Perception et génération de trajectoires en cobotique dans des environnements dynamiques

Guillaume Fuseiller

Thèse de doctorat

Université de Limoges

ED 610 - Sciences et Ingénierie des Systèmes, Mathématiques, Informatique (SISMI)

Xlim, équipe ReMix

Thèse pour obtenir le grade de

Docteur de l'Université de Limoges

Image, signal et automatique

Présentée et soutenue par

Guillaume Fuseiller

Le 6 décembre 2019

Perception et génération de trajectoires en cobotique dans des environnements dynamiques

Thèse dirigée par Ouiddad LABBANI-IGBIDA, Professeur des Universités, Institut XLIM, ENSIL-ENSCI, université de Limoges

Co-encadrée par Gilles MOURIOUX, Maitre de Conférence, Institut XLIM, ENSIL-ENSCI, université de Limoges

et Erick DUNO, Responsable standardisation process, VALEO

JURY :

Président

M. Youcef MEZOUAR, Professeur des universités, Institut PASCAL, université Clermont Auvergne

Rapporteurs

M. Pierre VIEYRES, Professeur des universités, laboratoire PRISME, université d'Orléans

M. Jean-Pierre GAZEAU, Ingénieur Recherche CNRS HDR, Institut PPRIME, université de Poitiers ENSMA

Remerciements

En tout premier lieu je tiens à remercier Erick DUNO pour avoir mis en place cette thèse au sein de l'entreprise Valeo et m'avoir accompagné et soutenu tout au long de ce travail.

Je remercie du fond du coeur Romain MARIE tout d'abord pour son soutient face à l'épreuve que représente la thèse, pour avoir été un collègue de travail fantastique et un ami.

Mes remerciements vont ensuite à Ouiddad LABBANI-IGBIDA et Gilles MOURIOUX qui ont encadrés cette thèse au sein de l'équipe ReMix du Laboratoire Xlim et supportés durant ces années.

Je remercie également mes collègues en commençant par ceux de Valeo, en premier Rachid pour nos passionnantes conversations et pauses cafés, Guillaume et Fernand, toute l'équipe des méthodes (Guillaume, David, Philippe, Jean-François, Aurélien, Joseph, Joseph, Jean-Pierre, Patrice, Patrice, Antonio, Dominique et Eric). Vient ensuite le tour des doctorants : Jeremy, Brice, Bilel, Hela, Fatima et Julien puis de l'ensemble des enseignants et chercheurs de la spécialité mécatronique de l'ENSIL-ENSCI : Joanny, Thierry, Thierry, David, Stéphane et Michel.

Je remercie L'ENSIL, devenue ENSIL-ENSCI, et son directeur Patrick LEPRAT pour m'avoir accueilli au sein de ses locaux.

Je remercie le groupe Valeo et particulièrement Valeo Materiaux de Friction et le site de Limoges qui a financé ces travaux de recherche et m'a accueilli à bras ouverts ainsi que le Laboratoire Xlim de l'université de Limoges.

Je remercie ma famille, mes parents, mon frère pour leur soutient, mes amis et enfin Solène qui m'a aidé à traverser l'étape difficile de la rédaction.

Sommaire

Avant-propos

Le groupe Valeo est un équipementier automobile comptant plus de 180 sites à travers le monde. L'activité de l'usine de Limoges est la production des garnitures d'embrayage à destination des véhicules de tourisme ou des véhicules industriels (poids lourds, tracteurs, bus). En tant qu'usine mère, les équipes développent les produits et processus qui sont par la suite reproduits à l'international (Brésil, Chine, Corée du Sud, Inde). Cherchant à intégrer la cobotique dans ses moyens de production, l'entreprise a mis en place une thèse CIFRE en partenariat avec l'équipe Robotique et Mécatronique (ReMix) du laboratoire Xlim. Ce manuscrit présente les résultats de cette collaboration.

Résumé

L'apparition au début des années 2010 des robots collaboratifs en parallèle du développement de l'industrie 4.0 a incité les industriels à repenser la robotique. L'intégration de robots au contact des humains permet de cumuler leurs points forts en termes de flexibilité et d'adaptabilité à des tâches complexes. Cependant de nombreux défis sont posés à la communauté robotique par ces nouvelles pratiques et notamment : comment sécuriser l'interaction entre l'Homme et le Robot. Nous proposons de traiter cette problématique en proposant un pipeline complet permettant de déplacer en autonomie le robot dans l'espace partagé avec l'humain tout en garantissant sa sécurité. Pour cela nous construisons une représentation de l'espace de travail du robot sous forme d'une grille d'occupation 3D sémantique à partir de la perception du robot de son environnement. Nous projetons alors en temps réel les obstacles et humains présents dans la zone de travail du robot vers l'espace des configurations, une représentation adaptée aux bras robotiques. En utilisant l'axe médian dans l'espace des configurations, la trajectoire du robot vers l'objectif est calculée au plus loin des obstacles. Nous adaptons cette trajectoire au changement dynamique de l'environnement et modulons sa vitesse en fonction de la distance séparant l'humain et le robot. L'intérêt de cette méthode a été démontré en simulation et avec des expérimentations réelles sur des robots collaboratifs en contexte industriel.

Mots clés

Cobotique, interaction homme-robot, planification de trajectoire, espace des configurations, bras robot industriel, environnement dynamique, carte d'occupation

Abstract

The appearance in the early 2010s of collaborative robots in parallel with the development of industry 4.0 prompted manufacturers to rethink robotics. The integration of robots in contact with humans makes it possible to combine their strengths in terms of flexibility and adaptability to complex tasks. However, many challenges are posed to the robotics community by these new practices and in particular: how to secure the interaction between Human and Robot. We propose to address this problem by proposing a complete pipeline allowing the robot to move autonomously in the space shared with humans while guaranteeing its safety. For this we build a representation of the robot's workspace in the form of a 3D semantic occupation grid from the robot's perception of its environment. We then project in real time the obstacles and humans present in the robot's work area to the configurations space, a representation adapted to robotic arms. Using the medial axis in the configurations space, the robot's trajectory towards the objective is calculated as far as possible from obstacles. We adapt this trajectory to the dynamic change of the environment and modulate its speed according to the distance separating the human and the robot. The interest of this method has been demonstrated in simulation and with real experiments on collaborative robots in an industrial context.

Keywords

Cobotic, human robot interaction, path planning, configurations space, industrial robot arm, dynamic environment, occupancy map

Chapitre 1.
Introduction générale

Sommaire du chapitre :

1.1 Contexte de la thèse

1.1.1 La cobotique dans l'industrie

Depuis son apparition au début des années 60, la robotique s'est progressivement imposée à l'ensemble du monde industriel, en remplaçant les humains dans des tâches de plus en plus complexes. Son utilisation est toutefois encadrée par des règlementations très strictes, qui définissent les précautions à prendre pour se prémunir contre les accidents. Historiquement, les robots restaient ainsi enfermés derrière des barrières de protection, limitant fortement leurs interactions avec les humains. Mais l'apparition à la fin du vingtième siècle de la robotique de service a changé la vision de la cohabitation entre l'Homme et le robot. On s'est ainsi rendu compte que ces derniers n'étaient pas cantonnés à la réalisation de tâches répétitives ou dangereuses, et qu'ils pouvaient tout à fait collaborer avec des opérateurs dans la réalisation de missions beaucoup plus complexes : on parle alors de cobotique (de l'anglais « collaborative robotics »).

Par essence, le robot est précis, capable de porter de lourdes charges, de répéter des gestes sans fatigue ni ennui, et de fonctionner sans interruption. Pour autant, il n'est pas, à l'heure actuelle, capable de s'adapter à un changement de situation, et les perturbations extérieures peuvent rapidement l'empêcher de mener sa tâche à bien. L'humain est au contraire très adaptable et peut, par expérience, avoir une connaissance de la tâche et de son environnement bien supérieure au robot. En combinant ces points forts, on obtient alors un moyen de production plus flexible qu'une machine robotisée et plus rapide qu'un opérateur seul.

Néanmoins, pour profiter d'une collaboration directe entre humains et robots, les mesures de sécurité qui imposaient une séparation non négociable des espaces de travail doivent être revues. A la place d'une barrière physique, c'est maintenant l'ensemble des briques logicielles et matérielles qui doivent pouvoir garantir l'intégrité physique des opérateurs. Depuis une dizaine d'année, cette révolution s'est accélérée grâce à l'arrivée sur le marché de "cobots"(Figure 1), dont la conception et/ou les fonctions de sécurité permettent une collaboration "sûre" entre l'humain et le robot. Celle-ci est obtenue en évitant les contacts ou en limitant leurs conséquences (par exemple avec des liaisons compliantes). Reste néanmoins plusieurs défis, notamment sur le plan logiciel, pour que la collaboration Homme-robot soit entièrement effective.

Figure 1 : Exemple de cobot : le Yumi de ABB.

1.1.2 Scénario de travail

Nous proposons dans cette thèse d'explorer cette collaboration Homme/machine à travers un cas d'application concret : la fabrication de garnitures d'embrayage. Il s'agit d'une pièce en matériau composite qui assure la transmission du couple entre le volant moteur et le mécanisme d'embrayage. Sa production, complexe, implique de nombreuses étapes détaillées en annexe Annexe 1 :. Nous nous intéressons ici en particulier aux étapes de formage, pressage et ébavurage qui sont regroupées au sein d'un même espace de travail nommé « îlot de cuisson » (cf. Figure 2).

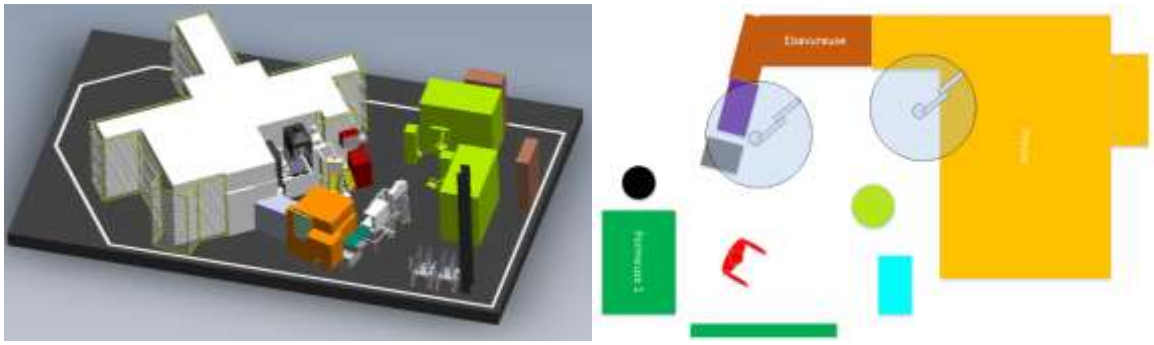


Figure 2 : Vue 3D d'un îlot de cuisson (à gauche) et schéma d'implantation (à droite). Les disques bleus représentent le positionnement envisagé des robots.

L'étape de formage consiste à mettre sous la forme d'un disque, du fil imprégné issu des étapes précédentes, à l'aide d'une machine nommée formeuse. Celle-ci va amener entre deux plateaux pressés l'un contre l'autre, le fil, en décrivant une épitrochoïde[1]. La pièce ainsi obtenue est un disque d'épaisseur régulière nommé préforme.

Cette préforme est alors introduite dans un moule, qui entre dans une presse, pour y suivre un cycle de thermocompression. Cette transformation va permettre de lier les épaisseurs de fil entre elles, et de donner une forme quasi définitive à la garniture

[1] Sinusoïde portée par un cercle

(diamètres intérieur et extérieur et forme des rainures). Cette couronne est appelée ébauche.

À la suite du moulage dans les presses, les ébauches présentent des bavures dues à l'écoulement du surplus de matière. L'étape d'ébavurage consiste à les éliminer à l'aide de bandes abrasives.

Historiquement, la manipulation des pièces entre ces trois étapes est réalisée par un opérateur. Comme le montrent les flèches de flux sur la Figure 3, celui-ci est chargé :

a. d'alimenter les formeuses en fil,
b. de collecter les préformes et de les amener à la presse,
c. de charger la presse,
d. de récupérer les ébauches à la sortie de l'ébavureuse,
e. d'effectuer un contrôle visuel de ces dernières,
f. de les placer sur un chariot nommé luge,
g. et enfin de remplacer les luges pleines partant vers l'étape de production suivante.

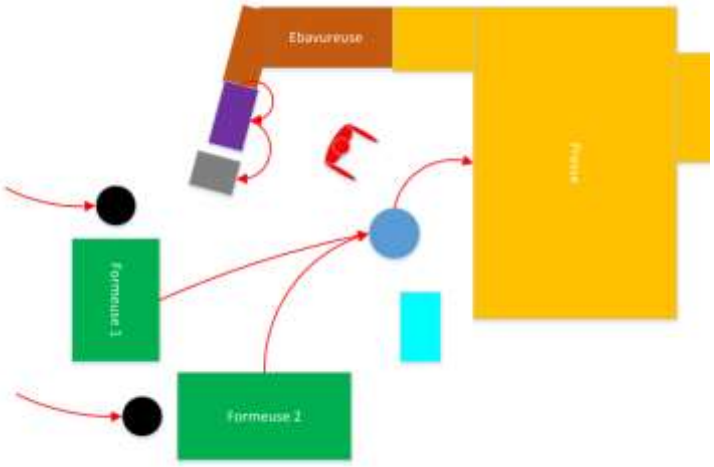


Figure 3 : Implantation d'un îlot de cuisson, les flèches symbolisent les flux de pièces.

La robotisation de ces tâches est rendue difficile par la présence d'aléas pouvant perturber le déroulement du cycle de production. Par exemple, des pièces peuvent rester collées dans les moules. Celles-ci doivent alors être retirées avant de pouvoir être rechargées. Dans ces situations, l'intervention d'un opérateur est requise, nécessitant l'arrêt de robots traditionnels. Pour limiter ces phases d'arrêt, les travaux de thèse ont porté sur l'utilisation d'un système cobotique, basé sur le scenario suivant :

L'opérateur reste chargé des formeuses et de l'alimentation en préforme (tâches a et b), ainsi que du renouvellement des luges (tâche g). Les tâches c à f sont normalement dévolues au robot, afin de profiter de sa productivité importante. Dans les cas non gérés par le robot (en particulier les aléas évoqués précédemment), l'humain doit collaborer

avec lui en intervenant sur ses tâches. Le robot adapte alors son comportement dès qu'il perçoit l'humain, et reprend son cycle de façon automatique au départ de ce dernier.

Pour parvenir à cela, trois zones distinctes sont définies :

- ✓ La zone du robot est exclusivement réservée à celui-ci. L'humain n'a pas la permission d'y pénétrer. D'après la norme ISO/TS 15066 (AFNOR 2016), si cette zone est violée, le robot doit impérativement se mettre en arrêt d'urgence.
- ✓ La zone de l'humain est interdite au robot, sa programmation ou sa conception doivent le garantir. L'opérateur peut s'y déplacer sans influencer le comportement du robot.
- ✓ La zone collaborative correspond à l'espace de travail partagé dans lequel les deux acteurs peuvent intervenir conjointement. Le robot adapte son comportement (ses trajectoires) en fonction de l'activité de l'humain.

1.2 Positionnement de la thèse

L'une des principales difficultés liées à la robotique est la multiplicité et la complexité des tâches nécessaires à implémenter avant de pouvoir correctement exécuter une mission. Dans le contexte cobotique, le robot doit en plus garantir la sécurité de l'humain. Cela implique qu'il soit capable de réagir en temps réel à des événements survenant dans son espace de travail. Avant d'entrer dans le détail des contributions de cette thèse, il est donc important de détailler ces différentes tâches et d'identifier les challenges qui y sont associés.

1.2.1 Perception et modélisation de l'espace

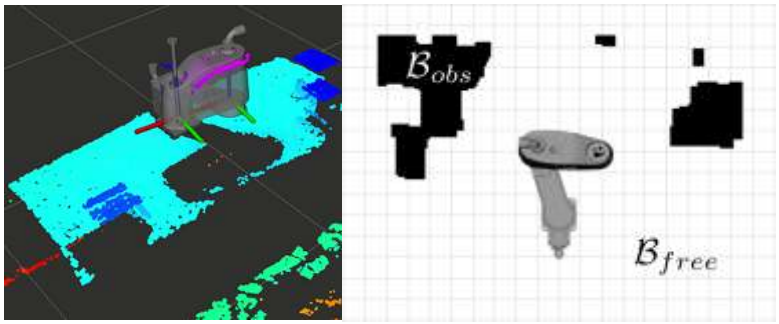


Figure 4 : De la perception à la modélisation de l'espace de travail du robot.

La première d'entre elles est de fournir au robot un moyen de percevoir ce qui l'entoure. Cela implique de choisir et de positionner un ensemble de capteurs, de manière adéquate. Ces décisions sont influencées par la nature même de l'environnement. Il est donc important d'identifier les contraintes qui découlent de ses spécificités. D'après le scénario évoqué précédemment, nous faisons les hypothèses suivantes :

- La scène est hautement dynamique : les actions de l'humain, très rapides, requièrent une fréquence d'observation élevée. Nous nous imposons pour cela une fréquence minimale de 20Hz. De plus, le contexte industriel implique une évolution de l'espace de travail, certes moins rapide, mais à prendre en compte (cycle solaire, présence/mouvements d'objets divers, …). Les traitements associés doivent donc y être robustes.
- La tâche est collaborative : plusieurs opérateurs peuvent être présents dans l'espace de travail. Leur sécurité étant prioritaire, les modalités de perception choisies doivent permettre de les localiser précisément et de les distinguer des autres obstacles. De plus, le champ d'observation doit couvrir l'ensemble de l'espace de travail, et éviter les zones mortes induites par le robot ou les humains.

La perception seule, aussi parfaite soit-elle, ne suffit pas à automatiser les actions du robot. L'étape suivante consiste donc à formaliser les observations issues des différents capteurs (potentiellement hétérogènes) au sein d'une modélisation de l'environnement exploitable par les algorithmes de prise de décision. Cette représentation est soumise aux mêmes hypothèses que la perception (haute dynamicité et présence d'humains). Elle doit donc être mise à jour en temps réel, contenir les informations de typage des obstacles, et garantir la fiabilité de son contenu.

1.2.2 Planification et exécution de trajectoires

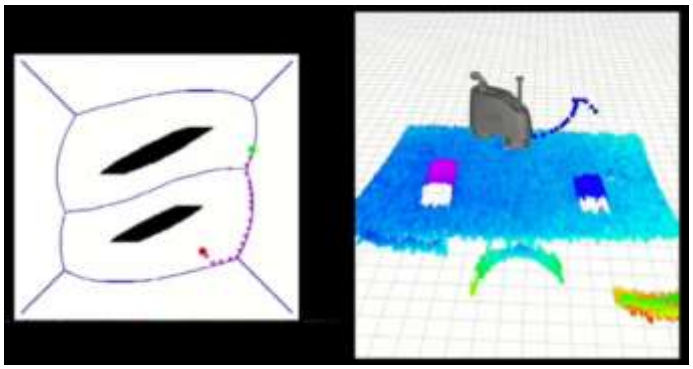


Figure 5 : Planification et réalisation d'une trajectoire.

Une fois doté d'une représentation de son environnement, le robot doit choisir l'action à mener pour réaliser sa mission. Cette tâche n'est pas traitée dans le cadre de cette étude. Plus spécifiquement, nous faisons l'hypothèse qu'à un instant donné, l'information est connue. Il est toutefois intéressant de noter que dans le cadre de la collaboration Homme robot, des algorithmes aboutis d'ordonnancement existent et peuvent être adaptés au cas de cette étude (Alami et al. 2006).

L'action à réaliser étant connue, il reste une dernière tâche préalable au mouvement : la planification d'une trajectoire. Cette problématique a souvent été étudiée, mais les

implications de la présence d'humain et la nécessité de garantir la sécurité de ces derniers complexifie cette dernière.

Une fois la trajectoire planifiée, la dernière tâche consiste à commander le robot pour suivre cette trajectoire. Les robots utilisés dans notre contexte ayant tous un contrôleur de mouvement, nous nous reposerons sur lui pour générer les déplacements.

1.3 Contributions de la thèse

Comme évoqué précédemment, les travaux présentés dans ce manuscrit s'intéressent à l'ensemble des problématiques de l'application collaborative, depuis la perception de l'environnement, jusqu'à la génération de trajectoires sûres permettant de réaliser la mission. Les principales contributions de la thèse sont :

- Une représentation multisensorielle de l'espace de travail :

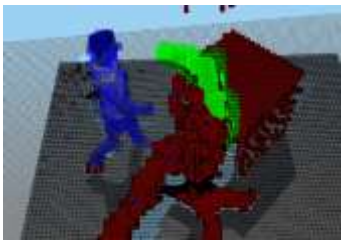


Figure 6 : Représentation de l'espace de travail (colorisée en fonction des informations sémentiques).

Après avoir sélectionné et positionné les capteurs adéquats pour observer la scène, nous avons proposé une représentation sous forme de grille d'occupation, enrichie d'informations sémantiques. Celle-ci précise notamment la nature des obstacles, permettant en particulier d'identifier et de localiser les humains dans l'espace de travail. Cela permettra par la suite de définir pour le robot des comportements adaptés.

- La projection temps-réel des obstacles dans l'espace des configurations :

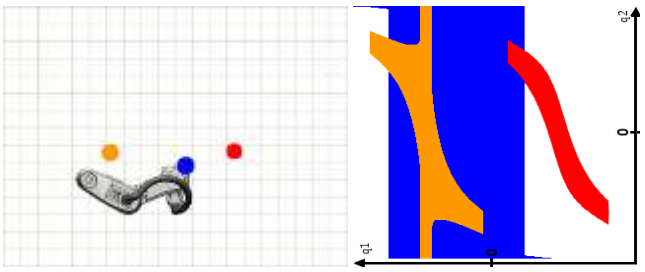


Figure 7 : Exemples de projections de points obstacles dans l'espace de travail vers l'espace des configurations.

L'espace des configurations est une représentation adaptée au bras manipulateur, puisqu'elle réduit sa complexité géométrique à un unique point dans un espace à N dimensions (avec N le nombre d'articulations). Nous avons

donc proposé une méthode optimisée de projection des obstacles du monde cartésien vers l'espace des configurations.

Basée sur le calcul préalable d'une table de valeurs (Look Up Table), elle permet de construire cette représentation dans des délais compatibles avec la dynamique de notre application. Cette méthode permet notamment de projeter l'information sémantique de la carte d'occupation, et aussi de planifier des trajectoires adaptées et sécurisantes pour les opérateurs directement dans l'espace des configurations.

- **La planification et la mise à jour des trajectoires sur l'axe médian de l'espace des configurations :**

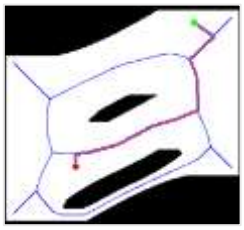


Figure 8 : Planification de trajectoire sur l'axe médian de l'espaces des configurations.

L'axe médian, aussi appelé squelette, est par définition le chemin le plus éloigné des obstacles. Cette propriété fait de lui un candidat idéal pour générer des trajectoires sûres. Nous avons donc proposé un planificateur basé sur l'axe médian dans l'espace des configurations.

Afin de répondre au besoin de réactivité de notre application, nous avons d'abord travaillé à l'optimisation de l'extraction du squelette en N dimensions, en proposant une nouvelle stratégie pour l'obtention de la carte des distances (étape intermédiaire).

Nous avons ensuite intégré les informations sémantiques dans le calcul de l'axe médian afin de l'éloigner des obstacles humains. En finalité, cette transformation augmente la distance entre le robot et les opérateurs, et renforce ainsi la sécurité des trajectoires choisies.

Compte tenu de la dynamique importante de l'environnement de travail, le squelette évolue au fil du temps. Pour cette raison, nous avons finalement proposé une méthode de mise à jour continue de la trajectoire courante, qui garantit la sécurité des humains tout en maintenant la continuité des mouvements du robot.

1.4 Plan du manuscrit

La suite de ce manuscrit est organisée de la manière suivante :

Le chapitre II présente un état de l'art des différentes problématiques abordées par la thèse. Après un tour d'horizon des travaux portant sur la cobotique dans l'industrie et une revue rapide des normes encadrant son utilisation, nous exposons les principaux travaux de recherche traitant de la planification de trajectoire, notamment en environnement dynamique.

Le chapitre III décrit le processus de construction des différentes représentations utilisées. Nous le débutons en justifiant le choix des capteurs utilisés et leur positionnement. Par la suite, nous explicitons la génération de la carte d'occupation de l'espace de travail, et en particulier la labélisation des différents obstacles. Pour finir, nous abordons la projection vers l'espace des configurations, en commençant par le pré-calcul de la Look Up Table, avant de détailler son utilisation.

Le chapitre IV est dédié à la planification de trajectoires. Partant de la représentation de l'espace des configurations, nous explicitons le calcul de la carte des distances et l'extraction du squelette. Nous détaillons ensuite le processus de planification, à savoir : la méthode de sélection d'une trajectoire sur l'axe médian, sa mise à jour réactive, et l'échantillonnage des points de passage envoyés au contrôleur du robot.

Le chapitre V est consacré à la validation expérimentale de l'ensemble de nos propositions, à travers l'utilisation de notre pipeline dans deux contextes expérimentaux différents, à la fois en simulation et en environnement réel. Les premiers résultats correspondent à une version simplifiée du scénario de travail, et sont obtenus sur un robot à deux degrés de liberté. La seconde partie provient d'expérimentations faites à partir d'un démonstrateur en adéquation avec le contexte industriel de la thèse.

Le chapitre VI conclut cet ouvrage et en fait le bilan. Nous revenons alors sur l'ensemble du travail réalisé, et suggérons des pistes pour la poursuite de ces travaux ainsi que des perspectives d'extensions à d'autres domaines.

Chapitre 2.
Etat de l'art

Sommaire du chapitre :

Dans ce second chapitre, nous présentons l'état de l'art relatif aux travaux menés. En premier, nous commençons par aborder la collaboration Homme-robot. Puis nous discuterons de la planification de trajectoires. Nous abordons le sujet dans sa globalité en présentant les méthodes génériques puis le cas plus spécifique de l'interaction Homme-robot.

2.1 Collaboration Homme robot

Cette section débute par la présentation des modes de collaboration prévus par les normes applicables puis présente les travaux du domaine. Ceux-ci seront présentés au travers de trois thématiques, tout d'abord la sécurisation de l'interaction puis la perception de l'environnement dans le contexte cobotique et enfin la quantification du risque pour les humains.

2.1.1 Normes industrielles

Du point de vue des industriels, la robotique collaborative est définie par la norme ISO/TS 15066 (AFNOR 2016). Celle-ci prévoit quatre modes de collaboration utilisables en fonction des applications souhaitées (cf. Figure 9) :

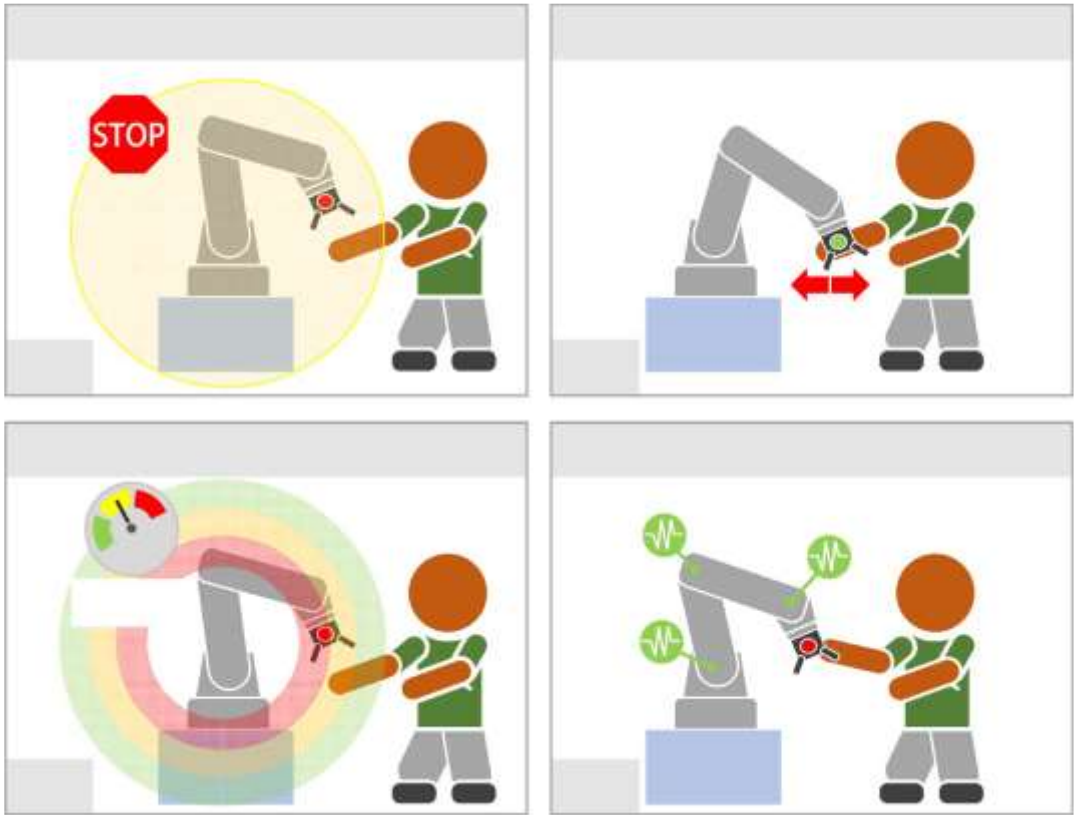


Figure 9 : Les quatre modes de collaboration selon (Villani et al. 2018). En haut à gauche : arrêt nominal de sécurité, en haut à droite : guidage manuel, en bas à gauche : surveillance de la vitesse et de la séparation, en bas à droite : limitation de la puissance et de la force.

- L'arrêt contrôlé nominal de sécurité : dans ce mode de collaboration, le robot s'arrête pour laisser l'opérateur agir. Ainsi aucun mouvement du robot n'est possible tant que l'opérateur est dans l'espace de travail collaboratif. Ce mode nécessite un dispositif de détection de personnes dans l'espace de travail collaboratif. Il permet un redémarrage automatique du robot lorsque l'opérateur quitte l'espace collaboratif.
- Le guidage manuel : comme dans le mode précédant, le robot s'arrête pour laisser entrer l'opérateur dans l'espace de travail collaboratif, mais celui-ci peut, via une commande locale, déplacer le robot. Ce mode permet par exemple de réaliser l'apprentissage de trajectoire ou le déplacement de lourdes charges guidé par l'opérateur.
- La surveillance de la vitesse et de la séparation : dans ce mode de fonctionnement, le robot peut bouger alors que l'opérateur est dans la zone de travail collaboratif mais doit se maintenir à une distance de sécurité de l'opérateur. La trajectoire du robot (position et vitesse) doit permettre de maintenir cette distance de sécurité, si tel n'est pas le cas, le robot doit s'arrêter ou changer de trajectoire.
- La limitation de la puissance et de la force : le dernier mode de travail collaboratif permet aussi le mouvement du robot dans l'espace de travail collaboratif et même le contact entre le robot et l'opérateur. Une appréciation des risques doit être réalisée afin de déterminer les limitations de force, de vitesse et/ou de puissance à appliquer au robot.

2.1.2 La question de la sécurité

L'un des aspects fondamentaux dans la collaboration Homme/robot est la sécurité des opérateurs lorsqu'ils évoluent à l'intérieur de l'espace de travail du robot. De nombreuses stratégies ont été élaborées au fil du temps, afin de garantir l'intégrité physique des opérateurs, ou, à minima, de minimiser les risques de blessures ou la gravité de ces dernières. Pour les présenter, nous proposons de suivre la classification introduite par Ikuta et al. (Ikuta, Ishii, et Nokata 2003), présentée par la Figure 10.

<table>
<tr><th></th><th></th><th>Control Strategy</th><th>Design Strategy</th></tr>
<tr><td rowspan="2">Pre-Collision</td><td>avoid collision</td><td>distance</td><td>-</td></tr>
<tr><td rowspan="2">minimize impact force</td><td>speed</td><td rowspan="2">weight</td></tr>
<tr><td rowspan="4">Post-Collision</td><td rowspan="4">attenuation diffusion</td><td rowspan="4">stiffness</td><td>moment of inertia</td></tr>
<tr><td>cover</td></tr>
<tr><td>surface</td></tr>
<tr><td>joint compliance</td></tr>
<tr><td>shape</td></tr>
</table>

Figure 10 : Classification des solutions de prévention des risques dans la collaboration Homme-robot. D'après [Ikuta03].

Deux grandes familles de stratégies y sont décrites. D'un côté, les stratégies post-collisions partent du principe que des contacts Homme-robot non désirés peuvent survenir. Elles visent donc à définir des moyens (matériels et logiciels) permettant d'en réduire les conséquences. De l'autre, les stratégies pré-collisions cherchent à anticiper en temps réel la survenue de ces dernières, et définissent donc des solutions garantissant qu'elles ne pourront se produire, ou, à minima, qu'elles préserveront l'intégrité physique des opérateurs.

Il est à noter que cette façon de classer les travaux n'est pas propre à (Ikuta, Ishii, et Nokata 2003), et qu'elle semble commencer à faire consensus dans la communauté, puisque reprise dans de très nombreux travaux : (Heinzmann et Zelinsky 2003), (Kulić et Croft 2006), (Broquère 2011), (Robla-Gomez et al. 2017). Dans (Robla-Gomez et al. 2017), la nomenclature inclut en outre une différentation en fonction des capteurs utilisés.

2.1.2.1 Les stratégies post-collision

Afin de minimiser les conséquences corporelles lors de collisions entre un opérateur et le robot, la principale des stratégies consiste à prendre des précautions au moment de sa conception. Plusieurs travaux (Suita et al. 1995), (Hun-Ok Lim et Tanie 1999), (Yamada et al. 2005), (Ulmen et Cutkosky 2010) ont ainsi proposé de couvrir l'ensemble du robot d'un revêtement viscoélastique (Figure 11), ce qui a le double avantage d'amortir la violence des chocs, et d'offrir des capteurs de contact sur l'ensemble de la surface du robot. Il devient alors possible de définir des lois de commande adaptées capables d'ajouter une sécurité supplémentaire lors des contacts.

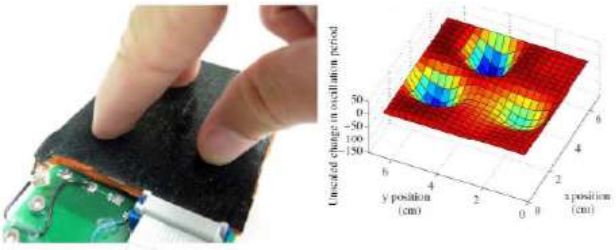


Figure 11 : Exemple de revetement viscoélastique proposé dans [Ulmen10], capable de détecter avec précision les zones de contact.

Cette volonté de définir des lois de commande réactives capables d'absorber une partie de l'énergie du système (due à sa masse et à sa vitesse) lors des collisions a d'ailleurs fait l'objet de plusieurs travaux, souvent menés par les mêmes équipes que celles ayant développées les peaux capacitives. On citera ainsi les travaux de (Ulmen et Cutkosky 2010), dont la loi de commande a mené à une réduction significative de l'énergie libérée lors d'un impact, ou encore ceux de (D. Kulic et Croft 2004), qui utilisent une commande en rigidité des actionneurs (basés sur des actionneurs compliants décrits juste en dessous).

Bien sûr, de nombreux autres travaux indépendants traitent également du sujet. Un état de l'art plutôt complet est d'ailleurs proposé dans (Santis et al. 2008).

La dernière grande catégorie de travaux visant à anticiper la survenue de collisions en minimisant leurs conséquences se concentre directement sur la structure mécanique du robot, en remplaçant les parties traditionnellement rigides et lourdes par des structures légères (basées par exemple sur des matériaux composites (Hirzinger et al. 2002) et/ou élastiques). On citera notamment les travaux orientés sur la mise au point d'actionneurs à rigidité variable (Pratt et Williamson 1995), (Bicchi et al. 2005), (Ahmed et Kalaykov 2010), qui se différencient par la façon dont cette variation de rigidité est introduite. Ainsi, dans (Pratt et Williamson 1995), un ressort est placé en série entre la sortie de l'actionneur et le segment du robot qu'il permet de commander. Dans (Bicchi et al. 2005), les articulations sont cette fois commandées par deux actionneurs antagonistes couplés à des ressorts. Ce sont d'ailleurs ces derniers qui sont utilisés dans (D. Kulic et Croft 2004) pour mettre en place une commande jouant sur la rigidité des actionneurs. Plus récemment, (Ahmed et Kalaykov 2010) ont proposé de remplacer les traditionnels ressorts par du fluide magnéto-rhéologiques, capable de passer de l'état liquide à l'état solide (et inversement) en quelques millisecondes. Dans les faits, la littérature est très riche sur le sujet. Un état de l'art sur le sujet est d'ailleurs proposé dans (Vanderborght et al. 2013).

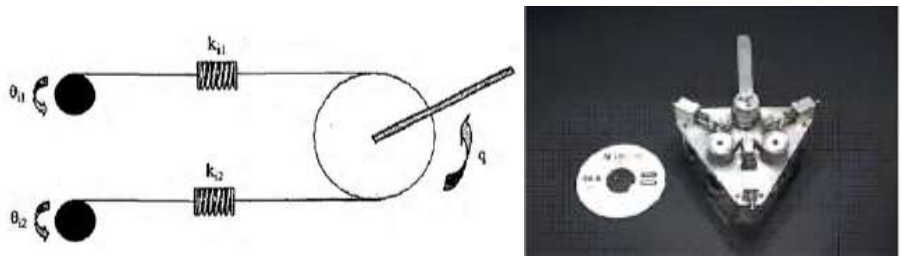


Figure 12 : (gauche) Principe de la liaison compliante proposée dans (D. Kulic et Croft 2004). (droite) Actionneur à rigidité variable proposé dans (Tonietti, Schiavi, et Bicchi 2005).

2.1.2.2 Les stratégies de pré-collision

Bien qu'il soit indispensable d'anticiper la survenue de contact Homme/robot, se contenter de stratégies post-collisions ne peut être satisfaisant. En effet, même en garantissant des conséquences sur l'intégrité des humains dans les limites du supportable, la douleur et les blessures restent une réalité. De très nombreux chercheurs se sont donc penchés sur la problématique visant à anticiper les collisions, et donc à pouvoir adapter le comportement en amont, permettant ainsi de les éviter, ou, à minima, d'en minimiser l'impact.

2.1.2.2.1 Percevoir l'environnement

Pour parvenir à de tels résultats, la première des conditions est de fournir au système un moyen de percevoir la scène. Historiquement, la stratégie la plus utilisée consistait à équiper le robot de capteurs de proximité afin de détecter en amont la survenue d'une collision. Cette stratégie a ainsi été étudiée pour différents types de technologies (infrarouge (Cheung et Lumelsky 1989), capacitifs (Yong Yu et Gupta 1999), (Hoffmann et al. 2016), ultrason (Llata et al. 1998) ou laser (Yong Yu et Gupta 1999)). Bien qu'incapables de capturer l'ensemble de la scène, ces solutions offrent l'avantage de ne subir aucun problème d'occultations. Elles présentent néanmoins l'inconvénient de ne capturer qu'une portion très réduite de l'espace de travail, et sont donc limitées dans un contexte de collaboration Homme/robot.

Dans le cas où l'espace de travail est immobile, c'est-à-dire où le robot n'est pas équipé d'une base mobile, une seconde stratégie plus récente (Svenstrup et al. 2009), (Otani, Bouyarmane, et Ivaldi 2018) consiste à utiliser les outils de capture de mouvement (motion capture). L'idée est alors d'équiper les opérateurs d'un ensemble de balises qui sont ensuite détectées par des capteurs observant la scène, permettant ainsi une localisation précise et une lecture temps-réel des mouvements réalisés. Ces approches sont néanmoins contraignantes, car elles nécessitent un équipement spécifique pour les opérateurs.

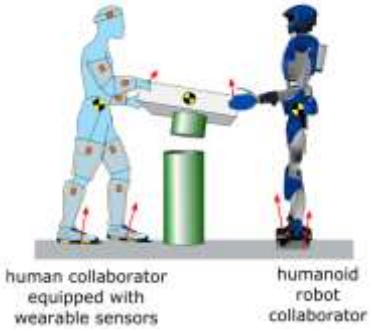


Figure 13 : Exemple de collaboration homme robot basé sur le principe de la motion capture. Extrait de (Otani, Bouyarmane, et Ivaldi 2018).

La dernière stratégie, qui est également la plus prometteuse (et donc la plus explorée) repose sur l'utilisation de capteurs observant la scène. Fixes dans l'environnement, ils n'imposent aucune contrainte aux opérateurs (bien souvent peu sensibles aux problématiques technologiques liées à leur métier). Notons toutefois que ce type d'approches souffre des problèmes d'occultations, ce qui impose dans la plupart des cas de multiplier les modalités de perception et les points de vue. L'écrasante majorité des travaux s'articule autour de trois grandes familles de capteurs, souvent couplées pour combiner leurs avantages : les caméras couleurs, les capteurs télémétriques (LiDaR multi-couches, caméras Time Of Flight, …) et les caméras RGB-D. (Henrich et Kuhn 2006)

utilisent par exemple une caméra couleur pour identifier et surveiller la zone de fonctionnement du robot et ainsi l'arrêter avant une collision. (Puls, Graf, et Wr 2012) utilisent une caméra time-of-flight pour percevoir le positionnement de l'humain et de ses bras afin de mettre à jour sa modélisation dans leur Framework de contrôle. Enfin, (Flacco et al. 2012) se servent d'une camera RGB-D pour déterminer en temps réel la distance séparant des points de contrôle répartis sur le robot des obstacles.

Notons pour finir que plusieurs travaux se sont également intéressés au placement optimal de ces capteurs, avec pour objectif de minimiser les occultations. On citera par exemple les travaux de (Flacco et Luca 2010) ou encore (Ceriani et al. 2013).

2.1.2.2.2 Evaluer le risque

Plusieurs travaux ont cherché à évaluer le risque encouru par un humain en cas de contact avec un robot. Par exemple, (Burgard, Brock, et Stachniss 2008) appliquent les méthodes des crashs test automobile à l'évaluation des conséquences d'une collision homme-robot. (Povse et al. 2010) proposent de tester la peine ressentie en appliquant les chocs sur un dispositif mécanique après que celui-ci ait été étalonné grâce au ressentie de volontaires. (Haddadin et al. 2012) quantifient les blessures provoquées par un robot en fonction des outils manipulés et des vitesses d'impact. (Park et Song 2009) dressent un tableau des limites de force acceptable pour diverses parties du corps notamment la tête et le cou. Cette classification des limites acceptables par partie du corps est d'ailleurs reprise par la norme ISO/TS 15066 (AFNOR 2016). Enfin, les travaux récents de (Fischer 2018) proposent une analyse de risque en temps réel pour estimer les conséquences d'un éventuel choc à venir.

2.2 Planification de chemins

2.2.1 Introduction

Comme nous venons de le voir, la voie royale pour sécuriser les interactions Homme-robot passe par l'évitement des collisions (non désirées). Pour autant, le robot a une tâche à effectuer, et doit nécessairement se déplacer pour y parvenir. L'une des thématiques fondamentales du domaine (qui est d'ailleurs la principale des problématiques abordées lors de cette thèse) est ainsi la planification de chemin, sur lequel nous allons maintenant nous pencher. Bien sûr, cette thématique n'est pas propre au contexte collaboratif, et a donc fait l'objet d'une littérature très riche qui s'étale sur plusieurs décennies. Pour un tour d'horizon général de ces travaux, nous proposons au lecteur l'excellent livre de Lavalle (LaValle 2006) qui dédie plus de 1000 pages aux algorithmes de planification (au sens large).

L'environnement dans lequel évolue le robot est appelé espace de travail, noté W. Il comprend le robot lui-même, mais aussi les obstacles qu'il doit éviter (notamment

humains dans notre contexte), et les objets avec lesquels il doit interagir. En tant que tel, l'espace de travail est rarement directement utilisé pour la planification de chemin. En effet, un robot étant composé de plusieurs corps rigides, déterminer si une position de ce dernier est réalisable implique de vérifier pour chacun d'entre eux si la portion de l'espace qu'ils occupent est libre. Pour pallier ce problème, Lozano-Pérez introduit dans (Lozano-Perez 1983) la notion de configuration q, qui décrit entièrement une posture du robot via le positionnement de chacune de ses articulations. L'espace correspondant à l'ensemble des configurations réalisables d'un robot est ainsi appelé *espace des configurations* C. Traditionnellement, cet espace est décomposé en deux sous-espaces :

- L'espace libre C_{free}, correspondant à l'ensemble des configurations libres de collision dans l'espace de travail.
- Son complément C_{obs} correspondant à l'ensemble des configurations induisant au moins un point de contact entre l'un des corps du robot, et un obstacle de l'environnement.

Le problème de la planification de chemin vise ainsi à établir dans l'espace des configurations, une trajectoire allant d'une configuration initiale du robot q_I jusqu'à une configuration finale q_F servant d'objectif, et ce, sans jamais passer par C_{obs} (cf. Figure 14). Pour résoudre cette problématique, plusieurs grandes stratégies ont été élaborées dans la littérature.

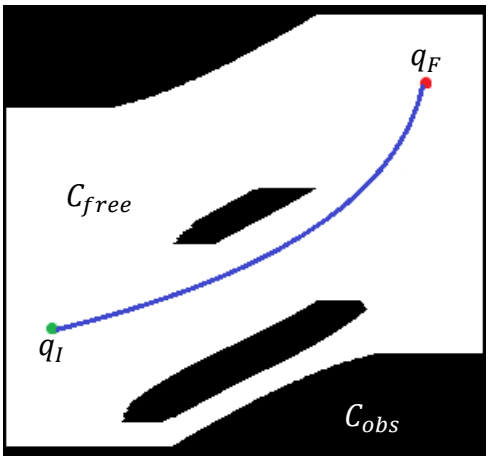


Figure 14 : Principe de la planification de chemin dans l'espace des configurations.

2.2.2 Probability Road Maps

La première, connue sous l'acronyme PRM (pour Probability Road Map) a été introduite par (Kavraki et al. 1996) et (Svestka et Overmars 1997). Elle est basée sur une phase d'apprentissage hors ligne où la connexité de l'environnement est modélisée par un graphe construit incrémentalement (cf. Figure 15). L'idée est de choisir plus ou moins aléatoirement un ensemble de configurations, et de ne conserver que celles appartenant

à l'espace libre C_{free}. Une fois cette étape réalisée, les configurations restantes sont connectées par des relations d'adjacence, à condition que le chemin les connectant appartienne lui aussi à C_{free}. Le résultat de cette phase d'apprentissage est un graphe utilisable ensuite en ligne pour trouver rapidement un chemin connectant la configuration courante du robot q_I et à la configuration q_F qu'il souhaite atteindre. Il suffit en effet de connecter q_I et q_F au graphe, puis d'appliquer des algorithmes de plus court chemin (A*, Dijkstra, …) pour trouver rapidement un chemin réalisable par le robot.

Ce type d'approches présente toutefois deux limites majeures. D'une part, la phase d'apprentissage nécessite de considérer un environnement globalement statique, faute de quoi le graphe construit n'est plus du tout représentatif de la réalité de l'espace de travail en cours de mission. Notons tout de même que certains travaux (Leonard Jaillet et Siméon 2004), (J. van den Berg, Ferguson, et Kuffner 2006) proposent des stratégies pour rendre les PRM utilisables en environnements dynamiques.

D'autre part, du fait de son côté aléatoire, ce type d'algorithmes ne possède pas la propriété de complétude, c'est-à-dire qu'il ne garantit pas de trouver un chemin connectant q_I et q_F s'il existe (à l'inverse, il ne sera pas non plus capable d'indiquer avec certitude que le chemin désiré n'existe pas). Au mieux, comme prouvé dans (Hsu, Latombe, et Kurniawati 2006), les PRM sont probabilistiquement complets, c'est-à-dire qu'en augmentant le nombre de configurations considérées dans le graphe construit, on finit par converger vers la complétude.

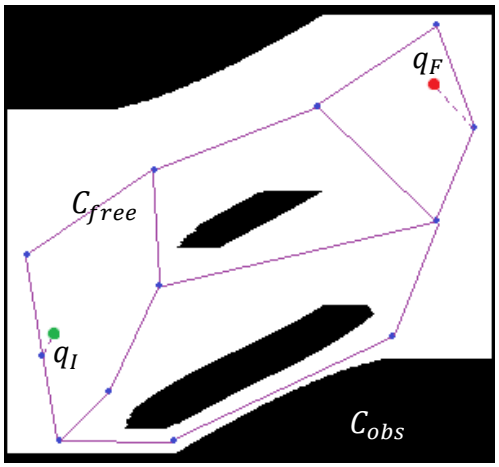


Figure 15 : Principe de fonctionnement des algorithmes Probality Road Map.

Malgré ces limites importantes, les PRM présentent l'énorme avantage d'être très rapides, et sont donc encore aujourd'hui utilisés dans des contextes temps-réel variés (Yan, Zhuang, et Xiao 2012) et pour des robots avec de nombreux degrés de liberté (Villumsen et Kristiansen 2017). De même, de nombreuses contributions ont porté sur des variantes plus performantes des PRM. Par exemple, dans (Amato et al. 1998) et (H.

Yeh et al. 2012), les configurations formant le graphe sont choisies de sorte à être à la frontière des obstacles, permettant ainsi de les longer, et donc de minimiser la distance parcourue. A l'inverse, dans (Lien, Thomas, et Amato 2003) et (H. Y. C. Yeh et al. 2014), les configurations sont choisies le plus loin possible des obstacles (plus spécifiquement sur l'axe médian, qui, comme nous le verrons par la suite, est central dans cette thèse), ce qui permet de maximiser la distance aux obstacles, donc la sécurité de la mission.

2.2.3 Rapidly-exploring Random Trees

La seconde grande famille d'algorithmes utilise le principe des RRT (Rapidly-exploring Random Trees), tel qu'initialement proposé par LaValle ((LaValle 1998),(LaValle et James J. Kuffner 2001)). Elle se démarque des PRM d'une part par le fait qu'elle ne nécessite pas d'étape hors-ligne, et d'autre part car elle ne vise pas à modéliser la connexité de l'ensemble de l'espace des configurations, mais seulement de la portion qui permet de connecter q_I à q_F. Plus spécifiquement, l'idée est de partir de l'une des deux configurations (ou les deux), et, par sauts successifs plus ou moins aléatoires (en fonction de la variante), de chercher à les connecter par un graphe qui s'utilisera exactement de la même façon que pour une PRM (cf. Figure 16). Tout le challenge est alors de proposer un processus d'expansion le plus efficace possible, de sorte à minimiser le temps requis pour planifier un chemin réalisable par le robot.

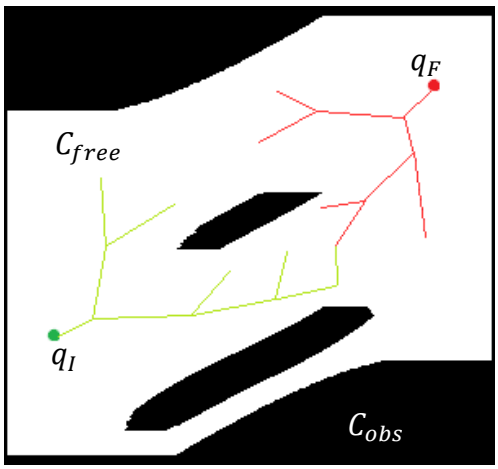


Figure 16 : Principe de fonctionnement des algorithmes Rapidly-exploring Random Trees.

Ainsi, dans la version originale de l'algorithme (LaValle 1998), l'expansion se fait par sauts successifs de taille constante, ce qui représente clairement l'approche la plus élémentaire, de nombreuses fois revisitée depuis. Par exemple, dans (L. Jaillet et al. 2005), les auteurs introduisent le ADD-RRT, où la taille des sauts est automatiquement adaptée en fonction des derniers sauts effectués. Dans le même ordre d'idée, (Knepper et Mason 2012) met à jour la direction et la taille des sauts en fonction des dernières

collisions détectées. Pour finir sur un dernier exemple corrélé à notre travail, il est bon de noter que l'axe médian a également été utilisé dans (Denny et al. 2014) pour guider la construction du RRT, permettant ainsi de maximiser la distance aux obstacles dans le choix de la trajectoire à réaliser.

Comme souligné dans (Broquère 2011), l'utilisation des RRT est aujourd'hui la stratégie la plus populaire pour la planification de chemin (Ellekilde et Petersen 2013), (Wei et Ren 2018), (Zhang et al. 2018). Ces algorithmes permettent en effet de planifier en temps-réel un chemin réalisable pour un robot possédant de nombreux degrés de liberté, tout en évoluant dans un environnement dynamique. Pour autant, les RRT ne sont pas sans défauts : ils permettent certes de générer des trajectoires réalisables en un temps réduit, mais ces dernières ne sont pas optimales. De plus, ces algorithmes ne garantissent que la complétude probabiliste, ce qui peut être une limite dans certains contextes.

2.2.4 Méthodes complètes

Les méthodes complètes, à l'inverse des méthodes basées sur l'échantillonnage, raisonnent sur l'ensemble de l'espace des configurations. Elles ont l'avantage d'être complètes (de trouver une solution si elle existe ou d'affirmer qu'il n'y en a pas) mais présentent l'inconvénient d'être lourde en terme de calcul. Lavalle (LaValle 2006) affirme même que certaines sont inapplicables bien que théoriquement intéressantes.

La première catégorie de travaux repose sur l'hypothèse que les obstacles et le robot sont composés uniquement de formes polygonals. Il suffit alors de chercher pour chaque couple d'arrêtes (robot-obstacle) les conditions d'intersection et d'en déduire ainsi la partie occupée de l'espace des configurations. Par exemple, (Chazelle 1987) déduit grace à cette décomposition, les cellules d'espace libre dans lesquelles peut naviguer le robot.

La deuxième catégorie est celle dans laquelle s'inscrit notre travail, est la décomposition cellulaire de l'espace des configurations. (Gonzalez, Kloetzer, et Mahulea 2017) proposent un comparatif des trajectoires obtenues en fonction du type de décomposition et notamment la décomposition en cellules trapézoïdales telles que présenté dans (M. de Berg et al. 2008) ou rectangulaires (Kloetzer et Ghita 2011). Il apparait que le temps de calcul pour des environnements 2D contenant une dizaine d'obstacles est proche de la seconde pour les meilleurs. Ce qui les rend peu compatibles avec la planification en environnements dynamiques. Notons que pour tous ces algorithmes l'espace des configurations est supposé connue.

2.2.5 Le cas de la collaboration Homme/robot

La planification de chemin dans le cas particulier de l'interaction homme robot nécessite des dispositions particulières pour assurer la sécurité des humains. (Sisbot et al. 2007), s'intéressent à la robotique de service et distinguent deux catégories : dans la première,

le planificateur ne considère que la base mobile pour la navigation du robot, alors que dans la seconde, l'ensemble des degrés de liberté sont pris en compte pour permettre la manipulation d'objets.

La plupart des travaux prennent en compte le positionnement de l'humain pour déplacer le robot. Par exemple Yoshimi et al. proposent une méthode de suivi de personne (Yoshimi et al. 2006). (Pacchierotti, Christensen, et Jensfelt 2006) proposent à l'inverse d'eviter l'humain se trouvant sur le passage du robot et (Vasquez et al. 2012) proposent des méthodes de navigation pour des fauteuils roulants autonomes en prenant en compte un espace personnel autour des humains à respecter.

Le cas de la manipulation qui est plus proche de notre application prend en compte l'ensemble du robot. Plusieurs travaux ont proposé de minimiser le coût des critères de danger introduits par (Ikuta, Ishii, et Nokata 2003) tout au long de la trajectoire du robot, d'abord (Nokata, Ikuta, et Ishii 2002) qui considèrent uniquement l'effecteur puis (Dana Kulic et Croft 2005) qui prennent en compte le modèle complet du robot. Siméon et al. présentent le planificateur Move3D basé sur des méthodes probabilistes appliquées sur une carte de coûts pour l'ensemble des degrés de liberté du robot (Simeon, Laumond, et Lamiraux 2001). (L. Jaillet, Cortes, et Simeon 2008) appliquent une méthode RRT sur ces mêmes cartes de coût.

Bien que la prise en compte de l'humain dans les travaux de planification semble plus le parti pris de la robotique sociale que de la robotique industrielle qui préfère des méthode réactives (Flacco et al. 2012), les principes de planification et notamment le respect d'une distance de sécurité restent applicables à tous les cas de collaboration Homme-robot.

Chapitre 3.
Perception et représentation de l'espace

Synthèse :

Dans ce chapitre, nous avons présenté, dans un premier temps, les capteurs que nous avons choisis pour percevoir l'environnement, ainsi que les méthodes de placement des capteurs et de leur calibration permettant une couverture adéquate de l'espace de travail. Nous avons ensuite explicité la construction d'une carte d'occupation 3D basée sur le parcours unique de chaque cellule. Nous avons introduit une sémantique dans la grille d'occupation nous permettant de différencier la nature des obstacles. Dans la dernière partie, nous avons montré comment construire en ligne l'espace des configurations d'un robot. Nous avons utilisé pour cela une table de valeurs pré-calculée, permettant notamment de projeter l'information sémantique de type d'obstacles dans l'espace des configurations. Enfin, nous avons présenté quelques exemples de projections et avons explicité la représentation de l'espace des configurations pour ces cas d'étude.

Sommaire du chapitre :

3.1 Introduction

La première tâche à considérer est la perception de l'environnement. Celle-ci est primordiale pour toute application de collaboration Homme-robot. En effet, le robot doit non seulement percevoir les humains pour adapter son comportement à leur présence, mais aussi maitriser le reste de la scène pour planifier les trajectoires qui lui permettront d'atteindre ses objectifs. Nous l'avons vu dans le Chapitre 1, le contexte industriel nous impose certaines hypothèses sur l'environnement de travail :

- La scène est hautement dynamique. La perception et les traitements permettant de construire les représentations doivent se faire en un temps réduit pour garantir la réactivité de la chaine complète.
- La scène est complexe puisque composée de différents outils de production, la multiplicité en nombre et en genre des capteurs est nécessaire pour obtenir une couverture complète.
- La scène est collaborative. La présence d'humains (un ou plusieurs) doit être prise en compte et intégrée dans la représentation.
- La finalité de la représentation est de permettre une planification de trajectoires pour que le robot effectue ses missions en toute sécurité.

Nous proposons pour cela la création dans un premier temps, d'une carte d'occupation de l'espace dans laquelle chaque cellule obstacle est labélisée selon le type d'obstacle qu'elle contient. Nous projetons ensuite l'ensemble des obstacles dans l'espace des configurations. Cet espace a été initialement utilisé pour la planification de trajectoires par (Lozano-Perez 1983). Il est défini d'après (LaValle 2006b) comme « l'ensemble des transformations possibles qui puissent être appliquées à un robot ». Cela signifie que cet espace représente toutes les configurations possibles d'un robot.

La Figure 17 présente le pipeline complet de la construction des représentations que nous utilisons. Ce chapitre détaille tour à tour chacune des étapes qui le composent. En premier, nous présentons le choix des capteurs et leur positionnement, puis nous explicitons la construction de la carte d'occupation et enfin nous détaillons la construction de l'espace des configurations. De plus, nous donnons à la fin du chapitre des exemples d'espaces des configurations tirés de nos expérimentations.

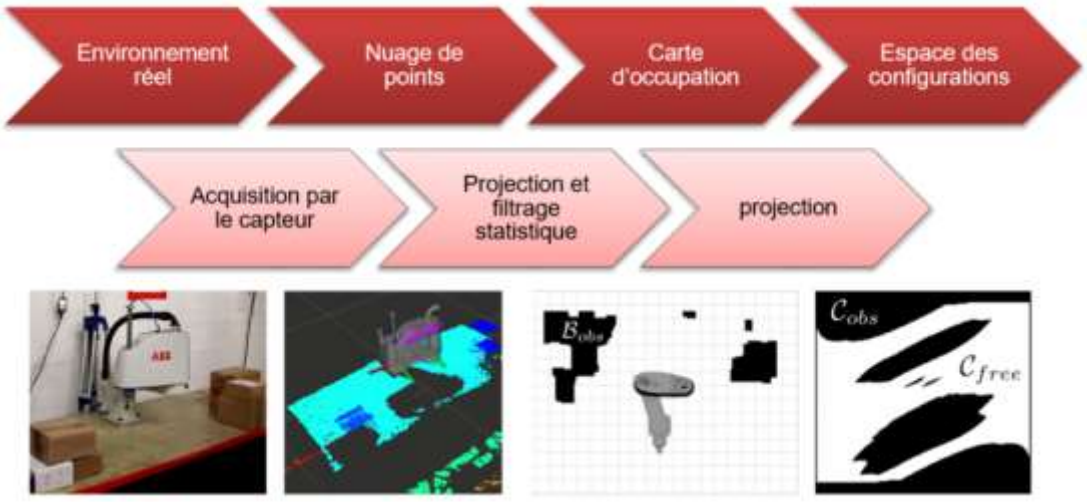


Figure 17 : Pipeline de la construction de la représentation de l'espace.

3.2 Capteurs pour l'acquision de l'espace de travail

Pour pouvoir construire une représentation fidèle de l'espace de travail, nous devons l'observer avec un ensemble de capteurs. Nous avons testé plusieurs capteurs dont voici une rapide présentation :

3.2.1 LiDAR 2D

Le premier capteur que nous utilisons est un LiDAR (light detection and ranging). Ce capteur permet d'acquérir la distance aux obstacles dans un plan 2D. Il est très utilisé dans la littérature, que ce soit pour la détection de personnes ou la navigation de robots mobiles (Ben-Said, Stéphant, et Labbani-Igbida 2016). Ce capteur est fait d'un télémètre LASER monté sur une base rotative et renvoie pour chaque point détecté (θ_P,ρ_P), ses coordonnées polaires dans le repère du Lidar. En notant $TH_{monde,lidar}$ la matrice de transformation homogène donnant la pose du lidar dans le monde, la position d'un point détecté dans le monde devient alors :

$$Pose_{P,monde} = TH_{monde,lidar} * \begin{Bmatrix} \rho_P * cos(\theta_P) \\ \rho_P * sin(\theta_P) \\ 0 \\ 1 \end{Bmatrix} \quad (3\text{-}1)$$

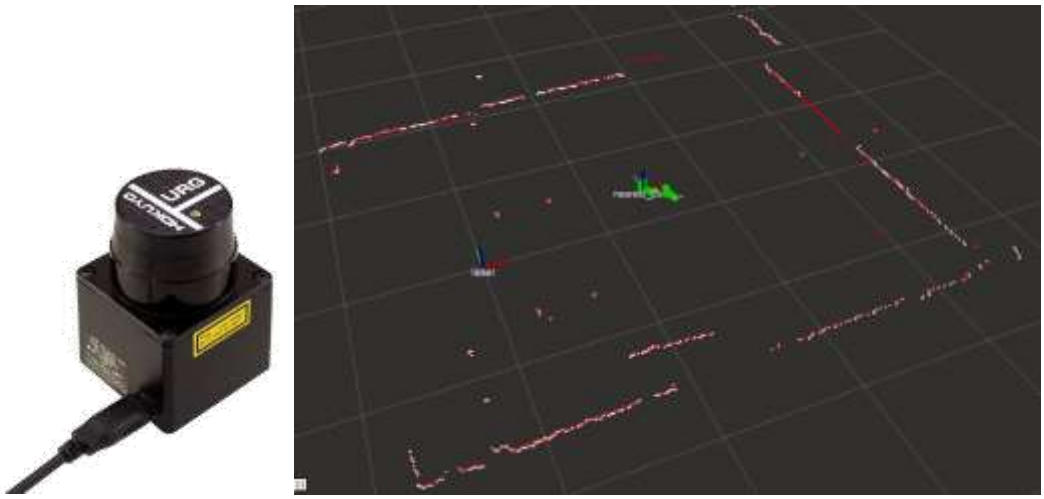


Figure 18 : Capteur LIDAR et exemple de nappe de points obtenue.

3.2.2 Caméra 3D time of flight

Le second capteur utilisé est une caméra 3D, il s'agit d'une caméra utilisant la mesure du temps de vol d'une onde lumineuse pour déterminer la distance des objets vus par chacun des pixels. Elle offre une meilleure tolérance aux variations de lumière ambiante qu'une caméra RGB-D à lumière structurée tel que la Kinect. La caméra nous renvoie pour chaque pixel, ses coordonnées X_P,Y_P,Z_P ainsi qu'une valeur de luminosité. De la même façon, $TH_{monde,cam3D}$ étant la pose de la caméra dans le monde, la position d'un point obstacle vu est, dans le monde :

$$Pose_{P,monde} = TH_{monde,cam3D} * \begin{Bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{Bmatrix} \quad (3\text{-}2)$$

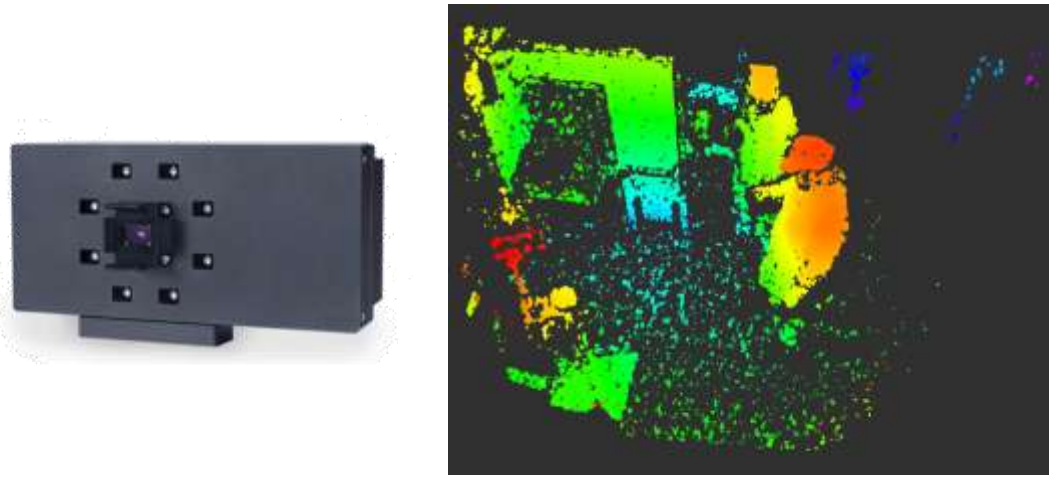


Figure 19 : Camera Time Of Flight et exemple de nuage de points obtenu

3.2.3 Caméra RGB

Une des faiblesses de la caméra 3D time of flight relativement aux caméras RGB-D est l'absence de capteur couleur. Pour pallier ce manque, nous avons ajouté des caméras

RGB permettant de coloriser notre carte d'occupation. Le modèle de sténopé relie un point dans le monde à sa projection sur le capteur (su,sv).

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_u & 0 \\ 0 & f_y & c_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} * TH_{cam,monde} * \text{Pose}_{\text{P,monde}} \quad (3\text{-}3)$$

Avec f_x, f_y, c_u, c_v les paramètres intrinsèques de la caméra et $TH_{cam,monde}$ les paramètres extrinsèques.

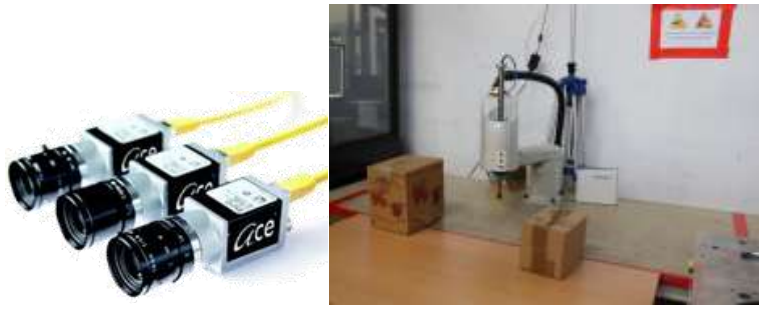


Figure 20 : caméras RGB et exemple d'image couleur

3.2.4 Caméra thermique

Afin de faciliter la détection des humains dans notre scène de travail, nous avons testé l'usage d'une caméra thermique. Celle-ci renvoie pour chaque pixel de l'image une valeur proportionnelle à sa température. Les humains étant à une température connue et constante, nous voulions les identifier grâce à cette propriété. Nous avons testé cette hypothèse sur le site de Valeo Limoges. Le process de fabrication mis en œuvre à Limoges impliquant de nombreuses étapes de cuissons, de nombreux éléments des scènes de travail observées sont à des températures proches des humains ou supérieures et il était donc difficile de discriminer les humains par l'intermédiaire des caméras thermiques (cf. Figure 21). Compte tenu des contraintes de l'application, nous n'avons pas utilisé ce capteur dans la suite de ce travail.

Figure 21 : image d'une caméra thermique en situation

3.2.5 Positionnement des capteurs

Le positionnement des capteurs dans l'environnement de travail est effectué en fonction des contraintes de l'environnement. Bien que des méthodes de placement optimal aient été mises au point telles que dans (Flacco et Luca 2010) pour la collaboration Homme robot, leur mise en application dans notre contexte est entravée par la complexité de l'environnement. La mise en place de ces capteurs est faite selon les points suivants :

- Un capteur Lidar est placé à une hauteur permettant la détection des jambes (20 à 30cm du sol) dans un plan parallèle au sol et le plus proche possible du robot pour couvrir l'espace de travail collaboratif.
- Deux caméras 3D sont placées en hauteur, inclinées vers l'espace de travail et si possible en arrière du robot de façon à observer la zone collaborative. Si plusieurs caméras sont utilisées, elles sont disposées de part et d'autre du robot pour permettre un point de vue redondant sur cette zone et éviter les phénomènes de masquage.
- Des caméras couleurs sont juxtaposées aux caméras 3D pour partager le même point de vue, facilitant la colorisation des nuages de points.

3.2.6 Calibration des capteurs de l'environnement de travail

La calibration de l'ensemble du dispositif d'acquisition nécessite une attention particulière pour permettre une fusion précise des données issues des capteurs. Nous avons mis au point une méthode de calibration à l'aide d'une mire 3D.

La procédure de calibration se déroule en plusieurs étapes :

- Calibration des paramètres intrinsèques des caméras couleur et de profondeur.
- Déterminer les paramètres extrinsèques des caméras.
- Déterminer la position du Lidar

Voici le détail de chacune des étapes :

La première étape consiste à calibrer la caméra couleur afin d'obtenir ses paramètres intrinsèques (f_x,f_y les distances focales et c_u,c_v les coordonnées de la projection du centre optique) du modèle de sténopé (éq. (3-3)) en utilisant les outils de calibration disponibles par exemple dans OpenCV.

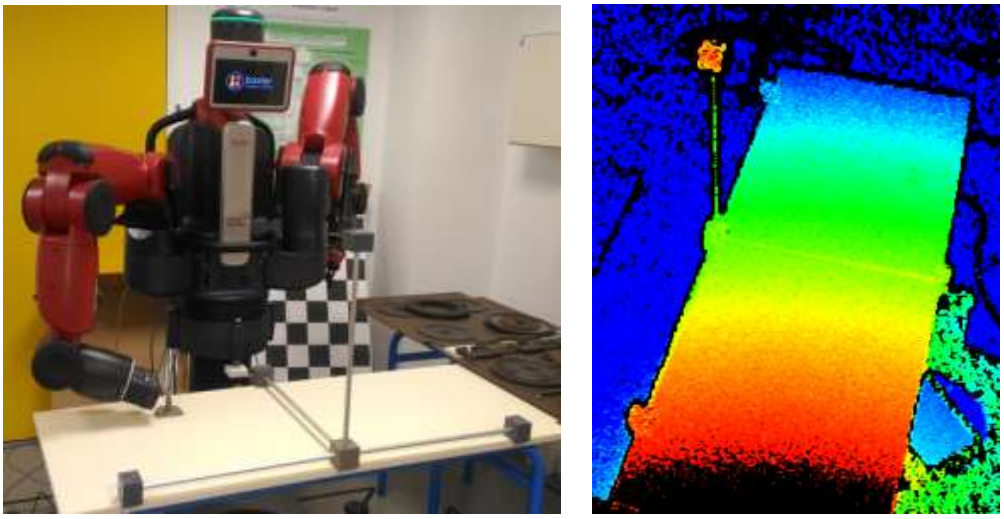
La position relative de la caméra couleur par rapport à la caméra de profondeur peut être déterminée sur les plans des caméras. En effet les caméras étant physiquement reliées l'une à l'autre par une pièce conçue par nos soins, la transformation entre les deux dispositifs est connue.

Dans notre application, le monde est centré autour du robot. Les paramètres extrinsèques des caméras correspondent donc aux positions relatives entre les caméras et la base du

bras du robot. Nous déterminons ces transformations en passant par un repère intermédiaire matérialisé par une mire 3D (cf. Figure 22). L'utilisation d'une pointe de calibration est très courante avec les bras robots industriels et permet de définir des repères dans l'environnement du robot. En sélectionnant trois points à portée du robot nous pouvons définir un repère associé à la mire : le premier point (P_1) est l'origine du repère, le second pris dans la direction X permet de définir le vecteur $\vec{x}$ et le troisième pris dans le plan XY permet de déterminer d'abord le vecteur $\vec{z} = \vec{x} \wedge \overrightarrow{xy}$ puis $\vec{y} = \vec{z} \wedge \vec{x}$. La matrice de transformation entre le robot et la mire est :

$$TH_{robot,mire} = \begin{pmatrix} \overrightarrow{x_x} & \overrightarrow{y_x} & \overrightarrow{z_x} & P_{1,x} \\ \overrightarrow{x_y} & \overrightarrow{y_y} & \overrightarrow{z_y} & P_{1,y} \\ \overrightarrow{x_z} & \vec{y}_z & \vec{z}_z & P_{1,z} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}4)$$

La mire est de dimension suffisante pour être visible par la caméra 3D et nous pouvons déterminer dans le nuage de points, les positions des points particuliers que sont l'origine, la direction X et la direction Y. De la même façon nous obtenons $TH_{caméra,mire}$.



a) b)

Figure 22 : Calibration de l'espace de travail : (a) la mire de calibration 3D est posée devant le robot Baxter dont le bras droit porte la pointe et (b) visualisation de la mire 3D dans le nuage de point de la caméra.

La position relative entre les caméras et le robot est obtenue par l'inversion de la matrice $TH_{caméra,mire}$ et une multiplication de matrices :

$$TH_{robot,caméra} = TH_{robot,mire} * TH_{caméra,mire}^{-1} \quad (3\text{-}5)$$

La dernière étape consiste à déterminer la position du Lidar dans le repère du robot, pour cela nous utilisons la pointe de calibration du robot pour "couper" la nappe LIDAR en

trois points. L'utilisation de la pointe permet de déterminer le point exact où celle-ci est à l'altitude du Lidar. Grâce aux trois points, nous pouvons calculer la transformation entre le robot et le Lidar en suivant la même méthode.

3.3 Cartographie de l'espace de travail

L'ensemble des données issues des capteurs doit être traité et fusionné. Pour cela, nous utilisons une carte d'occupation. Celle-ci permet de représenter l'espace de travail comme un ensemble de cellules, lesquelles sont mises à jour par les données issues des capteurs dès qu'une nouvelle acquisition est faite.

3.3.1 Les cartes d'occupation

Les cartes d'occupation ont été introduites par (Moravec et Elfes 1985) dans les années 80 pour la robotique mobile. Il s'agit d'une approximation discrète de l'environnement par une grille de cellules. A chacune de ces cellules est associée une probabilité de présence d'un obstacle. La probabilité d'occupation d'une cellule est mise à jour en connaissant la position du capteur sur la carte, sa mesure et la modélisation de cette mesure. Les cartes d'occupation ont ensuite évolué en ajoutant des informations importantes pour la navigation des robots, comme par exemple l'élévation du terrain. Parmi les plus utilisées aujourd'hui, nous pouvons citer Octomap présenté dans (Wurm et al. 2010) qui est une carte d'occupation 3D basée sur le principe des octree pour économiser de l'espace mémoire. Le framework d'Octomap est aujourd'hui très complet et permet par exemple la mise à jour de la carte par l'intermédiaire d'un nuage de points. C'est notamment l'outil qui est utilisé par MoveIt (l'outil de planification de trajectoire intégré à ROS) pour construire l'environnement à partir de données capteurs, notamment les cameras 3D. Cependant, pour notre application, il est nécessaire de pouvoir connaitre non seulement la probabilité de présence d'un obstacle mais aussi le type d'obstacle (humain ou matériel). De plus, l'apport de l'Octree est particulièrement important dans le cas de cartes de grande dimension alors que notre carte est restreinte à l'espace de travail collaboratif et varie avec une dynamique importante. Ainsi, nous avons programmé notre propre carte d'occupation.

3.3.2 Construction d'une carte 2D

Nous avons mis en place une carte d'occupation en 2D pour nos premières expérimentations réalisées avec un robot plan. Celle-ci est visible sur la Figure 17. La mise à jour est réalisée à l'aide du nuage de points issu d'une caméra 3D. Pour chaque point du nuage de points PC_i, nous déterminons la cellule B_{xy} sur laquelle il se projette verticalement, puis nous mettons à jour la probabilité d'obstacle à l'instant t, $p_{obs,t}$, dans cette cellule selon l'altitude du point. En fixant Z_{limit} l'altitude au-delà de laquelle on considère un objet comme étant un obstacle au robot, nous obtenons :

- Si $Z_{PC_i} < Z_{limit}$:

$$p_{obs,t}(B_{xy}) = \max(0, p_{obs,t-1}(B_{xy}) - Inc_{proba}) \quad (3\text{-}6)$$

- Sinon ($Z_{PC_i} \geq Z_{limit}$) :

$$p_{obs,t}(B_{xy}) = \min(1, p_{obs,t-1}(B_{xy}) + Inc_{proba}) \quad (3\text{-}7)$$

Avec Inc_{proba}, l'incrément de variation de la probabilité d'occupation.

3.3.3 Construction d'une carte en 3D

Nous avons ensuite construit une carte d'occupation en 3D bien plus complète qu'une projection sur une carte 2D. Nous avons développé une structure pour contenir les informations nécessaires à notre application.

3.3.3.1 Caractéristiques de notre carte

Notre carte d'occupation est constituée de cellules cubiques. Chacune de ces cellules est porteuse de quatre informations : une probabilité d'obstacle, une probabilité de connaissance, une information photométrique (couleur) ainsi qu'un label définissant le type d'obstacle.

La probabilité d'obstacle comme dans le paragraphe précédent est la croyance en la présence d'un obstacle dans cette cellule. La probabilité de connaissance correspond à la croyance de la justesse de notre information d'occupation. En effet, dans un environnement de production, il peut y avoir des zones invisibles des capteurs à cause d'un obstacle, du robot ou de l'humain. Dans ces zones d'ombre, il est impossible de mettre à jour la probabilité d'obstacle par une mesure, nous exprimons ce vieillissement de l'information en réduisant la probabilité de connaissance avec le temps. Cette probabilité est remise au maximum dès lors qu'une mise à jour est possible.

L'information photométrique donne la couleur de l'objet occupant la cellule.

La labélisation des cellules a pour but de les distinguer et de les traiter en fonction de leurs caractéristiques (cf. Figure 23 & Figure 24). Les différents labels utilisés sont les suivants :

- *Inconnue* : ce sont les cellules dont la probabilité de connaissance est plus faible que le seuil minimum de connaissance.
- *Libre* : ce sont les cellules dans lesquelles il n'y a pas d'obstacles, leur probabilité d'obstacle est inférieure au seuil défini pour l'absence d'obstacle et leur probabilité de connaissance est suffisamment élevée.
- *Robot* : ce sont les cellules occupées par le robot, leur probabilité d'obstacle est nulle (sinon le robot serait dans l'obstacle) et leur probabilité de connaissance est maximum (la position du robot est donnée par ses capteurs internes).

- *Fond* : ce sont les cellules occupées par des obstacles immuables, par exemple le sol ou des éléments de machines immobiles, ces objets étant statiques, leurs probabilités d'obstacle et de connaissance sont maximum et elles ne seront pas mise à jour. Ces obstacles sont définis à l'avance.
- *Humain* : ce sont les cellules occupées par des humains, leur probabilité de connaissance est suffisante, leur probabilité d'obstacle est supérieure au seuil défini pour la présence obstacle et le processus de mise à jour les a labélisées comme tel.
- *Obstacle* : ce sont les autres cellules dont la probabilité de connaissance est suffisante et la probabilité d'obstacle est supérieure au seuil défini. Elles correspondent à tous les autres obstacles (pièces de machine en mouvement, autre robot, chariot, boite et autres).

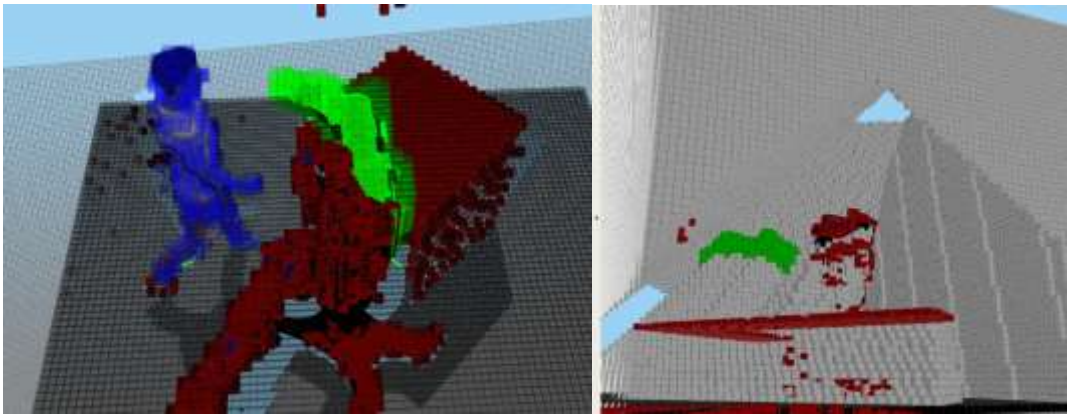


Figure 23 : Carte d'occupation de l'espace de travail avec les différents labels de cellules. En noir les cellules "fond", en rouge les cellules "obstacles", en vert les cellules "robot", en bleu les cellules "humain" et en blanc les cellules « inconnues ». Les cellules "libre" sont transparentes et pour un souci de clarté les cellules « inconnue » ne sont pas représentées sur l'image de gauche.

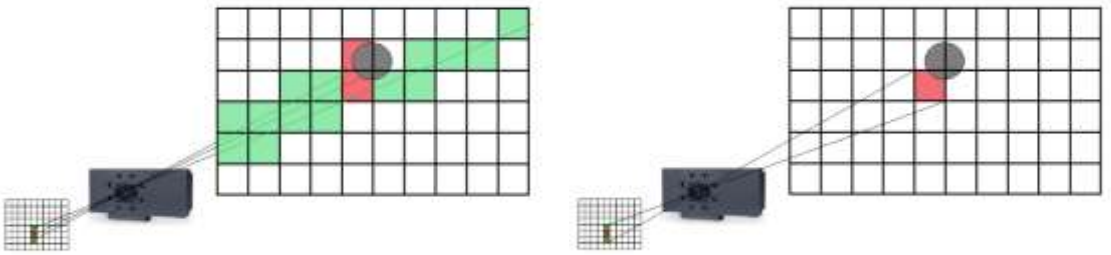
3.3.3.2 Mise à jour de la grille d'occupation à partir des données capteurs

La mise à jour de la carte d'occupation est faite grâce aux données issues des capteurs. La fusion des données est réalisée lors de ce processus. Notre choix de capteurs permet de regrouper les informations en deux catégories, les données de mesure d'un point 3D détecté (issues d'un lidar ou d'une caméra 3D) et les données colorimétriques (données par une caméra RGB).

La méthode classique pour traiter les données issues d'un lidar ou d'une caméra 3D est d'utiliser la technique du lancer de rayon (cf. Figure 24). Celle-ci consiste à parcourir pour chaque point donné par le capteur le trajet reliant le capteur au point obstacle détecté ou à la limite de détection. Il faut tout d'abord déterminer la première cellule de la carte parcourue par le rayon, puis calculer la cellule suivante pour avancer de proche en proche jusqu'à atteindre le point obstacle ou l'extrémité de la carte d'occupation. La

probabilité d'obstacle de la case occupée par le point obstacle est augmentée, tandis que celle des cases ayant été traversées par le rayon est diminuée, la probabilité de connaissance de ces cellules est mise au maximum. Cette méthode est utilisée par beaucoup d'algorithmes de carte d'occupation (OctoMap par exemple). Plusieurs problèmes sont inhérents à cette méthode notamment une incertitude dans les zones frontalières : lorsqu'une cellule proche d'un obstacle est traversée par plusieurs rayons, ces rayons peuvent être porteurs d'informations contraires sur la présence d'obstacles dans cette cellule et la probabilité de présence d'un obstacle peut être fluctuante. Ce problème peut être résolu simplement en verrouillant la mise à jour d'une cellule occupée comme dans (Hornung et al. 2013), mais il met en avant l'inconvénient de cette méthode de parcourir plusieurs fois la carte. De plus, cela est relativement coûteux en temps de calcul.

Nous proposons donc une approche différente basée sur le parcours de la carte d'occupation et la mise à jour en une fois de chacune des cellules au lieu de considérer chaque rayon. Nous parcourons l'ensemble des cellules et déterminons les rayons qui la traversent pour déduire l'état de la cellule (cf. Figure 24). Voici les étapes de notre algorithme :



a) b)

Figure 24 : Méthodes de mise à jour de la carte d'occupation. a) par lancer de rayon. b) notre méthode.

a. Nous commençons par convertir le nuage de points issu des caméras 3D. L'information donnée est de type XYZ, il s'agit des coordonnées cartésiennes des points détectés dans le repère du capteur. Nous les convertissons en coordonnées sphériques dans ce même repère. Nous ne conservons que le paramètre ρ, la distance entre le repère et le point obstacle, en effet les angles θ et ϕ sont constants pour chaque pixel de la caméra. Nous pouvons ainsi calculer la projection d'un point du monde sur le capteur grâce aux paramètre de calibration et à l'équation (3-3).

$$u = \frac{su}{s} = f_x * \frac{X}{Z} + c_u$$
$$v = \frac{sv}{s} = f_y * \frac{Y}{Z} + c_v \quad (3\text{-}8)$$

Or :

$$\rho = \sqrt{X^2 + Y^2 + Z^2}$$
$$\tan(\theta) = \frac{X}{Y} \quad (3\text{-}9)$$
$$\frac{\tan\phi}{\cos\theta} = \frac{Y}{Z}$$

D'où :

$$u = f_x * \tan(\theta) + c_u$$
$$v = f_y * \frac{\tan(\phi)}{\cos(\theta)} + c_v \quad (3\text{-}10)$$

b. Nous parcourons ensuite l'ensemble de la carte d'occupation et pour chaque cellule, nous calculons sa position relative au capteur puis nous en déduisons la position u, v du centre de la cellule sur le capteur grâce à l'équation (3-10). Nous calculons ensuite l'étalement (δu,δv) de la cellule sur le capteur (c.a.d. le nombre de pixels sur lesquels est projetée la cellule) à partir de la taille de la cellule. La projection de la cellule sur le plan image est approximée par un cercle de rayon r_{xy}. Pour être pleinement conservateur, ce rayon peut être choisi comme étant celui de la sphère englobante du cube de la cellule de longueur d'arête a :

$$r_{xy} = \sqrt{3 * \left(\frac{a}{2}\right)^2} = \sqrt{3} * \frac{a}{2} \quad (3\text{-}11)$$

Cependant, cette valeur peut s'avérer problématique en générant de trop nombreux faux positifs, on peut donc la réduire. Néanmoins, nous ne prendrons pas un cercle de rayon inférieur à celui de la sphère inscrit dans le cube :

$$r_{xy,min} = a \quad (3\text{-}12)$$
$$\delta u = f_x * \frac{r_{xy}}{Z}$$
$$\delta v = f_y * \frac{r_{xy}}{Z} \quad (3\text{-}13)$$

c. Nous parcourons alors les pixels ainsi identifiés sur chacune des caméras 3D et les rayons lidar traversant la cellule. Si parmi l'ensemble des rayons considérés, un point est détecté à l'intérieur de la cellule, alors la probabilité d'obstacle est augmentée. Sinon et si un ou plusieurs rayons traversent la cellule, c'est-à-dire si un ρ est plus grand que la distance cellule-capteur, alors la probabilité d'obstacle

est diminuée. Dans ces deux cas, la probabilité de connaissance est mise au maximum. La troisième possibilité est qu'aucun rayon n'ait atteint ou traversé la cellule, dans ce cas, la probabilité d'obstacle ne change pas tandis que la probabilité de connaissance diminue. Si la cellule est considérée comme occupée à la fin de cette étape, sa couleur est donnée en déterminant la projection de la cellule sur le capteur de la caméra RGB la plus proche du capteur ayant vu la cellule obstacle.

3.3.3.3 Modélisation du robot

Notre méthode de mise à jour détaillée ci-dessus permet de labéliser les cellules selon trois catégories : *libre*, *obstacle* ou *inconnue* en plus des cellules *fond* immuables. Pour labéliser les cellules correspondant au robot, nous utilisons le modèle géométrique direct du robot, mis à jour par les valeurs de ses codeurs, pour le positionner dans l'espace. Puis nous déterminons les cellules que celui-ci occupe grâce à des volumes englobants (cf. Figure 25 & Annexe Annexe 2 :). Ces cellules seront mises à jour avec une probabilité de connaissance maximum, une probabilité d'obstacle nulle et le label ***robot***.

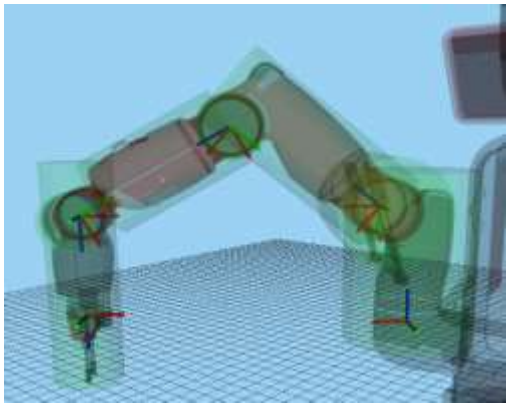


Figure 25 : Volume englobant le robot (en vert) permettant la labélisation des cellules robot.

3.3.3.4 Détection des personnes

La détection et la labélisation des humains dans la carte est une étape essentielle pour garantir l'interaction sans risque avec le robot. Les méthodes pour détecter les humains sont nombreuses, nous pouvons notamment citer la détection de piétons par traitement d'images grâce aux caméras RGB (Dollar et al. 2012), la segmentation des nuages de points telle que réalisée avec la Kinect (Shotton et al. 2011) ou la détection de jambes par un Lidar (Xavier et al. 2005). L'avantage de cette dernière méthode est de permettre une détection rapide et légère en termes de calculs mais, en revanche, elle ne permet de détecter qu'une petite partie de l'humain.

Nous avons donc basé notre détecteur d'humain sur un détecteur de jambes, à partir duquel nous initialisons un algorithme de labélisation de cellules. Celui-ci fonctionne de la manière suivante : le détecteur de jambes renvoie la position à laquelle l'humain se

trouve dans le plan. Nous labélisons *humain* les cellules se trouvant dans le plan du Lidar et dans un cercle restreint autour de la position donnée. Puis, nous appliquons ce label à toutes les cellules occupées contiguës et nous renouvelons cette étape jusqu'à ce que :

- Il n'y ait plus de cellules contiguës
  ou
- Les cellules contiguës soient en dehors du cylindre maximum atteignable par un humain.

Ce processus peut aussi être aidé par un test sur la couleur des cellules. En effet, si les opérateurs portent un uniforme d'une couleur distincte de l'environnement, un test permet d'éliminer des cellules qui ne correspondent pas à des humains.

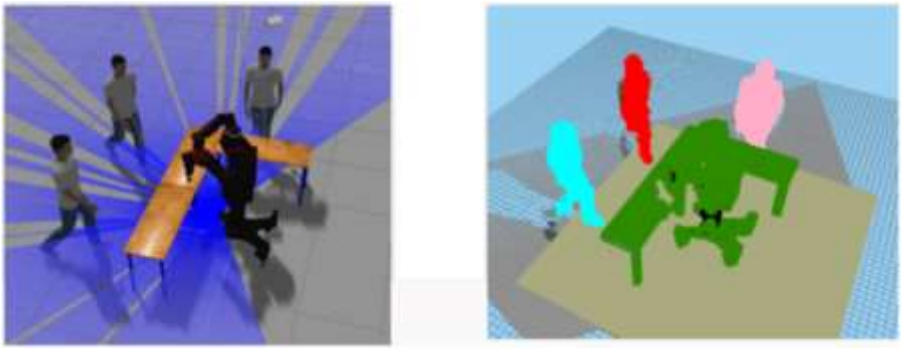


Figure 26 : Labélisation de cellules en humain dans notre carte d'occupation (à droite) à partir d'un environnement simulé sous Gazebo (à gauche).

La Figure 26 montre un exemple de labélisation d'humain dans un environnement simulé. Les humains en bleu clair, rouge et vert clair peuvent être différentiés puisque leur labélisation est faite à partir de trois positions données par le détecteur de jambes. Nous avons aussi tester notre algorithme en situation réelle. La Figure 27 montre des résultats de cette labélisation obtenus sur des ilots de cuisson situés dans l'usine Valeo de Limoges.

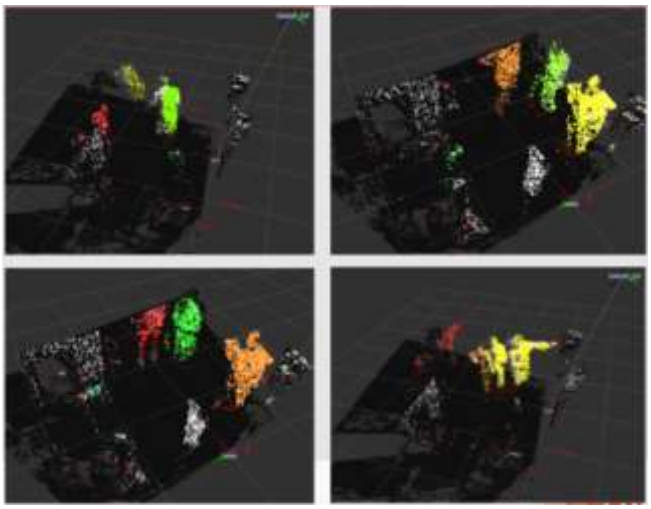


Figure 27 : Exemple de labélisation d'opérateur en situation réelle (ilots de cuisson Valeo).

3.3.4 Simplification en vue de la projection

Avant de projeter les obstacles dans une autre représentation, il est nécessaire de simplifier notre carte d'occupation qui contient six labels. En effet, l'ensemble de l'information contenue ne peut être projeté. Nous avons donc réparti les labels dans deux catégories : les cellules obstacles et les cellules non obstacles :

- Les cellules obstacles contiennent les labels *fond*, *obstacle*, *humain* et *inconnue*.
- Les cellules non obstacles contiennent les labels *libre* et *robot* (nous supposons que l'auto-collision est gérée par le contrôleur du robot).

Ce sont les cellules obstacles qui seront projetées pour construire la seconde représentation. La prise en compte des humains sera faite en projetant dans un second temps les cellules labélisées *humain*.

3.4 Projection vers l'espace des configurations

Nous allons définir ici l'espace des configurations, présenter quelques méthodes de construction et d'approximation puis introduire notre méthode pour projeter en temps réel les obstacles à l'intérieur de l'espace des configurations.

3.4.1 Définition

L'espace des configurations C, peut-être défini en robotique comme l'ensemble des configurations q, atteignables par un robot. L'exemple le plus simple est un robot de la taille d'une particule qui se déplace dans un plan XY, l'espace des configurations de ce système sera le plan XY lui-même. Si ce robot n'est plus une particule mais une forme géométrique quelconque se déplaçant toujours dans le plan XY mais pouvant changer d'orientation α, l'espace des configurations devient le plan XY augmenté d'une dimension, l'orientation α du robot. Pour un bras poly-articulé, une configuration q, est définie par la position de chacune des articulations q_i du robot :

$$q = \{q_1, q_2, q_3, \dots, q_N\} \qquad (3\text{-}14)$$

Avec N, le nombre d'articulations du robot. L'espace des configurations d'un robot à N articulations est donc de dimension N.

Considérant un robot A, composé de plusieurs éléments géométriques A_i, se déplaçant dans un environnement comportant des obstacles B_j, la partie accessible de l'espace des configurations est définie comme :

$$C_{free} = \left\{ q \in C \mid \bigcup A_i^q \cap \bigcup B_j = \emptyset \right\} \qquad (3\text{-}15)$$

Par opposition, l'espace des configurations inaccessible C_{obs} est défini comme :

$$C_{obs} = \left\{ q \in C \mid \bigcup A_i^q \cap \bigcup B_j \neq \emptyset \right\} \qquad (3\text{-}16)$$

Cette partie inaccessible correspond aux configurations, pour lesquelles le robot est en collision avec un ou plusieurs obstacles de l'environnement.

La partie de C_{obs} générée par un obstacle particulier est notée $C_{obs}(B_j)$ et :

$$C_{obs} = \bigcup C_{obs}(B_j) \qquad (3\text{-}17)$$

3.4.2 Calcul de l'espace des configurations

3.4.2.1 Méthodes basées sur l'échantillonnage de l'espace des configurations

Le parti pris de la plupart des travaux utilisant l'espace des configurations pour la planification de trajectoires est de ne pas calculer explicitement l'espace des configurations. Ces méthodes utilisent alors un échantillonnage de l'espace des configurations pour déterminer un chemin possible dans C_{free}.

Elles ont pour objectif de créer un graphe modélisant la traversabilité de l'espace des configurations. Ceux-ci fonctionnent en 3 étapes :

- Choisir des échantillons : La première étape consiste à choisir un ensemble de configurations q qui seront les nœuds du graphe. Les méthodes de sélection de ces échantillons sont abordées dans le Chapitre 2.
- Tester si ces configurations sont accessibles ou en collision grâce à un détecteur de collisions.
- Vérifier la possibilité de relier deux nœuds par une arête en vérifiant qu'il n'y ait pas de collision sur le parcours.

Les détecteurs de collisions fonctionnent en calculant l'intersection entre les corps du robot dans la configuration q, A_i^q et les obstacles B_j, si cette intersection est non vide, c'est qu'il y a collision.

La vérification d'une arête est classiquement faite en échantillonnant celle-ci et en vérifiant que chacun des échantillons est libre de collisions. Le choix du nombre d'échantillons est fait en fonction de la finesse de détection voulue et du temps de calcul disponible.

3.4.2.2 Méthodes exactes

Le calcul exact de l'espace des configurations est complexe, il faut déterminer pour chaque configuration de l'espace si un des corps du robot est en collision avec un obstacle. Compte tenu de la dimension élevée de l'espace des configurations d'un robot manipulateur, son calcul à l'aide d'un détecteur de collisions serait beaucoup trop long. Il existe des méthodes permettant de calculer arithmétiquement l'espace des

configurations à partir de la description algébrique du robot et des obstacles. (LaValle 2006) donne des exemples de calculs, de la collision de segments en 1D jusqu'à la construction de l'espace des configurations d'un robot triangulaire dans un environnement contenant un obstacle rectangulaire. Le calcul de la collision 1D est effectué en utilisant la différence de Minkowski entre l'obstacle et le robot (en position 0) :

$$C_{obs} = O \ominus A(0) \qquad (3\text{-}18)$$

La Figure 28 illustre ce calcul pour un robot $A = [-1,2]$ et un obstacle $O = [0,4]$. Cet exemple est facilement généralisable à la 2D ou 3D tant que le robot n'est capable que de translations. Dès lors que le robot est capable de rotation, le problème se complexifie et l'obtention d'un calcul algébrique de l'espace des configurations d'un bras poly-articulé est chose impossible.

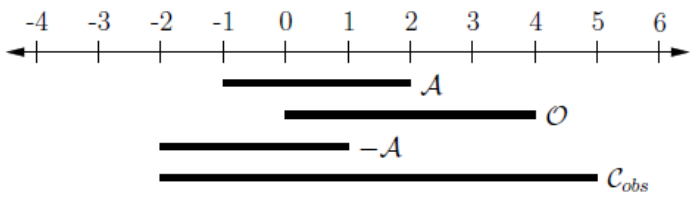


Figure 28 : Calcul de l'espace des configurations dans un cas 1D grâce à la différence de Minkowski (issu de (LaValle 2006)).

3.4.3 Notre méthode de construction

Nous construisons un espace des configurations discrétisé et calculons le statut de chaque cellule (occupé ou libre). Notre entrée pour la construction de l'espace des configurations est la carte d'occupation représentant l'espace de travail décrite précédemment. Cette carte est composée de cellules dont la taille et la position sont connues et invariantes. Seul le statut de la cellule change entre deux états libre ou occupé. Nous considérons chacune des cellules comme un obstacle potentiel (B_j dans l'équation *(3-16)*) et calculons sa projection dans C_{obs}.

Le calcul de l'espace des configurations étant long à effectuer, nous avons choisi de pré calculer la projection des obstacles et de le stocker dans une Look Up Table.

3.4.3.1 Modélisation du robot

Pour calculer la projection des obstacles dans C_{obs}, nous devons choisir une modélisation de robot. Nous modélisons les articulations du robot en utilisant la convention de Denavit-Hartenberg, grâce à quoi nous pouvons connaitre la matrice de transformation homogène liant chaque repère au monde (cf. Annexe Annexe 2 :).

Pour modéliser la géométrie du robot, nous définissons une approximation de chacun des corps du robot par un ensemble de formes 3D de base. Nous avons choisi trois formes de base pour modéliser les robots : la sphère, le cylindre et le parallélépipède rectangle. Chaque forme est positionnée par rapport au repère associé à ce corps, les cylindres et les parallélépipèdes doivent être orientés selon les axes du repère. Ramenés en 2D, les formes deviennent des disques ou des rectangles.

A titre d'exemple, pour modéliser le robot SCARA en 2D, nous avons utilisé deux rectangles, un associé à chaque corps, comme illustré sur la figure suivante :

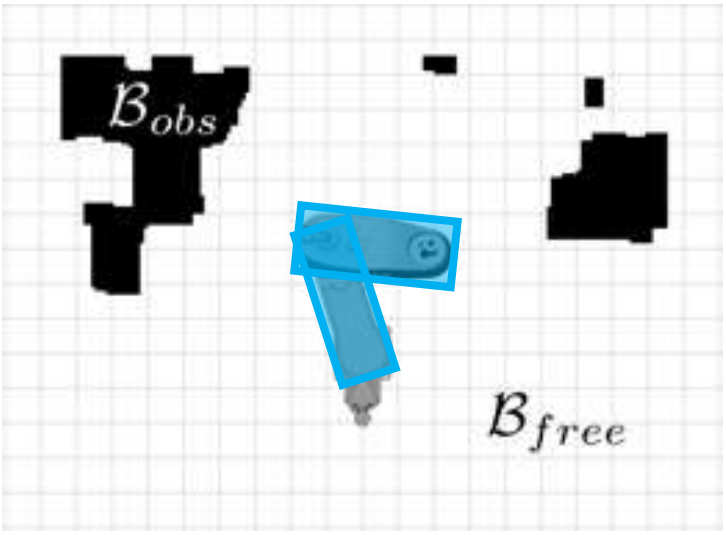


Figure 29 : Modélisation 2D du robot SCARA.

La modélisation du Baxter est quant à elle faite par 7 cylindres englobant chacun des corps du robot.

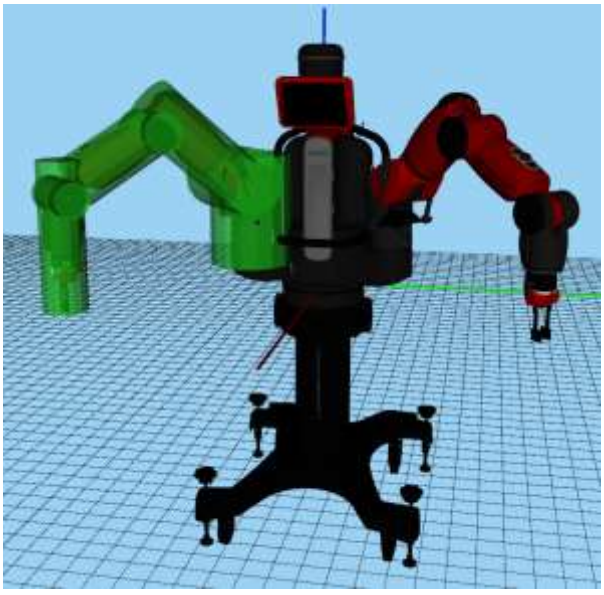


Figure 30 : Modélisation des segments du bras droit du robot Baxter.

3.4.3.2 Calcul des collisions entre le robot et les cellules de l'espace de travail

A partir de la modélisation que nous avons faite du robot, nous cherchons à déterminer la projection des obstacles de l'espace de travail vers l'espace des configurations. Notre espace de travail étant découpé en cellules cubiques pouvant être *libre* ou *obstacle*, nous calculons la projection de chacune des cellules de l'espace de travail dans l'espace des configurations.

Soit B_{xyz} une cellule de l'espace de travail localisée en x, y, z. La projection de cette cellule est donnée par :

$$C_{obs}(B_{xyz}) = \left\{ q \in C \mid \bigcup A_i^q \cap B_{xyz} \neq \emptyset \right\} \quad (3\text{-}19)$$

Si nous reprenons l'exemple du robot SCARA modélisé en 2D. Nous cherchons alors :

$$C_{obs}(B_{xy}) = \left\{ q \in C \mid (A_1^q \cup A_2^q) \cap B_{xy} \neq \emptyset \right\} \quad (3\text{-}20)$$

La Figure 31 montre la projection d'une cellule de la carte d'occupation 2D dans l'espace des configurations 2D du SCARA. La partie bleue correspond à une collision entre le corps A_1 du robot et la cellule et la partie verte à une collision avec le corps A_2. Notons qu'il est possible qu'une partie du vert soit masquée par le bleu.

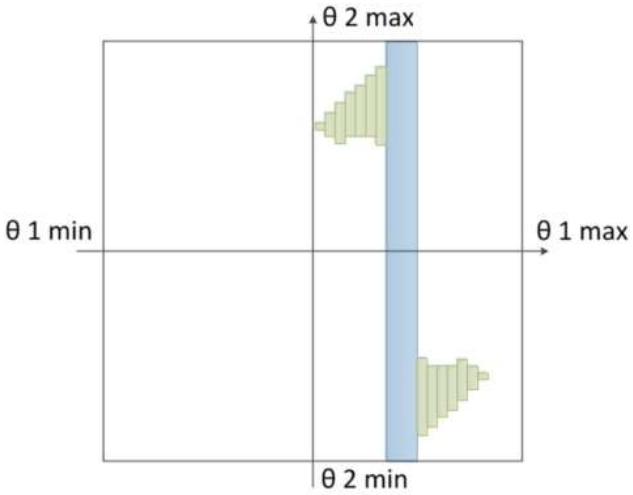


Figure 31 : Projection d'une cellule de la carte d'occupation 2D dans l'espace des configurations d'un SCARA (2D).

Pour accélérer les calculs, nous pouvons calculer séparément la partie bleue de la partie verte. Pour cela, nous nous plaçons d'abord dans le repère 0 du SCARA et cherchons les valeurs de θ_1 pour lesquelles A_1 est en collision avec la cellule. La Figure 32 donne la paramétrisation pour le calcul de la plage de collision.

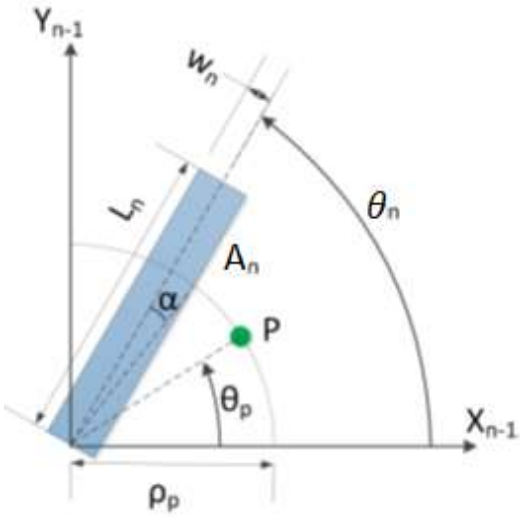


Figure 32 : Recherche de la plage de collision entre un corps du robot et une cellule de la carte d'occupation.

La plage collision entre le corps du robot (de dimension $L_n * 2W_n$) et la cellule approximée par un point P (en vert dans la Figure 32) de coordonnées polaires dans le repère 0 (ρ_P,θ_P) est :

Si $\rho_P < \sqrt{L_n^2 + W_n^2}$:

$$C_{obs}(B_{xy}, A_1) = \{q \in C \mid \theta_1 \in [(\theta_p - \alpha), (\theta_p + \alpha)]\} \quad (3\text{-}21)$$

Avec $\alpha = \arcsin\left(\frac{W_n}{\rho_P}\right)$

Sinon :

$$C_{obs}(B_{xy}, A_1) = \emptyset \quad (3\text{-}22)$$

La zone de bleue de la Figure 31 correspond donc à une plage de θ_1 inaccessible. Nous stockons dans la Look Up Table uniquement les limites de cette plage, cette méthode permet un gain de mémoire conséquent. Pour la zone verte (Figure 31), nous utilisons la même méthode de calcul mais la position du point P dans le repère 1 est dépendante de la valeur de θ_1. Nous ferons donc ce calcul pour chaque position possible de θ_1 donnée par la discrétisation de l'espace des configurations. Nous pouvons alors écrire :

$$C_{obs}(B_{xy}, A_2) = \sum_{\theta_i = \theta_{1,min}}^{\theta_{1,max}} \{q \in C \mid \theta_1 = \theta_i \; et \; \theta_2 \in [(\theta_p - \alpha), (\theta_p + \alpha)]\} \quad (3\text{-}23)$$

En décomposant ainsi le calcul de la projection d'une cellule obstacle dans l'espace des configurations, l'équation (3-20) devient :

$$C_{obs}(B_{xy}) = \left\{\theta_1 \mid A_1^{\theta_1} \cap B_{xy} \neq \emptyset\right\} \cup \left\{(\theta_1, \theta_2) \mid A_2^{(\theta_1, \theta_2)} \cap B_{xy} \neq \emptyset\right\} \quad (3\text{-}24)$$

Cette équation est généralisable aux robots à N degrés de liberté :

$$C_{obs}(B_{xyz}) = \bigcup_{n=1}^{N} \left\{ (\theta_1, \dots, \theta_N) \mid A_n^{(\theta_1,\dots,\theta_N)} \cap B_{xyz} \neq \emptyset \right\} \quad (3\text{-}25)$$

Nous cherchons donc les valeurs de θ_i pour lesquelles $A_i^{(\theta_1,\dots,\theta_i)}$ est en collision avec B_{xyz} et les calculons grâce à l'Algorithme 1 :

Comme le montre l'équation (3-25), $C_{obs}(B_{xyz})$ est l'union des collisions de chacun de corps A_n du robot. La position de chacun des corps n'étant due qu'à son articulation et celles le précédant, nous allons parcourir pour chaque corps l'ensemble des positions possibles des articulations le précédant. C'est le rôle de la boucle for ligne 4 et de la fonction récursive ligne 8 de l'Algorithme 1.

Cette fonction *parcours recursif* a comme entrée l'indice de l'articulation actuelle (qui commence à 0), l'indice du corps à calculer, le point obstacle et la configuration courante du robot. Si l'indice courant est celui à calculer, nous appelons la fonction *Calcul collision* qui renvoie les configurations où $A_{Ica}^{(\theta_1,\dots,\theta_i)}$ est en collision avec B_{xyz}. Sinon, nous parcourons toutes les positions possibles de l'articulation courante, modifions la configuration courante et rappelons la fonction *parcours recursif* en incrémentant l'indice de l'articulation courante.

La fonction *Calcul collision* effectue un changement de repère, en déplaçant le point obstacle dans le repère de l'articulation à calculer. En effet, vu que le calcul est effectué pour toutes les possibilités de positionnement des articulations précédentes, celles-ci peuvent être considérées comme fixes. Nous calculons la matrice de passage du monde vers le repère courant en multipliant tour à tour les matrices données par la modélisation de Denavit Hartenberg et nous transposons le point obstacle. Puis, pour chaque volume composant le corps, nous appelons la fonction calculant les plages d'intersections entre le volume et le point obstacle.

```
1: CS_obs = ∅
2: P = (x, y, z)
3: Q = θ_1, ..., θ_N,
4: for n ∈ [1, N] do
5:    CS_obs.merge(Parcours recursif(0, n, P, Q))
6: end for
7:
8: CS_obs Parcours recursif(indice courant Ico, indice à
   calculer Ica, point P, configuration Q)
9: CS_obs = ∅
10: if (Ico=Ica) then
11:    CS_obs.merge(Calcul collision(Ica, P, Q))
12: else
13:    for θ_Ico ∈ [θ_Ico,min, θ_Ico,max] do
14:       Q[Ico] = θ_Ico
15:       CS_obs.merge(Parcours recursif(Ico + 1, n, P, Q))
16:    end for
17: end if
18: return CS_obs
19:
20: CS_obs Calcul collision(indice à calculer Ica, point P,
   configuration Q)
21: CS_obs = ∅
22: TH = I(4)
23: for n ∈ [1, Ica − 1] do
24:    THliaison = DenavitHatenberg(n, q[n])
25:    TH = TH_liaison * TH
26: end for
27: P = TH * P
28: for all A_Ica ∈ robot[Ica] do
29:    switch A_Ica.type
30:    case sphère :
31:    CS_obs.merge(intersect sphere(P, A_Ica, Q, Ica ))
32:    case sphère :
33:    CS_obs.merge(intersect cylinder(P, A_Ica, Q, Ica ))
34:    case sphère :
35:    CS_obs.merge(intersect
       parallelepipede(P, A_Ica, Q, Ica ))
36: end for
```

Algorithme 1 : Calcul de la projection d'un point xyz dans l'espace des configurations.

Le changement de repère permet de ramener la recherche des contacts robot-obstacle à un problème 2D. En effet, le corps A_{Ica} n'est sujet qu'à la rotation θ_{Ica} autour de $\overrightarrow{z_{Ica-1}}$. Les fonctions intersect projettent le problème en 2D avant de calculer les configurations en collision. L'Algorithme 2 explicite la fonction *intersect cylindre*.

```
1: intersect cylinder(point P, volume , configuration Q, indice à calculer Ica)
2: CS_obs = ∅
3: if A_Ica.orientation=='Z' then
4:   if ((P.Z < A_Ica.position.Z + A_Ica.hauteur/2) && P.Z )> (A_Ica.position.Z
   - A_Ica.hauteur/2)) then
5:     C = new circle(A_Ica.position.X, A_Ica.position.Y, A_Ica.rayon)
6:     intersection_inters[] = intersect_circle(P, C)
7:   end if
8: else
9:   if ((P.Z < A_Ica.position.Z + A_Ica.rayon) && P.Z ) > (A_Ica.position.Z -
   A_Ica.rayon)) then
10:    if A_Ica.orientation=='X' then
11:      R = new_rectangle(A_Ica.position.X, A_Ica.position.Y, A_Ica.hauteur,
   A_Ica.rayon*2)
12:    else
13:      R = new_rectangle(A_Ica.position.X, A_Ica.position.Y, A_Ica.rayon*2,
   A_Ica.hauteur)
14:    end if
15:    intersection_inters[] = intersect_circle(P, R)
16:  end if
17: end if
18: for all inter in inters do
19:   obs_el = new CS_obs_element()
20:   for n ∈ [1,Ica-1] do
21:     obs_el.min[n] = Q[n]
22:     obs_el.max[n] = Q[n]
23:   end for
24:   obs_el.min[n] = inter.min
25:   obs_el.max[n] = inter.max
26:   for n ∈ [Ica+1,N] do
27:     obs_el.min[n] = robot.articulation[n].min
28:     obs_el.max[n] = robot.articulation[n].max
29:   end for
30:   C_obs.add(obs_el)
31: end for
32: return C_obs
```

Algorithme 2 : Calcul de l'intersection d'un cylindre et d'un point obstacle.

Celle-ci commence par tester l'orientation du cylindre, si celui-ci est orienté selon l'axe z alors sa projection en 2D sera un cercle (cf. Figure 33, b)), sinon ce sera un rectangle (cf. Figure 33, a))

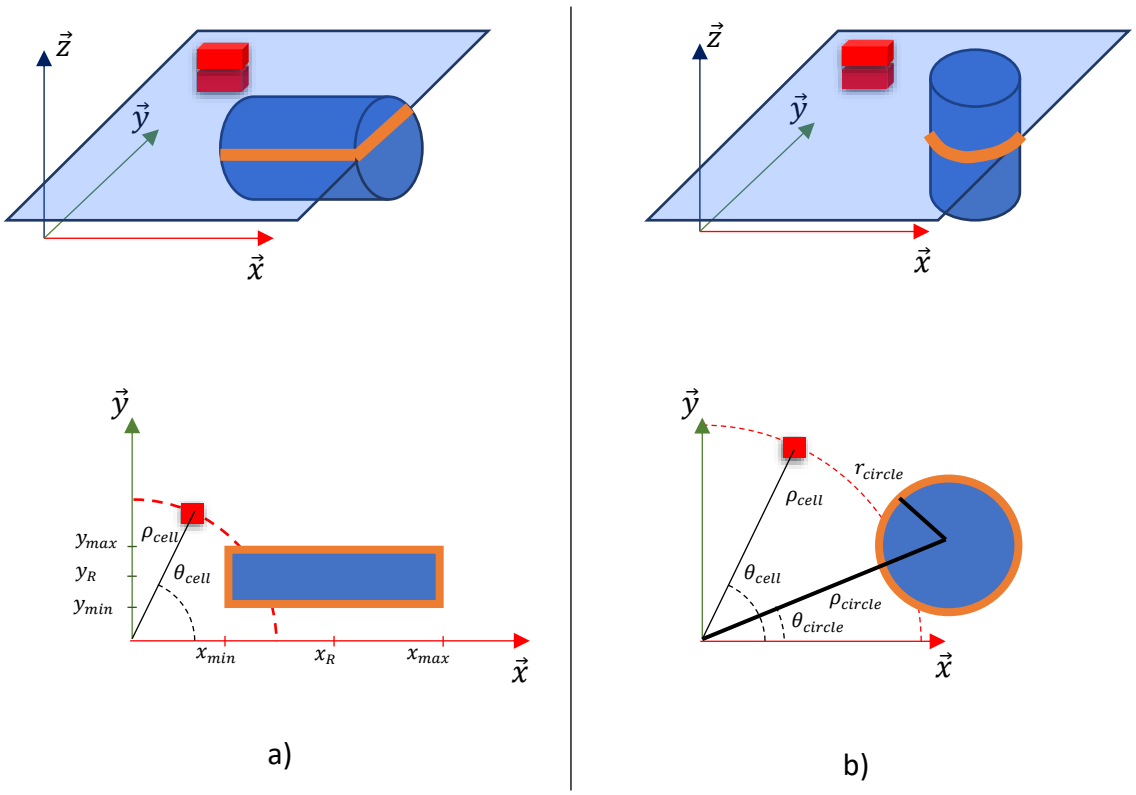


Figure 33 : Projection en 2D du calcul de collision entre une cellule de l'espace et un cylindre, a) orienté selon l'axe x, b) orienté selon l'axe z.

La plage de collision entre un cercle et la cellule obstacle est :

Si $\rho_{cell} \in [\rho_{circle} - r_{circle},\ \rho_{circle} + r_{circle}]$:

$$\theta_{cell} - \theta_{circle} - d\theta < \theta_{n,collision} < \theta_{cell} - \theta_{circle} + d\theta \qquad (3\text{-}26)$$

Avec $d\theta$ donné par le théorème d'Al-Kashi :

$$d\theta = \arccos\left(\frac{\rho_{circle}^2 + \rho_{cell}^2 - r_{cricle}^2}{2 * \rho_{circle} * \rho_{cell}}\right) \qquad (3\text{-}27)$$

Si le calcul de $d\theta$ n'est pas possible, c'est que le point obstacle se trouve à l'intérieur du cercle quel que soit θ_n. Et si ρ_{cell} n'est pas dans la plage donnée au-dessus alors il n'y a pas de collision.

Le calcul de la collision entre la cellule obstacle et un rectangle est plus complexe, nous utilisons l'Algorithme 3 pour déterminer la ou les plages où il y a collision.

1: intersect_rectangle(point P, rectangle R)
2: x_{min} = R.x − R.lx/2
3: x_{max} = R.x + R.lx/2
4: y_{min} = R.y − R.ly/2
5: y_{max} = R.y + R.ly/2
6: $\rho_1 = \sqrt{(x_{min}^2 + y_{min}^2)}$
7: $\rho_2 = \sqrt{(x_{max}^2 + y_{min}^2)}$
8: $\rho_3 = \sqrt{(x_{min}^2 + y_{max}^2)}$
9: $\rho_4 = \sqrt{(x_{max}^2 + y_{max}^2)}$
10: $\rho_{cell} = \sqrt{(P.X^2 + P.Y^2)}$
11: θ_{cell} = atan2(P.Y,P.X)
12: On_X_axis = $x_{max} > 0$ && $x_{min} < 0$
13: On_Y_axis = $y_{max} > 0$ && $y_{min} < 0$
14: closest = min(ρ_1, ρ_2, ρ_3, ρ_4)
15: **if** On_X_axis **then**
16: closest = min(closest,abs(x_{min}),abs(x_{max}))
17: **end if**
18: **if** On_Y_axis **then**
19: closest = min(closest,abs(y_{min}),abs(y_{max}))
20: **end if**
21: **if** On_X_axis && On_Y_axis **then**
22: closest = -1
23: **end if**
24: farest = max(ρ_1, ρ_2, ρ_3, ρ_4)
25: **if** ρ_{cell} <closest || θ_{cell} >farest **then** $\emptyset$
26: **end if**
27: Liste_ $\theta_{intersect} = \emptyset$
28: candidate_X = $\sqrt{(\rho_{cell}^2 + y_{max}^2)}$
29: **if** x_{min} < candidate_X < x_{max} **then**
30: Liste_$\theta_{intersect}$.add(atan2(y_{max}, candidate_X))
31: **end if**
32: **if** $-x_{min}$ < candidate_X < $-x_{max}$ **then**
33: Liste_$\theta_{intersect}$.add(atan2(y_{max}, -candidate_X))
34: **end if**
35: candidate_X2 = $\sqrt{(\rho_{cell}^2 + y_{min}^2)}$
36: **if** x_{min}<candidate_X2<x_{max} **then**
37: Liste_$\theta_{intersect}$.add(atan2(y_{min}, candidate_X2))
38: **end if**
39: **if** $-x_{min}$<candidate_X2<$-x_{max}$ **then**
40: Liste_$\theta_{intersect}$.add(atan2(y_{min}, -candidate_X2))
41: **end if**
42: candidate_Y = $\sqrt{(\rho_{cell}^2 + x_{min}^2)}$
43: **if** y_{min}<candidate_Y<y_{max} **then**
44: Liste_$\theta_{intersect}$.add(atan2(candidate_Y, x_{min}))
45: **end if**
46: **if** $-y_{min}$<candidate_Y<$-y_{max}$ **then**
47: Liste_$\theta_{intersect}$.add(atan2(-candidate_Y, x_{min}))
48: **end if**

```
candidate_Y2 = √(ρ²_cell + x²_max)
if y_min<candidate_Y2<y_max then
    Liste_θ_intersect.add(atan2(candidate_Y2, x_max))
end if
if −y_min<candidate_Y2<−y_max then
    Liste_θ_intersect.add(atan2(-candidate_Y2, x_max))
end if
Liste_θ_intersect.sort
if Liste_θ_intersect == ∅ then inter(robot.articulation[n].min,
robot.articulation[n].max)
end if
θ_test = Liste_θ_intersect[0] + (Liste_θ_intersect[1]- Liste_θ_intersect[0])/2
X_test = ρ_cell ∗ cos(θ_cell)
Y_test = ρ_cell ∗ sin(θ_cell)
if X_test ∉ [x_min,x_max] ‖ Y_test ∉ [y_min,y_max] then
    Liste_θ_intersect.pushfront(robot.articulation[n].min)
    Liste_θ_intersect.pushback(robot.articulation[n].max)
end if
Liste_inter = ∅
for i ∈ [0,Liste_θ_intersect.count/2] do
    Liste_inter.add( θ_cell - Liste_θ_intersect[i], θ_cell - Liste_θ_intersect[i+1])
end for
return Liste_inter
```

Algorithme 3 : Calcul de l'intersection d'un rectangle et d'un point obstacle

Explicitons le déroulement de l'algorithme. Nous commençons par calculer l'ensemble des variables nécessaires : les coordonnées de chaque côté du rectangle sur l'axe qui leur est perpendiculaire, la distance entre chaque angle du rectangle et l'axe de rotation et les coordonnées polaires du point obstacle. Nous déterminons ensuite si une collision est possible. En effet, si le point obstacle est plus loin que les quatre sommets du rectangle ou s'il est plus près que le sommet ou le coté le plus près du rectangle, il n'y a pas de collision possible. Dans le cas contraire, nous allons chercher les points collisions possibles. Pour chaque côté du rectangle, nous cherchons les coordonnées des deux intersections possibles entre la droite (colinéaire au segment) et le cercle parcouru par le point obstacle, puis nous vérifions si ces points sont sur le segment. La liste des points ainsi obtenue est triée et nous vérifions si la première plage correspond à une collision. Dans le cas contraire, nous ajoutons au début et à la fin de la liste les limites de l'articulation. Nous retournons alors une plage de collision pour chaque paire de points.

Remarque : Les méthodes de calcul présentées ci-dessus considèrent seulement des obstacles ponctuels. Cependant, nos cellules sont de taille non négligeable. Cette approximation est rendue possible par la surestimation du volume du robot. Cependant, un calcul plus précis des intersections entre le volume de cellule et ceux du robot est possible. En modélisant la cellule comme une sphère dont le rayon peut être choisi selon

le pessimisme voulu entre celui de la sphère inscrite dans le cube de la cellule ($r = \frac{a}{2}$) et celui de la sphère englobante ($\sqrt{3 * \frac{a}{2}}$), il suffit de quelques modifications dans les formules et algorithme présentés ci-dessus. Dans l'équation (3-27) r_{circle} devient $r_{circle} + r_{cell}$ et dans l'Algorithme 3, la recherche des candidats se fait avec $sqrt({\rho_{cell}}^2 - (x_{max} + r_{cell})^2)$ ou $sqrt({\rho_{cell}}^2 - (x_{min} - r_{cell})^2)$ et idem pour Y.

3.4.3.3 Construction de la table de valeurs (Look Up Table)

Les résultats de la recherche de collision vus précédemment sont stockés dans une Look Up Table. Celle-ci se présente sous la forme d'un tableau à 3 entrées (une pour chaque coordonnée de l'espace cartésien) et contient donc pour chaque cellule de la carte d'occupation décrite en section (3.3), sa projection dans l'espace des configurations sous la forme d'une liste de zones non atteignables. Cette liste de zones a été préalablement compressée en suppriment les recouvrements de zones.

L'espace des configurations est initialisé vide et lorsque la carte d'occupation est disponible, nous parcourons pour chaque cellule occupée la liste des zones et incrémentons dans l'espace des configurations les cellules correspondantes. Une fois ce parcours terminé, nous obtenons l'espace des configurations où toutes les cellules dont les valeurs sont supérieures à 0 sont des obstacles.

La mise à jour à l'instant suivant peut-être réalisée en réinitialisant l'espace des configurations et en projetant à nouveau l'ensemble des cellules obstacle. Nous pouvons aussi ne projeter que le différentiel de la carte d'occupation aux deux instants. En incrémentant les valeurs des projections des nouveaux obstacles et en décrémentant celles des obstacles ayant disparu, nous pouvons effectuer la mise à jour plus rapidement.

La projection d'une cellule de la carte d'occupation ne dépendant que de la forme du robot, la Look Up Table peut être sauvegardée et réutilisée dans la même application ou une autre si la carte d'occupation est identique (taille et position des cellules).

Il est aussi possible de sauvegarder séparément des morceaux de Look Up Table correspondant aux accessoires du robot (ex : les préhenseurs) ou aux objets saisis, et ainsi adapter notre construction de l'espace des configurations en fonction des changements d'outils ou des saisis d'objets.

3.5 Représentation de l'espace des configurations et des obstacles associés.

3.5.1 Cas à deux dimensions

Prenons comme premier exemple un robot SCARA (seulement les deux premiers axes) et considérons un monde plan. Le robot est constitué de deux articulations dont les axes

sont perpendiculaires au plan du monde. Le modèle géométrique du robot est schématisé Figure 34 :

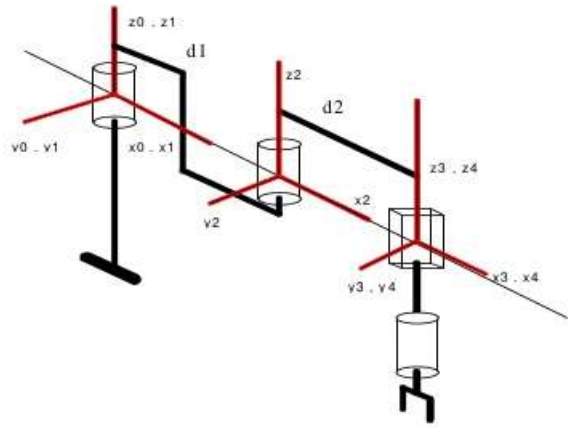


Figure 34 : Modèle Géométrique d'un robot SCARA, le placement des axes respecte la convention de Denavit Hartenberg

L'espace des configurations de ce robot est donc un plan de coordonnés (q_1, q_2). La Figure 35 un exemple de l'espace des configurations du SCARA et de la projection de trois obstacles ponctuels à l'intérieur de cet espace.

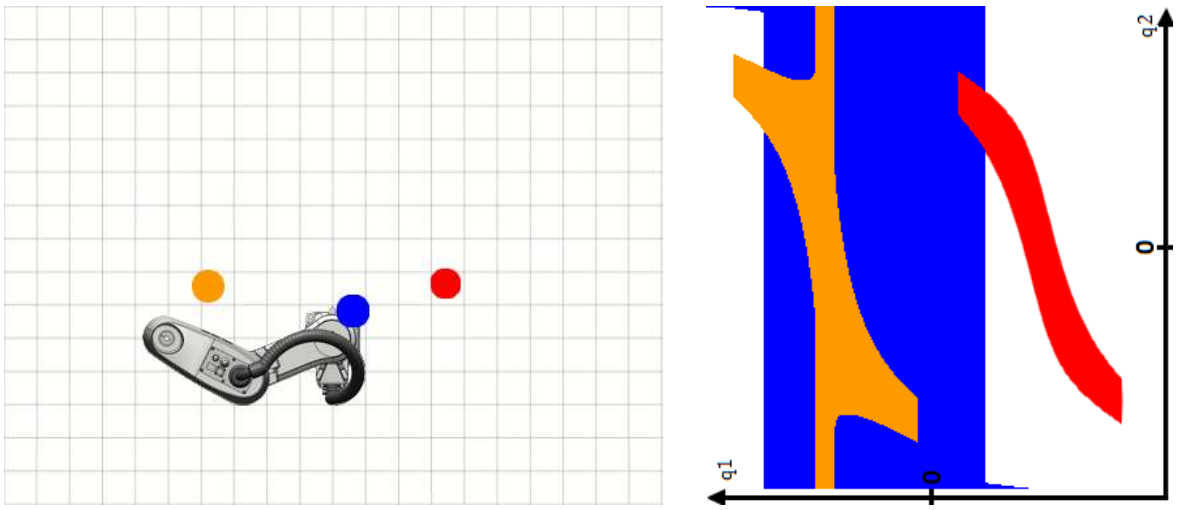


Figure 35 : Robot SCARA dans un mode 2D avec obstacles et espace des configurations à deux dimensions avec la projection de ces obstacles

3.5.2 Cas à trois dimensions

Comme second exemple, nous prenons ce même robot SCARA mais ajoutons à notre modèle le troisième axe qui est une glissière dont l'axe est aussi perpendiculaire au plan du sol. Nous avons un robot à trois degrés de liberté se déplaçant dans un monde en 3D. Son espace des configurations est donc un plan q_1, q_2 augmenté d'une élévation q_3. Dans ce cas précis, l'espace des configurations est hétérogène. Si l'effecteur fixé à l'axe 4 du robot (rotation autour d'un axe perpendiculaire au plan du sol) est une forme de révolution autour de z_4 alors la rotation q_4 ne modifie pas le positionnement géométrique

du robot. Ainsi le paramètre A_i^q des équations *(3-15)* et *(3-16)* ne dépend pas de q_4 et C_{obs} non plus. Nous pouvons donc représenter l'espace des configurations du robot SCARA sans cette 4ème dimension et déterminer les positions accessibles pour le robot.

Un autre exemple d'un espace des configurations à trois dimensions peut être donné en considérant les trois premiers axes d'un bras robot industriel. La plupart des bras industriels à 6 axes sont constitués de deux groupes d'articulations. Le premier groupe est appelé porteur et correspond aux trois premiers axes qui permettent de positionner l'effecteur dans l'espace, tandis que les trois derniers axes, dits, de poignet, permettent d'orienter l'effecteur (cf. Figure 36). Un espace des configurations en 3D permet donc de déterminer l'espace accessible pour le porteur.

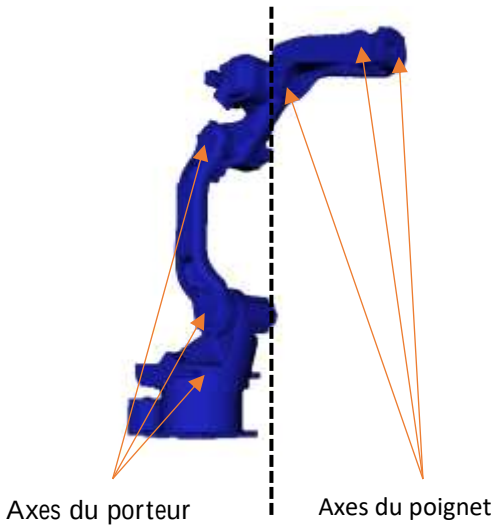


Figure 36 : Distinction des axes du porteur et du poignet sur le robot Yaskawa MH12.

Pour pouvoir prendre en compte l'effecteur dans le cas où nous modélisons un bras à 6 axes par un espace des configurations à 3 dimensions, il faut comme pour l'exemple du SCARA, garantir que C_{obs} ne dépend pas des axes de poignet, c'est-à-dire que q_4, q_5 ou q_6 ne modifient pas A_i^q. Deux méthodes permettent de satisfaire cette condition :

- La méthode la plus simple est d'englober l'effecteur et toutes les parties du robot qui sont mises en mouvement par les axes de poignet dans un ou plusieurs volumes. En remplaçant les A_i^q par ces volumes englobants, la projection de l'espace des configurations est indépendante de q_4, q_5 et q_6. L'inconvénient de cette méthode est de générer un nombre important de configurations inaccessibles car une position de l'espace des configurations 3D ne sera accessible que si toutes les configurations du poignet sont accessibles à cette position. En revanche, elle permet d'orienter l'effecteur dans n'importe quelle position dans l'espace accessible. Cette méthode est très efficace pour

des robots dont la partie poignet/effecteur est de petite taille (ex : le MH12 de Yaskawa).

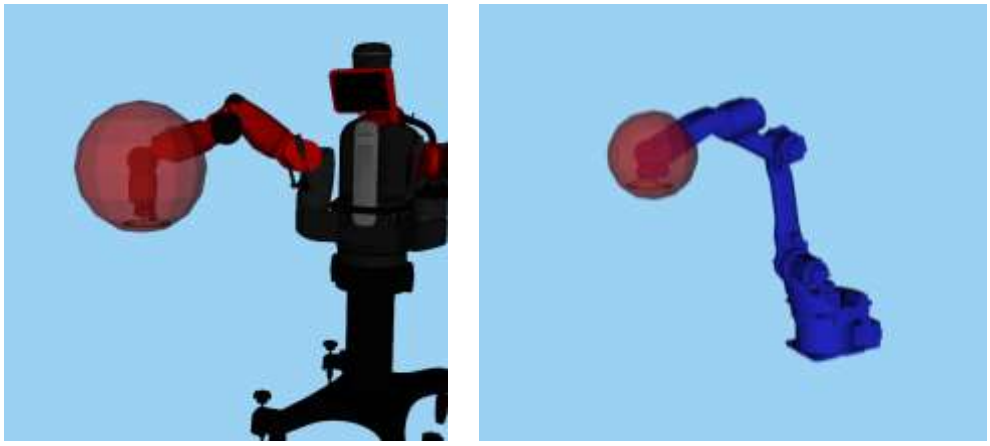


Figure 37 : Volume englobant les positions possibles de l'effecteur (ventouse) pour un Baxter et un MH12.

- La seconde méthode consiste à fixer l'orientation de l'effecteur. En asservissant q_4, q_5 et q_6 pour obtenir une orientation constante, la position du robot devient seulement dépendante de q_1, q_2 et q_3. Nous pouvons donc modéliser le robot avec un espace des configurations en 3D. L'inconvénient évident de cette méthode est l'impossibilité de modifier l'orientation de l'effecteur. En revanche, le nombre de configurations accessibles est plus important qu'avec la méthode précédente car le volume du poignet/effecteur n'est pas surévalué (cf. Figure 38). Cette méthode est adaptée pour les applications où l'orientation de l'effecteur n'a pas besoin de changer (ex : une application de pick and place).

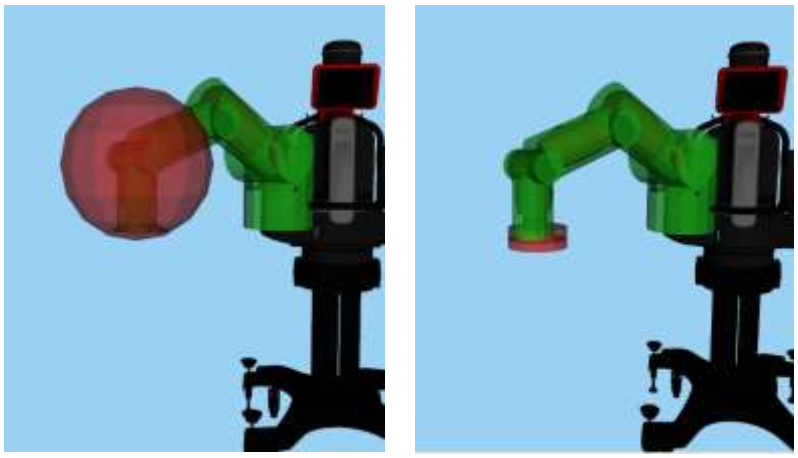


Figure 38 : Comparaison des volumes englobant l'effecteur. A gauche, l'effecteur peut prendre n'importe quelle orientation. A droite, l'orientation de l'effecteur est fixe.

La représentation des espaces de configuration en 3D peut être faite par un cube en trois dimensions. Cette représentation présente cependant un manque de lisibilité puisque visualisées sur un écran ou sur une feuille de papier, des zones de l'espace des configurations sont masquées. Nous préférons utiliser une représentation construite en

tranchant ce cube (cf. Figure 39). Si notre espace des configurations est discret, en disposant côte à côte les tranches faites au pas d'échantillonnage, nous obtenons une représentation 2D de l'espace 3D que nous pouvons donc visualiser en entier.

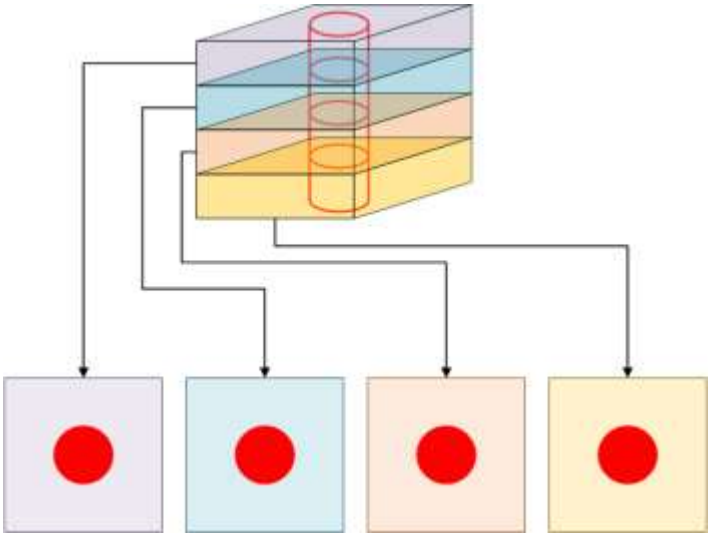


Figure 39 : principe de représentation de l'espace des configurations 3D contenant un obstacle cylindrique

3.5.3 Cas à quatre dimensions

Le dernier exemple d'espace des configurations que nous présenterons ici est celui d'un espace à 4 dimensions. Il existe des bras robot à 7 axes comme par exemple les bras qui équipent le Baxter. L'ajout d'une articulation supplémentaire au robot permet de modifier la position du coude du robot sans changer la position de l'effecteur, il existe donc une infinité de configurations possibles du robot pour une pose donnée de l'effecteur. La partie porteur de ces bras est donc composée de 4 axes. Par extension de l'exemple du bras à 6 axes présenté juste avant, nous pouvons définir l'espace des configurations accessible des bras 7 axes grâce à un espace des configurations à 4 dimensions.

Pour représenter l'espace des configurations 4D, nous utilisons une méthode semblable à celle de l'espace en 3D. Cet espace 4D est un hypercube que nous tranchons dans l'axe q_1, nous obtenons alors une ligne de cubes (tranches de l'hypercube) disposés sur un axe q_1. Nous tranchons alors ces cubes selon l'axe q_2 pour obtenir des plans q_3, q_4 répartis sur un plan q_1, q_2, comme illustré par la Figure 40. La Figure 41 montre un exemple d'espace des configurations en 4 dimensions dans lequel des obstacles ont été projetés. La position courante du robot est donnée par le point rouge. Un mouvement unitaire sur un seul des axes du robot déplace celui-ci sur un des points verts (dans le sens positif) ou bleus (dans le sens négatif).

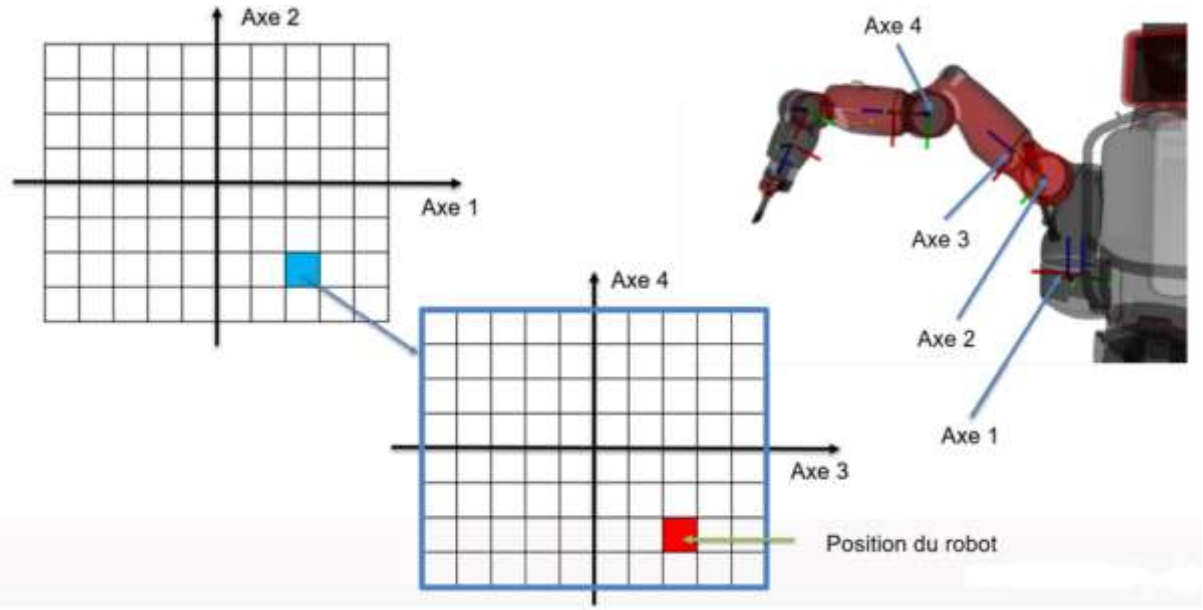


Figure 40 : Construction de l'espace des configurations en 4D pour un robot à 7 articulations (ici un bras du Baxter).

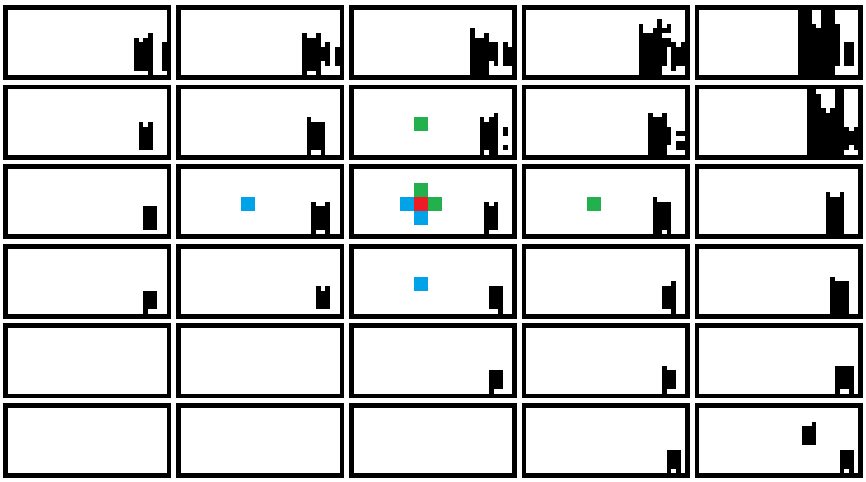


Figure 41 : Exemple d'espace des configurations en 4D.

3.5.4 Cas à cinq dimensions et plus

Il est possible de construire des espaces des configurations à 5, 6 ou 7 dimensions pour représenter complètement l'espace accessible aux robots à 6 et 7 axes. Cependant, nos méthodes de simplification permettent de s'en affranchir. De plus, la représentation des espaces de dimension 5 et plus est très complexe.

L'obtention d'une représentation telle que l'espace des configurations, dont nous avons détaillé la construction ici, est la première étape pour la réalisation d'une trajectoire sûre dans l'environnement collaboratif. Celle-ci contient les informations nécessaires à la planification d'un chemin comme par exemple la position des obstacles, mais aussi à la sécurisation de l'interaction, en distinguant les humains du reste. Le chapitre suivant se consacre donc à la planification de trajectoires au sein de l'espace des configurations.

Chapitre 4.
Génération de trajectoires sures

Synthèse :

Dans ce chapitre, nous avons introduit le fonctionnement de notre algorithme de planification de trajectoires. Nous avons d'abord explicité le calcul rapide de la carte des distances aux obstacles et les améliorations que nous avons apportées, en termes d'optimisation du temps de calcul mais aussi dans la prise en compte du facteur humain. L'axe médian, ensemble de points équidistants des obstacles, a été ensuite proposé comme support des trajectoires dans l'espace des configurations. Nous avons ensuite présenté une technique de fusion dynamique permettant d'éviter les à-coups du robot lors d'un changement de trajectoires lié à des perturbations dynamiques de l'axe médian. Enfin, nous avons exposé comment moduler la vitesse de déplacement du robot pour limiter l'énergie cinétique emmagasinée en fonction de la distance le séparant de l'humain.

Sommaire du chapitre :

4.1 Introduction

Pour atteindre notre objectif : déplacer de façon sûre un bras manipulateur dans une application cobotique, notre seconde tâche, est de planifier une trajectoire en s'appuyant sur la représentation décrite dans le chapitre précédant. Les contraintes telles que la dynamicité de l'environnent ou la nécessité de prendre en compte l'humain sont aussi présentes dans cette partie du travail. Pour cela, notre planificateur devra être rapide et ajuster en ligne les mouvements du robot en fonction de l'évolution de l'environnement.

Nous proposons alors de calculer la trajectoire en plusieurs étapes, comme exposées par la Figure 42. En premier lieu, nous calculons la carte des distances dans l'espace libre des configurations. Celle-ci nous permet ensuite d'obtenir l'axe médian, qui est l'ensemble des points équidistants des frontière de l'espace libre. Cette ensemble de points sert alors de support au choix d'un chemin reliant la position actuelle du robot à son objectif. Enfin, nous apporterons quelques ajustements après avoir échantillonné la trajectoire, selon la modulation de vitesse et la fusion dynamique, pour obtenir la trajectoire la plus sûre possible pour l'humain. Ce chapitre explicitera, dans cet ordre, l'ensemble des étapes évoquées.

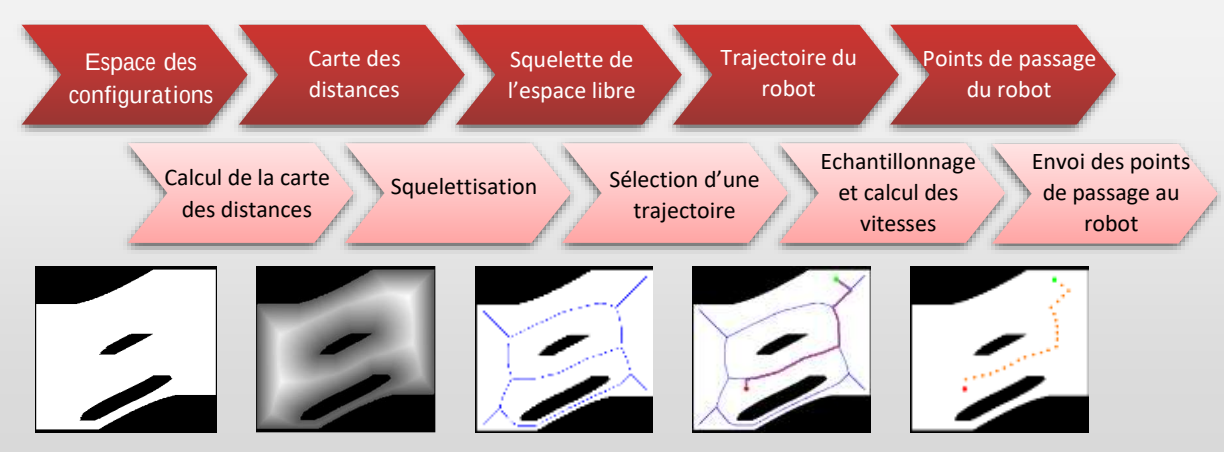


Figure 42 : pipeline du chapitre, de l'espace des configurations, construit précédemment, à la génération d'une trajectoire pour le robot

4.2 Calcul de la carte des distances

4.2.1 Concepts préliminaires

La carte des distances, introduite par (Pfaltz et Rosenfeld 1967), est un objet informatique, permettant de connaitre pour chaque point de l'espace, sa distance à l'obstacle le plus proche. La carte des projections contient en plus pour chacun de ces points, le (ou les) points de frontières les plus proches. La carte des distances est utilisée dans de nombreux domaines, tels que la reconnaissance de forme, la synthèse d'image, la compression de données et la robotique. Elle peut par exemple servir à dénombrer des

cellules en biologie ou servir de base à la navigation d'un robot. La Figure 43 donne des exemples de carte des distances.

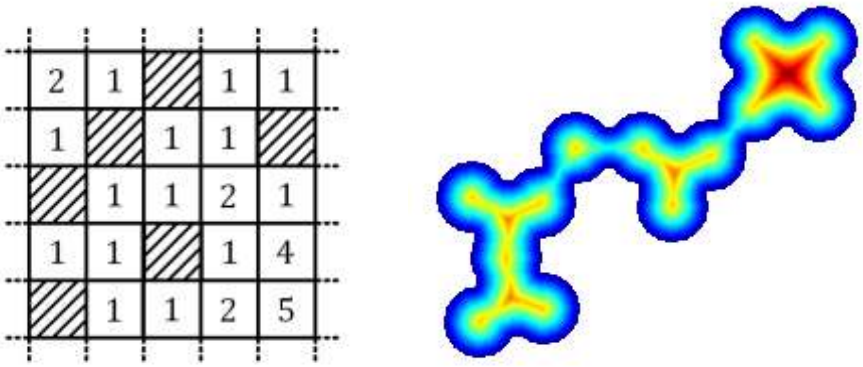


Figure 43 : à gauche, valeurs numériques correspondant à la carte des distances d'une grille binaire (les distances sont au carré), à droite, représentation de la carte des distances d'un ensemble de particules. L'échelle de couleur va du bleu (distance minimum) au rouge (distance maximum)[2].

Une définition formelle de la carte des distances est connue suit :

Etant donnés deux points $x, y \in \mathbb{Z}^n$, nous notons $d(x,y) \in \mathbb{R}^+$ la distance euclidienne séparant les deux points : $d(x,y) = \sqrt{\sum_{n=1}^{N}(x_n - y_n)^2}$. Par extension, la distance entre un point $x \in \mathbb{Z}^n$ et un ensemble de points $S \subset \mathbb{Z}^n$ est donnée par : $d(x,S) = \min_{y \in S} d(x,y)$.

Ce qui permet de définir la carte des distances et celle des projections :

Définition : La carte des distances D_{map} d'un sous-ensemble $S \subset \mathbb{Z}^n$ est la transformation de $\mathbb{Z}^n$ vers $\mathbb{R}^+$ qui associe à chaque point $x \in \mathbb{Z}^n$ sa distance à S : $D_{map}(x) = d(x,S)$.

Définition : La carte des projections P_{map} d'un sous ensemble $S \subset \mathbb{Z}^n$ est la transformation de $\mathbb{Z}^n$ vers S qui associe à chaque point $x \in \mathbb{Z}^n$ l'ensemble des points à distance minimum : $P_{map}(x) = \{y \in S | d(x,y) = D_{map}(x)\}$.

Remarque : Pour les points $x \in S$, $D_{map}(x) = 0$ et $P_{map}(x) = x$.

[2] Image de http://www.pfl-cepia.inra.fr/index.php?page=tutoImg-carte-distances

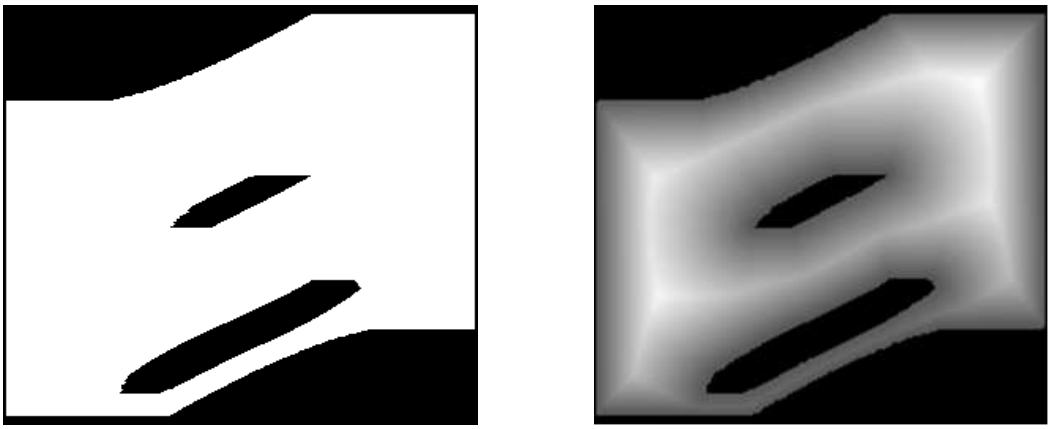


Figure 44 : Exemple de carte des distances dans notre contexte : à gauche, un espace des configurations en 2D (en blanc les configurations libres de collision, en noir les configurations obstacles), à droite, la carte des distances correspondante (en noir, les configurations dont la distance aux obstacles est nulle, en niveau de gris la distance aux obstacles des cellules libres, plus le gris est claire plus la configuration est éloignée).

La Figure 44 montre un exemple d'une carte d'occupation extraite de nos résultats experimentaux. Cet outil présente, pour nous, deux intérêts majeurs :

- Nous cherchons à déplacer le robot le plus loin des obstacles, ce qui peut être obtenu en suivant le squelette de la partie libre de l'espace des configurations. Ce dernier correspond aux crêtes de la carte des distances. Elle est donc une première étape de son calcul.
- Elle nous permet de connaitre la distance entre le robot et l'obstacle le plus proche. Cette information permet notamment d'adapter le comportement du robot, en particulier si l'obstacle est humain.

Le calcul de la carte des distances n'est pas trivial. La méthode la plus simple consiste à calculer, pour chaque cellule, sa distance à tous les points obstacles, et ne garder que la plus petite. Cette stratégie simpliste présente le défaut évident de nécessiter un important temps de calcul. De nombreux travaux ont donc cherché à diminuer la complexité de cette tâche. Comme discuté dans (Fabbri et al. 2008), ceux-ci peuvent être classés en trois catégories, en fonction de l'ordre de traitement des pixels :

- *Ordered propagation* (propagation ordonnée) : Cette catégorie d'algorithmes simule l'équation Eikonale. Un front d'onde est initialisé à la bordure des obstacles et se propage à vitesse constante. A son passage, chaque cellule reçoit comme valeur la distance parcourue par l'onde.
- *Raster scan* (parcours de matrice) : Ces algorithmes parcourent un à un chaque pixel de l'image, ligne par ligne. Les parcours peuvent être multiples en avançant et en reculant.
- *Independent scaning* (parcours indépendant) : Ces approches, telles que celle de (Pfaltz et Rosenfeld 1967), sont basées sur la réduction de dimension. Elles calculent en premier les distances 1D sur chaque ligne indépendamment. Ce

résultat intermédiaire est ensuite utilisé pour construire la carte des distances 2D en parcourant les colonnes une à une. Dans les espaces à plus de deux dimensions, la seconde étape est simplement répétée dans chacune d'elles.

Nous nous intéressons en particulier à la troisième famille d'algorithmes. Le calcul de la première dimension est simple et rapide. A l'inverse, les étapes suivantes sont complexes. Certaines propriétés de la distance euclidienne permettent toutefois de minimiser le temps de calcul tout en garantissant la justesse des résultats. Trois approches existent dans la littérature pour réaliser cela en conservant une complexité linéaire en regard du nombre de cellules.

La première approche repose sur les diagrammes de Voronoï. Le calcul explicite de ce diagramme est coûteux et non linéaire. Les travaux de (Breu et al. 1995) reposent sur le calcul efficient de l'intersection de celui-ci avec la ligne d'une image sans le construire explicitement. Une amélioration, proposée par, repose sur le fait que les points proches ont souvent la même projection. Se reposant sur le même concept, (Maurer et al. 2003) puis (Wang et Tan 2013) proposent des algorithmes plus efficients. Le dernier propose aussi une extension à plus de dimension.

La seconde approche se base sur des opérations morphologiques. (Shih et Mitchell 1992) montre que la carte des distances peut être calculée à l'aide d'une seule opération morphologique d'érosion en niveau de gris de la grille d'entrée. (Lotufo et Zampirolli 2001) ont ensuite proposé de décomposer l'élément structurant en un ensemble d'éléments 1D permettant d'obtenir un algorithme à parcours indépendant.

La dernière approche utilise l'intersection de paraboles telle qu'introduite initialement par (Saito et Toriwaki 1994). Pour augmenter la vitesse de calcul de la seconde phase et des suivantes, la distance aux obstacles est déduite en déterminant l'enveloppe inférieure d'un ensemble de paraboles. Cette approche a été reformulée et améliorée au fil du temps (Hirata 1996), (Meijster, Roerdink, et Hesselink 2000), (Felzenszwalb et Huttenlocher 2012). La complexité reste toujours proportionnelle à la taille de la grille mais le terme constant évolue.

4.2.2 Calcul de la carte des distances basé sur l'enveloppe de paraboles

Nous calculons la carte des distances en nous basant sur la dernière approche présentée ci-dessus. Nous détaillons maintenant l'algorithme présenté par (Felzenszwalb et Huttenlocher 2012) et présentons une amélioration pour réduire les temps de calcul.

4.2.2.1 Principe de fonctionnement

Considérant l'entrée de l'algorithme comme une grille M en N dimensions contenant pour chaque cellule une information binaire (obstacle ou pas obstacle), nous pouvons écrire :

$$M = M_{obs} \cup M_{free} \tag{4-1}$$

La carte des distances associée à M, $D_{map}(M)$ contient pour chaque cellule $x = (x_1, x_2, \dots, x_n)$ sa distance $D_{map}(x)$ au point de M_{obs} le plus proche :

$$\forall x \in M, D_{map}(x) = \min_{y \in M_{obs}} \sqrt{\sum_{n=1}^{N} (x_n - y_n)^2} \tag{4-2}$$

La décomposition proposée par (Saito et Toriwaki 1994) repose sur le calcul du premier élément de la somme puis l'ajout tour à tour des éléments suivants. Par souci de simplicité, nous détaillerons uniquement le cas à deux dimensions. La première étape consiste alors à calculer pour chaque ligne l une carte des distances 1D $g(x)$. Celles-ci sont regroupées dans la grille $G = \{g(x)\}$.

$$\forall x = (l, x_2), g(x) = \min_{y=(l,y_2) \in M_{obs})} |x_2 - y_2| \tag{4-3}$$

La seconde étape utilise comme entrée la grille G et parcours chaque colonne c pour en déduire la carte des distances :

$$\forall x = (x_1, c), D(x)^2 = \min_{1<l<l_{\max}} \{(x_1 - l)^2 + g(l, c)^2\} \tag{4-4}$$

Dans le cas à N dimensions, ce résultat est stocké dans une grille G_2 qui servira d'entrée à la répétition de cette étape parcourant la troisième dimension et ainsi de suite jusqu'à la Nième dimension.

En considérant une colonne c, la distance au carrée, séparant chacun des éléments de la colonne, de l'obstacle le plus proche sur la ligne l, est donnée par la fonction parabolique :

$$\mathcal{F}_l = (x_1 - l)^2 + g(l, c)^2 \tag{4-5}$$

Le sommet de cette fonction se situe à la position l et à la hauteur donnée par $g(l, c)$. L'équation devient alors :

$$\forall x, D(x)^2 = \min_{1<l<l_{\max}} \{\mathcal{F}_l\} \quad (4\text{-}6)$$

Il apparait alors que ce minimum correspond à l'enveloppe basse d'un ensemble de paraboles (cf. Figure 47).

4.2.2.2 Algorithme de calcul

L'algorithme utilisé par (Felzenszwalb et Huttenlocher 2012) est le suivant :

- Le calcul de la première étape est fait en suivant le pseudo code suivant :

```
Input: Binary image I = X ∪ X^c of size m * n
Output: 1-dimensional distance transform g(x) for each x ∈ I

for i=0 to m-1 do
    if (i, 0) ∈ X^c then
        g(i, 0) = 0
    else
        g(i, 0) = ∞
    for j=1 to n-1 do
        if (i, j) ∈ X^c then
            g(i, j) = 0
        else
            g(i, j) = g(i, j − 1) + 1
    for j=n-1 to 1 do
        if g(i, j + 1) < g(i, j) then
            g(i, j) = g(i, j + 1) + 1
```

Algorithme 4 : Calcul de $g(x)$, distance aux obstacles dans la première dimension

La Figure 45 illustre son fonctionnement. Chaque ligne étant traitée tour à tour, l'algorithme les parcourt d'abord en avançant et en enregistrant dans chaque cellule de G la distance au premier obstacle à gauche (∞ tant qu'aucun obstacle n'est vu). Ensuite, il effectue le chemin inverse en enregistrant les valeurs à l'obstacle à droite si celles-ci sont plus faibles (cf Figure 45 b.). Nous obtenons alors la grille G des distances 1D.

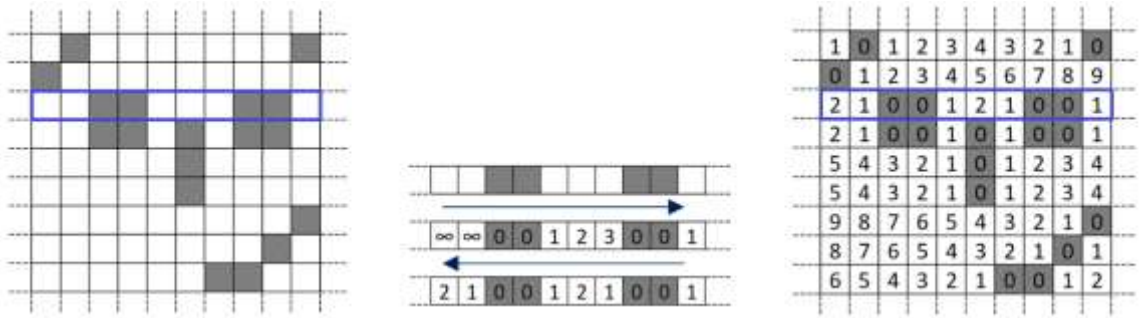


Figure 45 : Première étape de calcul pour l'obtention de la carte des distances. A gauche, la grille binaire d'entrée. Au milieu, le détail de calcul de la ligne 3, le parcours se fait d'abord de gauche à droite puis de droite à gauche en

gardant les valeurs les plus faibles. A droite, la grille G contenant pour chaque ligne la fonction $g(x)$ donnant pour chaque cellule, la distance à l'obstacle, de cette même ligne, le plus proche.

- L'étape suivante est calculée grâce à l'algorithme 2 :

```
1: Input: 1-dimensional dist. transform g(x), for each
   x ∈ I
2: Output: Euclidean distance transform 𝒟
3:
4: for j=0 to n-1 do
5:    k = 0, v[0] = 0, z[0] = −∞, z[1] = +∞
6:    for i=1 to m-1 do
7:       w = 1 + s(v[k], i)
8:       while w ≤ z[k] do
9:          k = k − 1
10:         w = 1 + s(v[k], i)
11:      k = k + 1, v[k] = i, z[k] = w, z[k + 1] = ∞
12:   for i=0 to m-1 do
13:      while z[k + 1] < i do k = k + 1
14:      𝒟²(i, j) = (i − v[q])² + g(v[q], j)²
```

Algorithme 5 : Calcul de D_{map} à partir de $g(x)$ en parcourant la seconde dimension

Bien que celui-ci ne construit pas explicitement toutes les paraboles, la Figure 46 montre l'ensemble des paraboles correspondant à la 5ème colonne de l'image pour éclairer notre propos. Chacune des paraboles donne la distance de chacun des points de la colonne, d'indice x_1, à l'obstacle le plus proche sur la ligne d'indice l.

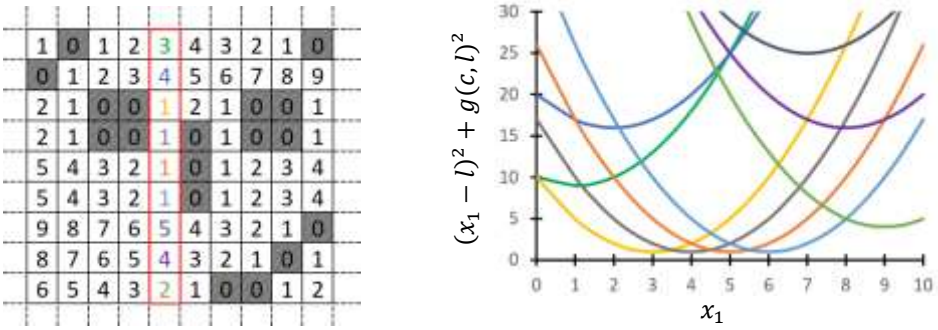


Figure 46 : représentation de l'ensemble des paraboles (à droite) de la colonne 5 de notre grille (à gauche).

Pour ne garder que les paraboles constituant l'enveloppe basse, celles-ci vont être décrites par trois informations, l'indice z qui représente le début de la zone où la parabole constitue l'enveloppe basse, v qui est l'indice où se situe son sommet, et $g(v)$, issue du résultat de la première étape qui nous donne la hauteur du sommet (cf Figure 47).

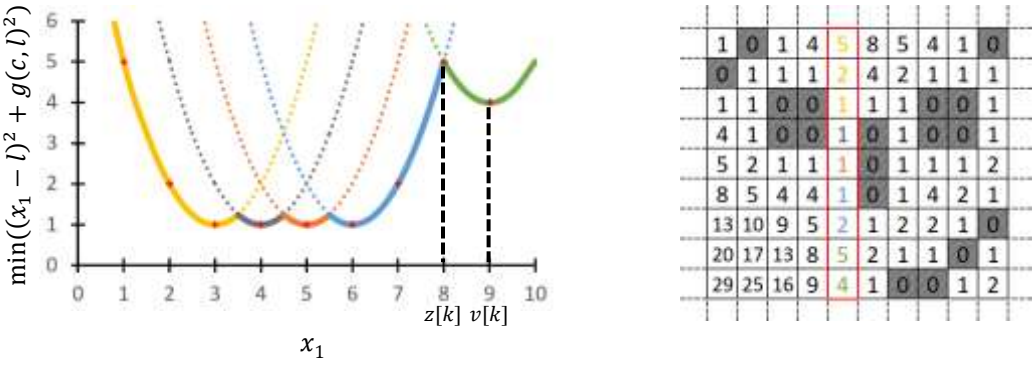


Figure 47 : enveloppe minimum des paraboles de la 5ième colonne (à gauche) et carte des distances au carré de notre exemple.

L'algorithme va donc, pour chaque colonne, tester les paraboles de chacune des lignes. Il commence par initialiser une liste de paraboles, puis en commençant par la première ajouter la parabole correspondante à chaque indice i. Cette parabole est alors comparée à la dernière de la liste, indicé k, en déterminant l'intersection entre les deux :

$$s(v[k], i) = \frac{i^2 - v[k]^2 + g(i, c) - g(v[k], c)}{2(i - v[k])} \qquad (4\text{-}7)$$

L'indice $w = 1 + trunc(s(v[k], i))$, représente le premier entier suivant l'intersection des paraboles, c'est-à-dire l'indice x_1 à partir duquel la nouvelle parabole est plus basse. Si w est inférieur ou égal à l'indice z de la précédente, alors cette dernière est supprimée de la liste, puisqu'elle ne fait plus partie de l'enveloppe basse, et le test est recommencé avec la parabole $k - 1$. Sinon w est plus grand, et alors elle est ajoutée à la liste.

Les valeurs de D_{map} sont obtenues en cherchant pour chaque cellule (x_1, c), la parabole dont la zone d'influence débute avant x_1, puis en calculant grâce à l'équation (4-4) la valeur en ce point. La racine carrée de cette valeur est alors calculée pour obtenir la distance à l'obstacle le plus proche. Néanmoins, si la grille compte d'autres dimensions, la valeur est stockée dans une grille qui sert d'entrée à une répétition de l'Algorithme 5. L'opération est alors renouvelée jusqu'à atteindre la Nième dimension.

4.2.3 Optimisation de l'algorithme

En observant l'Algorithme 5, il apparait que le calcul de l'intersection de paraboles est l'opération la plus coûteuse en temps de calcul et la plus répétée. Notre proposition d'amélioration de cet algorithme repose sur la réduction de l'occurrence de cette opération. Pour parvenir à cela, nous nous reposons sur deux hypothèses imposées par le contexte :

- Le calcul de la carte des distances est fait sur une grille discrète orthonormée. De plus, les obstacles occupent une partie non négligeable de l'espace et sont généralement regroupés. En conséquence, il arrive que souvent, sur une colonne, les paraboles issues de lignes côte à côte soient à la même hauteur, c'est-à-dire que les distances 1D soit les mêmes, telles que les lignes 3, 4, 5 et 6, colonne 5 de la Figure 46. Il est alors nécessaire de stocker chacune des paraboles correspondantes pour former l'enveloppe inférieure.
- La carte des distances est elle aussi définie sur une grille discrète. Cela induit que l'enveloppe de paraboles n'a pas besoin d'être exacte en continue mais seulement aux indices entiers.

A partir de ces deux propriétés, nous proposons de remplacer la succession de plusieurs paraboles de même élévation par un segment (cf. Figure 48)

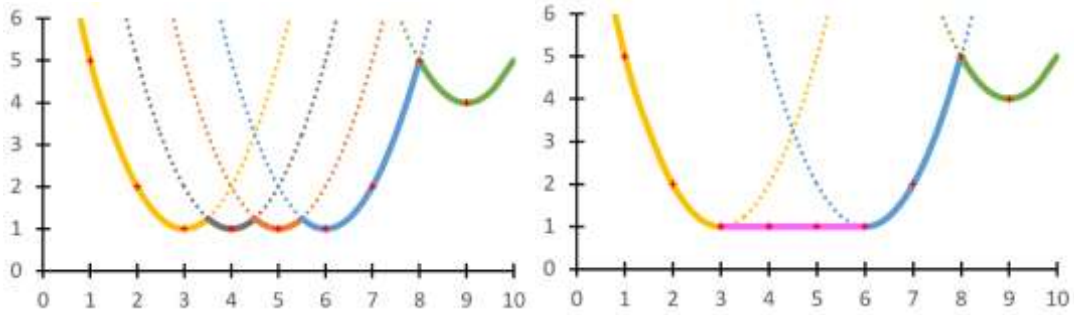


Figure 48 : Notre proposition : remplacer la suite de paraboles à même hauteur (à gauche) par un segment (à droite)

Cette approche apporte deux avantages. Premièrement, le calcul coûteux des intersections est fait moins souvent compte tenu de la réduction du nombre d'entités, formant l'enveloppe basse. Et puis, la construction du segment est réalisée grâce à une stratégie d'avance rapide, permettant alors d'éviter ce calcul.

4.2.3.1 Formulation théorique

Afin de mettre en place cette simplification, nous devons détailler deux points importants avant de présenter l'algorithme :

- Tout d'abord, le processus de création du segment est fait de tel sorte qu'aucun test d'intersection n'est nécessaire. Lors de l'ajout d'une nouvelle parabole à l'indice i, nous vérifions que si celle d'indice $i-1$ répond à trois critères :
  - son sommet est à la même hauteur,
  - elle fait partie de l'enveloppe basse,
  - sa zone d'influence débute à $i-1$ ou avant.

  Lorsque ces trois conditions sont remplies, nous vérifions si la distance 1D de la cellule $i+1$ est-elle aussi identique. Dès lors, nous avons trois points de suite à la

même hauteur et nous créons donc un segment dont les paramètre sont : g_{seg} sa hauteur, égale au sommet des paraboles et z_{seg} le début de sa zone d'influence, égale à $i-1$. Ensuite, celui-ci est prolongé en incrémentant i tant que $g(i,c) = g_{seg}$, c'est-à-dire tant que les paraboles sont à même hauteur. Enfin, lorsque le segment se termine nous ajoutons la parabole correspondant au dernier point du segment (la courbe bleue sur la Figure 48).

- Ensuite, l'intersection s, d'un segment horizontal avec une parabole d'indice k, est calculée en résolvant l'égalité suivante :

$$g_{seg}^2 = (s - v[k])^2 + g(v[k], c)^2 \quad (4\text{-}8)$$

Nous obtenons aisément une équation du second degré :

$$s^2 - 2.s.v[k] + v[k]^2 + g(v[k], c)^2 - g_{seg}^2 = 0 \quad (4\text{-}9)$$

Dont les solutions sont :

$$s = v[k] \pm \sqrt{g_{seg}^2 - g(v[k], c)^2} \quad (4\text{-}10)$$

De ces solutions, nous ne garderons que la plus petite. En effet, comme le processus de formation du segment se fait sans comparaison, il n'est donc pas nécessaire de vérifier sa position vis-à-vis des paraboles précédentes. C'est seulement lors de l'ajout d'une parabole survenant après la création du segment que nous devons les comparer. Le segment se trouve alors sur les lignes d'indices inférieurs à celui de la nouvelle parabole. De plus, il n'est pas nécessaire de vérifier le signe de $g_{seg}^2 - g(v[k], c)^2$. En effet, le cas où le somment de la parabole se situe au-dessus du segment ne peut pas se présenter. Lors de son ajout, celle-ci sera comparée à la demi-parabole suivant le segment, ajoutée lors de sa construction. Le segment ne peut donc être comparé qu'aux paraboles plus basses que lui.

4.2.3.2 Implémentation

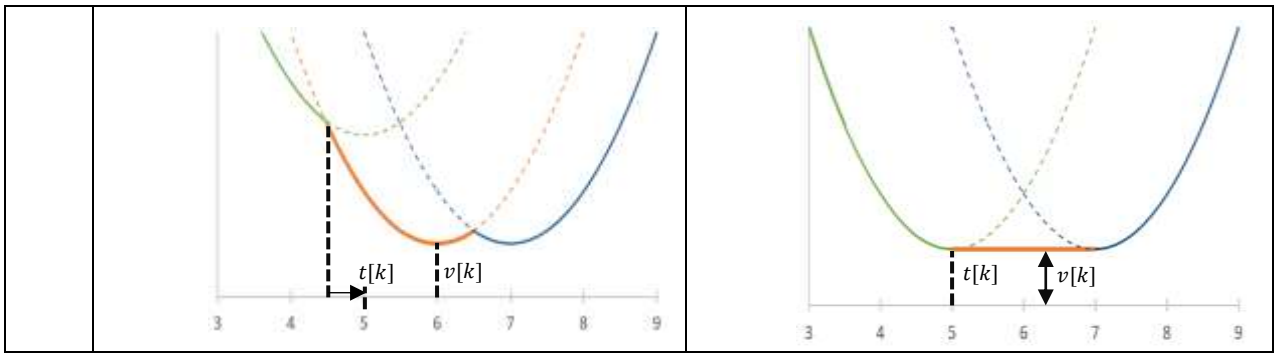
Pour intégrer les modifications présentées ci-dessus, nous avons implémenté une nouvelle version de l'Algorithme 5 :

```
1: Input: 1-dimensional dist. transform g(x), for each
   x ∈ I
2: Output: Euclidean distance transform 𝒟
3:
4: for j=0 to n-1 do
5:    k = 0, v[k] = 0, z[k] = −∞, z[k+1] = +∞, t[k] = 0,
      w = 0, i = 1
6:    while i < m do
7:       if g(i − 1, j) == g(i, j) and z[k] ≤ v[k] then
8:          i = i + 1
9:       if i < m and g(i − 1, j) == g(i, j) then
10:         k = k+1, z[k] = i−1, v[k] = g(i, j), t[k] = 1
11:         while i < m and g(i − 1, j) == g(i, j) do
12:            i = i + 1
13:      k = k+1, z[k] = i−1, z[k+1] = ∞, v[k] = i−1,
         t[k] = 0
14:      w = 1 + s(v[k], i)
15:      while w ≤ z[k] do
16:         k = k − 1
17:         if t[k] == 0 then w = 1 + s(v[k], i)
18:         else w = 1 + s2(v[k], i)
19:      k = k + 1, z[k] = w, z[k + 1] = ∞, v[k] = i,
         t[k] = 0, i = i + 1
20:   for i=0 to m-1 do
21:      while z[k + 1] < i do k = k + 1
22:      if t[k] == 0 then 𝒟²(i, j) = (v[q] − j)² +
         g(v[q], j)²
23:      else 𝒟²(i, j) = v[k]
```

Algorithme 6 : Algorithme de notre proposition

Afin de prendre en compte la variété d'éléments constituant l'enveloppe basse, nous avons introduit une nouvelle variable $t[k]$ dont la valeur définit le type : 0 si c'est une parabole, 1 si c'est un segment. $z[k]$ contient toujours l'indice du commencement de la zone d'influence de l'élément, tandis que $v[k]$ contient la hauteur du segment ou l'indice de position du sommet de la parabole. Le tableau suivant résume cela :

	parabole	segment
$v[k]$	Abscisse du sommet de la parabole	Hauteur du segment
$z[k]$	Début de la zone d'influence	Début de la zone d'influence
$t[k]$	0	1



Les modifications de l'algorithme apparaissent à partir de la ligne 7 où se trouve le test des conditions sur la parabole précédente, à savoir sommet à même hauteur, et zone d'influence débutant, au maximum, au niveau de son sommet. Ce second test permet aussi de valider qu'elle fait partie de l'enveloppe, sans quoi la valeur de w serait forcement plus grande que $i - 1$. La ligne 9 vérifie que la parabole suivante est compatible avec un segment, et si tel est le cas la ligne 10 ajoute le segment à la liste. Ensuite, les lignes 11 et 12 permettent l'avance rapide tant que les paraboles sont à même hauteur, et pour finir la ligne 13 ajoute la parabole qui termine le segment. A partir de la ligne 14, l'algorithme reprend normalement, si ce n'est le test ligne 17 qui permet de choisir le bon calcul d'intersection.

4.2.3.3 Résultats de notre optimisation

Une étude comparative de la rapidité de calcul de la carte des distances a été menée entre notre algorithme et les algorithmes les plus rapides à l'heure actuelle. Par soucis de clarté, celle-ci est présenté en détails en Annexe Annexe 3 :. En résumé cette étude montre que notre amélioration apporte un gain significatif de temps de calcul dans les environnements complexes (20% par rapport à [Felzenswalb12]). Il existe néanmoins un algorithme plus rapide mais celui-ci n'est pas aisément extensible à plus de deux dimensions (cf. Figure 49).

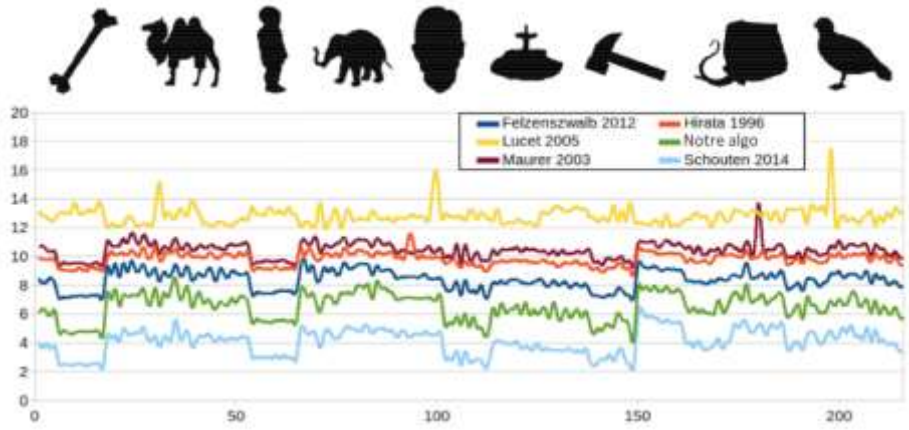


Figure 49 : temps de calcul avec chacun des algorithmes pour les images de (Sebastian et Kimia 2005)

La théorie et les résultats de notre proposition d'amélioration ont été soumis en vue d'une publication dans la revue « Image and Vision Computing ».

4.2.4 Application au contexte

Dans le contexte de notre étude, la collaboration Homme robot, le calcul de la carte des distances doit-être adapté pour améliorer la sécurisation de l'interaction. Avant de présenter notre stratégie d'adaptation, nous allons présenter la formulation de D_{map} dans l'espace des configurations puisque nous avons choisi de travailler dans cette représentation (cf Chapitre 3).

Nous rappelons que $C \subset \mathbb{Z}^n$ est l'espace des configurations discret, et qu'un point de cette espace (une configuration) est notée $q = (q_1, q_2, \dots, q_n)$. Les configurations pour lesquelles le robot entre en collision avec un obstacle sont notées q_{obs} et forment $C_{obs} \subset C$.

La carte des distances est calculée sur l'espace des configurations, et associe donc à chaque $q \in C$, sa distance $D_{map}(q)$ aux configurations présentant une collision :

$$D_{map}(q) = d(q, C_{obs}) \qquad (4\text{-}11)$$

De même, la carte des projections associe à chaque configuration $q \in C$ la ou les configuration(s) en collision les plus proches :

$$P_{map}(q) = \{q_{obs} \in C_{obs} | d(q, q_{obs}) = D_{map}(q)\} \qquad (4\text{-}12)$$

Comme nous l'avons présenté précédemment (cf. Chapitre 3-3.3-3.3.3-3.3.3.4), nous avons également projeté dans l'espace des configurations la sémantique du type d'obstacle afin de différencier les opérateurs. En isolant les obstacles humains dans un ensemble $C_{humain} \subset C_{obs}$, nous pouvons définir une carte des distances aux obstacles humain telle que :

$$D_{map,humain}(q) = d(q, C_{humain}) \qquad (4\text{-}13)$$

Cette carte nous donne donc la distance séparant chaque configuration d'une configuration présentant une collision avec un opérateur. Cependant, la séparation de l'information entre deux cartes ne permet pas de générer facilement un squelette adapté et complique notre tâche. C'est pourquoi nous proposons de fusionner les deux cartes afin de créer une carte des distances adaptée à partir de laquelle nous calculerons un axe médian adapté. Cette fusion est réalisée cellule par cellule grâce à l'équation suivante, où

k_h est un facteur que nous réglons pour obtenir l'éloignement voulu (cf Figure 50 et Annexe Annexe 4 :) :

$$\forall q \in C, D_{adapt}(q) = \min(D_{map}(q), \frac{D_{humain}(q)}{k_h}) \quad (4\text{-}14)$$

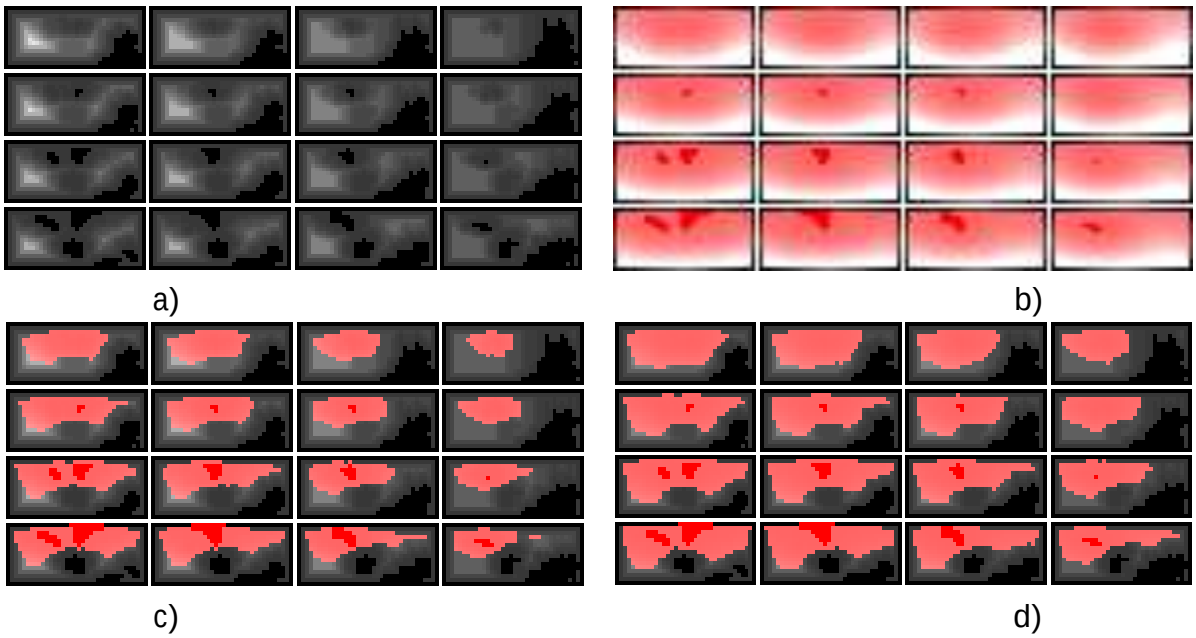


Figure 50 : fusion de la carte des distances pour tous les obstacles (a) avec celle calculée uniquement pour les obstacles humains (b), avec un coefficient k_h de 1,5 (c) et 3 (d). La zone modifiée dans les cartes fusionnées apparait en dégradé de rouge. Ces images sont des portions d'espaces des configurations 4D disponibles en entier en Annexe Annexe 4 :.

4.3 Planification de trajectoires

Nous introduisons dans cette section notre stratégie de génération de trajectoires basée sur la squelettisation de la partie libre de l'espace des configurations.

4.3.1 Navigation sur l'axe médian

L'axe médian a été introduit en premier par Harry Blum (H. F. Blum 1967) qui cherchait un descripteur de forme pouvant en capturer les caractéristiques topologiques. Il le présente par une analogie intuitive, le feu de prairie : Soient χ une forme contenue dans le plan $\mathbb{R}^2$, qui correspond à notre prairie et $\delta\chi$ les limites de cette forme. En supposant qu'un feu se déclare à un instant t sur l'ensemble de $\delta\chi$ et que celui-ci se propage à une vitesse uniforme dans toute la forme et dans toutes les directions. Ce feu envahi l'ensemble de la prairie, et l'axe médian se définit alors comme le lieu de rencontre des différents fronts de l'incendie.

Blum suggère aussi une seconde définition de l'axe médian. Compte tenu de l'uniformité de déplacement du front de flamme, la collision intervient à équidistance des points

d'origine de l'incendie, ceux-ci sont les points de $\delta\chi$ les plus proches. Si nous appelons x un point de la prairie et $P(x)$, la projection de x correspondant aux points de $\delta\chi$ les plus proches, alors l'axe médian $AM(\chi)$ se définit comme l'ensemble des points x dont le cardinal de $P(x)$ est supérieur ou égal à 2 :

$$AM(\chi) = \{x \in \chi \mid card(P(x)) \geq 2\} \quad (4\text{-}15)$$

(Pfaltz et Rosenfeld 1967) propose une troisième définition : Soit D un cercle inscrit à l'intérieur de la forme χ. Ce disque est maximal s'il n'existe aucun autre disque $D' \subset \chi$ tel que $D \subset D'$. Il apparait clairement qu'un disque maximal est en contact avec $\delta\chi$ en au moins deux points, nous retrouvons là, une équivalence avec la seconde définition de Blum. L'axe médian est à cet occasion nommé squelette, nous utilisons indifféremment les deux désignations.

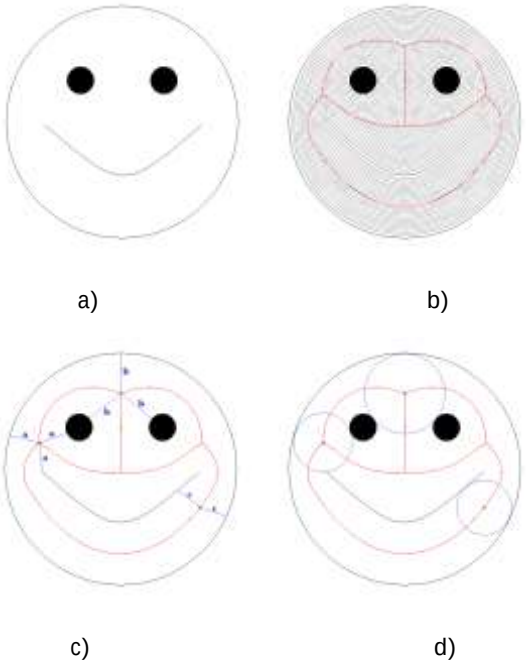


Figure 51 : différentes définitions de l'axe médian, en rouge : a) la "prairie", forme pour laquelle est calculé l'axe médian. b) l'analogie du feu de prairie, l'axe médian est le lieu de rencontre des differants fronts de la propagation de l'incendie en noir. c) l'axe median est l'ensemble des points ayant une projection de cardinal supérieur à deux. d) l'axe median est l'union des centres des disques maximaux (images issues de (MARIE 2014))

Par la suite, de nombreux travaux ont présenté des solutions pour le calculer dans des espaces discrets en 2D. On peut classer ces travaux selon les catégories suivantes :

- Les approches basées sur la morphologie mathématique : les points de la forme sont supprimés un à un dans un processus itératif utilisant un critère de voisinage (Lam, Lee, et Suen 1992),(Liu et al. 2011).

- Les approches basées sur l'utilisation des diagrammes de Voronoï : La frontière de la forme est échantillonnée et les régions de Voronoï sont calculées à partir des points obtenus. Puis les arêtes créées par deux points successifs sont supprimées et la résultante est une approximation de l'axe médian (Bai, Latecki, et Liu 2007).
- Les méthodes basées sur l'analogie du feu de prairie : celui-ci est simulé par divers outils mathématiques ou analytiques tels que des champs de potentiel (Ahuja et Chuang 1997) ou des contours actifs (Leymarie et Levine 1992).
- Les méthodes basées sur la carte de distances et/ou des projections : suite au calcul de ces résultats intermédiaires, un algorithme permet de localiser les points appartenant à l'axe médian en se basant par exemple sur la recherche des maxima locaux (Borgefors, Ragnemalm, et Sanniti di Baja 1989) ou en comparant les projections des points voisins (Chaussard, Couprie, et Talbot 2011).

L'axe médian a tout d'abord était utilisé par Blum pour la classification de forme (H. Blum et Nagel 1978) puis par de nombreux auteurs pour la caractérisation et la reconnaissance de formes (Sebastian et Kimia 2005),(Siddiqi et al. 1998),(Moayer et Fu 1975). D'autres ont par la suite utilisé l'axe médian comme moyen de compression d'images binaires (Brandt et Jain 1991), et enfin dans le contexte de la robotique pour la planification de trajectoires. Par exemple, (Garrido et al. 2011) utilise le squelette d'une carte d'occupation binaire avec un algorithme de fast-matching pour la navigation sûre d'un robot mobile. (Victorino, Rives, et Borrelly 2003) extrait implicitement l'axe médian de l'environnement pour définir une stratégie de contrôle réactive. D'autres approches calculent explicitement (Holleman et Kavraki 2000) ou implicitement (Yang et Brock 2004) le squelette pour aider les algorithmes PRM dans les passages étroits. Comme nous le faisons, certains travaux se basent sur l'axe médian dans l'espace des configurations, notamment le Medial Axis PRM (Wilmarth, Amato, et Stiller 1999) et ses extensions (Lien, Thomas, et Amato 2003),(H. Y. C. Yeh et al. 2014) dont les roadmaps sont générées grâce à des échantillons contraints sur le squelette. Il existe des travaux similaires avec les RRT dont (Denny et al. 2014). Cependant, aucun de ses travaux ne calcule explicitement l'axe médian de l'espace des configurations.

Les propriétés du squelette ont été étudiées par de nombreux auteurs ((Riley et Calabi 1964),(Matheron 1988),(Mattioli 1993) et (Schmitt et Mattioli 1993)). Nous ne citerons ici que celles nous intéressant ou ayant une incidence sur notre travail :

- Positionnement : l'axe médian se situe par définition au plus loin des bords de la forme. Cette position en fait une trajectoire de choix pour l'évitement d'obstacles.
- Finesse : le squelette est considéré d'épaisseur nulle dans le cas continu, dans le cas discret, son épaisseur se limite dans le meilleur des cas à un pixel. L'axe médian d'une forme 2D est donc un graphe, objet idéal pour définir une trajectoire.

- Connectivité : l'axe médian d'une forme entièrement connectée sera connecté. Au contraire, une forme séparée en deux par une frontière verra son axe médian coupé de la même façon. Cette propriété permet d'assurer la complétude d'un planificateur basé sur le squelette.
- Semi continuité : l'axe médian d'une forme subit une forte transformation lorsque la forme est modifiée, même lorsque la modification est mineure (cf. Figure 52). Cela conduit le plus souvent à l'apparition de branches parasites, induisant un besoin de filtrage tant à la génération du squelette que lors de son utilisation.

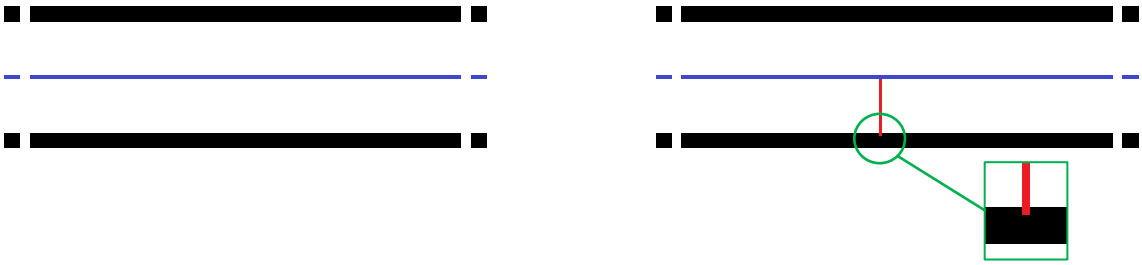


Figure 52 : Exemple d'apparition d'une nouvelle branche (en rouge) de l'axe médian (en bleu) d'un couloir lors d'une modification mineure (disparition d'un pixel).

Dans nos travaux, nous calculons le squelette à l'aide de l'algorithme du delta medial axis (δMA) introduit par (Marie, Labbani-Igbida, et Mouaddib 2013). Celui-ci se base sur la carte des projections et la carte des distances pour déterminer si un point fait partie du squelette ou non : un point appartient à l'axe médian si et seulement si, il existe un disque de rayon δ, dont le centre est sur le segment liant les projections de ce point et d'un de ses voisins, et si ce disque ne contient aucun point de la frontière de la forme. La distance au bord du point doit aussi être plus élevée que celle du voisin.

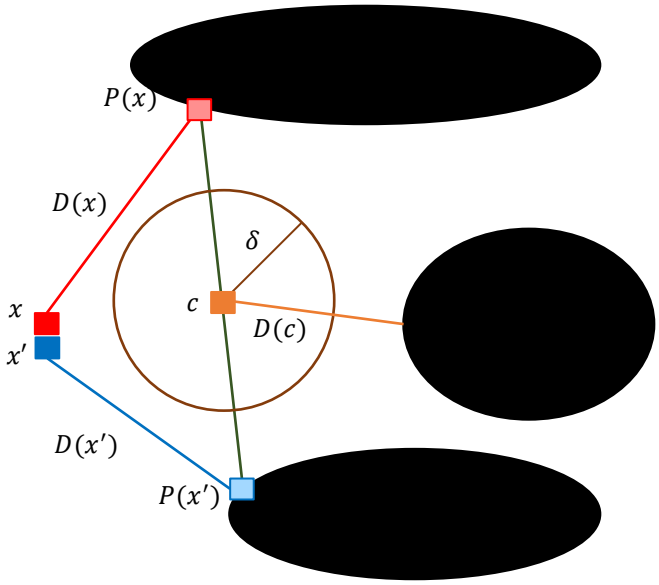


Figure 53 : illustration du delta medial axis

La formulation de ce principe peut être écrite de la façon suivante :

Soit x et x' deux points de χ, nous rappelons que $D(x)$ est la distance euclidienne entre x et le ou les points de $\delta\chi$ les plus proches et $P(x)$ la liste de ces points. Les auteurs, pour des raisons d'efficience algorithmique ne considèrent que le premier élément de cette liste. Nous définissons aussi $\Upsilon(x) = \{x' \mid d(x,x') \leq 1\}$ le voisinage de x correspondant aux points dont la distance euclidienne à x est inférieure ou égale à 1. La Figure 53 présente les notations permettant de définir le delta medial axis comme :

$$\delta MA(\chi) = \{x \in \chi \mid (\exists x' \in \Upsilon(x) \mid \begin{Bmatrix} \exists c \in [P(x), P(x')] \mid D(c) > \delta \\ \& \\ D(x) > D(x') \end{Bmatrix})\} \qquad (4\text{-}16)$$

L'algorithme nous permet donc d'obtenir une approximation du squelette dans un espace discret. De plus, le paramètre δ permet de filtrer les branches parasites pouvant apparaitre lors de déformations mineures de la forme. Il permet aussi de créer un critère de sécurité sur la génération de l'axe médian. En effet, aucune branche ne sera générée dans une partie de la forme où la distance aux obstacles est inférieure à δ. En utilisant le graphe généré par le delta medial axis pour la navigation du robot nous nous assurons de ne pas emprunter de passages trop étroits (trop proche des obstacles). Autre point positif de l'algorithme, il est utilisable en N dimentions. La Figure 54 montre un exemple de squelette calculé dans un espace des configurations en 4D.

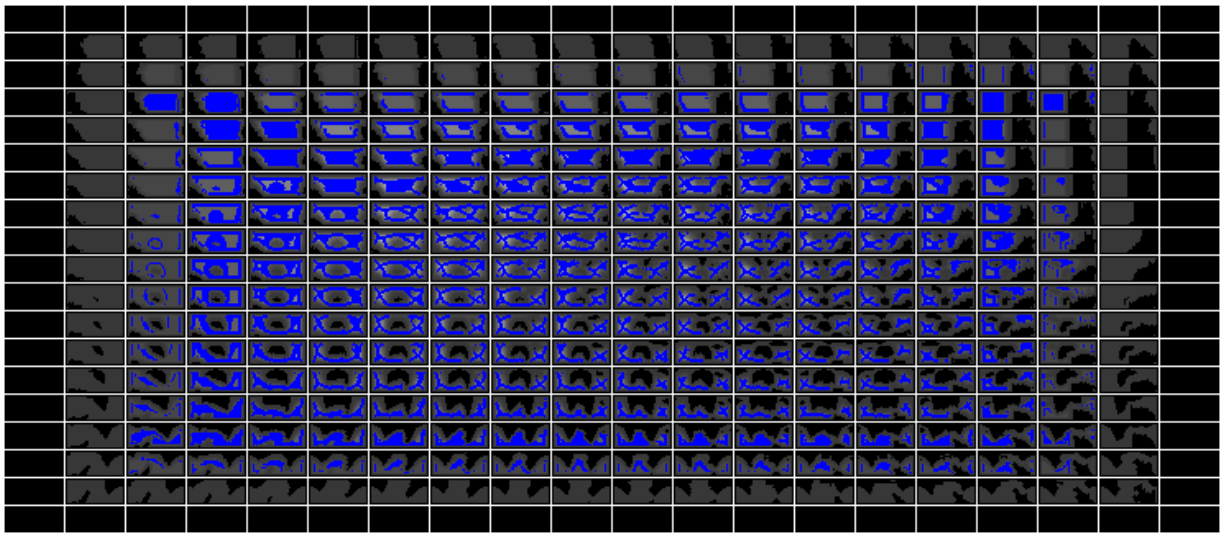


Figure 54 : exemple de squelette (en bleu) obtenu dans un espace en 4 dimensions.

4.3.2 Génération de trajectoires sur l'axe médian

Nous appliquons le δMA sur la partie libre de l'espace des configurations obtenue dans le Chapitre 3 en se basant sur les cartes des distances et des projections dont le calcul préalable est détaillé a été début de ce chapitre. Le déplacement du robot en suivant l'axe médian de l'espace des configurations permet de maximiser la distance aux obstacles

et ainsi la sécurité d'autant plus que la déformation de la carte des distances (2.4 de ce chapitre) permet d'éloigner le squelette des obstacles humains.

4.3.2.1 Rejoindre l'axe médian

La probabilité que les points de départ et d'arrivée du robot dans l'espace des configurations se trouvent sur l'axe médian est faible. En effet, dans les applications de Pick & Place, la destination du robot correspondant la plupart du temps à la pose ou la dépose d'une pièce et donc à une configuration proche des obstacles. Il en est de même pour la position de départ. Pour rejoindre le squelette en toute sécurité, nous allons créer deux branches artificielles au squelette en effectuant une monté de gradient jusqu'à atteindre ce dernier. En reprenant l'analogie du feu de prairie, cela revient à se déplacer en suivant la normale au front de l'incendie jusqu'à rejoindre le point de rencontre des flammes. Chacune des branches e_{added} est donc composée de I points $q_i \in C_{free}$ tels que :

$$e_{added} = \begin{cases} q_1 = q_{init} \\ q_i \mid \{ q_{i+1} \in \Upsilon(q_i) \;\&\; D(q_{i+1}) = \max\big(D(\{\Upsilon(q_i)\})\big) \\ q_I \in \delta MA(C_{free}) \end{cases} \qquad (4\text{-}17)$$

Avec : $q_{init} \in C_{free}$: le point initial de la montée de gradient correspondant au point de départ ou final de la trajectoire du robot.

4.3.2.2 Choisir une trajectoire sur le graphe issu de l'axe médian

Le squelette d'une forme 2D forme un graphe $G = (V, E)$ dans l'espace libre des configurations. Chaque sommet $v \in V$ représente un point d'intérêt topologique, à savoir, une intersection de plusieurs branches de l'axe médian ou la fin de l'une d'entre elles, positionnée à une configuration q. Chaque arrête $e \in E$, caractérisée par sa longueur l, correspond à un chemin sûr reliant entre eux deux sommets. Construire le graphe à partir du δMA n'est pas chose triviale et nous encourageons le lecteur à se référer à (MARIE 2014) pour un exemple de méthode d'extraction.

La recherche d'un chemin sur ce graphe reliant la position initiale du robot q_I à la position finale q_F est effectué en utilisant l'algorithme de Dijsktra. La Figure 55 résume l'obtention du chemin à partir du squelette.

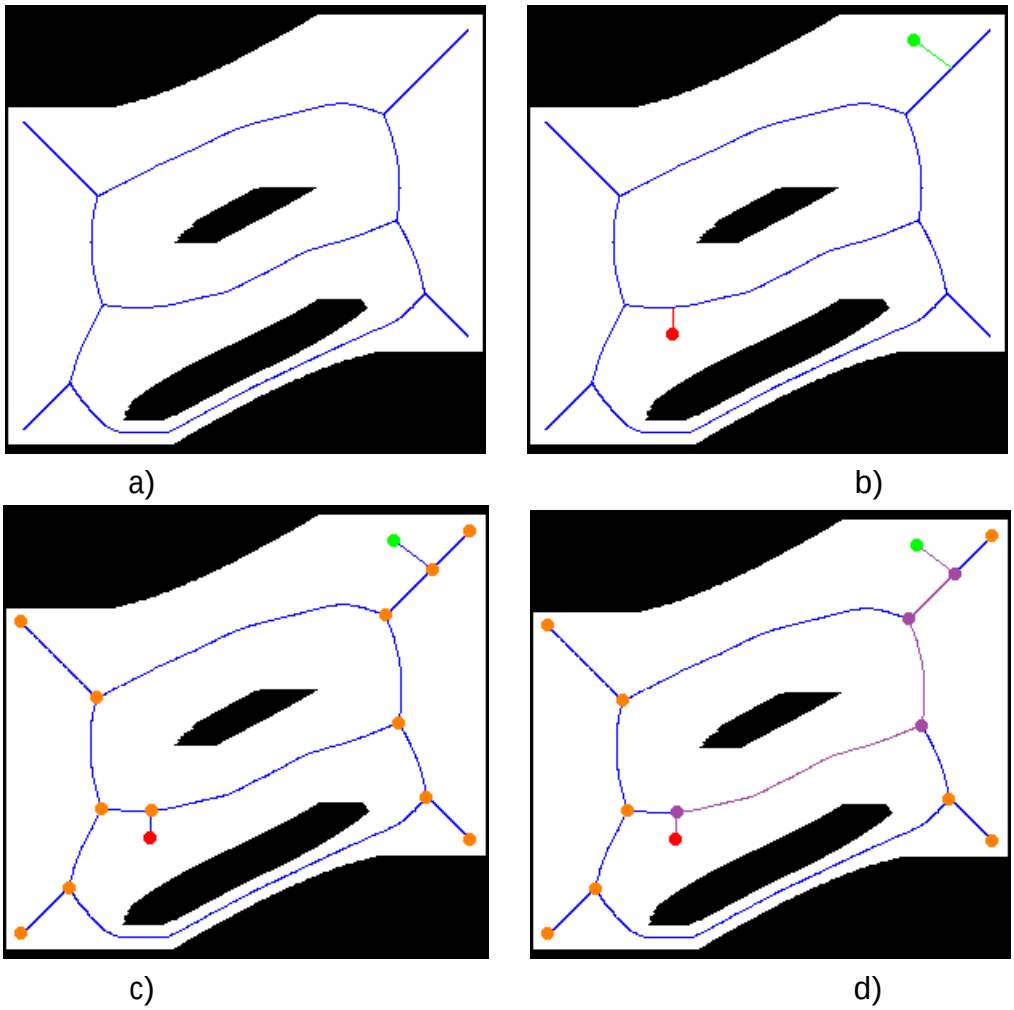


Figure 55 : obtention d'une trajectoire à partir de l'axe médian de l'espace des configurations. a) axe médian (en bleu) de la partie libre de l'espace des configurations. b) ajout des branches reliant les points de départ (vert) et d'arrivée (rouge) au squelette. c) extraction du graphe (arrêtes en bleue et sommets en orange). d) trajectoire sélectionnée par l'algorithme Dijsktra (en violet).

Notons les avantages de cette méthode :

- Le planificateur est complet : si un chemin existe, il le trouvera dans le cas contraire, il indiquera qu'il n'y a pas de solution.
- La planification est rapide : le graphe généré sur l'axe médian est de petite taille (peu de sommets), la recherche de solution est donc rapide.
- La planification est sûre : l'axe médian est par définition aussi loin des obstacles que l'ont peu l'être.

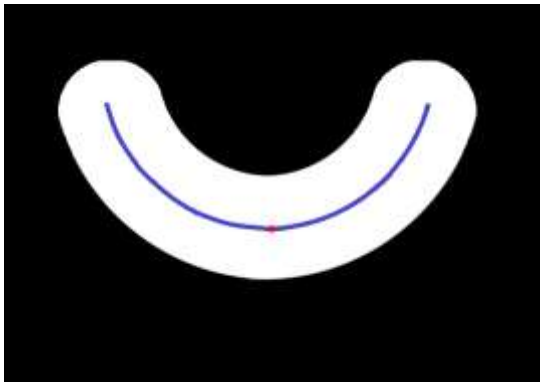
4.3.2.3 Choisir une trajectoire directement sur l'axe médian

Si le squelette est filaire en 2D, ce n'est plus le cas dès la 3ème dimension où celui-ci devient un ensemble de plans puis de volumes en 4D et ainsi de suite. Cette augmentation de dimension du squelette n'est pas sans poser de problème à la stratégie de planification présentée précédemment.

Tout d'abord l'extraction d'un graphe à partir de l'axe médian en ND n'est pas chose aisée. De plus, si même si l'extraction de ce graphe est possible, un nouveau problème se présente devant nous : les arrêtes qui en 2D étaient des lignes deviennent des surfaces en 3D puis des volumes en 4D, et les sommets passent de points à segments, puis à surfaces. L'utilisation d'un algorithme de plus court chemin serait toujours possible, mais il ne donnerait qu'une liste de sommets et arrêtes à traverser, et pas le chemin à choisir sur la surface d'une arrête ou le point d'un « sommet-segment » où effectuer le changement d'arrête.

Il existe quelques solutions à ce problème :

- Calculer de l'axe médian de l'axe médian : si le squelette d'une forme 3D est composé de surfaces, il est possible de calculer le squelette de ces surfaces pour obtenir une structure filaire qui permet l'extraction et l'utilisation aisée d'un graphe. Cette solution présente néanmoins deux défauts, le premier étant la réduction du nombre de configurations utilisables par le robot pour se déplacer conduisant à l'allongement des trajectoires sans pour autant augmenter la distance aux obstacles. Prenons l'exemple de la figure suivante, elle représente la coupe d'une forme 3D correspondant à deux demi-cylindres imbriqués. L'axe médian de cette forme est un demi tube (en bleu), le squelette de ce dernier serait une ligne passant par le point rouge. Une trajectoire générée à partir de cette ligne sera plus longue si on ne cherche pas à traverser cette coupe, cependant tous les points de l'axe médian en bleu sont équidistants du bord de la forme. Le second inconvénient de cette solution est sa complexité calculatoire. En effet, il faudrait calculer $N-1$ squelettes et donc cartes des distances et projections dans des référentiels non orthogonaux, cela allongerait considérablement le temps de calcul et donc le temps d'obtention d'une trajectoire sûre.



- La seconde possibilité est de calculer un chemin directement sur l'axe médian extrait à l'aide d'un algorithme adapté. (Benzaid et al. 2018) utilise par exemple le calcul d'un A* sur un squelette 3D pour la navigation d'un drone.

Cette solution plus simple présente elle aussi l'inconvénient d'être longue à calculer et est non compatible avec notre application.

Pour mener à bien notre planification de trajectoires sur le squelette ND, nous avons donc proposé une méthode permettant de calculer un algorithme de plus court chemin directement sur l'axe médian sans pour autant le calculer explicitement. Ce gain de calcul permet de réaliser une planification suffisamment rapide pour notre application.

Avant de détailler notre méthode de calcul, nous faisons un rappel rapide du fonctionnement d'un algorithme de plus court chemin :

Ces algorithmes fonctionnent à l'aide de deux listes. La première est celle des points déjà testés, la seconde est celle des points à tester.

-Etape 1 : à l'initialisation de l'algorithme, la liste 1 est vide tandis que la seconde contient uniquement le point de départ.

-Etape 2 : le premier point de la liste 2 est sélectionné et ses voisins sont analysés :

- Si un des voisins est l'arrivée, on a trouvé un chemin, l'algorithme se termine
- S'ils ne font pas parti de la liste 1, chacun des voisins est ajouté à la liste des points à tester, le coût de parcours est incrémenté.

-Etape 3 : la liste 2 est triée selon un critère choisi. Dans le cas du Dijkstra, c'est l'ordre d'apparition dans la liste, dans le cas d'un A*, c'est en fonction d'une heuristique H qui correspond à :

$$sH = Co + Di \qquad (4\text{-}18)$$

Avec Co, le coût pour parvenir jusqu'au point considéré, et Di, l'estimation que nous pouvons faire de la distance séparant le point considéré de l'arrivée. A la suite de l'étape 3, l'étape 2 est répétée jusqu'à trouver un chemin.

Notre proposition est de modifier l'étape deux en ajoutant le test du δMA avant d'ajouter les points à la liste des points à tester. Ainsi seuls les points appartenant au squelette seront utilisables par l'algorithme de planification.

4.4 Génération des trajectoires du robot

L'obtention d'un chemin sur l'axe médian permet de déplacer le robot dans l'environnement sans heurter d'obstacles. Néanmoins, cela ne suffit pas à garantir le déplacement sûr et régulier du robot. A cet effet, nous apportons plusieurs ajustements à la trajectoire issue du planificateur avant de l'envoyer au robot. Ce chemin est tout d'abord échantillonné et les traitements suivants sont faits pour chacun des points ainsi obtenus.

4.4.1 Adaptation dynamique de la trajectoire

En premier, nous avons mis au point un lissage de la trajectoire du robot. En effet, comme nous l'avons indiqué précédemment, le squelette peut subir de fortes modifications lors d'un faible changement de l'espace dans lequel il est calculé. L'espace des configurations étant très changeant compte tenu de la dynamicité de notre environnement, il en résulte une variation conséquente de la trajectoire calculée à chaque itération de notre algorithme. Pour éviter des mouvements saccadés du robot, nous avons ajouté une fusion dynamique de la trajectoire courante du robot avec la trajectoire nouvellement calculée.

Cette fusion, illustrée par la Figure 56, doit remplir deux objectifs :

- Permettre une transition souple entre les deux trajectoires,
- En cas de présence d'un obstacle proche de l'ancienne trajectoire, éloigner le résultat de la fusion de celui-ci.

Soit q_k, un point de la trajectoire actuelle du robot, issue de l'itération précédente, et q'_k, un point de la nouvelle trajectoire proposée par le planificateur, $k \in [1, K]$. Nous proposons q^*_k, le point de la trajectoire fusionnée, tel que :

$$q^*_k = q_k * (1 - \beta) + q'_k * \beta \qquad (4\text{-}19)$$

Avec :

$$\beta = \min(1, \frac{k}{K} + 1 - \tanh(\gamma * D_{map}(q_k))) \qquad (4\text{-}20)$$

Remarquons que, ici, D_{map} est calculée à la même itération que q'_k.

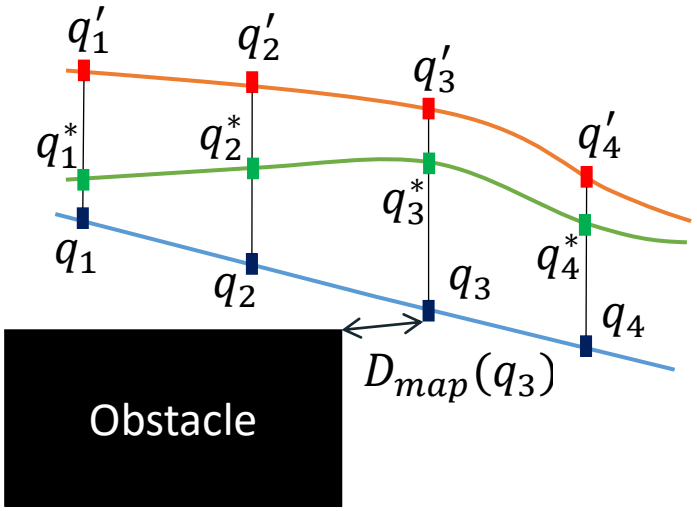


Figure 56 : fusion dynamique des trajectoires en cours (q_i) et nouvellement calculée (q_i')

En considérant le robot au point $k = 0$ de la trajectoire, la variation $\frac{k}{K}$ permet une transition progressive d'une trajectoire à l'autre sur K points. La seconde partie du coefficient β est construite à l'aide de la fonction tangente hyperbolique dont les caractéristiques correspondent à notre besoin. En effet, comprise entre 0 et 1 sur $\mathbb{R}^+$, elle tend vers 1 lorsque son paramètre tend vers l'infini, et sa tangente à l'origine n'est ni nulle ni infinie. La quantité $1 - \tanh(\gamma * D_{map}(q_k))$ évolue donc de 1 vers 0 lorsque la distance aux obstacles augmente. Le paramètre γ permet de régler cette décroissance, et la transition vers les valeurs de distances élevées est douce (cf Figure 57). L'ensemble est borné entre 0 et 1 par la fonction min.

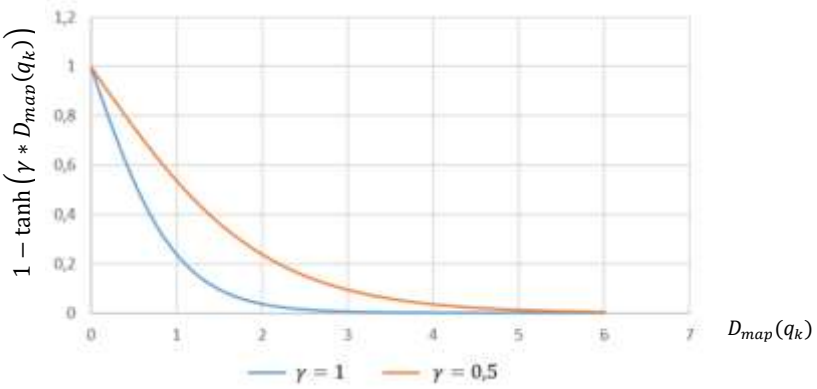


Figure 57 : évolution de $1 - tanh(\gamma * D_{map}(q_k))$ pour différentes valeurs de γ

4.4.2 Modulation de la vitesse du robot

Le second ajout fait à la trajectoire est la modulation de la vitesse. Comme nous l'avons présenté dans la section 2.1, la norme ISO/TS 15066 (AFNOR 2016) prévoit un mode de collaboration où la vitesse du robot doit-être modulée en fonction de la distance séparant l'humain de celui-ci. Cette modulation de vitesse doit notamment permettre au robot de s'arrêter avant d'entrer en collision avec l'humain.

Comme plusieurs travaux précédents, par exemple (Meguenani, Padois, et Bidaud 2015), nous régulons l'énergie cinétique du robot en fonction de la distance de séparation. En effet, le freinage d'un robot consiste à dissiper l'énergie cinétique de celui-ci, nous pouvons donc estimer que la distance d'arrêt du robot est proportionnelle à celle-ci. D'ailleurs, le théorème de l'énergie cinétique nous donne :

$$\Delta E_c = \sum W_{F_{ext/int}} \qquad (4\text{-}21)$$

Avec ΔE_c, la variation d'énergie cinétique du robot et $W_{F_{ext/int}}$, le travail exercé par une force intérieure ou extérieure sur le robot. En considérant qu'il n'y ait pas de contact

extérieur et que les autres forces internes au robot sont négligeables devant les couples de freinage, nous obtenons pour chaque corps du robot :

$$W_{frein,i} = Cp_{frein,i} * dq_i \quad (4\text{-}22)$$

Avec $Cp_{frein,i}$ le couple de freinage de l'articulation i du robot et dq_i la distance parcourue par l'articulation i lors du freinage. Celle-ci est toujours positive et le couple de freinage est négatif, donc le travail est négatif. L'énergie cinétique du robot est la somme des énergies accumulées par chacun des segments et peut s'écrire comme la moitié de la somme des vitesses d'articulation, $\dot{q}_i$, multipliée par l'inertie équivalente à chaque articulation, I_i^q. A l'arrêt du robot, E_c est nulle et donc sa variation devient :

$$-\frac{1}{2}\sum_i I_i^q \dot{q}_i{}^2 = \sum_i Cp_{frein,i} * dq_i \quad (4\text{-}23)$$

La détermination des valeurs d'inertie et de freinage nécessiterait une étude approfondie des capacités de freinage du robot en réalisant de nombreux tests. Etant donné que celles-ci dépendent du robot, de la configuration de ce dernier, de l'outil et plus généralement de la charge embarquée ainsi que de la vitesse du robot, cette étude devrait être reproduite pour chaque application et parfois plusieurs fois (avec charge ou non).

Dans un souci de simplicité et de généricité, notre choix est de chercher une simplification permettant d'approcher la valeur de l'énergie cinétique (en la surestimant) et les capacités de freinage du robot (en les sous-estimant).

Nous souhaitons choisir les vitesses $\dot{q}_i$ de telle sorte que le robot soit capable de s'arrêter avant $D_{map}(q)$, la distance séparant le robot de l'obstacle le plus proche dans l'espace des configurations. $D_{map}(q)$ est la somme quadratique des distances séparant q de l'obstacle selon chacune des dimensions de l'espace. Ces distances selon une dimension sont donc plus petites que $D_{map}(q)$, dire que:

$$dq_i \leq D_{map}(q) \quad (4\text{-}24)$$

Garantie que le robot s'arrêtera avant de toucher l'obstacle. Donc :

$$-\frac{1}{2}\sum I_i^q \dot{q}_i{}^2 \leq \sum Cp_{frein,i} * D_{map}(q) \quad (4\text{-}25)$$

Nous introduisons $\dot{q}_{max}$ tel que pour tout i, $\dot{q}_i < \dot{q}_{max}$, donc :

$$\frac{1}{2}\sum I_i^q \dot{q}^2 \leq \frac{1}{2}\dot{q}_{max}^2 \sum I_i^q \quad (4\text{-}26)$$

De la même façon nous choisissons I_{max} tel qu'il représente l'inertie du robot au pire cas, et $C_{frein,min}$ le couple de freinage minimum d'une articulation du robot. Dès lors :

$$\frac{1}{2}\sum \dot{q}_{max}^2 * I_{max} \leq \sum D_{map}(q) * Cp_{frein,min} \quad (4\text{-}27)$$

$$\frac{1}{2} * \dot{q}_{max}^2 * N * I_{max} \leq D_{map}(q) * N * Cp_{frein,min} \quad (4\text{-}28)$$

D'où :

$$\dot{q}_{max} = \lambda * \sqrt{D_{map}(q)} \quad (4\text{-}29)$$

Avec :

$$\lambda = \sqrt{\frac{2 * Cp_{frein,min}}{I_{max}}} \quad (4\text{-}30)$$

Nous pouvons donc calculer une valeur de vitesse maximale pour toutes les articulations permettant de garantir que le robot s'arrêtera avant de toucher l'obstacle. Evidement, compte tenu des simplification faites, ce calcul est très pessimiste et conduira à des vitesses très faibles de déplacement du robot. Cependant, dans nos applications, nous utiliserons des robots collaboratifs qui assurent par conception la sécurité de l'opérateur. Ainsi, il n'est pas nécessaire que le robot soit arrêté avant un contact. Nous adapterons la valeur de λ de façon à limiter l'énergie cinétique du robot en cas de collision, et apporter une sensation de sécurité à l'opérateur.

La vitesse de déplacement du robot est alors bornée par une vitesse maximale correspondant aux capacités du robot en fonctionnement collaboratif. De plus, lorsque le robot est trop proche des obstacles, la vitesse de déplacement est nulle évitant ainsi que le robot se déplace ou se mette en mouvement. La distance de sécurité d_{secu} est choisie telle que $d_{secu} < \delta$. De cette façon, il n'existe aucune branche de l'axe médian où le robot ne pourrait se déplacer.

Suite à ces ajustements, la trajectoire du robot est une suite de points de passage associés chacun à une vitesse qui peut être envoyée au contrôleur du robot afin de générer le mouvement. Cette trajectoire, conçue pour être sûre, sera alors exécutée dans l'environnement collaboratif. Le chapitre suivant présente les manipulations effectuées durant la thèse pour montrer les apports de notre méthodologie et les résultats associés.

Chapitre 5.
Expérimentations et résultats

Synthèse :

Dans ce dernier chapitre, nous avons présenté les expérimentations réalisées pendant la thèse ainsi que les résultats et les conclusions qui en découlent. Le premier lot d'expérimentations réalisées dans un cas simplifié, un robot à deux axes dans un monde 2D, a permis de mettre en évidence l'intérêt d'utiliser l'axe médian comme support des trajectoires. En effet, celui-ci permet d'éloigner le robot de l'humain, apportant ainsi un gain de sécurité. Nous avons aussi pu mettre en évidence dans ces tests l'intérêt de la fusion dynamique de trajectoires, qui permet d'obtenir un mouvement fluide du robot. Les résultats, issus du second lot d'expériences réalisées sur un robot à 7 degrés de libertés dans un monde 3D, montrent, en premier lieu, les aptitudes de notre travail pour la planification de trajectoires dans un environnement dynamique. L'utilisation d'indicateurs de risque, dédiés à la cobotique, a aussi permis de mettre en avant l'intérêt de prendre en compte l'humain dans la planification de trajectoires et de confirmer la pertinence de l'axe médian pour ce type d'applications.

Sommaire du chapitre :

5.1 Introduction

Les chapitres précédents ont permis de contextualiser le travail présenté ici, puis de détailler les solutions que nous avons proposées et mises en place pour répondre aux challenges qui nous sont adressés. Ce dernier chapitre présente les expérimentations qui ont été menées durant le travail de thèse, afin de mettre en avant les apports de notre méthode. Les résultats issus de celles-ci sont ensuite présentés et discutés.

Tout d'abord, nous présenterons les premiers tests qui ont été effectués sur un robot à deux articulations dans un monde en 2D. Ceux-ci ont pour objectif de montrer la pertinence d'un planificateur basé sur l'axe médian dans le contexte de la cobotique. Puis, nous présenterons le démonstrateur des travaux de thèse réalisé à l'aide d'un bras cobotique à 7 axes dans un environnement 3D. Celui-ci simule le fonctionnement d'un ilot de cuisson de l'usine Valeo et tend à démontrer l'intérêt d'une application collaborative dans ce contexte.

5.2 Expérimentation 1 : le robot SCARA dans un monde 2D

Le premier groupe d'expérimentations a été réalisé sur une version réduite du démonstrateur, composé d'un robot 2D dans un espace de travail plan. Celles-ci ont fait l'objet d'une publication (Fuseiller et al. 2018). Cette partie présente le protocole expérimental, l'ensemble du dispositif de test, ainsi que les résultats.

5.2.1 Objectifs et méthode d'évaluation

Les objectifs principaux de ces tests sont :

- Montrer la pertinence des trajectoires issues de l'axe médian dans le contexte collaboratif. Nous avons pour cela comparé les chemins de notre planificateur avec ceux d'un planificateur A*. Ce dernier permet d'obtenir une trajectoire parmi les plus courtes. Nous comparons le temps de parcours du robot ainsi que les distances aux obstacles dans l'espace de travail et dans l'espace des configurations.
- Montrer l'utilité de la modulation de vitesse et de la fusion de trajectoires dans un environnement dynamique. Nous étudions l'apport des modifications de trajectoires dynamiques en présence d'objets mobiles et des opérateurs.

5.2.2 Cas d'environnements statique

5.2.2.1 Dispositif et scénario expérimental :

Ces expériences ont été menées à bien avec un robot SCARA modèle IRB-910sc de la marque ABB[3] posé sur une table carrée de 1.6m de côté. La scène est observée par une

[3] Pour plus de détails : http://new.abb.com/products/robotics/industrialrobots/irb-910sc

caméra Time of Flight Balser ToF[4]. Cette dernière fournit un nuage de points avec une résolution de 640*480 à 15Hz prétraité par des filtres temporel et spatial, ainsi qu'un algorithme de rejection des outliers. Comme présenté dans le Chapitre 3, cette entrée est utilisée pour construire une grille d'occupation 2D avec une résolution de 1cm.

Dans un premier temps, nous avons appliqué notre approche en environnement réel avec des obstacles (cf. Figure 58). Nous l'avons ensuite testé en simulation sur douze environnements générés aléatoirement pour comparer notre planificateur face à un algorithme A*. Un échantillon des environnements de test est présenté dans la Figure 59, les autres sont visibles dans l'Annexe Annexe 5 :. L'objectif du robot est d'atteindre deux points cibles en traversant l'environnement de gauche à droite, puis de faire le chemin inverse à l'aide de chacun des planificateurs.

Pour l'algorithme A*, les obstacles sont augmentés de la valeur de δ afin qu'il respecte la distance de sécurité (cela apparait en gris foncé sur les images de l'espace des configurations de la Figure 59). Nous observons que l'algorithme A* donne deux trajectoires différentes à l'aller et au retour du robot, alors que notre planificateur donne le même résultat. Nous appliquons sur ces trois trajectoires la modulation de vitesse.

Figure 58 : robot SCARA IRB-910sc de ABB dans l'environnement de test réel

Premier point de comparaison : le temps de parcourt. Celui-ci est obtenue en simulant le mouvement du robot selon les trois trajectoires. Les deux autres critères sont la distance séparant le robot des obstacles dans l'espace des configurations (par un relevé de la position du robot sur la carte des distances calculée dans l'espace des configurations) et dans l'espace de travail (calcul de la distance la plus courte entre les points obstacles

[4] Pour plus de détails : https://www.baslerweb.com/en/products/cameras/3d-cameras/time-of-flight-camera/tof640-20gm_850nm/

de la carte d'occupation et ceux du robot). Ces distances sont présentées pour l'environnement 1 dans la Figure 61 (et pour les autres en Annexe Annexe 5 :) en fonction de la progression le long de la trajectoire. Nous avons pour cela normalisé la longueur de parcours à 1 (pour faciliter l'analyse des courbes, le retour du A* est présenté de la fin vers le début). Afin de pouvoir comparer quantitativement les distances pour l'ensemble des trajectoires, nous calculons, pour les deux espaces, les distances cumulées tout au long de la trajectoire :

$$D_{cumul} = \int_{t=0}^{t=1} D_{obs}(t)dt \qquad (5\text{-}1)$$

5.2.2.2 Résultats

Nous pouvons distinguer, dans les résultats de ces essais, deux comportements différents selon la disposition de l'espace des configurations :

- Les environnements peu encombrés (3 et 11) dans lesquels les configurations de départ et d'arrivée peuvent être reliés par une ligne droite sans rencontrer d'obstacles (Figure 59 d).
- Les environnements encombrés dans lesquels il est nécessaire de contourner les obstacles dans l'espace des configurations pour rejoindre l'arrivée (Figure 59 b et f).

Nous observons que le planificateur proposé produit des trajectoires plus rapides que l'algorithme A* dans les environnements encombrés. En effet, bien que les trajectoires produites par l'algorithme A* soient plus courtes, celles-ci sont proches des obstacles et la modulation de vitesse permet à nos chemins plus éloignés d'être parcourus plus rapidement. Ce résultat est important car dans notre contexte de production industrielle, nous pouvons assurer la sécurité et le confort de l'opérateur en s'éloignant sans pour autant augmenter le temps d'exécution des trajectoires du robot. Dans les environnements peu encombrés, l'absence d'obstacles permet des vitesses de déplacement identiques et l'algorithme A* devient plus rapide. La Figure 60 a) donne l'ensemble des temps de parcours.

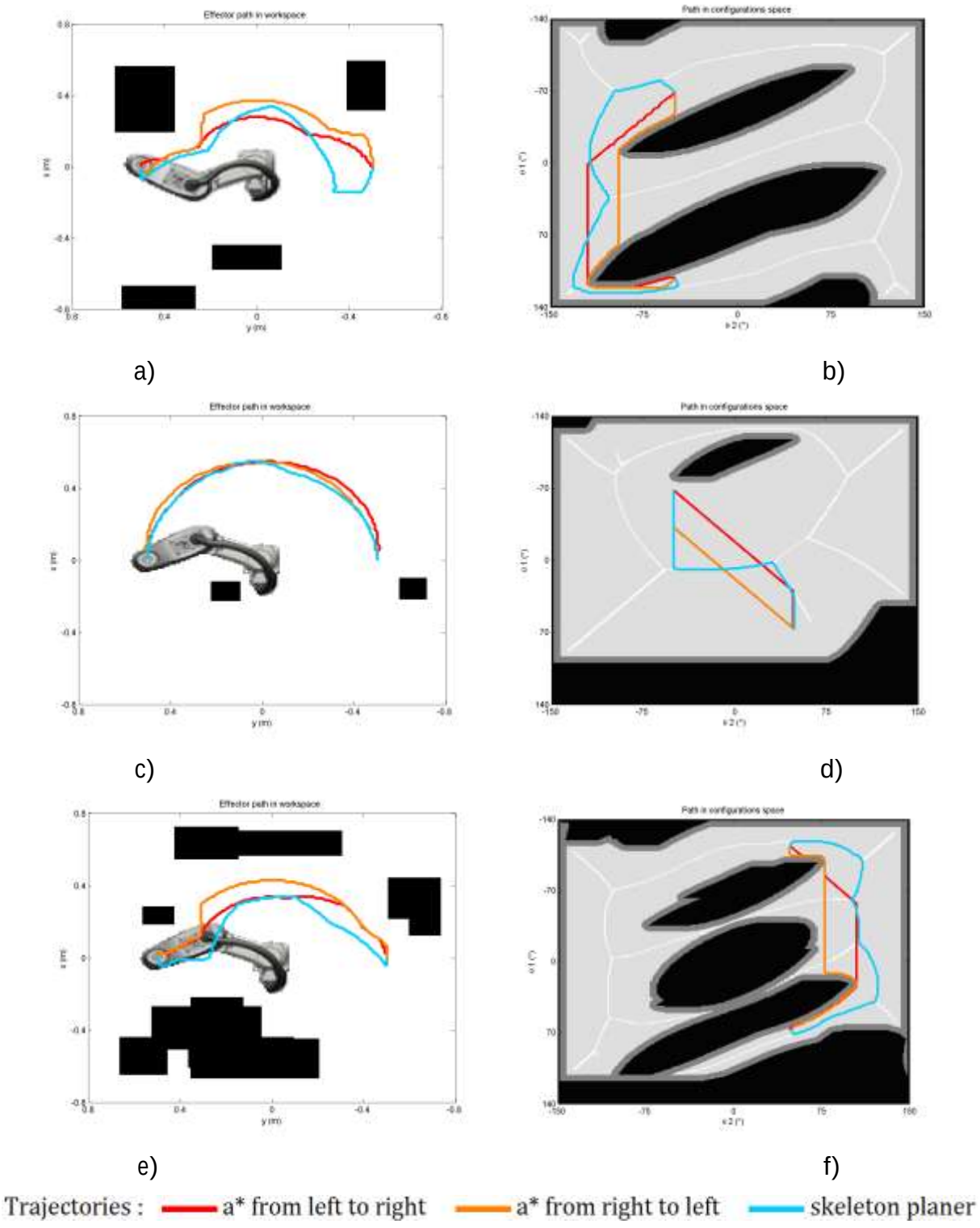


Figure 59 : Exemples de trajectoires de l'effecteur dans l'espace de travail (colonne de gauche) et des trajectoires du robot dans l'espace des configurations (colonne de droite) pour les environnements 1 (a et b), 3 (c et d) et 12 (e et f).

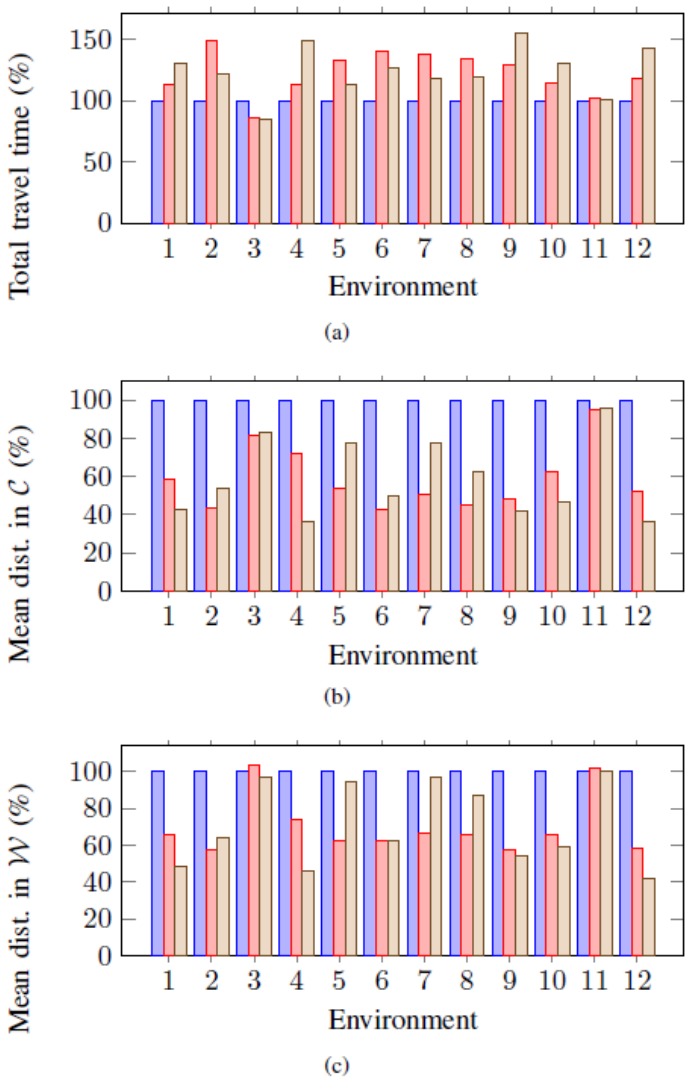


Figure 60 : Comparaison des trajectoires en terme de temps d'exécution (a), de distance cumulée aux obstacles dans l'espace des configurations (b) et dans l'espace de travail (c). Les valeurs sont en pourcentage, notre planificateur étant la référence (en bleu) (le A de gauche à droite en rouge et de droite à gauche en orange).*

En considérant la distance aux obstacles dans l'espace des configurations, nous observons que notre algorithme est toujours le plus éloigné. La Figure 61 montre l'exemple de l'environnement 1. Nous pouvons y observer qu'à tous les instants, notre trajectoire est la plus éloignée (il arrive que certains points des trajectoires du A* apparaissent plus éloignés, cependant il s'agit d'un effet de la normalisation des trajectoires qui est temporelle et non spatiale, ne permettant donc pas de superposer parfaitement le passage des mêmes zones). Cette observation est valable pour l'ensemble des environnements (voir Figure 60 b) et pour le détail, les courbes en Annexe Annexe 5 :). Cela permet de démontrer tout l'intérêt de l'axe médian comme support à la planification. Dans les environnements peu encombrés, la différence entre les chemins est moins prononcée, compte tenu de l'absence d'obstacle, il est normale que l'algorithme A* en soit presque autant éloigné que notre méthode.

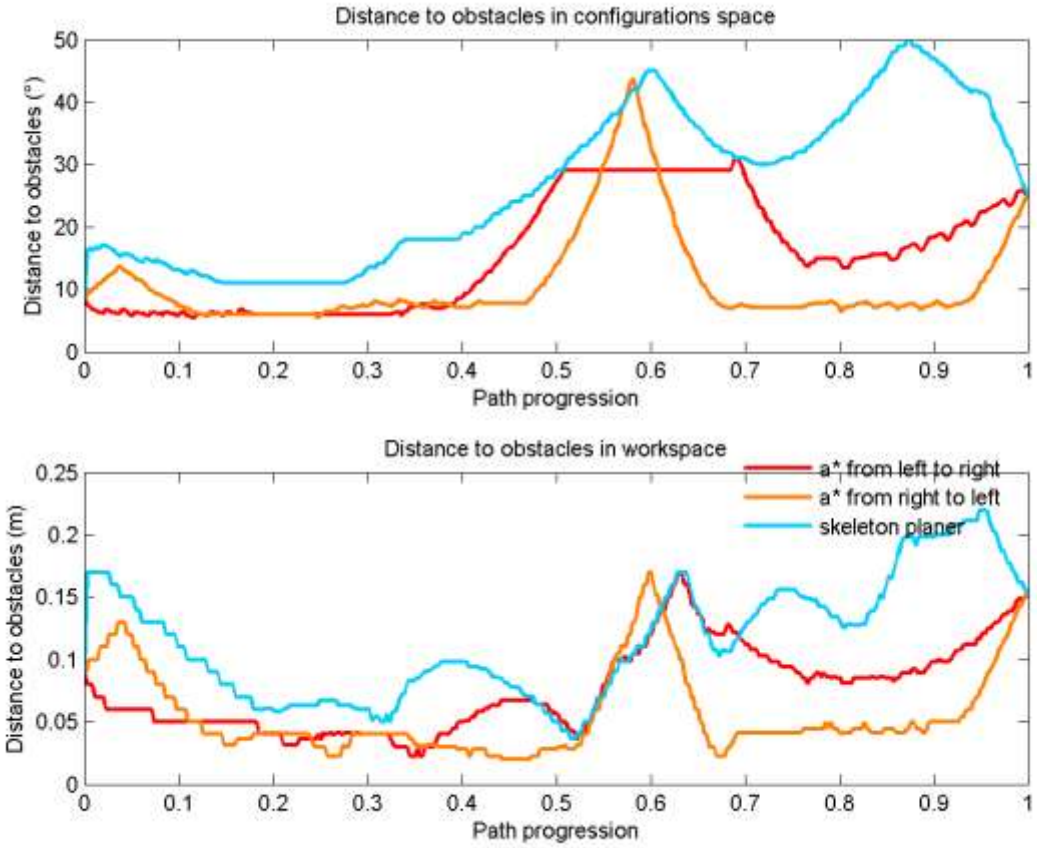


Figure 61 : Distance aux obstacles du robot dans l'environnement 1.

Concernant le critère distance aux obstacles dans l'espace de travail, la Figure 60 c) montre la comparaison des trois trajectoires, ici aussi notre méthode supplante le A* dans les environnements encombrés. La complexité géométrique du robot dans l'espace de travail rend les possibilités de s'éloigner des obstacles plus faibles. En observant la trajectoire de l'effecteur sur la Figure 59 e), nous notons que celui-ci s'éloigne le plus possible des obstacles. La Figure 61, montrant le détail pour l'environnement 1, permet de constater qu'à nouveau la distance à chaque instant de notre algorithme est supérieure au A* (le même problème de normalisation transpose une partie des courbes du A*) et cette observation reste valable pour l'ensemble des environnements, excepté pour ceux faiblement encombrés. Pour ces derniers, les valeurs de distances sont similaires dans les deux cas, car les trajectoires sont semblables et les obstacles très éloignés (voir Figure 59 c).

5.2.3 Cas d'environnements dynamique

5.2.3.1 Dispositif et scénario expérimental :

Le robot SCARA ayant été prêté par ABB et installé dans l'usine de Valeo, les règles de sécurité ne permettent pas de placer un humain directement dans l'espace de travail du robot. Pour pallier cela, nous avons mis au point le dispositif suivant : un opérateur est

installé devant une table, laquelle est observée par la caméra 3D positionnée au plafond. Face à l'opérateur est placé un écran représentant le robot dans l'espace de travail ainsi simulé tel que le montre la Figure 62. Un programme superviseur demande à intervalle régulier au robot de traverser la table de part en part, dans un sens puis dans l'autre, selon la perception faite de la scène avec l'opérateur et notre planificateur. Deux obstacles fixes sont disposés de part et d'autre de la table et l'humain a pour objectif de déplacer ses mains de manière à perturber le déplacement du robot. Durant cette expérimentation, la fusion des cartes des distances (humain et obstacle) n'est pas utilisée et tous les obstacles sont considérés identiques.

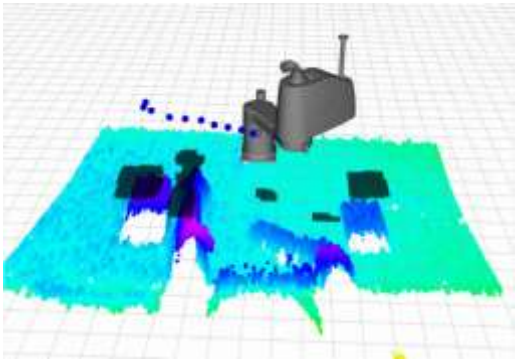
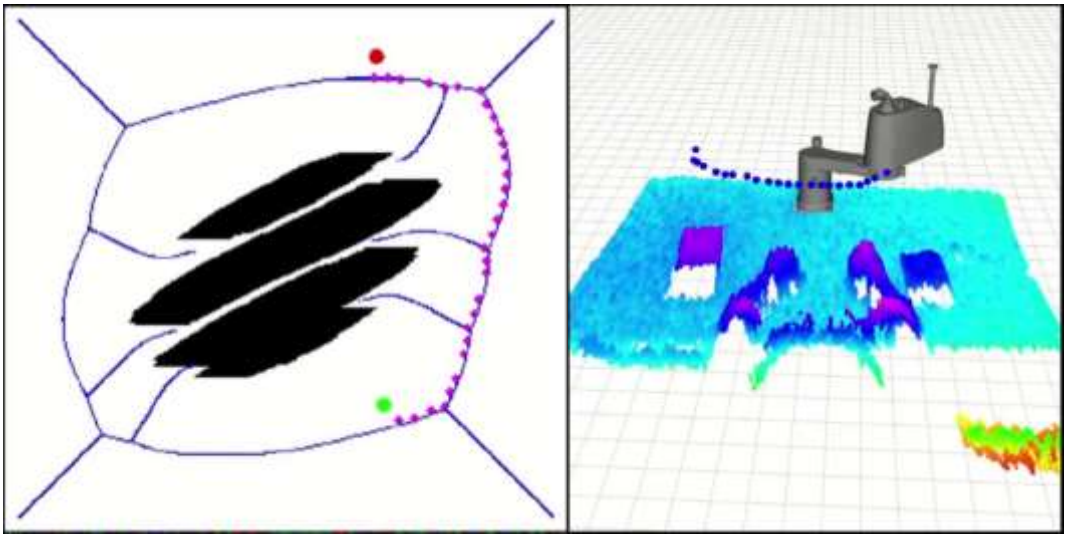


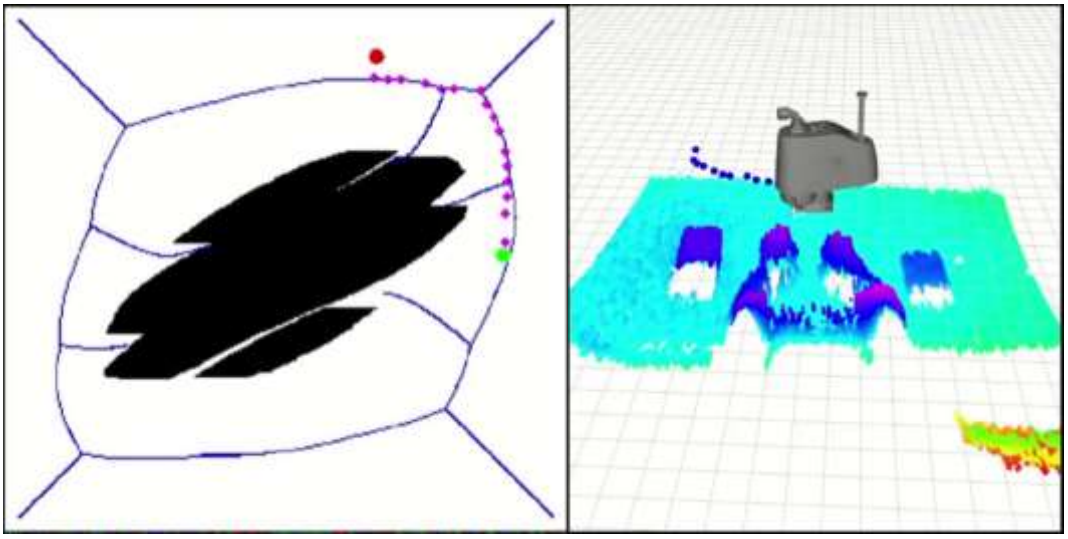
Figure 62 : Représentation du robot simulé dans l'environnement de travail réel observé par la caméra 3D. les points colorés correspondent au nuage de points issus de la caméra 3D, la surimpression en noir à la carte d'occupation et les points bleus à la trajectoire prévue de l'effecteur.

5.2.3.2 Résultats

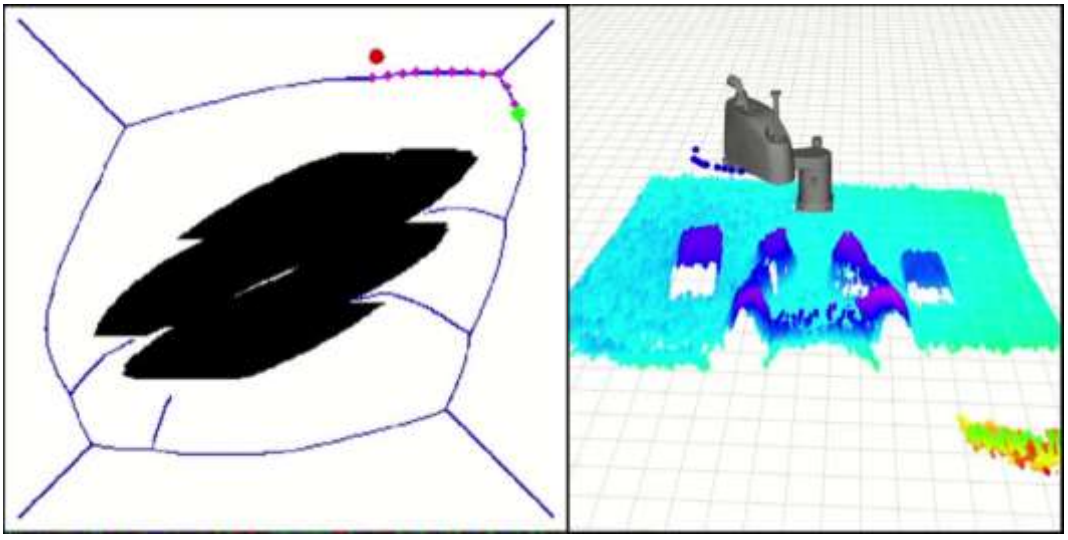
Les résultats de cette expérimentation montrent la capacité de notre planificateur à créer des trajectoires sûres pour traverser la table. La Figure 63 montre différents instants d'une trajectoire d'évitement des mains de l'opérateur. Le chemin planifié pour le robot évolue en fonction de la variation des obstacles dans l'espace des configurations, due aux mouvements des mains de l'opérateur. La fusion dynamique des trajectoires permet des transitions douces lors des modifications de l'axe médian. Nous pouvons apercevoir l'effet de celle-ci aux instants b) et d), les quelques points de la trajectoire (en rose) qui ne se superposent pas avec le squelette (en bleu) en sont issus. De plus, les tests ont confirmé que les algorithmes arrêtent le robot lorsque l'humain obstrue complètement le passage. L'ensemble des traitements de cette expérience ont été réalisés à 15Hz, cette fréquence s'est révélée satisfaisante pour l'évitement dynamique de l'humain, dont les mains se déplacent à environ 0,5 m/s.



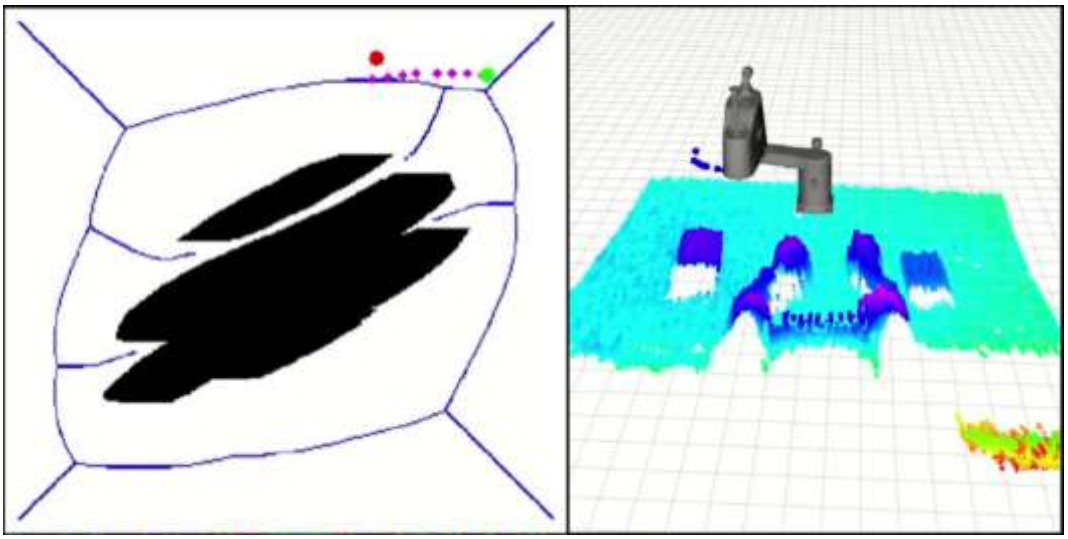
a) t= 0



b) t=1s



c) t=2s



d) t=2s 200ms

Figure 63 : Chronogramme d'un évitement dynamique des mains de l'opérateur. A gauche l'espace des configurations contenant les obstacles en noir, le squelette en bleu, l'objectif du robot en rouge et sa position actuelle en vert. La suite de points en violet correspond à la trajectoire planifiée.

5.2.4 Conclusions

Ce premier lot de tests a permis de mettre en avant dans un contexte simplifié les apports et les capacités de notre approche. Notre planificateur est en effet capable de construire des trajectoires sûres pour la collaboration Homme robot. Celles-ci sont mises à jour dynamiquement pour s'adapter aux mouvements de l'humain, sans que cela ne produise de saccades, avec suffisamment de réactivité. La productivité du robot est maintenue dans ce contexte en modulant la vitesse du robot pour être plus rapide lorsque cela est possible tout en maintenant, à proximité des opérateurs, des déplacements sûrs et socialement acceptables.

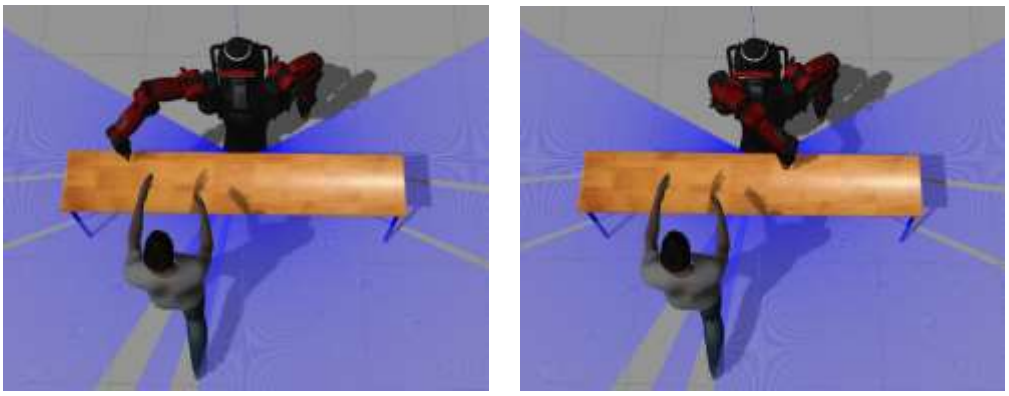
5.3 Expérimentation 2 : le cobot Baxter dans un monde 3D

Le second lot d'expérimentations a été réalisé en simulation sur un robot à 7 axes dans un monde 3D. L'objectif est de montrer la capacité de notre approche dans un travail collaboratif entre un humain et un robot. Nous analysons dans les résultats suivant l'apport de la considération de l'humain dans l'espace des configurations pour la génération de trajectoires sûres.

5.3.1 Dispositif et scénario expérimental

Nous utilisons pour ces essais le simulateur Gazebo intégré à l'environnent ROS (Robot Operating System). Celui-ci simule le fonctionnement d'un cobot Baxter de Rethink Robotics pourvu de 2 bras de 7 axes. Nous planifions les trajectoires du bras droit uniquement, le bras gauche étant positionné en replis. Devant le robot est disposé une

table représentant l'espace de travail collaboratif. Un modèle d'humain est placé face au robot, les bras devant lui de façon à perturber la trajectoire du robot (cf. Figure 64). L'objectif du robot et de traversé l'espace collaboratif en partant d'un emplacement à gauche de la table pour rejoindre le côté opposé. L'ensemble de la scène est observé par deux types de capteurs : un lidar placé juste devant le robot à 20cm de haut ainsi que deux caméras 3D placées en hauteur au-dessus et de part et d'autre du robot, l'ensemble cadencé à 10Hz.



a) b)

Figure 64 : Scène de travail de la simulation de coopération homme robot : a) robot dans la position de départ, b) robot dans la position finale.

A partir des données capteurs issues de la simulation, nous construisons notre représentation de l'environnement comme illustrée par la Figure 65. La grille d'occupation est constituée de cellules de 5 cm de côté et couvre une surface de 2m par 2m sur une hauteur de 3m centrée à la base du bras manipulateur. Les cellules labélisées *obstacle* et *humain* sont ensuite projetés dans l'espace des configurations grâce à notre Look Up Table. Lorsqu'un nouvel objectif de positionnement est envoyé, notre planificateur calcul en ligne un chemin sur l'axe médian. La suite de points ainsi générée est alors envoyée au contrôleur de la simulation du robot.

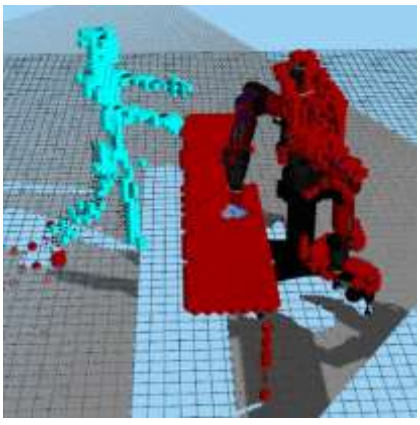


Figure 65 : Carte d'occupation issue de la simulation. Les cellules rouges sont des obstacles, les bleues sont des obstacles humains. Les autres types ne sont pas représentés pour plus de clarté.

Nous comparons plusieurs trajectoires réalisées dans cet environnement en faisant varier les deux paramètres principaux de notre planificateur : δ, la valeur de filtrage de l'axe médian, et k_h, le coefficient de la fusion des cartes de distances tout obstacle et obstacle humain.

5.3.2 Critères d'évaluation

Afin de montrer les apports de notre méthode, nous utilisons la méthode d'évaluation de risque introduite par (Fischer 2018) en l'adaptant à notre application. Celle-ci permet d'estimer en temps réel le risque d'une interaction Homme robot. Elle se compose de trois indicateurs dont nous détaillons la méthode de calcul à la suite :

- K : La probabilité d'un évènement dangereux.
- S : La sévérité d'un éventuel événement.
- P : La possibilité d'évitement de l'éventuel événement.

5.3.2.1 Calcul de l'indicateur K

La valeur de K est conditionnée à la distance séparant l'humain et le robot, D_{HR}. Celle-ci est comparée à la distance nécessaire au robot pour s'arrêter avant de toucher l'humain. L'auteur la nomme ρ_{tot} et la définit comme :

$$\rho_{tot} = D_{corpsR} + D_{arret} + D_{secuR} + D_{secuH} + D_{corpsH} \quad (5\text{-}2)$$

Ou ces distances, illustrées dans la Figure 66 correspondent à

- D_{corpsR} : la distance séparant le point considéré du robot à la limite de son enveloppe corporelle.
- D_{arret} : la distance nécessaire à l'arrêt du robot, prenant notamment en compte le temps de réaction de la chaine de commande et la distance parcourue par le robot durant le freinage.
- D_{secuR} : une distance choisie pour assurer la sécurité de l'interaction.
- D_{secuH} : la distance de sécurité pour l'humain, elle correspond à la longueur d'un bras qui d'après la norme ISO13855 (AFNOR 2010), est de 85cm.
- D_{corpsH} : la distance entre la ligne centrale de l'humain et la limite du tronc de celui-ci.

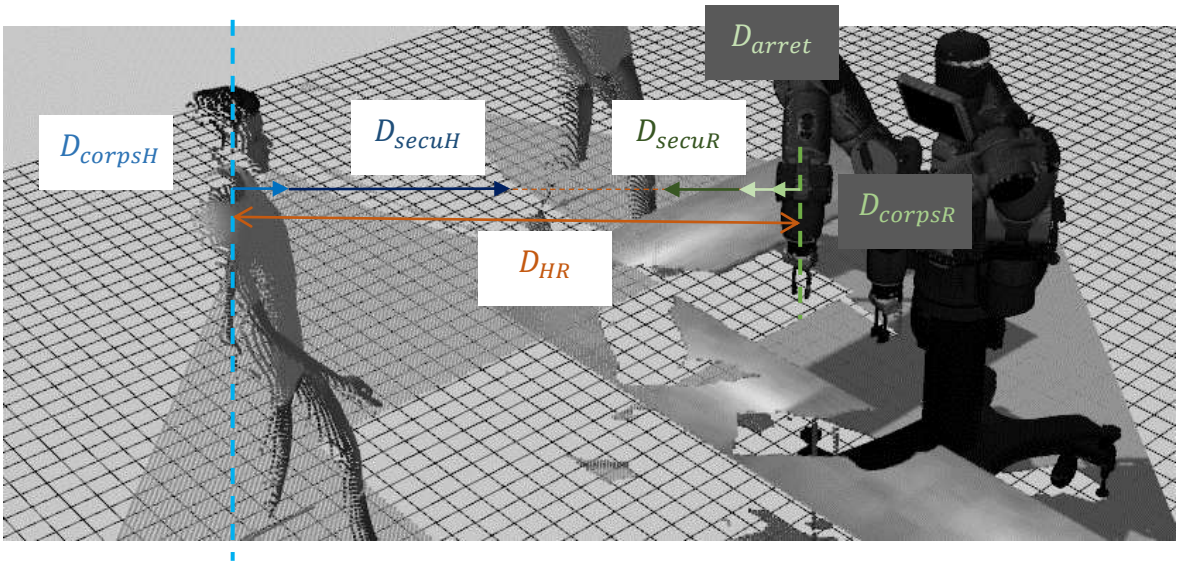


Figure 66 : Illustration des distances utilisées dans le calcul de l'indicateur K.

K se définit alors comme :

$$K = \begin{cases} 0 \; si \; D_{HR} \geq \rho_{tot} \\ \dfrac{\rho_{tot} - D_{HR}}{\rho_{tot} - D_{corpsR} - D_{corpsH}} \; si \; D_{corpsH} + D_{corpsR} < D_{HR} < \rho_{tot} \\ 1 \; si \; D_{corpsH} + D_{corpsR} \geq D_{HR} \end{cases} \quad (5\text{-}3)$$

Il est important de noter que dans ces travaux, l'humain n'est connu que par son squelette. Dans notre application, il est décrit par un ensemble de cellules de la carte d'occupation. Nous considérerons donc que D_{corpsH} comme nulle et évaluons K, S et P par rapport à la cellule *humain* la plus proche.

Nous choisissons $D_{arret} + D_{secuR}$ = 15cm. En effet, nous ne pouvons déterminer à chaque instant la valeur exacte de D_{arret} mais compte tenu des vitesses faibles de déplacement, elle est inclus dans la distance de sécurité.

5.3.2.2 Calcul de l'indicateur S

La valeur de S est donnée par la formule suivante :

$$S = C_{ol} * D_e * L_f * \left(1 + (M_{robot} * V_{robot} + M_{humain} * V_{humain}) * V_{convergence}\right) \quad (5\text{-}4)$$

Avec :

C_{ol} : l'indice de collaboration, il permet d'intégrer les capacités propres d'un robot à la collaboration. La valeur évolue de 10 pour un robot industriel classique et est réduite en fonction des capacités du robot (ex : compliance, matériaux amortissants, réaction dynamique). En se basant sur des valeurs exemples indicatives de l'auteur, nous

choisissons une valeur de 3 compte tenu que notre robot est à la fois compliant et intègre une détection de chocs et que notre planificateur est dynamique.

D_e : est la dangerosité de l'outil, elle traduit la capacité de l'outil à infliger des blessures. Un outil très dangereux tel une lame aura une valeur de 10 tandis qu'un outil sans danger tel que notre ventouse est coté à 2.

L_f : est le facteur de membre, il est lié aux forces admissibles par chacune des parties du corps. En pratique, il est donné en fonction de la hauteur de travail du robot : si l'outil est à hauteur de tête, il est égal à 1, 0.8 s'il est entre le bassin et les épaules et 0.5 s'il est plus bas.

M_{robot} et M_{humain} sont les masses du robot et de l'humain, V_{robot} et V_{humain} sont les vitesses respectives.

$V_{convergence}$ est la vitesse de convergence de l'humain et du robot, elle est illustrée dans la Figure 67 et calculée de la façon suivante :

$$\begin{aligned} \overrightarrow{V_{projrobot}} &= \overrightarrow{V_{robot}} \cdot \overrightarrow{U_{HR}} * \overrightarrow{U_{HR}} \\ \overrightarrow{V_{projhumain}} &= \overrightarrow{V_{humain}} \cdot \overrightarrow{U_{HR}} * \overrightarrow{U_{HR}} \\ \overrightarrow{V_{convergence}} &= \overrightarrow{V_{projrobot}} - \overrightarrow{V_{projhumain}} \end{aligned} \qquad (5\text{-}5)$$

Avec $\overrightarrow{U_{HR}}$ le vecteur unitaire dans la direction Homme robot.

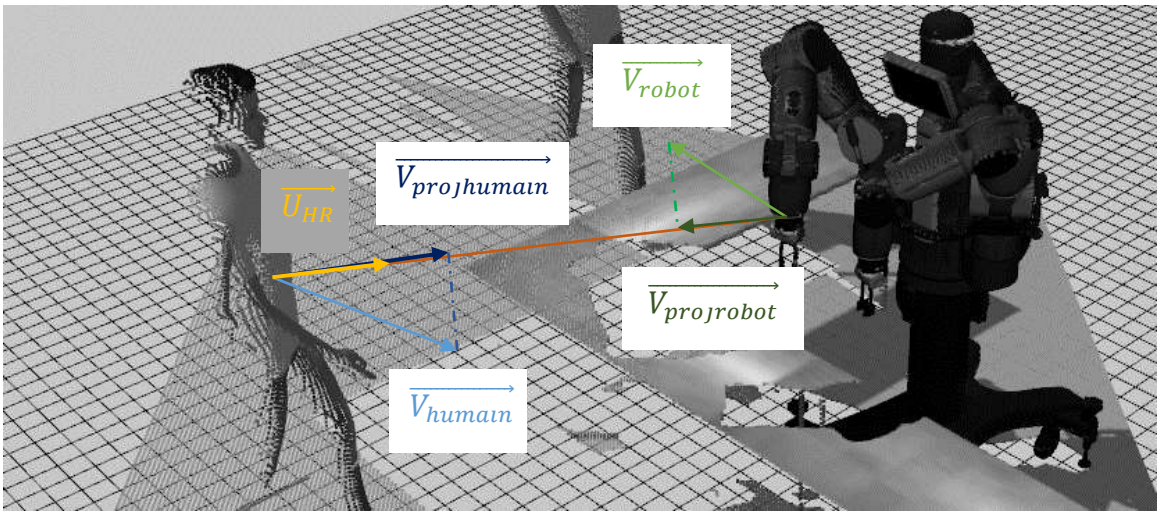


Figure 67 : illustration du calcul de la vitesse de convergence

Dans notre application, nous prendrons en compte la vitesse de l'effecteur ainsi que la vitesse de l'humain en entier, c'est-à-dire la vitesse de déplacement du tronc.

5.3.2.3 Calcul de P

Le calcul de P est sujet à deux coefficients :

E : l'encombrement de la zone de travail, il évolue de 1 à 10, 10 étant un environnement encombré de telle sorte qu'il serait impossible de se dégager lors d'un contact avec le robot, à l'inverse, 1 correspond à un environnement complètement vide.

H : est la probabilité d'occurrence d'un événement. Elle repose sur l'expérience de la cellule robotique ou sur celle de cellules similaires. La valeur de 1 est utilisée pour une cellule dans laquelle aucun accident n'est à déplorer durant son utilisation, tandis qu'une occurrence élevée d'incidents conduit à utiliser la valeur de 10.

La formule permettant d'obtenir la grandeur P est alors :

$$P = E * H * (1 + V_{robot}) \quad (5\text{-}6)$$

Dans notre application, nous choisissons $E = 2$. En effet, hormis le plan de travail commun, l'humain a tout l'espace nécessaire pour se dégager. Ne disposant d'aucun historique d'utilisation, nous prendrons $H = 5$.

5.3.2.4 Points de mesure du risque

Les critères d'évaluation K, S et P sont évalués pour un point du robot. Pour mesurer le risque sur l'ensemble du robot, nous les calculons en trois points correspondant au coude du robot, au poignet et à l'effecteur. Ces trois points, situés aux extrémités des segments du robot sont ceux où la vitesse est la plus élevée (à l'origine des repères de la Figure 68). La variable D_{corpsR} est alors déclinée en $D_{corpsR,épaule} = 0.15m$, qui englobe la pointe de l'épaule et $D_{corpsR,poignet} = D_{corpsR,effecteur} = 0.08m$.

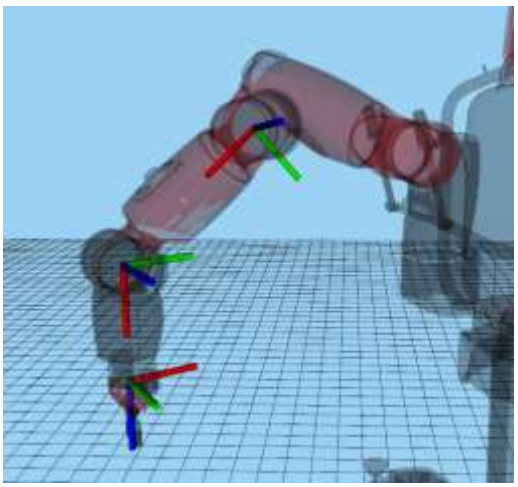


Figure 68 : repères associés aux points de contrôle des citères d'évaluation du risque : un pour l'épaule, un pour le poignet et un pour l'effecteur

5.3.3 Résultats

En premier, nous présentons les temps de calcul nécessaire à la réalisation des différentes étapes de notre solution, puis nous présenterons les résultats obtenus lors des simulations.

5.3.3.1 Temps de calcul

Les mesures suivantes ont été effectuées sur un ordinateur portable équipé d'un processeur à 4 cœurs (8 cœurs virtuels) Intel Xeon E3-1535M à 3.1 GHz, de 32Go de RAM et d'un disque dur SSD. Le système d'exploitation utilisé est Ubuntu 16.04 et l'ensemble des développements ont été réalisés en C++ dans le middleware ROS. Aucun des calculs n'est parallélisé ou déporté sur GPU.

Préalablement au lancement des tests, nous avons calculé la Look Up Table. La résolution de l'espace des configurations étant de 10°, compte tenu des limites des articulations du Baxter, la taille de l'espace des configurations est de $20 * 19 * 36 * 16$, soit 218 880 cellules. Il faut environ 3 heures et 10 minutes pour obtenir la table de projection pour un espace de cette taille. Néanmoins, celle-ci est calculée une seule fois et sauvegardée. Le temps nécessaire pour la charger en mémoire au lancement du programme de projection n'est que de deux minutes. La taille de la Look Up Table est d'environ 350 Mo dans la mémoire vive.

La première étape de notre chaine de traitement, est la mise à jour et la labélisation des 400 221 cellules de la carte d'occupation, qui prend 50ms. Ensuite, la projection des obstacles et de la sémantique associée vers l'espace des configurations dure 10ms. Le calcul de la carte des distances fusionnée nécessite 50ms de plus. Enfin, la dernière étape, le calcul d'une trajectoire sur l'axe médian, prends entre 50 à 60ms dans les cas favorables, 80ms lorsque la recherche est plus longue, et jusqu'à 140ms lorsqu'il n'y a pas de solution. En comparaison, il faut environs 450ms pour obtenir le squelette complet de l'espace des configurations, son calcul au fil de l'eau durant le A* apporte donc un vrai gain de temps.

Au total, notre chaine de traitement prend donc 170ms dans les cas favorables. Nous obtenons tout de même une fréquence en sortie du planificateur supérieure à 8 Hz du fait de la superposition de la fin des traitements d'une itération avec le début de la suivante grâce au multi cœur. Pour réduire le délai entre l'acquisition et l'envoi de la trajectoire au robot, nous pourrions paralléliser la plupart des calculs sur les 8 cœurs (tous sauf le A*) et ainsi réduire les délais de traitement à $\left(\frac{110}{8}\right) + 60 \approx 75ms$.

5.3.3.2 Influence des paramètres sur la trajectoire

Afin d'analyser l'influence des paramètres δ et k_h sur les chemins planifiés, nous avons réalisé des simulations avec 5 jeux de paramètres :

- $\delta = 2$ et $k_h = 1$: dans ce cas, l'influence des obstacles humains est nulle dans la fusion de la carte des distances.
- $\delta = 2$ et $k_h = 1.5$: les obstacles humains commencent à déformer la carte des distances.
- $\delta = 2$ et $k_h = 3$: cette influence est renforcée.
- $\delta = 2.5$ et $k_h = 1{,}5$: l'augmentation de δ réduit le nombre de points appartenant à l'axe médian et s'éloigne des obstacles.
- $\delta = 2.5$ et $k_h = 3$: l'axe médian et encore plus éloigné des obstacles humains.

La Figure 70 montre le squelette calculé dans les différents cas.

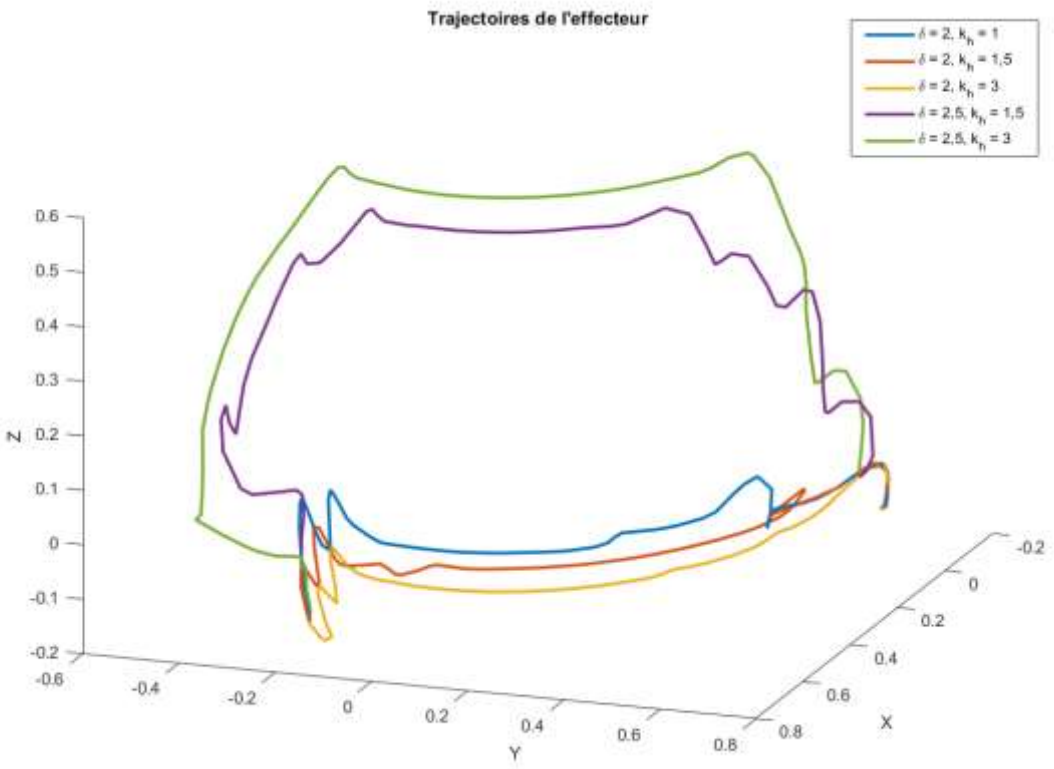
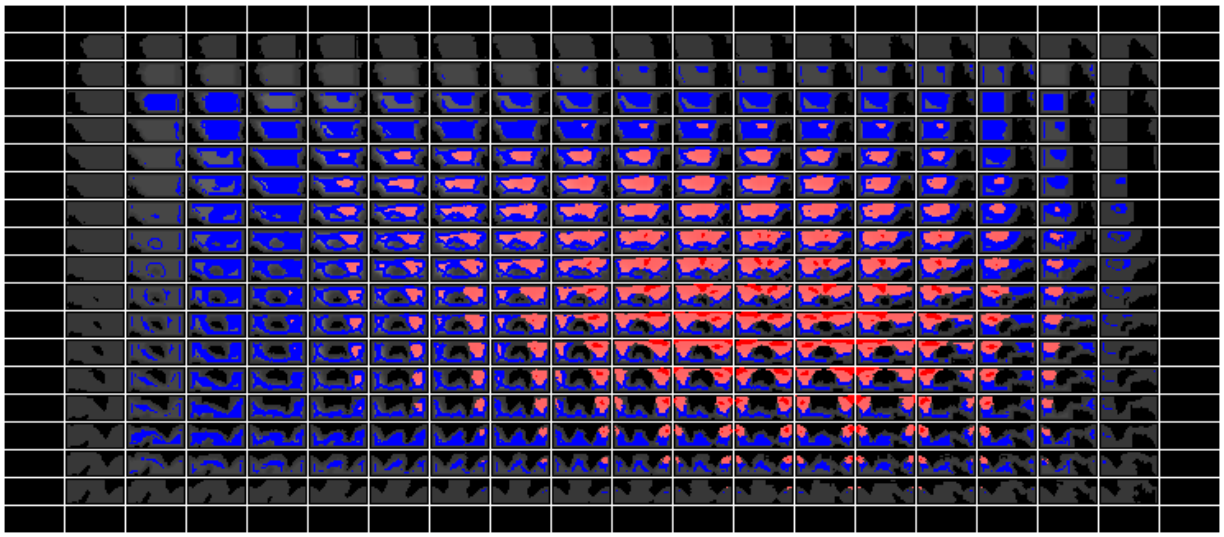


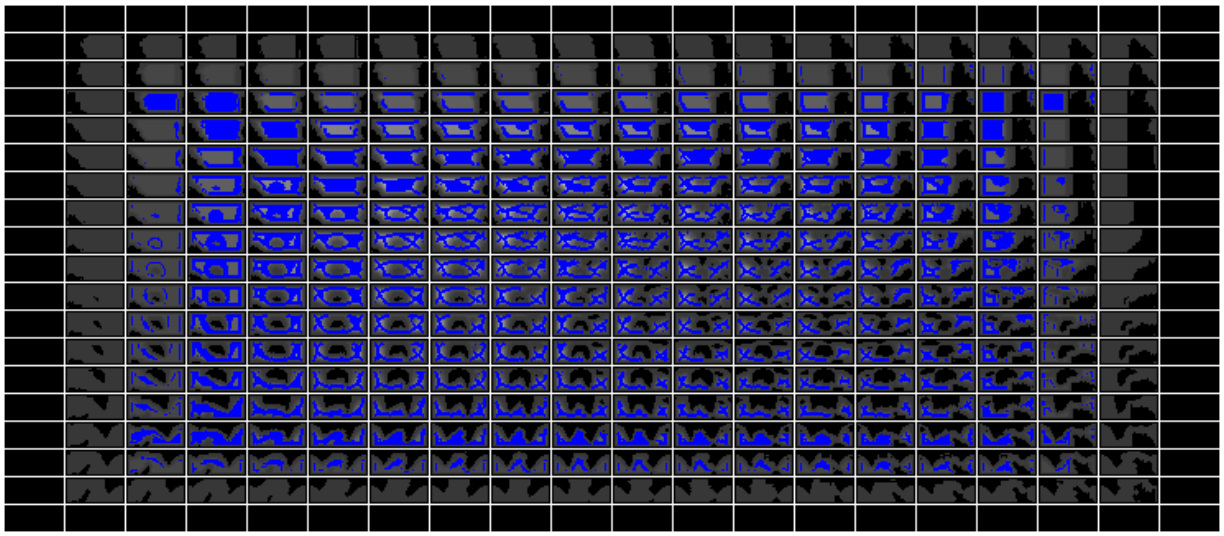
Figure 69 : Trajectoires de l'effecteur du Baxter.

La Figure 69 montre le chemin parcouru par l'effecteur du robot dans les 5 paramétrages. Pour aider le lecteur à se représenter l'évolution du robot face à l'humain, nous indiquons que :

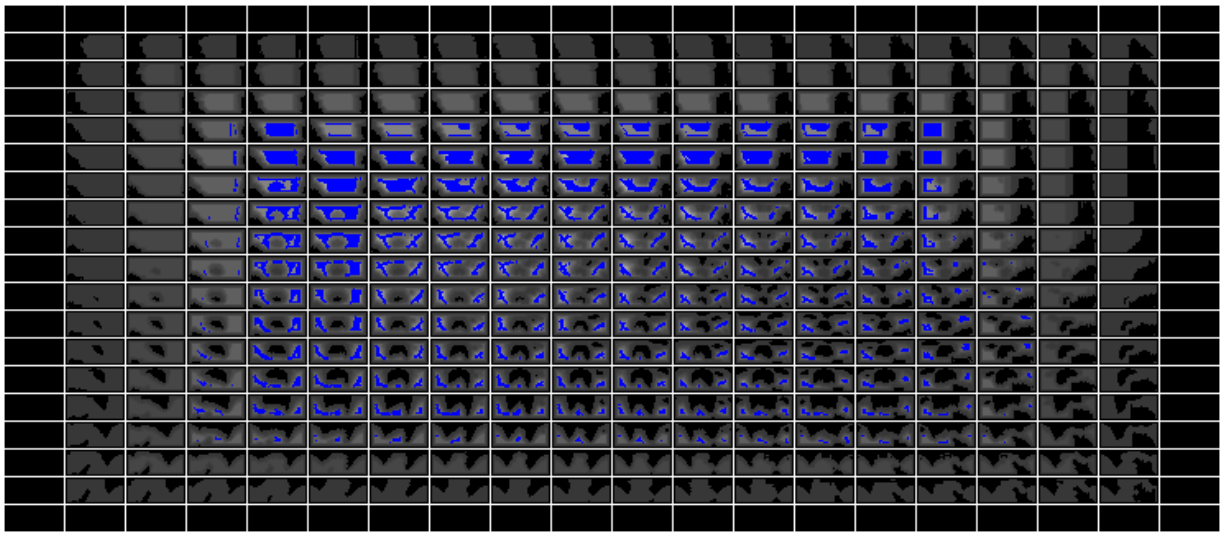
- L'opérateur se trouve de notre côté de la feuille (dans les X positifs)
- La base du bras se trouve à l'origine du repère (de l'autre côté de la feuille)
- Pour les trajectoires « basses » ($Z < 0.2$), l'effecteur du robot passe entre le corps du Baxter et les mains de l'opérateur, pour les autres il passe au-dessus.



a) $\delta = 2$ et $k_h = 3$



b) $\delta = 2$ et $k_h = 1$



c) $\delta = 3$ et $k_h = 1$

Figure 70 : Influence de δ et k_h sur le squelette (en bleu). a) $\delta = 2$ et $k_h = 3$. b) $\delta = 2$ et $k_h = 1$. c) $\delta = 3$ et $k_h = 1$.

Nous observons, Figure 69, une nette différences des trajectoires avec des valeurs de δ différentes. En effet, l'augmentation du paramètre de filtrage a supprimé certaines branches du squelette, et a conduit le manipulateur à emprunter un chemin passant au-dessus des bras de l'humain tandis que dans les autres simulations, celui-ci passe entre le tronc du Baxter et les mains de l'humain. Ce changement se traduit par une réduction de la probabilité d'apparition d'un évènement dangereux K comme illustrée par la Figure 71. Elle montre la valeur maximale entre les trois grandeurs K calculées (un pour chaque point de contrôle du robot) durant les 5 trajectoires. L'influence du paramètre k_h sur K est en revanche plus faible. Dans le cas de $\delta = 2.5$, nous observons bien une diminution de la probabilité mais nous ne pouvons noter de réel apports dans l'autre cas.

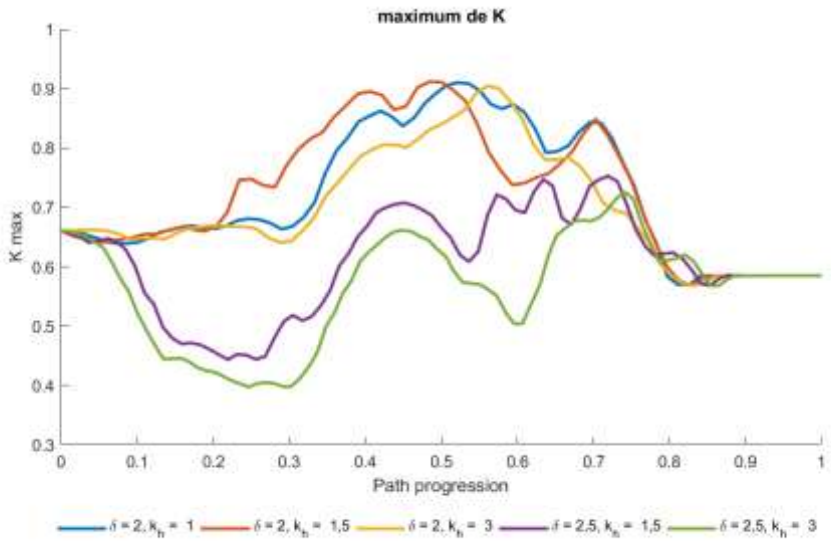


Figure 71 : Maximum des probabilités d'évènements dangereux, K, le long des trajectoires du robot.

Comme le facteur déterminant dans le calcul de l'indicateur K est la distance entre l'humain et les points de contrôle du robot, la Figure 72 montre son évolution pour chacun des points de contrôle. Pour l'effecteur, la distance évolue de façon semblable dans les 5 trajectoires et nous observons une légère augmentation de la distance avec l'augmentation de δ. En revanche, pour le poignet et le coude, le changement de chemin implique une forte augmentation de la distance. Nous notons en particulier que si c'est la distance du poignet qui est la plus faible tout au long des trajectoires pour $\delta = 2$, ce n'est pas le cas pour $\delta = 2.5$. L'influence de l'augmentation de k_h est toujours nulle pour les trajectoires basses tandis qu'elle est faible mais distincte lorsque le robot passe au-dessus des bras de l'humain.

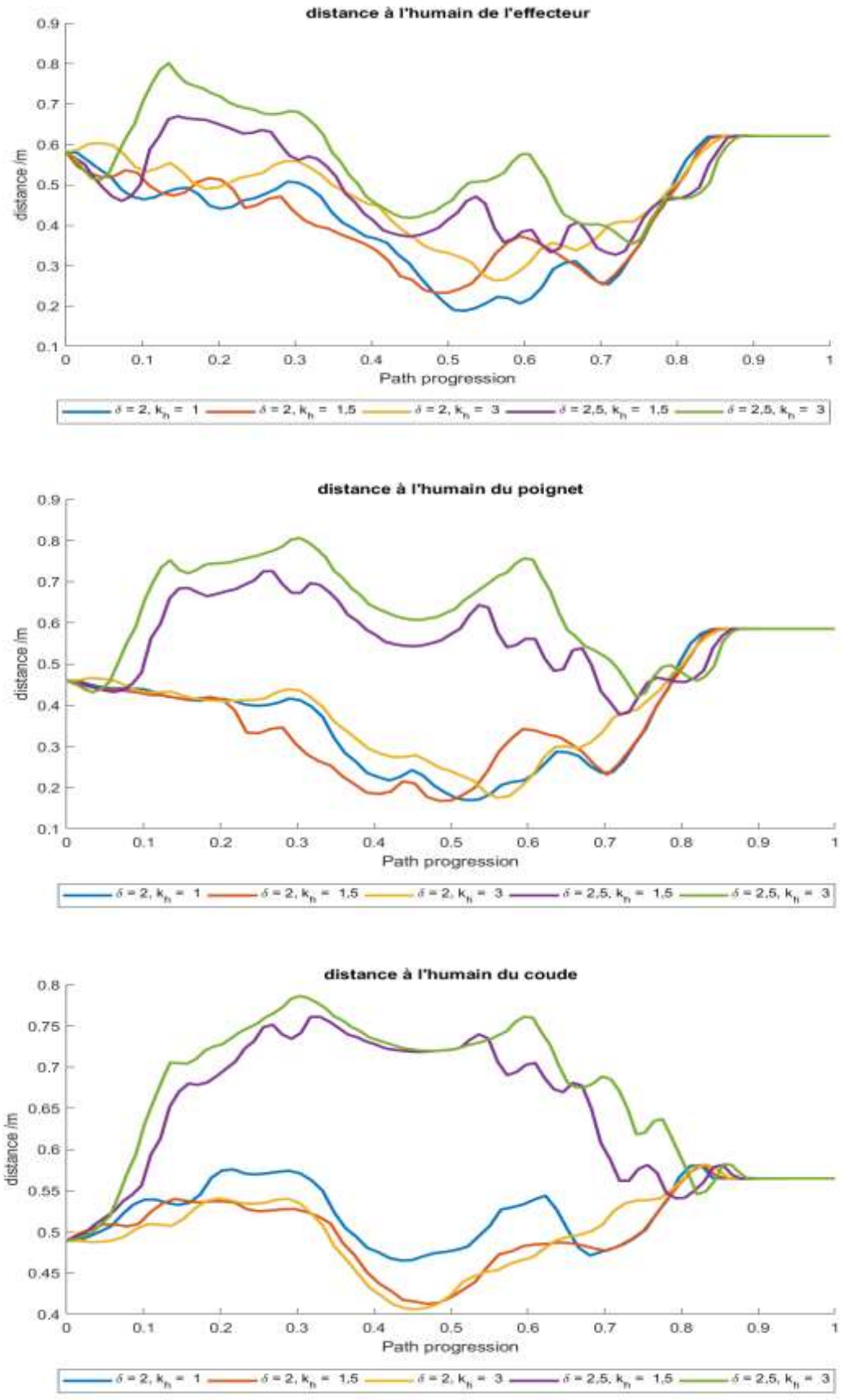


Figure 72 : Evolution de la distance entre chaque point de contrôle et la cellule labélisée humain la plus proche.

La faible influence de k_h sur les résultats précédents nous a amenés à penser que l'humain était probablement trop éloigné du robot pour que ce paramètre influe véritablement sur les trajectoires. Nous avons donc rapproché, dans la simulation, l'humain de la table et du robot de 10cm et réitéré l'ensemble des mesures. La Figure 73 montre les trajectoires pour les 5 ensembles de paramètres dans cette nouvelle situation. Elle montre une différence notable entre les trajectoires faites avec un $k_h = 3$ et les autres.

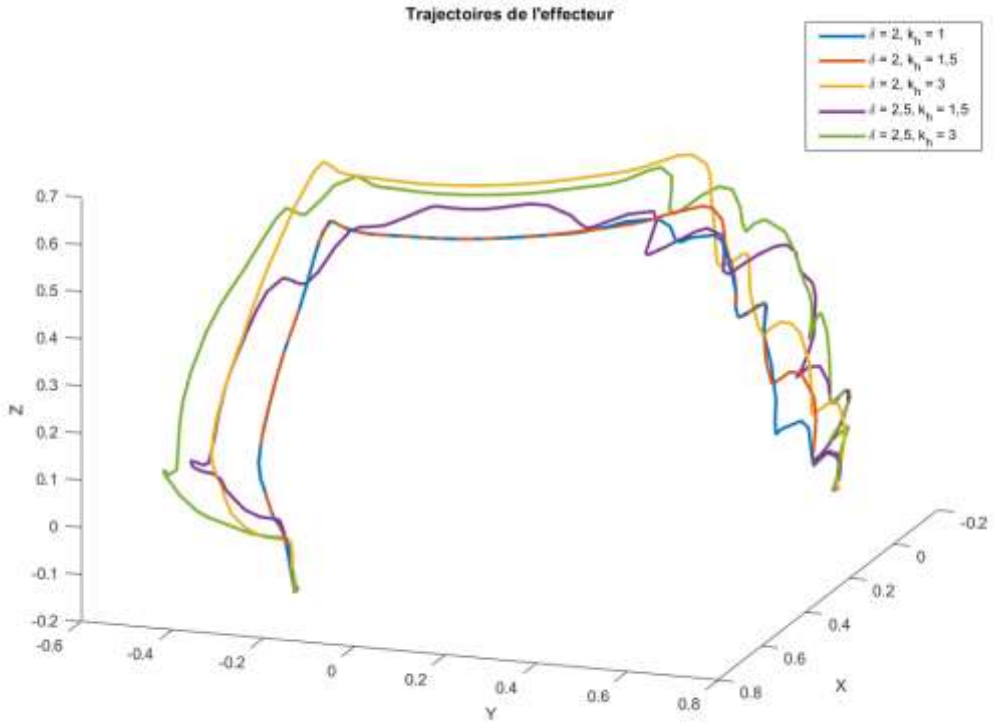


Figure 73 : Trajectoires de l'effecteur du robot lorsque l'humain est plus proche du robot.

La Figure 74 montre l'évolution du maximum des K des trois points de contrôle. Nous pouvons constater que lorsque l'humain est proche du robot, l'augmentation de k_h devient un paramètre déterminant pour éloigner le robot de l'opérateur. Néanmoins, nous pouvons aussi constater que la différence entre les trajectoires pour $k_h = 1$ et $k_h = 1.5$ est quasi inexistante, la zone d'influence de la distance à l'humain dans l'espace des configuration n'est sans doute pas traversée par la trajectoire dans ce cas (au contraire du cas où $k_h = 3$). Nous remarquons aussi que l'influence de δ est plus faible mais permet tout de même d'éloigner un peu plus le robot de l'opérateur dans certaines zones de la trajectoire.

Nous pouvons donc conclure que la modification du paramètre δ permet en toutes circonstances d'augmenter la distance entre le robot et les obstacles. Il faut tout de même faire attention à ne pas l'augmenter excessivement au risque de trop éroder l'axe médian et de le couper en plusieurs morceaux, empêchant ainsi la découverte d'un chemin. Le

paramètre k_h permet quant à lui d'empêcher le robot d'être trop proche de l'opérateur mais son influence se limite à la proximité immédiate de l'humain. En résumé, augmenter δ permet de choisir globalement des chemins plus éloignés tandis que k_h éloigne ces chemins de la proximité immédiate de l'opérateur.

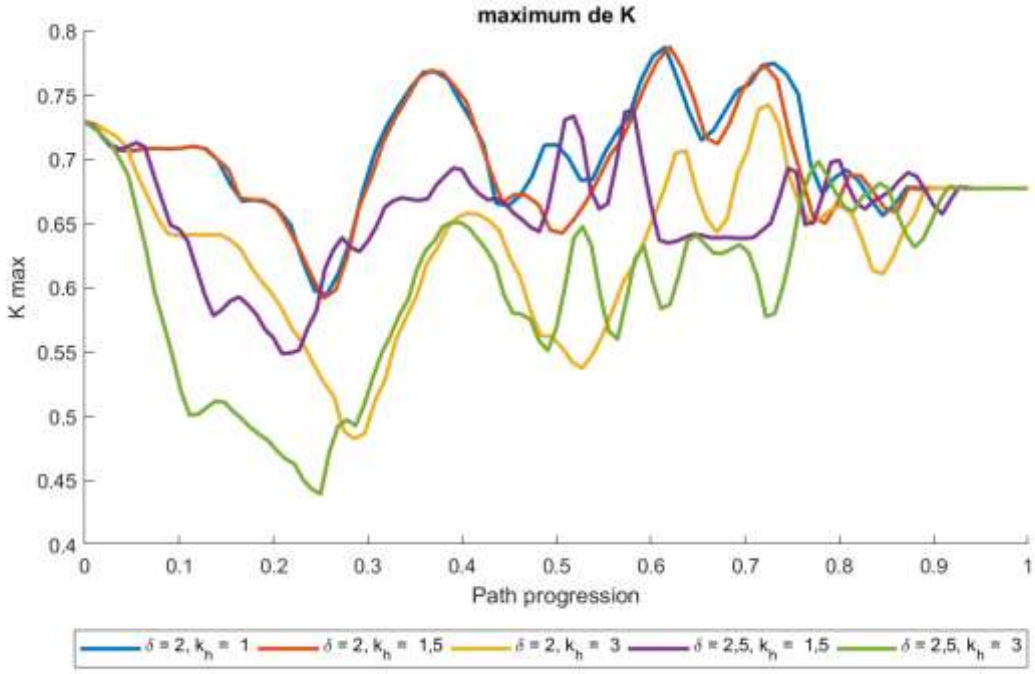


Figure 74 : Evolution du maximum des probabilités K pour chacun des ensembles de paramètres lorsque l'humain est plus proche du robot.

Les courbes d'influence sur les indicateurs de sévérité en cas d'incident, S, et de probabilité d'évitement P sont représentées Figure 75 et Figure 76. Nous constatons leur varaition est faiblement corrélée aux parmaètres δ et k_h. Leur analyse est rendue difficilecar celles-ci sont à la fois proche les unes des autres et bruitées. Nous avons alors calculé l'intégrale de chacune des courbes, en fonction du temps normalisé, afin de les comparées. Le Tableau 1 présentes les résultats. Notons que les valeurs dans le tableau ne sont significatives qu'en comparaison avec les autres situations.

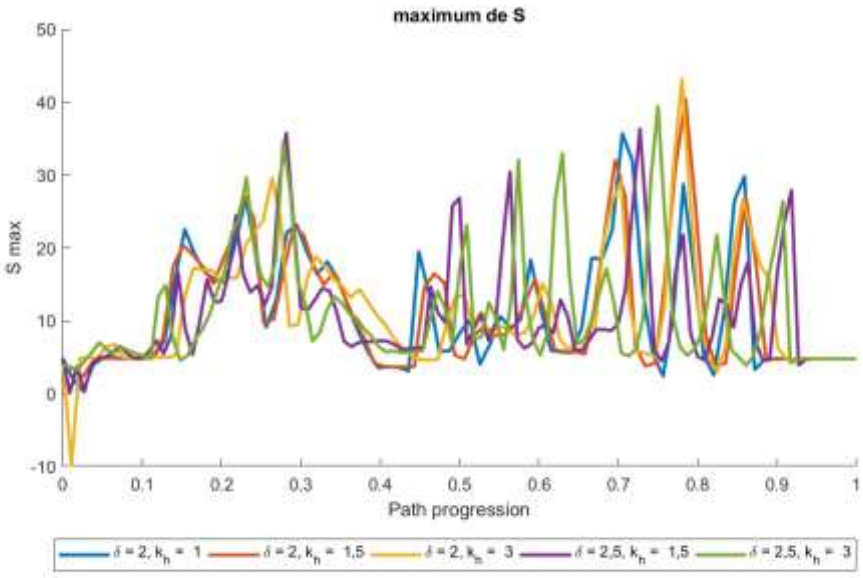


Figure 75 : Evolution du maximum des critères S pour chacune des trajectoires lorsque l'humain est proche.

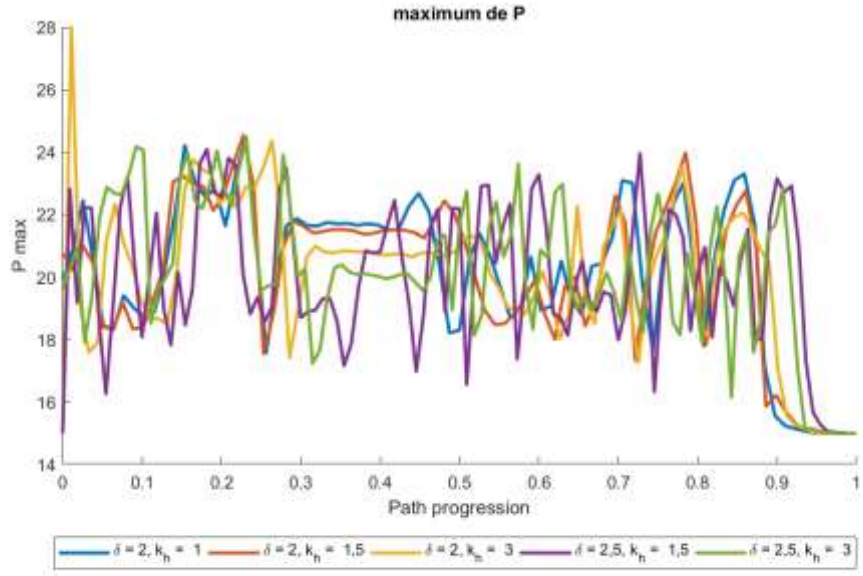


Figure 76 : Evolution du maximum des critères P pour chacune des trajectoires lorsque l'humain est proche.

	S, sévérité d'un évènement		P, probabilité d'évitement	
	Humain loin	Humain proche	Humain loin	Humain proche
$\delta = 2$ et $k_h = 1$	9.30	11.34	19.87	20.22
$\delta = 2$ et $k_h = 1.5$	8.32	11.63	19.92	20.14
$\delta = 2$ et $k_h = 3$	8.18	11.49	19.33	20.16
$\delta = 2.5$ et $k_h = 1.5$	12.51	10.88	20.58	20.02
$\delta = 2.5$ et $k_h = 3$	11.66	11.37	20.40	20.26

Tableau 1 : Intégrales du maximum de S et P pour l'ensemble des trajectoires.

Pour la probabilité d'évitement, nous observons que les différences entre les trajectoires sont faibles. On observe tout de même les trois chemins passant entre les mains de l'opérateur et le robot présentent une valeur de P plus faible que les autres. Les trajectoires basses étant plus proches des obstacles, la modulation de vitesse les rend plus lentes, et comme P dépend de la vitesse de déplacement du robot, il est normal que P soit plus élevé dans ces cas-là.

S, la sévérité de l'évènement est calculée en prenant en compte la vitesse du robot mais aussi le facteur de membre qui évolue en fonction de la hauteur du point de contrôle. Les trajectoires hautes rapprochant le poignet et l'effecteur de la tête de l'opérateur, il est logique d'observer une augmentation de la sévérité pour ces trajectoires-là.

5.3.4 Conclusions

Cette seconde partie des expérimentations a, en premier lieu, permis de mettre en évidence que le temps de calcul nécessaire à nos algorithmes est compatible avec une planification réactive, et en phase avec notre objectif initial. Une suite logique sera donc de mener les expérimentations en dynamique sur un robot réel. Des expérimentations avec le Baxter sont en cours de finalisation, après une phase d'optimisation de l'implémentation. L'évaluation des trajectoires selon des critères spécifiques à l'évaluation de la sécurité de la collaboration Homme robot a mis en évidence l'apport de la considération de l'humain dans la phase de planification. Les deux principaux paramètres de notre pipeline permettent, pour le premier, d'éloigner le robot des obstacles et, pour le second, de l'empêcher de s'approcher des obstacles humains en particulier. Ce comportement permet de diminuer fortement la probabilité d'apparition d'un évènement dangereux. Il n'y a pas d'influence notable sur la possibilité d'évitement mais le choix de certaines trajectoires plus loin des obstacles augmente la sévérité potentielle d'un incident. Mais en regard de la réduction du risque d'apparition de cet évènement, le risque global pour l'opérateur, à défaut d'être inférieur, n'est pas plus élevé.

Chapitre 6.
Conclusion

6.1 Synthèse

De nos jours, les interactions entre les robots et les humains sont limitées par les contraintes de sécurités et le manque d'autonomie décisionnelle des systèmes robotisés. Pour aller plus loin dans l'intégration de la collaboration au sein des chaines de production, nous nous sommes intéressés à deux problématiques majeures de la robotique :

- La perception et la représentation de l'environnement : pour permettre une autonomie des systèmes robotiques, la connaissance de l'espace de travail est une condition préalable.
- La planification de trajectoires : pour permettre au robot d'agir sur son environnement et d'accomplir sa mission dans un espace partagé avec l'humain tout en garantissant sa sécurité.

Pour répondre à ces deux questions, nous avons proposé et mis en place une chaine complète allant des capteurs à la génération automatique de trajectoires pour le robot dans des environnements dynamiques.

Nous exposons dans le chapitre trois la démarche nécessaire à la construction de la représentation de l'espace, en commençant par le choix des capteurs, notamment la caméra 3D pour la richesse de ses informations ou le lidar pour sa simplicité d'utilisation et sa grande fiabilité. Nos méthodes de construction, labélisation et mise à jour de la carte d'occupation permettent d'obtenir une information fiable et complète pour l'interaction Homme robot. Nous avons ensuite introduit une technique de construction en ligne de l'espace des configurations. Cette méthode permet aussi la projection des informations sémantiques, essentielles pour assurer la sécurité de l'humain.

A partir de l'espace des configurations ainsi obtenu, nous avons présenté dans le chapitre quatre une nouvelle approche basée sur l'axe médian de l'espace libre des configurations pour la planification de trajectoires. Dans l'objectif de calculer l'axe médian, qui par définition se situe à égale distance des obstacles l'entourant, nous avons apporté une amélioration certaine au calcul de la carte des distances dans les espaces à N dimensions. Afin d'améliorer la sécurité des opérateurs, nous avons alors proposé de déformer l'axe médian grâce aux information sémantiques, en pondérant les cartes de distances et ainsi l'éloigner encore plus des humains. Nous avons alors calculer le chemin du robot à l'aide d'un algorithme A* qui se propagent uniquement sur cet axe transformé. Nous avons aussi mis en place une modulation de la vitesse du robot de façon à réguler l'énergie cinétique emmagasinée par ce dernier en fonction de la distance qui le sépare de l'Homme. Enfin, pour limiter les saccades dûes à la forte dynamicité de l'environnement, une fusion de trajectoires permet une transition douce et évite les à-coups évitant un probable inconfort des opérateurs.

Les expérimentations réalisées et présentées dans le chapitre cinq démontrent les capacités de notre chaine de traitement à déplacer un robot en autonomie et sécurité pour l'humain dans un contexte collaboratif. Elles mettent en avant le fait que nos algorithmes permettent d'éloigner le robot de l'humain et que d'après des critères d'évaluation de risque cela réduit la probabilité de survenue d'un incident. Dans un contexte dynamique, la fusion de trajectoires permet un mouvement continue du robot et les mesures réalisées lors des tests permettent d'attester que notre système est utilisable pour la planification dynamique d'un robot à 7 axes.

6.2 Contributions majeures

Au cours de nos travaux, exposés ici, nous avons apporté trois contributions principales :

- Tout d'abord, nous avons conçu une représentation de l'espace adaptée à la collaboration Homme robot. Il s'agit d'une grille d'occupation sémantique permettant de différencier les obstacles selon leur nature et notamment les humains. Elle est construite et mise à jour par un ensemble de capteurs hétérogènes pour permettre en plus de la multiplicité des points de vue, la multiplicité des types de données facilitant la segmentation sémantique. La mise en évidence des humains dans l'espace de travail du robot est une étape préalable et nécessaire à tous les aspects de la collaboration, que ce soit la prise de décision haut niveau (quelles tâches doit accomplir le robot), la planification de trajectoires (comment déplacer le robot pour accomplir sa tâche) ou l'évaluation la sécurité de l'interaction (la sécurité de l'opérateur est-elle menacée par l'action du robot).

- Ensuite, nous avons proposé et implémenté une méthode de construction de l'espace des configurations en temps réel. Elle est basée sur le calcul en amont d'une Look Up Table, permettant d'associer à chaque point de l'espace de travail, l'ensemble des configurations pour lesquelles le robot est en collision avec lui. Ce calcul préalable est uniquement dépendant de la géométrie du robot et n'est donc à réaliser qu'une fois et sauvegardé pour un usage ultérieur. La projection rapide des obstacles dans l'espace des configurations permet aussi d'y apporter l'information sémantique en projetant séparément les différents types d'obstacles. La constitution d'un espace des configurations sémantique permet par la suite d'ajuster la planification en fonction de la présence des humains.
- Enfin, nous avons mis au point un planificateur de trajectoires dynamique sur un axe médian modifié dans l'espace des configurations. La construction de l'axe médian sur une carte des distances déformée par la présence des humains permet de l'éloigner de ces derniers. La trajectoire basée sur celui-ci est alors libre de collisions et maintenue à distance des opérateurs. Pour pallier au manque de stabilité du squelette, une fusion dynamique de la trajectoire courante avec la nouvellement proposée permet une transition douce, évitant les saccades et la sensation de gêne qui pourrait en résulter chez les humains. Pour finir, la modulation de la vitesse du robot en fonction de la distance à l'humain permet de limiter l'énergie d'un choc potentiel. Cet ensemble résulte en un planificateur de trajectoires sûres pour l'interaction Homme robot.

6.3 Perspectives et applications.

Pour faire suite aux travaux exposés ici et aller plus loin dans la planification de trajectoires sûres, les pistes suivantes pourraient-être explorées :

- ✓ Pour réduire le délai entre la perception de l'environnement et l'envoi de la trajectoire au robot, certains aspects de l'implémentation pourrait être améliorés. En premier, la parallélisation des calculs sur plusieurs cœurs permettrait de réduire le temps nécessaire à la mise à jour de la carte, à la projection et au calcul des cartes de distances. Ensuite, nos algorithmes manipulent des objets de grande taille en mémoire vive (look up table, carte d'occupation, espaces des configurations), les opérations d'accès pour le parcours, la lecture et l'écriture sont coûteuses et l'utilisation d'un objet informatique dédié et optimisé permettrait un gain de temps significatif.
- ✓ Les trajectoires créées par notre planificateur sont aujourd'hui composées des points de passage pris sur le squelette et envoyées brutes au contrôleur du robot. L'ajout d'une étape de lissage avant cette transmission permettrai d'apporter plus

de précision dans le contrôle de vitesse du robot et plus de régularité dans les trajectoires. Par exemple, l'usage de courbes B-spline permettrait de définir une trajectoire continue et dérivable pour contrôler les vitesses et les accélérations tout en satisfaisant les contraintes dynamiques du robot. De plus, compte tenu de la largeur des cellules de l'espace des configurations en 4D (10°), l'optimisation du chemin par les courbes permettrait de rester très proche de l'axe médian et ainsi de ne pas détériorer les contraintes de sécurité.

- ✓ L'utilisation de la métrique euclidienne dans l'espace des configurations est un choix pratique et donne des résultats satisfaisant mais ne reflète pas forcement la réalité de l'espace cartésien. Notamment, nous pouvons observer que deux mouvements de même amplitude dans l'espace des configurations peuvent aboutir à des mouvements très différents dans l'espace de travail. La recherche d'une métrique adaptée pourrait permettre de corriger ce défaut. Cependant, pour maintenir la rapidité de calcul des cartes des distances, la décomposition du calcul des distances dans chacune des dimensions doit être possible.
- ✓ L'ajout d'une modélisation des mouvements et des comportements de l'humain pourrait permettre d'anticiper ses positions futures et ainsi d'ajouter ces estimations à la carte d'occupation. Elles pourraient alors être projetées vers l'espace des configurations et influencer la planification de chemin. Cette anticipation augmenterait la réactivité de notre système accroissant encore la sécurité de l'interaction.

Les résultats encourageants présentés ici nous permettent d'envisager de nombreuses applications à nos contributions :

- ✓ La première application est la raison même qui nous a conduit à mener ces travaux : la collaboration d'un bras robotique avec des opérateurs dans un contexte industriel. La planification dynamique permet de maintenir un niveau de production correcte tout en garantissant la sécurité de l'opérateur. En comparaison avec les modes de fonctionnement où le robot est contraint de s'arrêter, l'interaction est plus fluide et plus efficiente.
- ✓ La seconde application qui pourrait utiliser notre travail est la génération de trajectoires pour les bras d'un robot de service ou celui d'une base industrielle mobile. En effet, les capacités de construction temps réel de la carte d'occupation et l'espace des configurations permettent son utilisation dans des environnements de dynamique élevée.
- ✓ La construction de l'espace des configurations en ligne peut servir de base à de nombreuses applications de planification pour les bras robotiques comme par exemple de la saisie d'objet.

- ✓ La carte d'occupation contenant les informations sémantiques nécessaires à la collaboration Homme robot peut elle aussi s'avérer utile en dehors de la planification de trajectoires, notamment et comme nous l'avons fait pour estimer les risques d'une situation de collaboration.
- ✓ L'axe médian déformé par les données sémantiques pourrait trouver des applications dans la navigation de robot mobile ou la voiture autonome (modification de la trajectoire de base, en fonction des caractéristiques des objets en présence ou d'humains).

Annexes

Annexe 1 : Le processus de production d'une garniture d'embrayage

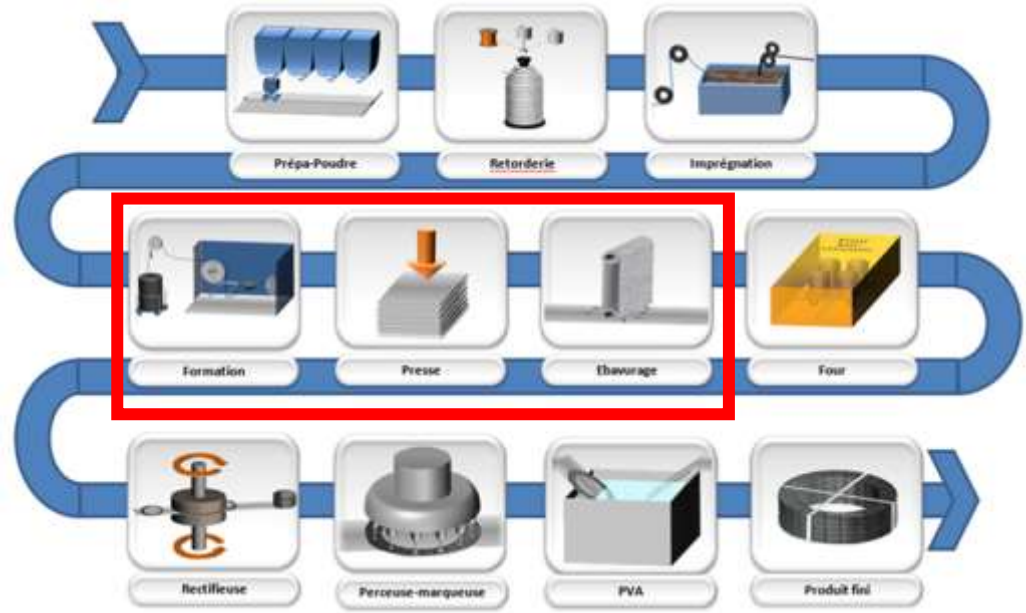


Figure 77 : processus de production d'une garniture d'embrayage

La figure ci-dessus décrit le processus de fabrication d'une garniture d'embrayage. Une partie de ce processus est l'objet de la tâche de robotisation collaborative (encadré en rouge). Il se décompose en plusieurs étapes :

- La préparation des poudres : lors de cette première étape un mélange de poudres est créé à partir de différents minéraux. Les matériaux et les proportions sont choisis en fonction des caractéristiques voulues pour la garniture.
- La retorderie : cette seconde étape consiste à assembler un fil composé de plusieurs brins torsadés. Ces brins peuvent être de différentes sortes : naturelle, synthétique (nylon) ou métallique (cuivre). La composition du fil est elle aussi donnée par les propriétés voulue pour la garniture.
- L'imprégnation : dans cette étape les produits des deux premières vont être assemblés pour former le matériau de la garniture. Pour cela, les fils sont imprégnés du mélange de poudre préalablement mis en phase aqueuse. Le fil est ensuite séché et mis en bobine.
- La formation : le fil imprégné est mis en forme de disque à l'aide d'une machine nommée formeuse. Celle-ci va amener entre deux plateaux pressés l'un contre l'autre le fil en décrivant une épitrochoïde. Cela permet d'obtenir un disque d'épaisseur régulière qui est nommé préforme.

- La presse : cette préforme est introduite dans un moule qui entre dans une presse pour suivre un cycle de thermocompression. Cette transformation va permettre de lier les épaisseurs de fil entre elles et de donner une forme quasi définitive à la garniture (diamètres intérieur et extérieur et forme des rainures), on appelle cette couronne l'ébauche.
- L'ébavurage : suite au moulage dans les presses, les ébauches présentent des bavures dûes à l'écoulement du surplus de matière. Celles-ci sont éliminées à l'aide de bandes abrasives.
- La cuisson : les ébauches sont ensuite conduites dans un four pour subir une étape de cuisson supplémentaire. Celle-ci permet de parfaire les liaisons créées lors du pressage et donne à la garniture ses caractéristiques physiques finales.
- La rectification : les ébauches sont alors amenées dans une rectifieuse pour leur donner leur épaisseur finale. C'est suite à cette étape que nous les nommons garnitures.
- Perçage-marquage : les garnitures sont percées pour permettre leur montage sur les mécanismes d'embrayage puis marquées afin de les identifier (marque, modèle, numéro de production).
- PVA : les pièces sont recouvertes d'un produit anti poussière pour préserver leur intégrité jusqu'à leur mise en service dans le véhicule fini.
- Conditionnement : enfin, les garnitures sont conditionnées et expédiée chez les embrayeurs afin d'être assemblés sur les mécanismes d'embrayage.

Annexe 2 : Modélisation de Denavit-Hartenberg

Cette convention permet de caractériser la transformation entre deux repères par seulement 4 paramètres mais il faut respecter les règles de placement des repères et des axes :

- ✓ L' axe $\overrightarrow{z_n}$ est porté par l'axe de liaison reliant le corps C_n et le corps C_{n+1}.
- ✓ L'axe $\overrightarrow{x_n}$ est porté par la normale commune à $\overrightarrow{z_{n-1}}$ et $\overrightarrow{z_n}$.
- ✓ L'axe $\overrightarrow{y_n}$ est choisi pour former un trièdre direct avec $\overrightarrow{z_n}$ et $\overrightarrow{x_n}$.

La transformation entre le repère R_{n-1} et le repère R_n est alors définie par :

- ➢ d_n la distance entre $\overrightarrow{x_{n-1}}$ et $\overrightarrow{x_n}$ selon $\overrightarrow{z_{n-1}}$.
- ➢ θ_n l'angle entre $\overrightarrow{x_{n-1}}$ et $\overrightarrow{x_n}$ autour de $\overrightarrow{z_{n-1}}$.
- ➢ r_n la distance entre $\overrightarrow{z_{n-1}}$ et $\overrightarrow{z_n}$ selon $\overrightarrow{x_n}$.
- ➢ α_n l'angle entre $\overrightarrow{z_{n-1}}$ et $\overrightarrow{z_n}$ autour de $\overrightarrow{x_n}$.

La matrice de transformation homogène associée au passage du repère R_{n-1} au repère R_n est obtenue en multipliant les matrices des rotation et translation unitaire de chacune des transformations précédentes :

$$TH_{n-1,n} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos(\theta_n) & -\sin(\theta_n) & 0 & 0 \\ \sin(\theta_n) & \cos(\theta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & r_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_n) & -\sin(\alpha_n) & 0 \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$TH_{n-1,n} = \begin{pmatrix} \cos(\theta_n) & -\cos(\alpha_n)\sin(\theta_n) & \sin(\alpha_n)\sin(\theta_n) & r_n\cos(\theta_n) \\ \sin(\theta_n) & \cos(\alpha_n)\cos(\theta_n) & -\sin(\alpha_n)\cos(\theta_n) & r_n\sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La transformation inverse est :

$$TH_{n-1,n}^{-1} = TH_{n,n-1} \begin{pmatrix} \cos(\theta_n) & \sin(\theta_n) & 0 & -r_n \\ -\cos(\alpha_n)\sin(\theta_n) & \cos(\alpha_n)\cos(\theta_n) & \sin(\alpha_n) & -d_n\sin(\alpha_n) \\ \sin(\alpha_n)\sin(\theta_n) & -\sin(\alpha_n)\cos(\theta_n) & \cos(\alpha_n) & -d_n\cos(\alpha_n) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Voici deux exemples de paramètrisation selon Denavit Hartenberg, le premier pour le robot SCARA d'ABB et le second pour un bras du robot Baxter :

liaison	d_n	θ_n	r_n	α_n
0->1	0	q_1	0,3 m	0
1->2	0	q_2	0,25 m	0
2->3	q_3	0	0	0
3->4	0	q_4	0	0

Tableau 2 : Paramètres de Denavit Hartenberg du robot ABB IRB910sc-3/0.55.

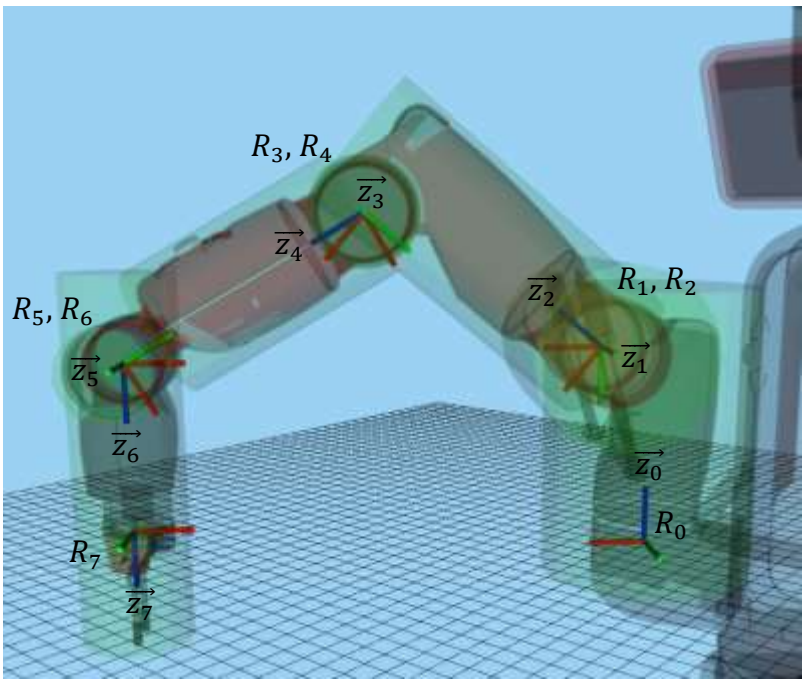


Figure 78 : Modèlisation de Denavit Hartenberg du bras droit de Baxter.

liaison	d_n	θ_n	r_n	α_n
0->1	0,27 m	q_1	0,07 m	$-90°$
1->2	0 m	q_2	0 m	$90°$
2->3	0,36 m	q_3	0,07 m	$-90°$
3->4	0 m	q_4	0 m	$-90°$
4->5	0,37 m	q_5	0,01 m	$90°$
5->6	0 m	q_6	0 m	$-90°$
6->7	0,23 m	q_7	0 m	$0°$

Tableau 3 : Paramètres de Denavit Hartenberg d'un bras du Baxter.

Annexe 3 : Résultats comparatifs de l'optimisation du calcul de la carte des distances.

Pour valider l'apport de notre optimisation relativement aux méthodes de l'état de l'art (notamment les travaux de Felzenswalb sur lesquels nous nous appuyons), nous avons mis en place un protocole expérimental complet permettant de les comparer dans des conditions équitables. Notons que notre optimisation n'est pas spécifique au contexte robotique, et que ces résultats ont donc été menés afin d'être les plus génériques possibles. Afin de comparer aux travaux existants, nous avons ainsi fait le choix de travailler en 2D, avec des banques d'images réputées du domaine. Les algorithmes considérés sont les plus performants de l'état de l'art. Ils incluent les méthodes proposées par (Hirata 1996; Maurer et al. 2003; Lucet 2005; Schouten et Broek 2014; Felzenszwalb et Huttenlocher 2012).

La première banque d'images considérée a été proposée dans (Fabbri et al. 2008), avec pour objectif de définir un moyen standardisé de comparer les algorithmes de transformée en distance. Elle contient principalement des images artificielles formées d'éléments géométriques simples (lignes, disques, carrés, pixels), et est donc très éloignée d'applications concrètes. Quelques exemples d'images de cette banque sont présentés Figure 79, tandis que les résultats comparatifs obtenus sont proposés dans le Tableau 4.

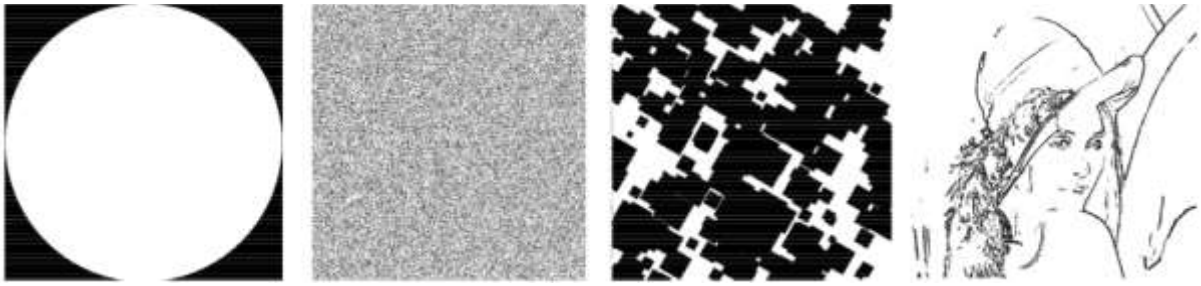


Figure 79 : Exemples d'images considérées dans la banque d'images définie par Fabbri (Fabbri et al. 2008). De gauche à droite : inscribed disk, random pixels (50%), random squares (75%), Lenna's edges (inverse). On note que le contenu de ces différentes images est fondamentalement différent, ce qui explique les fortes variations dans les résultats présentés dans le tableau ci-dessous.

images	Timing (in *ns/pixel*) per algorithm											
	Proposed		Felzenswalb		Hirata		Maurer		Lucet		Schouten	
	ave	rms	ave	rms	ave	rms	ave	rms	ave	rms	ave	rms
turning line	7.86	1.67	7.64	1.61	8.07	1.69	8.20	1.70	6.11	0.3	19.90	8.11
turning line, inverse	4.20	0.05	6.69	0.07	8.79	0.08	8.65	0.197	9.03	0.07	1.79	0.05
inscribed circle	8.99	0.26	9.23	0.36	9.90	0.14	10.70	0.31	11.98	4.81	44.88	21.68
inscribed circle, inverse	5.09	0.25	7.20	0.17	9.04	0.19	9.09	0.35	12.95	4.62	3.82	0.53
random pixels (see Fig.4)	18.92	6.50	19.21	5.19	18.60	4.15	19.92	5.11	23.10	3.50	18.52	8.73
random squares	6.69	1.51	7.97	0.81	9.29	0.32	9.33	0.40	17.59	0.80	7.10	3.83
point in corner	5.76	0.12	7.81	0.14	9.45	0.14	8.89	0.11	10.35	2.91	1.86	0.07
point in corner, inverse	4.00	0.15	6.43	0.17	8.71	0.20	8.58	0.16	12.18	2.95	1.74	0.05
Lenna's edges	16.34		15.42		14.30		15.95		17.05		13.68	
Lenna's edges, inverse	7.51		9.07		10.56		10.77		16.36		4.74	

Tableau 4 : Temps d'exécution obtenus pour la banque d'images définie dans [Fabbri08], pour l'ensemble des algorithmes considérés.

Globalement, les résultats obtenus sont en faveur du travail de (Schouten et Broek 2014). Pour information, il s'agit à l'heure actuelle de l'algorithme reconnu comme étant le plus rapide. Il est néanmoins non généralisable aux dimensions supérieures à 2 (il présente une complexité exponentielle au nombre de dimensions), ce qui le rend totalement inutilisable dans notre contexte.

Cela étant dit, comparé aux autres algorithmes de l'état de l'art (qui sont dans les faits beaucoup plus proches du notre, tant dans la théorie que dans la complexité algorithmique), la solution proposée est significativement plus performante, avec une amélioration du temps de calcul allant en moyenne de 13% par rapport à (Felzenszwalb et Huttenlocher 2012), jusqu'à 41% par rapport à la solution proposée par (Lucet 2005).

On remarque néanmoins 3 scénarios qui vont à l'encontre de cette affirmation, à savoir : random pixels, turning line, et Lenna's edges. Pour expliquer ce phénomène, considérons le cas du scénario random pixels. Dans celui-ci, un ensemble d'images est générée en choisissant un nombre de pixels comme étant le fond de l'image (c'est à dire les obstacles) et en les plaçant aléatoirement, ce nombre augmentant d'image en image. Comme illustré Figure 80, l'algorithme proposé figure parmi les plus lents pour des pourcentages de pixels de fond faibles. A partir de 70%, la tendance s'inverse, ce qui démontre clairement la surcharge computationnelle induite par la combinaison de segments et de paraboles : puisque les pixels sont placés aléatoirement, les obstacles ne sont que très peu connectés. Or, pour être en mesure de produire des segments, l'algorithme recherche des paraboles adjacentes qui n'existent pas, ce qui a un coût (environ 5% de plus que l'algorithme original de Felzenszwalb). A partir de 70%, le fond se retrouve de plus en plus connecté. Ce coût et alors largement contrebalancé par le gain induit lors de l'ajout des segments, rendant ainsi notre algorithme nettement plus efficace que les autres.

En suivant la même logique, notre méthode est également légèrement plus lente lorsqu'elle est appliquée sur les contours de Lenna (par définition, les contours sont fins,

donc ne forment que très rarement des segments) et sur le scénario turning line (qui consiste en une droite d'un pixel d'épaisseur qui coupe l'image en deux).

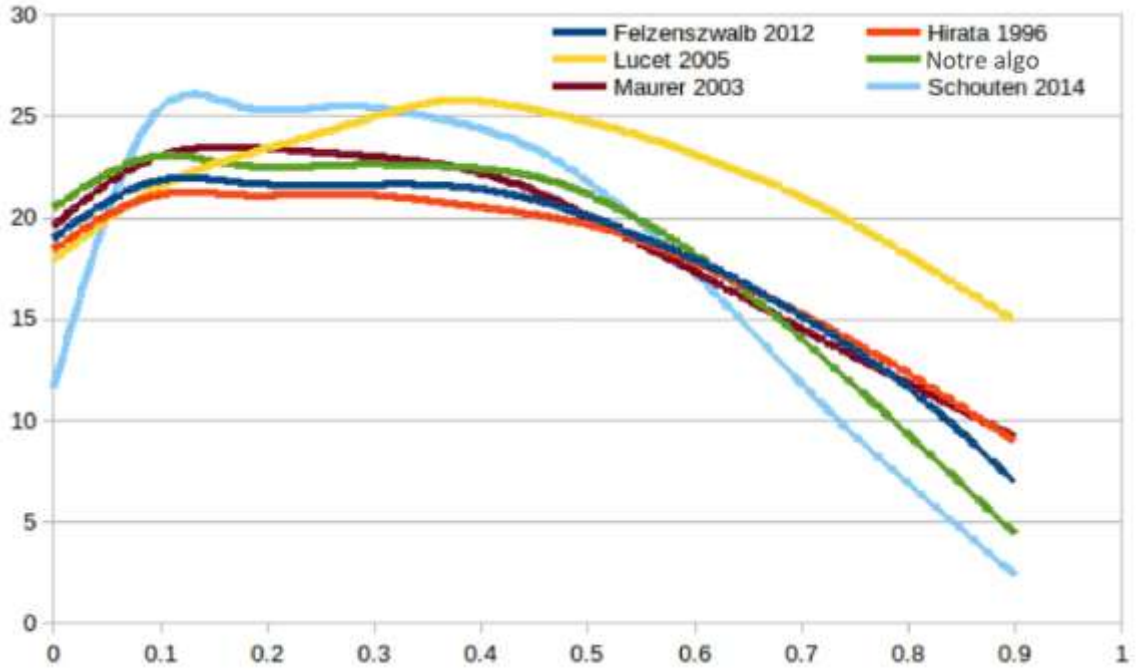


Figure 80 : Evolution du temps de calcul (en ns/pixel) pour chacun des algorithmes considérés, lorsqu'on augmente la proportion des pixels (placés aléatoirement) définissant le fond. Bien que plus lent au début, notre algorithme surclasse les autres méthodes (sauf FEED) à partir du moment où le fond devient suffisamment connecté.

Bien que la banque d'images proposée par (Fabbri et al. 2008) soit relativement exhaustive dans le type d'images qu'elle considère, il n'en reste pas moins que ces dernières sont purement artificielles, et qu'elles ne sont donc pas représentatives de cas d'application concrets. Pour cette raison, nous avons également appliqué les algorithmes à la banque d'images proposée dans (Sebastian et Kimia 2005). Elle est formée de 216 images représentants des entités du monde réel (principalement des objets et des animaux). Les temps de calcul pour chacune de ces images et pour chaque algorithme sont reportés Figure 81.

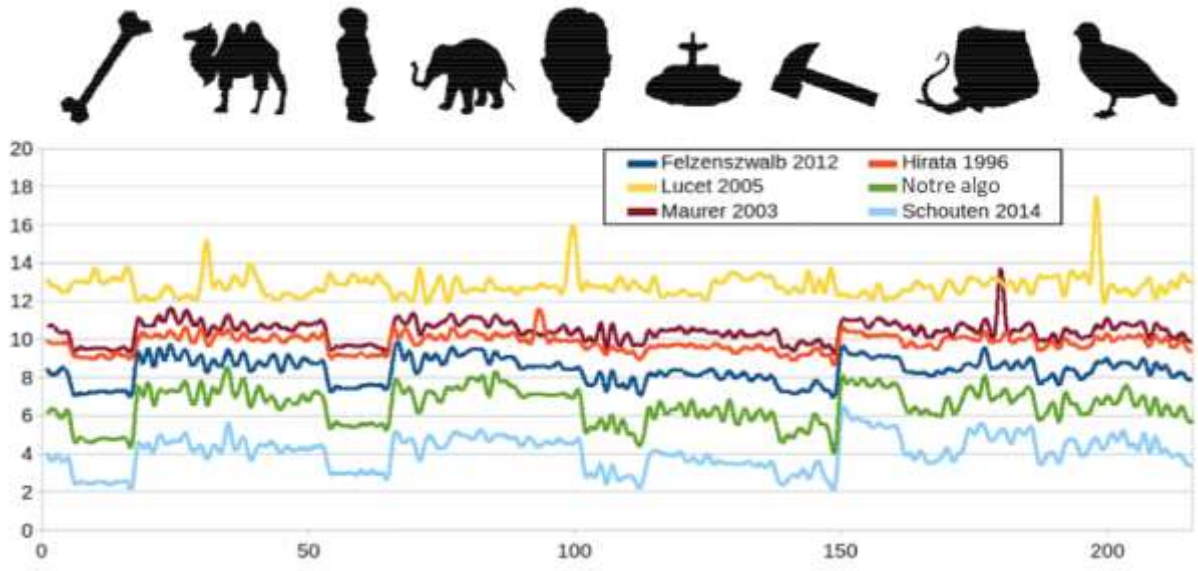


Figure 81 : Temps de calcul avec chacun des algorithmes pour les images de (Sebastian et Kimia 2005).

Clairement, la hiérarchie discutée précédemment est confirmée. L'algorithme FEED proposé par (Schouten et Broek 2014) est plus rapide, mais notre solution est juste derrière. On constate notamment un gain d'environ 20% par rapport à l'algorithme original (Felzenszwalb et Huttenlocher 2012).

L'ensemble de ces résultats expérimentaux confirment l'amélioration apportée par ce travail dans notre contexte. Comme nous l'avons vu précédemment, les obstacles projetés dans l'espace des configurations définissent des zones fortement connectées, ce qui, comme nous l'avons vu, est l'unique condition pour que notre solution surclasse les autres stratégies de l'état de l'art. Rappelons pour finir que la solution la plus rapide dans les résultats expérimentaux (FEED proposé par (Felzenszwalb et Huttenlocher 2012)), n'est pas utilisable dans notre contexte, car non généralisable aux dimensions élevées.

Annexe 4 : Fusion des cartes des distances

Cette annexe montre un exemple la fusion des cartes des distances pour tous les obstacles et les obstacles humains tel que présenté section 4.2.4. La Figure 82 montre l'environnement ayant servi à générer l'espace des configurations dans lequel sont calculées les cartes des distances.

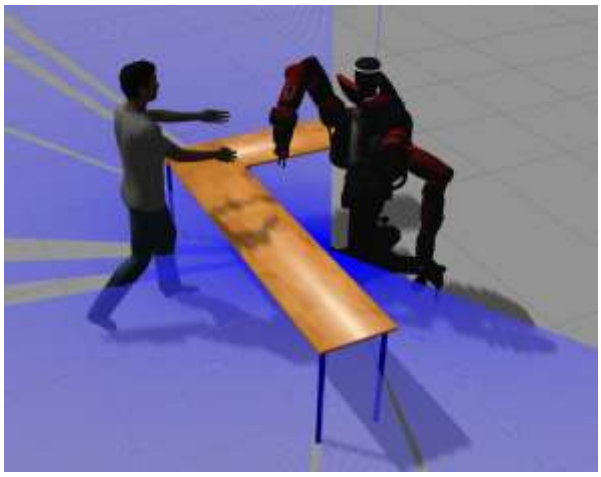


Figure 82 : Environnement en simulation servant au calcul des cartes des distances.

La Figure 83 montre localement l'influence de la fusion des cartes. La zone en rouge montre les cellules modifiées par la fusion par rapport à la carte des distances normale. Nous pouvons voir l'augmentation de cette zone conjointement avec k_h. Les Figure 84 et Figure 85 montrent cette influence sur la carte complète.

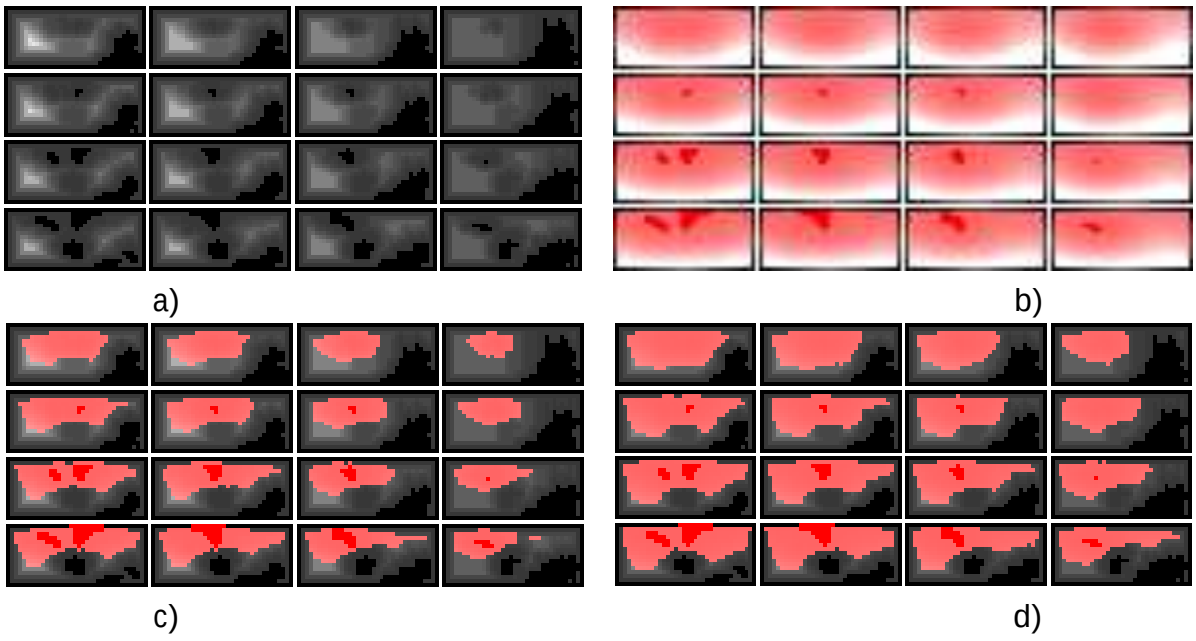


Figure 83 : zoom sur la fusion de la carte des distances pour tous les obstacles (a) avec celle calculée uniquement pour les obstacles humains (b), avec un coefficient k_h de 1,5 (c) et 3 (d).

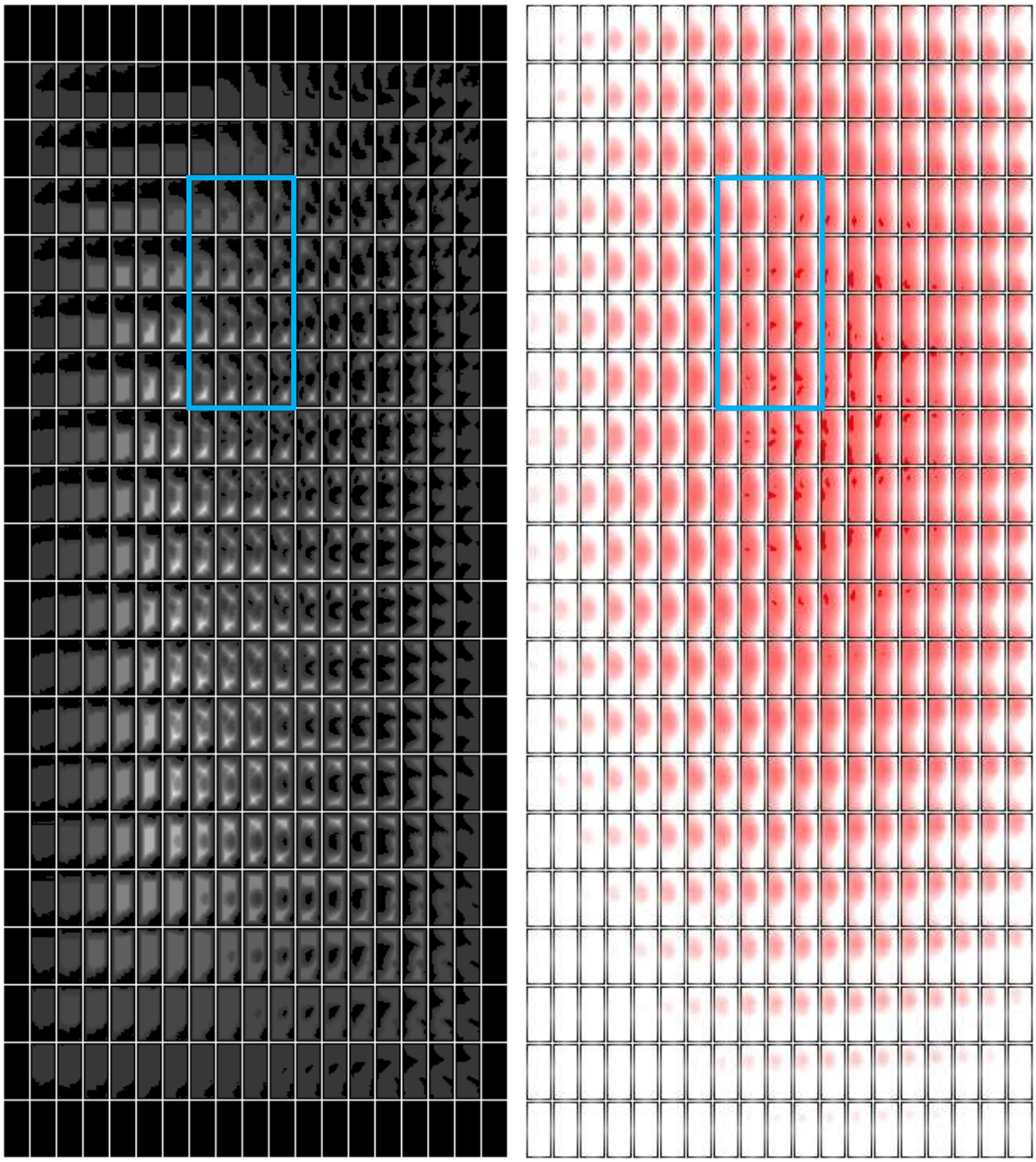
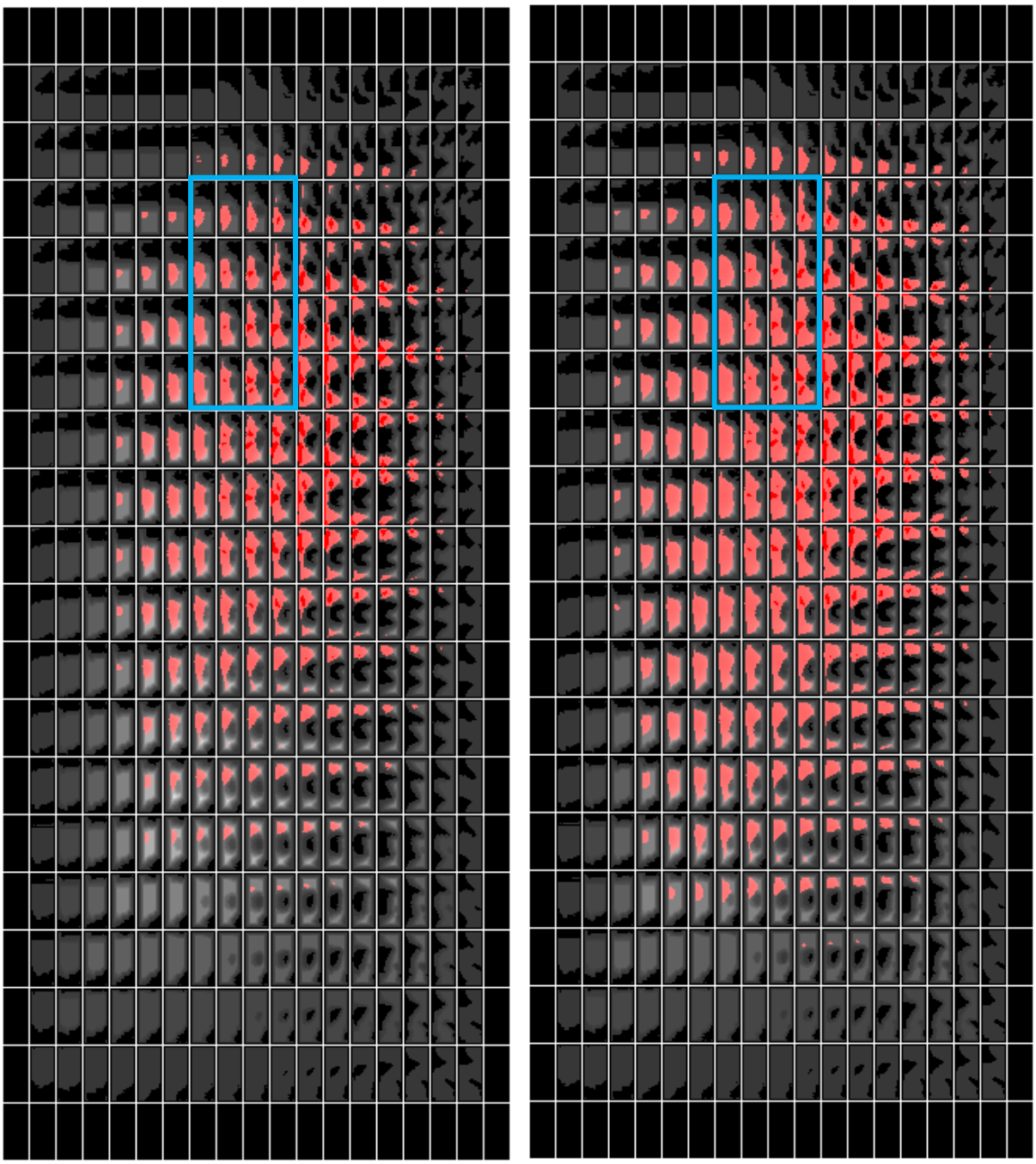


Figure 84 : Cartes des distances calculées a) pour tous les obstacles et b) seulement pour les obstacles humains



a) b)

Figure 85 : fusion des cartes des distances pour : a) $k_h = 1.5$ et b) $k_h = 3$

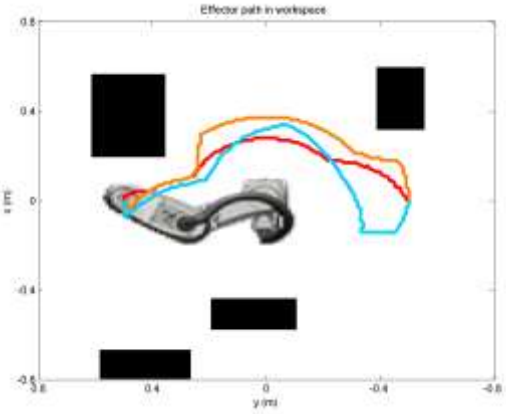
Annexe 5 : Résultats expérimentaux de la comparaison de notre planificateur avec l'algorithme A*

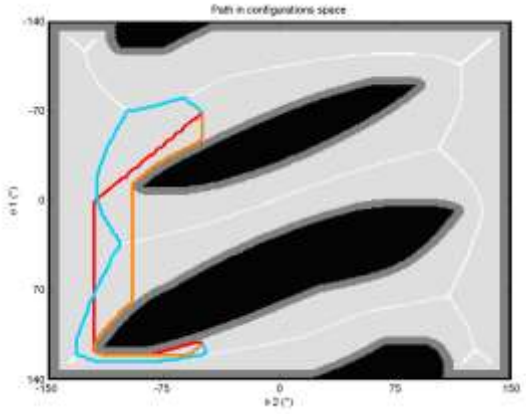
Cette annexe présente les résultats obtenus en simulations lors de la comparaison entre notre planificateur et un A*. Par souci de clarté, ceux-ci ont été synthétisés dans le manuscrit et nous allons ici présenter les résultats pour chacun des douze environnements de test. L'analyse des résultats, faite dans la section 5.2)5.2.2.2) du Chapitre 5, étant valide pour l'ensemble des tests, nous ne commenterons pas individuellement les environnements.

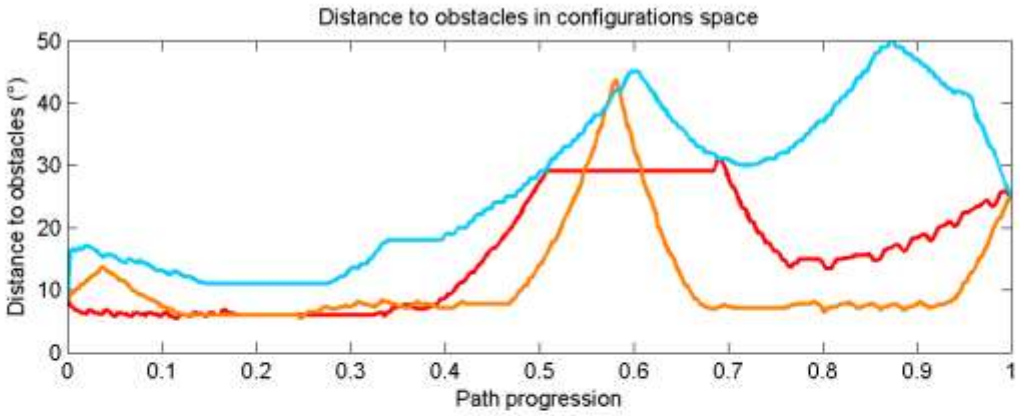
Pour rappel, les couleurs utilisées pour chacune des figures suivantes sont :

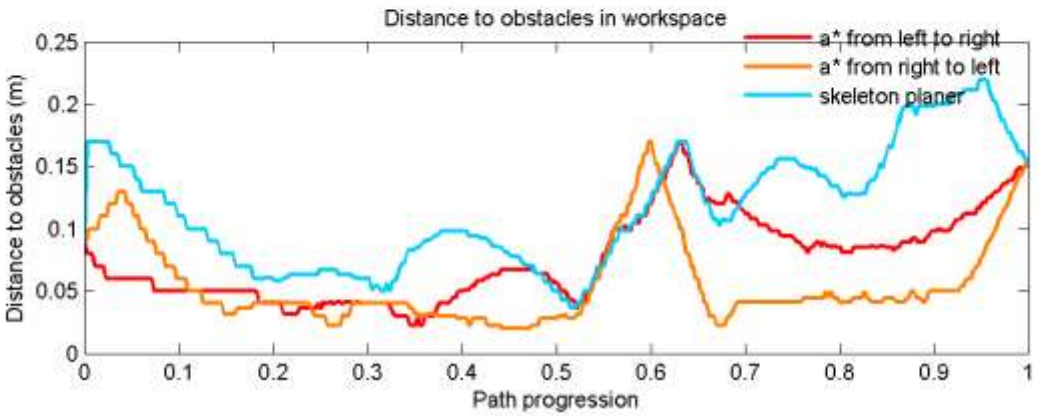
Rappelons également que les progressions des trajectoires ont été normalisées entre 0 et1.

1. Environnement 1

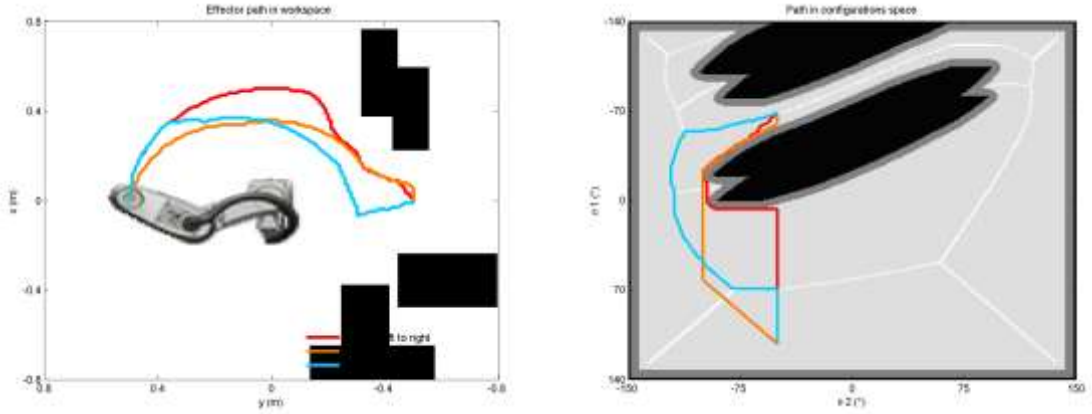


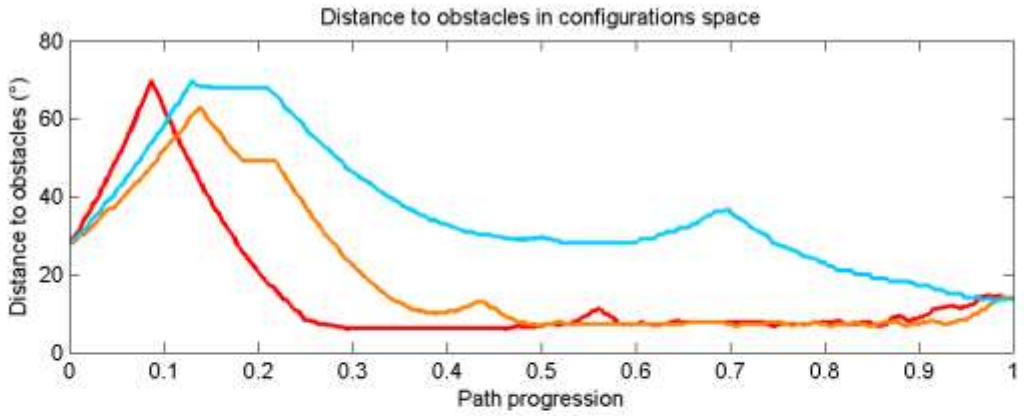


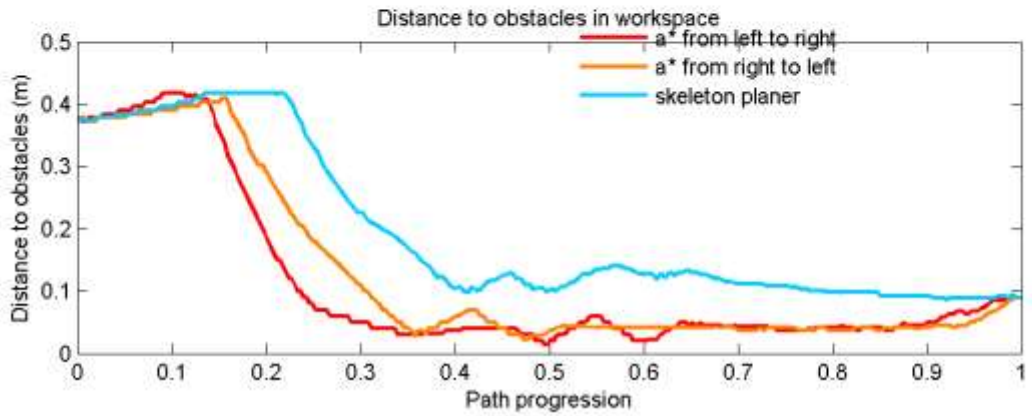




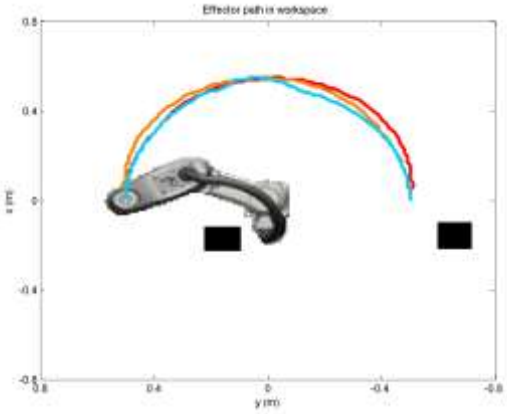
2. Environnement 2

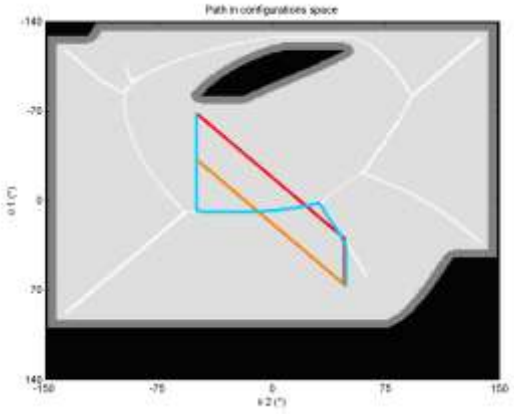


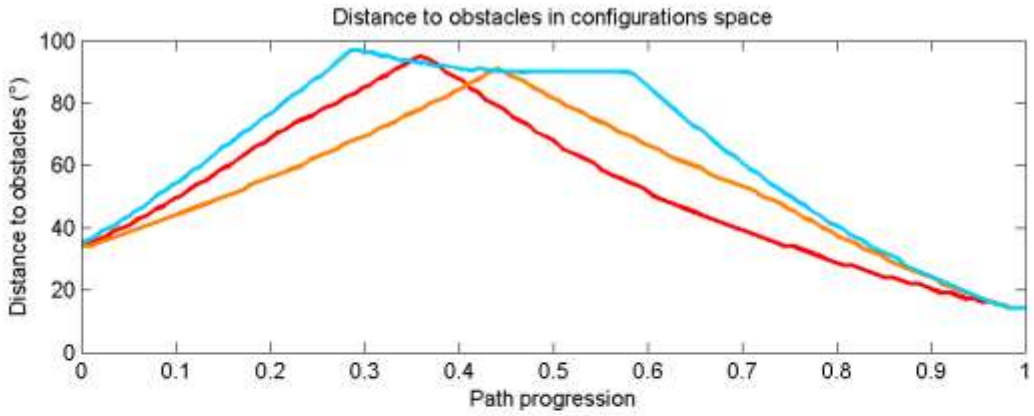


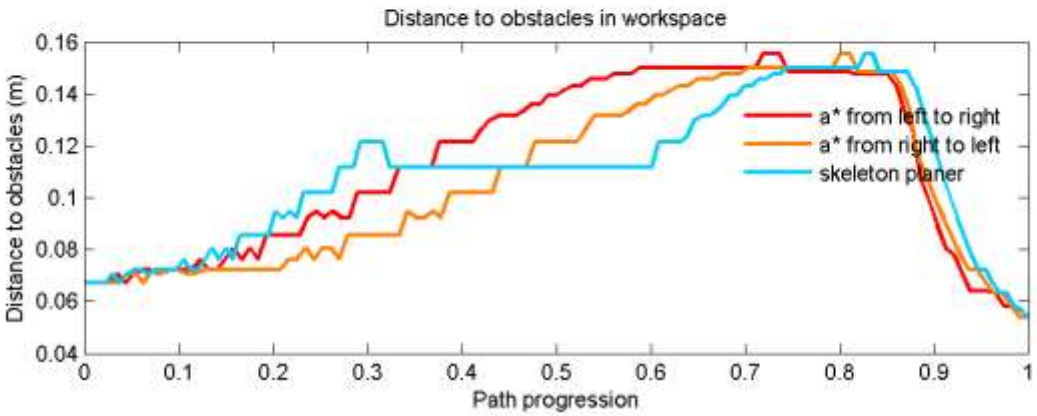


3. Environnement 3

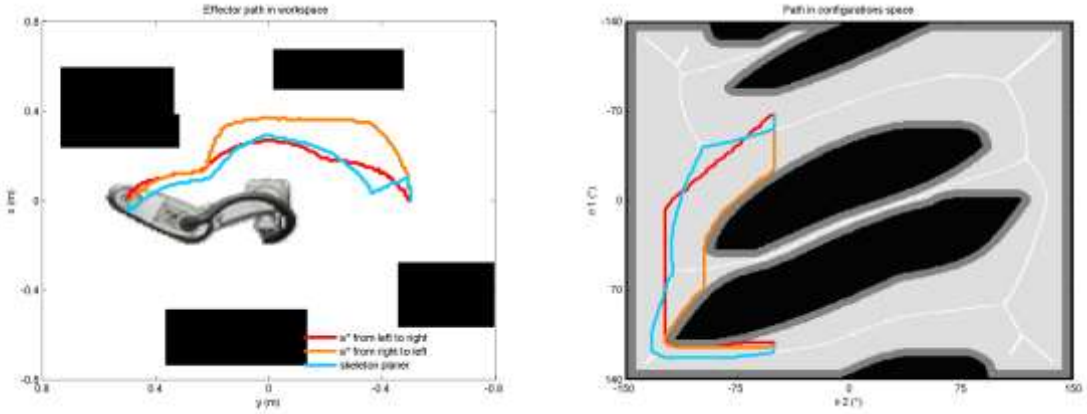


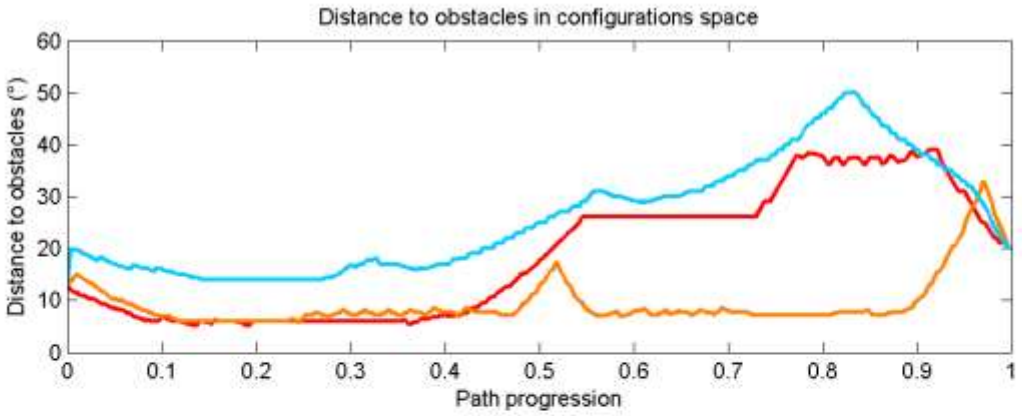


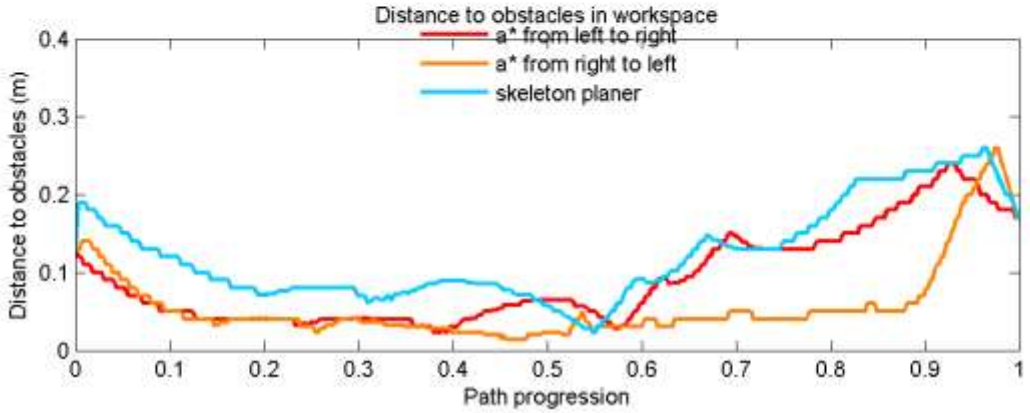




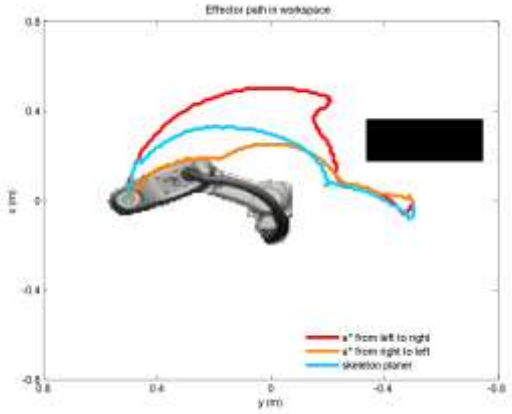
4. Environnement 4

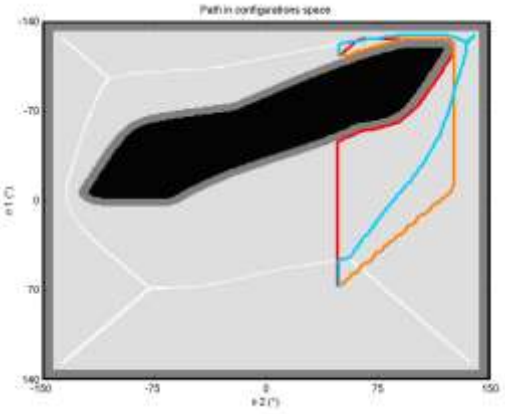


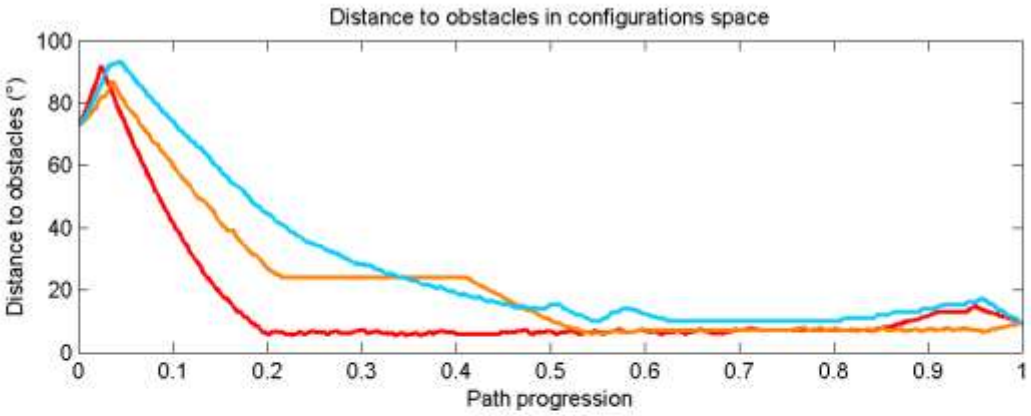


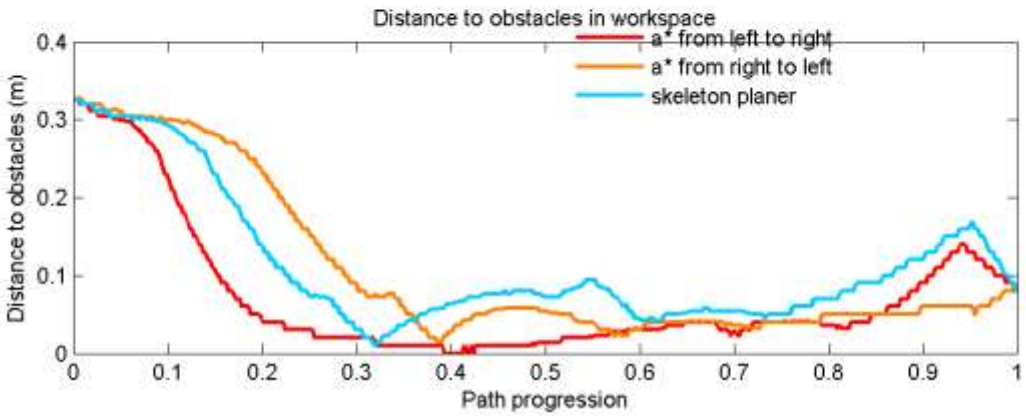


5. Environnement 5

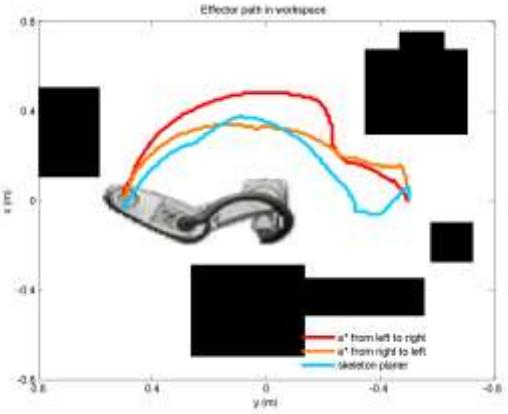


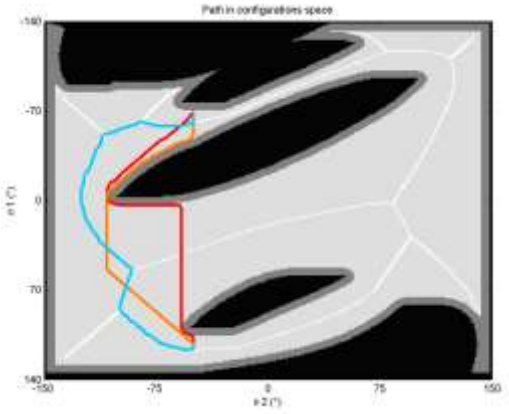


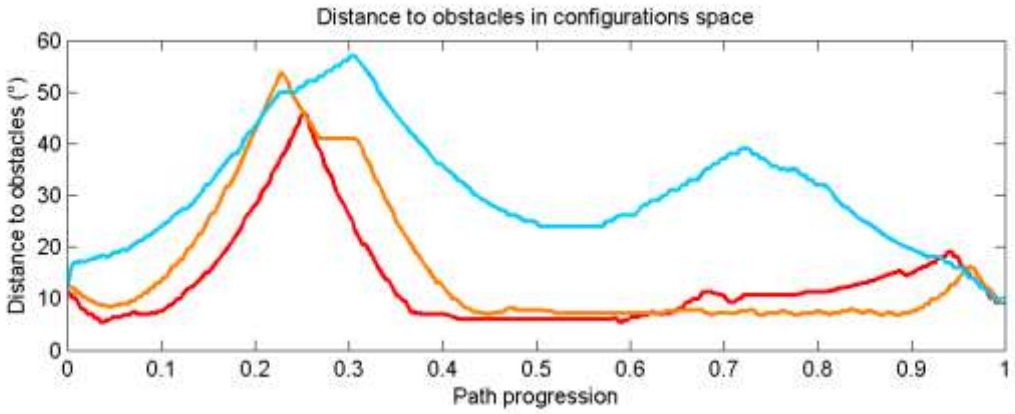


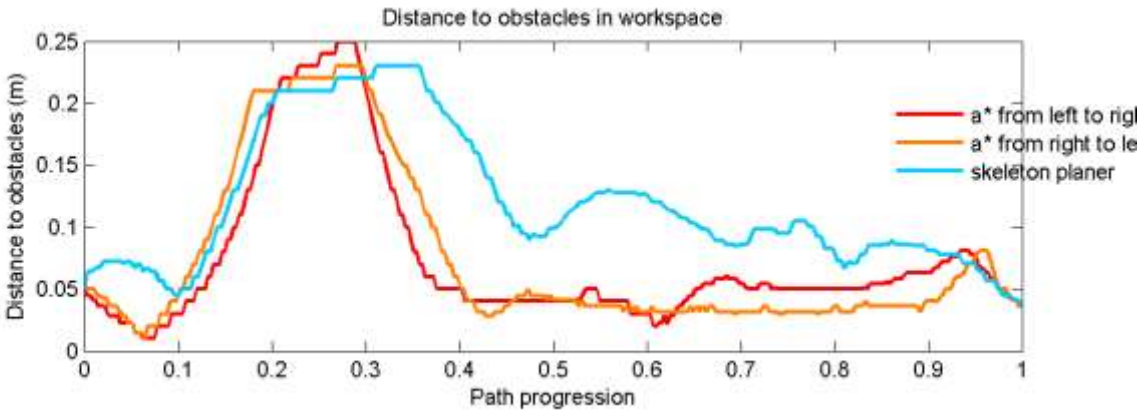


6. Environnement 6

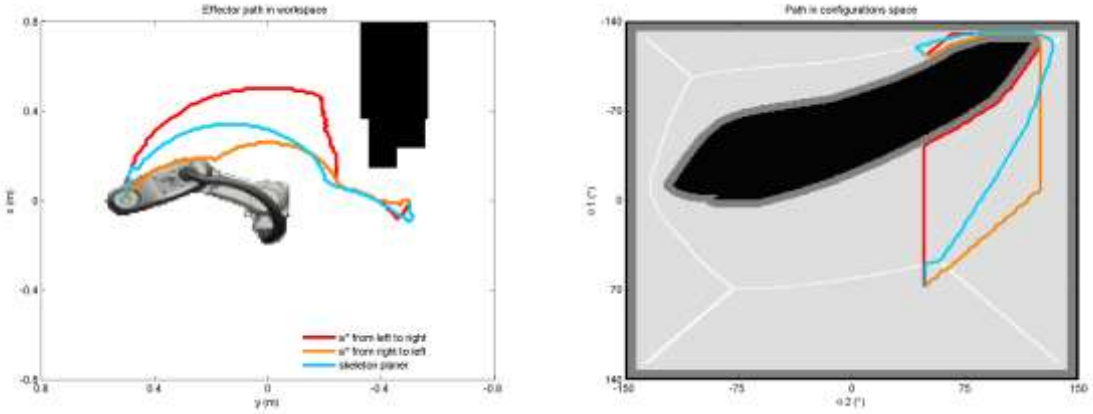


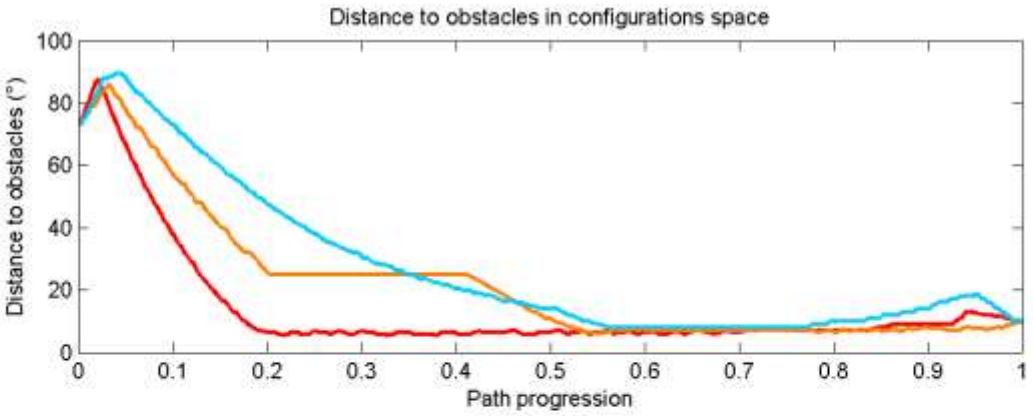


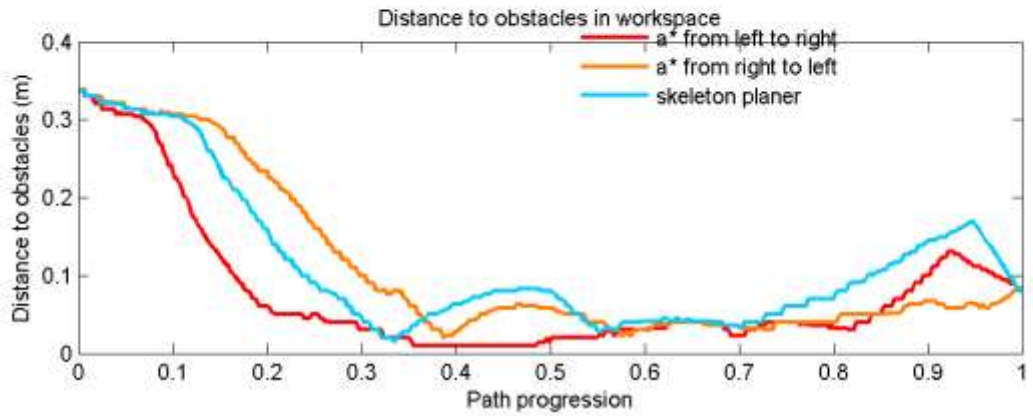




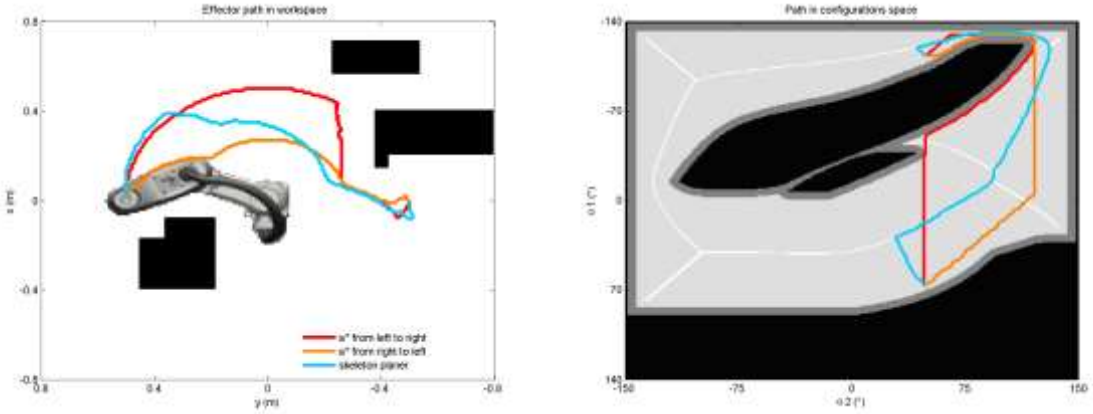
7. Environnement 7

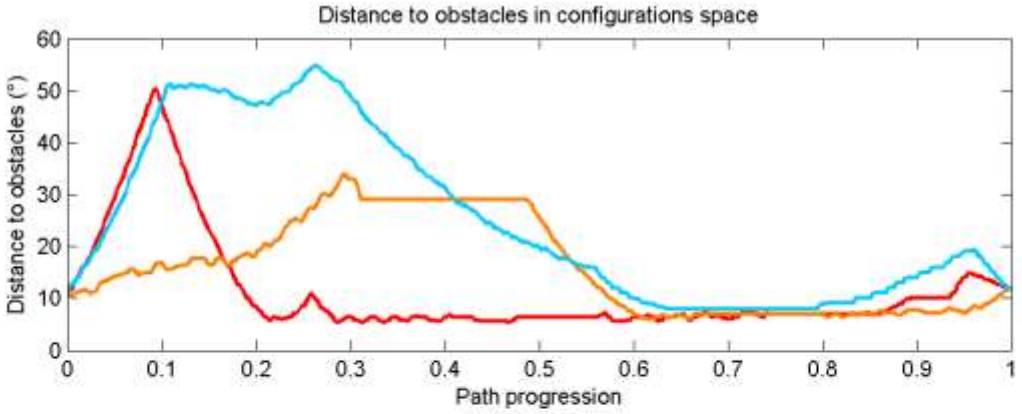


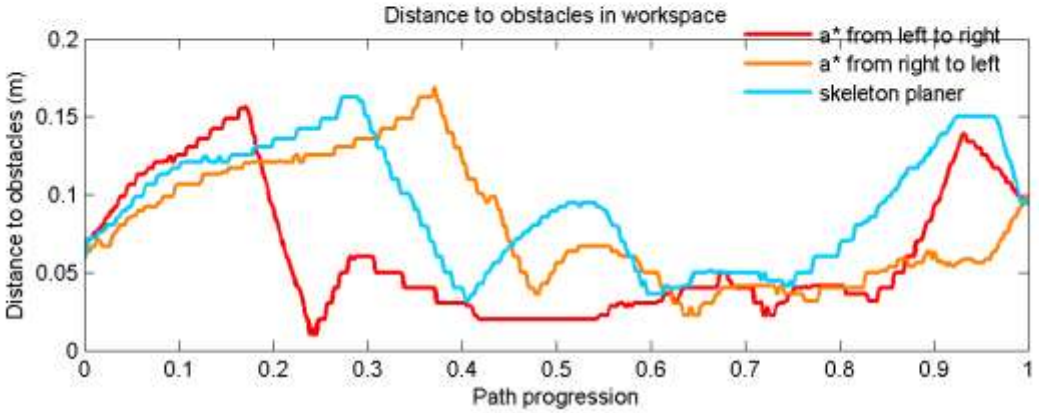




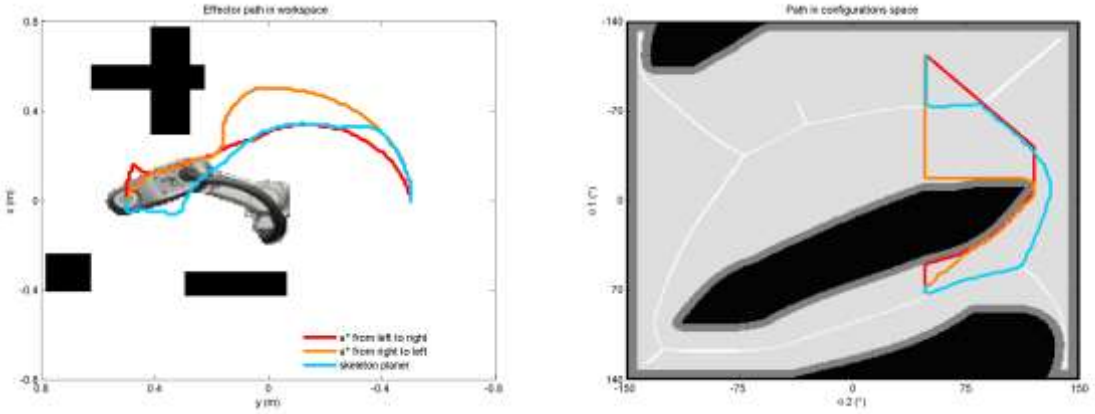
8. Environnement 8

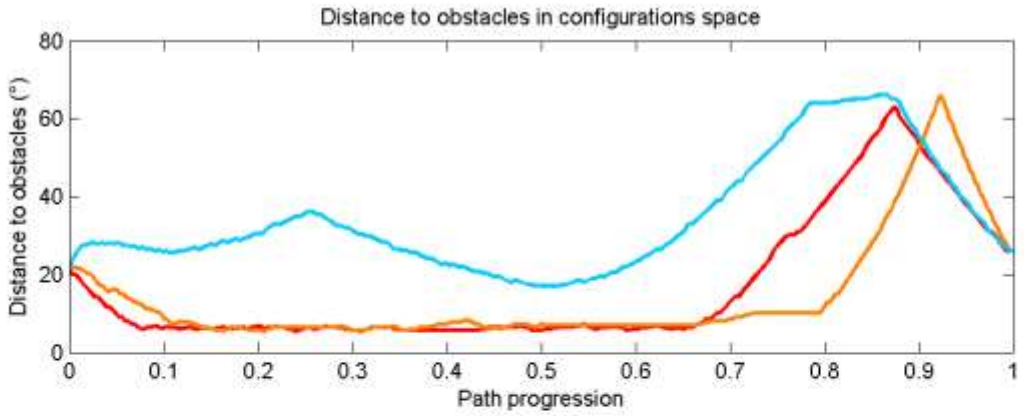


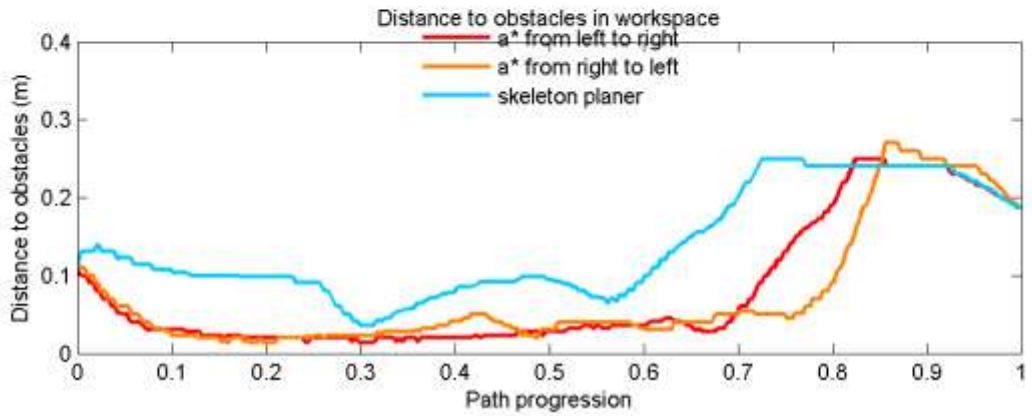




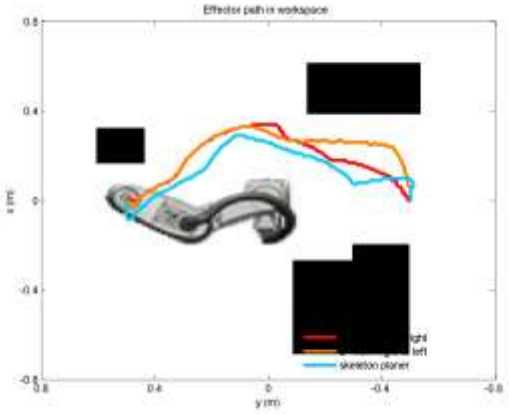
9. Environnement 9

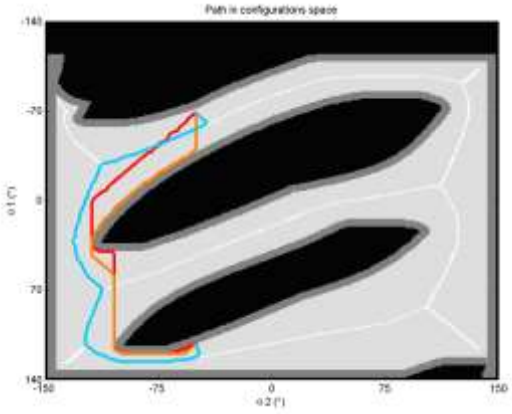


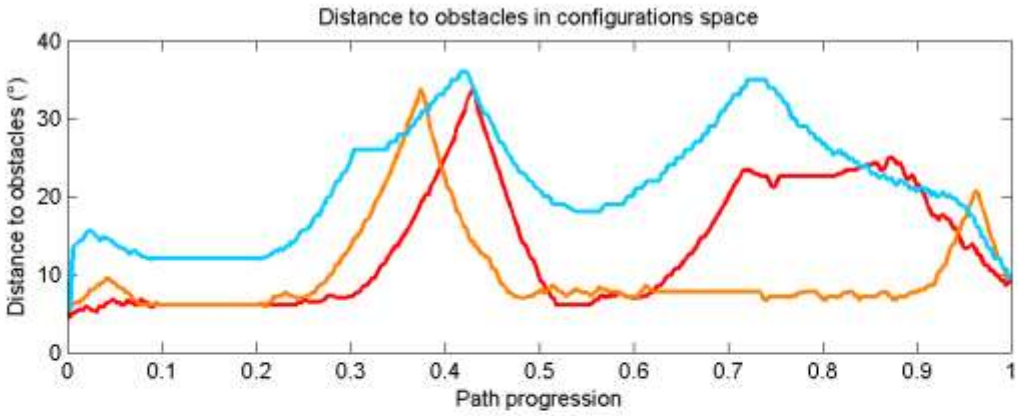


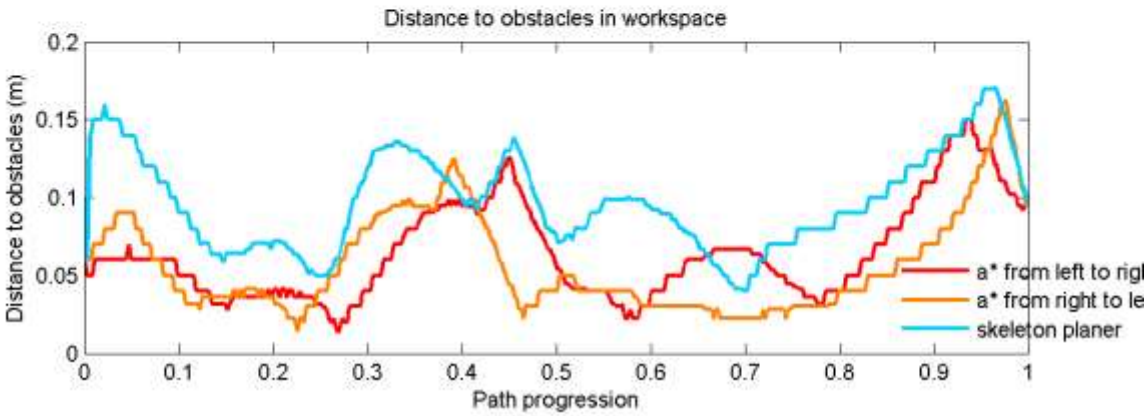


10. Environnement 10

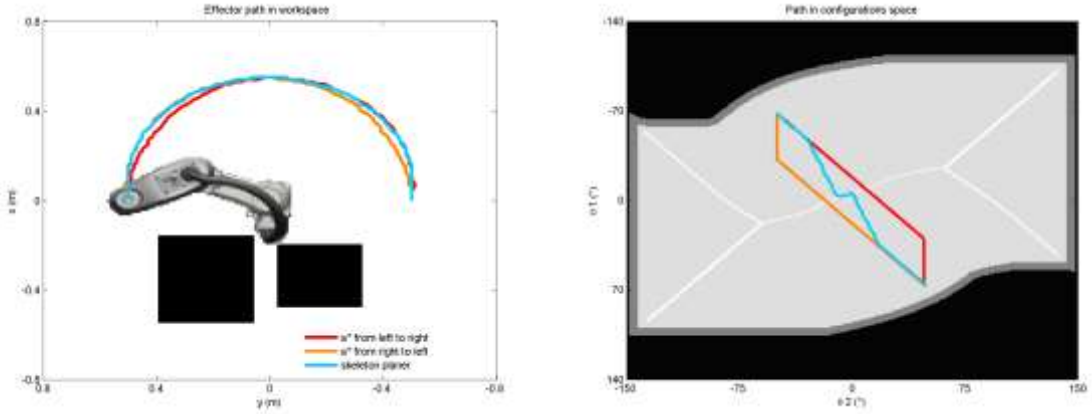


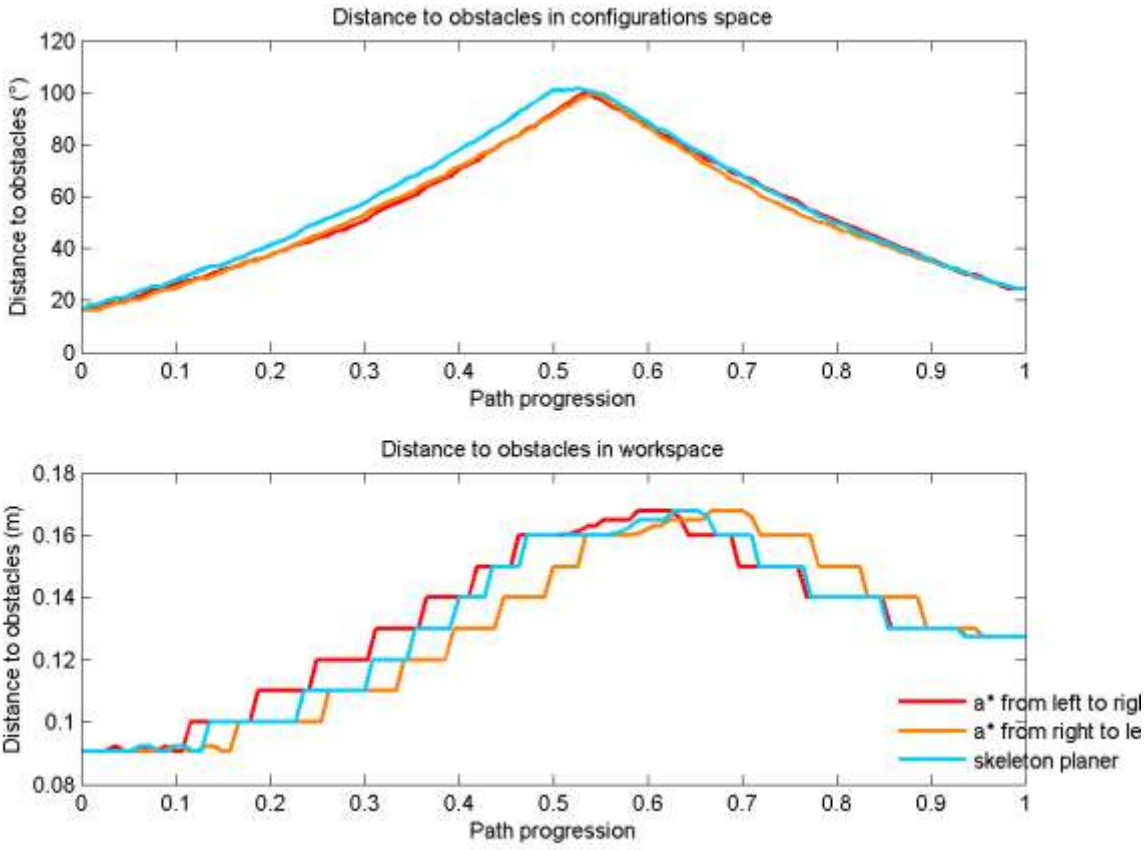




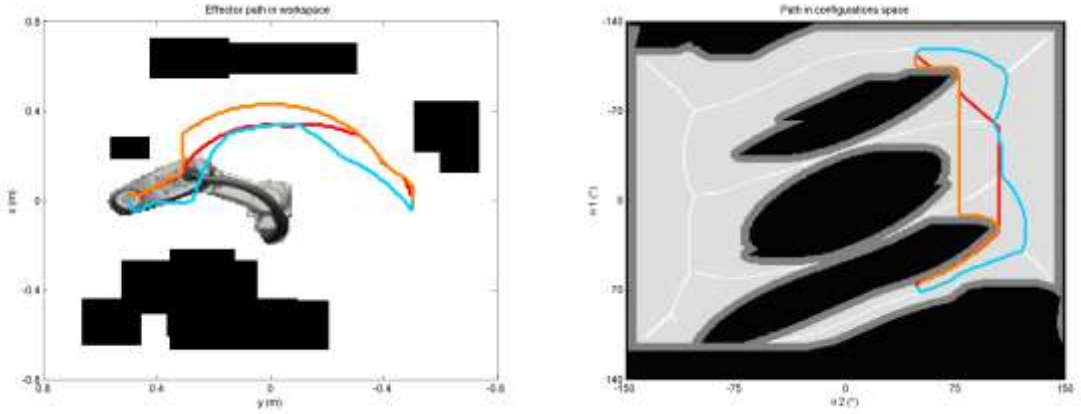


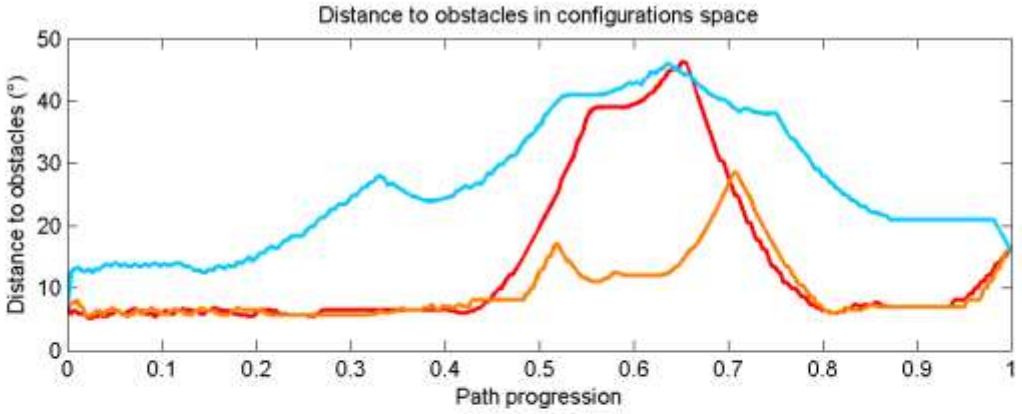
11. Environnement 11





12. Environnement 12





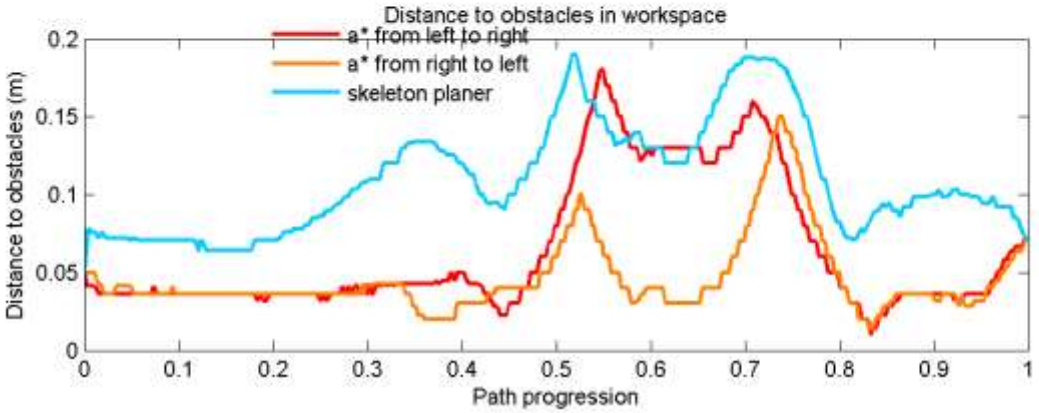


Table des figures

Table des algorithmes

Table des tableaux

Bibliographie

AFNOR. 2010. « NF EN ISO 13855 : Sécurité des machines - Positionnement des moyens de protection par rapport à la vitesse d'approche des parties du corps ». NF EN ISO 13855.

———. 2016. « ISO/TS 15066 : Robots et dispositifs robotiques - Robots coopératifs ». ISO/TS 15066.

Ahmed, M. R., et I. Kalaykov. 2010. « Static and dynamic collision safety for human robot interaction using magneto-rheological fluid based compliant robot manipulator ». In *2010 IEEE International Conference on Robotics and Biomimetics*, 370-75. https://doi.org/10.1109/ROBIO.2010.5723355.

Ahuja, Narendra, et Jen-Hui Chuang. 1997. « Shape Representation Using a Generalized Potential Field Model ». *IEEE Trans. Pattern Anal. Mach. Intell.* 19 (2): 169-76.

Alami, Rachid, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, et Raja Chatila. 2006. « Toward Human-Aware Robot Task Planning ». In *To Boldly Go Where No Human-Robot Team Has Gone Before, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-07, Stanford, California, USA, March 27-29, 2006*, 39–46. http://www.aaai.org/Library/Symposia/Spring/2006/ss06-07-006.php.

Amato, Nancy M., O. Burçhan Bayazit, Lucia K. Dale, Christopher J. H. Jones, et Daniel Vallejo. 1998. « OBPRM: an obstacle-based PRM for 3D workspaces ». In .

Bai, Xiang, Longin Jan Latecki, et Wenyu Liu. 2007. « Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution ». *IEEE Trans. Pattern Anal. Mach. Intell.* 29 (3): 449-62.

Ben-Said, H., J. Stéphant, et O. Labbani-Igbida. 2016. « Sensor-based control using finite time observer of visual signatures: Application to corridor following ». In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 418-23. https://doi.org/10.1109/IROS.2016.7759088.

Benzaid, Karima, Romain Marie, Noura Mansouri, et Ouiddad Labbani-Igbida. 2018. « Filtered Medial Surface Based Approach for 3D Collision-Free Path Planning Problem ». *J. Robotics* 2018: 4676720:1–4676720:9. https://doi.org/10.1155/2018/4676720.

Berg, J. van den, D. Ferguson, et J. Kuffner. 2006. « Anytime Path Planning and Replanning in Dynamic Environments ». In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2366-71. Orlando, FL, USA: IEEE. https://doi.org/10.1109/ROBOT.2006.1642056.

Berg, Mark de, Otfried Cheong, Marc van Kreveld, et Mark Overmars. 2008. *Computational Geometry: Algorithms and Applications*. 3e éd. Berlin Heidelberg: Springer-Verlag. https://www.springer.com/gp/book/9783540779735.

Bicchi, Antonio, Giovanni Tonietti, Michele Bavaro, et Marco Piccigallo. 2005. « Variable Stiffness Actuators for Fast and Safe Motion Control ». In *Robotics*

Research. The Eleventh International Symposium, édité par Paolo Dario et Raja Chatila, 527–536. Berlin, Heidelberg: Springer Berlin Heidelberg.

Blum, Harold Francis. 1967. « A transformation for extracting new descriptors of shape ». In *Models for the Perception of Speech and Visual Form*, 362--380.

Blum, Harry, et Roger N. Nagel. 1978. « Shape description using weighted symmetric axis features ». *Pattern Recognition* 10 (3): 167-80.

Borgefors, G, I Ragnemalm, et G. Sanniti di Baja. 1989. « A medical axis transform algorithm for compression and vectorization of document images ». In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, 1850-53 vol.3.

Brandt, J.W., et A. Jain. 1991. « The Euclidian Distance Transform : finding the local maxima and reconstructing the shape ». In *7th Scandinavian Conference on Image Analysis*, 974-81 vol.2.

Breu, Heinz, Joseph Gil, David G. Kirkpatrick, et Michael Werman. 1995. « Linear Time Euclidean Distance Algorithms ». *IEEE Trans. Pattern Anal. Mach. Intell.* 17 (5): 529–533.

Broquère, Xavier. 2011. « Planification de trajectoire pour la manipulation d'objets et l'interaction Homme-robot ». Theses, Université Paul Sabatier - Toulouse III. https://tel.archives-ouvertes.fr/tel-00644776.

Burgard, W., O. Brock, et C. Stachniss. 2008. « Safety Evaluation of Physical Human-Robot Interaction via Crash-Testing ». In *Robotics: Science and Systems III*. MITP. https://ieeexplore-ieee-org.ezproxy.unilim.fr/document/6280117.

Ceriani, N. M., G. Buizza Avanzini, A. M. Zanchettin, L. Bascetta, et P. Rocco. 2013. « Optimal placement of spots in distributed proximity sensors for safe human-robot interaction ». In *2013 IEEE International Conference on Robotics and Automation*, 5858-63. https://doi.org/10.1109/ICRA.2013.6631420.

Chaussard, J., M. Couprie, et H. Talbot. 2011. « Robust skeletonization using the discrete λ-medial axis ». *Pattern Recognition Letters* 32 (9).

Chazelle, B. 1987. « Approximation and decomposition of shapes ». *Algorithmic and Geometric Aspects of Robotics*, 145–185.

Cheung, E., et V. Lumelsky. 1989. « Development of sensitive skin for a 3D robot arm operating in an uncertain environment ». In *Proceedings, 1989 International Conference on Robotics and Automation*, 1056-61 vol.2. https://doi.org/10.1109/ROBOT.1989.100121.

Denny, J., E. Greco, S. Thomas, et N. M. Amato. 2014. « MARRT: Medial Axis biased rapidly-exploring random trees ». In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 90-97. https://doi.org/10.1109/ICRA.2014.6906594.

Dollar, P., C. Wojek, B. Schiele, et P. Perona. 2012. « Pedestrian Detection: An Evaluation of the State of the Art ». *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (4): 743-61. https://doi.org/10.1109/TPAMI.2011.155.

Ellekilde, Lars-Peter, et Henrik Gordon Petersen. 2013. « Motion planning efficient trajectories for industrial bin-picking ». *I. J. Robotics Res.* 32 (9-10): 991–1004. https://doi.org/10.1177/0278364913487237.

Fabbri, Ricardo, Luciano da Fontoura Costa, Julio C. Torelli, et Odemir Martinez Bruno. 2008. « 2D Euclidean distance transform algorithms: A comparative survey ». *ACM Comput. Surv.* 40 (1): 2:1–2:44.

Felzenszwalb, Pedro F., et Daniel P. Huttenlocher. 2012. « Distance Transforms of Sampled Functions ». *Theory of Computing* 8 (1): 415–428.

Fischer, Harrisson. 2018. « Analyse de risque et sécurisation de l'interaction homme-machine dans un contexte collaboratif temps réel ». http://www.theses.fr/s162120.

Flacco, F., T. Kröger, A. De Luca, et O. Khatib. 2012. « A depth space approach to human-robot collision avoidance ». In *2012 IEEE International Conference on Robotics and Automation*, 338-45. https://doi.org/10.1109/ICRA.2012.6225245.

Flacco, F., et A. De Luca. 2010. « Multiple depth/presence sensors: Integration and optimal placement for human/robot coexistence ». In *2010 IEEE International Conference on Robotics and Automation*, 3916-23. https://doi.org/10.1109/ROBOT.2010.5509125.

Fuseiller, G., R. Marie, G. Mourioux, E. Duno, et O. Labbani-Igbida. 2018. « Reactive path planning for collaborative robot using configuration space skeletonization ». In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 29-34. https://doi.org/10.1109/SIMPAR.2018.8376267.

Garrido, Santiago, Luis Moreno, Dolores Blanco, et Plotr Jurewitz. 2011. « Path planning for mobile robot navigation using Voronoi diagram and fast marching ». *International Journal of Robotics and Automation*, 154–176.

Gonzalez, R., M. Kloetzer, et C. Mahulea. 2017. « Comparative study of trajectories resulted from cell decomposition path planning approaches ». In *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, 49-54. https://doi.org/10.1109/ICSTCC.2017.8107010.

Haddadin, Sami, Simon Haddadin, Augusto Khoury, Tim Rokahr, Sven Parusel, Rainer Burgkart, Antonio Bicchi, et Alin Albu-Schäffer. 2012. « On Making Robots Understand Safety: Embedding Injury Knowledge into Control ». *The International Journal of Robotics Research* 31 (13): 1578-1602. https://doi.org/10.1177/0278364912462256.

Heinzmann, Jochen, et Alexander Zelinsky. 2003. « Quantitative Safety Guarantees for Physical Human-Robot Interaction ». *I. J. Robotics Res.* 22 (7-8): 479–504. https://doi.org/10.1177/02783649030227004.

Henrich, D., et S. Kuhn. 2006. « Modeling Intuitive behavior for safe human/robot coexistence cooperation ». In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 3929-34. https://doi.org/10.1109/ROBOT.2006.1642304.

Hirata, Tomio. 1996. « A Unified Linear-Time Algorithm for Computing Distance Maps ». *Information Processing Letters* 58 (3): 129–133.

Hirzinger, G., N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, et M. Schedl. 2002. « DLR's torque-controlled light weight robot III-are we reaching the technological limits now? » In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 2:1710-16 vol.2. https://doi.org/10.1109/ROBOT.2002.1014788.

Hoffmann, A., A. Poeppel, A. Schierl, et W. Reif. 2016. « Environment-aware proximity detection with capacitive sensors for human-robot-interaction ». In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 145-50. https://doi.org/10.1109/IROS.2016.7759047.

Holleman, Christopher, et Lydia E. Kavraki. 2000. « A Framework for Using the Workspace Medial Axis in PRM Planners ». In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*, 1408–1413.

Hornung, Armin, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, et Wolfram Burgard. 2013. « OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees ». *Autonomous Robots* 34 (3): 189-206. https://doi.org/10.1007/s10514-012-9321-0.

Hsu, David, Jean-Claude Latombe, et Hanna Kurniawati. 2006. « On the Probabilistic Foundations of Probabilistic Roadmap Planning ». *The International Journal of Robotics Research* 25 (7): 627-43.

Hun-Ok Lim, et K. Tanie. 1999. « Collision-tolerant control of human-friendly robot with viscoelastic trunk ». *IEEE/ASME Transactions on Mechatronics* 4 (4): 417-27. https://doi.org/10.1109/3516.809520.

Ikuta, Koji, Hideki Ishii, et Makoto Nokata. 2003. « Safety Evaluation Method of Design and Control for Human-Care Robots ». *The International Journal of Robotics Research* 22 (5): 281-97. https://doi.org/10.1177/0278364903022005001.

Jaillet, L., J. Cortes, et T. Simeon. 2008. « Transition-based RRT for path planning in continuous cost spaces ». In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2145-50. https://doi.org/10.1109/IROS.2008.4650993.

Jaillet, L., A. Yershova, S. M. La Valle, et T. Simeon. 2005. « Adaptive tuning of the sampling domain for dynamic-domain RRTs ». In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2851-56. https://doi.org/10.1109/IROS.2005.1545607.

Jaillet, Leonard, et Thierry Siméon. 2004. « A PRM-based motion planner for dynamically changing environments ». In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, September 28 - October 2, 2004*, 1606–1611. https://doi.org/10.1109/IROS.2004.1389625.

Kavraki, L. E., P. Svestka, J. C. Latombe, et M. H. Overmars. 1996. « Probabilistic roadmaps for path planning in high-dimensional configuration spaces ». *IEEE*

Transactions on Robotics and Automation 12 (4): 566-80. https://doi.org/10.1109/70.508439.

Kloetzer, M., et N. Ghita. 2011. « Software tool for constructing cell decompositions ». In *2011 IEEE International Conference on Automation Science and Engineering*, 507-12. https://doi.org/10.1109/CASE.2011.6042492.

Knepper, Ross, et Matthew Mason. 2012. « Real-time informed path sampling for motion planning search ». *The International Journal of Robotics Research* 31 (septembre): 1231-50. https://doi.org/10.1177/0278364912456444.

Kulic, D., et E. Croft. 2004. « Safe planning for human-robot interaction ». In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, 2:1882-1887 Vol.2. https://doi.org/10.1109/ROBOT.2004.1308098.

Kulic, Dana, et Elizabeth A. Croft. 2005. « Safe planning for human-robot interaction ». *J. Field Robotics* 22 (7): 383–396. https://doi.org/10.1002/rob.20073.

Kulić, Dana, et Elizabeth A. Croft. 2006. « Real-time safety for human–robot interaction ». *Robotics and Autonomous Systems* 54 (1): 1-12. https://doi.org/10.1016/j.robot.2005.10.005.

Lam, Louisa, Seong-Whan Lee, et Ching Y. Suen. 1992. « Thinning Methodologies-A Comprehensive Survey ». *IEEE Trans. Pattern Anal. Mach. Intell.*, 869–885.

LaValle, Steven M. 1998. « Rapidly-Exploring Random Trees: A New Tool for Path Planning ». In .

———. 2006. *Planning algorithms*. Cambridge university press. https://books.google.fr/books?hl=fr&lr=&id=-PwLBAAAQBAJ&oi=fnd&pg=PT9&dq=S.M.+LaValle.+Planning+algorithms.&ots=0gHt5kArts&sig=TavNUjoZZEd09L8gRPpDn5Cm1zw.

LaValle, Steven M., et Jr. James J. Kuffner. 2001. « Randomized Kinodynamic Planning ». *The International Journal of Robotics Research* 20 (5): 378-400.

Leymarie, Frederic F., et Martin D. Levine. 1992. « Simulating the Grassfire Transform Using an Active Contour Model ». *IEEE Trans. Pattern Anal. Mach. Intell.* 14 (1): 56-75.

Lien, Jyh-Ming, S. L. Thomas, et N. M. Amato. 2003. « A general framework for sampling on the medial axis of the free space ». In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 3:4439-44 vol.3.

Liu, Lu, Erin W. Chambers, David Letscher, et Tao Ju. 2011. « Extended grassfire transform on medial axes of 2D shapes ». *Comput. Aided Des.* 43 (11): 1496–1505.

Llata, J. R., E. G. Sarabia, J. Arce, et J. P. Oria. 1998. « Fuzzy controller for obstacle avoidance in robotic manipulators using ultrasonic sensors ». In *AMC'98 - Coimbra. 1998 5th International Workshop on Advanced Motion Control. Proceedings (Cat. No.98TH8354)*, 647-52. https://doi.org/10.1109/AMC.1998.743615.

Lotufo, Roberto, et Francisco A. Zampirolli. 2001. « Fast Multidimensional Parallel Euclidean Distance Transform Based on Mathematical Morphology ». In *14th*

Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2001), 15-18 October 2001, Florianopolis, Brazil, 100–105.

Lozano-Perez. 1983. « Spatial Planning: A Configuration Space Approach ». *IEEE Transactions on Computers* C-32 (2): 108-20. https://doi.org/10.1109/TC.1983.1676196.

Lucet, Yves. 2005. « A Linear Euclidean Distance Transform Algorithm Based on the Linear-Time Legendre Transform ». In *Second Canadian Conference on Computer and Robot Vision (CRV 2005), 9-11 May 2005, Victoria, BC, Canada*, 262–267.

Marie, R., O. Labbani-Igbida, et E. M. Mouaddib. 2013. « The delta-medial axis: A robust and linear time algorithm for Euclidian skeleton computation ». In *2013 IEEE International Conference on Image Processing*, 3523-26. https://doi.org/10.1109/ICIP.2013.6738727.

MARIE, Romain. 2014. « Exploration autonome et construction de cartes topologiques référencées vision omnidirectionnelle ». Université de Picardie Jules Verne.

Matheron, G. 1988. « Examples of topological properties of skeletons. » *Image Analysis and Mathematical Morphology, Theoretical advances* 2.

Mattioli, J. 1993. « Probl�mes inverses et relations diff�rentielles en morphologie math�matique ». *Th�se, Universit� Paris Dauphine*.

Maurer, Calvin R., Rensheng Qi, Vijay Raghavan, et Senior Member. 2003. « A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 265–270.

Meguenani, Anis, Vincent Padois, et Philippe Bidaud. 2015. « Control of robots sharing their workspace with humans: an energetic approach to safety ». In *IROS 2015*, 7. Hamburg, Germany. https://hal.archives-ouvertes.fr/hal-01179822.

Meijster, Arnold, Jos B. T. M. Roerdink, et Wim H. Hesselink. 2000. « A General Algorithm for Computing Distance Transforms in Linear Time ». In *International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing, ISMM 2000, Palo Alto, CA, USA, June 26-28, 2000*, 331–340.

Moayer, Bijan, et King-Sun Fu. 1975. « A syntactic approach to fingerprint pattern recognition. » *Pattern Recognition* 7 (1-2): 1-23.

Moravec, H., et A. Elfes. 1985. « High resolution maps from wide angle sonar ». In *1985 IEEE International Conference on Robotics and Automation Proceedings*, 2:116-21. https://doi.org/10.1109/ROBOT.1985.1087316.

Nokata, M., K. Ikuta, et H. Ishii. 2002. « Safety-optimizing method of human-care robot design and control ». In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 2:1991-96 vol.2. https://doi.org/10.1109/ROBOT.2002.1014833.

Otani, K., K. Bouyarmane, et S. Ivaldi. 2018. « Generating Assistive Humanoid Motions for Co-Manipulation Tasks with a Multi-Robot Quadratic Program Controller ». In

2018 IEEE International Conference on Robotics and Automation (ICRA), 3107-13. https://doi.org/10.1109/ICRA.2018.8463167.

Pacchierotti, Elena, Henrik I. Christensen, et Patric Jensfelt. 2006. « Embodied Social Interaction for Service Robots in Hallway Environments ». In *Field and Service Robotics*, édité par Peter Corke et Salah Sukkariah, 293-304. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg.

Park, J., et J. Song. 2009. « Collision analysis and evaluation of collision safety for service robots working in human environments ». In *2009 International Conference on Advanced Robotics*, 1-6.

Pfaltz, John, et Azriel Rosenfeld. 1967. « Computer representation of planar regions by their skeletons ». *Commun. ACM* 10 (février): 119-22. https://doi.org/10.1145/363067.363120.

Povse, B., D. Koritnik, R. Kamnik, T. Bajd, et M. Munih. 2010. « Industrial robot and human operator collision ». In *2010 IEEE International Conference on Systems, Man and Cybernetics*, 2663-68. https://doi.org/10.1109/ICSMC.2010.5641897.

Pratt, G. A., et M. M. Williamson. 1995. « Series elastic actuators ». In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, 1:399-406 vol.1. https://doi.org/10.1109/IROS.1995.525827.

Puls, Stephan, Jrgen Graf, et Heinz Wr. 2012. « Cognitive Robotics in Industrial Environments ». In *Human Machine Interaction - Getting Closer*, édité par Inaki Maurtua. InTech. http://www.intechopen.com/books/human-machine-interaction-getting-closer/cognitive-robotics-in-industrial-environments.

Riley, J., et L. Calabi. 1964. « Certain properties of circles inscribed in simple closed curves. » *Technical report 59281, Park Math. Lab. Inc*.

Robla-Gomez, S., Victor M. Becerra, J. R. Llata, E. Gonzalez-Sarabia, C. Torre-Ferrero, et J. Perez-Oria. 2017. « Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments ». *IEEE Access* 5: 26754-73. https://doi.org/10.1109/ACCESS.2017.2773127.

Saito, Toyofumi, et Jun-Ichiro Toriwaki. 1994. « New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications ». *Pattern Recognition* 27 (11): 1551-65.

Santis, Agostino De, Bruno Siciliano, Alessandro De Luca, et Antonio Bicchi. 2008. « An atlas of physical human–robot interaction ». *Mechanism and Machine Theory* 43 (3): 253-70. https://doi.org/10.1016/j.mechmachtheory.2007.03.003.

Schmitt, M., et J. Mattioli. 1993. « Morphologie math�matique ». *Masson*.

Schouten, Theo E., et Egon L. van den Broek. 2014. « Fast Exact Euclidean Distance (FEED): A New Class of Adaptable Distance Transforms ». *IEEE Trans. Pattern Anal. Mach. Intell.* 36 (11): 2159–2172.

Sebastian, Thomas B., et Benjamin B. Kimia. 2005. « Curves vs. skeletons in object recognition ». *Signal Processing* 85 (2): 247-63.

Shih, Frank Yeong-Chyang, et Owen Robert Mitchell. 1992. « A mathematical morphology approach to Euclidean distance transformation ». *IEEE Trans. Image Processing* 1 (2): 197–204.

Shotton, J., A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, et A. Blake. 2011. « Real-time human pose recognition in parts from single depth images ». In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1297-1304. https://doi.org/10.1109/CVPR.2011.5995316.

Siddiqi, Kaleem, Ali Shokoufandeh, Sven J. Dickinson, et Steven W. Zucker. 1998. *Shock Graphs and Shape Matching*.

Simeon, T., J. P. Laumond, et F. Lamiraux. 2001. « Move3D: A generic platform for path planning ». In *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century. (Cat. No.01TH8560)*, 25-30. https://doi.org/10.1109/ISATP.2001.928961.

Sisbot, E. A., L. F. Marin-Urias, R. Alami, et T. Simeon. 2007. « A Human Aware Mobile Robot Motion Planner ». *IEEE Transactions on Robotics* 23 (5): 874-83. https://doi.org/10.1109/TRO.2007.904911.

Suita, K., Y. Yamada, N. Tsuchida, K. Imai, H. Ikeda, et N. Sugimoto. 1995. « A failure-to-safety "Kyozon" system with simple contact detection and stop capabilities for safe human-autonomous robot coexistence ». In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 3:3089-96 vol.3. https://doi.org/10.1109/ROBOT.1995.525724.

Svenstrup, M., S. Tranberg, H. J. Andersen, et T. Bak. 2009. « Pose estimation and adaptive robot behaviour for human-robot interaction ». In *2009 IEEE International Conference on Robotics and Automation*, 3571-76. https://doi.org/10.1109/ROBOT.2009.5152690.

Svestka, Petr, et Mark H. Overmars. 1997. « Motion Planning for Carlike Robots Using a Probabilistic Learning Approach ». *I. J. Robotics Res.* 16 (2): 119–143. https://doi.org/10.1177/027836499701600201.

Tonietti, G., R. Schiavi, et A. Bicchi. 2005. « Design and Control of a Variable Stiffness Actuator for Safe and Fast Physical Human/Robot Interaction ». In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 526-31. https://doi.org/10.1109/ROBOT.2005.1570172.

Ulmen, J., et M. Cutkosky. 2010. « A robust, low-cost and low-noise artificial skin for human-friendly robots ». In *2010 IEEE International Conference on Robotics and Automation*, 4836-41. https://doi.org/10.1109/ROBOT.2010.5509295.

Vanderborght, B., A. Albu-Schaeffer, A. Bicchi, E. Burdet, D. G. Caldwell, R. Carloni, M. Catalano, et al. 2013. « Variable impedance actuators: A review ». *Robotics and Autonomous Systems* 61 (12): 1601-14. https://doi.org/10.1016/j.robot.2013.06.009.

Vasquez, Dizan, Procópio Stein, Jorge Rios-Martinez, Arturo Escobedo, Anne Spalanzani, et Christian Laugier. 2012. « Human Aware Navigation for Assistive

Robotics ». In *ISER - 13th International Symposium on Experimental Robotics - 2012*. Québec, Canada. https://hal.inria.fr/hal-00743628.

Victorino, Alessandro Corrêa, Patrick Rives, et Jean-Jacques Borrelly. 2003. « Safe navigation for indoor mobile robots. Part I: A sensor-based navigation framework ». *International Journal of Robotics Research* 22 (12).

Villani, Valeria, Fabio Pini, Francesco Leali, et Cristian Secchi. 2018. « Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications ». *Mechatronics* 55 (novembre): 248-66. https://doi.org/10.1016/j.mechatronics.2018.02.009.

Villumsen, Sigurd Lazic, et Morten Kristiansen. 2017. « PRM Based Motion Planning for Sequencing of Remote Laser Processing Tasks ». *Procedia Manufacturing* 11: 300-310. https://doi.org/10.1016/j.promfg.2017.07.109.

Wang, Jun, et Ying Tan. 2013. « Efficient Euclidean distance transform algorithm of binary images in arbitrary dimensions ». *Pattern Recognition* 46 (1): 230–242.

Wei, Kun, et Bing-yin Ren. 2018. « A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm ». *Sensors* 18: 571. https://doi.org/10.3390/s18020571.

Wilmarth, Steven A., Nancy M. Amato, et Peter F. Stiller. 1999. « MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space ». In *1999 IEEE International Conference on Robotics and Automation, Marriott Hotel, Renaissance Center, Detroit, Michigan, May 10-15, 1999, Proceedings*, 1024–1031.

Wurm, Kai M., Armin Hornung, Maren Bennewitz, Cyrill Stachniss, et Wolfram Burgard. 2010. « OctoMap : A Probabilistic , Flexible , and Compact 3 D Map Representation for Robotic Systems ». In *Proceedings of the ICRAWorkshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation (IROS)*.

Xavier, J., M. Pacheco, D. Castro, A. Ruano, et U. Nunes. 2005. « Fast Line, Arc/Circle and Leg Detection from Laser Scan Data in a Player Driver ». In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005*, 3930-35. https://doi.org/10.1109/ROBOT.2005.1570721.

Yamada, Y., T. Morizono, Y. Umetani, et H. Takahashi. 2005. « Highly soft viscoelastic robot skin with a contact object-location-sensing capability ». *IEEE Transactions on Industrial Electronics* 52 (4): 960-68. https://doi.org/10.1109/TIE.2005.851654.

Yan, F., Y. Zhuang, et J. Xiao. 2012. « 3D PRM based real-time path planning for UAV in complex environment ». In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1135-40. https://doi.org/10.1109/ROBIO.2012.6491122.

Yang, Yuandong, et O. Brock. 2004. « Adapting the sampling distribution in PRM planners based on an approximated medial axis ». In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 5:4405-4410 Vol.5.

Yeh, H., S. Thomas, D. Eppstein, et N. M. Amato. 2012. « UOBPRM: A uniformly distributed obstacle-based PRM ». In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2655-62. https://doi.org/10.1109/IROS.2012.6385875.

Yeh, H. Y. C., J. Denny, A. Lindsey, S. Thomas, et N. M. Amato. 2014. « UMAPRM: Uniformly sampling the medial axis ». In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 5798-5803.

Yong Yu, et K. Gupta. 1999. « Sensor-based roadmaps for motion planning for articulated robots in unknown environments: some experiments with an eye-in-hand system ». In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, 3:1707-14 vol.3. https://doi.org/10.1109/IROS.1999.811724.

Yoshimi, T., M. Nishiyama, T. Sonoura, H. Nakamoto, S. Tokura, H. Sato, F. Ozaki, N. Matsuhira, et H. Mizoguchi. 2006. « Development of a Person Following Robot with Vision Based Target Detection ». In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5286-91. https://doi.org/10.1109/IROS.2006.282029.

Zhang, H., Y. Wang, J. Zheng, et J. Yu. 2018. « Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments ». *IEEE Access* 6: 53296-306. https://doi.org/10.1109/ACCESS.2018.2871222.