

Système de cartographie 3D et de planification pour la manipulation robotique d'objets non rigides

par

Félix Nadon

Thèse soumise à l'Université d'Ottawa dans le cadre des exigences du programme de
PhD en Génie Électrique et Génie Informatique

L'Institut de génie électrique et de génie informatique d'Ottawa-Carleton
École de science informatique et de génie électrique
Faculté de Génie
Université d'Ottawa

Résumé

La manipulation intelligente des matériaux non rigides permettrait l'utilisation de robots à usage général dans l'automatisation de plusieurs tâches délicates ou demandant beaucoup de main-d'oeuvre, allant de l'assemblage industriel à l'exploration spatiale en passant par les procédures médicales, la manipulation des aliments et les travaux domestiques. Bien qu'une variété de stratégies aient été proposées au fil du temps pour manipuler les objets non rigides, celles-ci sont généralement spécifiques à une tâche particulière et imposent des contraintes sévères sur le type d'objet manipulé et l'environnement. Il n'existe pas encore de système global capable de planifier et d'exécuter, dans son ensemble, la saisie et la mise en forme d'un objet non rigide arbitraire, en particulier lorsque l'environnement n'est que partiellement observable et que les propriétés de l'objet sont initialement inconnues. L'objectif général de cette thèse est de contribuer au développement d'un tel système.

L'atteinte de cet objectif général passe par trois axes de contribution principaux, soit la conception globale du système, le développement d'un module de cartographie original adapté à la représentation d'objets déformables tridimensionnels, et la conception de mesures heuristiques pour guider l'observation et le remodelage de ces objets. Ainsi, le système global forme une solution intégrée pour le remodelage d'objets déformables tridimensionnels dans un environnement partiellement observable.

Cette recherche propose un système de cartographie original qui se distingue des solutions existantes en étant axé sur la manipulation d'objets déformables tridimensionnels, ainsi que des algorithmes de comparaison de formes adaptés à ce problème.

En termes de planification de tâche, le système doit coordonner les actions des différents éléments en fonction de la tâche à effectuer et des contraintes de l'environnement de travail. Après une prise de vue initiale, la tâche à accomplir est définie et des heuristiques d'observation sont intégrées dans des algorithmes pour observer la forme initiale de l'objet d'intérêt le plus complètement possible. Une fois cette forme obtenue, des heuristiques de manipulation sont proposées pour la sélection des points de prise et l'estimation du mouvement à accomplir. Un point de vue est également sélectionné pour effectuer le suivi de la forme en cours de déformation et, à la fin de la manipulation, la forme finale et la forme cible sont comparées. À cette fin, différentes mesures originales sont proposées pour évaluer la qualité du remodelage.

L'évaluation expérimentale du système proposé débute avec un robot physique équipé d'une caméra RGBD, permettant d'évaluer, de comparer et d'optimiser différentes méthodes pour la fusion des observations et la segmentation d'un objet d'intérêt dont la forme est

initialement inconnue. Pour plus de répétabilité, la simulation est ensuite utilisée pour évaluer les autres composantes du système. Ainsi, les résultats obtenus montrent que les heuristiques d'observation permettent d'observer la forme de l'objet d'intérêt avec efficacité et précision. Les algorithmes de comparaison et les heuristiques de manipulation proposés permettent de choisir des points de prise et des déplacements visuellement plausibles qui, une fois la manipulation exécutée, permettent de rapprocher la forme de l'objet d'intérêt de la forme cible. Globalement, le système et les algorithmes développés dans le cadre de cette thèse représentent un avancement significatif vers un système intégré capable de gérer, de bout à bout et avec un minimum d'intervention humaine dans un environnement 3D, l'observation, la comparaison, la planification, la manipulation et le suivi de forme pour le remodelage d'un objet déformable dont les propriétés de forme et de matériau sont initialement inconnues.

Remerciements

Si le voyage est plus important que la destination, on n'en est pas moins heureux d'apercevoir celle-ci — et la prochaine étape — à la fin d'un long parcours. Débuté à l'aube d'une pandémie et jonché de péripéties et de revers, le moins qu'on puisse dire est que celui-ci n'a pas été de tout repos ! Et comme tout long voyage, son résultat n'aurait sans doute pas été ce qu'il est sans l'appui et la contribution de nombreuses personnes tout au long de la route.

Ainsi, je souhaite d'abord remercier mon directeur de thèse, le professeur Pierre Payeur, pour son appui et ses conseils précieux tout au long de mon parcours. Je remercie également les membres de mon comité d'évaluation, les professeurs Jean-Philippe Roberge, Carlos Rossa, Eric Lanteigne et Jean-Lou De Carufel pour leurs commentaires avisés et leur patience dans la révision de ce volumineux manuscrit. Je remercie de même les professeurs Claude Laguë, Emil Petriu et Carlos Rossa pour leurs recommandations au moment de la proposition de thèse.

Un immense merci à ma famille et à mes amis qui m'ont soutenu chacun à leur manière pendant toutes ces années consacrées à ce projet. Des moments de découragement aux réussites célébrées, les mots ne sauraient exprimer l'impact de votre soutien sur mon parcours.

Je remercie aussi le programme de bourses d'études supérieures du Canada au niveau du doctorat du Conseil de recherches en sciences naturelles et en génie du Canada ainsi que celui de bourse d'études supérieures de l'Ontario pour leur appui financier.

Enfin, je souhaite remercier les nombreux autres que je ne saurais nommer mais qui, de près ou de loin, ont contribué à l'accomplissement de ce projet.

À ceux qu'on a perdus en chemin.

Table des matières

Liste des tableaux	x
Liste des figures	xi
Liste des algorithmes	xvi
Nomenclature	xvii
Chapitre 1 Introduction	1
1.1 Motivation et contexte	1
1.2 Définition du problème et solutions existantes	3
1.3 Objectifs	5
1.4 Organisation	7
Chapitre 2 Revue de Littérature	8
2.1 Manipulation d'objets déformables	8
2.1.1 Objets linéaires	9
2.1.2 Objets plans	11
2.1.3 Objets tridimensionnels	13
2.1.4 Diversité des propriétés d'objets déformables	19
2.2 Cartographie et représentation de scène	21
2.2.1 Cartographie des formes et de l'espace	21
2.2.2 Environnements voxélisés	25
2.3 Segmentation et comparaison de formes en 3D	26
2.3.1 Segmentation	26
2.3.2 Comparaison de formes	28
2.4 Planification de la manipulation	29

2.4.1	Planification de tâche	30
2.4.2	Planification de mouvement	32
2.4.3	Planification de la déformation	36
2.5	Applications	37
2.5.1	Agriculture	37
2.5.2	Sauvetage et premiers soins	38
2.5.3	Aérospatiale	40
2.6	Limites technologiques actuelles	40
Chapitre 3	Conception générale du système, considérations expérimentales et stratégie de planification	44
3.1	Conception du système	45
3.1.1	Scénario et problèmes	45
3.1.2	Séquencement des fonctionnalités requises	47
3.1.3	Structure du système de manipulation	49
3.2	Considérations expérimentales	51
3.2.1	Robot physique	51
3.2.2	Simulation	52
3.3	Stratégie globale de planification	53
3.3.1	Modèle de planification	54
3.3.2	Séquence de cartes et d'algorithmes	56
3.4	Sommaire	59
Chapitre 4	Représentation de l'environnement et cartographie	60
4.1	Choix d'une représentation	61
4.2	Module de cartographie	64
4.2.1	Concepts	64
4.2.2	Conversion de nuages de points	65
4.2.3	Optimisation de la fusion de cartes probabilistes	71
4.2.4	Mécanisme de filtres	88
4.3	Algorithmes auxiliaires	89
4.3.1	Algorithmes multifonctions	89
4.3.2	Remplissage des coquilles incomplètes	92
4.4	Sommaire	97

Chapitre 5	Algorithmes de segmentation en 3D	99
5.1	Segmentation par couleur	100
5.2	<i>Fast Level-Sets</i> 3D pour la segmentation de voxels	101
5.3	Algorithme hybride	108
5.4	Conversion en masque d'objet	110
5.5	Évaluation expérimentale	111
5.5.1	Méthodologie	111
5.5.2	Comparaison de la performance de segmentation	114
5.5.3	Évaluation visuelle des performances	120
5.6	Sommaire	122
Chapitre 6	Sélection du prochain point de vue pour la complétion de forme	123
6.1	Heuristique de prochain point de vue	123
6.1.1	Critère d'intégralité de l'objet	124
6.1.2	Calcul du prochain point de vue	128
6.2	Algorithmes d'observation	135
6.2.1	Avec contraintes de positionnement	136
6.2.2	Sans contraintes de positionnement	140
6.3	Évaluation expérimentale de la sélection du prochain point de vue	142
6.3.1	Simulation	142
6.3.2	Environnement physique	155
6.3.3	Comparaison expérimentale	161
6.4	Sélection d'un point de vue pour le suivi de la manipulation	162
6.5	Sommaire	165
Chapitre 7	Comparaison de formes et sélection de prise	166
7.1	Alignement	167
7.2	Association des voxels	170
7.3	Raffinement de l'association des voxels	174
7.3.1	Filtre de moyennage	175
7.3.2	Ajustement des points fixes	175
7.4	Évaluation de l'association des voxels	176
7.4.1	Résultat de l'association	177
7.4.2	Effet du filtre de moyennage	182
7.4.3	Méthode d'association des graphes	185

7.5	Sélection des points de prise	188
7.5.1	Sélection des maximums globaux	189
7.5.2	Sélection des maximums locaux	190
7.6	Évaluation de la sélection de prise	191
7.7	Sommaire	196
Chapitre 8	Résultats globaux et analyse	198
8.1	Résultats globaux	198
8.1.1	Transformation de l'éponge à la sphère	199
8.1.2	Transformation de la sphère en cœur	203
8.1.3	Redresser un tuyau	206
8.1.4	Courber en « S » un long cylindre droit	211
8.1.5	Résultats numériques	214
8.2	Analyse	216
Chapitre 9	Conclusion	219
9.1	Sommaire	219
9.2	Contributions	220
9.3	Travaux futurs	222
Références		224
Annexe A	Moments 3D	245
Annexe B	Résultats supplémentaires	247
B.1	Sélection du prochain point de vue	247
B.1.1	Observation de la sphère avec des voxels de 3 mm	247
B.1.2	Observation de la longue éponge	249
B.1.3	Note sur la segmentation en simulation	249
B.2	Exemples de séquences d'observation	250
B.3	Résultats globaux	256
B.3.1	Transformation d'une sphère en ovale	256
B.3.2	Pliage de la longue éponge	258
Annexe C	Calcul de la courbure	261
C.1	Mesures de la courbure locale	261
C.2	Évaluation expérimentale	264

Annexe D	Architecture logicielle et implémentation du module de cartographie	269
D.1	Bibliothèque de voxels	269
D.2	Application d’une séquence de filtres	271
D.3	Interface ROS2	275
D.3.1	Dépôt de cartes OpenVDB	275
D.3.2	Noeud d’interface (API)	276
D.3.3	Noeud de maintenance	278
D.3.4	Noeud de visualisation	278
D.3.5	Parallélisme et performance	279

Liste des tableaux

4.1	Caractéristiques de différentes représentations de l'environnement	62
5.1	Résultat des algorithmes de segmentation 3D sur la sphère simulée	115
5.2	Résultat des algorithmes de segmentation 3D sur la sphère réelle	116
5.3	Résultats des algorithmes de segmentation 3D sur la longue éponge (forme remplie, individuellement, 3 mm)	117
5.4	Résultats des algorithmes de segmentation 3D sur la longue éponge (forme remplie, scène composée, 3 mm)	117
5.5	Résultats des algorithmes de segmentation 3D sur la longue éponge (forme remplie, scène composée, 6 mm)	118
5.6	Résultat des algorithmes de segmentation 3D sur la balle (forme remplie, scène composée, 3 mm)	119
5.7	Score IoU des algorithmes de segmentation 3D (scène composée, 6 mm) . .	119
6.1	Nombre de voxels dans les formes de référence	144
6.2	Performance des calculs NBV avec différentes limites d'observations par calcul	146
6.3	Performance des calculs NBV sans division en groupes	152
6.4	Performance des calculs NBV sans suppression des intersections	154
6.5	Performance des calculs NBV comparée avec la méthode de référence	162
7.1	Résultats numériques de comparaison de formes 3D	180
7.2	Temps de calcul pour la comparaison de formes 3D	181
8.1	Résultats numériques pour les déformations	215

Liste des figures

2.1	Exemples d'objets classés selon leur rigidité et élasticité.	20
3.1	Exemple de scénario considéré	45
3.2	Composantes et données principales du problème de manipulation d'objets déformables	49
3.3	Aperçu des composantes clés du système de manipulation d'objets déformables.	50
3.4	Fenêtre du simulateur avec objet et caméra virtuels.	53
3.5	Algorithme de manipulation sans modèle prédictif	55
3.6	Cartes créées à l'étape d'initialisation.	56
3.7	Cartes déterminant la forme de l'objet à manipuler.	57
3.8	Processus de comparaison des formes pour la sélection de prise	58
4.1	Illustration de l'algorithme DDA	67
4.2	Configuration du robot pour tester la mise à jour de la carte d'occupation . .	71
4.3	Évolution des probabilités d'occupation avec la fusion bayésienne (10 mm, $\pm 2\%$)	73
4.4	Évolution des probabilités d'occupation avec la fusion logarithmique (10 mm, $\pm 2\%$)	73
4.5	Évolution des probabilités d'occupation avec la fusion logarithmique tronquée (10 mm, $\pm 2\%$)	74
4.6	Évolution des probabilités d'occupation avec la fusion par somme pondérée (10 mm, $\pm 2\%$)	74
4.7	Évolution des probabilités d'occupation avec la fusion bayésienne après le retrait de l'objet (10 mm, $\pm 2\%$)	75
4.8	Évolution des probabilités d'occupation avec la fusion logarithmique après le retrait de l'objet (10 mm, $\pm 2\%$)	76

4.9	Évolution des probabilités d'occupation avec la fusion logarithmique tronquée après le retrait de l'objet (10 mm, $\pm 2\%$)	76
4.10	Évolution des probabilités d'occupation avec la fusion par somme pondérée après le retrait de l'objet (10 mm, $\pm 2\%$)	77
4.11	Évolution des probabilités d'occupation avec la fusion bayésienne (3 mm, $\pm 5\%$)	78
4.12	Évolution des probabilités d'occupation avec la fusion logarithmique (3 mm, $\pm 5\%$)	78
4.13	Évolution des probabilités d'occupation avec la fusion logarithmique tronquée (3 mm, $\pm 5\%$)	79
4.14	Évolution des probabilités d'occupation avec la fusion par somme pondérée (3 mm, $\pm 5\%$)	79
4.15	Évolution des probabilités d'occupation avec la fusion bayésienne après le retrait de l'objet (3 mm, $\pm 5\%$)	80
4.16	Évolution des probabilités d'occupation avec la fusion logarithmique après le retrait de l'objet (3 mm, $\pm 5\%$)	81
4.17	Évolution des probabilités d'occupation avec la fusion logarithmique tronquée après le retrait de l'objet (3 mm, $\pm 5\%$)	81
4.18	Évolution des probabilités d'occupation avec la fusion par somme pondérée après le retrait de l'objet (3 mm, $\pm 5\%$)	82
4.19	Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,1$	83
4.20	Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,3$	84
4.21	Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,5$	84
4.22	Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,7$	85
4.23	Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,9$	85
4.24	Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,1$	86
4.25	Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,4$	87
4.26	Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,6$	87
4.27	Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,8$	88
4.28	Concept du processus de remplissage	93
4.29	Illustration pas à pas du processus de remplissage.	96
5.1	Illustration des deux méthodes pour l'initialisation de Φ	107
5.2	Tranches de Φ selon la figure 5.1 après une itération de l'algorithme FLS .	107
5.3	Exemples d'objets utilisés pour tester les algorithmes de segmentation 3D .	112
5.4	Exemple de grilles de voxels pour la segmentation 3D	112

5.5	Grille de voxels résultant de mesures simulées sur la sphère	113
5.6	Optimisation des seuils de segmentation	114
5.7	Formes segmentées pour la sphère	120
5.8	Formes segmentées pour la balle dans la scène composée	121
5.9	Résultats de la segmentation 3D d'un objet complexe	121
6.1	Exemple de séquence de points d'observation	124
6.2	Suppression des intersections	126
6.3	État initial pour la suppression des intersections	127
6.4	Objet segmenté et intersections	127
6.5	Résultat de suppression des intersections	128
6.6	Calcul des points d'observation potentiels	129
6.7	Sélection du sens de l'axe principal pour le prochain point de vue	130
6.8	Régions d'incertitude dans une carte d'occupation	134
6.9	Observation d'un objet avec le manipulateur robotique et positionnement du capteur sur l'effecteur	136
6.10	Positionnement de la caméra pour l'observation	138
6.11	Objets virtuels pour tester les méthodes de sélection des points d'observation.	143
6.12	Progression du processus d'observation pour la sphère (voxels de 8 mm) . .	147
6.13	Progression du processus d'observation pour la sphère (voxels de 6 mm) . .	148
6.14	Progression du processus d'observation pour le cylindre (voxels de 8 mm) .	149
6.15	Progression du processus d'observation pour le lapin (voxels de 8 mm) . . .	150
6.16	Évolution du nombre de voxels connus et inconnus par calcul NBV sans division en groupes	153
6.17	Évolution du nombre de voxels connus et inconnus par calcul NBV sans suppression des intersections	154
6.18	Séquence d'observations avec le robot physique.	156
6.19	Trajectoires du robot et représentation de l'environnement correspondant à la figure 6.18.	157
6.20	Exemple d'erreur introduite par l'imprécision de la caméra avec un modèle d'occupation ternaire.	158
6.21	Scénario semblable à la figure 6.20 avec un modèle probabiliste.	159
6.22	Observations par le robot physique avec et sans division en groupes	160
6.23	Positionnement de la caméra pour le suivi de la manipulation.	164

7.1	Formes utilisées pour tester la comparaison de formes 3D	167
7.2	Exemples de formes alignées	169
7.3	Exemple d'association de voxels (2D)	170
7.4	Association de voxels avec graphe bipartite	172
7.5	Comparaison des trois méthodes d'association avec un cas 2D.	174
7.6	Exemple de vecteurs d'association sans et avec le filtre de moyennage	175
7.7	Exemples de vecteurs d'association	177
7.8	Échelle de couleurs pour les distances	182
7.9	Effet du filtre de moyennage à différents rayons (sphère à ovale)	183
7.10	Effet du filtre de moyennage à différents rayons (longue éponge droite à pliée)	184
7.11	Forme de poutre utilisée dans les tests	184
7.12	Effet du filtre de moyennage à différents rayons (poutre droite à courbe) . .	184
7.13	Comparaison des méthodes d'association — tuyau et balle	185
7.14	Comparaison des méthodes d'association — éponge et sphère	186
7.15	Comparaison des critères pour l'association de voxels <i>Level-Sets</i> entre l'éponge et la sphère	187
7.16	Sélection de prise pour courber la poutre (sans alignement)	192
7.17	Sélection de prise pour courber la poutre (avec alignement)	192
7.18	Sélection de prise pour courber le tuyau (sans alignement)	193
7.19	Sélection de prise pour courber le tuyau (avec alignement)	193
7.20	Sélection de prise pour plier la longue éponge (sans alignement)	193
7.21	Sélection de prise pour plier la longue éponge (avec alignement)	193
7.22	Sélection de prise pour déformer la sphère en ovale (sans alignement) . . .	194
7.23	Sélection de prise pour déformer la sphère en ovale (avec alignement) . . .	194
7.24	Sélection de prise pour transformer le cœur en sphère (sans alignement) . .	194
7.25	Sélection de prise pour transformer le cœur en sphère (avec alignement) . .	195
7.26	Sélection de prise pour transformer l'éponge en sphère (sans alignement) .	195
7.27	Sélection de prise pour transformer l'éponge en sphère (avec alignement) .	195
8.1	Scène initiale — Éponge	200
8.2	Points de prise (8 cm) — Éponge à sphère	200
8.3	Résultat de la manipulation	201
8.4	Comparaison des formes initiale et finale avec la forme cible	201
8.5	Résultat de la manipulation avec des prises distantes d'au moins 5 cm . . .	202
8.6	Résultat de la manipulation avec les points de prise du même côté de l'objet	203

8.7	État initial pour la sphère et le cœur	204
8.8	Prises et résultat avec des prises distantes d'au moins 8 cm	205
8.9	Prises et résultat avec des prises distantes d'au moins 4 cm	205
8.10	Prises et résultat avec des prises distantes d'au moins 7 cm	206
8.11	Situation initiale pour le tuyau	207
8.12	Prises et suivi pour le tuyau	208
8.13	Résultats pour le tuyau	209
8.14	Résultats pour le tuyau en optimisant les maximums locaux	210
8.15	Situation initiale (avant la manipulation) pour le cylindre	212
8.16	Résultat du remodelage pour le cylindre	213
A.1	Exemples de direction des axes principaux	246
B.1	Progression du processus d'observation pour la sphère (3 mm)	248
B.2	Progression du processus d'observation pour la longue éponge	249
B.3	Exemples d'artefacts de couleur en simulation.	250
B.4	Séquence d'observations pour la sphère en simulation.	251
B.5	Séquence de points de vue pour observer le cylindre	253
B.6	Séquence de points de vue pour observer le cylindre (suite)	254
B.7	Séquence de points de vue pour observer le cylindre (suite)	255
B.8	Points de prise (5 cm) pour remodeler la sphère en ovale	256
B.9	Résultat de la manipulation pour la sphère et l'ovale	257
B.10	Comparaison des formes initiale et finale avec la forme cible	257
B.11	Situation initiale pour le pliage de la longue éponge	258
B.12	Déformation de la longue éponge avec des prises de 8 cm	259
B.13	Déformation de la longue éponge avec des prises de 5 cm	260
B.14	Comparaison des formes initiale et finale avec la forme cible	260
C.1	Illustration des mesures de courbure locale	262
C.2	Résultat du calcul de courbure (plan, 12 mm)	265
C.3	Résultat du calcul de courbure (plan, 3 mm)	266
C.4	Résultat du calcul de courbure (vecteurs normaux, 12 mm)	267
C.5	Résultat du calcul de courbure (vecteurs normaux, 3 mm)	268
D.1	Artefacts de rotation	271
D.2	Structure de l'interface ROS2 conçue pour le module de cartographie. . . .	275

Liste des algorithmes

4.1	Mise à jour d'une carte avec la valeur d'un nuage de points	66
4.2	Mise à jour d'une carte d'occupation ternaire	68
4.3	Mise à jour d'une carte d'occupation probabiliste	70
4.4	Recherche de voxels	91
4.5	Remplissage	95
4.6	Remplissage par masque 3D	97
5.1	<i>Fast Level-Sets</i> 3D	103
5.2	Initialisation de Φ	106
5.3	Algorithme de segmentation 3D hybride	109
6.1	Suppression des intersections	126
6.2	Calcul du meilleur point de vue pour un groupe de voxels inconnus	131
6.3	Calcul des prochains points de vue	132
6.4	Division en groupes des voxels inconnus	133
6.5	Identification des points de vue non-informatifs	135
6.6	Observation d'un objet avec contraintes de positionnement	139
6.7	Observation d'un objet sans contraintes de positionnement	141
7.1	Ajustement des points fixes	176
7.2	Identification des régions de surface à maximum global	190
C.1	Calcul de la courbure — plan le mieux ajusté	262
C.2	Calcul de la courbure — angle des vecteurs normaux	263
D.1	Mise à jour d'une carte avec filtre	272
D.2	Application d'une séquence de filtres	274

Nomenclature

Variables

M_a	Carte de voxels nommée a
M_a^b	Sous-carte b de la carte M_a
$\emptyset$	Valeur indéfinie ou arrière-plan dans une carte
$\emptyset$	Structure de données (ensemble, liste, etc.) vide

Opérateurs

$\mathcal{B}_M$	Boîte englobante des voxels actifs de M
$\kappa_v(p)$	Conversion des coordonnées du point p en voxels
$\kappa_p(v)$	Conversion des coordonnées du voxel v en unités réelles
$\Lambda(v)$	Valeur du voxel v
$\Gamma(v)$	État d'activité du voxel v ; 0 (inactif) ou 1 (actif)
$\Lambda_M(v), \Gamma_M(v)$	Valeur ou état du voxel v dans la carte M
$\eta_r(v)$	Voisins de rayon r du voxel v
$ x $	Valeur absolue du nombre ou cardinalité de l'ensemble x
$\ x\ $	Norme euclidienne du vecteur x
$\neg$	« NON » logique
$\vee$	« OU » logique
$\wedge$	« ET » logique
$\leftarrow$	Assignment d'une valeur à une variable
$\leftarrow$	Ajout d'un élément à une liste, ensemble, etc.
$\oplus$	
$\leftarrow$	Retrait d'un élément d'une liste, ensemble, etc.
$\ominus$	

Chapitre 1

Introduction

1.1 Motivation et contexte

L’environnement dans lequel les humains évoluent quotidiennement est constitué d’une multitude d’objets aux formes et aux propriétés variées, et une grande partie de ces objets se comportent de manière non rigide lorsqu’ils sont manipulés. Bien que les robots acquièrent de plus en plus d’autonomie pour travailler dans des environnements flexibles, la manipulation générale d’objets non rigides est une tâche qui demeure particulièrement difficile pour eux, en partie à cause de la diversité des propriétés et des comportements de ces objets lorsqu’ils sont manipulés.

Dans le cadre de cette thèse, un objet non rigide ou déformable est défini comme un objet dont la forme varie au cours de la manipulation. Cette définition inclut les cordes, les câbles et les fils métalliques, tous les tissus et matériaux semblables, les feuilles minces et flexibles, le caoutchouc et la mousse, ainsi qu’une variété de matières organiques dont la nourriture, les organes et les tissus vivants. La manipulation intelligente de ces matériaux permettrait l’utilisation de robots à usage général dans l’automatisation de plusieurs tâches délicates ou demandant beaucoup de main-d’œuvre, allant de l’assemblage industriel aux procédures médicales en passant par la manipulation des aliments et les travaux domestiques. Bien que plusieurs tâches des domaines ci-dessus pourraient être accomplies efficacement par le développement d’équipement spécialisé ou par la téléopération — par exemple certains travaux agricoles, les opérations de sauvetage, l’assemblage industriel ou la transformation alimentaire — l’équipement spécialisé devient rapidement inopérant lorsque le contexte change, forçant un redéveloppement à chaque changement. De plus, le développement de

robots intelligents devient particulièrement pertinent dans le contexte des missions spatiales de longue durée. Par exemple, dans le cas du projet « Lunar Gateway », la station spatiale devra opérer de manière autonome la grande majorité du temps puisque seulement 8h de téléopération par semaine sont prévues [1]. Ainsi, des systèmes robotiques autonomes devront prendre en charge l'entretien et la réparation de la station, de même que la supervision des expériences scientifiques. L'entretien autonome de la station spatiale ainsi que l'établissement d'une éventuelle base permanente à la surface de la lune requièrent des robots capables de gérer les objets déformables comme les câbles, les toiles et les structures gonflables, ainsi que d'effectuer les opérations agricoles nécessaires à une présence humaine à long terme [2]. Similairement, les délais de communication à l'échelle spatiale motivent le développement de robots autonomes pouvant accomplir des tâches complexes [3].

Globalement, cette thèse revisite le problème de l'analyse des formes en trois dimensions (3D) dans une perspective innovatrice, soit celle de la représentation, du suivi et de la planification de la déformation des objets non rigides ou déformables. En effet, la question de modéliser des objets volumétriques solides a largement été explorée dans la littérature et de nombreuses représentations et algorithmes de manipulation des données ont déjà été proposés. Les exemples classiques incluent la numérisation d'artéfacts en muséologie virtuelle ou en études architecturales, la création de cartes de navigation pour les robots et véhicules autonomes, la reconnaissance de formes ou d'objets en 3D, ou la manipulation robotique d'objets rigides prédéterminés. Ces représentations reposent en grande majorité sur l'hypothèse simplificatrice que les objets d'intérêt sont composés de matériaux rigides dont les caractéristiques physiques empêchent la déformation de l'objet. Cette hypothèse élimine d'emblée toutes les contraintes dynamiques liées au changement de forme et réduit le problème à la gestion des transformations homogènes résultant du déplacement d'un solide dans l'espace 3D.

En contrepartie, les méthodes de modélisation d'objets tridimensionnels non rigides ont attiré l'intérêt des chercheurs et des informaticiens dans le contexte de la production d'animations réalistes en cinématographie ou dans l'industrie des jeux vidéos, ou encore pour la simulation mécanique dans la conception de produits ou de machines. Dans ce contexte, les programmeurs exploitent diverses représentations pour imposer un comportement désiré à une forme tridimensionnelle. Les exemples classiques incluent la modélisation de personnages, la déformation d'objets courants soumis à des impacts ou des forces extérieures, ou encore le mouvement d'objets naturels tels que des arbres ou des liquides. Dans toutes ces situations, le comportement dynamique de la forme, incluant son déplacement dans l'espace, est entièrement prédéfini et préprogrammé sur la base des besoins de l'application et des lois

fondamentales de la physique. Il en résulte généralement des approximations plus ou moins fidèles et visuellement attrayantes de la déformation des objets, qui dépendent de la définition préalable de la forme et des propriétés mécaniques précises de l’objet simulé pour obtenir un résultat plausible.

En contraste avec ces problèmes classiques, la recherche présentée dans cette thèse étudie la modélisation d’objets tridimensionnels non rigides dans une perspective différente, soit celle de la représentation et du suivi de la forme d’objets dont la forme et la nature des matériaux sont initialement inconnues, et dont la forme évolue pendant leur manipulation par des bras ou des mains robotiques. Les hypothèses sur la rigidité ou sur la connaissance à priori de la réponse dynamique des matériaux sont exclues. L’étude se concentre sur la prise de mesures 3D pour estimer la forme initiale d’un objet solide, pour suivre l’évolution de sa forme pendant la manipulation, et pour gérer la manipulation afin de guider un processus de remodelage de l’objet jusqu’à une nouvelle forme cible désirée en 3D.

1.2 Définition du problème et solutions existantes

Considérons un scénario où un robot se retrouve dans un environnement inconnu comportant de multiples objets, certains étant déformables, et des obstacles. L’utilisateur identifie d’abord un objet non-rigide dont les caractéristiques physiques sont inconnues pour le robot, puis la position et la forme finales désirées pour cet objet. Afin de pouvoir effectuer la transformation et le déplacement, le robot doit être en mesure d’identifier les obstacles dans l’espace de travail, d’observer et de caractériser l’objet d’intérêt et d’identifier les points de prise les plus favorables sur celui-ci pour le déplacer et pour en modifier la forme. Le robot doit également être en mesure d’estimer les caractéristiques physiques de l’objet pour modifier les points de prise et les forces appliquées au besoin. Une panoplie de modèles et de stratégies ont été proposés au fil du temps pour manipuler les objets non rigides. Cependant, ces stratégies sont généralement spécifiques à une tâche particulière et tendent à imposer des contraintes sévères sur le type d’objet manipulé.

Les algorithmes de planification de tâche et de mouvement pour les systèmes robotiques s’intéressent principalement aux cas traditionnels de manipulation d’objets rigides, ou encore à la planification avec des descripteurs abstraits pour les actions et l’état de l’environnement. Dans le contexte des objets non rigides, une difficulté s’impose en raison des changements de forme des objets, qui doivent être inclus dans la description de l’état de l’environnement. Au

niveau de la planification de tâche, les algorithmes existants se penchent habituellement sur l'ordonnancement de tâches et d'actions atomiques telles que « ramasser une boîte » ou « se rendre dans une pièce ». Ainsi, l'état de l'environnement peut être représenté par une collection de faits binaires, accompagnée d'une description assez statique de l'impact des différentes actions sur ces faits. Lorsque ces informations et les contraintes de l'environnement peuvent être encodées par un langage de planification, il existe des algorithmes efficaces pour générer un plan d'action hors-ligne et pour adapter ce plan aux changements dans l'environnement. Ce genre de description de l'environnement est cependant mal adapté aux objets non rigides possédant un nombre infini de degrés de liberté et de configurations possibles. De même, il n'est pas efficace de décrire l'impact des actions sur la forme de l'objet par une mise à jour de faits binaires, en particulier si le modèle de la dynamique de l'objet doit être adapté en cours de manipulation. Ainsi, une description statique de l'environnement et des actions n'est pas suffisante pour permettre la planification de la déformation des objets non rigides. Il est de même difficile d'établir une séquence d'actions en enchaînant les pré- et postconditions des différentes actions. Les méthodes de planification typiquement utilisées dans la littérature étant ainsi mal adaptées à la manipulation d'objets déformables, on se tourne vers une représentation de tâche plus flexible avec un modèle de planification basé sur les actions du robot et sur un prédicteur de l'effet de ces actions sur la forme de l'objet. Cette méthode de planification a deux prérequis, soit une mesure heuristique permettant de guider le planificateur pendant que le prédicteur apprend le comportement de l'objet, et une représentation de l'état de l'environnement qui inclut la forme de l'objet manipulé.

En plus de regrouper les données provenant des capteurs et de permettre la détection de collision pour la planification de mouvement, cette représentation doit aussi supporter l'identification et le suivi de la forme de l'objet manipulé à mesure qu'elle change. Elle demande donc d'être mise à jour en continu tout en permettant d'effectuer efficacement les opérations nécessaires à la segmentation, au suivi et à la planification des changements de forme de l'objet d'intérêt. De même, la représentation choisie doit supporter une opération de comparaison de formes pour déterminer si la tâche de remodelage a été accomplie (c'est-à-dire si la forme de l'objet manipulé correspond à la forme souhaitée) et pour guider la planification vers cet objectif. Le tout doit également demeurer efficace en mémoire et en temps de calcul pour maintenir des délais d'exécution raisonnables. Puisque peu de représentations disponibles répondent à ces critères et que celles qui opèrent dans un environnement partiellement observable sont axées sur l'exploration avec des robots mobiles plutôt que sur la manipulation, il devient nécessaire de développer un système de cartographie répondant à ces besoins. Dans le

contexte robotique, le concept de « cartographie » fait habituellement référence à l’exploration d’un environnement inconnu par des robots mobiles, notamment pour la détection d’obstacles et le calcul de trajets sécuritaires en 2D ou 3D. Il peut aussi inclure la modélisation de grands espaces ou objets comme des forêts, des routes, des monuments ou des immeubles. Dans cette thèse, on étend ce concept à des scénarios de manipulation à grande ou à petite échelle en y ajoutant la mesure et le suivi de la forme de l’objet à manipuler, quelle que soit sa taille.

La recherche présentée s’inscrit dans le contexte d’un projet de grande envergure visant à développer les capacités robotiques de manipulation d’objets déformables. Ce projet à long terme comporte plusieurs facettes de développement dont certaines ont été complétées préalablement à cette thèse, notamment la quantification de l’élasticité à partir de mesures de force et de distance [4], la segmentation simple des objets déformables pleinement observables en contours 2D [5] et leur manipulation multisensorielle [6], de même que le suivi de déformation en 2D et la classification des objets selon leurs caractéristiques élastiques/plastiques [7]. Les travaux de maîtrise précédents [8] se sont basés sur cette capacité de suivi de la déformation pour développer la sélection de prise et le contrôle en-main d’un objet déformable à partir d’une vue 2D. Cette méthodologie de sélection de prise a été adaptée en trois dimensions avec un manipulateur robotique dans [9]. En parallèle et de paire avec la recherche présentée dans cette thèse, un modèle adaptatif permettant de prédire, au cours de la manipulation, le comportement d’objets déformables aux caractéristiques initialement inconnues est développé [10–12]. Dans ce contexte, cette thèse examine la conception générale d’un système de manipulation d’objets déformables et des différents algorithmes requis (chapitre 3) et élabore des stratégies originales de cartographie (chapitre 4), de segmentation (chapitre 5), d’observation (chapitre 6) et de planification (chapitre 7) visant à amalgamer et à coordonner, d’une manière adaptée à ce problème, l’ensemble des ressources et processus nécessaires.

1.3 Objectifs

La vision à long terme à laquelle cette thèse contribue est le développement d’un système robotique de bout en bout capable de planifier et de contrôler, dans un environnement partiellement observable, la déformation d’un objet non rigide dont la forme originale et la dynamique du matériau sont initialement inconnues.

Les objectifs spécifiques suivants permettent de combler les lacunes actuelles et de propulser le développement du système global :

1. Développer un système de cartographie original supportant la manipulation tridimensionnelle d'objets déformables dans un environnement qui n'est que partiellement observable, avec une représentation efficace de l'environnement et de l'objet à manipuler, incluant :
 - (a) Développer une stratégie de segmentation permettant d'identifier l'objet à manipuler parmi les objets environnants.
 - (b) Développer un algorithme de comparaison de formes en 3D axé sur la manipulation d'objets déformables.
 - (c) Établir des mesures heuristiques permettant de guider l'observation et la manipulation.
 - (d) Effectuer le suivi en temps réel de la forme et de la pose de l'objet au cours de la manipulation.
2. Concevoir un système de planification permettant de gérer la manipulation d'un objet non rigide en vue de modifier sa forme. Ceci implique de :
 - (a) Créer un cadre de travail global définissant les composantes du système, leurs responsabilités et leurs interactions.
 - (b) Identifier les régions de l'environnement et de l'objet sur lesquels diriger l'attention du robot, notamment pour identifier les obstacles et la forme de l'objet.
 - (c) Planifier les actions d'observation et de manipulation nécessaires en respectant les contraintes de l'objet et de l'environnement.
 - (d) Prendre en compte l'incertitude dans l'environnement et le modèle de l'objet.

Pour atteindre ces objectifs, une série de propositions originales sont formulées qui concernent la conception du système global, l'acquisition et l'encodage des modèles 3D pour des objets déformables, la segmentation des formes dynamiques en 3D, la sélection automatisée de points de vue pour permettre à des capteurs de profondeur limités au 2.5D de recueillir un maximum d'information sur la forme complète en 3D avant et pendant le processus de remodelage, et la comparaison efficace de formes en 3D pour guider la planification, notamment la sélection des points de prise sur l'objet et le mouvement de ceux-ci.

1.4 Organisation

Le reste de cette thèse est organisé comme suit : le chapitre 2 positionne la recherche présentée dans cette thèse dans le contexte de la littérature récente sur la manipulation d'objets déformables, la cartographie et la planification. Le chapitre 3 présente la conception globale du système de manipulation d'objets déformables proposé dans cette thèse tout en définissant plus précisément le scénario considéré ainsi que les différentes étapes et composantes nécessaires, de même que les considérations pour l'évaluation expérimentale. Il présente également le modèle de planification proposé pour ce scénario et l'orchestration des différentes données et algorithmes détaillés dans les chapitres subséquents. Le chapitre 4 détaille les méthodes de cartographie proposées, notamment le choix d'une représentation de l'environnement, la conception du module de cartographie original, ainsi que des algorithmes multifonctions reliés à la représentation de l'environnement. Le chapitre 5 présente des méthodes originales de segmentation 3D pour les grilles de voxels, adaptées au suivi de la forme d'objets déformables. Le chapitre 6 présente les méthodes proposées pour identifier les points de vue permettant à une caméra mobile d'observer la forme complète de l'objet d'intérêt. Le chapitre 7 détaille les méthodes originales proposées pour faire la comparaison de formes 3D voxélisées ainsi que la mesure heuristique résultante pour guider la sélection et le déplacement des points de prise. Les différentes méthodes proposées sont évaluées expérimentalement dans les chapitres correspondants, puis le chapitre 8 les applique à des scénarios de remodelage d'objets déformables de bout en bout, permettant une évaluation et une analyse globales du système et des méthodes proposées. Enfin, le chapitre 9 conclut cette thèse avec une discussion des contributions originales et des pistes d'expansion future. En plus de ces chapitres principaux, des détails supplémentaires sont fournis en annexe concernant l'analyse des moments 3D dans un environnement voxélisé (annexe A), des cas de test supplémentaires (annexe B), l'utilisation de la courbure locale pour la segmentation (annexe C), ainsi que l'architecture logicielle et l'implémentation du module de cartographie (annexe D).

Chapitre 2

Revue de Littérature

Ce chapitre propose un survol de la littérature pertinente à la recherche présentée, en particulier les avancées récentes reliées à la manipulation robotique d’objets déformables (2.1). On y discute également de la littérature concernant la cartographie et les représentations discrètes d’objets (2.2), les méthodes de segmentation et de comparaison de formes en 3D (2.3), les algorithmes de planification de tâche et de mouvement dans un contexte d’environnement dynamique et de gestion des changements (2.4), de même que les applications potentielles de la recherche présentée (2.5). Il se conclut par une analyse des limites de la littérature actuelle dans le contexte de la manipulation robotique d’objets déformables et de la manière dont elles sont abordées par la recherche présentée dans cette thèse (2.6).

2.1 Manipulation d’objets déformables

La manipulation d’objets déformables ainsi que les différentes techniques et applications reliées ont été étudiées en détail dans [13, 14]. Bien que le domaine ait gagné en popularité depuis la publication de ces articles, leurs conclusions générales demeurent pertinentes. En effet, la vaste majorité des travaux s’appliquent à résoudre une tâche précise avec un type d’objet bien particulier. Les stratégies de manipulation sont développées pour correspondre aux méthodes utilisées pour représenter la forme et l’état de l’objet déformable, lesquelles dépendent en grande partie des contraintes de la tâche effectuée ainsi que du type d’objet déformable qui est manipulé. Une caractéristique essentielle du type d’objet manipulé est sa dimensionnalité, c’est-à-dire si l’objet peut être représenté par un modèle linéaire (cordes, fils, câbles), plan (tissu, cuir, feuilles de métal ou plastique), ou tridimensionnel (éponges, aliments,

organes). Les méthodes d'apprentissage machine, dont la popularité est toujours croissante, permettent de résoudre des scénarios difficiles à modéliser, mais posent tout de même des défis liés au format des données d'entrée, à la spécificité des tâches effectuées, et à l'obtention d'une quantité suffisante de données d'entraînement. Dans une brève revue sur la manipulation d'objets déformables avec plusieurs robots, Herguedas et al. [15] notent de même que différentes catégories de tâches de manipulation semblent évoluer indépendamment et qu'il existe peu de travaux qui combinent les catégories dans et entre les différentes classifications.

2.1.1 Objets linéaires

La manipulation d'objets déformables linéaires tels les cordes, fils et câbles étant essentielle à un grand nombre d'applications, elle est toujours bien présente dans la littérature récente. Ces objets étant simples à modéliser, il est possible de planifier des opérations de mise en forme complexes comme le nouage [16] avec une représentation de la topologie du nœud et de l'effet de différentes actions sur celle-ci. Appliquer des méthodes d'apprentissage profond à cette même tâche [17] permet de mettre en lumière un des défis principaux de la manipulation d'objets déformables : puisque la forme et donc l'état de la corde change constamment pendant la manipulation, il est difficile d'établir manuellement une correspondance entre l'état courant de l'objet et le prochain mouvement à effectuer par le robot.

La forme de l'objet représentant un état de grande dimensionnalité, il est pertinent de chercher à faciliter la planification en obtenant une représentation simplifiée. Par exemple, Sintov et al. [18] pré-calculent un descripteur de forme pour accélérer un planificateur par échantillonnage. Une autre approche est de transférer l'état de l'objet dans un espace latent de faible dimension dans lequel il est plus facile pour un algorithme d'apprentissage de relier les mouvements du robot aux changements d'état. Par exemple, Zhou et al. [19] conçoivent des caractéristiques de forme basées sur des marqueurs et des nuages de points, sur lesquels ils appliquent ensuite des techniques de transformation de la dimension, alors que Qi et al. [20] préfèrent des réseaux autoencodeurs. Ces méthodes introduisent cependant plusieurs limitations importantes sur l'objet manipulé et sa représentation.

Le contrôle de forme sans modèle représente aussi un défi, en particulier lorsqu'on considère des câbles lourds comme dans l'industrie aéronautique [21]. Avec de tels câbles, l'impact de leur dynamique sur la commande du robot est trop important pour être ignoré [22], mais la déformabilité de l'objet augmente la difficulté de modéliser et de prédire cet effet pour

contrôler les perturbations. Les caractéristiques de retour de courbure¹ peuvent aider à capturer le comportement de tels câbles sous l'effet de la gravité lors du contrôle visuel [23]. Si la dynamique du câble est connue, le suivi des performances de la tâche dans le monde réel aide à configurer un environnement de simulation dans un cadre « *sim-to-real-to-sim* » [24], ce qui permet d'obtenir un environnement de simulation à haute fidélité comblant l'écart entre la simulation et les performances dans le monde réel.

Les contacts entre les objets déformables manipulés et l'environnement doivent également être pris en compte, puisqu'ils peuvent faire office de points de contrôle supplémentaires pour manipuler l'objet [25]. C'est aussi une contrainte dans les scénarios de *bin-picking* [26] où les prises doivent être sélectionnées de manière à minimiser l'incertitude de la pose et l'enchevêtrement des objets. Plus directement, sélectionner les points optimaux pour placer des fixations selon la forme qu'on souhaite donner à un câble [27] est une décision importante dans les processus d'assemblage. À l'inverse, planifier les mouvements et opérations permettant de déformer et d'enfiler une corde entre des obstacles présente une augmentation significative de la complexité par rapport à la manipulation dans un environnement libre [28].

Avec leurs multiples applications industrielles, le routage et la mise en forme de câbles sont des opérations communes dans la manipulation d'objets déformables. Les approches basées sur l'apprentissage sont courantes pour gérer les différentes parties de ce processus, avec des stratégies telles que l'intégration d'une simulation différentiable dans la boucle de manipulation [29], l'apprentissage d'un modèle d'objet global [30], ou l'utilisation de l'analyse physique pour combler l'écart entre la simulation et le monde réel [31]. Cette analyse et cette modélisation de la dynamique des objets déformables [32] constituent un défi continu pour les applications qui nécessitent une manipulation en temps réel [33], car l'analyse par éléments finis (FEM) traditionnelle est très coûteuse en termes de calcul, même pour les objets linéaires simples. Ces approches demandent aussi une connaissance préalable des propriétés physiques de l'objet et un réglage fastidieux des paramètres du modèle. D'autres types de modèles sont également étudiés pour éviter les difficultés de la modélisation physique. Par exemple, les relations géométriques abstraites et les interactions entre les points clés de l'objet peuvent être représentées sous forme de graphe et évaluées par apprentissage automatique [34].

Laezza et al. [35] notent que la plupart des méthodes robotiques de manipulation d'objets déformables sont développées et démontrées dans des environnements personnalisés, ce qui rend difficile l'évaluation et la comparaison des algorithmes. Ainsi, ils développent un bac à sable contenant six environnements impliquant des objets linéaires déformables. Ils visent

1. *curvature feedback*

en particulier une intégration facile avec les algorithmes d'apprentissage par renforcement, permettant de les découpler du système robotique et de l'environnement pour une évaluation plus précise.

D'autres applications de la manipulation d'objets linéaires déformables comprennent des tâches de fabrication telles que le soudage de câbles USB [36] et le contrôle de la forme et du trajet d'un câble reliant un robot télécommandé à sa station de base ou à sa source d'énergie, comme l'envisage Moreira [37] pour un sous-marin. Il existe un travail similaire par Sanchez et al. [38] sur la manipulation d'outils attachés. Une autre tâche intéressante pour la manipulation d'objets linéaires est la suture automatisée [39], qui pourrait permettre aux robots de fournir des solutions plus rapides et plus précises dans un cadre de chirurgie minimalement invasive.

2.1.2 Objets plans

Les objets déformables planaires tels que le tissu, le cuir, le papier et d'autres feuilles minces attirent également beaucoup l'attention des chercheurs car ces objets correspondent à une grande variété d'utilisations industrielles et domestiques.

Avec ces objets, les travaux récents sont communément dirigés vers l'accomplissement de tâches spécifiques plutôt que sur le contrôle de forme. Ces tâches complexes avec des objets déformables plans sont bien adaptées aux méthodologies d'apprentissage profond², qui demeurent des approches populaires dans la littérature. On utilise par exemple l'apprentissage par imitation pour lisser un morceau de tissu jusqu'à ce qu'il soit à plat sur son support [40] et pour des manipulations plus complexes [41] visant à modifier la configuration du tissu [42, 43]. Dans ces méthodes, en plus des trajectoires pour amener le tissu dans la configuration cible, le système doit apprendre les effets de la vitesse et de l'accélération du manipulateur, qui peuvent avoir un impact important sur le résultat final en retournant ou en lançant le tissu. L'apprentissage par renforcement peut aussi permettre à un robot de gérer la dynamique complexe d'un objet déformable sans modélisation explicite [44], alors que la parallélisation de l'apprentissage [45] permet d'évaluer différentes stratégies en parallèle. La précision des simulateurs pour les objets déformables tels que les tissus [46] reste cependant un défi qui crée des limitations importantes. En effet, les approches d'apprentissage machine requièrent une grande quantité de données, laborieuses à obtenir dans le monde réel [47]. Elles peuvent cependant être intégrées avec succès aux algorithmes traditionnels pour leur fournir des

2. *deep learning*

capacités améliorées, telles que l'interprétation des données tactiles pour la détection du glissement lors de la saisie de tissus sur une surface plane [48].

La planification de manipulations successives requiert de calculer l'effet de chaque manipulation sur le tissu avant son exécution, ce qui peut être coûteux. Li et al. [49] utilisent une base de données précalculée des états des objets pour augmenter la vitesse de recherche des planificateurs. La simulation de tissu dans le logiciel de modélisation 3D Maya est utilisée pour créer un ensemble de données contenant « toutes » les configurations possibles d'un vêtement lorsqu'il est saisi par différents points (chaque nœud du modèle maillé est utilisé comme point de saisie sous l'effet de la gravité). Cette base de données est ensuite utilisée pour reconnaître la forme actuelle de l'objet en fonction de la correspondance 3D et planifier les opérations suivantes.

Malhan et al. [50] travaillent sur l'amélioration dynamique d'un plan de manipulation pour la pose de feuilles composites sur des moules rigides. Dans ce contexte, lorsque le robot exécute un plan de mouvement précalculé, l'état de la feuille peut être différent de la prédiction, ce qui peut introduire des défauts dans la superposition des couches. C'est l'un des principaux problèmes de la manipulation robotique d'objets déformables. Malhan et al. utilisent un modèle de régression à processus gaussien entraîné en simulation pour corriger automatiquement ces défauts de superposition en ajustant la prise et le plan de mouvement du robot. Ce système est développé dans [51], qui se concentre sur l'automatisation de la sous-tâche de préhension pour le processus de drapage de grandes pièces composites. Contrairement aux petites pièces abordées dans [50], les grandes pièces sont drapées en plusieurs sections, entre lesquelles le robot doit saisir à nouveau la feuille. Malhan et al. utilisent un simulateur et une recherche dans l'espace d'états pour trouver la séquence des emplacements de saisie qui permettent de compléter correctement le processus de drapage. De plus, ils notent que le comportement des feuilles composites peut changer avec le temps, au fur et à mesure que les matériaux vieillissent. Ils développent donc également un système permettant d'ajuster le plan en cas de divergence entre le modèle et le comportement réel.

Aranda et al. [52] utilisent le contrôle de forme basé sur des gabarits³ pour contrôler la forme/projection 2D d'une feuille déformable isométriquement. Ils utilisent des gabarits pour encoder des informations sur la géométrie et le comportement de l'objet, qui sont ensuite appliquées pour trouver la forme 3D de l'objet qui conduira à la projection 2D souhaitée. Un problème clé de la manipulation d'objets déformables est aussi souligné par les auteurs :

3. *template-based shape control*

« la perception requiert le suivi de la forme 3D de l’objet déformable à partir d’images 2D monoculaires, ce qui est un problème sévèrement sous-contraint. Comme les robots rigides ont moins de degrés de liberté qu’un objet déformable, le contrôle de la forme devient sous-actionné. » [52]

La perception et la représentation d’un objet déformable plan sous une forme exploitable par un algorithme est toujours un défi, qui est souvent résolu par l’utilisation de marqueurs placés sur l’objet [53], qui donnent la position d’un point bien précis et permettent de reformuler le pliage d’une feuille comme un problème d’optimisation. D’autres stratégies incluent l’utilisation de la simple boîte englobante de l’objet, par exemple pour faciliter le transport d’un drap par plusieurs robots [54], ou encore l’extraction (en simulation) de points clés sur l’objet pouvant être traités par des réseaux neuronaux à graphe⁴ [55].

Un problème difficile dans le traitement des objets non rigides est de suivre leur forme en constante évolution pendant la manipulation et de gérer les occlusions qui peuvent faire perdre au robot la vue d’une partie de l’objet. Chi et Berenson [56] s’attaquent à ce problème pour les cordes et les tissus avec un suivi de séquence RGBD⁵ basé sur la dérive de points cohérents. Leur système est capable de déduire la forme de l’objet derrière les occlusions, et de reprendre le suivi une fois l’occlusion levée — même si l’objet a été déformé pendant l’occlusion. Cette méthode est affinée dans [57] en s’appuyant sur la dérive contrainte de points cohérents déformables pour traiter les interactions avec les obstacles (par exemple, enrouler une corde autour d’un cylindre) et l’auto-intersection à l’aide d’un modèle de diminution de la rigidité.

2.1.3 Objets tridimensionnels

La généralisation de la manipulation robotique à des tâches impliquant des objets déformables en 3D n’est pas un problème simple, car il y a une croissance exponentielle de la complexité de la modélisation lorsque le nombre de dimensions requises pour représenter un objet augmente. Néanmoins, de telles tâches ont commencé à attirer l’attention ces dernières années en raison de l’évolution des capacités des robots et des applications avancées auxquelles ces manipulations ouvrent la porte. Zhu et al. [58] présentent une vue d’ensemble des défis actuels de la manipulation d’objets déformables en tous genres, de la conception du préhenseur à la planification et au contrôle. En particulier, ils notent la difficulté de la

4. *graph neural networks*

5. *Red-Green-Blue-Depth* : observations de couleur et de profondeur

modélisation et de la simulation (soulignée par le manque de simulateurs disponibles) qui limite l'applicabilité des approches guidées par les données. Une question ouverte provient également de la représentation des configurations des objets déformables, car ils notent le défi de concevoir une représentation qui soit à la fois compacte et précise pour la planification et le contrôle, tout en permettant de traiter la forme toujours changeante de ces objets, même en cas d'occlusion ou de bruit. Ils évoquent également l'importante capacité à établir et à rétablir automatiquement le contact entre l'objet et le robot, ainsi que la nécessité pour le robot d'effectuer un raisonnement sémantique sur les objets afin de déterminer si une configuration donnée de l'objet correspond à l'objectif (c'est-à-dire ce que signifie le fait qu'un objet soit plié, enveloppé, etc.) Dans une perspective industrielle, Makris et al. [59] examinent l'assemblage automatisé d'objets non rigides. Bien que leur définition d'une pièce non rigide soit assez restrictive (« ses déformations sont plus grandes qu'au moins une de ses propres dimensions »), ils notent que peu de résultats de la recherche ont trouvé leur chemin vers les environnements industriels.

Apprentissage machine Une approche courante pour les tâches de manipulation complexes est l'apprentissage automatique, qui prend souvent la forme d'un apprentissage par renforcement ou par démonstration. Yin et al. [60] présentent une revue des travaux récents axés sur l'apprentissage qui couvrent la modélisation et le contrôle des objets déformables. Présentant beaucoup de points communs avec [13] et [14], ils passent en revue les approches de modélisation pour la perception et la simulation, ainsi que pour générer des exemples artificiels pour les approches de contrôle basées sur les données. Ils abordent certains des compromis qui doivent être faits entre la précision et la vitesse, et qui affectent les capacités de modélisation de grandes déformations plutôt que de petites déformations autour de la forme au repos. Ils notent qu'une grande partie du travail actuel sur la manipulation d'objets déformables est spécifique à l'application, et donc que les modèles choisis et les approches de contrôle sont rarement bien généralisés. En ce qui concerne la manipulation, ils notent que la plupart des travaux sont axés sur la prise et le déplacement ou sur des interactions simples (piquer, presser), et que des interactions plus complexes, comme la manipulation en-main, imposent des exigences plus strictes quant aux capacités des mains dextres et aux capteurs intégrés à la main, qui ne sont pas encore largement disponibles. En plus des discussions sur les simulateurs, ils notent que l'exploitation de la dynamique des objets déformables avec des mouvements rapides, comme d'étendre une corde ou une nappe en la lançant, n'a pas encore suscité beaucoup d'intérêt. De plus, ils remarquent que les garanties théoriques d'exactitude ou de stabilité sont difficiles à

obtenir sans hypothèses fortes, et souvent négligées dans la manipulation et le contrôle des objets déformables en raison de la complexité de leur dynamique.

Avec les méthodes basées sur les données, il est pertinent d'intégrer des connaissances d'experts dans le système, en particulier dans des domaines où il y a peu de place pour l'erreur, comme la chirurgie. Dans l'apprentissage par renforcement, ceci peut correspondre à sélectionner les caractéristiques utiles au suivi de l'achèvement de la tâche pendant que le robot apprend les actions optimales à entreprendre [61], ou encore de développer un « module de connaissance de la situation » qui combine les données des capteurs, les informations de simulation, les connaissances d'un expert et la programmation logique pour établir une représentation sémantique de la situation qui informe la planification de la manipulation [62].

Les méthodes d'apprentissage par renforcement et par démonstration étant plutôt spécifiques à une tâche ou à une manipulation précise, une autre approche est d'apprendre une correspondance directe entre les mouvements du robot et les caractéristiques de rétroaction⁶ utilisées. Ceci inclut, par exemple, d'utiliser une approche d'apprentissage en ligne pour faire correspondre des contacts ponctuels avec l'objet et un descripteur de déformation extrait directement d'un nuage de points [63], ou le contrôle de la projection 2D d'un objet telle qu'observée par une caméra fixe [60]. Ces méthodes ont l'avantage de ne pas nécessiter une grande quantité de données d'entraînement préalables ou une connaissance approfondie des caractéristiques de l'objet, deux éléments qui peuvent être difficiles à obtenir avec les objets déformables tridimensionnels. En particulier dans [60], bien que cela réduise la vitesse de convergence du modèle, on peut manipuler à la fois des objets rigides ou déformables puisque les matrices représentant le comportement de l'objet sont initialisées en demandant au robot d'effectuer de petits mouvements aléatoires autour des configurations initiales, puis améliorées à mesure que la manipulation progresse.

Modèles physiques Une approche plus classique consiste à s'appuyer sur des modèles de simulation tels que les modèles par éléments finis (FEM) pour planifier et contrôler la déformation des objets mous. Bien que ces modèles puissent demander beaucoup de calcul pour les objets 3D, ils offrent une grande précision et un grand réalisme si les propriétés mécaniques de l'objet sont connues à l'avance. Les techniques de résolution inverse pour ces simulations ont récemment été améliorées, permettant de calculer les forces à appliquer pour façonner un objet [64]. Arriola-Rios et al. [65] présentent une revue plus détaillée des différentes techniques présentes dans la littérature pour la modélisation d'objets 3D déformables avant le

6. *feedback features*

contrôle robotique. Ils notent en particulier les défis dus aux contraintes en temps réel et à la connaissance imprécise des propriétés de l'objet, ce qui rend les modèles physiquement précis tels que les FEM difficiles à utiliser en pratique. Ils constatent également que les paramètres de nombreux modèles ne sont pas directement liés aux propriétés physiques de l'objet, ce qui, même s'ils créent des résultats visuellement et physiquement plausibles, nécessite un réglage fastidieux des paramètres du modèle pour chaque objet. Afin de fournir des solutions diverses, ils tentent également de séparer les problèmes interdépendants de la représentation des formes et de la modélisation de la dynamique. Ils concluent qu'un défi important reste à relever : développer un système capable d'échanger des informations précises entre les différents modules de modélisation des formes/déformations, d'estimation des paramètres, de simulation, de perception, de planification et de contrôle. La prédiction des effets des actions du robot sur la forme de l'objet et le système robot-objet [66] est une composante essentielle d'un planificateur par recherche et échantillonnage et demeure un défi significatif. Saghour et al. [67] combinent la simulation FEM et l'apprentissage machine pour relier les actions du robot à la déformation de l'objet, la simulation permettant aussi d'estimer la forme de l'objet complet. La simulation est basée sur une connaissance approximative des propriétés mécaniques de l'objet alors que l'apprentissage machine met à jour le modèle en fonction des différences entre le comportement prédit et observé de l'objet pour développer un mécanisme de contrôle par rétroaction visuelle⁷ pour modifier la forme de l'objet à partir de points de prise fixes.

Descripteurs fréquentiels D'autres approches récentes explorent l'utilisation de modèles d'objets basés sur des descripteurs fréquentiels, par exemple la relation entre les mouvements des points saisis et la variation des coefficients de Fourier en considérant un modèle masse-ressort [68]. De tels descripteurs fréquentiels sont intéressants car ils permettent d'isoler le comportement de déformation à basse fréquence de l'objet, qui est résistant aux erreurs de mesure et à d'autres bruits à haute fréquence. L'analyse de ces descripteurs avec un graphe modal [69, 70] permet, une fois que le graphe est adapté à l'objet et aux points de saisie, de prédire le mouvement du robot qui aboutira à une forme donnée. Les coordonnées sphériques utilisées pour dériver la représentation de Fourier dans [68] et l'utilisation d'un graphe basé sur un modèle ellipsoïde dans [70] sont cohérents avec l'utilisation d'un ballon ou d'un bloc de pâte à modeler comme objet manipulé, mais peuvent être plus difficiles à appliquer à des objets de forme complexes.

7. *visual shape servoing*

Estimation des propriétés Étant donné que les approches basées sur les modèles nécessitent une connaissance préalable des propriétés mécaniques d'un objet, un problème important est l'estimation dynamique de ces paramètres pour des objets inconnus. Les stratégies pour résoudre ce problème passent souvent par la comparaison des données réelles avec une simulation, par exemple en combinant les informations de suivi RGBD avec une simulation par éléments finis interactive [71], ou en apprenant la réponse en force de l'objet à partir de quelques contacts à point unique pour ensuite simuler sa déformation lorsqu'il est sondé avec des objets de formes variées [72]. Des réseaux auto-organiseurs basés sur des graphes peuvent aussi être appliqués à la prédiction simultanée de la forme et de la déformation [73]. Cette estimation des propriétés mécaniques de l'objet permet éventuellement de mesurer sa forme en se basant uniquement sur des données tactiles [74], ce qui contribue à réduire les problèmes liés aux occlusions dans la manipulation en 3D avec une rétroaction visuelle [75, 76].

Sélection de prise Un autre sujet d'intérêt pour la manipulation sûre d'objets déformables est la planification et la surveillance de la préhension. Cui et al. [77] utilisent un réseau neuronal convolutionnel 3D profond pour fusionner les données visuelles et tactiles afin de catégoriser la force de préhension. Ils définissent les catégories de force de préhension comme « glissante », « appropriée » ou « excessive » en observant le comportement de l'objet, ce qui rapproche le robot de la capacité naturelle des humains à appliquer la force appropriée. Xu et al. [78] considèrent le problème de la saisie d'objets creux déformables tels que des gobelets en papier. En se basant sur le profil de rigidité de l'objet, ils calculent l'emplacement de la prise de manière à ce que la force de saisie et la déformation de l'objet soient minimales. Cela se traduit par la saisie de l'objet dans les zones où il est plus rigide. Cependant, leur mesure ne semble pas prendre en compte le centre de masse de l'objet, ce qui peut entraîner des prises dangereuses, comme la prise par le bas (où il est plus rigide) d'un gobelet en plastique rempli plutôt que par le haut (ce qui réduirait le risque de renversement). Zaidi et al. [79] présentent un pipeline complet de planification de prise pour assurer la stabilité d'une prise à trois doigts sur des objets déformables en 3D. Partant du principe qu'un triangle équilatéral est la prise la plus stable avec trois doigts, ils développent une stratégie de « pré-prise » qui sélectionne les points de contact en ajustant un tel triangle sur la surface de l'objet, aussi près que possible de son centre de masse. Ensuite, ils ajustent la position de la main de telle sorte que les points de contact sélectionnés soient atteignables compte tenu des contraintes cinématiques de la main. Pour calculer la force de prise, ils simulent de manière itérative un modèle d'interaction entre

la main et l'objet afin de déterminer la force minimale qui empêche le glissement.

Séquencement des points de contact Dans RoboCraft [80], un robot doté d'une pince parallèle personnalisée est entraîné à déformer la pâte à modeler en diverses formes 3D. Les observations visuelles brutes RGBD sont transformées en une représentation basée sur les particules. La dynamique des particules est traitée à l'aide de réseaux neuronaux à graphes pour apprendre le comportement de l'objet après 10 minutes de manipulations exploratoires. Le modèle d'objet résultant est utilisé pour planifier la série de pressions avec la pince parallèle qui déformera la pâte à modeler en une forme cible. Même si la variété des manipulations possibles est limitée par la forme physique de la pince, il est possible de donner à la pâte à modeler des formes 3D relativement complexes, et l'entraînement du modèle dans le monde réel permet d'éviter les problèmes de transfert entre la simulation et le monde réel causés par l'imprécision des simulateurs pour les objets déformables. En termes de perception, le nuage de points obtenu par la caméra RGBD est sous-échantillonné et simplement segmenté par couleur pour former la représentation basée sur les particules. Cette méthode repose sur l'hypothèse qu'un seul objet d'intérêt existe dans l'environnement sans aucun obstacle et que sa couleur est suffisamment distincte. En raison de la structure des nuages de points du monde réel, il n'existe pas de correspondance unique entre les particules de différentes observations. La fonction d'erreur est donc définie comme une combinaison de la distance de déplacement (*Earth-mover's distance*, EMD) et de la distance de Chamfer. L'EMD est calculée en faisant correspondre les particules des différentes images sous la forme d'un problème d'affectation de somme linéaire, puis en sommant les distances euclidiennes entre les paires de particules. La distance de Chamfer, quant à elle, associe chaque particule à son voisin le plus proche dans l'autre ensemble et additionne les distances au carré. La somme pondérée des deux mesures est plus performante que chacune d'entre elles prise isolément. Une tâche semblable de sculpture de pâte à modeler est considérée dans [81], en utilisant un modèle de reconstruction de nuage de points préentraîné doublé d'un réseau transformeur⁸ pour prédire les déformations de l'objet. Ils utilisent ensuite un algorithme d'échantillonnage d'actions basé sur les différences entre le nuage de points de l'objet et le nuage de points de la forme cible pour informer la planification. Ils notent en particulier que le choix de la représentation pour la forme de l'objet (ici le nuage de points) est un élément critique pour sélectionner à la fois le système de perception et le modèle de dynamique de l'objet.

8. *transformer network*

Cuiral-Zueco et al. [82] présentent une méthodologie visant à optimiser le positionnement de plusieurs préhenseurs pour des tâches de contrôle de forme. Tout d’abord, une trajectoire de déformation de référence (séquence de formes d’objets) est calculée en considérant le scénario d’actionnement complet, c’est-à-dire si un manipulateur peut être placé à chaque point de l’objet. Cette trajectoire de référence est ensuite utilisée pour calculer une mesure d’erreur qui définit l’importance de chaque point du contour dans la tâche de contrôle de la forme. Un processus d’optimisation sélectionne ensuite les positions des pinces en fonction de cette mesure et du nombre de pinces disponibles. Les déformations considérées sont celles d’objets planaires artificiels (ils n’utilisent pas de simulateur physique, mais uniquement la représentation mathématique des objets/processus) à l’aide de pinces fixes à point unique. Il n’y a aucune contrainte sur la perception ou le suivi de l’objet. Cependant, ils notent l’importance d’analyser la déformation sur l’axe du temps, car certains points peuvent ne pas sembler importants dans les premiers stades de la déformation, mais le devenir à des stades ultérieurs.

2.1.4 Diversité des propriétés d’objets déformables

Le facteur qui a le plus grand impact sur la manipulation d’objets déformables est sans doute le type d’objet manipulé. Il importe donc de discuter des différentes caractéristiques de ces objets et de leur impact sur la tâche. Typiquement, les objets déformables sont classifiés selon le nombre de dimensions de leur modèle minimal (c-à-d. linéaire, plan ou 3D)[13] ou encore selon leur élasticité [83]. Dans le cadre de la recherche présentée dans cette thèse, il est utile d’ajouter une deuxième mesure à l’axe élastique-plastique, soit la « rigidité » de l’objet (axe rigide-souple). Ceci mène à une classification à deux axes qui est représentée à la figure 2.1.

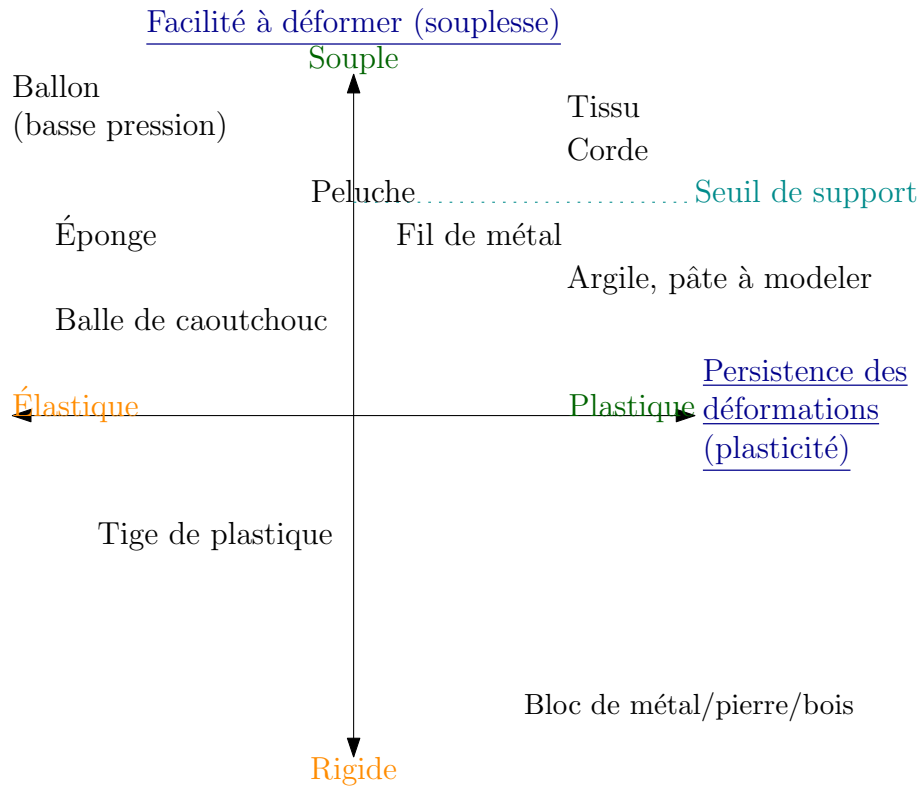


FIGURE 2.1 : Exemples d’objets classés selon leur souplesse et plasticité. On note que les objets dans le coin supérieur droit se déforment aussi sous l’effet de leur propre poids.

Une propriété des objets déformables qui a un grand impact sur les différentes actions de manipulation est l’élasticité. Par exemple, un objet parfaitement élastique comme une éponge reprendra sa forme originale dès qu’il ne sera plus soumis à des contraintes externes. Ainsi, relâcher une prise — par exemple pour changer d’outil ou ajuster la position de la prise — implique de recommencer la manipulation. À l’inverse, un objet plastique comme de la pâte à modeler implique que chaque prise laissera une trace permanente sur l’objet. La plasticité ouvre la porte à des manipulations plus complexes pouvant être composées d’une séquence de prises et d’opérations de roulement et d’étirement pour la mise en forme.

D’un autre côté, certains objets sont non rigides au point de se déformer sous l’influence de leur propre poids. Contrôler et planifier la forme de ces objets implique d’accorder une attention particulière aux surfaces, aux points de support, à la force de gravité et à la tension appliquée perpendiculairement à la déformation. Ces objets présentent souvent des caractéristiques de rigidité directionnelle, comme une corde qui est élastique ou même non-déformable en étirement dans le sens de la longueur mais complètement non rigide perpendiculairement à celle-ci ou en compression. Certaines opérations sur ces objets, telles le nouage et le pliage,

peuvent demander des séquences d’actions complexes et prédéfinies.

En plus de l’élasticité et de la rigidité des objets, une complexité supplémentaire est ajoutée s’il devient nécessaire de considérer les effets de la masse de l’objet manipulé sur le robot [22, 84] ainsi que les mouvements dynamiques rapides (p. ex. fouetter une corde ou lancer un tissu pour l’étendre) [85]. Ces types de manipulations demandent une bonne connaissance de la dynamique de l’objet manipulé.

2.2 Cartographie et représentation de scène

Une fonction clé de tout système robotique autonome est de créer et maintenir une représentation de l’environnement permettant d’extraire les informations nécessaires à l’accomplissement de sa tâche. Dans le cas de la manipulation d’objets déformables, ceci représente un défi supplémentaire puisque la représentation de l’environnement doit inclure la forme changeante de ces objets. Cette section débute donc par l’étude des techniques visant à cartographier un environnement inconnu avec différentes représentations. On accorde ensuite une attention particulière aux environnements voxélisés et aux possibilités qu’ils offrent.

2.2.1 Cartographie des formes et de l’espace

Dans le contexte robotique, la cartographie⁹ implique d’explorer un environnement inconnu, généralement pour différencier l’espace vide de l’espace occupé (obstacles). Cette exploration inclut non seulement la collecte passive des données de capteurs en une carte globale, mais aussi des stratégies de positionnement et d’observation pour révéler un maximum d’espace inconnu tout en accomplissant la tâche souhaitée. Ce domaine ayant une longue histoire dans le contexte de la navigation pour les robots mobiles, on s’intéresse ici particulièrement aux méthodes récentes présentant un intérêt pour la manipulation d’objets déformables. Pour ce faire, on étend la définition habituelle de la cartographie pour correspondre à un scénario de manipulation. En plus de l’identification de l’espace libre et des obstacles, ce concept inclut donc l’observation et la complétion en 3D de la forme de l’objet à manipuler, qu’il s’agisse d’un grand objet manipulé par plusieurs robots ou d’un petit objet avec une manipulation en-main. Dans le contexte du remodelage d’objets non rigides, l’objectif est d’identifier la forme initiale de l’objet d’intérêt et de suivre son évolution tout au long de la manipulation.

9. *mapping*

Cai et al. [86] s'appuient sur une structure de carte hybride et sur le faible taux d'erreur des capteurs LiDAR modernes pour construire un système de cartographie d'occupation avec des performances en temps réel. Ils enregistrent l'espace occupé sur une grille de hachage à accès direct, l'espace inconnu sur une octree pour une efficacité multirésolution, et n'enregistrent pas l'espace inoccupé connu. Pour mettre à jour la carte, au lieu de la méthode traditionnelle consistant à insérer les points mesurés dans la grille et à effectuer une reprojection¹⁰ vers la caméra, ils projettent les cellules de l'octree sur l'image de profondeur générée par la mesure LiDAR et comparent la profondeur minimale et maximale de la cellule à la zone correspondante de l'image. Cela permet d'obtenir un algorithme de mise à jour efficace qui évite l'opération coûteuse de reprojection. En raison de la précision des capteurs LiDAR modernes, ils n'effectuent pas non plus de mises à jour probabilistes et marquent uniquement l'espace comme inconnu ou occupé.

Furrer et al. [87] représentent une scène 3D initialement inconnue comme une base de données de modèles d'objets construits progressivement à partir d'observations segmentées. En supposant que le capteur est bien localisé dans une scène statique, les segments d'objets sont obtenus à partir de l'observation que des objets distincts forment généralement des frontières concaves les uns avec les autres. Les segments sont ensuite fusionnés à l'aide d'un algorithme de regroupement de Markov avec correspondance des points clés et de la géométrie pour récupérer des modèles et des instances d'objets. Ces modèles d'objets sont ensuite utilisés pour compléter les formes observées et vérifier que les parties déduites s'intègrent dans la géométrie de la scène afin d'identifier les instances d'objets.

En appliquant des stratégies modernes d'apprentissage profond, Xu et al. [88] présentent une paire de réseaux neuronaux pour construire une représentation de scène 3D à partir d'images de profondeur et prédire le flux volumétrique de la scène en fonction des actions du robot. En considérant des actions telles que la poussée planaire, le flux de scène est utilisé pour déformer la représentation de la scène entre les étapes en agrégeant les observations successives dans la représentation de la scène. Avec un maximum de 5 objets rigides, leur réseau maintient la permanence de l'objet et peut être utilisé pour manipuler des objets occultés. Alors que l'entraînement nécessite 20h sur quatre GPU, ils atteignent une vitesse d'inférence de 15 images par seconde sur un seul GPU.

Wurm et al. [89] étendent le paradigme traditionnel de la cartographie d'occupation à un scénario de manipulation en adoptant une hiérarchie d'octrees. En d'autres termes, au lieu de regrouper toutes les observations dans une carte monolithique comme c'est normalement

10. *raytracing*

le cas pour la navigation, ils encodent chaque objet de la scène comme son propre octree et stockent les relations sémantiques spatiales entre eux (par exemple, le vase est sur la table). Cela permet de mieux modéliser les objets mobiles et d'utiliser différentes résolutions d'octree pour différents objets, ce qui accroît l'efficacité. Toutefois, cela repose sur l'hypothèse que les mesures du nuage de points peuvent être parfaitement segmentées et que chaque segment peut être attribué à l'objet correct avant d'être intégré dans la carte appropriée, ce qui est difficile à réaliser en pratique.

La couche de voxels spatio-temporels [90] est une composante de cartographie pour le système de navigation NAV2 [91] basé sur ROS [92]. Elle crée une carte d'occupation des voxels qui tire parti d'un algorithme de décomposition des voxels basé sur le temps pour permettre la représentation d'obstacles dynamiques et est basé sur OpenVDB [93]. Toutefois, étant donné qu'il est axé sur la navigation des robots mobiles, les voxels n'emmagasinent qu'une valeur d'occupation et il n'y a pas de support pour la segmentation des objets ou d'autres manipulations directes de la carte. Un autre système de cartographie d'occupation globale basé sur OpenVDB est présenté par Hagmanns et al. [94]. Ils exploitent le modèle VDB et développent des algorithmes d'insertion efficaces pour intégrer les mesures des nuages de points dans la grille de voxels. Comme d'autres, cela permet de surmonter les limites du système le plus communément utilisé, OctoMap [95], en termes de performances d'insertion en temps réel et de volume de carte fixe. Cependant, ce système ne prend pas directement en charge la segmentation des objets, la comparaison et d'autres opérations cartographiques essentielles au suivi des objets déformables, et il n'enregistre pas d'informations sur les voxels au-delà de l'occupation.

Chen et al. [96] développent un système basé sur les particules pour construire une carte d'occupation continue en quatre dimensions dans un environnement dynamique. Ils proposent un schéma de mise à jour efficace basé sur des sous-espaces de voxels et de pyramides pour réduire le coût de calcul de leur solution et ainsi éviter d'avoir recours à un GPU pour initialiser, mettre à jour et rééchantillonner les particules, ce qui permet d'estimer en temps réel l'occupation d'un point arbitraire de la carte sur un processeur embarqué. Cependant, à l'instar d'autres cartes d'occupation basées sur les particules, peu d'informations sont enregistrées sur l'association entre les obstacles physiques et les particules qui les représentent. Il est donc difficile de segmenter et de suivre la forme d'un objet spécifique dans le temps, et la forme d'un obstacle dynamique tend à dégénérer en une masse informe sur un horizon de prédiction court.

Ren et Qureshi [97] utilisent une approche de détection neuronale pour un manipulateur œil-en-main afin de reconstruire un environnement d'armoire encombrée avec le nombre minimal de points de vue cinématiquement réalisables. Le scénario consiste en une armoire sans porte de dimensions arbitraires remplie d'un arrangement inconnu d'objets domestiques provenant de l'ensemble de données YCB [98]. Ils développent deux réseaux neuronaux profonds pour résoudre ce problème : l'un prend en entrée la scène existante et produit le meilleur prochain point de vue sans collision et réalisable sur le plan cinématique, alors que l'autre prédit la couverture de la scène obtenue par un point de vue donné. La combinaison de ces réseaux permet une exploration efficace de la scène sans connaissances préalables. Une fois qu'un objet a une couverture suffisante pour être associé à un modèle connu, il est utilisé pour compléter la scène afin de compenser l'observabilité partielle et de réduire le nombre d'interactions nécessaires. Contrairement à cela, les objets déformables ne peuvent pas être associés à des formes rigides connues. Notre objectif n'est donc pas de maximiser la couverture de la scène, qui n'est pas limitée à une armoire et peut être arbitrairement grande, mais plutôt d'obtenir une représentation complète de la forme d'un objet cible avant et pendant la manipulation. En outre, bien que leur stratégie permette de compléter la scène avec un nombre minimal de points de vue, elle nécessite un temps de calcul relativement long, compris entre une et trois minutes, bien qu'il ne soit pas clair s'il s'agit d'un temps de calcul par vue ou pour l'ensemble de la séquence de dix points de vue.

Une autre technique pour déterminer le prochain point de vue dans un scénario de reconstruction à petite échelle consiste à utiliser directement la densité des observations existantes [99]. En ciblant les régions où les mesures ne sont pas assez denses, ceci permet d'obtenir une bonne couverture des surfaces directement à partir des nuages de points, sans passer par une structure de données intermédiaire. Cette méthode a l'avantage de résulter en de courts mouvements et faibles temps d'observation, mais exige de conserver la totalité des nuages de points capturés et ne permet pas directement de distinguer l'espace libre de l'espace occupé par des obstacles. Une technique semblable consistant à pointer le capteur vers les éléments de surface non observés voisins des éléments observés a aussi été étudiée avec une représentation de fonction de distance signée (TSDF¹¹) encodée par une grille de voxels [100, 101].

Pour gagner en efficacité dans la planification du déplacement d'un observateur mobile, il est pertinent de chercher à planifier une séquence de points de vue plutôt que seulement le point de vue suivant. Si l'étendue de l'objet à observer est connue à l'avance, une approche usuelle est

11. *Truncated Signed Distance Function*

de distribuer les points d'observation sur un sphéroïde centré sur l'objet en question. Dans cet ordre d'idées, Hu et al. [102] utilisent des réseaux neuronaux profonds pour reconstruire une représentation implicite de la surface de l'objet (de sorte que les petites sections manquantes peuvent être reconstruites sans devoir être observées directement) et déterminer d'un seul coup l'ensemble minimum de points d'observation distribués sur la demi-sphère entourant l'objet qui permet de reconstruire celui-ci.

2.2.2 Environnements voxélisés

Le principe des grilles d'occupation 2D et parfois 3D est bien connu et souvent utilisé pour la navigation et la cartographie pour les robots mobiles. Pour élargir le champ d'application des grilles d'occupation 3D à la manipulation d'objets déformables, on peut se tourner vers le concept plus général de la voxélisation, qui fait référence à la division d'un espace continu en unités de volume discrètes appelées voxels. Cette technique a été largement utilisée dans le domaine de l'infographie et a également trouvé quelques utilisations dans la vision par ordinateur en 3D.

Les premières recherches en infographie portaient principalement sur la voxélisation de surfaces polygonales, de maillages et de scènes [103–105]. Ces articles soulignent l'utilité des grilles de voxels et des structures de données complémentaires pour des tâches telles que la simplification et la réparation de modèles, la détermination de la visibilité, les transformations¹² 3D, la visualisation de volumes et la détection de collisions. Un article plus récent sur ce sujet [106] ajoute les tâches de « géométrie solide constructive » à la liste précédente et développe des algorithmes pour tirer parti du parallélisme des données sur les GPU pour une voxélisation rapide des surfaces et des solides.

Dans la communauté de la vision par ordinateur, la voxélisation a trouvé des utilisations récentes dans les réseaux neuronaux convolutionnels (CNN) 3D. La voxélisation est normalement utilisée pour structurer les données de nuages de points (qui ne peuvent pas être traités tels quels par les CNN) afin d'être utilisées par des réseaux qui effectuent des tâches telles que la reconstruction de surface [107] ou la détection [108, 109] et la reconnaissance d'objets [110]. Parmi les principaux défis rencontrés par ces efforts de recherche, citons l'utilisation élevée de la mémoire qui limite le traitement par CNN aux petites grilles de voxels, ainsi que la perte d'informations lorsque plusieurs points sont fusionnés dans un seul voxel.

12. *morphing*

En ce qui concerne la voxélisation des nuages de points, la formule de base consiste à diviser l'espace observé en une grille régulière et à définir les voxels qui contiennent au moins un point. Cette méthode a été utilisée avec succès pour la reconstruction architecturale de façades de bâtiments à partir de nuages de points pris au niveau de la rue [111] et pour des balayages LiDAR aériens visant à séparer les voxels du sol de ceux appartenant à d'autres éléments (p. ex. plantes, édifices) [112]. Parmi les défis à relever, on note la définition de la taille appropriée des voxels pour préserver les détails, ainsi que le traitement des trous dans le nuage de points (c'est-à-dire les voxels qui devraient faire partie de l'objet mais qui ne contiennent pas de point). Garcia et Ottersten [113] présentent un algorithme de voxélisation de surface et de solide qui traite explicitement les données manquantes dans le nuage de points. Étant donné un nuage de points d'un objet (qui ne comprend naturellement que sa surface et peut comporter des points manquants en raison d'occlusions), ils commencent par voxéliser et solidifier les limites de l'objet. Ils remplissent ensuite efficacement le volume de l'objet pour permettre une représentation complète. Ces opérations sont effectuées en combinant des tranches de l'objet dans les trois dimensions afin de respecter la forme naturelle de l'objet.

2.3 Segmentation et comparaison de formes en 3D

Peu importe la représentation de l'environnement, pour effectuer la manipulation d'un objet déformable en contrôlant sa forme, il importe de pouvoir extraire cette forme des données représentant l'environnement, puis de la comparer avec la forme souhaitée. Cette section présente un résumé des méthodes de segmentation et de comparaison de forme pour les objets tridimensionnels.

2.3.1 Segmentation

Dans le contexte de la manipulation robotique, et particulièrement avec les objets déformables (section 2.1), la segmentation de données 3D est habituellement faite au niveau du nuage de points, avant de fusionner les différentes observations dans une représentation globale. Cette segmentation des nuages de points est souvent faite par un simple seuillage sur la couleur ou la profondeur, comme dans [80, 89], quoique d'autres critères tels la détection de bordures concaves aient été proposés [87]. Dans ces méthodes, les parcelles de nuages de points résultant de multiples observations ou de la sur-segmentation doivent ensuite être fusionnées pour former un modèle d'objet complet. Cette sur-segmentation est aussi présente

dans les méthodes basées sur les supervoxels [114, 115] et pour accélérer les méthodes basées sur la division de graphes¹³ [116].

De nombreuses approches récentes sont dirigées vers l’application des techniques d’apprentissage profond à la segmentation de scène, où l’objectif est d’étiqueter chaque objet dans une scène 3D statique et complètement observée. La majorité des approches citées dans [117, 118] travaillent sur des nuages de points représentant des scènes complètes. Bien que les approches par apprentissage profond puissent donner de bons résultats, elles font face à plusieurs défis. La plus notable est le coût de calcul élevé pour le traitement de données tridimensionnelles à haute résolution, ce qui demande du matériel puissant pour s’approcher d’une vitesse de traitement en temps réel. L’entraînement de ces grands modèles pose aussi des difficultés en raison de la rareté des ensembles de données 3D pertinents et de la difficulté de l’étiquetage manuel des données 3D. Plusieurs modèles ont également du mal à être généralisés pour segmenter des objets qui n’étaient pas représentés dans leurs ensembles de données d’entraînement ou lorsque l’échelle est différente, ce qui est particulièrement difficile pour les petits objets.

Xie et al. [119] proposent une approche d’apprentissage profond en deux parties pour obtenir des masques de segmentation 2D précis pour des objets inconnus apparaissant dans une image RGBD. Leur premier réseau utilise les informations de profondeur et raisonne en 2D et 3D pour estimer les centres des objets et calculer des masques de segmentation grossiers. En raison du bruit inhérent aux capteurs de profondeur, ces estimations initiales sont très inégales et irrégulières. Leur second réseau prend en entrée ces masques et l’image RGB pour se focaliser sur les objets individuels, en faisant « glisser » le masque sur les bords des objets. Bien qu’ils aient entraîné leur système sur des scènes virtuelles non photoréalistes, celui-ci obtient d’excellentes performances dans des environnements réels sans entraînement supplémentaire. Ils suggèrent que c’est parce que les données de profondeur se généralisent bien entre la simulation et la réalité, et parce que le raffinement du masque basé sur les régions de surface est une tâche plus facile que la segmentation complète des objets dans les images RGB.

En ce qui concerne les approches plus traditionnelles, une solution basée sur des *Level-Sets* sous supervision humaine a été développée pour la segmentation d’images médicales de tumeurs cérébrales [120]. Les méthodes basées sur la division de graphes [121] permettent de segmenter et de reconstruire des objets tridimensionnels à partir de séquences d’images en se basant sur un modèle de couleur adaptatif et un critère de minimisation d’énergie,

13. *graph cuts*

mais se révèlent coûteuses en temps de calcul. L'utilisation de supervoxels pour regrouper les voisinages de points semblables permet d'accélérer le traitement de ces méthodes [115].

Les avantages des approches traditionnelles et d'apprentissage profond sont combinées dans [122] pour la segmentation d'instances d'objets à partir de nuages de points dans un scénario de manipulation sur table. Le nuage de points est d'abord traité avec des réseaux neuronaux profonds pour extraire des caractéristiques qui sont ensuite utilisées dans un algorithme de regroupement plus traditionnel. Un autre réseau neuronal traduit les groupements de points en masques d'instance pour les objets observés. Dans [116], les méthodes traditionnelles par superpoints et par graphes basées sur les nuages de points sont utilisées comme méthodes primaires, alors que des méthodes par apprentissage profond sont utilisées pour calculer les poids des arêtes du graphe et pour effectuer l'opération de division du graphe.

2.3.2 Comparaison de formes

La comparaison de la forme et de la pose des objets 3D est une exigence de base pour évaluer la qualité du remodelage et de la manipulation. Pour les objets rigides, la plupart des approches consistent à extraire un modèle d'objet d'une base de données d'objets connus [123] pour la reconnaissance d'objets (qui est aujourd'hui principalement réalisée par apprentissage profond) ou pour un traitement ultérieur. Cependant, la manière de calculer la distance entre deux formes 3D varie selon la façon dont la forme est représentée. Des outils et des mesures telles que le plus proche voisin ont été développés pour les nuages de points et les modèles maillés et sont proposés dans divers logiciels [124, 125]. Comme l'ont observé Novotni et Klein [126], la notion humaine de similarité de forme relève davantage de la différence des volumes que de la correspondance des caractéristiques. Après avoir aligné les modèles maillés des objets sur leurs axes principaux et normalisé leurs volumes, ils effectuent essentiellement une opération de « ou exclusif » entre les deux modèles. Ils font cela avec une séquence de coquilles décalées pour chaque objet, en travaillant par tranches 2D, pour calculer des « histogrammes de distance » pour les objets, qui sont utilisés pour construire une mesure de similarité de forme. Ils notent que l'opération de « ou exclusif » doit être calculée plusieurs fois pour trouver le meilleur alignement des objets en raison du manque d'informations sur l'orientation des axes principaux.

Du point de vue de l'infographie, Breen et Whitaker [127] utilisent une approche par *Level-Sets* pour transformer un objet en un autre. Ils commencent par aligner les centres et les axes principaux des objets, puis mettent à l'échelle l'objet source de manière non uniforme le

long de ces axes pour produire un point de départ. Le modèle de *Level-Sets* transforme ensuite la surface en fonction de l'objet cible. Bien que cette méthode produise une transformation de forme fluide et crédible, elle ne tient pas compte des objets réalistes et de leurs propriétés physiques.

En ce qui concerne les objets voxélisés, une approche courante consiste à simplement compter le nombre de voxels qui doivent être déplacés pour que la source corresponde à la cible, ainsi que la distance totale qu'ils doivent parcourir. Adoptant toutes cette approche de base, les méthodes proposées dans [128–130] calculent la distance totale de déplacement de différentes manières. La simple distance euclidienne entre un voxel source (« positif ») et sa cible la plus proche (« négatif ») est utilisée initialement [128], mais il est noté qu'une meilleure solution est atteinte par une correspondance optimale entre les voxels positifs et négatifs. Pour ce faire, on trouve la correspondance parfaite à poids maximum dans un graphe pondéré reliant tous les voxels positifs à tous les voxels négatifs et en appliquant l'algorithme hongrois pour la correspondance optimale dans les graphes bipartites [129]. Dans [130], il est également noté que la compacité de l'objet n'est pas un bon prédicteur de la similarité de forme. On remarque que ces articles considèrent des objets normalisés en translation, rotation et volume (c'est-à-dire contenant le même nombre de voxels). Si la normalisation du volume est utile pour obtenir une idée de la similarité générale des formes, elle ne convient pas si l'objectif est de déformer un objet réel en une autre forme, car dans ce cas, les voxels représenteront des mesures physiques de volume plutôt qu'un niveau abstrait de discrétisation. En outre, les objets déformables peuvent s'étirer et se comprimer, ce qui peut entraîner l'ajout ou la suppression de voxels dans la représentation. De plus, les voxels ne peuvent pas être déplacés trop loin de leur position initiale et doivent rester attachés à leurs voisins. Ainsi, bien que la mesure de similarité de base présentée dans ces articles convienne pour évaluer la qualité du remodelage, elle devra être utilisée avec une « stratégie de déplacement des voxels » différente pour planifier la manipulation.

2.4 Planification de la manipulation

La littérature sur la planification pour la manipulation robotique est abondante et variée. Plutôt que de tenter ici une revue de la longue histoire de ces méthodes, on souhaite plutôt s'attarder à l'impact qu'ont les objets déformables sur ce problème classique. Par définition, les objets déformables apportent un élément dynamique supplémentaire à la manipulation robotique. En particulier pour la manipulation d'objets dont les propriétés de forme et de

matériau sont initialement inconnues, il importe que le système robotique soit capable de réagir aux changements de forme de l'objet et de gérer l'incertitude entourant ceux-ci. Les sections suivantes explorent la littérature récente sur deux aspects de la planification, soit la planification de tâche et la planification de mouvement, qui dénotent respectivement l'agencement des actions et sous-tâches pour accomplir un objectif de haut niveau et la sélection d'un « itinéraire » à travers des états intermédiaires pour compléter un mouvement. Dans cette exploration, une attention particulière est portée à la planification dans un environnement dynamique ou changeant, informant la manière dont cette planification s'applique à la manipulation d'objets déformables.

2.4.1 Planification de tâche

Arora et al. [131] présentent une solide revue de la planification des actions dans une perspective d'ingénierie des connaissances. Ils couvrent les différents langages, modalités et formalismes utilisés pour décrire les actions et leurs effets sur l'environnement. Ils discutent en outre de nombreuses approches d'apprentissage et de la manière dont elles peuvent être utilisées dans des applications de planification de tâches/actions, soulignant l'énorme effort nécessaire pour décrire manuellement les actions dans des environnements complexes.

Cette description de l'environnement et des actions du robot est un problème clé de la planification de tâche, mais qui demeure difficile à appliquer à des objets de forme changeante et de comportement inconnu. Chaque méthode pour décrire ces éléments est accompagnée d'une classe d'algorithmes conçus spécialement pour fonctionner avec celle-ci. Par exemple, Paulius et al. [132] préfèrent une représentation des connaissances basée sur les graphes pour la planification des tâches robotiques. Ils définissent un « réseau fonctionnel orienté objet » qui traite les actions comme une fonction dont les objets (et leurs états) sont les entrées et les sorties. Ils incluent dans ce réseau le niveau de difficulté de chaque tâche, défini comme la probabilité que le robot réussisse à accomplir cette tâche. Cette mesure de difficulté peut être définie pour chaque robot sans modifier le graphe de connaissances sous-jacent et permet de calculer la probabilité de réussite globale de tâches complexes à étapes multiples. Ils développent également un algorithme de planification qui peut s'appuyer sur un assistant humain pour permettre au robot d'accomplir avec succès des tâches difficiles tout en minimisant l'effort requis de la part de l'humain. Wang et al. [133] planifient des tâches de haut niveau dans des environnements non structurés en apprenant à combiner des compétences de bas niveau. Les compétences sont apprises avec des méthodes de processus gaussiens pour dériver les

différents paramètres de chaque compétence (par exemple, le diamètre d'une tasse ou d'un bol pour une tâche de versement), et sont formellement décrites avec leur effet prévu sur l'état de l'environnement et les conditions préalables requises pour que cet effet soit atteint. En adaptant les paramètres des compétences acquises et en les enchaînant, il est possible d'accomplir une tâche arbitraire dans un environnement non structuré.

La planification de tâche pure étant difficile à appliquer à des robots réels, elle est souvent combinée à la planification de mouvement dans les problèmes dits de planification de tâche et de mouvement (TAMP¹⁴). Mansouri et al. [134] notent les défis présentés par la combinaison de la planification de tâches discrètes de haut niveau avec la planification de mouvements continus de bas niveau dans ces problèmes. Comme ils s'attendent à ce qu'il n'existe pas de solution générale, « unique », à ce problème, ils présentent des lignes directrices sous la forme de cinq questions qui devraient être abordées lors de la conception d'un tel système. Il s'agit de trouver des solutions pour (1) diviser le problème en différents niveaux d'abstraction et trouver une solution qui obéit à toutes les contraintes à tous les niveaux d'abstraction ; (2) raisonner conjointement dans des représentations symboliques/discrètes et continues ; (3) intégrer l'apprentissage machine dans les modèles de tâches et d'environnement ; (4) permettre la prise de décision en ligne ; et (5) traiter l'incertitude, en particulier dans la perception du robot. Ils discutent de quelques approches communes pour traiter ces questions, en particulier (1) et (2), et terminent par des exemples de scénarios. Garrett et al. [135] fournissent un examen approfondi des algorithmes de planification de tâches et de mouvements. Bien que le champ d'application soit limité aux scénarios avec des actions déterministes dans un monde de corps rigides entièrement observables, il englobe les scénarios robotiques traditionnels dans de nombreux environnements industriels et domestiques. Les auteurs notent également des questions importantes pour les futures directions de recherche, telles que les environnements impliquant des objets déformables et dynamiques, le traitement de l'incertitude et l'intégration de méthodes basées sur l'apprentissage.

Kaelbling et Lozano-Pérez [136] ont abordé la question de l'incertitude dans la planification en travaillant dans un « espace de confiance ». C'est-à-dire qu'une probabilité est attribuée à chaque état et résultat d'action possibles (par exemple, le robot sait qu'un objet a 50% de probabilité de se trouver dans un certain tiroir. L'ouverture de ce tiroir fera passer cette « confiance » à 0 ou 100%). En raison de l'incertitude des observations de l'environnement, les objectifs sont également spécifiés avec une valeur de probabilité ou de confiance (par exemple, « le robot doit être sûr à 90% que l'objet se trouve à l'emplacement cible »). Ainsi,

14. *Task And Motion Planning*

la planification se fait sur une distribution de probabilité plutôt que sur des états discrets et déterministes. Cependant, ils notent que la construction d'un graphe de plan qui se ramifie sur chaque résultat possible d'une action est prohibitivement coûteuse. Afin d'éviter ce problème, ils construisent une approximation déterministe de la dynamique attendue de l'environnement pour construire un plan non ramifié. Tout en exécutant le plan, ils surveillent l'environnement et replanifient s'il y a des déviations par rapport aux résultats attendus, en mettant à jour le niveau de confiance du robot en conséquence. Plus récemment, Adu-Bredu et al. [137] ont appliqué des concepts similaires pour des problèmes à long horizon dans un environnement de cuisine simulé. En tenant compte des incertitudes dans l'estimation des poses des objets et des positions des articulations du robot, ils formulent le TAMP comme un problème hybride de satisfaction de contraintes et le résolvent par propagation de croyances non paramétriques avec passage de messages.

Kim et al. [138] considèrent les problèmes de planification de tâches et de mouvements géométriques, qui sont une sous-classe de TAMP impliquant le déplacement de plusieurs objets dans des configurations cibles parmi des obstacles mobiles (par exemple, remplir un réfrigérateur). Pour relever les défis que représentent les espaces de recherche hybrides et les contrôles de validité d'état coûteux en calcul, ils combinent la planification traditionnelle avec l'apprentissage machine. En d'autres termes, ils développent un cadre qui apprend des prédicteurs de guidage de recherche. Cela permet de guider efficacement le planificateur vers l'objectif en se basant sur l'expérience passée, tout en conservant la possibilité de corriger toute erreur commise par les prédicteurs appris. Leur graphe de planification se compose de paires d'états et d'actions abstraites et est traversé du début à la fin en se ramifiant d'abord sur les actions abstraites possibles à partir de l'état actuel, puis sur un échantillonnage aléatoire des paramètres continus (par exemple, l'orientation de la prise, la direction du mouvement) de chaque action. La recherche est guidée par la vérification de la faisabilité de chaque action et par une heuristique apprise.

2.4.2 Planification de mouvement

Dans cette section, on s'intéresse principalement à la planification de mouvement dans les contextes où le problème a un grand nombre de degrés de liberté ainsi que dans un environnement dynamique. Bien que les travaux présentés ne soient pas nécessairement applicables directement à la manipulation d'objets déformables, on peut s'inspirer des stratégies développées pour gérer les caractéristiques d'un espace de configuration infini et d'un comportement

parfois difficile à prévoir.

Qureshi et al. [139] ont développé des « réseaux de planification de mouvement », un algorithme de planification de mouvement qui utilise l'apprentissage profond pour planifier rapidement des chemins optimaux. Cet algorithme prend en entrée les données brutes des capteurs décrivant l'environnement (par exemple, un nuage de points) et produit la trajectoire du robot sous la forme d'une série de points dans l'espace d'état. Ce réseau neuronal profond est divisé en deux parties, à savoir un encodeur, qui convertit les données environnementales en une représentation latente, et un planificateur, qui prend la description de l'environnement et les états initial et cible pour générer un « état suivant » qui rapprochera le robot de l'objectif. Ce principe d'encoder l'état de l'environnement et de l'objet à manipulé dans un espace latent est une approche utilisée pour réduire la complexité de la planification et trouver des chemins simples dans un espace de planification très large [140], mais peut réduire l'interprétabilité des états latents pour les tâches connexes à la planification.

La planification pour les systèmes ayant un grand nombre de degrés de liberté tels que la manipulation multi-bras peut rapidement devenir intraitable, car la complexité du calcul augmente de façon exponentielle avec la dimensionnalité du système. Akbari et al. [141] considèrent des scénarios tels que la mise en place de divers objets sur une table avec un robot bi-manuel. Pour minimiser la quantité d'appels coûteux de planification de mouvement, ils s'appuient sur diverses heuristiques pour orienter la recherche vers l'objectif. En particulier, ils utilisent le raisonnement géométrique pour aider à déterminer le placement des objets, les poses de saisie et les trajectoires de mouvement en fonction de la connaissance de la scène et des solutions de cinématique inverse des robots. Ce raisonnement géométrique est intégré dans un planificateur heuristique *FastForward* qui sélectionne un plan géométriquement réalisable. Le planificateur de mouvement n'est appelé pour confirmer le choix qu'une fois l'action sélectionnée.

Au niveau de la planification de mouvement, les algorithmes les plus utilisés sont typiquement basés sur le principe de l'algorithme RRT [142] (*rapidly-exploring random tree*), qui échantillonne l'espace de configuration du robot pour trouver un chemin allant de la configuration initiale à la configuration cible. Bien que l'algorithme original ne soit ni optimal ni exhaustif¹⁵, plusieurs versions améliorées ont été développées au fil du temps, notamment pour utiliser une heuristique qui dirige l'algorithme vers l'objectif ou pour permettre la recherche efficace dans un espace de configuration très large créé par des équipes de robots [143].

15. dans le sens où il n'est pas garanti que l'algorithme trouve une solution même si elle existe

Shome et Bekris [144] travaillent sur des opérations de saisie et placement via des transferts entre plusieurs manipulateurs robotiques. Afin de s'attaquer à la dimensionnalité élevée du problème, ils tirent parti du découplage inhérent entre les différents bras. Un ensemble limité de configurations possibles de prise, de placement et de transfert fait que les contraintes n'affectent qu'un sous-ensemble de bras, ce qui permet au planificateur d'explorer différentes configurations possibles pour les autres bras. Cela permet d'utiliser des heuristiques pour obtenir rapidement des solutions initiales tout en raffinant le mouvement des autres bras pendant les manipulations. Ils basent leur implémentation sur la méthode dRRT¹⁶ et démontrent un fonctionnement en temps réel pour au moins cinq bras manipulateurs de grande taille. L'algorithme dRRT [145] s'attaque à la planification du mouvement pour de multiples robots à degrés de liberté élevés, où il serait excessivement coûteux de tenter une recherche dans l'espace composite de toutes les configurations des robots. Au lieu de cela, dRRT construit des feuilles de route découplées pour chaque robot et effectue la recherche dans une représentation implicite de l'espace composite qui correspond au produit des feuilles de route des robots individuels, réduisant ainsi l'espace de recherche à une taille gérable.

Aiswarya et Chowdhury [146] explorent plus en avant les capacités de l'algorithme RRT à générer des trajectoires dans des environnements dynamiques. Ils testent une variété de scénarios et de configurations dans un environnement 2D facilement visualisable et analysent la forme et l'efficacité des trajectoires obtenues. Ils considèrent en particulier le scénario où les obstacles ne sont pas initialement connus du robot. Un scénario similaire de planification des mouvements du robot dans des environnements dynamiques a également été exploré par Otte et Frazzoli [147] dans le développement de l'algorithme RRTX. En particulier, ils développent une méthode de replanification qui remodèle et répare dynamiquement l'arbre du plan au lieu de l'élaguer et de recommencer. De plus, ils soutiennent que la replanification devrait être effectuée de l'objectif vers le départ, car elle évite de devoir changer la racine de l'arbre à chaque replanification, et qu'elle correspond à l'observation selon laquelle les capteurs des robots ont tendance à détecter les changements près des feuilles de l'arbre, réduisant ainsi le nombre de chemins affectés par un changement. Ils notent en outre que, dans les environnements dynamiques où des obstacles peuvent se déplacer ou apparaître soudainement, des plans de secours et des « modèles de maintien » doivent être stockés à chaque nœud de sorte que le robot dispose d'un chemin sûr à suivre pendant la replanification au cas où le plan actuel serait soudainement invalidé.

16. *discrete Rapidly-exploring Random Tree*

Une approche précédente de replanification avec RRT* a été présentée par Karaman et al. [148]. Ils ont tiré parti de l'idée selon laquelle de nombreux systèmes robotiques ont besoin de plus de temps pour se déplacer que pour planifier un mouvement. Ainsi, ils calculent rapidement un plan initial et en engagent seulement la partie initiale. Pendant que le robot exécute la partie engagée, l'algorithme réenracine l'arbre à la fin de cette section et recherche un chemin plus optimal vers l'objectif. Cette séquence est répétée jusqu'à ce que le robot atteigne la région de l'objectif. Une approche plus récente de l'optimisation des algorithmes RRT* est présentée par Li et al. [149]. Ils commencent par l'algorithme RRT* informé, qui utilise une heuristique elliptique pour diriger la recherche vers le chemin optimal. Cependant, ils notent que si l'heuristique hyper-sphéroïde optimale couvre une trop grande partie de l'espace de recherche, l'algorithme informé n'est pas différent de sa forme de base. Pour résoudre ce problème, ils utilisent une série de tels hyper-sphéroïdes qui glissent le long du chemin optimal actuel. Cela permet de contraindre le processus d'échantillonnage sans aboutir à une région heuristique trop grande.

Dovgopolik et al. [150] modifient l'algorithme RRT* bidirectionnel pour une plus grande efficacité vers l'objectif. L'algorithme RRT* bidirectionnel fait croître un arbre à partir du nœud de départ et un autre à partir du nœud de l'objectif et tente de les connecter. Cependant, les arbres peuvent généralement croître dans n'importe quelle direction, y compris à l'opposé l'un de l'autre. L'algorithme modifié optimise cela en s'assurant que les nouveaux nœuds ne sont ajoutés que s'ils se trouvent entre les deux arbres, ce qui limite le nombre de nœuds inutiles. Cependant, comme certains obstacles ne peuvent être contournés qu'en faisant pousser les arbres l'un en face de l'autre, ils incluent également une vérification des « angles limites ». En d'autres termes, ils font également pousser des branches perpendiculaires aux nœuds de départ et d'objectif dans toutes les directions afin de s'assurer qu'un obstacle entourant l'objectif peut être contourné, même si cela nécessite de faire pousser les arbres dans des directions opposées. Dobson et al. [151] présentent l'algorithme dRRT*, qui améliore dRRT avec des garanties d'optimalité asymptotiques dans un algorithme informé, à tout moment. L'algorithme dRRT traditionnel décrit précédemment n'offrait pas de garanties d'optimalité ni de comportement en tout temps (c'est-à-dire la capacité de trouver rapidement une solution initiale avant de la raffiner au fur et à mesure de la progression du mouvement). Un examen de dRRT et de plus amples détails sur dRRT* sont fournis dans [143], ainsi que des résultats expérimentaux plus complets qui montrent la capacité de l'algorithme à fournir rapidement une solution initiale et à la raffiner en temps réel.

Les algorithmes utilisés dans l’exploration autonome peuvent fournir des indices sur la planification dans des environnements inconnus et partiellement observés. Wang et al. [152] utilisent une méthode d’apprentissage profond basée sur des informations contextuelles et des connaissances préalables pour prédire la distribution de l’occupation d’un espace occlus. Dans les algorithmes traditionnels, les zones inconnues qui ne peuvent pas être observées en raison d’occlusions sont supposées être soit entièrement occupées (pessimiste), soit entièrement libres (optimiste) jusqu’à ce qu’elles puissent être observées. Le réseau de prédiction d’occupation fournit des estimations plus précises de l’état réel de l’espace occlus, permettant une planification plus efficace des trajectoires du robot. Le réseau apprend en utilisant une méthodologie auto-supervisée avec des données non étiquetées et sans connaissance de la vérité absolue en simulant des trajectoires de navigation. Lindqvist et al. [153] développent un algorithme RRT multi-objectifs pour combiner la planification de trajectoire et l’exploration d’un espace inconnu dans un seul planificateur. Basé sur un cadre d’optimisation multi-objectifs, leur planificateur prend en compte le gain d’information, la distance parcourue et les coûts d’actionnement pour choisir une trajectoire vers des objectifs pseudo-aléatoires dans la zone inexplorée. Ils notent que le gain d’information est facilement calculé en comptant le nombre de voxels inconnus qui entreront dans le champ de vision du capteur le long de la trajectoire.

Dans le cas de la manipulation d’objets déformables, les algorithmes basés sur le principe du RRT se heurtent à deux problèmes pour planifier la déformation de l’objet. Premièrement, la forme de l’objet doit être incluse dans la configuration du système. Puisqu’un objet déformable a un espace de configuration pratiquement infini, il est difficile de générer une configuration aléatoire du système robot-objet qui respecterait à la fois les contraintes de l’objet et du robot. La deuxième problématique tient à la difficulté de relier deux états arbitraires de l’objet. En effet, la base de l’algorithme RRT est de générer une configuration aléatoire et de la relier à la configuration la plus proche pour former un arbre. Cependant, puisque les transitions peuvent être à sens unique et que le modèle de l’objet n’est pas complètement connu, il n’est généralement pas possible de déterminer directement le mouvement ou la transition qui permettrait de relier deux états arbitraires. Ces deux problématiques poussent à explorer l’espace d’action du robot plutôt que son espace de configuration.

2.4.3 Planification de la déformation

McConachie et al. [154] notent que l’obtention d’un modèle précis d’un objet déformable est difficile et coûteux en calcul, et que les différents modèles approximatifs présentent tous

différents inconvénients. Ainsi, un système robotique pourrait utiliser différents modèles à des fins différentes. L'un des moyens proposés pour résoudre ce problème consiste à séparer la planification globale du contrôle local. En utilisant la planification globale pour générer le mouvement approximatif de l'objet, il suffit de déplacer le système vers une configuration où un contrôleur local peut alors accomplir la tâche en affinant la forme de l'objet. Cela permet effectivement d'utiliser un modèle d'objet différent pour le planificateur global et le contrôleur local, tout en simplifiant les deux implémentations en évitant la nécessité de simuler entièrement l'objet. Une autre approche qu'ils proposent dans [155, 156] est de sélectionner dynamiquement le modèle d'objet qui est utilisé pour chaque étape de planification et de contrôle. Ils proposent de formuler la tâche comme un problème de bandit à plusieurs bras¹⁷, où chaque bras est associé à un modèle d'objet différent. Dans ce problème classique d'apprentissage par renforcement, le système apprend l'utilité de chaque bras/action en équilibrant l'exploration de nouvelles actions avec l'exploitation de l'option qu'il croit actuellement avoir une utilité maximale. Dans ce cas, le problème est plus difficile car chaque action (génération et exécution d'un plan de mouvement basé sur le modèle sélectionné) modifie l'état du système, ce qui change potentiellement l'utilité (réduction attendue de l'erreur de tâche) de chaque modèle. Ils appliquent cette méthode à des tâches de manipulation de cordes et de tissus interagissant avec des objets statiques dans un environnement de simulation. On note que si le modèle optimal change en fonction de la partie de la tâche considérée, tous les modèles qu'ils essaient sont au moins quelque peu fonctionnels pour chaque partie de la tâche, de sorte que l'utilité de sélectionner le meilleur est limitée.

2.5 Applications

Cette section recense quelques domaines d'application potentiels pour la recherche présentée ainsi que les solutions existantes développées dans ces domaines.

2.5.1 Agriculture

L'agriculture est un secteur économique essentiel qui nécessite beaucoup de travail manuel dans des conditions souvent difficiles et pour lequel la disponibilité de la main-d'oeuvre peut varier. Cela offre des opportunités intéressantes pour l'automatisation robotique. Oliveira et al. [157] présentent une large revue des technologies commerciales et de recherche existantes

17. *multi-armed bandit problem*

pour la robotique agricole et l'agriculture « intelligente », y compris l'utilisation de robots pour la préparation des champs, la plantation, le traitement, la surveillance, la récolte, l'estimation et l'optimisation du rendement.

Dans de nombreuses tâches agricoles et notamment dans la récolte des fruits et légumes, l'interaction avec des objets déformables est double : d'une part, l'objet à saisir est déformable et ne doit pas être endommagé, et d'autre part, le robot doit souvent interagir avec des objets déformables dans l'environnement tels que des branches et des feuilles pour atteindre l'objet cible. En raison de la spécialisation élevée et des volumes de production de l'agriculture moderne, ces problèmes sont souvent gérés au niveau de la conception de dispositifs mécaniques [158, 159], avec un intérêt pour la détection et la segmentation des articles plus difficiles à reconnaître [160, 161]. Les préhenseurs souples sont également étudiés pour leur conformité inhérente et leur toucher délicat, qui sont essentiels lors de la manipulation de produits fragiles de formes et de tailles variées [162].

La récolte et la gestion des arbres fruitiers nécessitent un peu plus d'intelligence dans la gestion de la déformation par le robot, car il peut y avoir plus d'occlusions et d'obstacles déformables dus aux branches et aux autres fruits. Des travaux sont donc consacrés à la modélisation et à la simulation 3D de ces arbres [163], à la planification de la trajectoire pour la cueillette des fruits [164, 165], ainsi qu'au déplacement et à la séparation des fruits cibles des branches ou des fruits occlusifs [166, 167]. La taille et la gestion des arbres fruitiers présentent également un intérêt [168]. Cependant, même avec la capacité de déplacer intelligemment les branches occlusives, beaucoup d'efforts sont encore consacrés à la conception mécanique des manipulateurs robotiques [169] et des pinces pour ces tâches.

Parmi les autres sujets intéressants dans le domaine de l'automatisation agricole, citons le développement d'un robot pour le greffage autonome [170] et la récolte de viande [171–173], qui sont également liés à la manipulation d'objets déformables.

2.5.2 Sauvetage et premiers soins

Une autre application potentielle de la manipulation robotique d'objets déformables, étroitement liée au domaine médical, est le sauvetage de personnes blessées dans des environnements à haut risque tels que les zones de guerre et les catastrophes naturelles ou industrielles. Dans ces scénarios, les robots ont le potentiel de réduire le risque de blessure pour les premiers répondants et d'augmenter leur efficacité, notamment avec des systèmes télérobotiques partiellement autonomes [174, 175]. En considérant l'évacuation des blessés, ces situations

présentent le besoin de manipuler et protéger l'objet déformable particulier qu'est le corps humain, animé ou non. À l'opposé des progrès de la robotique chirurgicale, peu de travaux semblent avoir été réalisés pour les robots de sauvetage autonomes ou semi-autonomes, même s'ils impliquent souvent des soins de santé. La plupart des efforts actuels semblent encore orientés vers la conception mécanique et le fait d'« atteindre la cible » avec le robot, qu'il s'agisse du contrôle de la stabilité de conduite [176], de la gestion de l'énergie [177], ou de la conception générale du véhicule [178, 179]. En particulier, bien que la conception de [178] ait accéléré l'évacuation des blessés lors d'une simulation de formation en situation réelle, elle nécessitait toujours une équipe de secouristes pour placer la victime sur la civière. La conception de [179] est particulièrement intéressante dans la mesure où elle est l'une des rares à inclure des bras sur une civière mobile, permettant potentiellement au robot d'effectuer un sauvetage autonome et des tâches de premiers soins de base.

Outre le déplacement, deux capacités principales sont requises pour un système de sauvetage autonome : placer le patient sur la civière et effectuer les premiers soins de base (par exemple, pour arrêter la perte de sang). Les travaux qui se rapprochent le plus de la mise en place autonome d'un patient sur une civière concernent la « manipulation par le bras entier », où le robot utilise l'ensemble du bras au lieu de seulement l'effecteur, offrant ainsi plus de soutien lors de la manipulation du patient. Cette approche a été étudiée dans des contextes tels que le sauvetage d'une personne en train de se noyer [180] et la création d'un système télérobotique pour aider une infirmière à tenir le patient lors de l'exécution de diverses tâches [181]. Dans le même ordre d'idées, Peiros et al. [182] examinent le repositionnement autonome de personnes inconscientes dans le cadre d'un scénario de recherche et de sauvetage en concevant des contraintes mathématiques sur le problème d'optimisation de la trajectoire afin de garantir le respect des limites biomécaniques du corps humain et d'éviter ainsi les blessures lors du repositionnement. Pour les premiers soins de base, des travaux ont été réalisés sur la planification du mouvement conforme [183] et le contrôle par retour de force [184] pour les tâches de bandage dans le contexte des soins aux personnes âgées.

Wang et Zhu [185] fournissent un examen détaillé et un historique de la manipulation d'objets déformables dans des scénarios de soins de santé (non chirurgicaux). Inspirés par les progrès des modèles d'apprentissage qui réduisent la dépendance aux modèles d'objets, ils s'éloignent de la classification habituelle des objets déformables basée sur la forme [13, 14, 186] et se concentrent plutôt sur les comportements distincts des matériaux (par exemple, les fluides, les textiles, les agrégats, les élastomères, les matériaux poreux), car ils trouvent que cela est plus pertinent pour les différentes tâches effectuées dans les scénarios de soins de santé. Ils

observent que les textiles sont les types d'objets les plus étudiés, tandis que les objets poreux et granuleux n'ont reçu qu'une attention limitée. Ils insistent sur les défis posés par la complexité de l'interaction robotique avec des objets déformables, notamment les grandes quantités de données et les fonctions de récompense difficiles à concevoir requises pour les approches basées sur l'apprentissage.

2.5.3 Aérospatiale

En plus des travaux d'assemblage notés dans la section 2.1, les missions aérospatiales ont plusieurs particularités qui demandent le développement de solutions robotiques flexibles et intelligentes. En effet, les missions complexes et à longue distance dans l'espace demandent une grande autonomie des systèmes robotiques puisque les méthodes traditionnelles nécessitant un certain niveau d'intervention humaine ne sont pas viables dans plusieurs cas [187]. C'est notamment le cas pour le contexte de la robotique intra-véhiculaire dans le projet *Lunar Gateway* [3], où l'on considère une station spatiale en orbite lunaire qui sera sans équipage de neuf à onze mois sur douze, avec des délais et des limites de téléopération importants. Dans ces conditions, de nombreuses tâches d'entretien devront être effectuées par des robots autonomes, notamment sur des objets déformables. De même, la téléopération directe amène son lot de défis en raison des délais de communication et des interfaces complexes [188]. Dans un contexte plus rapproché, les satellites et les manipulateurs spatiaux étant conçus pour être aussi légers que possibles pour réduire les coûts de lancement, ils contiennent généralement des tiges longues et fines et d'autres éléments flexibles. La plupart des tâches d'entretien sont actuellement effectuées par des sorties extra-véhiculaires (EVA) des astronautes en orbite terrestre basse, ce qui ne serait pas possible pour les orbites plus élevées et stimule le besoin de manipulateurs robotiques capables de gérer de telles tâches sans dépendre de la téléopération directe [189].

2.6 Limites technologiques actuelles

Comme on peut le constater par le nombre et la variété des études publiées dans les dernières années, la manipulation robotique d'objets déformables est un domaine en pleine expansion. Il n'y a pas encore de consensus sur la meilleure stratégie à adopter, même pour manipuler les objets les plus simples, mais les besoins d'automatisation de plus en plus nombreux poussent les chercheurs à se tourner vers des objets plus complexes à modéliser. Bien

qu’une bonne partie des travaux récents soient toujours concentrés sur la manipulation des objets linéaires et plans, plusieurs stratégies sont maintenant développées pour les objets tridimensionnels, plus complexes à modéliser. Les méthodologies sont aussi variées que les objets manipulés et les applications proposées, mais on remarque une prépondérance de travaux utilisant l’apprentissage machine sous une forme ou une autre. Bien que ces approches soient intéressantes pour gérer la grande complexité des objets déformables, elles demeurent limitées par leur besoin d’acquisition de grands volumes de données, le manque de simulateurs gérant les objets déformables, et leur spécificité à une tâche ou à un type d’objets. On remarque aussi l’absence d’approches systémiques globales permettant d’appliquer dans des contextes plus généraux les différentes solutions de manipulation robotique d’objets déformables. Une proposition originale de cette thèse consiste donc à articuler une architecture de perception, représentation, planification et contrôle globale pour ce type d’application.

Cartographie et représentation de scène L’exploration d’environnements inconnus et partiellement observables est un domaine bien étudié dans le contexte des robots mobiles, mais moins complet dans celui de la manipulation. Les méthodes traditionnelles d’optimisation du prochain point de vue peuvent être lentes à calculer alors que celles par apprentissage machine présentent les difficultés habituelles liées aux données d’entraînement et à la spécificité de l’environnement dans lequel elles évoluent. Les systèmes de cartographie disponibles permettent la reconstruction d’un environnement, mais offrent peu de support pour la manipulation des objets présents, en particulier ceux dont la forme change au cours de la manipulation. Les implémentations existantes des grilles de voxels pour la robotique visent généralement la cartographie d’occupation pour la navigation de robots mobiles. Ainsi, les données enregistrées pour chaque voxel se limitent habituellement à une valeur d’occupation à trois états (vide, occupé, inconnu), avec peu de support inhérent pour des données plus riches telles que la couleur, la rigidité, l’appartenance à un objet et d’autres données qui seraient nécessaires pour faciliter la segmentation, le suivi et la manipulation d’objets déformables. Elles ne supportent pas non plus l’implémentation des opérations avancées de segmentation, comparaison de formes et prédiction de la déformation, des problématiques auxquelles cette thèse propose des solutions.

Segmentation La segmentation est une étape initiale essentielle à la manipulation d’objets dans un environnement réel non structuré. Cependant, dans le contexte de la manipulation robotique d’objets déformables, celle-ci est souvent faite par un simple seuil de couleur ou

de profondeur appliqué globalement. Les approches traditionnelles utilisées pour les images n'étant pas toujours compatibles avec les données sous forme de nuages de points ou d'images de profondeur, des approches basées sur l'apprentissage machine ont été proposées. Celles-ci demeurent cependant limitées par la disponibilité des données d'entraînement, particulièrement difficiles à obtenir pour les objets déformables, ainsi que par les exigences de mémoire et de temps de calcul du traitement des données 3D. Il est donc pertinent de développer des méthodes robustes et efficaces pour la segmentation de données tridimensionnelles dans le contexte du suivi de forme d'un objet déformable.

Comparaison de formes 3D Au niveau de la comparaison de formes 3D, on note que les méthodes utilisées dépendent fortement de la représentation choisie pour les formes à comparer. Les méthodes existantes, qu'elles soient basées sur des algorithmes traditionnels ou sur l'apprentissage machine, permettent d'obtenir une mesure de similarité entre deux formes et de les aligner, mais offrent peu d'informations sur la manière de transformer une forme en une autre en respectant les contraintes liées à la manipulation d'objets physiques. Le développement d'une mesure plus informative à cet égard est un des sujets étudiés dans cette thèse.

Planification La problématique de modifier la forme d'un objet non rigide s'inscrit à la croisée de la planification de tâche et de mouvement. En effet, il est à la fois nécessaire de planifier une séquence d'opérations de déformation et d'états intermédiaires (pour faire une forme complexe avec un objet plastique) et de déterminer les mouvements à effectuer pour produire la déformation souhaitée. Cependant, la complexité de représentation des objets déformables fait en sorte qu'il est difficile d'appliquer les algorithmes existants pour résoudre l'un ou l'autre des problèmes, et qui plus est lesdits algorithmes sont généralement spécifiques aux situations pour lesquelles ils ont été développés. De plus, il est nécessaire de planifier, en parallèle des opérations de manipulation, le positionnement et le déplacement des capteurs permettant d'identifier et de suivre la forme et les propriétés physiques de l'objet d'intérêt tout au long de la manipulation. La recherche présentée dans cette thèse contribue à résoudre ces difficultés en développant une représentation flexible des objets déformables, soutenant un algorithme de planification prédictive basé sur les actions pour modifier la forme de l'objet. On développe de plus une mesure heuristique pour guider le choix des actions, qui est une composante essentielle d'une planification efficace lorsque l'espace de configurations et d'actions est vaste.

Domaines d'application Bien que les objets déformables soient communs, leur manipulation par des robots autonomes hors d'un laboratoire est encore rare. Plusieurs applications peuvent être desservies par l'utilisation de matériel spécialisé à la tâche, mais il reste plusieurs défis à résoudre pour une application plus universelle qui s'approche de la performance du travail humain. Ces défis sont particulièrement présents dans des environnements inconnus, partiellement observables ou non-structurés comme dans des scénarios de sauvetage, lorsque l'expertise humaine est indisponible comme dans les missions spatiales, et lorsqu'il y a une grande variabilité dans les objets manipulés et les conditions de travail en agriculture et en soins de santé. La résolution de ces défis passe par une plus grande autonomie des systèmes robotiques, qui doivent posséder les capacités nécessaires pour travailler dans des environnements inconnus tout en planifiant et en gérant de manière autonome la déformation des objets manipulés. Le développement de ces capacités demande à son tour le développement de systèmes capables de gérer de bout en bout l'observation, la planification et la manipulation d'objets déformables aux propriétés inconnues.

Globalement, la littérature actuelle sur la manipulation d'objets déformables se heurte à des limites concernant la réutilisabilité des méthodes et des modules logiciel ainsi que leur applicabilité à de nouveaux contextes. Les modèles disponibles pour représenter les objets et l'environnement dans la planification sont soit inadaptés au remodelage d'objets déformables, soit peu généralisables à des objets arbitraires aux propriétés initialement inconnues. On note ainsi le manque d'outils permettant de gérer systématiquement la représentation, la simulation et la planification de la manipulation de manière générale. Le développement de tels outils structurés s'appuyant sur les principes des grilles d'occupation 3D de même que sur la planification de tâche et de mouvement (TAMP), ainsi que leur intégration en un système cohérent, sont au cœur des objectifs de cette thèse. Ces développements sont clés à l'avancement des capacités robotiques de manipulation d'objets non rigides.

Chapitre 3

Conception générale du système, considérations expérimentales et stratégie de planification

Avant d'élaborer sur les méthodes individuelles dans les prochains chapitres, il importe de définir avec plus de précision le problème étudié dans cette thèse, ainsi que d'établir une stratégie globale pour l'approcher. Ainsi, ce chapitre aborde d'abord la conception générale d'un système de manipulation d'objets déformables (section 3.1) en présentant le scénario considéré d'une manière globale et en le décomposant en une série de sous-problèmes à résoudre (3.1.1). Il présente ensuite la séquence de fonctionnalités requises (3.1.2) et propose une approche systémique basée sur des composantes générales et réutilisables permettant d'accomplir les tâches de manipulation souhaitées (3.1.3). Cette partie de la recherche a été publiée dans [190, 191] et est supportée par une architecture logicielle flexible permettant de définir efficacement les opérations et processus nécessaires à la tâche à accomplir.

La section 3.2 présente ensuite la conception et les contraintes liées à l'implémentation des divers éléments du système proposé pour son évaluation expérimentale. Enfin, la section 3.3 se base sur la définition du scénario considéré et sur la conception du système proposé pour construire un modèle de planification et une feuille de route des différentes données et algorithmes qui seront détaillés dans les chapitres subséquents.

3.1 Conception du système

3.1.1 Scénario et problèmes

De manière générale, le problème considéré est une tâche générique de manipulation d'objets déformables, applicable à la fois à des scénarios de manipulation en-main ou à plus grande échelle. Ainsi, dans un environnement initialement inconnu et partiellement observable, un système robotique équipé d'une configuration quelconque de préhenseurs et de capteurs mobiles (p. ex., caméras RGBD) doit modifier la forme et la pose d'un objet d'intérêt dont les propriétés physiques (matériau) sont initialement inconnues pour qu'elles correspondent à une forme et à une pose cibles définies par l'utilisateur (figure 3.1).

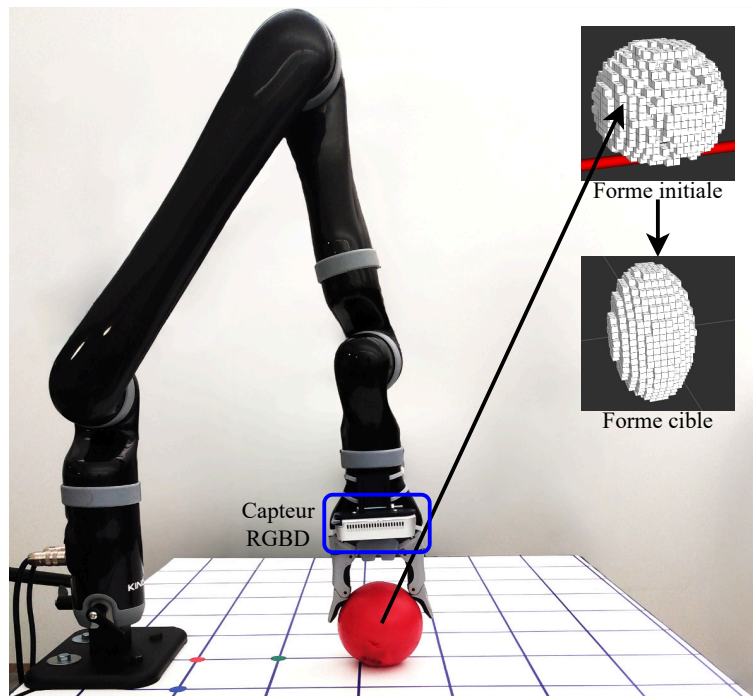


FIGURE 3.1 : Exemple de scénario considéré avec un manipulateur robotique, une caméra RGBD et un objet déformable avec ses formes initiale et cible.

Plus précisément, on peut définir la configuration suivante :

1. L'environnement contient :
 - (a) un certain nombre de manipulateurs robotiques,
 - (b) un certain nombre d'obstacles statiques,
 - (c) un certain nombre de caméras RGBD fixes (peut être zéro),
 - (d) un seul objet d'intérêt, composé d'un matériau non rigide.

2. Au moins un des manipulateurs robotiques est équipé d'une caméra RGBD.
3. Au moins un des manipulateurs robotiques est équipé d'un préhenseur adapté à l'objet d'intérêt.
4. Les propriétés physiques de l'objet d'intérêt (forme, poids, élasticité, rigidité) sont initialement inconnues.
5. L'état de l'objet est défini par sa position, son orientation et sa forme.
6. L'état cible de l'objet est prédéfini par une source externe (ici l'opérateur humain).

Ce scénario place le robot devant un nombre significatif de sous-problèmes qui doivent être résolus pour obtenir un système de manipulation robotique autonome et fonctionnel (généralement abrégé par « le système » dans cette thèse). Les éléments principaux à considérer incluent :

1. la définition de la tâche,
2. la gestion des capteurs visuels et tactiles,
3. la cartographie et la représentation de l'environnement,
4. la comparaison entre la forme de l'objet et la forme cible,
5. l'évaluation et la gestion des besoins (régions à observer, propriétés à identifier, stabilité de prise, etc.),
6. la modélisation de la dynamique de l'objet,
7. la planification de tâche et de mouvement.

La littérature actuelle propose certaines pistes de solution aux différents sous-problèmes présentés, mais au meilleur de nos connaissances aucun système global n'a réussi à les intégrer d'une manière qui répond aux besoins de la manipulation d'objets déformables tridimensionnels. Tout d'abord, la définition de tâche reste un défi significatif puisqu'elle implique de spécifier le processus de changement de la forme d'un état initial à un état final (forme désirée) pour un objet tridimensionnel. Plusieurs algorithmes de contrôle de forme demandent d'identifier des caractéristiques sensorielles précises qui peuvent être complexes à définir manuellement. L'utilisation d'outils de modélisation 3D pour définir la forme finale désirée de l'objet demeure une option simple, mais sous-exploitée et sera la stratégie privilégiée dans cette thèse.

Au niveau des capteurs, la mise en registre et la fusion de données provenant de capteurs visuels et de distance est un sujet déjà bien développé. Cependant, les systèmes de cartographie qui utilisent ces données sont axés sur la détection d'obstacles et la navigation de robots mobiles, le plus souvent dans des environnements fixes avec un mouvement en 2D. L'identification, la

segmentation et le suivi d'un objet manipulé est un point rarement considéré par ces systèmes et représente une contribution originale de cette thèse, en particulier pour des objets non-rigides. Pour la comparaison de formes, l'alignement de données tridimensionnelles est une procédure bien connue et, tel que discuté au chapitre 2, des algorithmes existent pour la comparaison de différentes représentations des volumes. Cependant, le résultat de ces algorithmes est habituellement une vague mesure de similarité, et aucune méthode n'a été proposée qui vise à extraire des informations utiles au guidage de la manipulation d'objets déformables tridimensionnels. L'extraction de ces informations, appelées « guides heuristiques », est une contribution originale significative de cette thèse.

La planification de la manipulation implique de choisir et de séquencer les actions du robot de manière à atteindre l'objectif, c'est-à-dire d'amener la forme de l'objet à correspondre à la forme cible. L'identification de ces actions amène son lot de sous-problèmes, notamment l'observation à partir de points de vue multiples, la complétion de forme, l'identification et la sélection des points de prise, l'assignation des tâches aux différents manipulateurs présents, l'approche et la saisie de l'objet, le contrôle de forme en-main, le maintien de la stabilité de la prise, la coordination des différents manipulateurs et, finalement, la planification des mouvements permettant d'obtenir la déformation désirée. Des solutions existantes à plusieurs de ces sous-problèmes ont été présentées au chapitre 2. Cependant, la planification et le séquençage à long terme de ces différentes actions demande l'utilisation d'un modèle capable de prédire le comportement de l'objet. Cette planification demande aussi d'équilibrer la vitesse de déformation avec le maintien de la stabilité de prise et l'acquisition d'informations sur l'environnement et sur le comportement de l'objet. Enfin, il faut effectuer la planification et le contrôle adaptatif des trajectoires des manipulateurs robotiques pour exécuter les actions planifiées. Cette thèse apporte également des contributions originales à ces aspects, notamment au niveau de l'optimisation du positionnement des capteurs pour la complétion de la forme de l'objet d'intérêt ainsi que de la sélection des points de prise et des mouvements de ceux-ci.

3.1.2 Séquencement des fonctionnalités requises

Les sous-problèmes définis à la section 3.1.1 permettent d'établir une séquence de données et de processus menant à l'accomplissement de la tâche de manipulation d'objet déformable. Cette séquence globale est une extension du cadre originalement proposé dans [190], invitant la conception d'un système de systèmes pour organiser et gérer les multiples composantes et processus qui entrent en jeu dans l'accomplissement de la tâche de manipulation. Les

différentes étapes de cette séquence de données et de processus sont énumérées ci-dessous et présentées graphiquement à la figure 3.2.

1. L'utilisateur définit la tâche en identifiant l'objet à manipuler et la forme qu'on souhaite lui donner (forme cible), dont un exemple est présenté à la figure 3.1.
2. Les capteurs produisent des observations RGBD. Celles-ci sont collectées dans un processus de cartographie pour former une représentation globale et en extraire la forme courante de l'objet à manipuler et l'état de l'environnement.
3. En fonction de l'état de l'environnement, les actions initiales du robot sont choisies pour déplacer les capteurs vers des points de vue optimaux pour compléter la forme initiale de l'objet d'intérêt.
4. À chaque étape de la manipulation, la forme courante de l'objet d'intérêt est comparée avec la forme cible pour :
 - (a) déterminer si la tâche a été accomplie,
 - (b) produire des mesures pour guider la planification du remodelage (guides heuristiques).
5. Parallèlement, l'état de l'environnement provenant des capteurs du robot est analysé pour identifier les priorités du système, dont l'observation des régions incertaines de l'objet ou de l'environnement. Le résultat est une série de mesures et de critères visant à optimiser la planification des actions d'observation et de manipulation effectuées par le robot.
6. En fonctions des mesures, guides et priorités produits par la comparaison de formes et l'évaluation des besoins, un processus de planification sélectionne et séquence les actions à accomplir par le robot pour observer et manipuler l'objet.
7. La séquence d'actions choisie est transmise au module de contrôle du robot, qui gère le déplacement de l'effecteur et des capteurs attachés au robot pour observer et modifier la forme de l'objet.

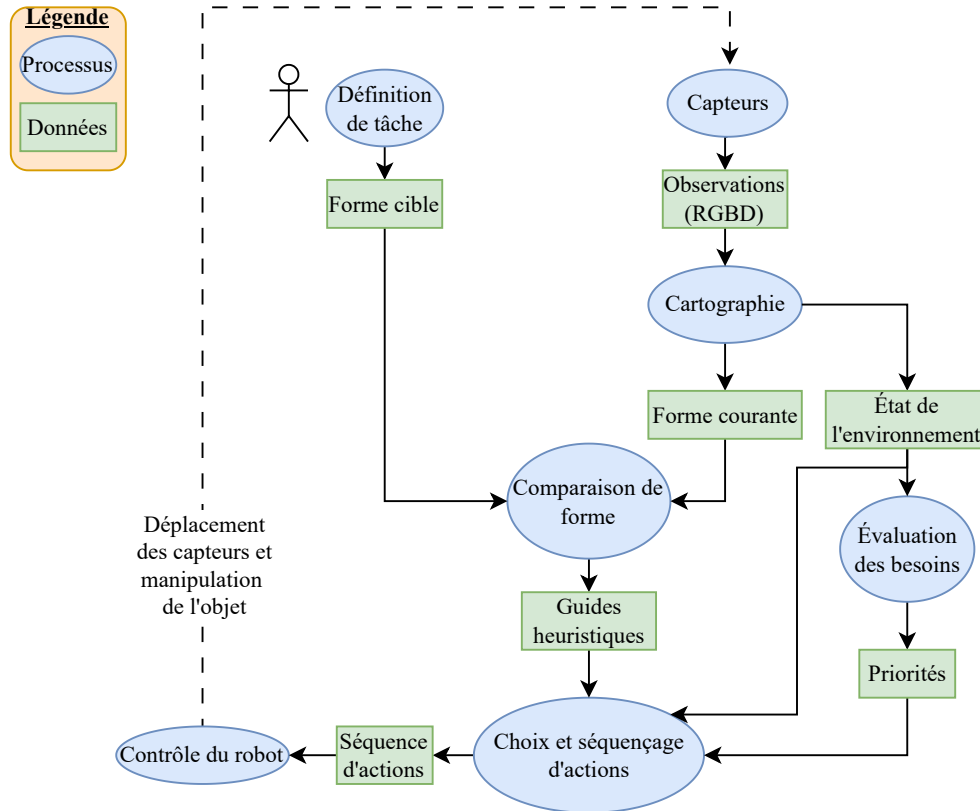


FIGURE 3.2 : Identification et séquençage des composants et données principales du problème de manipulation d'objets déformables.

3.1.3 Structure du système de manipulation

Les processus décrits ci-dessus sont regroupés en un système de manipulation d'objets déformables dont la conception forme une contribution originale de cette thèse. Le système proposé est basé sur l'interaction de cinq composantes clés montrées à la figure 3.3. Un module de cartographie original est développé pour correspondre aux besoins spécifiques de la manipulation d'objets déformables tridimensionnels. Ce module recueille les données provenant des capteurs et les analyse pour en extraire la forme et l'état de l'objet d'intérêt, les obstacles présents dans l'environnement, ainsi qu'une variété de mesures heuristiques liées à la perception et à la manipulation pour informer la planification de tâche. Ces mesures heuristiques sont utilisées par un module de planification de tâche pour générer des objectifs spécifiques à la tâche, lesquels sont ensuite transmis au module de planification de mouvement et à l'environnement robotique. Dans le cadre du scénario considéré pour l'évaluation expérimentale, ce planificateur de tâche a deux rôles principaux. D'abord, il utilise les heuristiques de perception pour sélectionner des points de vue optimaux permettant aux capteurs RGBD

d'observer la forme complète de l'objet d'intérêt et de la suivre durant la manipulation. Ensuite, il utilise les heuristiques de manipulation pour proposer des actions dirigeant la forme et la pose de l'objet d'intérêt vers la forme cible. La planification de mouvement est implémentée à travers le module logiciel MoveIt [192], qui gère la génération de trajectoires sécuritaires permettant aux différents effecteurs de rejoindre les poses spécifiées par le planificateur de tâche. L'environnement robotique, pour sa part, considère à la fois un système physique et la simulation. Le tout est conçu de manière à pouvoir être intégré avec une composante de modélisation et de prédiction dynamique de la déformation [12] développée en parallèle dans notre groupe de recherche mais non intégrée dans cette thèse. L'objectif éventuel de cette composante sera de pré-valider les actions proposées par le planificateur de tâche en prédisant leur effet sur la forme de l'objet d'intérêt. Globalement et en contraste avec les méthodes spécifiques à une tâche précise présentées dans la littérature, cette architecture originale intègre les fonctionnalités nécessaires en un système capable de s'adapter à différentes tâches de manipulation et de remodelage d'objets déformables, notamment dans un environnement 3D partiellement observable lorsque les propriétés physiques de l'objet d'intérêt sont initialement inconnues.

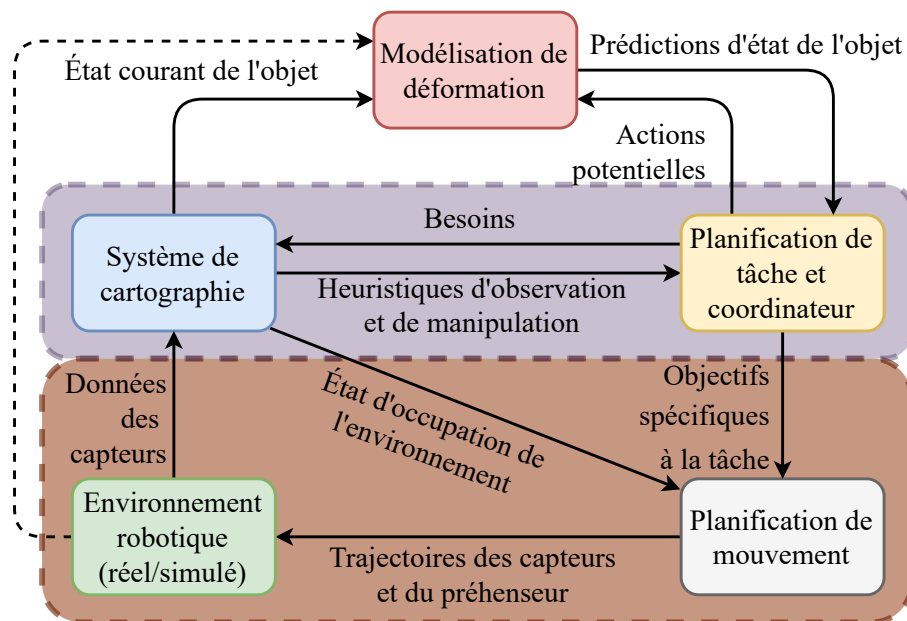


FIGURE 3.3 : Aperçu des composantes clés du système de manipulation d'objets déformables. Les algorithmes originaux proposés dans cette thèse sont concentrés dans l'encadré mauve alors que l'encadré brun indique les composantes développées pour l'évaluation expérimentale.

3.2 Considérations expérimentales

Pour préciser le contexte dans lequel le système conçu à la section précédente évolue, cette section présente les considérations expérimentales et les contraintes définissant l’environnement robotique introduit à la figure 3.3. Ces détails permettront notamment d’adapter les différents éléments du module de planification de tâche aux spécificités de l’environnement robotique et de la tâche précise considérée.

3.2.1 Robot physique

L’évaluation expérimentale est développée en considérant la configuration montrée à la figure 3.1, soit un robot Kinova Jaco de deuxième génération dont la base est montée de manière fixe. Un point de montage est ajouté près de l’effecteur du robot pour supporter une caméra RealSense D435i dans une configuration œil-en-main. Cette configuration permet de déplacer la caméra dans l’espace de travail du robot pour observer l’environnement et l’objet à manipuler sous plusieurs angles, et donc de déterminer sa forme. L’origine du référentiel global est situé au centre de la base du robot à la hauteur de l’anneau gris le plus bas, environ 5 cm au-dessus de la surface de travail.

La structure de l’environnement robotique physique introduit à la section 3.1.3 est réalisée en utilisant un mélange de bibliothèques fournies par le fabricant et tierces parties. Grâce à celles-ci, les trois composantes externes du système — le robot, la caméra et la planification de mouvement — fonctionnent avec un ensemble de nœuds ROS2 [92], lesquels présentent une variété de canaux d’informations et de services permettant de leur transmettre des commandes et de recevoir des données. Le module de cartographie fonctionnant aussi avec un ensemble de nœuds ROS2 (décrit à l’annexe D.3), il revient au module de planification de tâche de coordonner tous ces éléments pour accomplir la manipulation. Ainsi, il indique au module de cartographie quelles cartes créer à partir de quelles données, prépare les contraintes pour la planification de mouvement et les transmet à la bibliothèque MoveIt [192] pour obtenir une séquence de mouvements puis, après avoir validé le plan auprès de l’utilisateur par mesure de sécurité, envoie les commandes requises au robot pour l’exécuter. Des modifications au module de planification de tâche permettent donc d’adapter le système aux particularités de l’environnement et de la tâche à exécuter, comme les contraintes du robot, l’espace de travail, ou d’autres connaissances préalables du scénario.

Précision de la caméra La caméra RealSense D435i est configurée pour produire une carte de profondeur de 848×480 pixels avec un champ de vision de $87^\circ \times 58^\circ$, et peut mesurer la profondeur entre 17 cm et 3 m avec une erreur de $\pm 2\%$ sur la profondeur mesurée à 2 m. Il est possible de calculer la précision dans le plan de l'image (la taille d'un pixel) avec l'équation (3.1), où Z_w , fov et l représentent respectivement la profondeur du point mesuré, le champ de vision et la taille de l'image (en pixels) dans la dimension considérée.

$$\frac{mm}{px} = Z_w \cdot \frac{2 \tan\left(\frac{fov}{2}\right)}{l} \quad (3.1)$$

Étant donné l'espace de travail du robot, une distance de mesure typique à laquelle on peut observer l'objet à manipuler est entre 30 cm et 50 cm. Ainsi, à 30 cm, on a une précision de mesure d'environ 0,68 mm horizontalement et verticalement (dans le plan de l'image) alors que la précision sur la profondeur sera de ± 6 mm. À 50 cm, on trouve 1,12 mm par pixel dans le plan de l'image et ± 10 mm pour la profondeur.

Lorsque les mesures de la caméra sont converties en nuage de points, ces valeurs représentent l'incertitude dans la position des points mesurés. En comparant ceci avec la taille de certains des objets à manipuler, par exemple une sphère de 10 cm de diamètre ou une éponge de $7 \times 4 \times 13$ cm, on note en particulier que la zone d'incertitude de 20 mm créée par l'erreur de profondeur autour de la surface de l'objet peut rendre particulièrement difficile la localisation de la surface de ces petits objets.

3.2.2 Simulation

Étant donnés les défis liés à la précision de la caméra, on se tourne vers la simulation pour évaluer les algorithmes de cartographie, modélisation, et planification proposés dans cette thèse. Des avancées récentes dans la modélisation dynamique accélérée par GPU permettant maintenant de simuler de grandes déformations sur des objets tridimensionnels [193], cette approche nous permet de tester et de valider les algorithmes en contournant les limitations matérielles des capteurs et appareils commerciaux.

L'environnement de simulation est basé sur IsaacSim [194] (figure 3.4) fonctionnant en mode autonome, ce qui permet un contrôle programmatique complet du simulateur, dont notamment le positionnement de la caméra et le déplacement de points de manipulation sur l'objet. Ainsi, l'instance d'IsaacSim est incorporée dans un nœud ROS2 personnalisé, assurant la compatibilité avec la version de ROS2 utilisée par le reste du système ainsi qu'avec

les interfaces développées pour le robot réel. Ceci offre de même une intégration optimale avec le module de planification de tâche/coordonateur, qui peut ainsi facilement contrôler le mouvement des objets et de la caméra, la publication des nuages de points, et le déroulement général de la simulation. Notons que la version utilisée d'IsaacSim supporte uniquement les déformations élastiques. Cette limite a cependant peu d'impact sur les expériences puisque, en l'absence de connaissances préalables sur les propriétés de l'objet, on planifie un seul mouvement sans changement de prise.

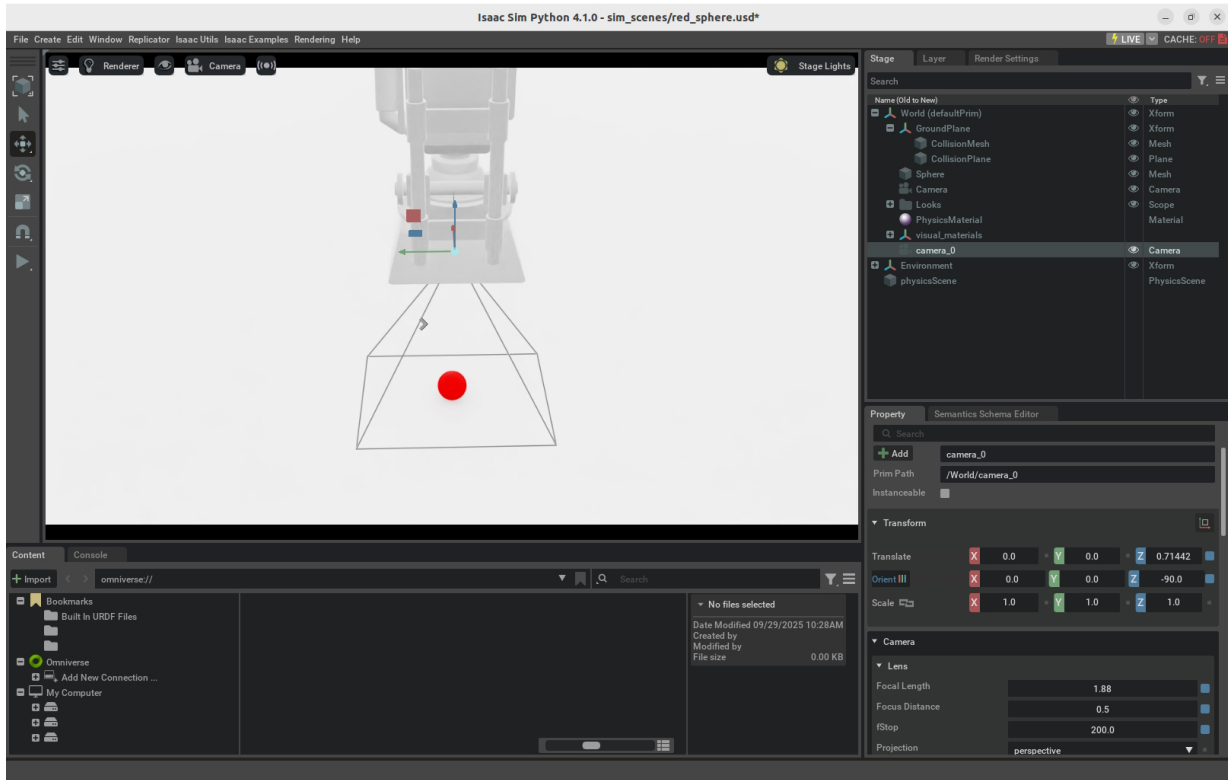


FIGURE 3.4 : Fenêtre du simulateur avec objet et caméra virtuels.

3.3 Stratégie globale de planification

Le module de planification de tâche décrit à la section 3.1.3 fait la coordination entre les différents éléments du système pour orchestrer la manipulation d'objet déformable suivant le cadre développé dans les sections précédentes. Cette section détaille le modèle de planification proposé (3.3.1) ainsi que l'agencement des différentes étapes, cartes et algorithmes (détaillés dans les chapitres subséquents) menant au remodelage d'un objet déformable initialement inconnu (3.3.2). Ces étapes incluent l'initialisation, l'observation de la forme de l'objet d'intérêt, sa comparaison avec la forme cible pour planifier les points de prise, ainsi que le suivi de la manipulation et l'évaluation du résultat du remodelage. L'agencement de ces différents

éléments forme une stratégie globale de perception et de planification pour l’observation et le remodelage, servant de feuille de route pour les composantes développées dans les prochains chapitres.

3.3.1 Modèle de planification

Dans le contexte où le modèle prédictif [12] permettant d’anticiper l’impact des actions du robot sur la forme de l’objet n’est pas encore intégré dans le système décrit et pendant sa période d’adaptation à l’objet manipulé, il est pertinent de se tourner vers un modèle de planification plus local, basé uniquement sur les mesures heuristiques fournies par le module de cartographie. Pour y arriver, un algorithme original de manipulation en trois temps est décrit à la figure 3.5. Ainsi, on commence par explorer l’environnement de travail en déplaçant successivement les capteurs de vision jusqu’à ce que la forme de l’objet à manipuler soit suffisamment observée et précise. L’étape de planification implique ensuite de faire la comparaison entre la forme initiale de l’objet et la forme cible pour déterminer les points de prise et estimer les forces à appliquer sur l’objet. Enfin, l’étape de manipulation proprement dite regroupe trois processus en parallèle, soit le mouvement du bras manipulateur pour déformer l’objet, l’évaluation et le maintien continu de la stabilité de prise, et le suivi de la forme de l’objet par les capteurs visuels mobiles. Pour les expériences précédant l’intégration avec le modèle prédictif, il sera difficile pour le système de s’adapter au niveau d’élasticité/plasticité de l’objet manipulé. Si l’objet est élastique, le relâcher ramènera sa forme au point de départ alors que s’il est plastique, on peut diviser la manipulation en plusieurs prises successives pour améliorer le résultat. Cette décision de modifier la prise ou non est donc laissée à un opérateur humain.

Pour valider le concept proposé, on développe d’abord un prototype en imposant les limites ci-dessous, de manière à permettre une évaluation expérimentale initiale qui sera étendue dans des travaux futurs.

- Un bras « observateur » équipé d’une caméra RGBD peut se déplacer indépendamment du bras manipulateur équipé du préhenseur.
- Les préhenseurs effectuent des prises « rigides » ponctuelles, ce qui fixe le choix de la configuration des prises et élimine le besoin du suivi de contact pour le maintien de la stabilité de la prise.
- On effectue une seule itération de l’algorithme (un seul mouvement des points de prise).

Les différentes étapes de cette manipulation sont détaillées dans les sections suivantes.

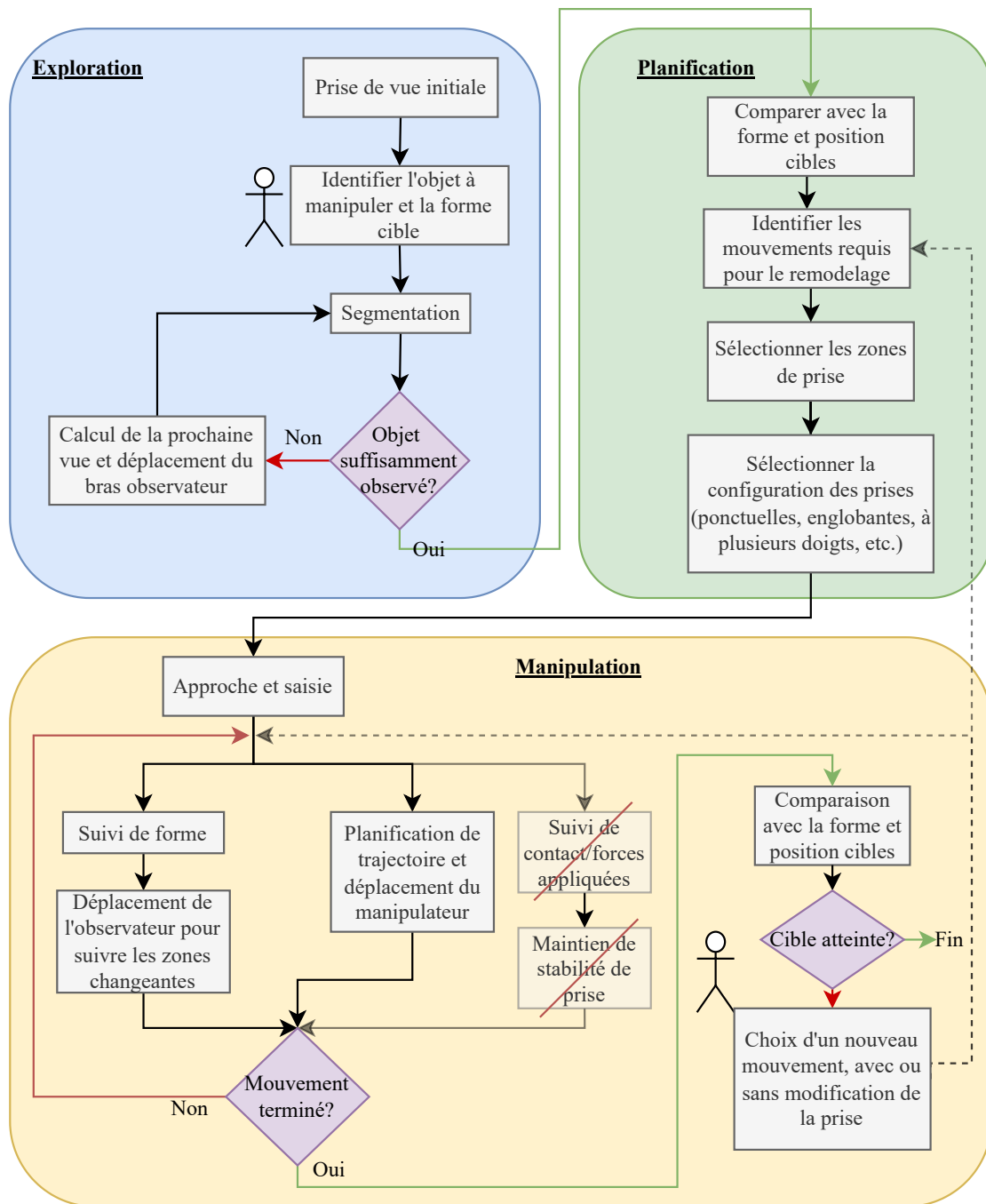


FIGURE 3.5 : Algorithme de manipulation sans modèle prédictif. Les éléments rayés sont exclus de l'évaluation expérimentale.

3.3.2 Séquence de cartes et d'algorithmes

Le scénario de manipulation considéré ainsi que la conception générale du système et du modèle de planification décrits dans les sections précédentes donnent lieu au choix d'une représentation de l'environnement sous forme de cartes voxélisées pour encoder les différentes informations nécessaires. Les détails de ce choix et le formalisme entourant les cartes voxélisées seront discutés au chapitre 4. Cette section présente pour sa part la séquence de cartes et d'algorithmes requis pour accomplir la tâche de remodelage selon le modèle de planification proposé.

Initialisation La séquence d'initialisation consiste à établir la connexion avec l'environnement robotique, qu'il soit réel ou simulé, à créer les cartes de base pour que le système puisse percevoir l'environnement, ainsi qu'à déplacer la caméra dans quelques positions prédéfinies et fixes pour obtenir un aperçu initial de l'espace de travail. Les cartes créées à cette étape sont montrées à la figure 3.6.

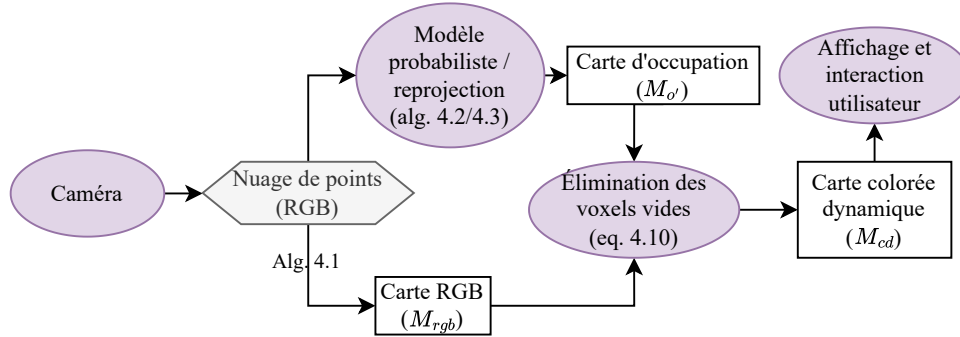


FIGURE 3.6 : Cartes créées à l'étape d'initialisation.

Ainsi, le nuage de points provenant de la caméra donne lieu à deux cartes selon les conversions décrites à la section 4.2.2. Tout d'abord, on définit la carte RGB, M_{rgb} , qui enregistre la couleur (au format rouge-vert-bleu) des points mesurés selon l'algorithme 4.1. De même, on définit la carte d'occupation $M_{o'}$, qui enregistre la probabilité que chaque voxel soit occupé. Tel que discuté à la section 4.2.2, $M_{o'}$ peut encoder cette valeur de manière ternaire (valeurs dans $\{0, 1, \emptyset\}$, alg. 4.2) ou probabiliste (valeurs dans $\mathbb{R}$, alg. 4.3). Selon cette définition, M_{rgb} conserve la couleur de tous les voxels observés. Pour permettre le suivi dynamique du mouvement ou de la déformation d'un objet, on doit y réinitialiser les voxels qui étaient précédemment occupés mais ne le sont plus. Ceci est réalisé par un processus d'élimination des voxels vides décrit par l'équation (4.10). Le résultat de ce processus est

la carte colorée dynamique, M_{cd} , (abrégée « carte colorée ») qui est affichée à l'utilisateur. Une fois que les cartes initiales sont créées et qu'on a complété les prises de vue fixes pour couvrir l'espace de travail, la dernière étape du processus d'initialisation est d'identifier l'objet à manipuler. Ceci est fait simplement en demandant à l'utilisateur de cliquer sur un point de cet objet dans une interface graphique.

Observation de la forme initiale Une fois qu'un point sur l'objet à manipuler a été identifié, la prochaine étape consiste à extraire la forme de cet objet de la carte colorée. Ceci est fait en plusieurs étapes donnant lieu aux cartes montrées à la figure 3.7.

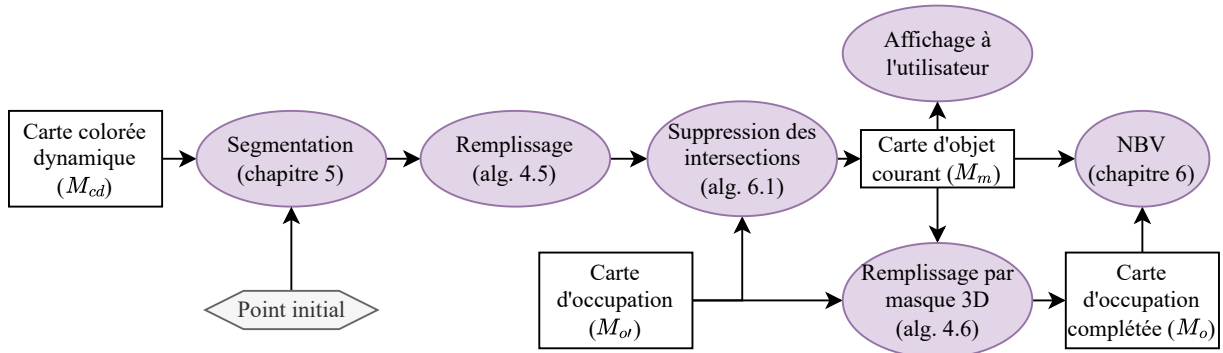


FIGURE 3.7 : Cartes déterminant la forme de l'objet à manipuler.

La première étape pour extraire la forme de l'objet est la segmentation, qui vise à identifier la coquille de l'objet. Avec le point sélectionné par l'utilisateur pour identifier l'objet d'intérêt, on applique l'un des algorithmes de segmentation décrits au chapitre 5 à la carte colorée dynamique pour en extraire les voxels appartenant à l'objet d'intérêt. La deuxième étape consiste à remplir la coquille pour compléter l'espace central non-observable ainsi que les petits défauts et voxels manquants. Ceci est accompli par la procédure décrite à la section 4.3.2. Le résultat de ce remplissage est ensuite transféré à la carte d'occupation pour que les voxels internes de l'objet soient marqués comme occupés dans celle-ci. Ceci est accompli par la procédure de remplissage par masque 3D décrite à la section 4.3.2. Enfin, la troisième étape consiste à supprimer de la carte complétée les intersections potentielles avec d'autres objets, qui nuisent à l'observation. Cette procédure est détaillée à la section 6.1.1. Ces étapes donnent lieu à la carte d'objet courant M_m et à la carte d'occupation complétée M_o , qui sont ensuite utilisées dans le processus d'observation (abrégé « NBV » pour *Next-Best-View*) décrit au chapitre 6 et publié dans [195], qui déplace la caméra pour observer les parties manquantes de l'objet d'intérêt jusqu'à ce que sa forme complète soit connue.

Comparaison de formes 3D et sélection de prise Une fois la forme initiale de l'objet observée, la prochaine étape dans l'objectif de remodelage d'un objet déformable tridimensionnel est d'identifier la meilleure manière de modifier sa forme avec un manipulateur robotique, notamment le positionnement des points de contact et les mouvements que ces points doivent effectuer. La clé de cette étape est la méthode originale de comparaison de formes 3D décrite au chapitre 7 et publiée dans [196].

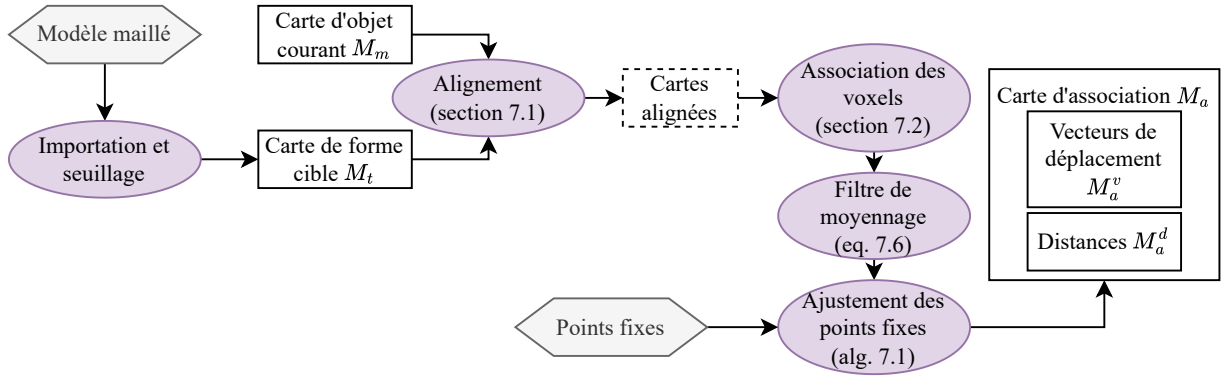


FIGURE 3.8 : Processus de comparaison des formes pour la sélection de prise

Le processus montré à la figure 3.8 débute par le chargement d'une forme cible, ainsi que par la définition optionnelle par l'utilisateur de points « fixes » sur l'objet à manipuler. Ces points fixes permettent de simuler les cas où l'objet est maintenu en place par une pince ou une autre contrainte externe et manipulé par un seul robot. Si aucun point fixe n'est sélectionné, on demande à l'utilisateur de choisir s'il souhaite déplacer l'objet vers la position de la forme cible ou laisser celui-ci en place pendant le remodelage. Suite à la définition de ces options par l'utilisateur, la première étape est d'aligner la carte d'objet courant M_m et la carte de forme cible M_t selon la procédure décrite à la section 7.1. Ces cartes sont ensuite comparées par la procédure d'association des voxels décrite à la section 7.2, qui associe à chaque voxel de M_m un vecteur encodant le déplacement requis pour rejoindre sa position souhaitée dans M_t . Tel que décrit à la section 7.3, ce résultat doit être raffiné par un filtre de moyennage (éq. (7.6)) et l'ajustement des points fixes (alg. 7.1). Le résultat est la carte d'association M_a . Enfin, les points de prise optimaux et leur déplacement souhaité peuvent être obtenus par l'analyse de M_a en fonction du principe de diminution de la rigidité [197, 198] selon les méthodes décrites à la section 7.5.

Suivi de la manipulation et évaluation du remodelage Une fois que les points de prise et leurs mouvements souhaités sont identifiés, il convient de repositionner la caméra pour observer le déplacement des points de prise et le changement de forme de l’objet. Ceci aide notamment à effectuer le contrôle en boucle fermée des déplacements et à surveiller la forme de l’objet pour maintenir la stabilité de la manipulation et détecter un comportement imprévu de l’objet. Un point d’observation est sélectionné selon la procédure décrite à la section 6.4, dont l’objectif est d’assurer un suivi des régions à haute déformation pendant le mouvement. Une fois celui-ci complété, on observe à nouveau la forme de l’objet sous plusieurs angles selon le processus de la section 6.2. Cette procédure vise à assurer une représentation la plus complète possible de la nouvelle forme de l’objet, qui peut ensuite être comparée avec la forme cible et avec la forme initiale pour évaluer si la manipulation a permis de rapprocher la forme de l’objet de la forme cible. Cette évaluation est faite en utilisant les algorithmes de comparaison définis pour l’alignement (section 7.1) et l’association des voxels (section 7.2) ou avec la mesure d’intersection sur union (IoU) 3D (section 5.5.1). Selon les résultats de cette comparaison, on pourrait choisir de faire une nouvelle manipulation à partir de la sélection de prise. Le processus de remodelage de bout en bout est évalué expérimentalement au chapitre 8.

3.4 Sommaire

La complexité de la manipulation robotique et du remodelage des objets déformables tridimensionnels implique la conception et la coordination d’un nombre significatif de données, composantes et processus qui interagissent. Après avoir introduit les conditions du scénario de manipulation considéré, ce chapitre a défini de manière générale la structure du système de manipulation ainsi que les considérations expérimentales ayant une incidence sur la formulation des différents modules de la solution. De plus, ce chapitre présente le modèle de planification et les différentes étapes du remodelage telles que gérées par le module de planification de tâche, qui coordonne les actions des différents éléments du système en fonction de la tâche à effectuer et des contraintes de l’environnement robotique. Il présente ainsi la séquence de données, de méthodes et d’algorithmes proposés pour répondre aux exigences de la tâche de remodelage d’objet déformable définie ici. Les prochains chapitres détailleront le formalisme et la conception algorithmique de ces méthodes.

Chapitre 4

Représentation de l'environnement et cartographie

La représentation de l'environnement est un élément clé d'un système robotique puisqu'elle fait le pont entre la perception, l'interprétation des données et la planification. Pour permettre au système d'évoluer dans un environnement initialement inconnu, cette représentation doit être intégrée dans un processus de cartographie qui explore l'environnement et met à jour la représentation en fonction des nouvelles observations. Dans le contexte de la robotique, la cartographie¹ correspond habituellement à déterminer l'espace libre et occupé dans l'environnement de travail d'un robot mobile. Dans le cadre de cette thèse, on utilise une définition plus générale du concept. Ainsi, une « carte » correspond à une structure pouvant être interrogée pour obtenir la valeur associée à une position quelconque dans l'espace. Cette valeur peut représenter une probabilité d'occupation, mais aussi une couleur, l'appartenance à un objet, ou toute autre information pertinente. Selon ce principe, la « cartographie » correspond aux processus et méthodes pour construire des cartes à partir des données des capteurs, pour les mettre à jour, et pour en extraire les informations nécessaires à la tâche. La représentation de l'environnement et le processus de cartographie doivent être sélectionnés avec soin puisque ces éléments ont une influence cruciale sur la manière dont le système robotique peut accomplir ses tâches. La section 4.1 évalue différentes représentations par rapport aux besoins de la manipulation robotique d'objets déformables, puis la section 4.2 propose un module de cartographie original basé sur la représentation sélectionnée. Enfin, la section 4.3 présente des algorithmes d'usage général développés dans cette thèse et intégrés dans le module de cartographie.

1. *mapping*

4.1 Choix d’une représentation

Pour soutenir la planification dans le contexte du contrôle de forme d’un objet non rigide initialement inconnu, le système robotique doit avoir accès à une représentation interne de l’objet manipulé et de l’environnement. Contrairement à la manipulation d’objets rigides, l’état de l’objet d’intérêt ne peut pas être représenté simplement par des translations et rotations dans un cadre de référence global ; la forme changeante de l’objet doit être incluse dans la représentation. De plus, la représentation choisie doit soutenir les opérations suivantes pour permettre la planification :

1. détection de collision,
2. comparaison de formes tridimensionnelles,
3. segmentation d’un objet d’intérêt à manipuler,
4. calcul du gain d’information par une pose de la caméra,
5. interopérabilité avec le modèle prédictif qui doit s’adjoindre au système.

Puisque l’état de l’objet inclut sa forme, il est aussi important que la représentation puisse être encodée efficacement afin d’éviter une utilisation excessive de la mémoire par le planificateur. La performance de différentes représentations pour ces opérations est décrite ci-dessous et résumée dans le tableau 4.1.

Nuages de points Cette représentation correspond aux données brutes provenant des capteurs de distance et caméras RGBD. Bien que cette représentation offre la meilleure interopérabilité avec les systèmes externes et qu’il existe des algorithmes pour la comparaison [199], le manque d’informations volumétriques est un défaut majeur pour son utilisation dans le système. En effet, le nuage de points n’encode aucune information sur ce qu’il y a entre les points, et aucune donnée sur l’espace occupé, vide ou inconnu. La détection de collision et le calcul du gain d’information deviennent alors difficiles à réaliser. De plus, un nuage de points dense demande beaucoup de mémoire.

Modèles maillés Une représentation standard dans la modélisation et l’animation 3D, les modèles maillés offrent l’avantage de permettre de représenter des surfaces continues et fermées, sans toutefois permettre une représentation explicite des volumes. Cette représentation est pratique pour définir la forme cible (qui peut être développée dans un logiciel de modélisation 3D) et peut être obtenue à partir d’un nuage de points par triangulation ou par un autre algorithme de maillage. La détection de collision est possible mais exige le calcul de

TABLEAU 4.1 : Caractéristiques de différentes représentations de l'environnement.

	Points	Maillage	Voxels	Fréquentiel	Impli-cite
Détection de collision	Difficile	Calculs d'intersection	Simple	Indéterminé	Oui
Comparaison de formes	Sensible au bruit	Sensible au bruit	Petit effet de filtrage	Filtrage intégré	Non
Segmentation	Couleur et distance	Non	Comme pour les images	Non	N.D.
Gain d'information	Non	Non	Simple	Non	N.D.
Encodage efficace	Non	Dépend de la densité	Oui (arbre)	Oui (coefficients)	Oui
Interopérabilité	Meilleure	Variable	Conversion en points	Non	Non

l'intersection entre les surfaces. Les algorithmes de comparaison existent et sont implémentés dans plusieurs logiciels et bibliothèques externes [124, 125]. Dans les travaux de Rowlands [9], on note cependant que l'utilisation de ces modèles pour comparer un objet déformé à une forme cible n'est pas une stratégie optimale. En effet, l'étape de maillage d'un nuage de points provenant du capteur RGBD peut être vue comme une opération d'interpolation qui augmente la précision du modèle au-delà de celle du capteur. Étant données la faible précision des caméras RGBD et la quantité de bruit significative dans les données brutes, ceci résulte en un modèle 3D imparfait avec beaucoup d'ondulations et de protrusions qui ne correspondent pas à l'objet réel, compliquant ainsi la comparaison avec le modèle 3D sans imperfections de la forme cible.

Modèles voxélisés La division de l'espace de travail en unités de volume discrètes offre plusieurs avantages pour la manipulation robotique. D'abord, on peut avoir un encodage explicite de l'occupation de la totalité de l'espace, ce qui facilite la détection de collision et le calcul du gain d'information par une nouvelle pose de caméra. Ensuite, la conversion entre un nuage de points et un modèle voxélisé est simple à faire dans les deux sens et offre un

effet de filtrage et de lissage intégré si les voxels sont plus gros que la résolution du nuage de points. Enfin, la comparaison peut être faite simplement en comptant le nombre de voxels différents entre deux modèles, ou par des algorithmes plus avancés [129]. Cependant, on note que les bibliothèques de voxélisation utilisées en robotique sont typiquement assez limitées, supportant seulement des valeurs ternaires (occupé/libre/inconnu) pour les voxels et limitant parfois le nombre et la taille des voxels. Ces bibliothèques sont habituellement utilisées dans un contexte de navigation en 2D (parfois 3D) plutôt que de manipulation. De plus, la rotation des modèles voxélisés peut donner lieu à des artefacts de la même façon que la rotation des images. Toutefois, l'utilisation d'un environnement voxélisé permet d'appliquer en 3D plusieurs algorithmes développés pour les images et un encodage en arbre permet d'obtenir une représentation efficace en mémoire et en temps de calcul.

Représentations fréquentielles Dans un contexte de rétroaction visuelle² d'objets déformables avec une vue 2D, Navaro-Alarcon et Liu [200] ont obtenu des résultats intéressants en encodant le contour de l'objet avec une série de Fourier tronquée. Cette représentation a l'avantage de fournir un vecteur relativement court (quelques coefficients) pour le stockage et la comparaison, ainsi que des caractéristiques de filtrage et de lissage intégrées. Bien qu'il serait possible d'étendre cette technique pour représenter la surface d'un objet déformable en 3D, il ne semble pas possible d'effectuer la détection de collision avec cette représentation, qui complique aussi l'encodage de l'espace libre et l'interopérabilité avec le modèle prédictif.

Représentations implicites Récemment, les représentation neurales implicites telles les champs de radiance neuronaux (NeRFs³) présentent une alternative aux cartes d'occupation standard [201]. Le principe de ces méthodes d'apprentissage profond est d'entraîner un modèle à prédire la probabilité d'occupation ou l'appartenance à un objet de n'importe quel point dans l'espace à partir d'observations partielles. Ces modèles ont l'avantage d'encoder une représentation continue de l'environnement, ce qui évite les problèmes de précision liés aux représentations discrètes. Ils permettent aussi de compléter les sections manquantes, ce qui permet d'obtenir un modèle sans trous et d'estimer la forme complète avec relativement peu d'observations. Cependant, ces modèles abstraits offrent peu de possibilités d'être manipulés directement, par exemple pour faire la segmentation ou la comparaison de formes.

2. *visual servoing*

3. *Neural Radiance Fields*

Au vu des avantages et inconvénients des différentes représentations évaluées, on se tourne vers un modèle voxélisé unifié pour représenter l’environnement ainsi que l’objet manipulé. En particulier, on choisira une implémentation offrant un accès direct aux voxels par coordonnées cartésiennes pour faciliter le développement d’algorithmes avancés pour la segmentation, la comparaison et d’autres opérations.

4.2 Module de cartographie

Pour assurer la généralité du système proposé, la représentation sélectionnée doit être intégrée dans un processus de cartographie qui demeure indépendant de la tâche à effectuer ainsi que de la plate-forme robotique utilisée, tout en soutenant les opérations nécessaires à la manipulation d’objets déformables initialement inconnus. Comme illustré à la section 2.2, ces caractéristiques sont rarement présentes dans les systèmes de cartographie disponibles présentement. Cette section propose donc un module de cartographie original qui offre les options nécessaires au remodelage d’objets déformables tridimensionnels tout en regroupant les fonctionnalités liées à la représentation de l’environnement et en les isolant des autres composantes du système robotique. Les détails d’architecture logicielle et d’implémentation de ce module sont fournis à l’annexe D.

4.2.1 Concepts

Soit une discrétisation de l’environnement en voxels de dimension cartésienne δ où le point $p \in \mathbb{R}^3$ aux coordonnées cartésiennes (x, y, z) correspond au voxel $v \in \mathbb{Z}^3$ aux coordonnées d’index $(i, j, k) = (\lfloor \frac{x}{\delta} \rfloor, \lfloor \frac{y}{\delta} \rfloor, \lfloor \frac{z}{\delta} \rfloor)$. On suppose que chaque voxel v peut conserver une valeur $\Lambda(v)$ numérique ($\Lambda(v) \in \mathbb{R}$), vectorielle ($\Lambda(v) \in \mathbb{R}^3$) ou booléenne ($\Lambda(v) \in \{0, 1\}$) — le type de valeur est défini au niveau de la grille et non du voxel — ainsi qu’un état $\Gamma(v) \in \{0, 1\}$ qui distingue les voxels « actifs » ou « d’intérêt » des voxels « inactifs » ou « d’arrière-plan ».

On souhaite intégrer cette représentation dans un système de cartographie permettant de conserver en parallèle plusieurs cartes de l’environnement (par exemple, la probabilité d’occupation et la couleur), de distinguer l’objet d’intérêt, la forme cible et le reste de l’environnement, et d’effectuer des opérations sur ces cartes en conservant les résultats des différentes étapes du processus de manipulation. On définit donc une carte M_a comme étant le regroupement d’un nom (a) et d’une grille de voxels, ainsi que d’autres métadonnées facilitant la mise à

jour et l'interprétation des voxels de cette carte. Pour pouvoir conserver ensemble plusieurs valeurs, on définit les sous-cartes $M_a^b, M_a^c, \dots$ qui partagent le nom, les métadonnées et la taille de voxel de M_a et où $b, c, \dots$ identifient les sous-cartes de M_a . Pour maintenir la généralité et la réutilisabilité du mécanisme de cartographie proposé, les cartes et les opérations sur celles-ci sont confinées au module de cartographie. Les autres composantes du système robotique accèdent aux cartes et les manipulent par leur nom, donnant lieu à un dépôt de cartes $R : a \rightarrow M_a$ qui fait le lien entre le nom d'une carte (a) et celle-ci (M_a).

L'interaction principale avec le module de cartographie se fait par la création de cartes et par leur interrogation. Pour répondre aux besoins de la manipulation d'objets déformables, une carte peut être créée à partir d'un modèle maillé pour définir la forme cible, d'un flux de nuages de points pour collecter les données des capteurs, ou en effectuant une opération sur une ou plusieurs autres cartes pour faire des calculs ou sauvegarder un état intermédiaire. Puisque le système évolue dans un environnement dynamique, les cartes doivent être mises à jour en continu pour s'adapter aux changements de l'environnement et de la forme de l'objet manipulé. Pour permettre d'équilibrer la réactivité des cartes et la performance de calcul, plusieurs approches de mise à jour sont considérées, notamment à la demande, à intervalles réguliers, ou à chaque mise à jour des données sources. Le mode de mise à jour fait partie des métadonnées spécifiées à la création d'une carte avec son nom et sa résolution.

4.2.2 Conversion de nuages de points

Pour une carte basée sur un flux de nuages de points, deux types de cartes distincts peuvent être créés. Dans le premiers cas, on souhaite conserver directement les valeurs présentes dans le nuage de points, comme la couleur ou d'autres informations provenant de capteurs (p. ex., capteur de force). Le deuxième cas est celui où on souhaite créer une carte d'occupation identifiant l'espace libre et l'espace occupé. Dans les deux cas, on suppose que les coordonnées du nuage de points sont dans le même cadre de référence que la carte à mettre à jour.

L'algorithme 4.1 décrit la mise à jour d'une carte où l'on conserve les valeurs du nuage de points. La procédure de base est la suivante : pour chaque point p du nuage de points, on assigne au voxel correspondant v la valeur $\Lambda(p)$ du point si le voxel était précédemment indéfini⁴ ou, dans le cas contraire, la moyenne de la valeur du point et de la valeur du voxel, et on le marque comme étant actif. Pour simplifier la notation dans les algorithmes, on définit les

4. On utilise $\Lambda(v) = \emptyset$ (encodé comme NaN/« pas un nombre ») et $\Gamma(v) = 0$ pour identifier les voxels non définis.

opérateurs κ_v et κ_p pour faire la conversion entre les coordonnées en unités réelles (x, y, z) des points et celles en coordonnées d'index (i, j, k) des voxels selon l'équation (4.1), où δ est la dimension d'un voxel.

$$\begin{aligned}\kappa_v(p) &= \kappa_v(x, y, z) = \left(\left\lfloor \frac{x}{\delta} \right\rfloor, \left\lfloor \frac{y}{\delta} \right\rfloor, \left\lfloor \frac{z}{\delta} \right\rfloor \right) \\ \kappa_p(v) &= \kappa_p(i, j, k) = (i \cdot \delta, j \cdot \delta, k \cdot \delta)\end{aligned}\tag{4.1}$$

Algorithme 4.1 Mise à jour d'une carte avec la valeur d'un nuage de points

Entrées

M , la carte à mettre à jour

3: P , le nuage de points

Pour tout $p \in P$, faire

$v \leftarrow \kappa_v(p)$

6: **Si** $\Lambda_M(v) = \emptyset \wedge \neg \Gamma_M(v)$ **alors** ▷ v n'est pas défini

$\Lambda_M(v) \leftarrow \Lambda_P(p)$

Sinon

9: $\Lambda_M(v) \leftarrow \frac{\Lambda_P(p) + \Lambda_M(v)}{2}$

$\Gamma_M(v) \leftarrow 1$

La mise à jour d'une carte d'occupation est une procédure plus délicate puisqu'il est nécessaire d'effectuer une opération de reprojection⁵ entre chaque point mesuré et la position de la caméra pour identifier l'espace non-occupé. La procédure de reprojection utilisée correspond à un algorithme d'analyseur différentiel numérique (DDA⁶) standard [202] illustré à la figure 4.1 et reformulé en 3D.

Cet algorithme calcule, pas à pas, les voxels traversés par un rayon $r = (x_r, y_r, z_r) \in \mathbb{R}^3$ reliant le point $p = (x_p, y_p, z_p) \in \mathbb{R}^3$ et la position de la caméra $c = (x_c, y_c, z_c) \in \mathbb{R}^3$. Ainsi, on commence par calculer le rayon à tracer r ainsi que le nombre de pas à effectuer s , de même que la taille d de ces pas dans les trois dimensions, de sorte qu'on fasse s pas d'un voxel dans la plus grande dimension de r :

$$\begin{aligned}r &= p - c \\ s &= \max(|x_r|, |y_r|, |z_r|) \\ d &= \frac{r}{s}\end{aligned}\tag{4.2}$$

5. *raytracing*

6. *Digital Differential Analyser*

Ensuite, il suffit de commencer la recherche au point c et de faire s pas en additionnant d à chaque fois pour parcourir tous les voxels sur la ligne reliant les points p et c . Notons que l'algorithme standard prévoit des coordonnées dans $\mathbb{R}^2$ et des pixels de taille 1. En plus du passage à trois dimensions, les algorithmes proposés dans cette thèse doivent faire la conversion des coordonnées en unités réelles du nuage de points dans $\mathbb{R}^3$ aux coordonnées d'indexage de la grille de voxels dans $\mathbb{Z}^3$ (équation (4.1)). De plus, la mise à jour de la carte d'occupation comporte des différences significatives si on utilise un modèle d'occupation ternaire (occupé/vide/inconnu) ou probabiliste. Ces deux cas sont discutés plus en détails ci-dessous.

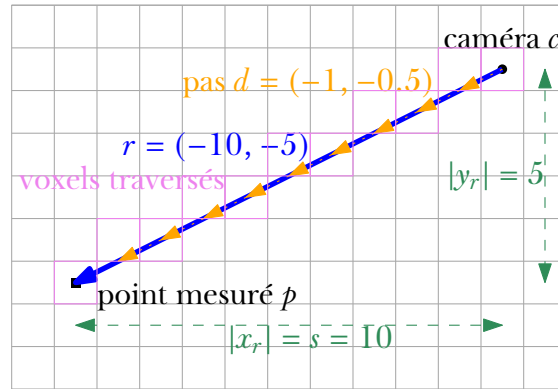


FIGURE 4.1 : Illustration de l'algorithme DDA. Dans le cas d'un modèle d'occupation probabiliste, on fait des pas supplémentaires pour définir la probabilité d'occupation des voxels « derrière » le point mesuré.

Modèle d'occupation ternaire L'algorithme 4.2 définit la procédure de mise à jour d'une carte d'occupation ternaire à partir d'un nuage de points. Puisque la procédure de reprojection est coûteuse et qu'il y a généralement plus d'un point par voxel (la résolution de la grille est plus faible que celle de la caméra), la mise à jour des voxels est d'abord effectuée sur une grille temporaire T , ce qui permet de vérifier si un voxel a déjà été visité et donc de l'ignorer dans les itérations subséquentes. Expérimentalement, on remarque que les erreurs d'arrondissement numériques dans la reprojection font en sorte que des voxels qui devraient être occupés sont parfois marqués comme vides selon l'ordre de traitement des points, en particulier si l'axe optique de la caméra forme un angle aigu avec la surface observée. Cette situation est partiellement corrigée par la vérification à la fin de l'algorithme (ligne 23).

Algorithme 4.2 Mise à jour d'une carte d'occupation ternaire

Entrées

M , la carte à mettre à jour

3: P , le nuage de points

c , la position de la caméra dans le cadre de référence de la carte

$T \leftarrow$ une grille de voxels vide avec la même résolution que M

6: $B \leftarrow \emptyset$ ▷ pour conserver les voxels utilisés

$c' \leftarrow \kappa_v(c)$

Pour tout $p \in P$, **faire**

9: $v \leftarrow \kappa_v(p)$

$B \leftarrow v$

$\oplus$

Si $\neg \Gamma_T(v)$ **alors**

12: $r \leftarrow v - c'$

$s \leftarrow \max(|x_r|, |y_r|, |z_r|)$ ▷ équation (4.2)

$d \leftarrow \frac{r}{s}$

15: $i \leftarrow c'$

$\Lambda_T(v) \leftarrow 1$; $\Gamma_T(v) \leftarrow 1$

$\Lambda_M(v) \leftarrow 1$; $\Gamma_M(v) \leftarrow 1$

18: **Pour** $\zeta \leftarrow 1 \dots \lfloor s \rfloor$, **faire**

Si $\neg \Gamma_T(i)$ **alors**

$\Lambda_T(i) \leftarrow 0$; $\Gamma_T(i) \leftarrow 1$

21: $\Lambda_M(i) \leftarrow 0$; $\Gamma_M(i) \leftarrow 0$

$i \leftarrow i + d$

Pour tout $w \in B$, **faire**

24: **Si** $\neg \Gamma_M(w)$ **alors**

$\Lambda_M(w) \leftarrow 1$

$\Gamma_M(w) \leftarrow 1$

Modèle d'occupation probabiliste La mise à jour d'une carte d'occupation probabiliste ressemble au cas ternaire, mais demande des étapes supplémentaires pour calculer la probabilité d'occupation de chaque voxel. Le modèle est inspiré de [203, 204] et consiste à appliquer une décroissance gaussienne de la probabilité d'occupation autour de chaque point mesuré, puis de les fusionner. Cependant, étant donnés la densité du nuage de points et le fonctionnement de la caméra (section 3.2.1), on note que l'incertitude est principalement présente le long de

l'axe optique de la caméra (valeur de profondeur) et est comparativement négligeable dans le plan de l'image. Ainsi, pour gagner du temps, cette incertitude est appliquée seulement sur la ligne reliant la caméra au point considéré. La formule de [203], modifiée pour appliquer l'incertitude uniquement sur cet axe, est montrée à l'équation (4.3), où p_i est la probabilité d'occupation d'un point i le long de l'axe, ρ_i est la distance entre le point i et la caméra, ρ_m est la distance entre le point mesuré m et la caméra, e_c est l'erreur nominale de profondeur de la caméra (p. ex., 0,02 pour $\pm 2\%$), $e_r = \rho_m \cdot e_c$ est la marge d'erreur au point mesuré m et $\sigma_r = e_r^2$ est la variance à ce point. Le facteur k_g peut être ajusté pour régler la vitesse de convergence de la fonction [204] et est défini à une valeur de 0,5 déterminée empiriquement.

$$p_i = \frac{1}{2} \left(1 + e^{-\frac{2((\rho_i - \rho_m) + 2e_r)}{e_r}} \right)^{-1} + k_g e^{-\frac{(\rho_i - \rho_m)^2}{\sigma_r}} \quad (4.3)$$

Au point mesuré, $\rho_i = \rho_m$ et l'équation devient $p_m = \frac{1}{2(1+e^{-4})} + k_g$ (où p_m dénote la probabilité d'occupation au point mesuré). L'algorithme 4.3 décrit la procédure de mise à jour où, pour chaque point observé, on applique l'algorithme DDA (figure 4.1) et les équations (4.3) à (4.6) pour calculer et fusionner la probabilité d'occupation de chaque voxel sur la ligne reliant ce point à la caméra. Cet algorithme bénéficie de la même optimisation que pour le modèle ternaire en ignorant les voxels déjà visités. Plusieurs versions de la formule de fusion entre la probabilité d'occupation précédente et la probabilité courante à la ligne 32 sont testées pour déterminer le nombre d'observations nécessaires pour converger à un modèle acceptable et la rapidité à laquelle les changements dans la scène, comme le déplacement d'un objet, sont propagés. L'équation (4.4) est une formule bayésienne classique utilisée dans [203], où p_u représente la probabilité d'occupation mise à jour, p_p représente la probabilité précédente, et p_c représente la probabilité calculée dans l'itération courante. L'équation (4.5) est tirée de [94] et utilise un modèle de fusion logarithmique pour accélérer la convergence. Elle est testée telle quelle ainsi qu'une version tronquée, une modification originale qui limite sa valeur à l'intervalle $[0, 1]$. Enfin, l'équation (4.6) est une conception originale qui utilise un facteur de dégradation temporelle k_d pour effectuer la fusion, lequel est empiriquement réglé à 0,25.

$$p_u = \frac{p_p \cdot p_c}{p_p \cdot p_c - (1 - p_p) \cdot (1 - p_c)} \quad (4.4)$$

$$p_u = p_p + \log_{10} \left(\frac{p_c}{1 - p_c} \right) \quad (4.5)$$

$$p_u = k_d \cdot p_p + (1 - k_d)p_c \quad (4.6)$$

Algorithme 4.3 Mise à jour d'une carte d'occupation probabiliste

Entrées

- M , la carte à mettre à jour
- 3: P , le nuage de points
- c , la position de la caméra dans le cadre de référence de la carte
- k_g , un coefficient pour la décroissance gaussienne
- 6: k_d , un coefficient pour la fusion de probabilités
- e_c , l'erreur de profondeur de la caméra (p. ex., 0,02 pour $\pm 2\%$)
- t_p , un seuil de probabilité minimale pour considérer un voxel comme occupé
- 9: $T \leftarrow$ une grille de voxels vide avec la même résolution que M
- $c' \leftarrow \kappa_v(c)$
- Pour tout** $p \in P$, **faire**
- 12: $v \leftarrow \kappa_v(p)$
- Si** $\neg \Gamma_T(v)$ **alors**
- $r \leftarrow v - c'$ ▷ paramètres pour DDA
- 15: $s \leftarrow \max(|x_r|, |y_r|, |z_r|)$ ▷ éq. (4.2)
- $d \leftarrow \frac{r}{s}$
- $\rho_m \leftarrow \|r\|$
- 18: $e_r \leftarrow \rho_m \cdot e_c$
- $\sigma_r \leftarrow e_r^2$
- $p_m \leftarrow \frac{1}{2(1+e^{-4})} + k_g$ ▷ probabilité au point mesuré p
- 21: $\Lambda_T(v) \leftarrow p_m$; $\Gamma_T(v) \leftarrow 1$
- $p_p \leftarrow \Lambda_M(v)$ ou 0,5 si non défini
- $p_u \leftarrow k_d p_p + (1 - k_d) p_m$ ▷ éq. (4.6), interchangeable avec (4.4) et (4.5)
- 24: $\Lambda_M(v) \leftarrow p_u$; $\Gamma_M(v) \leftarrow (p_u > t_p)$
- $i \leftarrow c'$
- Pour** $\zeta \leftarrow 1 \dots (\lfloor s + \frac{\sigma_r}{2} \rfloor + 1)$, **faire** ▷ probabilité sur la ligne reliant p à c
- 27: **Si** $\neg \Gamma_T(i)$ **alors**
- $\rho_i \leftarrow \|i - c'\|$
- $p_c \leftarrow \frac{1}{2} \left(1 + e^{-\frac{2((\rho_i - \rho_m) + 2e_r)}{e_r}} \right)^{-1} + k_g e^{-\frac{(\rho_i - \rho_m)^2}{\sigma_r}}$ ▷ éq. (4.3)
- 30: $\Lambda_T(i) \leftarrow p_c$; $\Gamma_T(i) \leftarrow 1$
- $p_p \leftarrow \Lambda_M(i)$ ou 0,5 si non défini
- $p_u \leftarrow k_d p_p + (1 - k_d) p_c$ ▷ éq. (4.6), interchangeable avec (4.4) et (4.5)
- 33: $\Lambda_M(i) \leftarrow p_u$; $\Gamma_M(i) \leftarrow (p_u > t_p)$
- $i \leftarrow i + d$
-

4.2.3 Optimisation de la fusion de cartes probabilistes

Pour tester la mise à jour de la carte d'occupation (algorithme 4.3), on considère la configuration de la figure 4.2a, où on place la caméra perpendiculairement à la surface de travail à une distance de 50 cm de celle-ci. Un objet est positionné sous la caméra de manière à pouvoir obtenir une vue en coupe de la carte d'occupation qui traverse le centre de l'objet (4.2b). Pour évaluer la qualité des résultats, on considère en particulier deux mesures, soit la vitesse de convergence du modèle d'occupation et la largeur de la zone d'incertitude autour des surfaces mesurées. On cherche à obtenir un modèle qui demande peu d'observations pour répondre aux changements comme l'ajout ou le retrait d'un objet, tout en minimisant l'incertitude autour des objets.

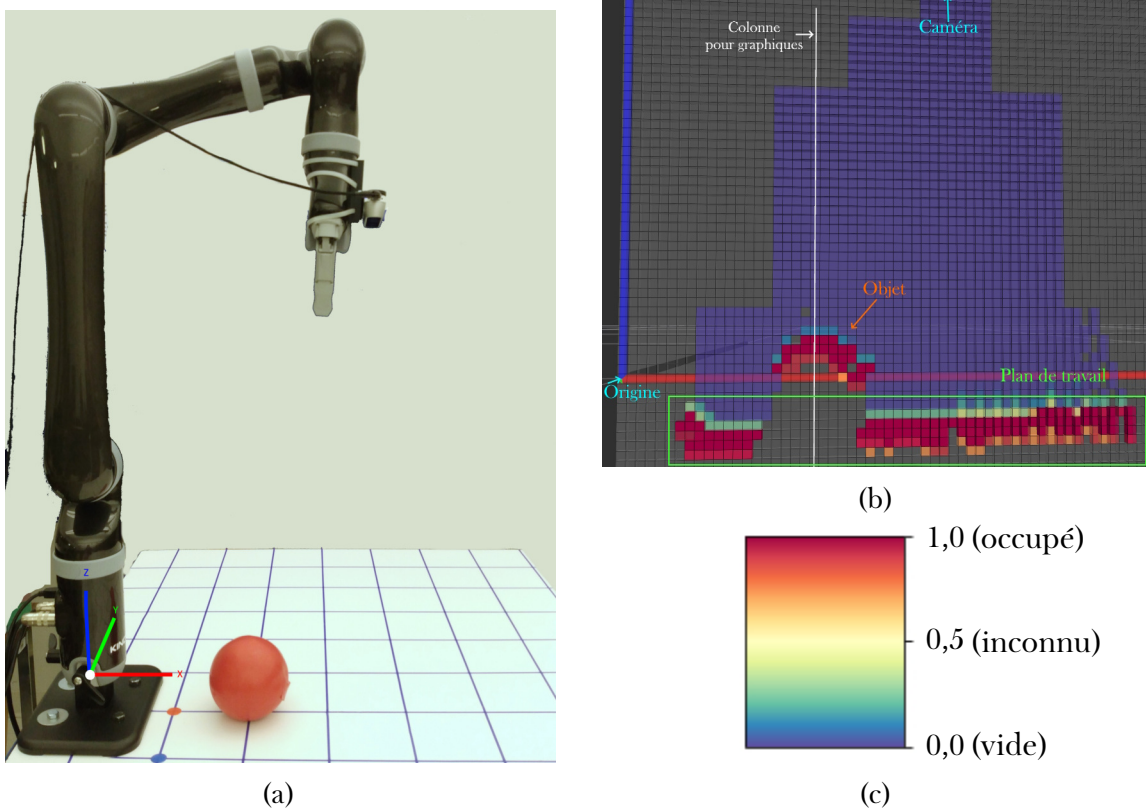


FIGURE 4.2 : Configuration du robot avec le référentiel global (a), exemple de tranche de carte d'occupation (b) et échelle de couleurs représentant la probabilité d'occupation (c) pour tester la mise à jour de la carte d'occupation.

4.2.3.1 Modèles de fusion

Cette section compare les résultats obtenus en utilisant les équations (4.4) à (4.6) pour fusionner de nouvelles observations dans une carte d'occupation probabiliste. Les figures 4.3 à 4.6, où n indique le nombre d'observations prises, montrent l'évolution d'une vue en coupe de la carte d'occupation initialement vide lorsqu'on observe une sphère de 10 cm de diamètre posée sur une table, à partir de la position fixe décrite ci-dessus, en considérant des voxels de 10 mm, une marge d'erreur e_c de $\pm 2\%$ pour la caméra, et les paramètres $k_g = 0,5$ et $k_p = 0,25$. Ces figures montrent d'abord la vue en coupe de la carte d'occupation probabiliste en représentant la probabilité d'occupation sur l'intervalle $[0, 1]$ selon l'échelle de couleur de la figure 4.2c pour différentes valeurs de n . La probabilité d'occupation $\Lambda_{M_o}(x, y, z)$ (où x, y, z sont les coordonnées du point en voxels) pour la colonne de voxels identifiée par une ligne blanche est ensuite tracée graphiquement, permettant une observation plus précise de l'évolution des valeurs représentant la zone d'incertitude autour de la surface mesurée. On rappelle que l'origine du référentiel global est situé à la base du robot, environ 5 cm au-dessus de la surface de travail.

Dans la figure 4.2b et les résultats subséquents, on peut remarquer certains artefacts liés aux mécanismes d'économie d'espace de la bibliothèque de voxels sous-jacente à l'implémentation du module de cartographie. Il est à noter que ces artefacts ne sont pas reliés à la méthode de fusion, mais proviennent du fait que les voxels « inactifs » (où $\Gamma(v) = 0$) sont traités à une plus faible résolution lorsque possible, et particulièrement lorsque ce sont des voxels « d'arrière-plan » ($\Gamma = 0$, $\Lambda = \emptyset$, en gris dans les images). Ainsi, lorsque suffisamment de voxels dans un bloc aligné sur les axes ont une valeur semblable, le bloc complet est mis à cette valeur. Dans les figures, ceci apparaît comme des carrés de 8×8 voxels formant une structure en escalier dans le champ de vision de la caméra et occasionnellement dans les voxels associés à la surface de travail. Dans les scénarios considérés, puisque les voxels associés aux surfaces sont définis comme « actifs », cette perte de précision dans les voxels vides et occasionnellement dans l'intérieur des objets ne cause pas de problèmes en pratique.

Pour l'acquisition initiale, les modèles de fusion bayésienne (figure 4.3) et logarithmique (tronquée ou non, figures 4.4 et 4.5) convergent vers un modèle où la totalité de l'espace incertain autour des objets est considéré occupé, menant à une région occupée qui dépasse la surface réelle de l'objet. On note que la méthode logarithmique converge plus rapidement que la méthode bayésienne, mais qu'à cette étape il n'y a pas de différence significative entre la méthode logarithmique tronquée ou non, bien que la méthode logarithmique non-tronquée mène à des probabilités d'occupation supérieures à 1,0 ou inférieures à 0,0. Pour sa part et

malgré les artefacts d'optimisation discutés ci-dessus, la méthode par somme pondérée (figure 4.6) maintient la région d'incertitude entre la surface de l'objet ou de la table et la caméra, menant à une région occupée plus mince suivie d'une région incertaine et d'une région vide.

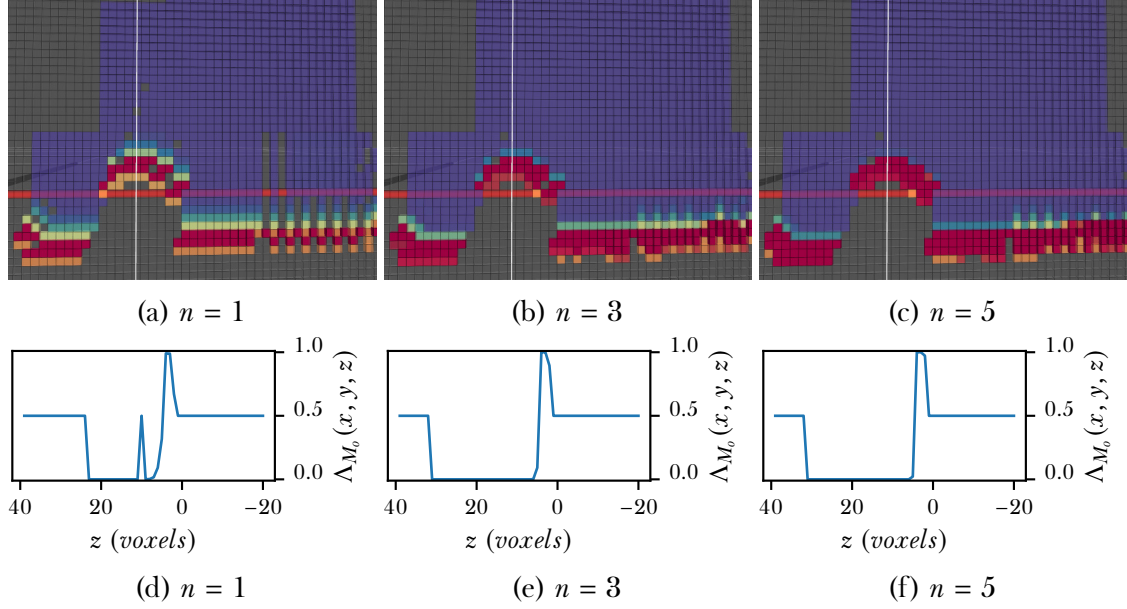


FIGURE 4.3 : Évolution des probabilités d'occupation avec la fusion bayésienne après n observations (résolution : 10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

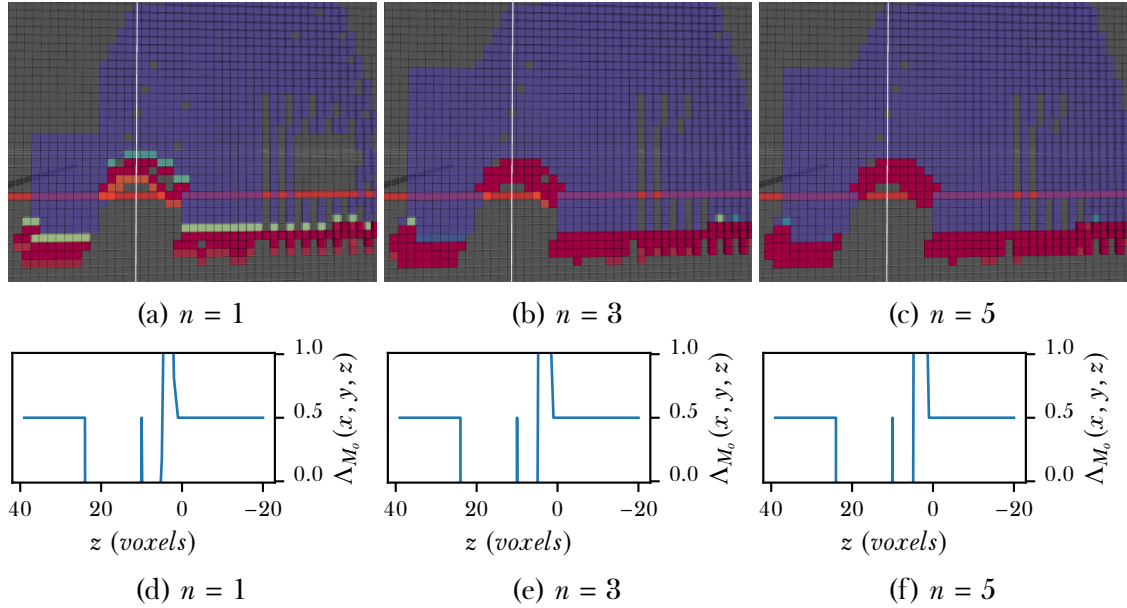


FIGURE 4.4 : Évolution des probabilités d'occupation avec la fusion logarithmique (résolution : 10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

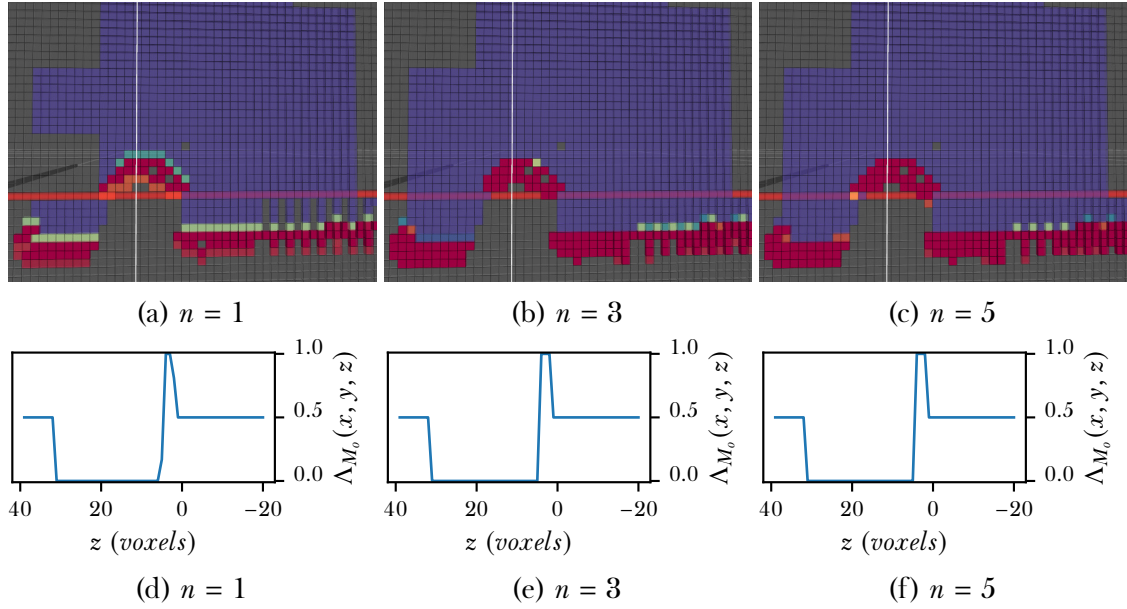


FIGURE 4.5 : Évolution des probabilités d'occupation avec la fusion logarithmique tronquée (résolution : 10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

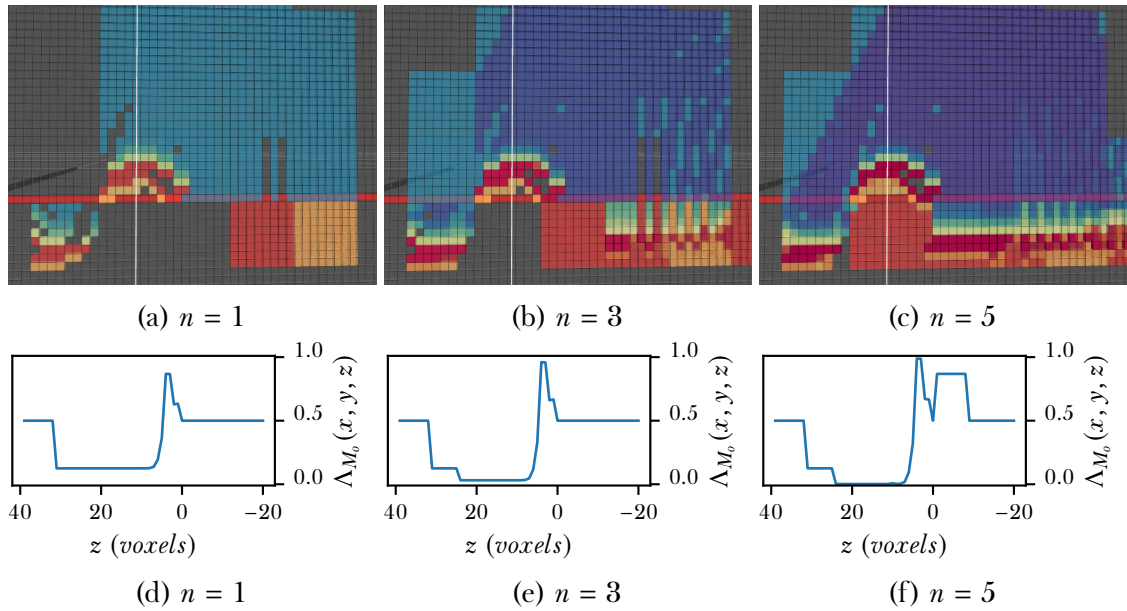


FIGURE 4.6 : Évolution des probabilités d'occupation avec la fusion par somme pondérée (résolution : 10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

Les figures 4.7 à 4.10 montrent la même scène dont l'objet observé a été retiré entre les observations 5 et 6. Lorsque l'objet est retiré, on remarque que la fusion bayésienne n'élimine pas complètement la région qui était précédemment occupée par l'objet, même après 5 nouvelles observations (figure 4.7). La méthode logarithmique converge plus rapidement, et on remarque que la version tronquée (figure 4.9) réagit plus rapidement aux changements que la version non tronquée (figure 4.8), au point qu'une partie de l'espace vide inactif est réinitialisé à la valeur d'arrière-plan (en gris) après un certain nombre d'itérations (figure 4.8c)⁷. La méthode par somme pondérée (figure 4.10) réagit plus lentement que la méthode logarithmique tronquée, mais elle maintient une décroissance plus graduelle de la probabilité d'occupation de l'espace précédemment occupé par l'objet.

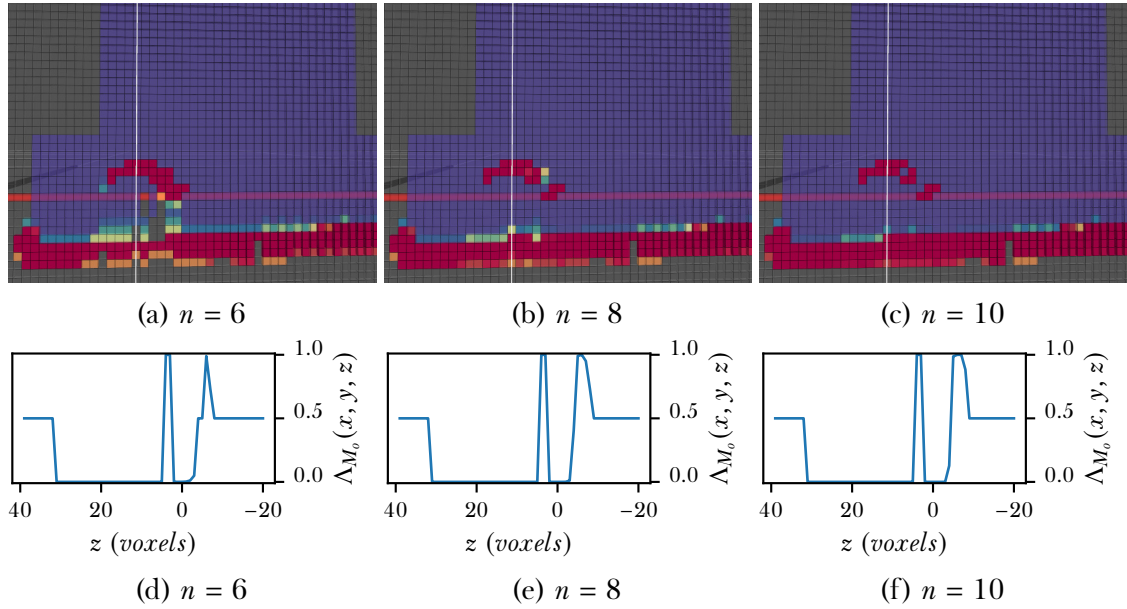


FIGURE 4.7 : Évolution des probabilités d'occupation avec la fusion bayésienne après le retrait de l'objet (résolution : 10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée en blanc.

7. Ceci est un résultat des mécanismes d'économie d'espace dans la structure en arbre d'OpenVDB.

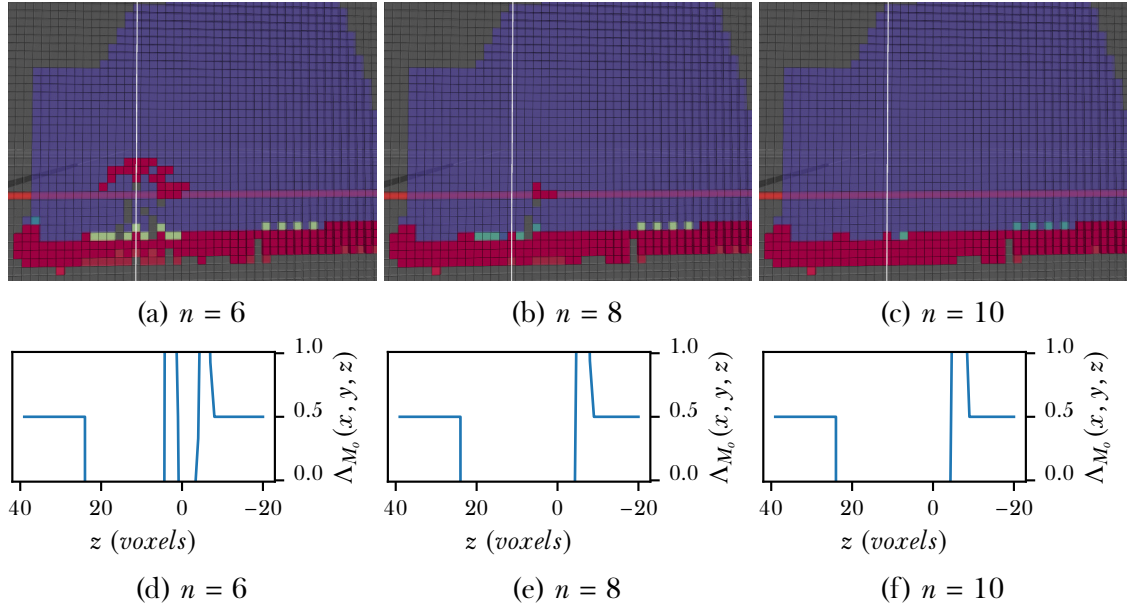


FIGURE 4.8 : Évolution des probabilités d'occupation avec la fusion logarithmique après le retrait de l'objet (résolution : 10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée en blanc.

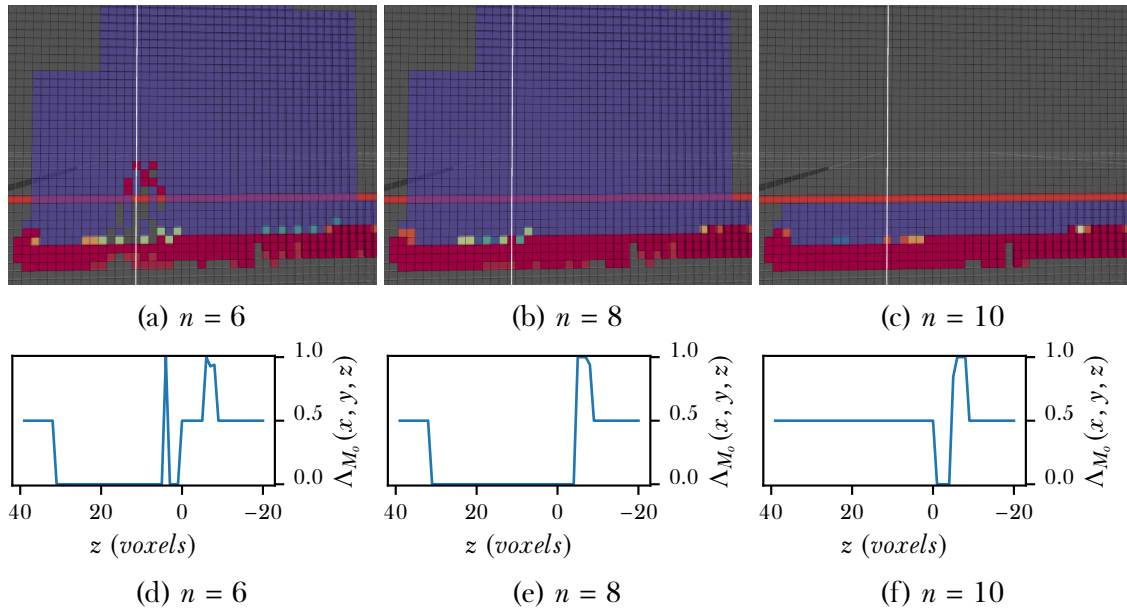


FIGURE 4.9 : Évolution des probabilités d'occupation avec la fusion logarithmique tronquée après le retrait de l'objet (10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée en blanc.

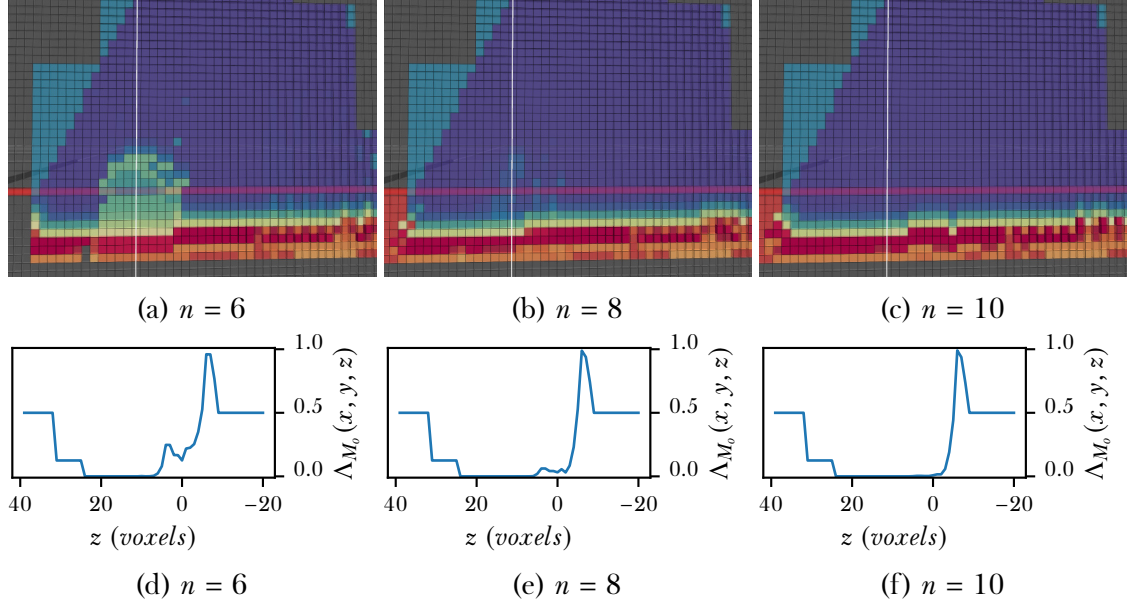


FIGURE 4.10 : Évolution des probabilités d’occupation avec la fusion par somme pondérée après le retrait de l’objet (10 mm/voxel; erreur de profondeur : $\pm 2\%$). (a)–(c) vue en coupe de la carte d’occupation, (d)–(f) probabilité d’occupation pour la colonne marquée en blanc.

Pour mieux différencier les modèles de fusion, on répète l’expérience avec des voxels de 3 mm et une marge d’erreur plus prononcée de $\pm 5\%$ pour la caméra. Les figures initiales (4.11–4.14) incluent aussi une forme de référence rouge translucide qui indique la taille et la position approximatives de la sphère observée. On remarque que la forme de référence est légèrement décalée par rapport aux mesures de la caméra, ce qui s’explique en partie par le positionnement manuel imprécis de l’objet physique et de la forme de référence, mais aussi par l’étirement des surfaces dans l’axe de la caméra causé par l’erreur de profondeur. Pour les modèles de fusion bayésienne (figure 4.11), logarithmique (4.12) et logarithmique tronquée (4.13), le modèle converge encore une fois vers une grande région occupée, qui dépasse significativement la surface de l’objet, mais la plus haute résolution permet de maintenir une mince zone d’incertitude près de cette région. La fusion par somme pondérée (figure 4.14) échange cette grande région occupée pour une zone d’incertitude plus large avec une décroissance plus graduelle de la probabilité d’occupation.

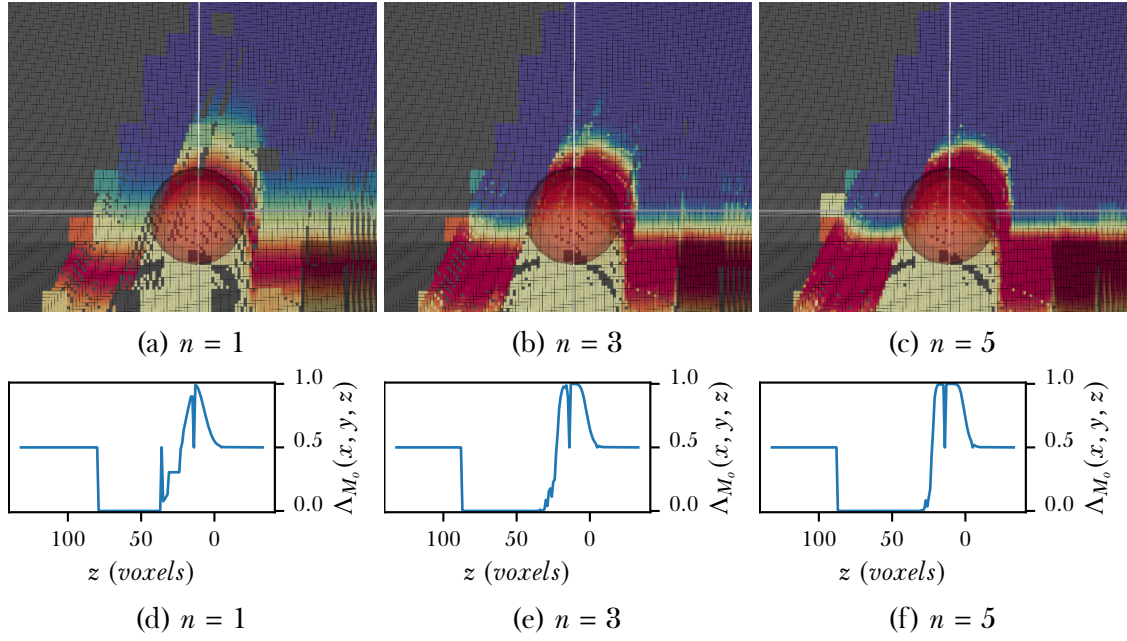


FIGURE 4.11 : Évolution des probabilités d'occupation avec la fusion bayésienne (résolution : 3 mm/voxel; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

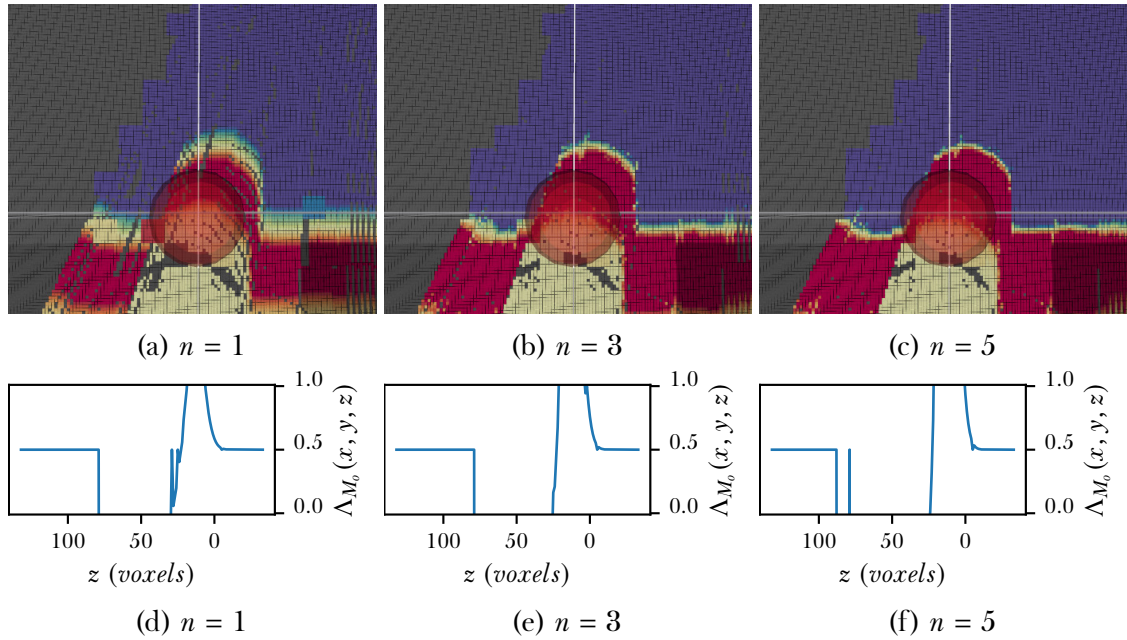


FIGURE 4.12 : Évolution des probabilités d'occupation avec la fusion logarithmique (résolution : 3 mm/voxel; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

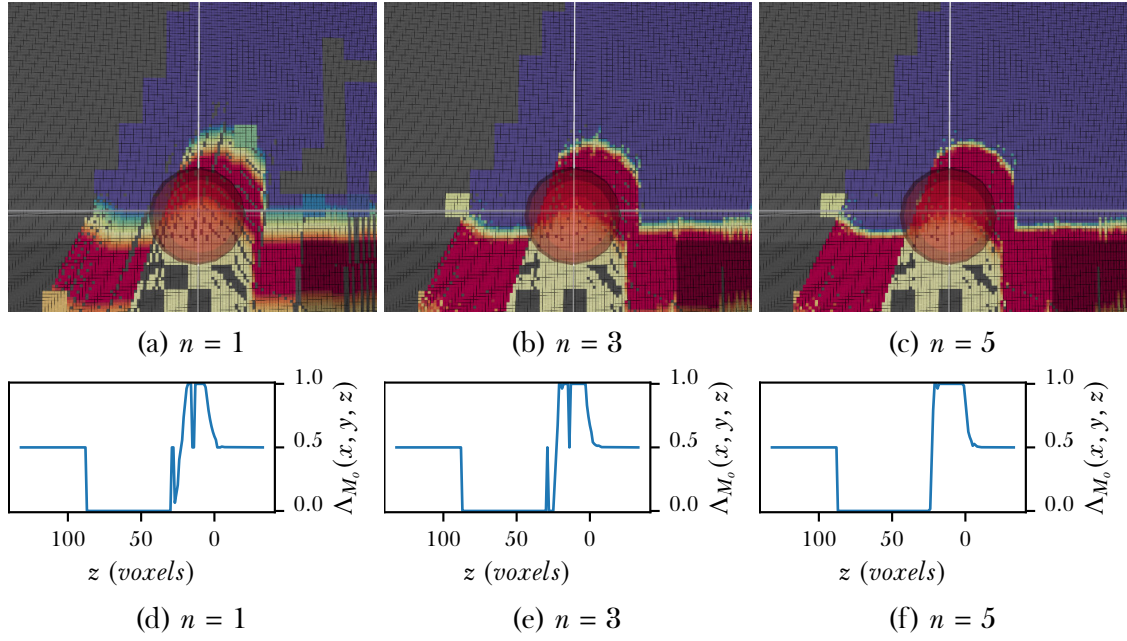


FIGURE 4.13 : Évolution des probabilités d'occupation avec la fusion logarithmique tronquée (résolution : 3 mm/voxel; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

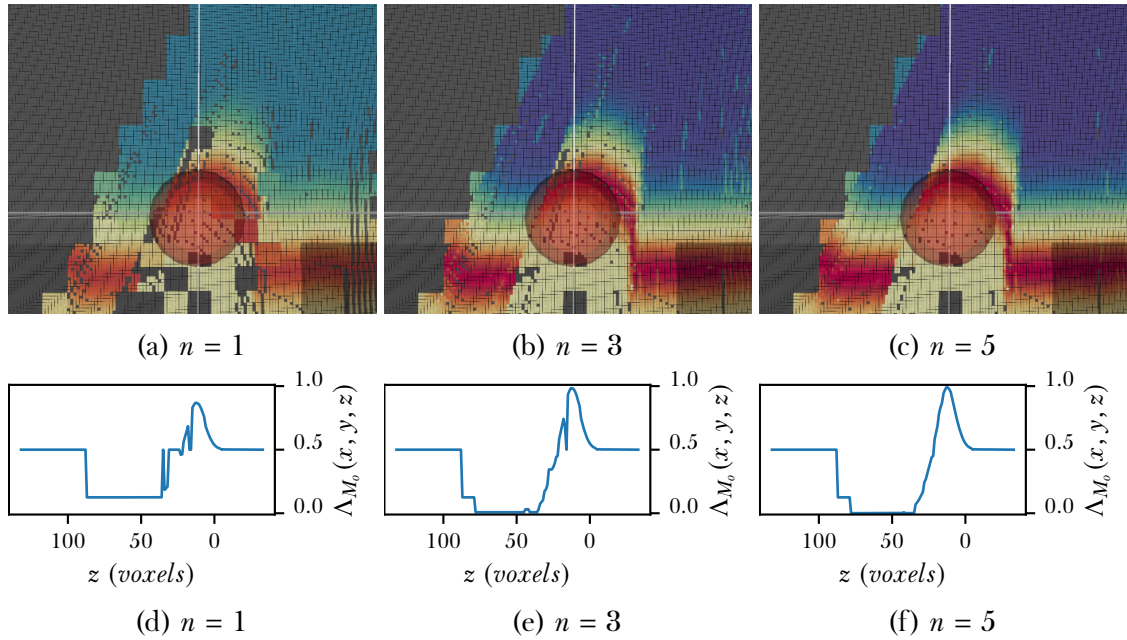


FIGURE 4.14 : Évolution des probabilités d'occupation avec la fusion par somme pondérée (résolution : 3 mm/voxel; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée par la ligne blanche.

Lorsque l'objet est retiré, les modèles de fusion bayésienne (figure 4.15) et logarithmique (4.16) ne permettent pas d'éliminer complètement la région précédemment occupée par l'objet après 5 nouvelles mesures. Les modèles logarithmique tronqué (4.17) et par somme pondérée (4.18) convergent plus rapidement, et le modèle par somme pondérée maintient une décroissance plus graduelle de la probabilité d'occupation.

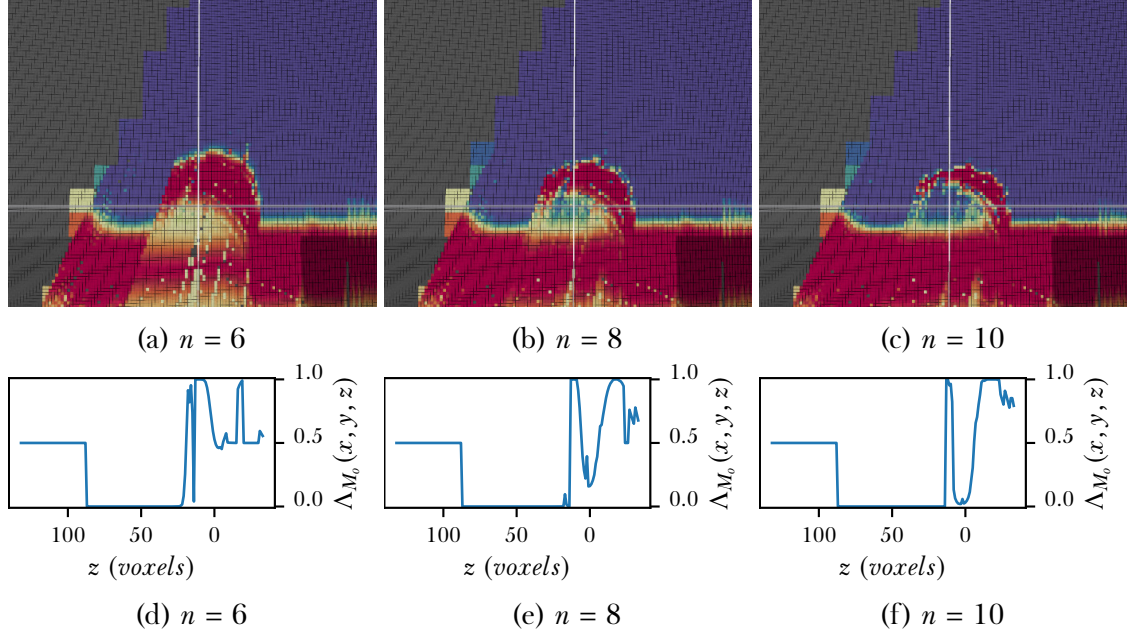


FIGURE 4.15 : Évolution des probabilités d'occupation avec la fusion bayésienne après le retrait de l'objet (résolution : 3 mm/voxel ; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée en blanc.

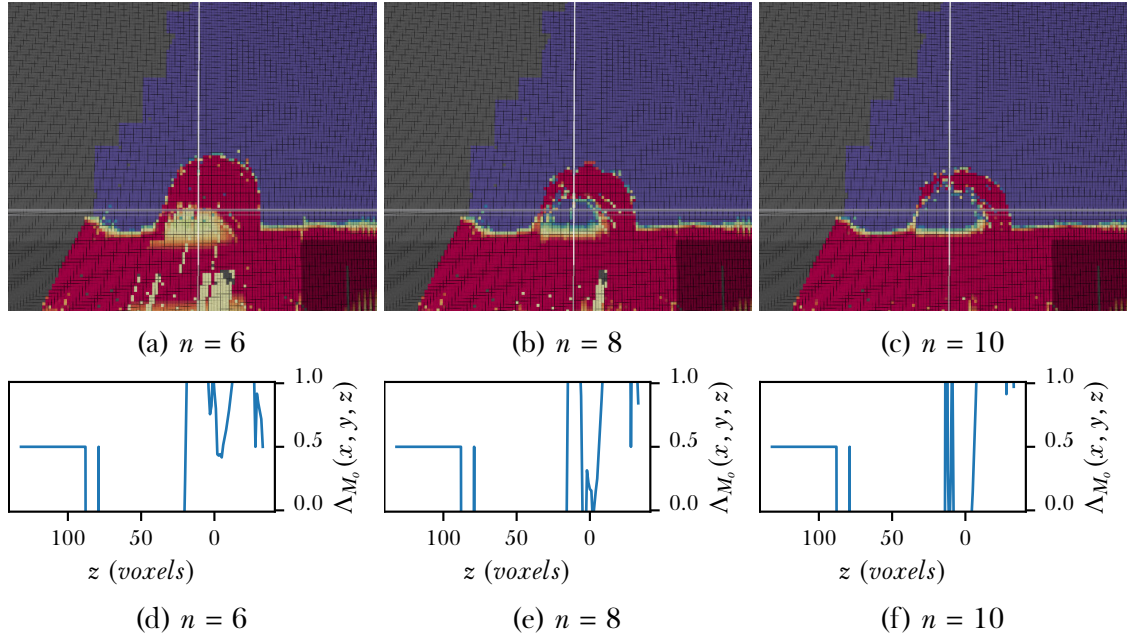


FIGURE 4.16 : Évolution des probabilités d'occupation avec la fusion logarithmique après le retrait de l'objet (résolution : 3 mm/voxel ; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée en blanc.

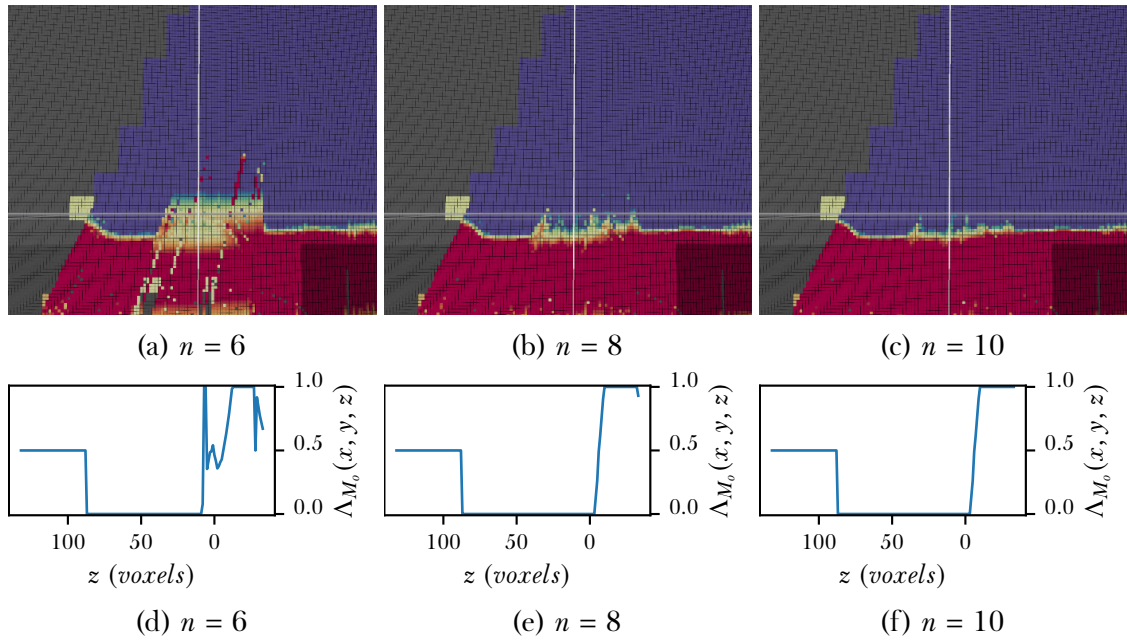


FIGURE 4.17 : Évolution des probabilités d'occupation avec la fusion logarithmique tronquée après le retrait de l'objet (3 mm/voxel ; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d'occupation, (d)–(f) probabilité d'occupation pour la colonne marquée en blanc.

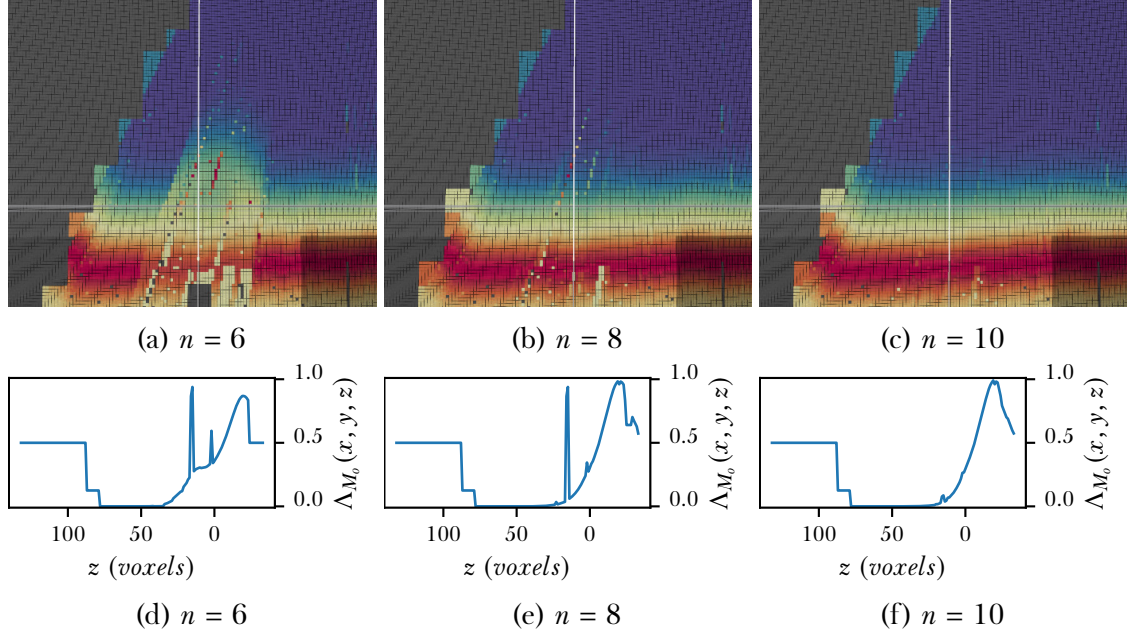


FIGURE 4.18 : Évolution des probabilités d’occupation avec la fusion par somme pondérée après le retrait de l’objet (3 mm/voxel ; erreur de profondeur : $\pm 5\%$). (a)–(c) vue en coupe de la carte d’occupation, (d)–(f) probabilité d’occupation pour la colonne marquée en blanc.

Au vu de ces résultats, les méthodes de fusion logarithmique tronquée et par somme pondérée sont des options valides. Dans un contexte de suivi dynamique de la forme d’un objet où l’imprécision de la caméra cause des erreurs fréquentes dans le nuage de points, la région d’incertitude plus large et plus graduelle présentée par la fusion par somme pondérée est cohérente avec les mesures imprécises et offre une zone tampon pour la planification. La décroissance plus graduelle après le retrait de l’objet permet également d’équilibrer la réactivité et la résistance aux mesures erronées. On opte donc pour cette méthode.

4.2.3.2 Paramètres

En plus du modèle de fusion, deux paramètres affectent la fusion des valeurs d'occupation, soit k_g , qui affecte la largeur de la courbe de probabilité d'occupation pour les nouvelles mesures (équation (4.3)), et k_d , qui règle l'importance relative de la valeur précédente et de la nouvelle mesure dans la fusion par somme pondérée (équation (4.6)). Pour optimiser ces valeurs, on reprend le modèle de fusion par somme pondérée avec des voxels de 3 mm et une marge d'erreur de la caméra de $\pm 5\%$, répétant l'expérience où l'objet est retiré entre les observations 5 et 6. L'évolution des valeurs d'occupation est montrée dans les figures 4.19 à 4.23 en fixant k_d à 0,25 et en faisant varier k_g . Ainsi, lorsque k_g est bas, la probabilité d'occupation demeure incertaine et répartie très largement sur l'espace, même après plusieurs observations. À l'inverse, lorsque k_g est élevé, la probabilité d'occupation est plus concentrée et a une décroissance moins graduelle en s'éloignant de la surface réelle.

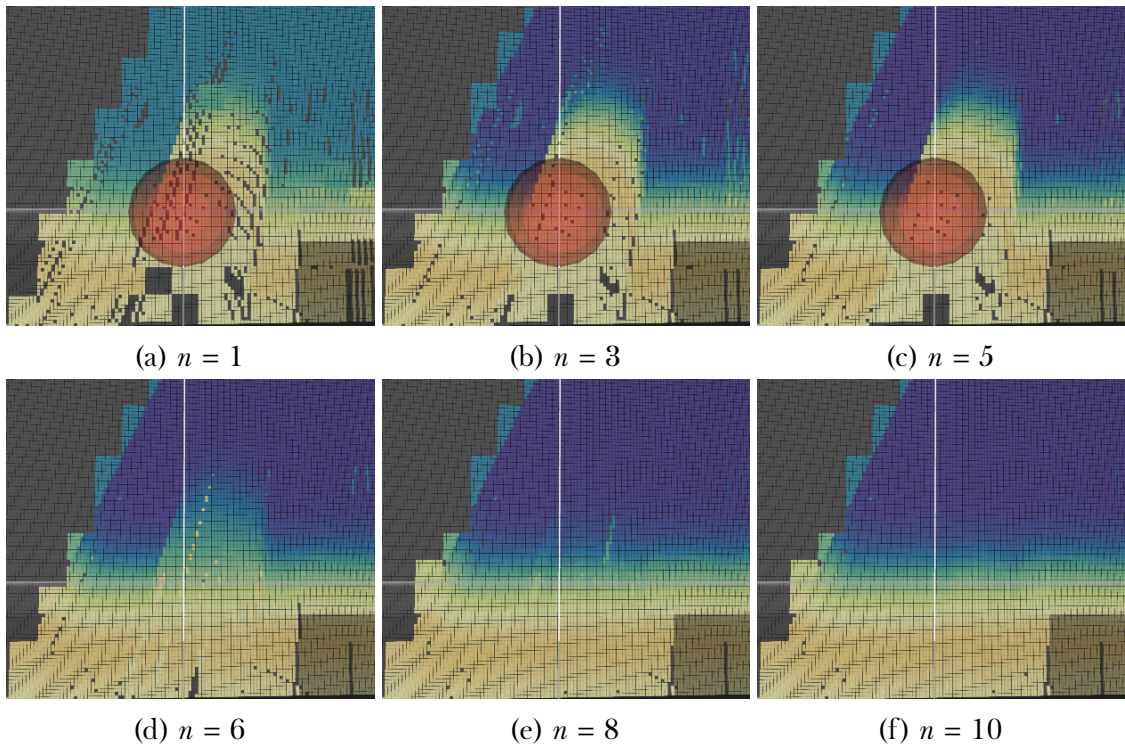


FIGURE 4.19 : Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,1$.

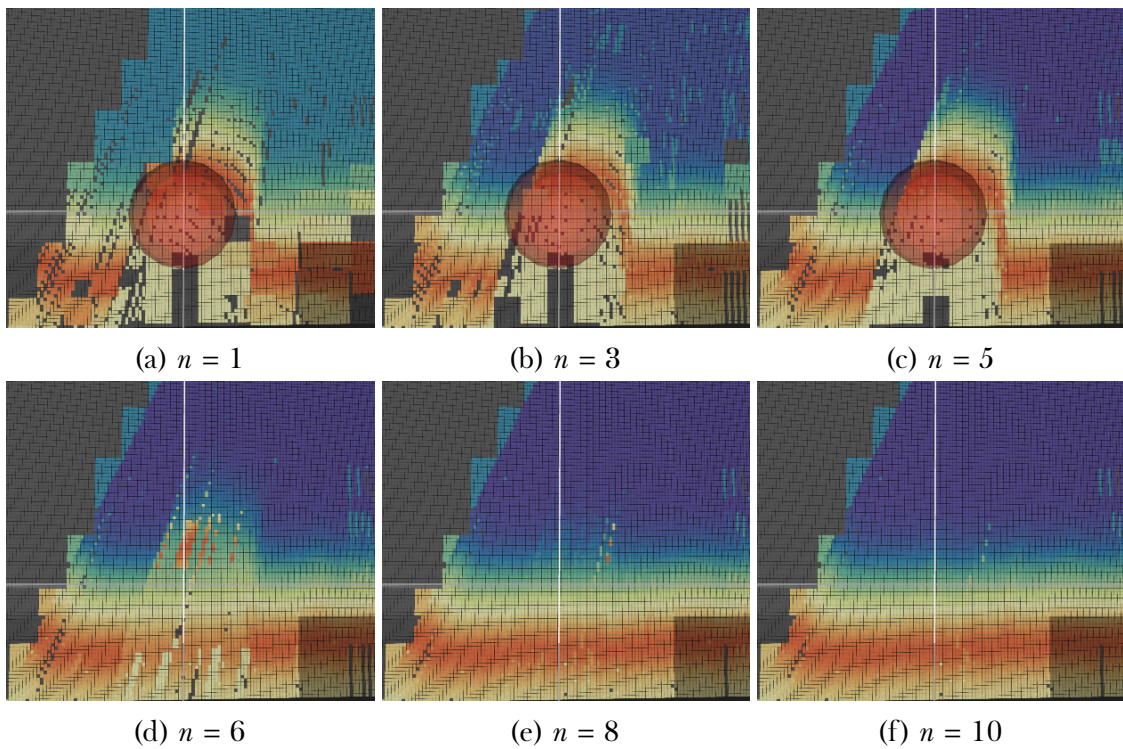


FIGURE 4.20 : Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,3$.

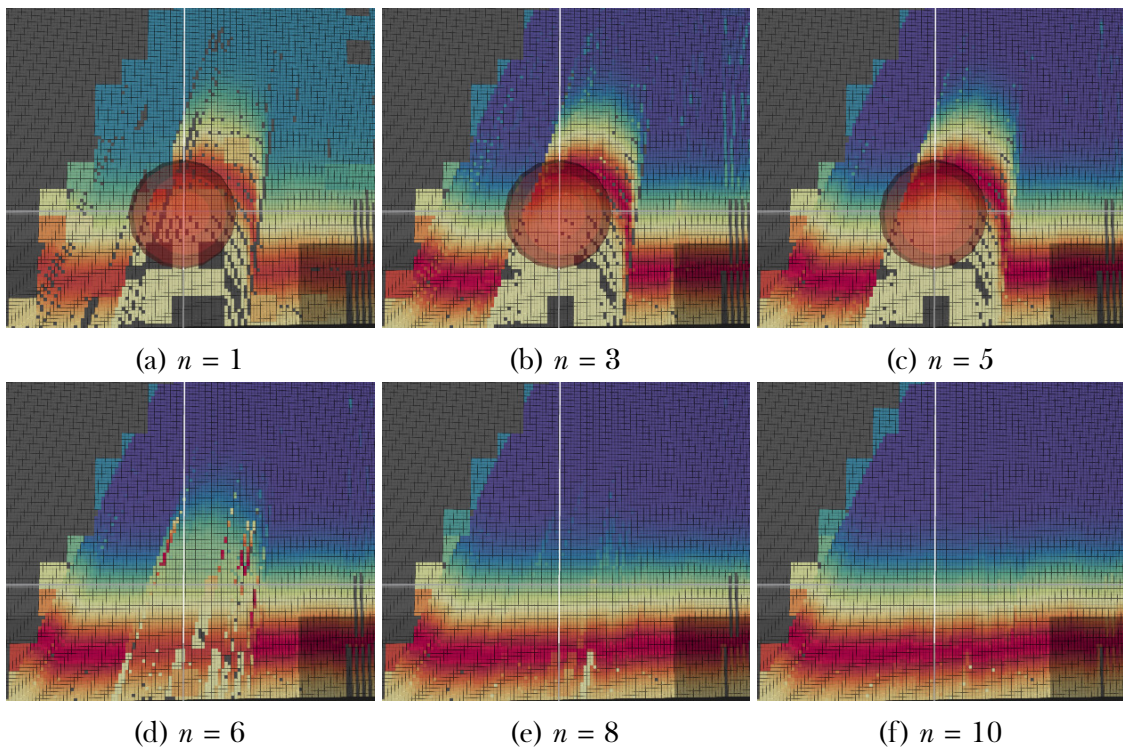


FIGURE 4.21 : Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,5$.

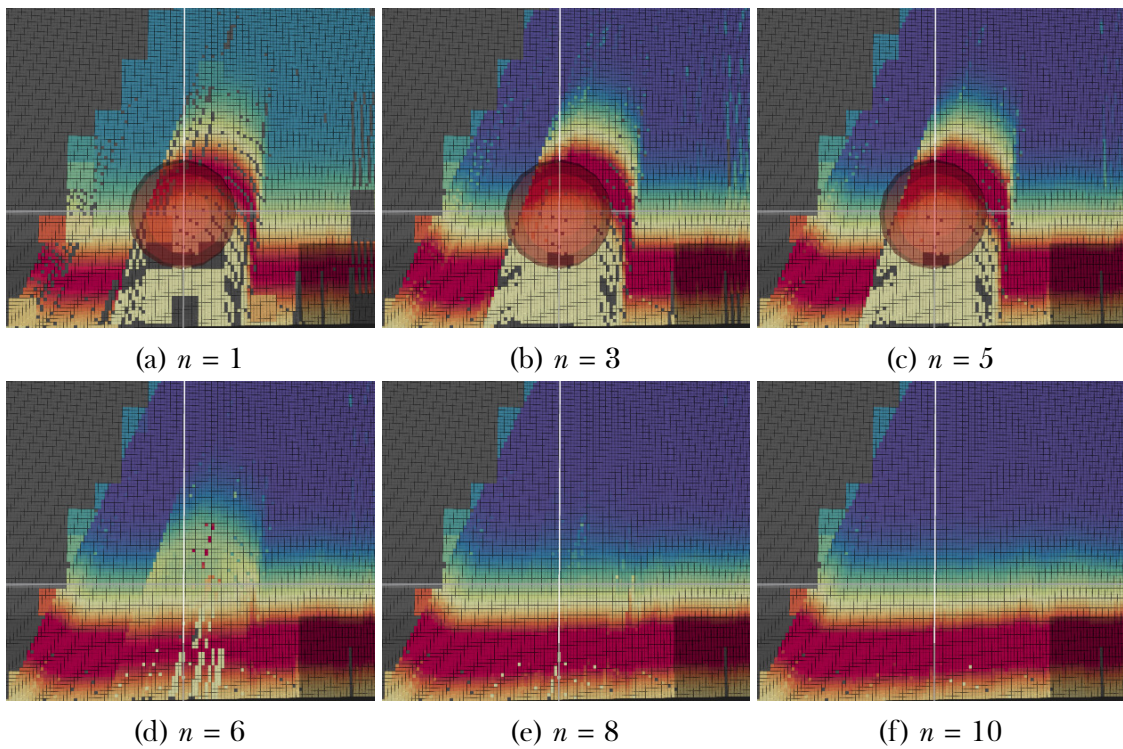


FIGURE 4.22 : Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,7$.

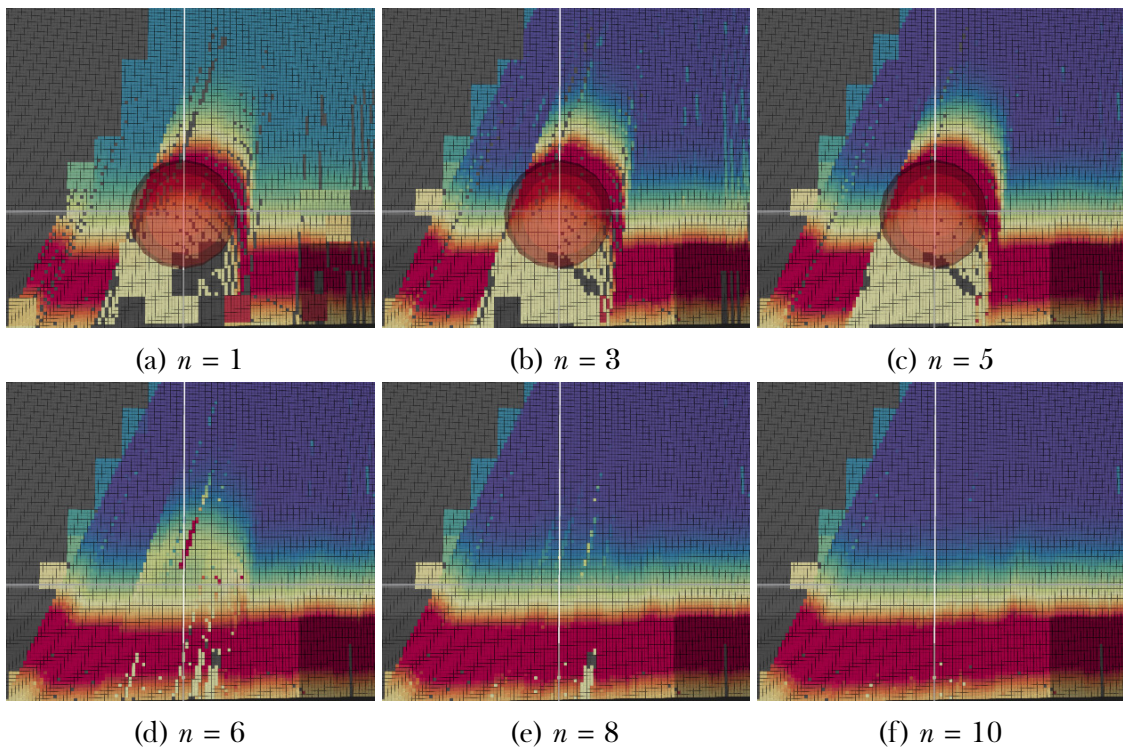


FIGURE 4.23 : Évolution des probabilités d'occupation avec $k_d = 0,25$ et $k_g = 0,9$.

Les figures 4.24 à 4.27 présentent les résultats obtenus en fixant k_g à 0,5 et en faisant varier k_d . Notons que le cas avec $k_d = 0,25$ est déjà montré à la figure 4.21. En ce qui concerne k_d , une petite valeur implique que la probabilité d'occupation est mise à jour presque immédiatement à chaque observation (nonobstant les erreurs numériques dans la reprojection, qu'on peut observer à la figure 4.24). À l'inverse, une grande valeur mène à une mise à jour plus lente qui maintient plus longtemps la persistance de l'objet après son retrait.

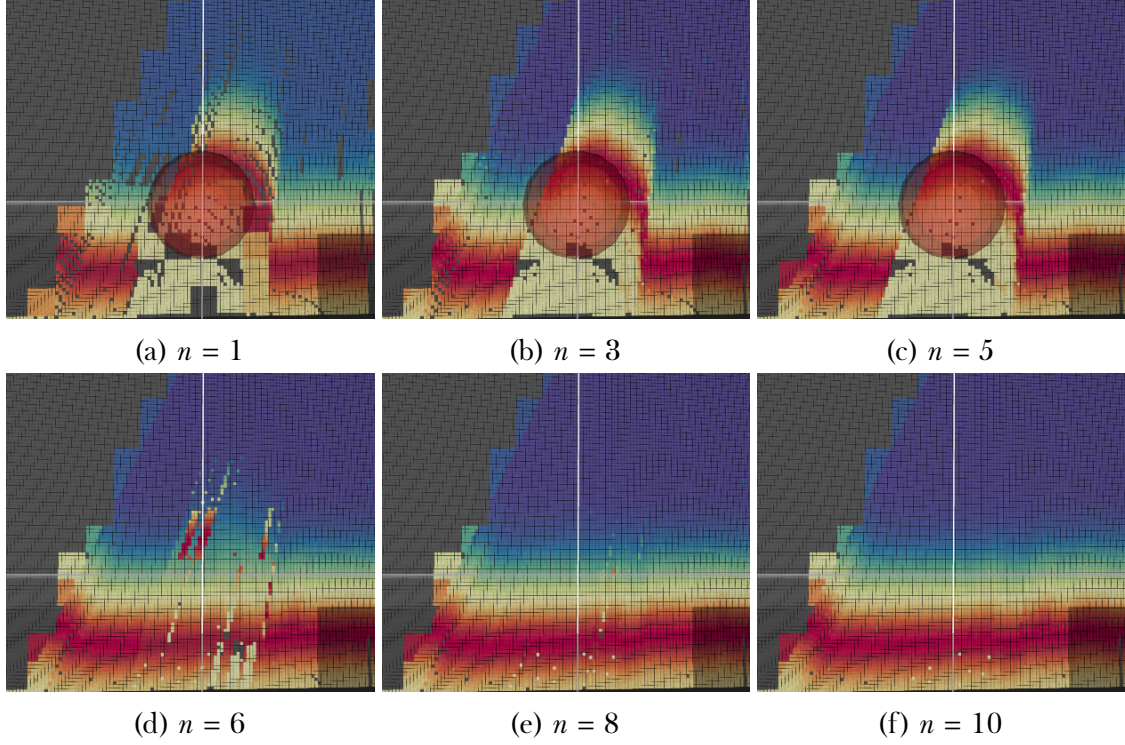


FIGURE 4.24 : Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,1$.

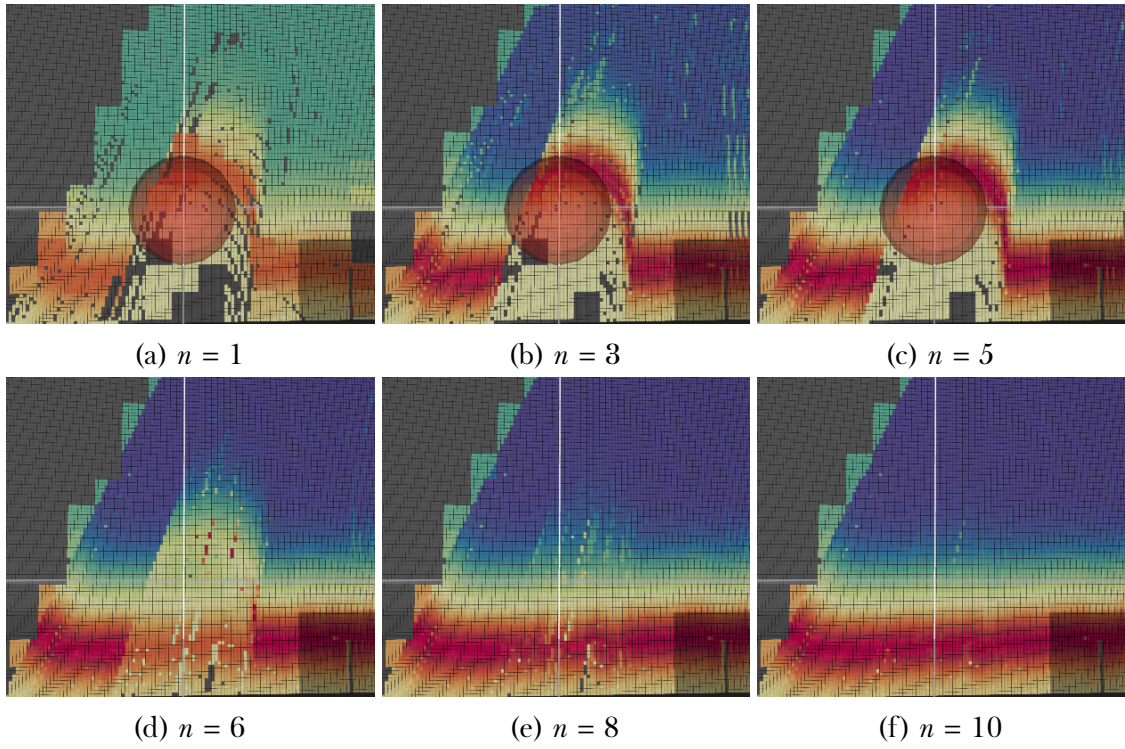


FIGURE 4.25 : Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,4$.

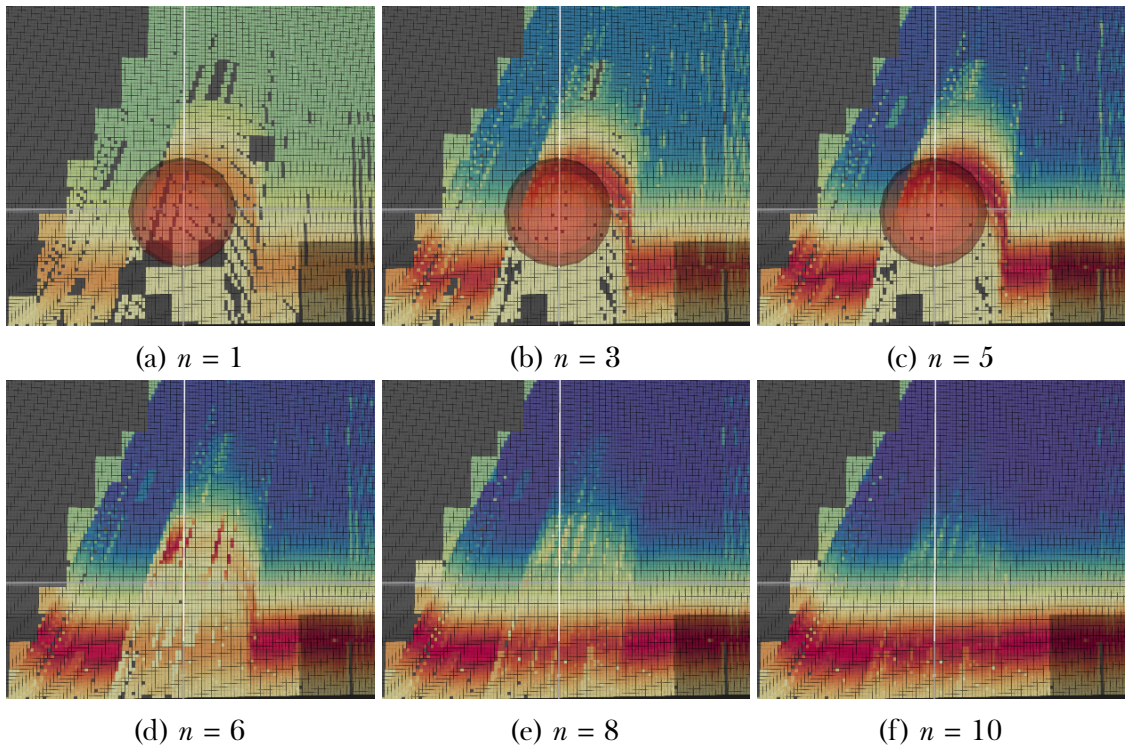


FIGURE 4.26 : Évolution des probabilités d'occupation avec $k_g = 0,5$ et $k_d = 0,6$.

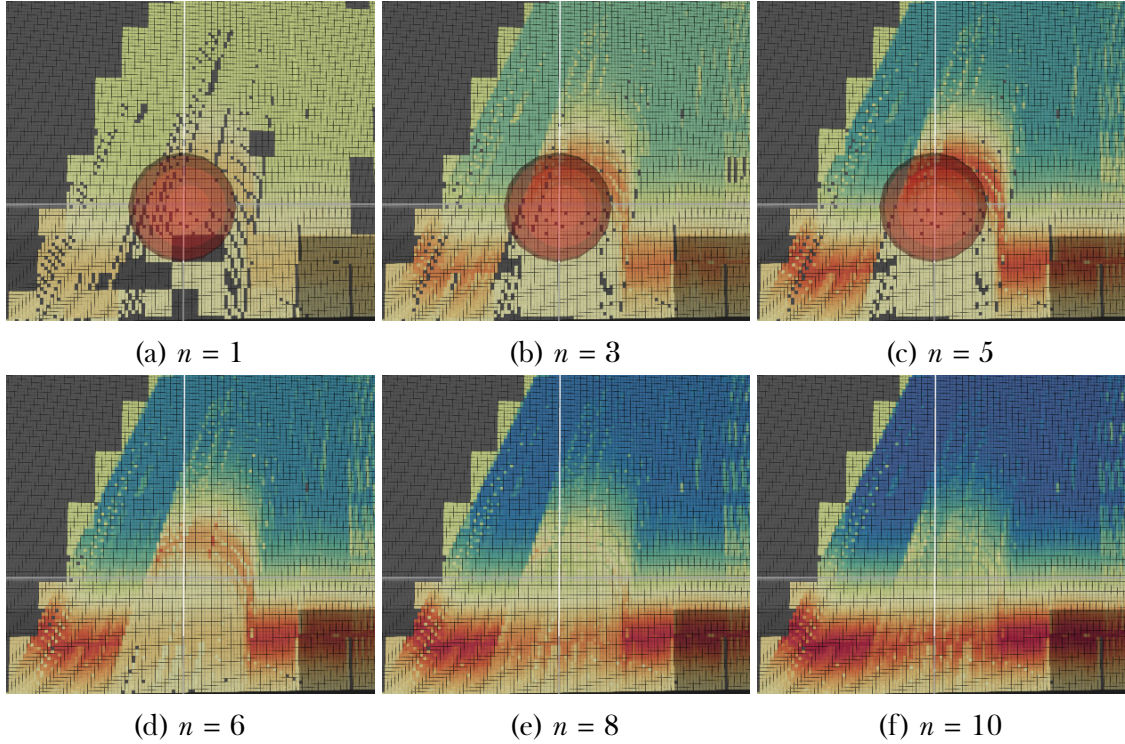


FIGURE 4.27 : Évolution des probabilités d’occupation avec $k_g = 0,5$ et $k_d = 0,8$.

Sur la base de ces résultats, les valeurs choisies de $k_d = 0,25$ et $k_g = 0,5$ permettent d’équilibrer la vitesse de convergence et le maintien d’une zone d’incertitude raisonnable considérant la précision de la caméra et le nombre d’observations à prendre pour chaque point de vue.

4.2.4 Mécanisme de filtres

Un élément clé pour distinguer le système de cartographie original proposé des solutions existantes telles que [90, 94, 95] est la possibilité d’effectuer diverses opérations et manipulations sur les cartes. Dans le contexte de la manipulation d’objets déformables, ces opérations peuvent inclure des filtres dans le sens traditionnel du traitement d’images comme un moyennage, mais aussi des opérations plus avancées comme la segmentation d’un objet d’intérêt, l’addition ou la soustraction des voxels de deux cartes, la comparaison de formes, et tout autre calcul qui transforme une ou plusieurs cartes. Pour éviter de transférer les cartes aux autres parties du système de manipulation robotique, ce qui réduirait l’indépendance du module de cartographie des autres composantes et nuirait à la performance, ces opérations sont confinées au module de cartographie par un mécanisme de filtres. Dans ce contexte, on définit un filtre

comme une fonction $F_o : M_{i1} \dots M_{in}, q_1 \dots q_n \rightarrow M_{o1} \dots M_{on}$ qui prend en entrée une ou plusieurs cartes M_{ix} , effectue une opération sur celles-ci, et a en sortie une ou plusieurs autres cartes M_{ox} . Cette fonction peut aussi prendre des paramètres q_x permettant de contrôler son fonctionnement. La majorité des algorithmes présentés dans cette thèse sont implémentés à travers ce mécanisme.

Puisque les cartes et les fonctions de filtre sont confinées au module de cartographie, les autres composantes spécifient les opérations à effectuer en utilisant une notation post-fixée avec les noms des cartes et des opérateurs. Ceci permet de combiner et d'enchaîner les opérations jusqu'à atteindre le résultat souhaité. Par exemple, si une carte M_r est créée en spécifiant la séquence de filtres et paramètres $(M_1, M_2, F_1, M_3, F_2), (q_{1a}, q_{2a}, q_{2b})$ on effectue les opérations $M_r \leftarrow F_2 (F_1 (M_1, M_2, q_{1a}), M_3, q_{2a}, q_{2b})$. Ces séquences d'opérateurs et de paramètres sont stockées dans les métadonnées de la carte résultante M_r pour permettre sa mise à jour. L'application de ces séquences de filtres est détaillée à l'annexe D.2.

Par défaut, si la carte est configurée pour être mise à jour automatiquement, ceci sera fait lorsque n'importe quelle carte identifiée dans la séquence d'opérations est mise à jour. Pour éviter les mises à jour redondantes si un filtre dépend de plusieurs cartes qui sont mises à jour simultanément à partir du même nuage de points, il est possible de limiter le déclenchement des mises à jour à une liste personnalisée.

4.3 Algorithmes auxiliaires

Les définitions et algorithmes présentés ci-dessous sont utiles à la définition de plusieurs autres algorithmes proposés dans cette thèse. On les regroupe donc ici pour en faciliter la lecture.

4.3.1 Algorithmes multifonctions

Boîte englobante des voxels actifs Puisque les grilles de voxels représentent un espace 3D infini, il convient de limiter les opérations aux régions d'intérêt. Ainsi, pour une carte M , on définit la boîte englobante⁸ des voxels actifs de M , $\mathcal{B}_M$, comme l'intervalle des coordonnées

8. *bounding box*

où M contient des voxels actifs. Ainsi, $\Gamma_M(v) = 0$ pour tout voxel $v \notin \mathcal{B}_M$. Plus formellement,

$$\begin{aligned}
\mathcal{B}_M &= [b_l, b_h] = [(x_l, y_l, z_l), (x_h, y_h, z_h)], x, y, z \in \mathbb{Z} \\
\mathbf{A}_M &= \{v \mid \Gamma_M(v) = 1\} \\
b_l &= (\min(x_i), \min(y_i), \min(z_i)) \mid i \in \mathbf{A}_M \\
b_h &= (\max(x_i), \max(y_i), \max(z_i)) \mid i \in \mathbf{A}_M \\
v \in \mathcal{B}_M &\Leftrightarrow b_l \leq v \leq b_h \Leftrightarrow (x_l \leq x_v \leq x_h) \wedge (y_l \leq y_v \leq y_h) \wedge (z_l \leq z_v \leq z_h)
\end{aligned} \tag{4.7}$$

Voisins d'un voxel On définit l'opérateur $\eta_r(v)$ selon l'équation (4.8) pour représenter les voxels voisins de v dans un rayon $r \in \mathbb{N}$. Ainsi, $\eta_0(v)$ représente l'ensemble des six voxels qui partagent une face avec v , $\eta_1(v)$ représente les 26 voxels (cube de $3 \times 3 \times 3$) entourant v , $\eta_2(v)$ les 124 voisins (cube de $5 \times 5 \times 5$) et ainsi de suite.

$$\eta_r(v) = \begin{cases} \{(x_v \pm 1, y_v, z_v), (x_v, y_v \pm 1, z_v), (x_v, y_v, z_v \pm 1)\} & \text{si } r = 0 \\ \{n \in [(x_v - r, y_v - r, z_v - r), (x_v + r, y_v + r, z_v + r)] \mid n \neq v\} & \text{si } r > 0 \end{cases} \tag{4.8}$$

Recherche de voxels Plusieurs algorithmes demandent de trouver dans une carte G un voxel v correspondant à certains critères définis par une fonction C autour d'un point dans l'espace 3D p . Par exemple, lorsqu'on se base sur un point sélectionné par l'utilisateur dans une interface graphique pour déterminer le point de départ d'une segmentation, G correspond à la carte à segmenter, p est le point sélectionné par l'utilisateur dans l'interface graphique, et $C(\Lambda, \Gamma)$ peut être défini selon l'équation (4.9) de sorte que v est le premier voxel actif rencontré.

$$C(\Lambda, \Gamma) = \begin{cases} 1 & \text{si } \Gamma = 1 \\ 0 & \text{autrement} \end{cases} \tag{4.9}$$

Puisque la précision de la sélection dans l'interface graphique peut varier, $v' = \kappa_v(p)$ ne correspond pas nécessairement au voxel d'intérêt. Ainsi, l'algorithme 4.4 définit une procédure de recherche simple basé sur un ensemble « exploré » E et « frontière » F . Les voisins de v' sont ajoutés à F et explorés tour à tour jusqu'à ce qu'on trouve un voxel qui corresponde au critère C . Pour éviter que la recherche ne croisse dans une direction vide, elle est limitée à $\mathcal{B}_G$, la boîte englobante des voxels actifs de G (équation (4.7)).

Algorithme 4.4 Recherche de voxels

Procédure RECHERCHE(G, p, C)

Entrées

- 3: G , la carte sur laquelle on travaille
 p , le point de départ de la recherche
 $C(\Lambda, \Gamma)$, le critère de recherche

6: **Sorties**

 le premier voxel v trouvé pour lequel $C(\Lambda_G(v), \Gamma_G(v)) = 1$

$v' \leftarrow \kappa_v(p)$

9: $E \leftarrow \emptyset$

$F \leftarrow \emptyset$

$F \leftarrow v'$
 $\oplus$

12: **Tant que** $F \neq \emptyset$, **faire**

$v \leftarrow$ un élément de F

$F \leftarrow v$
 $\ominus$

15: $E \leftarrow v$
 $\oplus$

Si $C(\Lambda_G(v), \Gamma_G(v)) = 1$ **alors Retourner** v

Sinon

18: **Pour tout** $n \in \eta_1(v) \mid n \in \mathcal{B}_G$, **faire**

Si $n \notin E$ **alors**

$F \leftarrow n$
 $\oplus$

Moments 3D L'analyse des moments est une méthode bien connue en traitement d'images pour obtenir les propriétés d'une forme telles le centroïde et l'orientation. En 3D, cette méthode est habituellement utilisée avec une notation continue pour des calculs de forces et de rotation. L'annexe A détaille les équations nécessaires pour l'obtention du centroïde et des trois axes principaux d'une forme 3D voxélisée. Puisque les deux premiers axes principaux représentent les directions dans lesquelles la forme est la plus allongée, le 3e axe principal est la direction perpendiculaire au plan le mieux ajusté à la forme.

Suppression des voxels vides Une carte $M_{x'}$ enregistrant les valeurs du nuage de points et mise à jour selon l'algorithme 4.1 conservera une valeur pour tous les voxels mesurés. Pour permettre le suivi dynamique lorsque les objets observés sont déplacés ou changent de forme, il est nécessaire d'en supprimer les voxels pour lesquels une mesure a été prise précédemment, mais qui ne sont plus occupés. Cette opération nécessite une carte d'occupation M_o où $\Lambda_{M_o}(v)$ définit la probabilité d'occupation du voxel v . Ainsi, l'équation (4.10), où τ_o est un seuil minimal pour considérer un voxel occupé, élimine de $M_{x'}$ les voxels considérés vides dans M_o pour créer la carte M_x (notons que la condition $\Lambda_{M_o} \geq \tau_o$ est semblable à $\Gamma_{M_o} = 1$ mais permet d'utiliser un seuil différent de celui utilisé pour définir les voxels occupés dans M_o).

$$\begin{aligned}\Lambda_{M_x}(v) &= \begin{cases} \Lambda_{M_{x'}}(v) & \text{si } \Lambda_{M_o}(v) \geq \tau_o \\ \emptyset & \text{autrement} \end{cases} \quad \forall v \in M_{x'} \mid \Gamma_{M_{x'}}(v) = 1 \\ \Gamma_{M_x}(v) &= \begin{cases} 1 & \text{si } \Lambda_{M_o}(v) \geq \tau_o \\ 0 & \text{autrement} \end{cases} \quad \forall v \in M_{x'} \mid \Gamma_{M_{x'}}(v) = 1\end{aligned}\tag{4.10}$$

4.3.2 Remplissage des coquilles incomplètes

Par nature, une carte voxélisée produite à partir d'un nuage de points obtenu d'une caméra RGBD ne contient pas d'informations sur les voxels situés à l'intérieur des objets observés. Cependant, pour la plupart des étapes de la manipulation d'objets déformables (détection de collision, association de voxels/planification, calcul de prochain point de vue, etc.), il est préférable que tous les voxels appartenant à l'objet d'intérêt ou aux obstacles soient pris en compte. Il importe donc de remplir la coquille pour assigner une valeur aux voxels internes. Le processus proposé ressemble à celui utilisé dans [113] et consiste à traverser le volume de l'objet ligne par ligne en s'assurant que seuls les voxels internes soient remplis. Dans cette procédure, si la coquille n'a pas de trous, on peut considérer qu'un voxel est interne si, sur la ligne d'un voxel, on a un nombre impair de murs de la coquille avant et après celui-ci, tel qu'illustré à la figure 4.28. En plus du choix de valeur et d'état d'activité à assigner aux voxels internes, l'application de cette procédure en 3D présente des subtilités quant au choix et à l'ordre des axes utilisés pour parcourir la coquille ligne par ligne, en particulier si la coquille représente une vue partielle d'un objet. De plus, on peut choisir d'utiliser ou non le résultat du remplissage sur un axe précédent lorsqu'on traite les axes subséquents. Ces options peuvent donc être ajustées aux conditions expérimentales, notamment l'orientation des prises de vue initiales.

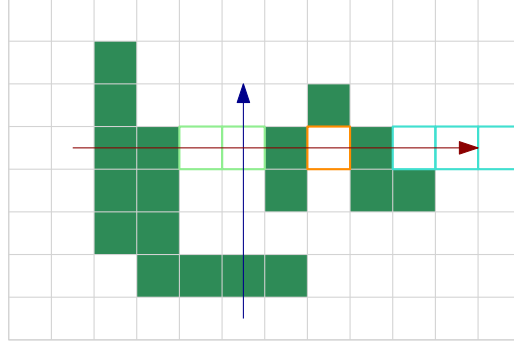


FIGURE 4.28 : Concept du processus de remplissage. Lorsqu'on fait le remplissage en suivant la flèche horizontale rouge, on traverse une région contenant des voxels internes à remplir (cadres vert pâle), puis des voxels vides externes (orange) et enfin une région « interne » (précédée d'un nombre impair de murs) mais non fermée par un dernier mur (bleu). Si on fait ensuite le remplissage selon la flèche verticale bleue, on doit choisir si on utilise ou non les voxels vert pâle remplis précédemment dans la détection de transition.

La procédure de remplissage proposée à l'algorithme 4.5 part du principe de détecter les transitions entre l'extérieur et l'intérieur de la coquille à remplir. Comme illustré à la figure 4.28, ceci est fait en conservant un drapeau ι indiquant si on se trouve présentement à l'intérieur et en détectant les transitions « vide $\rightarrow$ solide » et « solide $\rightarrow$ vide » (lignes 22 à 28) en comparant le voxel précédent p et le voxel courant c . L'algorithme original présenté ici est semblable à celui présenté dans [113], mais étendu pour plus de flexibilité dans la définition d'un voxel « solide » avec $X(v)$ et pour supporter la détection de transition lorsque la coquille comporte plus d'un voxel d'épaisseur — dans [113], on détecte uniquement le passage d'un voxel vide à un voxel solide alors que la méthode proposée vérifie aussi le passage de solide à vide. Puisqu'on travaille avec des observations partielles, il est possible que la coquille soit ouverte d'un côté et donc que la région intérieure continue indéfiniment. Ainsi, contrairement à [113], on ne remplit pas immédiatement les voxels intérieurs vides mais on attend d'atteindre une transition vers l'extérieur (ligne 29). Ceci est illustré par les voxels bleus de la figure 4.28 : ils seraient remplis par la procédure de [113] mais pas par l'algorithme proposé. Enfin, plutôt que d'effectuer le remplissage sur une séquence fixe de tranches, l'algorithme 4.5 tire parti des conventions d'indexage des coordonnées pour effectuer le remplissage sur une séquence d'axes arbitraires α . Ces particularités font aussi en sorte qu'il est possible de choisir de considérer ou non les voxels remplis lors d'une itération précédente dans la détection des transitions intérieur/extérieur.

L'algorithme 4.5 procède donc comme suit. Soit G la grille de voxels à remplir et la fonction $X_M(v) : v \rightarrow \{0, 1\}$ qui indique si le voxel v est considéré « solide » dans la grille M . Le remplissage étant une procédure en une dimension, il est nécessaire de diviser le volume de G en tranches 2D puis en rangées 1D. Puisque l'ordre de cette division peut avoir un impact sur le résultat, en particulier lorsque les observations sont incomplètes, on cherche à spécifier cet ordre de manière flexible. Ainsi, pour un ensemble de coordonnées q , on utilise la convention d'indexage $q = (q_0, q_1, q_2) = (x, y, z)$, ce qui permet de spécifier un axe $a \in \{0, 1, 2\}$ sur lequel effectuer le découpage. Ainsi, si $a = 1 \equiv y$, $(a + 1) \bmod 3$ est l'axe suivant, c'est-à-dire $2 \equiv z$, et $(a + 2) \bmod 3$ est l'axe précédent, $0 \equiv x$. L'espace d'indexage de G étant infini, on limite naturellement le remplissage à $\mathcal{B}_G = [b_l, b_h] = [(b_{l_0}, b_{l_1}, b_{l_2}), (b_{h_0}, b_{h_1}, b_{h_2})]$, la boîte englobante des voxels actifs de G (équation (4.7)). Ainsi, les tranches sur l'axe a vont de b_{l_a} à b_{h_a} , les rangées (sur l'axe suivant) vont de $b_{l_{(a+1) \bmod 3}}$ à $b_{h_{(a+1) \bmod 3}}$ et les colonnes de $b_{l_{(a+2) \bmod 3}}$ à $b_{h_{(a+2) \bmod 3}}$.

Pour chaque rangée, on définit le drapeau $\iota \in \{0, 1\}$ dont les valeurs indiquent respectivement qu'on est à l'extérieur ou à l'intérieur d'un objet, et l'ensemble F qui conserve les voxels intérieurs à remplir. On parcourt ensuite chaque rangée en conservant les voxels courant c et précédent p , qu'on utilise avec ι et X pour détecter les transitions de l'extérieur vers l'intérieur (ligne 22) et de l'intérieur vers l'extérieur (ligne 28). Lorsqu'on est à l'intérieur ($\iota = 1$), on ajoute les voxels vides (où $X_M(c) = 0$) à F pour les remplir lorsqu'on atteint une transition de l'intérieur vers l'extérieur (ligne 29). Si on termine la ligne sans atteindre une telle transition, on supprime le contenu de F pour éviter de remplir par erreur le côté ouvert d'une coquille partielle. Ce processus est illustré pour une ligne à la figure 4.29.

Algorithme 4.5 Remplissage

Entrées

G , la grille de voxels à remplir

3: k_v , la valeur à affecter aux voxels remplis
 k_a , l'état d'activité à affecter aux voxels remplis
 $X(v)$, un opérateur qui détermine si le voxel v est « solide » ▷ p. ex., $\Gamma_G(v) = 1$

6: $\alpha = (a_1, a_2, \dots, a_n)$, $a_i \in \{0, 1, 2\}$ la séquence d'axes sur laquelle découper le volume pour effectuer le remplissage
 $b_l = (b_{l_0}, b_{l_1}, b_{l_2}), b_h = (b_{h_0}, b_{h_1}, b_{h_2}) \leftarrow \mathcal{B}_G$ ▷ éq. (4.7)
 $W \leftarrow G$ ou une copie de G ▷ pour considérer le remplissage des axes précédents ou non

9: **Pour tout** $a \in \alpha$, **faire**

Pour $s_0 \leftarrow b_{l_a} \dots b_{h_a}$, **faire** ▷ chaque tranche 2D

Pour $s_1 \leftarrow b_{l_{((a+1) \bmod 3)}}, \dots, b_{h_{((a+1) \bmod 3)}} ,$ **faire** ▷ chaque rangée 1D

12: $\iota \leftarrow 0$
 $F \leftarrow \emptyset$
 $c = (c_0, c_1, c_2) \leftarrow \emptyset$

15: **Pour** $s_2 \leftarrow b_{l_{((a+2) \bmod 3)}}, \dots, b_{h_{((a+2) \bmod 3)}} ,$ **faire** ▷ chaque colonne/voxel
 $p \leftarrow c$
 $c_a \leftarrow s_0$

18: $c_{((a+1) \bmod 3)} \leftarrow s_1$
 $c_{((a+2) \bmod 3)} \leftarrow s_2$
Si $\iota = 0$ **alors**

21: **Si** $X_W(c) \wedge (p = \emptyset \vee \neg X_W(p))$ **alors**
 $\iota = 1$ ▷ transition extérieur à intérieur

Sinon

24: **Si** $\neg X_W(c)$ **alors**
 $F \leftarrow c \oplus$
Sinon

27: **Si** $p = \emptyset \vee \neg X_W(p)$ **alors**
 $\iota \leftarrow 0$ ▷ transition intérieur à extérieur
 $\Lambda_G(f) \leftarrow k_v \quad ; \quad \Gamma_G(f) \leftarrow k_a \quad \forall f \in F$

30: $F \leftarrow \emptyset$

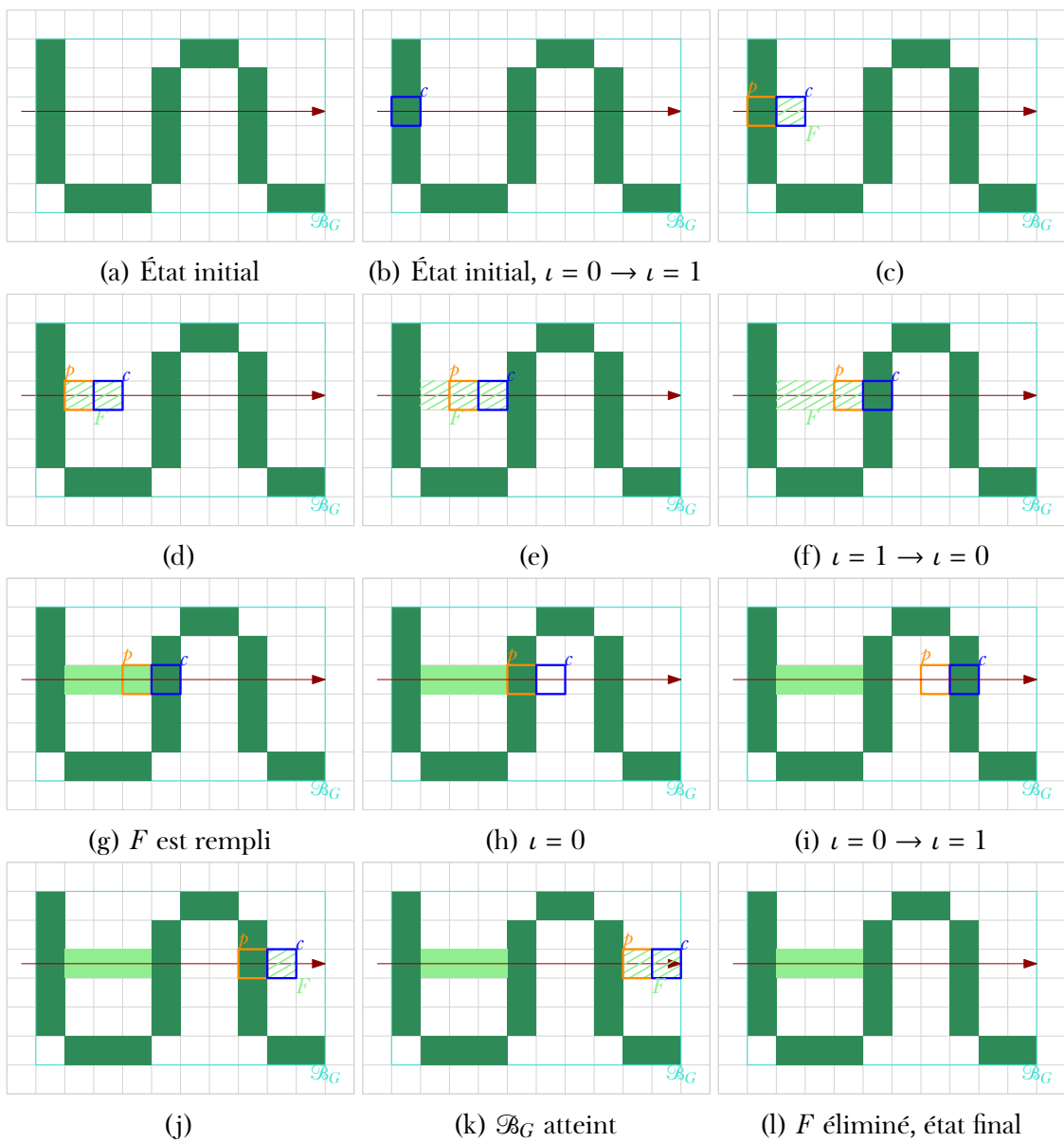


FIGURE 4.29 : Illustration pas à pas du processus de remplissage.

Remplissage par masque 3D Une autre méthode de remplissage peut être utilisée lorsqu'on souhaite propager le résultat du remplissage d'une carte à une autre. Ceci peut par exemple servir lorsqu'on a une grille de voxels remplie représentant l'objet à manipuler et qu'on souhaite marquer l'intérieur de l'objet (non-observable) comme occupé dans une carte d'occupation. Le principe est semblable à l'utilisation d'un masque dans le traitement d'images 2D : soit une grille de voxels M telle que $\Gamma_M(v) = 1$ indique les voxels v à remplir et G , une grille de voxels quelconque sur laquelle effectuer le remplissage. L'algorithme 4.6 affecte la valeur et l'état d'activité appropriés aux voxels d'arrière-plan de G selon M . Ceci permet de gagner en efficacité par rapport à l'algorithme de remplissage complet (4.5) lorsque M est disponible.

Algorithme 4.6 Remplissage par masque 3D

Entrées

- M , une grille de voxels où $\Gamma_M(v)$ indique les voxels à remplir
 - 3: G , la grille de voxels sur laquelle effectuer le remplissage
 - k_v , la valeur à affecter aux voxels remplis
 - k_a , l'état d'activité à affecter aux voxels remplis
 - 6: **Pour tout** $v \mid \Gamma_M(v) = 1$, **faire**
 - Si** $\neg\Gamma_G(v) \wedge \Lambda_G(v) = \emptyset$ **alors**
 - $\Lambda_G(v) \leftarrow k_v$; $\Gamma_G \leftarrow k_a$
-

4.4 Sommaire

Le remodelage d'objets déformables par un système robotique nécessite l'inclusion des informations sur la forme changeante de l'objet dans la représentation de l'état de l'environnement. En plus de la traditionnelle détection de collision et des calculs d'observabilité, cette représentation doit supporter des opérations avancées telles la segmentation de l'objet d'intérêt et la comparaison de formes tridimensionnelles, en plus d'offrir une bonne interopérabilité avec un modèle prédictif de la réponse de l'objet manipulé et les autres composantes logiciel. Ce chapitre présente une analyse des différentes représentations disponibles et propose un module de cartographie répondant aux besoins de la manipulation et du remodelage d'objets déformables tridimensionnels initialement inconnus dans un environnement partiellement observable.

Basé sur des grilles de voxels efficaces encapsulées dans une interface de cartes nommées, le système de cartographie original proposé dans ce chapitre se distingue des solutions existantes

[90, 94, 95] en permettant de définir des cartes en appliquant des opérations arbitrairement complexes à d'autres cartes. Ces opérations sont définies à travers un mécanisme de filtres qui permet notamment la mise à jour en continu des cartes lorsque les données d'entrée sont modifiées tout en assurant que les cartes puissent être conservées, visualisées, analysées et manipulées à l'intérieur du module de cartographie, indépendamment de la tâche à accomplir, du robot utilisé et de l'environnement de travail. Ce mécanisme permet au planificateur de tâche de définir des séquences d'opérations relativement complexes, comme une segmentation suivie d'un alignement, d'une comparaison puis d'un moyennage, sans s'inquiéter des détails du module de cartographie ou de la bibliothèque de voxels sous-jacente.

De plus, le module de cartographie propose des méthodes indépendantes du robot et de la tâche pour la création et le traitement des cartes voxélisées, notamment pour la conversion de nuages de points en cartes d'occupation ternaires ou probabilistes, pour l'analyse des moments 3D et des axes principaux, et pour le remplissage des coquilles d'objets dans les cartes. Ce remplissage inclut notamment un algorithme amélioré qui supporte le remplissage de coquilles partielles. Les détails d'architecture logiciel et d'implémentation du module de cartographie sont discutés à l'annexe D.

Chapitre 5

Algorithmes de segmentation en 3D

Une étape critique dans la manipulation robotique est la segmentation, qui vise à distinguer l’objet à manipuler du reste de l’environnement. Cette étape est particulièrement importante dans la manipulation d’objets déformables puisqu’on doit suivre et réobserver la forme changeante de l’objet au cours du processus de manipulation pour surveiller le progrès et la sûreté de la tâche. Dans ces applications, les méthodes de segmentation 2D sont bien établies, mais le coût en calcul des méthodes 3D incite souvent un retour à un seuillage simple sur la couleur ou la profondeur de nuages de points individuels (sections 2.1 et 2.3.1). Dans un environnement partiellement observable, ceci implique d’effectuer la segmentation avant de fusionner les observations, ignorant le contexte plus large de la scène. Pour relâcher ces contraintes, plusieurs algorithmes de segmentation sont définis dans les sections ci-dessous et intégrés dans le mécanisme de filtres du module de cartographie (section 4.2.4), ce qui permet de proposer un processus où ils sont appliqués après avoir regroupé les données des capteurs.

Étant données la puissance de calcul limitée des robots autonomes et les contraintes de temps réel imposées par le suivi de la forme d’un objet en cours de manipulation, des algorithmes efficaces sont requis pour la segmentation. De plus, il y a peu d’exemples d’objets déformables tridimensionnels dans les ensembles de données publics, et l’annotation de données 3D est longue et complexe. Ainsi, les algorithmes présentés ici se détournent des approches d’apprentissage machine coûteuses en données et en puissance de calcul, s’inspirant plutôt de méthodes classiques pour le traitement d’image avec l’objectif d’étudier leur application aux données 3D voxélisées et d’améliorer leur performance tout en maintenant leurs avantages en termes de temps de calcul et d’applicabilité à différents scénarios. Ainsi, on compare la segmentation simple par couleur (section 5.1) avec une formulation 3D originale de la segmentation par *Fast Level-Sets* (section 5.2) et une méthode hybride combinant des avantages des deux méthodes (section 5.3). Ces méthodes de segmentation sont suivies par

une brève discussion de la conversion des résultats de la segmentation en masques d’objet binaires ou probabilistes (section 5.4). Enfin, selon l’idée où les objets sont séparés par des bordures concaves [87], on étudie l’utilisation de la courbure locale comme information utile à la segmentation et à la manipulation. L’utilisation directe de cette mesure n’ayant cependant pas été retenue comme une méthode de segmentation viable dans le cadre de cette thèse, cette étude est présentée à l’annexe C. Bien que les méthodes proposées aient des similarités conceptuelles avec les méthodes basées sur les supervoxels [114, 115], elles ont un objectif différent. Les méthodes par supervoxels sont habituellement utilisées comme étape de pré-traitement pour sur-segmenter une scène statique, complète et de taille fixe de manière à accélérer des méthodes plus coûteuses visant à étiqueter chaque élément de la scène. En contrepartie, les méthodes proposées visent à extraire et à suivre la forme changeante d’un objet d’intérêt spécifique dans une scène de taille indéfinie.

5.1 Segmentation par couleur

Connaissant la position et la valeur (qui correspond ici à la couleur) d’un point sur l’objet d’intérêt, la méthode la plus simple pour séparer celui-ci du reste de l’environnement consiste à extraire de la carte tous les voxels dont la valeur est suffisamment proche de celle du point connu. Ainsi, posons la grille de voxels G où $\Lambda_G(v)$ contient une valeur pour les voxels v où $\Gamma_G(v) = 1$ et est indéfini ailleurs. Une telle grille peut être construite, par exemple, à partir d’un nuage de points produit par une caméra RGBD. À partir de G , on peut construire une grille booléenne M telle que $\Gamma_M(v) = 1$ si v est un voxel sur l’objet d’intérêt et $\Gamma_M(v) = 0$ ailleurs en appliquant un seuil τ selon l’équation (5.1), où v' est un voxel qu’on sait faire partie de l’objet d’intérêt :

$$\Gamma_M(v) \leftarrow 1 \ \forall \ v \in \mathcal{B}_G \mid \|\Lambda_G(v) - \Lambda_G(v')\| \leq \tau \quad (5.1)$$

En utilisant la distance euclidienne pour comparer les valeurs, ce processus supporte la conversion de $\Lambda_G(v)$ en différents espaces de couleur (intensité, RGB, HSV ou YUV) ainsi que la sélection des canaux à utiliser. Ceci permet donc de choisir une combinaison adaptée à l’environnement, puisque certains modèles de couleur et canaux peuvent se montrer plus résistants aux variations d’éclairage ou de texture.

Cependant, cette méthode doit calculer la distance de la valeur souhaitée pour tous les voxels de la grille, ce qui peut se révéler inutilement coûteux lorsque l’objet à extraire comporte peu de voxels par rapport à la grille complète. De plus, elle suppose que seul l’objet à extraire est proche de la valeur souhaitée. Pour pallier à ces deux défauts, une méthode alternative

est proposée pour limiter la recherche aux voxels voisins de ceux ayant déjà été marqués comme faisant partie de l'objet à extraire. Cette recherche est faite d'une manière semblable à l'algorithme 4.4, ce qui correspond à initialiser M avec $\Gamma_M(v') \leftarrow 1$ et à appliquer l'équation (5.2) à répétition, c'est-à-dire qu'on marque comme actifs dans M les voxels qui sont voisins d'un voxel actif de M et dont la valeur est plus proche de la référence que le seuil τ , jusqu'à ce qu'aucun nouveau voxel ne soit ajouté à M .

$$\Gamma_M(n) \leftarrow 1 \quad \forall n \in \eta_1(v) \mid \Gamma_M(v) = 1, \|\Lambda_G(n) - \Lambda_G(v')\| \leq \tau \quad (5.2)$$

5.2 *Fast Level-Sets* 3D pour la segmentation de voxels

La segmentation par *Fast Level-Sets* est utilisée dans le traitement d'image pour identifier un objet d'intérêt [205]. Cette technique, qui se montre relativement résistante aux variations de texture et de couleur sur l'objet à détecter, est basée sur la permutation des points de l'image entre deux listes, L_{in} et L_{out} , qui représentent respectivement la bordure interne et la bordure externe de l'objet. Cette permutation est faite avec l'aide de deux représentations intermédiaires de même format que l'image à traiter, soit $\Phi(p)$ qui indique si le pixel p est à l'intérieur de l'objet ou non, et $F_{ext}(p)$, une fonction de vitesse contrôlant l'évolution de la bordure.

Φ et F_{ext} prennent traditionnellement les valeurs suivantes :

$$\Phi(p) = \begin{cases} -1 & \text{si } p \text{ est dans } L_{in} \\ 1 & \text{si } p \text{ est dans } L_{out} \\ -3 & \text{si } p \text{ est dans l'objet} \\ 3 & \text{si } p \text{ est hors de l'objet} \end{cases} \quad (5.3)$$

$$F_{ext}(p) = -(I(p) - m_{in})^2 + (I(p) - m_{out})^2 \quad (5.4)$$

où $I(p)$ est l'intensité de l'image au point p , m_{in} est la moyenne de $I(p)$ pour tous les points p où $\Phi(p) < 0$ et m_{out} la moyenne de $I(p)$ où $\Phi(p) > 0$. On note que (5.4) correspond au cas où I est une image à un seul canal, où chaque partie de l'équation correspond au carré de la distance entre I et $m_{in/out}$. Dans nos travaux précédents [8], pour considérer le cas d'une image à plusieurs canaux, on définit $m_{in/out}$ comme le vecteur contenant la moyenne de chaque canal i (indépendamment), ce qui permet de reformuler F_{ext} :

$$\begin{aligned} F_{ext}(p) &= -\|I(p) - m_{in}\|^2 + \|I(p) - m_{out}\|^2 \\ &= -\sum_i (I(p)_i - m_{ini})^2 + \sum_i (I(p)_i - m_{outi})^2 \end{aligned} \quad (5.5)$$

Une méthode originale est ici proposée pour adapter cet algorithme de *Fast Level-Sets* multicanaux afin de segmenter un objet d'intérêt dans une grille de voxels 3D. Tout d'abord, on note que, contrairement à une image, les voxels peuvent être vides, ce qui est représenté par une valeur indéfinie ($\emptyset$) dans les grilles. Ces valeurs doivent être vérifiées et ignorées dans les calculs pour Φ , $m_{in/out}$ et F_{ext} . De plus, on n'utilise pas la transformation en coordonnées log-polaires exploitée dans [8] puisque cette transformation ainsi que l'ordonnancement du contour qui en résulte sont moins bien définis en 3D. Enfin, les grilles de voxels représentant un espace d'indexage infini, il n'est pas possible d'initialiser Φ simplement à une position prédéfinie dans l'image et on doit minimiser le nombre d'itérations à travers les voxels de la grille, notamment dans le calcul des moyennes. À cette fin, les moyennes $m_{in/out}$ et les compteurs $c_{in/out}$ sont sauvegardés dans les métadonnées de Φ pour économiser l'étape d'initialisation entre les itérations, et une méthode originale d'initialisation de Φ est proposée à l'algorithme 5.2.

L'algorithme 5.1 présente la version 3D proposée de l'algorithme de *Fast Level-Sets* multicanaux précédemment introduit en 2D dans [8]. Après l'initialisation des listes de bordure $L_{in/out}$, des valeurs moyennes $m_{in/out}$ et compteurs $c_{in/out}$ (lignes 8–13), l'algorithme commence par effectuer N_a cycles de segmentation (lignes 36–48) où les voxels sont permutés entre L_{in} et L_{out} en fonction de si F_{ext} indique qu'ils sont plus proche de la moyenne des voxels intérieurs m_{in} ou extérieurs m_{out} , ajustant ces moyennes à chaque permutation, avant de mettre à jour L_{in} , L_{out} et F_{ext} . On applique ensuite N_g cycles de lissage (lignes 49–59) où un noyau gaussien g de moyenne S_g et de diamètre W_g est appliqué aux voxels de L_{in} et L_{out} (lignes 52 et 57). Le tout est répété jusqu'à ce que la condition de fin (ligne 47) soit vraie ou que le nombre d'itérations maximal K ait été atteint.

En plus de l'initialisation de Φ , les changements principaux apportés par rapport à [8] concernent l'utilisation des coordonnées cartésiennes 3D au lieu de log-polaires 2D, la vérification des voxels vides, et la mise à jour en continu des moyennes m_{in} et m_{out} lorsque des voxels sont permutés entre les catégories plutôt que d'être recalculées pendant la mise à jour de F_{ext} . De plus, la flexibilité de l'algorithme a été améliorée en séparant le nombre de cycles de lissage N_g de la taille W_g du noyau gaussien utilisé et en utilisant des paramètres génériques pour les valeurs à affecter aux différentes régions de Φ . La sortie de l'algorithme est aussi la carte Φ complète plutôt que le contour L_{in} . On note également que $\Lambda_G(v)$ est utilisé de manière générique dans l'algorithme pour représenter la valeur du voxel v et soutient à la fois les valeurs simples et l'extraction d'un ou plusieurs canaux d'intérêt dans une grille à valeurs vectorielles.

Algorithme 5.1 *Fast Level-Sets* 3D

Entrées

- G , une grille de voxels à valeur vectorielle ou réelle
3: Φ , le résultat d'une itération précédente
 N_a, N_g, W_g, S_g, K , les paramètres de la segmentation
 $\beta = 1, \lambda = 3$, les valeurs à affecter à la bordure et à l'intérieur/extérieur

6: Sorties

Φ , une grille à valeur réelle au format *Level-Sets*

▷ initialisation des structures

- $L_{in} \leftarrow v \forall v \mid \Lambda_{\Phi}(v) = -\beta \wedge \Gamma_{\Phi}(v) = 1$
9: $L_{out} \leftarrow v \forall v \mid \Lambda_{\Phi}(v) = \beta \wedge \Gamma_{\Phi}(v) = 1$
 $m_{in} \leftarrow \text{MOYENNE}(\Lambda_G(v)) \mid \Lambda_{\Phi}(v) = -\lambda$
 $c_{in} \leftarrow |\{v \mid \Lambda_{\Phi}(v) = -\lambda\}|$
12: $m_{out} \leftarrow \text{MOYENNE}(\Lambda_G(v)) \mid \Lambda_{\Phi}(v) = \lambda$
 $c_{out} \leftarrow |\{v \mid \Lambda_{\Phi}(v) = \lambda\}|$
 $g(x, y, z) \leftarrow \exp\left(\frac{-0.5(x^2 + y^2 + z^2)}{S_g^2}\right), (x, y, z) \in [-W_g, W_g]$
15: $F_{ext} \leftarrow \text{UPDATE_FEXT}()$ ▷ Calcule $F_{ext}(p) \forall p \in L_{in} \cup L_{out}$ (équation (5.5))

Procédure UPDATE_LIN()

Pour tout $v \in L_{in}$, **faire**

- 18: **Si** $\Lambda_{\Phi}(n) < 0 \forall n \in \eta_0(v)$ **alors** ▷ voisinage eq. (4.8)
 $\Lambda_{\Phi}(v) \leftarrow -\lambda$; $\Gamma_{\Phi}(v) \leftarrow 0$
 $L_{in} \leftarrow v$
21: $u \leftarrow \Lambda_G(v)$
 $m_{in} \leftarrow \frac{m_{in} \cdot c_{in} + u}{c_{in} + 1}$
 $c_{in} \leftarrow c_{in} + 1$

24: Procédure UPDATE_LOUT()

Pour tout $v \in L_{out}$, **faire**

- Si** $\Lambda_{\Phi}(n) > 0 \forall n \in \eta_0(v)$ **alors** ▷ voisinage eq. (4.8)
27: $\Lambda_{\Phi}(v) \leftarrow \lambda$; $\Gamma_{\Phi}(v) \leftarrow 0$
 $L_{out} \leftarrow v$
 $u \leftarrow \Lambda_G(v)$
30: $m_{out} \leftarrow \frac{m_{out} \cdot c_{out} + u}{c_{out} + 1}$
 $c_{out} \leftarrow c_{out} + 1$
-

Algorithme 5.1 (suite) *Fast Level-Sets* 3D

```

     $k \leftarrow 0$  ;  $F_1 \leftarrow 0$ 
33: Tant que  $(k \leq K) \wedge \neg F_1$  , faire
     $k \leftarrow k + 1$ 
     $i \leftarrow 1$ 
36: Tant que  $i \leq N_a \wedge \neg F_1$  , faire ▷ Cycle de segmentation
     $i \leftarrow i + 1$ 
    Pour tout  $v \in L_{out}$  , faire
39: Si  $F_{ext}(v) > 0$  alors
        SWITCH_IN(v)
        UPDATE_LIN( )
42: Pour tout  $v \in L_{in}$  , faire
        Si  $F_{ext}(v) < 0$  alors
            SWITCH_OUT(v)
45: UPDATE_LOUT( )
        UPDATE_FEXT( )
        Si  $(F_{ext}(v) \leq 0 \forall v \in L_{out}) \wedge (F_{ext}(v) \geq 0 \forall v \in L_{in})$  alors
48:  $F_1 \leftarrow 1$  ▷ sortir du cycle de segmentation
    Pour  $i \leftarrow 1 \dots N_g$  , faire ▷ cycle de lissage
        Pour tout  $v \in L_{out}$  , faire
51:  $n \leftarrow \eta_{W_g}(v)$  ▷ voisinage de rayon  $W_g$ , eq. (4.8)
         $r \leftarrow \sum (\Lambda_\Phi(n) \cdot g(x, y, z))$  ▷ Produit élément par élément
        Si  $r < 0$  alors
54: SWITCH_IN(c)
        UPDATE_LIN( )
        Pour tout  $v \in L_{in}$  , faire
             $n \leftarrow \eta_{W_g}(v)$  ▷ voisinage de rayon  $W_g$ , eq. (4.8)
57:  $r \leftarrow \sum (\Lambda_\Phi(n) \cdot g(x, y, z))$  ▷ Produit élément par élément
            Si  $r > 0$  alors
                SWITCH_OUT(v)
                UPDATE_LOUT( )

```

Algorithme 5.1 (suite) Fast Level-Sets 3D

60: **Procédure** SWITCH_IN(v)

$$\Lambda_{\Phi}(v) \leftarrow -\beta \quad ; \quad \Gamma_{\Phi}(v) \leftarrow 1$$

$$L_{out} \stackrel{\ominus}{\leftarrow} v$$

63: $L_{in} \stackrel{\oplus}{\leftarrow} v$

Pour tout $y \in \eta_0(v)$, **faire**

▷ voisinage eq. (4.8)

Si $\Lambda_{\Phi}(y) = \lambda$ **alors**

66: $\Lambda_{\Phi}(y) \leftarrow \beta \quad ; \quad \Gamma_{\Phi}(y) \leftarrow 1$

$$L_{out} \stackrel{\oplus}{\leftarrow} y$$

$$u \leftarrow \Lambda_G(y)$$

69: $m_{out} \leftarrow \frac{m_{out} \cdot c_{out} - u}{c_{out} - 1}$

$$c_{out} \leftarrow c_{out} - 1$$

Procédure SWITCH_OUT(v)

72: $\Lambda_{\Phi}(v) \leftarrow \beta \quad ; \quad \Gamma_{\Phi}(v) \leftarrow 1$

$$L_{in} \stackrel{\ominus}{\leftarrow} v$$

$$L_{out} \stackrel{\oplus}{\leftarrow} v$$

75: **Pour tout** $y \in \eta_0(v)$, **faire**

▷ voisinage eq. (4.8)

Si $\Lambda_{\Phi}(y) = -\lambda$ **alors**

$$\Lambda_{\Phi}(y) \leftarrow -\beta \quad ; \quad \Gamma_{\Phi}(y) \leftarrow 1$$

78: $L_{in} \stackrel{\oplus}{\leftarrow} y$

$$u \leftarrow \Lambda_G(y)$$

$$m_{in} \leftarrow \frac{m_{in} \cdot c_{in} - u}{c_{in} - 1}$$

81: $c_{in} \leftarrow c_{in} - 1$

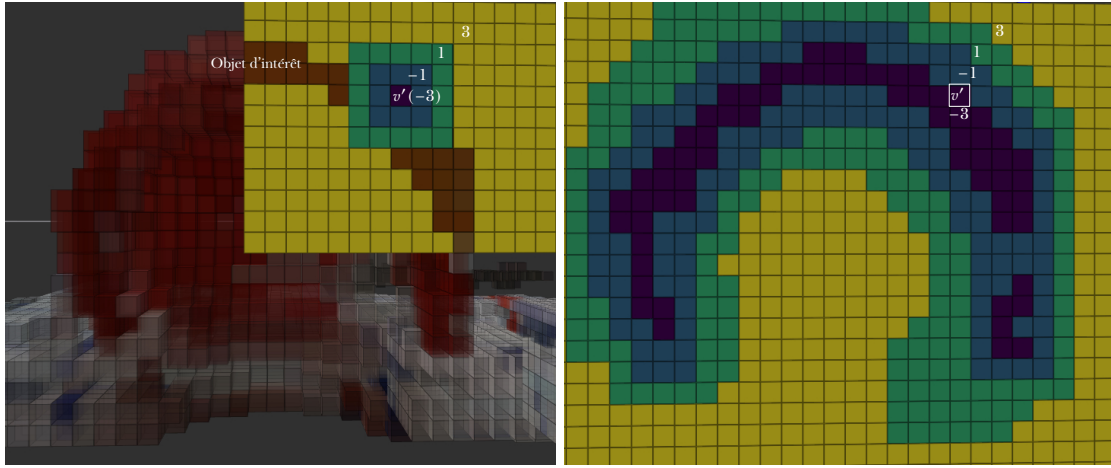
Initialisation de Φ Le *Fast Level-Sets* 3D demande obligatoirement une valeur initiale pour la grille Φ . Puisque la grille de voxels est infinie, cette valeur initiale ne peut pas être à une position fixe comme dans [8], mais doit être calculée à partir du point de départ p sélectionné par l'utilisateur pour identifier l'objet d'intérêt. Deux solutions originales sont proposées pour effectuer cette initialisation, soit à partir du seul point sélectionné, ou du résultat d'un autre algorithme de segmentation. Ces deux solutions sont décrites par l'algorithme 5.2 où $S(G, p)$ est le résultat d'un autre algorithme de segmentation appliqué sur la grille G à partir du point p (c-à-d. une pré-segmentation).

Algorithme 5.2 Initialisation de Φ

Entrées

- p , le point sélectionné par l'utilisateur sur l'objet d'intérêt
 - 3: G , la grille de voxels à segmenter
 - $v' \leftarrow \text{RECHERCHE}(G, p, C : \Gamma(v) = 1)$ ▷ algorithme 4.4
 - $T \leftarrow$ une grille vide semblable à G avec $\Gamma_T(v') = 1$ ou $S(G, p)$ ▷ choix de configuration
 - 6: $\Phi(v) \leftarrow 3 \forall v \in \mathcal{B}_T$
 - Pour tout** $v \in T \mid \Gamma(v) = 1$, **faire**
 - $\Phi(\eta_2(v)) \leftarrow 1$ ▷ voisinage eq. (4.8)
 - 9: $\Phi(\eta_1(v)) \leftarrow -1$
 - $\Phi(v) \leftarrow -3$
-

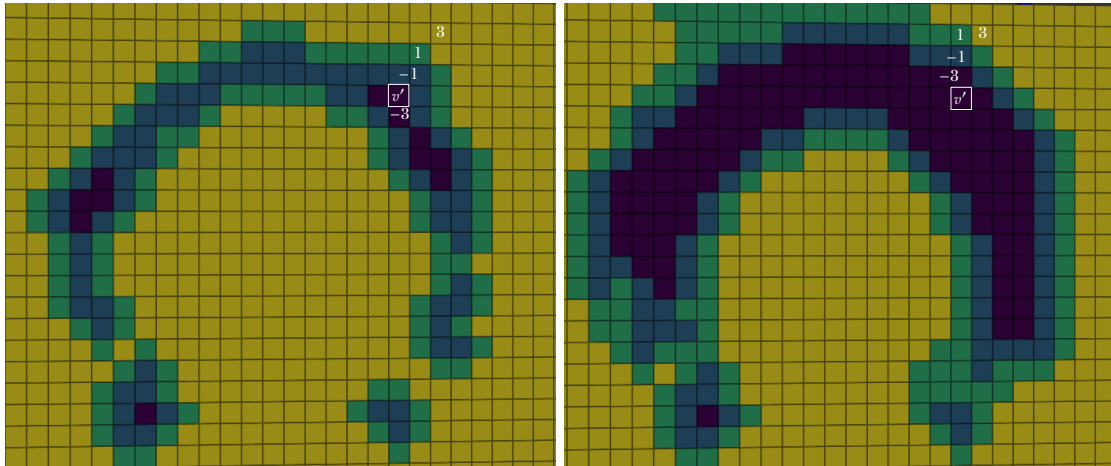
Dans cette procédure, on définit une grille T avec la même résolution que G et où $\Gamma_T(v) = 1$ pour les voxels qu'on sait initialement faire partie de l'objet d'intérêt (c-à-d. c ou $S(G, p)$). Φ est ensuite rempli avec la valeur « extérieure » (3) alors que les voxels initiaux où $\Gamma_T = 1$ reçoivent la valeur « intérieure » (−3) et sont entourés par un voxel de bordure intérieure (−1) et un voxel de bordure extérieure (1). Le résultat est illustré pour une tranche 2D à la figure 5.1, où 5.1a montre le résultat de l'initialisation à partir d'un seul voxel v' et 5.1b celui de l'initialisation avec une pré-segmentation selon l'équation (5.2). Les erreurs introduites par cette procédure, comme la définition de voxels de bordure à l'intérieur de l'objet, sont rapidement corrigées lorsqu'on applique l'algorithme de *Fast Level-Sets* 3D (figure 5.2). On note que le point p est nécessaire seulement lors de l'initialisation puisque le *Fast Level-Sets* 3D utilise ensuite la carte Φ mise à jour à chaque itération.



(a) Initialisé à partir de v' seulement

(b) Initialisé à partir d'une pré-segmentation avec l'équation (5.2)

FIGURE 5.1 : Illustration des deux méthodes pour l'initialisation de Φ sur une représentation voxelisée d'un objet sphérique (voxels rouges). Dans (a), la visualisation de Φ est tronquée pour laisser paraître l'objet d'intérêt à titre de référence.



(a) Initialisé à partir de v' seulement

(b) Initialisé à partir d'une pré-segmentation avec l'équation (5.2)

FIGURE 5.2 : Tranches de Φ selon la figure 5.1 après une itération de l'algorithme FLS

5.3 Algorithme hybride

Au vu des avantages et des inconvénients de la segmentation par couleur et du *Fast Level-Sets* 3D, il est pertinent de développer une stratégie de segmentation combinant les bénéfices de ces deux méthodes. Ainsi, la méthode de segmentation 3D hybride originale proposée part du même principe que la segmentation par couleur limitée aux voisins (équation (5.2)), mais remplace la comparaison directe avec le point de référence par une moyenne inspirée des principes du *Fast Level-Sets*.

En effet, dans le *Fast Level-Sets* 3D, l'inclusion ou l'exclusion d'un voxel est déterminée selon la fonction de vitesse F_{ext} . Cette fonction mesure essentiellement si la valeur d'un voxel est plus proche de la moyenne des voxels intérieurs ($F_{ext} > 0$) ou des voxels extérieurs ($F_{ext} < 0$), moyennes qui sont mises à jour à chaque itération. Ainsi, la segmentation par *Fast Level-Sets* classique est résistante aux variations de couleur et de texture de l'objet à extraire et, grâce au cycle de lissage avec flou gaussien, produit un contour lisse malgré les imperfections des données à segmenter. Cependant, ce lissage ainsi que le maintien des bordures intérieure et extérieure de l'objet mènent à l'inclusion de pixels (ou, en 3D, voxels) supplémentaires lors de la conversion en masque binaire. Bien que cet effet puisse être négligeable voire souhaitable dans le cas d'images à haute résolution, les dimensions physiques représentées par un voxel sont généralement beaucoup plus grandes que celles d'un pixel en raison des limitations des capteurs et du traitement. Dans cet environnement à basse résolution, les effets de lissage du *Fast Level-Sets* 3D peuvent introduire une erreur significative, qui peut notamment être observée en comparant la largeur des régions « intérieures » ($\Phi < 0$) des images de la figure 5.2 avec la coquille réelle de l'objet d'intérêt montrée à la figure 5.1a. Même si cette erreur peut être amoindrie en éliminant le cycle de lissage ($N_g = 0$ dans l'algorithme 5.1), le *Fast Level-Sets* 3D demande également un temps de calcul plus important que la segmentation par couleur limitée aux voisins (équation (5.2)), puisqu'il peut nécessiter plusieurs itérations avant de converger.

Partant de ces principes, les concepts de la méthode hybride proposée à l'algorithme 5.3, sont entre la segmentation par couleur limitée aux voisins et une seule itération du cycle de segmentation du *Fast Level-Sets* 3D. Plus formellement, cette méthode considère une grille de voxels G dont on cherche à extraire un masque d'objet d'intérêt M à partir d'un point p défini par l'utilisateur et d'un seuil de similarité τ . On commence par trouver le voxel v' tel que $\Gamma_M(v') = 1$ qui est le plus proche du point p avec l'algorithme 4.4. Comme dans cet algorithme, on définit les ensembles « exploré » E (initialement vide) et « frontière » F

(initialisé à $\{v'\}$). On définit aussi m_{in} et c_{in} qui, comme dans le *Fast Level-Sets* 3D, représentent respectivement la moyenne des valeurs des voxels faisant partie de l'objet d'intérêt et leur nombre. On choisit ensuite un voxel aléatoire $v \in F$ qu'on fait passer de F à E . La valeur (couleur) de ce voxel, $\Lambda_G(v)$, est comparée à m_{in} . Si leur différence est plus petite que τ , on met à jour m_{in} et c_{in} , on ajoute v au masque de l'objet d'intérêt en définissant $\Lambda_M(v)$ et $\Gamma_M(v)$ à 1, et on ajoute à F les voisins de v qui n'ont pas déjà été visités (non présents dans E). Cette procédure est répétée jusqu'à ce que F soit vide.

Algorithme 5.3 Algorithme de segmentation 3D hybride

Entrées

- G , la grille de voxels à segmenter
- 3: p , un point sur l'objet d'intérêt sélectionné par l'utilisateur
- τ , un seuil de similarité pour la segmentation

Sorties

- 6: M , un masque binaire pour l'objet d'intérêt
 - $M \leftarrow$ une grille vide de même résolution que G
 - $v' \leftarrow \text{RECHERCHE}(G, p, C : \Gamma = 1)$ ▷ alg. 4.4
 - 9: $E \leftarrow \emptyset$
 - $F \leftarrow \{v'\}$
 - $m_{in} \leftarrow \Lambda_G(v')$
 - 12: $c_{in} \leftarrow 1$
 - Tant que $F \neq \emptyset$, faire**
 - $v \leftarrow$ un élément de F
 - 15: $F \xleftarrow{\ominus} v$; $E \xleftarrow{\oplus} v$
 - Si $\|m_{in} - \Lambda_G(v)\| \leq \tau$ alors**
 - $m_{in} \leftarrow \frac{m_{in} \cdot c_{in} + \Lambda_G(v)}{c_{in} + 1}$
 - 18: $c_{in} \leftarrow c_{in} + 1$
 - $\Lambda_M(v) \leftarrow 1$; $\Gamma_M(v) \leftarrow 1$
 - $F \xleftarrow{\oplus} n \ \forall n \in \eta_1(v) \mid n \notin E$
-

5.4 Conversion en masque d'objet

Une fois la segmentation complétée, il convient de convertir le résultat de l'algorithme sélectionné en un format utilisable pour la suite des choses. Généralement, ceci est un masque binaire enregistré dans une carte M_m telle que :

$$\Lambda_{M_m}(v), \Gamma_{M_m}(v) = \begin{cases} 1 & \text{si } v \text{ fait partie de l'objet d'intérêt} \\ 0 & \text{autrement} \end{cases} \quad (5.6)$$

Les algorithmes de segmentation par couleur (section 5.1) et hybride (5.3) produisent directement un tel masque binaire. Dans le cas du *Fast Level-Sets 3D* (5.2), on doit effectuer une opération de seuillage sur Φ :

$$\Lambda_{M_m}(v), \Gamma_{M_m}(v) \leftarrow 1 \quad \forall v \in \mathcal{B}_\Phi \mid \Lambda_\Phi(v) < 0 \quad (5.7)$$

Pour soutenir l'incertitude dans la planification, il est pertinent de souhaiter un masque probabiliste plutôt que binaire. Pour ce faire, on peut définir la carte M_p telle que $\Lambda_{M_p}(v)$ indique la probabilité que le voxel v fasse partie de l'objet d'intérêt. Une telle carte peut être obtenue en utilisant le masque binaire M_m défini précédemment pour extraire la probabilité d'occupation des voxels correspondants avec l'équation (5.8) où M_o est une carte d'occupation probabiliste.

$$\begin{aligned} \Lambda_{M_p}(v) &\leftarrow \Lambda_{M_o}(v) \quad \forall v \in \mathcal{B}_{M_m} \\ \Gamma_{M_p}(v) &\leftarrow \Gamma_{M_m}(v) \quad \forall v \end{aligned} \quad (5.8)$$

5.5 Évaluation expérimentale

Cette section compare les résultats obtenus avec les algorithmes de segmentation présentés dans ce chapitre. On commence d’abord par discuter de la méthodologie utilisée pour cette comparaison (section 5.5.1) avant de comparer les résultats numériques de la segmentation de différents objets (section 5.5.2). S’ensuit une évaluation visuelle des performances de segmentation avec des objets pour lesquels il est difficile d’obtenir une référence numérique (section 5.5.3).

5.5.1 Méthodologie

Les algorithmes de segmentation sont testés avec une variété d’objets déformables réels montrés à la figure 5.3. Les nuages de points sont collectés avec la caméra RealSense D435i dans la configuration de la figure 3.1 en observant les scènes de test sous différents angles, puis agrégés dans des grilles de voxels à différentes résolution telles que montrées à la figure 5.4.

Pour obtenir une représentation numérique de la qualité de la segmentation, on construit des grilles de référence en voxélisant un modèle maillé avec la même forme et les mêmes dimensions que l’objet d’intérêt. Cette grille de référence est alignée avec le résultat de la segmentation par l’analyse des moments 3D (section 7.1) et les deux sont comparées avec une mesure d’intersection sur union (IoU) 3D au niveau des voxels (équation (5.9)). Puisque ceci représente une mesure de superposition des volumes et que les grilles segmentées représentent par nature des coquilles vides, elles sont remplies selon la méthode de la section 4.3.2 pour éviter qu’une petite erreur d’alignement ne cause une différence majeure dans le score IoU.

Étant données la faible précision des nuages de points produits par la caméra physique et la nature approximative des modèles de référence, on évalue d’abord la mesure IoU en prenant des mesures virtuelles sur une version simulée de la sphère (figure 5.3a) qui correspond exactement à sa forme de référence. Ceci donne lieu à la grille de voxels de la figure 5.5.

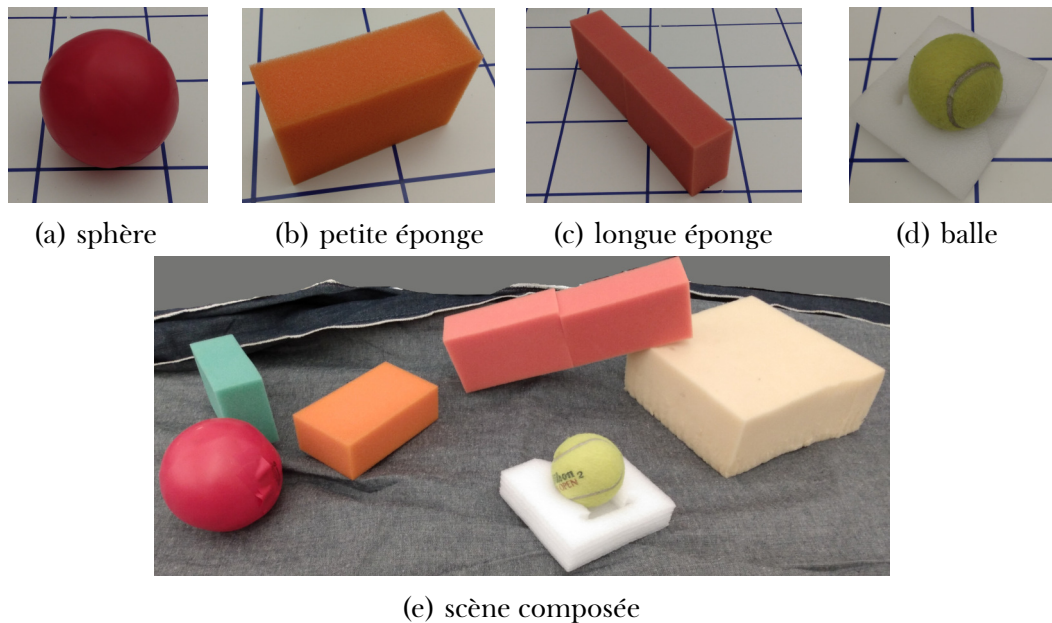


FIGURE 5.3 : Exemples d'objets utilisés pour tester les algorithmes de segmentation 3D

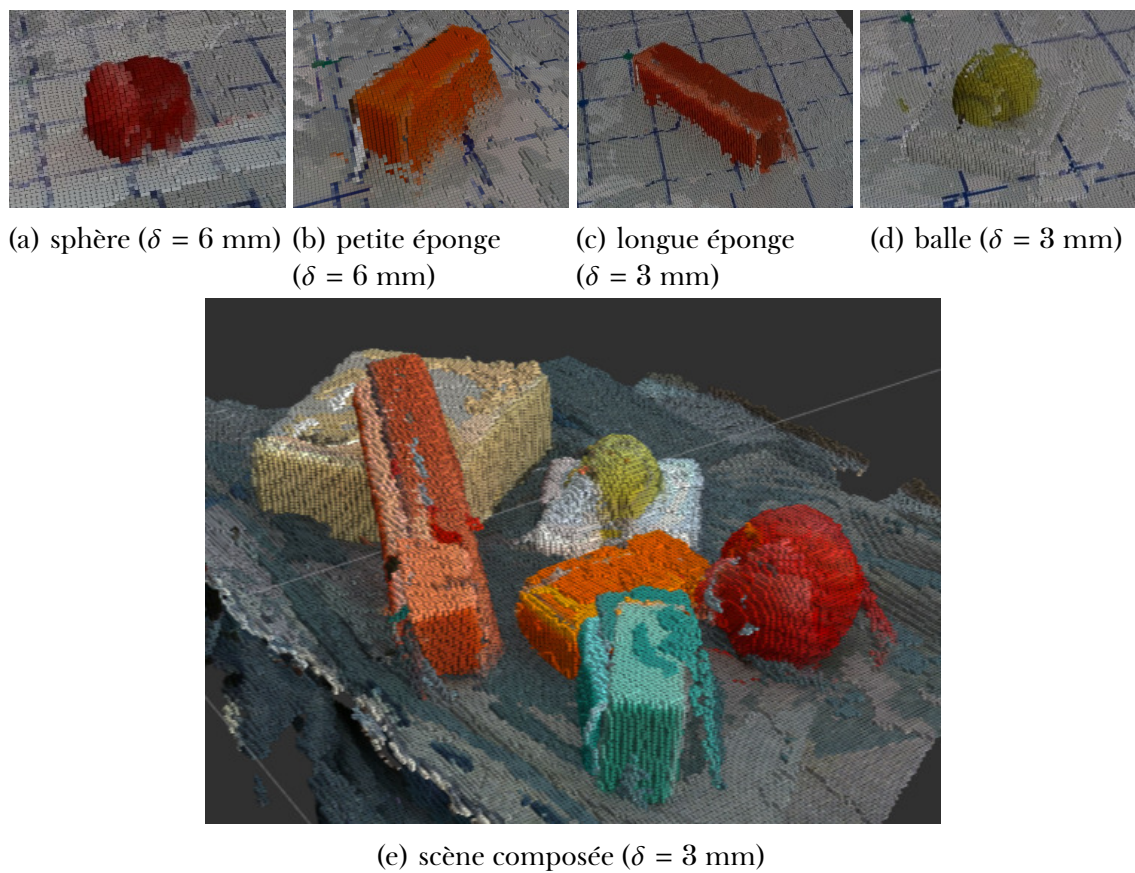


FIGURE 5.4 : Exemple de grilles de voxels pour la segmentation 3D avec différentes tailles de voxel δ

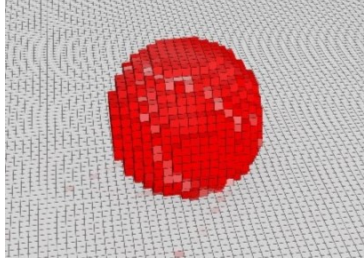


FIGURE 5.5 : Grille de voxels résultant de mesures simulées sur la sphère (5.3a) avec des voxels de 6 mm

Mesure IoU 3D En plus des mesures de comparaison détaillées qui seront présentées au chapitre 7, il est souhaitable d’avoir une mesure standardisée de la similarité des formes pour valider les différentes méthodes proposées. Celle-ci est fournie par la mesure d’intersection sur union (IoU), qui est communément utilisée pour comparer les algorithmes de segmentation d’objet 2D à une valeur de référence. Cette mesure est généralement définie par l’équation (5.9), où B est la boîte englobante réelle de l’objet et $\hat{B}$ est la boîte englobante détectée.

$$IoU = \frac{|B \cap \hat{B}|}{|B \cup \hat{B}|} \quad (5.9)$$

Ainsi, cette mesure est normalisée par la taille de l’objet à détecter et se rapproche de 1 à mesure que la qualité de la segmentation augmente. Pour évaluer les algorithmes de segmentation plus précis, on peut aussi l’utiliser pour comparer le masque binaire identifiant l’objet avec sa valeur de référence, auquel cas B dans (5.9) fait référence à ce masque plutôt qu’à la boîte englobante. Deux conditions sont nécessaires pour pouvoir appliquer cette mesure à des grilles de voxels 3D :

1. Convertir les voxels en valeurs binaires (inclus ou exclus).
2. Obtenir une valeur de référence pour le masque de l’objet.

Pour la première condition, on peut simplement utiliser l’état d’activité des voxels ainsi que leur proximité à la valeur d’arrière-plan pour déterminer s’ils sont inclus dans le masque ou non. La seconde condition est cependant plus difficile à remplir en 3D, puisqu’il n’est pas possible de manuellement sélectionner les voxels appartenant à l’objet comme c’est le cas pour les images. Il est donc nécessaire d’obtenir un modèle 3D de l’objet et de le convertir en carte voxélisée, comme on le fait pour définir la forme cible de la déformation. Naturellement, ce processus est plus adapté aux formes simples et aux objets simulés, pour lesquels on a le modèle 3D original, que pour les objets réels complexes, pour lesquels il peut être impossible d’obtenir un modèle 3D précis.

5.5.2 Comparaison de la performance de segmentation

Les algorithmes de segmentation basés sur la couleur requièrent la définition manuelle d'un seuil. Pour sélectionner une valeur appropriée, le score IoU pour différents algorithmes et seuils est représenté à la figure 5.6 pour quelques objets, ce qui permet de confirmer que la forme remplie permet une comparaison plus appropriée que le résultat direct de la segmentation (coquille vide). Pour le cas de la sphère simulée montré à la figure 5.6a, bien que le même modèle 3D soit utilisé pour créer les mesures simulées et la forme de référence, le score IoU demeure significativement sous la note parfaite de 1,0, avec des plateaux aux environs de 0,4 pour la coquille vide et 0,8 pour la forme remplie. Ceci s'explique par les différences dans le processus de voxélisation pour un modèle maillé et pour un nuage de points, ainsi que par la perte de précision inévitable dans la rotation et l'alignement de grilles voxélisées. Il est aussi intéressant de noter la forme des différentes courbes de la figure 5.6 : l'algorithme hybride présente une montée et une baisse plus rapide, avec un plateau plus plat que les autres versions. Ceci peut présenter des avantages en réduisant le besoin de micro-ajustements du seuil de segmentation.

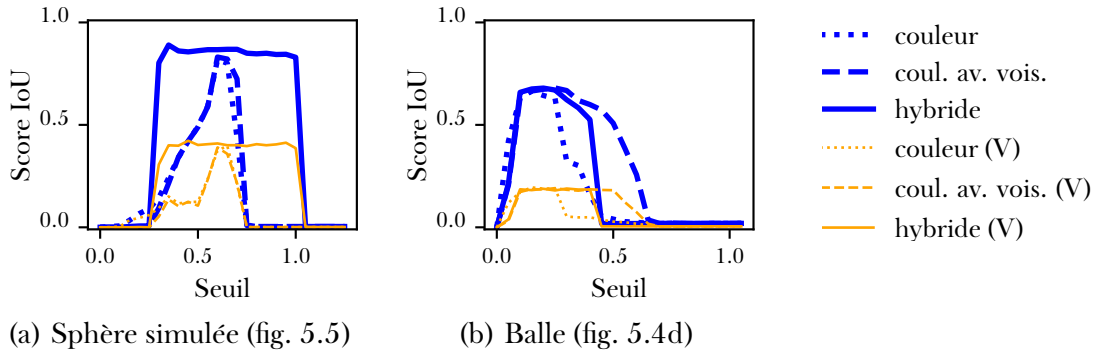


FIGURE 5.6 : Optimisation des seuils de segmentation RGB pour différentes grilles de voxels. Les lignes bleues présentent le score IoU pour la forme remplie alors que les lignes oranges (V) présentent celui pour les coquilles vides.

Les résultats des algorithmes de segmentation sur la sphère de la figure 5.3a sont présentés au tableau 5.1 pour les mesures simulées de la figure 5.5 et au tableau 5.2 pour les mesures physiques de la figure 5.4a. Comme le montre la comparaison des deux grilles de voxels avec l'objet réel, la sphère est nettement déformée dans la grille de voxels résultant des mesures physiques en raison des imprécisions des nuages de points. Cela se reflète dans le nombre de voxels et entraîne une baisse comparable des scores IoU. Il s'agit d'un problème récurrent qui complique la définition d'une valeur de référence appropriée pour la segmentation.

Dans le tableau 5.1 et les suivants, on compare sept variantes des algorithmes de segmentation. Il s'agit du seuillage par couleur (équation (5.1)), appelé « couleur », du seuillage par couleur avec recherche de voisins (équation (5.2)), appelé « couleur avec voisins », de l'algorithme « hybride » proposé à la section 5.3, et de quatre configurations de l'algorithme *Fast Level-Sets* 3D proposé à la section 5.2. Ces configurations initialisent Φ à partir du voxel initial uniquement (FLS-P, figure 5.1a) ou à partir d'une pré-segmentation avec l'algorithme hybride (FLS-H, figure 5.1b), et incluent ou non le cycle de lissage (FLS-S ou FLS-N). Cela conduit aux quatre combinaisons FLS-PS, FLS-HS, FLS-PN et FLS-HN. L'objectif d'initialiser l'algorithme FLS à partir de l'algorithme hybride plutôt qu'à partir de la segmentation simple par couleur (ou avec voisins) est de vérifier si l'application des multiples cycles de segmentation et de lissage du FLS permettent de raffiner la segmentation par rapport à l'utilisation directe de l'algorithme hybride.

Trois statistiques sont présentées pour chaque algorithme, soit le temps requis pour effectuer la segmentation, le nombre de voxels dans le résultat de la segmentation, et le score IoU entre le résultat et la référence. Pour les variantes du *Fast Level-Sets* 3D, le temps d'exécution est présenté deux fois : la première inclut le temps nécessaire pour le calcul initial des moyennes $m_{in/out}$ alors que la seconde est pour une deuxième passe de l'algorithme sur la grille segmentée.

TABLEAU 5.1 : Résultat des algorithmes de segmentation 3D sur la sphère simulée (fig. 5.3a) avec des voxels de 6 mm. La forme de référence a approximativement 760 voxels pour la coquille vide et 2880 voxels pour la forme remplie.

Méthode	Temps (s)	Coquille vide		Forme remplie	
		Voxels segmentés	IoU	Voxels segmentés	IoU
couleur	2,15	1249	0,3741	3225	0,8177
coul. av. vois.	0,012	1242	0,3901	3163	0,8310
hybride	0,015	1101	0,4094	2956	0,8670
FLS-PS	4,94 ; 0,25	41	0,0000	42	0,01466
FLS-HS	4,34 ; 0,17	3558	0,1990	4506	0,6300
FLS-PN	4,19 ; 0,013	1115	0,3742	2966	0,8543
FLS-HN	4,11 ; 0,015	3545	0,2003	4502	0,6304

TABLEAU 5.2 : Résultat des algorithmes de segmentation 3D sur la sphère réelle (fig. 5.3a) avec des voxels de 6 mm (forme remplie). La forme de référence a approximativement 2880 voxels.

Méthode	Temps (s)	Voxels segmentés	IoU
couleur	0,59	3778	0,6435
coul. av. vois.	0,017	3542	0,6738
hybride	0,021	3747	0,6795
FLS-PS	7,77 ; 0,69	2093	0,4772
FLS-HS	7,44 ; 0,56	5528	0,5059
FLS-PN	7,39 ; 0,032	4354	0,6082
FLS-HN	6,99 ; 0,047	6253	0,4434

Il est aussi intéressant de comparer les résultats pour un objet isolé dans sa scène avec ceux pour le même objet dans une scène plus complexe, avec différentes résolutions. Ceci est présenté dans les tableaux 5.3, 5.4 et 5.5 pour la longue éponge de la figure 5.3c par elle-même et dans la scène composée avec des voxels de 3 et 6 mm. Dans ces tableaux, on note que passer de la scène simple de la figure 5.4c (tableau 5.3) à la scène composée de la figure 5.4e avec la même résolution (tableau 5.4) augmente significativement le temps requis pour la segmentation avec le seuillage par couleur, de même que pour le calcul initial des moyennes avec le FLS — à l’exception de la variante FLS-PS, qui n’a pas convergé dans le cas de l’objet individuel mais a plutôt atteint le nombre maximal d’itérations. Cependant, on note qu’il y a peu d’impact sur le temps d’exécution pour l’algorithme de segmentation par couleur limité aux voisins, pour l’algorithme hybride et pour la deuxième passe des algorithmes FLS.

L’environnement de la scène composée est plus complexe et, en raison du plus grand nombre de points de vue nécessaires pour l’observer, comporte plus d’artefacts et d’erreurs de mesure. Ceci augmente la difficulté de segmentation et réduit la correspondance entre la forme de référence et les mesures réelles. Ces conditions se traduisent par une baisse du score IoU pour la plupart des algorithmes, notamment la segmentation par couleur et limitée aux voisins. En ce qui concerne l’impact de la résolution, de plus grands voxels (tableau 5.5) réduisent clairement le temps de calcul nécessaire pour tous les algorithmes. L’effet de filtrage supplémentaire résultant de plus gros voxels facilite aussi la segmentation pour certains algorithmes, en particulier le seuillage limité aux voisins et l’algorithme hybride. Cependant, la performance de plusieurs variantes du FLS est réduite, notamment celles qui incluent le cycle de lissage (variantes « S »). De plus, bien que différents algorithmes mènent

à des scores IoU plus élevés dans différents scénarios, on note que l'algorithme hybride se classe systématiquement parmi les meilleurs en termes de score IoU tout en demandant significativement moins de temps de calcul que le seuillage simple et toutes les variantes du FLS.

TABLEAU 5.3 : Résultats des algorithmes de segmentation 3D sur la longue éponge (forme remplie) avec des voxels de 3 mm. La forme de référence a approximativement 32 600 voxels.

Méthode	Temps (s)	Voxels segmentés	IoU
couleur	1,58	44 181	0,6190
coul. av. vois.	0,15	43 890	0,6250
hybride	0,18	44 194	0,6244
FLS-PS	109,60 ; 106,98	10 481	0,2074
FLS-HS	16,28 ; 6,55	60 961	0,5290
FLS-PN	24,36 ; 0,26	50 546	0,5082
FLS-HN	15,31 ; 0,23	66 384	0,4652

TABLEAU 5.4 : Résultats des algorithmes de segmentation 3D sur la longue éponge (forme remplie) dans la scène composée avec des voxels de 3 mm. La forme de référence a approximativement 32 600 voxels.

Méthode	Temps (s)	Voxels segmentés	IoU
couleur	2,41	46 989	0,1158
coul. av. vois.	0,10	22 208	0,3850
hybride	0,15	41 362	0,5000
FLS-PS	45,46 ; 3,88	25 885	0,4888
FLS-HS	40,66 ; 2,75	62 723	0,5018
FLS-PN	42,43 ; 0,20	47 543	0,5618
FLS-HN	39,37 ; 0,25	67 359	0,4615

TABLEAU 5.5 : Résultats des algorithmes de segmentation 3D sur la longue éponge (forme remplie) dans la scène composée avec des voxels de 6 mm. La forme de référence a approximativement 5000 voxels.

Méthode	Temps (s)	Voxels segmentés	IoU
couleur	0,931	8313	0,1572
coul. av. vois.	0,022	3925	0,4292
hybride	0,031	5970	0,5653
FLS-PS	5,89 ; 0,81	2199	0,3916
FLS-HS	5,84 ; 0,53	9980	0,5007
FLS-PN	5,76 ; 0,049	6848	0,6091
FLS-HN	5,66 ; 0,063	11 105	0,4486

Le tableau 5.6 présente les résultats de la segmentation de la balle jaune (fig. 5.4d) dans la scène composée. Encore une fois, on remarque le peu de temps d'exécution requis par la segmentation limitée aux voisins et l'algorithme hybride comparativement au seuillage simple et au *Fast Level-Sets* 3D. Dans cette scène, la balle a une couleur très distincte des autres objets et relativement peu de distortion, ce qui facilite la segmentation et la comparaison avec la forme de référence. Cependant, les variantes FLS ont une performance inférieure aux algorithmes plus simples. Le tableau 5.7 complète les résultats avec le score IoU des autres objets dans la scène composée avec des voxels de 6 mm. Ces résultats soulignent la faible performance des variantes FLS à basse résolution (p. ex., comparer la colonne « Balle » avec le tableau 5.6) ainsi que les défis liés à la distortion due au bruit des capteurs, qui est particulièrement évident pour les éponges bleue et orange dans la grille de la figure 5.4e.

TABLEAU 5.6 : Résultat des algorithmes de segmentation 3D sur la balle dans la scène composée avec des voxels de 3 mm (forme remplie)

Méthode	Temps (s)	Voxels seg.	Voxels ref.	IoU
couleur	1,79	7137	5724	0,5780
coul. av. vois.	0,025	6609	5729	0,5739
hybride	0,040	8731	5724	0,5538
FLS-PS	101,27 ; 1,37	2370	5734	0,3643
FLS-HS	102,17 ; 0,92	12 191	5743	0,4490
FLS-PN	101,85 ; 0,053	8403	5726	0,5722
FLS-HN	101,92 ; 0,068	13 032	5722	0,4149

TABLEAU 5.7 : Score IoU des algorithmes de segmentation 3D sur la scène composée avec des voxels de 6 mm (tous les objets, formes remplies). Voir le tableau 5.5 pour la longue éponge.

Méthode	Éponge bleue	Éponge orange	Sphère	Balle	Bloc
couleur	0,08361	0,2760	0,1552	0,2650	0,1358
coul. av. vois.	0,1484	0,3227	0,4343	0,5649	0,2578
hybride	0,1473	0,3646	0,5034	0,6303	0,2692
FLS-PS	0,02839	0,3310	0,1837	0,2564	0,1588
FLS-HS	0,3712	0,3581	0,3265	0,3297	0,3467
FLS-PN	0,3859	0,2982	0,2080	0,2342	0,2568
FLS-HN	0,5066	0,3009	0,2185	0,2212	0,3075

Parmi les douze cas présentés dans cette section, l’algorithme hybride obtient le meilleur score IoU dans cinq cas. La variante FLS-PN obtient le meilleur score dans trois cas, tandis que la segmentation par couleur, la segmentation par couleur limitée aux voisins, les variantes FLS-HS et FLS-HN obtiennent toutes le meilleur score une seule fois chacune. De plus, les algorithmes hybride et limité aux voisins sont nettement plus rapides que les algorithmes de segmentation simple par couleur et *Fast Level-Sets* 3D. L’algorithme hybride est également plus résistant aux changements de résolution et de complexité des scènes. Lorsque l’algorithme hybride est utilisé comme étape de pré-segmentation pour le *Fast Level-Sets* 3D afin d’initialiser Φ , l’étape FLS améliore la segmentation par rapport à l’algorithme hybride seul dans cinq cas : trois fois avec l’étape de lissage (FLS-HS) et deux fois sans (FLS-HN). Dans les cas où l’algorithme hybride donne un score IoU inférieur à celui d’un autre algorithme, il demeure toujours parmi les meilleurs résultats en termes de score et de temps de calcul.

5.5.3 Évaluation visuelle des performances

En plus de la comparaison numérique, on présente ci-dessous des images de grilles de voxels segmentées pour ancrer les scores IoU avec des observations qualitatives pratiques. Ceci permet aussi d'évaluer la performance de segmentation pour des objets complexes qui ne permettent pas de définir une forme de référence. La figure 5.7 montre les coquilles segmentées pour la sphère avec des voxels de 6 mm (figure 5.4a). On note que la segmentation par seuillage de couleur sélectionne des voxels superflus qui sont éliminés par l'algorithme hybride et la recherche limitée aux voisins. Dans ce cas, on remarque que le cycle de lissage du *Fast Level-Sets* 3D produit un résultat plus lisse lorsqu'une pré-segmentation est effectuée avec l'algorithme hybride (FLS-HS), mais que l'algorithme échoue complètement lorsqu'il est initialisé à partir d'un seul point (FLS-PS).

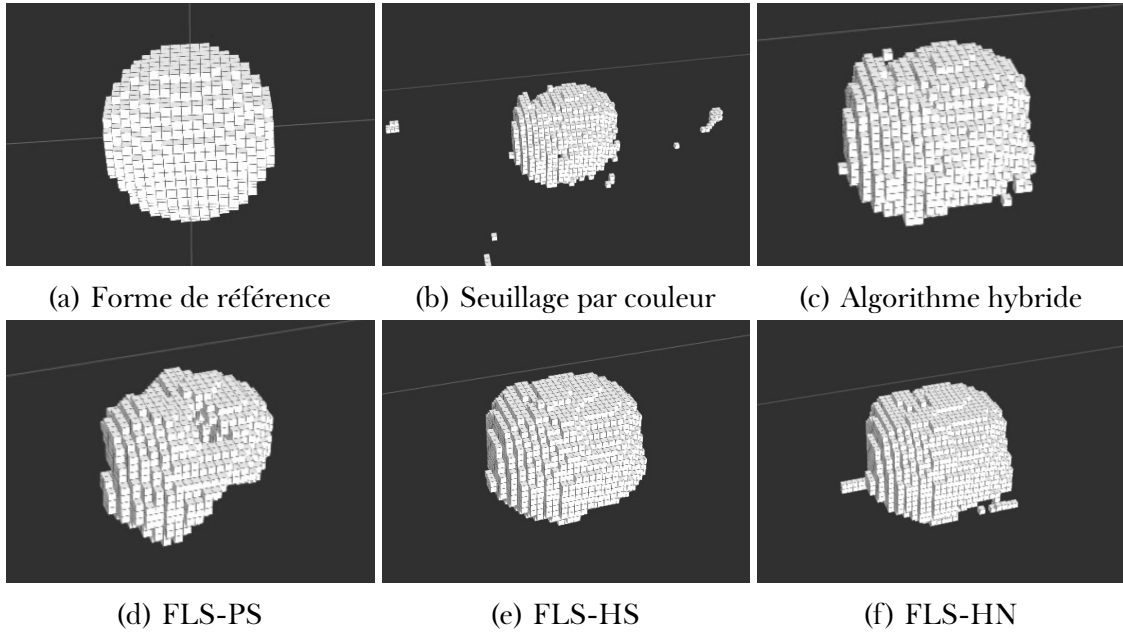


FIGURE 5.7 : Formes segmentées pour la sphère avec des voxels de 6 mm (fig. 5.4a)

La figure 5.8 présente les observations visuelles des coquilles vides correspondant à certains résultats du tableau 5.6. Visuellement, l'algorithme hybride obtient un meilleur résultat malgré un score IoU légèrement plus faible que le seuillage par couleur, qui peut avoir tiré un plus grand bénéfice des corrections introduites par l'algorithme de remplissage. Les résultats des algorithmes FLS-PS et FLS-PN sont cohérents avec leur faible score IoU, alors que l'algorithme FLS-HS complète la balle mais, comme les autres variantes FLS, ne peut la séparer de son support.

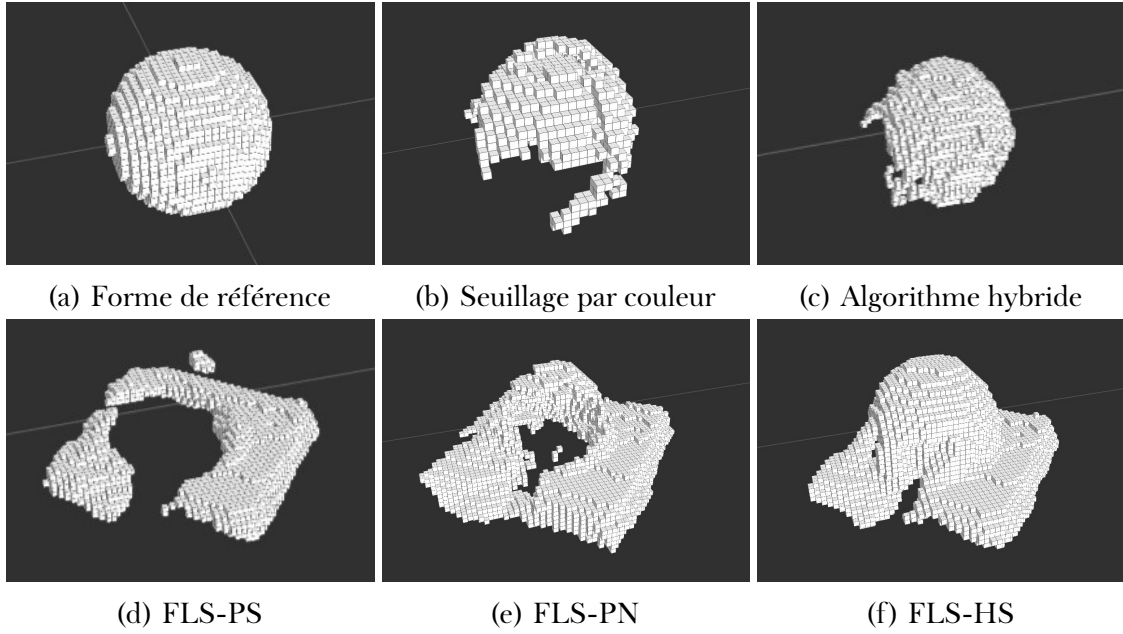


FIGURE 5.8 : Formes segmentées pour la balle dans la scène composée avec des voxels de 3 mm (figure 5.4e et tableau 5.6)

La figure 5.9 montre les résultats de la segmentation pour un objet informe obtenu en drapant une petite serviette sur un verre (figure 5.9a). Visuellement, les algorithmes hybride et de recherche limitée aux voisins obtiennent des résultats acceptables, alors que le seuillage par couleur inclut des voxels superflus et que les variantes FLS échouent à la tâche de segmentation.

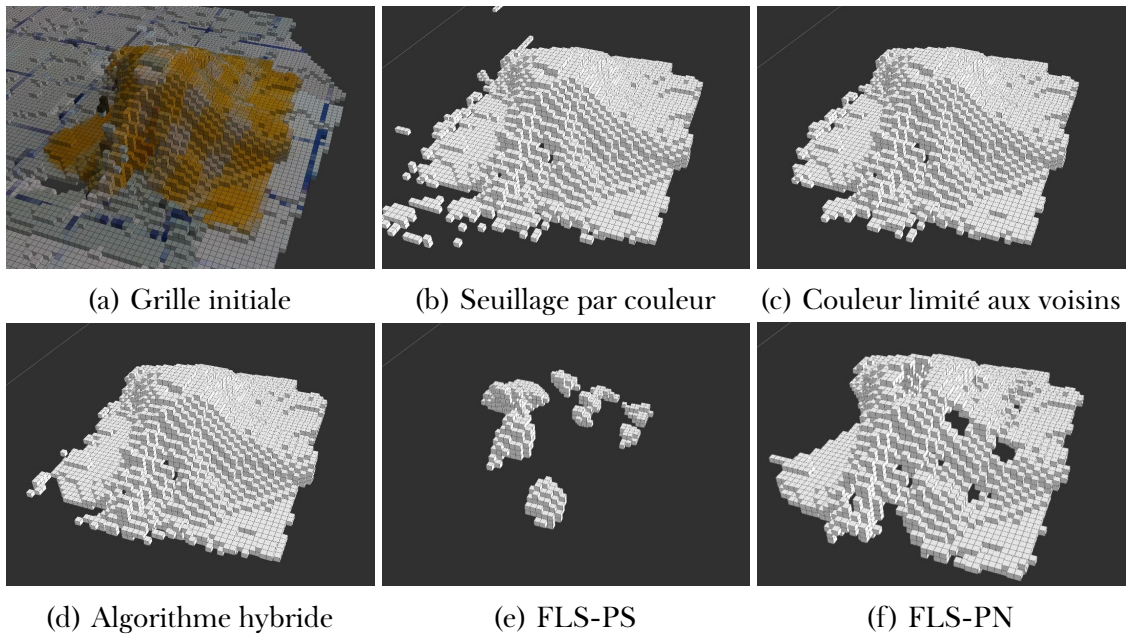


FIGURE 5.9 : Résultats de la segmentation 3D d'un objet complexe avec des voxels de 6 mm. les résultats pour FLS-HS et FLS-HN sont identiques à (a) et sélectionnent la grille entière

5.6 Sommaire

Ce chapitre propose et évalue une série d’algorithmes de segmentation dans le contexte de l’identification de la forme d’un objet d’intérêt dans un environnement voxélisé. En particulier, on compare la segmentation traditionnelle par seuillage de couleur telle qu’utilisée dans [67, 80, 81] avec une formulation 3D multi-canaux originale de l’algorithme *Fast Level-Sets* [8, 205]. Ayant observé les avantages et les inconvénients de ces algorithmes dans un environnement 3D discrétisé, on propose une approche hybride originale qui combine les avantages des deux classes d’algorithmes.

L’évaluation expérimentale montre que l’approche hybride proposée a une performance de segmentation supérieure au seuillage de couleur et au *Fast Level-Sets* 3D également proposé dans cette thèse, tout en restant rapide à calculer et sans dépendre de grands ensembles de données ou d’apprentissage machine coûteux. Dans les résultats, on note une quantité significative de bruit provenant de l’imprécision du capteur. Bien que les algorithmes présentés aient différents niveaux de résistance à ce bruit, il complique la définition d’une valeur de référence pour l’évaluation quantitative, puisque les voxels ne peuvent pas être sélectionnés manuellement aussi facilement que les pixels d’une image. Ainsi, la technologie des capteurs RGBD et les méthodes pour définir une valeur de référence pour la segmentation 3D sont des facteurs limitants dans l’évaluation de la segmentation pour les grilles de voxels.

Chapitre 6

Sélection du prochain point de vue pour la complétion de forme

Dans un environnement 3D partiellement observable, les points de vue fixes initiaux permettent seulement l'observation d'une partie de la forme de l'objet d'intérêt. Puisque la forme de l'objet est changeante et initialement inconnue, cette vue partielle ne permet pas de déduire la forme complète de l'objet, par exemple à partir d'une base de données de formes connues. Un mécanisme d'observation stratégique est donc nécessaire pour identifier les parties manquantes de l'objet d'intérêt et les observer. Ce chapitre présente le processus de sélection des points de vue permettant de compléter la forme de l'objet d'intérêt. Cette partie de la recherche a été publiée dans [195].

6.1 Heuristique de prochain point de vue

Après l'extraction de la partie déjà observée de l'objet, il est nécessaire de déterminer le prochain point de vue où positionner la caméra mobile montée sur le bras robotique pour compléter la forme initiale de l'objet, tel qu'illustré à la figure 6.1. Comme discuté à la section 2.2.1, les méthodes d'optimisation traditionnelles ou par apprentissage machine pour déterminer le point de vue optimal sont coûteuses en temps de calcul et en données, et les algorithmes de planification gagnent à être guidés par une mesure heuristique simple à calculer. Ainsi, on propose dans cette section une mesure heuristique originale pour guider un planificateur vers des points d'observation qui seront informatifs pour compléter la forme de l'objet d'intérêt. Cette mesure est développée pour être indépendante des spécificités de l'environnement de travail et des contraintes de positionnement des capteurs, ce qui permet de l'utiliser pour construire une variété d'algorithmes d'observation adaptés aux tâches à effectuer.

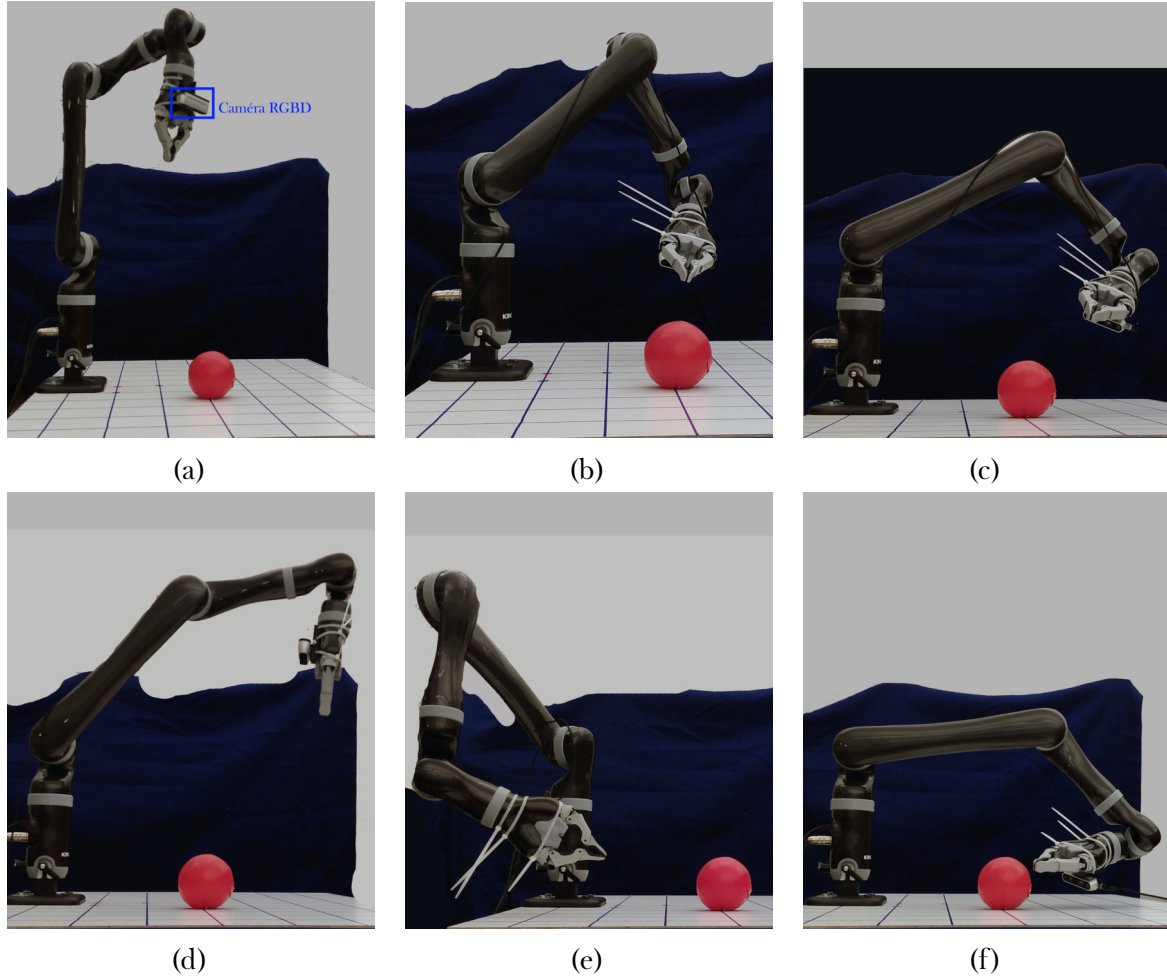


FIGURE 6.1 : Exemple de séquence de points d'observation où l'on déplace la caméra RGBD pour compléter la forme d'un objet d'intérêt

6.1.1 Critère d'intégralité de l'objet

Dans l'objectif d'observer la forme complète d'un objet initialement inconnu, il convient d'abord d'établir un critère pour déterminer si les données obtenues correspondent à la forme complète de l'objet d'intérêt ou s'il reste des parties non observées. Puisque l'objet est déformable et n'a pas nécessairement une forme connue au repos, il n'est pas possible de simplement comparer les mesures à une base de données de formes connues. De plus, les méthodes basées sur la densité des observations [99] sont peu commodes car elles requièrent de conserver la totalité des nuages de points provenant de la caméra et ne tiennent pas compte des observations erronées, deux contraintes qui sont simplifiées par l'encodage de la probabilité d'occupation sur une grille de voxels. La méthode proposée pour déterminer si l'objet d'intérêt est complet est donc basée sur l'identification des voxels inconnus (c'est-à-dire

pas suffisamment observés) dans la boîte englobante de l'objet d'intérêt. Ainsi, étant données une carte voxélisée M_m représentant le masque d'objet d'intérêt segmenté, de même qu'une carte M_o des probabilités d'occupation, M_m est considérée représenter la forme complète de l'objet d'intérêt s'il n'existe pas de voxel $v \in \mathcal{B}_{M_m}$ tel que $\tau_l \leq \Lambda_{M_o}(v) \leq \tau_h$, où τ_l , τ_h représentent les seuils inférieur et supérieur pour la probabilité d'occupation d'un voxel inconnu. En d'autres termes, tous les voxels dans $\mathcal{B}_{M_m}$ doivent être considérés occupés ou vides dans M_o .

Étant donné ce critère, deux opérations supplémentaires sont requises sur M_o et M_m pour améliorer la précision de la mesure. D'abord, l'intérieur de l'objet est impossible à observer mais devrait être considéré occupé plutôt qu'inconnu. Ainsi, on applique à M_m la procédure de remplissage décrite à la section 4.3.2 (algorithme 4.5), qu'on transfère à M_o avec le remplissage par masque 3D (algorithme 4.6). Ensuite, les zones de contact entre les différents objets présents dans l'environnement, par exemple entre l'objet d'intérêt et le plan de travail, créent des régions où il est difficile voire impossible d'observer la surface de ces objets. Il est donc nécessaire de traiter ces régions pour pouvoir estimer la forme de l'objet. Plutôt que d'introduire la supposition d'une surface de travail horizontale plane et de la supprimer directement par détection de plan, on propose une approche plus flexible qui maintient l'applicabilité du module de cartographie à différents scénarios.

Soient la carte M_m qui définit un masque d'objet d'intérêt tel que $\Gamma_{M_m}(v) = 1$ indique que le voxel v fait partie de l'objet représenté par M_m , et la carte M_o qui définit une grille d'occupation telle que $\Gamma_{M_o}(v)$ indique que le voxel v est considéré solide/occupé. M_m a idéalement été préalablement remplie par l'algorithme 4.5. Si l'objet représenté par M_m a des intersections avec d'autres éléments de l'environnement, la boîte englobante de M_m , $\mathcal{B}_{M_m}$, peut contenir des voxels qui sont considérés occupés dans M_o mais ne font pas partie de l'objet représenté par M_m (figure 6.2).

Selon la forme des objets et leur alignement avec les axes, il est possible que ces régions ne soient pas remplies proprement par l'algorithme 4.5 ou causent des erreurs d'interprétation par rapport à l'appartenance ou non de ces voxels à M_m . Pour éviter ces erreurs, on propose d'éliminer de M_m les voxels qui pourraient être interprétés comme faisant partie d'un objet voisin. La procédure originale proposée, décrite à l'algorithme 6.1, consiste à créer une grille T contenant les voxels des objets voisins de M_m , c'est-à-dire ceux qui sont considérés occupés dans M_o mais ne font pas partie de l'objet dans M_m . On applique ensuite la procédure de remplissage à T avant de supprimer de M_m les voxels présents dans T . Cette procédure a pour effet de définir comme ne faisant pas partie de M_m les voxels dont l'appartenance à l'objet est incertaine en raison d'une intersection.

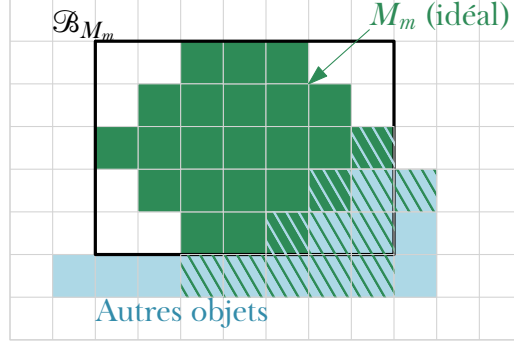


FIGURE 6.2 : L'appartenance des voxels hachurés à l'objet d'intérêt est incertaine, par exemple en raison de la non-observabilité, de la qualité de la segmentation et du remplissage, et d'erreurs numériques. La procédure de suppression des intersections vise à les exclure de l'objet d'intérêt.

Algorithme 6.1 Suppression des intersections

Entrées

M_o , la carte d'occupation où $\Gamma_{M_o}(v) = 1$ indique que le voxel v est occupé

3: M_m , le masque de l'objet d'intérêt où $\Gamma_{M_m}(v) = 1$ indique que v fait partie de l'objet

$T \leftarrow$ une grille vide semblable à M_m

Pour tout $v \in \mathcal{B}_{M_m}$, **faire**

6: $\Gamma_T(v) \leftarrow \Gamma_{M_o}(v) \wedge \neg \Gamma_{M_m}(v)$

$T \leftarrow \text{REPLISSAGE}(T)$

▷ algorithme 4.5

Pour tout $v \in \mathcal{B}_{M_m} \mid \Gamma_T(v) = 1$, **faire**

9: $\Lambda_{M_m}(v) \leftarrow \emptyset$; $\Gamma_{M_m}(v) \leftarrow 0$

Exemple de suppression des intersections Les figures 6.3 à 6.5 montrent l'utilité de la procédure de suppression des intersections. À la figure 6.3, l'imprécision des mesures et les artefacts de segmentation font en sorte que la boîte englobante de l'objet segmenté inclut des voxels inconnus et inobservables qui sont sous le plan de travail, ce qui empêche de confirmer que l'objet est complet. Sans introduire la supposition d'une surface de travail horizontale et plane, les intersections entre l'objet d'intérêt et les objets voisins sont identifiées à la figure 6.4 avant et après qu'on leur applique la procédure de remplissage (section 4.3.2). La suppression des intersections « remplies » de l'objet d'intérêt résulte en la figure 6.5, où on constate que la boîte englobante de l'objet d'intérêt ne dépasse plus sous le plan de travail et n'inclut plus les voxels inconnus non observables, ce qui permet de valider que la forme complète de l'objet a été observée.

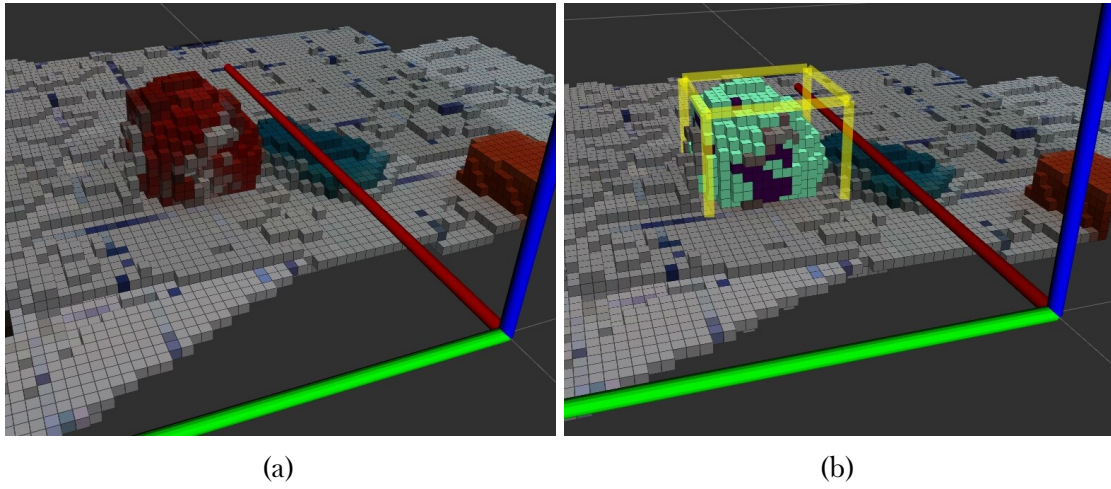


FIGURE 6.3 : État initial pour la suppression des intersections avec la scène colorée (a) et l'objet segmenté avec sa boîte englobante (b), qui déborde sous le plan de travail.

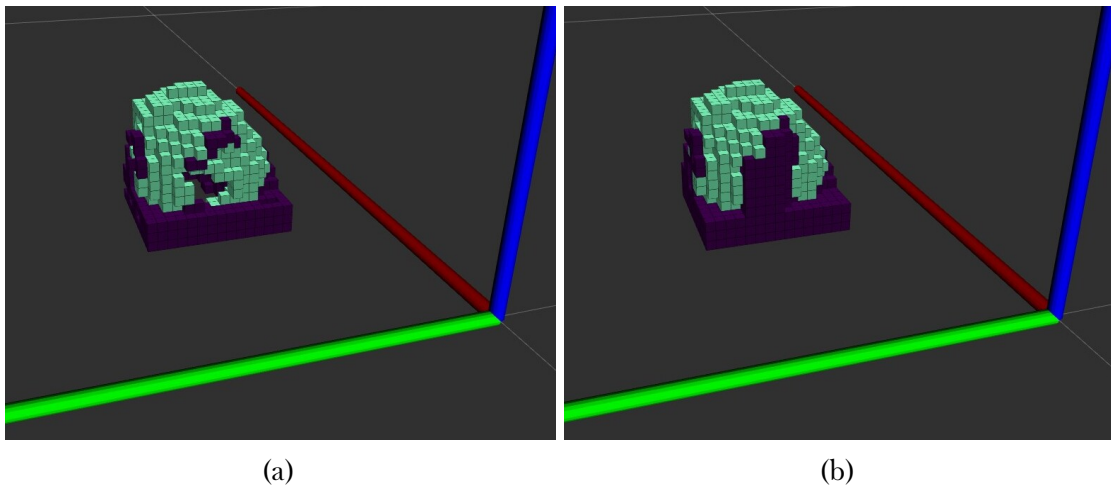


FIGURE 6.4 : Objet segmenté (vert) et intersections (mauve). Les intersections identifiées en (a) sont remplies en (b)

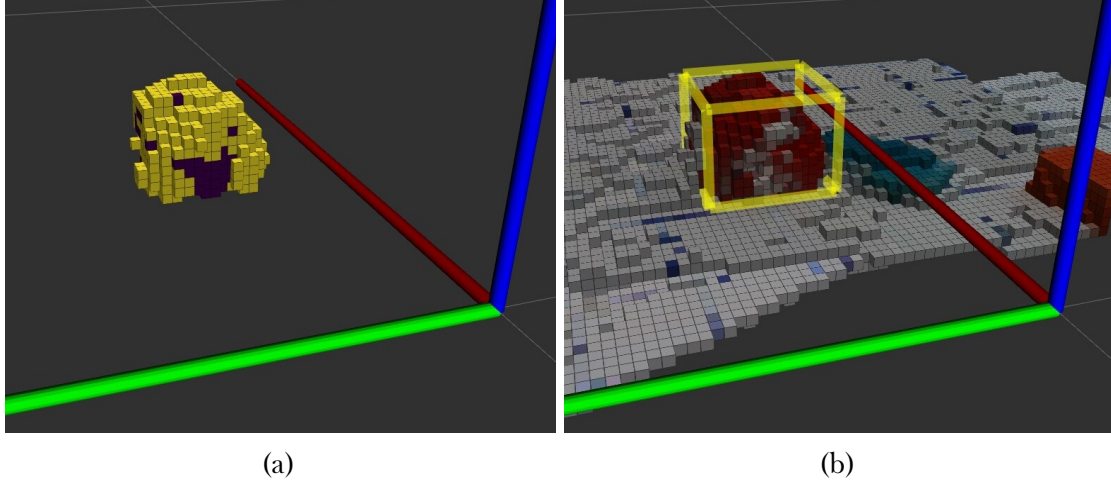


FIGURE 6.5 : Résultat de suppression des intersections : Masque d'objet (a) et boîte englobante sur la carte colorée (b), qui n'inclut plus de voxels superflus. Les voxels mauves en (a) représentent des voxels remplis dans M_m et non T .

6.1.2 Calcul du prochain point de vue

Étant donné le critère défini pour déterminer si l'objet est complet, la mesure originale proposée pour la sélection du prochain point de vue est basée sur l'analyse des parties manquantes de l'objet d'intérêt. Considérant un environnement de travail potentiellement infini, l'objectif n'est pas d'optimiser une séquence de points de vue pour observer l'environnement, mais de rapidement calculer une mesure indiquant quelles régions devraient être observées pour compléter la forme de l'objet d'intérêt et ainsi pouvoir planifier sa déformation. Le principe de base consiste à identifier les voxels non observés faisant partie de la boîte englobante de l'objet d'intérêt puis, en utilisant l'analyse des moments 3D, de déterminer un angle d'observation qui fait en sorte qu'un maximum de ces voxels seront dans le plan d'image de la caméra RGBD mobile. Étant donné le fonctionnement de la caméra et la projection 3D à 2D (ou 2.5D) qui en résulte, on déduit que le plan d'image doit être placé parallèlement au plan le mieux ajusté au groupe de voxels à observer, ce qui revient à aligner l'axe optique de la caméra avec le 3e axe principal de ce groupe de voxels.

Cette idée de calculer un vecteur d'observation perpendiculaire aux régions à observer ressemble à celle utilisée dans [100, 101]. Cependant, ceux-ci utilisent un vecteur normal au gradient de la surface en bordure des régions explorées. Par contraste, notre mesure propose des vecteurs normaux au plan le mieux ajusté des groupes de voxels non explorés proches de l'objet d'intérêt. Bien que cette mesure ne considère pas directement les occlusions (ce qui demande une procédure de reprojection plus coûteuse) et n'ait pas de garantie formelle

d'optimalité en termes de la minimisation du nombre de points de vue nécessaires pour observer l'objet, elle est rapide à calculer, indépendante du robot et de la caméra utilisés, et permet d'obtenir un point de départ pour guider un algorithme de planification plus complet prenant en compte les besoins spécifiques de l'application. La figure 6.6 présente l'enchaînement des algorithmes permettant d'appliquer cette mesure pour calculer une série de points de vue potentiels pour observer les voxels manquants dans l'objet d'intérêt.

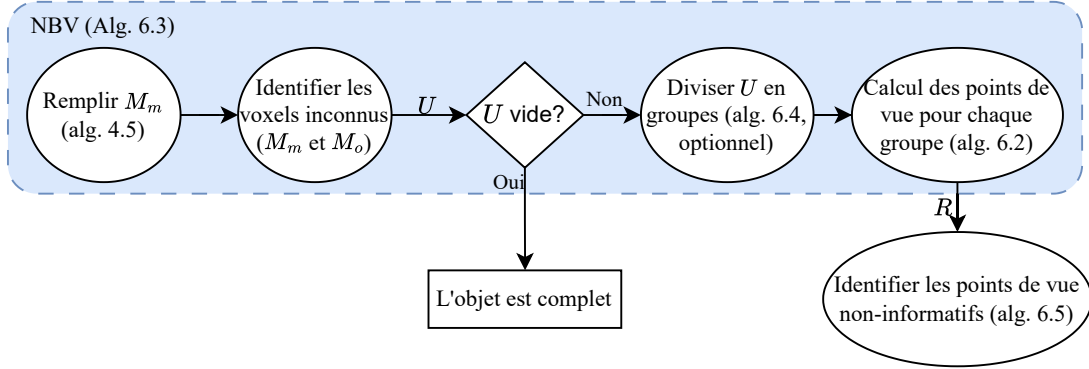


FIGURE 6.6 : Séquence d'algorithmes pour calculer les points d'observation potentiels.

Ainsi, l'algorithme 6.2 décrit la procédure originale proposée pour calculer les points de vue pour observer un groupe de voxels dont l'état d'occupation est inconnu. Étant donnée une grille de voxels U où $\Gamma_U(v) = 1$ indique que le voxel v est inconnu dans la boîte englobante de l'objet d'intérêt $\mathcal{B}_{M_m}$, l'analyse des moments 3D permet de déterminer le centroïde et les axes principaux de U (annexe A). Un point d'observation convenable est alors de diriger l'axe optique de la caméra vers ce centroïde en le gardant parallèle au 3e axe principal (c'est-à-dire perpendiculaire au plan le mieux ajusté à U). La grille des voxels inconnus U peut être construite simplement en définissant $\Gamma_U(v) = 1$ pour tous les voxels $v \in \mathcal{B}_{M_m}$ dont la probabilité d'occupation Λ_{M_o} indique un voxel inconnu ou non observé. Cependant, ceci mène à un seul point de vue suggéré, ce qui demandera de recalculer la mesure entre chaque observation pour obtenir le prochain point de vue. Pour améliorer l'efficacité et calculer un plus grand nombre de points de vue plus spécifiques, U peut être divisé en groupes de voxels contigus (algorithme 6.4), permettant d'appliquer l'algorithme 6.2 à chacun de ces groupes. Le principe de l'algorithme 6.2 consiste donc à sélectionner les axes principaux du groupe de voxels en ordre inverse (c'est-à-dire que le 3e axe principal est le meilleur point de vue, alors que le 2e et le 1er sont conservés comme options de rechange). Cependant, puisque les axes principaux ne sont pas orientés (annexe A), il est nécessaire de définir un critère pour sélectionner l'orientation appropriée. Le critère proposé est comme suit : Soit p et p' les points de vue obtenus en se déplaçant d'une distance égale quelconque (p. ex., 1 m) dans les deux

sens possibles de l'axe considéré à partir du centroïde k du groupe de voxels à observer. On choisit entre p et p' le point qui est le plus éloigné du centroïde c des voxels connus de l'objet, puisqu'il sera du côté vide/inconnu de celui-ci. Ceci est illustré à la figure 6.7.

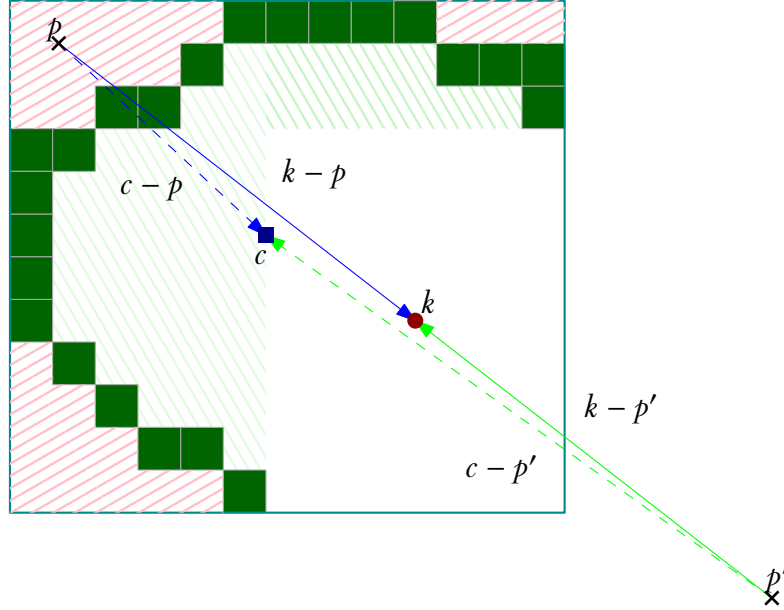


FIGURE 6.7 : Sélection du sens de l'axe principal pour le prochain point de vue. Les voxels verts sont la coquille de l'objet d'intérêt M_o et le grand cadre turquoise représente la boîte englobante de cet objet $\mathcal{B}_{M_o}$. La région hachurée rose (vers le haut) est connue et vide, alors que la région hachurée verte (vers le bas) est remplie par l'algorithme 4.5 et la région vide est inconnue. Les points d'observation potentiels p et p' sont situés à égale distance du centroïde k de la région inconnue que l'on vise à observer ($\|k - p\| = \|k - p'\|$) le long du 3e axe principal de celle-ci. Puisque p' est plus éloigné du centroïde c des voxels connus ($\|c - p\| < \|c - p'\|$), $k - p'$ est le vecteur d'observation préférable.

Pour faciliter le transfert et l'interprétation des résultats entre les différentes parties du système de manipulation robotique et les algorithmes 6.2, 6.3 et 6.5, on définit une structure de données pour encapsuler l'analyse du prochain point de vue pour un groupe de voxels inconnus :

NBVChunk :

- taille** : le nombre de voxels dans le groupe
- cible** : le point sur lequel centrer les observations
- vues** : une liste de points de vue pour observer le groupe
- vues_inverses** : la liste des points de vue « opposés » à **vues**
- vues_informatives** : une liste booléenne indiquant si on prévoit que les points de vue seront utiles

Algorithme 6.2 Calcul du meilleur point de vue pour un groupe de voxels inconnus

Procédure NBV_PARTIEL(G, c)

Entrées

- 3: G , une grille de voxels telle que $\Gamma_G(v) \mid v \in \mathcal{B}_{M_o}$ implique que v est un voxel inconnu
 c , le centroïde de l'objet d'intérêt

Sorties

- 6: R , une structure de type NBVChunk
calculer les moments 3D de G et en extraire les propriétés ▷ annexe A
 $k \leftarrow$ le centroïde de G
- 9: $A \leftarrow$ les trois axes principaux de G , en ordre inverse
 $V, V' \leftarrow \emptyset$
- Pour tout** $a \in A$, **faire**
- 12: $a' \leftarrow -a$
 $o_1 \leftarrow a + k$
 $o_2 \leftarrow a' + k$
- 15: **Si** $\|o_1 - c\| > \|o_2 - c\|$ **alors**
 $V \leftarrow_{\oplus} o_1$ et $V' \leftarrow_{\oplus} o_2$
Sinon
- 18: $V \leftarrow_{\oplus} o_2$ et $V' \leftarrow_{\oplus} o_1$
- $R.taille \leftarrow |\{v \in G \mid \Gamma_G(v) = 1\}|$ ▷ le volume de G
 $R.cible \leftarrow k$
- 21: $R.vues \leftarrow V$
 $R.vues_inverses \leftarrow V'$
-

L'algorithme principal pour déterminer les prochains points de vue potentiels est présenté à l'algorithme 6.3 et regroupe différentes options pour calculer les points de vue à partir d'une grille représentant le masque de l'objet d'intérêt ainsi que de la carte d'occupation. En particulier, on peut définir si on souhaite identifier les voxels inconnus de manière probabiliste (en définissant un intervalle de probabilités d'occupation pour considérer un voxel inconnu) ou binaire (basé sur l'état d'activité de la carte d'occupation). De plus, il est possible de diviser ou non les voxels inconnus de l'objet en plusieurs groupes, permettant ainsi d'obtenir un plus grand nombre de points de vue potentiels, lesquels sont aussi plus spécifiques. Enfin, on peut choisir d'effectuer ou non le remplissage de la grille du masque d'objet.

Ainsi, à partir de la carte M_m représentant l'objet à observer et de la carte d'occupation M_o , cet algorithme crée une grille temporaire U à valeurs booléennes où tous les voxels de la boîte englobante des voxels actifs de M_m , $\mathcal{B}_{M_m}$ qui sont considérés inconnus dans M_o sont définis comme actifs. Si U ne contient aucun voxel actif après cette étape, alors l'objet est considéré comme étant complètement observé. Si U contient des voxels actifs, on les divise optionnellement en groupes avec l'algorithme 6.4 avant d'appliquer l'algorithme 6.2 à chaque groupe pour calculer les points de vue potentiels pour observer ce groupe.

Algorithme 6.3 Calcul des prochains points de vue

Procédure NBV(M_m, M_o, X)

Entrées

- 3: M_m , le masque de l'objet à observer
 M_o , la carte d'occupation
 $X(v)$, un opérateur qui détermine si le voxel v est inconnu, p. ex. $\tau_l < \Lambda_{M_o}(v) < \tau_h$

6: **Sorties**

R , une liste de structures **NBVChunk** décrivant des points de vue potentiels

 Appliquer l'algorithme 4.5 pour compléter M_m ▷ optionnel

- 9: $U \leftarrow$ une grille de voxels semblable à M_m

$\Gamma_U(v) \leftarrow 1 \ \forall v \in \mathcal{B}_{M_m} \mid X_{M_o}(v) = 1$

Si $\Gamma_U(v) = 0 \ \forall v \in U$ **alors**

- 12: $R \leftarrow \emptyset$ ▷ L'objet est complet

Sinon

$c \leftarrow$ le centroïde de M_m ▷ annexe A

- 15: $G \leftarrow \{U\}$ ou **DIVISER**(U) ▷ Algorithme 6.4, selon la configuration

$R \leftarrow \text{NBV_PARTIEL}(M, c) \ \forall M \in G$ ▷ algorithme 6.2

L'algorithme 6.4, pour sa part, est responsable de diviser la grille des voxels inconnus U en ses différents groupes contigus. Ceci permet d'obtenir un plus grand nombre de points de vue plus spécifiques pour observer l'objet, en particulier lorsqu'il est déjà presque complet. Cet algorithme est basé sur le même principe d'exploration décrit à l'algorithme 4.4 : on sélectionne un voxel actif quelconque dans la grille de voxels inconnus U , et on l'ajoute ainsi que ses voisins actifs à une nouvelle grille C en les retirant de U . On poursuit la recherche ainsi jusqu'à ce qu'il n'y ait plus de voxels actifs connectés au voxel original, puis on recommence à partir d'un autre point de départ. Les groupes d'un seul voxel sont ignorés car ceux-ci ne sont pas informatifs dans le calcul des points de vue, et la liste des groupes de voxels résultante de l'algorithme est triée en ordre décroissant du nombre de voxels actifs (c'est-à-dire le champ taille de la structure NBVChunk correspondante) pour que les points de vue observant un plus grand nombre de voxels soient considérés en premier.

Algorithme 6.4 Division en groupes des voxels inconnus

Procédure DIVISER(U)

Entrées

3: U , une grille de voxels où $\Gamma_U = 1$ pour les voxels inconnus

Sorties

L , une liste de grilles de voxels contenant les différents groupes de voxels inconnus

6: **Tant que** $\neg(\Gamma_U(v) = 0 \forall v)$, **faire**

$C \leftarrow$ une grille de voxels semblable à U

$v \leftarrow v \in U \mid \Gamma_u(v) = 1$ choisi aléatoirement

9: $F, E \leftarrow \emptyset$

$F \leftarrow v$
 $\oplus$

Tant que $F \neq \emptyset$, **faire**

12: $p \leftarrow p \in F$ choisi aléatoirement

$E \leftarrow p$
 $\oplus$

Si $\Gamma_U(p) = 1$ **alors**

15: $\Gamma_C(p) \leftarrow 1$

$\Gamma_U(p) \leftarrow 0$

$F \leftarrow n \forall n \in \eta_1(p) \mid n \notin E$
 $\oplus$

18: **Si** $\|C\| > 1$ **alors** ▷ on ignore les groupes contenant un voxel seul

$L \leftarrow C$
 $\oplus$

trier L en ordre décroissant du nombre de voxels actifs

Vérification des vues informatives Comme discuté à la section 3.2.1, l'imprécision des mesures de distance de la caméra dans un modèle d'occupation probabiliste mène à une zone d'incertitude autour de la surface de l'objet. Cette zone ne peut pas être précisée au-delà d'un certain point en augmentant le nombre de prises de vues, donc un point de vue où l'axe optique de la caméra traverse cette zone ne sera pas informatif. Les voxels inconnus dans cette zone peuvent être différenciés de ceux résultant d'observations manquantes par le fait qu'ils forment une mince couche entre les voxels occupés représentant l'objet et les voxels inoccupés à l'extérieur de celui-ci, comme montré à la figure 6.8. Cette condition peut être vérifiée par une opération de reprojection sur une courte distance le long du vecteur d'observation, comme décrit dans l'algorithme 6.5.

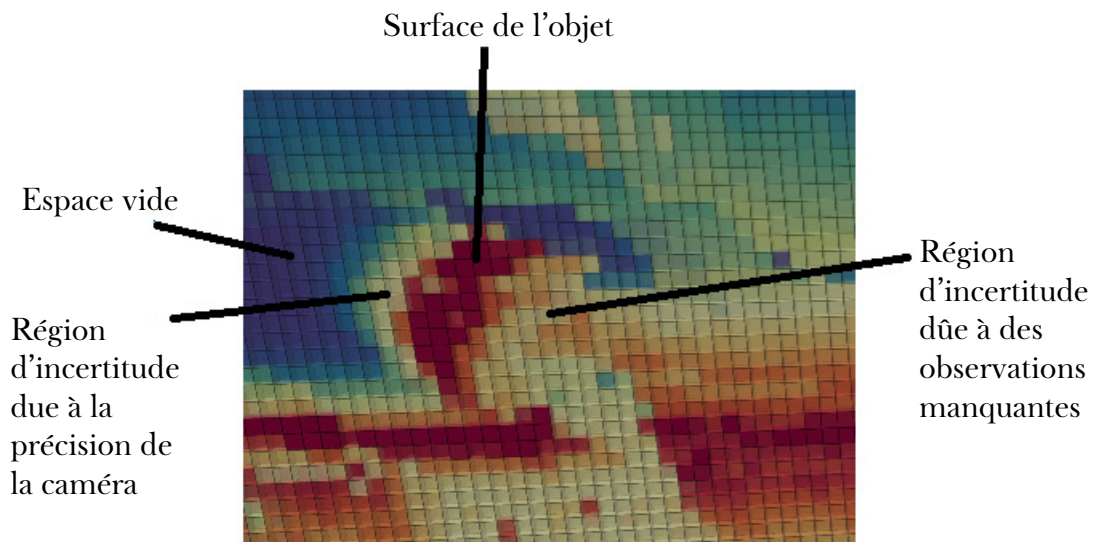


FIGURE 6.8 : Vue en coupe de la carte d'occupation montrant les différentes régions d'incertitude autour d'un objet principalement observé à partir du dessus et de la gauche.

Algorithme 6.5 Identification des points de vue non-informatifs

Entrées

- V , une structure NBVChunk
- 3: M_o , une grille d'occupation probabiliste
 M_m , la grille d'objet d'intérêt
 $[l, h]$, l'intervalle des valeurs $\Lambda_{M_o}(v)$ où v est considéré inconnu
- 6: k , un rayon d'expansion
 $B \leftarrow \mathcal{B}_{M_m}$ agrandie de k voxels dans chaque direction
 $c \leftarrow V.cible$
- 9: **Pour tout** point de vue $p \in V.vues \cup V.vues_inverses$, **faire**
 $v \leftarrow \frac{c-p}{\|c-p\|}$
 $s \leftarrow \max(|x_v|, |y_v|, |z_v|)$ ▷ équation (4.2)
- 12: $d \leftarrow \frac{v}{s}$
 $f \leftarrow 0$
 $i \leftarrow c$
- 15: **Tant que** $\neg f \wedge i \in B$, **faire**
 $f \leftarrow \Lambda_O(i) < l \wedge (\Lambda_O(n) < l \forall n \in \eta_1(i))$
 $i \leftarrow i - d$
- 18: $V.vues_informatives(p) \leftarrow \neg f$
-

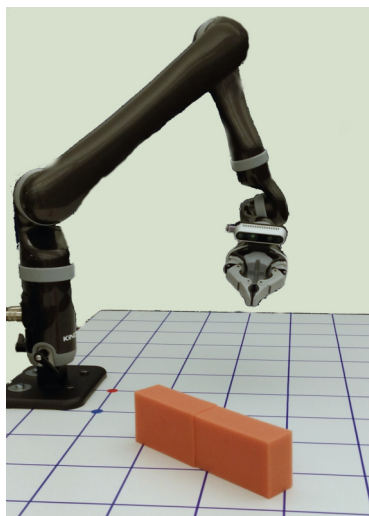
6.2 Algorithmes d'observation

Les mesures heuristiques décrites dans la section précédente sont intégrées dans le module de cartographie et demeurent indépendantes de la tâche à effectuer ainsi que des contraintes du système robotique et de l'environnement de travail. Pour appliquer ces mesures dans les diverses conditions expérimentales possibles, elles sont combinées à une connaissance des paramètres de la tâche dans des algorithmes d'observation originaux, qui utilisent ces informations pour sélectionner le prochain point de vue. Puisqu'ils dépendent des conditions expérimentales, ces algorithmes font partie du module de planification de tâche/coordonateur et non du module de cartographie. Deux exemples de tels algorithmes sont proposés ci-après, soit une version avec des contraintes de positionnement de la caméra et une sans celles-ci. La version avec contraintes correspond par exemple à l'utilisation d'un bras manipulateur avec une caméra œil-en-main alors que la version sans contraintes peut correspondre à une caméra se déplaçant librement en simulation. Les deux versions supposent un environnement où

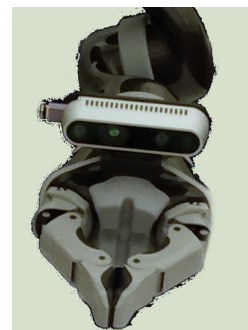
l'objet d'intérêt est posé sur une surface horizontale plane avec peu d'obstacles, mais peuvent être adaptés indépendamment des mesures heuristiques et du module de cartographie pour mieux correspondre à d'autres scénarios.

6.2.1 Avec contraintes de positionnement

Pour le premier algorithme d'observation proposé, on considère le cas d'une caméra œil-en-main installée près de l'effecteur d'un bras robotique à base fixe. On suppose également que ce robot est intégré avec un système de planification de mouvement qui prend en charge l'évitement des obstacles et permet de positionner le robot « proche » d'un point de vue même s'il n'est pas parfaitement atteignable. Dans ce scénario, la contrainte principale sur les points de vue est l'espace de travail du robot. En effet, en observant la construction du bras manipulateur (figure 6.9), on peut se rendre compte que l'espace de travail où il est possible de positionner la caméra et de la pointer vers l'objet est limité, en particulier pour observer le côté de l'objet le plus éloigné de la base du robot. Cependant, l'élément stochastique de la planification de mouvement fait en sorte que le robot tend à se déplacer même si l'effecteur est déjà proche de la pose souhaitée, réduisant le risque que celle-ci reste bloquée dans un minimum local.



(a)



(b)

FIGURE 6.9 : Observation d'un objet avec le manipulateur robotique (a) et positionnement du capteur sur l'effecteur (b).

Ainsi, le défi dans ce scénario est de construire les contraintes et commandes à transmettre au système de planification de mouvement pour positionner la caméra proche des meilleurs points de vue proposés par l’heuristique de prochain point de vue de la section 6.1. Avec MoveIt [192], le planificateur de trajectoire utilisé, le modèle de planification « sous-contrainte » permet de spécifier une région de l’espace où positionner l’effecteur du robot, mais son orientation doit être corrigée dans une étape subséquente. De plus, le critère principal pour classer les différents points d’observation suggérés par les algorithmes 6.2 et 6.3 est la capacité du robot à positionner la caméra dans la pose souhaitée. Ceci donne lieu à l’algorithme 6.6.

Cet algorithme présente une itération de la procédure d’observation proposée soumise aux contraintes de positionnement de la caméra décrites ci-dessus. Ainsi, on commence par obtenir les mesures heuristiques d’observation pour l’état courant de la carte d’objet d’intérêt M_m et de la carte d’occupation M_o . Comme décrit à la section 6.1, ceci résulte en une liste de structures NBVChunk qui encodent chacune les informations d’un groupe de voxels à observer. Chaque groupe a trois points de vue potentiels (c’est-à-dire le point de vue optimal [3e axe principal] et deux options de rechange, selon l’algorithme 6.2) et, puisque les groupes sont triés par taille et les points de vue par leur perpendicularité au groupe, on parcourt automatiquement les points de vue suggérés selon leur potentiel d’observation. Si l’heuristique n’indique pas que l’objet est complet, on parcourt tous les points de vue potentiels jusqu’à ce qu’on en trouve un qui puisse être rejoint par la caméra. Pour ce faire, on ignore d’abord les points de vue jugés « non informatifs » par l’heuristique, et on ajuste les points situés sous la surface de travail pour être légèrement au-dessus de celle-ci. Ensuite, on calcule le vecteur v qui indique la direction reliant le centroïde du groupe de voxels à observer, c , au point d’observation souhaité p , ainsi que les points ajustés, c' et p' , respectivement selon la distance minimale (d_o) et maximale (d_c) d’observation, définies en fonction des spécificités de la caméra et de l’environnement (section 3.2.1).

Le système de planification de mouvement est ensuite utilisé pour prendre en compte la cinématique du manipulateur déplaçant la caméra et l’évitement des obstacles. Ce faisant, on tente deux options de positionnement. Tout d’abord, l’option se rapprochant le plus du vecteur d’observation suggéré par l’heuristique de la section 6.1 est de positionner la caméra directement sur celui-ci, soit la ligne $c' - p'$, avec son axe optique parallèle à cette ligne et pointé vers c (moyennant des tolérances de positionnement et d’orientation pour aider à la convergence de la planification de mouvement). Si ceci est impossible, la deuxième option est de positionner la caméra « du bon côté » de l’objet à observer et de pointer son axe optique vers celui-ci. Ceci correspond à placer la caméra dans la demi-sphère de rayon d_c dont la

face plane est centrée sur c' et normale à $c' - p'$, puis à pointer l'axe optique vers c (figure 6.10). Si on réussit à positionner la caméra selon une de ces deux options, on peut prendre une observation, mettre à jour les cartes et redémarrer l'algorithme à partir du calcul des heuristiques en faisant un nouvel appel à l'algorithme 6.3. Autrement, on passe au prochain point de vue dans la liste. Cette procédure est répétée jusqu'à ce que l'objet soit considéré complet selon le critère de la section 6.1.1, ou jusqu'à ce que tous les points de vue soient ignorés. Ce dernier scénario implique que les régions inconnues restantes sont impossibles à observer avec cet algorithme, une condition qui dépend de la taille et du positionnement de l'objet d'intérêt dans l'espace de travail du robot.

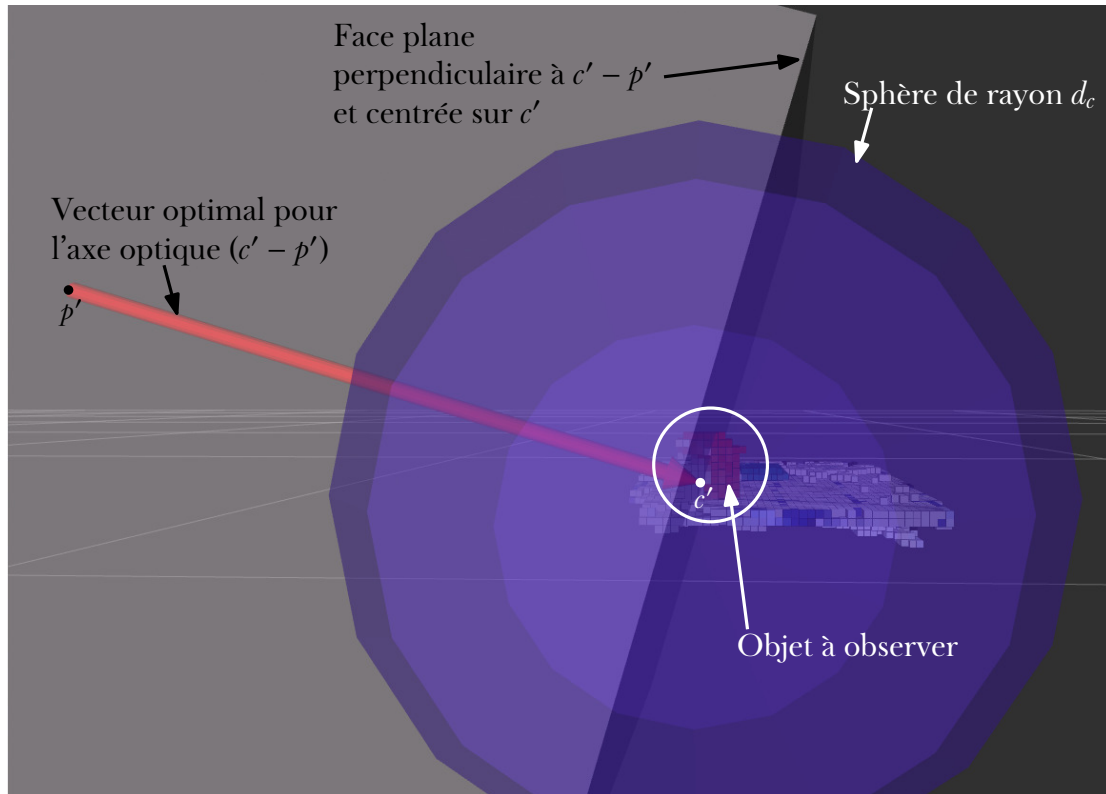


FIGURE 6.10 : Positionnement de la caméra pour observer une partie manquante de l'objet d'intérêt. Les contraintes de position sont établies pour que MoveIt tente de positionner la caméra à l'intérieur de l'intersection de la sphère mauve et du cube gris (partiellement hors de vue). L'orientation de la caméra est ajustée par la suite pour pointer l'axe optique vers c' .

Algorithme 6.6 Observation d'un objet avec contraintes de positionnement

Entrées

- M_m , la carte représentant l'objet à observer
- 3: M_o , la carte d'occupation
- d_c, d_o , la distance maximale et minimale entre la caméra et l'objet
- d_z , la hauteur minimum au-dessus du plan de travail
- 6: **Procédure** $\Pi(K_p, K_o)$ Appel au planificateur de mouvement pour calculer une trajectoire qui positionne la caméra selon les contraintes K_p et aligne l'axe optique avec K_o
- Procédure** $\Theta(P)$ Déplacer le robot selon la trajectoire P , prendre une observation et reprendre au début de l'algorithme
- $R \leftarrow \text{NBV}(M_m, M_o)$ ▷ algorithme 6.3
- 9: **Si** $R = \emptyset$, l'objet est complet (terminer l'algorithme)
- Pour tout** $G \in R$, **faire** ▷ structure NBVChunk
- $c \leftarrow G.cible$
- 12: **Pour tout** $p \in G.vues$, **faire** ▷ $p = (x_p, y_p, z_p)$
- Si** $\neg G.vues_informatives(p)$, passer au point de vue suivant.
- Si** $z_p < 0$ **alors** ▷ sous la surface de travail
- 15: $z_p \leftarrow d_z$
- $v \leftarrow \frac{p-c}{\|p-c\|}$
- $c' \leftarrow c + d_o v$
- 18: $p' \leftarrow c + d_c v$
- $P_l \leftarrow \Pi(\text{sur la ligne } c' - p', c' - p')$
- Si** P_l existe **alors** $\Theta(P_l)$
- 21: $C_s \leftarrow$ demi-sphère de rayon d_c , sa face plane centrée sur c' et normale à $c' - p'$
- $P_s \leftarrow \Pi(\text{dans } C_s, \emptyset)$
- Si** P_s existe **alors**
- 24: $m \leftarrow$ la position de la caméra à la dernière étape de P_s
- $P_p \leftarrow \Pi(m, c - m)$
- Si** P_p existe **alors** $\Theta(P_p)$
- 27: **Si non**, passer au point de vue suivant.
- Si non**, passer au point de vue suivant.
-

6.2.2 Sans contraintes de positionnement

Le deuxième algorithme d'observation proposé considère une caméra qui peut être déplacée librement et instantanément sans intégration avec un système de planification de mouvement, par exemple en simulation. Puis qu'il n'y a pas de contraintes liées au positionnement de la caméra ni d'élément stochastique lié à la planification de mouvement, ceci mène à quelques différences dans l'algorithme pour la sélection des meilleurs points de vue :

- Puisque tous les points de vue peuvent être rejoints exactement tels que spécifiés, on peut ignorer les points de vue problématiques (p. ex., sous le plan de travail) plutôt que de tenter de s'en approcher ;
- On doit éviter les minimums locaux et ajustements pouvant faire en sorte qu'on reste bloqué dans un point de vue non-informatif ;
- On doit éviter que la caméra soit placée à l'intérieur de l'objet. Par exemple, si la caméra est positionnée à une distance fixe du point à observer, cette distance doit être plus grande que toutes les dimensions de l'objet.

La procédure d'observation résultante est présentée à l'algorithme 6.7. Celui-ci est semblable à l'algorithme avec contraintes de positionnement (6.6), mais présente quelques différences fondamentales. D'abord, les points de vue situés sous la surface de travail sont ignorés pour éviter de bloquer l'algorithme dans un point de vue non-informatif (par exemple, si le point de vue optimal souhaité est complètement sous la table — ce qui est fréquent avec une vue initiale du dessus — le point de vue ajusté peut ne pas causer une mise à jour significative des cartes, faisant en sorte qu'on se retrouve avec le même point de vue optimal à la prochaine itération). Ensuite, pour contourner les limites de la planification de mouvement, la caméra est positionnée à un point p' situé directement sur le vecteur d'observation suggéré à une distance fixe d du groupe de voxels à observer.

Puisque les positions de la caméra ne sont pas limitées par la cinématique d'un manipulateur robotique, la plupart des points de vue proposés par l'heuristique seront utilisables. Ainsi, plutôt que de recalculer l'heuristique à chaque observation (donc généralement dès le premier groupe de voxels à observer), on peut gagner en efficacité en observant plusieurs groupes avant de recalculer l'heuristique. Cependant, il est possible d'observer plusieurs groupes d'un coup et donc d'avoir des mesures redondantes. Pour limiter le nombre de groupes observés de manière efficace, on se base sur l'observation suivante : au début de la séquence d'observations, la majorité des voxels inconnus de l'objet d'intérêt sont concentrés en un petit nombre de grands groupes alors qu'à la fin ils sont répartis en petits groupes distribués sur tout le volume

de l'objet. Ainsi, les observations au début du processus ont un grand impact sur la forme générale et la boîte englobante de l'objet d'intérêt, alors que celles à la fin visent plutôt à compléter de petites régions manquantes. Il est donc logique de recalculer l'heuristique plus souvent au début du processus, lorsque l'objet est partiel, qu'à la fin lorsqu'il est presque complet. Plusieurs critères peuvent correspondre à cette observation. Celui proposé dans l'algorithme 6.7 (ligne 21) est simple à calculer et consiste à limiter le nombre de groupes observés au nombre de fois que l'heuristique a été calculé dans le processus. Un autre critère testé est de recalculer l'heuristique après qu'on ait observé un groupe dont le nombre de voxels est inférieur à la moyenne. Notons que la performance de ces différents critères est difficile à comparer avec le robot physique en raison d'un manque de répétabilité des expériences.

Algorithme 6.7 Observation d'un objet sans contraintes de positionnement

Entrées

- M_m , la carte représentant l'objet à observer
 - 3: M_o , la carte d'occupation
 - τ_o la probabilité d'occupation pour qu'un voxel soit considéré vide
 - d , la distance souhaitée entre la caméra et l'objet
 - 6: n_l , le nombre de groupes à observer
 - $R \leftarrow \text{NBV}(M_m, M_o)$ ▷ algorithme 6.3
 - Si** $R = \emptyset$, l'objet est complet (terminer l'algorithme)
 - 9: $n_o \leftarrow 0$
 - Pour tout** $G \in R$, **faire** ▷ structure NBVChunk
 - $c \leftarrow G.\text{cible}$
 - 12: **Pour tout** $p \in G.\text{vues}$, **faire** ▷ $p = (x_p, y_p, z_p)$
 - Si** $\neg G.\text{vues_informatives}(p) \vee z_p < 0$ **alors**
 - Passer au point de vue suivant.
 - 15: $v \leftarrow \frac{p-c}{\|p-c\|}$
 - $p' \leftarrow c + dv$
 - Si** $\Lambda_{M_o}(p') > \tau_o$ **alors** passer au point de vue suivant. ▷ optionnel
 - 18: Déplacer la caméra au point p' et aligner son axe optique avec $c - p$
 - Prendre une observation.
 - $n_o \leftarrow n_o + 1$
 - 21: **Si** $n_o \geq n_l$ **alors**
 - Reprendre au début de l'algorithme avec $n_l \leftarrow n_l + 1$
 - Sinon**, Passer au groupe G suivant
-

6.3 Évaluation expérimentale de la sélection du prochain point de vue

Cette section présente l'évaluation expérimentale des mesures et méthodes pour l'identification des points de vue pour observer les parties manquantes de l'objet d'intérêt ainsi que l'intégration de ces mesures dans les algorithmes d'observation avec différents paramètres. Pour plus de répétabilité, on commence par comparer les différents paramètres en simulation à la section 6.3.1 avant d'élaborer sur les enjeux liés à l'environnement physique à la section 6.3.2. Afin d'encadrer l'évaluation du processus d'observation de l'objet d'intérêt, il convient d'abord de résumer ici la définition de trois concepts discutés dans les sections précédentes :

Le nombre de calculs de l'heuristique correspond au nombre d'appels à l'algorithme 6.3. Comme discuté à la section 6.1, cet algorithme résulte en plusieurs points de vue proposés répartis sur un ou plusieurs groupes de voxels inconnus. Ceci est habituellement abrégé « nombre de calculs (ou d'itérations) NBV » (*Next-Best-View*) dans les figures et tableaux de cette section.

Le nombre de points de vue proposés est le résultat direct de l'algorithme 6.3. Comme discuté à l'algorithme 6.2, un nombre fixe de points de vue est proposé pour chaque groupe de voxels inconnus identifié par les algorithmes 6.3 et 6.4.

Le nombre d'observations correspond au nombre de prises de vue réelles avec la caméra simulée ou physique. Ceci est le résultat des algorithmes d'observation discutés à la section 6.2. Ce nombre est généralement inférieur au nombre de points de vue proposés puisque ceux-ci sont utilisés ou non en fonction des différents critères des algorithmes d'observation.

6.3.1 Simulation

Les méthodes proposées pour la sélection de points de vue sont évaluées dans l'environnement de simulation basé sur IsaacSim (section 3.2.2), où le contrôle de l'environnement et l'élimination des erreurs de positionnement de l'objet, de la caméra et des mesures permettent une meilleure répétabilité et comparabilité des résultats. Dans cet environnement, les objets d'intérêt sont placés sur un plan de travail et observés avec une caméra RGBD virtuelle mobile qui produit en sortie un nuage de points similaire à celui obtenu par un capteur RealSense D435i mais sans erreur de positionnement sur les points. Trois vues initiales sont prises du dessus pour permettre à l'utilisateur d'identifier l'objet d'intérêt. Ces vues correspondent à

celles qu'il est possible d'obtenir avec le robot physique, c'est-à-dire que la caméra est d'abord positionnée perpendiculairement à la surface de travail horizontale, centrée sur celle-ci à une hauteur de 50 cm. Les deux autres positions correspondent à celle obtenues lorsque le robot fait ensuite une rotation de 45° dans les deux sens sur sa première articulation (axe vertical de la base) pour élargir le champ de vision initial. En simulation, la caméra peut être positionnée exactement au point d'observation souhaité, ce qui élimine les contraintes liées à la planification de mouvement et permet de tester les méthodes proposées comme des approches directes pour sélectionner les points de vue.

Ainsi, on teste les heuristiques d'observation et les paramètres de l'algorithme 6.7 avec les cinq objets virtuels montrés à la figure 6.11, soit la sphère de 10 cm de diamètre, l'éponge ($7 \times 4 \times 13$ cm) et la « longue éponge » ($7 \times 4 \times 26$ cm) semblables aux objets utilisés pour tester la segmentation au chapitre 6.3, auxquels on ajoute un cylindre de 10 cm de diamètre et 2 m de longueur et le lapin Stanford d'une hauteur de 45 cm. Pour chaque objet, une forme de référence est obtenue en voxélisant directement le modèle maillé. Le nombre de voxels dans les formes de référence avec différentes résolutions est présenté au tableau 6.1.

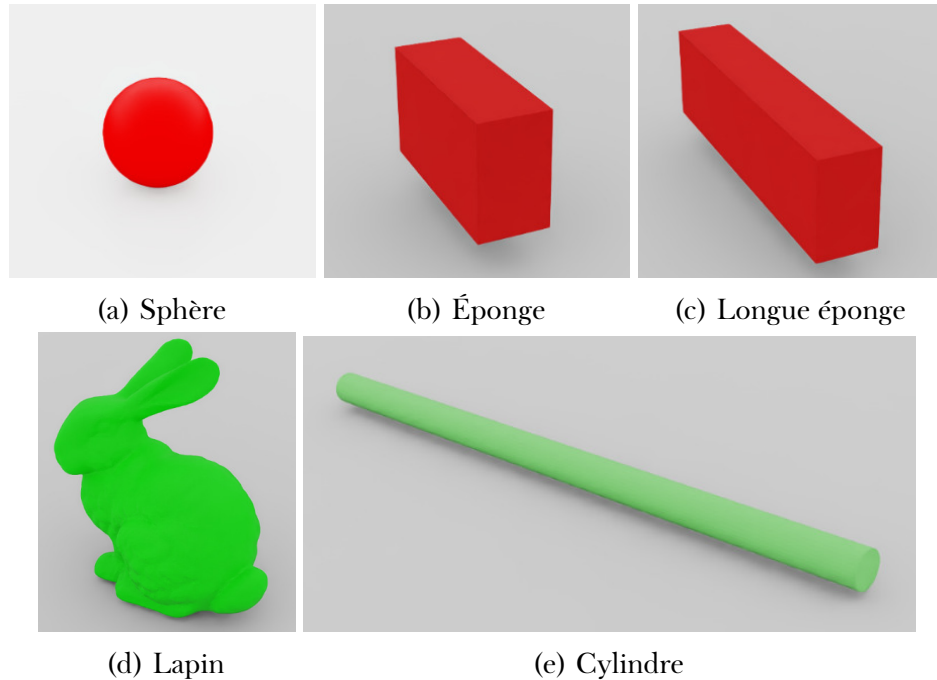


FIGURE 6.11 : Objets virtuels pour tester les méthodes de sélection des points d'observation.

TABLEAU 6.1 : Nombre de voxels dans les formes de référence

Objet (résolution)	voxels	Objet (résolution)	voxels
Sphère (8 mm)	1 293	Longue éponge (8 mm)	2 409
Sphère (6 mm)	2 881	Cylindre (8 mm)	35 642
Sphère (3 mm)	20 831	Lapin (8 mm)	42 936
Éponge (8 mm)	1 241		

Pour les tests en simulation, les nuages de points sont collectés dans des cartes à différentes résolutions. Puisqu’il n’y a pas d’incertitude sur la position des points, la carte d’occupation M_o utilise un modèle d’occupation ternaire (occupé/vide/inconnu). La couleur est conservée dans une carte M_{cd} selon la figure 3.6, c’est-à-dire qu’on y supprime les voxels considérés vides dans M_o (équation (4.10)). L’objet d’intérêt est segmenté de M_{cd} selon le principe de la figure 3.7, c’est-à-dire qu’on applique l’algorithme de segmentation hybride (section 5.3) ainsi que la procédure de remplissage (section 4.3.2) et de suppression des intersections (section 6.1.1).

6.3.1.1 Nombre de groupes à observer

Comme discuté à la section 6.2.2, lorsque l’espace à observer est divisé en groupes, le nombre de groupes à observer entre chaque calcul des heuristiques du prochain point de vue est un élément important dans l’optimisation du processus d’observation. Ainsi, on teste quatre critères pour limiter le nombre d’observations :

1. Observer tous les groupes retournés par l’heuristique.
2. Observer un seul groupe (le plus grand).
3. Observer, du plus grand au plus petit, un nombre de groupes égal au nombre de calculs de l’heuristique (numéro d’itération), ce qui représente une croissance linéaire du nombre d’observations à mesure que le processus avance.
4. Observer des groupes (du plus grand au plus petit) jusqu’à ce qu’on en observe un plus petit que la moyenne.

Ces stratégies sont testées en simulation avec les objets décrits précédemment. Ici, on considère quatre conditions de fin pour terminer les observations :

1. l’objet est complet,
2. tous les points de vue proposés sont ignorés,

3. le nombre de groupes, voxels inconnus et voxels connus n'ont pas changé en trois itérations,
4. le processus est visiblement bloqué sans atteindre l'une ou l'autre des conditions précédentes (interruption manuelle).

Le tableau 6.2 résume le nombre total d'observations, le nombre de calculs de l'heuristique, le nombre total de groupes de voxels inconnus (qui correspond au nombre de points de vue potentiels identifiés), le temps total de calcul de l'heuristique, le pourcentage final des voxels de l'objet d'intérêt par rapport à sa forme de référence (tableau 6.1), et la condition de fin qui a été atteinte. Notons que l'éponge est considérée complète dès le premier calcul des heuristiques car sa position et sa forme font en sorte que le remplissage peut éliminer tous les voxels inconnus à partir des trois vues fixes initiales.

Les figures 6.12 à 6.15 montrent la progression du processus d'observation pour les différents cas du tableau 6.2. Trois séries de graphiques sont présentées pour chaque cas et présentent les données provenant de l'algorithme 6.3, soit le pourcentage de voxels dans l'objet par rapport à sa forme de référence, le pourcentage de voxels inconnus dans la boîte englobante $\mathcal{B}$ de l'objet d'intérêt, le nombre de groupes de voxels inconnus, et le nombre moyen de voxels dans chacun de ces groupes.

Les figures 6.12 et 6.13 présentent l'évolution de l'observation de la sphère lorsque les grilles de voxels ont une résolution de 8 et 6 mm, respectivement. Avec les voxels de 8 mm, les quatre méthodes d'observation présentent un comportement semblable : l'observation converge rapidement vers les valeurs attendues, c'est-à-dire proche de 100% de voxels dans l'objet d'intérêt et 0% de voxels inconnus dans $\mathcal{B}$. Puisqu'il y a peu de groupes de voxels inconnus et que peu d'itérations sont nécessaires avant que l'objet soit considéré complet, il n'y a pas de différence majeure entre les différentes méthodes. Dans ces circonstances, le fait d'observer tous les groupes de voxels inconnus à chaque itération présente toutefois un léger avantage : puisque cette méthode fait les deux observations nécessaires dès la première itération, elle économise un cycle de calcul des heuristiques.

TABLEAU 6.2 : Performance des calculs NBV avec différentes limites d'observations par calcul

Objet (ré- solution)	Limite d'observations	Nb. obs.	Nb. NBV	Nb. Groupes	Temps NBV total (s)	% voxels final	Fin
Sphère (8 mm)	tous les groupes	2	2	2	0,30	104,25	Complet
	un groupe	2	3	3	0,38	101,70	Complet
	numéro d'itération	2	3	3	0,79	102,01	Complet
	taille du groupe	3	3	3	0,37	104,18	Complet
Sphère (6 mm)	tous les groupes	16	4	140	13,88	2519,54	Interrompu
	un groupe	3	4	6	0,64	105,00	Complet
	numéro d'itération	8	6	10	2,16	125,16	Complet
	taille du groupe	4	3	5	0,60	106,32	Complet
Sphère (3 mm)	tous les groupes	9	3	9	0,97	103,80	Complet
	un groupe	10	10	4886	299,62	1187,98	Interrompu
	numéro d'itération	10	5	961	87,54	103,01	Complet
	taille du groupe	9	5	16	2,88	102,05	Complet
Longue éponge (8 mm)	tous les groupes	5	5	6	0,56	95,48	Pas de progrès
	un groupe	5	6	7	0,68	95,43	Pas de progrès
	numéro d'itération	5	6	7	0,67	95,43	Pas de progrès
	taille du groupe	9	9	10	1,03	95,48	Pas de progrès
Cylindre (8 mm)	tous les groupes	170	40	171	52,67	102,90	Interrompu
	un groupe	10	11	204	12,73	43,24	Pas de progrès
	numéro d'itération	28	8	136	11,19	103,16	Complet
	taille du groupe	33	11	100	14,85	100,19	Pas de progrès
Lapin (8 mm)	tous les groupes	120	4	142	33,24	105,64	Aucune vue valide
	un groupe	20	20	165	15,28	1,89	Interrompu
	numéro d'itération	31	8	224	19,32	168,34	Interrompu
	taille du groupe	48	24	292	53,72	103,48	Complet
Éponge (8 mm)	S.O.	0	1	0	0,12	96,05	Complet

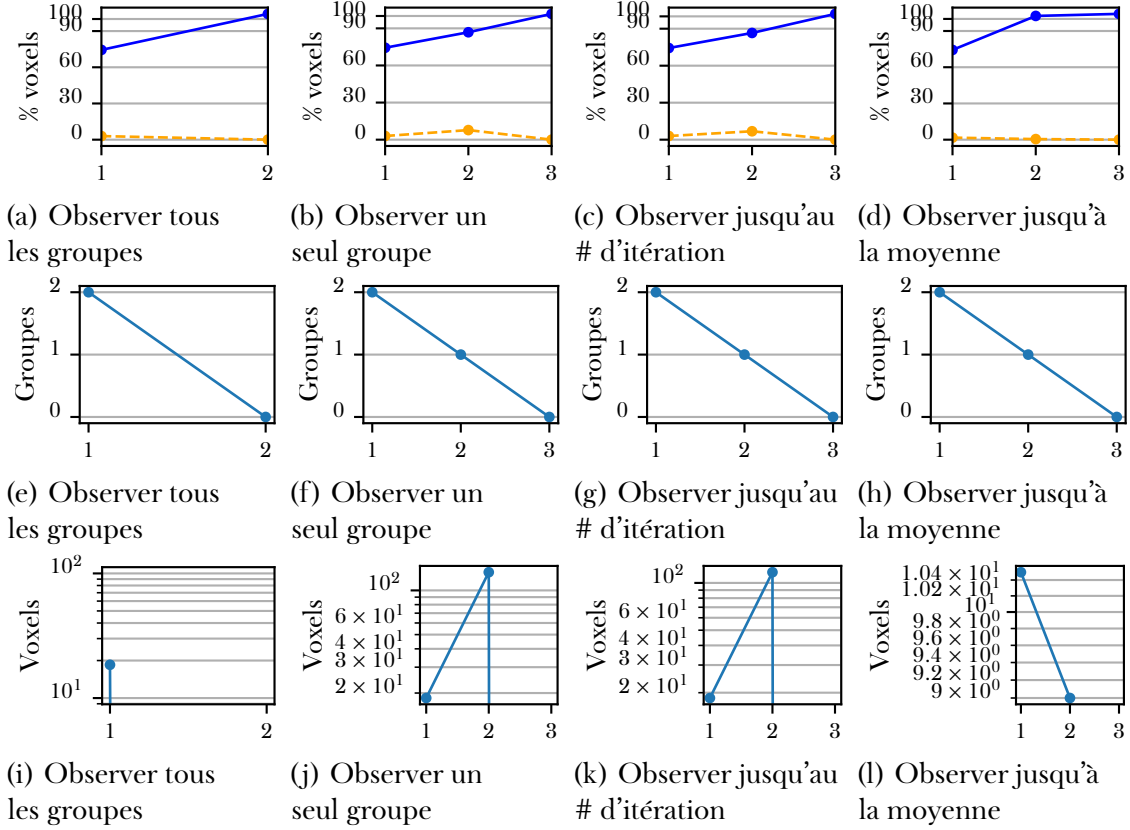


FIGURE 6.12 : Progression du processus d'observation pour la sphère (voxels de 8 mm) pour différentes limites du nombre d'observations par calcul NBV. Les axes horizontaux indiquent le numéro d'itération NBV, c'est-à-dire le nombre de calculs de l'heuristique. (a)–(d) montrent le pourcentage de voxels dans l'objet d'intérêt (ligne bleue pleine) par rapport à sa forme de référence et le pourcentage de voxels inconnus (ligne orange pointillée) dans la boîte englobante $\mathcal{B}$; (e)–(h) montrent le nombre de groupes de voxels inconnus ; (i)–(l) montrent le nombre moyen de voxels inconnus par groupe.

Avec des voxels de 6 mm (figure 6.13), on remarque un plus grand nombre de voxels inconnus par groupe ainsi qu'un plus grand nombre de groupes. Ceci demande donc un plus grand nombre d'observations avant de converger, mais la plupart des méthodes réussissent tout de même à compléter l'objet en un petit nombre d'itérations. On remarque aussi des pics au-dessus de 100% pour le nombre de voxels dans l'objet d'intérêt. Ces pics correspondent à des erreurs de segmentation faisant en sorte que des voxels superflus soient inclus dans l'objet d'intérêt, allant jusqu'à sélectionner la grille de voxels entière. Bien que ces erreurs soient généralement résolues dans une itération subséquente, elles sont accompagnées d'un pic dans le nombre de voxels inconnus et de groupes à observer. Ceci peut mener à un grand nombre d'observations superflues lorsque celles-ci ne sont pas strictement limitées : dans le

cas où on tente d'observer tous les groupes entre chaque itération, on aurait eu ici plus de 120 observations supplémentaires si le processus n'avait pas été interrompu manuellement (figure 6.13e).

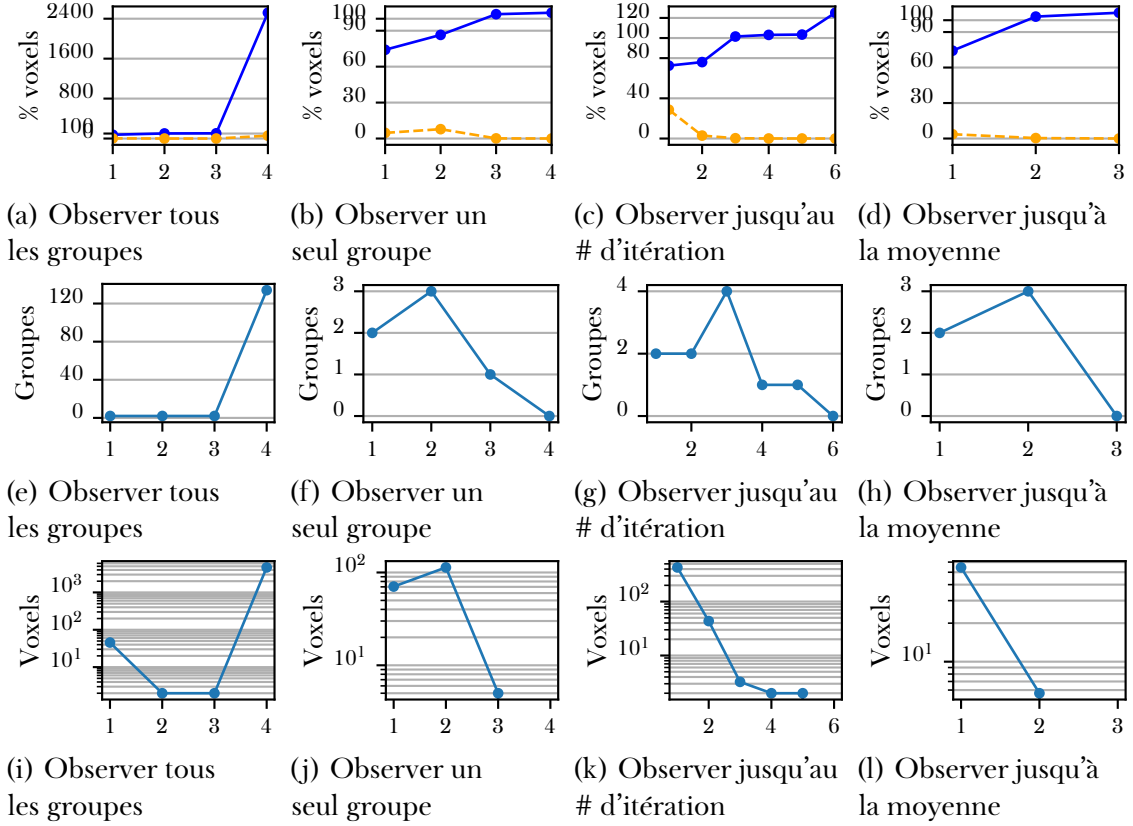


FIGURE 6.13 : Progression du processus d'observation pour la sphère (voxels de 6 mm) pour différentes limites du nombre d'observations par calcul NBV.

La figure 6.14 présente le progrès de l'observation pour le grand cylindre. Cet objet étant significativement plus grand que les précédents, on s'attend à ce qu'il demande un nombre plus important d'observations pour être complètement mesuré. De plus, l'objet est plus grand que le champ de vision de la caméra, ce qui fait en sorte que la boîte englobante est périodiquement agrandie pour accommoder de nouvelles portions de l'objet. Ceci est visible pour toutes les stratégies comme des oscillations dans le pourcentage de voxels inconnus dans $\mathcal{B}$ ainsi que des pics dans le nombre de groupes et le nombre de voxels par groupe. Ici, on remarque que l'observation de tous les groupes à chaque itération converge proche de 100% en moins de 10 itérations, mais qu'on continue longtemps à observer de petits groupes de voxels. À l'inverse, l'observation d'un seul groupe par itération reste bloquée car le plus grand groupe (qui est le seul qu'on tente d'observer) ne produit pas un point de vue informatif. L'observation

d'un nombre de groupes limité adaptativement par le numéro d'itération ou la moyenne du nombre de voxels par groupe permet d'éviter ces cas d'erreur et d'optimiser la convergence de l'observation.

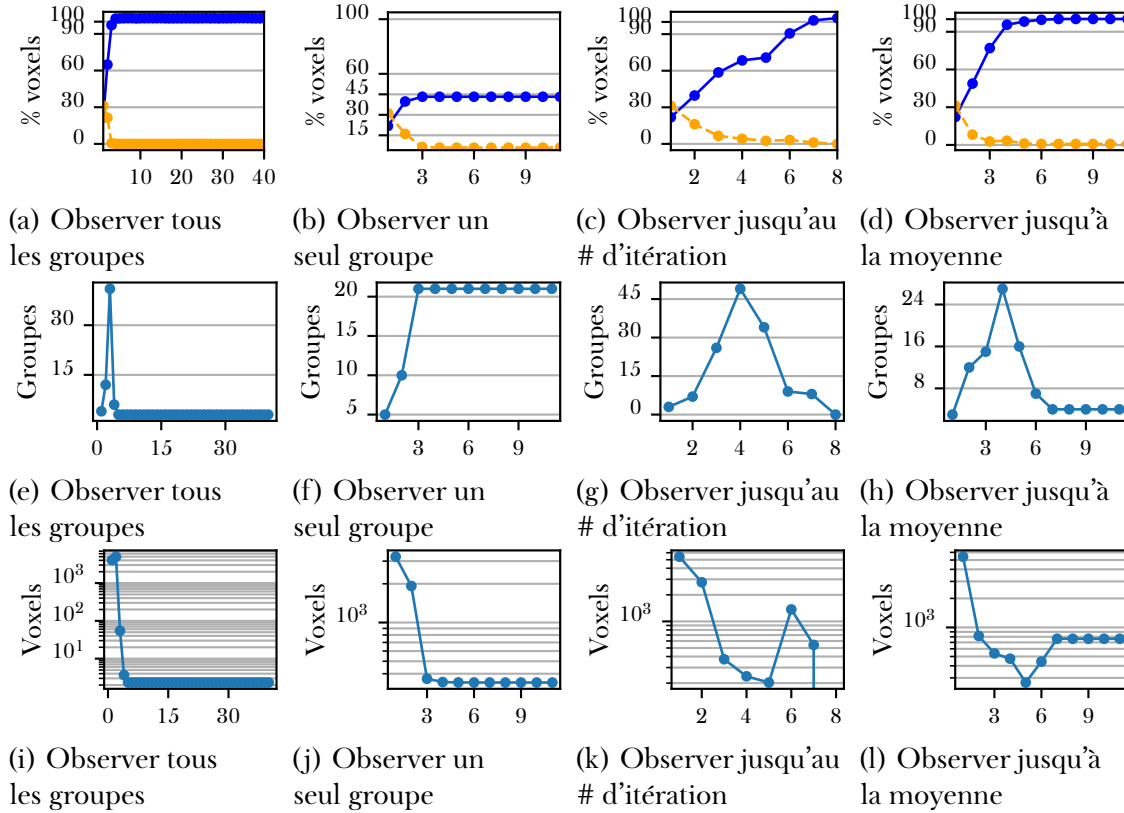


FIGURE 6.14 : Progression du processus d'observation pour le cylindre (voxels de 8 mm) pour différentes limites du nombre d'observations par calcul NBV.

La figure 6.15 présente le déroulement du processus d'observation pour le lapin avec des voxels de 8 mm. Ici encore, on remarque une convergence autour de 100% pour la plupart des méthodes malgré quelques pics liés aux erreurs de segmentation. Dans le cas où on observe un seul groupe par itération, l'observation reste bloquée à un faible pourcentage de complétion, indiquant que le point de vue associé au plus grand groupe est peu informatif. Observer un plus grand nombre de groupes entre les calculs des heuristiques permet d'éviter ce problème en augmentant la quantité d'information obtenue entre les calculs. Cependant, observer tous les groupes augmente les risques de se retrouver avec un grand nombre d'observations superflues dans le cas d'une erreur de segmentation, ici plus de 80 observations supplémentaires selon la figure 6.15e.

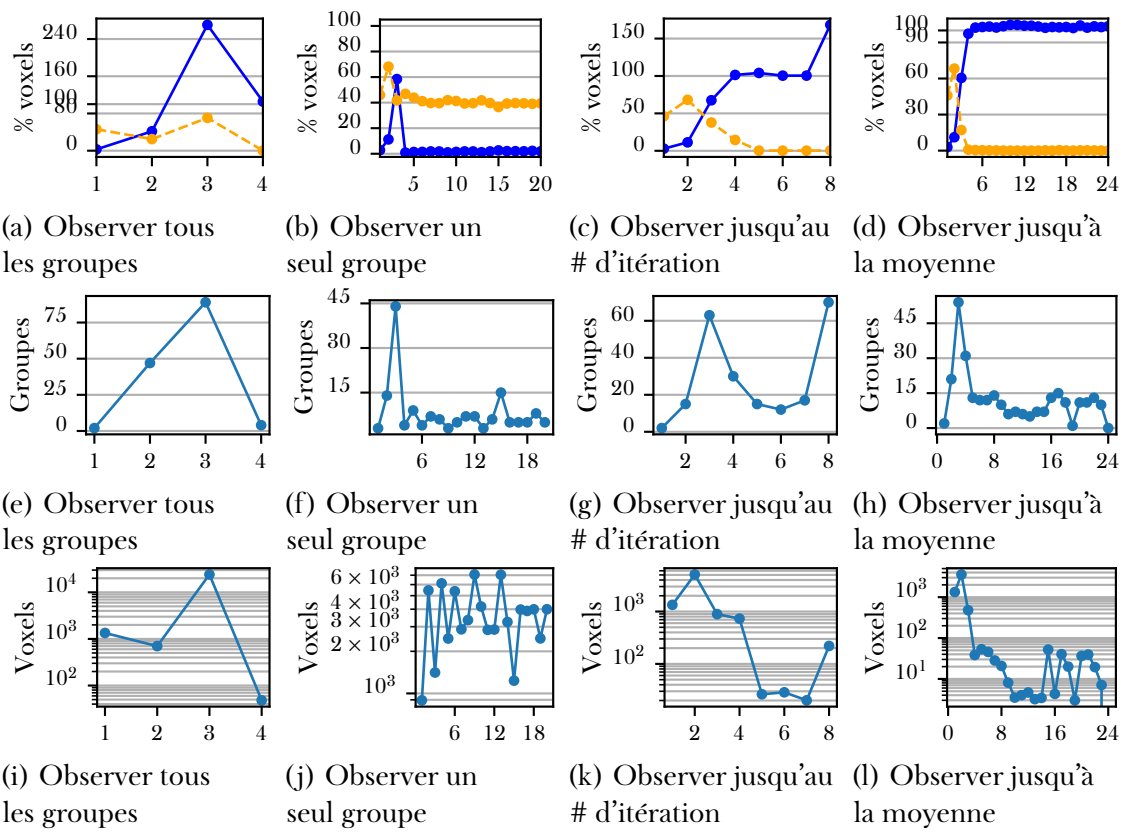


FIGURE 6.15 : Progression du processus d'observation pour le lapin (voxels de 8 mm) pour différentes limites du nombre d'observations par calcul NBV.

Dans ces résultats, on remarque que l'utilisation d'une stratégie adaptative pour limiter le nombre de groupes observés selon le numéro d'itération ou la taille des groupes permet souvent une observation plus complète et plus rapide que l'observation de tous les groupes ou d'un seul groupe. On remarque aussi que le nombre de voxels (et la forme segmentée) de l'objet d'intérêt converge proche de 100% même lorsque toutes les vues sont ignorées ou qu'il n'y a pas de progrès, indiquant que ces conditions résultent plus souvent d'une limitation dans la détection de si l'objet est complet plutôt que du calcul des points de vue.

En ce qui concerne le temps de calcul, on note que la méthode proposée est efficace. Dans les cas du tableau 6.2, le temps de calcul moyen (sans parallélisme) est de moins d'une seconde pour calculer l'heuristique NBV (algorithme 6.3) et obtenir tous les points de vue suggérés de l'itération dans 64 % des cas. Ce temps est proportionnel à la boîte englobante de l'objet d'intérêt $\mathcal{B}$ et au nombre de groupes de voxels inconnus identifiés, de sorte que le calcul des points de vue pour un grand objet à haute résolution est plus coûteux. Chaque calcul de l'heuristique NBV résulte en plusieurs points de vue proposés (un par groupe de voxels inconnus), atteignant un temps de calcul moyen de 0,13 secondes par point de vue potentiel. Bien que de plus grandes valeurs soient présentes, elles sont expliquées par des erreurs de segmentation faisant en sorte que la scène entière soit incluse dans $\mathcal{B}$, augmentant dramatiquement le nombre de voxels dans le calcul.

Ces erreurs de segmentation sont discutées plus en détails à l'annexe B.1 et peuvent notamment être constatées dans les cas de la sphère avec des voxels de 6 mm observant tous les groupes et de la sphère avec des voxels de 3 mm observant un seul groupe, où le nombre final de voxels dans la forme segmentée est respectivement de 25 et 10 fois le nombre attendu dans la forme de référence. Les cas avec des pics significativement supérieurs à 100% dans les figures 6.12 à 6.15 sont aussi affectés. En éliminant ces valeurs erronées, on obtient un temps de calcul moyen de 0,9 secondes par calcul des heuristiques. Globalement, ces temps de calcul sont avantageux par rapport aux 10 secondes par point de vue rapporté dans [99] et aux 28 secondes (sur GPU) pour générer et évaluer cinq points de vue dans [100], en considérant différents outils de calcul. Les résultats détaillés pour la sphère avec des voxels de 3 mm et la longue éponge sont présentés à l'annexe B.1. Des exemples de séquences d'observation sont présentés à l'annexe B.2.

6.3.1.2 Impact de la division en groupes

Pour valider l'importance de la division de l'espace inconnu en groupes de voxels contigus, on répète l'expérience sans faire cette division, en utilisant des voxels de 8 mm. Puisqu'il y a un seul groupe de voxels par calcul des heuristiques, on a toujours une seule observation par calcul NBV qui indique que l'objet n'est pas complet, et le nombre de groupes total est égal au nombre d'observations. Le tableau 6.3 résume la performance et la condition de fin atteinte alors que la figure 6.16 montre la progression du pourcentage de voxels inconnus et connus. Pour les objets dont le volume est relativement uniformément distribué comme la sphère, on parvient tout de même à observer efficacement tous les voxels et à détecter quand l'objet est complet. Cependant, pour les autres objets, l'algorithme reste bloqué à tenter d'observer les voxels inconnus par un point de vue peu efficace. Ceci est particulièrement évident dans le cas du cylindre : à cause de sa forme allongée, les voxels inconnus sont distribués à ses deux extrémités, faisant en sorte que le centroïde et les axes principaux des voxels inconnus poussent l'heuristique à aligner la caméra sur le centre de l'objet, qui est déjà connu. En ce qui concerne le lapin, on note que sa forme de référence a un grand nombre de voxels, et donc que le faible pourcentage de voxels inconnus restants représente tout de même une portion significative de l'objet.

Dans ce petit échantillon, on note que le temps de calcul atteint une moyenne de 0,86 secondes par calcul NBV, dont le nombre correspond aussi au nombre de groupes observés et de points de vue suggérés. La division en groupes n'a donc pas un impact significatif sur le temps de calcul total pour proposer des points de vue, mais elle réduit le temps de calcul par point de vue proposé puisque plus de points de vue sont proposés avec la division en groupes que sans elle, pour un temps de calcul total semblable.

TABLEAU 6.3 : Performance des calculs NBV sans division en groupes (voxels de 8 mm)

Objet	Nb. obs.	Nb. NBV	Temps NBV total (s)	% voxels final	Fin
Sphère	3	4	0,54	104,18	Complet
Longue éponge	9	9	1,20	95,39	Interrompu
Cylindre	16	16	11,20	57,24	Interrompu
Lapin	18	18	44,53	99,19	Interrompu
Éponge	0	1	0,07	96,05	Complet

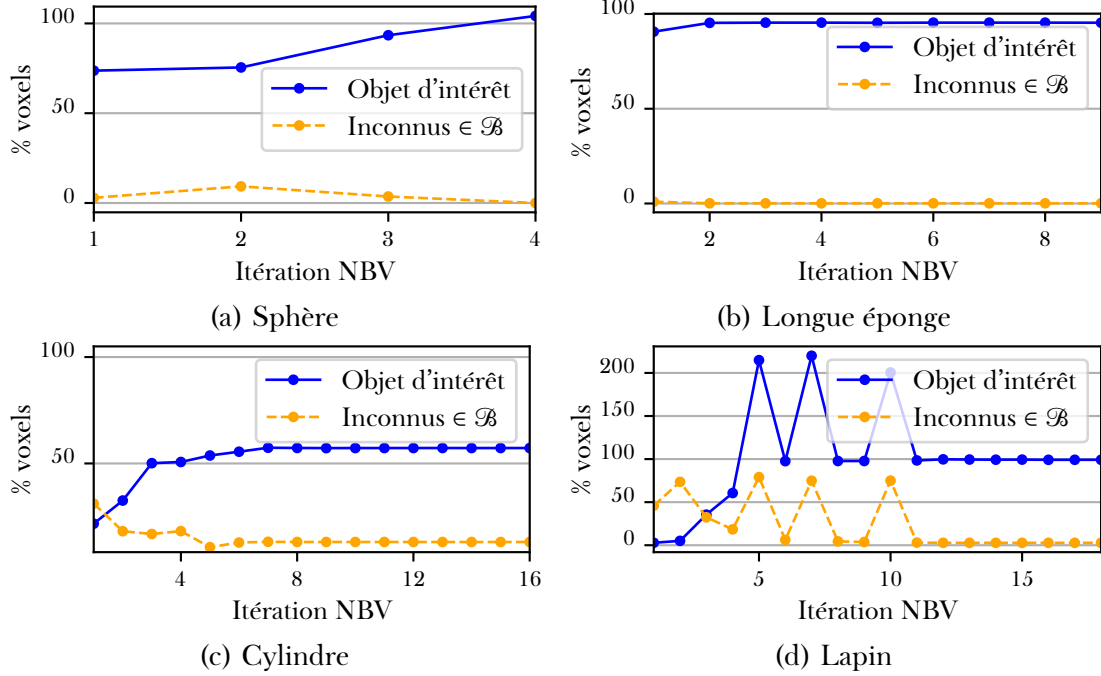


FIGURE 6.16 : Évolution du nombre de voxels connus et inconnus par calcul NBV sans division en groupes (voxels de 8 mm)

6.3.1.3 Impact de la suppression des intersections

Pour tester l'impact de la suppression des intersections, on répète l'expérience sans supprimer les intersections pour le cas qui a obtenu la meilleure performance, soit avec la division des voxels inconnus en groupes et en limitant le nombre d'observations au numéro d'itération du calcul des heuristiques. En simulation, on peut s'attendre à ce qu'il y ait peu d'intersections à supprimer car le seul contact est avec le plan de travail, et celui-ci est parfaitement plat grâce à l'absence d'erreurs de position dans les nuages de points. À part le dessous de l'objet, il y a donc peu de régions non-observables telles que discutées à la section 6.1.1. Comme on peut le voir en comparant le tableau 6.4 et la figure 6.17 au tableau 6.2 et aux figures 6.12 à 6.15 (c), ceci est partiellement vrai. Bien que les objets convergent vers un nombre de voxels similaire aux résultats obtenus à la section 6.3.1.1, on remarque que l'observation du cylindre, qui avait été complétée avec succès avec la suppression des intersections (tableau 6.2), reste bloquée sans celle-ci (tableau 6.4). La suppression des intersections amène donc un petit avantage pour détecter si l'objet est complet, même en simulation. Notons que la suppression des intersections étant un processus séparé du calcul des points de vue, elle n'affecte pas directement le temps de calcul, mais plutôt la taille de la boîte englobante de l'objet d'intérêt et le nombre de voxels inconnus dans celle-ci.

TABLEAU 6.4 : Performance des calculs NBV sans suppression des intersections (voxels de 8 mm).

Objet	Nb. obs.	Nb. NBV	Temps NBV total (s)	% voxels final	Fin
Sphère	2	3	0,87	97,68	Complet
Longue éponge	5	6	0,81	95,39	Pas de progrès
Cylindre	65	15	23,33	96,42	Pas de progrès
Lapin	55	38	55,76	107,14	Interrompu
Éponge	0	1	0,07	96,05	Complet

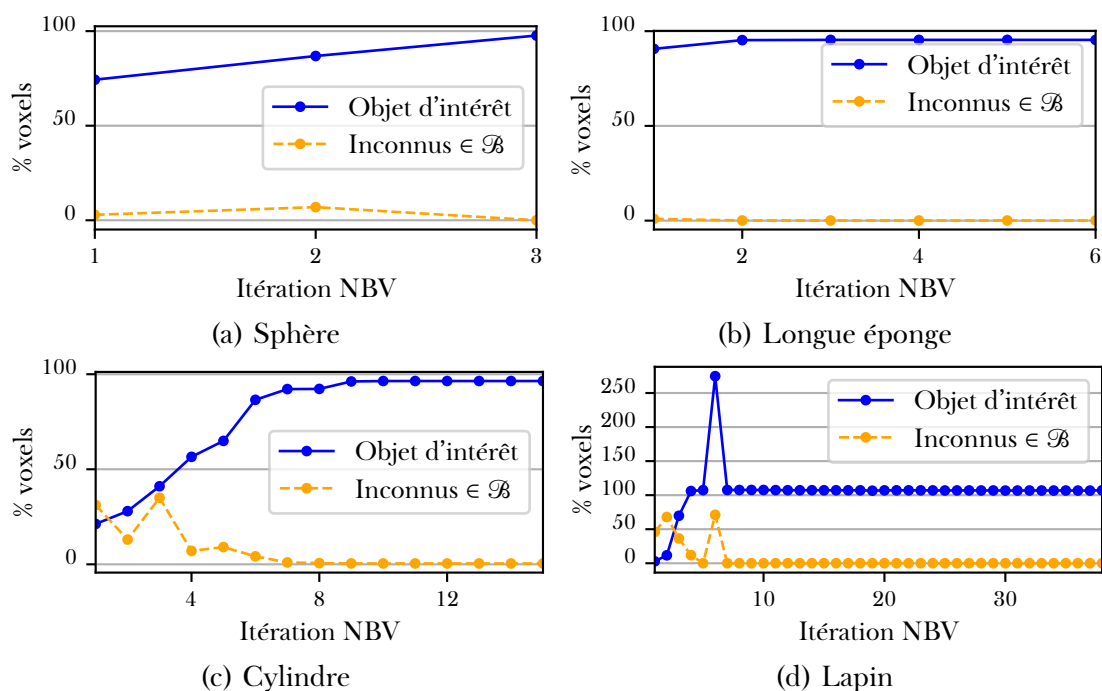


FIGURE 6.17 : Évolution du nombre de voxels connus et inconnus par calcul NBV sans suppression des intersections (voxels de 8 mm)

6.3.2 Environnement physique

La figure 6.18 présente une séquence d’observations avec le robot physique, en utilisant le modèle d’occupation probabiliste et en limitant le nombre d’observations au numéro d’itération. La figure 6.19 montre la trajectoire du robot et l’évolution de la représentation de l’environnement (carte colorée) pour chaque observation. Bien que l’objet soit considéré complet à la fin de cette séquence, on remarque que la planification de mouvement peine à diriger le capteur vers l’objet pour observer la portion éloignée de la base du robot, notamment la face verticale à droite de l’éponge. On note aussi que le robot tend à faire de grands mouvements pour déplacer la caméra entre des points de vue qui sont relativement rapprochés les uns des autres. Ceci est dû aux contraintes de la cinématique du robot ainsi qu’au fonctionnement du planificateur de trajectoire. L’optimisation des mouvements du robot ne faisant pas partie des objectifs de cette thèse, les trajectoires générées par MoveIt sont utilisées telles quelles.

Ainsi, dans l’environnement physique, la performance de l’heuristique du prochain point de vue et de l’algorithme d’observation dépendent non seulement de la précision de la caméra et de la segmentation, mais aussi de la limite majeure qu’est la capacité du robot à positionner la caméra près du point de vue sélectionné. En effet, les contraintes de la cinématique du robot, l’imprécision du contrôleur et les approximations dans le planificateur de mouvement permettent rarement de positionner la caméra exactement à la position et dans l’orientation souhaitées. La capacité du robot à se rapprocher de la pose désirée dépend aussi de facteurs difficiles à contrôler comme le positionnement de l’objet dans l’espace de travail du robot et les éléments stochastiques de la planification de mouvement. Ces facteurs aléatoires sont un obstacle à la comparaison formelle des différents paramètres de sélection du prochain point de vue. À la place, on présente ici quelques exemples des cas d’erreur rencontrés et de leurs solutions, menant au processus d’observation décrit à la section 6.2.1 et aux différentes optimisations et vérifications dans l’heuristique de prochain point de vue.

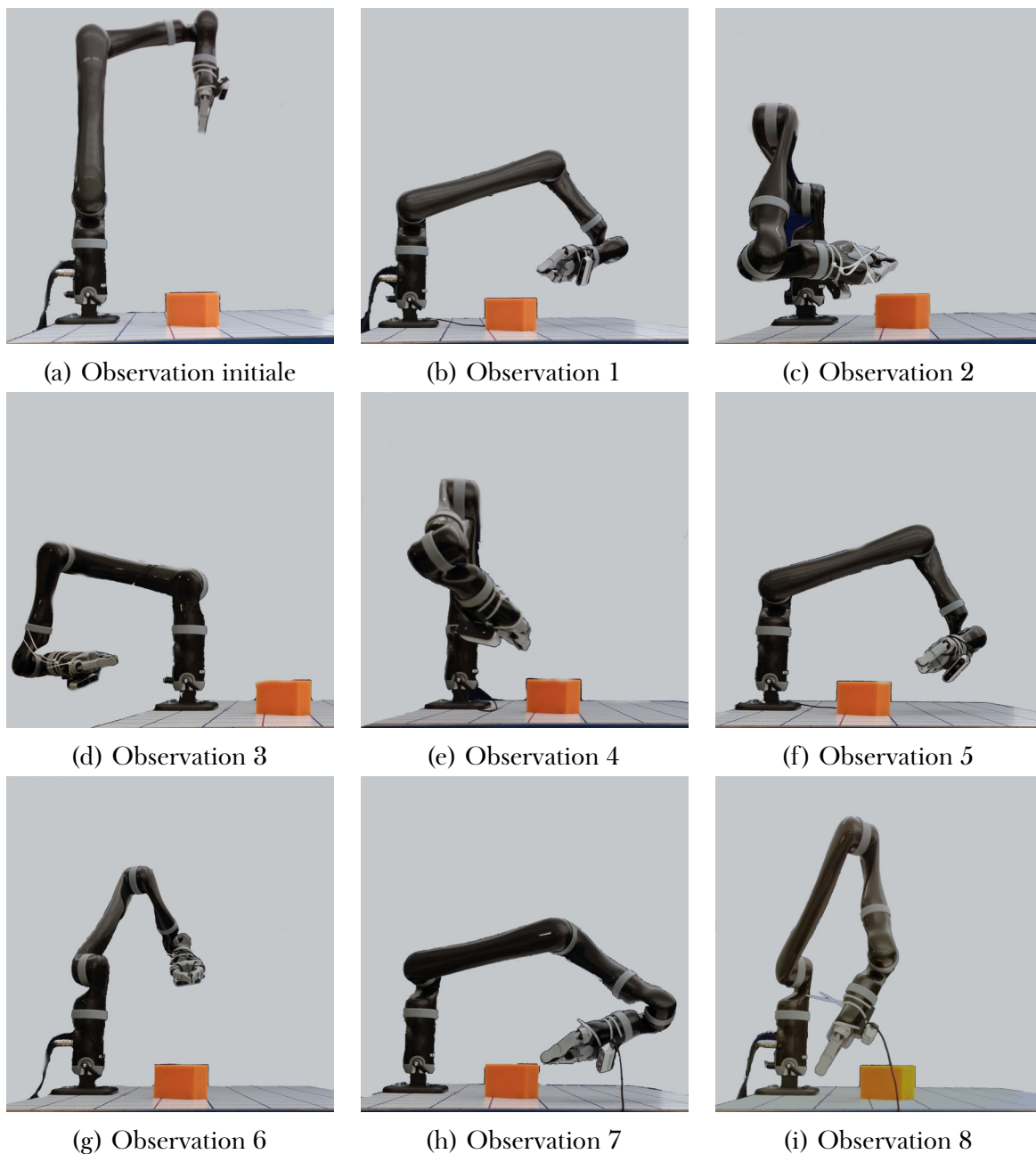
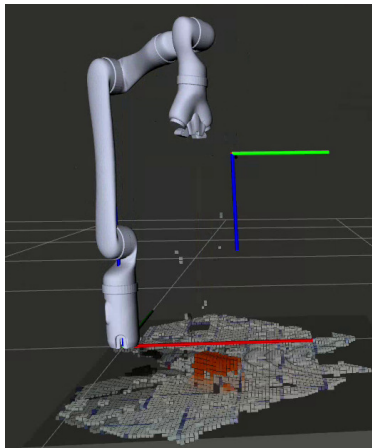
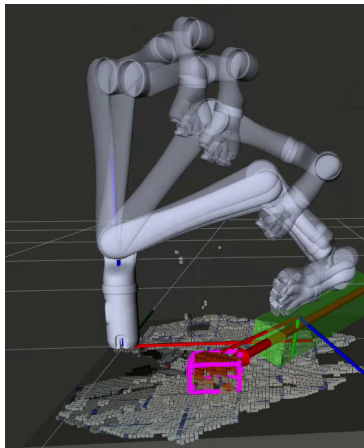


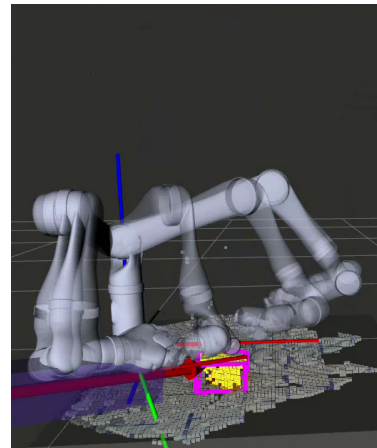
FIGURE 6.18 : Séquence d'observations avec le robot physique.



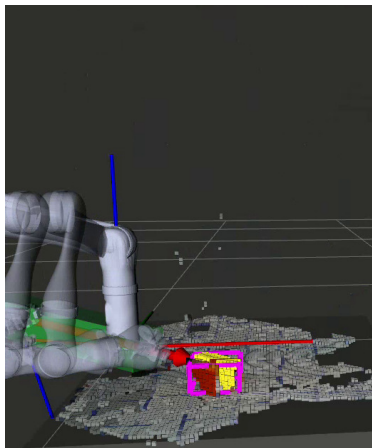
(a) Observation initiale



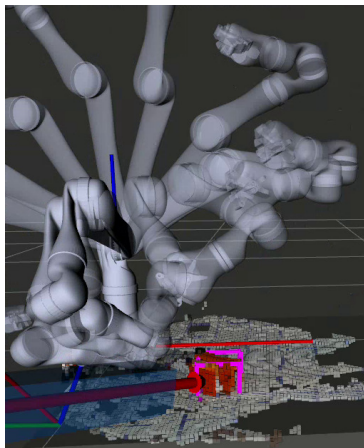
(b) Observation 1



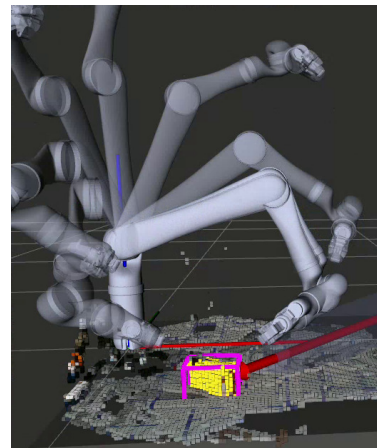
(c) Observation 2



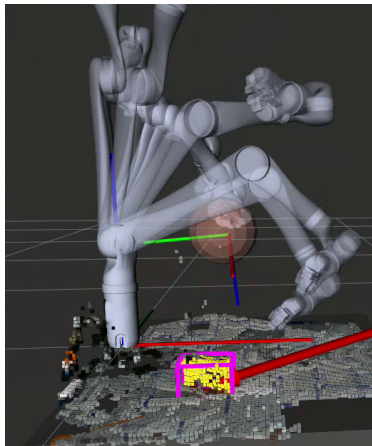
(d) Observation 3



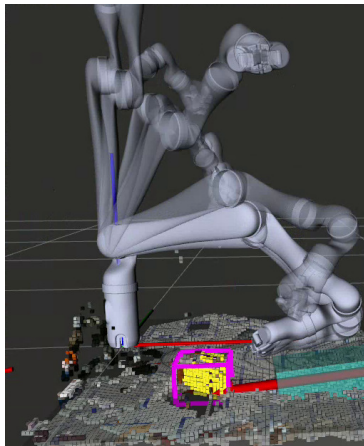
(e) Observation 4



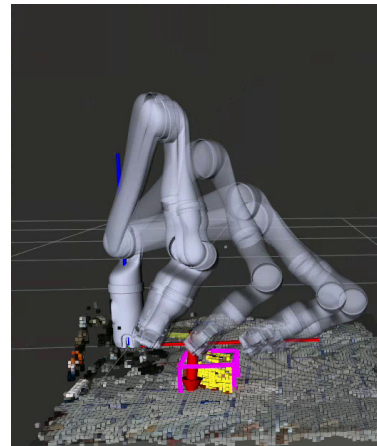
(f) Observation 5



(g) Observation 6



(h) Observation 7



(i) Observation 8

FIGURE 6.19 : Trajectoires du robot et représentation de l'environnement correspondant à la figure 6.18.

Modèle d'occupation ternaire Avec une carte d'occupation configurée pour un modèle ternaire (occupé/vide/inconnu), les mesures erronées causées par l'imprécision de la caméra physique ont un impact majeur sur les résultats. La figure 6.20 montre la carte M_{cd} , qui représente la vue en couleurs de l'environnement de travail avec la sphère rouge comme à la figure 6.1, de même qu'une tranche de la carte d'occupation ternaire correspondante M_o après les trois vues fixes initiales (a, d) ainsi qu'après une et deux observations à partir de points de vue différents (b, e et c, f, respectivement). À la figure 6.20 (b, c et e, f), des parties préalablement identifiées de l'objet (en 6.20a, d) sont supprimées par une mesure erronée. Ceci fait en sorte que l'objet est considéré complet avant de l'être réellement — dans ce cas-ci parce que sa boîte englobante ne correspond plus qu'à un seul voxel. À la figure 6.21, on répète l'expérience avec un modèle d'occupation probabiliste représentant l'état d'occupation sur l'intervalle $[0, 1]$. Dans ce cas, on note que les observations précédentes ne sont pas complètement effacées par une mesure erronée. Par exemple, bien que l'objet montré à la figure 6.21b ait perdu quelques voxels précédemment mesurés en (a), la carte d'occupation en (e) n'est pas complètement réinitialisée par rapport à celle en (d), contrairement au cas de la figure 6.20. Il est donc possible de prendre plus d'observations et d'obtenir une convergence plus fiable (mais plus lente) de la probabilité d'occupation.

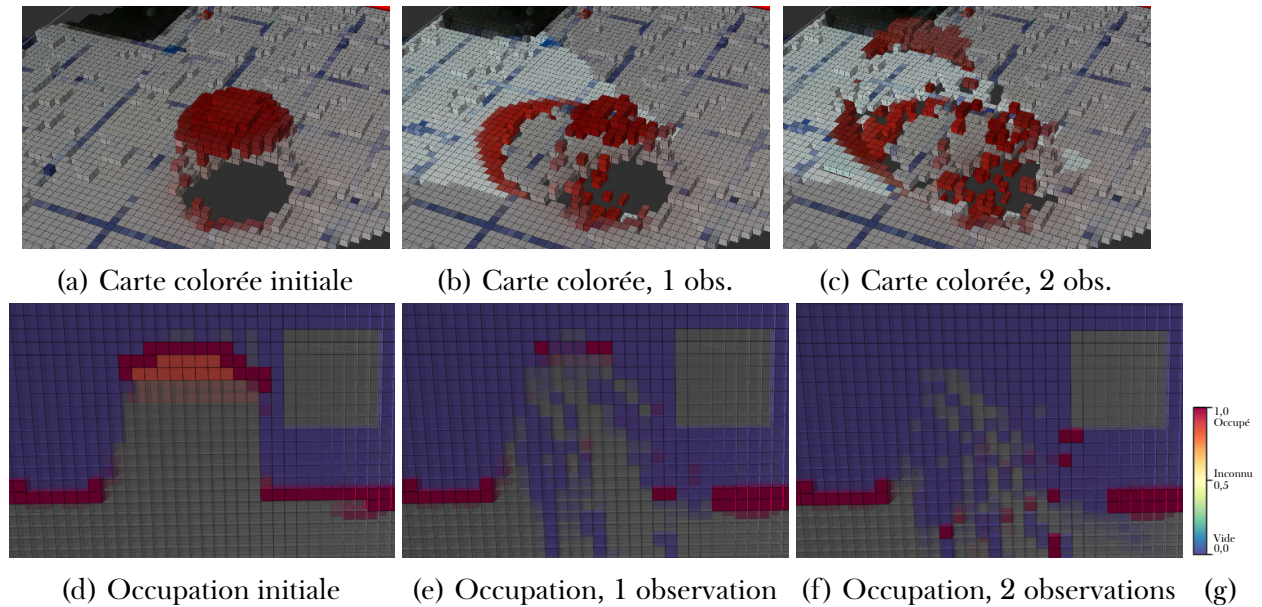


FIGURE 6.20 : Exemple d'erreur introduite par l'imprécision de la caméra avec un modèle d'occupation ternaire. (a)–(c) carte colorée dynamique selon la figure 3.6 montrant l'objet d'intérêt après les vues initiales fixes (a) et d'autres observations (b,c). (d)–(f) tranche verticale de la carte d'occupation correspondante, remplie selon le processus décrit à la figure 3.7 avec une valeur de 0,8.

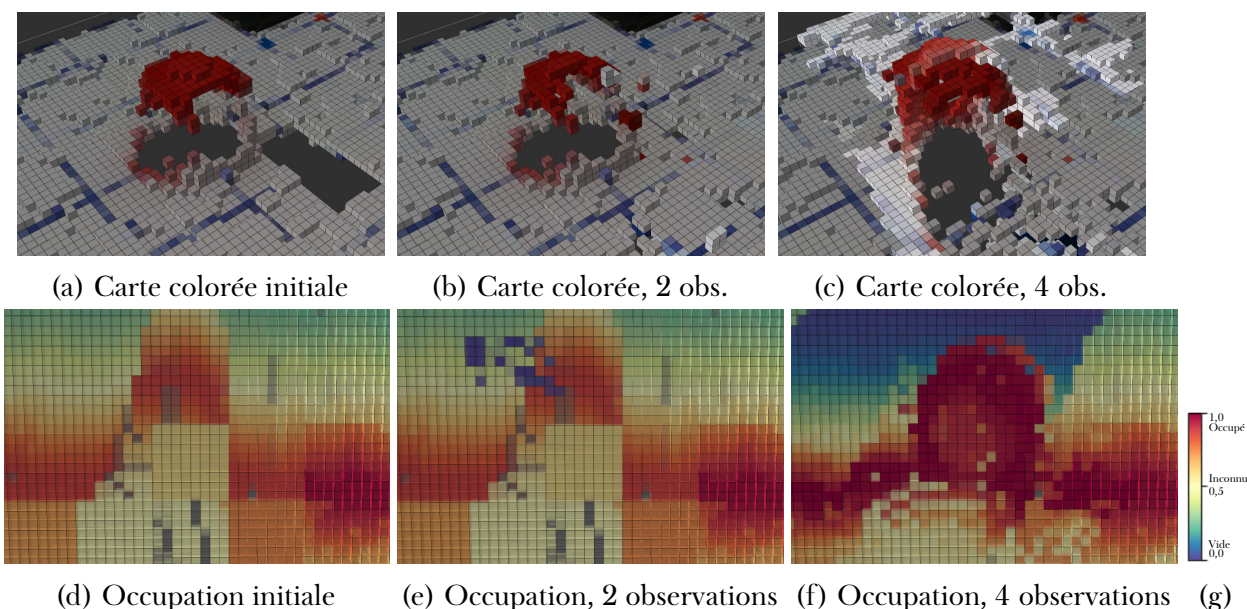
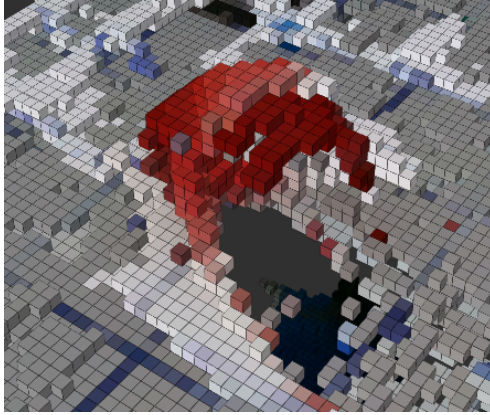


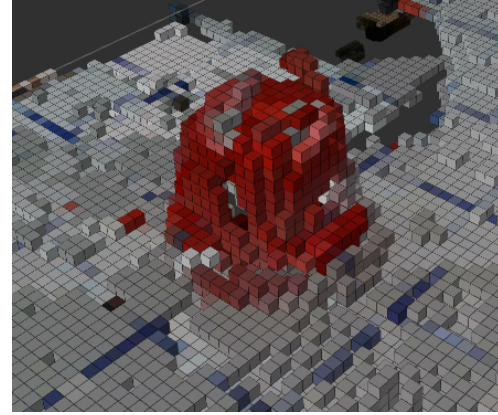
FIGURE 6.21 : Scénario semblable à la figure 6.20 avec un modèle probabiliste. La convergence demande plus d’observations mais réduit éventuellement les erreurs.

Planification de mouvement et division en groupes Avec le robot physique, le positionnement de la caméra au point de vue sélectionné n’est pas garanti et dépend du positionnement de l’objet dans l’espace de travail ainsi que des capacités de l’algorithme de planification de mouvement et du contrôleur du robot. Puisque tous les points de vue proposés ne permettront pas d’obtenir une observation, le nombre de propositions est un facteur important dans le résultat final. Selon l’algorithme d’observation (6.6), si les voxels inconnus ne sont pas divisés en groupes, on obtient trois points de vue potentiels et donc six essais de planification de mouvement par observation. Si la planification de mouvement échoue à trouver une solution pour ces contraintes, il est impossible d’observer la forme complète de l’objet. Si on divise les voxels inconnus en groupes, on obtient plus de points de vue potentiels et donc de meilleures chances d’observer l’objet complet, bien que le résultat final dépende encore du positionnement de l’objet et des autres facteurs aléatoires en jeu. La différence de résultat peut être observée à la figure 6.22. Sans la division des voxels inconnus en groupes (figure 6.22 a et c), le nombre de points de vue proposés est limité. Dans le cas présenté, il est seulement possible d’observer environ 25% de l’objet d’intérêt avant d’atteindre une situation où aucun des points de vue proposés ne peut être rejoint par le robot. Avec la division en groupes et le même objet placé au même endroit (figure 6.22 b et d), on peut observer une plus grande partie de l’objet avant que tous les points de vue proposés ne soient inaccessibles. Notons que l’observation complète d’un objet d’intérêt est difficile à obtenir dans l’environnement physique puisqu’elle

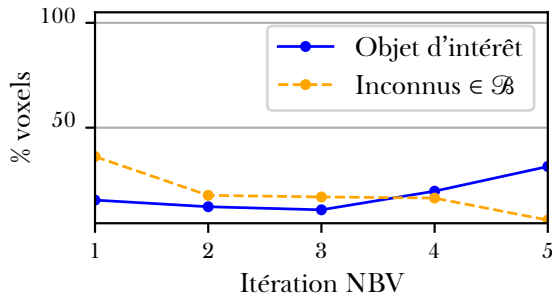
dépend de l'erreur de la caméra, du positionnement de l'objet dans l'espace de travail du robot, ainsi que des facteurs non déterministes dans la planification de mouvement et les différents algorithmes.



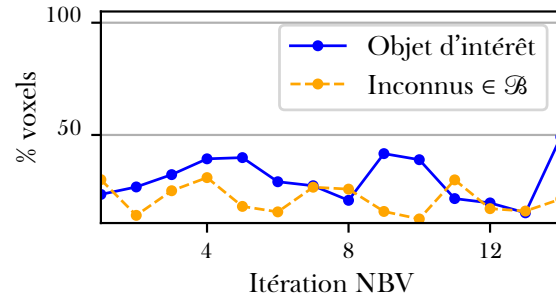
(a) Carte colorée finale sans division en groupes



(b) Carte colorée finale avec division en groupes



(c) Évolution des voxels connus sans division en groupes



(d) Évolution des voxels connus avec division en groupes

FIGURE 6.22 : Observations par le robot physique avec et sans division en groupes. Les graphiques en (c) et (d) comparent le nombre de voxels dans l'objet par rapport à sa forme de référence.

6.3.3 Comparaison expérimentale

Pour comparer les résultats obtenus avec une méthode de référence, on utilise la méthode discutée dans [102] qui consiste à distribuer un nombre fixe de points de vue visant le centroïde de l'objet d'intérêt sur la demi-sphère entourant (la partie connue de) celui-ci. Quelques modifications sont apportées à cette méthode pour en simplifier l'intégration dans le système global :

1. Plutôt que d'utiliser un réseau neuronal pour prédire la qualité de chaque point de vue, on compte directement le nombre de voxels inconnus présents dans l'intersection de la boîte englobante de l'objet et d'un modèle prédéfini du champ de vision de la caméra.
2. La distribution optimale des points de vue sur la demi-sphère étant un problème complexe [206], on opte pour une distribution aléatoire en assurant une distance euclidienne minimale entre les points d'observation.
3. Les points de vue proposés par cette méthode sont classés en ordre décroissant d'optimalité et utilisés dans les algorithmes d'observation de la section 6.2.

Concernant les stratégies pour limiter le nombre d'observations par itération, on note que celle d'observer tous les groupes (qui correspondent ici aux points de vue proposés) a peu d'intérêt avec la méthode de référence, qui propose toujours le même nombre de points de vue par itération, et revient ici à faire une distribution uniforme des points d'observation. Similairement, limiter le nombre d'observations au numéro d'itération est une simple croissance linéaire vers ce même cas. Les résultats de la section 6.3.1.1 ont démontré qu'il est peu efficace de prendre une seule observation par itération, en particulier lorsque le temps de calcul pour proposer des points de vue est important. L'option privilégiée avec la méthode de référence est donc de prendre des observations jusqu'à ce qu'on utilise un point de vue pour lequel le nombre de voxels inconnus intersectant le champ de vision de la caméra est plus faible que la moyenne.

Le tableau 6.5 présente les résultats obtenus avec la méthode de référence, rapportant d'une manière semblable au tableau 6.2 le nombre d'observations totales effectuées, le nombre de calculs NBV effectués, ainsi que le temps total de calcul, le pourcentage de voxels dans la forme observée par rapport à la forme de référence, ainsi que la condition de fin atteinte. Le nombre de points de vue proposés par calcul, n , est fixé à 10 pour équilibrer le temps de calcul et la distribution des points de vue. La distance minimale entre les points de vue est fixée à $2\pi r/n$ où r est la distance d'observation, soit 1,0 m pour le cylindre et 0,5 m pour les autres objets. Dans les conditions présentées, la méthode de référence requiert en moyenne

9,40 secondes pour générer et évaluer la qualité de chaque point de vue. De plus, à l'exception de la longue éponge pour laquelle la procédure de remplissage permet de mieux compenser pour les voxels manquants, la méthode de référence est incapable d'observer complètement les objets malgré un grand nombre de points de vue proposés et d'observations. Ceci est particulièrement évident dans les cas d'objets comme le cylindre et le lapin, où des points de vue visant le centroïde de la partie connue sont incapables d'observer les parties manquantes.

TABLEAU 6.5 : Comparaison de la performance des calculs NBV entre la méthode de référence [102] et la méthode proposée (observations limitées selon la taille du groupe de voxels observé)

Objet (résolution)	Nb. obs.	Nb. NBV	Temps NBV total (s)	% voxels final	Fin
Méthode de référence					
Sphère (8 mm)	66	8	632,25	64,66	Interrompu
Sphère (6 mm)	46	10	714,95	35,54	Interrompu
Longue éponge	12	3	56,55	86,92	Complet
Cylindre	22	5	1338,99	46,81	Interrompu
Lapin	47	10	329,82	39,13	Interrompu
Méthode proposée (tableau 6.2)					
Sphère (8 mm)	3	3	0,37	104,18	Complet
Sphère (6 mm)	4	3	0,60	106,32	Complet
Longue éponge	9	9	1,03	95,48	Pas de progrès
Cylindre	33	11	14,85	100,19	Pas de progrès
Lapin	48	24	53,72	103,48	Complet

6.4 Sélection d'un point de vue pour le suivi de la manipulation

En plus de l'observation de la forme initiale de l'objet, il est nécessaire de sélectionner des points de vue pour observer l'objet en cours de manipulation de manière à effectuer un suivi de forme, qui peut notamment être utile pour le contrôle de la manipulation. En pratique, le positionnement optimal de la caméra pour ce suivi dépend du nombre et de la distribution des points de prise sur l'objet (qui sera étudiée au chapitre 7), du type de prise

et de préhenseur, ainsi que des autres contraintes de la tâche à effectuer. Pour proposer un positionnement adapté aux conditions expérimentales, on suppose qu'une seule caméra est disponible, que les mouvements des points de prise sont encodés dans une carte M_a^v telle que chaque point de prise p effectuera un mouvement linéaire suivant un vecteur $\Lambda_{M_a^v}(p) - p$ sur une distance $\Lambda_{M_a^d}(p)$ selon la définition de la section 3.3.2, et que les occlusions proviennent principalement de l'objet manipulé et non du manipulateur lui-même. L'objectif est d'assurer un suivi minimal des régions à haute déformation (proches des points de prise) pendant le mouvement, puis d'observer à nouveau la forme générale de l'objet une fois le mouvement complété.

Le positionnement proposé de la caméra pour le suivi de la manipulation est basé sur les mêmes principes que pour l'observation des régions inconnues (section 6.1), c'est-à-dire que le plan d'image devrait être aligné sur le plan où on souhaite la plus grande précision, et que la caméra devrait être positionnée du côté de ce plan opposé au centroïde de l'objet manipulé pour réduire les occlusions. Ainsi, on reprend le calcul des moments 3D (annexe A) en considérant les points de départ et d'arrivée des vecteurs de déplacement associés aux points de prise, pondérés par la longueur du déplacement $\Lambda_{M_a^d}(p)$ selon l'équation (6.1) où P est l'ensemble des points de prise p sélectionnés et λ représente les coordonnées du voxel associé au point p dans M_a , c'est-à-dire $\Lambda_{M_a^v}(p)$. Le plan le mieux ajusté et le centroïde des vecteurs de déplacement des points de prise sont calculés de la même manière qu'à l'annexe A.

$$M_{ijk} = \sum_{p \in P} \left(x_p^i y_p^j z_p^k \Lambda_{M_a^d}(p) \right) + \sum_{\lambda = \Lambda_{M_a^v}(p), p \in P} \left(x_\lambda^i y_\lambda^j z_\lambda^k \Lambda_{M_a^d}(p) \right) \quad (6.1)$$

Ces informations permettent de positionner la caméra de la manière illustrée à la figure 6.23, c'est-à-dire avec son axe optique perpendiculaire au plan le mieux ajusté au déplacement prévu des points de manipulation et intersectant le centroïde de ceux-ci. Comme pour l'observation de la forme de l'objet, ceci mène à deux possibilités pour le positionnement de la caméra. Pour minimiser les occlusions par l'objet, on choisit le point du côté où la caméra sera plus proche du centroïde des mouvements prévus que de celui de l'objet manipulé.

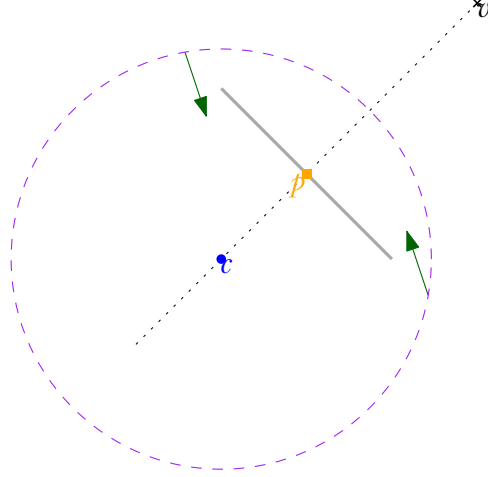


FIGURE 6.23 : Positionnement de la caméra pour le suivi de la manipulation. La caméra est placée au point v avec son axe optique (ligne pointillée) pointé vers le centroïde p des vecteurs de déplacement des points de manipulation (verts) et perpendiculairement au plan le mieux ajusté à ceux-ci (barre grise pleine), du côté le plus éloigné du centroïde c de l'objet manipulé (cercle rose pointillé).

Une fois que le mouvement est terminé, on déplace à nouveau la caméra pour observer de manière plus complète la forme finale de l'objet, permettant d'évaluer la correspondance de celle-ci avec la forme cible. Pour ce faire, on commence par réinitialiser la carte d'occupation M_o à l'état « non exploré » dans la région de l'objet d'intérêt M_m , c'est-à-dire que, pour tout voxel $v \in B$, $\Lambda_{M_o}(v) \leftarrow \emptyset$ et $\Gamma_{M_o}(v) \leftarrow 0$, où $B = k\mathcal{B}_{M_m}$ est la boîte englobante de l'objet courant agrandie d'un facteur k (arbitrairement 2) dans chaque dimension. Ensuite, on prend une observation à partir du point de vue ayant servi à suivre la manipulation avant de répéter la procédure d'observation décrite à la section 6.2 pour reconstruire la carte de l'objet d'intérêt. Cette réobservation après la manipulation permet d'évaluer la qualité du suivi de forme par rapport à la forme finale de l'objet ainsi que de valider la qualité du remodelage, deux éléments qui seront évalués expérimentalement au chapitre 8.

6.5 Sommaire

Contrairement aux objets rigides, la forme d'un objet déformable 3D ne peut pas être déterminée en la faisant correspondre avec un modèle connu et doit plutôt être complètement observée avant, après et pendant la manipulation. Ce chapitre propose une méthode rapide pour identifier des points de vue convenant à l'observation des parties manquantes d'un objet d'intérêt avec un capteur RGBD. Cette méthode est basée sur l'analyse de la distribution spatiale des portions inconnues de l'objet et est accompagnée de méthodes supplémentaires pour déterminer si une grille de voxels donnée représente la forme complète d'un objet et pour éliminer les voxels non observables. Ces méthodes, indépendantes des propriétés des capteurs et des robots utilisés, sont intégrées dans des algorithmes d'observation qui prennent en compte les suppositions par rapport à l'environnement et aux observateurs disponibles pour optimiser l'observation de l'objet. En étudiant les différents paramètres des algorithmes d'observation proposés, on conclut que la division de l'espace inconnu en groupes de voxels contigus et triés par taille est une stratégie efficace pour proposer des points de vue informatifs pour compléter la forme de l'objet d'intérêt. De même, limiter le nombre d'observations de manière adaptative entre le calcul des points de vue permet des gains d'efficacité par rapport à l'utilisation d'un seul ou de tous les points de vue proposés. La distinction entre la performance des deux limites évaluées, soit une croissance linéaire en fonction du nombre d'itérations complétées et une limite selon la taille des groupes de voxels inconnus observés, est cependant moins nette et dépend de l'objet observé ainsi que des conditions expérimentales.

Par rapport aux méthodes présentées dans la littérature [97, 99, 100, 102], la méthode proposée gagne en efficacité en évitant les calculs coûteux visant à quantifier précisément le volume d'espace inconnu qui peut être observé par les points de vue proposés. Elle délègue également les longs calculs de planification de mouvement à une étape subséquente à la proposition des points de vue, ce qui permet d'éviter ces calculs lorsque seul un sous-ensemble des points de vue proposés est utilisé, ainsi que d'optimiser l'attribution des points de vue à différents observateurs (par exemple plusieurs bras robotiques équipés de capteurs et observant la scène conjointement) sans changer la méthode utilisée pour proposer des points de vue. Ces gains d'efficacité sont importants dans le contexte de la manipulation d'un objet déformable, qui doit être réobservé à plusieurs reprises durant la manipulation pour suivre sa forme changeante. L'évaluation expérimentale montre l'efficacité et l'utilité de la méthode proposée, qui est significativement plus rapide que les méthodes alternatives tout en proposant des points de vue qui permettent d'observer la forme complète de l'objet.

Chapitre 7

Comparaison de formes et sélection de prise

Une fois que la forme complète de l'objet d'intérêt est obtenue, on souhaite comparer celle-ci avec la forme cible pour deux raisons. D'abord, puisque la tâche consiste à remodeler l'objet vers la forme cible, il est essentiel de pouvoir déterminer si la forme courante de l'objet est suffisamment proche de la forme cible pour pouvoir considérer la tâche accomplie. Dans une perspective de planification à long terme avec un modèle prédictif, ceci permet également de sélectionner l'action qui rapprochera le plus la forme courante de l'objet de la forme cible. Ensuite, puisque les propriétés du matériau de l'objet à remodeler sont initialement inconnues, on ne peut pas utiliser un modèle ou une approche en simulation pour optimiser le positionnement initial des points de prise. On doit donc se tourner vers une mesure géométrique résultant de la comparaison pour effectuer cette sélection. Ainsi, ce chapitre présente une mesure originale de similarité de formes qui est adaptée à la réalité du remodelage des objets déformables. On commence par présenter la méthode utilisée pour aligner la pose 3D de la forme initiale et de la forme cible (section 7.1) avant de détailler la méthode proposée pour la comparaison (section 7.2). Des méthodes pour raffiner les résultats de la comparaison sont discutées à la section 7.3, et la mesure d'association des voxels est évaluée expérimentalement à la section 7.4. L'utilisation des informations provenant de la mesure de comparaison proposée pour la sélection des points de prise est détaillée à la section 7.5 et évaluée à la section 7.6. Cette partie de la recherche a été publiée dans [196].

Au travers de ce chapitre, on évalue les méthodes proposées en utilisant les formes virtuelles montrées à la figure 7.1, voxélisées à partir de modèles maillés avec des voxels binaires de 6 mm.

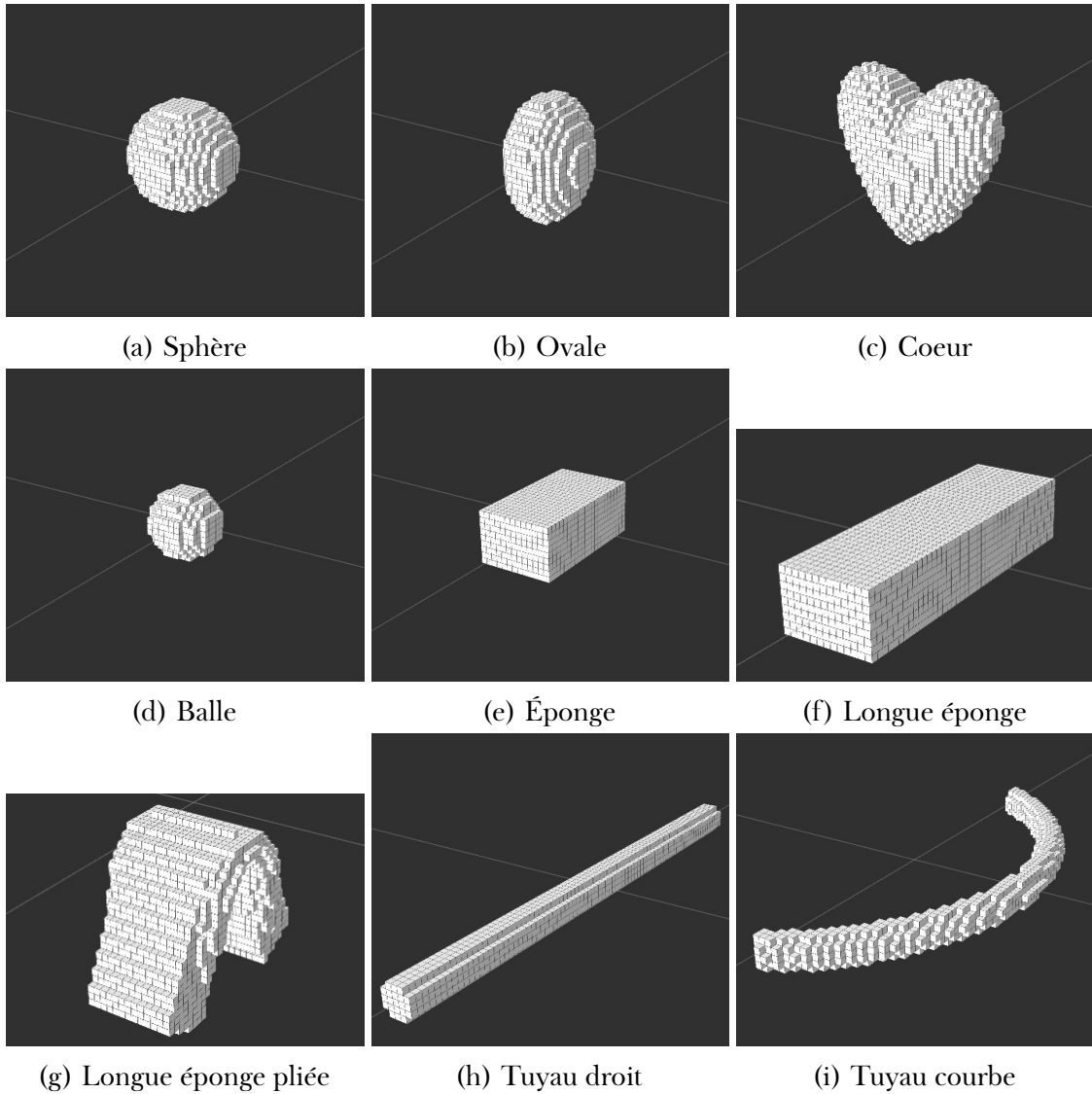


FIGURE 7.1 : Formes utilisées pour tester la comparaison de formes 3D (voxels de 6 mm)

7.1 Alignement

Pour optimiser la comparaison, il est utile de considérer séparément la pose et la forme des objets. En effet, s'il n'y a aucune intersection entre les objets, la différence (en nombre de voxels) entre les objets sera toujours égale à leur taille, ce qui n'est pas très utile pour la planification. Ainsi, il importe d'abord d'obtenir la transformation qui permet d'aligner le centroïde et les axes principaux de l'objet d'intérêt sur ceux de la cible. Ces propriétés des formes peuvent être obtenues par l'analyse des moments tridimensionnels (annexe A) et utilisées pour calculer la transformation nécessaire dans une procédure d'alignement.

Suite à l'analyse des moments, il est possible d'aligner l'objet d'intérêt O sur la forme cible C . Considérant $a_i = \begin{bmatrix} x_{oi} \\ y_{oi} \\ z_{oi} \end{bmatrix}$, $i \in \{1, 2, 3\}$, les axes principaux de l'objet O , et b_i (construit similairement) les axes principaux de l'objet C , la matrice de rotation R telle que $b_i = R a_i$ est obtenue par

$$R = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^{-1} \quad (7.1)$$

La translation nécessaire est pour sa part obtenue plus simplement par la soustraction des centroïdes. Ces deux composantes permettent de développer la matrice de transformation homogène qui permet d'aligner l'objet d'intérêt sur la cible :

$$Tr = \begin{bmatrix} & \bar{x}_c - \bar{x}_o \\ [R] & \bar{y}_c - \bar{y}_o \\ & \bar{z}_c - \bar{z}_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.2)$$

Puisque les axes principaux ne sont pas orientés, il y a $2^3 = 8$ orientations possibles (deux directions pour chaque axe) de l'objet d'intérêt qui sont alignées avec la forme cible. Tel que noté dans [126], pour choisir la meilleure orientation, il est nécessaire de faire huit fois le calcul des équations (7.1) et (7.2) ainsi que la comparaison des formes. Ainsi, il est utile d'avoir un algorithme de comparaison simple et rapide pour aligner les objets avant de faire la comparaison avec un algorithme plus avancé. Comme dans [126], cette comparaison simple peut être faite par le « ou exclusif » des deux objets alignés. Dans un environnement voxélisé, ceci correspond à compter le nombre de voxels différents entre les deux objets ou, considérant les objets O et C comme des ensembles de voxels binaires, $|(O \cup C) - (O \cap C)|$. Dans le cas où les voxels n'encodent pas des valeurs binaires, il est nécessaire de les convertir en voxels binaires ou de définir l'opération du « ou exclusif » d'une manière appropriée. Pour l'alignement des cartes, la mesure de différence proposée entre deux cartes O et C avec le même encodage est calculée par l'équation (7.3), soit la somme des valeurs absolues de la différence de valeur des voxels correspondants dans les deux cartes. Cette mesure est équivalente au « ou exclusif » pour des voxels binaires tout en permettant la comparaison de valeurs plus complexes.

$$d_{diff} = \sum_{x,y,z} |\Lambda_O(x, y, z) - \Lambda_C(x, y, z)| \quad (7.3)$$

Évaluation expérimentale La figure 7.2 montre quelques exemples de formes alignées selon la procédure ci-dessus. Certaines combinaisons ayant des résultats semblables ou difficiles à visualiser en 2D, elles sont exclues des figures et tableaux subséquents par économie d'espace. En général, les résultats sont visuellement plausibles même lorsque les formes à aligner sont très différentes. Cependant, puisque l'alignement est basé sur l'analyse des centroïdes et des axes principaux, on note quelques cas qui résultent en un alignement sous-optimal des formes. Ceci se produit lorsque l'alignement des centroïdes des deux formes fait en sorte que toutes les combinaisons d'alignement des axes principaux ne mènent à aucune intersection entre les deux formes (p. ex., figure 7.2g), ou lorsque la déformation souhaitée modifie l'ordre des axes principaux d'un objet (à la figure 7.2f, l'éponge droite en bleu est tournée de 90° par rapport à l'orientation optimale pour effectuer la déformation).

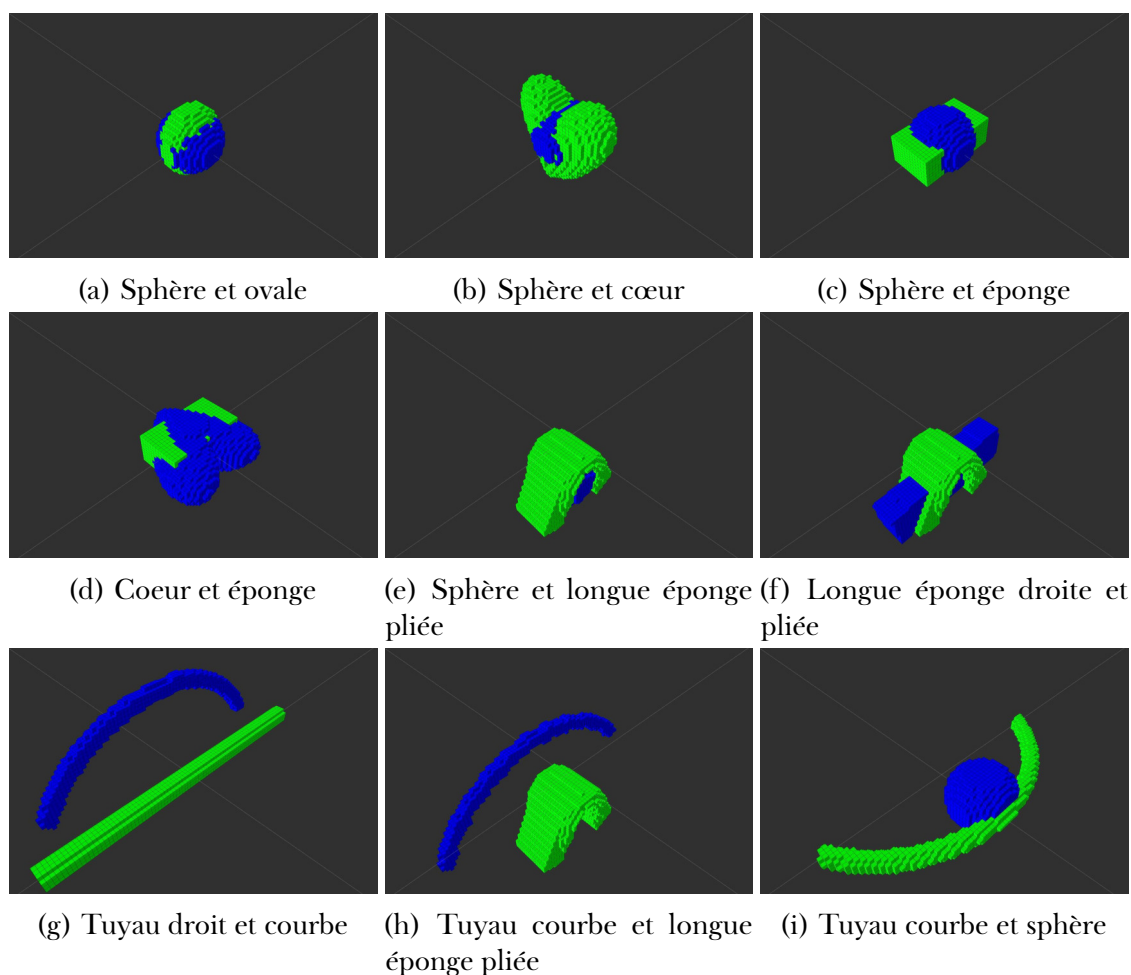


FIGURE 7.2 : Exemples de formes alignées

7.2 Association des voxels

Une fois les formes alignées, une comparaison plus approfondie est requise. Puisque l'équation (7.3) est principalement une mesure de l'intersection des formes, elle ne fournit pas beaucoup d'informations sur la similarité de formes qui ont une petite intersection ou lorsqu'une forme est complètement incluse dans l'autre. Pour pallier à ces limites et obtenir une mesure plus utile au remodelage d'objets déformables, la méthode originale présentée dans cette thèse propose plutôt d'établir une correspondance entre les voxels de l'objet et de la cible. Cette association de la totalité des voxels de la forme courante à ceux de la forme cible représente un champ vectoriel des mouvements désirés des voxels, dont la somme des longueurs des vecteurs est une mesure de la différence entre les formes. Le champ vectoriel lui-même est aussi interprété comme une mesure des zones subissant la plus grande déformation et utilisé comme heuristique pour diriger la planification des mouvements du manipulateur robotique. Le principe de cette association est illustré pour un cas 2D à la figure 7.3.

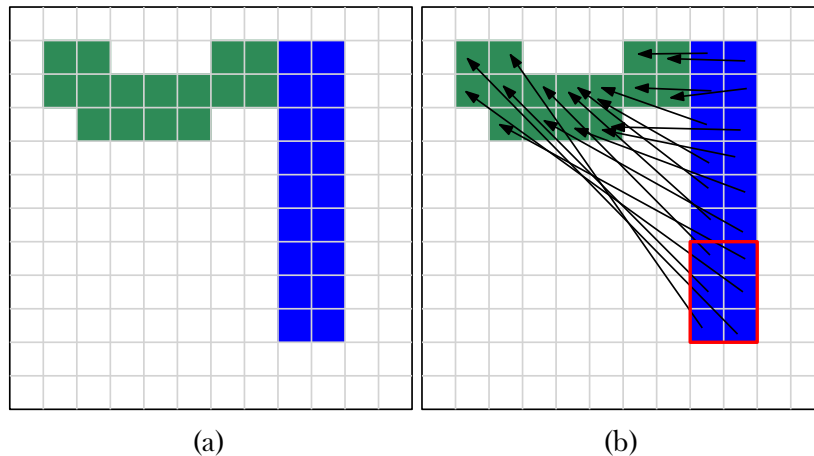


FIGURE 7.3 : (a) Exemple de formes initiale (bleu) et cible (vert) et (b) association entre leurs « voxels » montrant la région qui subit le plus grand changement (cadre rouge).

L'idée de mesurer la distance (totale ou moyenne) que les voxels de la forme initiale doivent parcourir pour correspondre à ceux de la forme cible a été proposée dans [128] comme mesure plus appropriée que la différence des volumes pour la comparaison des formes. La stratégie décrite dans cet article commence par diviser les parties non-correspondantes des formes en voxels positifs (voxels de la forme initiale qui ne sont pas sur la forme cible) et négatifs (ceux de la forme cible qui ne sont pas sur la forme initiale). Ensuite, il s'agit d'établir une correspondance entre les voxels positifs et négatifs pour pouvoir calculer la distance que chaque voxel doit parcourir. Plusieurs stratégies ont été développées pour établir cette

correspondance entre les voxels, que ce soit par la simple distance (euclidienne ou *earth-mover*) ou par la création d'un graphe bipartite et l'utilisation d'un algorithme d'optimisation des connections [129, 130].

Cependant, il est à noter que ces algorithmes considèrent des formes normalisées en volume, c'est-à-dire que les formes comparées ont le même nombre de voxels et qu'on établit une correspondance un-à-un complète entre les voxels positifs et négatifs. Bien que ceci soit utile pour comparer des formes à des échelles différentes, il importe de se rappeler que notre objectif est de déformer un objet pour le faire correspondre le plus précisément possible à une forme cible. Puisque les objets non rigides peuvent changer de volume s'ils sont comprimés ou étirés, il est utile de conserver la différence entre les volumes des objets. De plus, la normalisation en volume fait en sorte que les voxels des deux objets ne représentent pas les mêmes dimensions physiques, ce qui n'est pas souhaitable ici. Ainsi, puisque l'objet d'intérêt pourrait avoir un nombre de voxels différent de la forme cible, la correspondance un-à-un employée dans ces algorithmes ne permet pas d'obtenir un portrait complet de la déformation nécessaire. Enfin, lorsqu'on associe les voxels de la forme originale à la forme cible, la réalité physique et l'intégrité de l'objet doivent être respectées — deux voxels qui sont voisins dans la forme originale ont peu de chances d'être déplacés à des extrémités opposées dans la forme cible. Ainsi, il est inadéquat d'associer uniquement les voxels qui sont différents entre les deux formes.

Pour que la comparaison donne une idée de la déformation nécessaire, il est pertinent d'inclure la totalité des voxels des deux formes dans l'association plutôt que seulement ceux qui sont différents. Contrairement aux autres méthodes faisant l'association entre les voxels ou points de formes 3D [80, 128–130], la méthode proposée dans cette thèse ne cherche pas seulement à obtenir une mesure de similarité en calculant la distance totale ou moyenne que les voxels doivent parcourir, mais plutôt un champ vectoriel qui décrit le mouvement nécessaire des différentes parties de l'objet. Il est donc essentiel de définir cette association entre les voxels des deux formes à comparer d'une manière qui prend en compte les déplacements possibles, et de conserver les vecteurs résultant de l'association.

Ainsi, la méthode originale proposée pour faire l'association des voxels entre deux grilles, illustrée à la figure 7.4, se déroule en deux étapes qui ont chacune diverses configurations possibles. Pour ces étapes, on définit O et C , deux grilles de voxels représentant respectivement la forme originale et la forme cible. La première étape consiste à créer un graphe G reliant les voxels de O et de C . Chaque nœud du graphe représente un voxel et contient ses coordonnées ainsi qu'un identificateur spécifiant s'il représente un voxel de O ou de C . Tous les nœuds n_o

représentant un voxel de O sont reliés à tous les nœuds n_c représentant un voxel de C . L'arête reliant les nœuds n_{oi} et n_{cj} a un poids w_{oicj} qui est déterminé en fonction des coordonnées, de la valeur et de l'état d'activité des voxels qu'ils représentent.

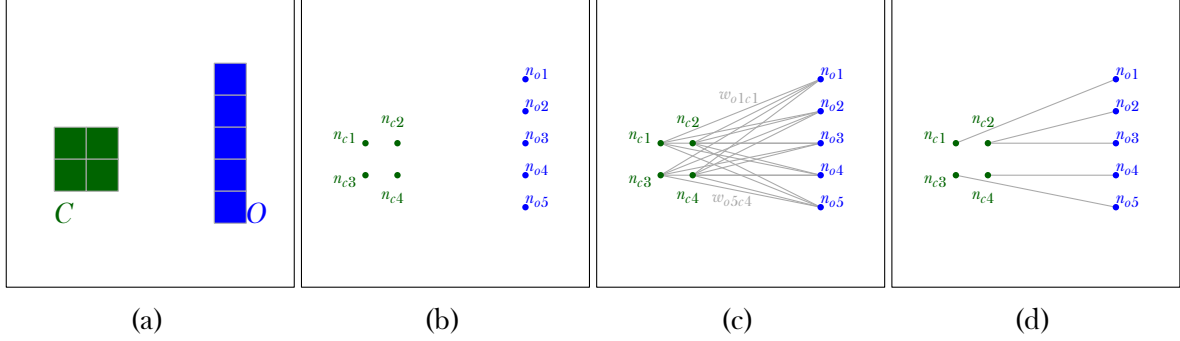


FIGURE 7.4 : Association de voxels avec graphe bipartite. À partir des voxels des formes originale O et cible C (a), on crée les nœuds correspondants n_{oi} , n_{cj} (b) et les arêtes de poids w_{oicj} (c). On sélectionne les arêtes à conserver (d) selon la méthode appropriée. Celles-ci sont orientées de O à C pour donner un champ vectoriel comme à la figure 7.3.

Pour planifier la déformation, le poids d'une arête est principalement déterminé par la distance euclidienne entre les voxels qu'elle connecte, mais il peut être souhaitable de ne pas connecter les voxels dont l'état d'activité ou la valeur sont trop différents. L'équation (7.4) représente la fonction proposée pour déterminer le poids w de l'arête reliant les nœuds n_o et n_c , où a et s sont des paramètres qui définissent respectivement si on souhaite considérer l'état d'activité et le signe des voxels et où $\text{coords}(n)$ représente les coordonnées du voxel représenté par le nœud n . Un poids infini implique que ces deux nœuds ne devraient pas être connectés. Le signe et l'état d'activité sont particulièrement pertinents lorsque les cartes d'entrée sont au format *Level-Sets*, puisqu'il est ainsi possible de différencier les voxels intérieurs de la surface lors de l'association. Ainsi, le résultat de la première étape est un graphe bipartite reliant les voxels de la forme initiale à ceux de la forme cible, dont les arêtes sont pondérées par la distance que les voxels de la forme initiale doivent parcourir pour atteindre la position correspondante sur la forme cible.

$$w(n_o, n_c, a, s) = \begin{cases} \infty & \text{si } a \wedge (\Gamma(n_o) \neq \Gamma(n_c)) \\ \infty & \text{si } s \wedge (\text{signe}(\Lambda(n_o)) \neq \text{signe}(\Lambda(n_c))) \\ \|\text{coords}(n_o) - \text{coords}(n_c)\| & \text{autrement} \end{cases} \quad (7.4)$$

Une fois que ce graphe est construit, la deuxième étape est de sélectionner les arêtes qui offrent la meilleure association. Trois méthodes tirées de la théorie des graphes sont étudiées :

- **Association complète de poids minimal**¹ : Dans un graphe bipartite avec les partitions (U, V) , on choisit l'association de taille $\min(|U|, |V|)$ qui minimise la somme du poids des arêtes sélectionnées. Ainsi, si les grilles O et C n'ont pas le même nombre de voxels, les voxels supplémentaires de la grille la plus grande n'auront pas d'association.
- **Couverture d'arêtes de poids minimum**² : On choisit un ensemble d'arêtes qui fait en sorte que tous les nœuds soient inclus, tout en minimisant la somme du poids des arêtes sélectionnées.
- **Couverture d'arêtes de cardinalité et poids minimums** : On choisit un ensemble d'arêtes qui fait en sorte que tous les nœuds soient inclus, en minimisant d'abord le nombre d'arêtes choisies puis la somme du poids des arêtes. Ceci est fait en commençant avec l'association complète de poids minimal et en y ajoutant l'arête de plus faible poids pour chaque nœud non inclus dans l'association.

Les trois méthodes sont illustrées avec un exemple 2D à la figure 7.5. En considérant le contexte physique du remodelage d'objets déformables, on observe que l'association complète de poids minimal n'est pas une option appropriée puisqu'elle laisse des voxels sans correspondance. En effet, pour planifier la déformation, il est nécessaire que tous les voxels soient pris en compte. La distinction entre les deux autres méthodes est plus subtile. Puisque le but de cette association est de déterminer le déplacement des voxels de la forme originale vers la forme cible, une association de cardinalité autre que 1 : 1 implique que des voxels soient fusionnés ou divisés alors que l'objet est comprimé ou étiré. Sans autres informations sur la manière dont la déformation est effectuée, on peut supposer que l'étirement ou la compression est distribué assez uniformément sur l'objet, c'est-à-dire que plusieurs voxels se divisant en deux est plus probable qu'un voxel se divisant en un grand nombre. Cette logique correspond à la couverture de cardinalité et poids minimum. La logique inverse, soit de concentrer l'étirement ou la compression, pourrait cependant être plus appropriée dans certaines applications et correspond à la couverture de poids minimum. Les trois méthodes d'association demeurent accessibles dans l'interface de filtres du module de cartographie, permettant de choisir la plus appropriée selon l'application.

1. *minimum weight full matching*

2. *minimum weight edge cover*

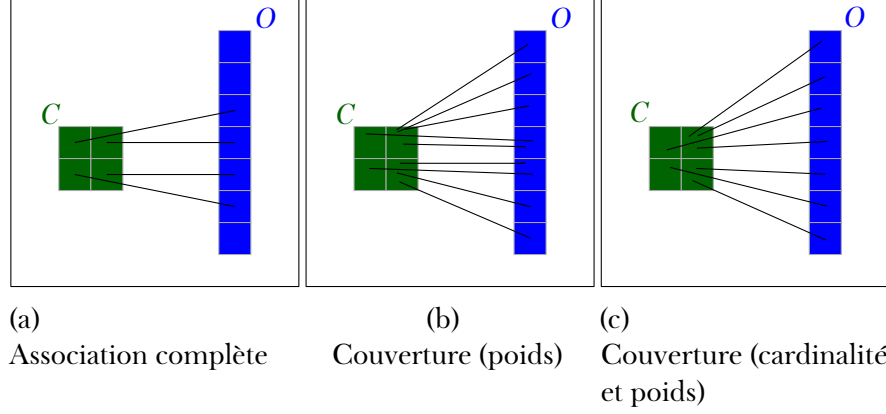


FIGURE 7.5 : Comparaison des trois méthodes d'association avec un cas 2D.

Une fois l'association effectuée, elle est encodée dans une nouvelle carte à valeur vectorielle M_a . Ainsi, pour chaque voxel de la grille O , le voxel correspondant de M_a encode les coordonnées du ou des voxel(s) de la grille C qui y est associé. C'est-à-dire que pour un voxel $v \in O$, $\Lambda_{M_a}(v) = c$ implique que v doit être déplacé au point c pour remodeler l'objet O en la forme C . Par souci d'efficacité, M_a inclut une carte d'associations M_a^v et une carte de distances M_a^d telles que $\Lambda_{M_a^v}(v) = c \implies \Lambda_{M_a^d}(v) = \|v - c\|$, ce qui permet de mettre les distances à jour et de les interroger sans devoir les recalculer pour la carte complète à chaque fois. Une mesure de la distance entre les formes représentées par O et C peut donc être obtenue par la somme de M_a^d (équation (7.5)). Le champ vectoriel encodé dans M_a^v fournit des informations sur le déplacement souhaité des différentes parties de l'objet et est utilisé comme mesure heuristique pour guider les algorithmes de planification du remodelage.

$$d_{assoc} = \sum_{v \in M_a^d} \Lambda_{M_a^d}(v) \quad (7.5)$$

7.3 Raffinement de l'association des voxels

Une fois qu'une association est obtenue entre les voxels de la forme originale et ceux de la forme cible, il convient d'effectuer quelques ajustements pour améliorer la qualité de cette association en tant que mesure heuristique pour la sélection de prise et le remodelage de l'objet. Ceux-ci sont l'application d'un filtre de moyennage pour lisser la mesure ainsi que l'ajustement des vecteurs de déplacement calculés pour prendre en compte les points fixes définis par l'utilisateur.

7.3.1 Filtre de moyennage

Expérimentalement, on remarque que le champ vectoriel dense résultant de la procédure d'association décrite précédemment contient une certaine quantité de bruit (figure 7.6a). En effet, selon la taille des voxels et la méthode de pondération des arêtes, plusieurs d'entre elles peuvent avoir un poids similaire, surtout pour des voxels voisins. Ceci permet aux erreurs numériques et à l'ordonnancement non déterministe des structures de données d'introduire un élément stochastique dans le choix des arêtes conservées, menant à un champ vectoriel qui n'est pas lisse. Ceci peut mener à des résultats indésirables lorsqu'on cherche à utiliser des valeurs spécifiques dans ce champ vectoriel pour guider la sélection de prise et le remodelage de l'objet. Pour pallier à cette situation, on applique un filtre de moyennage à la carte d'association M_a (c'est-à-dire à M_a^v et à M_a^d), où chaque voxel v est mis à jour avec la moyenne de ses voisins dans un rayon r , tel qu'indiqué à l'équation (7.6) où on utilise la notation $\Lambda(v) \neq \emptyset$ pour indiquer que v n'est pas un voxel d'arrière-plan, et où N_v est construit à partir du voisinage η_r (équation (4.8)) en sélectionnant uniquement les voxels répondant à ce critère. Ce filtre d'usage général est intégré au module de cartographie.

$$N_v = \{n \in \eta_r(v) \mid \Lambda_{M_a}(n) \neq \emptyset\}$$

$$\Lambda_{M_a}(v) \leftarrow \frac{\sum_{n \in N_v} (\Lambda_{M_a}(n)) + \Lambda_{M_a}(v)}{|N_v| + 1} \quad \forall v \in \mathcal{B}_{M_a} \mid \Lambda_{M_a}(v) \neq \emptyset \quad (7.6)$$

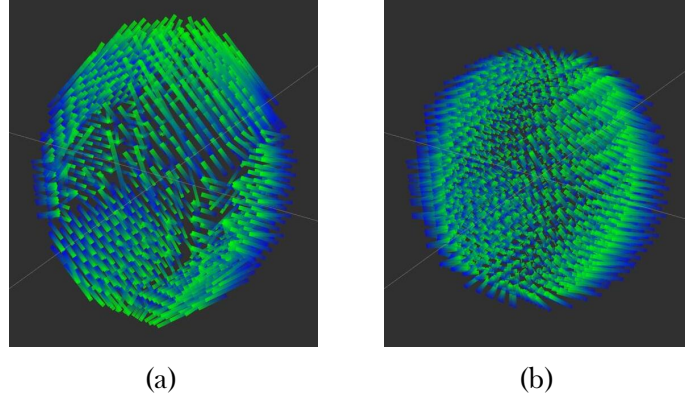


FIGURE 7.6 : Exemple de vecteurs d'association (carte M_a^v) sans (a) et avec (b) le filtre de moyennage.

7.3.2 Ajustement des points fixes

Dans le contexte où l'objet à manipuler peut être attaché de manière rigide à un objet fixe, il est nécessaire d'ajuster les mouvements désirés des voxels pour correspondre à cette

réalité. Tel que discuté à la section 3.3.2 (figure 3.8), ces cas sont déterminés en demandant à l'utilisateur de sélectionner un ou des points à considérer comme fixes sur la forme initiale de l'objet d'intérêt. La procédure d'ajustement des points fixes consiste donc à prendre une grille représentant le mouvement souhaité des voxels de l'objet manipulé, comme celle obtenue par l'association des voxels entre la forme initiale et la forme cible, et à soustraire de chaque vecteur le déplacement calculé pour le point qui a été marqué comme fixe, tel que décrit à l'algorithme 7.1. Puisque le choix des points fixes dans l'interface graphique n'est pas nécessairement précis et que l'association des voxels peut contenir des valeurs erronées en raison du bruit, la valeur de déplacement souhaitée pour le voxel marqué comme fixe est obtenue en prenant la moyenne de ce voxel et de ses voisins η_r , à un rayon configurable r (équation (4.8)). En présence de plusieurs points fixes, il serait nécessaire de vérifier que ces points correspondent à la réalité de la déformation souhaitée et de faire les ajustements nécessaires dans le cas contraire (p. ex., il est impossible de plier en deux un objet droit dont les deux bouts sont fixes). L'algorithme 7.1 considère donc uniquement le cas d'un seul point fixe.

Algorithme 7.1 Ajustement des points fixes

Entrées

- M_a , la grille de voxels à valeurs vectorielles à ajuster
 - 3: p , le point sélectionné comme fixe par l'utilisateur
 - r , le rayon de moyennage
 - $v \leftarrow \text{RECHERCHE}(M_a, p, C : \Lambda(v) \neq \emptyset)$ ▷ algorithme 4.4
 - 6: $N \leftarrow \{n \in \eta_r(v) \mid \Lambda_{M_a}(n) \neq \emptyset\}$ ▷ voisinage, éq. (4.8)
 - $d \leftarrow \frac{\sum_{n \in N} (n - \Lambda_{M_a^v}(n)) + v - \Lambda_{M_a^v}(v)}{|N| + 1}$
 - $\Lambda_{M_a^v}(c) \leftarrow \Lambda_{M_a^v}(c) + d \ \forall c \in M_a$ ▷ M_a^v et M_a^d sont mis à jour simultanément
-

7.4 Évaluation de l'association des voxels

Suite à l'alignement, les formes de la figure 7.1 sont comparées avec le score IoU et l'association des voxels. Puisqu'il s'agit de voxels binaires, l'association est pondérée seulement par la distance entre les voxels, en utilisant la méthode de l'association de cardinalité et poids minimums. Cette section présente d'abord les résultats visuels et numériques pour l'association des voxels avec différentes paires de formes avant de s'intéresser plus en détails à l'effet du filtre de moyennage et de la méthode d'association des graphes.

7.4.1 Résultat de l'association

La figure 7.7 montre les vecteurs résultants de l'association des voxels entre les formes comme des lignes passant du bleu au vert. Globalement, ces résultats montrent un déplacement plausible des voxels pour passer d'une forme à l'autre, même lorsqu'elles sont très différentes ou que l'alignement était sous-optimal. Cependant, on note une certaine quantité de bruit dû à des éléments stochastiques comme les erreurs d'arrondissement numériques ou l'ordonnancement des voxels lors de l'association. Notons également que les voxels qui ne subissent pas de déplacement ne sont pas représentés, et que certains vecteurs peuvent être difficiles à voir en raison de leur taille ou de l'angle de la visualisation.

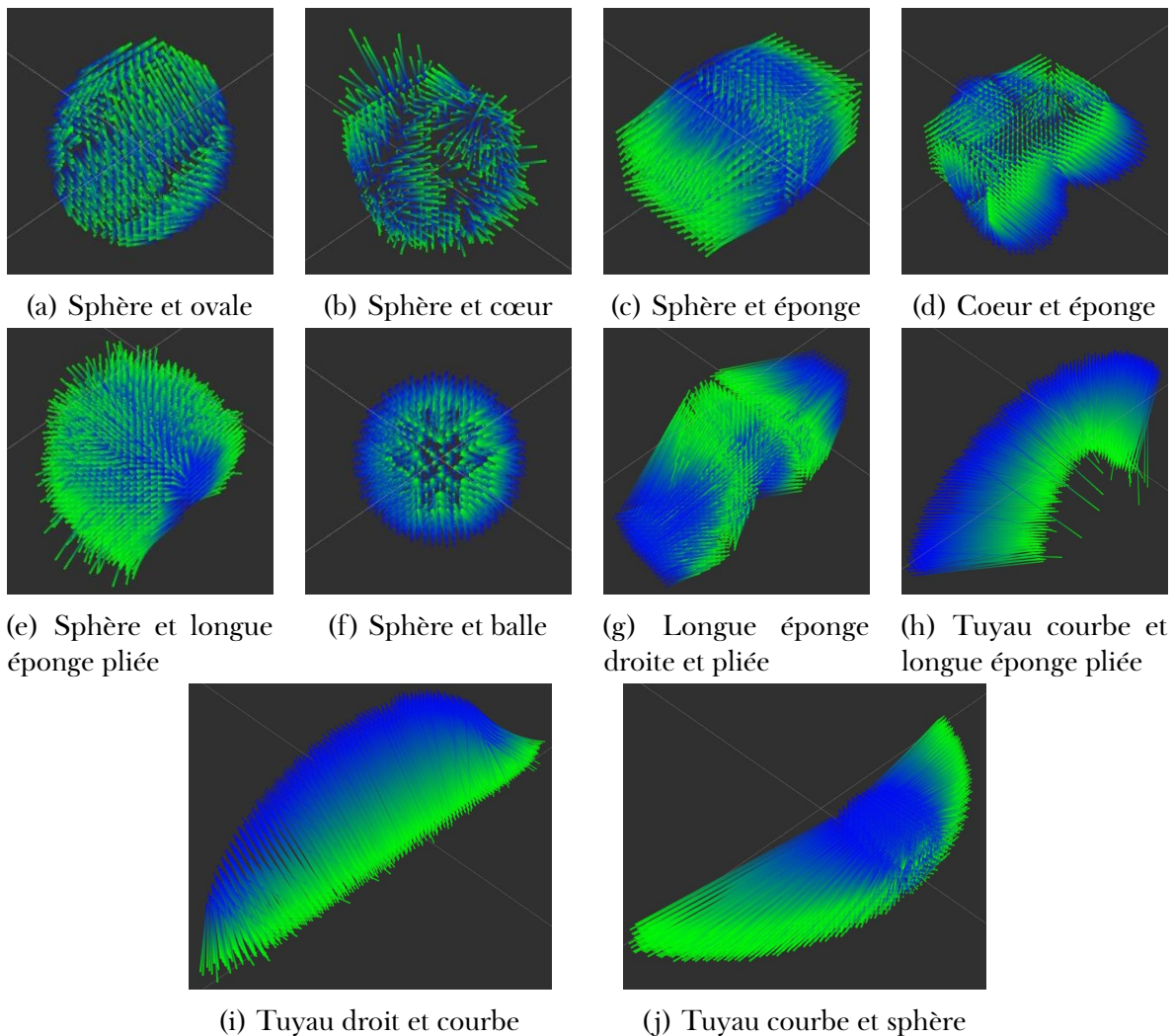


FIGURE 7.7 : Exemples de vecteurs d'association

Le tableau 7.1 présente les valeurs numériques pour différentes méthodes de comparaison de formes. La différence simple utilisée dans l’alignement (d_{diff} , équation (7.3)) est ici équivalente au « ou exclusif » de [126] puisqu’on utilise des voxels binaires, alors que le score IoU est une mesure standard pour la comparaison de formes 2D, reportée ici en 3D (équation (5.9)). Les mesures détaillées originales sont la distance totale d’association des voxels (d_{assoc} , équation (7.5)), ainsi que la moyenne de cette distance, obtenue en la divisant par le nombre de voxels dans la forme originale ($|O|$ ou $|M_a^d|$). Notons que d_{diff} et le score IoU sont des mesures d’intersection de volumes alors que d_{assoc} (totale ou moyenne) est une mesure du travail requis pour transformer la forme originale en la forme cible. À part le score IoU et la moyenne de d_{assoc} , ces valeurs ne sont pas normalisées par le nombre de voxels. Le tableau 7.2 présente les temps de calcul correspondants ainsi que, pour l’alignement et l’association, le temps requis pour effectuer les transformations et autres opérations sur les grilles de voxels en plus de la comparaison. Les temps de calcul sont rapportés pour un traitement avec un seul fil d’exécution sur CPU³ et des algorithmes implémentés avec un minimum d’optimisation du temps de calcul (annexe D).

Dans le tableau 7.1, on remarque que la simple différence et le poids d’association sont plus robustes que le score IoU et permettent une comparaison même lorsqu’il n’y a pas d’intersection entre les formes. Puisque ces mesures ne sont pas normalisées, la différence relative entre plusieurs paires de formes à différentes échelles peut être difficile à interpréter (par exemple, est-ce que la sphère est plus proche de l’ovale que le tuyau droit ne l’est du tuyau courbe?). Cependant, l’interprétation de la distance d’association d_{assoc} en tant que mesure du travail requis pour transformer une forme en l’autre (en termes de déplacement des voxels) offre une piste pour effectuer cette comparaison, particulièrement lorsque les échelles sont semblables (par exemple, transformer la sphère en ovale demande moins de travail que transformer l’éponge rectangulaire en cœur). Les mesures proposées permettent correctement de déterminer quelle forme est la plus proche d’une autre (par exemple, la sphère est plus proche de l’ovale que du cœur) et de suivre si une forme s’est rapprochée ou éloignée de la cible lors de la manipulation.

3. AMD Ryzen 5 3600 et 32 Go de mémoire vive sur un ordinateur de bureau

En termes de temps de calcul, on note que la simple différence utilisée pour l’optimisation de l’alignement (équation (7.3)) est significativement plus rapide à calculer que les autres mesures — bien que la différence et le score IoU aient tous deux une complexité de $O(|\mathcal{B}_{OUC}|)$ pour comparer les grilles de voxels O et C , la différence peut être implémentée avec les algorithmes optimisés d’OpenVDB [93] qui utilisent la structure d’arbre sous-jacente. La procédure d’alignement complète effectue ce calcul huit fois en plus de l’application des transformations sur les cartes. En ce qui concerne l’association, la construction du graphe implique de créer $|O| \times |C|$ arêtes avant d’optimiser les connexions et de créer la grille contenant les vecteurs de déplacement résultant de cette optimisation. C’est un processus qui peut se montrer relativement lent en fonction du nombre de voxels des formes à comparer et de la complexité de l’optimisation des connections. Bien que le temps de calcul pour l’association demeure dans des limites utilisables pour les petits objets, l’algorithme bénéficierait d’une implémentation optimisée et d’une parallélisation des calculs pour faciliter son application à des objets avec un grand nombre de voxels.

TABLEAU 7.1 : Résultats numériques de comparaison de formes 3D

Objet O	Objet C	d_{diff}	score IoU	d_{assoc}	$\frac{d_{assoc}}{ O }$
sphère	sphère	0	1,0000	0,00	0,00
sphère	ovale	956	0,7010	2276,89	0,79
sphère	cœur	3386	0,4409	1698,66	0,59
sphère	tuyau courbe	3807	0,0325	30 348,50	11,05
sphère	tuyau droit	3952	0,0828	34 741,29	12,06
sphère	balle	2082	0,2773	4459,64	1,55
sphère	éponge	2050	0,4621	7174,08	2,49
ovale	cœur	3306	0,4353	1025,34	0,40
ovale	tuyau courbe	3461	0,0437	26 455,97	10,75
ovale	tuyau droit	3460	0,1130	31 451,37	12,30
ovale	balle	1758	0,3125	4020,09	1,57
ovale	éponge	1190	0,6303	3840,11	1,50
cœur	tuyau courbe	5855	0,0573	46 078,56	7,91
cœur	tuyau droit	6534	0,0767	43 473,19	7,45
cœur	balle	5023	0,1372	24 608,10	4,23
cœur	éponge	3908	0,3717	12 740,05	2,18
tuyau courbe	tuyau droit	2920	0,0000	20 663,48	18,21
tuyau courbe	éponge	3687	0,0176	13 595,54	12,05
tuyau courbe	longue éponge	5497	0,0568	6517,95	5,78
tuyau courbe	longue éponge pliée	5519	0,0000	21 177,16	18,74
tuyau droit	éponge	3510	0,1210	25 264,38	14,15
balle	éponge	1992	0,2733	584,45	0,73
balle	longue éponge	4332	0,1474	895,29	1,12
éponge	longue éponge	2340	0,5349	1321,00	0,49
longue éponge	longue éponge pliée	5971	0,2141	24 348,18	5,03

TABLEAU 7.2 : Temps de calcul pour la comparaison de formes 3D (secondes)

Objet <i>O</i>	Objet <i>C</i>	Alignement	Calcul de d_{diff}	IoU	Association des voxels
sphère	sphère	3,67	0,22	0,36	147,47
sphère	ovale	2,85	0,17	0,33	65,54
sphère	cœur	2,98	0,18	0,79	235,69
sphère	tuyau courbe	2,49	0,14	1,54	49,89
sphère	tuyau droit	3,44	0,21	1,55	53,76
sphère	balle	3,39	0,23	0,36	24,19
sphère	éponge	2,97	0,18	0,36	76,58
ovale	cœur	2,86	0,18	0,78	199,27
ovale	tuyau courbe	2,32	0,13	1,13	40,10
ovale	tuyau droit	3,45	0,21	1,14	50,71
ovale	balle	3,30	0,20	0,28	17,78
ovale	éponge	2,89	0,18	0,29	67,43
cœur	tuyau courbe	3,27	0,14	2,07	110,53
cœur	tuyau droit	4,53	0,24	2,60	136,43
cœur	balle	4,19	0,21	0,88	43,60
cœur	éponge	3,80	0,19	0,81	182,36
tuyau courbe	tuyau droit	3,03	0,19	1,89	19,37
tuyau courbe	éponge	3,31	0,19	0,88	28,96
tuyau courbe	longue éponge	3,23	0,20	0,92	56,92
tuyau courbe	longue éponge pliée	2,59	0,14	3,29	49,76
tuyau droit	éponge	3,11	0,20	0,57	49,95
balle	éponge	2,71	0,17	0,18	21,27
balle	longue éponge	2,75	0,18	0,34	36,24
éponge	longue éponge	2,79	0,18	0,27	167,90
longue éponge	longue éponge pliée	2,15	0,09	1,09	338,55

7.4.2 Effet du filtre de moyennage

La figure 7.9 montre le résultat de l'association des voxels pour passer de la sphère à l'ovale (voxels binaires, association de cardinalité et poids minimums). Ces résultats sont constitués de deux valeurs, soit les vecteurs résultant de l'association, représentés par des lignes passant du bleu (sphère) au vert (ovale), et la longueur de chacun de ces vecteurs, projetés sur la sphère selon l'échelle de couleurs de la figure 7.8. Sans le filtre de moyennage, on remarque plusieurs valeurs irrégulières, c'est-à-dire des vecteurs qui divergent significativement de leur entourage par leur orientation ou leur longueur. Ces valeurs résultent d'erreurs d'arrondissement numérique, d'arêtes de poids identiques, et d'autres imperfections semblables dans l'association, et ne correspondent pas au champ vectoriel lisse et dense qui représenterait la déformation graduelle d'une forme à une autre. L'application d'un filtre de moyennage selon l'équation (7.6) corrige efficacement ces erreurs, avec un rayon optimal entre 2 et 3 voxels pour la résolution utilisée à la figure 7.9. Cependant, lorsque le rayon du filtre est trop grand, il peut éloigner le résultat de la réalité de la déformation — avec un rayon de 8 voxels, on remarque même une inversion des zones à haute et à basse déformation. Les figures 7.10 et 7.12 montrent respectivement l'effet du filtre sur le scénario de longue éponge pliée et, pour une meilleure visualisation en 2D, sur une déformation semblable avec une poutre rectangulaire de $15\text{ m} \times 2\text{ m} \times 1\text{ m}$ avec des voxels de 50 cm (figure 7.11). Ces objets rectangulaires supportent l'application d'un filtre de plus grand rayon que la sphère, mais on note que celui-ci peut tout de même modifier significativement la forme résultante. Par exemple, à la figure 7.12, la forme cible (extrémité verte) passe d'une courbe gaussienne à un arc de cercle.

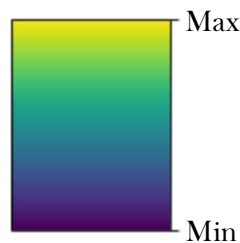
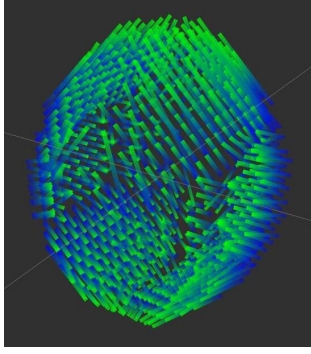
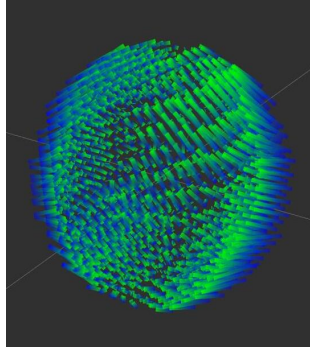


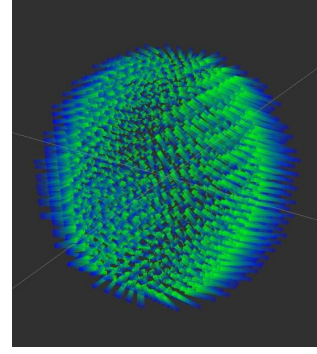
FIGURE 7.8 : Échelle de couleurs pour les distances



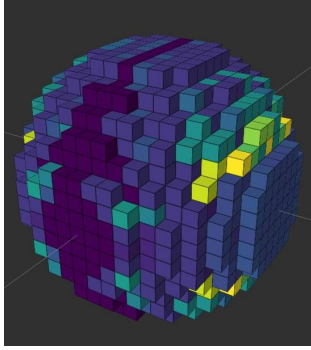
(a) Vecteurs, sans moyennage



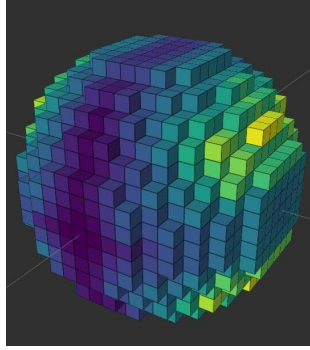
(c) Vecteurs, $r = 1$



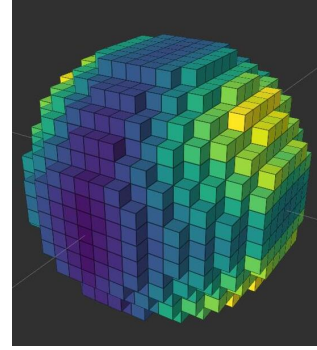
(e) Vecteurs, $r = 2$



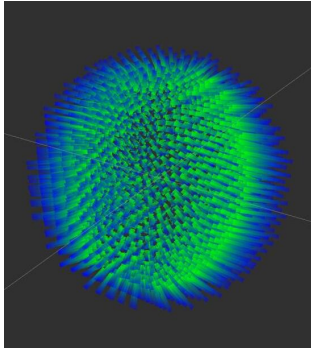
(b) Distances, sans moyennage



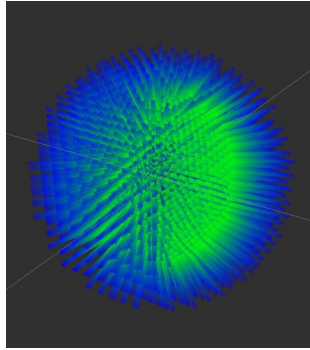
(d) Distances, $r = 1$



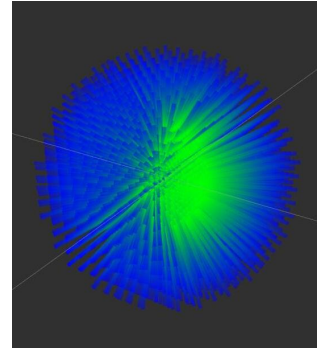
(f) Distances, $r = 2$



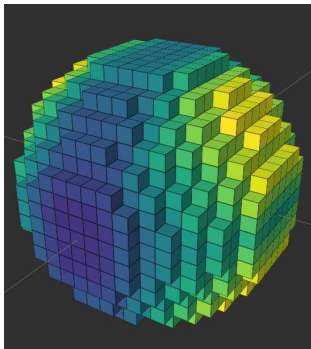
(g) Vecteurs, $r = 3$



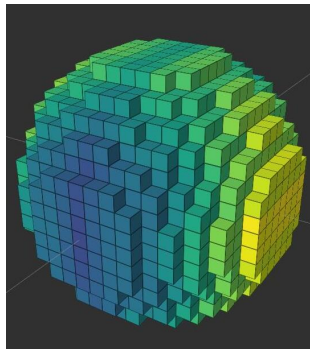
(i) Vecteurs, $r = 5$



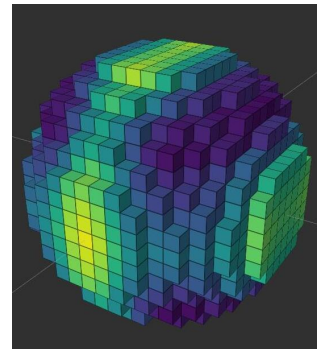
(k) Vecteurs, $r = 8$



(h) Distances, $r = 3$



(j) Distances, $r = 5$



(l) Distances, $r = 8$

FIGURE 7.9 : Effet du filtre de moyennage à différents rayons r (en voxels) après l'association des voxels (sphère à ovale)

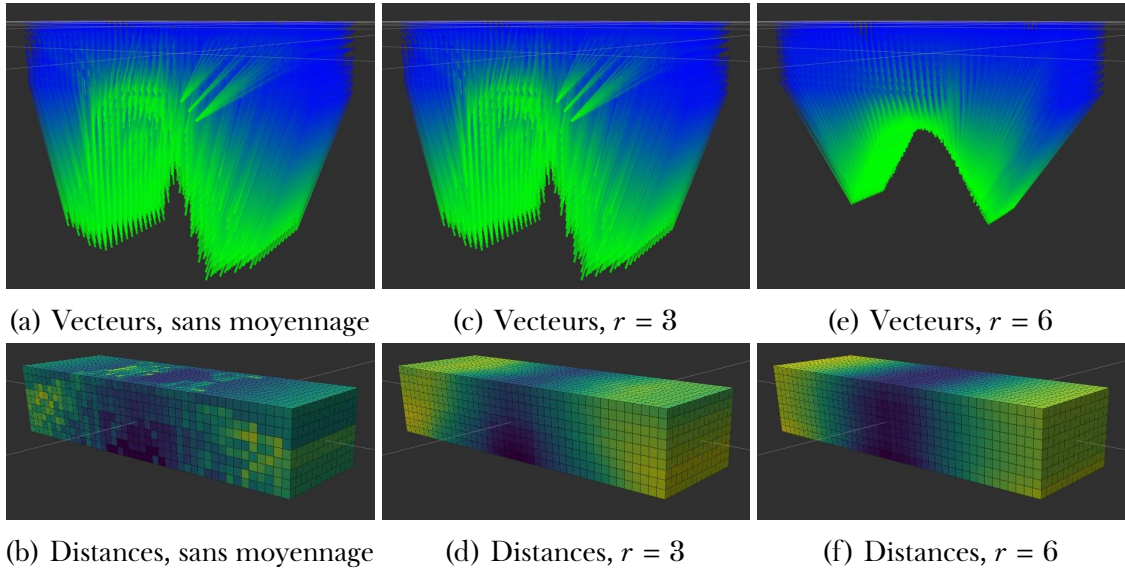


FIGURE 7.10 : Effet du filtre de moyennage à différents rayons r (en voxels) après l'association des voxels (longue éponge droite à pliée)

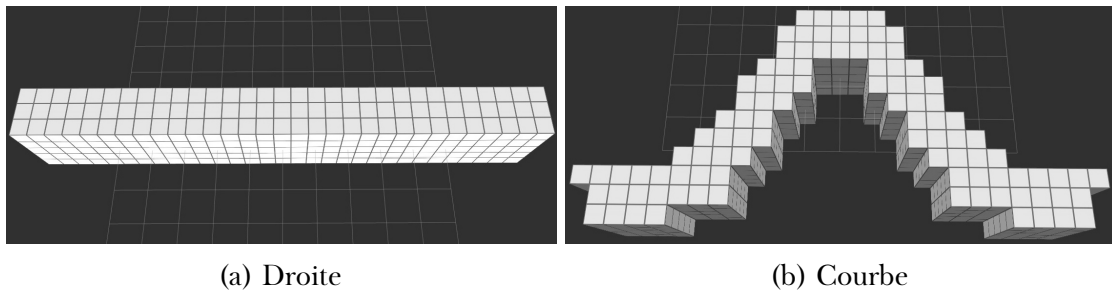


FIGURE 7.11 : Poutre de $15\text{ m} \times 2\text{ m} \times 1\text{ m}$ avec des voxels de 50 cm

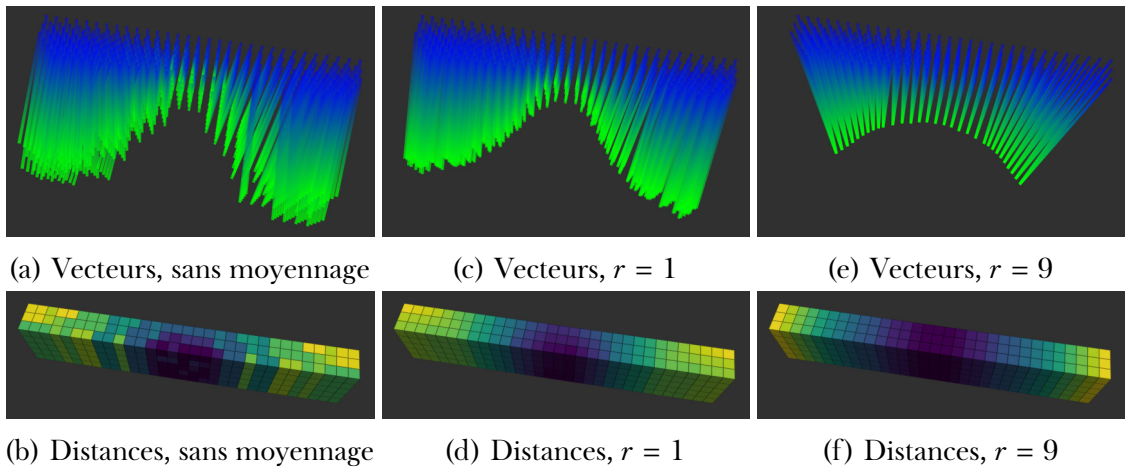


FIGURE 7.12 : Effet du filtre de moyennage à différents rayons r (en voxels) après l'association des voxels (poutre droite à courbe)

7.4.3 Méthode d'association des graphes

La figure 7.13 montre le résultat des différentes méthodes d'association des voxels discutés à la section 7.2 pour l'association du tuyau courbe et de la balle avec des voxels binaires de 6 mm, et la figure 7.14 pour l'association de l'éponge et de la sphère. Les deux figures présentent les vecteurs et les distances après l'application du filtre de moyennage (section 7.3.1) avec un rayon de 1 voxel. On remarque que l'association complète de poids minimal mène à des voxels qui n'ont pas de correspondance (représentés par des traits rouges), ce qui rend cette méthode inutilisable pour guider la déformation d'objets ayant un nombre de voxels différent de la forme cible. La couverture de poids minimal produit un résultat plus uniforme mais, dans les cas présentés, l'extrémité verte des vecteurs ne correspond pas tout à fait à la forme cible — il manque la portion inférieure de la sphère en 7.13b et on voit un ovale plutôt qu'une sphère en 7.14b. La couverture de cardinalité et poids minimaux est la combinaison des deux autres résultats et produit une mesure visuellement raisonnable pour le guidage de la déformation.

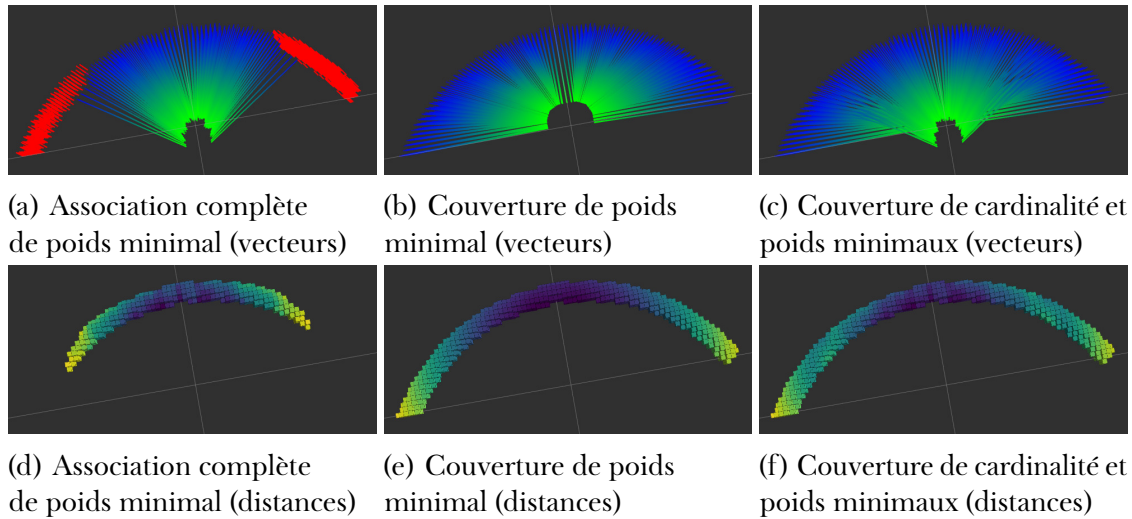


FIGURE 7.13 : Comparaison des méthodes d'association — tuyau et balle

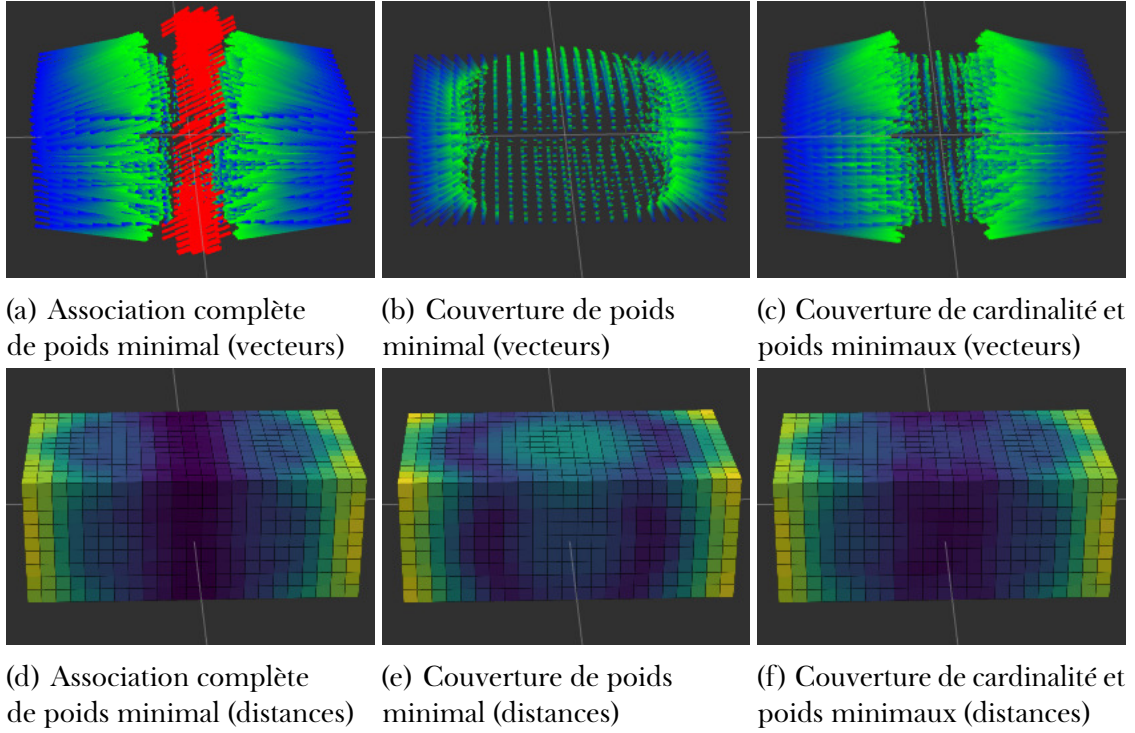


FIGURE 7.14 : Comparaison des méthodes d'association — éponge et sphère

Avec des formes encodées au format *Level-Sets*, où $\Lambda(v) < 0$ indique un voxel interne, $\Lambda(v) > 0$ indique un voxel externe, $\Gamma(v) = 1$ indique un voxel de bordure et $\Gamma(v) = 0$ indique un voxel à l'intérieur/extérieur de l'objet, il est possible d'utiliser ces informations supplémentaires pour l'association. En particulier, on peut limiter les associations aux voxels qui partagent le même état d'activité et le même signe, faisant en sorte que les voxels internes soient seulement associés aux voxels internes et que les voxels de bordure soient seulement associés aux voxels de bordure. La figure 7.15 présente les différentes combinaisons de ces critères pour l'association de l'éponge et de la sphère, avec la couverture de cardinalité et poids minimaux et un filtre de moyennage avec un rayon de 1 voxel. On remarque que l'inclusion de l'un ou l'autre des critères supplémentaires améliore l'uniformité des résultats par rapport à la distance seule, en particulier le signe. La combinaison des trois critères est l'option qui donne le résultat le plus visuellement plausible pour le guidage de la déformation.

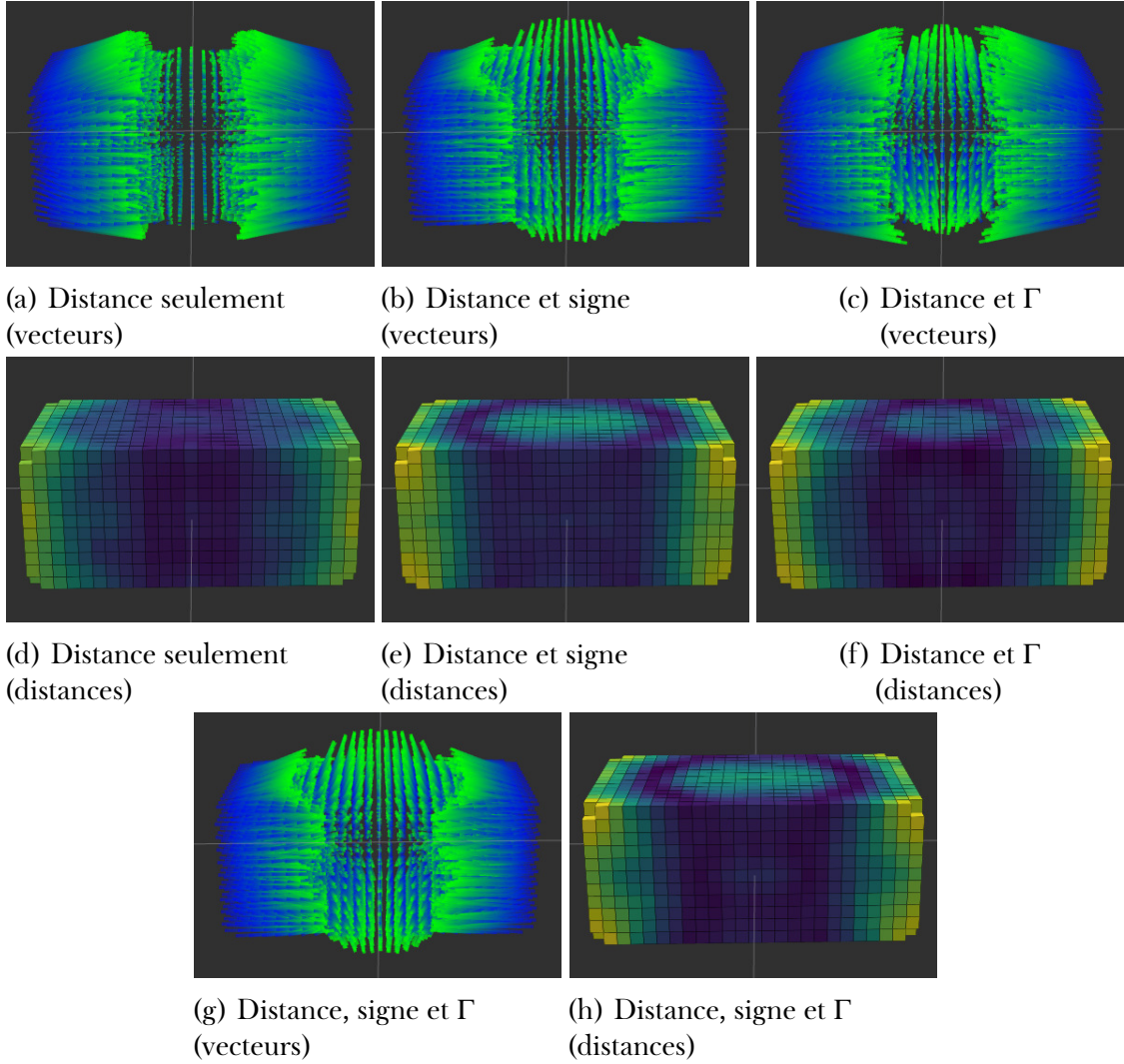


FIGURE 7.15 : Comparaison des critères pour l'association de voxels *Level-Sets* entre l'éponge et la sphère

7.5 Sélection des points de prise

La manipulation robotique d'un objet déformable en vue de son remodelage implique de distribuer les points de prise sur l'objet d'une manière qui permet un contrôle adéquat de sa forme pour rapprocher celle-ci de la cible. Sans connaissance préalable des propriétés du matériau de l'objet, il n'est pas possible d'utiliser une approche de simulation pour distribuer ces points et planifier leur déplacement. Ainsi, les méthodes de comparaison de formes 3D introduites dans ce chapitre sont ici étudiées en tant que méthodes d'analyse géométrique pour la sélection des points de prise. Dans cette optique, une méthode pour estimer le comportement d'un objet déformable sans connaissance préalable des propriétés de son matériau est le principe de diminution de la rigidité [197]. Celui-ci est basé sur l'observation que les parties de l'objet proches des points manipulés suivront le mouvement de ceux-ci avec un effet qui diminue proportionnellement à la distance du point manipulé. Cette idée peut être appliquée à la sélection des points de prise en positionnant ceux-ci près des parties de l'objet où le plus grand déplacement est souhaité [198].

Pour effectuer la sélection de prise, on suppose qu'une carte M_g est disponible où $\Gamma_{M_g}(v) = 1$ indique que le voxel v fait partie de l'objet à manipuler et où $\Lambda_{M_g}(v)$ est proportionnel à la qualité du voxel v en tant que point de prise. Cette mesure de qualité d'un point de prise dépend particulièrement du choix du préhenseur utilisé, ainsi que des spécificités de la tâche à effectuer. Pour le remodelage général d'objets déformables, le principe de diminution de la rigidité indique que M_a^d , la carte des distances de l'association des voxels, correspond à cette hypothèse (on veut placer les points de prise aux endroits qui doivent subir le plus grand déplacement). Selon le préhenseur et la tâche, il pourrait être souhaitable d'en faire une mesure pondérée incluant une mesure de la courbure locale de l'objet [8, 207] ou la distance de son centre de masse [79]. En effet, la nature du préhenseur de même que le nombre, la disposition et les mouvements que ses parties en contact avec l'objet peuvent effectuer affectent inévitablement les manipulations possibles, à la fois pour la manipulation en main de petits objets ou la manipulation collaborative de grands objets. Ces contraintes doivent naturellement être prises en compte dans la sélection des points de prise pour obtenir les résultats souhaités. Ceci s'intègre dans le cadre proposé de deux manières. Premièrement, la nature du préhenseur affecte les régions de l'objet sur lesquelles il est préférable de positionner les points de prise, par exemple si on préfère de grandes régions planes ou de petits rebords, si on préfère les régions plus ou moins rigides, ou plus ou moins proches du centre de masse de l'objet. Ces contraintes peuvent être directement incluses dans le calcul de M_g et donc

contrôler la distribution des points de prise proposés sur l'objet. Deuxièmement, la géométrie du préhenseur et les mouvements possibles affectent la sélection et l'agencement des points de prise retenus parmi ceux proposés. Par exemple, on peut vouloir limiter la sélection à des points situés de part et d'autre de l'objet pour une pince parallèle, ou former un triangle équilatéral avec une pince à trois doigts, avec différentes distances minimales entre les points de contact. Cette sélection intervient entre le calcul de M_g et le choix final des points de prise, c'est-à-dire qu'on identifiera des points où Λ_{M_g} est élevé et dont la disposition respecte les contraintes géométriques imposées par le robot.

Aux fins des expériences discutées dans cette thèse, on suppose que les points de prise peuvent se déplacer librement dans l'espace, et qu'ils n'ont pas de contraintes liées à la forme de l'objet, à leur position sur celui-ci ou entre eux, et que les préhenseurs agrippent l'objet de manière rigide et fixe aux points de prise. Ceci correspond par exemple à des prises ponctuelles avec plusieurs manipulateurs ou à des prises englobantes sur différentes sections de l'objet. Cette hypothèse permet de poser $M_g = M_a^d$, soit la carte d'association représentant les distances. Ainsi, le concept de base de la sélection de prise est que le module de planification de tâche interroge le module de cartographie pour obtenir les maximums de M_g , et distribue les prises sur ceux-ci en fonction des contraintes du préhenseur (par exemple les critères de stabilité de prise discutés dans [8]). Deux algorithmes sont proposés ci-dessous pour obtenir ces points, soit la sélection des maximums globaux et la sélection des maximums locaux.

7.5.1 Sélection des maximums globaux

Le premier algorithme de sélection de prise est présenté à l'algorithme 7.2. Cette méthode choisit simplement les n points sur la surface de l'objet qui ont la plus grande valeur dans M_g . Pour éviter que les points choisis soient trop près les uns des autres, on définit une distance minimale r entre eux. Cet algorithme commence par parcourir tous les voxels v de l'objet pour lesquels $\Gamma_{M_g}(v) = 1$ afin d'identifier les voxels de surface (dont au moins un des voisins ne fait pas partie de l'objet). Les voxels de surface sont placés dans H , un tas à priorité croissante⁴ qui les maintient triés par valeur $\Lambda_{M_g}(v)$. Les maximums sont ensuite déterminés en choisissant le voxel $v \in H$ pour lequel $\Lambda_{M_g}(v)$ est maximal, et en le supprimant de H ainsi que ses voisins dans un rayon r . Le résultat est L , une liste des maximums globaux de M_g séparés par une distance minimale de r , triés en ordre décroissant de $\Lambda_{M_g}(v)$. Ainsi, après l'exécution de cet algorithme, le module de planification de tâche sélectionne simplement les n premiers

4. *max heap*

éléments de L pour y placer les points de manipulation.

Algorithme 7.2 Identification des régions de surface à maximum global

Entrées

M_g , une grille de voxels à valeurs réelles

3: r , la distance minimale entre les prises (en voxels)

Sorties

L , une liste de paires $(v_i, \Lambda(v_i))$ où $\|v_i - v_j\| > r \forall i, j$; en ordre décroissant de $\Lambda_{M_g}(v_i)$

6: $H \leftarrow \emptyset$

Pour tout $v \in \mathcal{B}_{M_g} \mid \Gamma_{M_g}(v) = 1$, **faire**

Si $\neg(\Gamma_{M_g}(n) = 1 \forall n \in \eta_0(v))$ **alors** ▷ voxel de surface

9: $H \leftarrow_{\oplus} (v, \Lambda_{M_g}(v))$

Tant que $H \neq \emptyset$, **faire**

$(v, \Lambda) \leftarrow \max_{\Lambda}(H)$

12: $L \leftarrow_{\oplus} (v, \Lambda)$

$H \leftarrow_{\ominus} (v, \Lambda)$

$H \leftarrow_{\ominus} (n, \Lambda(n)) \forall n \in \eta_r(v)$

7.5.2 Sélection des maximums locaux

Le deuxième algorithme de sélection de prise consiste à choisir les points sur la surface de l'objet qui sont des maximums locaux dans M_g . Similairement à la sélection des maximums globaux, on définit une distance minimale r entre les points sélectionnés, qui peut être vue comme la distance à laquelle le manipulateur fait effet ou encore la taille de la pince ou du préhenseur utilisé. L'algorithme de sélection est le même (7.2) si ce n'est que la condition de la ligne 8 est modifiée pour correspondre à l'équation (7.7), où d est la distance à considérer pour le maximum (habituellement 0), c'est-à-dire que v doit être un voxel de surface et un maximum local dans un rayon de d pour être considéré comme un point de prise potentiel.

$$\neg(\Gamma_{M_g}(n) = 1 \forall n \in \eta_0(v)) \wedge (\Lambda_{M_g}(n) < \Lambda_{M_g}(v) \forall n \in \eta_d(v)) \quad (7.7)$$

Cet algorithme a deux intérêts particuliers. D'abord, les maximums locaux sont souvent distribués plus uniformément que les maximums globaux sur la surface de l'objet. Ensuite, selon le principe de diminution de la rigidité et si M_g est cohérent avec M_a^d , le nombre de

maximums locaux identifiés par cet algorithme permet de déterminer combien de points de prise sont nécessaires pour effectuer la déformation. Cet estimé du nombre de points de prise est donc obtenu directement à partir de la comparaison géométrique, sans nécessiter d’optimisation prédictive coûteuse.

7.6 Évaluation de la sélection de prise

La sélection de prise est basée sur les résultats de l’association des voxels, en choisissant les points où la distance à parcourir pour faire correspondre les deux formes est la plus grande. Selon la section 7.5, deux méthodes sont évaluées, soit la sélection des maximums globaux et la sélection des maximums locaux. Les figures 7.16 à 7.27 montrent les résultats de cette sélection, avec ou sans alignement préalable des formes, pour une série de paires de formes. Notons que l’alignement est effectué à la main pour isoler la comparaison de formes de celui-ci. Dans ces figures, la forme originale montre la distance associée à chaque voxel selon l’échelle de couleur de la figure 7.8, la forme cible est indiquée en blanc, les vecteurs d’association sont représentés par des lignes passant du bleu (forme initiale) au vert (forme cible), et les régions de prise identifiées sont indiquées par des sphères dont la teinte correspond à la qualité relative du point de prise : une région identifiée par une sphère plus foncée correspond à une valeur de $M_g = M_a^d$ plus élevée (donc une plus grande distance à parcourir selon la section 7.5) et sera choisie en premier si le nombre de manipulateurs est limité. Dans ces résultats, la taille des voxels et la taille des régions de prise sont sélectionnées en fonction de la taille de l’objet de manière à faciliter la visualisation.

À travers ces résultats, on remarque que l’alignement préalable des formes est une étape essentielle pour l’obtention de points de prise pertinents. En particulier avec la sélection des maximums globaux, les points de prise privilégiés se retrouvent concentrés du même côté de l’objet lorsque les formes ne sont pas alignées. Ceci ne permettrait pas une déformation correcte de l’objet si le nombre de points de prise est limité, ce qui est particulièrement évident dans les figures 7.16 et 7.22. La sélection des maximums locaux souffre du même problème puisque, sans alignement, la distance entre les formes a une plus grande influence sur les résultats que leur différence. De manière générale, les points de prise les plus optimaux (plus foncés) obtenus par la sélection des maximums globaux sont semblables à ceux obtenus par la sélection des maximums locaux. Cependant, la sélection des maximums globaux propose un grand nombre de points de prise, qui doivent être limités manuellement, alors que la sélection des maximums locaux propose directement les points les plus intéressants.

De plus, on remarque dans la figure 7.27 que les points de prise ont tendance à être positionnés près des coins et bords des objets puisque ce sont les endroits qui demandent le plus grand déplacement. Cependant, ces points ne représentent pas des choix optimaux pour plusieurs types de préhenseurs puisqu'ils peuvent plus facilement glisser. Dans ces cas, l'ajout d'une mesure de la courbure dans la sélection de prise serait pertinent [8]. En plus de cette considération, d'autres contraintes doivent être définies en fonction de la géométrie et du fonctionnement du préhenseur utilisé. En effet, le positionnement direct des points de prise aux points suggérés correspond à des prises fixes indépendantes. Avec une pince parallèle, on chercherait plutôt à effectuer une prise englobante sur une région à haute déformation (par exemple en augmentant la distance r entre les prises dans l'algorithme 7.2), ou à sélectionner deux points de prise positionnés de part et d'autre de cette région. Un principe semblable s'applique au choix d'une prise triangulaire avec un préhenseur à trois doigts.

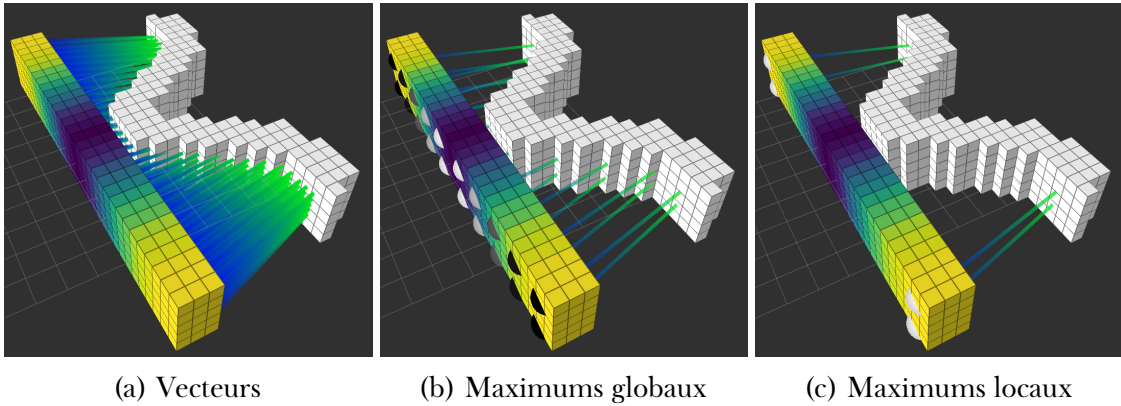


FIGURE 7.16 : Sélection de prise pour courber la poutre (sans alignement)

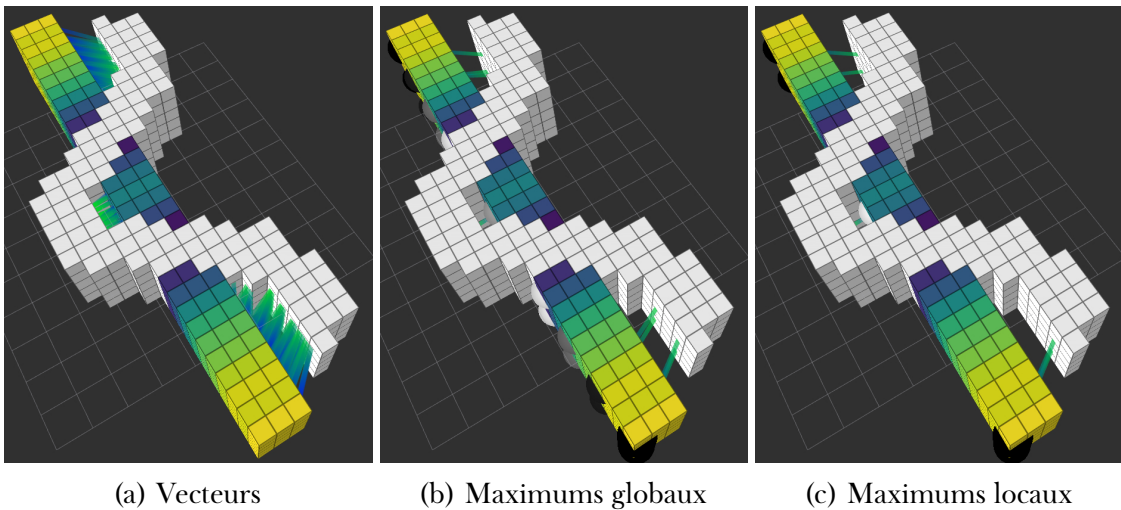


FIGURE 7.17 : Sélection de prise pour courber la poutre (avec alignement)

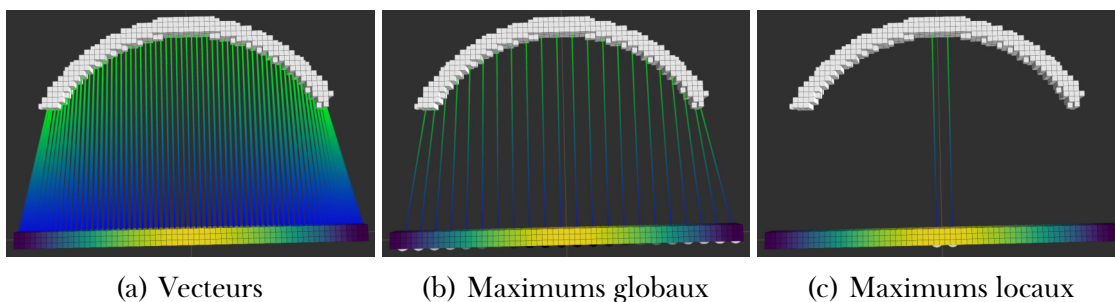


FIGURE 7.18 : Sélection de prise pour courber le tuyau (sans alignement)

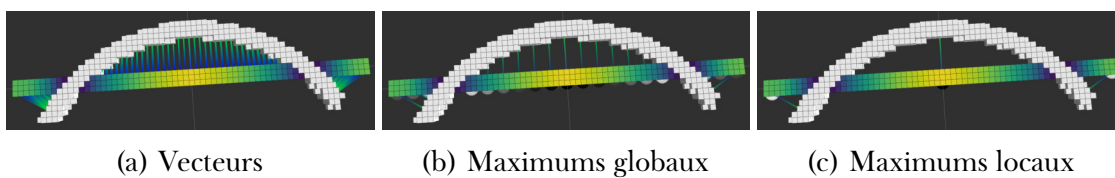


FIGURE 7.19 : Sélection de prise pour courber le tuyau (avec alignement)

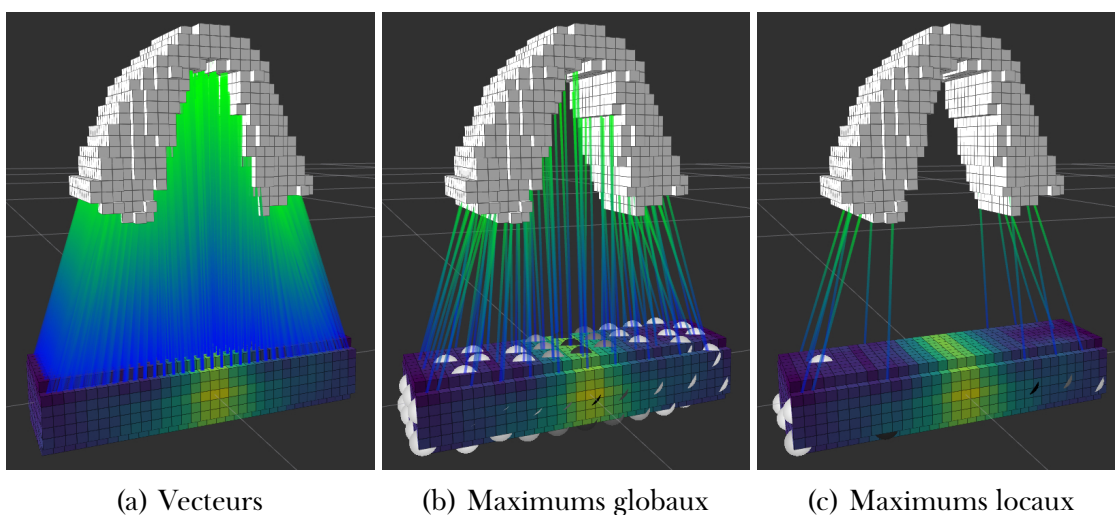


FIGURE 7.20 : Sélection de prise pour plier la longue éponge (sans alignement)

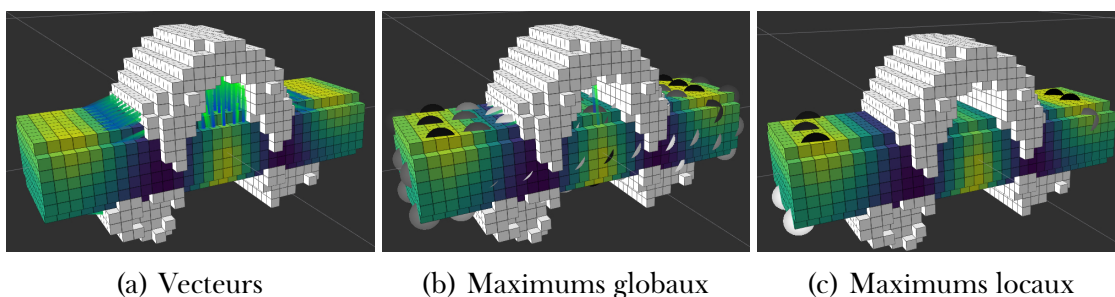


FIGURE 7.21 : Sélection de prise pour plier la longue éponge (avec alignement)

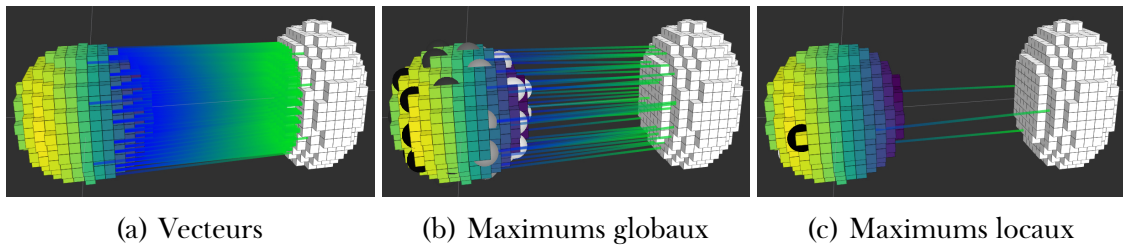


FIGURE 7.22 : Sélection de prise pour déformer la sphère en ovale (sans alignement)

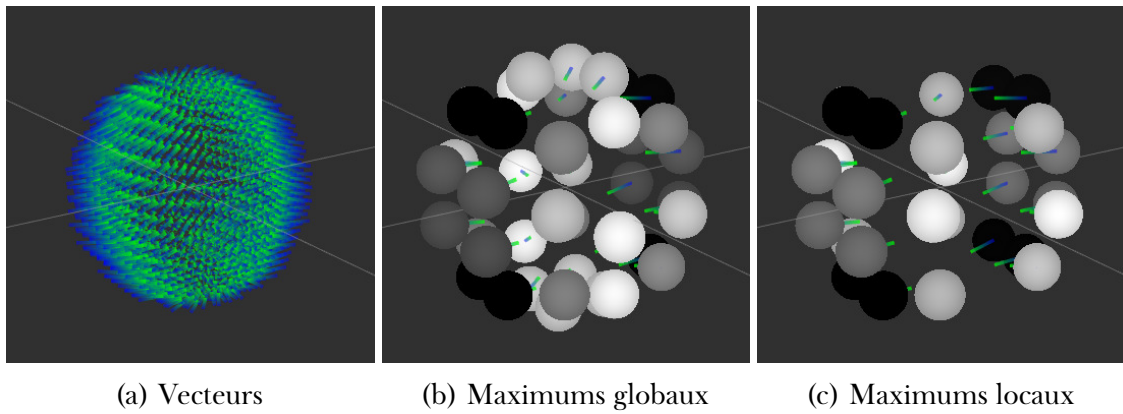


FIGURE 7.23 : Sélection de prise pour déformer la sphère en ovale (avec alignement)

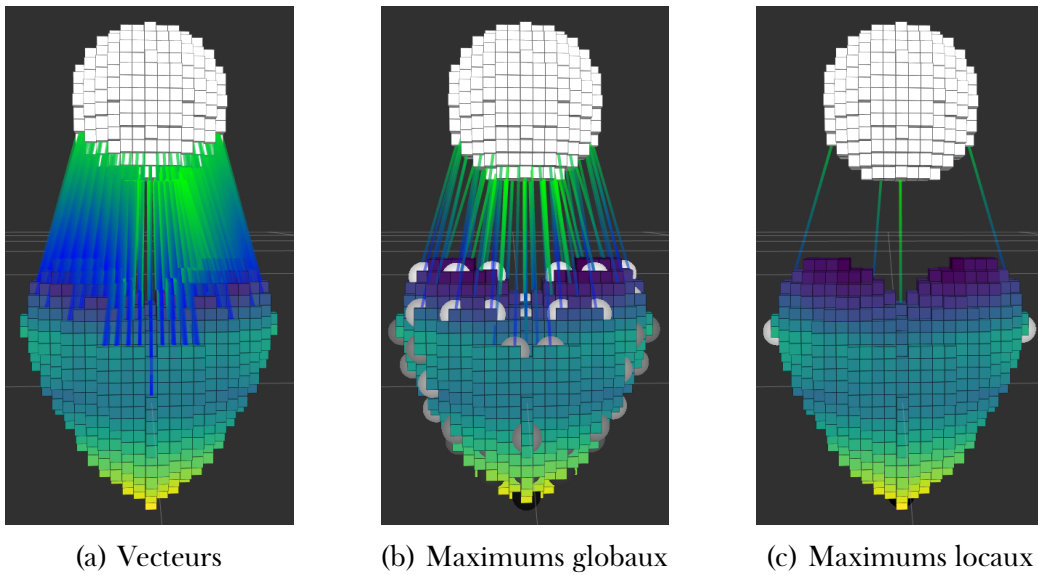
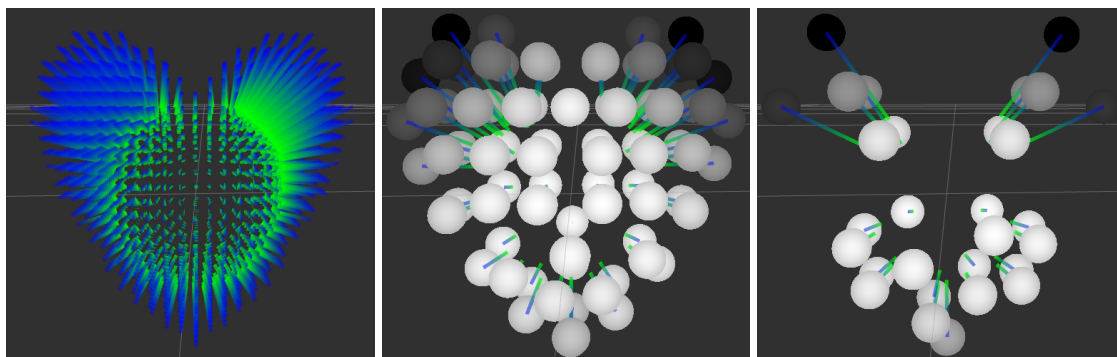


FIGURE 7.24 : Sélection de prise pour transformer le cœur en sphère (sans alignement)

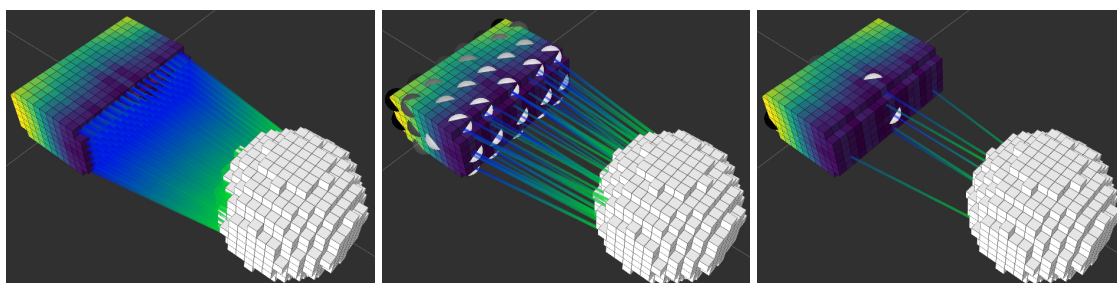


(a) Vecteurs

(b) Maximums globaux

(c) Maximums locaux

FIGURE 7.25 : Sélection de prise pour transformer le cœur en sphère (avec alignement)

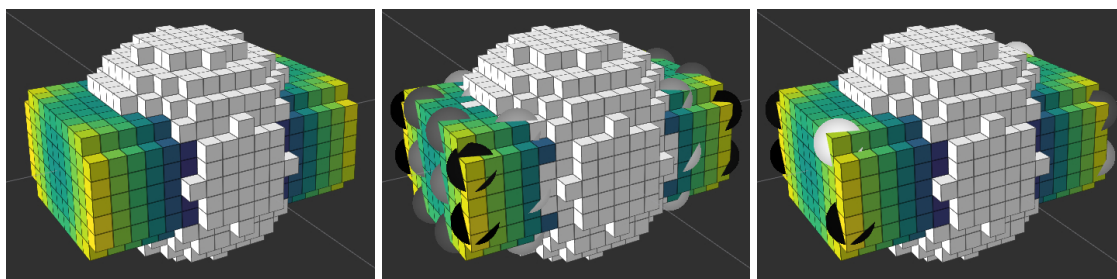


(a) Vecteurs

(b) Maximums globaux

(c) Maximums locaux

FIGURE 7.26 : Sélection de prise pour transformer l'éponge en sphère (sans alignement)



(a) Vecteurs

(b) Maximums globaux

(c) Maximums locaux

FIGURE 7.27 : Sélection de prise pour transformer l'éponge en sphère (avec alignement)

7.7 Sommaire

Ce chapitre propose des mesures et guides pour diriger le remodelage d’objets déformables dont les propriétés du matériau sont initialement inconnues dans un environnement 3D représenté par des grilles de voxels. En particulier, on conçoit une mesure de comparaison originale adaptée à la réalité du remodelage des objets déformables, permettant une association optimale des voxels entre la forme initiale et la forme cible tout en préservant les informations de volume et de forme. Ces propriétés permettent de définir des mesures additionnelles pour identifier le nombre, la position et le mouvement souhaité des points de prise requis pour le remodelage de la forme originale vers la forme cible avec des manipulateurs robotiques, indépendamment du préhenseur utilisé.

La mesure de comparaison originale proposée est comparée avec des méthodes inspirées de la littérature. Le score IoU est une mesure standard pour évaluer la performance de la segmentation 2D en comparant les formes résultantes, alors que la différence d_{diff} utilisée pour l’alignement à l’équation (7.3) est équivalente au « ou exclusif » utilisé dans [126] lorsqu’elle est appliquée à des voxels binaires. Les résultats démontrent que ces deux mesures sont peu informatives lorsque l’intersection entre les formes comparées est faible, et (7.3) est plus flexible que le score IoU ou que la mesure de [126] puisqu’elle permet de comparer des grilles de voxels encodant des valeurs autres que binaires. La mesure présentée dans [130] correspond à la mesure d’association proposée avec les contraintes que les deux formes aient le même nombre de voxels, qu’on considère seulement les voxels qui ne font pas partie de l’intersection des deux formes, et qu’on utilise l’association complète de poids minimal. Cependant, la normalisation en volume n’est pas compatible avec le remodelage d’objets déformables réels, et l’association complète de poids minimal échoue lorsque les formes n’ont pas le même nombre de voxels (section 7.4.3). La méthode proposée pour l’association des voxels (section 7.2) généralise cette mesure en éliminant ces contraintes. Elle permet ainsi l’extraction de mesures pour guider le positionnement et le mouvement des points de prise pour le remodelage, une caractéristique qui n’a pas de comparables directs dans la littérature existante. La méthode proposée pour la sélection des points de prise, et particulièrement l’utilisation des maximums locaux, permet d’identifier le nombre, la position et le mouvement des points de prise sans nécessiter de simulation ni de connaissances sur le matériau de l’objet, qui est un avantage par rapport aux exemples de [80, 82].

Globalement, bien que la méthode proposée soit relativement coûteuse à calculer pour les grands objets, elle permet de comparer des formes avec des volumes différents et demeure

informative lorsqu'il n'y a aucune ou peu d'intersection entre les formes, ou lorsque l'une d'entre elles est complètement incluse dans l'autre. De plus, les informations détaillées fournies par cette méthode permettent de déterminer des points de contrôle et des trajectoires plausibles pour le remodelage de l'objet avec des manipulateurs robotiques, qui peuvent être sélectionnés et raffinés en fonction des contraintes de la tâche et du préhenseur utilisé.

Chapitre 8

Résultats globaux et analyse

Ce chapitre présente la validation globale des algorithmes et méthodes proposés aux chapitres précédents en les appliquant à des scénarios de bout en bout de remodelage d'objets déformables (section 8.1). La validation expérimentale se conclut à la section 8.2 par une analyse globale des résultats présentés ici ainsi qu'aux chapitres précédents.

8.1 Résultats globaux

Cette section présente les résultats pour le processus complet d'observation, comparaison et déformation pour une série de scénarios en simulation, examinant autant la manipulation en main de petits objets et la manipulation collaborative de grands objets. Dans ces résultats, lorsque plusieurs scénarios utilisent la même forme de départ, on ne répète pas le processus d'observation puisque celui-ci produit des résultats presque identiques à chaque fois. Pour tester le remodelage en simulation, les prises sur l'objet sont représentées par des « sphères de contrôle » de taille variable, qui agrippent l'objet de manière rigide lorsqu'elles font une intersection avec lui de sorte que les points agrippés ne peuvent pas glisser hors des prises. Ceci permet de déplacer précisément les points désirés sur l'objet et limite le glissement et les pertes de contrôle dues à l'instabilité de la simulation. En modifiant la taille de la sphère de contrôle, on peut modéliser une prise fixe sur un point, ou encore une prise englobante sur une section de l'objet. Ainsi, les points choisis pour la manipulation sur l'objet sont déplacés linéairement et précisément vers leur point associé sur la forme cible. Puisque la simulation permet une grande précision sur le positionnement des sphères de contrôle, leur mouvement est fait en boucle ouverte. Étant donné que les différentes cartes utilisées sont configurées pour être mises à jour automatiquement, le suivi de forme en cours de déformation se fait naturellement lorsque les nouveaux nuages de points sont transmis par la caméra virtuelle.

Sauf indication contraire, on utilise des voxels de 8 mm, l'algorithme de segmentation hybride (section 5.3) avec un seuil adapté à l'objet d'intérêt (généralement autour de 0,3), et on divise les voxels inconnus en groupes (section 6.1), dont on observe un nombre égal au numéro d'itération avant de recalculer les heuristiques d'observation (section 6.3.1.1). L'association est faite en utilisant la méthode de la couverture d'arêtes de cardinalité et poids minimums (section 7.2), et la sélection de prise utilise les maximums locaux (section 7.5). Notons que les points de prise sélectionnés sont montrés de la même manière qu'à la section 7.6. Pour faciliter la visualisation, les marqueurs indiquant les points de prise ont 50% de la taille de la zone d'influence de la prise et les sphères de contrôle utilisées dans la simulation ont 25% de celle-ci. Par exemple, si la distance r dans l'algorithme 7.2 est définie à trois voxels (24 mm), les sphères grises montrant les points de prise potentiels ont un rayon de 12 mm et les sphères de contrôle rose ont un rayon de 6 mm. Notons également que le remplissage de la forme segmentée affecte aux voxels remplis une valeur inférieure à celle des voxels provenant de la segmentation. Ainsi, lorsqu'un masque d'objet segmenté est affiché, on peut distinguer les voxels provenant de la segmentation (valeur maximale, en jaune) de ceux provenant du remplissage (valeur minimale, en mauve). Les résultats visuels sont regroupés par scénario dans les sections 8.1.1 à 8.1.4, alors que la section 8.1.5 présente les valeurs numériques pour la comparaison de formes avant et après la déformation. Quelques cas supplémentaires sont présentés à l'annexe B.3. Dans les figures montrant des cartes voxélisées, les axes rouge, vert et bleu représentent respectivement les axes X , Y et Z du référentiel global.

8.1.1 Transformation de l'éponge à la sphère

Le premier scénario testé est de comprimer l'éponge rectangulaire ($7 \times 4 \times 13$ cm) pour en faire une sphère de 10 cm de diamètre. La figure 8.1 présente la sélection du point identifiant l'objet d'intérêt dans la carte colorée (M_{cd} , figure 3.6) après les trois vues fixes initiales ainsi que le masque d'objet segmenté et rempli (M_m , figure 3.7). Comme noté à la section 6.3, avec des voxels de 8 mm, l'éponge est considérée complète dès les vues fixes initiales. Il n'y a donc pas de séquence d'observation pour compléter la forme originale dans ce cas-ci.

La figure 8.2 présente le résultat de la comparaison de la forme initiale et de la forme cible (section 7.2), ainsi que les points de prise sélectionnés et l'angle d'observation pour le suivi de la manipulation (section 7.5). Dans ce cas-ci, on utilise un rayon d'influence de 8 cm (10 voxels) pour les points de prise, de sorte que seuls deux points de prise sont identifiés par l'algorithme de sélection.

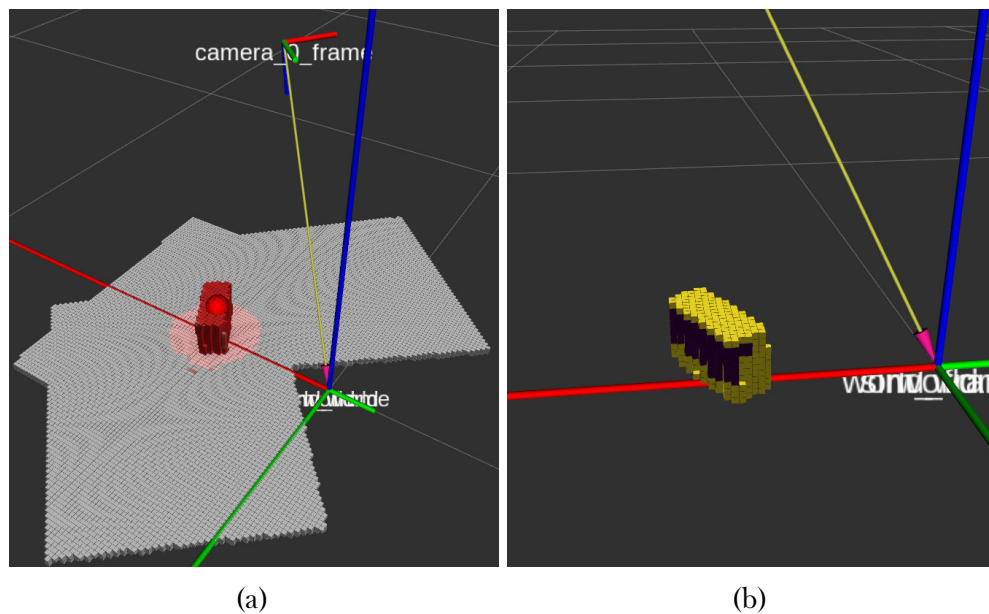


FIGURE 8.1 : (a) Sélection de l'objet d'intérêt dans la scène initiale et (b) forme segmentée de l'éponge. Dans (b), les voxels jaunes sont ceux obtenus de la segmentation alors que les mauves proviennent de la procédure de remplissage.

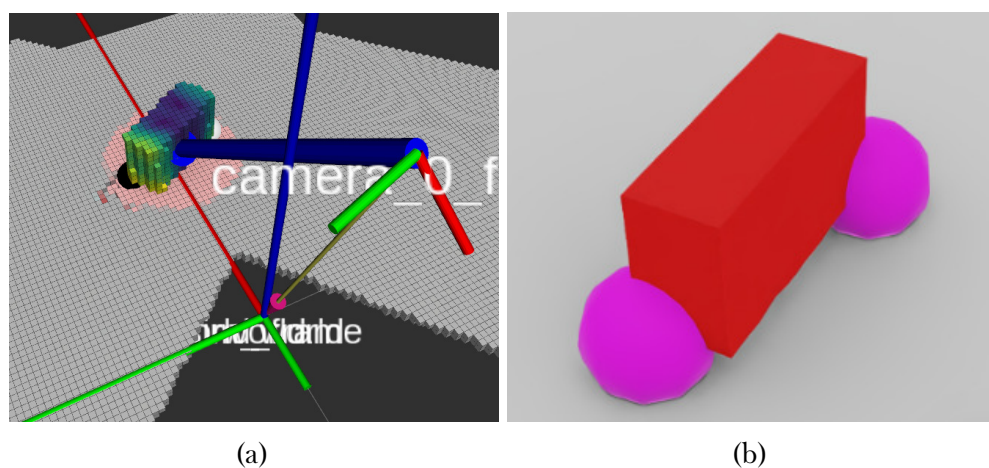


FIGURE 8.2 : (a) Résultat de comparaison, points de prise et vecteur d'observation (flèche bleue) pour le suivi de la manipulation avec des prises distantes d'au moins 8 cm. (b) Points de prise en simulation pour transformer l'éponge rectangulaire en sphère.

La figure 8.3 présente le résultat de la manipulation en simulation ainsi que le masque d'objet d'intérêt avant et après la réobservation de l'objet déformé. Visuellement, le mouvement des points de prise permet de rapprocher l'éponge d'une forme sphérique malgré une certaine instabilité de la simulation en 8.3a. La forme segmentée obtenue à partir de seulement l'angle d'observation pour le suivi (8.3b) ressemble au résultat de la simulation bien qu'elle représente seulement une vue partielle de la forme de l'objet. Cette forme est semblable à celle, plus complète, obtenue après la réobservation avec plusieurs points de vue (8.3c), confirmant le choix du point d'observation pour le suivi de la manipulation et la capacité du système de cartographie à effectuer le suivi de l'objet d'intérêt. On note aussi que des artefacts liés à l'instabilité de la simulation (notamment en bas à gauche de l'objet à la figure 8.3a) sont transposés dans la forme obtenue après la réobservation à la figure 8.3c.

La figure 8.4 compare la forme cible avec les formes initiale et finale. Visuellement, la forme finale est plus proche de la forme cible que la forme initiale, malgré un léger décalage vertical.

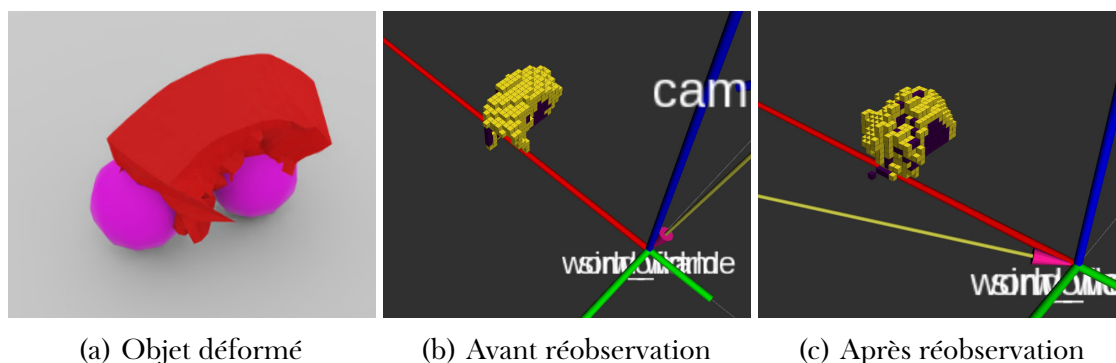


FIGURE 8.3 : Résultat de la manipulation

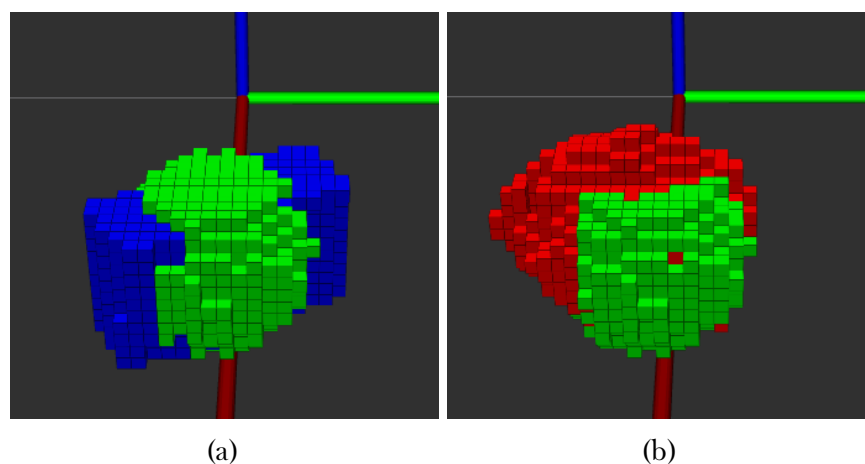


FIGURE 8.4 : Comparaison des formes initiale (bleu) et finale (rouge) avec la forme cible (vert)

Puisque les points de prise utilisés précédemment ont une zone d'influence relativement grande par rapport à l'objet, ce qui empêche de les rapprocher suffisamment, on répète l'expérience en les réduisant à une zone de 5 cm. La figure 8.5 montre les points de prise sélectionnés dans ce cas ainsi que les résultats de la manipulation, en simulation et comparativement à la forme cible. Comparativement au cas avec les prises espacées de 8 cm, on remarque un contrôle plus précis de la forme de l'objet malgré une certaine instabilité de la simulation. Le résultat est ainsi plus proche de la forme cible que précédemment.

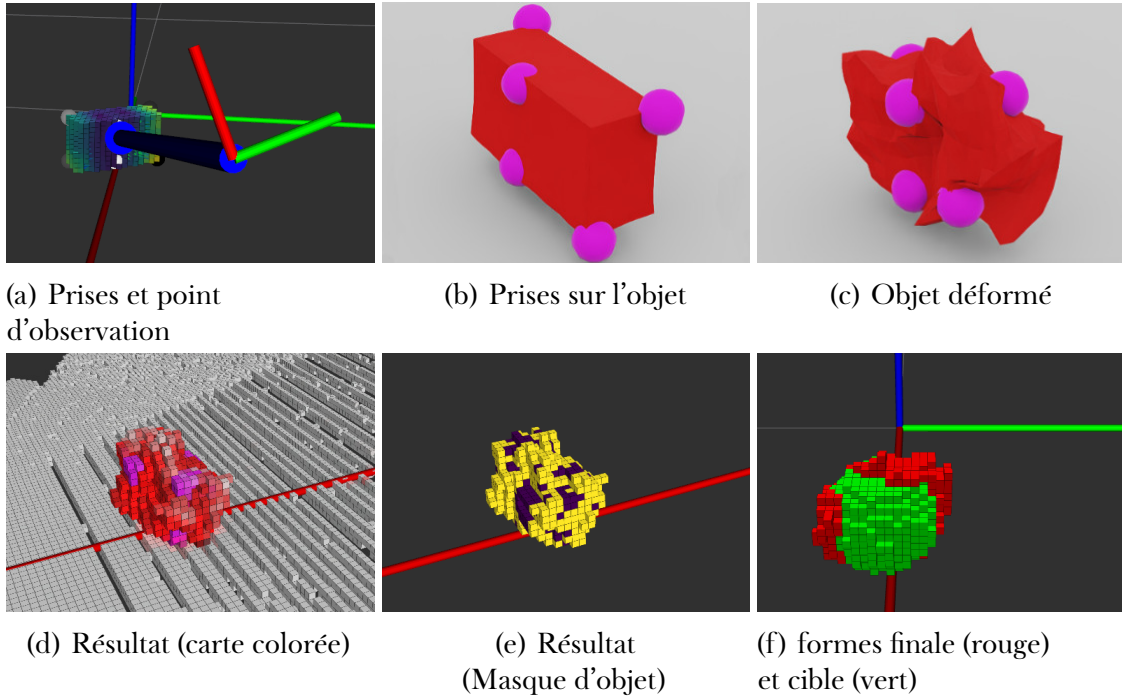


FIGURE 8.5 : Résultat de la manipulation avec des prises distantes d'au moins 5 cm

Enfin, on répète le scénario en limitant les points de prise aux deux points présentant le plus grand déplacement, sans autre considération. Comme on peut le constater à la figure 8.6, ceci résulte en des points de prise qui sont situés du même côté de l'objet. Ainsi, le résultat de la manipulation est principalement un déplacement plutôt qu'une déformation. Ce déplacement fait en sorte que le point de référence pour l'objet d'intérêt n'est plus sur l'objet mais sur une des sphères de contrôle, menant à un échec du suivi de la déformation. Lorsque le nombre de points de prise est limité, il est donc nécessaire de tenir compte de contraintes plus avancées comme le bouclage des forces¹ plutôt que de naïvement sélectionner les n meilleurs points. Une approche alternative serait aussi d'augmenter graduellement la taille de la zone d'influence des points de prise jusqu'à ce que le nombre de maximums locaux choisis corresponde au

1. *force closure*

nombre de préhenseurs disponibles. L'utilité de cette méthode serait cependant dépendante de la taille et de la géométrie de l'objet à remodeler.

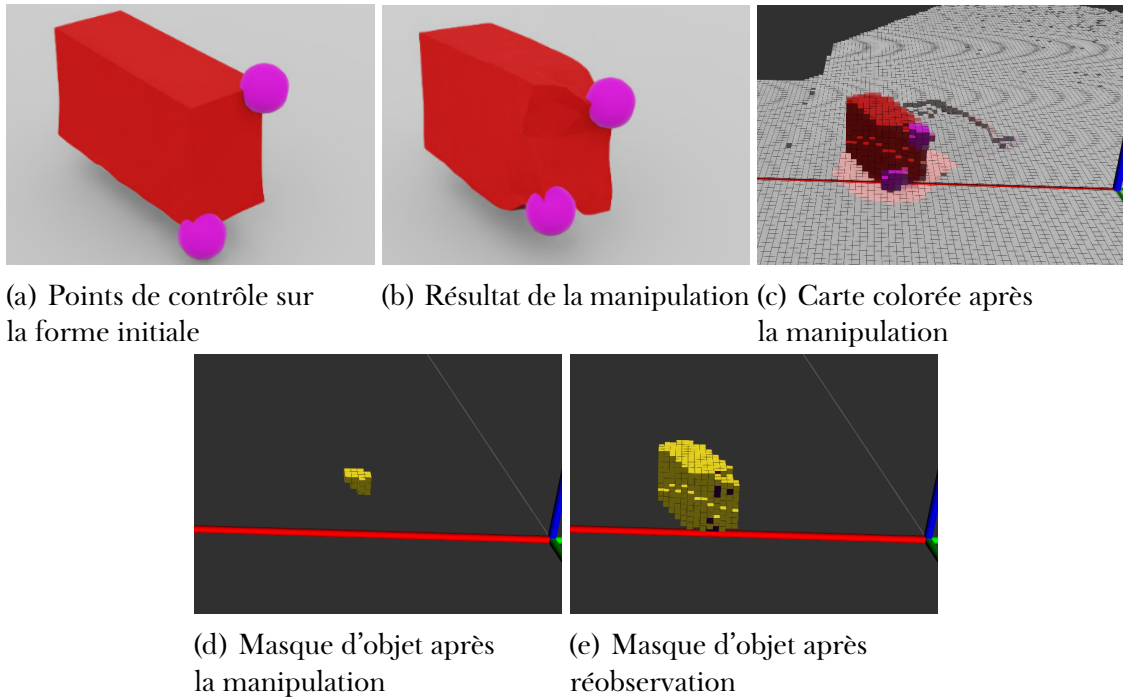


FIGURE 8.6 : Résultat de la manipulation avec les points de prise du même côté de l'objet. Le point de référence doit être modifié avant la réobservation pour correspondre à l'objet d'intérêt en (e)

8.1.2 Transformation de la sphère en cœur

Une forme de cœur peut se montrer intéressante pour étudier les déformations car ses concavités et protrusions permettent un positionnement intuitif des points de contrôle optimaux [8]. Dans ce cas, on cherche à comprimer la sphère de 10 cm de diamètre pour former un cœur d'environ 9 cm de haut, 9 cm de large et 5 cm d'épaisseur. En raison du niveau de détail de cette forme, on augmente la résolution pour avoir des voxels de 4 mm. La figure 8.7 montre la situation initiale ainsi que la forme cible alignée. La figure 8.8 montre les prises sélectionnées et le résultat avec des prises espacées de 8 cm, et la figure 8.9 avec des prises de 4 cm. Comme on peut le constater, la taille et le nombre des points de contrôle est un facteur clé dans l'obtention de résultats visuellement plausibles. En effet, avec les prises de 8 cm, les deux points de prise sont parfaitement alignés pour former la base du cœur (8.8 c et d), mais leur espacement ne permet pas d'ajouter la 3e prise qui serait nécessaire pour former le creux. Avec les prises de 4 cm (8.9), on obtient un plus grand nombre de points de prise qui

tendent d'aplatir la sphère en plus des trois points attendus pour former le cœur. Bien que ceci permette de rapprocher la sphère de la forme cible, la distribution de ces points de prise et les propriétés de matériau de la sphère font en sorte qu'il est plus difficile d'évaluer le résultat visuellement. Lorsqu'on optimise manuellement la taille des prises à 7 cm, on obtient les trois points de prise attendus pour former la base du cœur et son creux, tel que montré à la figure 8.10. Dans cette figure, on constate que le creux du cœur demeure difficile à former en raison des propriétés du matériau, qui s'aplatit plutôt que de former le creux attendu. En pratique, le choix de la taille des points de prise dépend grandement du préhenseur utilisé, allant de prises quasi ponctuelles aux doigts plus ou moins larges des différentes pinces jusqu'à une prise englobante. Lorsque ce choix est plus flexible, on remarque que la qualité visuelle du résultat dépend en partie des propriétés du matériau de l'objet. Une planification prédictive tenant compte de ces propriétés [12] serait donc requise pour optimiser le choix de la taille des points de prise.

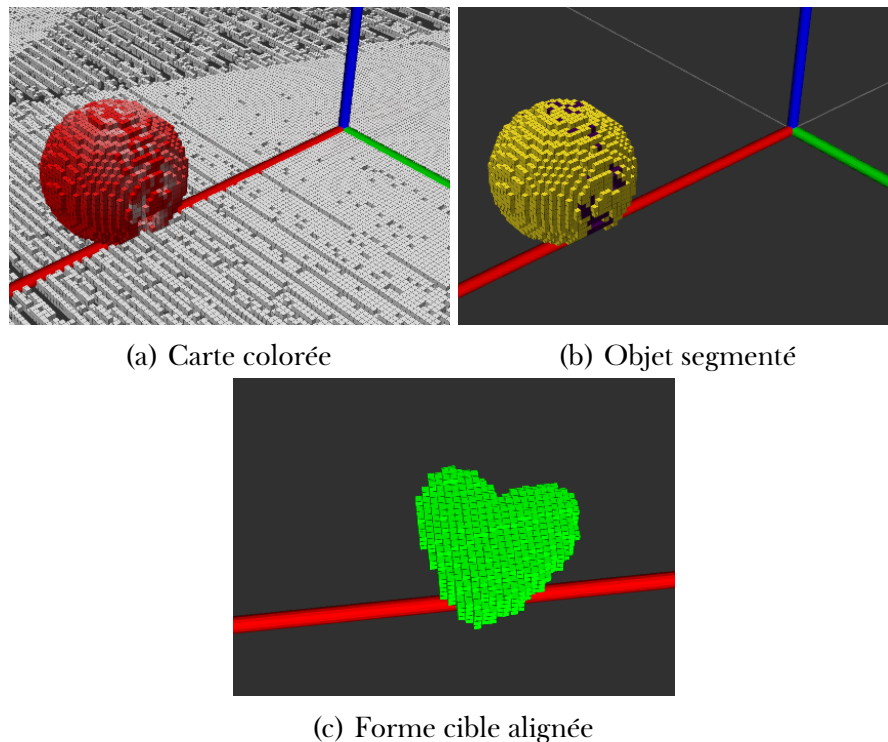


FIGURE 8.7 : État initial pour la sphère et le cœur. La forme cible étant complètement occluse par la forme initiale, il n'est pas possible de montrer les deux sur la même image.

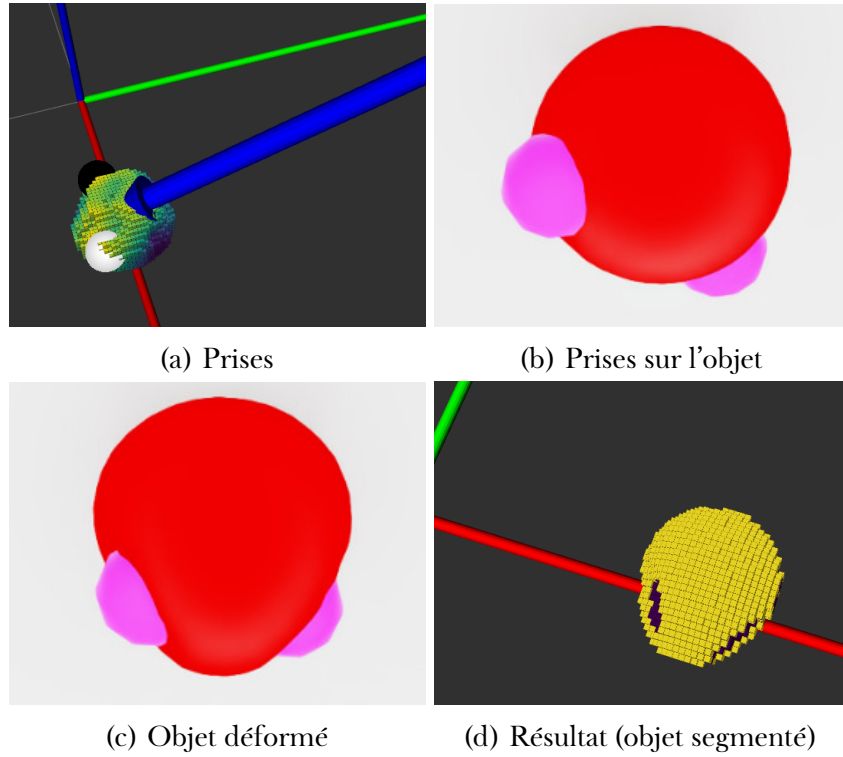


FIGURE 8.8 : Prises et résultat avec des prises distantes d'au moins 8 cm

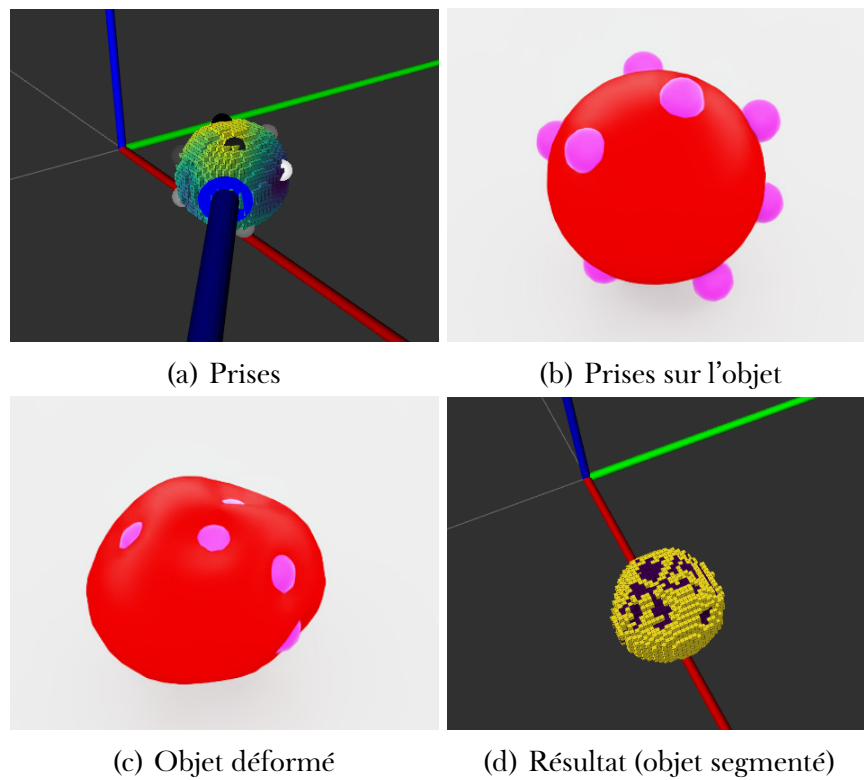


FIGURE 8.9 : Prises et résultat avec des prises distantes d'au moins 4 cm

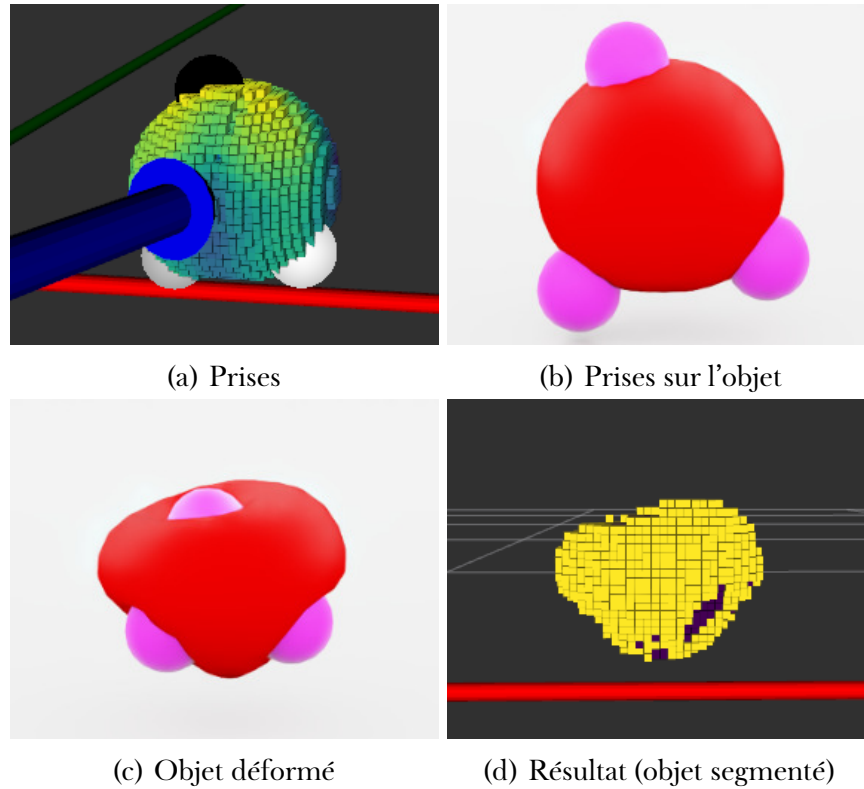


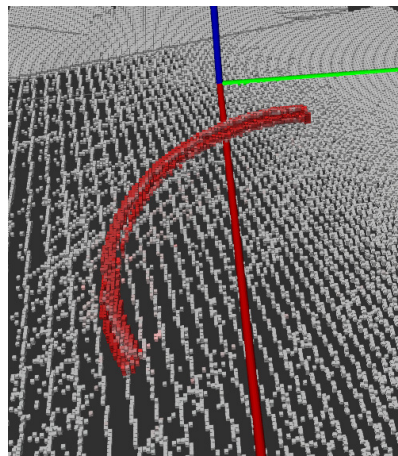
FIGURE 8.10 : Prises et résultat avec des prises distantes d'au moins 7 cm

8.1.3 Redresser un tuyau

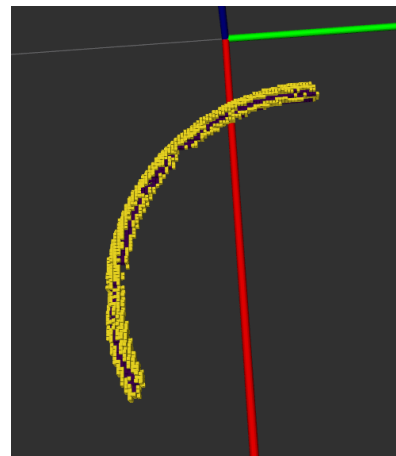
Le prochain scénario consiste à redresser un tuyau courbé d'environ 2 cm de diamètre et 50 cm de longueur. En raison de son petit diamètre, on utilise ici des voxels de 4 mm plutôt que 8. La figure 8.11 montre la situation initiale avec la carte colorée, l'objet segmenté et la forme cible. La figure 8.12 montre les prises sélectionnées et les cartes en cours de manipulation avec le suivi en temps réel. Enfin, la figure 8.13 montre les résultats finaux. Puisque la déformation est grande, on doit sélectionner un nouveau point de référence sur l'objet à la fin de la manipulation pour pouvoir identifier correctement la forme résultante.

Bien que le résultat soit visuellement très proche de la forme cible, on remarque que le nombre de prises proposées est significativement plus grand que le minimum attendu de deux ou trois. Ceci indique que la carte des distances d'association utilisée pour la sélection de prise et visible en 8.12a comporte des irrégularités qui créent des maximums locaux superflus. Ceci pourrait justifier l'utilisation d'un filtre de moyennage plus agressif pour adoucir la carte, ou encore un plus grand rayon d pour l'identification des maximums locaux à l'équation (7.7). En effet, lorsqu'on augmente le rayon d de vérification des maximums locaux de 0 à 3 dans (7.7),

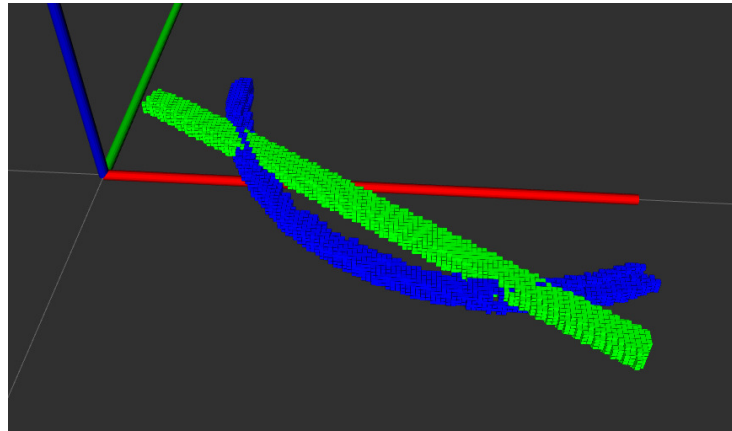
on obtient les trois maximums locaux attendus, comme montré à la figure 8.14. Bien que la forme de l'objet soit légèrement plus difficile à contrôler dans ce cas, les résultats obtenus sont visuellement proches de ceux montrés à la figure 8.13 avec un plus grand nombre de points de prise répartis sur l'objet, justifiant ainsi l'utilisation des maximums locaux (avec un rayon approprié) pour identifier le nombre de points de prise nécessaires. Plus généralement, ces résultats confirment aussi la capacité du système à faire le suivi de la déformation et la mise à jour en continu des cartes. L'objet manipulé est suivi et segmenté tout au long de la manipulation (figure 8.12 c et d) et est très proche du résultat obtenu après une réobservation complète (figure 8.13 c et d) jusqu'à ce que le point de référence de la segmentation se retrouve hors de l'objet (8.13b). La forme finale de l'objet est presque identique à la forme cible (8.13e), confirmant la pertinence d'utiliser les vecteurs d'association résultant de la comparaison pour guider le mouvement des points de prise.



(a) Carte colorée



(b) Objet segmenté



(c) Objet (bleu) et forme cible (vert)

FIGURE 8.11 : Situation initiale pour le tuyau

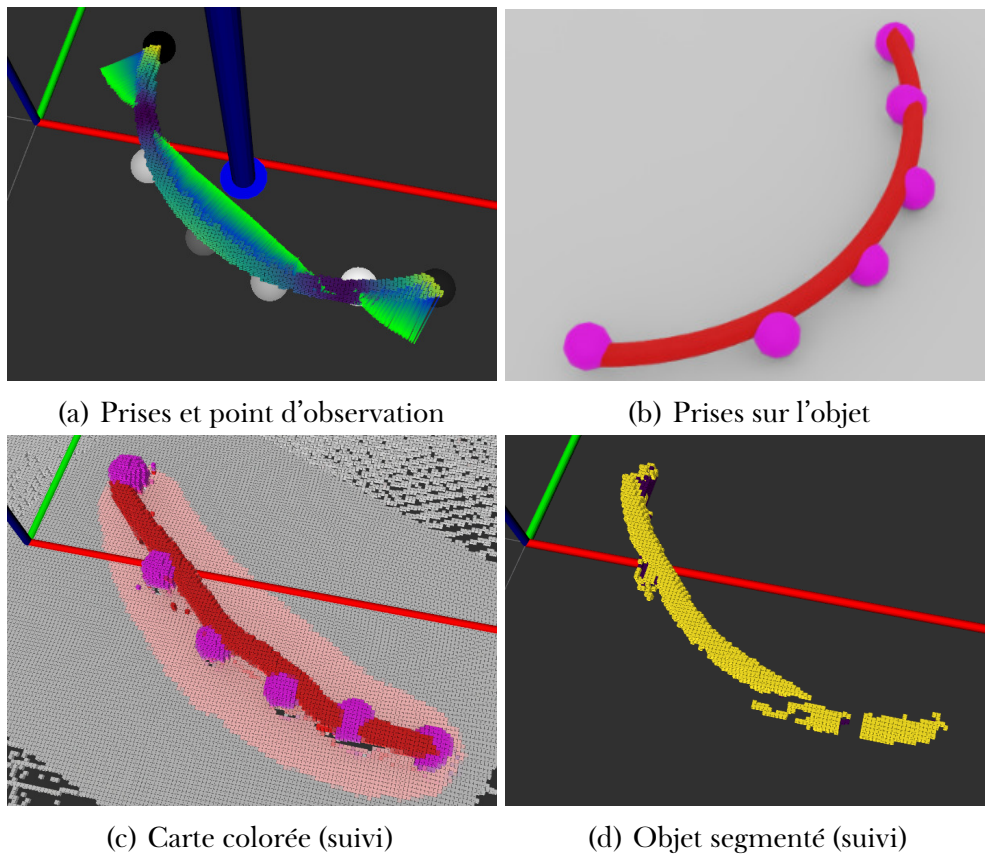
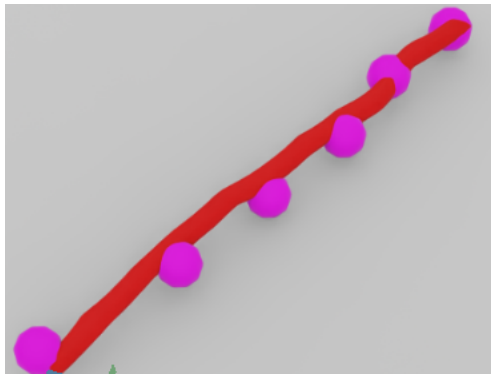
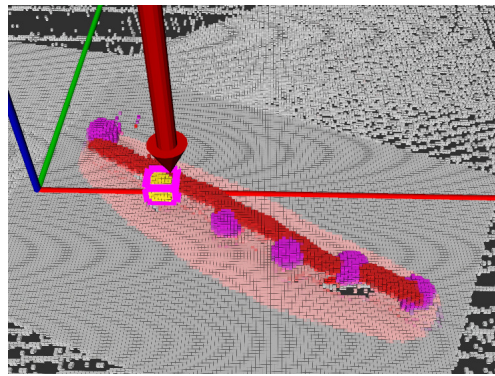


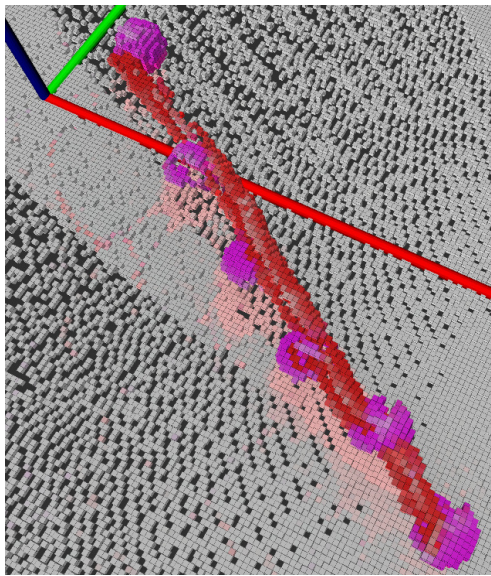
FIGURE 8.12 : Prises et suivi pour le tuyau



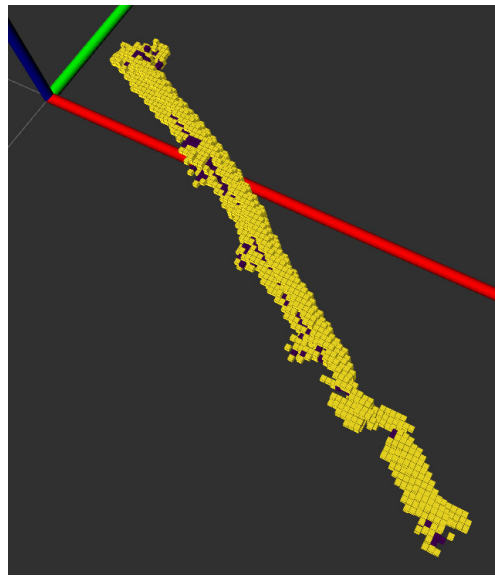
(a) Objet déformé



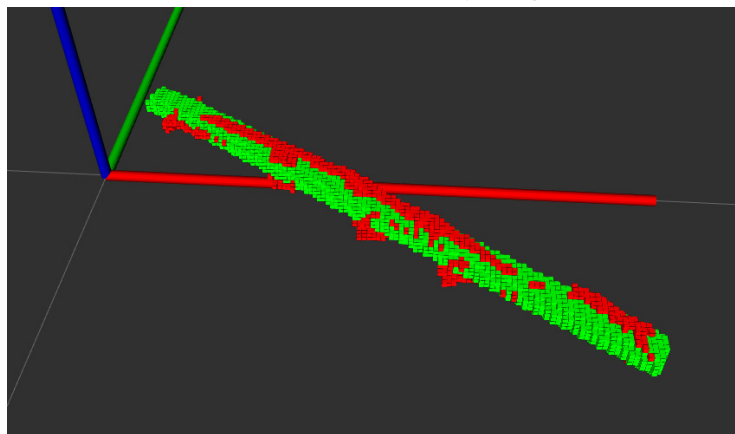
(b) Erreur de suivi



(c) Carte colorée après réobservation



(d) Objet segmenté après réobservation



(e) Résultat (rouge) et cible (vert)

FIGURE 8.13 : Résultats pour le tuyau

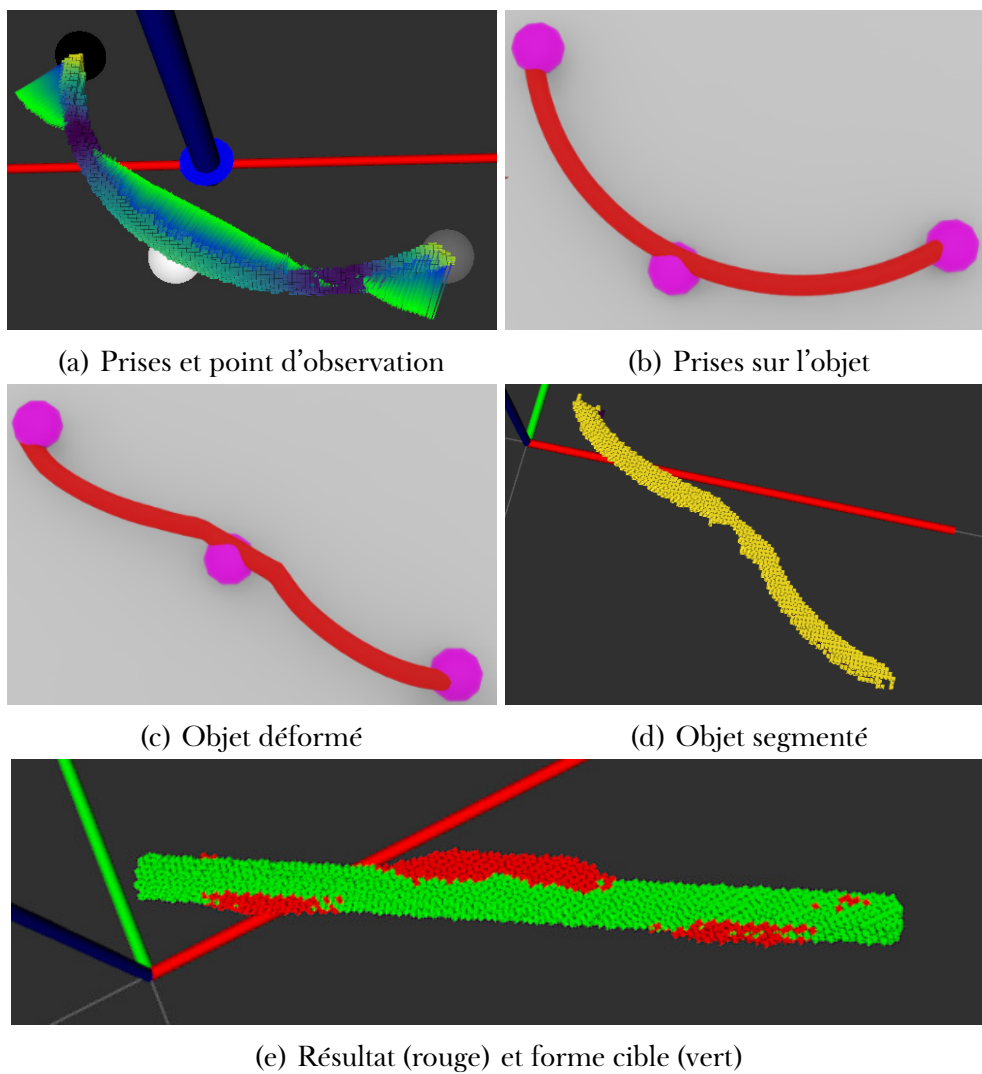
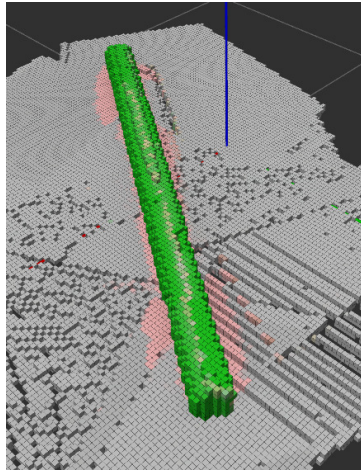


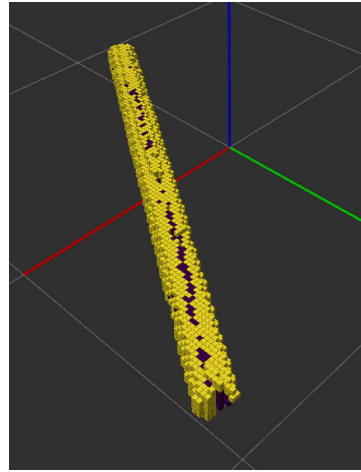
FIGURE 8.14 : Résultats pour le tuyau en optimisant les maximums locaux

8.1.4 Courber en « S » un long cylindre droit

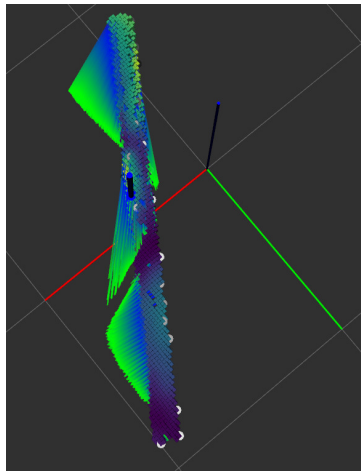
Le dernier scénario consiste à former un « S » avec le cylindre de 2 m de long par 10 cm de diamètre. En raison de sa taille, on réduit la résolution à des voxels de 16 mm pour accélérer le traitement. Une séquence de points de vue pour observer la forme initiale est montrée à l'annexe B.2 (figures B.5 à B.7). La figure 8.15 montre la situation initiale avec les points de prise choisis et la figure 8.16 montre le résultat de la manipulation. Ce cas est plus complexe que les précédents car il présente plusieurs points d'inflexion avec des déplacements requis différents. Malgré cette difficulté supplémentaire, la comparaison de formes par association des voxels (figure 8.15c) produit des vecteurs de déplacement qui correspondent aux mouvements attendus, et les points de prise sont concentrés dans les régions qui demandent le plus de contrôle, soit les extrémités et les creux du « S ». Le résultat obtenu après le déplacement des points de prise est une forme qui est très proche de la cible (figure 8.16d), confirmant que les vecteurs résultant de l'association des voxels offrent une bonne approximation du mouvement requis par les points de prise, même lorsque les propriétés du matériau de l'objet sont inconnues et que le contrôle est effectué en boucle ouverte. Les points de prise étant petits par rapport à la taille de l'objet et à la déformation souhaitée, on remarque qu'ils pourraient être regroupés ou que leur taille pourrait être augmentée pour obtenir un effet semblable avec un plus petit nombre de points de prise.



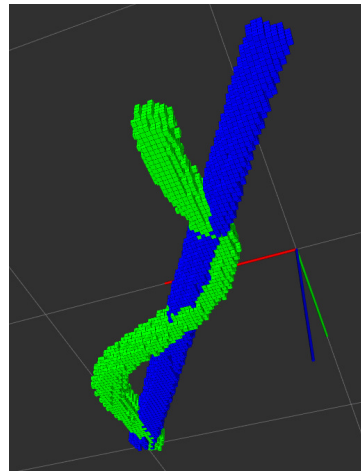
(a) Carte colorée



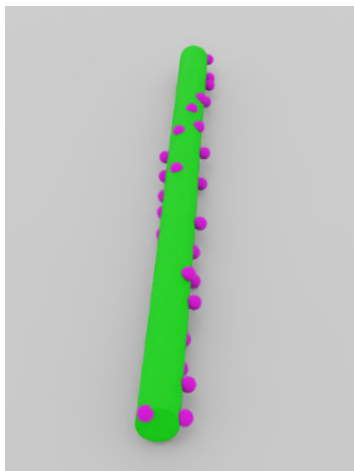
(b) Objet segmenté



(c) Prises et point d'observation

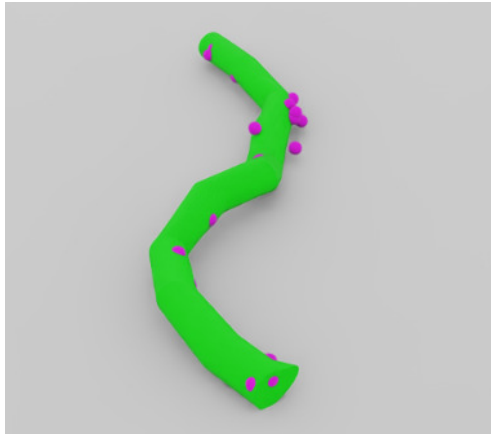


(d) Objet (bleu) et cible (vert)

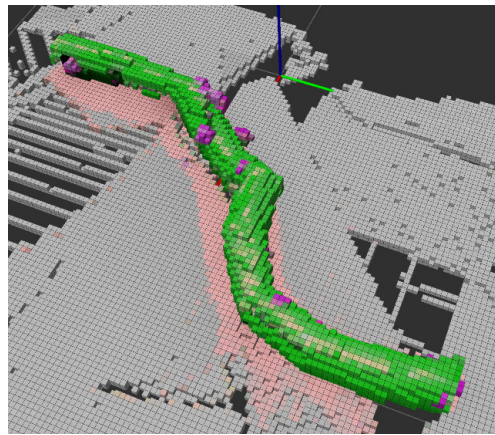


(e) Prises sur l'objet

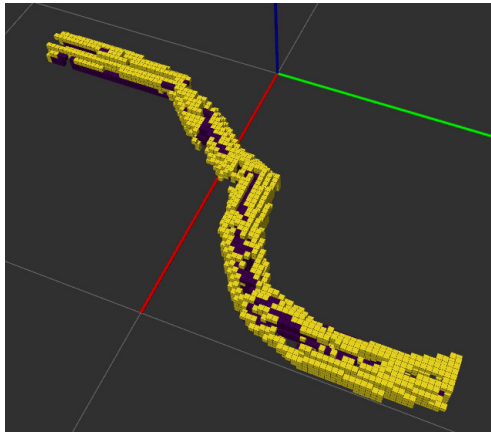
FIGURE 8.15 : Situation initiale (avant la manipulation) pour le cylindre



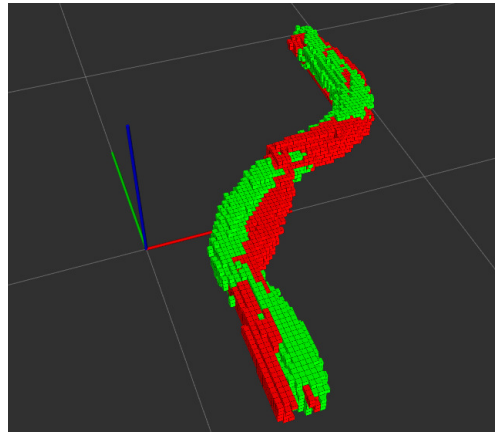
(a) Objet déformé



(b) Carte colorée après la manipulation



(c) Objet segmenté après la manipulation



(d) Objet (rouge) et cible (vert) après la manipulation

FIGURE 8.16 : Résultat du remodelage pour le cylindre

8.1.5 Résultats numériques

Le tableau 8.1 présente les valeurs numériques avant et après la déformation pour les scénarios ci-dessus selon les différents modes de comparaison utilisés à la section 7.4, soit la mesure IoU 3D (section 5.5.1), la différence utilisée pour l'alignement (d_{diff} , équation (7.3)), et le poids d'association, qui correspond à la distance que les voxels de la forme courante doivent parcourir pour atteindre la forme cible (d_{assoc} , équation (7.5)). La plupart des cas étudiés montrent une nette amélioration des différentes mesures (c'est-à-dire une augmentation du score IoU et une diminution des autres mesures), indiquant que la manipulation a rapproché la forme de l'objet de la forme cible. Les quelques exceptions sont cohérentes avec les résultats visuels : pour le scénario « éponge à sphère » (section 8.1.1), les cas avec des prises de 8 cm et deux prises de 5 cm ont causé un déplacement de l'objet d'intérêt qui réduit sa correspondance avec la forme cible. Bien que le déplacement soit moins clair visuellement, le scénario « sphère à cœur » (section 8.1.2) avec des prises de 8 cm peut s'expliquer de manière semblable puisque les deux prises sont situées du même côté de l'objet et ne respectent donc pas le bouclage des forces. Il est aussi intéressant de comparer les différentes tailles des points de prise pour le cas de la sphère qu'on tente de remodeler en cœur. Bien que le résultat avec des prises de 4 cm soit difficile à évaluer visuellement par rapport au prises de 8 cm (section 8.1.2), les résultats numériques montrent que la forme finale avec les prises de 4 cm est significativement plus proche de la forme cible que celle avec les prises de 8 cm. En général, les résultats numériques sont cohérents avec les observations visuelles et montrent que les prises et déplacement choisis sont en mesure de rapprocher l'objet d'intérêt de la forme cible.

TABLEAU 8.1 : Résultats numériques pour les déformations (comparaison avec la forme cible)

Scénario	IoU		d_{diff}		d_{assoc}		$\frac{d_{assoc}}{ M_a^d }$	
	Initial	Final	Initial	Final	Initial	Final	Initial	Final
Éponge et sphère (prises de 8 cm)	0,3952	0,4011	1067	1058	2044	2780	1,9524	2,5689
Éponge et sphère (prises de 5 cm)	0,3952	0,5007	1067	849	2044	1202	1,9524	1,1521
Éponge et sphère (2 prises de 5 cm)	0,3952	0,0182	1067	1244	2044	113	1,9524	4,1768
Sphère et ovale (ann. B.3)	0,6842	0,6114	582	640	1067	851	0,8757	0,8599
Sphère et cœur (prises de 8 cm)	0,3670	0,2988	4146	4746	12 344	14 525	1,8806	2,2243
Sphère et cœur (prises de 4 cm)	0,2852	0,3520	5879	4309	20 141	10 438	2,2794	1,5902
Sphère et cœur (prises de 7 cm)	0,2852	0,3557	5879	4261	20 140	11 091	2,2794	1,7024
Longue éponge (prises de 8 cm, ann. B.3)	0,2279	0,2466	2613	2309	8684	4701	3,8546	2,6543
Longue éponge (prises de 5 cm, ann. B.3)	0,2287	0,3008	2602	2137	8659	4767	3,8657	2,4498
Tuyau ($d = 0$)	0,0531	0,2892	5807	3720	17 866	2960	7,7985	1,2835
Tuyau ($d = 3$)	0,0531	0,0946	5807	5313	17 866	7459	7,7985	3,3892
Cylindre	0,1587	0,2596	6954	4728	55 861	4815	11,6257	1,6884

8.2 Analyse

Globalement, les résultats présentés dans cette thèse illustrent un système intégré capable de gérer l’observation, la représentation, la planification, la manipulation et le suivi de la déformation d’un objet non rigide tridimensionnel initialement inconnu. Au meilleur de nos connaissances, aucun tel système n’a encore été proposé dans la littérature. Couplé à un modèle d’occupation ternaire ou probabiliste pour s’adapter à la performance de la caméra, le module de cartographie et son mécanisme de filtres permettent la mise à jour en continu des cartes voxélisées représentant l’environnement, l’objet d’intérêt et les autres informations nécessaires à l’accomplissement de la tâche souhaitée. Cette capacité à créer et à manipuler des cartes pour extraire et mettre à jour en continu des informations à partir de cartes existantes distingue le système de cartographie proposé des solutions existantes telles que [90, 94, 95], axées sur l’enregistrement des données et la navigation.

Les algorithmes de segmentation présentés sont en mesure d’identifier la forme d’un objet d’intérêt initialement inconnu même dans des scénarios où il y a une certaine quantité de bruit et de variations de couleur, quoiqu’une optimisation des paramètres spécifique à chaque scénario demeure nécessaire pour obtenir les meilleurs résultats. En particulier, l’algorithme de segmentation hybride proposé a une performance de segmentation plus fiable que la segmentation par couleur communément utilisée [67, 80, 81] ou le *Fast Level-Sets* [8, 205], tout en demeurant parmi les meilleurs résultats en termes de performance et de temps de calcul.

Les mesures pour la sélection du prochain point de vue proposent des angles d’observation raisonnables qui, couplés à un algorithme d’observation adapté au robot et à l’environnement, permettent d’observer et de compléter efficacement la forme de l’objet d’intérêt. La méthode proposée est également plus rapide que les résultats présentés dans la littérature [97, 99, 100], ce qui est confirmé en la comparant avec une méthode de référence tirée de [102]. D’autres optimisations liées au robot permettraient des gains d’efficacité supplémentaires, par exemple l’ordonnancement des points de vue proposés pour minimiser la distance à parcourir et les sauts. Dans l’optique de développer des mesures et un systèmes indépendants de la tâche, de l’environnement et du robot, ces optimisations présentes dans plusieurs algorithmes de sélection du prochain point de vue n’étaient toutefois pas un objectif de cette thèse.

En ce qui concerne le critère pour détecter si l’objet d’intérêt est complet, on remarque qu’il résulte en une certaine tendance au perfectionnisme faisant en sorte que le système continue à tenter d’observer les quelques voxels manquants ou difficiles à observer (ce qui

dépend de la résolution, des paramètres de la caméra et de la distance d'observation) longtemps après que l'objet soit visuellement complet, restant parfois bloqué indéfiniment. Notons que cette question — est-ce qu'un ensemble de voxels donné représente la forme complète d'un objet initialement inconnu — est particulièrement difficile à résoudre avec certitude. Une fois que la forme initiale est obtenue, la comparaison avec la forme cible permet d'obtenir une approximation visuellement plausible de l'association entre les parties de la forme initiale et de la forme cible. Quoique cette association demande un temps de calcul significatif en fonction du nombre de voxels dans les objets, elle permet une comparaison plus robuste et informative que le score IoU et l'intersection des volumes [126], permettant de guider le remodelage d'objets réels plutôt que de comparer des formes abstraites comme dans [130].

Cette association et le principe de diminution de la rigidité permettent de proposer des points de prise plausibles pour le remodelage de l'objet d'intérêt vers la forme cible. On note ainsi que la taille et le nombre des régions de prise proposées sont des facteurs critiques au succès de la manipulation. En particulier lorsque le nombre de prises est limité en deçà du nombre proposé par l'identification des maximums locaux, la simple distance de déplacement n'est pas un critère suffisant pour assurer la sélection optimale des points de prise. Des facteurs additionnels comme le bouclage des forces ou la courbure devraient être pris en compte pour optimiser cet élément [8]. Malgré ces contraintes, la méthode proposée pour la sélection des points de prise ne demande aucune connaissance des propriétés de l'objet ni de simulation pour optimiser leur positionnement, ce qui est un avantage par rapport à la littérature existante [80, 82].

Concernant la déformation en tant que telle, on note que le simple déplacement en boucle ouverte (sans rétroaction) des points de prise semble beaucoup plus adapté aux grandes déformations (p. ex., tuyau, cylindre) qu'aux petites (p. ex., avec la sphère) pour l'obtention d'un résultat visuellement proche de la forme cible, bien que les scores numériques soient généralement améliorés par la manipulation. On remarque aussi les limites du simulateur et des paramètres de matériau choisis pour les objets, qui mènent occasionnellement à des artefacts de simulation (figures 8.3a et 8.5c) et semblent mieux adaptés aux déformations en compression qu'en étirement. Le suivi en temps réel de la déformation demeure limité par le nombre de capteurs disponibles, les occlusions par l'objet et les manipulateurs, ainsi que par le délai de mise à jour des cartes — à haute résolution, les cartes n'ont pas toujours le temps d'être mises à jour avant l'arrivée du prochain nuage de points. Cependant, la réobservation de l'objet sous plusieurs angles après la manipulation ou entre les différentes étapes de celle-ci permet de pallier à ces difficultés et d'obtenir la forme de l'objet déformé.

Ainsi, en intégrant le système de cartographie proposé avec les divers algorithmes, méthodes et mesures développées dans cette thèse, on obtient un système capable de gérer l'acquisition de données et l'extraction des informations nécessaires au remodelage d'objets déformables initialement inconnus dans un environnement partiellement observable. Cette portion du système reste de plus indépendante des contraintes spécifique du robot utilisé et de l'environnement de travail. Les résultats obtenus montrent que l'intégration de ce système avec les composantes liées à l'environnement robotique et à la planification de tâche et de mouvement permet d'effectuer, de bout en bout, une tâche de remodelage d'un objet déformable tridimensionnel dont les propriétés sont initialement inconnues, incluant l'observation initiale, la planification, le remodelage lui-même et la validation du résultat. Ceci représente une avancée significative par rapport aux capacités présentées dans la littérature existante.

En général, l'élément le moins robuste du système demeure sa capacité à détecter si l'objet d'intérêt est complet, qui est un problème particulièrement difficile lorsqu'on ne connaît pas sa forme au préalable. Celui-ci est suivi par l'alignement des formes qui, parce que basé uniquement sur l'analyse des axes principaux et du centroïde, tend à aligner les objets par leurs diagonales plutôt que par leur orientation naturelle et peine à aligner les objets qui n'ont pas d'intersection lorsque leurs centroïdes coïncident. En ce qui concerne la comparaison et l'association des voxels, l'amélioration principale requise est l'optimisation de l'implémentation pour réduire le temps de calcul et, pour la sélection de prise, l'inclusion de critères autres que la simple distance pour optimiser la taille et le nombre de points proposés. Au niveau de la segmentation, on note que malgré une performance de segmentation inférieure, le *Fast Level-Sets* conserve deux avantages significatifs sur l'algorithme hybride, soit l'évitement de la définition manuelle d'un seuil et la possibilité de faire le suivi d'une forme changeante sans la nécessité d'un point de référence fixe sur celle-ci après la première itération. Bien que les différents algorithmes laissent place à des améliorations individuelles, leur intégration donne lieu à un système global capable de gérer avec succès les tâches de remodelage d'objets déformables initialement inconnus.

Chapitre 9

Conclusion

9.1 Sommaire

Cette thèse présente un système de cartographie 3D et de planification pour la manipulation robotique d'objets déformables. Ce système flexible est capable de gérer, de bout en bout et dans un environnement tridimensionnel partiellement observable, la déformation et le remodelage d'un objet non rigide dont les caractéristiques de forme et la dynamique du matériau sont initialement inconnues. La manipulation robotique d'objets déformables est un problème complexe du fait de l'évolution de la forme des objets au cours de la manipulation. La gestion de cette complexité demande la conception et l'intégration efficace en un système cohérent de composantes réutilisables qui demeurent fonctionnelles dans des scénarios opérationnels impliquant une grande variabilité. Cette thèse présente le développement et l'intégration de tels composantes, en particulier un système de cartographie 3D original permettant d'aggréger et d'analyser les données des capteurs pour en extraire des mesures générales et indépendantes du robot et de la tâche pour l'observation et la manipulation. Ceci est intégré avec des composantes plus spécifiques à la tâche pour la planification de tâche et de mouvement pour former un système opérant dans un environnement robotique réel ou simulé.

En plus de la conception du système global, cette thèse propose divers algorithmes, méthodes et mesures pour optimiser les différentes étapes de l'observation et du remodelage d'objets déformables tridimensionnels. Ainsi, concernant la segmentation de l'objet d'intérêt dans des grilles de voxels, on compare les méthodes basées sur le seuillage de couleur à une formulation 3D multi-canaux originale de l'algorithme *Fast Level-Sets*. Ceci permet de proposer une méthode de segmentation hybride originale combinant les avantages des deux classes d'algorithmes. On propose également des méthodes pour raffiner et compléter la forme segmentée.

En ce qui concerne l’observation de la forme complète de l’objet, cette thèse présente une méthode basée sur l’analyse géométrique permettant d’identifier des points de vue appropriés pour observer les parties manquantes de l’objet d’intérêt avec des capteurs visuels et de profondeur. Cette mesure est rapide à calculer et indépendante des contraintes spécifiques du capteur, du robot et de l’environnement, soutenant son utilisation pour construire des algorithmes d’observations adaptables à différents scénarios.

Pour planifier la manipulation, cette thèse propose une méthode pour guider le remodelage d’objets 3D représentés par des grilles de voxels. Cette méthode est basée sur une mesure de comparaison de formes voxélisées originale qui respecte les contraintes et la réalité physique du remodelage d’objets déformables de diverses tailles. Cette mesure permet ainsi la définition d’heuristiques pour identifier le nombre, la position et le mouvement souhaité des points de prise pour une tâche de remodelage donnée.

L’évaluation expérimentale confirme la validité et l’utilité des mesures et méthodes proposées pour l’observation et le remodelage d’objets déformables tridimensionnels, ainsi que leur intégration dans un système global. Ainsi, le système et les algorithmes développés dans le cadre de cette thèse offrent, par rapport à la littérature existante, un progrès significatif vers un système intégré pour l’observation, la planification et l’exécution dans une tâche de remodelage d’un objet déformable tridimensionnel initialement inconnu.

9.2 Contributions

Les contributions originales de cette thèse sont concentrées en trois axes principaux, soit la conception du système global, la cartographie et la planification. Ainsi, on retrouve les contributions principales suivantes :

1. Conception d’un système global pour le remodelage d’objets déformables dont les propriétés sont initialement inconnues dans un environnement 3D partiellement observable.
2. Développement d’un système de cartographie pour le remodelage d’objets déformables. Celui-ci supporte les opérations nécessaires au remodelage dont la segmentation, la comparaison de formes et le suivi de la déformation.
3. Développement d’une méthode rapide et efficace pour identifier des points de vue permettant de compléter la forme de l’objet d’intérêt, indépendamment des propriétés du robot et de la caméra.

4. Développement d'une méthode de comparaison de formes 3D adaptée au remodelage d'objets déformables et permettant d'extraire des mesures pour guider la sélection des points de prise et leur déplacement.

Ces contributions principales s'accompagnent d'une série de contributions complémentaires, dont notamment :

1. Une formulation 3D de la segmentation par *Fast Level-Sets* et le développement d'un algorithme de segmentation hybride original.
2. Le développement d'algorithmes de remplissage et de raffinement post-segmentation pour réduire les suppositions liées à l'environnement.
3. Le développement d'algorithmes d'observation adaptant les points de vue proposés aux contraintes de l'environnement.
4. La conception de modules réutilisables et indépendants de la tâche et de l'environnement pour les différentes composantes du système.

Jusqu'à présent, les travaux effectués dans le cadre de cette thèse ont directement mené à quatre publications. Les défis concernant l'observation, la modélisation, la prédiction et le contrôle de la forme d'objets déformables tridimensionnels ont été identifiés dans [190] avec un cadre de travail initial pour coordonner la manipulation d'objets non rigides tridimensionnels quelconques. Dans [191], ce cadre de travail initial a été étendu et intégré au sein d'une architecture globale avec les modules de cartographie, de modélisation et de planification présentés dans cette thèse. Les méthodes proposées pour la sélection de points de vue pour observer et compléter la forme de l'objet d'intérêt ont été publiées dans [195], alors que celles pour la comparaison de formes voxélisées et la sélection des points de prise ont été publiées dans [196]. Des publications supplémentaires sont en préparation pour couvrir les autres éléments du système dont les algorithmes de segmentation pour des grilles de voxels, les détails du module de cartographie, et l'intégration globale des différents éléments présentés.

9.3 Travaux futurs

Les travaux présentés dans cette thèse laissent place à une série de directions possibles pour le futur. En premier lieu on note que la prochaine étape du projet à long terme sur la manipulation d'objets déformables dans le cadre duquel cette thèse s'inscrit serait l'intégration du module de modélisation de la dynamique d'objet et de prédiction de forme [12] dans le système global développé ici. Cette intégration permettrait au système de gérer des manipulations complexes demandant une planification en plusieurs étapes tout en s'adaptant aux propriétés physiques de l'objet. De même, elle permettrait un raffinement des points de prise et des mouvements effectués, augmentant ainsi la précision et la qualité du remodelage. Pour effectuer cette intégration, le travail principal serait le développement d'un planificateur à long terme tirant parti du modèle prédictif et des mécanismes de cartographie pour choisir et séquencer plusieurs actions en tenant compte de l'incertitude dans la forme de l'objet et dans son comportement. Ce planificateur avancé permettrait également de prendre en compte des actions plus délicates comme la torsion de l'objet à remodeler, et de poursuivre sur des robots physiques l'étude des concepts validés en simulation dans cette thèse.

En plus de cette étape majeure, on peut entrevoir un certain nombre d'améliorations futures ciblant les différents éléments présentés. Pour la segmentation, l'amélioration principale consisterait à ramener certains avantages du *Fast Level-Sets* dans l'algorithme hybride, de manière à éviter la définition manuelle d'un seuil de segmentation et la nécessité d'un point de référence fixe sur l'objet après la première itération. L'algorithme hybride évolutif résultant de ces modifications aurait ainsi le potentiel d'offrir la meilleure performance et convivialité d'utilisation dans différents scénarios et environnements.

Au niveau de l'observation, les algorithmes pour détecter si les voxels segmentés représentent la forme complète de l'objet ainsi que pour en faire le remplissage et le raffinement pourraient être améliorés pour augmenter la robustesse du processus et ainsi réduire le nombre d'observations superflues pour compléter la forme de l'objet. Ces améliorations passeraient sans doute par le raffinement des critères pour identifier les voxels non-observables et détecter si la forme est complète. De plus, l'algorithme d'observation devrait être optimisé en fonction du robot utilisé, par exemple pour combiner une séquence de points de vue proposés en une trajectoire qui réduit le déplacement nécessaire pour le robot.

Concernant la comparaison entre la forme initiale et la forme cible, le développement d'un critère plus évolué que le simple calcul des axes principaux permettrait un alignement plus naturel des formes, particulièrement lorsqu'il y a peu d'intersections entre celles-ci. Le

développement de mesures plus spécifiques pour le calcul du poids des arêtes dans l'association des voxels pourrait optimiser la correspondance entre les voxels de la forme initiale et de la forme cible. La conception d'un algorithme d'association plus efficace en temps de calcul permettrait également une meilleure performance à haute résolution.

Bien que l'optimisation de l'association entre les formes initiale et cible bénéficierait aussi à la sélection de prise, il serait également pertinent d'étudier et d'intégrer au système différents critères de sélection de prise permettant de s'adapter aux contraintes de la tâche, du robot et de l'environnement. On pense notamment aux différents types et géométries de préhenseurs, au bouclage des forces et à la courbure locale. Pour l'exécution et le suivi de la manipulation, il conviendrait de développer un modèle de contrôle plus adapté aux petites déformations que le déplacement en boucle ouverte des points de prise, et d'ajuster les paramètres de simulation pour mieux correspondre aux objets réels et éviter les artefacts. Dans un contexte où plusieurs capteurs pourraient être disponibles, il serait aussi intéressant d'effectuer le suivi de la manipulation sous plusieurs angles.

Références

- [1] E. Ackerman, “Meet the Lunar Gateway’s Robot Caretakers,” *IEEE Spectrum*, Avr. 2022. [En ligne]. Disponible : <https://spectrum.ieee.org/lunar-gateway-robots>
- [2] C. Warner, “NASA Outlines Lunar Surface Sustainability Concept,” Mar. 2020. [En ligne]. Disponible : <http://www.nasa.gov/feature/nasa-outlines-lunar-surface-sustainability-concept>
- [3] A. A. J. B. R. Morris, “Deliberation For Intra-Vehicle Robotic Activities In Space,” dans *2021 IEEE International Conference on Autonomous Systems (ICAS)*, Aoû. 2021, pp. 1–5. doi : 10.1109/ICAS49788.2021.9551182
- [4] A.-M. Cretu, P. Payeur, et E. Petriu, “Neural Network Mapping and Clustering of Elastic Behavior From Tactile and Range Imaging for Virtualized Reality Applications,” *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 9, pp. 1918–1928, Sep. 2008. doi : 10.1109/TIM.2008.919904
- [5] A.-M. Cretu, E. M. Petriu, P. Payeur, et F. F. Khalil, “Deformable Object Segmentation and Contour Tracking in Image Sequences Using Unsupervised Networks,” dans *Canadian Conference on Computer and Robot Vision*, Mai 2010, pp. 277–284. doi : 10.1109/CRV.2010.43
- [6] F. F. Khalil, P. Payeur, et A.-M. Cretu, “Integrated Multisensory Robotic Hand System for Deformable Object Manipulation,” dans *LASTED Technology Conferences / 705 : ARP / 706 : RA / 707 : NANA / 728 : CompBIO*. ACTA Press, Nov. 2010. doi : 10.2316/P.2010.706-063
- [7] F. Hui, “Visual Tracking of Deformation and Classification of Object Elasticity with Robotic Hand Probing,” Thèse de maîtrise, Université d’Ottawa / University of Ottawa, Aoû. 2017. doi : 10.20381/ruor-20757
- [8] F. Nadon, “Sélection de prise et contrôle d’une main robotique pour la manipulation d’objets non rigides,” Thèse de maîtrise, Université d’Ottawa / University of Ottawa, Déc. 2019. doi : 10.20381/ruor-24234
- [9] S. Rowlands, “Robotic Control for the Manipulation of 3D Deformable Objects,” Thèse de maîtrise, Université d’Ottawa / University of Ottawa, Aoû. 2021. doi : 10.20381/ruor-26774

- [10] A. Valencia, “3D Shape Deformation Measurement and Dynamic Representation for Non-Rigid Objects under Manipulation,” Thèse de maîtrise, Université d’Ottawa / University of Ottawa, Jul. 2020. doi : 10.20381/ruor-24946
- [11] A. J. Valencia et P. Payeur, “Deformation Modeling for the Robotic Manipulation of 3D Elastic Objects using Physics-Informed Graph Neural Networks,” dans *20th Conference on Robots and Vision (CRV)*, Jun. 2023, pp. 194–201. doi : 10.1109/CRV60082.2023.00032
- [12] A. J. Valencia et P. Payeur, “Efficient Dynamic Modeling for 3D Deformable Object Manipulation via Physics-Informed Learning,” dans *22nd International Conference on Advanced Robotics*. San Juan, Argentina : IEEE, Déc. 2025.
- [13] F. Nadon, A. Valencia, et P. Payeur, “Multi-Modal Sensing and Robotic Manipulation of Non-Rigid Objects : A Survey,” *Robotics*, vol. 7, no. 4, p. 74, Nov. 2018. doi : 10.3390/robotics7040074
- [14] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, et Y. Mezouar, “Robotic manipulation and sensing of deformable objects in domestic and industrial applications : A survey,” *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, Jun. 2018. doi : 10.1177/0278364918779698
- [15] R. Herguedas, G. López-Nicolás, R. Aragüés, et C. Sagüés, “Survey on multi-robot manipulation of deformable objects,” dans *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 977–984. doi : 10.1109/ETFA.2019.8868987
- [16] M. Yan, G. Li, Y. Zhu, et J. Bohg, “Learning Topological Motion Primitives for Knot Planning,” dans *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) IROS 2020*. Las Vegas, NV, USA (Virtual) : IEEE, Oct. 2020, p. 8. doi : 10.1109/IROS45743.2020.9341330
- [17] K. Suzuki, M. Kanamura, Y. Suga, H. Mori, et T. Ogata, “In-air Knotting of Rope using Dual-Arm Robot based on Deep Learning,” dans *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 6724–6731. doi : 10.1109/IROS51168.2021.9635954
- [18] A. Sintov, S. Macenski, A. Borum, et T. Bretl, “Motion Planning for Dual-Arm Manipulation of Elastic Rods,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6065–6072, Oct. 2020. doi : 10.1109/LRA.2020.3011352
- [19] P. Zhou, J. Zhu, S. Huo, et D. Navarro-Alarcon, “LaSeSOM : A Latent and Semantic Representation Framework for Soft Object Manipulation,” *IEEE Robotics and Automation Letters*, pp. 1–1, 2021. doi : 10.1109/LRA.2021.3074872
- [20] J. Qi, G. Ma, P. Zhou, H. Zhang, Y. Lyu, et D. Navarro-Alarcon, “Towards latent space based manipulation of elastic rods using autoencoder models and robust centerline extractions,” *Advanced Robotics*, vol. 36, no. 3, pp. 101–115, Fév. 2022. doi : 10.1080/01691864.2021.2004222

- [21] R. Lagneau, A. Krupa, et M. Marchal, “Automatic Shape Control of Deformable Wires Based on Model-Free Visual Servoing,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, Oct. 2020. doi : 10.1109/LRA.2020.3007114
- [22] F. Mou et D. Wu, “Control Method for Robotic Manipulation of Heavy Industrial Cables,” dans *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aoû. 2021, pp. 963–969. doi : 10.1109/ICMA52036.2021.9512586
- [23] P. Chang et T. Padır, “Model-Based Manipulation of Linear Flexible Objects with Visual Curvature Feedback,” dans *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2020, pp. 1406–1412. doi : 10.1109/AIM43001.2020.9159044
- [24] P. Chang et T. Padır, “Sim2Real2Sim : Bridging the Gap Between Simulation and Real-World in Flexible Object Manipulation,” dans *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, Nov. 2020, pp. 56–62. doi : 10.1109/IRC.2020.00015
- [25] J. Zhu, B. Navarro, R. Passama, P. Fraisse, A. Crosnier, et A. Cherubini, “Robotic Manipulation Planning for Shaping Deformable Linear Objects With Environmental Contacts,” *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 16–23, Jan. 2020. doi : 10.1109/LRA.2019.2944304
- [26] J. Dirr, D. Gebauer, et R. Daub, “Localization and Grasp Planning for Bin Picking of Deformable Linear Objects,” dans *Procedia CIRP*, ser. 16th CIRP Conference on Intelligent Computation in Manufacturing Engineering, vol. 118, Italy, Jan. 2023, pp. 235–240. doi : 10.1016/j.procir.2023.06.041
- [27] K. Chen, Z. Bing, F. Wu, Y. Meng, A. Kraft, S. Haddadin, et A. Knoll, “Contact-Aware Shaping and Maintenance of Deformable Linear Objects With Fixtures,” dans *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 1–8. doi : 10.1109/IROS55552.2023.10341726
- [28] M. Yu, K. Lv, C. Wang, Y. Jiang, M. Tomizuka, et X. Li, “Generalizable whole-body global manipulation of deformable linear objects by dual-arm robot in 3-D constrained environments,” *The International Journal of Robotics Research*, Sep. 2024. doi : 10.1177/02783649241276886
- [29] C. Ying et K. Yamazaki, “Motion Generation for Shaping Deformable Linear Objects with Contact Avoidance Using Differentiable Simulation *,” dans *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Déc. 2023, pp. 1–8. doi : 10.1109/ROBIO58561.2023.10355033
- [30] M. Yu, K. Lv, H. Zhong, S. Song, et X. Li, “Global Model Learning for Large Deformation Control of Elastic Deformable Linear Objects : An Efficient and Adaptive Approach,” *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 417–436, Fév. 2023. doi : 10.1109/TRO.2022.3200546

- [31] D. Tong, A. Choi, L. Qin, W. Huang, J. Joo, et M. K. Jawed, “Sim2Real Neural Controllers for Physics-Based Robotic Deployment of Deformable Linear Objects,” *The International Journal of Robotics Research*, vol. 43, no. 6, pp. 791–810, Mai 2024. doi : 10.1177/02783649231214553
- [32] N. Lv, J. Liu, et Y. Jia, “Dynamic Modeling and Control of Deformable Linear Objects for Single-Arm and Dual-Arm Robot Manipulations,” *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2341–2353, Aoû. 2022. doi : 10.1109/TRO.2021.3139838
- [33] H. Du, Q. Jiang, et W. Xiong, “Dynamic geometrical configuration predictions during robotic manipulation for automated cable assembly,” *Journal of Manufacturing Systems*, vol. 64, pp. 121–132, Jul. 2022. doi : 10.1016/j.jmsy.2022.06.001
- [34] X. Ma, D. Hsu, et W. S. Lee, “Learning Latent Graph Dynamics for Visual Manipulation of Deformable Objects,” dans *2022 International Conference on Robotics and Automation (ICRA)*, Mai 2022, pp. 8266–8273. doi : 10.1109/ICRA46639.2022.9811597
- [35] R. Laezza, R. Gieselmann, F. T. Pokorny, et Y. Karayiannidis, “ReForm : A Robot Learning Sandbox for Deformable Linear Object Manipulation,” dans *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi’an, China : IEEE, Mai 2021, pp. 4717–4723. doi : 10.1109/ICRA48506.2021.9561766
- [36] Y. Gao, Z. Chen, Y.-H. Liu, M. Fang, et J. Lin, “Vision-Based Grasping and Manipulation of Flexible USB Wires,” dans *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Déc. 2019, pp. 1083–1088. doi : 10.1109/ROBIO49542.2019.8961848
- [37] M. L. Moreira, “Visual servoing on deformable objects : An application to tether shape control,” Thèse de doctorat, Université de Toulon, Avr. 2019. [En ligne]. Disponible : <https://tel.archives-ouvertes.fr/tel-02534996>
- [38] D. Sanchez, W. Wan, et K. Harada, “Dual-Armed Manipulation Planning for Tethered Tools,” dans *Workshop on Robotic Manipulation of Deformable Objects (ROMADO), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, p. 4.
- [39] L. Cao, X. Li, P. T. Phan, A. M. H. Tiong, H. L. Kaan, J. Liu, W. Lai, Y. Huang, H. M. Le, M. Miyasaka, K. Y. Ho, P. W. Y. Chiu, et S. J. Phee, “Sewing up the Wounds : A Robotic Suturing System for Flexible Endoscopy,” *IEEE Robotics Automation Magazine*, vol. 27, no. 3, pp. 45–54, Sep. 2020. doi : 10.1109/MRA.2019.2963161
- [40] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, K. Yamane, S. Iba, J. F. Canny, et K. Goldberg, “Deep Imitation Learning of Sequential Fabric Smoothing from an Algorithmic Supervisor,” dans *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA (Virtual) : IEEE, Oct. 2020, p. 8. doi : 10.1109/IROS45743.2020.9341608

- [41] B. Jia, Z. Pan, Z. Hu, J. Pan, et D. Manocha, “Cloth Manipulation Using Random-Forest-Based Imitation Learning,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2086–2093, Avr. 2019. doi : 10.1109/LRA.2019.2897370
- [42] R. Jangir, G. Alenyà, et C. Torras, “Dynamic Cloth Manipulation with Deep Reinforcement Learning,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 4630–4636. doi : 10.1109/ICRA40945.2020.9196659
- [43] R. Lee, J. Abou-Chakra, F. Zhang, et P. Corke, “Learning Fabric Manipulation in the Real World with Human Videos,” dans *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan : IEEE, Mai 2024, pp. 3124–3130. doi : 10.1109/ICRA57147.2024.10610062
- [44] X. Zhang, C. Li, et N. Xi, “Cable Assembly Based on Robot Manipulation and Control,” dans *2022 12th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Jul. 2022, pp. 597–602. doi : 10.1109/CYBER55403.2022.9907067
- [45] M. H. Daniel Zakaria, M. Aranda, L. Lequière, S. Lengagne, J. A. Corrales Ramón, et Y. Mezouar, “Robotic Control of the Deformation of Soft Linear Objects Using Deep Reinforcement Learning,” dans *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, Aoû. 2022, pp. 1516–1522. doi : 10.1109/CASE49997.2022.9926667
- [46] H. Shehawy, D. Pareyson, V. Caruso, S. De Bernardi, A. M. Zanchettin, et P. Rocco, “Flattening and folding towels with a single-arm robot based on reinforcement learning,” *Robotics and Autonomous Systems*, vol. 169, Nov. 2023. doi : 10.1016/j.robot.2023.104506
- [47] F. Coltraro, J. Borràs, M. Alberich-Carramiñana, et C. Torras, “Tracking cloth deformation : A novel dataset for closing the sim-to-real gap for robotic cloth manipulation learning,” *The International Journal of Robotics Research*, Fév. 2025. doi : 10.1177/02783649251317617
- [48] N. Fiedler, Y. Jonetzko, et J. Zhang, “A Multimodal Pipeline for Grasping Fabrics from Flat Surfaces with Tactile Slip and Fall Detection,” dans *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Déc. 2023, pp. 1–6. doi : 10.1109/ROBIO58561.2023.10354620
- [49] Y. Li, Y. Wang, Y. Yue, D. Xu, M. Case, S.-F. Chang, E. Grinspun, et P. K. Allen, “Model-Driven Feedforward Prediction for Manipulation of Deformable Objects,” *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1621–1638, Oct. 2018. doi : 10.1109/TASE.2017.2766228

- [50] R. K. Malhan, R. Jomy Joseph, A. V. Shembekar, A. M. Kabir, P. M. Bhatt, et S. K. Gupta, “Online Grasp Plan Refinement for Reducing Defects During Robotic Layup of Composite Prepreg Sheets,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France : IEEE, Mai 2020, pp. 11 500–11 507. doi : 10.1109/ICRA40945.2020.9196876
- [51] O. M. Manyar, J. Desai, N. Deogaonkar, R. J. Joesph, R. Malhan, Z. McNulty, B. Wang, J. Barbic, et S. K. Gupta, “A Simulation-Based Grasp Planner for Enabling Robotic Grasping during Composite Sheet Layup,” dans *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi’an, China : IEEE, Mai 2021, pp. 930–937. doi : 10.1109/ICRA48506.2021.9560939
- [52] M. Aranda, J. A. C. Ramon, Y. Mezouar, A. Bartoli, et E. Özgür, “Monocular Visual Shape Tracking and Servoing for Isometrically Deforming Objects,” dans *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 7542–7549. doi : 10.1109/IROS45743.2020.9341646
- [53] Z. Cui, K. Huang, B. Lu, et H. K. Chu, “Automated Folding of a Deformable Thin Object through Robot Manipulators,” dans *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 4241–4248. doi : 10.1109/IROS45743.2020.9341239
- [54] R. Herguedas, G. López-Nicolás, et C. Sagüés, “Collision-free transport of 2D deformable objects,” dans *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, Oct. 2021, pp. 430–435. doi : 10.23919/ICCAS52745.2021.9650027
- [55] Z. Weng, F. Paus, A. Varava, H. Yin, T. Asfour, et D. Kragic, “Graph-based Task-specific Prediction Models for Interactions between Deformable and Rigid Objects,” dans *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 5741–5748. doi : 10.1109/IROS51168.2021.9636660
- [56] C. Chi et D. Berenson, “Occlusion-robust Deformable Object Tracking without Physics Simulation,” dans *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6443–6450. doi : 10.1109/IROS40897.2019.8967827
- [57] Y. Wang, D. McConachie, et D. Berenson, “Tracking Partially-Occluded Deformable Objects while Enforcing Geometric Constraints,” dans *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi’an, China : IEEE, Mai 2021, pp. 14 199–14 205. doi : 10.1109/ICRA48506.2021.9561012
- [58] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. LI, J. Pan, W. Yuan, et M. Gienger, “Challenges and Outlook in Robotic Manipulation of Deformable Objects,” *IEEE Robotics Automation Magazine*, pp. 2–12, 2022. doi : 10.1109/MRA.2022.3147415

- [59] S. Makris, F. Dietrich, K. Kellens, et S. J. Hu, “Automated assembly of non-rigid objects,” *CIRP Annals*, Jun. 2023. doi : 10.1016/j.cirp.2023.05.003
- [60] H. Yin, A. Varava, et D. Kragic, “Modeling, learning, perception, and control methods for deformable object manipulation,” *Science Robotics*, vol. 6, no. 54, Mai 2021. doi : 10.1126/scirobotics.abd8803
- [61] S. A. Pedram, P. W. Ferguson, C. Shin, A. Mehta, E. P. Dutson, F. Alambeigi, et J. Rosen, “Toward Synergic Learning for Autonomous Manipulation of Deformable Tissues via Surgical Robots : An Approximate Q-Learning Approach,” dans *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, Nov. 2020, pp. 878–884. doi : 10.1109/BioRob49111.2020.9224421
- [62] D. Meli, E. Tagliabue, D. Dall’Alba, et P. Fiorini, “Autonomous tissue retraction with a biomechanically informed logic based framework,” dans *2021 International Symposium on Medical Robotics (ISMR)*, Nov. 2021, pp. 1–7. doi : 10.1109/ISMR48346.2021.9661573
- [63] Z. Hu, T. Han, P. Sun, J. Pan, et D. Manocha, “3-D Deformable Object Manipulation Using Deep Neural Networks,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4255–4261, Oct. 2019. doi : 10.1109/LRA.2019.2930476
- [64] F. Ficuciello, A. Migliozi, E. Coevoet, A. Petit, et C. Duriez, “FEM-Based Deformation Control for Dexterous Manipulation of 3D Soft Objects,” dans *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 4007–4013. doi : 10.1109/IROS.2018.8593512
- [65] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, et J. L. Wyatt, “Modeling of Deformable Objects for Robotic Manipulation : A Tutorial and Review,” *Frontiers in Robotics and AI*, vol. 7, 2020. doi : 10.3389/frobt.2020.00082
- [66] H. Deng, F. Ahmad, J. Xiong, et Z. Xia, “A Robot-Object Unified Modeling Method for Deformable Object Manipulation in Constrained Environments,” *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 6, pp. 4262–4273, Déc. 2024. doi : 10.1109/TMECH.2024.3371111
- [67] C. Saghour, D. Navarro-Alarcón, P. Fraisse, et A. Cherubini, “Dual-arm shaping of soft objects in 3D based on visual servoing and online FEM simulations,” *The International Journal of Robotics Research*, Déc. 2024. doi : 10.1177/02783649241301076
- [68] F. Makiyeh, F. Chaumette, M. Marchal, et A. Krupa, “Shape Servoing of a Soft Object Using Fourier Series and a Physics-Based Model,” dans *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 6356–6363. doi : 10.1109/IROS55552.2023.10342354

- [69] B. Yang, B. Lu, W. Chen, F. Zhong, et Y.-H. Liu, “Model-Free 3-D Shape Control of Deformable Objects Using Novel Features Based on Modal Analysis,” *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 3134–3153, Aoû. 2023. doi : 10.1109/TRO.2023.3269347
- [70] B. Yang, C. Sui, F. Zhong, et Y.-H. Liu, “Modal-graph 3D shape servoing of deformable objects with raw point clouds,” *The International Journal of Robotics Research*, vol. 42, no. 14, pp. 1213–1244, Déc. 2023. doi : 10.1177/02783649231198900
- [71] A. Sengupta, R. Lagneau, A. Krupa, E. Marchand, et M. Marchal, “Simultaneous Tracking and Elasticity Parameter Estimation of Deformable Objects,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 10 038–10 044. doi : 10.1109/ICRA40945.2020.9196770
- [72] Y. Zhu, K. Lu, et K. Hauser, “Semi-Empirical Simulation of Learned Force Response Models for Heterogeneous Elastic Objects,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 1646–1652. doi : 10.1109/ICRA40945.2020.9197077
- [73] A. J. Valencia et P. Payeur, “Combining Self-Organizing and Graph Neural Networks for Modeling Deformable Objects in Robotic Manipulation,” *Frontiers in Robotics and AI*, vol. 7, 2020. doi : 10.3389/frobt.2020.600584
- [74] J. M. S. Loza, “Shape sensing of deformable objects for robot manipulation,” Thèse de doctorat, Université Clermont Auvergne, Mai 2019. [En ligne]. Disponible : <https://tel.archives-ouvertes.fr/tel-02294752>
- [75] J. Sanchez, C. M. Mateo, J. A. Corrales, B. Bouzgarrou, et Y. Mezouar, “Online Shape Estimation based on Tactile Sensing and Deformation Modeling for Robot Manipulation,” dans *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 504–511. doi : 10.1109/IROS.2018.8594314
- [76] J. Sanchez, K. Mohy El Dine, J. A. Corrales, B.-C. Bouzgarrou, et Y. Mezouar, “Blind Manipulation of Deformable Objects Based on Force Sensing and Finite Element Modeling,” *Frontiers in Robotics and AI*, vol. 7, 2020. doi : 10.3389/frobt.2020.00073
- [77] S. Cui, R. Wang, J. Wei, F. Li, et S. Wang, “Grasp State Assessment of Deformable Objects Using Visual-Tactile Fusion Perception,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 538–544. doi : 10.1109/ICRA40945.2020.9196787
- [78] J. Xu, M. Danielczuk, J. Ichnowski, J. Mahler, E. Steinbach, et K. Goldberg, “Minimal Work : A Grasp Quality Metric for Deformable Hollow Objects,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 1546–1552. doi : 10.1109/ICRA40945.2020.9197062

- [79] L. Zaidi, J. A. Corrales Ramon, L. Sabourin, B. C. Bouzgarrou, et Y. Mezouar, “Grasp Planning Pipeline for Robust Manipulation of 3D Deformable Objects with Industrial Robotic Hand + Arm Systems,” *Applied Sciences*, vol. 10, no. 23, p. 8736, Jan. 2020. doi : 10.3390/app10238736
- [80] H. Shi, H. Xu, Z. Huang, Y. Li, et J. Wu, “RoboCraft : Learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks,” *The International Journal of Robotics Research*, Déc. 2023. doi : 10.1177/02783649231219020
- [81] A. Bartsch, C. Avra, et A. B. Farimani, “SculptBot : Pre-Trained Models for 3D Deformable Object Manipulation,” dans *IEEE International Conference on Robotics and Automation*. Yokohama, Japan : IEEE, Mai 2024, pp. 12 548–12 555. doi : 10.1109/ICRA57147.2024.10610899
- [82] I. Cuiral-Zueco, G. López-Nicolás, et H. Araujo, “Gripper positioning for object deformation tasks,” dans *International Conference on Robotics and Automation (ICRA)*, Mai 2022, pp. 963–969. doi : 10.1109/ICRA46639.2022.9812304
- [83] F. Hui, P. Payeur, et A.-M. Cretu, “Visual Tracking of Deformation and Classification of Non-Rigid Objects with Robot Hand Probing,” *Robotics*, vol. 6, no. 1, p. 5, Mar. 2017. doi : 10/gfs798
- [84] A. Gonnochenko, A. Semochkin, D. Egorov, D. Statovoy, S. Zabihifar, A. Postnikov, E. Seliverstova, A. Zaidi, J. Stemmler, et K. Limkraitassiri, “Coinbot : Intelligent Robotic Coin Bag Manipulation Using Artificial Brain,” dans *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, Fév. 2021, pp. 67–74. doi : 10.1109/ICARA51699.2021.9376455
- [85] M. C. Nah, A. Krotov, M. Russo, D. Sternad, et N. Hogan, “Manipulating a Whip in 3D via Dynamic Primitives*,” dans *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 2803–2808. doi : 10.1109/IROS51168.2021.9636257
- [86] Y. Cai, F. Kong, Y. Ren, F. Zhu, J. Lin, et F. Zhang, “Occupancy Grid Mapping Without Ray-Casting for High-Resolution LiDAR Sensors,” *IEEE Transactions on Robotics*, vol. 40, pp. 172–192, 2024. doi : 10.1109/TRO.2023.3323936
- [87] F. Furrer, T. Novkovic, M. Fehr, M. Grinvald, C. Cadena, J. Nieto, et R. Siegwart, “Modelify : An approach to incrementally build 3D object models for map completion,” *The International Journal of Robotics Research*, vol. 42, no. 3, pp. 45–65, Mar. 2023. doi : 10.1177/02783649231166977
- [88] Z. Xu, Z. He, J. Wu, et S. Song, “Learning 3D Dynamic Scene Representations for Robot Manipulation,” dans *Proceedings of the 2020 Conference on Robot Learning*. PMLR, Oct. 2021, pp. 126–142. [En ligne]. Disponible : <https://proceedings.mlr.press/v155/xu21b.html>

- [89] K. M. Wurm, D. Hennes, D. Holz, R. B. Rusu, C. Stachniss, K. Konolige, et W. Burgard, "Hierarchies of octrees for efficient 3D mapping," dans *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 4249–4255. doi : 10.1109/IROS.2011.6094571
- [90] S. Macenski, D. Tsai, et M. Feinberg, "Spatio-temporal voxel layer : A view on robot perception for the dynamic world," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, Mar. 2020. doi : 10.1177/1729881420910530
- [91] S. Macenski, F. Martín, R. White, et J. Ginés Clavero, "The marathon 2 : A navigation system," dans *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. doi : 10.1109/IROS45743.2020.9341207
- [92] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, et W. Woodall, "Robot operating system 2 : Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022. doi : 10.1126/scirobotics.abm6074
- [93] K. Museth, "VDB : High-resolution sparse volumes with dynamic topology," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–22, Jun. 2013. doi : 10.1145/2487228.2487235
- [94] R. Hagmanns, T. Emter, M. Grosse Besselmann, et J. Beyerer, "Efficient Global Occupancy Mapping for Mobile Robots using OpenVDB," dans *IEEE/RSJ IROS Workshop Agile Robotics : Perception, Learning, Planning, and Control*, Kyoto, Oct. 2022. doi : 10.24406/publica-1081
- [95] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, et W. Burgard, "OctoMap : An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Avr. 2013. doi : 10.1007/s10514-012-9321-0
- [96] G. Chen, W. Dong, P. Peng, J. Alonso-Mora, et X. Zhu, "Continuous Occupancy Mapping in Dynamic Environments Using Particles," *IEEE Transactions on Robotics*, vol. 40, pp. 64–84, 2024. doi : 10.1109/TRO.2023.3323841
- [97] H. Ren et A. H. Qureshi, "Robot Active Neural Sensing and Planning in Unknown Cluttered Environments," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2738–2750, Aoû. 2023. doi : 10.1109/TRO.2023.3262114
- [98] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, et A. M. Dollar, "The YCB object and Model set : Towards common benchmarks for manipulation research," dans *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517. doi : 10.1109/ICAR.2015.7251504
- [99] R. Border et J. D. Gammell, "The surface edge explorer (SEE) : A measurement-direct approach to next best view planning," *The International Journal of Robotics Research*, Fév. 2024. doi : 10.1177/02783649241230098

- [100] R. Monica et J. Aleotti, “Contour-based next-best view planning from point cloud segmentation of unknown objects,” *Autonomous Robots*, vol. 42, no. 2, pp. 443–458, Fév. 2018. doi : 10.1007/s10514-017-9618-0
- [101] G. Hardouin, F. Morbidi, J. Moras, J. Marzat, et E. M. Mouaddib, “Surface-driven Next-Best-View planning for exploration of large-scale 3D environments,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 501–15 507, Jan. 2020. doi : 10.1016/j.ifacol.2020.12.2376
- [102] H. Hu, S. Pan, L. Jin, M. Popović, et M. Bennewitz, “Active Implicit Reconstruction Using One-Shot View Planning,” dans *IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan : IEEE, Mai 2024, pp. 12 477–12 483. doi : 10.1109/ICRA57147.2024.10611542
- [103] A. Kaufman, D. Cohen, et R. Yagel, “Volume graphics,” *Computer*, vol. 26, no. 7, pp. 51–64, Jul. 1993. doi : 10.1109/MC.1993.274942
- [104] D. Cohen-Or et A. Kaufman, “Fundamentals of Surface Voxelization,” *Graphical Models and Image Processing*, vol. 57, no. 6, pp. 453–461, Nov. 1995. doi : 10.1006/gmip.1995.1039
- [105] D. Haumont et N. Warzée, “Complete Polygonal Scene Voxelization,” *Journal of Graphics Tools*, vol. 7, no. 3, pp. 27–41, Jan. 2002. doi : 10.1080/10867651.2002.10487563
- [106] M. Schwarz et H.-P. Seidel, “Fast parallel surface and solid voxelization on GPUs,” *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 179 :1–179 :10, Déc. 2010. doi : 10.1145/1882261.1866201
- [107] H. Wu, Z. Xu, C. Liu, A. Akbar, H. Yue, D. Zeng, et H. Yang, “LV-GCNN : A lossless voxelization integrated graph convolutional neural network for surface reconstruction from point clouds,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 103, p. 102504, Déc. 2021. doi : 10.1016/j.jag.2021.102504
- [108] Z. Su, P. S. Tan, et Y.-H. Wang, “DV-Det : Efficient 3D Point Cloud Object Detection with Dynamic Voxelization,” *arXiv :2107.12707 [cs]*, Jul. 2021. [En ligne]. Disponible : <http://arxiv.org/abs/2107.12707>
- [109] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, et V. Vasudevan, “End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds,” dans *Conference on Robot Learning*. PMLR, Mai 2020, pp. 923–932. [En ligne]. Disponible : <https://proceedings.mlr.press/v100/zhou20a.html>
- [110] D. Maturana et S. Scherer, “VoxNet : A 3D Convolutional Neural Network for real-time object recognition,” dans *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 922–928. doi : 10.1109/IROS.2015.7353481

- [111] T. Hinks, H. Carr, L. Truong-Hong, et D. F. Laefer, “Point Cloud Data Conversion into Solid Models via Point-Based Voxelization,” *Journal of Surveying Engineering*, vol. 139, no. 2, pp. 72–83, Mai 2013. doi : 10.1061/(ASCE)SU.1943-5428.0000097
- [112] L. Wang, Y. Xu, et Y. Li, “Aerial Lidar Point Cloud Voxelization with its 3D Ground Filtering Application,” *Photogrammetric Engineering & Remote Sensing*, vol. 83, no. 2, pp. 95–107, Fév. 2017. doi : 10.14358/PERS.83.2.95
- [113] F. Garcia et B. Ottersten, “CPU-Based Real-Time Surface and Solid Voxelization for Incomplete Point Cloud,” dans *22nd International Conference on Pattern Recognition*, Aoû. 2014, pp. 2757–2762. doi : 10.1109/ICPR.2014.475
- [114] J. Papon, A. Abramov, M. Schoeler, et F. Worgotter, “Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2027–2034. [En ligne]. Disponible : https://openaccess.thecvf.com/content_cvpr_2013/html/Papon_Voxel_Cloud_Connectivity_2013_CVPR_paper.html
- [115] X. Dong, Y. Xiao, Z. Chen, J. Yao, et X. Guo, “GPU-based supervoxel segmentation for 3D point clouds,” *Computer Aided Geometric Design*, vol. 93, p. 102080, Fév. 2022. doi : 10.1016/j.cagd.2022.102080
- [116] H. Guo, H. Zhu, S. Peng, Y. Wang, Y. Shen, R. Hu, et X. Zhou, “SAM-Guided Graph Cut for 3D Instance Segmentation,” dans *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, et G. Varol, Eds. Cham : Springer Nature Switzerland, 2025, pp. 234–251. doi : 10.1007/978-3-031-73195-2_14
- [117] Y. He, H. Yu, X. Liu, Z. Yang, W. Sun, S. Anwar, et A. Mian, “Deep learning based 3D segmentation in computer vision : A survey,” *Information Fusion*, vol. 115, p. 102722, Mar. 2025. doi : 10.1016/j.inffus.2024.102722
- [118] S. Neshev, K. Tonchev, A. Manolova, et V. Poulkov, “3D Scene Segmentation : A Comprehensive Survey and Open Problems,” *IEEE Access*, vol. 13, pp. 110 457–110 496, 2025. doi : 10.1109/ACCESS.2025.3583136
- [119] C. Xie, Y. Xiang, A. Mousavian, et D. Fox, “Unseen Object Instance Segmentation for Robotic Environments,” *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1343–1359, Oct. 2021. doi : 10.1109/TRO.2021.3060341
- [120] A. E. Lefohn, J. E. Cates, et R. T. Whitaker, “Interactive, GPU-Based Level Sets for 3D Segmentation,” dans *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003*, ser. Lecture Notes in Computer Science, R. E. Ellis et T. M. Peters, Eds. Berlin, Heidelberg : Springer, 2003, pp. 564–572. doi : 10.1007/978-3-540-39899-8_70
- [121] N. D. F. Campbell, G. Vogiatzis, C. Hernández, et R. Cipolla, “Automatic 3D object segmentation in multiple views using volumetric graph-cuts,” *Image and Vision Computing*, vol. 28, no. 1, pp. 14–25, Jan. 2010. doi : 10.1016/j.imavis.2008.09.005

- [122] Z. Jiang, Y. Xue, Y. Zhao, X. Huang, et H. Li, “Multi-scale deep learning and clustering-based tabletop object instance segmentation for robot manipulation,” *International Journal of Advanced Robotic Systems*, vol. 21, no. 5, Nov. 2024. doi : 10.1177/17298806241278165
- [123] J. W. H. Tangelder et R. C. Velkamp, “A survey of content based 3D shape retrieval methods,” *Multimedia Tools and Applications*, vol. 39, no. 3, p. 441, Déc. 2007. doi : 10.1007/s11042-007-0181-0
- [124] P. Cignoni, C. Rocchini, et R. Scopigno, “Metro : Measuring error on simplified surfaces,” *The Eurographics Association*, p. 8, 1998.
- [125] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, et G. Ranzuglia, “MeshLab : An Open-Source Mesh Processing Tool,” *Eurographics Italian Chapter Conference*, p. 8 pages, 2008. doi : 10.2312/LOCALCHAPTEREVENTS/ITALCHAP/ITALIANCHAPCONF2008/129-136
- [126] M. Novotni et R. Klein, “A geometric approach to 3D object comparison,” dans *Proceedings International Conference on Shape Modeling and Applications*, Mai 2001, pp. 167–175. doi : 10.1109/SMA.2001.923387
- [127] D. Breen et R. Whitaker, “A level-set approach for the metamorphosis of solid models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 173–192, Avr. 2001. doi : 10.1109/2945.928169
- [128] E. Bribiesca, “Measuring 3-D shape similarity using progressive transformations,” *Pattern Recognition*, vol. 29, no. 7, pp. 1117–1129, Jul. 1996. doi : 10.1016/0031-3203(95)00150-6
- [129] H. Sánchez-Cruz et E. Bribiesca, “A method of optimum transformation of 3D objects used as a measure of shape dissimilarity,” *Image and Vision Computing*, vol. 21, no. 12, pp. 1027–1036, Nov. 2003. doi : 10.1016/S0262-8856(03)00119-7
- [130] H. Sánchez-Cruz et R. M. Rodríguez Dagnino, “Normalization of a 3D-Shape Similarity Measure with Voxel Representation,” *Computación y Sistemas*, vol. 10, no. 4, pp. 372–387, Jun. 2007. [En ligne]. Disponible : http://www.scielo.org.mx/scielo.php?script=sci_abstract&pid=S1405-55462007000200005&lng=es&nrm=iso&tlng=en
- [131] A. Arora, H. Fiorino, D. Pellier, M. Métivier, et S. Pesty, “A review of learning planning action models,” *The Knowledge Engineering Review*, vol. 33, 2018. doi : 10.1017/S0269888918000188
- [132] D. Paulius, K. S. P. Dong, et Y. Sun, “Task Planning with a Weighted Functional Object-Oriented Network,” dans *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi’an, China : IEEE, Mai 2021, pp. 3904–3910. doi : 10.1109/ICRA48506.2021.9561680

- [133] Z. Wang, C. R. Garrett, L. P. Kaelbling, et T. Lozano-Pérez, “Learning compositional models of robot skills for task and motion planning,” *The International Journal of Robotics Research*, Avr. 2021. doi : 10.1177/02783649211004615
- [134] M. Mansouri, F. Pecora, et P. Schüller, “Combining Task and Motion Planning : Challenges and Guidelines,” *Frontiers in Robotics and AI*, vol. 8, 2021. doi : 10.3389/frobt.2021.637888
- [135] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, et T. Lozano-Pérez, “Integrated task and motion planning,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021. doi : 10.1146/annurev-control-091420-084139
- [136] L. P. Kaelbling et T. Lozano-Pérez, “Integrated task and motion planning in belief space,” *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, Aoû. 2013. doi : 10.1177/0278364913484072
- [137] A. Adu-Bredu, N. Devraj, P.-H. Lin, Z. Zeng, et O. C. Jenkins, “Probabilistic Inference in Planning for Partially Observable Long Horizon Problems,” dans *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 3154–3161. doi : 10.1109/IROS51168.2021.9636685
- [138] B. Kim, L. Shimanuki, L. P. Kaelbling, et T. Lozano-Pérez, “Representation, learning, and planning algorithms for geometric task and motion planning,” *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 210–231, Sep. 2021. doi : 10.1177/02783649211038280
- [139] A. H. Qureshi, Y. Miao, A. Simeonov, et M. C. Yip, “Motion Planning Networks : Bridging the Gap Between Learning-Based and Classical Motion Planners,” *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, Fév. 2021. doi : 10.1109/TRO.2020.3006716
- [140] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, et D. Kragic, “Enabling Visual Action Planning for Object Manipulation Through Latent Space Roadmap,” *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 57–75, Fév. 2023. doi : 10.1109/TRO.2022.3188163
- [141] A. Akbari, F. Lagriffoul, et J. Rosell, “Combined heuristic task and motion planning for bi-manual robots,” *Autonomous Robots*, vol. 43, no. 6, pp. 1575–1590, Aoû. 2019. doi : 10.1007/s10514-018-9817-3
- [142] S. M. LaValle et J. J. Kuffner, “Randomized Kinodynamic Planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, Mai 2001. doi : 10.1177/02783640122067453
- [143] R. Shome, K. Solovey, A. Dobson, D. Halperin, et K. E. Bekris, “dRRT* : Scalable and informed asymptotically-optimal multi-robot motion planning,” *Autonomous Robots*, vol. 44, no. 3, pp. 443–467, Mar. 2020. doi : 10.1007/s10514-019-09832-9

- [144] R. Shome et K. E. Bekris, “Anytime Multi-arm Task and Motion Planning for Pick-and-Place of Individual Objects via Handoffs,” dans *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Aoû. 2019, pp. 37–43. doi : 10.1109/MRS.2019.8901083
- [145] K. Solovey, O. Salzman, et D. Halperin, “Finding a needle in an exponential haystack : Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning,” *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, Avr. 2016. doi : 10.1177/0278364915615688
- [146] L. Aiswarya et A. R. Chowdhury, “Human Aware Robot Motion Planning using RRT Algorithm in Industry4.0 Environment,” dans *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*, Mar. 2021, pp. 351–358. doi : 10.1109/ISR50024.2021.9419511
- [147] M. Otte et E. Frazzoli, “RRTX : Asymptotically optimal single-query sampling-based motion planning with quick replanning,” *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, Jun. 2016. doi : 10.1177/0278364915594679
- [148] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, et S. Teller, “Anytime Motion Planning using the RRT*,” dans *2011 IEEE International Conference on Robotics and Automation*, Mai 2011, pp. 1478–1483. doi : 10.1109/ICRA.2011.5980479
- [149] C. Li, C. Wang, J. Wang, Y. Shen, et M. Q.-H. Meng, “Sliding-Window Informed RRT* : A Method for Speeding Up the Optimization and Path Smoothing,” dans *IEEE International Conference on Real-time Computing and Robotics*, Jul. 2021, pp. 141–146. doi : 10.1109/RCAR52367.2021.9517672
- [150] I. Dovgopolik, K. Artemov, S. Zabihiyar, A. Semochkin, et S. Kolyubin, “Fast and Memory-Efficient Planning in C-space : Modified Bi-directional RRT* Algorithm for Humanoid Robots,” dans *2021 International Conference "Nonlinearity, Information and Robotics" (NIR)*, Aoû. 2021, pp. 1–6. doi : 10.1109/NIR52917.2021.9666083
- [151] A. Dobson, K. Solovey, R. Shome, D. Halperin, et K. E. Bekris, “Scalable asymptotically-optimal multi-robot motion planning,” dans *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Déc. 2017, pp. 120–127. doi : 10.1109/MRS.2017.8250940
- [152] L. Wang, H. Ye, Q. Wang, Y. Gao, C. Xu, et F. Gao, “Learning-based 3D Occupancy Prediction for Autonomous Navigation in Occluded Environments,” dans *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 4509–4516. doi : 10.1109/IROS51168.2021.9636333
- [153] B. Lindqvist, A.-A. Agha-Mohammadi, et G. Nikolakopoulos, “Exploration-RRT : A multi-objective Path Planning and Exploration Framework for Unknown and Unstructured Environments,” dans *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 3429–3435. doi : 10.1109/IROS51168.2021.9636243

- [154] D. McConachie, M. Ruan, et D. Berenson, “Interleaving Planning and Control for Deformable Object Manipulation,” dans *Robotics Research*, ser. Springer Proceedings in Advanced Robotics, N. M. Amato, G. Hager, S. Thomas, et M. Torres-Torriti, Eds. Cham : Springer International Publishing, 2020, pp. 1019–1036. doi : 10.1007/978-3-030-28619-4_68
- [155] D. McConachie et D. Berenson, “Bandit-Based Model Selection for Deformable Object Manipulation,” dans *Algorithmic Foundations of Robotics XII : Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, ser. Springer Proceedings in Advanced Robotics, K. Goldberg, P. Abbeel, K. Bekris, et L. Miller, Eds. Cham : Springer International Publishing, 2020, pp. 704–719. doi : 10.1007/978-3-030-43089-4_45
- [156] D. Mcconachie et D. Berenson, “Estimating Model Utility for Deformable Object Manipulation Using Multiarmed Bandit Methods,” *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 967–979, Jul. 2018. doi : 10.1109/TASE.2018.2822669
- [157] L. F. P. Oliveira, A. P. Moreira, et M. F. Silva, “Advances in Agriculture Robotics : A State-of-the-Art Review and Challenges Ahead,” *Robotics*, vol. 10, no. 2, p. 52, Jun. 2021. doi : 10.3390/robotics10020052
- [158] J. Jun, J. Seol, et H. I. Son, “A Novel End-effector for Tomato Harvesting Robot : Mechanism and Evaluation,” dans *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, Oct. 2020, pp. 118–121. doi : 10.23919/ICCAS50221.2020.9268439
- [159] L. Cao et S. Miao, “Design of Chinese Cabbage Harvester,” dans *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, Oct. 2020, pp. 243–248. doi : 10.1109/ICMA49215.2020.9233714
- [160] J. L. Louedec, H. A. Montes, T. Duckett, et G. Cielniak, “Segmentation and detection from organised 3D point clouds : A case study in broccoli head detection,” dans *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2020, pp. 285–293. doi : 10.1109/CVPRW50498.2020.00040
- [161] Y. Qian, R. Jiacheng, W. Pengbo, Y. Zhan, et G. Changxing, “Real-time detection and localization using SSD method for oyster mushroom picking robot*,” dans *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Sep. 2020, pp. 158–163. doi : 10.1109/RCAR49640.2020.9303258
- [162] E. Navas, R. Fernández, D. Sepúlveda, M. Armada, et P. Gonzalez-de-Santos, “Soft Gripper for Robotic Harvesting in Precision Agriculture Applications,” dans *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Avr. 2021, pp. 167–172. doi : 10.1109/ICARSC52212.2021.9429797

- [163] F. Yandun, A. Silwal, et G. Kantor, “Visual 3D Reconstruction and Dynamic Simulation of Fruit Trees for Robotic Manipulation,” dans *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2020, pp. 238–247. doi : 10.1109/CVPRW50498.2020.00035
- [164] Z. Tang, L. Xu, et H. Xie, “Picking Trajectory Planning of Citrus Based on Improved Immune Algorithm and Binocular Vision,” dans *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Jun. 2020, pp. 6–10. doi : 10.1109/ICAICA50127.2020.9182606
- [165] Sayyida. Beegam. K et Meetha. V. Shenoy, “Cherry Plucking Strategies for Coffee Harvester,” dans *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, Avr. 2021, pp. 151–155. doi : 10.1109/ICCAR52225.2021.9463497
- [166] Y. Xiong, Y. Ge, et P. J. From, “Push and Drag : An Active Obstacle Separation Method for Fruit Harvesting Robots,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 4957–4962. doi : 10.1109/ICRA40945.2020.9197469
- [167] S. Mghames, M. Hanheide, et A. M. G. Esfahani, “Interactive Movement Primitives : Planning to Push Occluding Pieces for Fruit Picking,” dans *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA (Virtual) : IEEE, 2020, p. 8. doi : 10.1109/IROS45743.2020.9341728
- [168] A. You, F. Sukkar, R. Fitch, M. Karkee, et J. R. Davidson, “An Efficient Planning and Control Framework for Pruning Fruit Trees,” dans *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Mai 2020, pp. 3930–3936. doi : 10.1109/ICRA40945.2020.9197551
- [169] V. Tinoco, M. F. Silva, F. N. Santos, L. F. Rocha, S. Magalhães, et L. C. Santos, “A Review of Pruning and Harvesting Manipulators,” dans *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Avr. 2021, pp. 155–160. doi : 10.1109/ICARSC52212.2021.9429806
- [170] S. Chen, Y. C. Chiu, et Y. C. Chang, “Development of a Tubing-grafting Robotic System for Fruit-bearing Vegetable Seedlings,” *Applied Engineering in Agriculture*, vol. 26, no. 4, pp. 707–714, 2010. doi : 10.13031/2013.32055
- [171] E. Misimi, E. R. Øye, A. Eilertsen, J. R. Mathiassen, O. B. Åsebø, T. Gjerstad, J. Buljo, et Ø. Skotheim, “GRIBBOT – Robotic 3D vision-guided harvesting of chicken fillets,” *Computers and Electronics in Agriculture*, vol. 121, pp. 84–100, Fév. 2016. doi : 10.1016/j.compag.2015.11.021
- [172] N. Essahbi, B.-C. Bouzgarrou, et G. Gogu, “Soft material modeling for robotic task formulation and control in the muscle separation process,” *Robotics and Computer-Integrated Manufacturing*, vol. 32, pp. 37–53, Avr. 2015. doi : 10.1016/j.rcim.2014.09.003

- [173] A. Mason, I. de Medeiros Esper, O. Korostynska, L. E. Cordova-Lopez, D. Romanov, M. Pinceková, P. H. Bjørnstad, O. Alvseike, A. Popov, O. Smolkin, M. Manko, L. B. Christensen, K. Takács, et T. Haidegger, “RoBUTCHER : A novel robotic meat factory cell platform,” *The International Journal of Robotics Research*, Fév. 2024. doi : 10.1177/02783649241234035
- [174] L. Cheng et M. Tavakoli, “COVID-19 Pandemic Spurs Medical Telerobotic Systems : A Survey of Applications Requiring Physiological Organ Motion Compensation,” *Frontiers in Robotics and AI*, vol. 7, 2020. doi : 10.3389/frobt.2020.594673
- [175] S. Al-Hussaini, J. M. Gregory, N. Dhanaraj, et S. K. Gupta, “A Simulation-Based Framework for Generating Alerts for Human-Supervised Multi-Robot Teams in Challenging Environments,” dans *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct. 2021, pp. 168–175. doi : 10.1109/SSRR53300.2021.9597684
- [176] H. Peng, M. Wu, H. Lu, J. Xiao, C. Cheng, et J. Li, “Stability Control for Medical Rescue Robot in Unconstructed Environment,” dans *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, Déc. 2020, pp. 184–188. doi : 10.1109/ICARM49381.2020.9195272
- [177] Y. Xu, H. Zhao, X. Zhou, Bhushan, S. E. Ovrur, et T. Xia, “Energy Management for Medical Rescue Robot,” dans *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, Déc. 2020, pp. 44–49. doi : 10.1109/ICARM49381.2020.9195370
- [178] A. Mandow, J. Serón, F. Pastor, et A. García-Cerezo, “Experimental Validation of a Robotic Stretcher for Casualty Evacuation in a Man-Made Disaster Exercise,” dans *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2020, pp. 241–245. doi : 10.1109/SSRR50563.2020.9292633
- [179] X. Lin, S. Mahmud, S. Roman, A. Shaker, Z. Law, M. Lin, et J.-H. Kim, “Design of A Novel Transformable Centaur Robot with Multilateral Control Interface for Search and Rescue Missions,” dans *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Sep. 2020, pp. 200–205. doi : 10.1109/CACRE50138.2020.9230311
- [180] W. Yuan, K. Hang, H. Song, D. Kragic, M. Y. Wang, et J. A. Stork, “Reinforcement Learning in Topology-based Representation for Human Body Movement with Whole Arm Manipulation,” dans *2019 International Conference on Robotics and Automation (ICRA)*, Mai 2019, pp. 2153–2160. doi : 10.1109/ICRA.2019.8794160
- [181] P. Gliesche, C. Kowalski, M. Pfingsthorn, et A. Hein, “Towards Whole Arm Manipulation for Outpatient Care,” dans *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IRO, Las Vegas, NV, USA (Virtual)*, Oct. 2020, p. 5. [En ligne]. Disponible : <https://ras.papercept.net/proceedings/IROS20/3625.pdf>

- [182] E. Peiros, Z.-Y. Chiu, Y. Zhi, N. Shinde, et M. C. Yip, “Finding Biomechanically Safe Trajectories for Robot Manipulation of the Human Body in a Search and Rescue Scenario,” dans *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 167–173. doi : 10.1109/IROS55552.2023.10342353
- [183] Z. Huang, J. Li, J. Huang, J. Ota, et Y. Zhang, “Motion Planning for Bandaging Task With Abnormal Posture Detection and Avoidance,” *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 5, pp. 2364–2375, Oct. 2020. doi : 10.1109/TMECH.2020.2973674
- [184] J. Li, X. Gu, Q. Liu, H. Sun, W. Cen, Z. Huang, Y. Zhang, et Z. Huang, “Force Feedback Algorithm for Robot Arm to Bandage A Swaying Arm,” dans *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, Déc. 2020, pp. 38–43. doi : 10.1109/ICARM49381.2020.9195393
- [185] L. Wang et J. Zhu, “Deformable Object Manipulation in Caregiving Scenarios : A Review,” *Machines*, vol. 11, no. 11, p. 1013, Nov. 2023. doi : 10.3390/machines11111013
- [186] P. Jiménez, “Survey on model-based manipulation planning of deformable objects,” *Robotics and Computer Integrated Manufacturing*, vol. 28, no. 2, pp. 154–163, 2012. doi : 10/bzdc7
- [187] J. Moser, J. Hoffman, R. Hildebrand, et E. Komendera, “An Autonomous Task Assignment Paradigm for Autonomous Robotic In-Space Assembly,” *Frontiers in Robotics and AI*, vol. 9, 2022. doi : 10.3389/frobt.2022.709905
- [188] P. Kazanzides, B. P. Vagvolgyi, W. Pryor, A. Deguet, S. Leonard, et L. L. Whitcomb, “Teleoperation and Visualization Interfaces for Remote Intervention in Space,” *Frontiers in Robotics and AI*, vol. 8, p. 372, 2021. doi : 10.3389/frobt.2021.747917
- [189] E. Papadopoulos, F. Aghili, O. Ma, et R. Lampariello, “Robotic Manipulation and Capture in Space : A Survey,” *Frontiers in Robotics and AI*, vol. 0, 2021. doi : 10.3389/frobt.2021.686723
- [190] F. Nadon, A. J. Valencia, N. Sambandam, S. Rowlands, J. Dickens, et P. Payeur, “Toward a General Framework for 3D Deformable Object Grasping and Manipulation,” dans *Workshop on Robotic Manipulation of Deformable Objects (ROMADO), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, p. 4. [En ligne]. Disponible : <http://commandia.unizar.es/irosworkshop2020/>
- [191] F. Nadon, A. J. Valencia, et P. Payeur, “Robotic System Architecture Design for Manipulation of 3D Deformable Objects,” dans *IEEE/SICE International Symposium on System Integration (SII)*. Cancún, México : IEEE, Jan. 2026, pp. 813–818. doi : 10.1109/SII64115.2026.11404564
- [192] PickNick Robotics, “MoveIt Motion Planning Framework,” 2024. [En ligne]. Disponible : <https://moveit.ai/>

- [193] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, et A. Garg, “ORBIT : A Unified Simulation Framework for Interactive Robot Learning Environments,” *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, Jun. 2023. doi : 10.1109/LRA.2023.3270034
- [194] NVIDIA, “Isaac Sim - Robotics Simulation and Synthetic Data Generation,” 2024. [En ligne]. Disponible : <https://developer.nvidia.com/isaac-sim>
- [195] F. Nadon et P. Payeur, “A Fast Next-Best-View Approach for Voxelized Object Shape Estimation and Completion,” dans *International Instrumentation and Measurement Technology Conference (I2MTC)*. Nancy, France : IEEE, Mai 2026.
- [196] F. Nadon et P. Payeur, “3D Shape Comparison and Matching Methods for Reshaping Deformable Objects,” dans *International Instrumentation and Measurement Technology Conference (I2MTC)*. Nancy, France : IEEE, Mai 2026.
- [197] D. Berenson, “Manipulation of deformable objects without modeling and simulating deformation,” dans *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 4525–4532. doi : 10.1109/IROS.2013.6697007
- [198] F. Nadon et P. Payeur, “Automatic Selection of Grasping Points for Shape Control of Non-Rigid Objects,” dans *Proceedings of the IEEE International Symposium on Robot and Sensors Environments (ROSE 2019)*, Ottawa, ON, Jun. 2019, pp. 170–176. doi : 10.1109/ROSE.2019.8790383
- [199] “CloudCompare v2.6.1 - User manual.pdf.” [En ligne]. Disponible : <https://www.danielgm.net/cc/doc/qCC/CloudCompare%20v2.6.1%20-%20User%20manual.pdf>
- [200] D. Navarro-Alarcon et Y. H. Liu, “Fourier-Based Shape Servoing : A New Feedback Method to Actively Deform Soft Objects into Desired 2-D Image Contours,” *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 272–279, Fév. 2018. doi : 10/gc2t9t
- [201] S. Pan, L. Jin, H. Hu, M. Popović, et M. Bennewitz, “How Many Views Are Needed to Reconstruct an Unknown Object Using NeRF?” dans *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan : IEEE, Mai 2024, pp. 12 470–12 476. doi : 10.1109/ICRA57147.2024.10610617
- [202] D. F. Rogers, *Procedural Elements for Computer Graphics*. New York : McGraw-Hill, 1985.
- [203] P. Payeur, P. Hebert, D. Laurendeau, et C. Gosselin, “Probabilistic octree modeling of a 3D dynamic environment,” dans *Proceedings of International Conference on Robotics and Automation*, vol. 2, Avr. 1997, pp. 1289–1296 vol.2. doi : 10.1109/ROBOT.1997.614315

- [204] P. Payeur, D. Laurendeau, et C. Gosselin, “Range data merging for probabilistic octree modeling of 3D workspaces,” dans *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, Mai 1998, pp. 3071–3078 vol.4. doi : 10.1109/ROBOT.1998.680897
- [205] Y. Shi et W. C. Karl, “A Real-Time Algorithm for the Approximation of Level-Set-Based Curve Evolution,” *IEEE Transactions on Image Processing*, vol. 17, no. 5, pp. 645–656, Mai 2008. doi : 10.1109/TIP.2008.920737
- [206] X. Lai, D. Yue, J.-K. Hao, F. Glover, et Z. Lü, “Iterated dynamic neighborhood search for packing equal circles on a sphere,” *Computers & Operations Research*, vol. 151, p. 106121, Mar. 2023. doi : 10.1016/j.cor.2022.106121
- [207] A. Morales, G. Recatalá, P. J. Sanz, et A. P. Del Pobil, “Heuristic vision-based computation of planar antipodal grasps on unknown objects,” dans *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, 2001, pp. 583–588. doi : 10.1109/ROBOT.2001.932613
- [208] Academy Software Foundation, “OpenVDB : OpenVDB Overview.” [En ligne]. Disponible : <https://academysoftwarefoundation.github.io/opensvdb/overview.html#secTheGrid>

Annexe A

Moments 3D

Cette annexe reporte les équations pour le calcul des moments 3D et l'extraction des propriétés de centroïde et d'orientation dans la notation discrète communément utilisée pour les images, plus appropriée aux environnements voxelisés que la notation continue habituellement utilisée en 3D.

Dans un environnement tridimensionnel discret, le moment m_{ijk} est calculé par l'équation (A.1), où $V(x, y, z)$ est la valeur attribuée au voxel situé à la position (x, y, z) (typiquement 0 ou 1 pour simplifier l'interprétation des résultats) et où les sommes peuvent être définies sur la boîte englobante de l'objet. Notons que cette valeur ne correspond pas nécessairement à $\Lambda(x, y, z)$.

$$m_{ijk} = \sum_x \sum_y \sum_z x^i y^j z^k V(x, y, z) \quad (\text{A.1})$$

Ainsi, m_{000} correspond à la somme des valeurs des voxels, ce qui peut être interprété comme le volume (pour des voxels binaires) ou la « masse » de l'objet. Similairement au cas 2D, le centre de masse de l'objet, $c = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix}$ est obtenu par

$$\begin{aligned} \bar{x} &= \frac{m_{100}}{m_{000}} \\ \bar{y} &= \frac{m_{010}}{m_{000}} \\ \bar{z} &= \frac{m_{001}}{m_{000}} \end{aligned} \quad (\text{A.2})$$

Pour leur part, les axes principaux de l'objet correspondent aux vecteurs propres¹ de la matrice de covariance des moments centraux de deuxième ordre :

$$Cov = \begin{bmatrix} u_{200} & u_{110} & u_{101} \\ u_{110} & u_{020} & u_{011} \\ u_{101} & u_{011} & u_{002} \end{bmatrix} \quad (A.3)$$

$$\begin{aligned} u'_{ijk} &= \frac{m_{ijk}}{m_{000}} \\ u_{200} &= u'_{200} - \bar{x}^2 \\ u_{020} &= u'_{020} - \bar{y}^2 \\ u_{002} &= u'_{002} - \bar{z}^2 \\ u_{110} &= u'_{110} - \bar{x}\bar{y} \\ u_{101} &= u'_{101} - \bar{x}\bar{z} \\ u_{011} &= u'_{011} - \bar{y}\bar{z} \end{aligned} \quad (A.4)$$

Ainsi, le vecteur propre $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ qui correspond à la plus grande valeur propre de $Cov(obj)$ est la direction à partir de son centre de masse vers laquelle l'objet est le plus allongé. De plus, le vecteur propre qui correspond à la plus petite valeur propre de $Cov(obj)$ est la direction vers laquelle l'objet est le moins allongé, et ainsi la direction perpendiculaire au plan le mieux ajusté à l'objet.

Cependant, on note que les axes obtenus de cette manière ne sont pas orientés, c'est-à-dire qu'il y a deux directions possibles pour chaque axe tel qu'illustré à la figure A.1. Cette ambiguïté doit être prise en compte dans les algorithmes faisant usage de cette information.

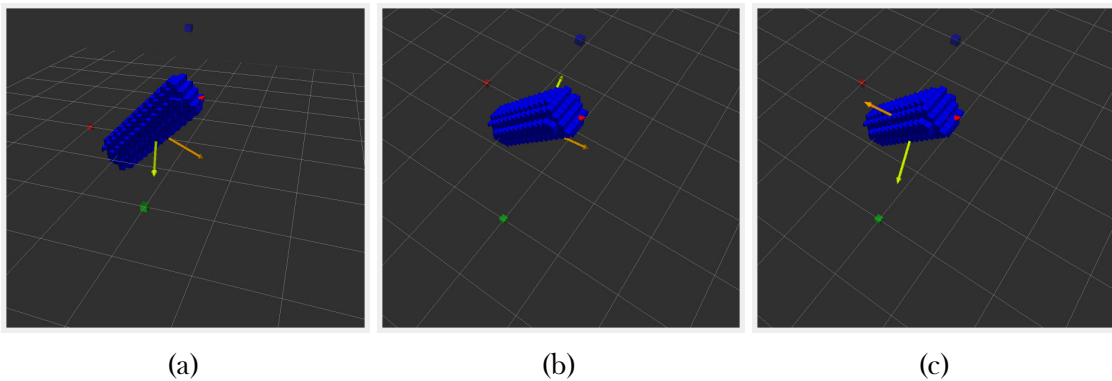


FIGURE A.1 : Exemples de direction des axes principaux, illustrés du plus au moins allongé par les flèches rouge, orange et jaune. Il y a deux directions possibles pour chaque axe pour un total de $2^3 = 8$ combinaisons. Les axes globaux sont montrés par les points rouge, vert et bleu.

1. *eigenvectors*

Annexe B

Résultats supplémentaires

B.1 Sélection du prochain point de vue

B.1.1 Observation de la sphère avec des voxels de 3 mm

Les résultats obtenus avec des voxels de 3 mm (figure B.1) sont semblables à ceux avec des voxels de 6 mm (figure 6.13). Les cas limitant le nombre d'observations à un seul groupe et au numéro d'itération présentent tous deux des pics liés à des erreurs de segmentation. Ces pics ralentissent le calcul des points d'observation potentiels en augmentant grandement le nombre de groupes et de voxels à considérer. Dans le cas de l'observation d'un seul groupe, on note que le deuxième pic n'est pas résolu à la prochaine observation. Ceci indique que le groupe observé à cette itération était situé loin des voxels causant le problème et n'a donc pas permis de les réobserver. Ceci est moins problématique avec la méthode limitant les observations au numéro d'itération puisque les quelques observations supplémentaires augmentent les chances de réobserver les voxels erronés.

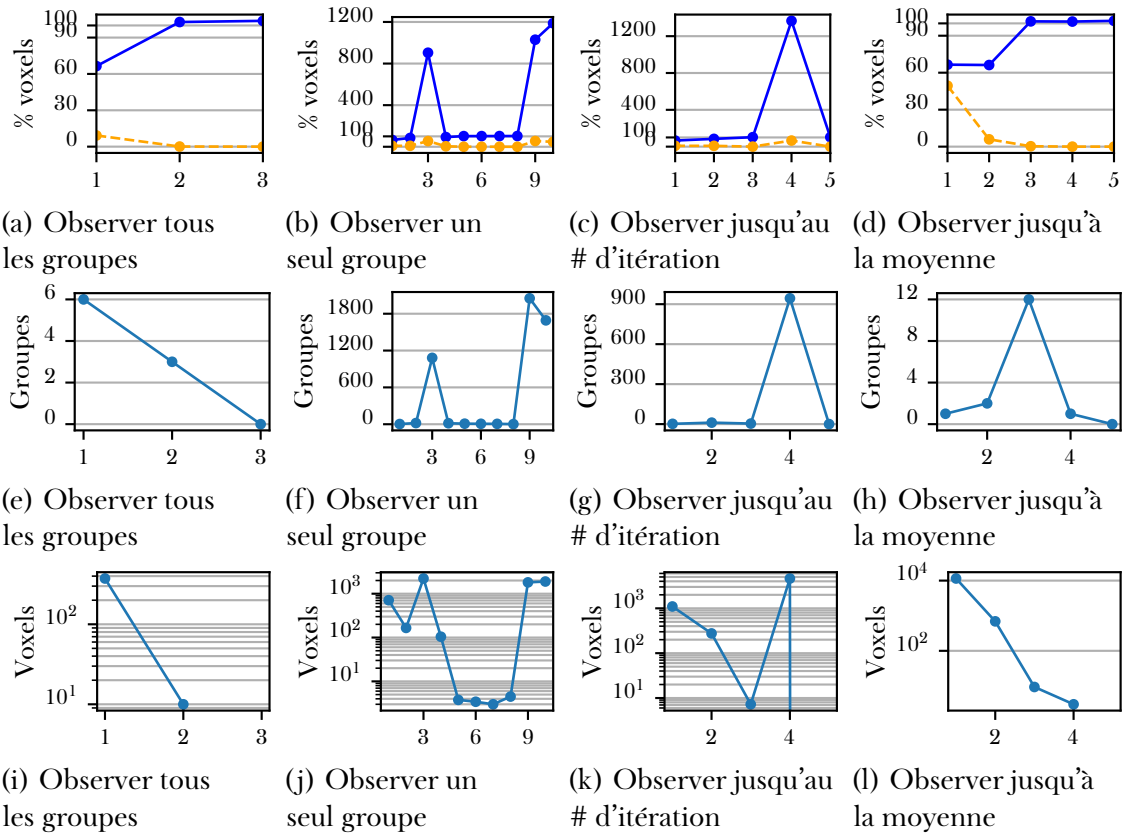


FIGURE B.1 : Progression du processus d'observation pour la sphère (voxels de 3 mm) pour différentes limites du nombre d'observations par calcul NBV.

B.1.2 Observation de la longue éponge

La figure B.2 présente le déroulement du processus d'observation pour la longue éponge avec des voxels de 8 mm. Dans ce cas, les quatre méthodes ont des résultats semblables, convergeant autour de 95% de voxels dans l'objet avec peu de voxels inconnus. Cependant, les dernières itérations ne présentent pas de progrès, ce qui indique que quelques voxels non-observables sont présents et ne peuvent pas être éliminés par le remplissage et la suppression des intersections.

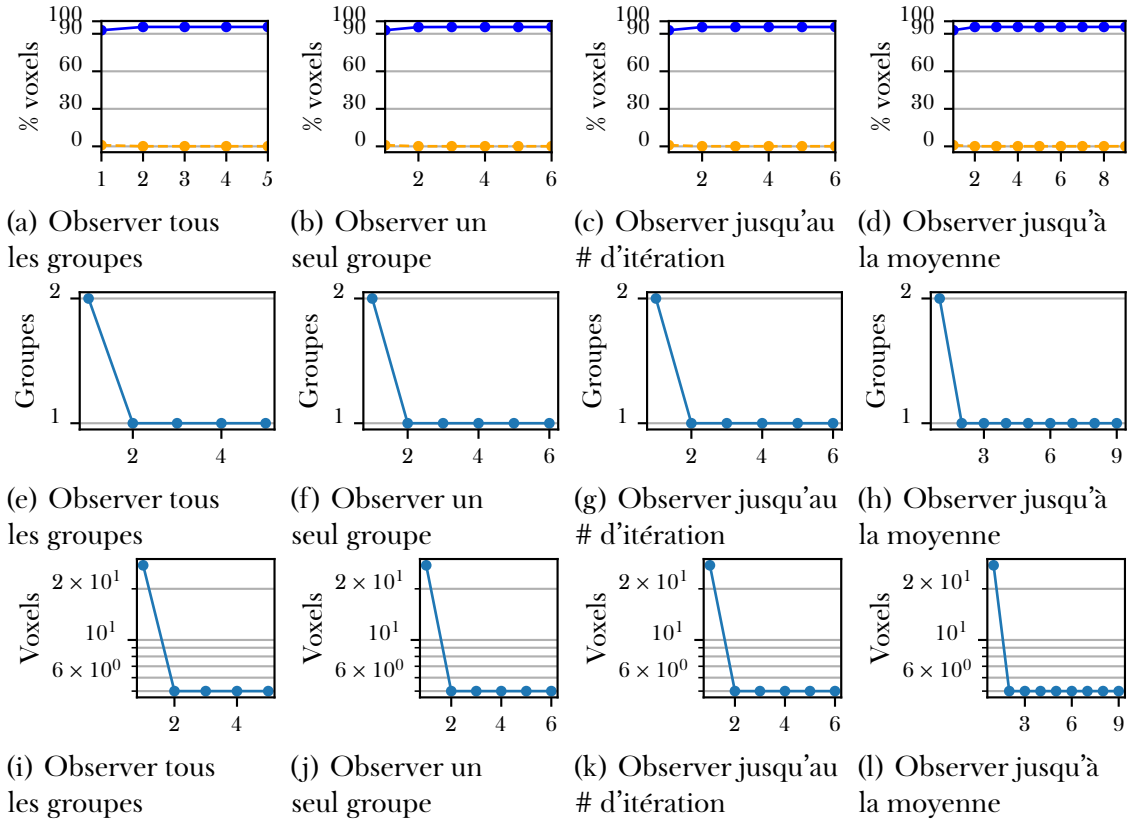


FIGURE B.2 : Progression du processus d'observation pour la longue éponge (voxels de 8 mm) pour différentes limites du nombre d'observations par calcul NBV.

B.1.3 Note sur la segmentation en simulation

Bien que les mesures en simulation éliminent l'erreur de position dans les nuages de points, elles contiennent tout de même des artefacts de couleur, montrés à la figure B.3. Ces artefacts prennent la forme d'un halo brun-rouge sur la surface de travail ainsi que de rayures sur l'objet où la couleur de l'objet est confondue avec celle de l'arrière plan, généralement aux

intersections des mesures provenant de différents points de vue. Lorsque la segmentation est effectuée dynamiquement avec une sélection itérative des points d’observation (plutôt que sur une grille de voxels fixes comme à la section 5.5.2), il arrive que le voxel utilisé comme référence pour la couleur de l’objet d’intérêt se retrouve dans l’un de ces artefacts. Cette situation, qui est plus fréquente à haute résolution, mène généralement à une erreur de segmentation où la grille complète est sélectionnée comme faisant partie de l’objet d’intérêt, mais qui est résolue lorsque les artefacts changent de position à la prochaine observation.

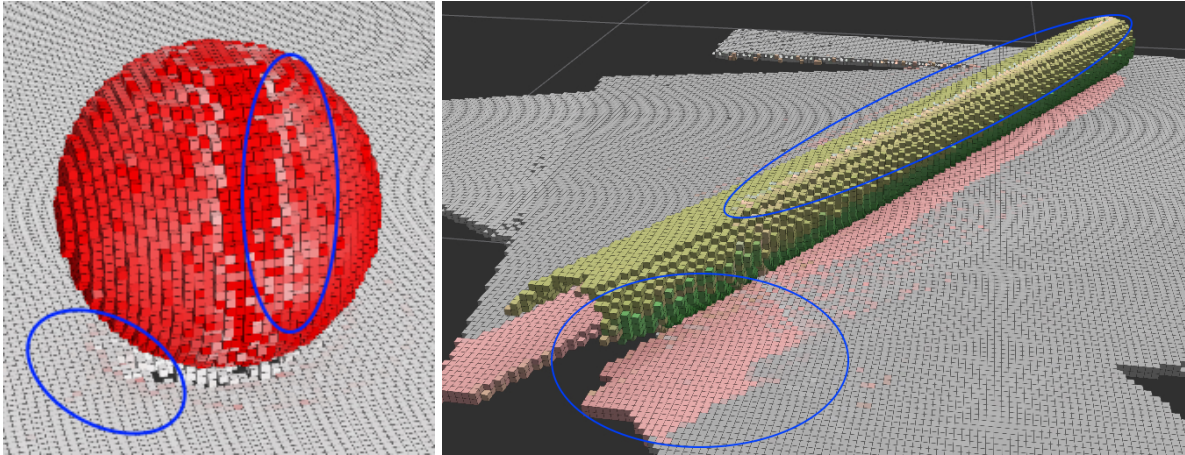


FIGURE B.3 : Exemples d’artefacts de couleur en simulation.

B.2 Exemples de séquences d’observation

La figure B.4 présente la séquence d’observations menant à l’obtention de la forme initiale complète pour la sphère de 10 cm de diamètre avec des voxels de 5 mm. Bien que le nombre d’observations par calcul des heuristiques NBV soit limité au numéro d’itération (indiqué « NBV » dans la figure), on constate que ce nombre n’est pas atteint, soit parce que plusieurs points de vue sont ignorés (p. ex., sous la surface de travail) ou parce que le nombre de groupes de voxels inconnus est plus petit que la limite. Dans cette séquence d’observations, on remarque que l’observation sous plusieurs angles est combinée à des observations légèrement décalées du même côté de l’objet (p. ex., B.4 d et e ou h et i). Ces observations visent à confirmer l’état d’occupation de petites sections qui n’ont pas été mesurées par les observations précédentes à cause de la résolution du nuage de points ou des erreurs d’arrondissement dans l’opération de reprojection.

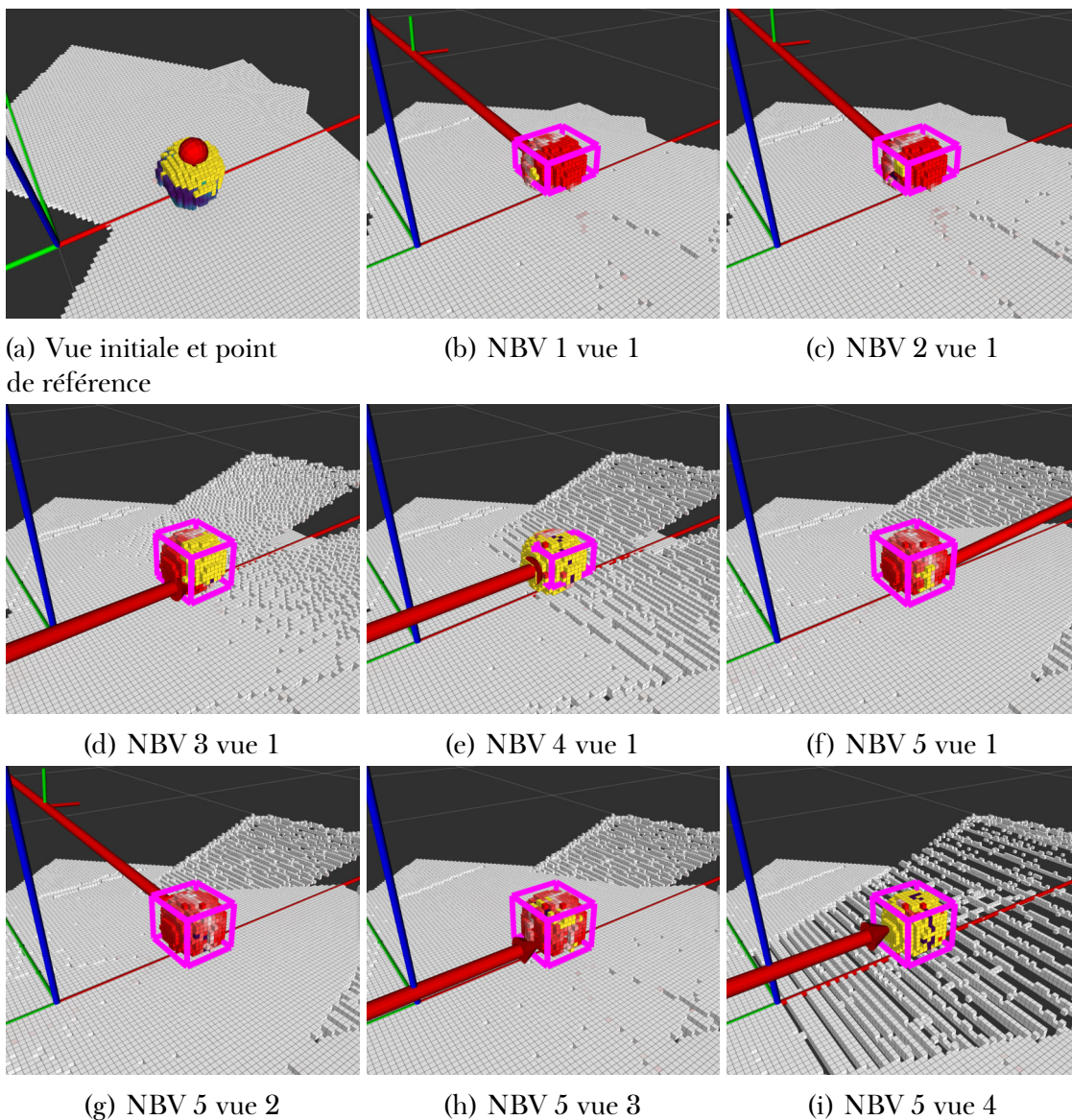


FIGURE B.4 : Séquence d'observations pour la sphère en simulation. La flèche rouge représente l'axe optique de la caméra au point de vue choisi et le cadre rose la boîte englobante de l'objet segmenté. Le masque segmenté (jaune et mauve) est superposé à la carte colorée (rouge). Les couleurs visibles peuvent donc varier selon l'ordre d'affichage des deux cartes.

Les figures B.5 à B.7 montrent une séquence de points de vue pour observer la forme initiale du cylindre de 2 m de long par 10 cm de diamètre avec des voxels de 16 mm. Dans cette séquence, on remarque que le critère utilisé pour détecter si l'objet est complet — c'est-à-dire une absence de voxels inconnus dans sa boîte englobante — permet d'observer correctement un objet qui est plus grand que le champ de vision de la caméra. En effet, la boîte englobante s'agrandit à mesure que des observations sont prises près des extrémités de l'objet (p. ex., B.5 c–e), de sorte que de nouvelles observations sont prises à mesure que les différentes parties de l'objet sont mesurées. Cependant, en raison de l'angle et des proportions de l'objet par rapport aux axes globaux, l'objet d'intérêt occupe seulement une petite portion de sa boîte englobante. Ceci mène à plusieurs points de vue qui observent des parties inconnues de la boîte englobante qui ne font pas partie de l'objet (p. ex. B.5h).

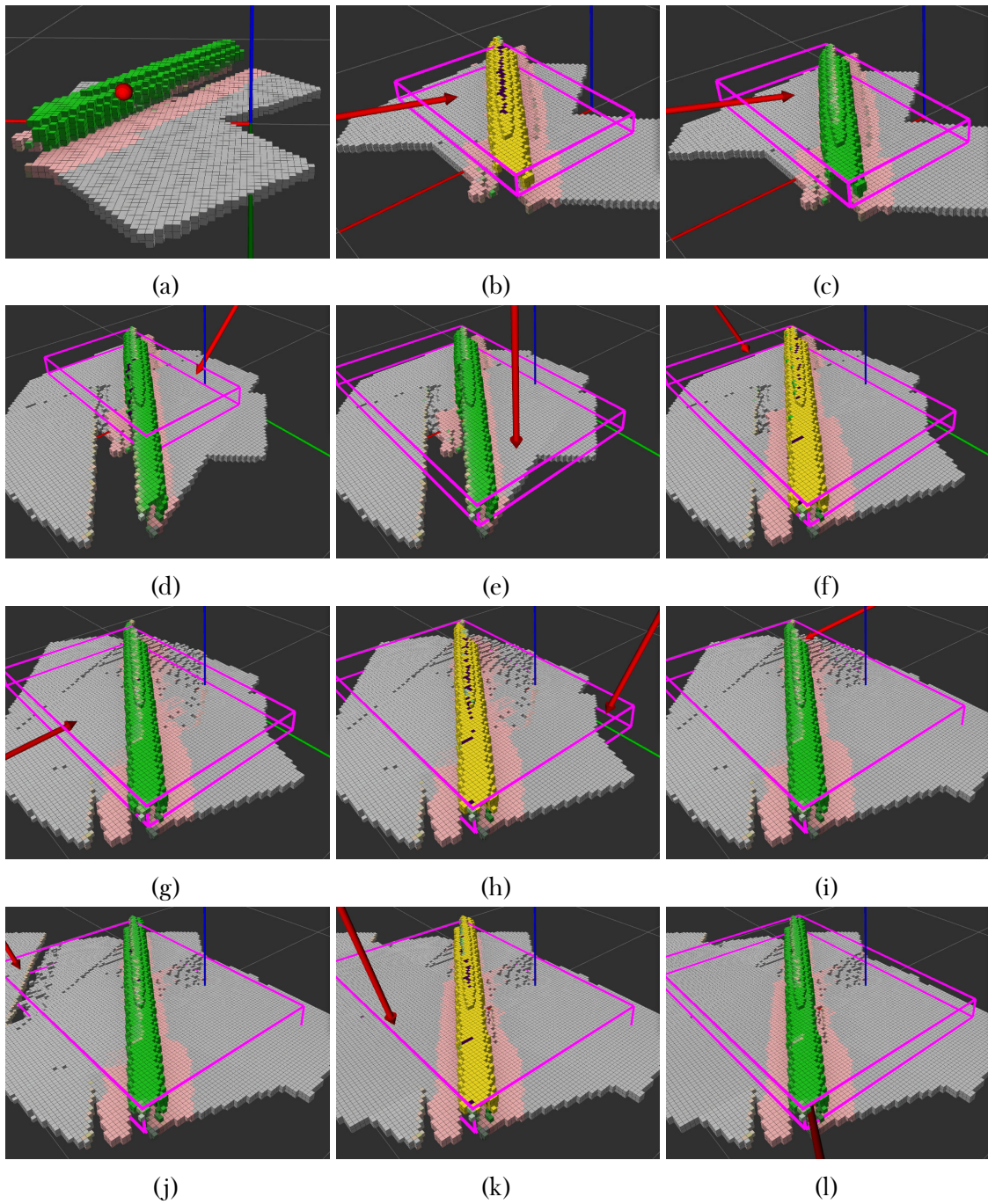


FIGURE B.5 : Séquence de points de vue (flèche rouge) pour observer le cylindre vert. Le masque d'objet segmenté (jaune et mauve) est superposé à la carte colorée.

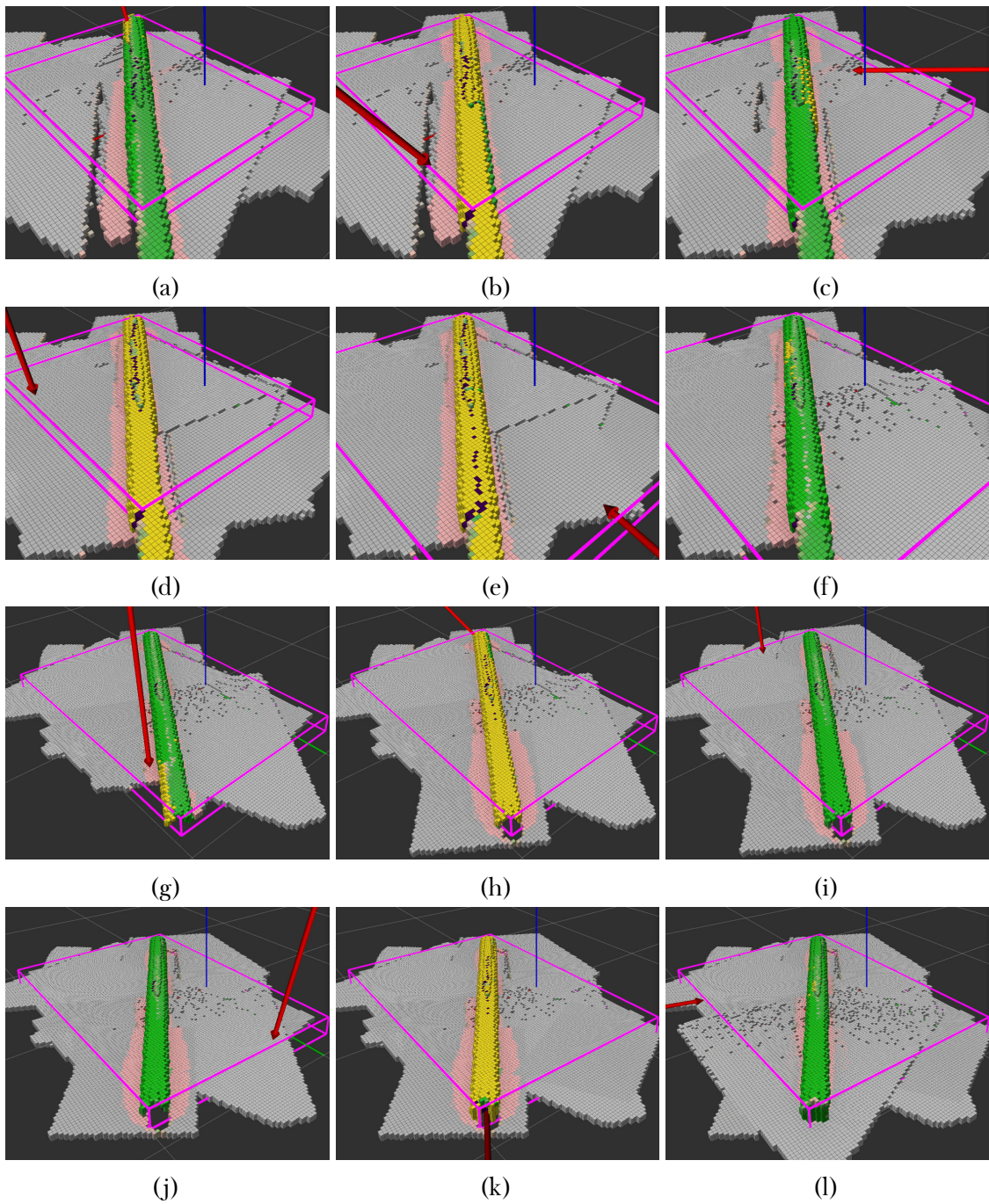


FIGURE B.6 : Séquence de points de vue pour observer le cylindre (suite)

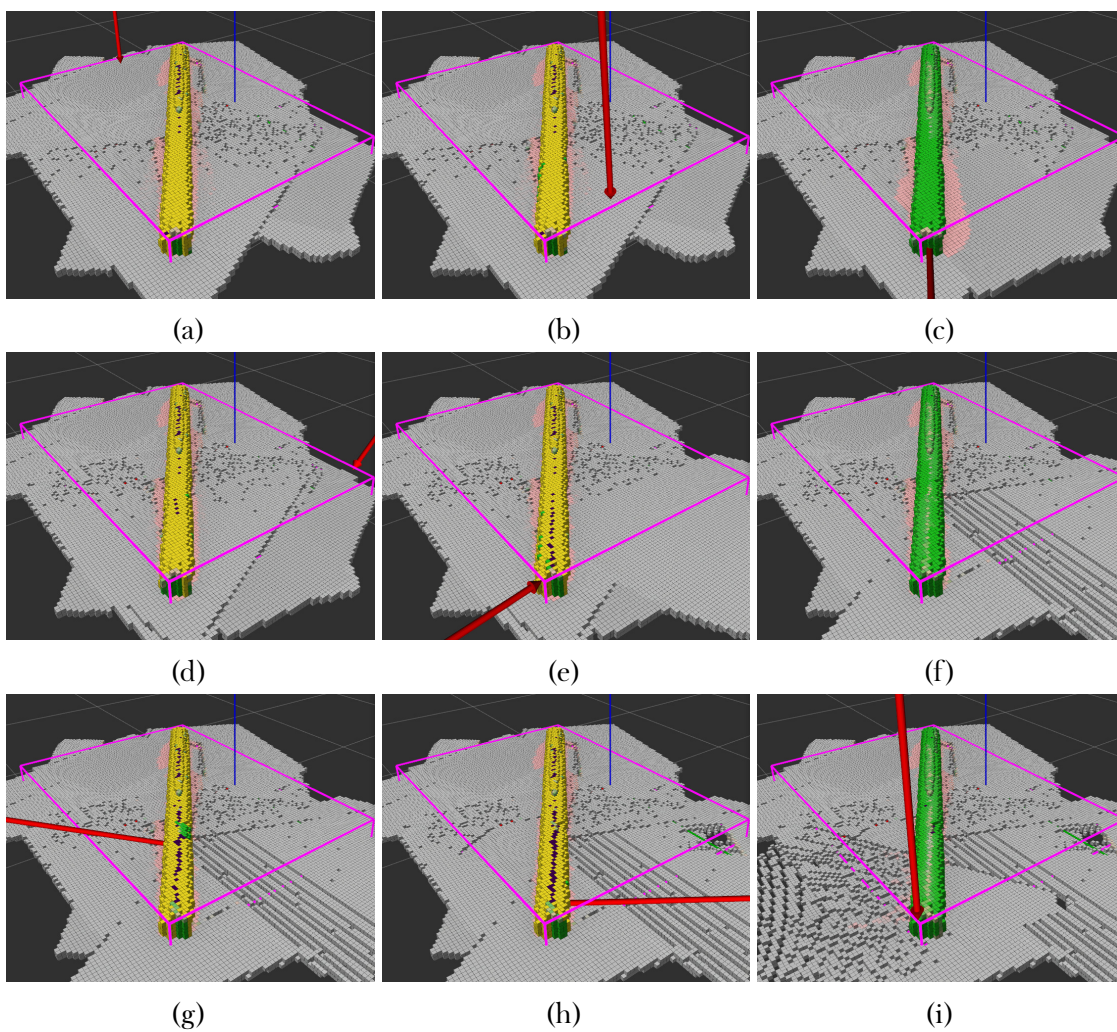


FIGURE B.7 : Séquence de points de vue pour observer le cylindre (suite)

B.3 Résultats globaux

B.3.1 Transformation d'une sphère en ovale

Un scénario additionnel considéré est la compression d'une sphère en une forme ovale. La figure B.8 montre d'abord le masque d'objet complété ainsi que les points de prise et l'angle d'observation pour le suivi de la manipulation. Ensuite, la figure B.9 montre le résultat de la manipulation en simulation ainsi que l'objet d'intérêt avant et après la réobservation. Enfin, la figure B.10 compare la forme cible avec les formes initiale et finale. Dans ces résultats, on remarque que la déformation complétée est relativement petite par rapport à la taille des voxels. Ainsi, bien que le résultat en simulation ait rapproché la sphère d'un ovale (B.9a), il est plus difficile d'évaluer visuellement la qualité du remodelage dans les cartes voxélisées (B.9 b–e). La forme résultant du suivi de la manipulation (B.9 b et c) est très proche de celle obtenue après la réobservation (B.9 d et e), confirmant un choix approprié du point de vue pour le suivi.

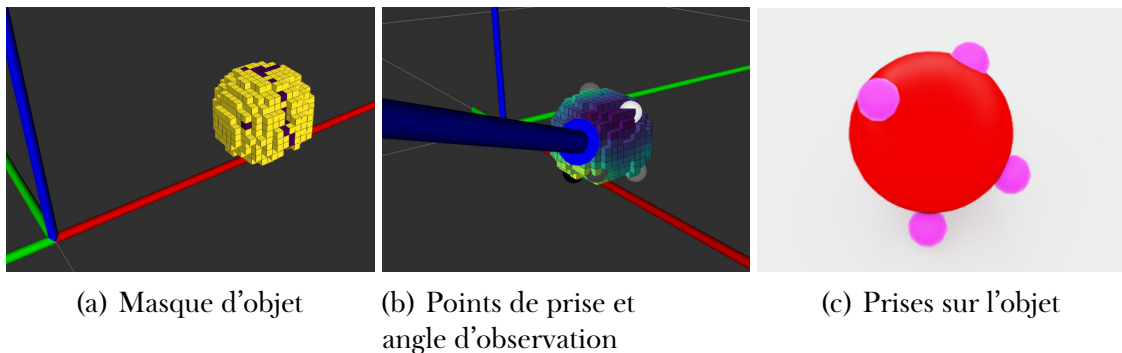


FIGURE B.8 : Points de prise (5 cm) pour remodeler la sphère en ovale

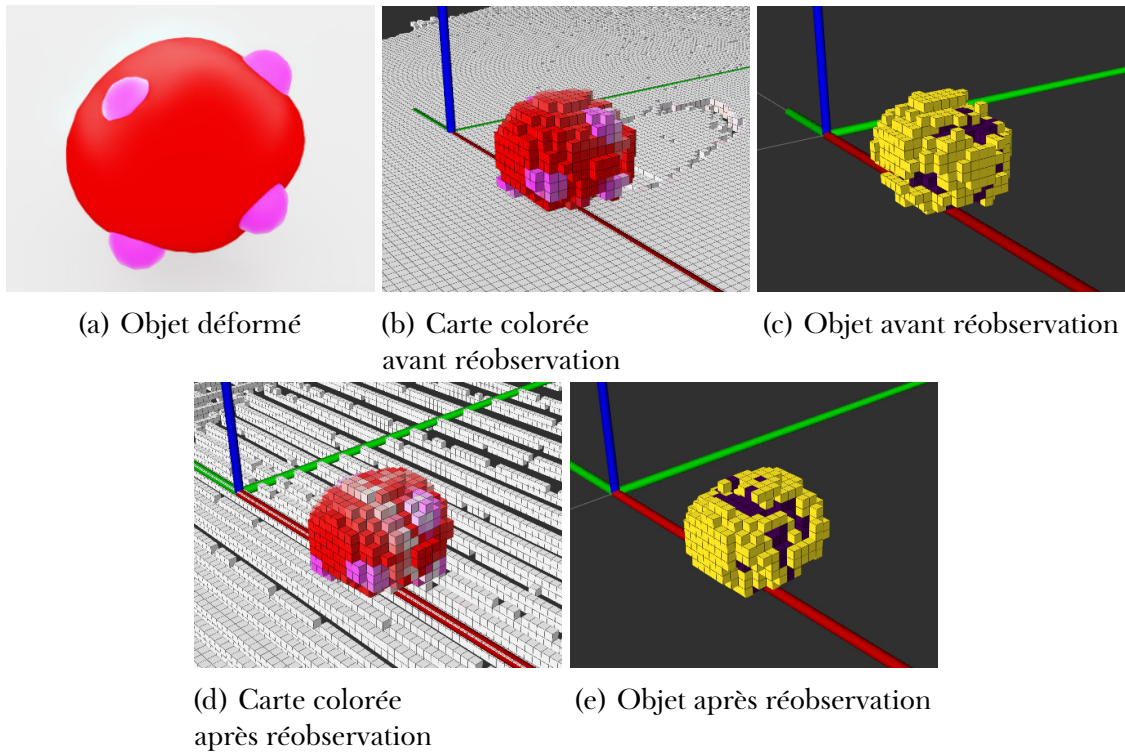


FIGURE B.9 : Résultat de la manipulation pour la sphère et l'ovale

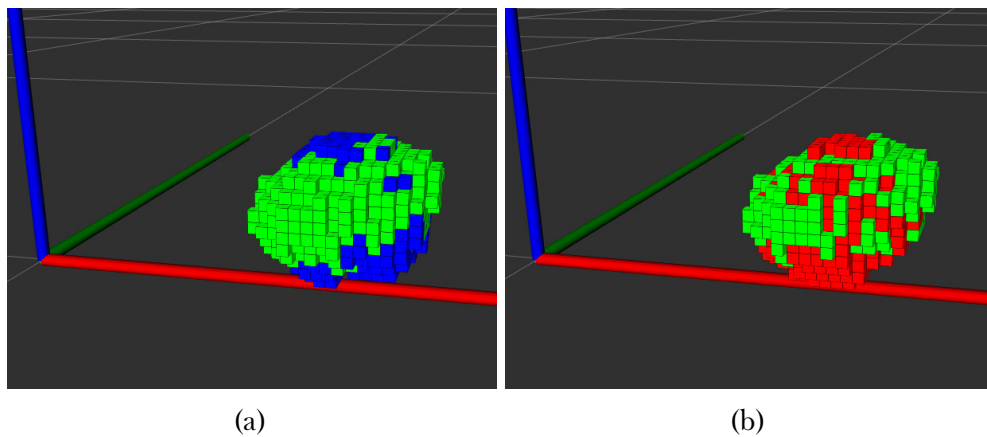


FIGURE B.10 : Comparaison des formes initiale (bleu) et finale (rouge) avec la forme cible (vert)

B.3.2 Pliage de la longue éponge

Ce scénario consiste à plier la longue éponge comme à la figure B.11 (a et b). Dans ce scénario, on note l'enjeu lié à l'alignement soulevé à la figure 7.2f, c'est-à-dire que l'alignement qui minimise la différence entre la forme initiale et la forme cible ne correspond pas nécessairement à l'alignement « naturel » pour effectuer le remodelage. La figure B.11 (b et c) montre la carte colorée initiale et l'objet d'intérêt une fois qu'il est complètement observé et segmenté. La figure B.12 montre les points de prise ainsi que le résultat de la manipulation avec des points de prise de 8 cm. Visuellement, ce résultat ressemble peu à la forme cible. Ainsi, on répète l'expérience avec des prises de 5 cm à la figure B.13, qui permet de constater qu'un plus grand nombre de plus petites prises améliore sensiblement le résultat. La figure B.14 compare la forme cible avec les formes initiale et finale. Dans ces résultats, on remarque que les prises de 8 cm et les prises de 5 cm réussissent à rapprocher l'objet de la forme cible. Cependant, aucun des deux cas ne réussit parfaitement à former le creux du pli en raison du problème d'alignement précédemment identifié, qui mène aussi à une torsion de l'objet perpendiculairement au mouvement souhaité. On remarque également une certaine instabilité de la simulation dans les figures B.12c et B.13c. Ces enjeux nuisent aux mesures et à la validation visuelle des résultats, bien que les valeurs numériques (tableau 8.1) montrent que la forme de l'objet s'est rapprochée de la cible.

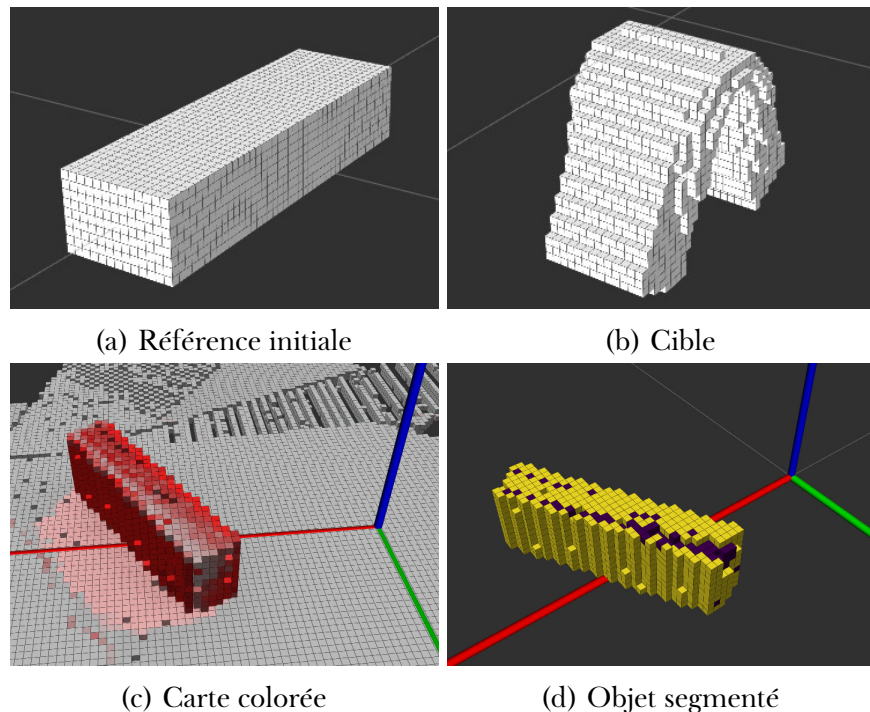
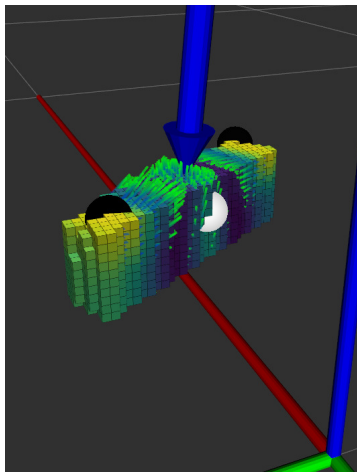
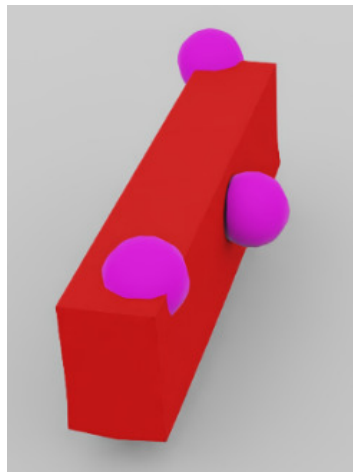


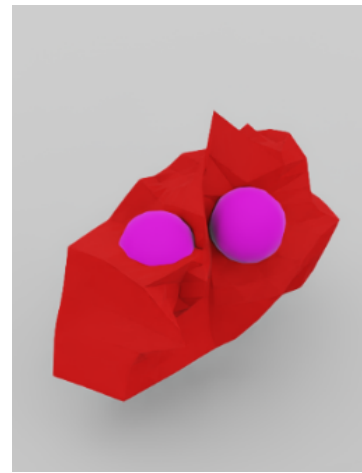
FIGURE B.11 : Situation initiale pour le pliage de la longue éponge



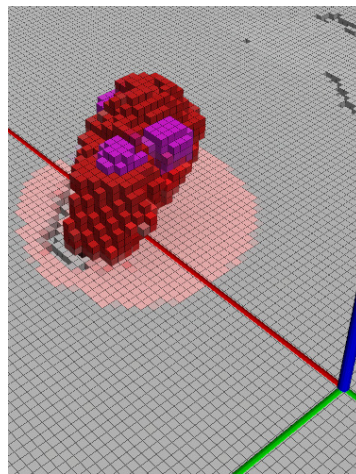
(a) Prises et angle d'observation



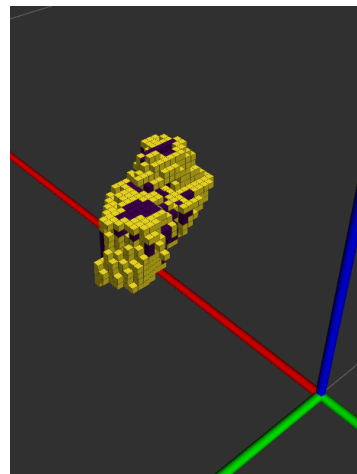
(b) Prises sur l'objet



(c) Objet déformé



(d) Carte colorée du résultat



(e) Résultat segmenté

FIGURE B.12 : Déformation de la longue éponge avec des prises de 8 cm

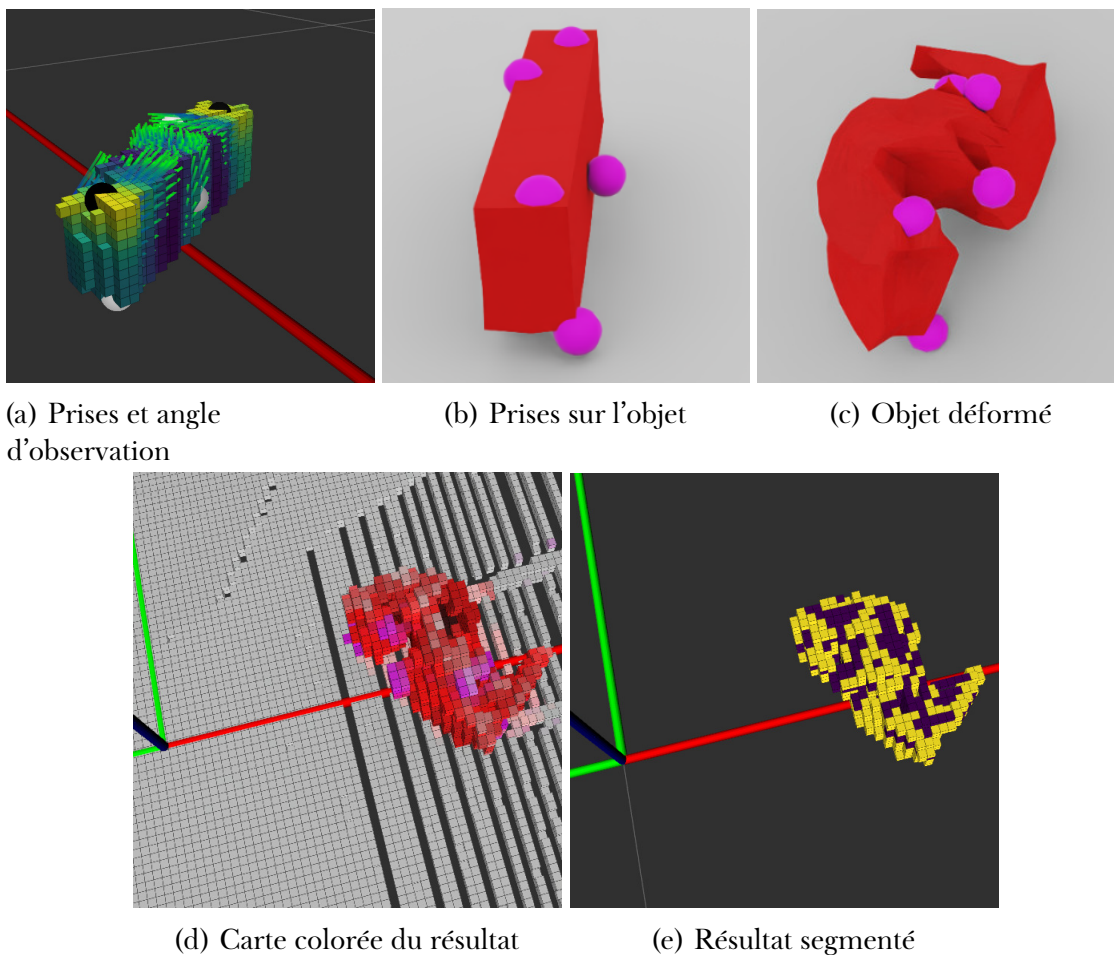


FIGURE B.13 : Déformation de la longue éponge avec des prises de 5 cm

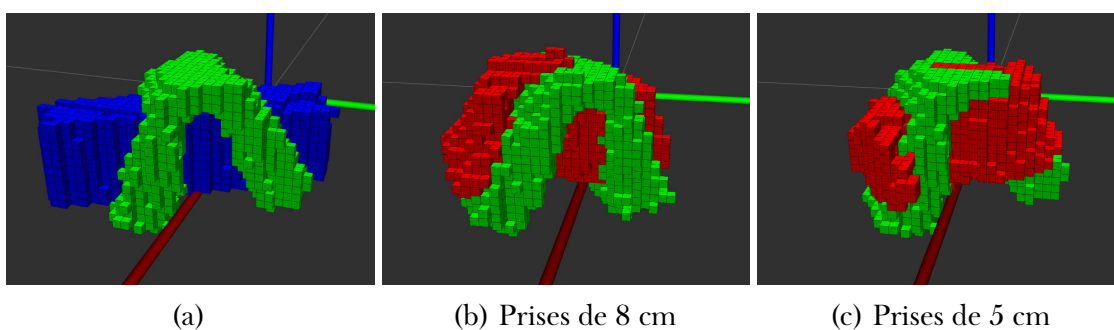


FIGURE B.14 : Comparaison des formes initiale (bleu) et finale (rouge) avec la forme cible (vert). Voir B.12c et B.13c pour une meilleure visibilité des formes finales.

Annexe C

Calcul de la courbure

C.1 Mesures de la courbure locale

Même avec un algorithme résistant aux variations de texture et d'éclairage, la couleur seule peut être insuffisante pour segmenter correctement un objet d'intérêt dans tous les scénarios, particulièrement lorsque plusieurs objets de couleur semblable sont à proximité les uns des autres. Ainsi, on peut s'inspirer de [87], qui suppose que les objets sont séparés par des bordures concaves, pour proposer la courbure locale comme mesure additionnelle dans la segmentation. Cette mesure peut aussi être utile pour évaluer la qualité d'un point de prise et favoriser le positionnement des doigts sur les faces planes de l'objet [8].

Cependant, on note que la mesure de concavité utilisée dans [87] requiert une connaissance des régions intérieure et extérieure de l'observation, puisqu'une bordure concave d'un côté d'une coquille mince est convexe de l'autre. Bien que cette information puisse être déduite à partir de la position de la caméra pour des observations individuelles, elle est plus difficile à obtenir dans une grille de voxels partielle résultant de la combinaison de plusieurs points de vue. Ainsi, on propose deux méthodes inspirées de celles utilisées en 2D dans [8] pour calculer cette mesure.

La première méthode proposée pour mesurer la courbure consiste à calculer la distance entre chaque voxel et le plan le mieux ajusté à la région environnante (calculé à l'aide des moments 3D, annexe A), tel que montré à la figure C.1a et à l'algorithme C.1. Avec cette méthode, la mesure de la courbure locale à un voxel v est la distance d entre v et le plan le mieux ajusté passant par le centroïde c de sa région avoisinante dans un rayon r , $\eta_r(v)$ (éq. (4.8)). La deuxième méthode proposée consiste à mesurer la différence d'angle du vecteur

normal au plan le mieux ajusté des régions voisines. Cette méthode, décrite à la figure C.1b et à l'algorithme C.2, consiste à calculer le vecteur normal au plan le mieux ajusté de $\eta_{r_p}(v)$, la région avoisinante de chaque voxel v à un rayon r_p , puis de calculer l'angle entre le vecteur calculé pour v et ses voisins à un rayon r_v , $\eta_{r_v}(v)$. La courbure mesurée au voxel v est l'angle minimum, maximum ou moyen (configurable) avec ses voisins.

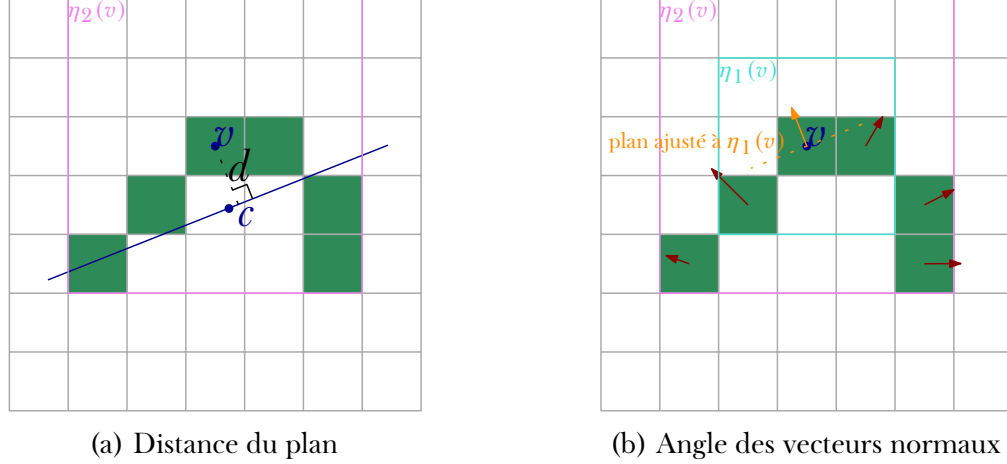


FIGURE C.1 : Illustration des mesures de courbure locale

Algorithme C.1 Calcul de la courbure — plan le mieux ajusté

Entrées

r , le rayon de la région à considérer

3: G , la grille de voxels à traiter

Sorties

C , une grille de voxels indiquant la courbure

6: **Pour tout** $v \in G \mid \Gamma_G(v) = 1$, **faire**

$S \leftarrow \{\eta_r(v), v\}$

► voisinage eq. (4.8)

calculer les moments 3D de la région $S \in G$

► annexe A

9: $c \leftarrow$ centroïde de la région S

$n \leftarrow$ 3e axe principal de la région S

► n est donc un vecteur normal au plan le mieux ajusté à S

12: $d \leftarrow v - c$

$\Lambda_C(v) \leftarrow |d \cdot n|$; $\Gamma_C(v) \leftarrow 1$

Algorithme C.2 Calcul de la courbure — angle des vecteurs normaux

Entrées

- G , la grille de voxels à traiter
- 3: r_p , le rayon de calcul du plan
 r_v , le rayon de calcul des angles

Sorties

- 6: C , une grille de voxels indiquant la courbure
 V , une grille à valeurs vectorielles de même résolution que G
- Pour tout** $v \in G \mid \Gamma_G(v) = 1$, **faire**
- 9: $S \leftarrow \{\eta_{r_p}(v), v\}$ ▷ voisinage eq. (4.8)
calculer les moments 3D de la région $S \in G$ ▷ annexe A
 $c \leftarrow$ centroïde de la région S
- 12: $\Lambda_V(v) \leftarrow$ 3e axe principal de la région S
 $\Gamma_V(v) \leftarrow 1$
- Pour tout** $v \in V \mid \Gamma_V(v) = 1$, **faire**
- 15: $A \leftarrow \emptyset$
 $N \leftarrow \eta_{r_v}(v)$
Si $\Gamma_A(n) = 0 \forall n \in N$ **alors**
- 18: $A \leftarrow 90^\circ$
 $\oplus$
Sinon
 $A \leftarrow \arccos \left(\frac{\Lambda_V(v) \cdot \Lambda_V(n)}{\|\Lambda_V(v)\| \cdot \|\Lambda_V(n)\|} \right) \forall n \in N$ ▷ reporté sur l'intervalle $[0, 90]^\circ$
- 21: $\Lambda_C(v) \leftarrow X(A)$; $\Gamma_C(v) \leftarrow 1$ ▷ X : minimum, maximum ou moyenne
-

C.2 Évaluation expérimentale

Pour les méthodes de mesure de courbure proposées à la section précédente, l’obtention d’une valeur de référence est encore plus complexe que pour la segmentation. Les résultats sont donc présentés sous forme visuelle en reprenant la scène composée (figure 5.3e) comme exemple.

Les figures C.2 et C.3 montrent le résultat du calcul de courbure basé sur la distance du plan (figure C.1a) avec des voxels de 12 et 3 mm, respectivement, et différents rayons r (en voxels) pour calculer le plan le mieux ajusté. Les figures C.4 et C.5, pour leur part, montrent le résultat du calcul de courbure basé sur les angles des vecteurs normaux (figure C.1b) avec des voxels de 12 et 3 mm respectivement, et différents rayons (en voxels) r_p pour calculer les vecteurs normaux et r_v pour les comparer. Le résultat attendu est une mesure de courbure élevée près des bords des objets rectangulaires et, selon l’interprétation où les objets sont séparés par des bordures concaves [87], à proximité des transitions entre les objets. À l’inverse, on s’attend à une courbure plus faible pour les objets sphériques dont la forme varie graduellement, et minimale pour les surfaces planes. Pour être utilisables en tant que critères directs pour la segmentation, ces mesures devraient permettre de distinguer les différents objets et les transitions entre ceux-ci.

Dans ces résultats, on remarque que, pour les deux algorithmes, le choix du rayon utilisé pour les calculs est un paramètre critique pour l’obtention de résultats corrects. Cependant, les valeurs optimales dépendent de la résolution et de la forme des objets — par exemple, avec la méthode du plan, les bords de la longue éponge sont les mieux identifiés avec un rayon entre 5 et 10 voxels lorsque ceux-ci font 3 mm (figure C.3d), mais avec des voxels de 12 mm les résultats sont meilleurs avec un rayon entre 3 et 5 (figure C.2b). Lorsque les rayons de calcul sont trop petits ou trop grands, on obtient un résultat flou qui ne permet pas d’identifier visuellement les bords des objets. De plus, on remarque une quantité significative de bruit dans la plupart des images, surtout lorsque les voxels sont grands, ce qui complique l’identification automatique des bords. Au vu de ces limites, l’utilisation directe de la courbure comme critère de segmentation n’a pas été retenue comme une méthode de segmentation viable dans le cadre de cette thèse. Cependant, les résultats obtenus démontrent que ces mesures ont un certain potentiel pour être intégrées avec d’autres critères dans un algorithme de segmentation multimodal ou pour la sélection de points de prise stables [8], deux aspects qui seront étudiés dans des travaux futurs.

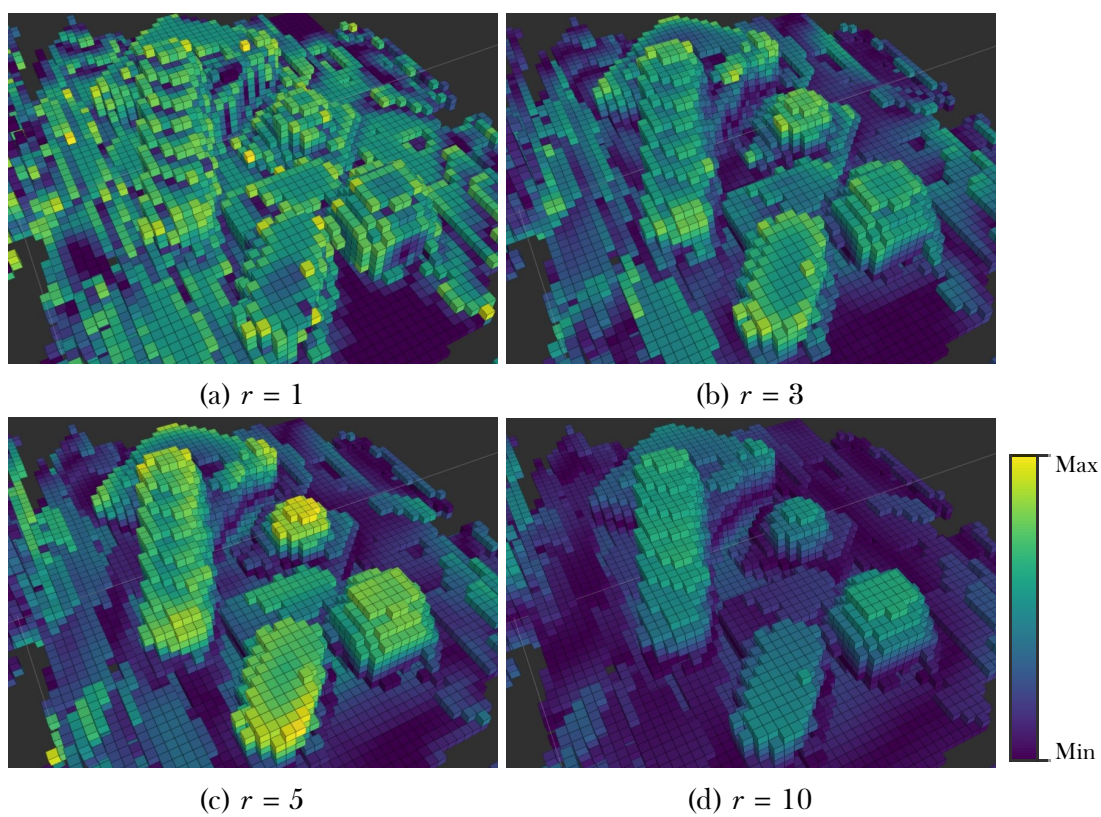


FIGURE C.2 : Résultat du calcul de courbure basé sur le plan avec des voxels de 12 mm.

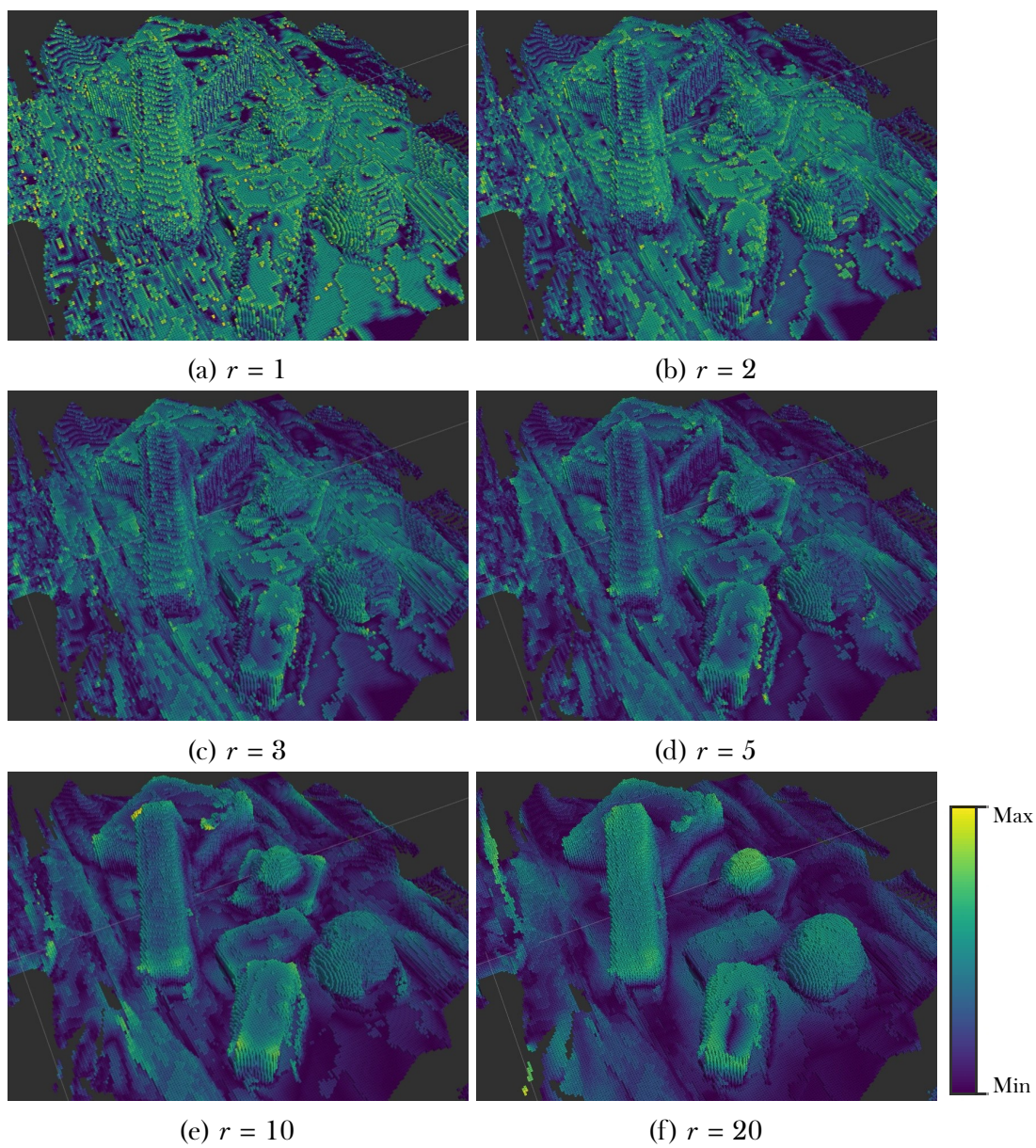


FIGURE C.3 : Résultat du calcul de courbure basé sur le plan avec des voxels de 3 mm.

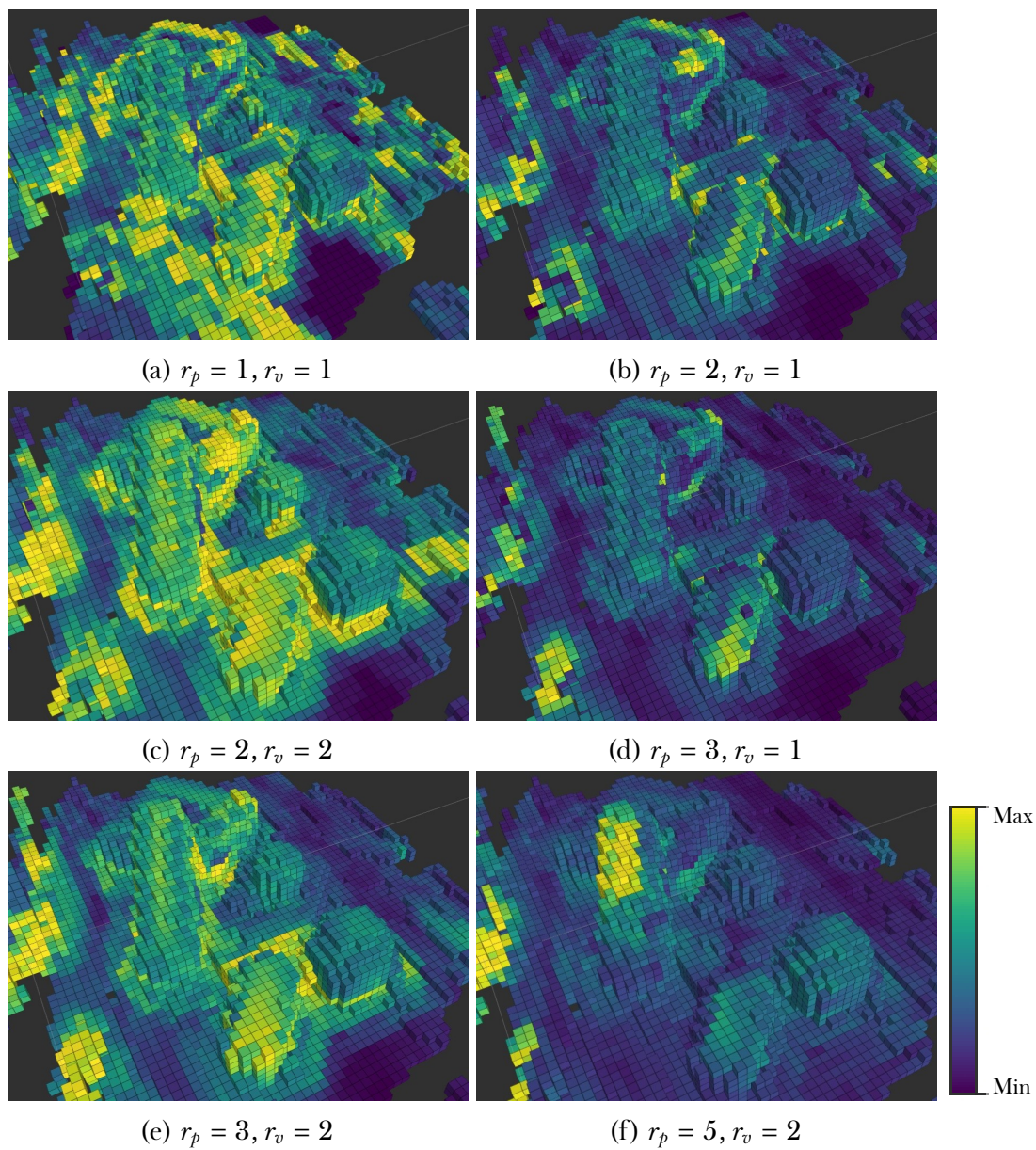


FIGURE C.4 : Résultat du calcul de courbure basé sur les vecteurs normaux avec des voxels de 12 mm.

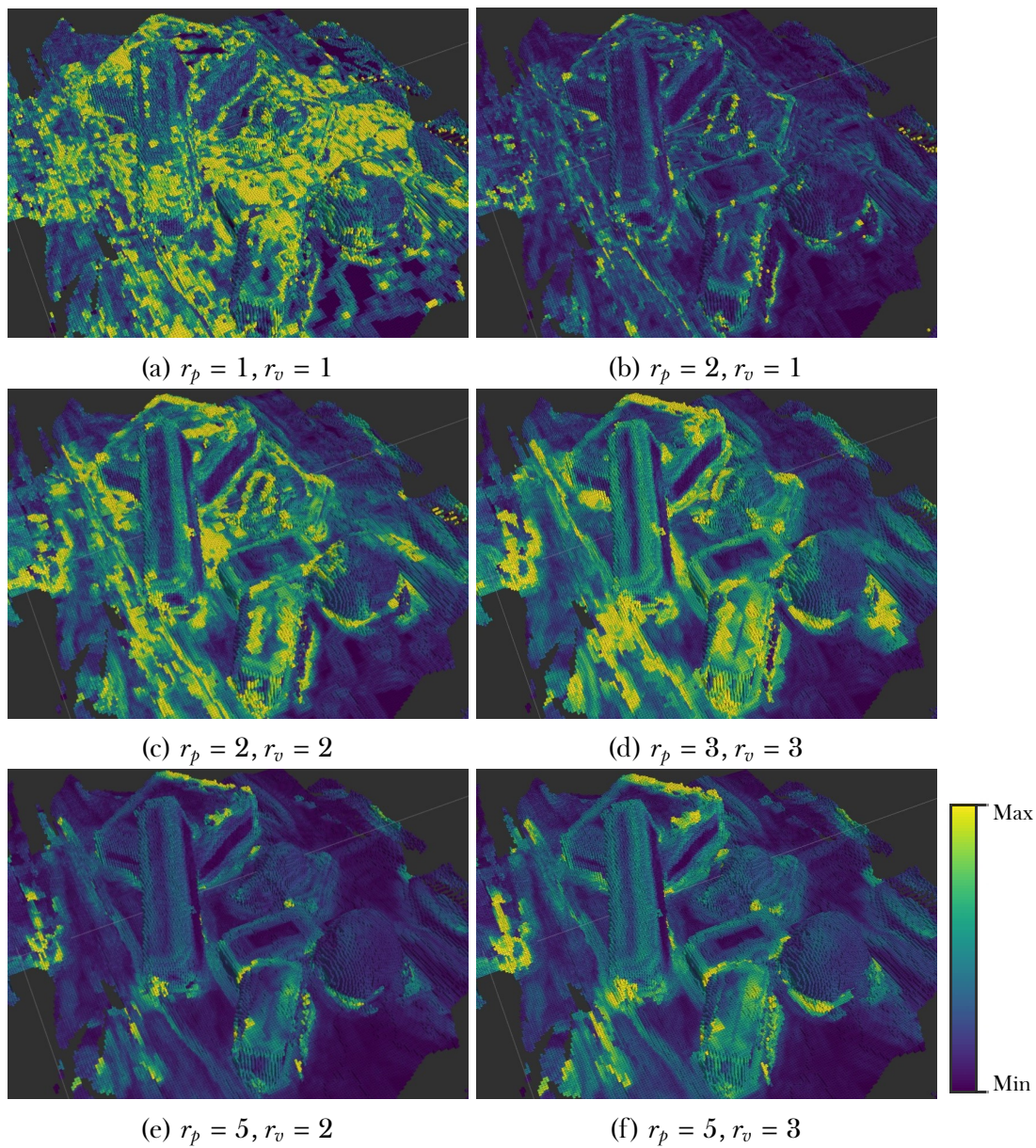


FIGURE C.5 : Résultat du calcul de courbure basé sur les vecteurs normaux avec des voxels de 3 mm.

Annexe D

Architecture logicielle et implémentation du module de cartographie

D.1 Bibliothèque de voxels

Le système de cartographie développé au chapitre 4 est basé sur des grilles de voxels efficaces fournies par la bibliothèque OpenVDB [93], qui encode les voxels dans une structure d'arbre peu profonde avec un encodage spécial dans la mémoire pour permettre un accès aléatoire rapide aux voxels ainsi qu'une faible utilisation de la mémoire. Comme il sera discuté ci-dessous, les caractéristiques clés de OpenVDB pour notre application incluent la représentation de voxels contenant une variété de valeurs, l'itération efficace à travers les voxels en sautant les voxels « vides », ainsi que l'accès direct aux voxels en utilisant des coordonnées cartésiennes. Ceci permet entre autres de transposer avec une relative facilité des algorithmes de traitement d'images bien établis vers la représentation 3D.

Bien qu'elle ait été développée pour l'animation par ordinateur, cette bibliothèque offre plusieurs caractéristiques utiles pour notre cas d'utilisation, dont l'encodage de l'espace dans une structure en arbre et le support efficace des opérations arithmétiques. L'encodage en arbre (pas nécessairement octrees) de OpenVDB est aussi « particulièrement adapté » [208] à la représentation de volumes constitués de régions intérieure et extérieure à valeurs fixes ainsi que d'une bande plus ou moins mince de voxels « actifs » (dont la valeur est changeante) entre les deux.

OpenVDB ne permet pas de stocker des valeurs et des types de données arbitraires et se limite aux nombres binaires, entiers ou décimaux, ainsi qu'à des vecteurs de trois éléments. Cette limitation fait en sorte qu'il n'est pas possible de conserver toutes les informations requises dans la même carte (par exemple la probabilité d'occupation, la couleur et la correspondance à l'objet d'intérêt ou à un obstacle). Ainsi, il est nécessaire de conserver en parallèle plusieurs cartes de l'environnement pour satisfaire aux différents besoins du système de manipulation robotique (segmentation, détection de collision). Cependant, ceci offre des avantages au niveau de la compression et simplifie grandement la définition des opérations logiques et arithmétiques. OpenVDB représente l'espace comme une grille infinie (dans les limites des variables d'index) de voxels indexés par coordonnées cartésiennes. Chaque voxel possède une valeur (qui dépend du type défini pour la grille) ainsi qu'un état booléen qui détermine si le voxel est « actif » ou non. Les régions semblables sont regroupées en « tuiles » dont la taille dépend des facteurs de branchement de la structure d'arbre sous-jacente. Pour les grilles à valeur réelle, OpenVDB inclut des fonctionnalités pour soutenir deux interprétations : la grille *Level-Sets*, constituée de deux zones inactives interprétées comme l'intérieur (valeurs négatives) et l'extérieur (valeurs positives) d'un objet et d'une mince bande active entre les deux, ainsi que la grille *fog volume*, où l'intérieur est à 1, l'extérieur est à 0 et une grande région active varie de 0 à 1 entre les deux.

Remarque au sujet des transformations Comme pour les images, le fait que l'espace voxélisé soit constitué d'une grille alignée aux axes fait en sorte que les opérations de rotation n'aient pas un résultat exact pour un angle quelconque (non multiple de 90°). Ceci peut donner lieu à des artefacts qui nuisent à la comparaison, notamment des trous ou des bosses dans l'objet transformé, comme on peut le voir à la figure D.1. Bien que des opérations morphologiques ou de filtrage pourraient réduire ces artefacts, il importe d'abord de noter qu'il y a deux méthodes possibles pour effectuer la transformation :

Soit O et T , les espaces voxélisés original et transformé. On a la transformation de coordonnées $f(p) \rightarrow p'$ qui relie les espaces de sorte que $O(p) = T(p')$. Le calcul de la valeur de $T(p')$ peut être fait en itérant à travers O pour déterminer les valeurs à « écrire » (c'est-à-dire $T(f(p)) \leftarrow O(p)$ pour chaque voxel $p \in O$) ou, à l'inverse, en itérant à travers T pour déterminer les valeurs à lire (donc $T(p') \leftarrow O(f^{-1}(p'))$ pour chaque voxel $p' \in T$). Dans le deuxième cas, on note qu'il est d'abord nécessaire de transformer la boîte englobante de O à T pour connaître l'étendue de l'espace transformé. Expérimentalement (voir la figure D.1), on note que la deuxième stratégie permet de réduire certains artefacts de rotation, notamment les trous.

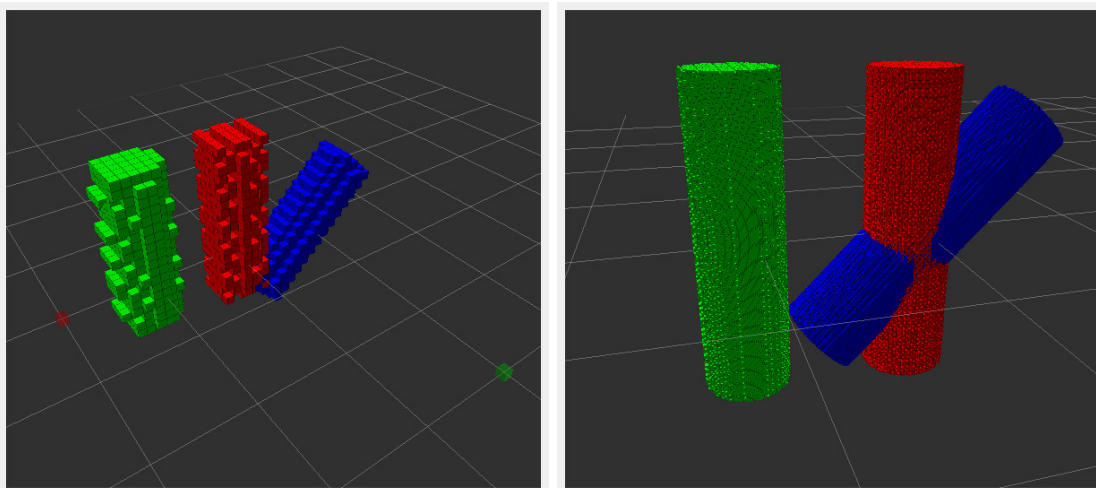


FIGURE D.1 : Voxélisation et transformation d'un modèle maillé à différentes résolutions. À partir du modèle original en bleu, la rotation avec la transformation directe (rouge) produit plus de trous que la transformation inverse (vert). À haute résolution (image de droite), les artefacts ne sont pas visibles à l'œil nu. Comparées avec une forme de référence transformée avant la conversion en voxels, la transformation directe obtient un score IoU de 0,5310 à basse résolution et de 0,6898 à haute résolution, et la transformation inverse des scores de 0,6232 et 0,8021 à basse et haute résolution, respectivement. Les deux versions transformées obtiennent un score IoU (l'une avec l'autre) de 0,4318 et 0,7784 à basse et haute résolution.

D.2 Application d'une séquence de filtres

L'algorithme D.1 gère la mise à jour d'une carte créée avec une séquence de filtres. À partir des listes d'opérations $H_M^{operations}$ et de paramètres $H_M^{parametres}$ définies dans les métadonnées H_M de la carte à mettre à jour M , on calcule le résultat de la séquence de filtres en appliquant l'algorithme D.2. Comme discuté précédemment, ce résultat M_r peut consister en une ou plusieurs cartes. Puisque le module de cartographie fonctionne avec un dépôt de cartes centralisé $R : a \rightarrow M_a$ et une interface par nom, la suite de l'algorithme consiste à fusionner les cartes et métadonnées de M_r avec les versions correspondantes enregistrées dans R . Ainsi, si M_r est une liste de cartes, on doit mettre à jour chaque sous-carte définie dans les métadonnées de la carte originale $H_M^{sous-cartes}$ et y ajouter celles qui n'y sont pas.

Algorithme D.1 Mise à jour d'une carte avec filtre

Entrées

- M , la carte à mettre à jour
- 3: H_M , les métadonnées de M
- $R : a \rightarrow M_a$, le dépôt de cartes ▷ section 4.2.1
- $M_r \leftarrow \Phi \left(H_M^{operations}, H_M^{parametres} \right)$ ▷ algorithme D.2
- 6: **Si** M_r est une liste de cartes **alors**
- $s \leftarrow H_M^{sous-cartes}$ (potentiellement vide)
- Pour tout** $G \in M_r$, **faire**
- 9: $n \leftarrow$ le nom de G
- Si** $n \in s$ **alors**
- $Z \leftarrow R(n)$
- 12: fusionner la carte et les métadonnées de G dans Z
- Sinon**
- $s \leftarrow n$
- 15: $R(n) \leftarrow G$
- Sinon**
- fusionner la grille et les métadonnées de M_r dans M
-

Pour sa part, l'algorithme D.2 gère l'application d'une séquence de filtres. Puisque les opérations à effectuer sont spécifiées en notation post-fixée dans la liste O , le principe de base consiste à enregistrer les opérandes successives au sommet d'une pile S jusqu'à ce qu'on rencontre un opérateur. Le nombre d'opérandes requis par cet opérateur est ensuite retiré du sommet de S , l'opération est exécutée et son résultat est enregistré au sommet de S avant de continuer à traverser la liste O . Puisque les filtres définis peuvent demander un nombre variable d'opérandes et produire un nombre variable de résultats, ce principe de base est modifié pour permettre l'utilisation de variables intermédiaires. Par exemple, supposons qu'on souhaite appliquer le filtre F_2 qui prend une seule opérande à un des résultats du filtre F_1 qui en produit deux. Avec la version de base, on spécifie la séquence $O = (M_i, F_1, F_2)$, ce qui correspond à évaluer $F_2(F_1(M_i))$. Puisque $F_1(M_i) \mapsto M_{o1}, M_{o2}$ produit deux résultats, après son évaluation on se retrouve avec la séquence (M_{o1}, M_{o2}, F_2) , ce qui correspond à évaluer $F_2(M_{o2}) \mapsto M_{o3}$, et le résultat de la séquence complète est $[M_{o1}, M_{o3}]$ plutôt que M_{o3} ou $F_2(M_{o1})$ (qui est impossible à obtenir).

Pour utiliser des variables intermédiaires, on définit l'opérateur $\mathfrak{s}$ pour le stockage de variables. On peut donc spécifier $O = (M_i, F_1, \mathfrak{s}, a, F_2), Q = ([noms = [b, a]])$ où b, a sont les paramètres de l'opérateur $\mathfrak{s}$ et correspondent aux noms à donner aux variables intermédiaires. Après l'évaluation de F_1 , ceci résulte en $(M_{o1}, M_{o2}, \mathfrak{s}, a, F_2)$. Les variables intermédiaires sont ensuite enregistrées de sorte qu'on a $V(a) = M_{o1}, V(b) = M_{o2}$ et donc la séquence restante (a, F_2) correspond à l'évaluation de $F_2(a) = F_2(M_{o1})$ qui produit le résultat escompté. De plus, pour faciliter l'implémentation des fonctions F_o qui ne sont généralement pas commutatives, on inverse l'ordre des opérandes lorsqu'on les retire de la pile (ligne 21) de sorte que l'opération $F_i(M_1, M_2)$ est spécifiée par la séquence (M_1, M_2, F_i) plutôt que (M_2, M_1, F_i) comme en notation post-fixée standard. Enfin, puisque le module de cartographie fonctionne avec un dépôt centralisé et une interface par noms, on a les structures $R : a \rightarrow M_a$ et $F : o \rightarrow F_o$ qui relient respectivement le nom d'une carte à la carte elle-même et le nom d'un opérateur à la fonction qui l'implémente et à son nombre d'opérandes, qui peut être variable selon les paramètres q .

Les différents opérateurs sont définis dans une structure de données qui, pour chacun d'entre eux, garde en mémoire les éléments suivants :

- le nom de l'opérateur;
- une fonction qui détermine le nombre d'opérandes à partir des paramètres de l'opérateur;
- la fonction permettant d'exécuter l'opérateur. Celle-ci prend en entrées une liste de grilles et un dictionnaire de paramètres, et retourne une grille ou une liste de grilles.

Ceci permet de facilement définir les opérateurs et d'encapsuler dans le mécanisme de filtres les différentes fonctions définies pour modifier les cartes, sans devoir accéder aux détails internes du module de cartographie et du dépôt de cartes. Ce mode de fonctionnement augmente donc la flexibilité du module de cartographie en permettant de définir et d'exécuter les opérations nécessaire à l'accomplissement de la tâche considérée.

Algorithme D.2 Application d'une séquence de filtres

Procédure $\Phi(O,P)$ **Entrées**

- 3: O , la liste d'opérations et d'opérandes en notation post-fixée
 Q , la liste des ensembles de paramètres pour les différentes opérations
 $F : o \rightarrow F_o$, la structure reliant le nom d'un opérateur o à sa fonction F_o
- 6: $R : a \rightarrow M_a$, le dépôt de cartes ▷ section 4.2.1
- $S \leftarrow$ une pile vide dont S_0 est le sommet
 $V : x \rightarrow M_x$ une association entre un nom et une carte ▷ pour stocker les variables
- 9: **Pour tout** $o \in O$, **faire**
 Si $o = \varsigma$ **alors** ▷ stockage de variable
 $q \leftarrow$ le prochain élément de Q
- 12: **Pour tout** $n \in q.noms$, **faire**
 $V(n) \leftarrow S_0$
- Sinon**
- 15: **Si** $o \in V$ **alors** ▷ lecture de variable
 $S_0 \leftarrow V(o)$
- Sinon**
- 18: **Si** $o \in F$ **alors** ▷ opération
 $q \leftarrow$ le prochain élément de Q
 $n \leftarrow$ le nombre d'opérandes de F_o en fonction de q
- 21: $l \leftarrow S_{n-1}, S_{n-2} \dots S_0$
 $r \leftarrow F_o(l, q)$
 $S_0 \leftarrow a \ \forall a \in r$
- 24: **Sinon** ▷ o est le nom d'une carte
 $g \leftarrow R(o)$
 $S_0 \leftarrow g$
- 27: **Si** S contient plus d'un élément **alors** **Retourner** S
 Sinon, **Retourner** S_0
-

D.3 Interface ROS2

Pour assurer la flexibilité du module de cartographie et faciliter sa réutilisation ainsi que son interopérabilité avec les autres composantes du système, il est conçu comme un paquet ROS2 autonome, tirant parti des interfaces de message et de service de ROS2 pour permettre la création de cartes ainsi que leur mise à jour, affichage et interrogation par les autres composantes du système global. Trois nœuds ROS2 sont conçus pour séparer les aspects de la création/interrogation des cartes, de leur mise à jour, et de la préparation de différentes visualisations. Un aperçu de cette structure est présenté à la figure D.2 et ses différentes composantes sont décrites ci-dessous.

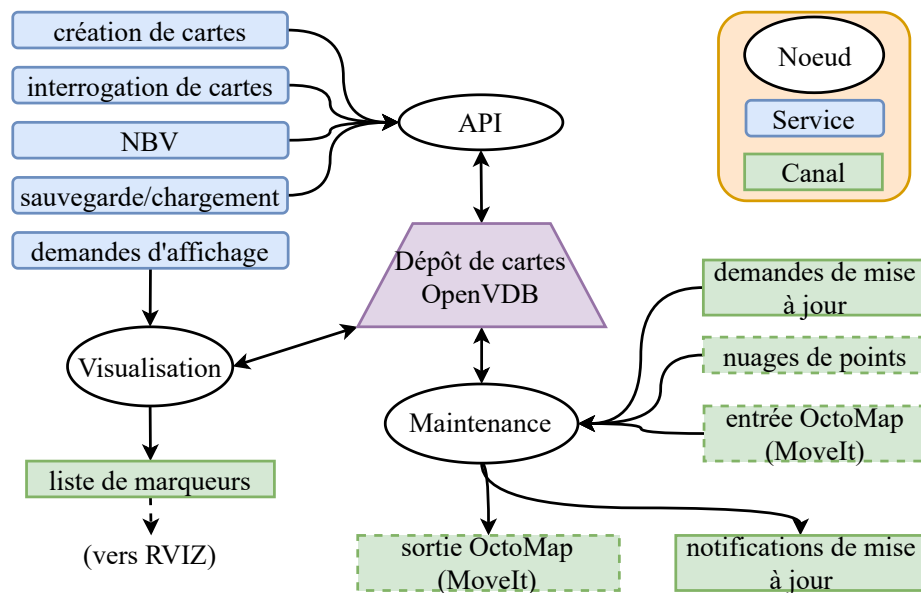


FIGURE D.2 : Structure de l'interface ROS2 conçue pour le module de cartographie.

D.3.1 Dépôt de cartes OpenVDB

Le dépôt de cartes OpenVDB a pour rôle principal de maintenir le lien entre le nom d'une carte et la structure OpenVDB associée (appelées *grilles* dans le vocabulaire d'OpenVDB). Ce module est donc responsable de maintenir la cohérence de la cartographie entre les nœuds, qui s'exécutent dans des fils ou processus différents (section D.3.5). Cette interface par nom est essentielle au fonctionnement du module de cartographie puisqu'elle permet d'isoler les grilles OpenVDB des autres parties du système, qui peuvent simplement référer aux cartes par leur nom. De plus, ce module encapsule certaines opérations courantes sur les grilles pour les rendre plus conviviales.

D.3.2 Noeud d'interface (API)

Le nœud d'interface (API) est responsable des interactions avec les autres parties du système. Ce nœud présente donc six services ROS2 pour permettre de créer des cartes, de les sauvegarder et charger d'un fichier, de les modifier, de les interroger, et de calculer les heuristiques du prochain point de vue. Ces services sont décrits ci-dessous.

Création de cartes Le service de création de cartes est le point d'entrée principal du module de cartographie pour les autres parties du système. Une carte est créée en spécifiant un nom, une source de données et les paramètres liés à celle-ci, la taille d'un voxel, ainsi que la méthode et le délai de mise à jour. Cinq sources de données sont implémentées, qui prennent chacune leurs propres paramètres :

Modèle maillé sous la forme d'un fichier standard stl. Les paramètres sont le chemin du fichier, ainsi qu'un facteur d'échelle et une transformation à appliquer avant l'importation.

Nuage de points provenant d'un canal ROS2. En plus du nom du canal d'où recevoir les données, les paramètres indiquent le type de carte à créer (valeur directe ou probabilité d'occupation) ainsi que les valeurs définissant le modèle de calcul pour l'occupation (ternaire ou probabiliste et les paramètres associés).

Scène de planification pour la grille d'occupation OctoMap maintenue par MoveIt.

Archive qui effectue une copie statique d'une autre carte.

Filtre dont les paramètres sont une séquence d'opérations ainsi que leurs options, tel que décrit à la section 4.2.4.

De même, trois méthodes de mise à jour sont utilisées. Les cartes peuvent être mises à jour manuellement (en publiant un message sur un canal), automatiquement lorsque la source de données change, ou encore à intervalles réguliers spécifiés par un délai de mise à jour. Notons qu'il est toujours possible de forcer la mise à jour manuelle d'une carte.

Sauvegarde et chargement Ces services permettent aux différentes parties du système de contrôler la sauvegarde et le chargement des cartes vers un fichier. La fonction de sauvegarde permet d'enregistrer une, plusieurs ou toutes les cartes du dépôt vers un fichier, en formatant le nom de fichier pour éviter les conflits. La fonction de chargement, pour sa part, permet de charger les cartes de manière statique (non mises à jour) ou dynamique (mise à jour selon les modalités définies à la création de la carte). Ceci permet donc d'enregistrer les cartes pour

analyse ultérieure, ou encore de suspendre le travail pour le reprendre à un moment ultérieur. Cette dernière fonctionnalité est particulièrement utile en simulation, où elle permet de sauter l'étape répétitive de l'acquisition de la forme initiale de l'objet à manipuler.

Modification Ce service permet d'effectuer des modifications ponctuelles à une carte existante. Trois types de modification sont implémentées :

- **Métadonnées** : La modification des métadonnées permet de modifier, par exemple, les paramètres de mise à jour d'une carte ou les options d'un filtre. Ceci est notamment utilisé pour mettre à jour le point de référence utilisé pour la segmentation (chapitre 5).
- **Filtre** : Permet d'appliquer un filtre quelconque à une carte (section 4.2.4).
- **Remplissage** : Permet de remplir une région rectangulaire alignée sur les axes avec une valeur arbitraire. Ceci est utilisé pour réinitialiser une portion de la carte d'occupation après la manipulation (section 4.3.2).

Interrogation Le service d'interrogation permet aux autres parties du système d'obtenir diverses informations sur les cartes, notamment :

1. Les métadonnées, qui peuvent inclure des résultats calculés par les opérations de filtres (notamment la transformation d'alignement, section 7.1).
2. La valeur et l'état d'activité de la carte à un ou plusieurs points.
3. Les points ou régions qui représentent des maximums globaux ou locaux, avec l'option de considérer ou non les voxels actifs ou inactifs. Cette fonctionnalité est utilisée pour la sélection de prise (section 7).
4. Les propriétés volumétriques de la carte, comme les moments 3D (annexe A), la taille des voxels en mètres, ou encore la boîte englobante des voxels actifs.

Prochain point de vue (NBV) Ce service est séparé du service d'interrogation pour offrir plus de convivialité dans la définition des paramètres et résultats. Il est une simple encapsulation de l'algorithme décrit à la section 6.1 pour effectuer la conversion de données avec le service ROS2 et calculer la boîte englobante de la carte remplie résultant de l'algorithme.

D.3.3 Noeud de maintenance

Le nœud de maintenance est responsable d’effectuer la mise à jour en continu des cartes gérées par le module de cartographie. Ceci implique notamment la gestion des canaux sur lesquels on reçoit les nuages de points, des minuteries pour les cartes qui sont mises à jour à intervalles réguliers, ainsi que de la chaîne de dépendances pour celles qui sont mises à jour automatiquement au changement des données source. Deux canaux sont maintenus par ce nœud pour recevoir les demandes de mise à jour et publier une notification lorsqu’une carte est rafraîchie. Ceci permet aux autres parties du système de demander la mise à jour manuelle d’une carte, mais aussi d’attendre qu’il y ait un changement. Ces canaux sont aussi utilisés à l’intérieur du nœud de maintenance : lorsqu’on publie une notification de mise à jour, on publie aussi une requête de mise à jour pour toutes les cartes à mise à jour automatique qui dépendent de celle qui a changé initialement. Ceci fait en sorte d’utiliser le mécanisme de passage de messages et d’ordonnancement de ROS2 pour gérer efficacement la mise à jour séquentielle d’un grand nombre de cartes sans bloquer le système.

Étant responsable de la mise à jour des cartes, le nœud de maintenance contient aussi les algorithmes pour la conversion des nuages de points en cartes (section 4.2.2), la conversion des grilles OctoMap vers OpenVDB, ainsi que le mécanisme de filtres décrit à la section 4.2.4.

D.3.4 Noeud de visualisation

Le nœud de visualisation a pour rôle de convertir les cartes vers un format qui peut être affiché dans RVIZ, une interface graphique de visualisation 3D intégrée dans ROS2. Pendant l’exécution du système de cartographie, le nœud de visualisation affiche un service ROS2 qui prend en entrée le nom d’une carte à afficher, le délai de mise à jour, et le type de visualisation à présenter. Les voxels de la carte sont convertis en marqueurs cubiques colorés selon le type de visualisation, puis transmis à RVIZ pour l’affichage. Par défaut, la fonction de visualisation détecte automatiquement la coloration appropriée pour les cartes à valeur réelle de type *fog volume* ou *Level-Sets* ainsi que pour les valeurs vectorielles encodant une couleur (type RGB). D’autres paramètres de visualisation sont fournis pour afficher les cartes d’occupation (coloration centrée sur 0.5) et les cartes vectorielles encodant un déplacement, comme celles résultant de l’algorithme d’association des voxels (section 7.2). Par souci d’efficacité, seuls les voxels marqués comme étant actifs sont affichés par ce service. Les cartes peuvent être affichées une seule fois à la réception de la requête, ou encore à intervalles réguliers. Le nœud de visualisation offre aussi un service ROS2 permettant aux autres parties du système d’afficher

plus facilement les flèches, points, boîtes englobantes et autres marqueurs facilitant le suivi de l'exécution des divers algorithmes.

En plus de ces fonctionnalités, le nœud de visualisation peut aussi être utilisé « hors-ligne » de manière interactive pour l'analyse et l'affichage de cartes sauvegardées. Ceci donne accès à un plus large éventail de fonctionnalités pour contrôler l'affichage, notamment pour sélectionner si on affiche ou non les voxels inactifs et d'arrière-plan, contrôler la coloration, limiter l'affichage à des régions particulières, ou visualiser la carte couche par couche.

D.3.5 Parallélisme et performance

Dans le modèle d'exécution de ROS2, les trois nœuds s'exécutent en parallèle. Cependant, ceux-ci doivent tous trois avoir accès au même dépôt de cartes et Python ne permet pas de parallélisme des calculs à l'intérieur du même processus. Le transfert des cartes entre les processus étant relativement coûteux, différentes variations sur le parallélisme sont testées. Un fonctionnement équilibré est atteint lorsque les nœuds d'interface et de maintenance s'exécutent dans le même processus et celui visualisation dans un 2^e processus. Le dépôt de cartes implémente un mécanisme de sérialisation avec des nœuds client et serveur qui, une fois configuré, est transparent du point de vue des trois nœuds du module. Ceux-ci peuvent donc stocker et retirer des cartes du dépôt qu'ils soient dans le même processus ou non, sans modifier leur implémentation. Étant donné que ce mécanisme est implémenté avec les interfaces ROS2, il peut également être configuré pour permettre l'accès direct aux grilles OpenVDB par les autres parties du système.