\mathcal{U}, x^1, \ldots, x^n)$ about $\mathbf{p} \in M$ and $(\mathcal{V}, \psi) = (\mathcal{V}, y^1, \ldots, y^n)$ 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 $\det\big(\delta F^i / \delta x^j(\mathbf{p})\big)$ is nonzero [158].*

Theorem 2 (Invariance of Dimension). *A nonempty SM of dimension m cannot be diffeomorphic to an n-dimensional SM unless $m = n$ [159].*

This dimensional invariance imposed using diffeomorphisms allows for mapping to and from constraint-free TMs. TMs can be represented implicitly and explicitly. Global representations are possible only if the dimension of the ambient space hosting the TM is greater than the dimension of the TM. So, an n-manifold lives in an $(n + 1)$ or higher-dimensional space. Hence, only manifolds of dimensions $0, 1$ and 2 can be visualized, and thus perceived for the PPP. Here, the PPS will be defined as a 2-manifold whenever possible and the path generated will be defined as a 1-manifold. Thus, the explicit representation will be defined for a 2-manifold, from the perspective of a surface (Definition 11 and Figs. 3 and 6), followed by the parametrization of a 1-manifold (Definition 12).

Definition 11 (Surface). *A subset $\mathcal{S}_{surface}$ of $\mathbb{R}^3$ is a surface if, for every point $\boldsymbol{p} \in \mathcal{S}_{surface}$, there is an open set $\mathcal{U}$ in $\mathbb{R}^2$ and an open set $\mathcal{W}$ in $\mathbb{R}^3$ containing $\boldsymbol{p}$, 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 $\rho : \mathcal{U} \to \mathcal{S}_{surface} \cap \mathcal{W}_{subset}$ 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* $\mathcal{S}_{surface}$ *[163].*

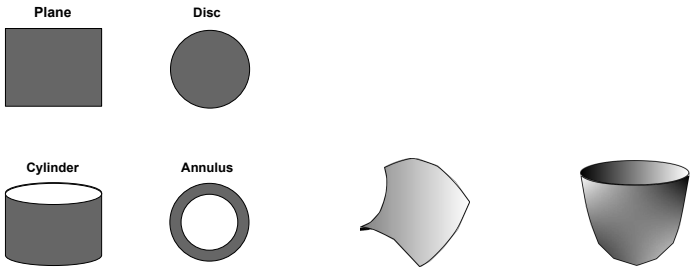


Fig. 6: Examples of surfaces. (a): common surfaces; (b): hyperbolic paraboloid; (c) elliptic paraboloid.

Definition 12 (Parametrized curve)**.** *A parametrized curve in* $\mathbb{R}^n$ *is a map* $\gamma : (\alpha, \beta) \to \mathbb{R}^n$*, for some* α, β*, such that* $-\infty \leq \alpha \leq \beta \leq \infty$ *[163].*

The open interval $(\alpha, \beta) = \{t \in \mathbb{R} \mid \alpha < t < \beta\}$. 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. To understand a geodesics, tangent spaces are introduced.

Definition 13 (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* $\mathcal{S}_{surface}$ *passing through* $\mathbf{p}$*. The tangent space* $\boldsymbol{T}_p\mathcal{S}$ *of* $\mathcal{S}_{surface}$ *at* $\mathbf{p}$ *is the set of all tangent vectors to* $\mathcal{S}_{surface}$ *at* $\mathbf{p}$ *[163].*

Definition 14 (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 [163].*

The proposed CFDMPP is based on generating geodesics as constraint-free paths on the PPS, while considering its underlying curvature. Some examples of geodesics are shown in Fig. 7. However, modifications to these paths may be necessary to overcome Dynamic Constraints (DCs). To this end, concepts known as homotopy and homology, are applied. Before providing definitions, an intuitive relationship between these concepts and the PPP is provided. Topological spaces lay the foundation for defining metrics, which formalize notions of continuity and differentiability. A surface, TM or $\mathcal{TS}$ helps define

the PPS. A path between $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ in the PPS is the image of a map, whose pre-image lies in some space, for example, the control input space. In a 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 evaluates path quality based on path length. Then, geodesics allow us to generate the shortest path on a surface, as defined by the metric on that surface. So, minimizing path length can be held as a reliable measure across the PPSs. If the PPS is no longer

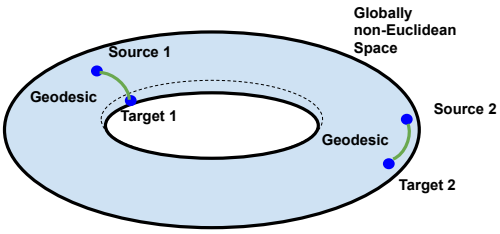


Fig. 7: A globally Non-Euclidean space such as a Genus-1 Torus (donut) where the shortest path between any two points is a geodesic.

pristine, that is, $\mathcal{P} \neq \mathcal{P}^{\text{free}}$, then a mathematical element has upset the PPS. An exclusion of a region (point) from the PPS has occurred, invalidating its original definition. This exclusion of free space now reflects a missing region, which eventually forms the basis of homotopy and homology. This causes a change in the topological property of connectedness of the space. Roughly speaking, a space is simply connected if every closed curve in the space can be shrunk to a point [164]. So in the PPP, the original shortest path may no longer be mathematically correct, if it appears to cut across the missing region. Then, a new shortest path must be planned. But, there may now exist alternative paths, each satisfying the path length measure.

This occurs as the path diverts around the missing region, while remaining on the PPS and reaching $\mathbf{p}^{\text{goal}}$. Recognizing the number of alternative paths is non-trivial for more than one constraint or missing region. Fig. 8 illustrates trivial and non-trivial cases. The differently coloured lines represent unique path types between their respective $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ pairs. The number of possible paths in a space with several missing regions, depends on two attributes: (i) the locations of $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, and (ii) the number of missing regions, as seen in Fig. 8. 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 path for its optimality, a list of possible paths needs to be created. The list requires distinguishing paths that are different from one another (diverting differently) and grouping together longer/shorter variants of a path type. This requirement of grouping and separating paths, is what leads to homotopy and homology.

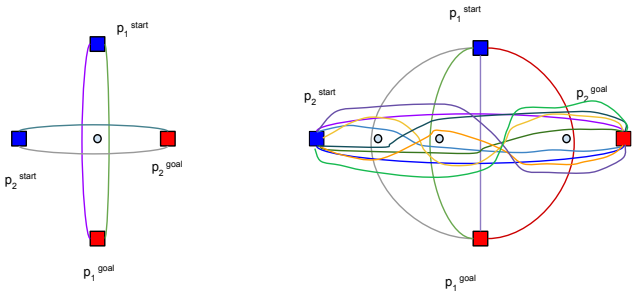


Fig. 8: Paths for different $\mathbf{p}^{\text{start}}$-$\mathbf{p}^{\text{goal}}$ pairs. Trivial case with one and non-trivial case with three missing regions on the left and right sides respectively.

2.2 Homotopy and Homology

Homotopy and homology give us a tool to uniquely characterize a path, by determining how many missing regions the path encloses. Consequentially, paths of different lengths that still enclose the same missing regions can be characterized as the same path-type. Inversely, paths of the same path lengths that enclose different missing regions will be classified as being different. 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 PPP, when the 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. This trait marks the direction of path as 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. Such 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 the best path across path types, and within the chosen path type.

2.2.1 Homotopy

Definition 15 (Homotopy)**.** *If f and f' are continuous maps: $X \to Y$, f is homotopic to f' if there is a continuous map $F : X \times I \to Y$, such that $F(x, 0) = f(x)$ 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 is denoted by $f \simeq f'$ [164].*

A path from x_0 to x_1 in space X is a continuous map $f : I \to X$, such that $f(0) = x_0$ and $f(1) = x_1$, where $I = [0, 1]$ is the unit interval [165].

Definition 16 (Path Homotopy)**.** *Two paths f and f' from $I \to X$, where $I = [0, 1]$, are said to be path homotopic, if they have the same initial and final points x_0 and x_1, respectively, and there exists a continuous map $F : I \times I \to X$, such that $\forall(s, t) \in I^2$,*

$$F(s,0) = f(s) \qquad F(s,1) = f'(s)$$
$$F(0,t) = x_0 \qquad F(1,t) = x_1$$

Then, F is called the path homotopy between f and f' and the homotopic paths are expressed as $f \simeq_p f'$. [164].

Fig. 9 path homotopy; specifically, that in the event of a constraint, $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$ still remain the same, but an alteration of the path is necessary.

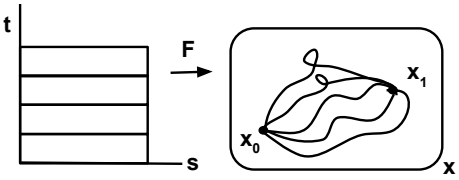


Fig. 9: Path homotopy with the unit plane on the left hand side and the resulting homotopic paths on the right hand side. The divisions of the unit plane are mapped to X as homotopic paths, with the same initial and final points.

Lemma 1. *The relations $\simeq$ (homotopy) and $\simeq_p$ (path homotopy) are equivalence relations. [164].*

Lemma 1 aids in categorizing different paths as equivalent, if they are homotopic. The proof of Lemma 1 is seen formally in [164], and informally here, for PPP contexts. If f is a path, denote its path-homotopy class by $[f]$. First, $f \simeq_p f$ is essential and trivial. Second, $f \simeq_p f'$ implies $f' \simeq_p f$, shows that path homotopy is sequence independent. Third, $f \simeq_p f'$ and $f' \simeq_p f''$ implying $f \simeq_p f''$, shows that path homotopy is path-quantity independent. Fig. 10 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. Fig. 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 t are (straight line) path homotopic, but h is not homotopic to t. It is intuitively clear that there exists no homotopy in that case, as one cannot deform h past the missing region at the origin. Thus, path homotopy and different homotopic classes relate to paths around missing regions in the PPP. The subsequent definitions are of use for a discretized PPS.

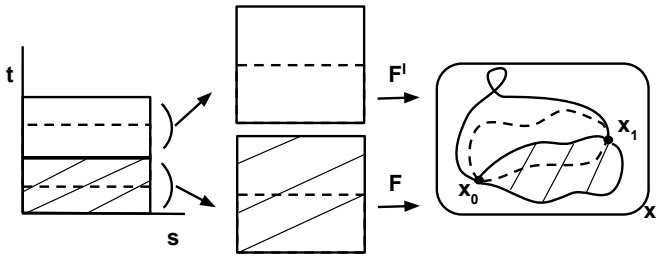


Fig. 10: If $F : f \simeq_p f'$ and $F' : f' \simeq_p f''$, then $G : f \simeq_p f''$.

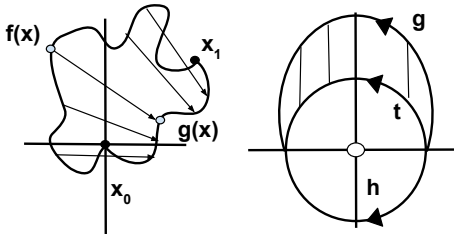


Fig. 11: Straight line homotopy. Left hand side: path homotopy between f and g. Right hand side: path homotopy between g and t, but not between t and h.

Definition 17 (Product of paths)**.** *If f is a path in X from x_0 to x_1, and g is a path in X from x_1 to x_2, then the product of f and g, denoted as $h = f * g$, is defined by [164]*

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

When a path is planned, it is typically generated incrementally, i.e, as a composition of path segments. Even as segments, they are used in the induced product operation on path-homotopy classes as

$$[f] * [g] = [f * g] \tag{3}$$

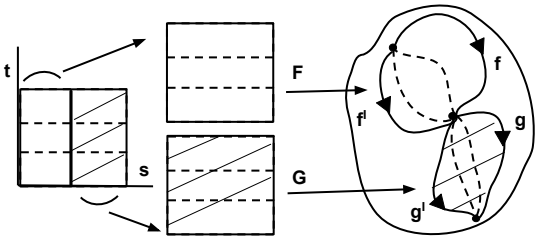


Fig. 12: The product operation of two path-homotopy classes.

Fig. 12 shows the significance of the product operation on path-homotopy classes. It illustrates that the product of two path-homotopy classes is the path-homotopy class of the product of paths. The associativity of the product operation on path-homotopy classes holds for any finite product of classes. That is, the path-homotopy class of the final path remains the same irrespective of the number of intermediate path segments to which the product is applied. Stated formally as a theorem [164]:

Theorem 3. *Let f be a path in X and $a_0, ..., a_n$ be real scalars, such that $0 = a_0 < a_1 < ..., a_n = 1$. Let $f_i : I \to X$ be the path that equals the positive linear map of $I = [0, 1]$ onto $[a_{i-1}, ai]$ followed by f. Then, $[f] = [f_1] * \cdots * [f_n]$.*

Theorem 3 is of prime significance for the discretized PPP. This work proposes discretizing the PPS and then commencing path planning. That is, path segments are generated incrementally and their composition 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, one is confident that theoretically, discretized path segments and the map resolution have 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 homology.

2.2.2 Homology

Homology theory is quite detailed, and beyond the scope of this work. 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 [165]. 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 the PPS X. These homology groups are topological invariants and are also homotopy invariants [165]. Intuitively speaking, homology allows

the detection of holes/missing regions 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 them to determine homotopic classes. As the authors of [136] (on whose work the homotopy implementation of this article 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 $f : I \to X$ with x_0 and x_1 as the initial and final points, one defines the reverse path from x_1 to x_0 as $\bar{f}(s) = f(1-s)$, where $s \in I$ [165]. Fig. 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.

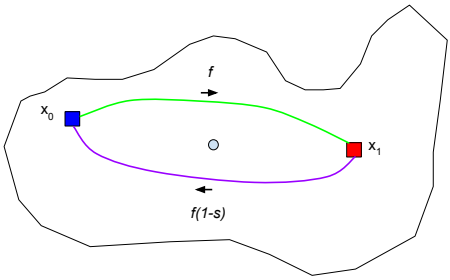


Fig. 13: Connection between homology and homotopy

One way to characterize a contour around a missing region is by looking at the number of its windings around that missing region. Consider a contour $L(z) \in \mathbb{C}$, to define the winding number. Assume that $L(z)$ encloses a point $p \in \mathbb{C}$, which can be considered as the missing region. Then, the winding number $v(L, p)$ of a closed loop or contour L 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 [166]. A simple loop does not self intersect. For loops that are not simple, the winding number is determined by the number of partitions into which the sub-loops of the loop break the space as seen in Fig. 14. Partitions that have a non-zero winding number v_i are defined as the inside of the loop,

and those with a zero-winding number, are defined as being outside of the loop. This is helpful to understand because of our interest in the presence of constraints/missing regions, which will later be defined as poles of functions in $\mathbb{C}$. The computation of winding numbers depend on where the poles are located with respect to the contour- inside or outside. The demonstration of this requires the introduction of a few definitions.

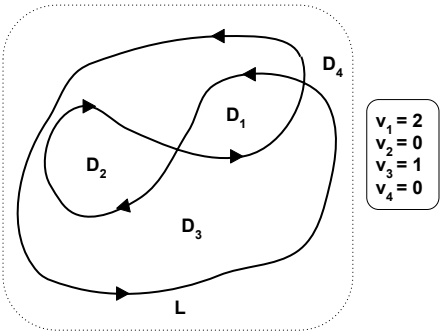


Fig. 14: Non-simple loop: winding number of its partitions

The following definitions deal with functions on the complex plane denoted by $\mathbb{C}$.

Definition 18 (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 its derivative at z_0 is continuous and finite, i.e., $f'(z_0) = \lim_{h\to 0}[f(z_0 + h) - f(z_0)]/h$ is unique and finite, for $h \in \mathbb{C}$.*

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 for the computation of integrals involving contours that enclose poles.

Theorem 4 (Cauchy's theorem for a disc [167])**.** *If f is a holomorphic function in an open disc, then*

$$\int_{\gamma} f(z) = 0 \tag{4}$$

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

Cauchy's theorem states that if a holomorphic mapping has no singularities inside a loop, its integral around the loop vanishes [166]. To illustrate this, consider Fig. 15, which has a loop K and a singularity. Using partitions as defined above, one can see that loop K does not encircle the singularity, and thus its integral around the loop must vanish.

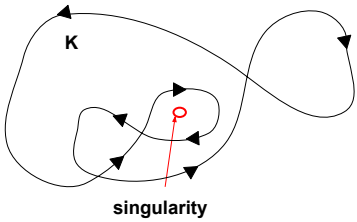


Fig. 15: Singularity located "outside" the loop; so integral vanishes

The disc in Theorem 4 can be 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 (4) holds true. Theorem 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 agree, provided that the mapping is analytic (holomorphic in $\mathbb{C}$) everywhere in the region lying between the two contours [166]. 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, then 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 5 (Cauchy's Integral Formula [167])**.** *Let f be holomorphic in an open set that contains the closure of a disc D. If C denotes the boundary circle of this disc with positive orientation, then*

$$f(z_0) = \frac{1}{2\pi i} \int_C \frac{f(z)}{z - z_0}\, dz \tag{5}$$

for any point $z_0 \in D$.

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 [166]. With theorem 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 $e^{i\theta}$. Since $f(z)$ has a pole at $z_0 = 0$, according to (5)

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

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, whose result is the integral of any curve K in $\mathbb{C}$, as as f acts on it; see Fig. 16. Since the result of the integral is the connecting arrow, it is irrespective of the type or length of contour. The contour length increases the detours taken, while undergoing the same final net revolutions around the pole.

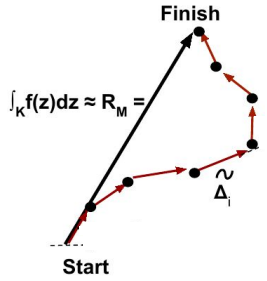


Fig. 16: Riemann Sum and winding number

Theorem 6. *If a contour sweeps only through analytic points as it is deformed, the value of the integral does not change [166].*

Hence, (5) 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, if and only if the two loops have the same winding number around p. Imaginably, this characteristic is what we aim to recreate for homology classes.

We now give some insight on how to numerically compute the integral in (5).

Definition 19 (Residue)**.** *If a holomorphic function f has a pole of order n at z_0, then the residue of f at z_0 is defined as [167]*

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

Theorem 7 (Residue Formula)**.** *Suppose that f is holomorphic in an open set containing a closed contour γ and its interior, except for poles at $z_1, ..z_n$ inside C. Then,*

$$\int_\gamma f(z)dz = 2\pi i \sum_{k=1}^{n} Res[f(z), z_k] \tag{6}$$

for any point $z \in D$ [167].

Recall that Theorems 4, 5 and 7 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 in Theorem 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 a contour K, which is acted on by a holomorphic function $f(z) = 1/(z - z_0)$. That is, the contour encloses the pole z_0 of the holomorphic function, as seen in Fig. 17. Applying the Residue Formula to this contour via f gives

$$\oint_K \frac{1}{(z - z_0)} = \oint_K f(z) = 2\pi i \,\mathrm{Res}[f(z), z_0]. \tag{7}$$

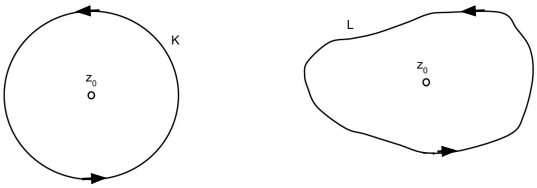


Fig. 17: Contours surrounding one pole

The value of the integral in (8) 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 Fig. 17. That is:

$$\oint_K \frac{1}{(z - z_0)} = 2\pi i \, v(K, z_0) \,\mathrm{Res}[f(z), z_0] \tag{8}$$

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

From (8), 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. The term residue is owed to the fact that it is the only part of the function that remains after the integration via the Cauchy Integral Formula [166]. 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 that 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 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 Fig. 17, as 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 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. To generalize to several poles, the *General Residue Theorem* states:

$$\oint_K f(z)dz = 2\pi i \sum_j v(K, s_j)\,\mathrm{Res}[f(z), s_j] \tag{9}$$

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 in Theorem 7 represents a special case of the General Residue Theorem represented by (9), when every pole is wound around once. In the context of this article, 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. Fig. 18 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 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

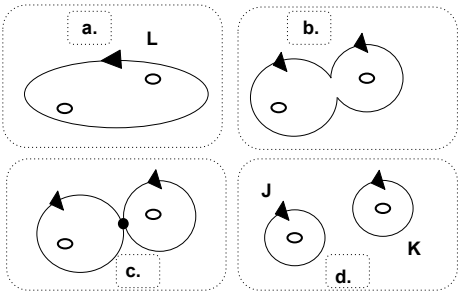


Fig. 18: Decomposition of a contour enclosing two singularities

homology class markers, we shall borrow the definition of homologous paths from [119] to indicate a relationship between contours in a homotopic class and those in the homology class (see Fig. 19) 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 article.

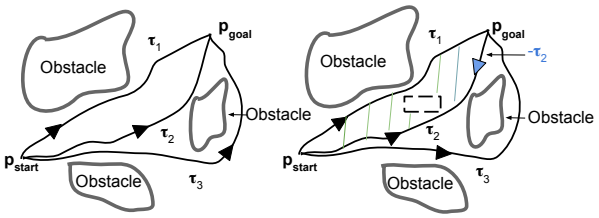


Fig. 19: Left: Homotopic paths τ_1 and τ_2, but not τ_3, with either of the other paths. Right: Homologous paths τ_1 and $-\tau_2$ enclose shaded area while τ_3 is not homologous with τ_1 or τ_2.

Definition 20 (Homologous Paths)**.** *Two paths τ_1 and τ_2 connecting the same start and end coordinates, $\boldsymbol{p}^{start}$ and $\boldsymbol{p}^{goal}$, 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, the aforementioned concepts are applied to propose the CFDMPP as a novel algorithm to solve the PPP in hand.

3 Methodology

3.1 Problem Statement

Consider a robotic system with at least one PPS, $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$, an initial knowledge of constraints, and a sensing mechanism to relay information about new constraints as it becomes available. Then, the goal is to develop a path planning algorithm which generates at least one constraint-free path from $\mathbf{p}^{\text{start}}$ to $\mathbf{p}^{\text{goal}}$ in a reactive manner without having to explicitly check for constraint violation on each generated path. To this end, we propose the Constraint-Free Discretized Manifolds-based Path Planner (CFDMPP), which is a path planner with the following characteristics:

- It formulates a CFM on the PPS.
- It produces a path in the shortest path class and at least one other alternative path class on the CFDM.
- It generates a one-time cost map which maybe reused based on some algorithm tuning parameters.

3.2 Algorithm Formulation

A high level view of the CFDMPP is revealed in Algorithm 1. The algorithm's four main steps are as follows:

- Line 1 constructs a CFM on the map of the PPS, by accounting for locations and bounds of the Static Constraints (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, which guarantees that the paths generated on it are constraint-free without having to explicitly validate the path against any constriant violations.
- Line 2 discretizes the CFM from Line 1 to produce a graph-based representation of the manifold. A graph traversal algorithm can then be applied to compute the shortest constraint-free path from $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$.
- Line 3 allows the generated path from Line 2 to be transformed to an alternative PPS, if necessary.
- 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 (i.e., falling object in a warehouse) or human beings. The second situation involves the requirement of an alternative path, based on a human decision after assessing the 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 1 CFDMPP

Require:
Initial (approximate) map of the environment
Bounds of the static constraints
Source and target locations in PPS: $\mathbf{p}^{\text{start}}$ and $\mathbf{p}^{\text{goal}}$

Ensure:
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 the DC responsible for path blockage ▷ Section 3.2.2
7: **end while**
8: **return** followed path

The rest of the section is broken down into four parts. Each part reflects the same sequence of steps seen in Algorithm 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 covered Section 3.2.2.

3.2.1 CFM Formulation

The following steps systematically formulate the CFM.

i. 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 |PPSs| and |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 SPPS_1, SPPS_2, ..., $\text{SPPS}_{|\text{SPPSs}|}$.

ii. Dimensions of Path Planning Spaces (PPSs): As noted in Definition 1, 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.

iii. 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 (2), 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 (2), may hold insufficient information about the the bounds of the PPS. 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 step (iv).

iv. Mapping Between Path Planning Spaces (PPSs): When a PPP has multiple equivalent PPSs, it implies that the PPPS 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 PPPS to those of a SPPS. We will denote the continuous mapping (if it exists) from PPS_i to PPS_j, $i, j \in \{1, 2, \ldots, |\text{PPSs}|\}$, as $f_{\text{PPS}_i}^{\text{PPS}_j} : \text{PPS}_i \to \text{PPS}_j$. Obviously, the existence of such a mapping does not guarantee the existence of its inverse, $\left(f_{\text{PPS}_i}^{\text{PPS}_j}\right)^{-1}$. 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 an Euclidean PPS (in the form of (2)) may not be Euclidean.

v. Information About Static Constraints (SCs): The environment may contain constraints, as defined in Definition 2, which have to be avoided by the path planner. Thus, their information must be accounted for in the algorithm. This step deals with SCs, i.e., constraints whose presence is known apriori. Five parameters are used to process SCs and they are: classification of SCs, their total number, 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.

(a) Classification of SCs present in the PPP: The SCs are classified into the categories specified in Definition 2, i.e., the robot's mechanical limits (ML), singularities (SG), no-go zones (NG) and obstacles (OB). All possible static constraints are accounted for, irrespective of which of the multiple PPSs they may lie in. The classification of constraints enables a better understanding of the kind of inoperable regions that the PPP is dealing with.

(b) 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}.

(c) 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 PPPS or any of the SPPSs), which is known as the native space of that constraint (see Fig. 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 Fig. 21, and as shall be formally defined in step (iv). 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 $n_{\mathrm{SC}}^{\mathrm{PPS}_i}$ be the number of SCs in PPS_i, where $i \in \{1, 2, |\mathrm{PPSs}|\}$. Likewise, let $n_{\mathrm{ML}}^{\mathrm{PPS}_i}$, $n_{\mathrm{SG}}^{\mathrm{PPS}_i}$, $n_{\mathrm{NG}}^{\mathrm{PPS}_i}$, $n_{\mathrm{OB}}^{\mathrm{PPS}_i}$, be the (non-negative) number of SCs of their respective category in PPS_i. Therefore,

$$n_{\mathrm{SC}} = \sum_{i=1}^{|\mathrm{PPSs}|} n_{\mathrm{SC}}^{\mathrm{PPS}_i} = n_{\mathrm{SC}}^{\mathrm{PPPS}} + \sum_{i=1}^{|\mathrm{SPPSs}|} n_{\mathrm{SC}}^{\mathrm{SPPS}_i} \tag{10}$$

$$n_{\mathrm{SC}}^{\mathrm{PPS}_i} = n_{\mathrm{ML}}^{\mathrm{PPS}_i} + n_{\mathrm{SG}}^{\mathrm{PPS}_i} + n_{\mathrm{NG}}^{\mathrm{PPS}_i} + n_{\mathrm{OB}}^{\mathrm{PPS}_i} \tag{11}$$

(d) Bounds of SCs: The expanse of every constraint is determined in the native PPS. The SCs may be defined by the bounds of the PPSs using (2), as an equation or simply as a point. This step is essential for two reasons. First, if there exist multiple PPSs, then 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.

(e) Constructing the disjoint constraint sets: Once the bounds for all n_{SC} SCs have been established, one 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

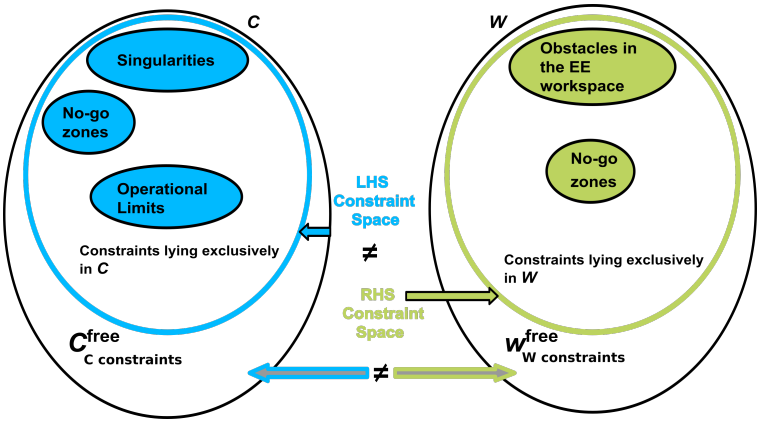


Fig. 20: Each uniquely defined static constraint may lie in any of the PPSs identified for a robotic system; therefore, that PPS becomes the native space of the constraint, and the constraint has no meaning in another PPS.

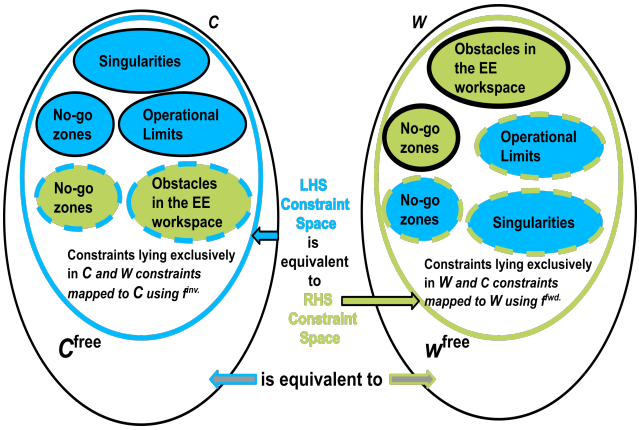


Fig. 21: In order for the original constraint to have an equivalence in another PPS, a mapping is required. Only after mapping the original constraints in their native space to the alternative PPS, their equivalence of constraints in both spaces are established.

the DCSs is detailed. A CS associated to a specific PPS_i, denoted by ${\mathrm{CS}_r}^{\mathrm{PPS}_i}$, 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 $f(\mathrm{SC}_i, \mathrm{SC}_j)$ to measure the proximity $\Delta_{\mathrm{CS}}^{ij}$ between two SCs, SC_i and SC_j, in the same constraint

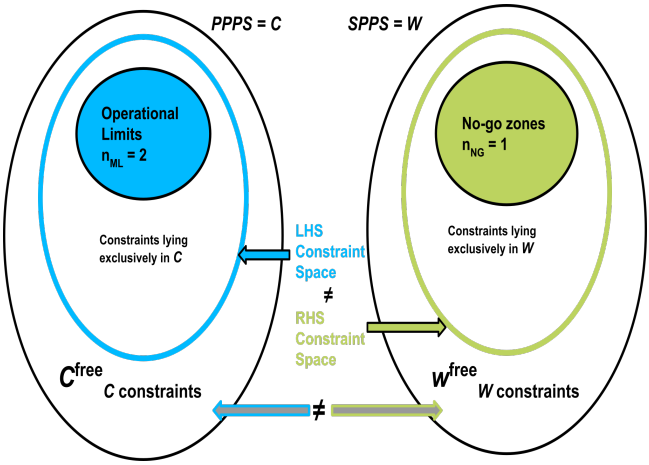


Fig. 22: The 2LPM example as seen in Step (v)(v)c. The native PPS for static constraints present in the PPP is seen here. Note that the constraints in their originally defined form do not have an equivalence in the alternative PPS.

set.

$$\Delta_{\mathrm{CS}}^{ij} = f(\mathrm{SC}_j, \mathrm{SC}_j) \tag{12}$$

where $i, j \in \{1, \ldots, n_{\mathrm{SC}}^{\mathrm{PPS}_i}\}$. There are several ways to define such a function depending on the user preference. Here, 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 constraint set ${\mathrm{CS}_r}^{\mathrm{PPS}_i}$ can be formally defined as

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

Let l_{uk}^{min} and l_{uk}^{max} be the respective lower and upper limits of the kth dimension of $\mathrm{SC}_u \in {\mathrm{CS}_r}^{\mathrm{PPS}_i}$ for an n-dimensional PPS_i. If ${\mathrm{CS}_r}^{\mathrm{PPS}_i} = \{\mathrm{SC}_1, \mathrm{SC}_2, \ldots, \mathrm{SC}_m\}$, then ${\mathrm{CS}_r}^{\mathrm{PPS}_i}$ is an n-dimensional Cartesian space in PPS_i; its lower and upper limits of the kth dimension are $\min\left(l_{1k}^{min}, l_{2k}^{min}, \ldots, l_{mk}^{min}\right)$ and $\max\left(l_{1k}^{max}, l_{2k}^{max}, \ldots, l_{mk}^{max}\right)$, respectively. Such limits, for $k = 1, \ldots, n$, define the boundary $\delta{\mathrm{CS}_r}^{\mathrm{PPS}_i}$ of ${\mathrm{CS}_r}^{\mathrm{PPS}_i}$. The set ${\mathrm{CS}_r}^{\mathrm{PPS}_i}$ may contain SCs of different categories as per Definition 2. 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 $\mathcal{CO}$-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,

$$\text{DCSs} = \bigcup_{i=1}^{|\text{PPSs}|} \text{DCSs}^{\text{PPS}_i} \tag{13}$$

$$|\text{DCSs}| = \left|\text{DCSs}^{\text{PPPS}}\right| + \sum_{i=1}^{|\text{SPPSs}|} \left|\text{DCSs}^{\text{PPS}_i}\right| \tag{14}$$

Algorithm 2 outlines the iterative procedure to compute $\text{DCSs}^{\text{PPS}_i}$, where the notation $\alpha(j)$ denotes the jth element of the set α.

vi. Potential Dynamic Constraints: 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 mechanism. The sensing algorithm is not part of the CFDMPP, so some basic assumptions about it is mentioned here. 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 as a reaction to a

Algorithm 2 Computing $\text{DCSs}^{\text{PPS}_i}$

Require:
$\alpha = \{\text{SC}_1, \text{SC}_2, \ldots\}$ ▷ Set of all SCs in PPS_i
Δ_{CS} ▷ User-defined proximity threshold for PPS_i
Ensure:
$\text{DCSs}^{\text{PPS}_i}$ ▷ Set of all DCSs in PPS_i

$\text{DCSs}^{\text{PPS}_i} \leftarrow \emptyset$
$k \leftarrow 0$ ▷ Number of DCSs
while $\alpha \neq \emptyset$ **do**
  $k \leftarrow k + 1$
  $\text{DCS}_k \leftarrow \alpha(1)$ ▷ DCS gets first SC available
  $\alpha \leftarrow \alpha \setminus \alpha(1)$
  $\text{SC}_{remove} \leftarrow \emptyset$
  for $j = 1$ to $|\alpha|$ **do** ▷ Compare with remaining CSs
    Compute the proximity Δ between SC_j and DCS_k using (12)
    if $\Delta \leq \Delta_{\text{CS}}$ **then**
      $\text{DCS}_k \leftarrow \text{DCS}_k \cup \alpha(j)$
      $\text{SC}_{remove} \leftarrow \text{SC}_{remove} \cup \alpha(j)$
    end if
  end for
  $\alpha \leftarrow \alpha \setminus \text{SC}_{remove}$
  $\text{DCSs}^{\text{PPS}_i} \leftarrow \text{DCSs}^{\text{PPS}_i} \cup \text{DCS}_k$
end while
return $\text{DCSs}^{\text{PPS}_i}$ ▷ Eventually, $\text{DCSs}^{\text{PPS}_i} = \{\text{DCS}_1, \text{DCS}_2, \ldots, \text{DCS}_k\}$

DC. 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.

DCs can be grouped into two kinds for the purposes of the CFDMPP. The first group 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 during the construction of the CFM. The second group of DCs involves 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 step (vii), 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.

vii. Choosing the PPS for Path Planning: Step (i) shows that a PPP can have several PPSs. Subsequently, steps (iii), (iv), (v) and (vi) obtain information about the PPSs and of constraints that lie in them. 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, because of three reasons. But before that, further notation is defined. 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 segment provides Evaluative Factors (EFs) 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 PPSs_{alt}. Now, we discuss the EFs and the reasons for their use and order of application.

(a) First EF: The first EF 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 PPS_{tasks}. Then, PPS_{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.

(b) Second EF: The second EF evaluates a PPS based on the number of DCSs it contains. This factor favors choosing the PPS containing the maximum number of DCSs. This is so because of the process used to formulate the PPM. Once a PPS is chosen as the PPS_{chosen}, it is formulated into a constraint-free PPM. The latter requires the PPS_{chosen} to be completely defined, as will be seen in step (viii). The complete definition of the PPS_{chosen} involves transforming the DCSs present in PPS_{alt} to PPS_{chosen}. The transformation is accomplished by applying the mapping between PPS_{chosen} to PPS_{alt}, to the DCSs in PPS_{chosen}. Assume that $\text{DCSs}^{\text{PPS}_{chosen}}$ and $\text{DCSs}^{\text{PPS}_{alt}}$ are the sets of DCSs in PPS_{chosen} and PPS_{alt}, respectively, and $\left|\text{DCSs}^{\text{PPS}_{chosen}}\right| > \left|\text{DCSs}^{\text{PPS}_{alt}}\right|$. Then, it is computationally less intensive to transform the DCSs in PPS_{alt} to PPS_{chosen}, rather than the other way around. It is thus, advantageous to choose the PPS that requires the least transformations of DCSs from other PPSs.

(c) Third EF: The third EF evaluates a 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.

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. To resolve that conflict, the factors are ranked in the following order of precedence (strongest to weakest):

$$\text{Second EF} > \text{Third EF} > \text{First EF} \tag{15}$$

That is, the Second EF 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 EF, will the Third EF be necessary. Similarly, unless there is a tie using the Third EF, the First EF is not necessary. Should the First EF also yield a tie, any PPS may be chosen from the last set of results.

viii. Complete Definition of Chosen Path Planning Space (PPS): Once the appropriate PPS_{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.

(a) First, all unmapped DCSs from all the alternative spaces in the set PPSs_{alt} are mapped to PPS_{chosen}. This can be done systematically using Algorithm 3.

Depending on the application, Algorithm 3 may result in native or/and mapped DCSs. It was noted earlier, that additional structure would be imposed on PPS_{chosen} to describe it as a manifold. The same topological representation can be applied to also represent any DCS mapped to the PPS_{chosen}. 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 is detailed later in step (ix).

(b) Second, re-evaluate the mapped DCSs in PPS_{chosen} to see if they can be merged with the old/new DCSs. In the case of any mapped DCSs, Algorithm 2 is applied on all the native and mapped DCSs to ensure that all CSs are actually disjoint. This gives the user yet another

chance to define a new value for the previously defined proximity measure Δ_{CS}.

Algorithm 3 Mapping all unmapped DCSs to PPS_{chosen}

Require:
PPS_{chosen}
$PPSs_{alt} = \{PPS_1, \ldots, PPS_{|PPSs|-1}\}$
$DCSs^{PPS_i}$ and $f^{PPS_{chosen}}_{PPS_i}$, $\forall\ PPS_i \in PPSs_{alt}$ ▷ DCSs and mapping information

Ensure:
PPS_{chosen} containing the native DCSs and those mapped from $PPSs_{alt}$

$\alpha \leftarrow PPSs_{alt}$ ▷ All alternative PPSs whose DCSs need processing
$DCS_{mapped} \leftarrow \emptyset$
while $\alpha \neq \emptyset$ **do**
  $PPS_{curr} \leftarrow \alpha(1)$▷ Alternative PPS whose DCSs are currently processed
  for $i = 1$ to $\left|DCSs^{PPS_{curr}}\right|$ **do** ▷ Transform every DCS in the alternative PPS_{curr}
    $DCS_a \leftarrow DCSs^{PPS_{curr}}(i)$
    $DCS_m \leftarrow f^{PPS_{chosen}}_{PPS_{curr}}(DCS_a)$
    $DCS_{mapped} \leftarrow DCS_{mapped} \cup DCS_m$ ▷ Store transformed DCS
  end for
  $\alpha \leftarrow \alpha \setminus PPS_{curr}$ ▷ Remove the PPS whose DCSs have been processed
end while
return PPS_{chosen}

ix. Constructing a Constraint-Free Manifold (CFM) in the Chosen Path Planning Space (PPS)
This section focuses on defining 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/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. The three main steps required to construct a CFM are as follows. 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 2.1.

First, the topological representation of the PPS is obtained using Definition 1. While one can start with any PPS, it is recommended to start with PPS_{chosen} for the subsequent steps. The following explanation uses the general notation of PPS, not to compromise generality. Basic information about the PPS is given in (1) and (2). The topology assigned to the PPS, as defined by (2), is that of the metric topology. Generally speaking, for a metric space (M, d), d is the metric defined on M, such that $d : M \times M \to \mathbb{R}$. 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 [165]. 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)[165]. 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[165]. 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., $\mathcal{S}_{\text{PPS}} = \prod_{i=1}^{l} p_i^{span}$. Each of the l dimensions is treated as a separate topological space. The set of points in i^{th} dimension is denoted by $\mathcal{S}_i = p_i^{span}$ and is assigned the metric topology $\mathcal{T}_i$, induced by $d = \|.\|_\infty$ 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, $\mathcal{S}_{\text{PPS}} = \mathcal{S}_1 \times \cdots \mathcal{S}_i \times \cdots \mathcal{S}_l = \prod_{i=1}^{l} p_i^{span}$, on which the product topology is thus induced [165]. 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

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

In either case, we choose to define additional metrics on $\mathcal{S}_{\text{PPS}}$. Metrics, such as $d = \|.\|_1$, $d = \|.\|_2$ and $d = \|.\|_\infty$ 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 $\mathcal{T}_1 \subseteq \mathcal{T}_2$[165]. Then, the topology induced by the different metrics $\|.\|_1, \|.\|_2$ and $\|.\|_\infty$ 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 (2), 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 $(\mathcal{S}_{\text{PPS}_i}, \mathcal{T}_{\text{PPS}_i}) = \mathcal{TS}_{\text{PPS}_i}$. 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 PPS_{chosen} is the first PPS that requires this step. Item (viii) showed that PPS_{chosen} now contains mapped and native (unmapped) DCSs in $\text{DCS}^{\text{PPS}_{chosen}}$, i.e., all possible regions to avoid during path planning. In the following discussion, the superscript "PPS_{chosen}" is dropped, since only the DCSs lying in PPS_{chosen} are tackled. The native DCSs are considered first, before passing on to the mapped DCSs. Every native DCS has bounds as defined in (1) and (2) with respect to PPS_{chosen}. That is, it is a subset of $\mathcal{TS}_{\text{PPS}_{chosen}}$. Removing it from $\mathcal{TS}_{\text{PPS}_{chosen}}$ 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 [165]. 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 $\mathcal{TS}_{\text{PPS}_{chosen}}$. Therefore, the constraint-free set of points is represented as a subspace of $\mathcal{TS}_{\text{PPS}_{chosen}}$, 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 and the sake of illustration, assume that the DCS's general form is $\prod_{i=1}^{l} p_i^{\text{DCS}_{span}}$ where $p_i^{\text{DCS}_{span}} \in (p_i^{\text{DCS}_{min}}, p_i^{\text{DCS}_{max}})$. Thus,

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

$$\mathcal{S}^{free}_{\text{PPS}_{chosen}} = \mathcal{S}_{\text{PPS}_{chosen}} \setminus \mathcal{S}_{\text{DCS}} = \prod_{i=1}^{l} p_i^{span} \setminus \prod_{i=1}^{l} p_i^{\text{DCS}_{span}}$$

$$\mathcal{S}^{free}_{\text{PPS}_{chosen}} = \prod_{i=1}^{l} p_i^{\text{free}_{span}} \; ; \; p_i^{free_{span}} \in (p_i^{free_{min}}, p_i^{free_{max}})$$

The subspace topology $\mathcal{T}^{free}_{\text{PPS}_{chosen}}$ is now that subset of $\mathcal{T}_{\text{PPS}_{chosen}}$, that is applicable to $\mathcal{S}^{free}_{\text{PPS}_{chosen}}$, which is a subset of $\mathcal{S}_{\text{PPS}_{chosen}}$. The constraint-free topological space of PPS_{chosen} is thus denoted as $\mathcal{TS}^{free}_{\text{PPS}_{chosen}}$. 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 PPS_{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 PPS_{chosen}. That is, any or all l-dimensions may be modified by ϵ_i^{min} and ϵ_i^{max}, for $i = 1, \ldots, l$, such that the new bounds become $(p_i^{\text{DCS}_{min}} - \epsilon_i^{min}, p_i^{\text{DCS}_{max}} + \epsilon_i^{max})$. 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. 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 $\mathcal{TS}_{\text{PPS}_{chosen}}$. Thus, it can be removed from $\mathcal{TS}_{\text{PPS}_{chosen}}$ in the same manner as the unmapped DCSs. A topological representation of the PPS is completely constraint-free, if and only if all DCSs have been removed from it. A similar process can be undertaken to construct equivalent topological spaces for the PPS_{alt}, denoted by $\mathcal{TS}^{free}_{\text{PPS}_{alt}}$. At this juncture, the topological representations of all PPSs 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 $f^{\text{PPS}_{alt}}_{\text{PPS}_{chosen}}$, relating PPS_{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 $f^{\text{PPS}_{alt}}_{\text{PPS}_{chosen}}$ 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 [165, Proposition 2.19]. Thus, all constraint-free topological spaces are topologically equivalent, by virtue of 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 PPS_{chosen}, specifically $\mathcal{TS}_{\mathrm{PPS}_{chosen}}$. Using Definition 3, a manifold can be constructed on a subset of $\mathcal{TS}_{\mathrm{PPS}_{chosen}}$, 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 11, 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 $\mathcal{TS}_{\mathrm{PPS}_{chosen}}$.

Choosing the manifold to construct 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. Every compact surface admits a polygonal presentation [165], 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 $\mathcal{TS}_{\mathrm{PPS}_{chosen}}$. Now, the construction of an annulus corresponding to one DCS and its surrounding area is shown. Connectedness and compactness of the annulus are also shown along the way.
The construction of an annulus on $\mathcal{TS}^{free}_{\mathrm{PPS}_{chosen}}$ 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 $\mathcal{TS}_{\mathrm{PPS}_{chosen}}$ can be mathematically deformed into an annulus. This annulus then, still adheres to the original topology that was attributed to $\mathcal{TS}_{\mathrm{PPS}_{chosen}}$. This is so, since every homeomorphism is

a local homeomorphism [165, 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 $\mathcal{TS}_{\text{PPS}_{chosen}}$, without loss of generality. $\mathcal{TS}_{\text{PPS}_{chosen}}$ is 2-dimensional whose span is $(-r_2, r_2) \times (-r_2, r_2)$, where $(r_2, r_2) \in \mathcal{S}_{\text{PPS}_{chosen}}$. The resulting region is an open square (boundary not included) centered at the origin of $\mathcal{TS}_{\text{PPS}_{chosen}}$, and whose sides are of length $\approxeq 2r_2$. This $\mathcal{TS}_{\text{PPS}_{chosen}}$ is assumed to contain one DCS, also centered at the origin, whose span is $[-r_1, r_1] \times [-r_1, r_1]$, where $[r_1, r_1] \in \mathcal{S}_{\text{PPS}_{chosen}}$ 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 $\mathcal{TS}^{free}_{\text{PPS}_{chosen}}$, when the DCS is removed from $\mathcal{TS}_{\text{PPS}_{chosen}}$, 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: $\mathcal{S}^{free}_{\text{PPS}_{chosen}} = \{\mathbf{p} \in \mathcal{S}_{\text{PPS}_{chosen}} : r_1 < \|\mathbf{p}\|_\infty < r_2\}$. 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 $\mathcal{TS}^{free}_{\text{PPS}_{chosen}}$, the reference to $\mathcal{TS}^{free}_{\text{PPS}_{chosen}}$ will no longer be made. Only the deformation of shapes describing the space will be referred to. The shape as defined so far, is referred to as representation-1 or R-1 and is given by:

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

In Fig. 23, R-1 is the area shown in gray, with dashed edges to show that boundaries are excluded. 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

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

where $\mathbf{p}^{\text{R-1}} \in$ R-1 and $\mathbf{p}^{\text{R-2}} \in$ R-2. Note that the homeomorphism is taking place in the same topological space; the notation $\mathbf{p}^{\text{R-1}}$ and $\mathbf{p}^{\text{R-2}}$ 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:

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

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 $h^{\text{R-2}}_{\text{R-1}}(\mathbf{p}^{\text{R-1}})$ does satisfy the definition of a homeomorphism. $h^{\text{R-2}}_{\text{R-1}}(\mathbf{p}^{\text{R-1}})$ is continuous, i.e., pre-images of open sets are open in the defined topology of $\mathcal{TS}^{free}_{\text{PPS}_{chosen}}$, and is also bijective. One can verify, that $h^{\text{R-2}}_{\text{R-1}}(\mathbf{p}^{\text{R-1}})$ 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, $h^{\text{R-2}}_{\text{R-1}}(r_1+\epsilon, r_1+\epsilon) = (r_1+\epsilon, r_1+\epsilon)^{(r_1+\epsilon)-r_1}/_{(r_2-r_1)(r_1+\epsilon)}$ takes the following values:

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

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 $h^{\text{R-2}}_{\text{R-1}}$ on a point $\mathbf{p}^{\text{R-1}}$ can be seen in Fig. 23, where the gray region or 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 $h^{\text{R-1}}_{\text{R-2}}(\mathbf{p}^{\text{R-2}})$, 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

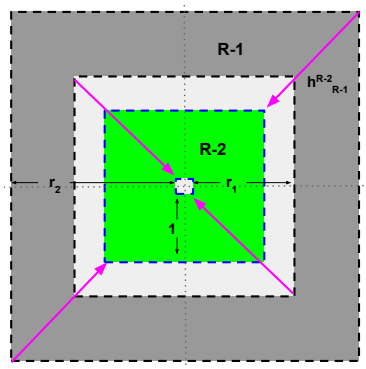


Fig. 23: Homeomorphism $h^{\text{R-2}}_{\text{R-1}}$ from R-1 to R-2

that $h^{\text{R-1}}_{\text{R-2}}(\epsilon,\epsilon) = (\epsilon,\epsilon)^{\epsilon(r_2-r_1)+r_1/\epsilon}$ becomes

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

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 $h^{\text{R-2}}_{\text{R-1}}(r_1+\epsilon, r_1+\epsilon)$. Other quadrants can be verified in a similar manner as well. It can be seen that this inverse is also continuous. Therefore, $h^{\text{R-2}}_{\text{R-1}}(\mathbf{p}^{\text{R-1}})$ 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 [165], 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 [165].

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 $f : I \to X$ 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 [165, Theorem 4.15]. Also, every continuous image of a path-connected space is path-connected [165, 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 $h^{\text{R-1}}_{\text{R-2}}$ 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

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

It can be seen in Fig. 24 as the area within the dashed green perimeter. The homeomorphism from R-2 to R-3 is seen as

$$h^{\text{R-3}}_{\text{R-2}}(\mathbf{p}^{\text{R-2}}) = \mathbf{p}^{\text{R-2}} \|\mathbf{p}^{\text{R-2}}\|_\infty / \|\mathbf{p}^{\text{R-2}}\|_2 = \mathbf{p}^{\text{R-3}}$$

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

$$h^{\text{R-2}}_{\text{R-3}}(\mathbf{p}^{\text{R-3}}) = \mathbf{p}^{\text{R-3}} \|\mathbf{p}^{\text{R-3}}\|_2 / \|\mathbf{p}^{\text{R-3}}\|_\infty = \mathbf{p}^{\text{R-2}}$$

Once again, $h^{\text{R-3}}_{\text{R-2}}$ can be seen to be continuous and bijective, with a

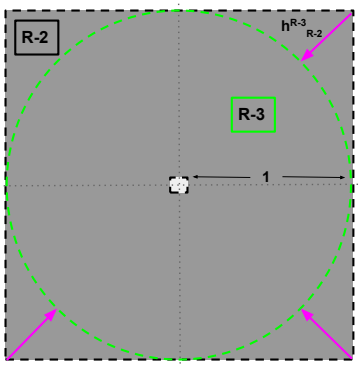


Fig. 24: Homeomorphism $h^{\text{R-3}}_{\text{R-2}}$ from R-2 to R-3

continuous inverse in the form of $h^{\text{R-2}}_{\text{R-3}}$. One can confirm that R-2 is indeed mapped to R-3 using a representative point (ϵ, ϵ), i.e., $h^{\text{R-3}}_{\text{R-2}}(\epsilon,\epsilon) = (\epsilon,\epsilon)\epsilon/\sqrt{2\epsilon^2}$ becomes

$$h^{\text{R-3}}_{\text{R-2}}(\epsilon,\epsilon) \begin{cases} \to (0,0) \text{ as } \epsilon \to 0 \\ = (p_1^{\text{R-3}}, p_2^{\text{R-3}}) \in (0,1) \times (0,1) \text{ such that} \\ \|.\|_2 < 1 \text{ as } \epsilon = c \text{ where } 0 < c < 1 \\ \to (1/\sqrt{2}, 1/\sqrt{2}) \text{ as } \epsilon \to 1 \end{cases}$$

This homeomorphism can be seen with the help of arrows in Fig. 24. Other quadrants yield the respective regions in R-3. A check of the inverse may

also be seen using $h^{\text{R-2}}_{\text{R-3}}(\epsilon,\epsilon) = (\epsilon,\epsilon)^{\sqrt{2}\epsilon}/\epsilon$, such that

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

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

$$\text{R-4} \equiv \{\mathbf{p} \in \mathbb{R}^2 \setminus \{0,0\} : r_i < \|\mathbf{p}\|_2 < r_o\} \tag{16}$$

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 Fig. 25 as the gray region. The homeomorphism from R-3 to R-4 is seen as

$$h^{\text{R-4}}_{\text{R-3}}(\mathbf{p}^{\text{R-3}}) = \mathbf{p}^{\text{R-3}}\left(\|\mathbf{p}^{\text{R-3}}\|_2 (r_o - r_i) + r_i / \|\mathbf{p}^{\text{R-3}}\|_2\right) = \mathbf{p}^{\text{R-4}}$$

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

$$h^{\text{R-3}}_{\text{R-4}}(\mathbf{p}^{\text{R-4}}) = \mathbf{p}^{\text{R-4}}\left(\|\mathbf{p}^{\text{R-4}}\|_2 - r_i\right)/(r_o - r_i)\|\mathbf{p}^{\text{R-4}}\|_2 = \mathbf{p}^{\text{R-3}}$$

One can verify that $h^{\text{R-4}}_{\text{R-3}}$ is continuous and bijective, with a continuous

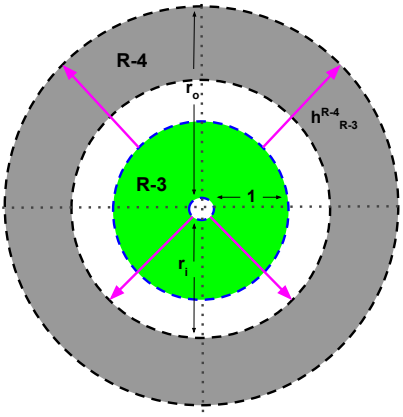


Fig. 25: Homeomorphism $h^{\text{R-4}}_{\text{R-3}}$ from R-3 to R-4

inverse in the form of $h^{\text{R-3}}_{\text{R-4}}$. Using the representative point (ϵ,ϵ), R-3 is

seen to map to R-4 by checking that $h^{\text{R-4}}_{\text{R-3}}(\epsilon,\epsilon) = (\epsilon,\epsilon)^{\sqrt{2}\epsilon(r_o-r_i)+r_i/\sqrt{2}\epsilon}$ obeys the following:

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

Other representative points, such as $(\epsilon, 0)$, may be checked by expecting that as ϵ tends to 0, the image tends to $(r_i/\sqrt{2}, 0)$, and so on. The inverse may be checked similarly by $h^{\text{R-3}}_{\text{R-4}}(\epsilon,\epsilon) = (\epsilon,\epsilon)^{\sqrt{2}\epsilon - r_i/\sqrt{2}\epsilon(r_o-r_i)}$.

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

Naturally, other regions can be shown to be mapped as well. The composition of the three homeomorphisms, $h^{\text{R-4}}_{\text{R-3}} \circ h^{\text{R-3}}_{\text{R-2}} \circ h^{\text{R-2}}_{\text{R-1}} = h^{\text{R-4}}_{\text{R-1}}$ 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 $h^{\text{R-4}}_{\text{R-1}}$. 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 [165, Proposition 2.31]. Formally, one can define charts and an atlas 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 $\{\mathcal{U}_\alpha, \phi_\alpha\} = \{\text{R-4}, Id\}$ is simply the identity map. That is, $Id : \text{R-4} \to \mathbb{R}^2$ 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 [160]. 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 11 by choosing the following parametrizations/patches. To define a surface patch, one recalls the definition of parametrization from Definition 12. The first patch is defined by $\gamma_\alpha(u, v) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^2 = (\gamma_1(u) \times \gamma_2(v))$ for $-r_o < u < r_o$ and $-r_o < v < r_o$.

$$\gamma_1(u) = u \tag{17}$$

$$\gamma_2(v) = v \tag{18}$$

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 4 locally on the patches. However, it can be quite tedious if there exist several patches. So, one invokes the following corollary from [159]:

Corollary 1. *Suppose $U \subseteq \mathbb{R}^n$ is an open subset, and $F : U \to \mathbb{R}^n$ is a smooth function whose Jacobian determinant is non-zero at every point in U. Then,*
(a) F is an open map.
(b) If F is injective, then $F : U \to F(U)$ 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 7. If there were more charts, then, 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.

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 $\mathcal{T}^{free}_{\text{PPS}_{chosen}}$ 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 [165]. The open cover of a space X is a collection $\mathcal{U}$ whose union is X, and a subcover of $\mathcal{U}$ is a subcollection 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 $U_1, \cdots U_k \in \mathbb{U}$ such that $X = U_1 \cup \cdots \cup U_k$. 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 8 (Heine-Borel)**.** *The compact subsets of* $\mathbb{R}^n$ *are exactly the closed and bounded ones. [159].*

R-4, as defined thus far in (16), is not closed, by definition. That is, it does not contain its boundary points. Then by Theorem 8, 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 8, 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 CFM and Planning Paths on the CFM

Item (ix) 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 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 2.

Proposition 2. *Every compact surface admits a polygonal presentation. [165, Proposition 6.14].*

With a CFDM at hand, one can now focus on factoring the presence of DC as described in Item (vi). Recall that the first group of DC 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 Item (v). 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:

$$\text{poles} = \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_i & \cdots & \mathbf{p}_m \end{bmatrix} \tag{19}$$

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 (19), 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 (19). That is f takes the following form:

$$f(z) = {}^{f_1(z)}\!/_{(z-\mathbf{p}_1)..(z-\mathbf{p}_i)..(z-\mathbf{p}_m)} \tag{20}$$

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:

$$L_e = \int_e f(z)dz = \int_0^1 f((1-\lambda)z_1 + z_2)(z_2 - z_1)d\lambda \tag{21}$$

where e denotes an edge of the graph. The above expression has borrowed notation from [134], and denotes the value of the integral of the holomorphic function acting on a contour segment as $L(e)$. In this work, $L(e)$ 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 Fig. 26 which provides the operational workflow of the 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 DC 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 $\mathbf{node}^{\text{start}}$ (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 $\mathbf{node}^{\text{start}}$ 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.

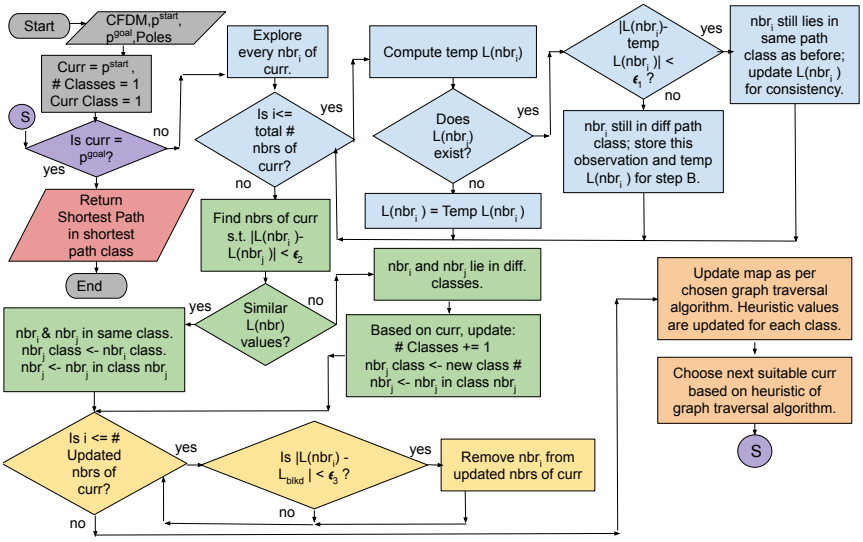


Fig. 26: Flowchart of the CFDMPP during the class marker value computation

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 $\mathbf{nbr}_i$ 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 class marker values are different for different classes. In Fig. 26, this stage's 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 **nbrs** of **curr**, the processing moves to the second section.

The second section involves the different possibilities of paths that occur by virtue of **curr** having multiple **nbrs**. The edge connecting $\mathbf{node}^{\text{start}}$ and $\mathbf{nbr}_i$ augments the existing contour from $\mathbf{node}^{\text{start}}$ to **curr**. However, since **curr** may have several **nbrs**, the existing contour may now experience n detours, caused by the respective n **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 Fig. 26, this stage's associated blocks are represented in green. Similar to the first stage, the class marker value of paths via $\mathbf{nbr}_i$ is denoted as $L(\mathbf{nbr}_i)$.

Now, all such n class marker values corresponding to n **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 **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 **nbrs** lie on distinct, new classes. Their class and node numbers are also updated. With this, the path and its detours via **curr** and its **nbrs** are fully explored.

The third section involves the second group of DC as explained in item (vi), 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 [134]. In Fig. 26, this stage is depicted 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 **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 **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 **nbrs** via **curr**. In Fig. 26, this stage is depicted 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 work) of choice chooses the next **curr** according to its heuristic.

The algorithm progresses until the final section in red is arrived at. The graph traversal algorithm, which in this case is A* returns the shortest path denoted by $\mathbf{p}_{\text{path}}$:

$$\mathbf{p}_{\text{path}} = \left[\mathbf{p}^{\text{start}} \ \cdots \ \mathbf{p}^{i} \ \cdots \ \mathbf{p}^{\text{goal}}\right] \tag{22}$$

where $\mathbf{p}^i$ is the ith 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 (22) 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 Fig. 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

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 PPS_{chosen} is represented as $\mathbf{p}_{\text{path}}{}^{\text{PPS}_{chosen}}$. Paths in other PPSs follow the same notation pattern. The path formulated in (22) is located in PPS_{chosen}. The transformation of the path to any PPS_{alt}, can be done by using the map defined by $f^{\text{PPS}_{alt}}_{\text{PPS}_{chosen}}$. That is, $\mathbf{p}_{\text{path}}{}^{\text{PPS}_{alt}} = f^{\text{PPS}_{alt}}_{\text{PPS}_{chosen}}(\mathbf{p}_{\text{path}}{}^{\text{PPS}_{chosen}})$. The mappings $f^{\text{PPS}_{alt}}_{\text{PPS}_{chosen}}$ 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 2. Therefore, the path obtained in any PPS can be transformed to another PPS without any loss of information.

4 Simulations

4.1 Setup

The performance of the proposed CFDMPP is demonstrated in a typical large warehouse environment replete with movable, multi-level shelving units and/or crates. A Costco warehouse is chosen for this purpose, since most of its aisle based shelving units are movable. The aisle space allows the operation of vehicles, for merchandise loading and offloading, in addition to human activities. Such a warehouse is the closest replica of the intended target environment to demonstrate the functionality of the CFDMPP. The robot environment is shown in Fig. 27 (not drawn to scale). Its dimensions are based on the average

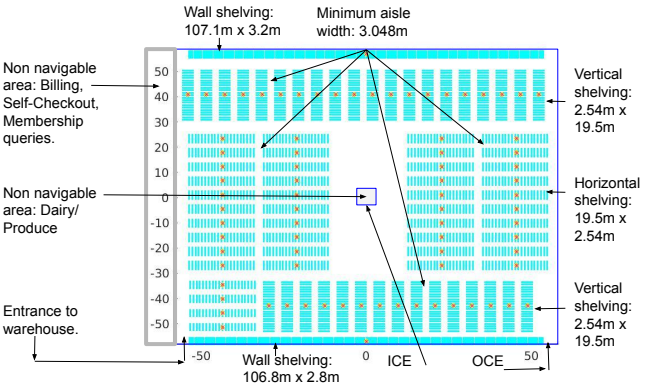


Fig. 27: Simulation environment based on the area of an average Costco Warehouse

Costco warehouse, which spans an area of 146,000 square foot (equivalent to 13,564 m^2)[1]. The singular PPS $=\mathcal{W}=$ PPPS has a dimension of $d^{\mathrm{PPPS}}=2$. The PPPS is approximated as

$$\mathrm{PPS}_{chosen}=\{(x,y)\mid(x,y)\in (-62.23\,\mathrm{m},58.23\,\mathrm{m})\times(-58.23\,\mathrm{m},58.23\,\mathrm{m})\}$$

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 no shelving units. Therefore, as seen in Fig. 27, 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. As such, $n_{\mathrm{NG}}=1+1$, $n_{\mathrm{SC}}=2$, and

$$NG_1=\{(x,y)\mid(x,y)\in [-3.536\,\mathrm{m},3.536\,\mathrm{m}]\times[-3.536\,\mathrm{m},3.536\,\mathrm{m}]\}$$
$$NG_2=\{(x,y)\mid(x,y)\in [-62.23\,\mathrm{m},-58.23\,\mathrm{m}]\times[-58.23\,\mathrm{m},58.23\,\mathrm{m}]\}$$

Applying Algorithm 2 with a $\Delta_{\mathrm{CS}}=0.5\,\mathrm{m}$ leads to two DCSs in PPS_{chosen} which are the same as the actual constraints, i.e., $\mathrm{DCS}_1=NG_1$ and $\mathrm{DCS}_2=NG_2$. 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 3.2. The constraint-free topological space is obtained as

$$\mathcal{S}^{free}_{\mathrm{PPS}_{chosen}}=\{\mathbf{p}=(x,y)\mid r_1<\|\mathbf{p}\|_\infty<r_2\} \tag{23}$$

[1] https://investor.costco.com/corporate-profile-2

for $r_1 = 3.5361\,\text{m}$ and $r_2 = 58.23\,\text{m}$. Using the same metrics and set of homeomorphisms, as seen in Section 3.2, the CFM is obtained. The R-1 representation of the constraint-free region obtained in (23) may be transformed to a CFM as R-4. The radii used in the homeomorphisms are $r_i = r_1 \times \|(1,1)\|_2 = 1.414 r_1 = 5\,\text{m}$, and $r_o = 58.23\,\text{m}$. The resulting CFM is then

$$\text{R-4} = \{\mathbf{p} = (x, y) \mid r_i < \|\mathbf{p}\|_2 < r_o\}$$

The CFDMPP is implemented using Matlab R2017b. The CFDM is produced from the CFM with the help of DistMesh (Matlab) [168, 169]. The discretization resolution of the environment (and other tuning parameters) are chosen with the following understanding. The fundamental layout of the warehouse remains constant. These are the locations of walls, structural columns, and pillars. Therefore, the discretization of the CFM representing it should 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 $0.1092\,\text{m}^2$ 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 Fig. 27. 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, as mentioned in Section 3.2, the 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 Unmanned Ground Vehicle (UGV), a clearance of approximately 0.67 m is added to the point cloud representations of the 81 shelving units. In Fig. 27, 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

Table 1: 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

in Table 1. 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. Thus, it is set by the user after at least one path is found. These parameters are discussed in Section 4.2. Note that parameters, such as mesh generation times and inter-node computation times, are a one-time cost associated with a specific discretized map. This is because the CFDMPP capitalizes on known information as a one-time initialization cost. Therefore, these times are treated as overheads. To this end, the same discretized map is used for all cases in this section. 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 Fig. 26.

4.2 Results and Analysis

Five test cases are simulated. They depict typical situations encountered in real-world warehouse environments, where UGVs are required 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. Each test 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. The results display the path in R-4 along 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.

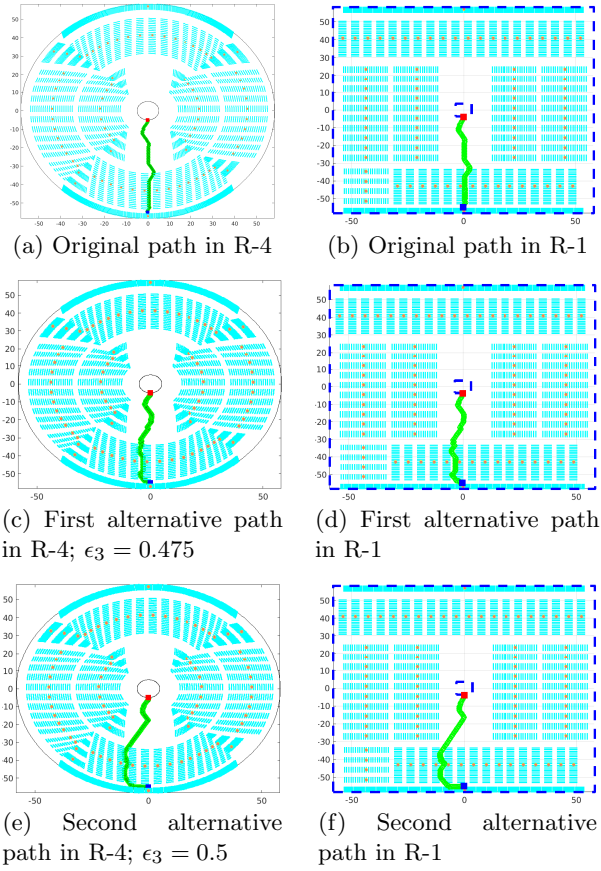
The results are summarized in Figs. 28 to 32. Cases 2, 4 and 5 depict different variants under the stocking and/or retrieving category; whereas case 3 is a clear example of a surveillance route across the length of the warehouse. Case 1 may be considered as a short range surveillance route and/or stocking/retrieval task route. In all cases, the path initially generated is seen first, followed by alternative paths. Figs. 28(a), 29(a), 30(a), 31(a) and 32(a) show paths as initially generated by the CFDMPP prior to blocking.

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. Observing Figs. 28(a), 29(a), 30(a), 31(a) and 32(a), 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. Note that these two cases require path planning over a longer range, whereas the remaining cases are for shorter distances. This behaviour 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. 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 fewer detours. This is what is noticed 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 are produced with minimal detours, barring those caused by the discretization resolution.

Arguably, different tuning parameters may be chosen. But, recall from Section 4.1, 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: Fig. 28(e) and Fig. 28(c) for case 1, and Fig. 30(e) and Fig. 30(c) for case 3. On the other hand, cases 2 , 4 and 5 are seen to produce one alternative path class each, in Figs. 29(c), 31(c) and 32(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.

The number of path classes produced via homotopic blocking now needs to be viewed in light of yet another parameter, ϵ_3. Parameters ϵ_1 and ϵ_2 together give rise to path classes and the separation amongst them, but ϵ_3 sets how a users chooses 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 Fig. 28(c). When the value of ϵ_3 is increased to 0.5, a more drastic change in the path is sought, which is what is obtained in Fig. 28(e). 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 Figs. 30(c) and 30(e), where an increase in ϵ_3 results in path classes that are increasingly different from the original class. However,



(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

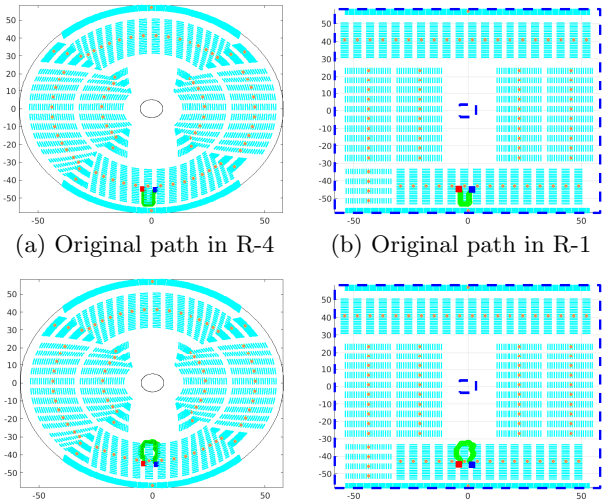
(e) Second alternative path in R-4; $\epsilon_3 = 0.5$ (f) Second alternative path in R-1

Fig. 28: Case 1: CFDMPP with and without blocking in Costco style warehouse

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.

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, for instance, 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



(a) Original path in R-4

(b) Original path in R-1

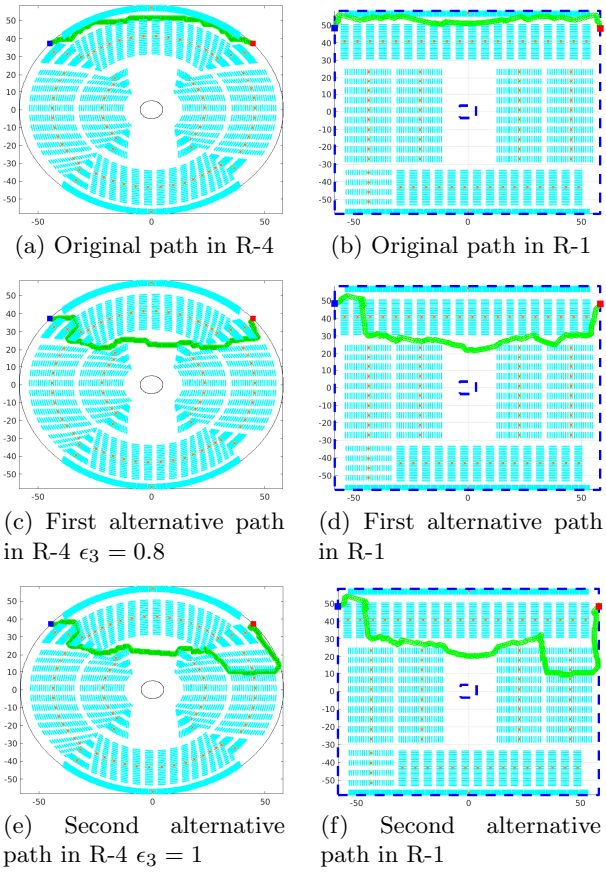
(c) First alternative path in R-4 $\epsilon_3 = 0.45$

(d) First alternative path in R-1

Fig. 29: Case 2: CFDMPP with and without blocking in Costco style warehouse

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 helpful 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 (traditional) 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



(a) Original path in R-4

(b) Original path in R-1

(c) First alternative path in R-4 $\epsilon_3 = 0.8$

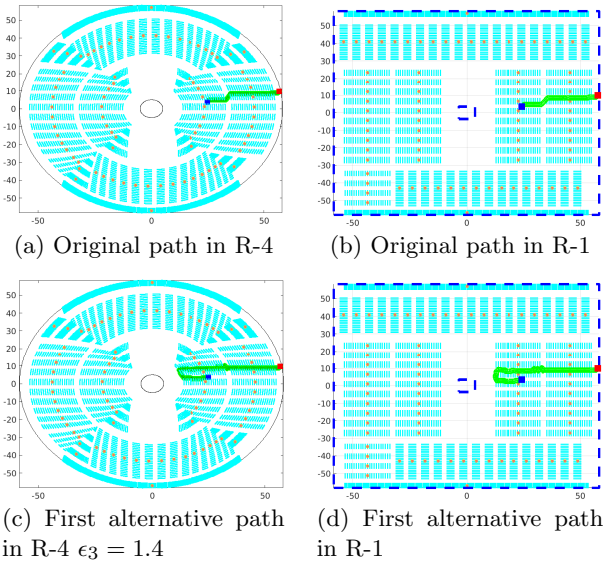
(d) First alternative path in R-1

(e) Second alternative path in R-4 $\epsilon_3 = 1$

(f) Second alternative path in R-1

Fig. 30: Case 3: CFDMPP with and without blocking in Costco style warehouse

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. This is very beneficial to the overall cost of the path planning strategy. Fig. 33 shows a pictorial analysis of the 5 cases discussed thus far. This figure shows the following information, in a related fashion: the original path lengths (without blocking) all normalized with respect to the minimum path length of all cases shown in green, the values of ϵ_3 used to query an alternative path class shown in magenta, the path lengths obtained via blocking normalized with respect to the original unblocked path length shown in blue and the number of alternative path classes produced shown in black. One can observe that the cases with the longest unblocked path lengths, lead to the highest number of alternative path classes. Correspondingly, those cases show that the change in ϵ_3 values also reflect in the change in path lengths of the resulting alternative paths. That



(a) Original path in R-4 (b) Original path in R-1

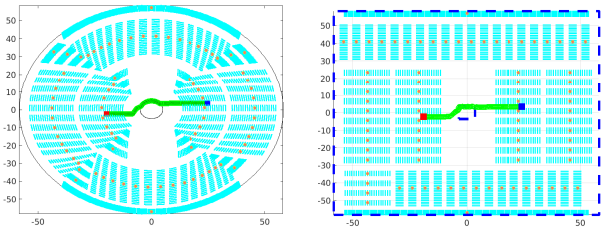
(c) First alternative path in R-4 $\epsilon_3 = 1.4$ (d) First alternative path in R-1

Fig. 31: Case 4: CFDMPP with and without blocking in Costco style warehouse

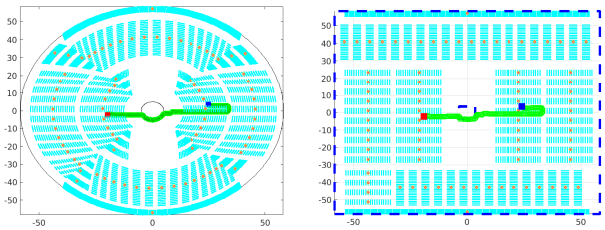
is, as the change in ϵ_3 values is greater, so is the change in the path lengths of the resulting paths. This is congruent with the theory, as higher ϵ_3 values indicate a further deviation from the chosen path class, and thus allowing for more windings around additional constraints. The number of alternative path classes is in addition to the original, unblocked path class.

SAs cannot handle such cases. As two of the most popular algorithms in the SA family, we now demonstrate how the Probabilistic Road Map (PRM) and Rapidly exploring Random Tree (RRT) algorithms fare in simulated blocking situations. In order to simulate that, both algorithms need to be run to produce initial paths. The PRM uses an open source Matlab implementation[2]. The maximum number of samples permitted to construct the tree is 1500. 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, 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. 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 an entire

[2] https://github.com/petercorke/robotics-toolbox-matlab



(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

Fig. 32: Case 5: CFDMPP with and without blocking in Costco style warehouse

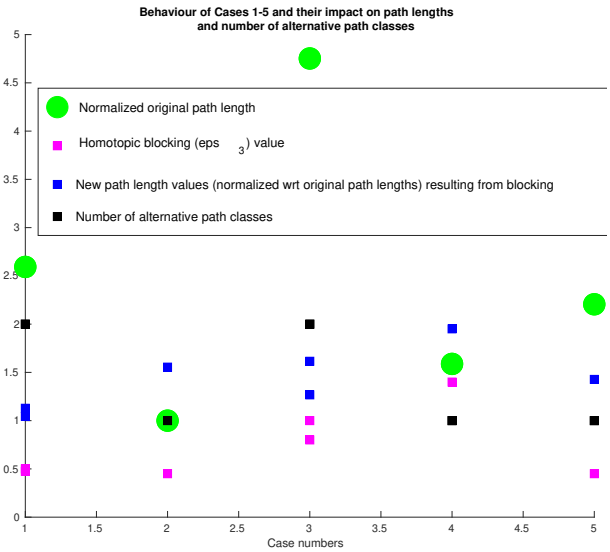


Fig. 33: Summary of the CFDMPP viewed through alternative path classes and thresholds

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

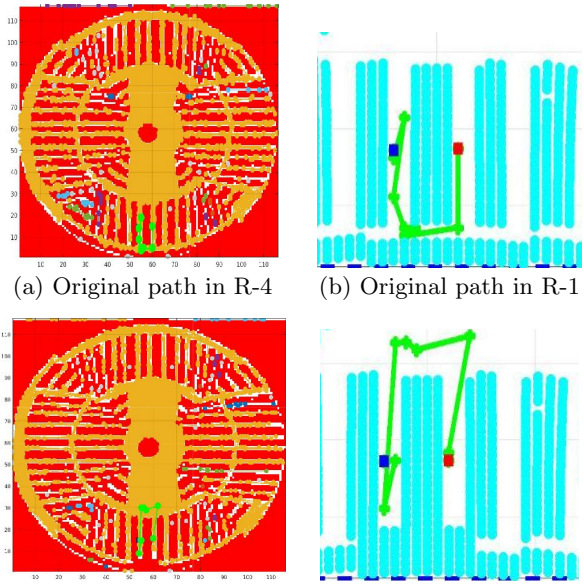
shelving units, to allow the algorithm to function. The simulated situation is the same as case 2, described earlier (see Fig. 29). The first path produced by the PRM in R-4 is seen in Fig. 34(a), and its equivalent path in R-1 (zoomed in) is depicted in Fig. 34(b). The cyan coloured shelving units appear skewed because of the transformation of their equivalent occupancy grid indices from R-4 to R-1.

Now, we attempt to simulate blocking. One cannot 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 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. Fig. 34(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 Fig. 34(d).

A similar exercise is attempted with the RRT, where the original path generated in R-4 is seen in Fig. 35(a) and a magnified equivalent of the path is shown in Fig. 35(b). The path produced via simulated blocking and its (magnified) equivalent R-1 version are demonstrated in Figs. 35(c) and 35(d), respectively. 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 a 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.



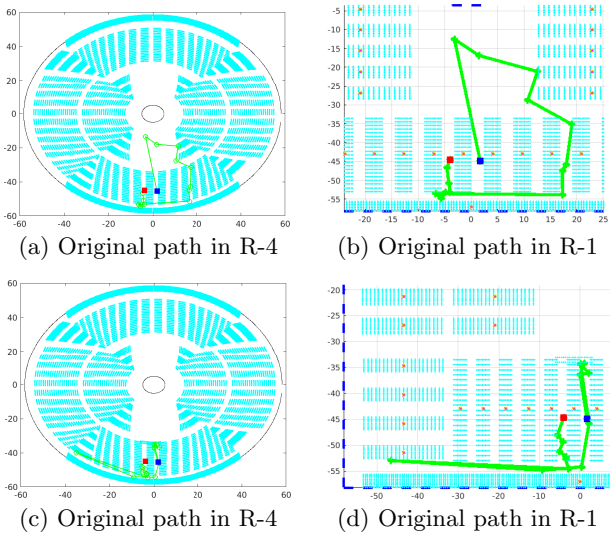
(a) Original path in R-4 (b) Original path in R-1

(c) Original path in R-4 (d) Original path in R-1

Fig. 34: Case 2: PRM with and without blocking in Costco style warehouse

5 Conclusion

The paper presents a novel PPP with three main objectives, which are typically lacking in most other path planners: The first is avoiding the periodic explicit check up for violating the constraints which are a priori known during the map construction process. Second, guaranteeing that the resultant paths



(a) Original path in R-4 (b) Original path in R-1

(c) Original path in R-4 (d) Original path in R-1

Fig. 35: Case 2: RRT with and without blocking in Costco style warehouse

are free of misinterpretations, 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. To that end, the CFDMPP is proposed and evaluated on a realistic warehouse environment. It was also benchmarked against other common algorithms, namely the PRM and RRT, to clearly highlight its superiority and ability to satisfy the aforementioned objectives.

The CFDMPP is proven to be advantageous thanks to the following characteristics. First, it maximizes knowledge of the constant regions of the map. So, when no further change in the map/constraints is encountered, the CFM is a complete representation of the constraint-free navigable PPS. Creating 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. Consequently, the second advantage is that the CFDMPP avoids the explicit verification for violating the constraints that are modelled out of the CFM. This allows paths to be formed on constraint edges without resource allocation for constraint violation. By modelling the PPS as a manifold and discretizing it, properties of continuity, transformations and discretization resolutions are combined to form the third advantage. This offers the much desired flexibility for the application dependant mesh resolution, which directly impacts the quality and number of paths produced. The prime advantage of the CFDMPP pertains to the paths produced on the CFDM. Apart from returning the shortest constraint-free paths on the CFDM, the CFDMPP uses homology and homotopic class-based path classification, avoidance and regeneration. 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. New constraint information is of consequence only when they render paths invalid. 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 outcome flexibility is preferred. Even with such variance in tuning parameters, the resulting paths do not suffer from local minima. All paths produced are viable constraint-free paths.

Despite the wealth of advantages, CFDMPP does come with a few challenges. They relate to a few aspects that are essential and inherent for the functioning of the algorithm. The CFDMPP has three tunable parameters. The first is pertaining to the path classes, while the second is the choice of meshing resolution. These two parameters are essential to initiate path planning. The third parameter is used to control the generation of 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 Section 4.2, the implications of the tuning parameters on the final results are multifaceted. The parameters present a unique intertwine of attributes for path classes that require deliberation. Therefore, while tuning allows for a higher flexibility, it may be viewed as a disadvantage, as it usually requires knowledge and/or experience with the algorithm to determine the appropriate parameter values.

Statements and Declarations

- Funding: This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). Grant RGPIN-2014-06512.
- Competing interests: The authors have no competing interests to declare that are relevant to the content of this article.
- Ethics approval: Not applicable
- Consent to participate: Not applicable
- Consent for publication: Not applicable
- Availability of data and materials: Not applicable
- Code availability: Not applicable
- Authors' contributions: All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Sindhu Radhakrishnan. The first draft of the manuscript was written by Sindhu Radhakrishnan and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

[1] Radhakrishnan, S.: Observable 2D SLAM and Evidential Occupancy Grids. Master's thesis, Carleton University (2014)

[2] Radhakrishnan, S., Gueaieb, W.: Reconfigurable EKF for 2D SLAM. In: 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), pp. 1–6 (2016). https://doi.org/10.1109/RTSI.2016.7740549

[3] Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation **12**(4), 566–580 (1996). https://doi.org/10.1109/70.508439

[4] M.LaValle, S.: Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, Ames, IA 50011 USA (June 1998)

[5] Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research **30**(7), 846–894 (2011)

[6] LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research **20**(5), 378–400 (2001)

[7] Elbanhawi, M., Simic, M.: Sampling-based robot motion planning: A review. IEEE Access **2**, 56–77 (2014). https://doi.org/10.1109/ACCESS.2014.2302442

[8] Hsu, D., Zheng Sun: Adaptively combining multiple sampling strategies for probabilistic roadmap planning. In: IEEE Conference on Robotics, Automation and Mechatronics, 2004., vol. 2, pp. 774–7792 (2004). https://doi.org/10.1109/RAMECH.2004.1438016

[9] Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), vol. 1, pp. 521–5281 (2000). https://doi.org/10.1109/ROBOT.2000.844107

[10] Akbaripour, H., Akbaripour, H., Masehian, E., Masehian, E.: 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)

[11] LaValle, S.M., Branicky, M.S., Lindemann, S.R.: On the relationship

between classical grid search and probabilistic roadmaps. The International Journal of Robotics Research **23**(7-8), 673–692 (2004)

[12] Lengyel, J., Reichert, M., Donald, B.R., Greenberg, D.P.: Real-time robot motion planning using rasterizing computer graphics hardware. Computer graphics (New York, N.Y.) **24**(4), 327–335 (1990)

[13] Stentz, A.: Optimal and efficient path planning for partially-known environments. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp. 3310–33174 (1994). https://doi.org/10.1109/ROBOT.1994.351061

[14] Dale, L.K., Amato, N.M.: Probabilistic roadmaps-putting it all together. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), vol. 2, pp. 1940–19472 (2001). https://doi.org/10.1109/ROBOT.2001.932892

[15] Dash, A.K., Chen, I.-M., Yeo, S.H., Yang, G.: 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), vol. 1, pp. 761–766 (2003)

[16] Carpin, S., Pillonetto, G.: Motion planning using adaptive random walks. IEEE Transactions on Robotics **21**(1), 129–136 (2005)

[17] Barraquand, J., Latombe, J.-C.: Robot motion planning: A distributed representation approach. The International Journal of Robotics Research **10**(6), 628–649 (2016)

[18] Kuffner, J.J., LaValle, S.M.: 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), vol. 2, pp. 995–10012 (2000). https://doi.org/10.1109/ROBOT.2000.844730

[19] Atramentov, A., LaValle, S.M.: Efficient nearest neighbor searching for motion planning. In: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), vol. 1, pp. 632–6371 (2002). https://doi.org/10.1109/ROBOT.2002.1013429

[20] Ferguson, D., Stentz, A.: Anytime RRTs. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5369–5375 (2006). https://doi.org/10.1109/IROS.2006.282100

[21] Lee, J., Pippin, C., Balch, T.: Cost based planning with RRT in outdoor environments. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 684–689 (2008). https://doi.org/10.1109/

IROS.2008.4651052

[22] Diankov, R., Kuffner, J.: Randomized statistical path planning. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1–6 (2007). https://doi.org/10.1109/IROS.2007.4399557

[23] Jaillet, L., Cortés, J., Siméon, T.: Sampling-based path planning on configuration-space costmaps. IEEE Transactions on Robotics **26**(4), 635–646 (2010)

[24] Chu, K., Lee, M., Sunwoo, M.: Local path planning for off-road autonomous driving with avoidance of static obstacles. IEEE Transactions on Intelligent Transportation Systems **13**(4), 1599–1616 (2012)

[25] Kavraki, L.E., Kolountzakis, M.N., Latombe, J.-.: Analysis of probabilistic roadmaps for path planning. IEEE Transactions on Robotics and Automation **14**(1), 166–171 (1998). https://doi.org/10.1109/70.660866

[26] Donald, B., Lynch, K.K.M.., Rus, D. (eds.): Algorithmic and Computational Robotics : New Directions 2000 WAFR, 1st edn. A K Peters/CRC Press, an imprint of Taylor and Francis, Boca Raton, FL (2001)

[27] Berenson, D., Srinivasa, S., Kuffner, J.: Task space regions: A framework for pose-constrained manipulation planning. The International Journal of Robotics Research **30**(12), 1435–1460 (2011)

[28] Jaillet, L., Porta, J.: Asymptotically-optimal path planning on manifolds. Robotics Science and Systems VIII (2012)

[29] Quinlan, S.: Real-time modification of collision-free paths. ProQuest Dissertations Publishing (1995)

[30] Quinlan, S., Khatib, O.: Elastic bands: connecting path planning and control. In: [1993] Proceedings IEEE International Conference on Robotics and Automation, pp. 802–8072 (1993). https://doi.org/10.1109/ROBOT.1993.291936

[31] Ademovic, A., Lacevic, B.: Path planning for robotic manipulators using expanded bubbles of free c-space. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), vol. 2016-, pp. 77–82 (2016). https://doi.org/10.1109/ICRA.2016.7487118

[32] Qureshi, A.H., Iqbal, K.F., Qamar, S.M., Islam, F., Ayaz, Y., Muhammad, N.: Potential guided directional-RRT* for accelerated motion planning in cluttered environments. In: 2013 IEEE International Conference on Mechatronics and Automation, pp. 519–524 (2013). https://doi.org/10.1109/ICMA.2013.6617971

[33] Qureshi, A.H., Mumtaz, S., Iqbal, K.F., Ali, B., Ayaz, Y., Ahmed, F., Muhammad, M.S., Hasan, O., Kim, W.Y., Ra, M.: Adaptive potential guided directional-RRT. In: 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1887–1892 (2013). https://doi.org/10.1109/ROBIO.2013.6739744

[34] Qureshi, A.H., Qureshi, A.H., Ayaz, Y., Ayaz, Y.: Potential functions based sampling heuristic for optimal path planning. Autonomous Robots **40**(6), 1079–1093 (2016)

[35] Lin, Y., Saripalli, S.: Sampling-based path planning for uav collision avoidance. IEEE Transactions on Intelligent Transportation Systems **18**(11), 3179–3192 (2017)

[36] Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., How, J.: Motion planning in complex environments using closed-loop prediction. In: AIAA Guidance, Navigation and Control Conference and Exhibit (2008). https://doi.org/10.2514/6.2008-7166. AIAA

[37] Wang, W., Zuo, L., Xu, X.: A learning-based multi-rrt approach for robot path planning in narrow passages. Journal of Intelligent & Robotic Systems **90**(1-2), 81–100 (2017)

[38] Suh, J., Gong, J., Oh, S.: Fast sampling-based cost-aware path planning with nonmyopic extensions using cross entropy. IEEE Transactions on Robotics **33**(6), 1313–1326 (2017)

[39] Qureshi, A.H., Ayaz, Y.: Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. Robotics and Autonomous Systems **68**, 1–11 (2015)

[40] Noreen, I., Khan, A., Ryu, H., Doh, N.L., Habib, Z.: Optimal path planning in cluttered environment using RRT-AB. Intelligent Service Robotics **11**(1), 41–52 (2017)

[41] Wei, K., Ren, B.: 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)

[42] Kang, G., Kim, Y.B., Lee, Y.H., Oh, H.S., You, W.S., Choi, H.R.: Sampling-based motion planning of manipulator with goal-oriented sampling. Intelligent Service Robotics **12**(3), 265–273 (2019)

[43] McMahon, T., Thomas, S., Amato, N.M.: Sampling-based motion planning with reachable volumes for high-degree-of-freedom manipulators. The International Journal of Robotics Research **37**(7), 779–817 (2018)

[44] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research **5**(1), 90–98 (1986)

[45] Siciliano, B.: Robotics Modelling, Planning and Control, 1st ed. 2009. edn. Advanced Textbooks in Control and Signal Processing. Springer, London (2009). https://doi.org/10.1007/978-1-84628-642-1

[46] Oriolo, G.: Motion Planning 3 Artifical Potential Fields. Professor Oriolo's notes for his class in Artificial Intelligence and Robotics. (2020). http://diag.uniroma1.it/oriolo/amr/slides/MotionPlanning3_Slides.pdf

[47] Rasekhipour, Y., Fadakar, I., Khajepour, A.: Autonomous driving motion planning with obstacles prioritization using lexicographic optimization. Control Engineering Practice **77**, 235–246 (2018)

[48] Volpe, R., Khosla, P.: Artificial potentials with elliptical isopotential contours for obstacle avoidance. In: 26th IEEE Conference on Decision and Control, vol. 26, pp. 180–185 (1987). https://doi.org/10.1109/CDC.1987.272738

[49] Faverjon, B., Tournassoud, P.: 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, vol. 4, pp. 1152–1159 (1987). https://doi.org/10.1109/ROBOT.1987.1087982

[50] Koditschek, D.: Exact robot navigation by means of potential functions: Some topological considerations. In: Proceedings. 1987 IEEE International Conference on Robotics and Automation, vol. 4, pp. 1–6 (1987). https://doi.org/10.1109/ROBOT.1987.1088038

[51] Khosla, P., Volpe, R.: Superquadric artificial potentials for obstacle avoidance and approach. In: Proceedings. 1988 IEEE International Conference on Robotics and Automation, pp. 1778–17843 (1988). https://doi.org/10.1109/ROBOT.1988.12323

[52] Volpe, R., Khosla, P.: Manipulator control with superquadric artificial potential functions: theory and experiments. IEEE Transactions on Systems, Man, and Cybernetics **20**(6), 1423–1436 (1990). https://doi.org/10.1109/21.61211

[53] Leica, P., Chavez, D., Rosales, A., Roberti, F., Toibero, J.M., Carelli, R.: Strategy based on multiple objectives and null space for the formation of mobile robots and dynamic obstacle avoidance. Revista Politécnica (Quito) **33**(1) (2014)

[54] Agirrebeitia, J., Avilés, R., de Bustos, I.F., Ajuria, G.: A new APF strategy for path planning in environments with obstacles. Mechanism and machine theory **40**(6), 645–658 (2005)

[55] Paikray, H.K., Das, P.K., Panda, S.: Optimal path planning of multi-robot in dynamic environment using hybridization of meta-heuristic algorithm. International journal of intelligent robotics and applications Online **6**(4), 625–667 (2022)

[56] Park, J.-M., Kim, D.-W., Yoon, Y.-S., Kim, H.J., Yi, K.-S.: 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)

[57] Gao, Y., Lin, T., Borrelli, F., Tseng, E., Hrovat, D.: Predictive Control of Autonomous Ground Vehicles With Obstacle Avoidance on Slippery Roads. Dynamic Systems and Control Conference, vol. ASME 2010 Dynamic Systems and Control Conference, Volume 1, pp. 265–272 (2010). https://doi.org/10.1115/DSCC2010-4263. ASME. https://doi.org/10.1115/DSCC2010-4263

[58] Yoon, Y., Shin, J., Kim, H.J., Park, Y., Sastry, S.: Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. Control Engineering Practice **17**(7), 741–750 (2009)

[59] Rasekhipour, Y., Khajepour, A., Chen, S., Litkouhi, B.: A potential field-based model predictive path-planning controller for autonomous road vehicles. IEEE Transactions on Intelligent Transportation Systems **18**(5), 1255–1267 (2017). https://doi.org/10.1109/TITS.2016.2604240

[60] Sgorbissa, A.: 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)

[61] Abbas, M.A., Milman, R., Eklund, J.M.: Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles. In: 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1–6 (2014). https://doi.org/10.1109/CCECE.2014.6901109

[62] Gao, Y., Gray, A., Tseng, H.E., Borrelli, F.: A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. Vehicle System Dynamics **52**(6), 802–823 (2014) https://doi.org/10.1080/00423114.2014.902537. https://doi.org/10.1080/00423114.2014.902537

[63] Montiel, O., Orozco-Rosas, U., Sepúlveda, R.: Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. Expert Systems with Applications **42**(12), 5177–5191 (2015)

[64] Aghababa, M.P.: 3D Path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles. Applied Ocean Research **38**, 48–62 (2012)

[65] Wang, C., Mao, Y.S., Du, K.J.: Simulation on local obstacle avoidance algorithm for unmanned surface vehicle. International Journal of Simulation Modelling **15**(3), 460–472 (2016)

[66] Kim, H., Cheang, U.K., Rogowski, L.W., Kim, M.J.: Motion planning of particle based microrobots for static obstacle avoidance. Journal of Micro-Bio Robotics **14**(1-2), 41–49 (2018)

[67] Wang, D., Wang, P., Zhang, X., Guo, X., Shu, Y., Tian, X.: An obstacle avoidance strategy for the wave glider based on the improved artificial potential field and collision prediction model. Ocean Engineering **206**, 107356 (2020)

[68] Campana, M., Lamiraux, F., Laumond, J.-P.: A gradient-based path optimization method for motion planning. Advanced Robotics **30**(17-18), 1126–1144 (2016)

[69] Koditschek, D.E., Rimon, E.: Robot navigation functions on manifolds with boundary. Advances in Applied Mathematics **11**(4), 412–442 (1990)

[70] Quillen, P., Muñoz, J., Subbarao, K.: Path planning to a reachable state using minimum control effort based navigation functions. The Journal of the astronautical sciences **66**(4), 554–581 (2019)

[71] Kowalczyk, W., Kowalczyk, W., Przybyla, M., Przybyla, M., Kozlowski, K., Kozlowski, K.: 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)

[72] Kowalczyk, W.: Rapid navigation function control for two-wheeled mobile robots. Journal of Intelligent & Robotic Systems **93**(3-4), 687–697 (2018)

[73] Tao, S., Tan, J.: Path planning with obstacle avoidance based on normalized r -functions. Journal of Robotics **2018**, 1–10 (2018)

[74] Stopp, A., Riethmuller, T.: 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, pp. 545–550

(1995)

[75] Murphy, R.R., Hughes, K., Marzilli, A., Noll, E.: Integrating explicit path planning with reactive control of mobile robots using trulla. Robotics and Autonomous Systems **27**(4), 225–245 (1999)

[76] Hughes, K., Tokuta, A., Ranganathan, N.: 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, vol. 1, pp. 469–476 (1992). https://doi.org/10.1109/IROS.1992.587377

[77] Mediavilla, M., González, J.L., Fraile, J.C., Ramón Perán, J.: Reactive approach to on-line path planning for robot manipulators in dynamic environments. Robotica **20**(4), 375–384 (2002)

[78] Diéguez, A.R., Sanz, R., López, J.: Deliberative on-line local path planning for autonomous mobile robots. Journal of Intelligent & Robotic Systems **37**(1), 1–19 (2003)

[79] Lamiraux, F., Bonnafous, D., Lefebvre, O.: Reactive path deformation for nonholonomic mobile robots. IEEE Transactions on Robotics **20**(6), 967–977 (2004)

[80] Simeon, T., Laumond, J.-P., Van Geem, C.V., Cortes, J.: Computer aided motion: Move3d within molog. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), vol. 2, pp. 1494–1499 (2001). https://doi.org/10.1109/ROBOT.2001.932822

[81] van den Berg, J., Overmars, M.: Planning time-minimal safe paths amidst unpredictably moving obstacles. The International Journal of Robotics Research **27**(11-12), 1274–1294 (2008)

[82] McFetridge, L., Ibrahim, M.Y.: A new methodology of mobile robot navigation: The agoraphilic algorithm. Robotics and Computer-Integrated Manufacturing **25**(3), 545–551 (2009)

[83] Belkhouche, F.: Reactive path planning in a dynamic environment. IEEE Transactions on Robotics **25**(4), 902–911 (2009)

[84] Belkhouche, F., Bendjilali, B.: Reactive path planning for 3-D autonomous vehicles. IEEE Transactions on Control Systems Technology **20**(1), 249–256 (2012)

[85] Wu, A., How, J.P.: Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. Autonomous Robots **32**(3),

227–242 (2012)

[86] Cockayne, E.J., Hall, G.W.C.: Plane motion of a particle subject to curvature constraints. SIAM Journal on Control **13**(1), 197–220 (1975)

[87] De Filippis, L., Guglieri, G., Quagliotti, F.: Path planning strategies for UAVs in 3D environments. Journal of Intelligent & Robotic Systems **65**(1), 247–264 (2012)

[88] Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., Jurišica, L.: Path planning with modified a star algorithm for a mobile robot. Procedia Engineering **96**, 59–69 (2014)

[89] Savkin, A.V., Hoy, M.: Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments. Robotica **31**(2), 323–330 (2013)

[90] Hossain, M.A., Ferdous, I.: 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)

[91] Cui, P., Yan, W., Wang, Y.: Reactive path planning approach for docking robots in unknown environment. Journal of Advanced Transportation **2017**, 1–11 (2017)

[92] Hildebrandt, A.-C., Klischat, M., Wahrmann, D., Wittmann, R., Sygulla, F., Seiwald, P., Rixen, D., Buschmann, T.: Real-time path planning in unknown environments for bipedal robots. IEEE Robotics and Automation Letters **2**(4), 1856–1863 (2017)

[93] Bevilacqua, P., Frego, M., Fontanelli, D., Palopoli, L.: Reactive planning for assistive robots. IEEE Robotics and Automation Letters **3**(2), 1276–1283 (2018)

[94] Ataka, A., Lam, H.-K., Althoefer, K.: Reactive magnetic-field-inspired navigation method for robots in unknown convex 3-D environments. IEEE Robotics and Automation Letters **3**(4), 3583–3590 (2018)

[95] Li, X., Zhao, G., Li, B.: Generating optimal path by level set approach for a mobile robot moving in static/dynamic environments. Applied Mathematical Modelling **85**, 210–230 (2020)

[96] Wang, B., Liu, Z., Li, Q., Prorok, A.: Mobile robot path planning in dynamic environments through globally guided reinforcement learning. IEEE Robotics and Automation Letters **5**(4), 6932–6939 (2020)

[97] Ginesi, M., Meli, D., Roberti, A., Sansonetto, N., Fiorini, P.: Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions. Journal of Intelligent & Robotic Systems **101**(4) (2021)

[98] Wada, H., Kinugawa, J., Kosuge, K.: Reactive motion planning using time-layered c-spaces for a collaborative robot pady. Advanced Robotics **35**(8), 490–503 (2021)

[99] Alonso-Mora, J., DeCastro, J.A., Raman, V., Rus, D., Kress-Gazit, H.: Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. Autonomous Robots **42**(4), 801–824 (2018)

[100] Kattepur, A., Purushotaman, B.: Roboplanner: a pragmatic task planning framework for autonomous robots. Cognitive Computation and Systems **2**(1), 12–22 (2020)

[101] Brooks, R.A.: Planning collision- free motions for pick-and-place operations. The International Journal of Robotics Research **2**(4), 19–44 (1983)

[102] Dorst, L., Mandhyan, I., Trovato, K.: The geometrical representation of path planning problems. Robotics and Autonomous Systems **7**(2), 181–195 (1991)

[103] Farber, M.: Topological complexity of motion planning. Discrete & Computational Geometry **29**(2), 211–221 (2003)

[104] Błaszczyk, Z., Carrasquel-Vera, J.G.: Topological complexity and efficiency of motion planning algorithms. Revista Matemática Iberoamericana **34**(4), 1679–1684 (2018)

[105] Trinkle, J.C., Milgram, R.J.: Complete path planning for closed kinematic chains with spherical joints. The International Journal of Robotics Research **21**(9), 773–789 (2002)

[106] Shvalb, N., Shoham, M., Liu, G., Trinkle, J.C.: Motion planning for a class of planar closed-chain manipulators. The International Journal of Robotics Research **26**(5), 457–473 (2007)

[107] Tanner, H.G., Kumar, A.: Towards decentralization of multi-robot navigation functions. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 4132–4137 (2005)

[108] Ryu, J.C., Ryu, J.C., Park, F.C., Park, F.C., Kim, Y.Y., Kim, Y.Y.: Mobile robot path planning algorithm by equivalent conduction heat flow topology optimization. Structural and multidisciplinary optimization

45(5), 703–715 (2012)

[109] Roy, D.: Algorithmic path planning of static robots in three dimensions using configuration space metrics. Robotica **29**(2), 295–315 (2011)

[110] Kennedy, M., Thakur, D., Ani Hsieh, M., Bhattacharya, S., Kumar, V.: Optimal paths for polygonal robots in SE(2). Journal of Mechanisms and Robotics **10**(2) (2018)

[111] Masehian, E., Amin-Naseri, M.R.: A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. Journal of Robotic Systems **21**(6), 275–300 (2004)

[112] Olmstead Muhs, J.C., Yang, J.: A geodesics-based model for obstacle avoidance. In: 2005 Digital Human Modeling for Design and Engineering Symposium (2005). https://doi.org/10.4271/2005-01-2692

[113] Xu, B., Xu, B., Stilwell, D.J., Stilwell, D.J., Kurdila, A.J., Kurdila, A.J.: Fast path re-planning based on fast marching and level sets. Journal of Intelligent & Robotic Systems **71**(3), 303–317 (2013)

[114] Bhattacharya, S., Pivtoraiko, M.: A classification of configuration spaces of planar robot arms for a continuous inverse kinematics problem. Acta applicandae mathematicae **139**(1), 133–166 (2015)

[115] Bhattacharya, S., Ghrist, R., Kumar, V.: Multi-robot coverage and exploration on riemannian manifolds with boundaries. The International Journal of Robotics Research **33**(1), 113–137 (2014)

[116] Wu, K., Lo, C., Lin, Y., Liu, J.: 3D path planning based on nonlinear geodesic equation. In: 11th IEEE International Conference on Control Automation (ICCA), pp. 342–347 (2014). https://doi.org/10.1109/ICCA.2014.6870943

[117] Qin, L., Yin, Q., Zha, Y., Peng, Y.: Dynamic detection of topological information from grid-based generalized voronoi diagrams. Mathematical Problems in Engineering **2013**, 1–11 (2013)

[118] Wu, K.-L., Ho, T.-J., Huang, S.A., Lin, K.-H., Lin, Y.-C., Liu, J.-S.: Path planning and replanning for mobile robot navigation on 3D terrain: An approach based on geodesic. Mathematical Problems in Engineering **2016**, 1–12 (2016)

[119] Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Autonomous Robots **33**(3), 273–290 (2012)

[120] Bhattacharya, S.: Towards optimal path computation in a simplicial complex. The International Journal of Robotics Research **38**(8), 981–1009 (2019)

[121] Zhang, B., Liu, Y., Lu, Q., Wang, J.: A path planning strategy for searching the most reliable path in uncertain environments. International Journal of Advanced Robotic Systems **13**(5), 172988141665775 (2016). https://doi.org/10.1177/1729881416657751

[122] Jaillet, L., Simeon, T.: Path deformation roadmaps: Compact graphs with useful cycles for motion planning. The International Journal of Robotics Research **27**(11-12), 1175–1188 (2008). https://doi.org/10.1177/0278364908098411

[123] Havoutis, I., Ramamoorthy, S.: Motion planning and reactive control on learnt skill manifolds. The International Journal of Robotics Research **32**(9-10), 1120–1150 (2013)

[124] Bohigas, O., Henderson, M.E., Ros, L., Porta, J.M.: A singularity-free path planner for closed-chain manipulators. In: 2012 IEEE International Conference on Robotics and Automation, pp. 2128–2134 (2012). https://doi.org/10.1109/ICRA.2012.6224899

[125] Bohigas, O., Henderson, M.E., Ros, L., Manubens, M., Porta, J.M.: Planning singularity-free paths on closed-chain manipulators. IEEE Transactions on Robotics **29**(4), 888–898 (2013). https://doi.org/10.1109/TRO.2013.2260679

[126] Porta, J.M., Jaillet, L., Bohigas, O.: Randomized path planning on manifolds based on higher-dimensional continuation. The International Journal of Robotics Research **31**(2), 201–215 (2012)

[127] Henderson, M.E.: Multiple parameter continuation: Computing implicitly defined k-manifolds. International Journal of Bifurcation and Chaos in Applied Sciences and Engineering **12**(3), 451–476 (2002)

[128] Jaillet, L., Porta, J.M.: Path planning under kinematic constraints by rapidly exploring manifolds. IEEE Transactions on Robotics **29**(1), 105–117 (2013). https://doi.org/10.1109/TRO.2012.2222272

[129] Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences - PNAS **93**(4), 1591–1595 (1996)

[130] Kimmel, R., Sethian, J.A.: Computing geodesic paths on manifolds. Proceedings of the National Academy of Sciences - PNAS **95**(15), 8431–8435 (1998)

[131] Sethian, J.A.: Fast marching methods. SIAM review **41**(2), 199–235 (1999)

[132] Liu, G., Trinkle, J., Yang, Y., Luo, S.: Motion planning of planar closed chains based on structural sets. IEEE Access **8**, 117203–117217 (2020)

[133] Cabello, S., Liu, Y., Mantler, A., Snoeyink, J.: Testing homotopy for paths in the plane. Discrete & Computational Geometry **31**(1), 61–81 (2004)

[134] Bhattacharya, S., Kumar, V., Likhachev, M.: Search-based path planning with homotopy class constraints. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. AAAI'10, pp. 1230–1237 (2010)

[135] Bhattacharya, S., Likhachev, M., Kumar, V.: Identification and representation of homotopy classes of trajectories for search-based path planning in 3D. In: Durrant-Whyte, H., Roy, N., Abbeel, P. (eds.) Robotics: Science and Systems VII. The MIT Press, One Broadway 12th Floor Cambridge, MA 02142 (2012). https://doi.org/10.7551/mitpress/9481.003.0007

[136] Bhattacharya, S., Likhachev, M., Kumar, V.: 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)

[137] Kim, S., Sreenath, K., Bhattacharya, S., Kumar, V.: Trajectory Planning for Systems with Homotopy Class Constraints. In: Latest Advances in Robot Kinematics (ARK), Innsbruck, Austria, pp. 83–90 (2012)

[138] Hernandez, E., Carreras, M., Ridao, P.: A comparison of homotopic path planning algorithms for robotic applications. Robotics and Autonomous Systems **64**, 44–58 (2015)

[139] Hernández, E., Carreras, M., Ridao, P.: A path planning algorithm for an AUV guided with homotopy classes. In: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling, vol. 21 (2011)

[140] Hernández, E., Carreras, M., Ridao, P.: 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 (2012)

[141] Jenkins, K.D.: 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)

[142] Lumelsky, V.J., Stepanov, A.A.: Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica **2**(1-4), 403–430 (1987)

[143] Kim, D., Kang, M., Yoon, S.-E.: Volumetric tree: Adaptive sparse graph for effective exploration of homotopy classes. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1496–1503 (2019)

[144] Vazquez-Leal, H., Marin-Hernandez, A., Khan, Y., Yıldırım, A., Filobello-Nino, U., Castaneda-Sheissa, R., Jimenez-Fernandez, V.M.: Exploring collision-free path planning by using homotopy continuation methods. Applied Mathematics and Computation **219**(14), 7514–7532 (2013)

[145] Eduardo De Cos-Cholula, H., Ulises Diaz-Arango, G., Hernandez-Martinez, L., Vazquez-Leal, H., Sarmiento-Reyes, A., Teresa Sanz-Pascual, M., Leobardo Herrera-May, A., Castaneda-Sheissa, R.: FPGA implementation of homotopic path planning method with automatic assignment of repulsion parameter. Energies (Basel) **13**(10), 2623 (2020)

[146] Diaz-Arango, G., Vazquez-Leal, H., Hernandez-Martinez, L., Manuel Jimenez-Fernandez, V., Heredia-Jimenez, A., Ambrosio, R.C., Huerta-Chua, J., De Cos-Cholula, H., Hernandez-Mendez, S.: Multiple-target homotopic quasi-complete path planning method for mobile robot using a piecewise linear approach. Sensors (Basel, Switzerland) **20**(11), 3265 (2020)

[147] Diaz-Arango, G., Sarmiento-Reyes, A., Hernandez-Martinez, L., Vazquez-Leal, H., Lopez-Hernandez, D.D., Marin-Hernandez, A.: Path optimization for terrestrial robots using homotopy path planning method. In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2824–2827 (2015)

[148] Park, J., Karumanchi, S., Iagnemma, K.: Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming. IEEE Transactions on Robotics **31**(5), 1101–1115 (2015)

[149] Yi, D., Goodrich, M., Seppi, K.: Homotopy-aware RRT: Toward human-robot topological path-planning. In: The Eleventh ACM/IEEE International Conference on Human Robot Interaction. HRI '16, pp. 279–286 (2016)

[150] Kala, R.: Homotopy conscious roadmap construction by fast sampling of

narrow corridors. Applied Intelligence (Dordrecht, Netherlands) **45**(4), 1089–1102 (2016)

[151] Kala, R.: Homotopic roadmap generation for robot motion planning. Journal of Intelligent & Robotic Systems **82**(3), 555–575 (2016)

[152] Yao, W., Qi, N., Zhao, J., Wan, N.: Bounded curvature path planning with expected length for dubins vehicle entering target manifold. Robotics and Autonomous systems **97**, 217–229 (2017)

[153] Liu, Y., Qi, N., Yao, W., Zhao, J., Xu, S.: Cooperative path planning for aerial recovery of a UAV swarm using genetic algorithm and homotopic approach. Applied Sciences **10**(12), 4154 (2020)

[154] Gregoire, J., Čáp, M., Frazzoli, E.: Locally-optimal multi-robot navigation under delaying disturbances using homotopy constraints. Autonomous Robots **42**(4), 895–907 (2018)

[155] Kolur, K., Chintalapudi, S., Boots, B., Mukadam, M.: Online motion planning over multiple homotopy classes with gaussian process inference. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2358–2364 (2019)

[156] LIU, Y., ZHENG, Z., QIN, F.: Homotopy based optimal configuration space reduction for anytime robotic motion planning. Chinese Journal of Aeronautics **34**(1), 364–379 (2021)

[157] Bloch, A., Camarinha, M., Colombo, L.J.: Dynamic interpolation for obstacle avoidance on riemannian manifolds. International Journal of Control **94**(3), 588–600 (2021)

[158] Tu, L.W.: An Introduction to Manifolds, 2nd edn. Universitext. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-7400-6

[159] Lee, J.M.: Introduction to Smooth Manifolds., 2nd edn. Graduate Texts in Mathematics ; v.218. Springer, New York, NY (2002). https://doi.org/10.1007/978-1-4419-9982-5

[160] Wilson, J.: Manifolds. Notes for graduate students via seminar by Dr.Jenny Wilson (2012). http://www.math.lsa.umich.edu/\texttildelowjchw/WOMPtalk-Manifolds.pdf

[161] Nicolaescu, L.I.: Lectures on the Geometry of Manifolds vol. 32, 2nd edn. Ringgold Inc, Portland (2008). http://search.proquest.com/docview/200118389/

[162] Nicolaescu, L.: Homeomorphisms vs. Diffeomorphisms. Professor

Nicolaescu's notes on the topic. (2003). https://www3.nd.edu/\texttildelowlnicolae/FYsem2003.pdf

[163] Pressley, A.: Elementary Differential Geometry, 2nd edn. Springer undergraduate mathematics series. Springer, London (2010). https://doi.org/10.1007/978-1-84882-891-9

[164] Munkres, J.R.: Topology, 2nd ed. edn. Prentice Hall, Upper Saddle River, NJ (2000)

[165] Lee, J..M.: Introduction To Topological Manifolds., 2nd edn. Springer, Dordrecht (2011)

[166] Needham, T.: Visual Complex Analysis. Oxford University Press, Oxford University Press Inc., New York (1997)

[167] Stein, E.M.: Complex Analysis. Princeton lectures in analysis ; 2. Princeton University Press, Princeton, N.J (2003)

[168] Persson, P.-O.: Mesh generation for implicit geometries. Ph.D. dissertation, Massachusetts Institute of Technology (February 2005). http://persson.berkeley.edu/thesis/persson-thesis.pdf

[169] Persson, P.-O., Strang, G.: A simple mesh generator in matlab. SIAM Review **46**(2), 329–345 (2004). https://doi.org/10.1137/S0036144503429121