Planification interactive de mouvement avec contact

Nassime Michel Blin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *12 Décembre 2017* **par :**

Nassime Michel BLIN

Planification interactive de mouvement avec contact

JURY

René Zapata	Professeur d'Université	Président du Jury
Fouad Bennis	Professeur d'Université	Rapporteur
Véronique Perdereau	Professeur d'Université	Rapporteur
Philippe Fillatreau	Maitre de Conférences	Membre du Jury
Jean-Yves Fourquet	Professeur d'Université	Membre du Jury
Michel Taïx	Maitre de Conférences	Membre du Jury

École doctorale et spécialité :
EDSYS : Robotique 4200046

Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes(LAAS-CNRS), Laboratoire Génie de Production (LGP)

Directeur(s) de Thèse :
Jean-Yves Fourquet, *Michel Taïx* et *Philippe Fillatreau*

Rapporteurs :
Fouad Bennis et *Véronique Perdereau*

Remerciements

Je souhaite remercier mes directeurs de thèse, Jean-Yves Fourquet, Michel Taïx et Philippe Fillatreau pour leurs qualités scientifiques et leur implication dans mon encadrement qui m'a beaucoup apporté. Je voudrais également remercier Véronique Perdereau et Fouad Bennis qui m'ont fait l'honneur d'être mes rapporteurs ainsi que René Zapata pour avoir assisté à ma soutenance en tant qu'examinateur.

Je voudrais ensuite remercier Justin Carpentier, Bruno Sarkis, Joseph Mirabel et Mathieu Geisert qui m'ont aidé à effectuer ce travail et sans qui je n'aurai pas eu les mêmes résultats.

Je souhaite aussi remercier les professeurs et chercheurs qui ont soutenu mon dossier et sans qui je n'aurais pas pu faire cette thèse : Olivier Stasse, Abdelhafid Elouardi et Emmanuelle Encrenaz.

À présent je souhaite remercier l'ensemble des personnes qui avec moi ont fait l'homme que je suis devenu aujourd'hui. En premier lieu, mes parents Louis et Nafissa Blin pour m'avoir donné la vie et une éducation sans pareil. Ma soeur et mes frères : Lina, Samy et Dany Blin dont je suis fier et que j'aime. Je voudrais ensuite remercier mes premiers amis avec qui j'ai fait l'école de la vie : Pierre-Olivier Haye, Antoine Dervieux, Pierre Vergne et Cédric Novais, vite rejoints par Charles Papelier, Nicolas Monin, Mikhail Stadnitchouk et Alice Mayer. Mon professeur d'histoire-géographie du lycée, M. Jobert, m'a donné le goût de sa discipline sans laquelle je serais bien différent. Je pense ensuite à Arnaud Iss et Benjamin Pillet amis et musiciens de mon premier groupe de musique et Pierre Berget et Guillaume Duchemin qui m'ont aidé à grandir chacun à leur façon.

Nombreuses sont les personnes que j'ai rencontrées au cours de cette thèse sans lesquelles ma vie eût été bien morne et que je voudrais remercier. Dans l'ordre et le désordre et que je n'ai pas déjà cité sont les gepettistes Mylène Campana, Nirmal Giftsun, Alexis Mifsud, les membres du groupe de musique MARACLAAS Maximillien Naveau, Mehdi Benallegue, Naoko Abe, mais aussi Andrea del Prete, Kevin Giraud-Esclasse, Galo Maldonado, Olivier Roussel, Andreas Orthey, Christian Vassallo et plus tard, Guilhem Saurel, Steve Tonneau, Pierre Fernbach, Rohan Budhiraja, Florent Forget, Thomas Flayols, François Bailly, Dinesh Atchuthan et Céline Pieters. Ceux de Tarbes qui sont Simon Cailhol, Corentin Delebarre, Karima Berkoune, Floran Barelli, Farouk Frigui, Guillaume Viné, Matthias Barus, Maël Thévenot, Camille Gillet et Pauline Mauvy. Mais aussi Hélène Thirion et Catherine Tessier pour leur travail et parce qu'elles m'ont permis de me déplacer à Gênes.

I also wish to thank the people I met when I was working in Genova, starting with Arash Ajoudani who accepted me in his research group. I wish to thank Luka Peternel, Nikol Guljemovic and Paul Gay, my friends from the laboratory. I met Dimitar Todorov as a bulgarian roomate but I soon became one of the bulgarian folks from Genova along with Rosalie Malcheva, Maggi Trapova, Vladimir Anastov, Volen Sevov and more. They all became friends and I wish to tell them благодаря.

Gli italiani che incontrato nella città della perdizione mi hanno aperto una nuova di-

menzione. Ho scoperto una nuova lingua e cultura ché mai dimenticaro, grazie a Virginia Grozio e le altre due sorelle.

Je voudrais remercier les personnes que j'ai rencontrées à la fin de mon doctorat et qui m'ont soutenu pendant cette période difficile, mes colocataires Petra Mullerová, Cuang Nguyen et la quasi-colocataire Kvetka Szolárová mais aussi Andrew Simpson qui m'a aidé d'un point de vue personnel.

Last, I wish to thank Amy LaViers who accepted me as a post-doctoral fellow in her laboratory at UIUC.

يا ماما شكرا لك كمان مرة لأنك ساعدتيني كل يوم لما كنت سعيد ولما كنت حزين. بحبك يا ماما.

Merci !

Une valse à mille temps
Offre seule aux amants
Trois cent trente-trois fois l'temps
De bâtir un roman

Jacques Brel

Résumé

La conception de nouveaux produits industriels nécessite le développement de prototypes avant leur déploiement grand public. Afin d'accélérer cette phase et de réduire les coûts qui en découlent, une solution intéressante consiste a utiliser des prototypes virtuels le plus longtemps possible en particulier dans la phase de conception. Certaines des étapes de la conception consistent à effectuer des opérations d'assemblage ou de désassemblage. Ces opérations peuvent être effectuées manuellement ou automatiquement à l'aide d'un algorithme de planification de mouvement. La planification de mouvement est une méthode permettant à un ordinateur de simuler le déplacement d'un objet d'un point de départ à un point d'arrivée tout en évitant les obstacles.

Le travail de recherche de cette thèse apporte des solutions afin d'améliorer l'interaction entre un humain et un algorithme de planification de mouvement pendant l'exploration de l'espace libre. Le temps de recherche est partagé entre l'humain et la machine selon un paramètre de partage d'autorité permettant de déterminer le pourcentage d'allocation du temps à l'une ou l'autre entité. L'utilisation simultanée de ces deux entités permet d'accélérer grandement la vitesse d'exploration par rapport à la vitesse d'un humain seul ou d'un algorithme seul.

Ces travaux apportent ensuite une nouvelle méthode de planification de mouvement avec contact permettant de générer des trajectoires à la surface des obstacles au lieu de les générer uniquement dans l'espace libre. La planification au contact permet d'effectuer des opérations spécifiques telles que le glissement ou l'insertion utiles pour la résolution de problèmes de planification dans des environnements encombrés.

Enfin, détecter les intentions d'un utilisateur lorsqu'il interagit avec une machine permet de lui fournir des ordres efficacement et intuitivement. Dans le cadre de la planification interactive au contact, un algorithme de détection d'intention est proposé. Ce dernier s'appuie sur l'utilisation d'un robot haptique permettant à un opérateur de ressentir les obstacles virtuels lors de la manipulation d'un objet virtuel dans un environnement de réalité virtuelle. L'algorithme interactif s'adapte en temps réel aux actions de l'opérateur pour une exploration pertinente de la surface des obstacles.

Ces travaux ont été menés en partie au laboratoire toulousain LAAS au sein de l'équipe Gepetto et en partie dans le laboratoire LGP de l'ENIT au sein de l'équipe DIDS. Nous remercions la région Midi-Pyrénées pour avoir financé ces recherches.

Abstract

Designing new industrial products requires to develop prototypes prior to their launch phase. An interesting solution to speedup the development phase and reduce its costs is to use virtual prototypes as long as possible. Some steps of the development consist in assembly or disassembly operations. These operations can be done manually or automatically using a motion planning algorithm. Motion planning is a method allowing a computer to simulate the motion of an object from a start point to a goal point while avoiding obstacles.

The following research work brings solutions for the interaction between a human operator and a motion planning algorithm of virtual objects for the exploration of free space. Research time is split between the human and the machine according to an authority sharing parameter determining the percentage of time allocated to one or the other entity. The simultaneous use of a human and a machine greatly speedup the exploration in comparison to the time needed by any of the former two alone.

This work then presents a new interactive motion planner with contact. This method permits to generate trajectories at the surface of obstacles instead of free space trajectories. Contact motion planning allows specific operations such as sliding or insertion. This greatly diminishes the solving time of motion planning problems in cluttered environments.

Detecting the intentions of a user when he interacts with a machine is a good way to convey orders efficiently and intuitively. An algorithm for interactive contact planning with intention detection techniques is proposed. This algorithm uses a haptic robot allowing a user to feel virtual obstacles when manipulating a virtual object in a virtual reality environment. The interactive algorithm adapts to the actions of the user in real time for a pertinent exploration of the surfaces of obstacles.

This work has been done partly in LAAS-CNRS laboratory in Toulouse in Gepetto team and partly in LGP-ENIT laboratory in Tarbes in DIDS team. We wish to thank the Midi-Pyrénées region for funding this research.

Table des matières

Chapitre 1

Introduction et contexte

Le développement de nouveaux produits industriels est de plus en plus contraint. Un nouveau besoin apparaît alors, celui d'accélérer toutes les étapes de la conception d'un produit et de réduire leurs coûts. Pour cela sont utilisées des maquettes virtuelles qui sont des prototypes numériques à la place de prototypes physiques jusque tard dans le cycle de développement d'un produit.

L'assemblage, le désassemblage et la maintenance de maquettes virtuelles peut se faire en utilisant des méthodes de planification de mouvement. Ces méthodes consistent à trouver pour un objet un chemin sans collisions entre un point de départ et un point d'arrivée. Un ordinateur est capable de résoudre un tel problème de planification mais ses performances se dégradent considérablement dans des environnements encombrés.

Il existe de nombreuses méthodes pour améliorer les performances d'un planificateur de mouvement et l'une d'elles consiste à utiliser conjointement un planificateur automatique et un opérateur humain. La machine a l'avantage d'être rapide et l'homme celui d'être intelligent. En combinant les capacités de l'un et de l'autre, il est possible d'augmenter la vitesse de résolution d'un problème.

Par définition, l'assemblage implique un contact entre les pièces assemblées. De plus le contact d'un objet avec son environnement peut permettre des tâches spécifiques utiles telles que l'assemblage ou l'insertion. Il apparaît donc utile de se munir de planificateurs de mouvements capables de planifier des trajectoires à la surface d'obstacles au lieu de rester dans l'espace libre. Un tel planificateur apporterait des performances supérieures tout en permettant des opérations de glissement, d'insertion.

L'utilisation conjointe d'un humain et d'une machine pour planifier un mouvement nécessite une interaction entre ces deux entités. Quand existe une telle interaction, il est souhaitable qu'elle soit la plus efficace, ergonomique et naturelle possible. Il faut pour cela utiliser des méthodes de détection d'intention qui permette à la machine d'aider l'humain efficacement.

Dans cette thèse nous répondons à ces trois problématiques que sont : l'interaction, le

contact et la détection d'intention qui seront développées dans des chapitres dédiés. Comme l'indique son intitulé, "Planification interactive de mouvement avec contact", elle vise un apport aux techniques et outils de simulation interactive dans le domaine de l'assemblage, du désassemblage et de la maintenance de produits industriels.

Nos travaux s'inscrivent dans un cadre collaboratif entre le LAAS entité du CNRS à Toulouse et le laboratoire génie de production, le LGP au sein de l'école d'ingénieurs ENIT à Tarbes. Le LAAS est notamment spécialisé en planification de mouvement et le LGP est spécialisé dans le génie mécanique et disposant d'une plate-forme de réalité virtuelle.

1.1 Contexte industriel

Le développement de nouveaux produits implique pour les entreprises un processus très contraignant. Il leur faut s'adapter à des impératifs économiques, de qualité mais aussi réglementaires qui tendent à devenir de plus en plus difficiles.

Nous présentons dans cette sous-section les méthodes et outils que les industries utilisent pour le développement de nouveaux produits mais aussi pendant leur maintenance ou la gestion de leur fin de vie.

1.1.1 Processus industriels et outils PLM

Pour faire face à leurs contraintes, les industriels mettent en œuvre des méthodes comme le PLM (Product Lifecycle Management) pour la conception de leurs produits. Ces méthodes permettent de considérer l'ensemble du cycle de vie d'un produit dès sa conception. Cela permet d'une part d'optimiser son cycle de vie et d'autre part de faciliter sa conception, sa fabrication, sa maintenance et la gestion de sa fin de vie.

Les outils PLM présentent de nombreux bénéfices que l'on peut classer en trois catégories. La première catégorie apporte des effets opérationnels pour les individus impliqués dans le processus de conception. Les outils PLM leur facilitent la tâche par une meilleure compréhension du produit et une réduction des erreurs de conception notamment. En second lieu, ces outils bénéficient à l'organisation des processus par la réduction des difficultés liées à l'échange de données, à une coordination plus efficace et à une gestion plus rigoureuse . Enfin, ces outils apportent des effets stratégiques sur les performances industrielles grâce à l'augmentation du degré d'innovation et à la réduction des coûts et des temps de production.

Le développement de produits part d'un besoin exprimé par le client. Des spécifications de produits sont déterminées conjointement par le client et le fournisseur. Alors démarre la démarche de conception. Elle s'appuie traditionnellement sur le cycle en V dans l'industrie même si les méthodes agiles prennent de plus en plus d'importance et s'utilisent en son sein.

Le cycle en V est un modèle de développement d'un projet en étapes successives à partir d'un besoin exprimé par le client. Ce besoin est traduit en spécifications puis en modules et

en fonctions élémentaires. La première phase de ce cycle est la conception du produit dans la phase descendante de la figure 1.1. La validation du produit s'effectue dans la deuxième phase, ascendante. Chaque étape de validation correspond à une étape de conception.

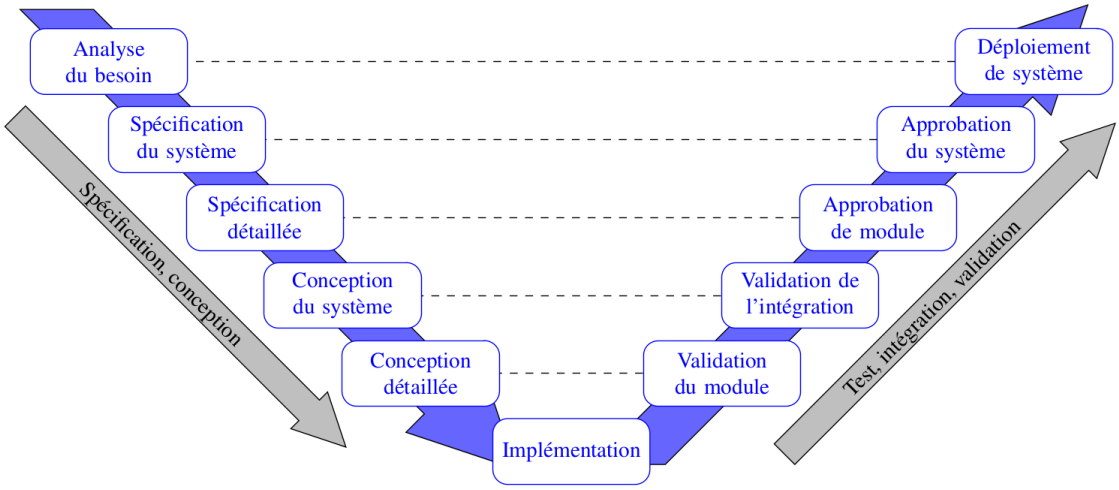


FIGURE 1.1 – Méthode de gestion dite cycle en V [12].

Les produits industriels sont de plus en plus complexes et impliquent un nombre toujours croissant de procédures de validation et donc de prototypes sur lesquels réaliser les tests. Cependant, un prototype est générateur de coûts et de délais. La solution est alors d'utiliser un prototype virtuel à la place d'un prototype physique.

Les entreprises effectuent donc l'ensemble des phases du développement du produit (conception, implémentation, test, évaluation) en privilégiant l'utilisation des maquettes virtuelles.

1.1.2 Prototypage virtuel

S'il était nécessaire par le passé de fabriquer un prototype physique, ce n'est plus le cas aujourd'hui grâce à l'informatique qui permet de créer et d'utiliser des maquettes numériques.

Les outils PLM utilisés avec des prototypes virtuels permettent de diminuer les coûts liés aux prototypes réels. Les prototypes virtuels permettent aussi de diminuer le temps de prototypage car une maquette numérique est bien plus rapide à créer. Les prototypes virtuels permettent dès la phase de conception de valider les actions, par exemple l'assemblage et la maintenance, associées au cycle de vie du produit avant de réaliser les prototypes physiques, faits plus tard et en moins grand nombre.

L'utilisation de prototypes virtuels pour la simulation de comportements physiques s'est donc répandue dans de nombreux domaines. Des logiciels spécialisés sont utilisés pour

l'analyse du comportement mécanique, thermique, magnétique, électrique ou de dynamique des fluides. Dans le cadre de la planification de mouvement, un logiciel simule la collision entre objets ce qui permet à un algorithme de planification de simuler un assemblage. En effet, chaque pièce d'un prototype doit être assemblée afin de vérifier qu'elle s'insère correctement, qu'elle a les bonnes dimensions, qu'elle fonctionne comme prévu. Ces pièces sont donc dessinées numériquement et assemblées manuellement dans la maquette numérique ou bien à l'aide d'un logiciel de planification de mouvement. La planification de mouvement consiste à trouver pour un objet, un chemin d'un point de départ à un point d'arrivée. Une définition plus précise sera donnée chapitre 2.

La réalité virtuelle (RV) est un des outils informatiques utilisés lorsque les tests réalisés sur maquette numérique nécessitent l'intervention d'un humain (assemblage, maintenance, ...). La RV permet l'immersion et l'interaction d'un acteur humain avec un environnement virtuel, numérique. Cette technique sera utilisée au cours de cette thèse et sera définie dans le détail au chapitre 5.

1.2 Contexte scientifique et contributions

Le travail ici présenté s'inscrit dans un contexte scientifique déjà riche de nombreuses contributions. Nous présentons ici les enjeux auxquels nous avons fait face, les réponses que nous y avons apporté puis les publications dans lesquelles nous apportons ces réponses.

1.2.1 Verrous

La planification interactive de mouvement consiste à faire collaborer un algorithme de planification de mouvement avec un opérateur humain au travers d'un périphérique d'interaction. Une définition plus précise est donnée au chapitre 3. Ce champ de recherche n'est pas nouveau ayant fait l'objet de nombreux travaux au LAAS, au LGP et dans de nombreux laboratoires étrangers. Nous proposons dans cette thèse une nouvelle façon de partager le temps d'exploration entre l'homme et la machine. Nous proposons une solution interactive originale pour l'exploration d'un espace dans le cadre de la planification de mouvement.

Un verrou scientifique important est que la plupart des travaux précédents ne prennent pas en compte le contact qui est nécessaire dans de nombreuses tâches. La planification de mouvement au contact sera définie précisément au chapitre 4. Il existe déjà des méthodes de planification au contact. Elles sont soit locales soit basées sur de la rétraction. À notre connaissance, il n'existe pas de méthodes globales et non basées rétraction. Cet important verrou sera levé par la proposition d'un planificateur de mouvement avec contact.

Enfin, la possibilité de prendre en compte les intentions de l'utilisateur est un domaine déjà bien exploré dont la définition précise sera donnée chapitre 5. Dans le cadre de la planification interactive de mouvement, il n'existait à notre connaissance qu'une seule contribution avec détection d'intention et très peu dans le cadre de la planification de mou-

vement avec contact. Nous proposons un algorithme utilisant une méthode de détection d'intention dans le cadre de la planification au contact.

1.2.2 Apports scientifiques et techniques

Cette thèse apporte plusieurs contributions scientifiques :

— Une méthode de partage du temps de planification de mouvement entre l'humain et la machine

— Une méthode permettant de la planification de mouvement au contact d'une surface choisie par l'humain

— Une méthode capable de prendre en compte l'intention de l'utilisateur humain pendant la planification de mouvement au contact

Elle apporte aussi plusieurs contributions techniques :

— Une architecture permettant de contrôler un objet dans l'espace à l'aide d'une souris 6D ou d'un bras haptique

— Une architecture permettant le travail en parallèle d'un environnement de réalité virtuelle et d'un planificateur de mouvement

1.2.3 Publications associées

Cette thèse a mené à trois publications dans des conférences internationales de référence en robotique :

Interactive Motion Planning with Contact. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016). Nassime Blin, Michel Taïx, Philippe Fillatreau, Jean-Yves Fourquet.

Tuning Interaction in Motion Planning with Contact. 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2016). Nassime Blin, Michel Taïx, Philippe Fillatreau, Jean-Yves Fourquet.

H-RRT-C : Haptic Motion Planning with Contact. 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2017). Nassime Blin, Michel Taïx, Philippe Fillatreau, Jean-Yves Fourquet.

1.3 Organisation du document

Bien qu'un important travail de développement et d'intégration ait été effectué au cours de cette thèse, il ne sera évoqué ici que succinctement. Ce mémoire est focalisé sur les aspects scientifiques de ce travail et son positionnement par rapport aux autres travaux.

Le chapitre 2 évoquera brièvement les fondements et concepts de la planification de mouvement non interactive. Le chapitre 3 se focalisera sur les méthodes de planification interactive de mouvement en espace libre. Le chapitre 4 présentera nos travaux en planification de mouvement au contact. Le chapitre 5 décrira nos contributions pour la détection d'intention dans le cadre de la planification interactive de mouvement. Suivra le chapitre 6 consacré à l'implémentation de nos algorithmes et aux expérimentations menées tout au long de la thèse. Enfin, nous rassemblerons nos conclusions et évoquerons des pistes de recherche futures au dernier chapitre.

Chapitre 2

Planification de mouvement automatique

Ce chapitre présente dans un premier temps les concepts élémentaires de la planification de mouvement dans un cadre purement automatique. Cela veut dire qu'aucun humain n'entre en jeu pendant la résolution du problème. Dans un second temps, ce chapitre présente les différentes méthodes de planification existantes à ce jour en mettant l'accent sur les méthodes dites probabilistes qui sont les méthodes utilisées tout au long de cette thèse.

2.1 Définition de la planification de mouvement

Cette section définit ce qu'est un problème de planification de mouvement puis des concepts indispensables pour comprendre la suite de ce mémoire : la définition d'un robot, de l'espace des configurations, de la planification de mouvement et enfin les principales notations utilisées en planification de mouvement.

2.1.1 Définition d'un robot

Dans cette thèse, un robot est un objet rigide, appelé OBJ, que nous souhaitons déplacer dans un espace. D'une manière générale, l'objet à déplacer, le robot, n'est pas nécessairement un objet rigide mais peut être constitué de plusieurs corps articulés. Les articulations peuvent être des rotations ou des translations. Cet ensemble de corps reliés par des articulations est appelé une chaîne cinématique. Un robot possède un certain nombre de degrés de liberté (ddl) fonction du nombre d'articulations qu'il possède et du nombre de paramètres dont il dispose pour se déplacer dans l'espace. Ce total détermine son nombre de ddl.

Dans la figure 2.1, le premier robot est fixe et ne possède que trois ddl, le second est un bras industriel fixe possédant six ddl. Le dernier est le robot humanoïde HRP-2 possédant 30 articulations. Ce dernier peut se déplacer librement dans l'espace, il possède donc 6 ddl

supplémentaires pour se positionner dans l'espace car un objet flottant est défini par trois positions et trois orientations, voir sous section suivante. HRP-2 possède donc un total de 36 ddl.

FIGURE 2.1 – Robots à différents degrés de liberté [42].

2.1.2 L'espace des configurations

Une configuration est un vecteur q dont chaque composante décrit la valeur de l'un des paramètres du robot, chaque paramètre étant borné. Dans le cas d'un objet rigide non articulé et flottant dans l'espace tridimensionnel, sa configuration possède six paramètres : trois positions x, y, z dans l'espace borné par les limites de l'espace de travail W mais également trois orientations θ, φ, ψ bornées car nécessairement inclues entre 0 et 2π [39, 68, 72]. Pour un objet flottant, la configuration est définie par un vecteur de dimension 6.

L'espace des configurations $\mathcal{C}$ représente l'ensemble de toutes les configurations du robot. Cela permet de traduire le problème du mouvement d'un robot à n dimensions dans $\mathbb{R}^2$ ou dans $\mathbb{R}^3$ en celui d'un point dans $\mathcal{C} \subset \mathbb{R}^n$. Dans le cas d'un corps flottant, $\mathcal{C} \subset SE(3)$. La dimension de la variété $\mathcal{C}$ est égale à la dimension du robot c'est à dire au nombre de paramètres indépendants décrivant entièrement la configuration du robot. Cet espace contient l'ensemble des obstacles et est borné.

Les obstacles dans le monde réel induisent des obstacles dans $\mathcal{C}$ et sont notés $\mathcal{C}_{obst}$ avec $\mathcal{C}_{obst} \subset \mathcal{C}$. Son complémentaire est l'espace libre, noté $\mathcal{C}_{libre}$. Sont exclues de $\mathcal{C}_{libre}$ les configurations au contact des obstacles $\mathcal{C}_{contact}$ ce qui fait que $\mathcal{C}_{libre}$ est un ouvert de $\mathcal{C}$.

2.1.3 La planification de mouvement

La planification de mouvement est historiquement définie par le problème du déménageur de piano [67] voulant déplacer ce dernier d'un endroit à un autre au milieu d'une pièce contenant d'autres objets. En effet, le déplacement d'un piano au sein d'une pièce se heurte à un problème : vu la forme du piano, existe-t-il une trajectoire praticable entre

ces derniers ? Pour résoudre ce problème de déménagement, il faut qu'il existe au moins un chemin valide que le déménageur doit effectuer entre une configuration de départ et une configuration d'arrivée. Un déplacement est dit valide s'il est intégralement situé dans un espace libre d'obstacles. La planification de mouvement consiste donc à trouver pour un robot une trajectoire valide. Cependant, le problème sera très différent selon le type d'objet à déplacer, un piano, une aiguille, un bras articulé, etc... D'une manière générale, le problème de la planification est de déplacer un objet. Cela peut être le cas d'un solide articulé ou non.

FIGURE 2.2 – Déménageur de piano [36].

La planification est utile en robotique mobile [78] et pour la conduite de voitures [41]. Elle sert aussi pour résoudre des problèmes de manipulation d'un objet [64, 65] ou encore pour la modélisation moléculaire [1].

La résolution d'un problème de planification de mouvement à l'aide d'une méthode indépendante du type de robot a été proposée pour la première fois par Lozano-Perez en 1983 [49]. Il s'agit de présenter le problème non pas dans l'espace de travail W mais dans l'espace des configurations $\mathcal{C}$.

La notion d'espace des configurations permet de transformer le problème de planification d'un robot dans l'espace de travail W en un problème de trajectoire d'un point dans l'espace des configurations [38].

Le problème de planification de mouvement est alors reformulé de la sorte : sachant $\mathcal{C}$, $\mathcal{C}_{obst}$, un configuration initiale q_{init} et une configuration finale q_{fin}, toutes deux appartenant à $\mathcal{C}_{libre}$, existe-il un chemin valide reliant q_{init} à q_{fin} ?

La complexité du problème de planification de mouvement croit exponentiellement avec

la dimension de $\mathcal{C}$. Dans ce cas le problème devient difficilement traitable.

2.1.4 Notations pour la planification de mouvement

Ici sont résumées les notations utilisées en planification de mouvement et tout au long de ce mémoire :

- L'espace de travail W ou environnement est l'espace dans lequel est décrite la géométrie du robot et des obstacles.
- Le nombre de degrés de liberté (ddl) est le nombre de paramètres indépendants nécessaires pour décrire la configuration d'un robot.
- Une configuration q est un vecteur dont chaque composante décrit la valeur d'un des degrés de liberté du robot.
- L'espace des configurations noté $\mathcal{C}$ est l'ensemble des configurations possibles pour un robot, le déplacement d'un robot dans W correspond au déplacement d'un point dans $\mathcal{C}$.
- L'espace libre $\mathcal{C}_{libre} \subset \mathcal{C}$ est l'ensemble des configurations de $\mathcal{C}$ qui ne sont pas en collision avec un obstacle.
- L'espace des obstacles $\mathcal{C}_{obst} \subset \mathcal{C}$ est l'ensemble des configurations de $\mathcal{C}$ qui sont occupées par des obstacles avec $\mathcal{C}_{libre} \cap \mathcal{C}_{obst} = \{\}$.
- L'espace au contact $\mathcal{C}_{contact}$ est l'ensemble des configurations où un point au moins du robot est en contact avec un point au moins d'un obstacle mais sans être en collision. Il s'agit de la frontière de $\mathcal{C}_{obst}$.
- Les configurations q_{init} et q_{fin} sont respectivement la configuration initiale et la configuration finale de la trajectoire à trouver.
- Un chemin est une fonction chemin : $[0, 1] \rightarrow \mathcal{C}$ qui relie deux configurations de $\mathcal{C}$

2.2 Méthodes de planification de mouvement

Nous présentons ici les différentes méthodes de planification de mouvement. Nous commençons par un très bref survol des méthodes déterministes [43, 17] car ces méthodes ne sont pas l'objet de nos recherches. Nous présentons ensuite le principe des méthodes probabilistes puis nous présentons les deux principales méthodes de cette catégorie, le PRM puis le RRT. Tous nos travaux sont basés sur la méthode RRT. Enfin, nous résumons les notations utilisées en planification de mouvement probabiliste.

2.2.1 Méthodes déterministes

Il existe plusieurs méthodes de planification de mouvement utilisant le formalisme décrit ci-dessus. Parmi les premières méthodes proposées, on trouve les méthodes dites déterministes. Ces méthodes donneront toujours le même résultat pour un problème donné.

Ces méthodes peuvent être exactes et sont alors dites complètes. Si elles sont complètes, elles trouvent une solution s'il en existe une sinon elles indiquent qu'il n'en existe pas.

Les méthodes déterministes dites approchées ne sont pas complètes mais si une solution existe elle sera trouvée en temps infini. On dit alors qu'elles sont complètes en résolution.

Les méthodes déterministes ne seront pas étudiées dans ce mémoire car elles sont trop coûteuses en temps pour des problèmes ayant de nombreux ddl. Aussi nous invitons le lecteur à se référer à l'ouvrage [43] pour plus de précisions. Elles ont une complexité algorithmique qui croît exponentiellement avec le nombre de ddl [15]. Afin de résoudre ce problème de complexité souvent rencontré en pratique, une nouvelle catégorie de méthodes ont étés introduites dans les années 1990, il s'agit des méthodes probabilistes.

2.2.2 Méthodes probabilistes

Les méthodes probabilistes perdent la propriété de complétude des méthodes déterministes mais sont très efficaces en pratique. Elles garantissent cependant une complétude en probabilité c'est à dire que la probabilité de trouver une solution, s'il en existe, tend vers 1 lorsque le temps tend vers l'infini. Autrement dit, elles garantissent de trouver une solution lorsque le temps tend vers l'infini.

Ces algorithmes se basent sur une fonction d'échantillonnage des configurations, alors appelé échantillon. Une configuration échantillonnée aléatoire q_{rand} appartient à $\mathcal{C}$ et est ajoutée à un graphe ou arbre selon des heuristiques propres à chaque algorithme, on appelle *roadmap* cette structure de données représentant l'espace des configurations. In fine, les échantillons reliés entre eux forment une trajectoire valide entre q_{init} et q_{fin}. Nous détaillons ici les deux principales méthodes que sont PRM et RRT. Ces deux méthodes se sont imposées au fil du temps comme méthodes de base de la planification de mouvement probabiliste. L'intérêt de ce type de méthode est qu'elles ne calculent pas C_{obst}, tous les tests de collision étant réalisés dans W. La façon dont les méthodes probabilistes explorent l'environnement dépend d'abord de la méthode d'échantillonnage aléatoire. L'échantillonnage le plus répandu est uniforme sur l'espace des configurations. Cela donne une probabilité égale pour chaque point de C d'être choisi. Le succès de ces méthodes est largement dépendant de la manière dont se fait l'échantillonnage et de la fonction qui permet de connecter ces échantillons.

2.2.3 PRM

La première de ces deux méthodes est appelée PRM (Probabilistic Roadmap Method). Elle a été introduite par Kavraki [38]. L'idée de cette méthode est d'échantillonner l'espace des configurations $\mathcal{C}$ pour trouver suffisamment de configurations sans collisions pour permettre de représenter l'espace des configurations libres : même nombre de composantes connexes, couverture de l'espace libre. L'algorithme détermine si les configurations échantillonnées sont en collision ou non grâce à un détecteur de collision. Celles qui sont dans

l'espace libre sont gardées et l'algorithme tente de les relier par une arête au graphe contenant les autres nœuds. Le lien se fait à l'aide d'une méthode locale permettant de relier un nœud à un autre le long d'un espace libre. Le graphe comporte initialement deux nœuds, la configuration initiale et la configuration finale. Lorsque l'on souhaite trouver le chemin entre deux nœuds du graphe, celui-ci est parcouru à l'aide d'un algorithme de type A*, voir figure 2.3 où est indiqué S la configuration de départ et G la configuration d'arrivée.

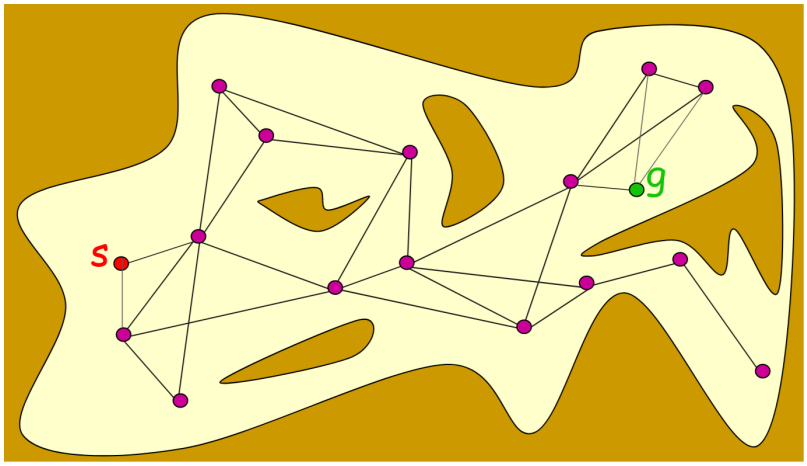


FIGURE 2.3 – Graphe PRM.

Cette méthode a l'avantage de pouvoir conserver le graphe entre deux requêtes de planification. Cela permet d'effectuer plusieurs planifications successives très rapidement sur le même environnement en gardant le même graphe mais avec des configurations initiales et finales différentes. Son désavantage est celui de toutes les méthodes de planification de mouvement probabilistes, il faut que le graphe soit suffisamment représentatif de l'espace libre C_{libre} avant qu'il ne soit utilisable pour des requêtes de planification de mouvement. Cette étape est donc généralement effectuée hors ligne car la génération d'une *roadmap* est coûteuse en temps ; le graphe de l'environnement obtenu est conservé pour des requêtes effectuées en ligne ultérieurement. Il existe de nombreuses variantes de cet algorithme [63, 33, 10, 18].

2.2.4 RRT

L'algorithme RRT, pour Rapidly-exploring Random Trees introduit par LaValle [45], est la méthode de base utilisée tout au long de ce mémoire. Elle fait partie de la famille des méthodes par diffusion de *roadmap*. Elle permet de construire une *roadmap* rapidement, à usage unique, dont le but n'est pas de couvrir l'intégralité de l'environnement mais simplement de répondre à une unique requête de planification. Au lieu de construire un graphe, le RRT construit un arbre qui est un graphe ne contenant pas de boucles.

Dans cet algorithme, chaque nœud possède un seul nœud parent excepté pour la racine de l'arbre qui est la configuration initiale. Chaque nœud peut posséder plusieurs nœuds fils, mais les feuilles de l'arbre n'en ont pas. La méthode RRT part du nœud racine pour explorer incrémentalement l'espace des configurations C afin d'atteindre la configuration finale. L'arbre RRT croît de proche en proche comme illustré à la figure 2.4.

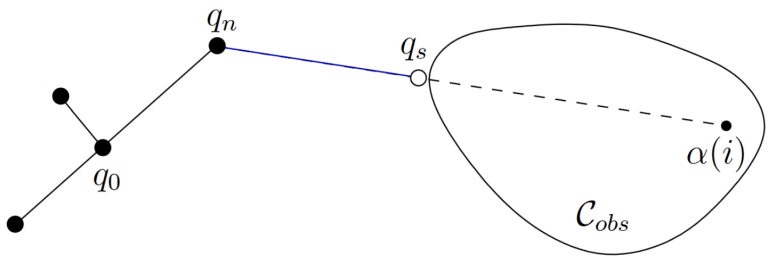


FIGURE 2.4 – Principe d'extension du RRT [45]

L'algorithme 1 décrit la méthode RRT. Cette méthode prend en argument un nombre N_{max} qui est le nombre maximal d'itérations que pourra faire cet algorithme. Soit une solution est trouvée avant d'atteindre N_{max}, soit l'algorithme stoppe. À chaque itération de l'algorithme, une configuration aléatoire est générée, ligne 2. Ligne 3, cette configuration q_{rand} est envoyée à l'algorithme Étendre.

Algorithm 1 RRT

Require: $W, q_{init}, q_{fin}, N_{max}, T$
1: **for** $i = 0$ **to** $i < N_{max}$ **do**
2: $q_{rand} \leftarrow$ ÉchantillonnageAléatoire()
3: $T \leftarrow$ Étendre(T, q_{rand})
4: **end for**

L'algorithme Étendre se charge de connecter la nouvelle configuration aléatoire q_{rand} à l'arbre T. Ligne 1, il cherche de l'arbre T, le nœud q_{near} le plus proche de q_{rand}. Ligne 2, une méthode Connexion() tente de relier les deux configurations. Si cette méthode réussit, les lignes 3 et 4 ajoutent à l'arbre le nouveau nœud q_{rand} et le nouvel arc depuis ce nouveau nœud jusqu'à q_{near} avant de retourner l'arbre mis à jour. S'il existe un obstacle le long du chemin alors la méthode Connexion(), ligne 7, renvoie la dernière configuration avant collision comme nouveau nœud et le chemin jusqu'à cette dernière pour la mise à jour de l'arbre.

Une variante très connue de RRT est la méthode BiRRT qui utilise deux *roadmaps*, une dont la racine est en q_{init} et une en q_{goal}. Cette méthode donne de bien meilleurs résultats en alternant l'ajout de nœuds à l'une puis l'autre *roadmap*.

Algorithm 2 Étendre

Require: T, q_{rand}
1: q_{near} = Plus_Proche_Noeud(T, q_{rand})
2: **if** Connexion(q_{rand} à q_{near}) OK **then**
3: T.Ajoute_Noeud(q_{rand})
4: T.Ajoute_Arc(q_{rand}, q_{near})
5: **return** T
6: **else**
7: **return** Échec
8: **end if**

2.2.5 Notations et définitions pour la planification de mouvement probabiliste

Ici sont résumées les notations utilisées en planification de mouvement probabiliste et tout au long de ce mémoire.

- Une configuration aléatoire est notée q_{rand} de l'anglais random pour aléatoire.
- La méthode d'échantillonnage est un algorithme permettant de choisir une configuration $q_{rand} \in \mathcal{C}$.
- Un échantillon est une configuration choisie par la méthode d'échantillonnage.
- Une méthode de détection de collision permet de déterminer si une configuration appartient à $\mathcal{C}_{libre}$ ou à $\mathcal{C}_{obst}$.
- La *roadmap* ou arbre est l'ensemble des configurations échantillonnées et ajoutées comme valides par la méthode de détection de collision puis reliées entre elles.

2.3 Conclusion

Nous avons introduit dans ce chapitre l'intérêt de la planification de mouvement, son formalisme et les principales méthodes utilisées. Nous avons mis l'accent sur les méthodes probabilistes car nous nous en servirons tout au long de ce mémoire. Les étapes importantes qui vont influencer le succès et les performances d'un PRM ou d'un RRT sont principalement la manière dont on échantillonne l'espace $\mathcal{C}$ et la fonction qui permet de relier ou d'étendre la *roadmap*.

Les méthodes aléatoires sont très efficaces pour traiter une grande variété de problèmes de planification de mouvement. Cependant, ces méthodes aléatoires perdent en efficacité dans des espaces encombrés. La probabilité de générer un échantillon aléatoire dans un espace libre devient très faible si cet espace est lui même très petit. A contrario, un humain saura généralement détecter un passage étroit à l'aide de sa vue. Par conséquent, nous croyons que la planification de mouvement serait bien plus efficace si un opérateur humain pouvait travailler conjointement avec un algorithme automatique.

Nous souhaitons modifier la manière d'explorer l'espace pour l'échantillonnage des configurations. Notre idée est de faire l'exploration de manière interactive en faisant collaborer un opérateur humain et un algorithme automatique. Nous proposons ainsi un algorithme de planification interactive de mouvement au chapitre suivant. En utilisant à la fois la vitesse de la machine et l'intelligence de l'humain, nous obtenons des résultats décrits au chapitre 6 qui dépassent largement les performances d'une machine seule ou d'un humain seul.

Chapitre 3

Planification interactive dans l'espace libre

Ce chapitre présente les possibilités d'interaction entre un homme et une machine lors de l'assemblage d'une pièce industrielle dans le cadre d'un problème de planification interactive de mouvement. Après avoir décrit les différents modes d'interaction avec l'humain, la question du partage d'autorité et ses principes pour l'interaction seront abordés. Seront ensuite présentés les outils utilisables pour cette interaction. Enfin, une solution algorithmique concernant la planification de mouvement interactive en espace libre sera exposée, suivie d'un cas d'école illustratif.

3.1 Problématique

Dès qu'une opération d'assemblage, de désassemblage ou de maintenance a lieu, un opérateur humain doit déplacer une pièce d'un produit industriel complexe au sein d'un environnement contraint.

Une opération représentative de ce type d'opération est l'assemblage d'un moteur d'essuie-glace de voiture. Il est nécessaire de l'insérer sous le capot dans un espace très réduit. Lors de la conception du prototype de voiture, chaque pièce est conçue séparément et la moindre erreur de conception géométrique empêchera son insertion. Aussi la faisabilité de l'assemblage devra être vérifié avant la mise en production.

Les essais permettant de vérifier qu'une pièce a la bonne géométrie consistent à trouver une trajectoire valide entre une configuration de départ et une configuration d'arrivée, configuration prévue pour un bon fonctionnement. Cette trajectoire peut être trouvée par un algorithme ou par un humain. La manipulation par l'opérateur humain doit donc permettre l'assemblage en condition réelle de la maquette numérique.

Pendant la manipulation de l'objet pour l'assemblage, l'opérateur humain utilise un certain nombre de sens et d'actions pour la mener à bien. En premier lieu, il lui faut voir

l'environnement dans lequel il évolue. Il doit pouvoir visualiser les obstacles, c'est-à-dire les objets présents le long de la trajectoire mais aussi l'objet qu'il porte en main. Il lui faut ensuite un moyen de déplacer l'objet et de se déplacer dans l'environnement. Il lui sera également utile de ressentir le contact selon les situations.

La recherche d'une trajectoire valide d'une configuration de départ à une configuration d'arrivée est le but des algorithmes de planification de mouvement. L'utilisation conjointe d'un algorithme et d'un humain lors de la planification de mouvement se nomme la planification interactive de mouvement. Elle nécessite d'adapter les algorithmes de planification pour permettre l'interaction homme-machine.

Il faut donc développer un moyen de transposer les sens et actions mentionnés plus haut par des sens et actions dans l'environnement virtuel. De nombreuses méthodes existent pour l'interaction et sont listées dans la section suivante.

3.2 État de l'art de la planification interactive

La planification interactive de mouvement permet à une machine ou algorithme de collaborer avec un opérateur humain afin de résoudre un problème de planification de mouvement plus rapidement que si l'un ou l'autre était seul.

Il existe trois catégories de planificateurs de mouvement interactifs comportant une ou plusieurs étapes. La première catégorie, présentée sous-section 3.2.1, comporte plusieurs étapes faisant par exemple intervenir l'opérateur humain seul avant de laisser complètement la main à un algorithme automatique agissant seul. La deuxième catégorie, présentée sous-section 3.2.2, dispose aussi de plusieurs étapes mais à la différence de la précédente catégorie, ces étapes alternent entre algorithme et humain de manière continue. Enfin, la troisième catégorie d'algorithmes présentée sous-section 3.2.3, ne comporte qu'une seule étape faisant interagir à la fois l'humain et l'algorithme.

3.2.1 Interaction en plusieurs étapes

He [30] propose un algorithme de planification interactive en deux étapes. En plus des configurations q_i et q_f initialement présentes dans un graphe PRM, il ajoute j configurations dites critiques q_{cj} à la main avec une interface haptique. Ces configurations permettent de prendre en compte des contraintes de production. L'algorithme tente dans une deuxième étape de connecter les configurations critiques ce qui accélère l'exploration de l'environnement. En effet, les configurations critiques permettent à l'algorithme d'étendre la *roadmap* avec plus d'indications que seulement la configuration de départ et celle d'arrivée.

Une autre méthode est de dessiner des chemins entiers à l'aide d'un périphérique d'interaction. Bayazit [6] utilise cette méthode où un chemin est dessiné manuellement dans une première étape. Ce chemin étant susceptible de comporter des portions en collision, l'algorithme dans une seconde étape, décale les portions non valides hors de l'espace en

collision pour les repousser soit dans l'espace libre soit dans l'espace de contact entre l'objet et les obstacles. Cet algorithme est une solution pour la planification au contact et sera détaillé dans le chapitre suivant 4.

Le problème lorsque l'humain agit en premier est qu'il est possible qu'il donne des indications impossibles à résoudre à la machine : un passage que l'humain a considéré faisable mais qui est en réalité trop étroit, une configuration critique ajoutée dans une composante non connexe de l'espace libre.

3.2.2 Interaction en plusieurs étapes avec alternance

Ici sont décrites les contributions de planification interactive ayant lieu en plusieurs étapes avec alternance. Ces deux contributions alternent entre une étape automatique suivie d'une étape collaborative pendant laquelle l'étape automatique peut être relancée sous certaines conditions. Elle sera suivie d'une nouvelle étape collaborative et ainsi de suite.

Le planificateur de Ladevèze avec approche globale [42] s'appuie sur une décomposition hors ligne de l'espace libre en octree [51]. Le processus de planification automatique est une première étape et a lieu en deux temps. Il fait d'abord appel à un algorithme A* qui cherche le plus court couloir de cellules adjacentes de la configuration de départ à la configuration d'arrivée afin de réduire l'espace libre à explorer aléatoirement dans une troisième étape. Il trouve ensuite une trajectoire précise à l'intérieur de ce couloir à l'aide d'un algorithme probabiliste de type RRT.

Une fois le couloir obtenu, l'étape collaborative de l'algorithme consiste en un guidage le long de cette trajectoire calculée lors de la première étape. Si l'opérateur s'en écarte significativement, le processus de planification A* puis RRT est relancé depuis la configuration courante. Une nouvelle trajectoire de guidage est calculée et guide de nouveau l'utilisateur. On a donc un processus continu alternant une étape de planification suivie d'une étape de guidage. L'intérêt de cette méthode est que le guidage se fait à l'intérieur d'une zone valide ayant pris en compte les obstacles lors d'un premier traitement. Cela permet d'accélérer le traitement temps réel mais demande un pré-traitement qui peut être long si l'environnement est complexe.

Le planificateur de Cailhol basé sur un découpage multiniveaux de l'environnement [13] commence par une étape de planification. Cette étape est divisée en deux phases. La première phase appelée planification grossière est une planification se servant d'informations sémantiques et topologiques afin de trouver le chemin topologique le moins coûteux à l'aide d'un algorithme de Dijkstra. Une seconde phase, dite de planification fine, trouve une trajectoire géométrique dans le chemin topologique choisi à l'aide d'un algorithme de planification adapté à chaque lieu topologique (par exemple RRT). Une fois le processus de

planification terminé, une deuxième étape commence où l'utilisateur est guidé le long du chemin géométrique. Il peut s'en écarter pour forcer l'ensemble du processus à recalculer les chemins topologiques et géométriques de la même manière que pour le planificateur avec approche globale de Ladevèze [42]

Cette solution est efficace pour trouver des trajectoires et utilise l'humain ainsi que la machine mais se heurte à une difficulté qui est celle de l'étiquetage sémantique des différents lieux à traverser.

3.2.3 Interaction en une seule étape

Contrairement aux travaux de la section précédente, il n'y a dans les contributions suivantes qu'une seule étape, l'algorithme et l'opérateur travaillant toujours en même temps. Nous évoquerons ici les principaux travaux de ce domaine, qui sont soit des méthodes sans guidage, soit des méthodes où l'utilisateur est guidé le long d'une trajectoire.

3.2.3.1 Sans guidage

Naderi [55] propose un algorithme de planification de mouvement temps-réel basé sur une méthode de type RRT. Il s'inspire des travaux de Otte [56] proposant une méthode capable de replanifier en temps réel une trajectoire en fonction d'obstacles mobiles. Sa méthode alternant entre la planification et la reconnexion d'arcs déjà existants est conçue pour permettre de multiples requêtes au sein d'un même environnement. L'opérateur peut repositionner à chaque itération la configuration de départ, d'arrivée et les obstacles. L'ajout ou le déplacement d'un obstacle dynamiquement coupe certains arcs de la *roadmap*. Grâce à leur méthode, la *roadmap* est reconnectée rapidement. La figure 3.1 donne le fonctionnement de cette méthode. La vignette (a) illustre la croissance de la *roadmap*, en (b) et (c) la configuration but est modifiée à la volée ainsi que la configuration d'origine et les obstacles.

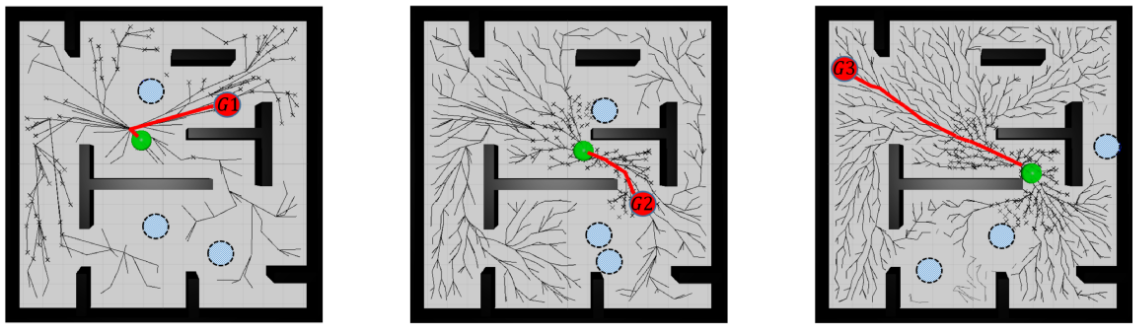


FIGURE 3.1 – Processus de replanification Naderi [55]

3.2.3.2 Avec guidage

Le planificateur local de Ladevèze [42] se base sur la configuration objectif pour le calcul d'une force d'attraction sans tenir compte des obstacles puis fait une interpolation linéaire entre la configuration courante et la configuration objectif afin de déterminer la direction de guidage. Un bras haptique servant d'interface utilisateur le guide vers la configuration objectif. C'est à l'utilisateur de prendre en compte les obstacles de l'environnement virtuel pour les éviter à l'aide d'un retour visuo-haptique. Il lui faut s'écarter de l'assistance proposée pour contourner les obstacles. Cette méthode a l'avantage de fournir une aide constante à l'utilisateur, quelle que soit la situation dans laquelle il se trouve, la complexité du problème ou son encombrement localement. Le désavantage de cette méthode est qu'elle peut se retrouver bloquée dans un minima local [40], l'opérateur devant alors gérer cette situation.

Une méthode continue avec guidage mais avec une approche probabiliste a été proposée par Flavigné. Son algorithme IRRT [22] permet d'influencer la propagation d'un RRT. L'utilisateur et le planificateur automatique échangent des pseudo forces dont la résultante contraint l'échantillonnage aléatoire de l'algorithme. Une pseudo force de guidage notée F_a est calculée en fonction des nœuds de la *roadmap* RRT et des obstacles environnants. Une pseudo-force notée F_u produite par l'utilisateur est mesurée au travers du bras haptique. La somme pondérée de ces deux forces donne une résultante F_r. Il en déduit une ellipsoïde à l'intérieur de laquelle se fait l'échantillonnage.

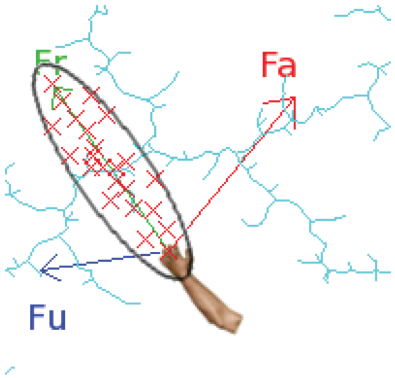


FIGURE 3.2 – Force attractive, utilisateur et résultante [22]

Ces approches continues doivent être exécutables en temps réel afin de permettre l'interaction avec un opérateur sans attente. Cela devient difficile à contrôler dans des environnements trop encombrés où les obstacles prennent beaucoup de place en mémoire. Cela devient aussi difficile à contrôler pour des robots à grande dimension car il est nécessaire

de calculer la valeur de chaque dimension en temps réel. Par conséquent, la plupart des contributions du domaine contrôlent des robots de faibles dimensions.

3.3 Partage de contrôle

Il existe un partage de contrôle lorsqu'au moins deux entités collaborent pour le contrôle d'un même système [12]. Ces deux entités peuvent être des humains ou des ordinateurs.

Cette collaboration entre entités est généralement mise en œuvre pour permettre à une machine d'assister un humain pour la commande d'un système afin effectuer une tâche. Ces systèmes de partage de contrôle sont soit des systèmes d'assistance soit des systèmes d'extension [70]. Un système d'assistance à l'humain allège sa charge cognitive tandis qu'un système d'extension apporte un contrôle que l'humain ne pourrait pas avoir.

Le partage de contrôle est constitué de deux éléments. D'abord un système de contrôle automatique qui génère des commandes pour contrôler le système [12] ensuite un contrôle d'autorité permettant de mixer les commandes issues du système automatique avec celle de l'humain pour produire celles soumises au système [52]. La figure 3.3 décrit le cadre général du partage de contrôle. Le bloc commande système de cette figure indique que les commandes générées par l'utilisateur et le système d'assistance peuvent être définies dans un espace différent de celui des actionneurs du système [12].

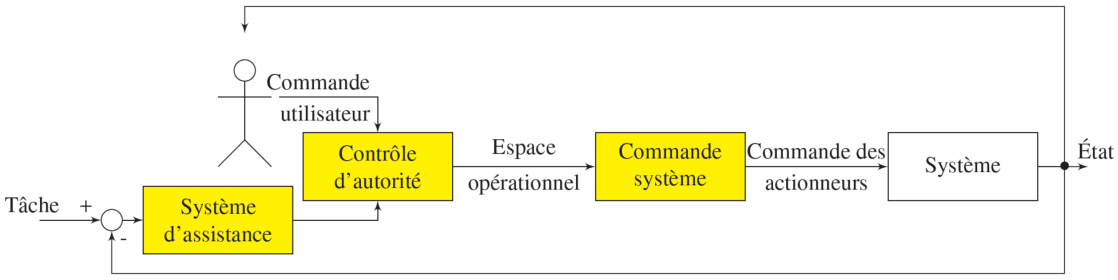


FIGURE 3.3 – Partage de contrôle

Le partage de contrôle peut être complété par un système de prédiction d'intention que nous traiterons au chapitre 5.

Disposant d'un contrôle d'autorité, il faut définir quelles sont les opérations à la charge du système automatique et quelles sont celles à la charge de l'humain. Le partage d'autorité définit la manière de fusionner les commandes de l'utilisateur et celles de la machine pour contrôler le système. Il s'agit d'arbitrer qui, de la machine ou de l'humain obtient l'autorité sur le système et de quelle façon. Cet arbitrage est soit statique, soit dynamique, c'est-à-dire qu'il peut évoluer au cours du temps.

Dans le cadre de la planification interactive de mouvement, le partage d'autorité peut se faire sur toutes les fonctions de la planification : la zone de recherche, l'évitement des

obstacles, le partage du temps. Nous reprenons ici certaines des contributions décrites à la section précédente que nous analysons sous l'angle du partage d'autorité.

Un des critères importants lors de la planification de mouvement est la zone de recherche. De nombreux travaux ont montré que la planification de mouvement est très rapide pour des environnements faiblement encombrés mais qu'elle devient petit à petit inefficace lorsque des zones à fort encombrement existent dans l'espace à explorer. Une des solutions souvent mises en œuvre par les planificateurs est alors de réduire l'espace à explorer pour se concentrer sur les zones difficiles. Dans le cadre de la planification de mouvement interactive, plusieurs méthodes ont été proposées.

Le premier type de méthode est le pré-traitement de l'environnement comme le fait Ladevèze [42]. L'étape d'interaction de son planificateur guide l'utilisateur le long d'une trajectoire contenue dans un sous espace de W précalculé hors ligne. Dans ce cas, la machine a autorité initialement pour la zone de recherche. L'utilisateur peut décider de quitter cette zone afin de relancer le prédécoupage. Le partage d'autorité est donc le suivant : la machine propose une zone de recherche et l'humain demande à la machine de la modifier ou non.

Une méthode de collaboration différente pour la définition de la zone de recherche est celle de Denny [19]. Il décrit une méthode contraignant la recherche par régions, une région étant définie comme un volume fermé dans l'environnement. L'algorithme permet alors la définition de régions de différentes sortes, des régions d'attraction, des régions à éviter et des régions recommandées par le planificateur. L'utilisateur sélectionne les régions à catégoriser à l'aide d'une simple souris à tout moment. Le partage d'autorité est donc le suivant : la machine explore tout l'espace et l'humain lui donne des ordres d'attraction ou d'évitement par région.

Pour rappel, la planification de mouvement a pour objectif de trouver une trajectoire valide pour un objet d'une configuration de départ à une configuration d'arrivée sans entrer en collision avec les obstacles. L'évitement des obstacles est donc une partie critique dans toute opération de planification de mouvement.

Certaines méthodes de planification interactive ne sont pas prévues pour garantir la non collision des obstacles. Par exemple, le planificateur de Cailhol [14] relaxe la contrainte de non collision dans certaines conditions.

Dans ce cas, la trajectoire de guidage est potentiellement en collision par endroits. Le partage d'autorité est donc le suivant : la machine guide le long d'une trajectoire, l'humain peut en dévier pour forcer le calcul d'une nouvelle trajectoire. Lorsque l'humain est chargé d'éviter les obstacles, la charge de l'ordinateur diminue ce qui permet aux planificateurs de ce type d'être efficaces dans des environnements complexes. En revanche, ils sont parfois mis en échec par leur incapacité à gérer les obstacles automatiquement, en particulier dans les passages étroits.

D'autres méthodes de partage d'autorité laissent la machine éviter les obstacles. Les méthodes interactives basées sur les planificateurs probabilistes ajoutent des configurations

à une *roadmap* seulement dans le cas où ces configurations sont inclues dans l'espace libre. C'est donc l'algorithme qui est capable d'éviter les obstacles.

Il existe deux stratégies lorsque l'algorithme doit éviter les obstacles. La première est celle mise en place dans la plupart des planificateurs probabilistes : si une configuration est en collision, elle est rejetée. La seconde dilate l'espace libre afin de permettre une petite interpénétration entre le robot et l'obstacle. Cette petite erreur est par la suite corrigée dans une étape ultérieure, comme dans les stratégies par rétraction détaillées au chapitre 4.1.

Enfin, le partage d'autorité se fait par un partage du temps de recherche entre l'humain et la machine selon des stratégies différentes. Les architectures de processeurs modernes possédant de nombreux coeurs de processeurs, il est aujourd'hui facile de dédier par exemple un coeur à la gestion de l'interface avec l'humain et de laisser les autres coeurs se charger de la planification de mouvement automatique. Bien que cette solution soit préconisée pour les applications industrielles, paralléliser les algorithmes de planification de mouvement est une spécialité à part entière, voir par exemple [57], et n'est pas l'objet de ces travaux. Ainsi, le partage du temps de la planification se fera toujours sur un seul coeur de processeur.

3.4 Périphériques d'interaction

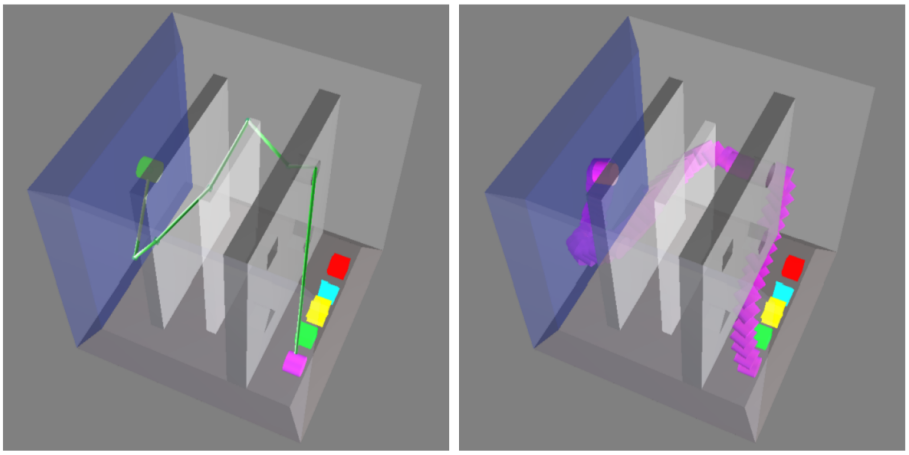
Le cadre qui nous intéresse dans cette thèse est celui de la planification de mouvement interactive. Divers périphériques sont utilisables afin d'interagir avec l'ordinateur pour la recherche d'une trajectoire valide. Ces périphériques sont soit sensoriels transmettant des informations de la machine vers l'utilisateur soit moteurs transmettant des informations de l'utilisateur vers la machine soit sensorimoteurs, étant capables des deux.

3.4.1 Périphériques sensoriels

Parmi les cinq sens, seule la vue sera traitée dans cette sous-section. Ce sens permet à l'utilisateur de voir l'environnement, ce qui lui permet d'appréhender le problème et d'afficher l'objet en déplacement. Dans le cadre de l'interaction continue, lorsque le planificateur automatique cherche une trajectoire, il est possible d'indiquer à l'écran la *roadmap*. Cela permet à l'utilisateur de savoir où la machine a déjà cherché et dans quelles zones elle rencontre des difficultés à trouver une trajectoire. L'utilisateur peut alors chercher dans des zones pertinentes.

Si l'interaction est en deux temps, la machine a déjà trouvé une trajectoire et la propose à l'utilisateur à l'écran en y affichant son résultat. Dans ce cas, l'utilisateur a le choix de dévier de la trajectoire s'il en préfère une autre. La figure 3.4 illustre cette méthode.

La vision est donc un premier canal d'informations disponible pour l'interaction. C'est un bouclage entre l'opérateur et le système manipulé fournissant un retour temps-réel sur les actions à mener. Elle fournit une première couche d'assistance à l'opérateur.



(a) Planification initiale. (b) Trajectoire planifiée.

FIGURE 3.4 – Une trajectoire est trouvée par la machine et proposée à l'utilisateur [13].

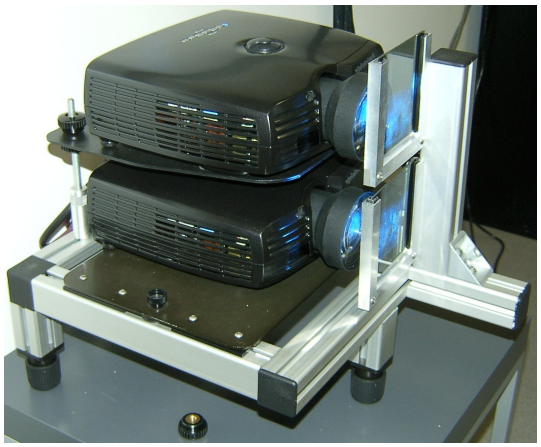


FIGURE 3.5 – Projecteurs pour la 3D

FIGURE 3.6 – Écran pour la vision 3D

3.4.1.1 Écran 2D

De manière évidente, l'écran d'ordinateur sera le premier choix pour l'affichage. Le principal problème d'un écran classique est que la scène 3D est projetée sur une surface 2D. Cela limite la perception de la géométrie. L'humain ne peut pas toujours interpréter correctement les objets vus.

FIGURE 3.7 – Lunettes polarisées passives

FIGURE 3.8 – Lunettes actives

3.4.1.2 Écran 3D

Une solution disponible aisément est une visualisation en trois dimensions. Ce type de visualisation diminue fortement les problèmes cités au paragraphe précédent car elle permet une meilleure immersion dans l'environnement ce qui facilite la tâche de l'opérateur.

Deux types de systèmes existent. Le premier est un système de stéréoscopie passive. Il est constitué de deux projecteurs munis chacun d'un filtre à polarisation circulaire. La polarisation est différente pour chaque projecteur. L'image de chaque projecteur est projetée sur un même écran qui conserve la polarisation de la lumière. L'utilisateur s'équipe alors d'une paire de lunettes 3D dont chaque verre est muni d'un filtre afin de visionner une image différente par oeil. Ce système est similaire à celui utilisé dans la plupart des cinémas bien qu'au cinéma, la polarisation soit verticale et horizontale.

Le second système est de la stéréoscopie active. Il nécessite pour cela une paire de lunettes actives. L'écran affiche successivement l'image de l'oeil droit puis celle de l'oeil gauche. Dans le même temps, la paire de lunettes occulte soit le verre droit soit le verre gauche. La période d'occultation étant plus petite que la durée de la persistance rétinienne, l'oeil humain ne constate pas de coupure et la vision se fait alors en 3D.

3.4.1.3 Casque RV

Une innovation récente en termes de visualisation est le casque de réalité virtuelle (Head Mounted Display HMD). Ce casque intègre à la fois de la vision stéréoscopique et des capteurs de mouvement. Cela permet de modifier la visualisation de l'utilisateur en fonction de ses mouvements.

FIGURE 3.9 – Casque de réalité virtuelle HTC Vive.

3.4.2 Périphériques moteurs

Les périphériques moteurs permettent de donner des ordres à la machine. Ils sont utilisés ici pour se déplacer dans la scène et déplacer l'objet à planifier.

Lorsque l'humain souhaite se déplacer dans la scène, il a besoin d'un périphérique spécifique pour se mouvoir dans l'environnent. Ceci afin de visionner les obstacles sous tous les angles et de trouver une trajectoire valide.

Déplacer l'objet à planifier peut se faire à l'aide du même périphérique que celui utilisé pour se déplacer dans la scène. Divers contrôleurs sont disponibles, chacun ayant des avantages différents. Certains périphériques d'ordinateur permettent d'interagir de manière plus intuitive que les traditionnels claviers et souris. Ils sont utiles notamment lors des simulations virtuelles. Ces périphériques possèdent plus de degrés de liberté leur permettant de contrôler un objet dans un univers virtuel. En réalité, définir la configuration d'un objet solide dans l'espace nécessite six paramètres ou dimensions, ce qui n'est pas ergonomique avec une souris standard (planaire).

3.4.2.1 Clavier, souris

Une souris standard (planaire) permet le déplacement dans une scène. Il existe de nombreux logiciels n'utilisant que ce périphérique, parfois couplés à un clavier.

L'industrie du jeu vidéo a développé ces dernières décennies les canons du contrôle d'avatar à l'aide d'un clavier et d'une souris. La plupart des logiciels qui nécessitent un déplacement dans l'espace reprennent tout ou partie de ce qui est devenu une norme de facto, tel que le zoom à la molette (et non le défilement).

3.4.2.2 Souris 6D

La souris 6D remplace avantageusement une souris traditionnelle. Cette souris offre la possibilité de modifier six paramètres à la fois, voir figure 3.10. Tandis qu'une souris habituelle évolue sur une surface planaire n'offrant que deux degrés de liberté, celle-ci permet

le déplacement dans les trois dimensions de l'espace et la modification de l'orientation de l'objet à l'aide de trois rotations.

FIGURE 3.10 – Souris 6D

3.4.2.3 Manette

L'utilisation d'une manette de jeu vidéo telle que celle de la Playstation ® ou celle de la Xbox ® permet de résoudre le problème du contrôle des rotations car elles disposent de deux joysticks. Il est alors possible de contrôler trois angles ainsi qu'un quatrième paramètre, souvent une vitesse de déplacement.

Ces manettes sont aujourd'hui utilisées dans la robotique, en particulier pour le contrôle de drones, qu'ils soient terrestres ou aériens. De nombreuses applications militaires notamment utilisent de tels contrôleurs qui s'avèrent plus efficaces qu'un clavier et une souris pour nombre de situations. D'autres manettes plus récentes existent et s'inspirent toutes du modèle Playstation ® .

Le déplacement d'un objet comme le déplacement dans une scène est très ergonomique à l'aide d'une manette de jeux vidéos qui permet les deux à la fois.

3.4.2.4 Capture de mouvement sans marqueurs

La kinect ® est un matériel initialement développé pour le jeu vidéo qui s'est avéré un capteur très performant. Une caméra équipe la kinect ® pour la capture 3D d'un mouvement permettant sa localisation dans l'espace. Un capteur de profondeur améliore l'analyse du mouvement. La kinect est motorisée afin de suivre un mouvement dans le plan frontal. Elle est en outre équipée d'une série de microphones utilisés pour la reconnaissance vocale. Ce périphérique est utilisé comme une manette mains-libres très utile pour les applications de réalité virtuelle.

La technologie Leap motion ® est une technologie utilisée en réalité virtuelle pour la reconnaissance de gestuelles. La technologie est basée sur des capteurs infrarouges et l'analyse de leurs données. Elle permet une interaction très naturelle avec l'ordinateur en utilisant le mouvement de ses doigts, voir figure 3.11.

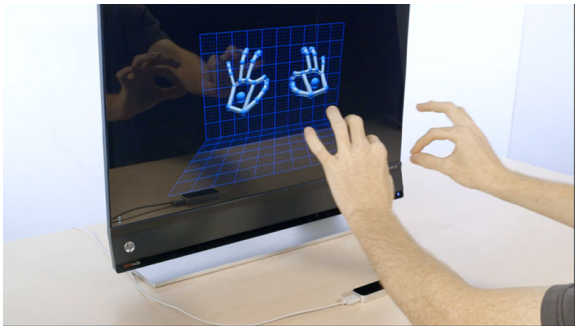


FIGURE 3.11 – Capteur Leap motion.

3.4.2.5 Capture de mouvement avec marqueurs

La capture de mouvement, ou motion capture, est une technologie couramment utilisée dans le cinéma, le jeu vidéo et dans la recherche en robotique pour ne citer que ces domaines. Elle permet de localiser des objets dans l'espace. Cette technologie utilise des caméras infra rouges qui émettent des flashs infrarouges. Des petites sphères, appelées marqueurs, dotées d'un revêtement réfléchissant la lumière infra rouge sont positionnées sur le corps dont on veut mesurer la position. La position des marqueurs est lue par chacune des caméras ce qui permet de trianguler la position du ou des corps dans l'espace.

Chaque objet possède une signature spécifique de marqueurs. Ils sont positionnés différemment pour chacun d'entre eux, voir par exemple la figure 3.12 qui est une manette possédant une signature unique. Il est également possible de positionner un grand nombre de marqueurs sur une chaîne cinématique tel que le corps humain, voir figure 3.13.

La capture de mouvement peut donc être positionnée sur la tête ou les bras de l'opérateur afin d'identifier sa position et de déplacer l'environnement en fonction de ses mouvements de manière complètement naturelle. La figure 3.14 est une paire de lunettes avec une signature permettant la localisation de l'opérateur afin d'adapter la vue à sa position dans l'espace. Cette technologie peut aussi être utilisée pour donner des ordres spécifiques [21].

3.4.3 Périphériques sensorimoteurs

Les périphériques sensorimoteurs sont capables à la fois de donner des ordres à la machine et de recevoir des informations de cette dernière. Les informations de la machine

FIGURE 3.12 – Le flystick équipé de marqueurs, repérable dans l'espace grâce à sa signature.

FIGURE 3.13 – Capture de mouvement corps complet.

sont transmises à l'utilisateur qu'il capte à l'aide de l'un de ses sens. Nous ne traitons ici que des périphériques sensorimoteurs haptiques.

Il existe certains périphériques dotés de retour d'effort permettant d'utiliser le sens

FIGURE 3.14 – Lunettes passives avec marqueurs

haptique. Certains joysticks pour le jeu vidéo notamment, sont motorisés et permettent un retour d'effort mais sur deux axes seulement. Les périphériques haptiques sont aujourd'hui utilisés dans de nombreux domaines d'application tel que l'apprentissage [5] [4], la planification [47], la chirurgie [3] [74] [27], le jeu vidéo [37].

3.4.3.1 Bras haptique

Un bras haptique est un périphérique sensorimoteur qui est une chaîne cinématique réversible ayant en général au moins trois degrés de liberté motorisés. Chaque degré de liberté d'un bras haptique n'est pas nécessairement motorisé. Un bras haptique est une interface bidirectionnelle. Elle permet d'agir sur le monde virtuel tout en fournissant un retour sensorimoteur utilisé soit pour un retour haptique soit pour guider l'opérateur. Les figures 3.15 et 3.16 montrent deux exemples différents de bras haptiques.

Des informations haptiques sont toujours utiles et parfois nécessaires lorsque la zone que l'utilisateur traverse est très encombrée et nécessite des mouvements très précis. Il se peut également que l'affichage ne permette pas d'appréhender toute la scène, typiquement lorsqu'une insertion empêche de voir où l'objet est déplacé ou tout simplement si l'environnement est trop encombré pour que l'utilisateur se souvienne de tout. Il faut garder à l'esprit que la visualisation se fait sur un écran et que seule est disponible la projection de l'environnement et du robot selon un certain point de vue.

Les informations haptiques sont transmises à l'utilisateur au travers du bras haptique sous forme de force ou de couple. Elles peuvent être utiles pour traverser les passages encombrés ou les opérations d'insertion décrites ci-dessus car le retour haptique est une information utilisable intuitivement, très efficace pour appréhender la géométrie d'un environnement, réel ou virtuel.

FIGURE 3.15 – Sigma7 : 7ddl.

FIGURE 3.16 – Phantom omni : 6ddl dont 3 motorisés.

3.4.3.2 Autres périphériques haptiques

Il existe d'autres types de périphériques haptiques comme par exemple les gants haptiques. Un tel équipement permet la manipulation avec retour d'effort afin de rendre l'expérience plus réaliste qu'avec un bras haptique. Nous pouvons citer le gant de VRGluv ® , celui de DextaRobotics ® et une contribution du monde académique, le gant de Jadhav ® [35].

Nous pouvons aussi citer les nouveaux types d'écrans tactiles dit haptiques [28]. Ces écrans présentent des zones proéminentes dynamiquement modifiables sur la surface de l'écran. Cela permet notamment de simuler des boutons rendant l'expérience tactile plus confortable.

3.5 Planification interactive dans l'espace libre

Nous proposons ici un algorithme de planification interactive de mouvement basé sur le partage du temps nommé I-RRT. Grâce à une interface permettant à l'utilisateur de déplacer l'objet planifié dans l'espace de travail et à une méthode de partage d'autorité, nous pouvons planifier des trajectoires en utilisant conjointement l'homme et la machine.

3.5.1 Approche

Nous détaillons ici l'approche choisie pour notre contribution par rapport à l'état de l'art et les raisons qui ont motivé ces choix. Nous y reprenons les trois points présentés aux sections précédentes 3.2, 3.3 et 3.4.

3.5.1.1 Interaction

Nous avons choisi une interaction en une seule étape permettant à la machine de collaborer avec l'utilisateur pour l'exploration de l'espace. Cela est obtenu au détriment de la vitesse de résolution du problème par rapport à une solution en plusieurs étapes si l'on considère uniquement l'étape avec l'humain. L'absence de pré-traitement permet en revanche de tester directement les objets après leur conception.

Cette méthode n'utilise pas de guidage haptique, à cause de l'utilisation d'un périphérique non motorisé. Il existe en revanche un guidage visuel par l'affichage de la *roadmap* à l'écran. L'utilisateur peut donc en temps réel savoir où il est préférable d'explorer.

3.5.1.2 Partage d'autorité

Le partage d'autorité entre l'algorithme et l'humain de notre méthode ne dépend pas de la zone de recherche. Les deux entités, l'algorithme et l'humain, peuvent explorer l'intégralité de l'environnement tout au long de la simulation. Nous avons choisi cette modalité de partage d'autorité car elle tire profit de la complétude probabiliste de l'algorithme probabiliste tout en laissant l'utilisateur explorer tout l'environnement.

Le rejet des configurations en collision est laissé entièrement à la charge de l'ordinateur. Les configurations en collisions seront donc toujours rejetées, qu'elles proviennent de la machine ou de l'humain.

Le temps de processeur est partagé entre l'utilisateur et l'ordinateur, l'autorité étant donnée à l'un puis à l'autre alternativement, comme expliqué à la section suivante 3.5.2.

3.5.1.3 Périphériques d'interaction

Dans une première approche nous avons choisi d'utiliser un simple écran d'ordinateur sans vision 3D. La raison est que nous avons implémenté cet algorithme sans environnement de réalité virtuelle afin d'éprouver ses capacités. Le moyen utilisé pour se déplacer dans l'environnement doit permettre de se déplacer à la fois en translation et en rotation. De plus il faudra pouvoir s'éloigner ou se rapprocher par des fonctions de zoom/dézoom. Le périphérique d'interaction choisi pour déplacer l'objet dans l'espace est dans ce cas une souris 6D conjointement avec une souris planaire. Ce choix permet de déplacer l'objet d'une manière naturelle. Cependant l'absence de retour d'effort ne permet pas une simulation immersive et limite les capacités exploratoires.

3.5.2 Algorithme interactif

Nous proposons un algorithme de planification de mouvement interactive continue, sans guidage. La recherche spatiale est complète pour les deux entités partageant le temps et avec gestion automatique des obstacles.

En parallèle des actions de l'utilisateur, nous utilisons un algorithme de planification de mouvement probabiliste, le RRT. Ce choix est justifié dans le cadre de requêtes uniques en environnement très contraint, ce qui est notre cas.

Pour rappel, à chaque itération du RRT, une configuration q_{rand} de l'objet à déplacer est aléatoirement tirée dans tout l'espace de travail. Si cette configuration q_{rand} s'avère faire partie de l'espace libre et s'il existe un chemin faisable entre q_{rand} et une autre configuration q_{near} déjà présente dans la *roadmap* alors on ajoute q_{rand} à la *roadmap* ainsi que le chemin de q_{near} à q_{rand}.

L'utilisateur positionne l'objet à planifier dans l'espace dont la configuration est q_{user}. On appellera cet objet OBJ. Cette configuration q_{user} est positionnée indépendamment de q_{rand}, la configuration aléatoire tirée par le planificateur automatique.

L'algorithme 3 propose une mise en œuvre des principes de planification interactive proposés dans la sous section précédente 3.5.1. Il a pour argument l'espace de travail W. Le second argument est T, la *roadmap* contenant au départ la configuration initiale q_{init} et la configuration finale q_{goal}. Le paramètre α est le coefficient de partage d'autorité, fixe pendant toute la durée de la simulation. La méthode prend enfin comme argument la configuration q_{user} positionnée par l'utilisateur à l'aide de son périphérique d'interaction.

Algorithm 3 Planification Interactive

Require: W, T, α, q_{user}
1: **loop**
2: $a \leftarrow \text{rand}(0, 1)$
3: **if** $a > \alpha$ **then**
4: $q_{current} \leftarrow q_{user}$
5: $T \leftarrow \text{Add\_Tree}(q_{current})$
6: **else**
7: $q_{current} \leftarrow \text{Random\_Shooter}()$
8: $T \leftarrow \text{Add\_Tree}(q_{current})$
9: **end if**
10: **end loop**

Chaque itération commence par tirer un nombre aléatoire a compris entre 0 et 1, ligne 2. Ce nombre détermine à qui l'autorité est donnée. Si $a > \alpha$ alors le planificateur donne autorité à l'humain, ligne 4. Dans ce mode humain, le planificateur au lieu de tirer une

configuration aléatoirement, lit la configuration q_{user} définie par l'utilisateur et l'envoie au planificateur à la place de la configuration aléatoire. Ainsi est-il possible de partager le temps de travail entre l'homme et la machine en attribuant à chacun un pourcentage de configurations envoyées au planificateur, de 0% (tout humain) à 100% (tout machine). La configuration q_{user} est sauvegardée et envoyée à la méthode Add_Tree(). Cette méthode détermine si la configuration en paramètre n'est pas en collision. Si tel est le cas, elle cherche alors un chemin jusqu'à la configuration la plus proche déjà présente dans T, ligne 5. Cette méthode retourne alors dans T un nouveau nœud et un nouvel arc si le chemin est faisable.

Dans le cas où $a < \alpha$, l'algorithme entre en mode automatique, ligne 7. Dans ce cas, la fonction Random_Shooter() tire une configuration aléatoire q_{rand}. Cette configuration est envoyée à la méthode Add_Tree() qui ajoute un nœud à la *roadmap*.

3.5.3 Schéma d'interaction global

La figure 3.17 illustre le fonctionnement de notre algorithme interactif. Deux boucles agissent en parallèle. La première boucle est humaine. Un opérateur humain utilise une souris 6D pour agir sur la *roadmap* qui est affichée à l'écran et visible en temps réel par ce même opérateur. L'opérateur humain se sert des modèles 3D et d'un détecteur de collision pour ajouter des configurations à la *roadmap*.

La deuxième boucle est celle de la machine. Elle possède aussi les modèles 3D de l'objet à planifier et de l'environnement et se sert du même algorithme de détection de collision pour ajouter des nœuds et des arcs à la *roadmap*, elle-même toujours visible par l'opérateur humain.

Les deux entités collaborent au travers de la *roadmap*, tentant l'une comme l'autre de l'enrichir par l'ajout de nœuds dans des espaces libres non encore connectés.

3.5.4 Exemple expérimental illustratif

Pendant la planification, un repère est affiché à l'intérieur de l'objet à planifier OBJ. C'est ce point que l'utilisateur contrôle à l'aide de la souris 6D ou tout autre périphérique d'interaction. C'est autour de ce point que l'objet tourne lorsque l'utilisateur en donne l'ordre. Mais c'est aussi et surtout ce point qui est affiché à l'écran sous la forme d'un repère. Ce repère correspond à une configuration q_{user} dans $\mathcal{C}$.

La figure 3.18 montre un objet de forme L se déplaçant d'un point à un autre. La trajectoire parcourue est clairement visible sous la forme de repères 3D composés de trois flèches, rouge, verte, bleue indiquant les orientations successives de l'objet. Les positions de ces repères indiquent les configurations générées par l'utilisateur à l'aide du périphérique d'interaction. Ces repères sont reliés entre eux par des arcs ici en jaune. Dans cette figure, l'utilisateur est le seul à générer des configurations donc $\alpha = 0$, le mode est totalement humain.

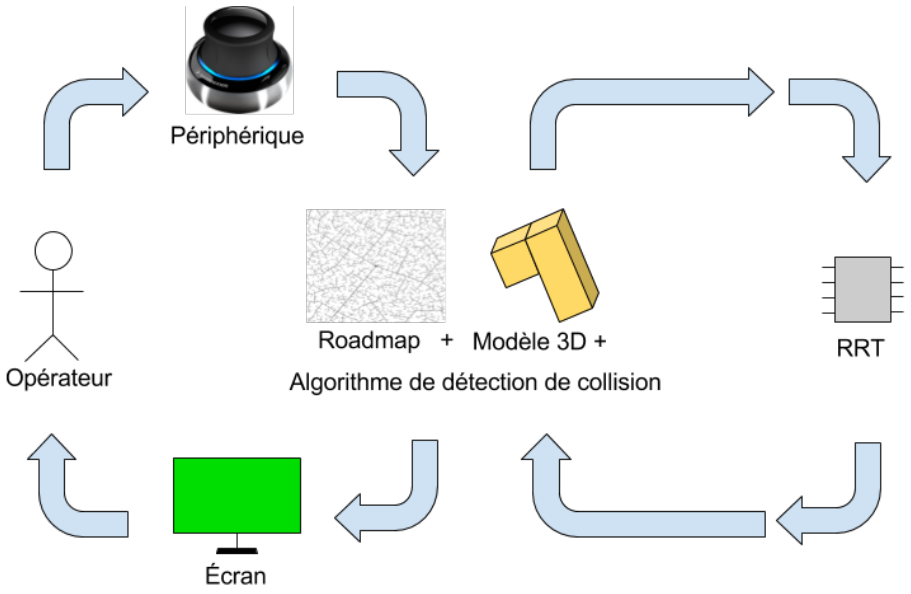


FIGURE 3.17 – Schéma global

La figure 3.19 quand à elle, montre l'exploration d'un espace situé au dessus d'un plan gris. On voit également des configurations visibles sous la forme de repères et reliés entre eux par des arcs jaunes. Ici, seules des configurations aléatoires sont générées, c'est-à-dire que $\alpha = 1$, le mode est totalement automatique.

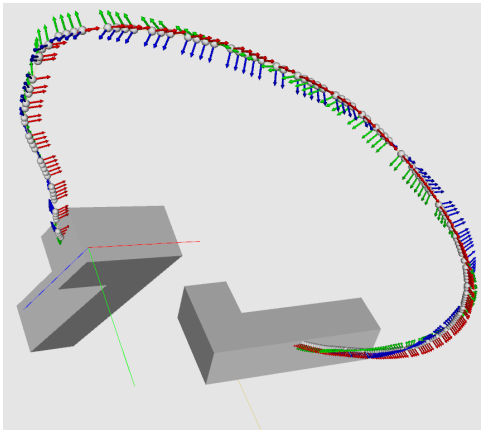


FIGURE 3.18 – Mode humain.

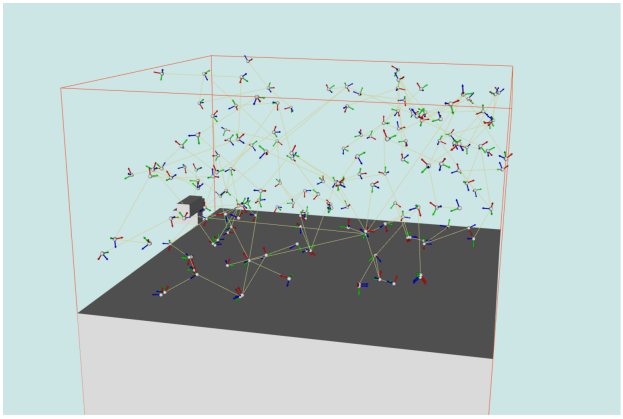


FIGURE 3.19 – Mode machine.

Ainsi par l'introduction d'une méthode de partage du temps nous avons permis au planificateur automatique de prendre en compte les actions d'un opérateur humain. À

l'aide d'un unique paramètre α, nous avons permis le partage d'autorité entre l'homme et la machine.

3.6 Exemple du labyrinthe

3.6.1 Expérience

Ici est présenté un exemple d'environnement qui n'est pas évident à résoudre pour un ordinateur. Il s'agit d'un labyrinthe formé de murs en trois dimensions que l'utilisateur ne peut pas franchir en hauteur. Cela l'empêche de passer par-dessus les murs tout comme l'algorithme.

Les deux figures 3.20 et 3.21 montrent l'objet déplacé bloqué contre les bornes hautes et basses du labyrinthe. L'objet en déplacement est une flèche en trois dimensions.

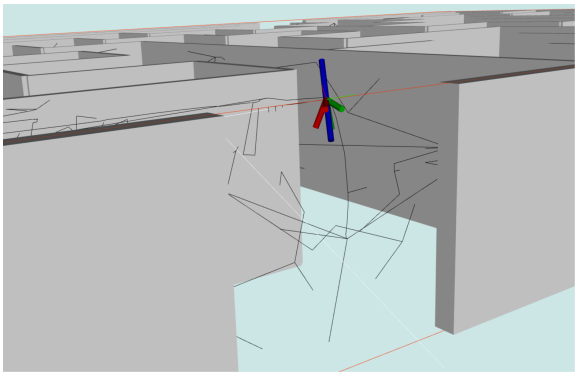


FIGURE 3.20 – Limitation plafond.

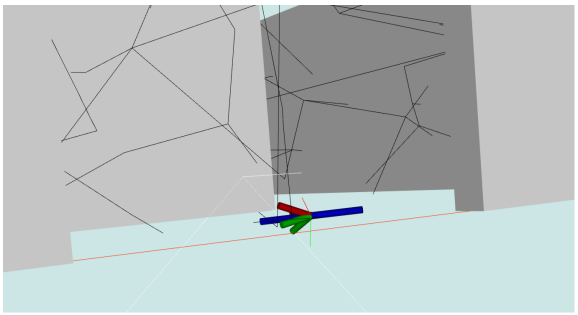


FIGURE 3.21 – Limitation plancher.

Résoudre un labyrinthe avec un algorithme de planification probabiliste peut prendre un temps assez long en comparaison du temps que prend un humain. Cela est dû au fait

que la plupart du temps, lorsqu'un nœud est ajouté à la *roadmap*, le chemin jusqu'à son plus proche voisin n'est pas valide. La deuxième raison est que la plupart des nœuds sont ajoutés à des endroits inintéressants car l'ordinateur échantillonne aléatoirement. En effet un labyrinthe possède majoritairement des zones d'espace libre mais un seul chemin résout le problème. Cela revient à un passage extrêmement long et étroit du point de vue espace des configurations. La probabilité est très petite de tirer aléatoirement un nœud qui puisse être connecté à la *roadmap* existante.

Enfin, lorsque les nœuds sont ajoutés hors collision et à un endroit utile, la plupart du temps, il existait déjà de nombreux nœuds présents à cet endroit ce qui réduit encore les performances.

A contrario, lorsqu'un humain guide le planificateur, ces phénomènes quoique n'ayant pas disparu, sont réduits autant que l'autorité est donnée à l'humain utilisant son intelligence pour accélérer le procédé.

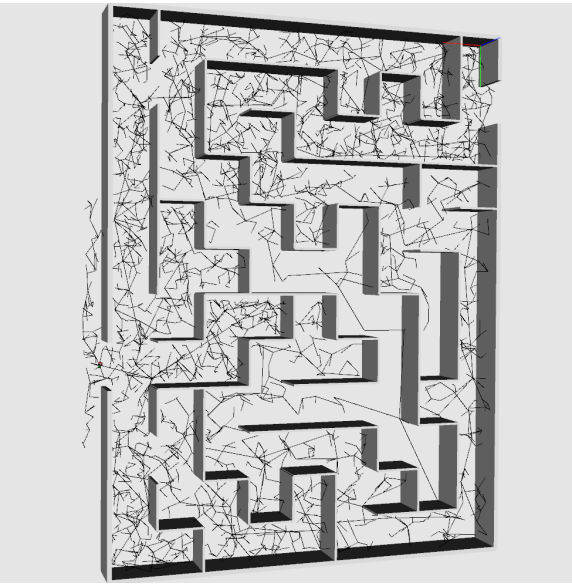


FIGURE 3.22 – Solution labyrinthe avec RRT.

Nous pouvons voir figure 3.22 que le RRT seul a pu résoudre le problème du labyrinthe. Il est clairement visible cependant que de nombreux arcs et nœuds sont redondants. En effet certains espaces ont plus de chances de contenir un nouveau nœud car proches d'un espace contenant déjà des nœuds. En second lieu, il existe de nombreux nœuds ajoutés dans des espaces inutiles car l'ordinateur ne peut pas savoir à l'avance où échantillonner.

La figure 3.23 montre le même problème résolu avec notre algorithme interactif. Non seulement les deux effets néfastes décrits ci-dessus ont beaucoup moins d'importance mais

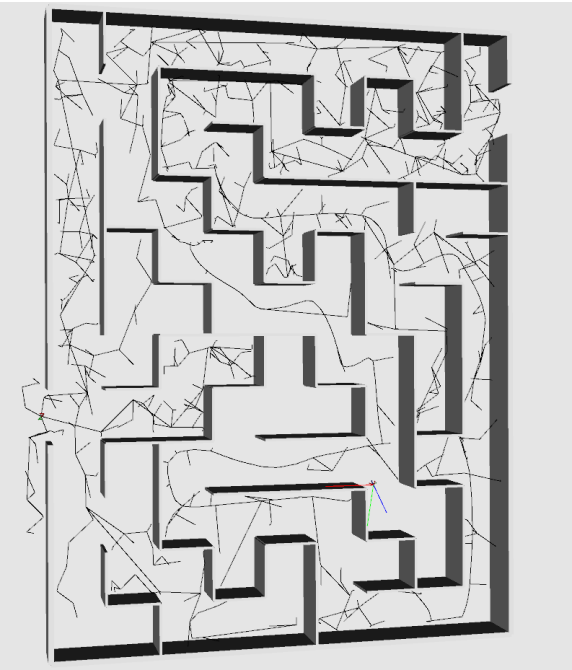


FIGURE 3.23 – Solution labyrinthe avec algorithme interactif.

en plus une solution sera trouvée beaucoup plus vite. Le RRT trouve une solution en trois minutes avec 3690 nœuds et 7378 arcs tandis que la deuxième expérience n'a pris qu'une minute avec 1453 nœuds et 2904 arcs avec une valeur de $\alpha = 0,5$.

3.6.2 Influence du paramètre alpha

L'aide d'un opérateur a radicalement changé la vitesse de résolution du problème. Très vite, lorsque des échantillons sont ajoutés dans un espace utile pour la recherche d'un chemin, les échantillons suivants ont tendance à être également ajoutés autour de ce dernier. La raison est qu'un nouveau nœud doit être connecté à un nœud valide pour être ajouté à la roadmap. La probabilité d'ajouter un nœud valide augmente dans les zones où il existe déjà d'autres nœud valides. On peut voir que dans la méthode interactive, le nombre de échantillonnés est beaucoup plus faible que dans la méthode RRT. En comparaison avec la méthode RRT, les performances de notre planificateur interactif I-RRT la dépassent radicalement. La méthode I-RRT génère moins de nœuds, d'arcs et résout le problème plus rapidement.

Le paramètre d'interactivité α permet la répartition du travail entre l'homme et la machine. Un algorithme trouvera facilement une solution à un problème de planification

dans un environnement faiblement encombré. À l'inverse dès que l'environnement devient très encombré ou que l'objet à déplacer doit emprunter des passages étroits, l'humain aura généralement plus de facilité que la machine à savoir où est le bon passage grâce à ses plus grandes capacités cognitives ou à sa connaissance métier.

Un environnement aura des zones dotées de caractéristiques propres. Certaines seront traitées efficacement par l'ordinateur, d'autres non. Un algorithme de planification se comportera différemment sur deux environnements différents. Il réagira aussi différemment selon la valeur du paramètre α.

Nous avons constaté qu'il existe une grande variabilité dans la vitesse de résolution du problème en fonction de ce paramètre pour un même opérateur. C'est-à-dire que toutes choses étant égales par ailleurs, la variation du paramètre α induit une variation dans la vitesse de résolution d'un problème.

En outre, deux opérateurs différents n'obtiendront pas les mêmes performances pour une même valeur de α car leurs performances dépendent de leur capacités cognitives mais aussi et surtout de leur entraînement.

Cela pose évidemment la question de la pertinence d'un α fixe car il est réglé en début de simulation et reste constant tout au long d'une même expérience.

3.6.3 Perspectives

Il serait souhaitable que ce paramètre évolue au fil du temps pour permettre de l'adapter à l'opérateur ou à l'environnement. Différentes alternatives au réglage fixe du paramètre α sont envisageables.

Une première possibilité serait d'afficher à l'écran un curseur allant de zéro à un, ainsi qu'un nombre représentant la valeur d'α. L'opérateur pourrait alors régler à sa guise la valeur de α. La simulation commencerait avec $\alpha = 0.5$ et l'opérateur pourrait modifier la valeur plusieurs fois au cours de la simulation en fonction du contexte applicatif ou des contraintes sur la planification.

Une autre possibilité serait que le paramètre α puisse s'adapter au fil du temps. Il serait alors possible de le modifier à chaque itération en fonction des mouvements de l'opérateur. On peut considérer par exemple, un accumulateur logiciel qui à chaque itération de l'algorithme ajoute à l'accumulateur, la vitesse de déplacement de l'utilisateur avec un certain gain, positif ou négatif. La valeur de l'accumulateur serait croissante au-delà d'une certaine vitesse et décroissante en deçà. La vitesse est un indicateur intéressant car elle témoigne de l'implication de l'utilisateur dans le processus interactif. S'il bouge beaucoup, il est raisonnable de penser qu'il le fait pour de bonnes raisons. Il faut alors lui donner plus de marge que s'il bouge peu.

3.7 Conclusion

Un exposé des méthodes existantes nous a permis d'illustrer les capacités des algorithmes interactifs et leur différences avec des méthodes totalement automatiques. Nous avons ensuite décrit la manière dont ces méthodes partagent l'autorité entre l'Homme et la machine et les différents outils matériels pour ce partage. Enfin, une analyse critique nous a permis de nous positionner face à l'état de la question et de proposer une méthode de planification de mouvement interactive originale.

Une des forces de notre méthode est sa simplicité de mise en œuvre avec de nombreux types de périphériques d'interaction. Cette méthode d'interaction dans l'espace libre permet la recherche dans l'ensemble de l'environnement gardant la complétude probabiliste. Elle permet un partage d'autorité réglable et donne d'excellents résultats par rapport au RRT, qui sont détaillés au chapitre 6.

On constate cependant qu'une opération d'assemblage implique par définition le contact entre les objets. Or, les techniques de planification de mouvement développées par la communauté évitent en général les obstacles. L'absence de contact peut devenir gênante lors d'opérations spécifiques telles que le glissement ou l'insertion. Le glissement peut également être utilisé pour guider la planification afin de rendre la tâche plus facile ou plus rapide à résoudre. Pour ces raisons, nous pensons qu'il serait très intéressant d'étudier une solution de planification de mouvement au contact des obstacles au lieu de chercher à toujours les éviter.

Chapitre 4

Contact et interaction

Nous avons présenté dans le chapitre précédent la planification de mouvement interactive dans l'espace libre. Notre objectif est de travailler sur l'assemblage de pièces pour les besoins du prototypage notamment. L'assemblage par définition implique le contact entre plusieurs pièces au moins pour le point d'arrivée. Il peut également arriver que le glissement de deux pièces l'une contre l'autre soit indispensable, comme lors d'une tâche d'insertion, ou simplement utile pour faciliter le processus. Ainsi une méthode permettant la génération de configurations au contact des obstacles serait une réponse à cette problématique.

Dans ce chapitre nous présenterons dans une première section ce qu'est le contact, les modes de contact entre un objet et un obstacle puis un état de l'art de la planification de mouvement au contact ainsi que notre positionnement par rapport à ces travaux. Les deux sections suivantes présenteront ensuite notre approche. Une méthode d'échantillonage au contact permettant de générer rapidement des configurations au contact entre un objet et un obstacle est présentée puis utilisée dans la section suivante pour réaliser de la planification interactive avec contact appelée I-RRT-C. Elle permet d'alterner des possibilités de planification dans l'espace libre et dans l'espace au contact par la mise en œuvre de notre méthode d'échantillonnage et de notre algorithme interactif présenté chapitre 3.

4.1 Contact et mouvement

Cette section présente d'abord l'utilité du contact pour le déplacement d'un objet. Elle détaille ensuite les types de contacts possibles dans un espace à trois dimensions, en translation et en rotation. Enfin, un état de l'art présente les contributions du domaine de la planification de mouvement au contact et notre positionnement par rapport à ces travaux.

4.1.1 Utilité du contact

La planification de mouvement au contact est utile dans de nombreux cas. Le contact a de nombreuses utilités que nous tenterons d'exposer ici, qu'il s'agisse de glissement ou d'insertion, d'opérations spécifiques nécessitant le contact ou d'utiliser le contact comme moyen pour accélérer l'exploration de l'environnement, réduire l'incertitude ou accéder à des mouvements autrement impossibles.

4.1.1.1 Opération de glissement

Le glissement est l'action de déplacer un objet au contact d'un obstacle. Nous pouvons facilement nous représenter l'utilité du glissement pour la manipulation d'un objet par exemple quand nous utilisons une brosse pour effacer un tableau.

Le glissement peut aussi être un moyen détourné pour réaliser une tâche. Nous voyons figure 4.1, un robot qui tente de porter un seau trop lourd. Le déplacer par glissement lui permet de rapprocher le seau pour ensuite le soulever. Dans cette situation le glissement est une étape indispensable.

Enfin, nous pouvons remarquer que le glissement permet dans certains cas de raccourcir les trajectoires en comparaison de celles entièrement dans l'espace libre.

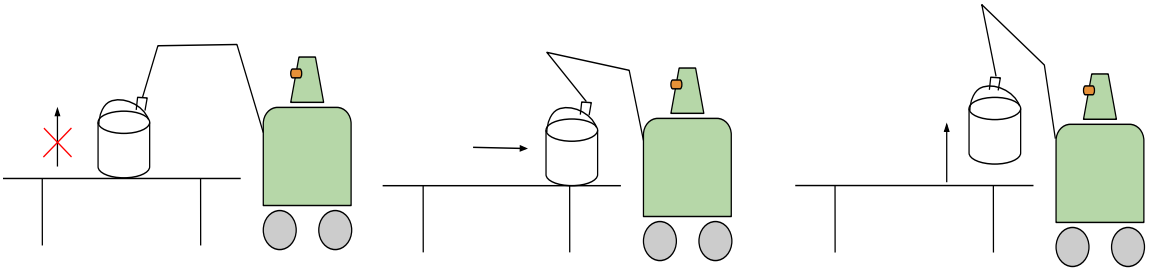


FIGURE 4.1 – Opération de glissement

4.1.1.2 Opération d'insertion

L'insertion est le fait d'introduire un objet dans un autre dans un emplacement prévu à cet effet. Il est généralement précédé par un glissement. C'est l'aboutissement d'un glissement à l'intérieur d'un objet qui donne une insertion. L'opération d'insertion est une tâche industrielle courante dans l'assemblage. Son intérêt est illustré ci-dessous avec un exemple.

Nous voyons figure 4.2 l'insertion d'une clef dans une serrure. La clef doit trouver une trajectoire depuis son point de départ (étape 1). Il lui est d'abord nécessaire d'entrer au contact avec l'obstacle (étape 2). Dans cette situation, un planificateur dans l'espace libre sera mis en échec à moins de lui donner un petit espace libre entre la clef et la serrure. L'étape 3 montre une opération de glissement pour se positionner au dessus de la serrure.

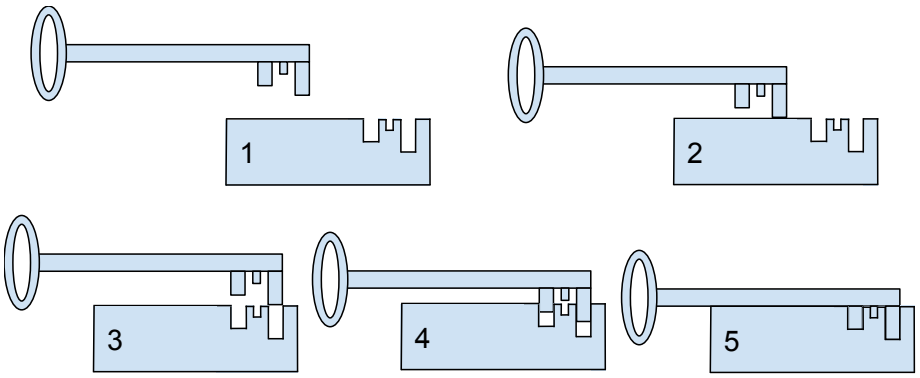


FIGURE 4.2 – Opération d'insertion

Nous voyons étape 4 se poursuivre le glissement en vue de l'insertion au point d'arrivée, étape 5.

4.1.1.3 Réduire l'incertitude

Nous pouvons nous servir de la planification au contact pour réduire l'incertitude sur l'environnement. Par exemple, un personnage tente d'insérer une clef dans une serrure sans l'aide de la vue. Il vient au contact puis se sert de la porte pour glisser en direction de la serrure. Il atteint son but beaucoup plus facilement que s'il fallait viser directement au bon endroit car il s'est servi du contact pour savoir qu'il était contre la porte. Il se sert ensuite du glissement pour trouver la serrure et enfin il insère la clef au bon endroit.

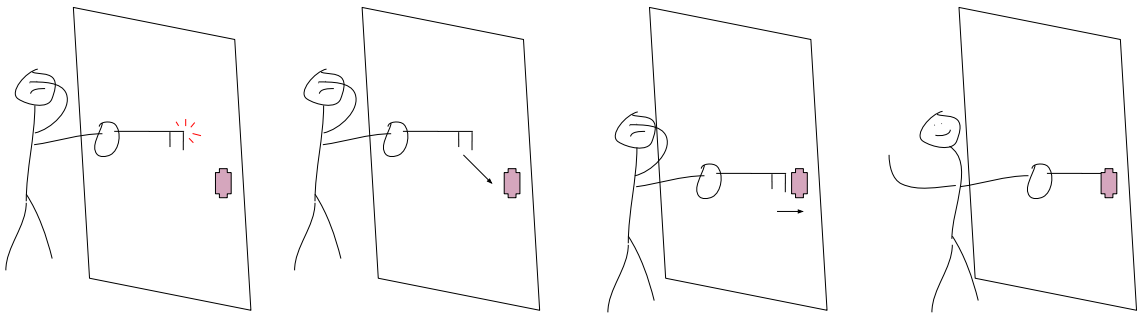


FIGURE 4.3 – Réduire l'incertitude : glisser et insérer sans vision.

4.1.2 Modes de contact

Lorsque l'on parle de contact, il est nécessaire de définir quel type de contact sera pris en compte. Dans notre cadre de travail, tous les solides sont constitués de facettes. Nos

modèles numériques ne comportent en effet que des faces planes. Il n'existe dans ce cas que des contacts ponctuels ou non ponctuels ce qui donne six différentes possibilités de contact dans un espace à trois dimensions.

4.1.2.1 Contact ponctuel

Il existe trois types de contacts ponctuels : le contact d'un point contre un point, le contact d'un point contre une droite, le contact d'un point contre une surface, voir figure 4.4

Si l'on veut planifier le mouvement d'un solide contre un obstacle et qu'on ne considère qu'un seul point de contact de ce solide, il est alors possible de considérer que ce point de contact de l'objet passe par une surface tangente au point de contact de l'obstacle ce qui revient à considérer que les trois premières possibilités de contact sont un seul et même cas.

4.1.2.2 Contact non ponctuel

D'autres possibilités existent pour le contact avec plus qu'un seul point. Il s'agit du contact d'une droite contre une droite toutes deux alignées, le contact d'une droite contre une surface et enfin le contact d'une surface contre une surface, voir figure 4.5.

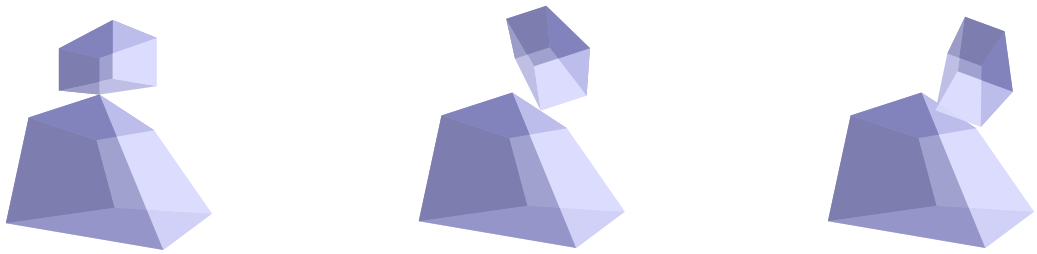


FIGURE 4.4 – Contact point contre point, point contre droite, point contre surface

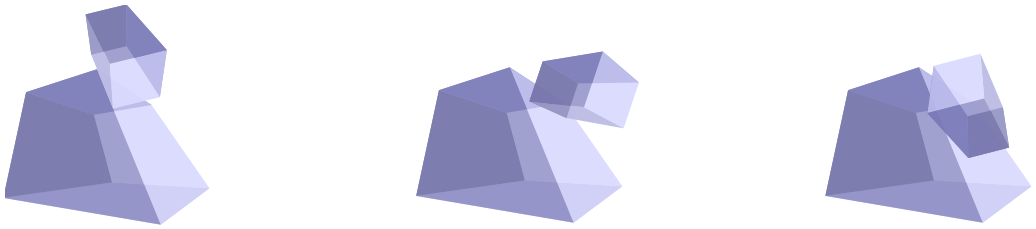


FIGURE 4.5 – Contact droite contre droite, droite contre surface, surface contre surface

4.1.2.3 Glissement au contact

Une fois un point de contact atteint, on peut glisser sur la surface en utilisant ou non les rotations pour garder une orientation fixe ou non à l'objet en glissement.

La première stratégie est de laisser libre la rotation de l'objet lorsqu'il est au contact. Cette stratégie est la plus couramment utilisée dans les travaux de planification de mouvement au contact.

Une autre possibilité est d'interdire toute rotation à l'arrivée au contact. Dans ce cas, l'orientation est définie avant d'entrer au contact et reste fixe pendant toute la durée de la planification au contact. En contrepartie de la réduction des degrés de liberté, cette stratégie permet de réduire les calculs pendant la planification au contact. Elle permet également de donner autorité à l'opérateur pour choisir une orientation spécifique pendant le contact si l'algorithme est interactif.

Une étude détaillée du mouvement au contact se trouve dans les travaux de Hirukawa [31].

4.1.3 État de l'art de la planification au contact

Ici sont présentées les principales contributions dans le domaine de la planification de mouvement au contact. Nous présentons d'abord les méthodes qui trouvent un contact en repoussant des portions de trajectoires de l'espace des obstacles vers l'espace au contact, que nous opposons aux méthodes qui trouvent un contact depuis l'espace libre vers l'espace au contact. Enfin, nous présenterons des méthodes de planification interactive de mouvement avec contact.

4.1.3.1 De $\mathcal{C}_{obst}$ vers $\mathcal{C}_{contact}$

La méthode la plus couramment utilisée dans le cadre de la planification de mouvement au contact est la planification de mouvement basée rétraction. Plusieurs algorithmes efficaces ont été développés et utilisant cette méthode. Cette méthode consiste à autoriser temporairement l'interpénétration entre les obstacles et OBJ puis à repousser les nœuds en collision hors de $\mathcal{C}_{obst}$.

Saha [62] trouve une première trajectoire avec un RRT en ayant réduit la taille des obstacles. Dans une seconde étape, un algorithme pousse les portions de trajectoire en collision hors des obstacles comme suit : lorsqu'une configuration q_r est en collision, il calcule une sphère centrée en q_r à l'intérieur de laquelle il échantillonne de nouvelles configurations. Le rayon de cette sphère croît progressivement jusqu'à qu'une configuration valide soit générée. Cette opération de déplacement de la configuration en collision est appelée la rétraction.

Zang [79] a donné la première définition formelle du problème de rétraction et propose un algorithme utilisant cette définition. Ayant une configuration q_r en collision, il cherche

le point q_m le plus proche de q_r et situé dans l'espace de contact $\mathcal{C}_{contact}$. L'algorithme s'approche ensuite itérativement de q_m.

La méthode précédente a été appliquée avec succès pour la planification de mouvement pour l'assemblage/désassemblage par Lee [48] avec un planificateur RRT rétractif et sélectif. Cette méthode choisit dans quel sens opérer la rétraction grâce à un test peu coûteux permettant de dire si la direction aléatoire choisie est judicieuse ou non. Grâce à une analyse en composantes principales, il détermine la meilleure direction de rétraction.

Les méthodes par rétraction apportent une bonne valeur ajoutée dans les passages étroits par rapport au RRT en accélérant nettement le processus. Cependant, ces méthodes ont tendance à échouer si l'environnement est fortement contraint. De plus, le besoin de plusieurs étapes de planification ralentit la tâche sans pour autant garantir d'obtenir des trajectoires valides.

4.1.3.2 De $\mathcal{C}_{libre}$ vers $\mathcal{C}_{contact}$

Une alternative à la rétraction est le contact direct sur la surface. Au lieu de passer à l'intérieur de l'obstacle et de repousser à la surface les portions en collision, l'idée est de venir directement de l'espace libre $\mathcal{C}_{libre}$ vers une surface appartenant à $\mathcal{C}_{contact}$.

Dans une première approche se servant du contact et donnant des résultats approchés, on trouve les contributions de Amato [2] et de Rodriguez [61]. Ces deux contributions se servent du contact lorsqu'une collision est détectée pour orienter l'échantillonnage dans des directions préférentielles. Ainsi, ils obtiennent des portions de trajectoires parallèles aux obstacles, parfois très proches du contact mais pas exactement au contact.

Redon [60] a proposé une technique très intéressante de planification locale. Soient deux configurations q_i et q_{i+1}. Supposons que l'arc (q_i, q_{i+1}) soit invalide car il existe une première configuration q_c en collision le long de cet arc. À partir de q_c, il considère qu'il existe m collisions entre le robot et les obstacles. Afin d'éviter les collisions, un déplacement élémentaire $d\boldsymbol{P}^i(q_c)$ au point de contact $\boldsymbol{P}^i$ doit satisfaire m contraintes de non pénétration. Il en déduit les mouvements admissibles. Cette méthode est efficace sans limitation dimensionnelle mais a le défaut d'être locale et de ne pas chercher explicitement le contact : elle cherche en fait à l'éviter.

Une contribution similaire est celle de Stilman [73]. Il décrit une méthode permettant d'échantillonner dans un espace tangent à une configuration en contact avec la surface d'un obstacle. Il dispose d'une configuration aléatoire q_{rand} et d'une configuration q_{near} au contact et proche de q_{rand} comme on peut le voir sur la figure 4.6. Il calcule ensuite q_s, une configuration à distance Δq de q_{near} sur le segment $[q_{near}q_{rand}]$. Il en déduit le déplacement sur l'espace tangent et trouve une nouvelle configuration q'_s projetée sur la surface qu'il peut ajouter à la *roadmap*. Le principal problème de cette méthode est qu'elle est coûteuse en temps de calcul.

La méthode de Stilman a été appliquée avec succès pour une méthode de planification de mouvement au contact par Sieverling [71]. La figure 4.7 montre un ensemble de confi-

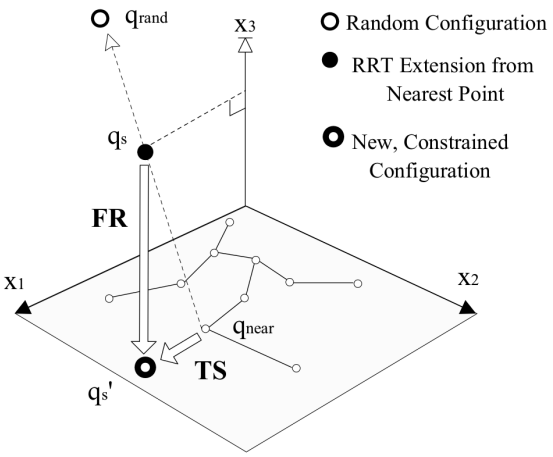


FIGURE 4.6 – Stilman génère des configurations au contact.

gurations x_s capable de rentrer dans un passage étroit grâce à un contact préalable sur la surface d'un obstacle. Ils alternent le mouvement au contact et le mouvement dans l'espace libre en fonction de la dernière action afin d'explorer l'espace libre et l'espace au contact. Cette méthode possède le même défaut que celle de Stilman sur laquelle elle s'appuie.

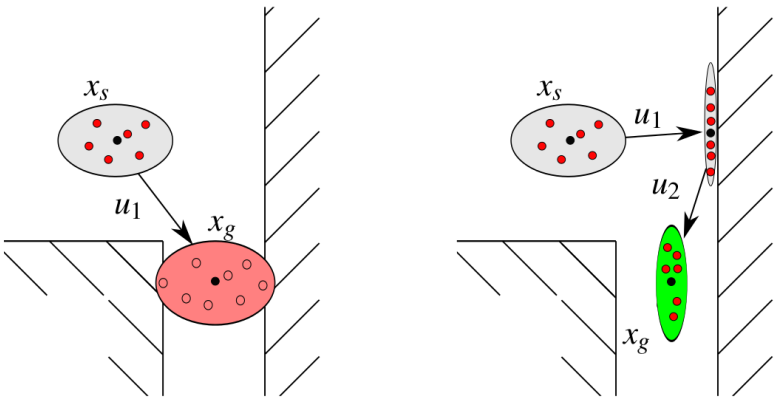


FIGURE 4.7 – Figure tirée de Sieverling [71]. À gauche sans contact, la probabilité est grande pour entrer en collision. À droite, avec contact, les configurations générées traversent le passage étroit sans problème.

4.1.3.3 Planification interactive au contact

La planification de mouvement interactive au contact nous permet de bénéficier à la fois d'un planificateur automatique explorant l'ensemble de l'espace libre et d'un opérateur pouvant guider la planification ou décider de venir au contact des obstacles pour des tâches spécifiques telles que le glissement ou l'insertion.

Cela permet de partager l'autorité s'agissant de l'exploration au contact. Plutôt que d'avoir un fonctionnement où une machine décide seule où et quand aller au contact, nous pouvons avoir un contrôle humain de ces tâches. À notre connaissance il existe très peu de contributions de planification interactive de mouvement avec contact.

Bayazit [6] propose une solution de planification de mouvement interactive se servant du contact mais dont le but n'est pas de générer des trajectoires au contact. Dans son cas, l'opérateur utilise un bras haptique et l'espace libre afin d'autoriser l'ajout de configurations en collision. Il dispose d'un algorithme de retour d'effort qui empêche l'utilisateur de rentrer dans les obstacles au delà d'un certain seuil fonction du retour d'effort. Par conséquent les nœuds en collision sont proches de la surface. Dans une deuxième étape de sa méthode, il isole pour chaque portion de trajectoire en collision la première et la dernière configuration valide et trace une droite entre elles. Il repousse alors ces nœuds en direction de cette droite. Cette méthode génère des trajectoires proches du contact mais pas exactement au contact, voir figure 4.8. Elle comporte aussi les défauts des méthodes de rétraction, comme décrit au paragraphe précédent.

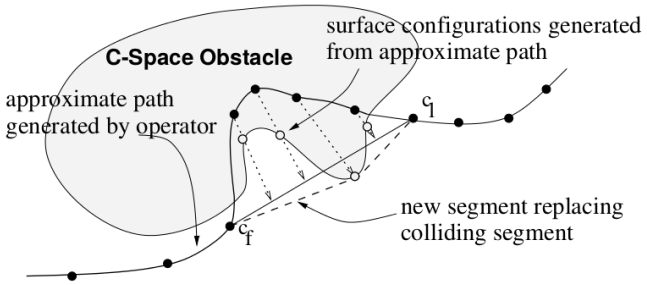


FIGURE 4.8 – Rétraction interactive proche du contact, Bayazit [6].

Yan [77] propose une solution de planification de mouvement interactive avec contact basée sur la rétraction. Cela rend cette méthode interactive. Dans une première étape, l'opérateur donne une trajectoire par le déplacement d'un objet dans l'espace de travail. Certaines portions de cette trajectoire sont en collision. Une seconde étape consiste à repousser les échantillons en collision hors des obstacles jusqu'à leur surface. Dans une troisième étape, il fait appel à un BiRRT pour connecter ensemble les configurations repoussées. Si le BiRRT ne parvient pas à les connecter, le processus est relancé en notifiant l'utilisateur. Cette méthode a le défaut de toutes les méthodes basées rétraction.

4.1.4 Approche

Nous avons choisi de mettre en œuvre un algorithme de planification de mouvement cherchant un contact de $\mathcal{C}_{libre}$ vers $\mathcal{C}_{obstacle}$. Nous nous interdisons toute configuration en collision. Nous souhaitons mettre en œuvre un algorithme capable d'échantillonner directement à la surface des obstacles en une seule étape. D'une part, cela permet d'accélérer le procédé en évitant le procédé itératif de rétraction des configurations en collision et, d'autre part, cela garantit la validité des trajectoires. En outre, nous gardons à l'esprit que nous souhaitons utiliser notre algorithme de planification au contact dans un cadre interactif.

Le mode de contact que nous avons choisi est ponctuel. Nous ne réorientons pas les objets à l'arrivée au contact. Nous avons choisi de laisser les rotations fixes pendant toute la durée du contact dans une première approche. Cela simplifie les calculs et nous permet de traiter plus facilement l'interaction au contact lorsque nous ferons de la planification interactive de mouvement en espace libre et au contact.

4.2 Échantillonnage au contact

Cette section présente notre contribution pour un échantillonnage au contact utilisant l'approche que nous avons choisie ci-dessus. Nous commençons par décrire notre méthode d'échantillonnage au contact puis notre proposition d'algorithme. Des exemples d'utilisation de cet algorithme terminent cette section.

4.2.1 Échantillonner au contact

Notre but est de trouver une méthode permettant d'échantillonner des configurations au contact d'un plan tangent à un point de contact. Cet échantillonnage est basé sur un changement de repère du repère monde au repère local au plan tangent. Nous voulons échantillonner sur le plan tangent au contact alors que la *roadmap* est construite dans le repère global.

Notre méthode est divisée en cinq parties :

— trouver un vecteur normal à la surface

— trouver un repère local à la surface

— trouver la distance à la surface du point échantillonné (différent du point de contact)

— tirer un échantillon

— changer de repère

4.2.1.1 1 : Trouver un vecteur normal à la surface

Afin de permettre l'échantillonnage au contact, il est nécessaire d'utiliser une librairie de détection de collision. Celle que nous utilisons possède une fonction de mesure de la

distance entre deux objets géométriques. Elle retourne la distance entre les deux points les plus proches de ces objets ainsi que les coordonnées de ces deux points. La plus petite distance est affichée à l'écran sous la forme d'un segment blanc reliant les deux points les plus proches entre le corps de l'objet et celui de l'obstacle comme indiqué dans l'exemple de la figure 4.9. Ce repère est une indication à l'utilisateur lui permettant de savoir sur quelle surface la planification au contact se fera.

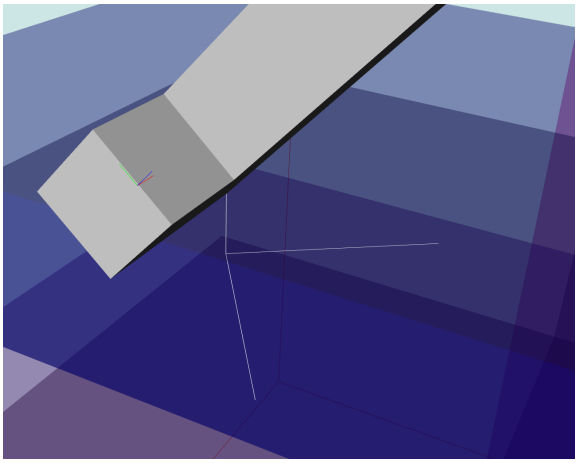


FIGURE 4.9 – Repère blanc affiché entre les deux points les plus proches.

Le point le plus proche sur l'objet est P_n, celui sur l'obstacle est P_o. Nous utilisons ces points pour en déduire le vecteur normal $\boldsymbol{n}_c$ à la surface d'échantillonnage Π, voir figure 4.10.

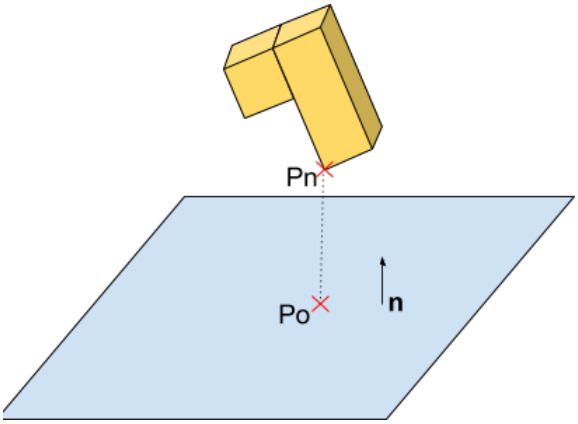


FIGURE 4.10 – Vecteur normal $\boldsymbol{n}$.

4.2.1.2 2 : Trouver un repère local à la surface

Nous utilisons pour trouver un repère local à la surface le processus d'orthonormalisation de Gram-Schmidt [66]. Cet algorithme permet à partir d'une famille de N vecteurs libres à N dimensions, d'orthonormaliser ces vecteurs les uns avec les autres. Cela veut dire qu'une fois l'algorithme terminé, chaque vecteur sera orthogonal à tous les autres vecteurs.

Autrement dit, $\forall$ vecteur $\boldsymbol{v}_i$ et $\boldsymbol{v}_j$ avec $i, j \in \mathbb{N}$, on a $\boldsymbol{v}_i.\boldsymbol{v}_j = 0$

Ce processus est initialisé avec le vecteur $\boldsymbol{n}_c$ porté par le segment $[P_oP_n]$. L'algorithme construit à partir de ce vecteur une base orthonormée qui est un repère local à la surface de contact dont on se servira à l'étape suivante, illustré figure 4.11.

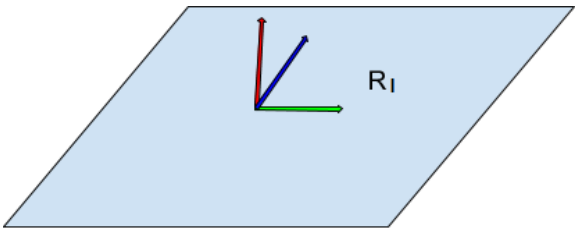


FIGURE 4.11 – $\boldsymbol{R_l}$ repère local.

4.2.1.3 3 : Trouver la distance à la surface du point échantillonné

Le repère attaché à OBJ a pour coordonnées P_c. Puisque l'objet à planifier au contact ne change jamais d'orientation, il existe une distance fixe entre P_c et le point de contact de l'objet P_n. Cette distance est appelée δ, voir figure 4.12. On a donc :

$$\delta = \boldsymbol{n}_c[P_nP_c] = constante \tag{4.1}$$

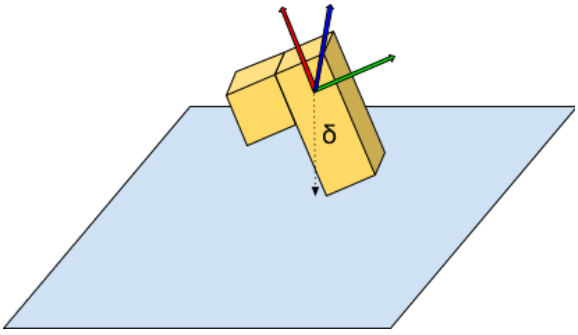


FIGURE 4.12 – Distance δ.

4.2.1.4 4 : Tirer un échantillon

La stratégie est la suivante : puisque le but est d'échantillonner aléatoirement sur une surface il suffit pour cela de tirer deux nombres aléatoires x et y pour échantillonner sur toute la surface.

Si maintenant on considère que le point à échantillonner est un sommet appartenant à un robot, le raisonnement est le même. Puisque le robot est un solide, donc possédant trois dimensions, il est nécessaire de fixer la troisième dimension z à un nombre fixe pour que le robot reste toujours à la même distance de la surface. En conclusion, on tire deux nombres aléatoires x et y et on a $z = constante$ car on ne considère que des surfaces planes.

4.2.1.5 5 : Changer de repère

Reste à savoir comment passer de la surface fictive où l'on échantillonne des configurations à la surface réelle de l'espace de travail qui a été choisie par l'opérateur. Cette opération est très simple car le repère local calculé grâce à l'algorithme de Gram-Schmidt est également une matrice de passage. Les trois axes de ce repère peuvent être considérés comme une matrice $\boldsymbol{R_l}$.

Tout point dans le repère local $\boldsymbol{R_l}$ possède également des coordonnées dans le repère monde $\boldsymbol{R_g}$, voir figure 4.13 . Si on a $\boldsymbol{p}_l$ un point quelconque dans le repère local alors son équivalent dans le repère global $\boldsymbol{p}_g$ est :

$$\boldsymbol{p}_g = \boldsymbol{R}_l \boldsymbol{p}_l \tag{4.2}$$

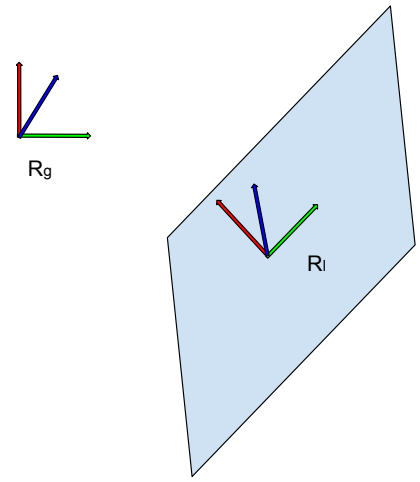


FIGURE 4.13 – Repère local, repère global.

4.2.1.6 6 : Exemple

Si un point de contact $\boldsymbol{p}$ a pour coordonnées dans le plan fictif :

$$\boldsymbol{p} = \begin{bmatrix} x & y & z \end{bmatrix}^t \tag{4.3}$$

Ses coordonnées $\boldsymbol{p}_{monde}$ dans le repère monde sont alors :

$$\boldsymbol{p}_{monde} = p\boldsymbol{R}_l \tag{4.4}$$

Nous avons supposé que l'orientation reste fixe pendant toute la durée de l'échantillonnage au contact. Cette orientation est définie par l'opérateur lors de la création du contact et est celle que l'objet avait avant l'entrée au contact. Les trois paramètres position sont calculés tel que suit :

$$q_x \leftarrow aléatoire \tag{4.5}$$

$$q_y \leftarrow aléatoire \tag{4.6}$$

$$q_z = 0 \tag{4.7}$$

$$\boldsymbol{p}_{local} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix} \tag{4.8}$$

Les coordonnées $\boldsymbol{p}_g$ de ce point dans le repère monde subissent une rotation :

$$\boldsymbol{p}_g = \boldsymbol{R}_l \boldsymbol{v}_{local} \tag{4.9}$$

Sachant que la configuration ajoutée à la *roadmap* est celle du repère attaché à l'objet, il nous faut translater ces coordonnées de δ pour garder au contact P_n et non P_c. On obtient au final :

$$\boldsymbol{p}_g = \boldsymbol{R}_l \boldsymbol{v}_{local} + \delta \tag{4.10}$$

Ainsi par une rotation puis une translation, il est possible de transformer les coordonnées locales d'un point du robot en coordonnées globales de ce même point puis d'effectuer une translation de ce même point pour que P_n reste au contact de la surface sélectionnée par l'opérateur, voir exemple en 2D figure 4.14.

4.2.2 Algorithme

L'algorithme 4 présente notre méthode d'échantillonnage au contact. Il prend en entrée le point de contact sur l'obstacle $\boldsymbol{P_o}$ et le point de l'objet le plus proche de l'obstacle $\boldsymbol{P_n}$. Il a également pour entrée la configuration q_{user} qui correspond à la configuration du périphérique d'interaction positionné par l'utilisateur. La variable N est le nombre d'échantillons au contact que l'algorithme doit calculer avant de se terminer.

L'algorithme 4 commence ligne 1 par l'appel à la fonction $TrouverRepèreLocal$. Elle prend pour arguments $\boldsymbol{P_o}$, $\boldsymbol{P_n}$ et q_{user}.

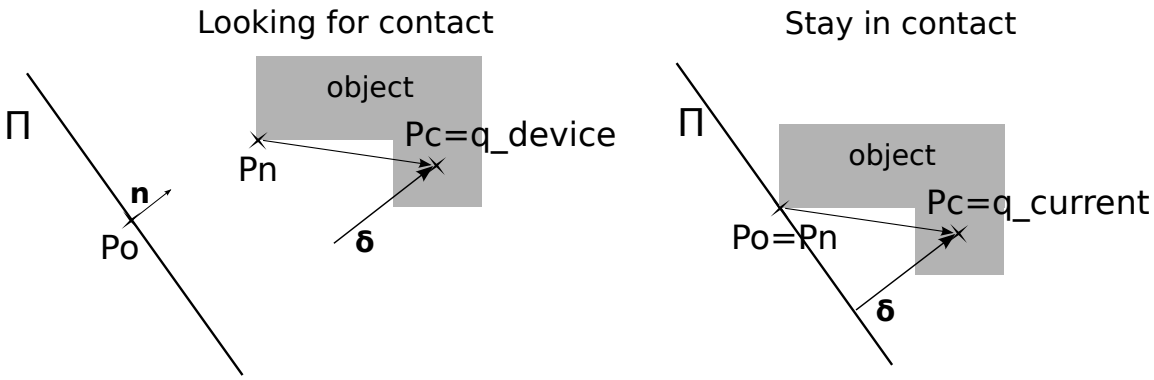


FIGURE 4.14 – Exemple en 2D

Algorithm 4 Contact Sampling

Require: $\boldsymbol{P_o}, \boldsymbol{P_n}, q_{user}, N, T$
1: $(\boldsymbol{\delta}, \boldsymbol{R}) \leftarrow TrouverRepèreLocal(\boldsymbol{P_o}, \boldsymbol{P_n}, q_{user})$
2: **for** $i < N$ **do**
3: $\quad q_{current} \leftarrow EchantillonnerAuContact(\boldsymbol{R}, \boldsymbol{\delta})$
4: $\quad T \leftarrow$ Add_tree($q_{current}$)
5: **end for**

Cette fonction commence par calculer $\boldsymbol{n}$, le vecteur normal à la surface de contact Π, de la façon suivante :

$$\boldsymbol{n} = \frac{\boldsymbol{P_n} - \boldsymbol{P_o}}{||\boldsymbol{P_n} - \boldsymbol{P_o}||} \tag{4.11}$$

Grâce au procédé d'orthonormalisation de Gram-Schmidt, la fonction cherche ensuite $\boldsymbol{R_l}$ le repère local à l'obstacle qui est aussi la matrice de passage d'un repère local au repère monde.

La fonction calcule ensuite $\boldsymbol{\delta}$ à l'aide de $\boldsymbol{P_n}$ et q_{user}.

À la ligne 2 commence une itération de N échantillonnages au contact. Ils sont calculés ligne 3 en faisant appel à la fonction $EchantillonnerAuContact()$. Cette fonction est détaillée dans la sous section précédente. Elle prend en arguments $\boldsymbol{R}$ et $\boldsymbol{\delta}$ et renvoie $q_{current}$.

Ligne 4, la nouvelle configuration $q_{current}$ est ajouté à la *roadmap*.

4.2.3 Exemples

Le fonctionnement de l'échantillonnage au contact est illustré ici à l'aide de trois exemples de planification au contact d'une surface, d'une droite puis d'un point. Dans le cas de l'échantillonnage d'un point contre une surface, la surface tangente est unique.

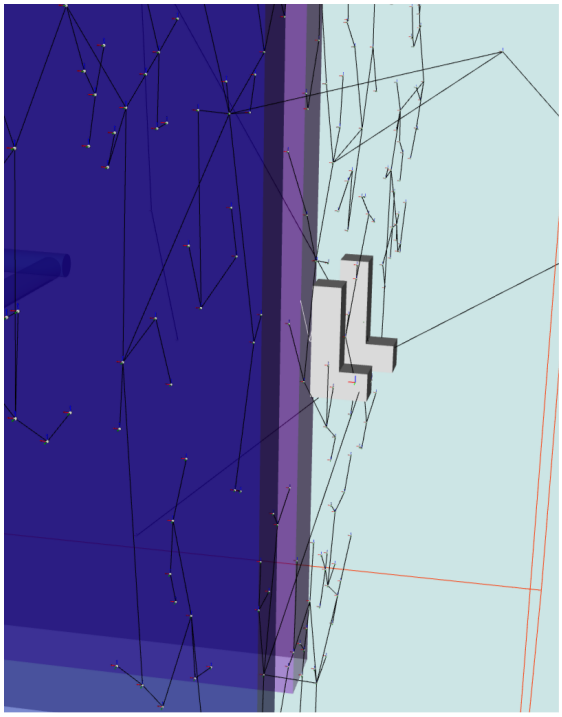


FIGURE 4.15 – Échantillonnage au contact.

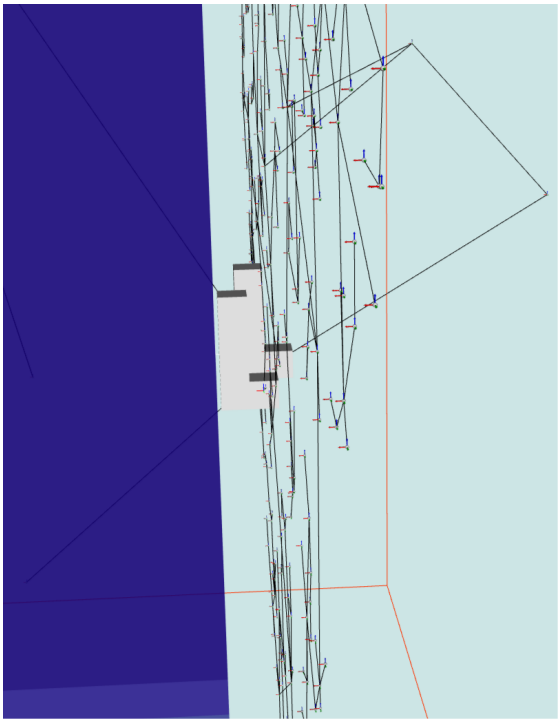


FIGURE 4.16 – Même scène avec point de vue différent.

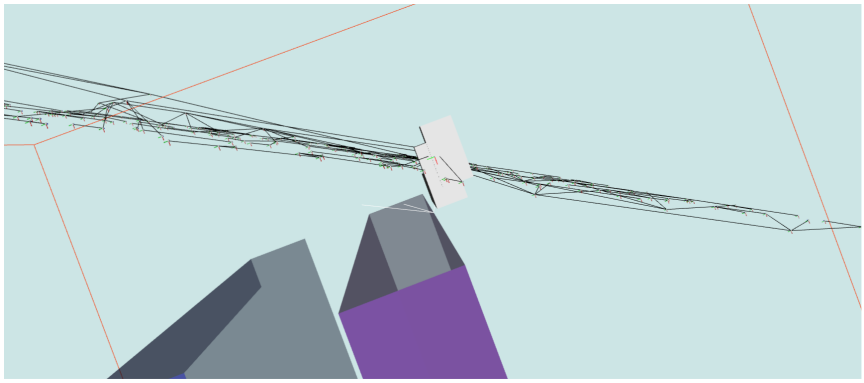


FIGURE 4.17 – Échantillonnage sur un plan passant par un segment.

Dans les deux autres cas, il existe une infinité de plan passant par un segment ou un point. Dans ces cas, le comportement est le suivant : le plan est défini par le vecteur $\boldsymbol{n}$. L'algorithme aura toujours le même comportement quelle que soit la situation.

Le premier exemple de contact d'un point sur une surface, figures 4.15 et 4.16, montre le même échantillonnage au contact de la surface plane en bleu, selon deux points de vue

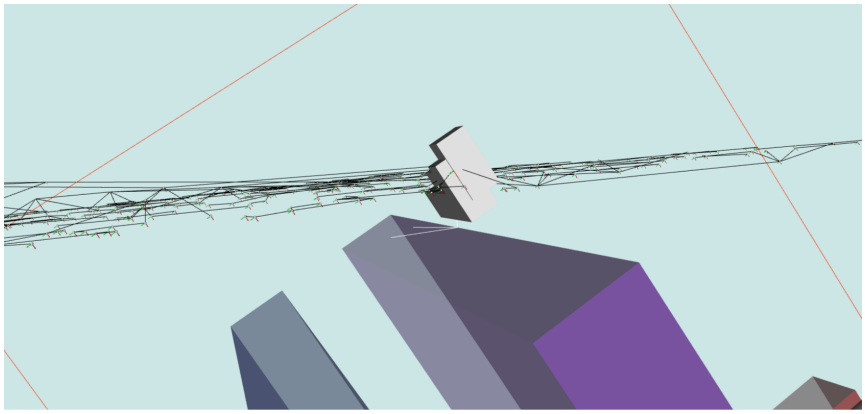


FIGURE 4.18 – Même scène avec point de vue différent.

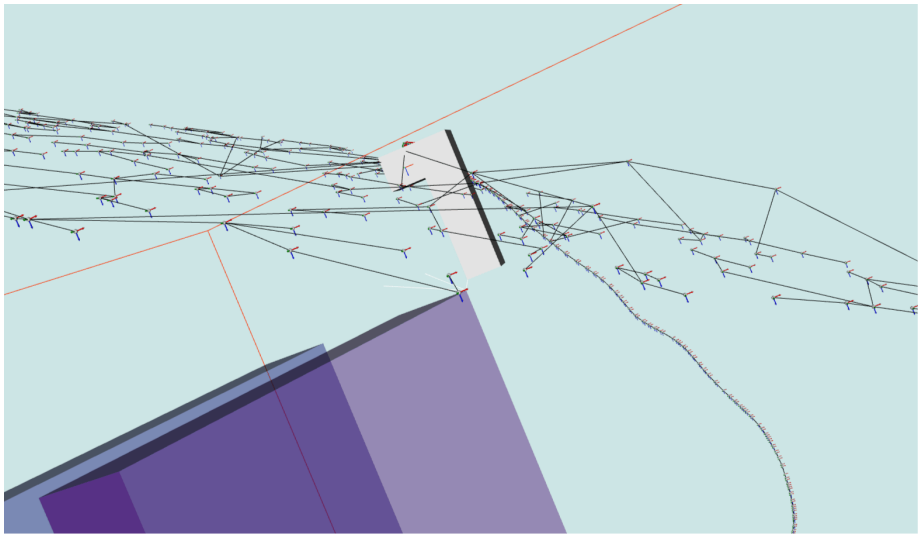


FIGURE 4.19 – Échantillonnage sur un plan passant par un sommet.

différents. Un grand nombre de configurations sont générées dans l'espace.

Le deuxième exemple expose le comportement de l'algorithme lorsque le plan de contact choisi n'est pas coplanaire avec la surface d'un obstacle mais seulement d'un segment appartenant à l'obstacle. Dans ce cas, l'échantillonnage se fait dans le plan tangent au segment. Les figures 4.17 et 4.18 illustrent cet exemple selon deux points de vue différents.

Le troisième exemple illustre le même mécanisme mais contre un sommet de l'obstacle. Des configurations sont générées de la même manière dans le plan traversé par le sommet choisi par l'opérateur. Les figures 4.19 et 4.20 illustrent cet exemple selon deux points de vue différents.

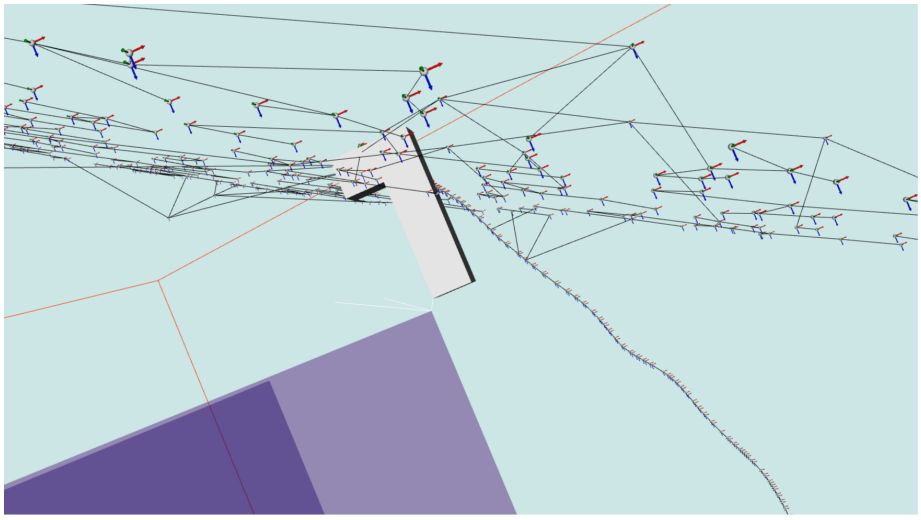


FIGURE 4.20 – Même scène avec point de vue différent.

4.2.4 Conclusion

Nous avons introduit une méthode d'échantillonnage au contact d'une surface plane capable de générer un grand nombre de configurations rapidement. Ces configurations ont une orientation fixe ce qui transforme le problème de placement d'un solide dans l'espace de dimension 6 à un problème de translation sur un plan de dimension 2. Cette solution d'échantillonnage est bien adaptée à une méthode de planification interactive de mouvement dans le but d'échantillonner au contact des obstacles car elle est compatible avec une interaction temps-réel et laisse l'algorithme calculer ce que l'opérateur aura du mal à effectuer : le suivi d'un contact.

4.3 Planification interactive de mouvement avec contact

Cette section présente notre algorithme de planification interactive avec contact qui utilise à la fois la planification interactive de mouvement dans l'espace libre présentée au chapitre 3 et l'échantillonnage au contact présentée précédemment.

Nous commençons par présenter les différentes modalités d'interaction possibles pour un tel algorithme. Dans un deuxième temps est présenté l'algorithme interactif avec contact appelé I-RRT-C. Un exemple illustratif sera détaillé à la fin de la section.

4.3.1 Modalités d'interaction

Afin de permettre à un opérateur et un algorithme de collaborer pour planifier tantôt dans l'espace libre, tantôt au contact, il nous faut définir les modalités d'interaction et

de partage d'autorité. Il existe en effet de nombreuses possibilités auxquelles nous avons réfléchi. Il nous faut savoir comment est détecté le contact, qui, de l'homme ou de la machine active et désactive le mode contact et qui se charge de la planification pendant ce mode.

4.3.1.1 Entrée en mode contact

L'entrée en mode contact peut se faire de différentes façons. L'opérateur déplace un objet dans l'espace libre au voisinage d'obstacles. Dans ce cas, le mode contact peut s'initier lorsque l'opérateur s'approche d'un obstacle. Un seuil à déterminer déclenche le mode qui a donc besoin d'un paramètre.

Une deuxième possibilité est d'attendre que l'opérateur entre en collision avec un obstacle. Le défaut de ce mode est que la détection du contact arrive trop tard, après la collision ; l'opérateur ayant depuis longtemps sélectionné une surface alors que le mode contact ne s'enclenche qu'à partir du moment où il entre en collision. L'opérateur a en effet très peu de chances de faire coïncider un point, segment ou face d'un objet contre la surface d'un obstacle s'il ne possède pas de périphérique avec retour d'effort. Dans cette situation, le seul retour est visuel, il faut donc se fier à la précision de l'affichage, du point de vue de la scène nécessairement partiel et de la précision du périphérique d'interaction. Le plus souvent, l'opérateur ne parvient pas exactement au contact mais est soit en dehors soit en collision. C'est donc le choix d'un périphérique d'interaction qui implique une solution.

Le mode contact peut être déclenché par une collision provoquée par la machine. Ce peut être une configuration en collision, à l'image de ce qui se produit parfois dans les algorithmes de rétraction. Ce peut être aussi une collision survenant lors de l'étape d'ajout d'un arc entre une configuration q_{rand} et une configuration q_{near}. Le défaut de ces approches est que l'opérateur n'a aucun contrôle sur la surface choisie.

4.3.1.2 Pendant le mode contact

Pendant le mode contact, la planification de mouvement est soit totalement aléatoire soit interactive, l'opérateur ayant dans ce cas une part d'autorité.

La rotation de l'objet est soit fixe soit variable pendant le contact. Les différentes possibilités sont les suivantes :

— la machine tire aléatoirement l'orientation de l'objet pendant le mode contact

— l'opérateur arrive au contact avec une certaine orientation et la machine la garde constante

— l'opérateur décide de la ou des orientations de l'objet pendant le contact

Laisser la machine générer des orientations aléatoires permet d'explorer plus de possibilités mais provoque plus collisions. Laisser l'opérateur décider d'une seule orientation confère la priorité à la décision de l'opérateur mais limite les possibilités.

Enfin, laisser l'opérateur choisir une ou plusieurs orientations pendant le mode contact permet d'explorer de nombreuses possibilités tout en donnant la priorité à l'opérateur mais nécessite soit un périphérique d'interaction adéquat tel qu'un bras haptique afin de donner des orientations valides, soit un algorithme permettant de décider de l'orientation en temps réel. Ces deux cas sont très différents

Dans le premier cas , un périphérique à retour d'effort et un algorithme de simulation des forces de friction sont utilisés. Au premier contact de type point contre point, un moment se crée en ce point de contact. L'objet viendra se maintenir contre l'obstacle d'abord contre un segment puis ensuite contre une face, selon les ordres de l'opérateur.

Dans le second cas, il faut un algorithme dédié à la recherche du contact d'un segment ou d'une surface, qui soit capable d'orienter l'objet automatiquement en fonction de sa géométrie et de celle de l'obstacle. Dans cette situation, l'homme a autorité pour choisir la surface de contact ainsi que l'orientation grossière de l'objet tandis que la machine a autorité pour l'orientation exacte de l'objet.

4.3.1.3 Sortie du mode contact

Enfin, la sortie du mode contact peut se faire en donnant autorité à la machine par exemple en quittant le mode contact la première fois qu'un échantillon au contact est en collision. Cela veut dire qu'il est en contact avec au moins une deuxième surface. Une autre possibilité est de laisser la machine tenter N échantillonnages, indépendamment de leur validité finale avant de quitter le mode.

Si l'autorité est laissée à l'opérateur, une possibilité est de mesurer la distance de OBJ à l'obstacle. Si la distance est supérieure à un seuil prédéterminé, le planificateur quitte le mode contact.

4.3.1.4 Choix spécifiques

Nous avons décidé de laisser l'autorité à l'opérateur pour l'entrée en mode contact car il saura interpréter l'environnement et déterminer quelles sont les surfaces dignes d'intérêt mieux que la machine. Nous proposons de mesurer une distance de l'objet à la surface et d'initier le mode quand cette distance est inférieure à un seuil. Ce seuil est choisi arbitrairement et sera étudié au chapitre expérimentations 6.

Pendant le mode contact nous laissons la machine explorer seule la surface car l'opérateur aura du mal à rester en contact avec ladite surface mais avec une orientation fixe car nous laissons la priorité à l'opérateur pour le choix de l'orientation. Nous disposons d'une souris 6D comme périphérique d'interaction ce qui nous empêche de laisser l'opérateur choisir l'orientation pendant le mode contact car sans retour sensoriel, il est probable qu'il choisisse des orientations en collision.

La sortie du mode contact se fait au bout de N itérations en contact afin de laisser la chance à la machine d'explorer suffisamment la surface choisie par l'opérateur. Si l'opérateur

n'a pas déplacé son objet, le seuil de distance sera de nouveau franchi, ce qui relancera le mode contact pour N nouvelles itérations.

Nous laissons donc l'opérateur décider manuellement du début du mode contact et s'il souhaite le relancer. Nous le laissons aussi décider manuellement de l'orientation de l'objet pendant ce mode.

4.3.2 I-RRT-C

L'algorithme 5 présente notre méthode d'échantillonnage au contact combinée avec notre algorithme de planification de mouvement interactive. Nous avons appelé cet algorithme interactif avec contact I-RRT-C pour *Interactive RRT in Contact*.

À chaque itération de l'algorithme, un nombre aléatoire a est tiré pour déterminer qui, de l'homme ou de la machine, obtiendra l'autorité. Si $a \leq \alpha$, l'autorité est donnée à la machine. Dans ce cas, une configuration q_{rand} aléatoire dans l'espace libre est tirée. Dans le cas contraire, $a > \alpha$, l'autorité est alors donnée à l'opérateur, c'est le mode utilisateur.

Dans le mode utilisateur, deux situations sont alors possibles déterminées par la distance de l'objet à l'obstacle le plus proche. Une fonction distance effectue un test de distance entre OBJ et chacun des obstacles. Les résultats sont classés et la fonction renvoie la plus petite distance. S'il existe une collision, la configuration portée par l'objet sera rejetée. Si la distance est supérieure ou égale à un seuil d prédéterminé alors cette configuration tente d'être reliée à la *roadmap*. Si enfin cette distance est inférieure à d, le planificateur entre en mode contact sinon il reste dans l'espace libre.

Dans le mode utilisateur dans l'espace libre, la configuration ajoutée à la *roadmap* est celle définie par l'opérateur à l'aide du périphérique d'interaction. Cette configuration se situe nécessairement dans l'espace libre.

Dans le mode utilisateur avec contact, la distance de OBJ à un obstacle est inférieure à d ; le planificateur considère donc que l'objet est très proche d'une surface appartenant à un obstacle. Cela veut dire que l'opérateur souhaite planifier au contact de cette surface. La configuration q_{user} est alors projetée sur la surface à explorer et l'algorithme y échantillonne N configurations. Au bout de ces N échantillons, le mode contact se termine. Une nouvelle itération de l'algorithme se produit alors. Par conséquent, aucun partage d'autorité n'existe pendant le contact : tous les échantillons seront générés automatiquement au contact si l'opérateur l'indique.

Ligne par ligne, le fonctionnement de l'algorithme est le suivant :

Ligne 2, la fonction Find_Nearest_Obstacle() retourne la paire de points les plus proches : $\boldsymbol{P_o}$ le point d'un obstacle le plus proche de $\boldsymbol{P_n}$ le point le plus proche sur l'objet.

Ligne 3, un nombre aléatoire entre 0 et 1 est choisi pour donner l'autorité soit à un opérateur (ligne 4), soit à la machine (ligne 11).

Ligne 5, un test sur la distance entre $\boldsymbol{P_o}$ et $\boldsymbol{P_n}$ détermine l'entrée en mode contact ou non. Si le test est valide, la fonction ContactSampling() est appelée, ligne 9.

Algorithm 5 Planification interactive avec contact : I-RRT-C

Require: $W, T, N, q_{user}, \alpha, d$

loop
  $a \leftarrow \text{rand}(0, 1)$
  if $a > \alpha$ **then**
    if $|\boldsymbol{P_o} - \boldsymbol{P_n}| \geq d$ **then**
      $(\boldsymbol{P_o}, \boldsymbol{P_n}) = \text{Find\_Nearest\_Obstacle}(q_{user})$
      $q_{current} \leftarrow q_{user}$
      $T \leftarrow \text{Add\_Tree}(q_{current})$
    else
      $\text{ContactSampling}(\boldsymbol{P_o}, \boldsymbol{P_n}, N, q_{user})$
    end if
  else
    $q_{current} \leftarrow \text{Random\_Shooter}()$
    $T \leftarrow \text{Add\_Tree}(q_{current})$
  end if
end loop

4.3.3 Exemple

Nous présentons ici un exemple de planification interactive au contact sur un plan puis nous discutons des paramètres liés à cet algorithme.

4.3.3.1 Contact contre un plan

La figure 4.21 montre deux tests successifs du mode contact en cinq étapes avec $\alpha = 0$.

Initialement, l'objet est dans l'espace libre (1). L'utilisateur déplace l'objet à l'aide de la souris 6D en laissant derrière lui une trajectoire constituée des nœuds et des arcs ajoutés à la *roadmap*.

Une fois arrivé suffisamment proche de la grande surface située au bas de la figure, le planificateur passe en mode automatique et échantillonne la surface en ajoutant rapidement un grand nombre de configurations au contact (2). On peut voir d'une part que les nœuds sont distribués aléatoirement sur la surface et d'autre part que les orientations sont toujours les mêmes. En effet, à chaque position est ajouté un repère, comme expliqué chapitre 3. Chacun de ces repères ayant toujours la même orientation, cela implique que toutes les configurations ont la même orientation.

À l'étape suivante (3), l'opérateur quitte le contact comme indiqué par les nœuds qu'il laisse derrière lui. Une fois dans l'espace libre, l'objet est tourné pour obtenir une nouvelle orientation que l'on peut voir en (4). Enfin, il revient au contact et le planificateur génère une nouvelle série de configurations au contact, toutes de même orientation (5).

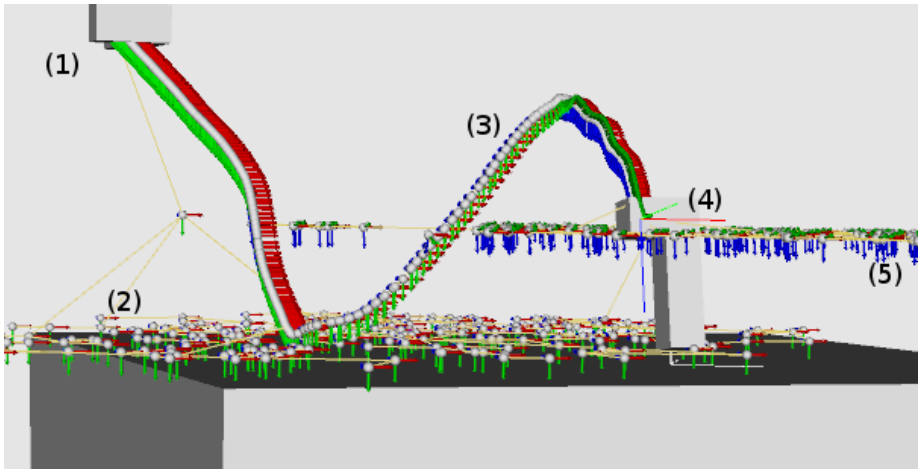


FIGURE 4.21 – Deux venues au contact différentes.

4.3.3.2 Influence des paramètres

Notre algorithme est paramétré par plusieurs variables que sont le paramètre d'interaction α, l'activation ou non du mode contact, sa distance d'activation d et le nombre N de nœuds échantillonnés au contact avant de quitter le mode contact. Chacun de ces paramètres influence la vitesse de résolution de l'algorithme, en plus de la spécificité de chaque environnement et de l'opérateur lui-même. Une discussion critique sera menée sur ces paramètres et leur influence dans le chapitre 6, Implémentation et résultats.

4.4 Conclusion

Ce chapitre a présenté la planification de mouvement au contact. Nous avons d'abord montré quel gains peuvent être tirés d'une telle planification par rapport à une planification totalement dans l'espace libre. Nous avons ensuite brièvement exposé les travaux existants dans ce domaine. Enfin nous avons détaillé les différents types de contacts en translation ou en rotation avant de positionner notre contribution d'algorithme de planification au contact.

Les méthodes de planification de mouvement au contact précédentes sont très coûteuses en temps en comparaison de celle présentée ici. Tant les méthodes par rétraction que celles par le calcul de la pseudo inverse de la jacobienne sont plus lourdes en temps de calcul à l'inverse de notre méthode qui génère rapidement beaucoup d'échantillons sur une surface de contact. En outre les méthodes par rétraction comportent un risque d'échec que nous n'avons pas.

Les principaux défauts de notre méthode sont d'une part l'absence de contrôle pendant le contact menant à des échantillonnages non pertinents. Ce défaut devra être réglé en

apportant un contrôle plus fin de la part de l'opérateur. D'autre part, le retour uniquement visuel des informations sur les collisions limite nos capacités de contrôle. Le périphérique d'interaction pourrait être remplacé par un bras haptique. Cela rendrait l'environnement virtuel plus immersif, permettant à l'opérateur de ressentir le contact afin d'accélérer la recherche de trajectoire.

Chapitre 5

Détection d'intention en réalité virtuelle

Dans ce chapitre, nous commençons par exposer ce qu'est la détection d'intention dans le cadre de la robotique. Nous proposons dans la deuxième section une méthode permettant la détection d'intention lors de la planification de mouvement au contact. Enfin, un nouvel algorithme appelé H-RRT-C utilise cette détection d'intention en réalité virtuelle dans le cadre de notre méthode de planification au contact.

Nous avons présenté dans le chapitre 4 une méthode pour planifier des mouvements au contact afin de permettre certains mouvement spécifiques tels que le glissement ou l'insertion. Nous avons ensuite proposé un planificateur interactif utilisant cette méthode. Nous avons déterminé que les principales faiblesses de cette méthode sont d'une part l'absence de retour haptique et d'autre part, l'absence de contrôle sur la planification au contact. Nous tâcherons de remédier à ces deux limitations dans le présent chapitre.

Ainsi exposerons-nous d'abord ce que sont la réalité virtuelle et la détection d'intention, le lien entre ces deux domaines étant de mettre en œuvre des méthodes d'interaction avec une machine de la manière la plus intuitive possible. Nous présentons ensuite une première contribution basée sur notre méthode de planification au contact, capable de s'adapter aux intentions de l'utilisateur pendant l'exploration du contact. Une deuxième contribution, appelée H-RRT-C pour "Haptic RRT with Contact" s'attachera à introduire une méthode pour l'utilisation immersive de cette méthode de planification avec détection d'intention en utilisant un bras haptique, une plate-forme de réalité virtuelle couplée à un planificateur interactif de mouvement.

5.1 Introduction

Nous exposons dans cette section ce qu'est la réalité virtuelle dont nous nous servirons pour les contributions décrites dans ce chapitre. Nous présentons ensuite l'intérêt de la

détection d'intention dans le cadre de la commande de robots puis un bref état de l'art des contributions du domaine. Est ensuite présenté un état de l'art de la planification interactive de mouvement avec utilisation d'un périphérique haptique car c'est le matériel que nous utiliserons pour notre contribution. Enfin, nous exposerons notre choix de périphérique d'interaction pour notre contribution de planification interactive de mouvement au contact avec détection d'intention.

5.1.1 La réalité virtuelle

5.1.1.1 Définition

On peut trouver une description détaillée de la réalité virtuelle, ses objectifs et ses méthodes dans le traité de réalité virtuelle [23]. Elle y est définie comme "domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3d, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudo-naturelle par l'intermédiaire de canaux sensorimoteurs".

5.1.1.2 Problématiques de la RV

Le traité de réalité virtuelle [23] explique que la réalité virtuelle a trois problématiques fondamentales illustrées par la figure 5.1. Elles sont "l'analyse et la modélisation de l'activité humaine en environnement réel et en environnement virtuel", "l'analyse et modélisation de l'interfaçage du sujet pour l'immersion et l'interaction dans l'environnement virtuel" et "la modélisation et réalisation de l'environnement virtuel".

Une personne en immersion et en interaction dans un environnement virtuel perçoit par ses sens naturels, décide et agit sur le monde virtuel ce qui peut être schématisé par une approche “perception, décision, action” .

La première problématique a pour objet l'analyse et la modélisation de l'activité humaine en environnement réel et en environnement virtuel, en bleu sur la figure 5.1. Il s'agit de le représenter et d'étudier ses comportements dans les simulations interactives. D'analyser le mouvement humain [25, 44], de concevoir des environnements [59] et des lois de commandes réalistes d'avatars et la synthèse de mouvements pour les humains virtuels [59, 75, 16]. Il s'agit enfin d'étudier les interactions entre humains réels et virtuels [46] et d'étudier des humains en immersion par exemple pour l'étude de phobies [20].

Une deuxième problématique, en orange sur la figure 5.1 est l'interfaçage des opérateurs pour l'immersion et l'interaction. Elle traite de la conception d'interfaces sensorielles, motrices ou sensorimotrices comme par exemple les systèmes de visualisation immersifs, les outils de capture de mouvements ou les bras haptiques [26]. On trouve par exemple les travaux du CEA-LIST réalisés avec les robots de la société Haption.

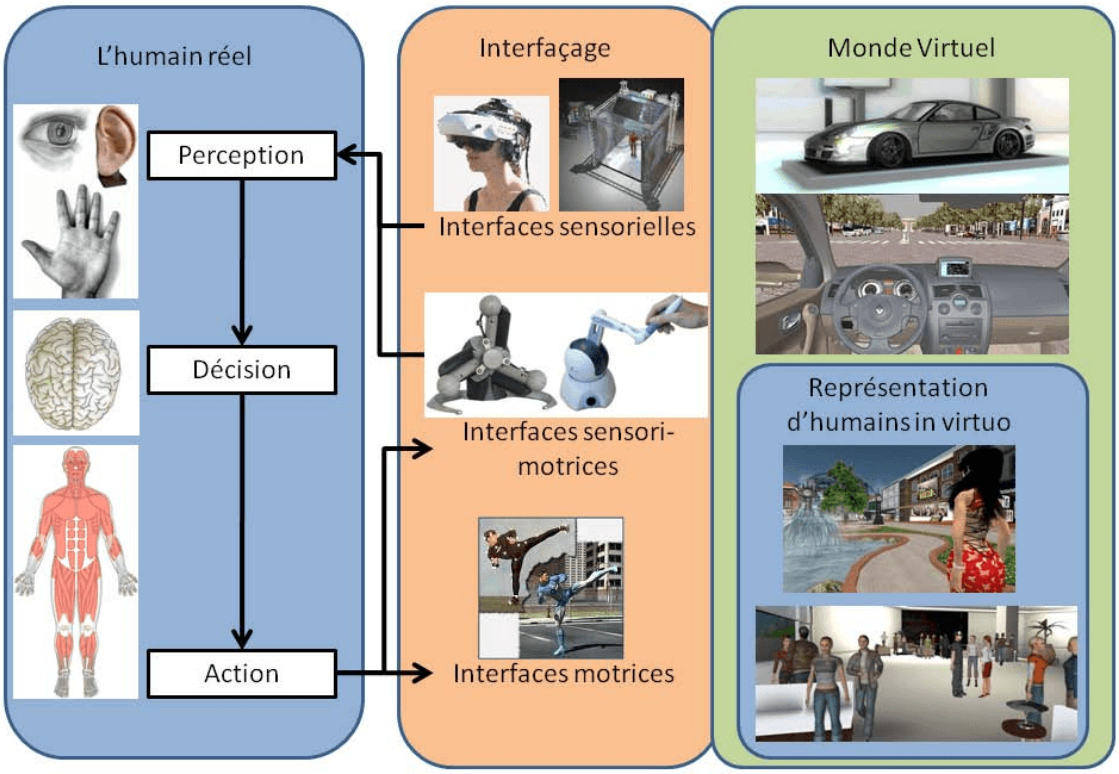


FIGURE 5.1 – Boucle de perception, décision, action.

La dernière problématique, en vert sur la figure 5.1 est la réalisation de fonctionnalités et la modélisation de rendus afin de rendre l'environnement virtuel le plus immersif possible. Il s'agit de mettre en œuvre des algorithmes temps réel afin que l'utilisateur ait une sensation de fluidité dans son interaction avec le monde virtuel. Les travaux de cette thèse ont pour objet d'étude cette dernière problématique.

5.1.1.3 Composition d'un système de réalité virtuelle

Un système de réalité virtuelle comporte nécessairement cinq caractéristiques [23].

Il doit être basé sur l'informatique. L'interface d'un monde virtuel avec un humain exploite le matériel informatique afin de créer une simulation dynamique réaliste. Les méthodes de la réalité virtuelles se retrouvent dans l'imagerie de synthèse, dans l'animation notamment.

Un tel système doit utiliser des interfaces, ou périphériques d'interaction. Ce sont soit des interfaces sensorielles informant l'utilisateur de l'évolution du monde virtuel, soit des interfaces motrices permettant d'agir sur ce dernier.

Il doit modéliser un environnement virtuel. Ce monde devient de plus en plus réaliste avec l'évolution des technologies. Les problématiques liées sont la modélisation, la numérisation et le traitement de données du monde virtuel.

Il doit interagir en temps réel avec l'utilisateur. Cette interaction sans latence permet à l'utilisateur de percevoir ce monde virtuel comme réel, malgré ses imperfections. C'est une caractéristique essentielle et une des contraintes les plus difficiles.

Enfin, une cinquième caractéristique est qu'il doit mettre l'utilisateur en état d'immersion autant que possible. L'immersion totale serait atteinte lorsque l'utilisateur ne percevrait pas de différence entre un monde réel et un monde virtuel.

5.1.2 Détection d'intention

La détection d'intention peut servir à commander un robot de manière plus intuitive que par un ordre direct. Dans ce cas, un algorithme avec détection d'intention comportera une couche algorithmique apportant une valeur ajoutée à la commande humaine. Elle peut être un mouvement complexe initié par une commande simple, ou un ensemble d'opérations qui humainement ne peuvent pas être commandées simultanément et nécessitent donc une intégration dans une couche logicielle. Il peut s'agir encore d'extrapoler les commandes de l'humain pour agir selon ses desiderata en termes de mouvements ou encore de réagir de manière autonome aux mouvements humains pour des fonctions d'accompagnement ou de sécurité. Il peut s'agir enfin d'accompagner un mouvement dans un cadre médical par exemple.

Nous présentons ici un bref état de l'art des contributions où la commande d'un robot se fait en détectant les intentions d'un humain.

La contribution de Shen [69] apporte une solution pour le mouvement des personnes en fauteuil roulant. L'utilisation classique d'un joystick empêche la personne d'utiliser ses deux mains ce qui forme un handicap supplémentaire. Par conséquent, il équipe cette personne de cinq centrales inertielles à neuf degrés de liberté (trois accéléromètres, trois gyroscopes, trois magnétomètres) positionnés sur son dos et ses bras. Elle peut ainsi contrôler le fauteuil roulant en bougeant naturellement le haut de son corps.

Wang [76] a développé une méthode pour le contrôle d'un robot jouant au tennis de table contre un humain. Dans un premier temps, il capture un ensemble de données de positions du joueur humain. Cet ensemble est utilisé pour entraîner la machine à reconnaître les différentes actions de l'humain et leur conséquence sur la position de la balle. Pendant l'expérience, un système de caméras observe la position de la balle et la position et l'orientation de la raquette de l'humain. Un bras robotique Barrett WAM réagit aux actions de l'utilisateur avant même qu'il ait touché la balle afin de lui permettre de se

déplacer aussi rapidement qu'il le faut. Cette méthode en ligne adapte le mouvement du robot en prenant en compte à la fois sa croyance initiale issue de l'analyse préliminaire et à la fois la détection de l'intention de l'adversaire par l'observation de ses mouvements.

Huang [34] contrôle un exosquelette pour bras droit à trois degrés de liberté ayant deux articulations au niveau de l'épaule et une au niveau du coude. Ce genre d'exosquelette est utile pour les personnes ayant perdu le plein usage d'un bras suite à un accident, il est utile aussi en tant qu'aide à l'effort. Le contrôle d'un tel système ne peut pas se faire à l'aide de commandes à mains, traditionnelles. Huang a donc développé un ensemble de capteurs de détection d'intention de l'utilisateur basé sur deux ensembles de quatre capteurs de pression (capteur FSR, force sensing resistor) positionnés sur le bras et l'avant bras à l'intérieur de coques attachées à l'exosquelette, voir figure 5.2. Ces capteurs détectent l'intention de l'utilisateur et contrôlent le bras en conséquence.

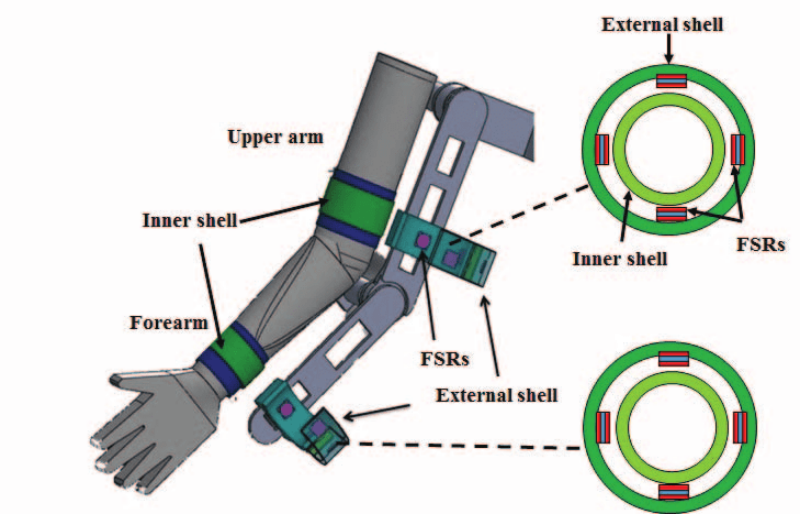


FIGURE 5.2 – Capteurs de pression pour exosquelette

Peternel [58] propose une approche en deux étapes pour le contrôle d'un bras robotique six axes en collaboration avec un humain. L'expérience menée est d'utiliser ce robot comme une aide pour le sciage d'une poutre en bois. L'humain tient une extrémité de la scie et le robot tient l'autre. Des capteurs électromyographiques disposés sur son bras droit mesurent l'effort fourni par l'opérateur. Le bras robotique lui même est de type collaboratif, muni de capteurs d'effort, à l'image d'un bras haptique. L'algorithme effectue la même tâche que l'humain et remplace celui-ci lorsqu'il fournit peu d'effort, ce dernier ne se chargeant plus que du guidage du bras.

Les méthodes citées ci-dessus utilisent des techniques de détection d'intention qui sont semblables à celles que nous voulons utiliser mais pour un domaine différent de celui qui nous intéresse. Nous souhaitons utiliser ces techniques pour explorer l'espace des configu-

rations dans le cadre de la planification interactive de mouvement.

5.1.3 Planification de mouvement haptique

Hassan [29] applique la méthode d'optimisation des colonies de fourmis pour trouver des trajectoires optimales dans le cadre de l'assemblage et du désassemblage. Cette méthode de planification de mouvement biomimétique fonctionne de manière itérative et distribuée. En un second temps, il guide un opérateur le long de la trajectoire en utilisant un bras haptique Phantom ® [50].

Brandt [11] contribue à la planification avec contrôle haptique et détection d'intention par un algorithme utilisé pour la commande d'un véhicule semi-autonome. La planification est de type champ de potentiel répulsif. Les obstacles sont analysés par les capteurs créant des zones à éviter. La conduite est interactive, le pilote ayant trois moyens d'influencer la planification : accélération/freinage, orientation de la voiture et intention de manœuvre. Lorsque l'interface de conduite détecte une intention de manœuvre, elle assiste haptiquement ce changement de trajectoire au travers du volant tout en prenant en compte les obstacles en replanifiant une nouvelle trajectoire.

Hou [32] propose une méthode interactive et utilisant un bras haptique pour la téléopération d'un drone volant. Son algorithme est basé sur une méthode RRT dont la configuration but est dynamiquement modifiée avec son interface haptique. Il utilise donc une stratégie de replanification fonction des commandes de l'utilisateur et de sa fenêtre de réaction afin d'éviter d'entrer en collision avec les obstacles. L'interface utilisée est un joystick à trois degrés de liberté. L'algorithme prend en compte les actions de l'utilisateur et adapte la position du joystick en fonction de la vélocité du drone.

Nous pouvons citer les travaux de Gang dans le domaine de la chirurgie haptique [24]. Ces travaux sont fondés sur l'utilisation d'un robot maître à trois degrés de liberté en rotation utilisé comme un bras haptique. Un robot esclave suit les commandes du maître avec de bonnes performances en contrôle et en retour haptique.

Nous évoquons dans ce paragraphe les contributions exposées dans les chapitres précédents mais traités sous l'angle de la fonction haptique. Bayazit [6] et Yan [77] utilisent un bras haptique dans le cadre de la planification de mouvement interactive. Ils se servent du bras pour dessiner des trajectoires inexactes que l'algorithme traite en un second temps, leur limitation étant d'utiliser une méthode basée sur la rétraction. Ladevèze [42] utilise le bras haptique comme périphérique de guidage de l'utilisateur, ce pour des questions d'apprentissage du mouvement. Le périphérique est en plus utilisé par l'utilisateur pour contraindre la machine à changer de trajectoire de guidage selon les ordres de l'utilisateur. Cailhol [14] se sert des mêmes fonctions dans son architecture multi-niveaux. Enfin, Flavigné [22] propose de biaiser la zone d'exploration du planificateur en prenant en compte l'intention de l'utilisateur mesurée comme une pseudo-force au travers d'un périphérique interactif, voir figure 5.3. Cette dernière méthode est celle qui se rapproche le plus de l'objectif qui nous intéresse. Sa limitation réside en ce qu'elle n'est pas prévue pour plani-

fier au contact des obstacles. De plus l'échantillonnage ne se déroule pas selon les ordres de l'opérateur mais suivant un compromis opérateur-ordinateur. Nous souhaitons que la planification se fasse selon les ordres de l'opérateur pour le choix de la zone à explorer.

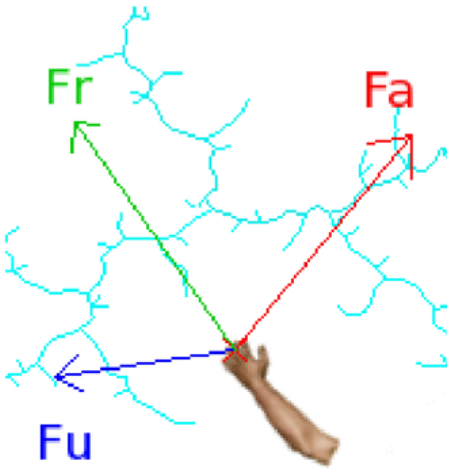


FIGURE 5.3 – Force attractive, utilisateur et résultante [22].

5.1.4 Choix spécifique

Dans le cadre de l'assemblage/désassemblage, il faut utiliser un périphérique ayant au moins six degrés de liberté afin de positionner et d'orienter un objet dans l'espace. I-RRT-C utilise une souris 6D non motorisée. La limitation principale de ce périphérique est l'absence de retour haptique, le choix du périphérique s'est porté sur un bras haptique.

Notre bras haptique est motorisé sur six degrés de liberté, ce qui permet de ressentir le contact de manière naturelle dans un monde virtuel. De plus, un bras haptique fournit une série d'informations produites par l'humain. Nous souhaitons nous servir de ces dernières afin de planifier des mouvements au contact beaucoup plus efficaces que ceux que nous avions produit jusque là.

5.2 Contact et force utilisateur

Cette première contribution apporte une réponse aux limitations de notre précédent algorithme, ***ContactSampling***. L'idée est d'utiliser les actions de l'utilisateur au travers d'un bras haptique pour s'y adapter en temps réel afin d'échantillonner au contact de manière plus naturelle.

5.2.1 Réduire la surface de contact

L'échantillonnage sur une surface permet d'effectuer des tâches spécifiques ou même de réduire la quantité de nœuds nécessaires pour trouver une solution à un problème. Cependant, la fait d'échantillonner sur une surface non bornée mène à des situations pathologiques. Un nœud aléatoirement généré a de grandes chances de se retrouver trop loin de la zone d'intérêt et risque en plus d'être en collision avec des obstacles. Si le nœud n'est pas en collision, il a beaucoup de chances d'avoir été généré dans une zone lointaine et inutile pour résoudre le problème. C'est une perte de temps dans les deux cas.

Pour limiter ces cas pathologiques, une solution est de réduire la surface d'échantillonnage au contact. La méthode ici choisie est de réduire la surface à un disque en première approximation. Il est nécessaire de garder un échantillonnage uniforme à l'intérieur de ce disque or la nouvelle fonction de répartition possède deux variables aléatoires ρ et θ. Des valeurs de ρ et θ choisies aléatoirement donnent un résultat semblable à celui de la figure 5.4. En effet, la densité de probabilité est indépendante de θ mais dépendante de ρ_e, la valeur aléatoire de ρ tirée pour un échantillon e. Plus ρ_e est grand, plus grand est le cercle

$$\int_0^{2\pi} \rho_e \, \mathrm{d}\theta \tag{5.1}$$

Par conséquent, plus une surface élémentaire $\mathrm{d}\rho\mathrm{d}\theta$ est éloignée du centre du cercle, plus sa densité de probabilité est faible. Afin de garder une densité de probabilité uniforme sur tout le disque comme illustré figure 5.5, il devient nécessaire de modifier la loi de probabilité selon la direction ρ.

La méthode finalement utilisée est la suivante :

Soit $rand()$, la fonction donnant un nombre aléatoire $k \in [0; 1]$ et soient deux variables ρ et θ,

$$\rho \leftarrow rand() \tag{5.2}$$
$$\rho \leftarrow \sqrt{\rho} \tag{5.3}$$
$$\theta \leftarrow rand() \tag{5.4}$$
$$\theta \leftarrow 2\pi\theta \tag{5.5}$$

Alors les coordonnées (x, y) du point de contact seront : $x = \rho cos(\theta)$ et $y = \rho sin(\theta)$.

Ce principe d'échantillonnage uniforme est utilisé avec des ellipses dans le reste de la section.

5.2.2 Intention utilisateur

Le retour d'effort permet de ressentir les obstacles, l'environnement, contrairement à la souris 6D utilisée jusqu'ici dans cette thèse. Ce retour d'effort augmente significativement l'efficacité de la recherche d'une trajectoire, en particulier dans les zones encombrées.

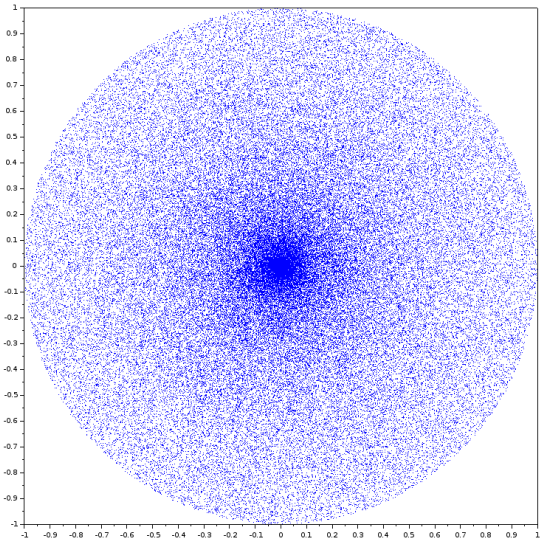


FIGURE 5.4 – Densité non uniforme.

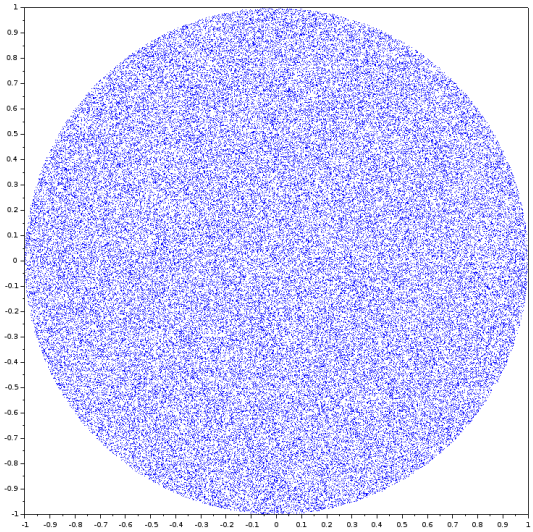


FIGURE 5.5 – Densité uniforme.

Disposant d'un bras haptique il est possible d'implémenter de nouveaux algorithmes utilisant les informations produites par l'opérateur pendant son utilisation. En effet, en plus d'obtenir des informations de position et d'orientation, nous pouvons obtenir des informations de la force appliquée à l'objet virtuel par l'utilisateur au travers de l'interface haptique.

Dans I-RRT-C, les configurations au contact étaient générées aléatoirement sur un plan tangent au point de contact. Ce plan était borné par les limites de l'espace de travail qui étaient trop grandes. Dans un nouvel algorithme que nous nommons ***HapticSampling***, les configurations au contact sont échantillonnées à l'intérieur d'une ellipse $\mathcal{E}$ centrée au point de contact afin de limiter la région d'échantillonnage. Il apparaît alors nécessaire de définir les paramètres de l'ellipse : sa surface, son élongation, son orientation. La position des échantillons est contrainte autour du point de contact défini par l'utilisateur, l'origine de $\mathcal{E}$. Cela permet de grandement réduire les configurations en collision ou celles qui sont loin de la zone qui intéressent l'utilisateur.

Lorsqu'il entre en collision avec un obstacle, l'utilisateur ressent une force de réaction grâce à l'utilisation d'un bras haptique et d'un algorithme de physicalisation fourni par le constructeur du bras. Il est alors possible de lire cette force et de l'utiliser pour modifier la planification en plus de la lecture de la position et de l'orientation du robot. La planification est modifiée par la lecture de trois paramètres fournis par le bras haptique :

— $\|\boldsymbol{f_u}\|$ détermine sa surface
— φ définit le ratio entre ses deux axes
— θ définit son orientation

5.2.2.1 La norme de la force utilisateur $\|\boldsymbol{f_u}\|$

La force utilisateur $\|\boldsymbol{f_u}\|$ est la force fournie par l'utilisateur qu'il soit dans l'espace libre ou en collision. Cette force est très faible s'il est dans l'espace libre mais non nulle. Elle représente la force tridimensionnelle appliquée par l'utilisateur sur l'objet virtuel OBJ et permet son déplacement dans l'espace. Dans le cas où l'objet n'est pas dans l'espace libre mais en collision dans l'espace de contact, cette force $\|\boldsymbol{f_u}\|$ croît lorsque l'utilisateur pousse plus fort l'objet contre un obstacle, voir figure 5.6.

S'il touche légèrement un obstacle puis quitte le contact, il est raisonnable de penser qu'il ne s'intéresse que peu à cet obstacle ou qu'il l'a touché par erreur. En revanche, si l'utilisateur pousse fortement l'objet contre un obstacle, nous admettons qu'il a choisi cette surface intentionnellement car il souhaite planifier un mouvement à son contact. Par conséquent, la surface de $\mathcal{E}$ est définie proportionnellement à $\|\boldsymbol{f_u}\|$ qui représente l'intérêt de l'utilisateur pour la surface en question.

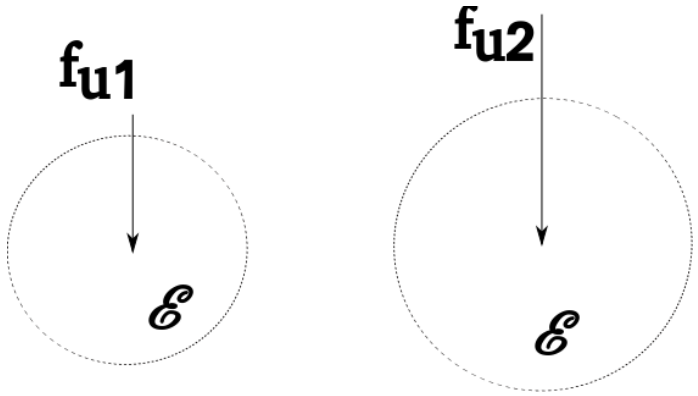


FIGURE 5.6 – Force utilisateur

5.2.2.2 L'angle φ à la normale

L'angle φ à la normale $\boldsymbol{n_c}$ de la surface de contact Π exprime l'intention de l'opérateur de se déplacer. Lorsqu'il entre en collision avec la surface d'un obstacle, notre algorithme calcule la normale $\boldsymbol{n_c}$ à la surface au premier point de contact. Si l'angle φ à cette normale est nul alors la force appliquée par l'utilisateur est orthogonale à la surface Π. Nous n'avons dans ce cas aucune information sur les intentions de l'utilisateur. Dans ce cas, l'angle est nul, l'axe mineur et majeur sont de même dimension et l'ellipse devient un disque.

En revanche, si l'utilisateur glisse le long de la surface, l'algorithme de physicalisation génère des forces de frottement qui induisent un angle entre φ et $\boldsymbol{n_c}$. La mesure de cet angle permet de déterminer les intentions de l'utilisateur. L'algorithme s'adapte en fonction de cet angle en changeant la surface d'échantillonnage d'un disque à une ellipse. Cela donne donc un peu plus de configurations le long d'une direction principale, dans la direction de

l'axe majeur de l'ellipse. L'axe mineur rétrécit dans le même temps réduisant le nombre de configurations dans la direction de l'axe mineur. Ce phénomène se poursuit jusqu'à que l'ellipse soit confondue avec une droite quand φ tend vers $\frac{\pi}{2}$. Les figures 5.7 et 5.8 illustrent ce mécanisme.

L'angle φ est calculé de la sorte :

$$\varphi = acos \frac{\boldsymbol{f_u}.\boldsymbol{n_c}}{\|\boldsymbol{f_u}\| * \|\boldsymbol{n_c}\|} \tag{5.6}$$

Avec s_a et b_a, la longueur de l'axe mineur et de l'axe majeur respectivement, nous calculons leur longueur de la manière suivante :

$$b_a = exp(\varphi) \qquad s_a = 1/b_a \tag{5.7}$$

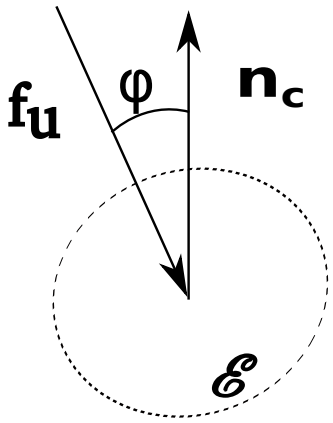


FIGURE 5.7 – Petit angle φ.

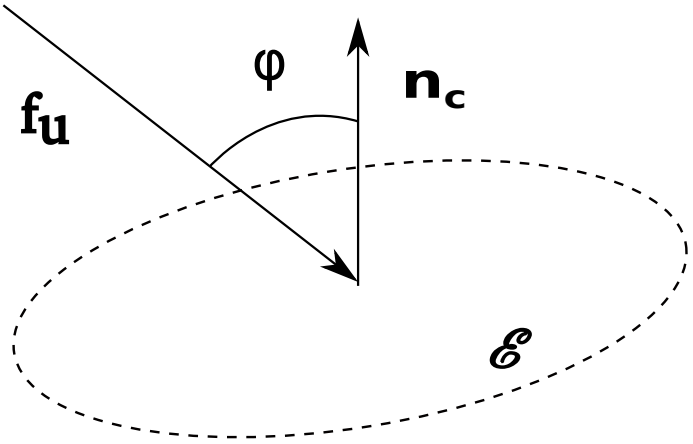


FIGURE 5.8 – Grand angle φ.

5.2.2.3 Le vecteur déplacement de position $\boldsymbol{\Delta}_p$

Il indique dans quelle direction l'utilisateur est actuellement en train de se déplacer avec p pour position. Sa projection sur la surface de contact est Δ_t avec t pour tangent. L'axe majeur de $\mathcal{E}$ est aligné avec ce dernier. Cela permet d'échantillonner dans la direction choisie par l'opérateur, voir figure 5.9.

Le vecteur $\boldsymbol{\Delta_p}$ est calculé à partir de la soustraction entre deux positions consécutives de OBJ. Nous utilisons $\boldsymbol{t_c}$, vecteur tangent à la surface de contact et $\boldsymbol{\Delta_p}$ pour calculer l'orientation θ de l'ellipse. D'abord, nous projetons $\boldsymbol{\Delta}_p$ sur la surface :

$$\boldsymbol{\Delta}_t = \boldsymbol{\Delta_p} - \boldsymbol{\Delta_p}.\boldsymbol{n_c} * \boldsymbol{n_c} \tag{5.8}$$

Puis nous calculons θ :

$$\theta = acos \frac{\boldsymbol{\Delta}_t . \boldsymbol{t_c}}{\|\boldsymbol{\Delta}_t\| * \|\boldsymbol{t_c}\|} \tag{5.9}$$

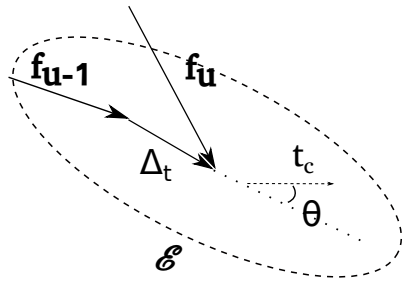


FIGURE 5.9 – Le mouvement utilisateur oriente l'ellipse

5.2.2.4 Paramètres de l'algorithme

Les trois paramètres φ, θ et $\|\boldsymbol{f_u}\|$ ont chacun une influence sur la vitesse d'exécution de l'algorithme. Ils sont de plus eux même paramétrés par avance avec des gains ou d'autres paramètres.

Le gain est de K_f pour $\|\boldsymbol{f_u}\|$. Il détermine la taille en mètres du rayon de $\mathcal{E}$ lorsque $a = b = 1$. Avec r le rayon du disque non encore élongé, on a donc :

$$r = K_f * \|\boldsymbol{f_u}\| \tag{5.10}$$

Ce gain influence les performances de l'algorithme en fonction de l'environnement.

Le gain K_φ pour le paramètre φ influence la manière dont évolue l'élongation de l'ellipse. On calcule donc a, l'axe majeur et b, l'axe mineur de la sorte :

$$b_a = K_\varphi * exp(\varphi) \tag{5.11}$$

$$s_a = 1/b_a \tag{5.12}$$

Enfin, le paramètre Δ_p est calculé par la soustraction de deux configurations consécutives. Utiliser plus de deux configurations peut faire office de filtre passe bas permettant de lisser le mouvement de l'utilisateur pour éliminer les mouvements parasites mais réduisant la réactivité de l'algorithme.

5.2.3 Algorithme

Nous présentons ici les détails de notre algorithme d'échantillonnage haptique ***HapticSampling***.

Ligne 2, nous lisons les informations données par le bras haptique que sont $\boldsymbol{f_u}$, la force utilisateur, $\boldsymbol{n_c}$, la normale à la surface de contact, $\boldsymbol{\Delta_p}$ le vecteur variation de position de l'utilisateur et $\boldsymbol{t_c}$ une tangente à la surface de contact.

Algorithm 6 HapticSampling

Require: W

1: **if** contact **then**
2: $\boldsymbol{f_u}, \boldsymbol{n_c}, \boldsymbol{\Delta_p}, \boldsymbol{t_c} \leftarrow$ HapticArm
3: $\varphi = angle(\boldsymbol{f_u}, \boldsymbol{n_c})$
4: $\theta = angle(\boldsymbol{\Delta_t}, \boldsymbol{t_c})$
5: Compute $\mathcal{E}(\varphi, \theta, \boldsymbol{f_u})$
6: $q_{current} \leftarrow$ContactSampling($\mathcal{E}$)
7: return $q_{current}$
8: **end if**

Ligne 3, nous calculons φ qui est l'angle entre $\boldsymbol{f_u}$ et $\boldsymbol{n_c}$ puis ligne 4 θ qui est l'angle entre $\boldsymbol{\Delta_t}$ et $\boldsymbol{t_c}$.

La ligne 5 utilise φ, θ et $\boldsymbol{f_u}$ pour calculer les caractéristiques de l'ellipse $\mathcal{E}$: $\|\boldsymbol{f_u}\|$, φ et θ.

Ayant défini une forme pour l'ellipse, la méthode ***ContactSampling*** est appelée ligne 6 qui échantillonne à l'intérieur de $\mathcal{E}$. Cette configuration aléatoire $q_{current}$ est renvoyée en sortie d'algorithme. La méthode ***ContactSampling*** est celle décrite au chapitre précédent.

5.2.4 Exemples

Les exemples qui suivent présentent la façon dont les trois paramètres $\|\boldsymbol{f_u}\|$, φ et θ sont utilisés pour échantillonner au contact de la surface d'un obstacle. Les situations décrites illustrent le fonctionnement de chaque paramètre. Dans chaque exemple, l'utilisateur pousse un cube contre un mur.

La figure 5.10 présente un exemple où l'utilisateur pousse légèrement le cube contre le mur. Les configurations aux contact sont groupées dans une petite surface autour du point de contact. L'utilisateur pousse de manière orthogonale. Par conséquent, l'ellipse est pratiquement un disque.

La figure 5.11 présente un cas où l'utilisateur a aussi poussé le cube orthogonalement à la surface. En revanche il a appuyé plus fort à l'aide du bras haptique. Par conséquent, l'ellipse est toujours quasi ronde mais beaucoup plus grande.

La figure suivante, figure 5.12 illustre un cas où l'ellipse est allongée. Cela est dû à un grand angle φ entre le vecteur force utilisateur $\boldsymbol{f_u}$ et la normale au contact $\boldsymbol{n}_c$.

Enfin, la figure 5.13 est le résultat de deux situations différentes. La première est un cas où l'utilisateur pousse l'objet légèrement avec un angle non nul et en un second temps, il pousse bien plus fort au même point avec un angle non nul aussi. Le résultat visible est une première ellipse entourée d'une seconde bien plus grande. Les bords de l'ellipse sont ici entourés de pointillés pour une question de clarté, étant entendu que ces pointillés ne

s'affichent pas pendant l'expérience.

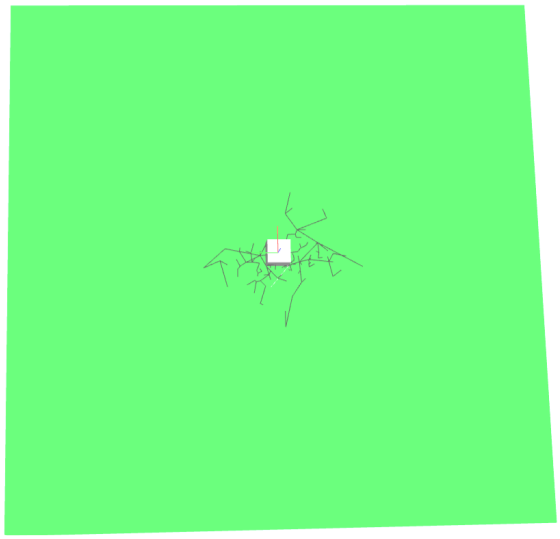


FIGURE 5.10 – Faible force.

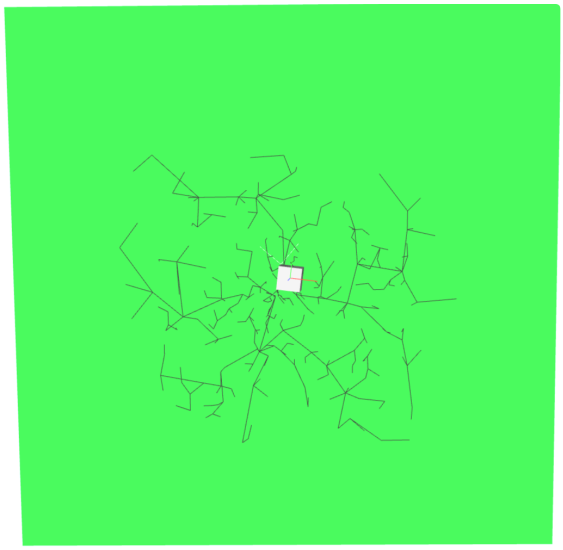


FIGURE 5.11 – Grande force.

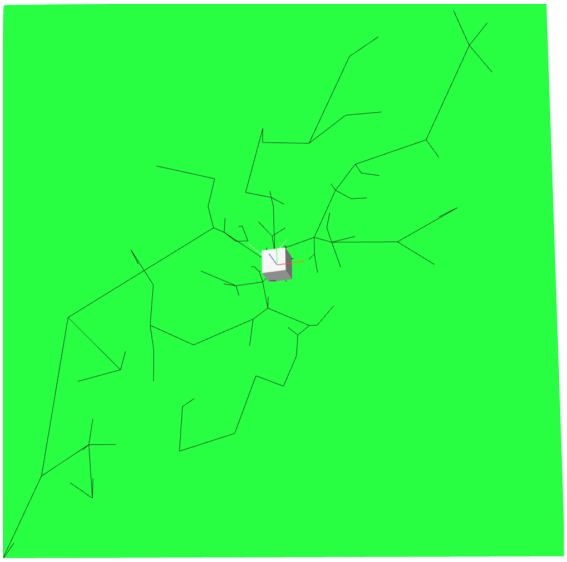


FIGURE 5.12 – Forme élongée, grand φ.

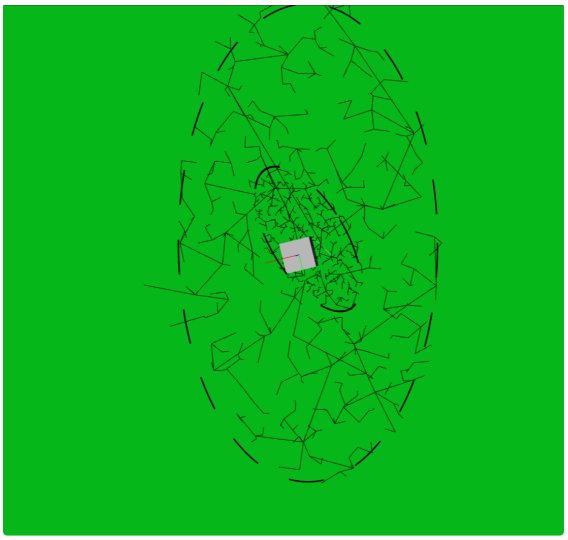


FIGURE 5.13 – Petite puis grande force.

5.2.5 Conclusion

Nous avons proposé une méthode capable d'échantillonner au contact d'une surface. Cette méthode borne la recherche à l'intérieur d'une ellipse puis modifie sa forme en fonction des commandes de l'opérateur. Elle s'adapte aux intentions de l'utilisateur et se déplace en avançant dans la direction qu'il souhaite explorer, en se déformant selon l'intérêt qu'il porte à cette surface.

La surface de l'ellipse est proportionelle à son intérêt pour la surface de contact, l'élongation de l'ellipse est fonction de l'angle à la normale au contact que l'utilisateur crée à l'aide de son objet virtuel et enfin, l'ellipse est orientée selon les mouvements de l'utilisateur.

Nous souhaitons à présent mettre en place un environnement de réalité virtuelle capable d'utiliser cette solution, couplée à un planificateur de mouvement. Nous présentons cette solution à la section suivante.

5.3 Planification au contact en RV et contrôle haptique

Nous proposons dans cette section une contribution originale utilisant notre méthode d'échantillonnage au contact avec détection d'intention couplée avec une architecture de réalité virtuelle et un planificateur de mouvement. Ce nouvel algorithme est appelé H-RRT-C pour "Haptic RRT with Contact".

Nous commencerons par détailler les modalités d'interaction de cette contribution et exposer brièvement son architecture. L'algorithme sera ensuite explicité avant de montrer un exemple illustratif de cette méthode.

5.3.1 Modalités d'interaction

Notre algorithme repose sur l'interaction entre un opérateur humain et un planificateur automatique. Ici sont détaillées les modalités d'interaction de cet algorithme. Nous présentons d'abord le matériel utilisé pour l'affichage et l'immersion, puis ceux qui sont utilisés pour se déplacer dans la scène et pour le déplacement de l'objet. Nous présentons ensuite ce nouvel algorithme appelé H-RRT-C avant de conclure.

5.3.1.1 Matériel

L'affichage de la manipulation se fait sur grand écran utilisant une stéréoscopie passive. L'écran 3D affiche la manipulation de l'objet. Un second écran 2D affiche l'avancement de la planification en affichant l'évolution de la *roadmap* en temps réel. Les lunettes 3D utilisées pour la visualisation sont équipées de marqueurs localisés par un système de capture de mouvement. Cela permet de déplacer le point de vue dans la scène virtuelle. Ainsi, l'humain qui se déplace dans l'espace réel ou en bougeant la tête adaptera le point de vue sur la scène virtuelle. Nous utilisons un bras haptique Virtuose 6D35-45 de la société Haption

qui est une interface bidirectionnelle. Elle permet à l'utilisateur d'agir en temps réel sur l'environnement virtuel et fournir un retour sensoriel à l'utilisateur.

Une description plus détaillée de ce matériel sera donnée au chapitre 6.

5.3.1.2 Interaction et partage d'autorité

L'interaction pour notre nouvel algorithme H-RRT-C se fait de la même manière que pour notre précédente contribution I-RRT-C. Le partage d'autorité en revanche, est plus fin laissant à l'opérateur le choix de la zone d'échantillonnage. Il peut aussi en changer en temps réel l'orientation de l'objet lorsqu'il est au contact grâce au retour d'effort empêchant les collisions.

5.3.2 Algorithme

L'utilisation de notre méthode d'échantillonnage haptique ***HapticSampling*** conjointement avec notre algorithme de planification de mouvement au contact nous permet de planifier des trajectoires pour un objet dans des environnements très contraints. Nous pouvons effectuer des tâches comme le glissement ou l'insertion en nous servant d'un retour haptique en plus du retour visuel. Cela nous permet un meilleur contrôle de l'échantillonnage au contact et apporte des résultats plus rapidement.

Notre nouvel algorithme autorise le partage d'autorité pendant le contact ce qui n'était pas possible avec notre précédente méthode I-RRT-C car elle planifiait au contact de manière automatique et aléatoire. Nous utilisons à présent les intentions de l'opérateur pendant le contact par la mesure de trois paramètres de détection d'intention, augmentant la vitesse générale de la planification.

Pendant le contact, l'utilisateur peut changer l'orientation de l'objet manipulé. Cela n'était pas possible dans notre méthode précédente. En effet, l'opérateur devait choisir une orientation spécifique avant d'entrer au contact. Nous pouvons à présent planifier au contact pour différentes orientations modifiables en temps réel.

Notre planificateur est soit en mode contact soit en mode espace libre. Dans le mode espace libre, un nombre aléatoire a détermine qui détient l'autorité, l'homme ou la machine. Le mode contact démarre lorsque la distance entre l'objet et l'obstacle est inférieure à d.

Ligne 2, la fonction Find_Nearest_obstacle trouve le couple $(\boldsymbol{P_o}, \boldsymbol{P_n})$ de points les plus proches entre les obstacles et l'objet manipulé.

Ligne 3, un nombre aléatoire a est tiré qui détermine le mode humain, ligne 5 ou le mode automatique, ligne 12.

Ligne 5, le test de distance choisit le mode contact ou espace libre. Ligne 6, le planificateur est en mode espace libre, il ajoute à la *roadmap* la configuration courante de l'objet.

Ligne 9 est le mode contact faisant appel à la méthode présentée au début de ce chapitre, elle même faisant appel à la méthode ***ContactSampling*** présentée au chapitre 4.

Algorithm 7 Haptic RRT with Contact : H-RRT-C

Require: $W, T, N, q_{user}, \alpha, d$

loop
  $(\boldsymbol{P_o}, \boldsymbol{P_n})$ = Find_Nearest_Obstacle(q_{user})
  $a \leftarrow$ rand$(0, 1)$
  if $a > \alpha$ **then**
    if $|\boldsymbol{P_o} - \boldsymbol{P_n}| \geq d$ **then**
      $q_{current} \leftarrow q_{user}$
      $T \leftarrow$ Add_Tree($q_{current}$)
    else
      HapticSampling($\boldsymbol{P_o}, \boldsymbol{P_n}, N, q_{user}$)
    end if
  else
    $q_{current} \leftarrow$ Random_Shooter()
    $T \leftarrow$ Add_Tree($q_{current}$)
  end if
end loop

5.4 Conclusion

Ce chapitre dans une première section a présenté la réalité virtuelle, technologie utilisée comme outil dans ce chapitre. Nous avons ensuite présenté ce qu'est la détection d'intention et son utilisation pour la commande de différents types de robots puis quelques travaux utilisant ce genre de méthode. Ont ensuite été présentées les contributions traitant de planification de mouvement qui utilisent des systèmes motorisés bidirectionnels, c'est-à-dire utilisés à la fois comme capteur et comme moteur. Nous avons enfin exposé le choix de périphérique d'interaction pour les contributions de ce chapitre.

Les deux sections suivantes ont respectivement présenté les contributions de cette thèse dans le domaine de la planification de mouvement au contact avec détection d'intention puis l'utilisation de cette méthode dans le cadre d'un planificateur interactif de mouvement dans une architecture de réalité virtuelle. Notre méthode est capable d'étendre une *roadmap* au contact d'une surface très rapidement. Cette *roadmap* croît à l'intérieur d'une ellipse dont la surface, les proportions et l'orientation sont calculées dynamiquement en fonction des ordres de l'opérateur. L'algorithme s'adapte aux intentions de l'utilisateur pour explorer des surfaces. La force de notre deuxième contribution est que l'utilisation d'un bras haptique en réalité virtuelle permet de faciliter grandement l'exploration par un utilisateur ainsi que la recherche du contact grâce au retour haptique.

Les principales voies d'amélioration de ces travaux seraient d'une part de pouvoir configurer automatiquement les nombreux paramètres de l'algorithme interactif et de l'algorithme de planification au contact. En effet, ces paramètres peuvent avoir une influence

considérable sur la vitesse de résolution du problème. La variabilité de chaque humain ajoute une seconde raison très importante sur la nécessité de les voir s'ajuster automatiquement. D'autre part, il serait intéressant de pouvoir planifier au contact sur plusieurs surfaces à la fois. Pour le moment une seule surface est prise en compte. Il serait bon de pouvoir projeter l'ellipse sur une surface voisine ce qui faciliterait la tâche de suivi de surfaces d'un objet.

Chapitre 6

Implémentation et résultats

Ce chapitre présente les apports de cette thèse et le travail de mise en œuvre qu'il a fallu fournir pour y parvenir.

Deux contributions techniques ressortent de cette thèse, deux architectures logicielles de planificateurs interactifs de mouvement. Ces deux architectures couplent un périphérique interactif avec une solution de visualisation et un planificateur automatique.

Nous présentons section 6.1 l'architecture utilisée pour l'algorithme interactif, I-RRT et pour l'algorithme interactif avec contact, I-RRT-C, ce dernier ajoutant la fonction contact à l'algorithme interactif. Ces deux méthodes utilisent comme périphérique d'interaction une souris 6D. Sont ensuite présentés section 6.2 les environnements de tests ainsi que les objets qui leur sont liés. Les deux sections suivantes présentent respectivement les résultats de l'algorithme I-RRT 6.3 puis ceux de l'algorithme I-RRT-C 4.3.

La cinquième section 6.5 décrit le travail de mise en œuvre de l'algorithme interactif avec contact haptique, H-RRT-C. Cet algorithme utilise cette fois un bras haptique comme périphérique d'interaction et se base sur I-RRT-C pour la partie planificateur interactif auquel sont ajoutés de nouveaux développements pour la gestion du bras haptique notamment. La section 6.6 présente enfin les résultats de nos expérimentations avec l'algorithme H-RRT-C.

6.1 Implémentation des algorithmes I-RRT et I-RRT-C

Nous présentons ici l'implémentation de notre algorithme interactif et de notre algorithme au contact. La différence entre les deux étant que I-RRT-C intègre en plus du premier une fonction de planification au contact. Ces méthodes utilisent toutes deux une souris 6D comme périphérique d'interaction.

Ainsi nous commencerons par présenter les outils logiciels utilisés pour ces algorithmes et modifiés pour satisfaire nos besoins. Nous détaillons ensuite le pilote de la souris 6D que nous avons développé et utilisé dans ces logiciels. Puis nous présenterons la manière dont

nous utilisons ces données pour le mouvement en translation et en rotation. Enfin, nous présenterons l'architecture du planificateur interactif.

6.1.1 Outils logiciels

Deux outils logiciels sont utilisés conjointement pour nos besoins en planification de mouvement. Le premier est HPP, un logiciel de planification, le second est un logiciel de visualisation, le gepetto viewer.

HPP [53] est un logiciel libre de planification de mouvement développé par l'équipe LAAS-Gepetto de Toulouse. HPP est un acronyme pour Humanoid Path Planner ou planificateur de chemin pour humanoïdes. Il contient donc des outils pour la planification de mouvement des robots humanoïdes mais ne se limite pas à ces derniers. Il peut en effet planifier tous les types de robots formés de chaînes cinématiques ouvertes ainsi que pour des robots solides comme étudiés dans la présente thèse.

Ce logiciel a été adapté pour notre besoin, la planification interactive, en y ajoutant le pilote décrit plus bas. Il est utilisé conjointement avec le gepetto viewer.

Le gepetto viewer est un logiciel libre développé par l'équipe gepetto permettant d'afficher les résultats de HPP. Il permet de se déplacer dans l'espace autour de la scène à l'aide de la souris. Il est également possible de pivoter le point de vue et de zoomer. Cela permet de placer la caméra selon tous les points de vue. Lorsque la planification est terminée, il est possible de jouer le mouvement à l'écran pour visualiser tout le processus.

6.1.2 Pilote de la souris 6D

Le fabriquant de la souris fournit un pilote et un SDK pour le développement d'applications utilisant son matériel mais son SDK n'est pas satisfaisant. En effet, il n'offre qu'un contrôle sommaire du matériel et n'a aucune documentation. Ainsi il paraissait plus rapide de développer notre propre pilote. Cela permet en outre de maîtriser tout son fonctionnement.

Le fonctionnement de la souris est le suivant : par défaut ses six paramètres valent zéro. La partie mobile de la souris est stabilisée à l'origine à l'aide de ressorts. Les axes de la souris sont bloqués par des butées dans toutes les directions. Lorsqu'un actionneur est poussé à son maximum, le paramètre correspondant vaut 511 ; il vaut -512 lorsqu'il est poussé à son minimum.

Chaque valeur envoyée par la souris correspond à une vitesse de déplacement dans l'univers virtuel : soit une vitesse de translation, soit une vitesse de rotation. La valeur maximale correspondra donc à une vitesse maximale, la valeur minimale à une vitesse maximale dans la direction inverse.

Si l'opérateur déplace les capteurs mécaniques jusqu'à leur maximum, la vitesse de déplacement dans le monde virtuel sature ce qui peut gêner le déplacement. Les vitesses de déplacement doivent donc être adaptées à la taille de l'environnement car cela influe beaucoup sur la performance de l'algorithme et la précision du déplacement. Le réglage

s'effectue par des gains en vitesse de translation et en vitesse rotation qui conviennent à l'utilisateur. Si l'imprécision est trop importante l'opérateur devient de moins en moins utile. Il faut alors réduire la vitesse jusqu'à un seuil acceptable.

6.1.3 Mouvement dans la scène

L'objet à déplacer est affiché au point de départ au début de la simulation. Ses coordonnées initiales sont $(0, 0, 0, 0, 0, 0)$. Les trois premiers paramètres sont la position dans l'espace en trois dimensions souvent appelés (x, y, z). Les trois derniers paramètres expriment la rotation exprimée en angles d'Euler autour de chacun des trois axes du repère cartésien monde. Ces angles souvent notés (θ, ϕ, ψ) peuvent être appelés le roulis, le tangage et le lacet ou roll, pitch, yaw.

6.1.3.1 Translation de l'objet

Lorsque l'objet est en translation, les valeurs lues sont des vitesses instantanées en translation. Chaque trame est envoyée par la souris avec une cadence constante. Elles contiennent chacune un vecteur vitesse de translation $\boldsymbol{v}_i$ pour chaque itération i. Il suffit alors d'intégrer cette vitesse dans un vecteur position $\boldsymbol{p}$. Cette valeur est intégrée avec un gain k_t, pour translation.

On obtient alors :

$$\boldsymbol{p}_i = \boldsymbol{p}_{i-1} + k_t \boldsymbol{v}_i \tag{6.1}$$

6.1.3.2 Changement du point de vue de la caméra

La scène vue dans le gepetto viewer est déplaçable à l'aide de la souris planaire. On peut voir par exemple un même environnement vu sous différents points de vue figures 6.1, 6.2, 6.3, 6.4, 6.5 et 6.6. On appelle caméra le point de vue de l'utilisateur sur la scène. Ces points de vue induisent une position et une orientation de la caméra. L'utilisateur peut placer la caméra sous n'importe quel point de vue. Un repère est attaché à la caméra, initialement aligné avec le repère monde. Il existe une matrice de passage $\boldsymbol{C}$ du repère monde au repère de la caméra. La matrice $\boldsymbol{C}$ est variable car à tout moment l'utilisateur peut changer le point de vue de la scène.

6.1.3.3 Couplage translation et changement du point de vue

La souris opère des déplacements dans le repère monde tandis que l'utilisateur s'attend à des déplacements dans le repère de la caméra. Il faut donc prendre en compte le point de vue de la caméra pour le déplacement effectué par l'utilisateur car s'il pousse la souris vers l'avant, il souhaite avancer, peu importe le point de vue actuel de la caméra.

Le véritable déplacement $\boldsymbol{d}_{réel}$ effectué par l'utilisateur est donc :

$$\boldsymbol{d}_{réel} = \boldsymbol{C}\boldsymbol{d}_i \tag{6.2}$$

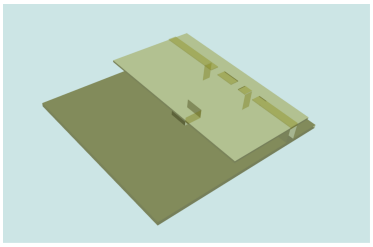


FIGURE 6.1

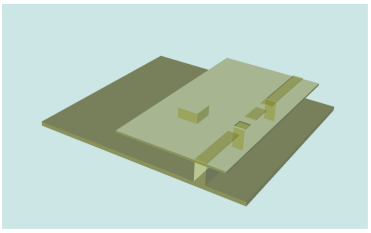


FIGURE 6.2

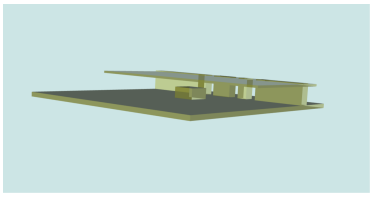


FIGURE 6.3

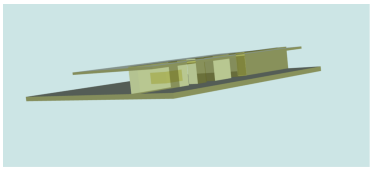


FIGURE 6.4

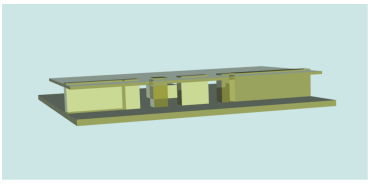


FIGURE 6.5

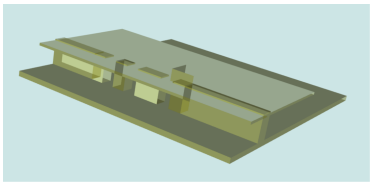


FIGURE 6.6

En conclusion,

$$\boldsymbol{p}_i = \boldsymbol{p}_{i-1} + k_t \boldsymbol{C}\boldsymbol{d}_i \tag{6.3}$$

6.1.3.4 Rotation de l'objet

Lorsque l'on change l'orientation de l'objet avec la souris, les valeurs lues sont des vitesses de rotation instantanée. Chaque trame envoie un vecteur vitesse de rotation $\boldsymbol{\omega}_i$. Un gain adapte la vitesse. À chaque itération, ce vecteur est intégré dans un vecteur orientation $\boldsymbol{o}$ contenant trois angles décrivant l'orientation de OBJ.

Une rotation n'est pas aussi simple à intégrer qu'une translation où une simple somme est suffisante. Nous utilisons dans ce pilote la méthode dite "exponential mapping". Elle consiste à transformer trois rotations élémentaires en une matrice de rotations élémentaires. Cette matrice est multipliée au vecteur $\boldsymbol{o}$ à chaque itération. La démonstration de cette méthode est détaillée dans l'ouvrage de Murray [54].

6.1.4 Limitations

Tout périphérique demande un temps d'adaptation et a un volume de travail limité. La souris 6D ne fait pas exception et il faut un entraînement pour maîtriser son fonctionnement. La prise en main n'est pas naturelle et il faut s'adapter à la rigidité de l'interface. Un nouvel opérateur devra prévoir un temps d'apprentissage.

D'autre part, l'objet déplacé par l'opérateur peut traverser les obstacles en utilisant notre méthode puisque rien n'empêche les collisions. Un exemple est montré figures 6.7 et

6.8. On voit clairement à la figure 6.8 que l'objet est en collision. Dans ce cas, le planificateur ne peut ajouter un nœud à la *roadmap* puisqu'il y a collision.

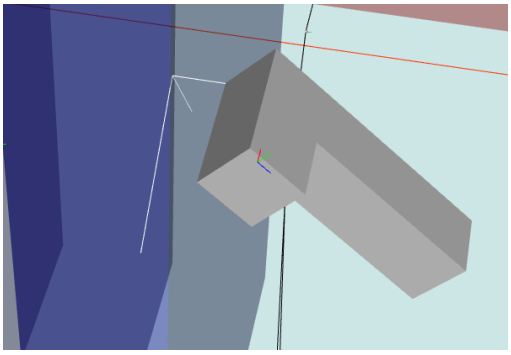


FIGURE 6.7 – Objet dans l'espace libre

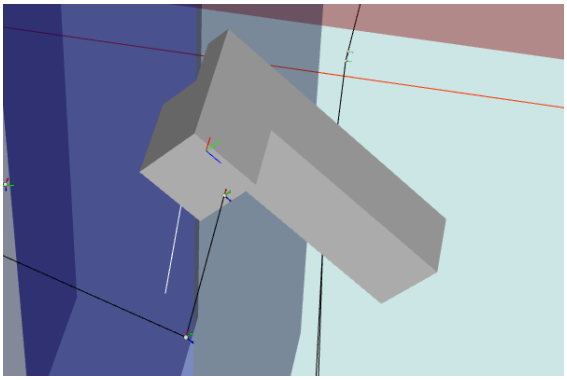


FIGURE 6.8 – Objet en collision.

6.1.5 Architecture logicielle

Disposant d'un planificateur de mouvement, d'un logiciel de visualisation et d'un pilote de souris, nous avons conçu une architecture de planification interactive. Ici est décrite l'architecture logicielle du système.

6.1.5.1 I-RRT

Le logiciel exécute trois threads différents.

Il existe trois threads dans la simulation. Le premier est le pilote de la souris appelé `DeviceThread` sur la figure 6.9. Le pilote de souris tout d'abord appelle sa méthode d'initialisation qui ouvre le fichier du périphérique d'interaction puis initialise les positions et rotations initiales ainsi que les vitesses de rotation et de translation. Le thread se charge ensuite de dépiler les trames envoyées par le périphérique.

Le second thread est celui de l'opérateur appelé `OperatorThread`. Ce thread se sert des données envoyées par le pilote (position, orientation) pour bouger l'objet planifié. Il commence par récupérer ces données et les convertit en une configuration utilisable par le planificateur. Il affiche ensuite l'objet à l'écran à la configuration positionnée par l'utilisateur.

Enfin, le troisième thread est celui du planificateur de mouvement appelé `IntearctivePlanner`. Il est chargé de la planification de mouvement par l'appel à chaque itération d'une méthode ajoutant une configuration à la *roadmap*. Cette méthode choisit le mode de planification : utilisateur ou automatique. Elle tire le paramètre α déterminant qui aura l'autorité.

Un second processus est exécuté, c'est le Gepetto Viewer utilisé pour la visualisation de l'expérience. Ce processus communique avec le précédent à l'aide d'un middleware CORBA

qui permet des appels de fonctions à distance. Ces processus sont commandés par un script python ce qui permet de modifier les conditions des expériences rapidement.

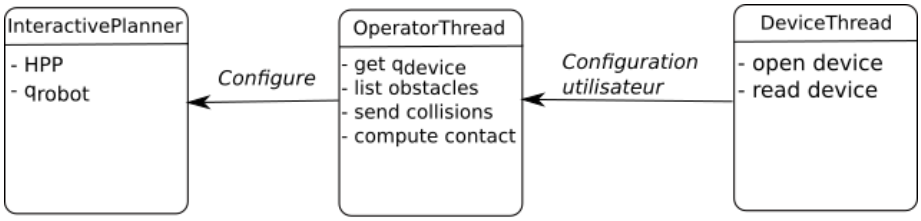


FIGURE 6.9 – Architecture logicielle du planificateur interactif.

6.1.5.2 I-RRT-C

L'algorithme I-RRT-C diffère de I-RRT dans son implantation uniquement par sa capacité à échantillonner au contact.

Le thread `InteractivePlanner` comporte alors un appel à une méthode de détection de collision qui itère sur tous les couples obstacle/robot un test de distance et effectue un classement de ces distances. En outre, le planificateur affiche à l'écran un segment entre le couple de points les plus proches ainsi qu'un repère sur la surface de l'obstacle, comme vu précédemment à la figure 4.9.

Si le mode est choisi comme mode humain, un deuxième test est effectué pour déterminer si l'utilisateur est suffisamment proche ou non d'un obstacle en utilisant le classement décrit plus haut. Si ce deuxième test est valide, démarrent alors les traitements concernant la planification au contact détaillés dans l'algorithme 4.

6.2 Environnements expérimentaux

Cette section présente les deux environnements utilisés tout au long des expériences de ce chapitre, qui présentent chacun des caractéristiques différentes. Cela permet de tester nos contributions sur une variété de problèmes afin d'analyser leurs comportements.

6.2.0.1 Plans croisés

La figure 6.10 présente un premier environnement. Il est encombré et se caractérise par des passages très étroits. Il est constitué de deux plans bleus entre lesquels existe un petit espace libre. Cet espace étant restreint, l'objet manipulé ne peut effectuer de rotation qu'autour de deux de ses axes. Deux plans rouges sont accolés aux plans bleus. Un seul passage permet de passer des uns aux autres comme il apparaît sur les figures 6.11, 6.12 et 6.13. L'espace libre des plans rouges étant bloqués par des barres, l'objet doit nécessairement passer par ce passage étroit. Ces deux plans rouges sont séparés par

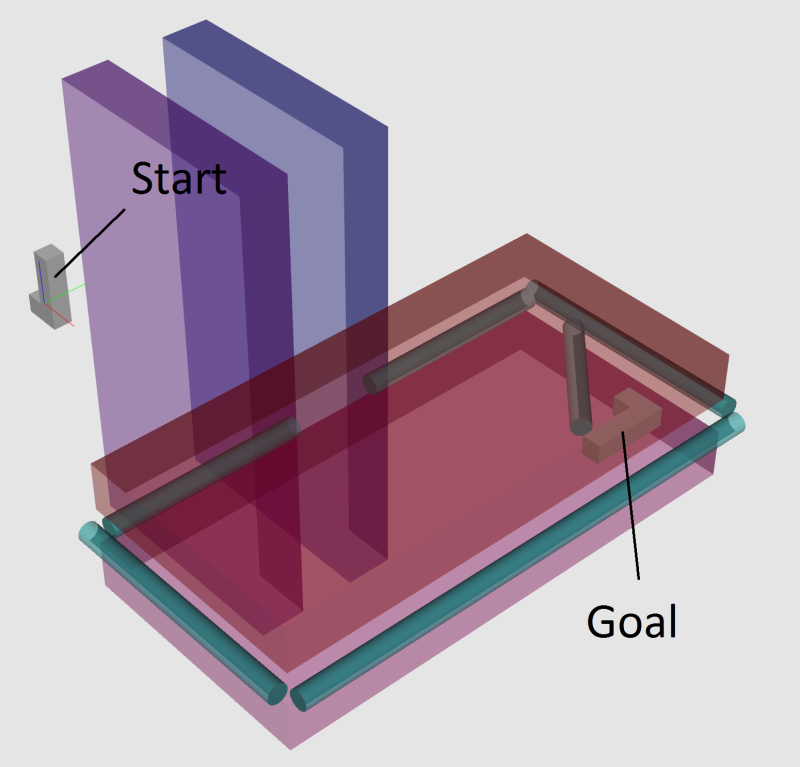


FIGURE 6.10 – Environnement 1 : Plans croisés

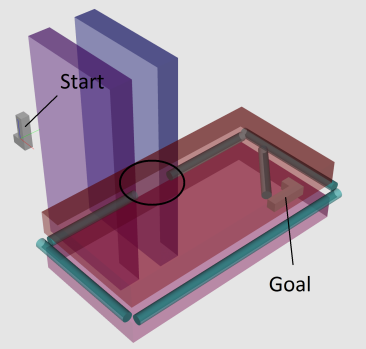


FIGURE 6.11 – Passage étroit entouré

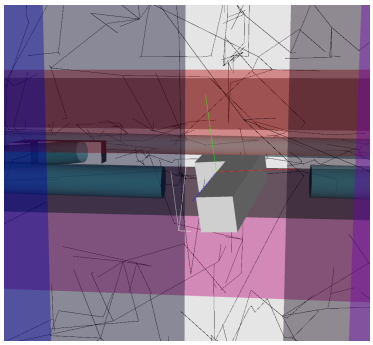


FIGURE 6.12 – Vue de côté

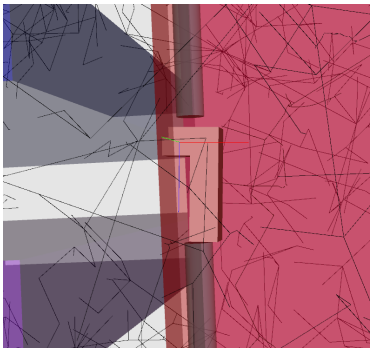


FIGURE 6.13 – Vue du dessus

un espace encore plus étroit que celui entre les plans bleus, l'objet manipulé ne pouvant tourner qu'autour d'un seul axe.

L'objet manipulé dans cet environnement a la forme d'un L d'une longueur de 1,6

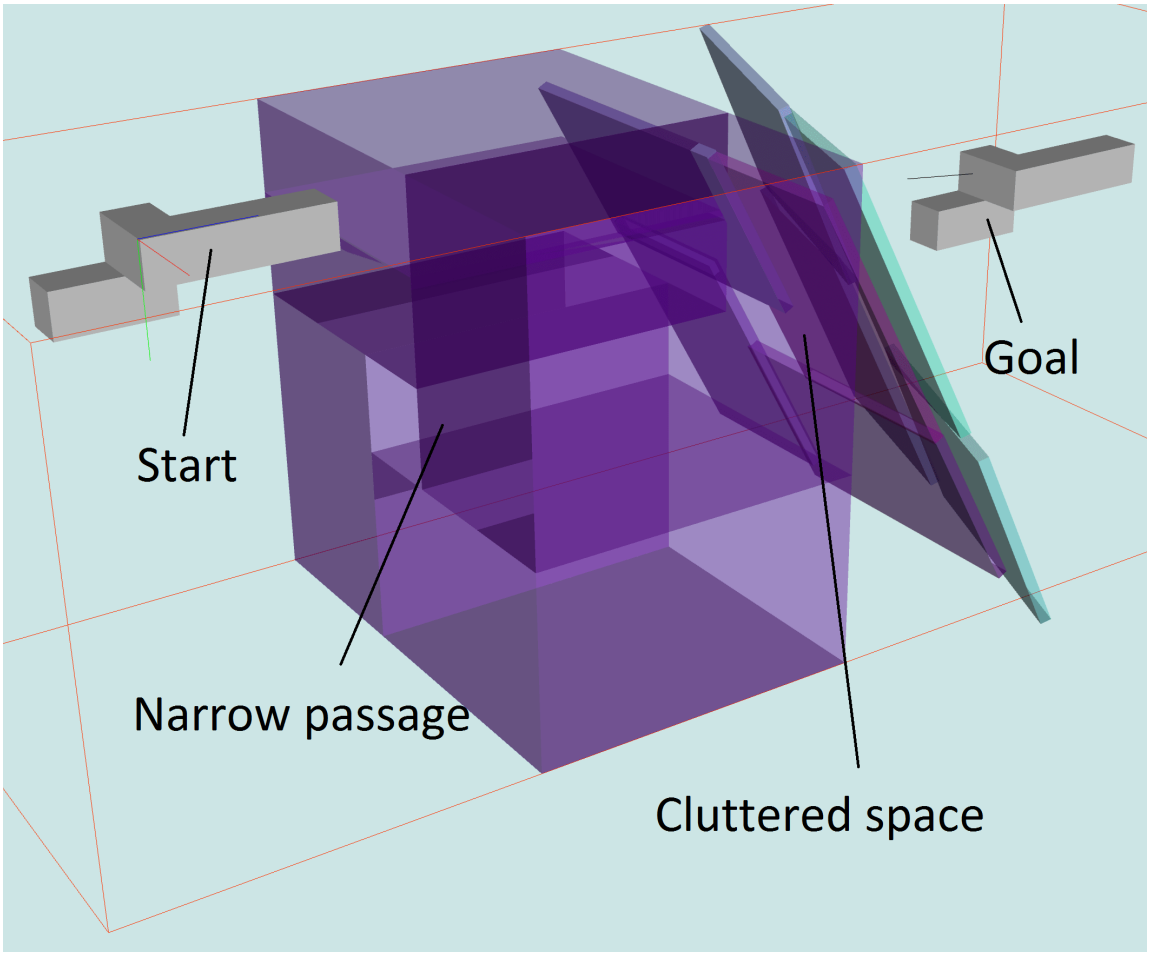


FIGURE 6.14 – Environnement 2 : Tunnel

mètres, une hauteur de 0,8 mètre et une largeur de 0,4 mètre. La distance entre les deux plans bleus est de 1,2 mètres et celle entre les plans rouges de 0,75 mètres.

Pour nos essais, la configuration initiale est située à l'extérieur des plans bleus. La configuration finale est située à l'intérieur des deux plans rouges, derrière une barre oblique, ce qui rend le problème encore plus difficile à résoudre et qui requiert un glissement pour être résolu efficacement.

6.2.0.2 Tunnel

Le deuxième environnement est composé de deux zones différentes, voir figure 6.14. Le premier espace est un long tunnel ayant quasiment la largeur de l'objet manipulé. Il faut s'y insérer avec précision.

La sortie de ce tunnel mène sur un premier plan incliné. Ici, l'objet doit effectuer une rotation autour d'un axe pour pouvoir traverser le trou au milieu de ce plan incliné. Il est suivi par un second plan, incliné différemment. Il faut que l'objet effectue deux rotations pour traverser ce second plan incliné. Étant donné les fortes contraintes dans cet espace, effectuer ces rotations sans assistance est une tâche très difficile pour un humain, car la marge d'erreur est presque nulle.

L'objet OBJ, dans cette expérience est modifié pour le rendre asymétrique selon tous les axes. La configuration initiale est à l'extérieur du tunnel et la configuration finale à la sortie des plans inclinés.

6.3 Résultats, algorithme interactif en espace libre : I-RRT

Cette section présente le résultat des expérimentations avec notre algorithme interactif I-RRT sur les deux environnements présentés dans la section précédente.

6.3.1 Plans croisés

Nous allons résoudre ce problème d'abord en utilisant l'algorithme BiRRT ce qui revient à $\alpha = 1$, ensuite en utilisant notre méthode interactive.

Le premier test utilisant un simple BiRRT a demandé à l'ordinateur 2h45 minutes pour trouver une solution qui apparaît sur la figure 6.15. Ont été tirés 27 600 nœuds ; 55 198 arcs ont étés ajoutés à la *roadmap*. Dans un environnement industriel, de telles performances ne sont pas acceptables, alors que ce problème n'est lui-même pas réaliste, car il comporte peu de surfaces.

Les expériences suivantes sont menées à l'aide d'une souris 6D et de notre algorithme interactif. La figure 6.16 montre un exemple de résultat avec un paramètre de partage d'autorité α égal à 0,5. Nous testons différentes valeurs de α pour voir laquelle donne les meilleures performances. Les résultats sont indiqués table 6.1.

Comme nous pouvions l'imaginer, l'aide d'un opérateur humain change radicalement la vitesse de résolution du problème. Très vite, lorsque des échantillons sont ajoutés dans une zone considérée comme intéressante par l'opérateur, la probabilité augmente que d'autres échantillons aléatoires soient ajoutés dans cette zone. De manière générale, le nombre de nœuds échantillonnés est très petit en comparaison d'une méthode complètement automatique.

En comparaison du BiRRT, notre planificateur atteint des performances bien supérieures. Cependant, le partage d'autorité a une forte influence sur la vitesse de résolution avec pour cet environnement, une performance maximale lorsque $\alpha = 0,95$ ce qui veut dire que 95% des échantillons sont tirés par l'algorithme. La raison est qu'il est bien difficile pour l'opérateur de déplacer correctement l'objet dans cet espace très contraint. Il a donc besoin d'une aide importante de la machine. Des valeurs encore plus hautes de α ne laissent pas suffisamment d'occasions à l'opérateur de donner des ordres à la machine pour qu'elle

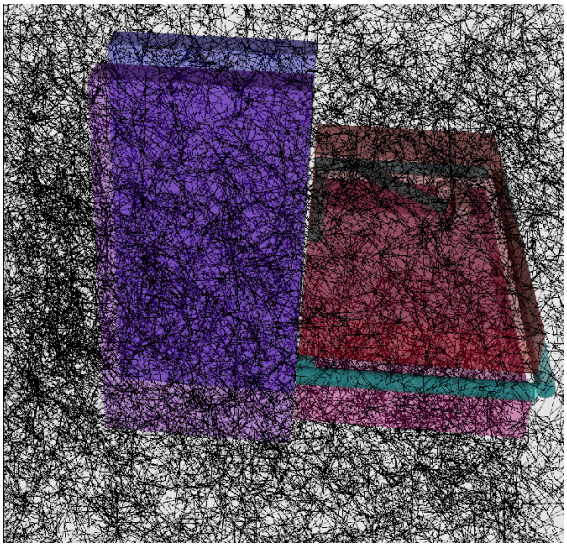


FIGURE 6.15 – Simple RRT

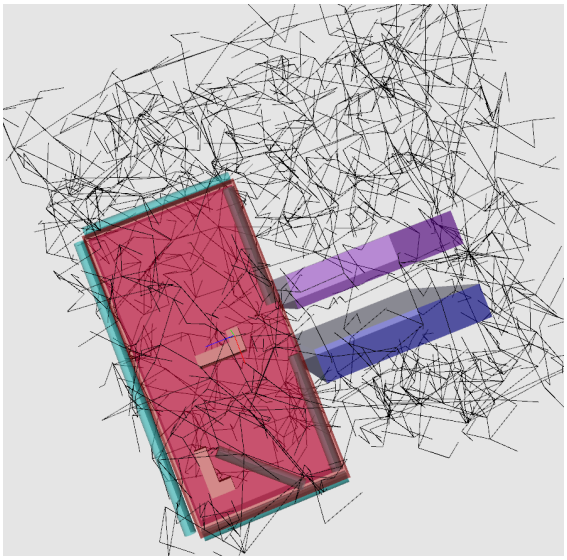


FIGURE 6.16 – I-RRT

Scenario	α	Time	Nodes	Edges
RRT	1	2h45m	27 600	55 198
Interactive	0.99	83s	2 454	4 906
Interactive	0.98	63s	2 059	4 116
Interactive	0.95	25s	891	1 780
Interactive	0.9	30s	1 284	1 074
Interactive	0.6	36s	1 386	2 770
Interactive	0.4	40s	1 372	2 742
Interactive	0.3	108s	2 135	4 268
Interactive	0.2	146s	2 476	4 950
Interactive	0.1	154s	1 878	3 754

TABLE 6.1 – Influence de α, I-RRT

puisse bénéficier de sa présence. Des valeurs trop hautes ralentissent donc le processus par rapport à d'autres valeurs plus basses.

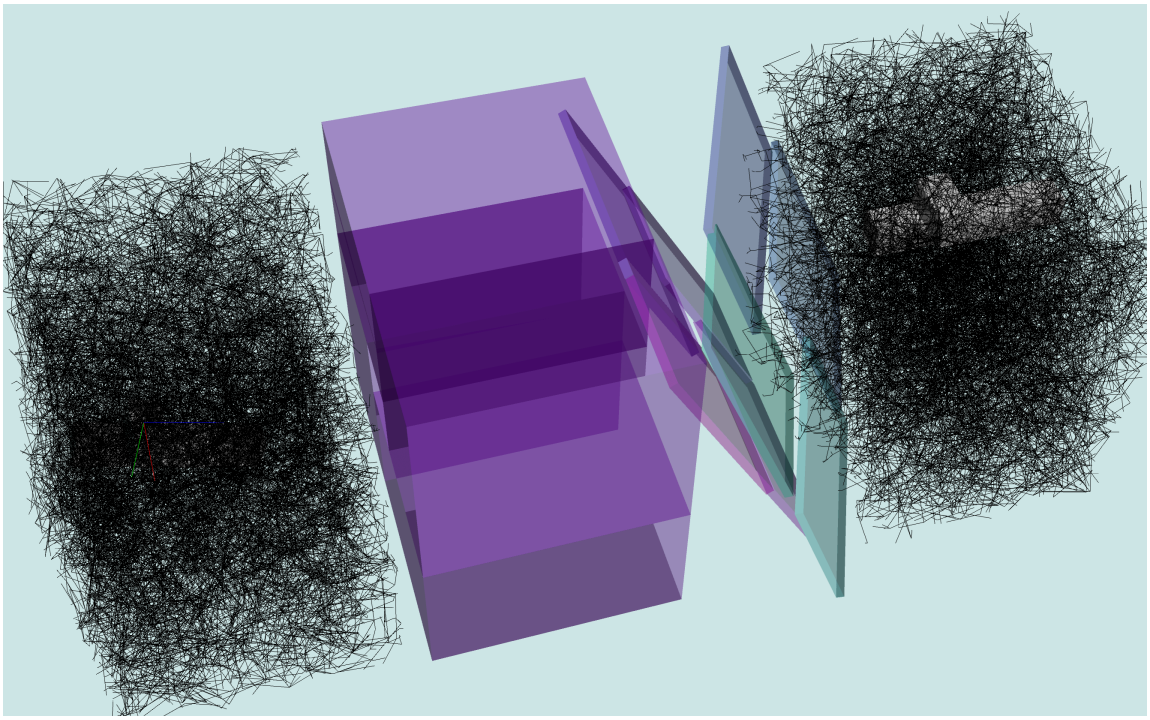


FIGURE 6.17 – Environnement 2, $\alpha = 1$

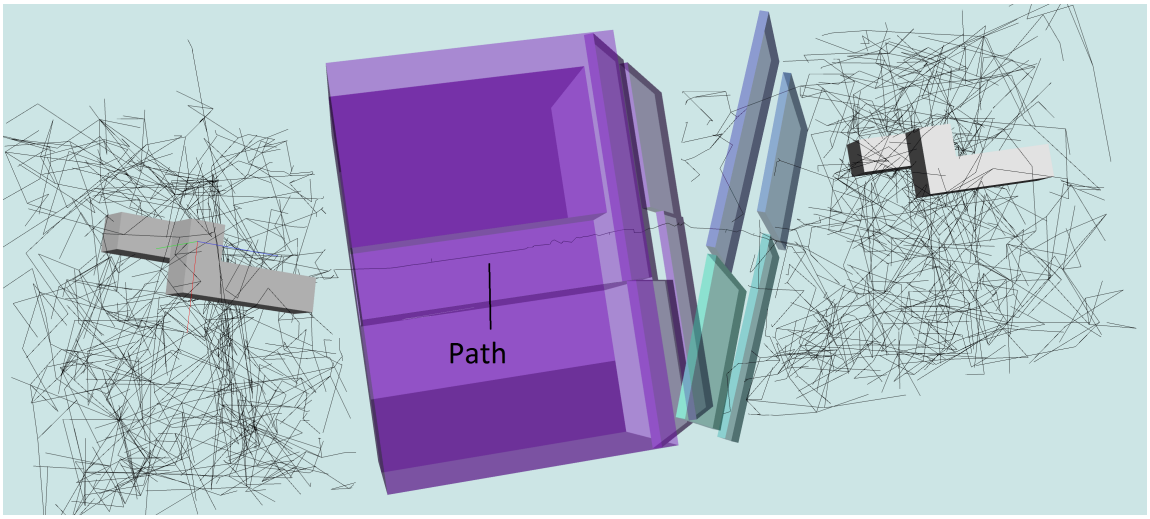


FIGURE 6.18 – Environnement 2, $\alpha = 0.5$

6.3.2 Tunnel

À l'instar du cas précédent, nous résolvons ce problème en utilisant un BiRRT standard, ce qui revient à $\alpha = 1$. Le résultat après environ douze heures de recherche est donné à la figure 6.17. Aucune solution n'étant trouvée à l'issue de ce temps, l'expérience à été interrompue.

Une deuxième expérience a été menée avec $\alpha = 0$, c'est à dire sans aucune aide de la part de l'ordinateur. Un opérateur expérimenté n'a pas réussi à traverser les plans inclinés, cette zone étant trop difficile à traverser sans assistance de l'ordinateur. L'expérience a également été interrompue.

Enfin, nous avons tenté de résoudre ce problème avec un facteur $\alpha = 0,5$, voir figure 6.18. Le résultat a finalement été trouvé par un opérateur expérimenté au bout de 29 minutes et 22 secondes. Ont étés générés 9 025 nœuds et 18 048 arcs. Il est en effet très difficile d'insérer OBJ dans le tunnel et encore plus difficile de lui faire traverser les plans inclinés sans aide. Comme on peut l'imaginer, une demi heure de manipulation précise est particulièrement exaspérante pour un opérateur et ne représente donc pas une solution réaliste. De même, douze heures calcul sans résultat alors qu'une solution existe n'est pas acceptable pour une application industrielle.

Nous voyons ici les limites de notre algorithme interactif en espace libre. Sans utiliser de plans tangents au contact comme aide pour la planification, il devient pratiquement impossible de résoudre cet environnement. Nous résoudrons ce même environnement en utilisant notre algorithme avec contact à la section suivante.

6.4 Résultats, algorithme interactif au contact : I-RRT-C

Cette section présente d'abord les résultats d'expérimentations de notre contribution interactive avec contact sur un premier environnement. Cet environnement nous permet de tester séparément l'influence de chaque paramètre de I-RRT-C afin d'analyser leur influence.

Les paramètres sont au nombre de quatre. Le premier paramètre α a été analysé à la section précédente. Le second paramètre est l'influence de α lorsque est activé le mode contact. Les deux paramètres restants sont le nombre N d'échantillons générés pendant le mode contact et la distance d de passage en mode contact. L'algorithme I-RRT-C est ensuite testé sur un autre environnement afin de comparer ses performances à celles de I-RRT.

6.4.1 Paramètre du mode contact

Nous tentons ici de résoudre le problème des plans croisés (voir section 6.2.0.1) en utilisant l'algorithme I-RRT-C. Dès que l'opérateur rapproche l'objet d'un obstacle, des échantillons sont générés le long d'un plan tangent à l'obstacle au point de contact. La

roadmap peut alors croître rapidement sur les surfaces choisies par l'opérateur.

Pour entrer à l'intérieur de la zone bornée par les plans rouges, l'utilisateur a intérêt à glisser le long de l'un de ces plans. Cela n'est pas possible par un échantillonneur aléatoire mais notre échantillonneur au contact résout le problème rapidement en restant au contact avec l'obstacle.

Les performances de l'algorithme au contact sont testées avec une série d'expérimentations faisant varier le paramètre de partage d'autorité α. Les figures 6.19, 6.20 et 6.21 montrent trois résultats différents avec trois valeurs différentes de α. La table 6.2 montre les résultats pour l'ensemble des expérimentations.

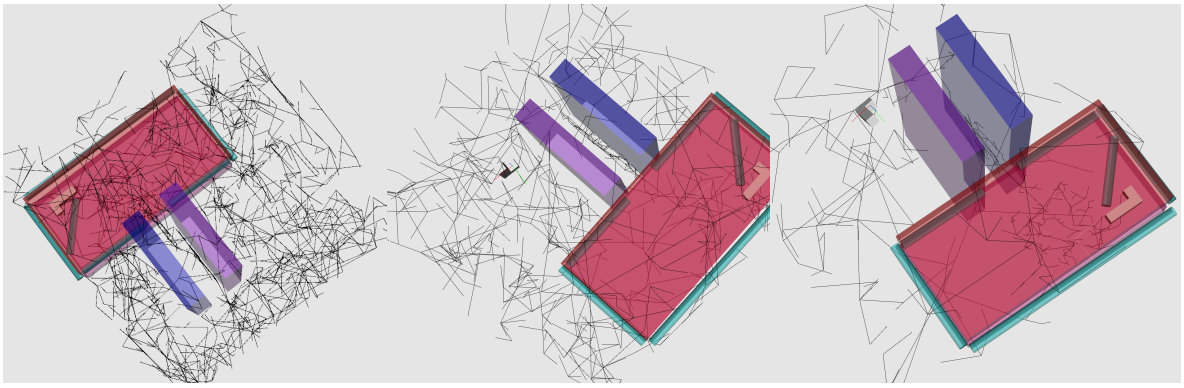


FIGURE 6.19 – $\alpha = 0.8$ FIGURE 6.20 – $\alpha = 0.2$ FIGURE 6.21 – $\alpha = 0.05$

Lorsque le paramètre de partage d'interaction α devient plus petit, le temps nécessaire pour résoudre le problème diminue aussi. Un point minimal est atteint lorsque $\alpha = 0,08$ ce qui veut dire que l'opérateur a autorité pendant 92% du temps de planification. Cela ne veut pas dire que l'ensemble des configurations ajoutées pendant ce temps sont définies par l'opérateur. En effet, lorsque l'opérateur s'approche d'une surface, le mode contact échantillonne selon l'orientation définie par l'opérateur mais aléatoirement sur le plan de contact.

Pour rappel, il existe trois modes opératoires différents :

— l'ordinateur explore aléatoirement

— l'opérateur explore l'espace libre

— l'opérateur active la planification au contact

Dans cette expérience, il faut que le paramètre α soit très petit car sinon l'ordinateur ajoute beaucoup de nœuds et d'arcs dans des espaces inutiles à la résolution du problème. Nous constatons que des valeurs plus petites que $\alpha = 0,08$ donnent de moins bons résultats. C'est parce que de trop petites valeurs font perdre le bénéfice de l'échantillonnage aléatoire ce qui ralentit aussi la planification.

α	Temps	Nœuds	Arcs
1	2h45m	27 600	55 198
0.8	40s	1 729	3 454
0.5	31s	1 314	2 626
0.2	21s	715	1 428
0.1	22s	679	1 356
0.08	14s	538	1 074
0.05	24s	547	1 092
0.04	21s	602	1 202
0.02	31s	658	1 314
0.01	25s	768	1 534
0	47s	894	1 786

TABLE 6.2 – Influence de α, I-RRT-C

Dans tous les cas, notre algorithme au contact donne de bien meilleurs résultats que la solution BiRRT, résolvant le problème beaucoup plus rapidement. Il est également environ deux fois plus rapide que la solution interactive sans contact, pour rappel, le meilleur temps obtenu avec cette méthode est de 25 secondes.

6.4.2 Paramètre du nombre d'échantillons au contact

À chaque fois que le mode contact commence, un nombre N d'échantillons sont générés à la surface définie par l'opérateur. Après cela, le mode contact est relancé si la condition de distance est valide.

Nous voulons déterminer l'influence de N sur l'efficacité de la planification. Nous avons ainsi conduit une série d'expérimentations pour observer le comportement de l'algorithme en fonction de N. Pendant ces tests, le paramètre α est fixé à 0,2 car c'est une valeur moyenne. Les résultats, donnés table 6.3 indiquent que N influence négativement le processus global lorsque sa valeur est trop extrême. Nous décidons donc de le laisser fixé à la valeur de $N = 5$ pour le restant des expérimentations, cette valeur donnant les meilleurs résultats pour cette expérience.

6.4.3 Paramètre distance d'entrée en contact

L'activation du mode contact se fait lorsque l'opérateur s'approche d'un obstacle. Dans toutes les expériences précédentes, le paramètre d d'activation du mode contact était fixé à $d = 0,1$ mètre. Nous déterminons ici l'influence de d pour la résolution de l'environnement

Nombre d'échantillons N	Temps (s)	Nœuds	Arcs
1	62	1 371	2 740
2	37	1 064	2 126
5	18	711	1 420
10	27	902	1 802
50	36	1 019	2 036
100	42	1 155	2 308
200	43	1 004	2 006
500	52	1 196	2 390

TABLE 6.3 – Influence du nombre d'échantillons au contact N

des plans croisés. Les autres paramètres sont fixés et valent $\alpha = 0,2$ et $N = 5$. Les résultats, table 6.4 ne montrent pas d'influence de d sur les performances de l'algorithme. Tout juste pouvons nous affirmer qu'une valeur trop grande empêchera l'utilisateur de naviguer en espace libre (ce que peut déjà faire la machine) et qu'une valeur trop petite empêchera l'utilisateur d'entrer au contact car lui laissant une trop petite fenêtre avant collision pour activer ce mode.

Distance d (m)	Temps (s)	Nœuds	Arcs
0.02	38	1 075	2 148
0.05	53	1 170	2 338
0.1	36	1 114	2 226
0.2	42	1 130	2 258
0.5	29	1 051	2 100
0.8	34	1 078	2 154
1	35	1 117	2 232

TABLE 6.4 – Influence de la distance d

6.4.4 Test sur l'environnement tunnel

Nous testons enfin notre algorithme avec un deuxième environnement pour vérifier ses performances.

Cet environnement, représenté figure 6.14, était pratiquement impossible à résoudre en n'utilisant que notre méthode interactive sans contact. Nous recommençons donc nos

expériences en utilisant à la fois l'interaction et le contact. La table 6.5 montre les résultats de cette expérience. Nous remarquons que l'on peut maintenant résoudre le problème en un temps très court, à comparer avec la demi heure en utilisant l'algorithme I-RRT avec $\alpha = 0.5$. Nous constatons que des valeurs de α proches de 1 ou de 0 dégradent considérablement les performances : un temps plus long est nécessaire, un plus grand nombre de nœuds et un plus grand nombre d'arcs sont nécessaires se traduisant par un temps de résolution plus long.

L'ordinateur et l'humain ont donc ici des rôles équilibrés. L'ordinateur a besoin que l'humain lui indique où échantillonner pour entrer dans le tunnel extrêmement contraint. La probabilité d'y rentrer aléatoirement est proche de zéro. C'est ce qui a mis en échec la méthode totalement aléatoire dans la section précédente. D'un autre côté, l'opérateur a absolument besoin d'une assistance pour traverser les plans inclinés. Il a besoin d'une assistance d'échantillonnage aléatoire et d'une possibilité d'utiliser le contact pour avancer plus vite. C'est l'absence de cette assistance qui a mis en échec la méthode totalement manuelle dans la section 6.3. Les valeurs donnant les meilleurs résultats sont ainsi incluses dans une fenêtre de valeurs comprises entre 0,6 et 0,9 environ.

La figure 6.22 montre le résultat pour $\alpha = 0,5$. Nous constatons qu'un seul et unique chemin est trouvé pour traverser le tunnel étroit.

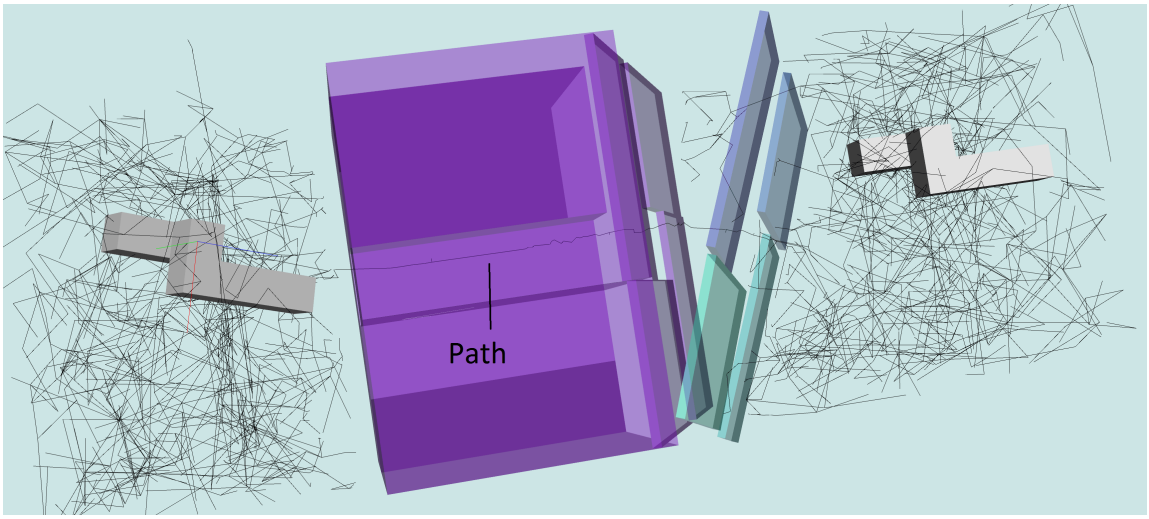


FIGURE 6.22 – Environnement 2, $\alpha = 0.5$

6.5 Implémentation de l'algorithme H-RRT-C

Ici est présentée l'architecture utilisée pour les expérimentations de notre algorithme interactif avec planification haptique et détection d'intention. L'intérêt de cette nouvelle

α	Temps (s)	Nœuds	Arcs
0.98	197	3 508	7 014
0.95	218	3 741	7 480
0.9	65	2 021	4 040
0.8	68	2 026	4 051
0.7	61	1 818	3 634
0.6	65	1 718	3 434
0.5	138	2 396	4 790
0.4	112	2 051	4 100
0.3	158	2 072	4 142
0.2	132	1 810	3 618
0.1	157	1 423	2 844
0.05	624	2 585	5 168

TABLE 6.5 – Environnement 2, influence de α, I-RRT-C

méthode est qu'un bras haptique est utilisé à la place d'une souris 6D. Cela permet de ressentir l'environnement afin d'améliorer la planification. En effet, dès lors que l'objet entre en collision avec l'obstacle, l'opérateur en est directement averti au travers du bras haptique. Il sait dans quelle direction se situe l'obstacle et, pour peu que l'objet soit suffisamment simple, il sait également quelle partie de ce dernier est en collision.

Nous présentons ici l'implémentation de notre algorithme interactif et haptique. Nous présentons d'abord la plate-forme de RV du LGP sur laquelle est développé l'algorithme. Dans une deuxième partie, nous verrons comment cette dernière est connectée au logiciel de planification de mouvement, HPP. Est enfin présentée l'architecture logicielle utilisée pour ce planificateur.

6.5.1 Plate-forme de réalité virtuelle

L'ensemble du matériel de Réalité Virtuelle fonctionne à l'aide de Virtools sous Windows XP. Ce logiciel permet d'écrire des scripts pour des simulations et d'interfacer ces périphériques tout en laissant la possibilité de programmer en C++ des algorithmes communiquant avec ces scripts. Nous présentons ici le matériel utilisé pour notre algorithme de planification de mouvement avec détection d'intention, H-RRT-C.

6.5.1.1 Affichages

Le système de visualisation 3D est un système de stéréoscopie passive. Il est constitué de deux vidéo-projecteurs ProjectionDesign F2 munis chacun d'un filtre à polarisation circulaire avec une polarisation différente pour chaque projecteur, voir figure 6.23. Les images sont projetées sur un écran de 3m de large sur 2,25m de haut conservant la polarisation de lumière, voir figure 6.24. Pour reconstituer l'image 3D, l'utilisateur doit s'équiper de lunettes munies de filtres du même type que ceux des projecteurs afin de conserver, pour chaque oeil, uniquement l'image qui lui est destinée, voir figure 6.25

La visualisation de l'assemblage se fait sur cette projection 3D de la scène. Cela permet une immersion de l'opérateur facilitant sa tâche d'insertion. Cette visualisation est un premier bouclage entre l'utilisateur et la machine. Une deuxième visualisation se fait sur un second écran où est affiché l'avancement de la planification de la même manière que pour nos précédentes contributions.

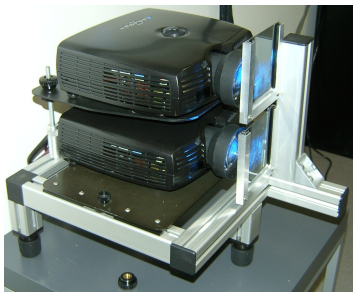


FIGURE 6.23 – Projecteurs

FIGURE 6.24 – Écrans

FIGURE 6.25 – Lunettes

6.5.1.2 Déplacement dans la scène

Nous utilisons un système de capture de mouvement pour tenir compte du point de vue de l'opérateur pour l'affichage de la scène virtuelle. Ce système de capture de mouvement optique est fourni par la société ART (Advanced Real-time Tracking GmbH). Il est constitué de quatre caméras Infrarouge (IR) disposées autour de l'écran. Si deux caméras suffisent à trianguler la position des sphères, un plus grand nombre de caméras permet d'augmenter la précision du système et de prévenir les problèmes d'occultation des marqueurs.

Un logiciel appelé DTrack traite les données issues des caméras IR. Il calcule les positions et orientations des différents objets équipés de marqueurs. Ce logiciel est hébergé sur une station dédiée équipée d'un périphérique prévu pour y connecter les caméras.

Les lunettes 3D utilisées pour la visualisation sont équipées de marqueurs localisés par un système de capture de mouvement. La technologie décrite ci-dessus permet alors à l'opérateur de se déplacer naturellement autour de la scène ou de bouger la tête afin de modifier son point de vue tout en ayant les mains libres. Le point de vue sur la scène

virtuelle est ainsi modifié en temps réel en fonction du placement des marqueurs donc des actions de l'utilisateur.

6.5.1.3 Déplacement de l'objet

Nous utilisons un bras haptique Virtuose 6D35-45 de la société Haption, voir figure 6.26, qui est une interface bidirectionnelle. Elle permet d'une part à l'utilisateur d'agir sur l'environnement virtuel et fournit d'autre part un retour sensoriel à l'utilisateur. Le bras est commandé par le logiciel IPSI, Interactive Physics Simulation Interface (IPSI) fourni par Haption. C'est un moteur physique pour la manipulation d'objets rigides intégrant une interface vers les périphériques haptiques de la société Haption. Il intègre un moteur physique temps réel.

FIGURE 6.26 – Bras haptique virtuose 6D35-45

6.5.2 Interface du bras haptique

6.5.2.1 Middleware ZeroMQ

Le logiciel de planification de mouvement HPP ne peut pas être compilé pour Windows tandis que la plate-forme de réalité virtuelle fonctionne sous Windows XP uniquement. Il était donc nécessaire de trouver un moyen pour la communication entre les deux systèmes.

Une première tentative a été de programmer une interface CORBA pour Windows et ainsi d'éviter toute modification du canal de communication de HPP. En effet, HPP possède déjà une interface CORBA dont il se sert notamment pour communiquer avec le gepetto viewer. Malheureusement devant l'ampleur et la difficulté du travail, un choix plus simple est de choisir un autre middleware alternatif afin de limiter le temps de développement. Le choix s'est porté sur ZeroMQ, middleware ayant les même fonctionnalités que CORBA. ZeroMQ, à la différence de CORBA, est non seulement libre mais aussi et surtout, multi plate-forme nativement ainsi que bien documenté avec des exemples fonctionnels directement utilisables.

6.5.2.2 Synchronisation des simulations

Une dernière difficulté a été de faire coïncider les modèles. En effet, deux simulations s'exécutent en parallèle. La première, sous HPP est identique à celles décrites précédemment. HPP fait de la planification de mouvement, interprète les données issues du périphérique d'interaction et affiche le tout à l'écran à l'aide de Gepetto Viewer. La seconde simulation, sous Virtools est celle de l'environnement immersif contenant un moteur physique et avec laquelle l'utilisateur interagit à l'aide d'un bras haptique. Cette seconde simulation se doit de contenir exactement les mêmes modèles de l'environnement et de OBJ. Les tailles doivent être identiques, ainsi que les positions et orientations initiales. La vitesse de déplacement d'OBJ doit être la même.

Les modèles sont inversés lorsqu'ils sont intégrés dans Virtools, par une symétrie axiale. Il est donc nécessaire de prendre en compte cela lors du dessin du modèle pour Virtools en le retournant préalablement. La convention des repères monde est inversée entre Virtools et HPP ce qui veut dire qu'un déplacement ou une rotation sous Virtools donnera un résultat opposé sous HPP. Les mouvements envoyés par Virtools à HPP sont donc adaptés à la norme HPP avant d'être envoyés au planificateur.

6.5.3 Architecture logicielle

L'architecture finale est décrite dans la figure 6.27.

Virtools exécute la simulation utilisée par l'opérateur du bras haptique. Cette simulation affiche l'état courant à l'écran et envoie des données à HPP par un réseau local grâce au middleware ZeroMQ. Ces données sont la position et l'orientation de l'objet et le vecteur 3D de la force utilisateur. Celui-ci peut-être nul s'il ne bouge pas le bras, jusqu'à très grand si l'utilisateur pousse fort contre un obstacle.

Du côté planificateur, un client ZeroMQ reçoit les données. À la place d'une souris 6D, il est possible d'indiquer au thread `DeviceThread` qu'un autre périphérique est utilisé. Le thread effectue alors les modifications nécessaires pour s'adapter au nouveau matériel et reçoit à la place de données souris, les données reçues par le client ZeroMQ. Ce mécanisme cache le matériel réel et n'envoie in fine que des configurations au thread opérateur sans que le planificateur ait à faire la différence entre les différentes options matérielles. Le reste de l'architecture est identique à celle décrite plus haut pour l'algorithme I-RRT-C.

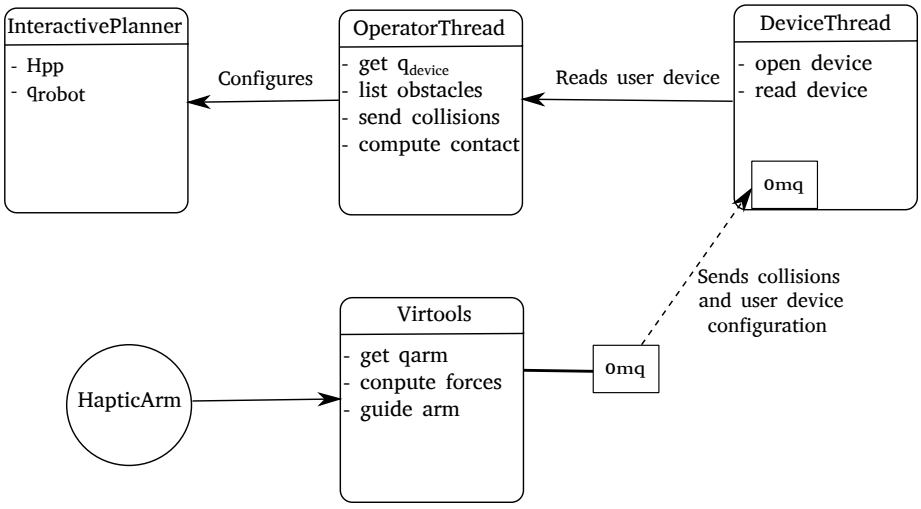


FIGURE 6.27 – Architecture logicielle H-RRT-C.

6.6 Expérimentations, algorithme haptique avec contact : H-RRT-C

Cette section présente dans un premier temps les résultats d'expérimentations de notre contribution interactive haptique sur les mêmes environnements que ceux décrits section 6.2. Les figures de cette section montrent ces environnements avec des couleurs différentes car une conversion a eu lieu afin de les adapter à Virtools mais les géométries sont les mêmes. Avoir les mêmes géométries nous permet de comparer les performances de H-RRT-C à celles de I-RRT et de I-RRT-C.

6.6.1 Tunnel

La première expérience se fait sur l'environnement constitué du tunnel suivi de deux plans inclinés. Pour rappel, le tunnel est très étroit, l'objet manipulé n'y pouvant s'y insérer que très difficilement. Il faut orienter l'objet avec une grande précision pour pouvoir traverser les deux plans inclinés qui suivent, voir figure 6.28.

Le problème lié à cet environnement est testé avec l'algorithme H-RRT-C. La table 6.6 contient les résultats des expériences pour diverses valeurs de α avec $N = 5$ et $d = 0,1$. La première ligne avec $\alpha = 0$ est une expérience menée sans génération aléatoire de nœuds mais avec tout de même l'aide de l'ordinateur pour la planification au contact. Plus de deux minutes sont nécessaires dans de telles conditions. Ce résultat est à comparer avec la demi heure nécessaire en utilisant notre algorithme interactif sans algorithme de contact ni retour d'effort ni aide de l'ordinateur, voir section 6.3.2. Le retour haptique a divisé par trente le temps nécessaire à la résolution du problème grâce au sens haptique très performant pour une tâche de planification de mouvement.

Lorsque nous utilisons l'algorithme avec $\alpha > 0$, les performances s'améliorent encore beaucoup atteignant un optimum avec $\alpha = 0,2$ et une durée de l'expérience de 27 secondes. Cette performance est à comparer avec notre meilleur résultat en utilisant l'algorithme I-RRT-C qui était de 61 secondes. H-RRT-C est ici deux fois plus rapide. La figure 6.29 montre le résultat de l'expérience pour $\alpha = 0,5$.

Nous constatons que le meilleur résultat est obtenu avec $\alpha = 0,2$. Une valeur proche de 1 ou de 0 donne de moins bons résultats. Des valeurs médianes donnent de meilleurs résultats avec un temps plus court et moins de nœuds et d'arcs. Comme dans le cas de l'algorithme I-RRT-C, à la fois l'opérateur et la machine sont utiles dans cet environnement. Cela est dû au fait que nous avons largement besoin de l'opérateur humain pour traverser le tunnel car cela est presque impossible pour l'ordinateur. Ce denier est cependant utile dans la seconde zone constituée des plans inclinés.

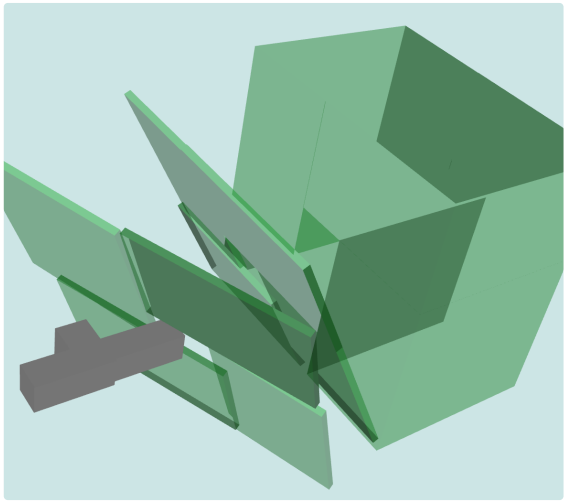


FIGURE 6.28 – Tunnel et plans inclinés

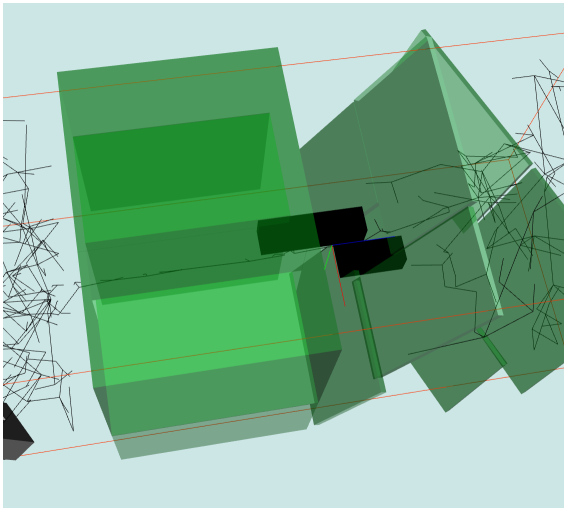


FIGURE 6.29 – alpha = 0.5

α	Temps (s)	Nœuds	Arcs
0	135	2 110	4 218
0.05	60	871	1 740
0.1	67	924	1 846
0.2	27	536	1 070
0.5	33	661	1 320
0.8	28	671	1 340
0.95	38	669	1 336

TABLE 6.6 – Influence de α, scénario 1, H-RRT-C.

6.6.2 Plans croisés

Nous testons ensuite l'algorithme H-RRT-C dans l'environnement des plans croisés. Pour rappel, il est constitué de deux paires de plans entre lesquels l'espace est très contraint. Il existe un petit passage au croisement de ces plans pour passer des uns aux autres, voir la description précise section 6.2.

Nous observons les performances de notre méthode selon la valeur du paramètre α. Les résultats sont donnés dans la table 6.7. Hormis pour les valeurs extrêmes de ce paramètre, nous sommes presque toujours plus rapides que pour la meilleure paramétrisation de la méthode I-RRT-C pour le même environnement. La valeur optimale est ici obtenue avec α autour de $0,1$ pour une durée d'expérience de 10 secondes, ce qui plus rapide que tout autre méthode, le meilleur temps obtenu avec I-RRT-C étant de 14 secondes.

La figure 6.30 montre le résultat pour $\alpha = 0$. La figure 6.31 illustre le phénomène de planification au contact. Nous pouvons y voir l'échantillonnage de nœuds à la surface de l'un des deux plans bleus lorsque l'utilisateur a poussé l'objet contre ce dernier.

α	Temps (s)	Nœuds	Arcs
0	15	484	966
0.005	18	778	1 548
0.01	10	669	1 336
0.1	12	862	1 722
0.3	20	926	1 850
0.6	21	954	1 897
0.9	39	1 262	2 522

TABLE 6.7 – Influence de α, scénario 2, H-RRT-C

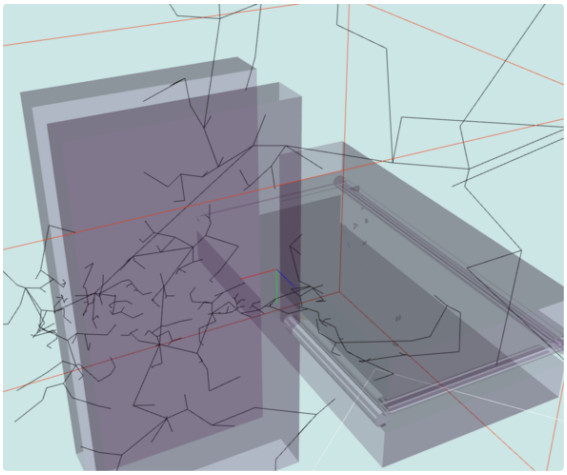


FIGURE 6.30 – alpha = 0

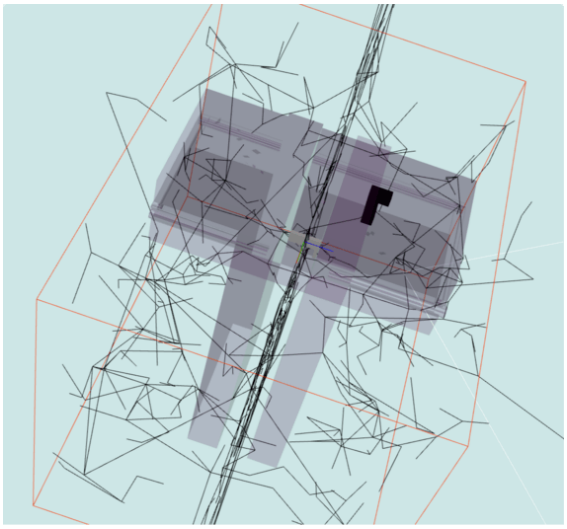


FIGURE 6.31 – alpha = 0.1

6.7 Conclusion

Ce chapitre a exposé le travail d'implémentation et d'expérimentation effectué au cours de cette thèse. Deux architectures ont été développées au cours de cette thèse. La première met en œuvre les algorithmes I-RRT et I-RRT-C. La deuxième englobe la première pour mettre en œuvre l'algorithme H-RRT-C.

La comparaison des performances de l'algorithme I-RRT par rapport à RRT montre que le premier est beaucoup plus rapide que le second grâce à la collaboration d'un humain avec un algorithme. L'humain étant capable de chercher directement au bon endroit, il donne des indications précieuses à l'algorithme, accélérant le processus global. Le gain attendu pour I-RRT par rapport à RRT est au moins de 300% comme constaté sur l'environnement exemple du labyrinthe. Le gain est de minimum deux ordres de grandeur sur des environnements encombrés comme ceux décrits dans ce chapitre.

Les tests effectués sur nos environnements en utilisant la solution I-RRT-C a montré que ce dernier était plus efficace que I-RRT dans les espaces encombrés. La possibilité de planifier au contact que permet I-RRT-C donne de très bons résultats : elle accélère la planification de mouvement et permet en plus des opérations d'insertion ce qui est très difficile à faire sans gestion du contact. Nous avons constaté que sur un même problème, l'ajout du mode contact permet d'accélérer la vitesse de résolution de 80% minimum.

Les expérimentations de l'algorithme H-RRT-C ont montré qu'il était supérieur à I-RRT-C en tous points. Il est aussi supérieur en termes de maniabilité car le retour haptique apporte une aide précieuse à l'utilisateur pour le déplacement en environnement encombré. Cet algorithme haptique est caractérisé non seulement par la présence du retour haptique

mais aussi par un échantillonnage au contact déterminé par trois paramètres que sont la surface de l'ellipse, sa direction et son élongation. En comparaison de I-RRT-C, ce nouvel algorithme est 50% à 200% plus rapide pour la résolution des deux problèmes étudiés ici.

Il sera nécessaire d'analyser séparément l'influence de chaque paramètre de l'algorithme H-RRT-C comme cela a été fait pour l'algorithme I-RRT-C. Cela permettra de savoir quels paramètres sont critiques et de quelle manière les régler en fonction de l'environnement et de l'utilisateur.

Chapitre 7

Conclusion

7.1 Synthèse

Nous avons présenté dans le premier chapitre le contexte de cette thèse, son positionnement industriel et son positionnement scientifique. Le second chapitre a présenté l'état de l'art de la planification de mouvement automatique en se focalisant sur les méthodes probabilistes qui sont celles utilisées dans nos travaux.

Nous avons présenté dans le troisième chapitre notre méthode de planification interactive de mouvement. Elle permet à un opérateur humain de travailler conjointement avec une machine pour rechercher une trajectoire valide entre une configuration de départ et une configuration d'arrivée. Nous avons développé cette solution en nous appuyant sur une souris 6D comme périphérique d'interaction et en partageant le temps entre l'homme et la machine. Ainsi, nous pouvons utiliser à la fois la capacité de la machine, la vitesse, et celle de l'homme, l'intelligence, ce qui permet de résoudre des problèmes de planification de mouvement beaucoup plus rapidement que si l'un ou l'autre protagoniste était seul. La cohabitation est effectuée à l'aide d'un paramètre de partage d'autorité α déterminant le pourcentage d'itérations allouées à la machine.

Nous avons exposé dans le quatrième chapitre notre travail concernant la planification de mouvement au contact. Il existe de nombreuses situations d'assemblage où le contact est une propriété utile pour effectuer un mouvement. Or la planification de mouvement vise généralement à éviter de rentrer en collision ou en contact avec des obstacles. À la différence d'autres méthodes de planification au contact, notre méthode est une méthode globale et n'est pas une méthode basée sur de la rétraction de portions de trajectoires invalides, comme le font la plupart des méthodes au contact.

La combinaison de notre méthode interactive présentée au chapitre 3 et notre méthode de planification au contact a été nommée I-RRT-C et a été décrite dans deux publications

dans les conférences IROS2016 [7] et RoMan2016 [8].

Dans le cinquième chapitre, nous avons présenté notre travail de planification interactive de mouvement avec contact et retour haptique appelé H-RRT-C. Cette solution est une amélioration substantielle de notre précédente solution de planification interactive de mouvement avec contact. Cette solution utilise à la fois un bras haptique pour planifier au contact de manière beaucoup plus efficace qu'avec un périphérique sans retour haptique mais aussi une méthode de détection d'intention permettant de prédire ce que l'utilisateur veut faire dans le cadre de la planification au contact.

Cela permet d'obtenir des trajectoires au contact beaucoup plus rapidement car elles nécessitent un plus faible nombre d'échantillons tout en explorant des zones plus pertinentes du point de vue de l'utilisateur. Cette solution, qui se base sur notre précédente contribution a été publiée dans un article de la conférence RoMan2017 [9].

Enfin, le sixième chapitre a décrit les deux solutions techniques que nous avons mises en œuvre au cours de cette thèse. La première est une architecture de planification interactive de mouvement intégrée au logiciel de planification de mouvement HPP développé par l'équipe Gepetto du LAAS. Cette architecture inclut un pilote de souris 6D et est utilisée pour le test des algorithmes I-RRT et I-RRT-C décrits dans ce chapitre. La deuxième architecture développée permet de coupler notre planificateur interactif à une plate-forme de réalité virtuelle présente au laboratoire LGP. Cela a permis d'utiliser un bras haptique comme périphérique d'interaction en plus d'une vision stéréoscopique. Les résultats d'expériences sur cette architecture ont permis de mesurer l'intérêt du retour haptique pour la planification interactive de trajectoire.

7.2 Perspectives

En ce qui concerne la planification interactive de mouvement, la possibilité de voir le paramètre de partage d'autorité α évoluer au cours du temps pour permettre son adaptation à l'opérateur ou à l'environnement pourrait améliorer les performances et l'ergonomie de l'algorithme. Différentes alternatives au réglage fixe du paramètre α sont envisageables mais il serait intéressant de pouvoir modifier ce paramètre à chaque itération en fonction des mouvements de l'opérateur. Ce paramètre déterminant le pourcentage d'itérations allouées à la machine, il serait utile qu'il s'adapte en temps réel aux actions de l'opérateur. Lorsque l'opérateur est très actif, ce paramètre devrait être plus bas pour lui laisser d'avantage d'initiative. Lorsqu'il est beaucoup moins actif, ce paramètre devrait augmenter. Il faudrait donc trouver une heuristique afin de mesurer l'activité de l'opérateur et de régler le paramètre en fonction.

En ce qui concerne la planification de mouvement au contact, il existe plusieurs possibilités d'amélioration de notre contribution. En premier lieu, il serait intéressant de tester

une variante de notre algorithme de planification au contact où les rotations pendant le mode contact ne sont pas fixes. Cela permettrait de soulager l'opérateur pour le réglage de l'orientation pendant le mode contact et de tester toutes les possibilités aléatoirement. Il peut exister de nombreux cas où le changement automatique d'orientation est utile. Supposons par exemple, qu'un objet doive glisser sur une surface et passer en dessous d'un autre obstacle. L'opérateur n'aurait alors qu'à poser l'objet contre la surface de glissement et l'algorithme trouvera au bout d'un moment une configuration qui permette de passer en dessous du deuxième obstacle.

En second lieu, nous pourrions implémenter une version de l'algorithme où l'échantillonnage au contact se limite aux bornes de la surface choisie, un triangle dans le cas où les modèles sont des meshs. Cela serait utile si les modèles possèdent de grandes surfaces mais contre productif dans le cas contraire.

Il serait également intéressant de modifier la façon dont commence ou termine le mode contact c'est-à-dire les modalités d'interaction de ce mode. La planification au contact pourrait par exemple commencer automatiquement dès la première détection d'une collision. Cela permettrait d'adapter cette méthode d'échantillonnage au contact pour des algorithmes totalement automatiques.

Nous pourrions aussi changer la manière dont se termine le mode contact, par exemple à la première collision contre une deuxième surface ou bien lorsqu'un échantillon est totalement dans l'espace libre, c'est-à-dire s'il n'est plus en contact avec la surface. L'intérêt de cette méthode permettrait de quitter le mode contact automatiquement ce qui serait utile dans le cadre d'un planificateur automatique mais pas forcément intéressant dans un cadre interactif.

Il serait particulièrement intéressant de prendre en compte plusieurs surfaces pour le mode contact. À l'arrivée au contact d'une seconde surface, il serait utile de se déplacer le long de l'espace d'intersection entre ces deux surfaces. Ce mécanisme pourrait être étendu à plus de deux surfaces : trois, quatre et plus. Un tel algorithme au contact déplacerait alors un objet au contact d'une, puis deux, puis trois surfaces et ainsi de suite jusqu'à se retrouver dans l'impossibilité de faire glisser l'objet. Une telle stratégie permettrait alors d'explorer des recoins très difficilement atteignables par un échantillonnage automatique. Échantillonner aléatoirement une configuration au contact, par exemple, de quatre surfaces a une probabilité qui tend vers zéro. Cela est encore plus vrai s'il faut glisser le long de ces surfaces. Or un tel algorithme serait capable de faire cette opération. Accumuler ainsi de nombreux contacts serait particulièrement utile dans le cadre de tâches d'insertion.

Le pendant de la précédente stratégie serait de prendre en compte les surfaces voisines comme précédemment mais cette fois-ci en ne gardant qu'un seul contact pour un but différent. Il serait bon de pouvoir projeter un échantillon au contact sur une surface voisine. Cela permettrait de planifier au contact même si les deux surfaces ne sont pas coplanaires. À partir d'une deuxième surface, ce même mécanisme pourrait être activé permettant de planifier au contact d'une troisième surface. Le contact se transmettant ainsi de proche en

proche, cela permettrait de planifier au contact de tout type de surface. L'utilité d'un tel algorithme serait de garder un contact sur un objet géométrique très complexe permettant donc le suivi de surfaces décomposées en de multiples facettes. Cela serait une bonne approximation de ce qu'un humain est capable de faire lorsqu'il suit une surface courbe, alors que notre présente contribution au contact n'en est pas capable.

L'algorithme gardant le plus de contacts possibles afin de s'insérer dans des espaces de moins en moins accessibles pourrait être appelé algorithme conservateur. Celui qui cherche un seul contact en libérant à chaque fois le précédent pour le suivi de surfaces pourrait être appelé algorithme libérateur. Les deux algorithmes ont des applications très différentes et pourraient tous deux être utilisés pour résoudre un même problème.

En ce qui concerne notre contribution de planification haptique avec détection d'intention, la principale voie d'amélioration de ces travaux serait de pouvoir automatiquement configurer les nombreux paramètres de l'algorithme interactif et de l'algorithme de planification au contact. En effet, ces paramètres peuvent avoir une influence considérable sur la vitesse de résolution du problème, comme pour l'algorithme I-RRT-C.

Il serait intéressant de pouvoir gérer le retour d'effort haptique en fonction de l'environnement. Le retour haptique implique une certaine rigidité sur le retour d'effort lorsqu'un contact survient. Cette rigidité est réglée par un gain et est la même dans toutes les directions. Or il serait intéressant de modifier cette rigidité selon certains axes en fonction de la tâche à effectuer de la même manière que Peternel le fait dans sa contribution avec détection d'intention [58]. Cela veut dire qu'un axe aurait besoin d'être très rigide pendant qu'un autre ne le devrait pas pour permettre à l'opérateur de se déplacer prioritairement selon cette direction.

Une autre voie d'amélioration serait d'être capable de gérer des chaînes cinématiques portant OBJ. Cela permettrait par exemple de simuler un bras humain portant l'objet. Les simulations d'assemblage gagneraient ainsi en réalisme. En effet, une simulation d'assemblage ne prenant pas en compte l'opérateur peut mener à un résultat positif (une trajectoire faisable) tout en comportant un risque de ne pas pouvoir être effectué par un humain car nécessitant des mouvements impossibles à réaliser. L'idée précédente de gestion des rigidités aurait un intérêt particulier dans ce cadre car elles simuleraient le comportement d'un bras humain lorsqu'il est en collision de plusieurs surfaces : il garde une forte rigidité dans une certaine direction tout en étant plus souple dans d'autres.

Enfin, nos deux contributions techniques mériteraient une étude clinique afin d'analyser leur ergonomie face à une variété de sujets. Cela permettrait d'évaluer l'efficacité de nos algorithmes, avec des opérateurs de différents niveaux d'expertise y compris des personnes non spécialistes du domaine.

Bibliographie

[1] Ibrahim Al-Bluwi, Thierry Siméon, and Juan Cortés. Motion planning algorithms for molecular simulations : A survey. *Computer Science Review*, 6(4) :125–143, 2012.

[2] Nancy M Amato, O Burchan Bayazit, and Lucia K Dale. Obprm : An obstacle-based prm for 3d workspaces. 1998.

[3] Roger Baumann. Haptic interface for virtual reality based laparoscopic surgery training environment. *Unpublished doctoral dissertation*, 1734, 1997.

[4] Benjamin Bayart and Abderrahmane Kheddar. Evaluation of an haptic step-guidance algorithm through a teaching 2d/3d path scenario. In *Haptic, Audio and Visual Environments and Games, 2007. HAVE 2007. IEEE International Workshop on*, pages 124–129. IEEE, 2007.

[5] Benjamin Bayart, Aurélien Pocheville, and Abderrahmane Kheddar. An adaptive haptic guidance software module for i-touch : example through a handwriting teaching simulation and a 3d maze. In *Haptic Audio Visual Environments and Their Applications, 2005. IEEE International Workshop on*, pages 6–pp. IEEE, 2005.

[6] O Burchan Bayazit, Guang Song, and Nancy M Amato. Enhancing randomized motion planners : Exploring with haptic hints. *Autonomous Robots*, 10(2) :163–174, 2001.

[7] Nassime Blin, Michel Taïx, Philippe Fillatreau, and Jean-Yves Fourquet. Interactive motion planning with contact. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2016*, 2016.

[8] Nassime Blin, Michel Taïx, Philippe Fillatreau, and Jean-Yves Fourquet. Tuning interaction in motion planning with contact. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pages 1000–1006. IEEE, 2016.

[9] Nassime Blin, Michel Taïx, Philippe Fillatreau, and Jean-Yves Fourquet. H-rrt-c : Haptic motion planning with contact. In *IEEE RO-MAN2017 26th IEEE International Symposium on Robot and Human Interactive Communication*, 2017.

[10] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 521–528. IEEE, 2000.

[11] Thorsten Brandt, Thomas Sattel, and Michael Bohm. Combining haptic human-machine interaction with predictive path planning for lane-keeping and collision avoidance systems. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 582–587. IEEE, 2007.

[12] Simon Cailhol. *Planification interactive de trajectoire en Réalité Virtuelle sur la base de données géométriques, topologiques et sémantiques.* PhD thesis, 2015. Thèse de doctorat dirigée par Fourquet, Jean-Yves et Fillatreau, Philippe Automatique et robotique Toulouse, INPT 2015.

[13] Simon Cailhol, Philippe Fillatreau, Jean-Yves Fourquet, and Yingshen Zhao. A hierarchic approach for path planning in virtual reality. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 9(4) :291–302, 2015.

[14] Simon Cailhol, Philippe Fillatreau, Yingshen Zhao, and Jean-Yves Fourquet. Hierarchic interactive path planning in virtual reality. In *Informatics in Control, Automation and Robotics*, pages 179–203. Springer, 2016.

[15] John Canny. *The complexity of robot motion planning.* MIT press, 1988.

[16] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3555–3561. IEEE, 2016.

[17] Howie M Choset. *Principles of robot motion : theory, algorithms, and implementation.* MIT press, 2005.

[18] Juan Cortes, Thierry Siméon, and Jean-Paul Laumond. A random loop generator for planning the motions of closed kinematic chains using prm methods. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 2141–2146. IEEE, 2002.

[19] Jory Denny, Read Sandström, Nicole Julian, and Nancy M Amato. A region-based strategy for collaborative roadmap construction. In *Algorithmic Foundations of Robotics XI*, pages 125–141. Springer, 2015.

[20] Michel J Dugas, Kylie Francis, and Stéphane Bouchard. Cognitive behavioural therapy and applied relaxation for generalized anxiety disorder : a time series analysis of change in worry and somatic anxiety. *Cognitive behaviour therapy*, 38(1) :29–41, 2009.

[21] Philippe Fillatreau, J-Y Fourquet, Romain Le Bolloc'H, Simon Cailhol, Adrien Datas, and Bernard Puel. Using virtual reality and 3d industrial numerical models for immersive interactive checklists. *Computers in Industry*, 64(9) :1253–1262, 2013.

[22] David Flavigné and Michel Taïx. Improving motion planning in weakly connected configuration spaces. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5900–5905. IEEE, 2010.

[23] Philippe Fuchs, Guillaume Moreau, and Alain Berthoz. Le traité de la réalité virtuelle volume 1 : L'homme et l'environnement virtuel, 2006.

[24] Han Gyeol Gang, Jiong Min Park, Seung-Bok Choi, and Jung Woo Sohn. Development of haptic system for surgical robot. In *Active and Passive Smart Structures and Integrated Systems 2017*, volume 10164, page 101641N. International Society for Optics and Photonics, 2017.

[25] Sylvie Gibet and Pierre-François Marteau. Analysis of human motion, based on the reduction of multidimensional captured data–application to hand gesture compression, segmentation and synthesis. *Articulated Motion and Deformable Objects*, pages 72–81, 2008.

[26] Florian Gosselin, Claude Andriot, Joan Savall, and Javier Martín. Large workspace haptic devices for human-scale interaction : A survey. *Haptics : Perception, Devices and Scenarios*, pages 523–528, 2008.

[27] Sébastien Grange, Terrence Fong, and Charles Baur. M/oris : a medical/operating room interaction system. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 159–166. ACM, 2004.

[28] Danny A Grant and Juan M Cruz-Hernandez. Method and apparatus for providing a fixed relief touch screen with locating features using deformable haptic surfaces, November 21 2007. US Patent App. 11/943,862.

[29] Syed Hassan and Jungwon Yoon. Haptic guided optimized aircraft maintenance assembly disassembly path planning scheme. In *Control Automation and Systems (ICCAS), 2010 International Conference on*, pages 1667–1672. IEEE, 2010.

[30] Xuejian He and Yonghua Chen. Haptic-aided robot path planning based on virtual tele-operation. *Robotics and computer-integrated manufacturing*, 25(4) :792–803, 2009.

[31] Hirohisa Hirukawa and Yves Papegay. Motion planning of objects in contact by the silhouette algorithm. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 722–729. IEEE, 2000.

[32] Xiaolei Hou, Xiaohua Wang, and Robert Mahony. Haptics-aided path planning and virtual fixture based dynamic kinesthetic boundary for bilateral teleoperation of vtol aerial robots. In *Control Conference (CCC), 2016 35th Chinese*, pages 4705–4710. IEEE, 2016.

[33] David Hsu, Gildardo Sánchez-Ante, and Zheng Sun. Hybrid prm sampling with a cost-sensitive adaptive strategy. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3874–3880. IEEE, 2005.

[34] Jian Huang, Weiguang Huo, Wenxia Xu, Samer Mohammed, and Yacine Amirat. Control of upper-limb power-assist exoskeleton using a human-robot interface based on motion intention recognition. *IEEE Transactions on Automation Science and Engineering*, 12(4) :1257–1270, 2015.

[35] Saurabh Jadhav, Vikas Kannanda, Bocheng Kang, Michael T Tolley, and Jurgen P Schulze. Soft robotic glove for kinesthetic haptic feedback in virtual reality environments. *Electronic Imaging*, 2017(3) :19–24, 2017.

[36] Claudia Elvira Esteves Jaramillo. *Motion planning : from digital actors to humanoid robots*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2007.

[37] George T Katanics and Philip A Groves. Method and apparatus for an interactive video game with physical feedback, April 11 1995. US Patent 5,405,152.

[38] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4) :566–580, 1996.

[39] Wisama Khalil and Etienne Dombre. *Modélisation, identification et commande des robots*. Hermès science publ., 1999.

[40] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1) :90–98, 1986.

[41] Dalhyung Kim, Woojin Chung, and Shinsuk Park. Practical motion planning for car-parking control in narrow environment. *IET control theory & applications*, 4(1) :129–139, 2010.

[42] Nicolas Ladeveze, Jean-Yves Fourquet, and Bernard Puel. Interactive path planning for haptic assistance in assembly tasks. *Computers & Graphics*, 34(1) :17–25, 2010.

[43] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[44] Jean-Paul Laumond, Mehdi Benallegue, Justin Carpentier, and Alain Berthoz. The yoyo-man. In *Robotics Research*, pages 217–233. Springer, 2018.

[45] Steven M LaValle. Rapidly-exploring random trees : A new tool for path planning. 1998.

[46] A. LaViers, C. Cuan, M. Heimerdinger, U. Huzaifa, C. Maguire, A. Niles, K. Bradley, K. Brooks Mata, R. McNish, I. Vidrin, and A. Zurawski. Choreographic and somatic approaches for the development of expressive robotic systems. In *Non publié*, 2017.

[47] Anatole Lécuyer, Abderrahmane Kheddar, Sabine Coquillart, Ludovic Graux, and Philippe Coiffet. A haptic prototype for the simulations of aeronautics mounting/unmounting operations. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 182–187. IEEE, 2001.

[48] Junghwan Lee, OSung Kwon, Liangjun Zhang, and Sung-eui Yoon. Sr-rrt : Selective retraction-based rrt planner. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2543–2550. IEEE, 2012.

[49] Tomas Lozano-Perez. Spatial planning : A configuration space approach. *IEEE transactions on computers*, (2) :108–120, 1983.

[50] Thomas H Massie, J Kenneth Salisbury, et al. The phantom haptic interface : A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, volume 55, pages 295–300. Chicago, IL, 1994.

[51] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2) :129–147, 1982.

[52] Stéphane Mercier, Catherine Tessier, and Frédéric Dehais. Authority management in human-robot systems. *IFAC Proceedings Volumes*, 43(13) :410–415, 2010.

[53] Joseph Mirabel, Steve Tonneau, Pierre Fernbach, Anna-Kaarina Seppälä, Mylène Campana, Nicolas Mansard, and Florent Lamiraux. Hpp : A new software for constrained motion planning. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 383–389. IEEE, 2016.

[54] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[55] Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. Rt-rrt* : a real-time path planning algorithm based on rrt. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 113–118. ACM, 2015.

[56] Michael Otte and Emilio Frazzoli. Rrtx : Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7) :797–822, 2016.

[57] Jia Pan, Christian Lauterbach, and Dinesh Manocha. g-planner : Real-time motion planning and global navigation using gpus. In *AAAI*, 2010.

[58] Luka Peternel, Tadej Petrič, and Jan Babič. Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1497–1502. IEEE, 2015.

[59] Nicolas Pronost. *Définition et réalisation d'outils de modélisation et de calcul de mouvement pour des humanoïdes virtuels*. PhD thesis, Rennes 1, 2006.

[60] Stephane Redon and Ming C Lin. Practical local planning in the contact space. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4200–4205. IEEE, 2005.

[61] Rodriguez, Xinyu Tang, Jyh-Ming Lien, and N. M. Amato. An obstacle-based rapidly-exploring random tree. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 895–900, May 2006.

[62] Mitul Saha, Jean-Claude Latombe, Yu-Chi Chang, and Friedrich Prinz. Finding narrow passages with probabilistic roadmaps : The small-step retraction method. *Autonomous robots*, 19(3) :301–319, 2005.

[63] Abraham Sanchez, J Abraham Arenas, and René Zapata. Non-holonomic path planning using a quasi-random prm approach. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2305–2310. IEEE, 2002.

[64] Jean-Philippe Saut, Anis Sahbani, Sahar El-Khoury, and Véronique Perdereau. Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2907–2912. IEEE, 2007.

[65] Jean-Philippe Saut, Anis Sahbani, and Véronique Perdereau. Generic motion planner for robot multi-fingered manipulation. *Advanced Robotics*, 25(1-2) :23–46, 2011.

[66] Erhard Schmidt. Über die auflösung linearer gleichungen mit unendlich vielen unbekannten. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 25(1) :53–77, 1908.

[67] Jacob T Schwartz and Micha Sharir. On the piano movers' problem : Iii. coordinating the motion of several independent bodies : The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2(3) :46–75, 1983.

[68] Lorenzo Sciavicco, Bruno Siciliano, and T Herbert. Modeling and control of robot manipulators. *Journal of Intelligent and Robotic Systems*, 21(1) :99, 1998.

[69] Bo Shen and Shuoyu Wang. An independent life support robot for the lower-limb handicapped and elderly : Task-intention-identification and assistive-motion-planning algorithms. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 1169–1176. IEEE, 2015.

[70] Thomas B Sheridan and William L Verplank. Human and computer control of undersea teleoperators. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE MAN-MACHINE SYSTEMS LAB, 1978.

[71] Arne Sieverling, Clemens Eppner, and Oliver Brock. Interleaving motion in contact and in free space for planning under uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)-(under review)*, 2017.

[72] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[73] Mike Stilman. Task constrained motion planning in robot joint space. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3074–3081. IEEE, 2007.

[74] Andreas Tobergte, Patrick Helmer, Ulrich Hagn, Patrice Rouiller, Sophie Thielmann, Sébastien Grange, Alin Albu-Schäffer, François Conti, and Gerd Hirzinger. The sigma. 7 haptic interface for mirosurge : A new bi-manual surgical console. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3023–3030. IEEE, 2011.

[75] Steve Tonneau. *Synthèse de mouvement pour des personnages virtuels en environnements contraints*. PhD thesis, Rennes, INSA, 2015.

[76] Zhikun Wang, Katharina Mülling, Marc Peter Deisenroth, Heni Ben Amor, David Vogt, Bernhard Schölkopf, and Jan Peters. Probabilistic movement modeling for intention inference in human–robot interaction. *The International Journal of Robotics Research*, 32(7) :841–858, 2013.

[77] Yu Yan, Emilie Poirson, and Fouad Bennis. Integrating user to minimize assembly path planning time in plm. In *IFIP International Conference on Product Lifecycle Management*, pages 471–480. Springer, 2013.

[78] Lei Zhang, Huaxi Zhang, Zheng Fang, René Zapata, and Marianne Huchard. A domain specific architecture description language for autonomous mobile robots. In *Information and Automation (ICIA), 2012 International Conference on*, pages 283–288. IEEE, 2012.

[79] Liangjun Zhang and Dinesh Manocha. An efficient retraction-based rrt planner. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3743–3750. IEEE, 2008.