HAL
open science

Planification de Chemin et adaptation de posture en environnement dynamique

Thomas Lopez

To cite this version:

Thomas Lopez. Planification de Chemin et adaptation de posture en environnement dynamique. Synthèse d'image et réalité virtuelle [cs.GR]. INSA de Rennes, 2012. Français. NNT : . tel-00767784

HAL Id: tel-00767784

https://theses.hal.science/tel-00767784

Submitted on 20 Dec 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

THÈSE INSA Rennes
sous le sceau de l'Université Européenne de Bretagne
pour obtenir le grade de
DOCTEUR DE L'INSA DE RENNES
Spécialité : Informatique

présentée par
Thomas LOPEZ
ÉCOLE DOCTORALE : MATISSE
LABORATOIRE : IRISA – UMR6074

Planification de chemin et adaptation de posture en environnement dynamique

Thèse soutenue le 9 Mars 2012
devant le jury composé de :

Bruno ARNALDI
Professeur à l'INSA de Rennes - IRISA / *Président*
Thierry SIMÉON
Directeur de recherche - LAAS-CNRS / *Rapporteur*
Céline LOSCOS
Professeur à l'Université de Reims - CReSTIC / *Rapporteur*
Jean-Pierre JESSEL
Professeur à l'Université Paul Sabatier - IRIT / *Examinateur*
Tsai-Yen Li
Professeur à National ChengChi University - Taiwan / *Examinateur*
Stéphane DONIKIAN
Directeur de Recherche INRIA Rennes - Golaem / *Directeur de thèse*
Fabrice LAMARCHE
Maître de conférence à l'Université de Rennes 1 - IRISA / *Encadrant de thèse*

Remerciements

Je tiens tout d'abord à remercier les différents membres de mon jury de thèse. Merci à Bruno Arnaldi qui m'a fait l'honneur de présider ce jury afin de conclure mes longues années d'études à l'INSA. Merci à Céline Loscos et à Thierry Siméon d'avoir accepté d'être mes rapporteurs de thèse ainsi que pour leurs rapports à la fois constructifs et gratifiants sur mes travaux. Merci enfin à Jean-Pierre Jessel qui a pu faire le déplacement afin de participer également à ce jury.

I would also have a special thanks for Tsai-Yen Li for coming from Taiwan especially for my PhD thesis defense. Thank you also Tsai-Yen for welcoming me in your lab twice. Those two stays were a really fulfilling experience for me and I will keep that souvenir with me ! Thanks to all the people in IM Lab for their really warm welcoming, I hope I will have the opportunity to meet everyone again, in Taiwan or anywhere else.

Je tiens également à remercier Stéphane Donikian d'avoir accepté de diriger cette thèse avec Fabrice Lamarche. Un grand merci ensuite à Fabrice qui a été mon encadrant pendant ces 3 ans, 5 mois et 9 jours ! Merci car j'ai été le premier doctorant (mais certainement pas le dernier) que tu as encadré, Merci également car tu as réussis à me garder pendant ces 1255 jours dans ton bureau sans jamais craquer (ou presque) ! ! ! Merci à toi pour ta disponibilité, pour les longs échanges et débats d'idées qui m'ont permis d'avancer pendant cette thèse. Que ce soit autour d'un café, d'une pause cigarette ou devant le tableau blanc, tous ces échanges m'ont enrichit et sont à l'origine des idées exposées aujourd'hui dans ce manuscrit ! Enfin, merci à toi de m'avoir toujours soutenu et d'avoir toujours cru en moi et en nos recherches même dans des moments de bonnes galères où les choses ne semblaient pas aller dans le bon sens... Merci enfin pour la bonne humeur dans le travail, pour avoir supporté ma musique dans le bureau pendant mon travail, pour les soirées rédactions jusqu'à pas d'heure ensemble et pour les débats philosophiques pendant les soirées de séminaire au vert (notamment autour du conte des trois petits cochons et du grand méchant loup). Merci encore pour bien des choses, et bon courage avec la relève des doctorants qui ont déjà commencé à me succéder et qui ont l'air de vouloir te donner du fil à retordre ! C'est en tous cas avec un grand plaisir que j'espère pouvoir continuer à collaborer avec toi pour la suite.

Je tiens également à remercier toutes les personnes de l'IRISA / INRIA que j'ai été amené à côtoyer depuis mon master jusqu'à aujourd'hui. Merci tout d'abord à tous les membres de l'ex-équipe de recherche BUNRAKU qui sont aujourd'hui séparés entre MimeTIC, VR4I et FRVsense. Je remercie tout d'abord Kadi Bouatouch qui m'a

permis d'approcher la recherche en m'encadrant durant mon master. Merci également Kadi pour les nombreux conseils que tu as pu me donner au cours de ma thèse. Merci à tous les permanents que j'ai côtoyé durant ces 4 années pour leur accueil et leur bonne humeur toujours présente qui ont fait de Bunraku l'une des équipes les plus conviviales que l'IRISA ! ! ! Merci également à tous les membres du M2S !
Je tiens à remercier aussi les nombreux ingénieurs, doctorants, post-docs et assistantes qui sont passés dans l'équipe au cours de ces années et qui ont participé à ce doctorat à leur manière ! Les nombreux repas ensemble et les pauses cafés plus que quotidiennes ont participé à la vie et à la bonne ambiance de l'équipe ! Désolé de ne pas pouvoir tous vous citer ici, la liste serait bien heureusement trop longue pour tenir sur ces pages !
Merci aussi à l'AGOS et particulièrement à Marie, qui m'ont permis de découvrir, de me développer et de m'améliorer dans de nombreux domaines extra-professionnels pendant ces quatre ans. Merci de nous aider à nous aérer les neurones et nous éloigner un peu de l'écran de temps à autre en nous mettant à disposition les ressources nécessaires pour participer à des activités telles que le Hockey-sur-Glace, le Squash ou le Badminton. Merci également à tous les sportifs de l'IRISA et d'en dehors que j'ai pu faire courir (ou qui m'ont fait courir) sur les divers terrains de sport. Merci à Pierrette, Laurence et les diverses personnes de la cafet pour leur accueil quotidien qui participe à faire de ce lieu de travail un endroit convivial ! Une petite pensée à tous les cafés qui m'ont permis de rester éveillé et aux centaines de touillettes sacrifiées pendant cette thèse !

Je tiens enfin à remercier toutes les personnes proches qui m'ont soutenues durant cette thèse. Merci à ma famille, à commencer par mes parents, merci d'avoir toujours cru en moi et de m'avoir donné depuis toujours l'opportunité de poursuivre mes objectifs tant personnels que professionnels. Un grand merci à ma sœurette adorée Lisa, à son mari Greg, merci pour votre présence et votre bonne humeur qui me font du bien :). Un merci aussi pour ma grand-mère Adrienne qui m'a toujours soutenu également. Je remercie tous mes amis : Tout d'abord ma petite femme et grand frère adoré, Ludo, ta présence à toujours été rassurante ! Merci d'être là, même malgré la distance ! Merci à Antho qui a pris ton relai avec une grande dévotion et que c'est un plaisir de compter parmi mes amis ! Un grand merci à ma tamagoshinette, Charline, pour ta présence sans faille dans les moments où j'avais besoin, même en étant à l'autre bout du globe ! Je remercie également mes amis de promo de l'INSA avec qui l'on arrive toujours à se retrouver, malgré mon emploi du temps supra-chargé ! Merci à mes co-bureaux, Philippe et Carl ! J'espère que ça n'a pas été trop dur de me supporter ;) prenez soin de Fab et ne l'amochez pas trop ! ! ! Faut qu'il puisse encore servir après vous ! Merci à mes petites sœurs de l'IRISA pour les moments passés ensembles et pour m'avoir permis de m'améliorer en anglais ! Merci à Aurore, pour nos nombreuses pauses café, pour nos discussions et un merci spécial pour mon pot de thèse ! Enfin pour terminer, je tiens à remercier des connaissances plus récentes, mais non moins importantes ! ! Merci à vous Claire, Julie, Émeric, *mon petit*, pour les nombreux moments passés dernièrement ensemble qui m'ont fait un bien fou au moral et m'ont permis de me retrouver un peu ! Enfin, merci à toi que je n'ai appris à connaître que récemment mais auprès de qui je me sens heureux et moi-même. Merci de m'équilibrer et d'être là pour partager ça avec moi, même dans mes moments de folie douce ;) !

Table des matières

Liste des figures iii

Introduction 1
- 1 Contexte de recherche . . . 1
- 2 Problématique . . . 4
- 3 Contributions . . . 6
- 4 Organisation de ce mémoire . . . 7

1 Bibliographie 9
- 1 Introduction à la planification de chemin . . . 10
- 2 Planification de chemin en environnements statiques . . . 11
  - 2.1 Décomposition en cellules . . . 12
  - 2.2 Cartes de cheminement . . . 16
- 3 Planification de chemin en environnements statiques avec des éléments dynamiques . . . 21
  - 3.1 Adaptation ponctuelle de la représentation de l'environnement pour la planification de chemin . . . 22
  - 3.2 Adaptation dynamique de la représentation de l'environnement pour la planification de chemin . . . 26
  - 3.3 Planification de trajectoire . . . 28
- 4 Planification de chemin en environnements complexes dynamiques . . . 30
- 5 Planification de chemin et mouvements de l'agent . . . 33
- 6 Conclusion . . . 38

2 Représentation de l'agent et des objets de l'environnement 41
- 1 Représentation de l'agent . . . 42
  - 1.1 Découpler animation de l'agent et planification de son chemin . . . 42
  - 1.2 Caractérisation de l'agent . . . 43
- 2 Caractérisation des objets . . . 45
  - 2.1 Préambule . . . 46
  - 2.2 Les Volumes d'Interaction : Caractérisation de l'impact topologique d'un objet . . . 47
  - 2.3 Mouvements des objets et Volumes d'Interaction . . . 57
- 3 Conclusion . . . 60

3 Représentation dynamique de la topologie 63

1 Caractérisation des relations entre deux objets de l'environnement 64
1.1 Caractérisation des accessibilités et obstructions entre deux objets . 64
1.2 Détection des interactions entre objets 67
2 Représentation de la topologie globale et de son évolution au cours du temps 69
2.1 Le Graphe Topologique 69
2.2 Représentation de la topologie et de son évolution au cours du temps 75
2.3 Suivi des modifications de Volumes d'Interaction 80
3 Conclusion 81

4 Planification de trajectoire en environnement dynamique 83
1 Planification de la trajectoire locale 84
1.1 Représentation de la topologie locale à un élément 84
1.2 Anticipation de l'évolution des obstructions locales dans le domaine spatio-temporel 87
1.3 Planification locale dans le domaine spatio-temporel 90
1.4 Réactivité de l'agent pendant la navigation 94
2 Planifier le déplacement de l'agent dans l'environnement dynamique 95
2.1 Raffinement des relations topologiques 96
2.2 Calcul et optimisation du chemin topologique 99
3 Planification hiérarchique 103
3.1 Association de la planification globale et locale 103
3.2 Gestion d'une obstruction de la Surface Navigable 104
4 Conclusion 108

5 Présentation des résultats obtenus 111
1 Architecture globale 111
2 Définition des environnements 113
2.1 Représentation de l'agent 113
2.2 Représentation pratique des éléments de l'environnement virtuel . . 114
3 Environnements de tests 115
4 Performances observées 116
5 Conclusion 118

Conclusion 119

Publications 125

Liste des références 125

Table des figures

1.1 Subdivision à l'aide de grilles régulières 13
1.2 Subdivision de l'environnement à l'aide de cylindres 14
1.3 Décomposition exacte basée sur une triangulation de Delaunay 15
1.4 Représentations de l'environnement basées sur des diagrammes de Voronoï . 18
1.5 Échantillonnage probabiliste de l'espace des configurations 19
1.6 Rapidly-exploring Random Trees et RRT-Connect 21
1.7 Extensions des RRTs pour la planification de chemin avec des obstacles dynamiques 24
1.8 Exemples de cartes de cheminement mises à jour dynamiquement 27
1.9 Planification de trajectoire permettant d'anticiper les obstacles dynamiques 28
1.10 Planification de trajectoire Utilisation des éléments dynamiques pour la navigation 31
1.11 Exemples de volumes englobants 34
1.12 Motion Patches 35
1.13 Arbres de recherche précalculés 37

2.1 Représentation de l'agent à l'aide de cylindres englobants 43
2.2 Représentation des transitions entre les capacités de déplacement à l'aide d'un graphe 45
2.3 Influence de la pente navigable sur la définition de la Surface Navigable . . 48
2.4 Caractérisation de la Surface Navigable pour une caisse 49
2.5 Caractérisation de la Surface Navigable pour un canapé 50
2.6 Influence des Surfaces Navigables sur la construction des Volumes Interdits 51
2.7 Caractérisation du Volume Interdit pour une caisse 52
2.8 Caractérisation du Volume Interdit pour un canapé 52
2.9 Définition du profil d'accessibilité 53
2.10 Caractérisation du Volume d'Accessibilité pour une caisse 55
2.11 Caractérisation du Volume d'Accessibilité pour un canapé 56
2.12 Degrés de liberté associé à un objet de l'environnement 57
2.13 Rotation d'un objet sans mise à jour des Volumes d'Interaction 58
2.14 Rotation d'un objet avec mise à jour des Volumes d'Interaction 60
2.15 Extraction des Volumes d'Interaction dans l'espace des configurations . . . 61

3.1 Caractérisation de l'accessibilité 65
3.2 Obstruction de la Surface Navigable d'un objet 66
3.3 Obstruction du Volume d'Accessibilité d'un objet 67

3.4 Mise à jour du Graphe Topologique 68
3.5 Caractérisation des relations topologiques 70
3.6 Mise à jour du Graphe Topologique 71
3.7 Exemple de fausse accessibilité 72
3.8 Aperçu de l'environnement de test 73
3.9 Exemple de détection d'obstructions 74
3.10 Exemple de détection d'accessibilités 76
3.11 Graphe Topologique associé à l'environnement de test 77
3.12 Détection de chemin à travers des surfaces déconnectées 78
3.13 Rappel de la mise à jour des Volumes d'Interaction 80

4.1 Exemple de construction des cartes de cheminement locales 86
4.2 Représentation des obstacles dans le domaine spatio-temporel 89
4.3 Test de validation des arcs 93
4.4 Fausse accessibilité 96
4.5 Échantillonnage d'une accessibilité entre deux surfaces navigables 97
4.6 Planification haut niveau dans le Graphe Topologique 100
4.7 Pondération des arcs d'accessibilité dans le Graphe Topologique et planification topologique 101
4.8 Environnement nécessitant une duplication 104
4.9 Graphe Topologique de l'environnement avant duplication 105
4.10 Graphe Topologique de l'environnement après duplication 109
4.11 Exemple de duplications multiples dans le Graphe Topologique 110

5.1 Vue d'ensemble de la structure de planification 112
5.2 Adaptation de posture de l'agent au cours de la navigation 114
5.3 Détection des changements topologiques à la volée 115
5.4 Exemples d'environnements de tests utilisés 116
5.5 Exemple d'environnement utilisant un moteur physique 117

Introduction

Les mondes virtuels sont aujourd'hui utilisés dans de nombreux domaines applicatifs. Ces mondes virtuels sont généralement peuplés à l'aide d'agents autonomes qui rendent ces environnements plus vivants. Dans le cadre de cette thèse nous nous sommes intéressés à la navigation de tels agents autonomes au sein de ces environnements virtuels. La planification de chemin est une tâche cruciale pour ces agents puisqu'elle va influencer directement leur autonomie de mouvement et leur capacité à agir au sein de ces mondes. Nous nous sommes focalisés au cours de nos travaux sur la planification de chemin pour des agents peuplant des environnements dynamiques dont la configuration évolue au cours de la simulation de manière non connue a priori. Afin de répondre aux contraintes temporelles imposées par de nombreuses applications interactives, telles que les jeux vidéo, nous nous sommes également donné l'objectif de proposer une solution de planification assez performante pour être compatible avec de telles applications. Ce mémoire de thèse présente donc une nouvelle méthode de planification de chemin au sein d'environnements dynamiques non connus a priori et compatible avec des applications interactives.

1 Contexte de recherche

La planification de chemin est une problématique de recherche qui a été largement abordée dans le domaine de la robotique. Que ce soit pour des robots se déplaçant dans des environnements réels ou pour des agents se déplaçant dans des environnements virtuels, le fond de cette problématique reste le même. Dans les deux cas, l'entité (robot ou agent virtuel) doit construire ou posséder une représentation de l'environnement l'entourant afin de pouvoir naviguer de manière autonome au sein de cet environnement. Cette capacité de navigation est cruciale puisqu'elle va permettre à un agent d'aller à la découverte de son environnement, augmentant ainsi ses possibilités d'action avec cet environnement ainsi qu'avec les autres agents qui le peuplent. Ces deux domaines ont donc, de manière similaire, cherchés à proposer des nouvelles solutions à la navigation autonome de l'entité. Toutefois, de nombreux points doivent être soulevés afin de doter un agent de cette autonomie de déplacement. L'entité doit tout d'abord être capable d'appréhender sa présence et sa géométrie au sein de l'environnement et de pouvoir s'y localiser, répondant ainsi à la question "Où suis-je ?". La configuration de l'environnement doit ensuite être identifiée, afin que l'entité déduise les accessibilités et obstructions existantes, répondant ainsi à la problématique "Où puis-je me rendre et comment ?". Finalement l'agent va devoir être à même de trouver son chemin au sein de cet environnement, répondant à la question "Comment puis-je atteindre cette

destination ?". Ces problématiques ont donc tout d'abord été traitées dans le domaine de la robotique depuis plusieurs décennies. Le développement récent, et de plus en plus important, des environnements virtuels a amené à adapter de nombreuses méthodes issues de la robotique à des agents virtuels et à rechercher de nouvelles solutions.

Par leur utilisation dans des domaines très variés, les mondes virtuels vont présenter des propriétés très différentes en fonction des applications visées. Cependant, le peuplement de ces environnements est un point crucial dans l'ensemble de ces applications puisqu'il va permettre de rendre ces mondes virtuels plus "vivants" et donc de créer par conséquent une meilleure immersion de l'utilisateur. Il est en effet très rare d'être en présence de lieux totalement vides dans nos environnements quotidiens et la création d'un monde virtuel peuplé renforce donc la crédibilité de celui-ci. Le peuplement de ces environnements par des agents virtuels autonomes va appartenir au domaine de l'animation comportementale. L'autonomie de ces agents va dépendre des diverses capacités qu'ils vont posséder. Parmi celles-ci, la capacité de naviguer de manière autonome est l'une des aptitudes les plus importantes. Cette aptitude va en effet influer directement sur l'autonomie de déplacement de ces agents au sein de l'environnement ainsi que sur leur capacité à aller agir avec celui-ci et avec les autres agents.

La navigation des agents virtuels est donc un centre d'intérêt commun à l'ensemble des domaines applicatifs utilisant des environnements virtuels. Toutefois, les besoins et les contraintes imposées ne sont pas les mêmes en fonctions des applications considérées. Ainsi, ces domaines applicatifs peuvent être divisés en deux parties : les applications interactives, telles que les jeux vidéo ou encore la réalité virtuelle, et les applications de production, telles que le cinéma d'animation ou la gestion de cycles de vie des produits. Dans le cadre des applications interactives, l'accent est mis sur les temps de réponse des méthodes proposées afin qu'un utilisateur agissant avec l'environnement virtuel ait une réponse directe à ses actions. À l'inverse, dans le cadre des méthodes de production, le résultat final sera en général produit hors-ligne à l'aide de méthodes coûteuses en temps de calcul. Toutefois, il est intéressant dans ce cadre applicatif de posséder des méthodes de prévisualisation rapides afin d'obtenir à l'avance une idée du résultat final en économisant du temps de calcul.

Applications interactives

Les jeux vidéo. Les environnements virtuels sont depuis toujours très présents dans les jeux vidéo où le joueur doit généralement évoluer à l'aide d'un avatar le représentant. Ces mondes virtuels sont généralement peuplés d'agents virtuels autonomes avec qui le joueur va pouvoir interagir et qui vont permettre, par leurs actions, d'enrichir le jeu. Grâce aux améliorations des performances des processeurs et des cartes graphiques, ces environnements virtuels se complexifient afin d'être toujours plus riches, que ce soit par des rendus des décors toujours améliorés ou par l'intégration de plus en plus courante de moteurs de simulation physique. Afin de planifier leur chemin au sein de tels environnement, les agents virtuels vont devoir être capables de réagir rapidement aux actions de l'utilisateur ou du moteur physique afin de prendre en compte les

modifications topologiques créées et de proposer des interactions riches au joueur sans que celui-ci n'ait à attendre.

Les jeux sérieux. De nouvelles méthodes de formations, baptisées "Jeux sérieux" ou encore *Serious Games*, se proposent d'offrir des formations plus attractives à travers des jeux pédagogiques, informatifs ou encore industriels. L'utilisation d'environnements virtuels dans ce contexte va permettre une meilleure immersion de la personne formée et en conséquence un meilleur apprentissage. Une fois encore, l'une des contraintes principale de ce genre d'application va être d'avoir une simulation interactive afin que la personne formée soit au mieux immergée dans la formation qui lui est proposée. Tout comme précédemment pour les jeux vidéo, les agents virtuels vont donc devoir être capable de naviguer de manière autonome, tout en réagissant et en s'adaptant rapidement aux actions de l'utilisateur.

La réalité virtuelle. Le but de la réalité virtuelle est d'immerger un humain réel au sein d'un environnement virtuel. Afin d'augmenter les interactions offertes à l'utilisateur mais également d'augmenter son immersion, des agents virtuels autonomes vont pouvoir être intégrés à la simulation. Les agents peuplant ces environnements vont ainsi être amenés à se déplacer auprès de l'utilisateur, voir même à agir en collaboration avec celui-ci. Les solutions proposées dans ce cadre applicatif doivent donc être interactives au maximum afin de renforcer l'immersion de l'utilisateur en proposant des agents capables de réagir directement à ses actions, comme par exemple en se déplaçant avec lui lors d'une tâche collaborative.

Applications de production

Le cinéma. Le domaine cinématographique utilise depuis de nombreuses années les environnements virtuels afin de réaliser des films d'animation, des effets spéciaux s'intégrant au sein de scènes réelles ou encore à intégrer des acteurs virtuels (respectivement réels) dans des scènes réelles (respectivement virtuelles). Certains films ont par exemple fait appel à ces techniques pour générer des armées d'acteurs virtuels s'affrontant sur des champs de bataille. Le cinéma ne propose pas d'interaction entre les spectateurs et le film. Ainsi, afin d'avoir des rendus de meilleure qualité, il est possible d'utiliser des méthodes beaucoup plus coûteuses pour la génération des environnements et des acteurs virtuels. Toutefois, la problématique de la navigation est toujours présente afin de générer les chemins que vont emprunter les acteurs virtuels dans les scènes générées. De plus, dans le cadre de certaines scènes comme le champ de bataille cité précédemment, cette planification de chemin dans l'environnement va devoir être faite dans des environnements complexes et changeants.

La gestion du cycle de vie des produits. Ces gestions de cycles de vie des produits sont aussi appelées PLM ou *Product Lifecycle Management*. Afin de pouvoir tester à moindre coût les produits considérés, les environnements virtuels sont aujourd'hui largement utilisés afin de simuler les diverses étapes du cycle de vie d'un produit. De plus, l'utilisation d'environnements virtuels de plus en plus réalistes va permettre de créer des éléments et d'en tester des propriétés diverses telles que les caractéristiques mécaniques ou physiques. La simulation de chaînes de montage va pouvoir amener à prendre en compte des environnements, tels que des usines par exemple, où les agents

sont amenés à naviguer en prenant en compte le déplacement d'éléments tels que des escabeaux ou des échafaudages afin de s'adapter aux contraintes imposées par l'espace de travail. Une fois encore, ce domaine applicatif nécessite donc l'utilisation de méthodes de navigation.

L'architecture. En modélisant des constructions dans des environnements virtuels avant leur construction réelle, les architectes disposent aujourd'hui d'un moyen d'avoir un aperçu du rendu final de leur réalisation et de vérifier par exemple l'intégration de ces réalisations au sein d'environnements réels déjà existants en les reproduisant à l'identique dans un monde virtuel. Peupler ces environnements à l'aide d'agents virtuels naviguant de manière autonome va permettre une mise en situation de l'environnement afin de rendre le bâtiment plus "vivant" lors de présentations. Il sera également possible d'effectuer des simulations de panique afin de vérifier, avant construction, divers critères de sécurité concernant le bâtiment.

Comme nous pouvons le voir à travers ces différentes applications, la navigation est généralement une capacité essentielle requise pour les agents virtuels. À la vue de ces différents exemples, la définition du problème de planification et la complexité de celui-ci vont largement dépendre de l'utilisation prévue et de son caractère interactif ou non. Selon les situations, les environnements, les agents ou les contraintes à considérer sont différentes. Toutefois, il ressort dans les deux cas le besoin de posséder une méthode de planification de chemin performante. Dans le cas des applications interactives, cela va en effet permettre une réaction rapide des agents aux modifications apportées par l'utilisateur ou par un autre facteur dans la topologie. Dans le cas des applications de production, l'utilisation de telles méthodes permettra d'avoir un outil de prévisualisation des résultats efficaces et économisera le temps d'attente des producteurs. En plus de l'aspect temporel et interactif, la définition du problème de planification de chemin va également dépendre de la complexité de l'environnement considéré. La gestion d'environnements statiques ou dynamiques dont l'évolution est connue ou non a priori sont autant de possibilités à considérer. Ces possibilités amènent ainsi chacune des contraintes différentes à la formulation du problème. La complexité du problème général de planification de chemin a été démontrée comme étant PSPACE-complet [Rei79, Can88]. Afin de réduire cette complexité, les méthodes proposées vont donc généralement se focaliser sur la résolution de problèmes spécifiques. Les différentes méthodes de planification proposées vont donc généralement se concentrer sur un contexte particulier, que ce soit la complexité de l'environnement, le nombre d'agents amenés à naviguer simultanément ou encore le besoin d'interactivité et de réactivité dont doit faire preuve l'agent durant sa planification.

2 Problématique

Les méthodes de planification de chemin actuelles s'intéressent généralement à adresser le problème de la navigation pour une topologie particulière d'environnements. De nombreuses méthodes se sont ainsi intéressées à des agents au sein d'environnements totalement statiques (cf. Bibliographie section 2). La prise en compte d'environnements

statiques va permettre de précalculer de nombreux éléments nécessaires à la planification des agents puisque la topologie de l'environnement ne va pas être modifiée par la suite. Afin de s'ouvrir à un panel plus important d'environnements, de nombreuses méthodes de planification se sont ensuite focalisées sur la navigation au sein d'environnements changeants et dynamiques (cf. Bibliographie section 3). Toutefois les éléments dynamiques pris en compte par ces méthodes sont généralement des obstacles que les agents doivent éviter au cours de leur navigation. Plus récemment, certaines méthodes (cf. Bibliographie section 4) ont proposé de gérer des environnements composés d'éléments dynamiques navigables mais elles font en contrepartie l'hypothèse que les mouvements des éléments dynamiques sont connus a priori afin d'anticiper les accès créés. Ces simplifications limitent les possibilités d'applications de ces méthodes. En effet, l'évolution de la topologie d'un environnement n'est généralement pas connue à l'avance puisque cette évolution pourra être la conséquence de l'action d'un utilisateur ou le résultat d'une simulation physique. De plus, les objets dynamiques présents dans un environnement peuvent bien entendu se comporter comme des obstacles à la navigation de l'agent mais ces objets, tels que des plateformes mobiles, des planches ou encore des escabeaux, vont également pouvoir aider un agent au cours de sa navigation en lui proposant de nouvelles opportunités de navigation. En simplifiant le problème de navigation, les solutions actuelles ne permettent donc pas de gérer des environnements de ce type.

Les recherches effectuées pendant cette thèse se sont focalisées sur la planification de chemin pour des agents peuplant un environnement virtuel dynamique. La dynamique au sein des environnements que nous considérons pourra résulter de plusieurs facteurs tels que des scénarios, des forces externes appliquées sur les éléments, comme par exemple la simulation de forces physiques, ou encore, dans le cadre des applications interactives, des conséquences de l'action d'un utilisateur. La prise en compte d'environnements physiques ou des actions de l'utilisateur va ainsi amener à gérer des situations où l'évolution de l'environnement ne peut pas être déterminée à l'avance. Afin de gérer de telles situations, nous nous sommes intéressés à produire une solution de planification au sein d'environnements dynamiques dont l'évolution au cours du temps n'est pas connue a priori. D'autre part, nous voyons les objets présents au sein d'un environnement comme des obstacles mais également comme des éléments qui peuvent être utilisés par des agents pour les aider au cours de la navigation. Une planche peut ainsi être utilisée par un agent comme un pont connectant deux zones, ou une caisse peut être utilisée pour accéder à un endroit surélevé. Nous nous sommes donc intéressés au cours de cette thèse à représenter cette accessibilité pouvant être créée par les objets afin de pouvoir offrir de nouvelles opportunités de navigation aux agents de l'environnement. De plus, à la vue des domaines d'application présentés précédemment, nous avons souhaité conserver tout au long de cette thèse l'idée de proposer une méthode performante pouvant être compatible avec des applications interactives ou des outils de prévisualisation rapide. La gestion d'un environnement dynamique dont l'évolution n'est pas connue a priori, la prise en compte de l'accessibilité créée par les objets, ainsi que la volonté de créer d'une solution compatible avec des applications interactives sont autant de contraintes faisant de cette planification de chemin un problème complexe

qui n'a pas encore été abordé à l'heure actuelle.

3 Contributions

Afin de proposer une solution de planification de chemin pouvant s'adapter aux différentes situations présentées précédemment, nous introduisons une méthode faisant peu d'hypothèses sur les propriétés de l'environnement. Notre méthode a donc pour objectif de planifier des chemins au sein d'environnements dont l'évolution n'est pas connue a priori et où les éléments présents ne sont pas seulement des obstacles à la navigation de l'agent mais peuvent également l'assister dans sa tâche de navigation. De plus, cette planification est effectuée avec des performances permettant l'utilisation dans des applications interactives. Différentes contributions ont permis, au cours de cette thèse, de produire une méthode permettant de répondre à ces différents objectifs. Les contributions apportées sont les suivantes :

Définition d'une représentation duale des objets : Nous proposons, dans un premier temps, une représentation duale des éléments permettant de caractériser les différentes interactions existantes entre un élément et un agent. Cette représentation permet d'identifier l'impact d'un objet sur son environnement que ce soit aussi bien en terme d'obstruction ou d'accessibilité. De plus, cette représentation est construite en prenant en compte les diverses capacités de navigation de l'agent. Cela permet d'associer à la représentation de l'objet, la gestion de différents déplacements et, donc, de l'adaptation de posture de l'agent en fonction des propriétés de l'objet. L'introduction de cette représentation duale nous permet également de faciliter le suivi et la mise à jour de la topologie globale, en associant à chaque objet sa propre représentation locale. De plus, cela permet de gérer de manière simplifiée le suivi de la topologie dans des environnements dynamiques dont l'évolution n'est pas connue a priori.

Construction de la topologie basée sur l'observation : Cette seconde contribution se focalise sur la détection et la représentation de la topologie globale de l'environnement au cours de la simulation. En considérant les représentations duales associées aux éléments, nous sommes capables de localiser et de caractériser les relations topologiques d'accessibilité et d'obstruction existant au sein d'un environnement. L'observation de ces relations topologiques, ainsi que le suivi de leur évolution, nous permet ainsi de définir une représentation globale de la topologie dynamique de l'environnement. De plus, en se basant sur les observations effectuées au fil de la simulation, nous sommes capables, sans ajouter d'information a priori, d'identifier des propriétés temporelles sur les relations topologiques existantes dans l'environnement. Nous pouvons ainsi détecter des liens statiques ou périodiques. Cette représentation permet, en agissant à la façon d'une mémoire de l'évolution de la topologie, d'anticiper de futures évolutions possibles dans l'environnement dynamique.

Planification hiérarchique : Notre dernière contribution réside dans l'introduction d'un nouvel algorithme de planification. Celle-ci va se baser sur la représentation dynamique de la topologie afin de proposer une navigation selon plusieurs critères tels que la minimisation de temps d'attente, une optimisation de la distance ou encore la définition de critères de sécurité au cours de la navigation entre les éléments dynamiques.

De plus, en se basant sur les informations temporelles fournies par la représentation topologique, nous sommes capables d'anticiper des chemins entre des éléments dynamiques déconnectés dans l'espace et dans le temps. Cette planification temporelle permet ainsi également de tirer profit des mouvements propres des éléments dynamiques afin de proposer des nouvelles solutions de navigation aux agents. Enfin, l'aspect temporel et dynamique de l'environnement sera pris en compte dans les différentes parties de la planification, afin de gérer à la fois les accessibilités dynamiques mais également d'éviter les collisions de l'agent avec les obstacles mobiles.

Les possibilités d'applications de notre méthode seront illustrées en présentant la mise en place et l'utilisation de chacune de nos contributions dans des environnements dynamiques. Afin de démontrer les capacités de notre méthode dans un contexte applicatif complexe, nous prendrons en exemple un environnement dynamique qui sera soumis aux lois physiques et aux interactions d'un utilisateur pouvant agir sur le monde en temps réel pendant la simulation. Dans de tels environnements les modifications apportées par le moteur physique ou par l'utilisateur seront donc imprévisibles et les agents virtuels devront donc adapter leur navigation de manière réactive en se servant de tous les éléments pouvant les aider au cours de cette tâche.

4 Organisation de ce mémoire

Ce mémoire de thèse va tout d'abord présenter une étude bibliographique du domaine de la planification de chemin. Cette étude va permettre d'introduire les bases de nos recherches mais également d'identifier les limitations des méthodes actuelles et de démontrer le besoin de nouvelles solutions. Suite à cet état de l'art, nous présenterons nos contributions en trois chapitres. Le premier s'intéressera à la représentation de l'agent et des objets géométriques de l'environnement. Dans cette partie nous introduirons la représentation duale associée aux objets, définissant ainsi les Volumes d'Interaction, qui nous permettront de représenter cet impact topologique d'un élément. Cela constitue une base pour la suite des contributions. Le chapitre suivant traitera de la représentation dynamique de la topologie basée sur l'observation à travers l'identification des relations existant entre les différents Volumes d'Interaction présents dans l'environnement. Nous y introduirons également une nouvelle structure, le Graphe Topologique, représentant à la fois la mémoire de l'environnement mais aussi une caractérisation dynamique de la topologie globale de l'environnement. Dans la partie suivante, nous verrons comment, en nous basant sur les Volumes d'Interaction et sur le Graphe Topologique, nous pouvons utiliser une planification de chemin hiérarchique afin de proposer une navigation efficace à l'aide de différents critères pour l'agent au sein de l'environnement dynamique. Nous présenterons dans le chapitre suivant différents résultats que nous avons obtenus au cours de cette thèse puis nous conclurons dans le dernier chapitre de ce manuscrit.

Bibliographie

1

Table des matières

1 **Introduction à la planification de chemin** **10**
2 **Planification de chemin en environnements statiques** **11**
2.1 Décomposition en cellules 12
2.2 Cartes de cheminement 16
3 **Planification de chemin en environnements statiques avec des éléments dynamiques** **21**
3.1 Adaptation ponctuelle de la représentation de l'environnement pour la planification de chemin 22
3.2 Adaptation dynamique de la représentation de l'environnement pour la planification de chemin 26
3.3 Planification de trajectoire 28
4 **Planification de chemin en environnements complexes dynamiques** . **30**
5 **Planification de chemin et mouvements de l'agent** **33**
6 **Conclusion** **38**

Le problème de la navigation au sein d'environnements est commun à de multiples domaines tels que la robotique ou l'informatique graphique. Que ce soit au sein d'environnements réels, comme c'est le cas en robotique, ou d'environnements virtuels, de plus en plus présents en informatique graphique, la navigation des robots ou des agents virtuels est un critère clé de leur autonomie dans l'environnement. Doter ces agents de la capacité de naviguer de manière autonome au sein d'environnements connus ou non est la problématique principale de la planification de chemin. Celle-ci a pour but de déterminer un chemin évitant les collisions avec les obstacles et permettant de se déplacer entre un point de départ, en général la position courante de l'agent, et un point cible, autrement dit le but de l'agent. De nombreuses méthodes de planification de chemin ont été proposées et permettent de prendre en compte différents types d'environnement. Les méthodes de planification se basent en grande partie sur la représentation faite de l'environnement et les propriétés de celui-ci. Nous verrons donc, après une introduction au problème de la planification de chemin, les solutions qui ont tout d'abord été proposées pour des environnements statiques, puis pour de la planification dans des environnements contenant des objets dynamiques pour en arriver finalement à des environnements totalement dynamiques.

1 Introduction à la planification de chemin

La problématique de la planification de chemin est définie ainsi : étant donné un robot possédant un nombre variable de degrés de liberté, et une description de l'environnement où ce robot est immergé, comment trouver un chemin libre de toutes collisions entre deux configurations du robot au sein de l'environnement. De nombreux exemples illustrant cette problématique peuvent être cités. Le rôle de la planification va être par exemple de déterminer un chemin entre deux positions dans un environnement. Il peut s'agir ainsi d'un agent virtuel cherchant son chemin dans un environnement tel qu'un Personnage Non Joueur (PNJ) dans un jeu vidéo. On peut également s'intéresser au déplacement d'un bras mécanique industriel possédant de nombreux degrés de libertés et étant placé dans un environnement très contraint. La nature du robot (ou de l'agent) considéré ainsi que les propriétés de l'environnement dans lequel il va évoluer vont définir différents types de problèmes. Vu la grande variété de problèmes abordés, la planification de chemin a fait l'objet de nombreuses recherches durant les dernières décennies, de nombreux résultats sont discutés dans les états de l'art proposés par Latombe [Lat91], Choset [CLH+05] et LaValle [LaV06].

La planification de chemin dépend en grande partie de l'environnement dans lequel l'agent va évoluer. La formulation des problèmes de planification de chemin va donc généralement reposer sur la représentation qui est faite de cet environnement, aussi appelée "espace de travail" ou *workspace*. Les différentes méthodes de planification proposées explorent une représentation duale de l'espace de travail afin d'identifier le chemin désiré. Cette représentation duale de l'espace de travail est appelée l'espace des configurations ou *C-Space* [Lp83]. Cet espace des configurations est défini comme un espace à N dimensions dans lequel chacun des N axes de la base représente un degré de liberté du robot. Dans cet espace, une configuration est un vecteur de dimension N représentant une posture entièrement définie de l'agent en fixant la valeur de ces N degrés de liberté. Une courbe dans cette espace des configurations va donc représenter une séquence de postures de l'agent et donc un mouvement de celui-ci. Afin d'identifier un chemin pour l'agent dans l'environnement, les algorithmes de planification de chemin vont donc chercher une séquence de configurations réalisables dans l'espace des configurations. Cet espace des configurations est généralement divisé en deux sous-ensembles : C-Free et C-Obstacle. C-Free représente l'ensemble des configurations admissibles, c'est-à-dire l'ensemble des configurations pour lesquelles l'agent n'entre pas en collision avec lui-même ou avec l'environnement. À l'opposé, C-Obstacle représente l'ensemble des configurations invalides, c'est-à-dire où l'agent est en collision avec lui-même ou son environnement. Planifier un chemin pour un agent au sein de son environnement revient donc à trouver une séquence valide de configurations dans le C-Space entre la configuration initiale de l'agent et une configuration cible. Afin d'être valide, cette trajectoire doit être composée d'une séquence de configurations appartenant à C-Free. Les méthodes de planification de chemin s'intéressent donc en grande partie à proposer une représentation de C-Free adaptée en particulier à un type de pro-

blèmes. La recherche de chemin lors de la planification va également être guidée par des critères tendant à optimiser le chemin généré. Certaines méthodes vont se focaliser sur la génération de chemin les plus courts possibles tandis que d'autres vont se concentrer sur des critères de sécurité de l'agent en cherchant par exemple à éloigner au maximum le chemin des obstacles.

Le problème de la planification de chemin a été démontré comme étant PSPACE-complet [Rei79, Can88]. Cette complexité implique donc de devoir faire des approximations ou de sacrifier des critères d'optimalité afin de simplifier le problème. Ces approximations peuvent porter sur la qualité des planifications effectuées ou par exemple sur l'occupation mémoire dédiée à l'algorithme ou au temps de calcul qui lui est alloué. Il est également possible de simplifier le problème en diminuant la dimension de l'espace de recherche (i.e. l'espace des configurations) et donc par conséquent en réduisant le nombre de degrés de liberté de l'agent considéré. Enfin, la complexité de la tâche de navigation va également être dépendante de la complexité de l'environnement considéré. Nous allons nous intéresser au cours de cette étude bibliographique à trois familles distinctes de problèmes de planification et aux solutions qui ont été proposées afin d'y répondre. Nous verrons ainsi, dans un premier temps, les méthodes proposées pour la planification à l'intérieur d'environnement statique. Nous aborderons ensuite le cas des environnements majoritairement statiques mais où il est possible d'avoir des obstacles mobiles, se déplaçant dans le temps. Nous finirons par des environnements entièrement dynamiques au cours de la simulation, considérant les objets mobiles à la fois comme des obstacles, mais également comme des aides à la navigation. Enfin, la planification de chemin et le mouvement de l'agent le long de ce chemin étant corrélés, nous verrons dans une dernière partie différentes méthodes qui ont été proposées afin de lier le chemin généré au mouvement de l'agent et également comment il est possible d'adapter les postures de celui-ci pour planifier des chemins tenant compte de contraintes environnementales variées.

2 Planification de chemin en environnements statiques

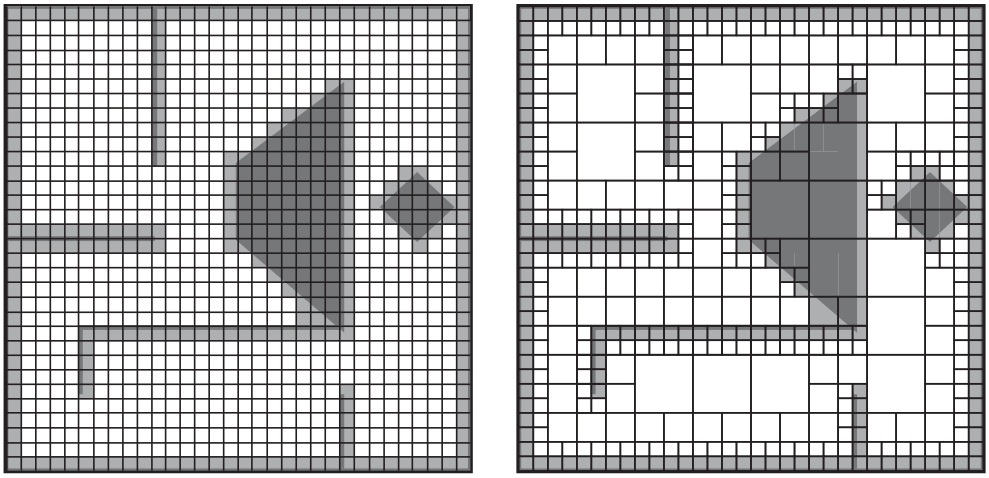
Les environnements statiques se caractérisent par leur structure qui n'est pas amenée à évoluer au cours du temps. Cette propriété implique donc qu'il n'y a pas de nouvelles accessibilités ou obstructions créées pendant le déroulement de la simulation. Puisque la configuration de ces environnements ne change pas, il est donc possible d'effectuer des précalculs afin de proposer des structures de données performantes lors de requêtes de planification de chemin ultérieures. Les méthodes proposées pour caractériser les environnements navigables vont se diviser principalement en deux familles. D'un côté, les méthodes de décomposition en cellules vont représenter l'espace navigable à l'aide de zones interconnectées de formes prédéfinies permettant d'appréhender les limites de C-Free et de C-Obstacles. D'un autre côté, les méthodes utilisant les cartes de cheminement vont représenter un sous-ensemble de C-Free par un ensemble de chemins standardisés permettant la navigation de l'agent.

2.1 Décomposition en cellules

La décomposition en cellules propose de diviser l'espace des configurations en cellules de formes prédéfinies. Celles-ci sont ensuite caractérisées selon leur appartenance à C-Free ou C-Obstacle afin de construire une structure de données plus adaptée à la planification de chemin. Les méthodes de décomposition en cellules se répartissent en deux familles : les décompositions approchées et les décompositions exactes. Les décompositions approchées construisent une représentation approchée de l'environnement en ne cherchant pas à caractériser précisément C-Free mais en en définissant un sous-ensemble dont la représentation est plus simple. Les décompositions exactes sont des structures généralement plus coûteuses à calculer mais qui permettent une description plus fine de l'environnement en décrivant exactement C-Free et, par conséquent, C-Obstacle.

2.1.1 Décomposition approchée en cellules

Les méthodes de décompositions approchées proposent l'utilisation d'une forme prédéfinie de cellules afin de caractériser l'environnement [Lat91, Kuf98, Kuf04, SYN01, LC03, Li04]. Ces cellules représentent des sous-ensembles de l'espace des configurations et sont interconnectées afin de capturer la connectivité de l'environnement. Les cellules obstruées par une partie de C-Obstacles sont ensuite annotées et interdites à la navigation lors de la planification de chemin. Lors de l'utilisation de décompositions approchées, le rôle de la planification va être d'identifier une séquence de cellules interconnectées et appartenant à C-Free permettant de relier la position courante de l'agent à son but. Ces décompositions en cellules sont généralement effectuées à l'aide de grilles régulières 2D composées de cellules carrées [Kuf98]. Les obstacles de l'environnement sont ensuite projetés dans cette grille permettant de marquer les cellules obstruées (voir Fig. 1.1a). La représentation de l'environnement sous la forme d'une grille régulière 2D permet de projeter les obstacles de l'environnement dans une bitmap 2D et de pouvoir ainsi profiter de la carte graphique afin d'accélérer la planification de chemin lors des requêtes. Au cours de cette planification, chaque cellule dans C-Free permet l'accès à ses 8 cellules avoisinantes si celles-ci appartiennent également à C-Free (4 cellules directement voisines, 4 cellules dans les diagonales). Cette exploration de la grille régulière peut également être améliorée en utilisant des heuristiques afin d'empêcher les changements de direction trop brusques de l'agent ce qui permet également de réduire l'espace de recherche [Kuf04]. Afin de gérer des environnements plus complexes, certaines techniques proposent l'utilisation de plusieurs grilles régulières dans un même environnement afin de pouvoir représenter simultanément plusieurs capacités de déplacement (voir Section 5) [SYN01]. Ceci permet également de supprimer la contrainte 2D amenée par l'utilisation d'une unique grille régulière et de gérer en conséquence des environnements 3D avec différents étages [LC03, Li04]. En se basant sur les caractéristiques de l'agent, cette approche permet, en plus, de localiser dans la grille régulière les accès existants entre des zones déconnectées de l'environnement, gérant ainsi la planification de chemin au sein d'environnements non connexes.



(a) Grille régulière

(b) Grille régulière couplée à un quad-tree

Figure 1.1 – 1.1a : Grille régulière composée de cellules carrées. 1.1b : Grille régulière couplée à un quad-tree afin de diminuer le nombre d'éléments pour représenter l'environnement. Les cellules grisées correspondent aux cellules annotées comme obstruées, les cellules blanches comme les cellules libres.

Les solutions proposées pour la modélisation de *C-Space* sous forme de grilles régulières requièrent de trouver un compromis entre la précision de la représentation utilisée, ou en d'autres termes la résolution de la grille, le coût de mémoire et le temps de planification. En effet, en utilisant une résolution plus fine permettant de décrire l'environnement précisément, l'information stockée en mémoire sera plus importante et le nombre d'éléments à considérer durant la planification sera également plus grand, abaissant dès lors les performances. D'un autre côté, si la résolution de la représentation n'est pas assez précise, il est possible de passer à côté de solutions existantes. D'autre part, les chemins générés ne seront pas de bonne qualité et la représentation de l'environnement sera assez éloignée de sa configuration réelle, générant ainsi des chemins peu réalistes. Plusieurs solutions alternatives ont été proposées afin de construire une décomposition approchée permettant de mieux gérer l'occupation mémoire. En se basant sur la structure des grilles régulières, il est, par exemple, proposé de définir une représentation hiérarchique permettant de représenter l'environnement avec plusieurs niveaux de détails. L'utilisation d'une structure hiérarchique permet de réduire le nombre de cellules dans la grille en conservant de l'information détaillée là où il y en a besoin. Cette structuration hiérarchique est faite à l'aide d'un *quadtree* (ou d'un autre type de 2^n-tree) structurant les données des cellules. Ce *quadtree* est une structure de donnée de type arbre dont les nœuds peuvent être soit des nœuds terminaux, soit divisés en 4 nœuds fils. L'espace des configurations est représenté dans un premier temps avec une résolution assez basse : chaque cellule est représentée par un nœud du *quadtree*. Les cellules peuvent se trouver dans l'un des trois cas suivant : (1) la cellule est entièrement incluse dans C-Free, (2) la cellule est entière incluse dans C-Obstacle,

(3) la cellule est à la frontière de C-Free et C-Obstacle et contient un sous ensemble de chaque. Dans les cas (1) et (2), la cellule contient toute l'information requise et n'a pas besoin d'être plus détaillée au niveau de la représentation. Dans le cas (3), il faut raffiner la représentation afin de caractériser plus précisément C-Free et C-Obstacle. Dans ce cas de figure, la cellule est divisée en 4 cellules filles de résolution équivalente et la même classification est faite entre les nouvelles cellules créées (voir Fig. 1.1b). L'utilisation d'un quadtree permet ainsi de réduire considérablement le nombre de cellules de la grille, et donc l'utilisation mémoire. Il est ainsi plus efficace de représenter de larges environnements ouverts en focalisant le raffinement sur les frontières entre C-Free et C-Obstacle.

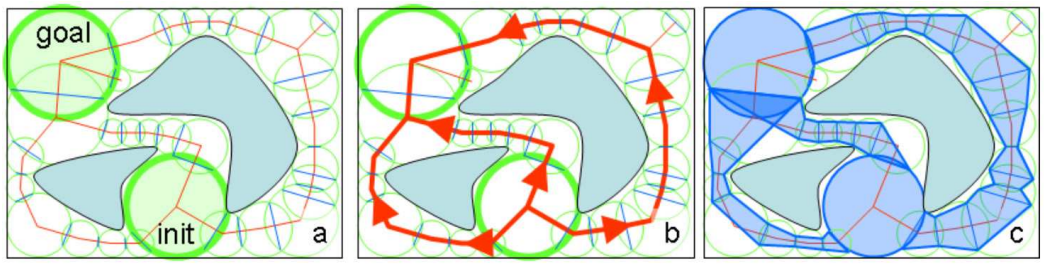


Figure 1.2 – Pettré [PLT05] propose une décomposition de C-Free à l'aide de cylindre (a). Cette décomposition permet l'identification des principaux chemins de navigation (b) et également la création de couloirs de navigation permettant de générer des trajectoires plus variées pour gérer la navigation d'une foule d'agent.

Diverses approches n'utilisant pas de structures à base de grilles régulières ont également été proposées. Il a ainsi été, par exemple, proposé de représenter l'environnement navigable à l'aide d'une décomposition en cylindres de navigation [PLT05]. L'utilisation de ces cylindres lui permet de créer une structure comprenant un nombre assez restreint d'éléments et permettant la navigation de plusieurs milliers d'agents en temps réel. Les axes médians de C-Free, c'est à dire les axes les plus éloignés des obstacles et présentant donc la plus grande sécurité au niveau de la navigation, sont tout d'abord identifiés. Des cylindres, centrés sur ces axes et dont le rayon correspond à la distance de l'obstacle le plus proche, sont définis le long de ces axes. Ces cylindres représentent donc des sous-espaces de C-Free où la navigation est possible. En connectant ensuite les cylindres qui s'intersectent, des couloirs de navigation où les entités peuvent naviguer librement sont créés (voir Fig.1.2). En identifiant les axes médians et l'éloignement maximum aux obstacles, cette méthode construit une représentation relativement proche d'un diagramme de Voronoï généralisé.

2.1.2 Décomposition exactes en cellules

À l'inverse des décompositions approchées, les décompositions exactes permettent de capturer entièrement la représentation de C-Free. Parmi les techniques utilisées on retrouve, par exemple, des décompositions en cellules verticales [LaV06] ou des triangulations de Delaunay. Ces méthodes cherchent à caractériser précisément C-Free en

utilisant les bordures des obstacles comme des contraintes formant les données d'entrées. Les méthodes basées sur les triangulations de Delaunay contraintes utilisent ainsi les frontières des obstacles de l'environnement afin de définir la triangulation [KBT03]. C-Free est donc représenté par un ensemble de triangles dont les arêtes libres représentent des connexions entre zones navigables et les arêtes contraintes des obstacles. La planification de chemin se ramène alors à identifier une séquence de triangles reliés par des bordures franchissables. En analysant de manière plus poussée la triangulation obtenue, certaines méthodes proposent d'extraire des informations supplémentaires sur l'environnement. L'analyse d'une triangulation de Delaunay contrainte peut par exemple permettre d'extraire les goulets d'étranglement présents dans un environnement [LD04]. L'extraction de ces goulets a pour objectif de détecter précisément les endroits les plus contraints de l'environnement permettant d'utiliser cette information lors de la navigation de l'agent. D'un autre côté, un raffinement de la triangulation de Delaunay contrainte peut être utilisé afin de pouvoir planifier les chemins les plus courts pour des agents de tailles différentes en identifiant également les passages contraints de l'environnement ainsi que des couloirs de navigation [Kal10].

Bien que la majorité des représentations exactes permettent une représentation d'un environnement 2D, certaines approches se sont intéressées à gérer des environnements 3D à l'aide de décompositions exactes [Lam09]. Une approche utilisant une subdivision exacte en 3D incluant les contraintes de sol et de plafond permet par exemple d'ajouter des informations de hauteur à l'analyse de l'environnement. Cette subdivision, dite prismatique, permet ensuite de générer des cartes 2D interconnectées et d'augmenter automatiquement les cartes 2D d'information tridimensionnelle permettant entre autre de différencier les obstacles franchissables, tels que les marches, des obstacles infranchissables, tels que les murs (voir Fig.1.3).

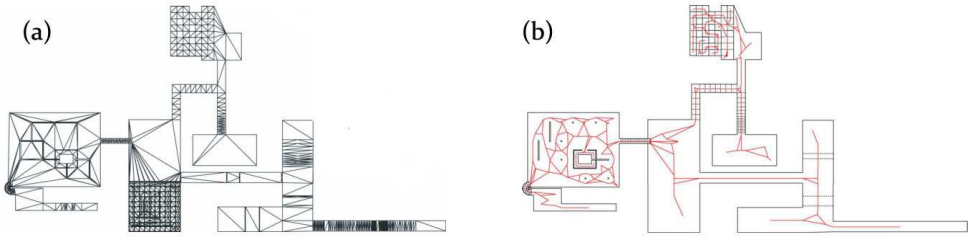


Figure 1.3 – Méthode de décomposition exacte proposée par Lamarche [Lam09]. (a) : Représentation de l'environnement à l'aide de cellules 2.5D prenant en compte les contraintes de hauteur. (b) : Carte topologique de l'environnement prenant en compte les capacités de déplacement de l'agent. On peut notamment l'observer dans le carré supérieur où les déplacements de l'agent entre les différents carrés sont limités par les hauteurs de marche entre ces carrés.

Les méthodes de décompositions en cellules, exactes ou approchées, s'adressent principalement à des problèmes de planification de chemin de faibles dimensions. En effet, lorsque la dimension des espaces de recherche augmente, il est plus difficile de caractériser et de représenter les frontières entre C-Free et C-Obstacle. Dans le cas des méthodes

exactes, les représentions de l'environnement deviennent plus difficiles et surtout beaucoup plus coûteuses en mémoire au-delà d'un espace de recherche en 2 dimensions. Pour les représentations approchées, elles deviennent difficilement utilisables au-delà de 4-5 dimensions. Les cartes de cheminement, que nous allons maintenant aborder, ont donc été conçues pour la planification de chemin dans des espaces de plus grandes dimensions.

2.2 Cartes de cheminement

Les cartes de cheminement (ou *roadmap*) permettent de représenter un sous-ensemble de C-Free en déterminant des configurations libres de l'environnement et en reliant ces configurations par des arcs de transition étant également contenus dans C-Free. En procédant ainsi les cartes de cheminement permettent une exploration beaucoup plus rapide de l'espace des configurations. Les différentes méthodes utilisant des cartes de cheminement vont donc s'intéresser principalement à déterminer des configurations libres de C-Free et à les connecter. Cela est fait en essayant de proposer la meilleure couverture possible sur l'ensemble de l'environnement afin de le représenter au mieux. Les techniques utilisant les cartes de cheminement se décomposent en deux familles : les méthodes à requêtes multiples (*multiple-query*) et les méthodes à requêtes simples (*simple-query*). Dans le premier cas, une exploration de l'ensemble de l'environnement est effectuée a priori et ré-utilisée ensuite à de multiples reprises pour la planification de plusieurs chemins. Dans le second cas, l'environnement est exploré seulement partiellement au moment de la demande de planification, chaque nouvelle requête entraînant une nouvelle exploration partielle de C-Free.

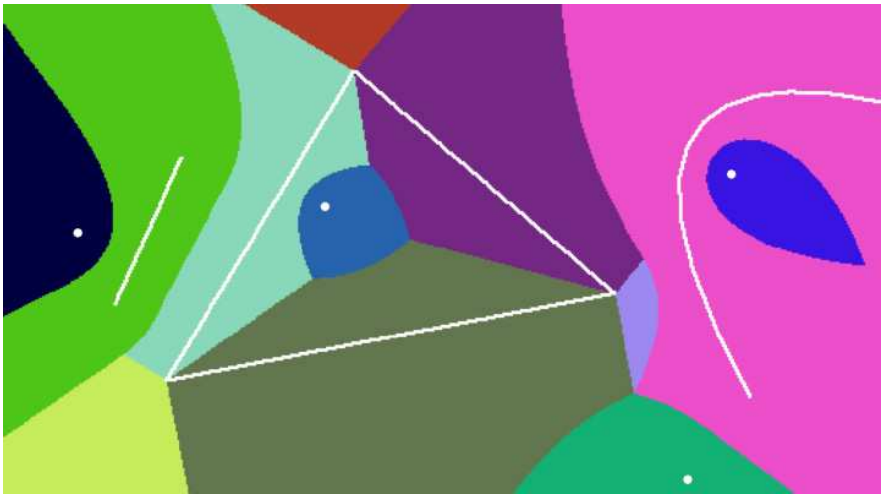
2.2.1 Méthodes à requêtes multiples

Les méthodes à requêtes multiples sont généralement composées de deux phases. Dans un premier temps, une phase de construction permet d'explorer l'espace des configurations et de capturer la topologie de l'environnement dans la carte de cheminement. Dans un second temps, la phase de requête permet d'explorer la carte de cheminement afin de calculer un chemin pour l'agent. La carte de cheminement représentant l'environnement est donc calculée au cours d'une phase de précalcul et utilisée au cours de la simulation pour répondre aux différentes tâches de planification de l'agent. Lors de la planification, la position courante de l'agent ainsi que sa destination sont rattachées aux cartes de cheminement. La construction des cartes de cheminement est en général basée soit sur des diagrammes de Voronoï généralisés soit sur des méthodes probabilistes. Il existe toutefois d'autres méthodes permettant de représenter l'environnement à l'aide de carte de cheminements, en les couplant par exemple avec des quad-tree [WL08].

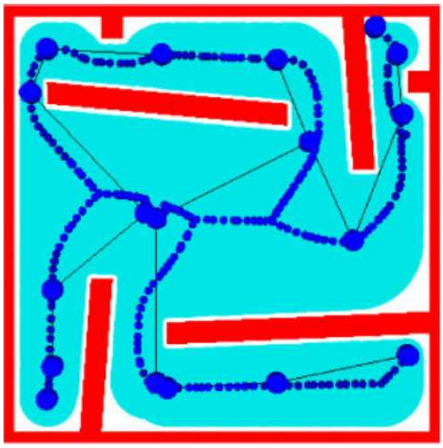
Les diagrammes de Voronoï généralisés permettent de calculer les axes d'éloignement maximum des obstacles. En renseignant les frontières de C-Obstacle, le calcul d'un diagramme de Voronoï généralisé va en effet déterminer les ensembles de points les plus éloignés des obstacles en identifiant les bordures des cellules de Voronoï. L'utilisation de ces points pour la création d'une carte de cheminement permet donc de générer des

chemins présentant une bonne marge de sécurité vis-à-vis des obstacles et de limiter les chemins rasant les bordures de C-Free. Afin d'accélérer le calcul et le rendu de telles cartes de cheminement, certaines méthodes [HKL+99] utilisent des algorithmes de rendus issus de l'informatique graphique afin de représenter le diagramme de Voronoï obtenu dans un Z-buffer (voir Fig. 1.4a) grâce à une parallélisation des calculs sur la carte graphique. La méthode des "cartes de couloirs", ou *Corridor Maps*, utilise aussi les diagrammes de Voronoï dans la construction de leur carte de cheminement [GO07] (voir Fig. 1.4b). En effet, les nœuds de la carte de cheminement sont des configurations qui sont éloignées au maximum des obstacles entourants. Ces configurations sont de plus annotées de leur distance d'éloignement aux obstacles ce qui permet ainsi de leur attribuer un rayon de sécurité permettant de définir un cercle autour de cette configuration. L'ensemble des configurations et des chemins qui sont contenus au sein de ce cercle sont donc par définition libres de toute collision avec l'environnement. En joignant l'ensemble des configurations de la carte de cheminement et en faisant l'union de leurs cercles de sécurité respectifs, on obtient donc des couloirs de navigation au sein desquels l'agent va pouvoir naviguer librement. Les chemins générés entre les configurations de la carte sont ensuite lissés lors de la navigation afin d'obtenir des trajectoires plus naturelles mais restant toujours à l'intérieur des couloirs de navigation (voir Fig. 1.4c).

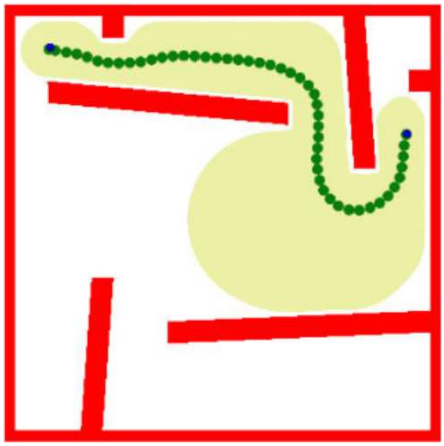
Parmi les méthodes probabilistes à requêtes multiples, l'une des plus utilisée est sans doute celle des cartes de cheminement probabilistes, ou *Probabilistic RoadMaps* (PRM). Les PRMs ont été introduites simultanément par Kavraki [KL94] et Overmars [OS94] avant de faire l'objet d'une étude commune de leur part [KSLO96]. Pendant la phase de construction de la carte de cheminement, l'exploration de l'espace des configurations est faite de manière aléatoire en tirant des configurations au hasard. Cet échantillonnage tend à suivre généralement une distribution uniforme des échantillons mais de nombreuses méthodes proposent l'utilisation d'autres lois de répartitions en fonction de l'environnement étudié. Un algorithme de détection de collision est utilisé pendant l'échantillonnage afin de déterminer si les configurations sont valides ou si elles sont en collision avec C-Obstacle. Une fois l'échantillonnage terminé, un planificateur local est utilisé afin de connecter les configurations valides à leurs voisins. Si ceux-ci répondent à des critères de proximité tels que les k plus proches voisins, ou que les configurations sont à l'intérieur d'une sphère de proximité, alors ils sont connectés. Si le chemin qui permet de les relier reste dans C-Free alors il est conservé dans la carte de cheminement(voir Fig.1.5a). Une fois la carte de cheminement créée, la planification de chemin se fait simplement par l'insertion des configurations de départ et d'arrivée dans la carte de cheminement et par l'utilisation d'un algorithme tel qu'un Dijkstra ou un A* afin de trouver le chemin. Certaines méthodes proposent d'ajouter à la PRM de l'information sur les agents et de biaiser ainsi les chemins générés. La prise en compte de ces meta-information permet l'utilisation de différents algorithmes d'exploration des PRMs afin de gérer des groupes d'entités avec des priorités d'explorations différentes [BA02]. En fonction des comportements, les agents d'un groupe vont par exemple chercher à couvrir tout l'environnement, à atteindre communément un but dont la position est inconnue, ou encore à attendre qu'un " berger " guide leur groupe.



(a) Représentation utilisant des cellules de Voronoï [HKL+99]

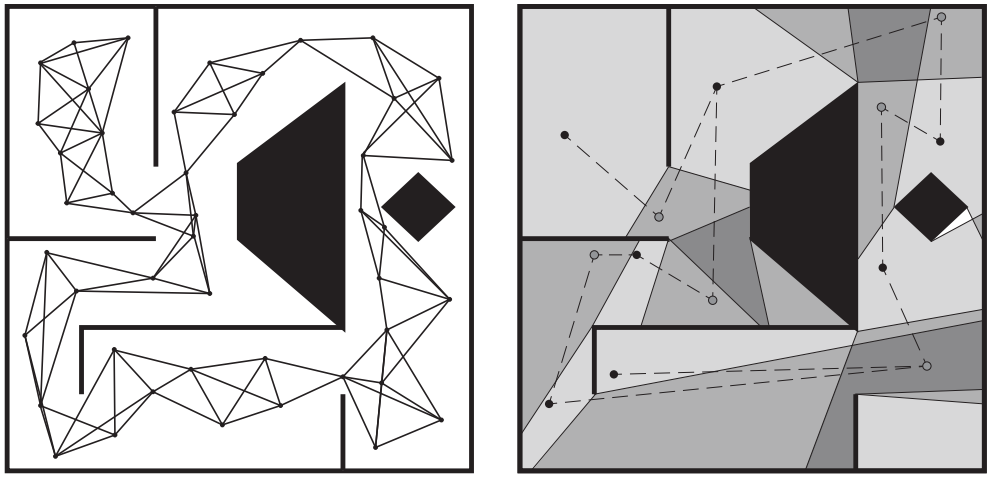


(b) Corridor Maps [GO07]



(c) Planification utilisant les Corridor Maps

Figure 1.4 – 1.4a : En utilisant les bordures des obstacles (en blanc) comme contraintes dans l'environnement, HoffIII propose une représentation de l'environnement dans des bitmaps en utilisant des cellules de Voronoï. 1.4b et 1.4c : Geraerts utilise les points les plus éloignés des obstacles afin de construire une carte de cheminement (bleu foncé) et associe également à la carte des couloirs de sécurité (bleu clair). Un chemin (vert) est ensuite identifié puis lissé tout en restant éloigné des obstacles.



(a) Probabilistic RoadMap (PRM) **(b)** Visibility PRM

Figure 1.5 – 1.5a : Le premier environnement montre une carte de cheminement probabiliste où chaque nœud est connecté à ses k plus proches voisins ($k = 4$). 1.5b : Visibility PRM. Les nœuds noirs sont des gardiens et les cercles gris des connecteurs. Les zones gris clair sont observées par un gardien, les gris moyen par deux gardiens et les plus sombres par 3 gardiens. La zone blanche n'est observée par aucun gardien mais la couverture de l'environnement est considérée comme suffisante.

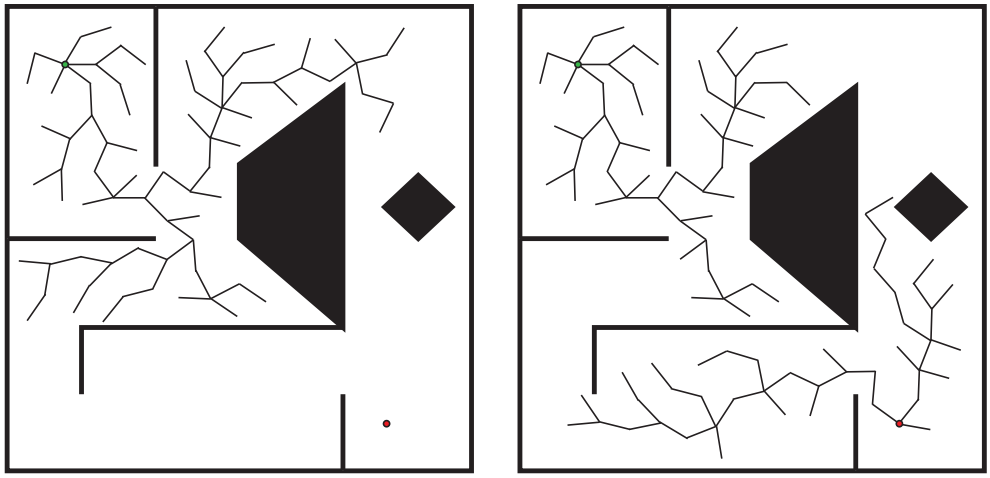
Selon les environnements considérés et le recouvrement désiré de la carte de cheminement sur l'environnement, les PRMs construites peuvent être très coûteuses en mémoire au regard du nombre de nœuds et d'arcs nécessaires. Afin de réduire le nombre de nœuds, et donc l'occupation mémoire, tout en gardant une couverture maximale de l'environnement, plusieurs études se sont intéressées à la création de *visibility-PRM* [SLN00, SGLM03]. Dans ces *visibility-PRMs*, la visibilité dénote la possibilité de pouvoir trouver un chemin direct, c'est à dire sans passage par un nœud intermédiaire de la carte de cheminement, entre deux configurations de l'environnement sans entrer en collision avec C-Obstacle. Ces cartes de cheminement sont constituées de deux types de nœuds : les gardiens et les connecteurs. Les gardiens sont des nœuds qui ne sont pas visibles les uns des autres. Chaque gardien se voit rattaché à une zone de visibilité qui contient l'ensemble des configurations de C-Free auxquelles il peut accéder directement. Un gardien ne devant être visible par aucun autre, le nombre de gardiens nécessaires pour couvrir l'environnement est donc restreint. Les connecteurs sont ensuite des configurations qui sont visibles par deux gardiens ou plus et ils permettent donc de connecter ses gardiens en créant des chemins entre eux. Un connecteur est ajouté seulement lorsqu'il crée une nouvelle connexion entre deux gardiens ce qui permet de limiter également le nombre de connecteurs utilisés (voir Fig. 1.5b).

2.2.2 Méthodes à requêtes simples

Contrairement aux méthodes à requêtes multiples qui cherchent à construire une représentation de l'environnement dans son ensemble, les méthodes à requêtes simples

proposent d'explorer et de construire la représentation d'un sous-ensemble de l'environnement lorsqu'une requête de planification est formulée. Ces méthodes favorisent donc la création d'une structure par requête plutôt que la construction d'une structure globale plus lourde (notamment en mémoire) mais utilisée pour l'ensemble des requêtes de planification. L'une des structures les plus utilisée s'appelle les *Rapidly-exploring Random Trees* (RRT) dont le but est d'explorer l'espace de configuration à l'aide d'un arbre de recherche [LaV98]. La racine de l'arbre d'exploration est fixée au niveau de la configuration courante de l'entité et des échantillons sont ensuite tirés aléatoirement dans l'environnement, entrainant la croissance de l'arbre d'exploration (voir Fig. 1.6a). En limitant l'expansion maximale de l'arbre à chaque pas d'exploration, la croissance de l'arbre est biaisée et tend à explorer en priorité les zones de l'environnement qui n'ont pas été explorées jusqu'alors, ce qui permet de générer rapidement un arbre possédant une bonne couverture de l'espace de recherche. L'exploration de l'environnement s'arrête lorsque la destination de l'agent est atteinte par l'arbre. Contrairement aux PRMs, cet arbre de par sa construction permet de conserver une structure d'exploration qui est toujours connexe et toujours reliée à la configuration courante de l'entité. Afin d'améliorer les performances des RRTs et de connecter plus rapidement la configuration de départ à la configuration cible de l'agent lors de la planification de chemin, une amélioration de l'algorithme de base propose l'utilisation simultanée de deux arbres de recherche pour explorer l'environnement dans une méthode appelée RRT-connect [KL00]. La racine du premier arbre est fixée comme précédemment à la configuration courante de l'entité, tandis que la racine du second est rattachée à la configuration cible. Les deux arbres sont étendus l'un après l'autre et, après chaque phase d'extension, l'algorithme essaie de voir s'il est possible de trouver une connexion directe des deux arbres afin d'accélérer la recherche (voir Fig. 1.6b). Bien que les méthodes basées sur les RRT produisent des résultats rapides, la solution générée n'est pas optimale. Afin de palier à cela, certaines méthodes proposent en se basant sur les RRTs de converger vers des solutions optimales [KWP+11] ou encore d'utiliser un paramètre faisant un compromis lors de la planification entre continuer l'exploration de l'environnement ou raffiner la représentation existante en ajoutant de nouveaux arcs de meilleurs qualité à la représentation courante [APD11]. L'utilisation de ces arbres de recherche permet tout d'abord de trouver des chemins dans des environnements très contraints sans pré-calculs, ni paramètres à ajuster par l'utilisateur. De plus ces algorithmes permettent l'exploration, et donc la planification de chemin, dans des environnements de très grandes dimensions et sont donc très adaptés à des problèmes de planification avec de nombreux degrés de liberté.

Les solutions abordées jusqu'ici ont été proposées pour des environnements statiques. Puisque l'environnement n'est pas modifié lors de la simulation de nombreux pré-calculs peuvent être effectués. Cela simplifie a posteriori la tâche de planification. Toutefois, les environnements où sont amenés à naviguer des robots ou des agents virtuels, sont rarement entièrement statiques. Il convient donc maintenant de s'intéresser, et de s'interroger, sur la manière de prendre en compte des modifications pouvant arriver au cours de la simulation. En considérant avoir des objets qui peuvent bouger, ou être déplacés dans l'environnement, comment est-il possible de planifier un chemin ? Comment



(a) Rapidly-exploring Random Trees (RRT)

(b) RRT Connect

Figure 1.6 – 1.6a : Exploration de l'environnement à l'aide d'un Rapidly-exploring Random Tree. La position courante de l'agent est le point en haut à gauche de l'environnement et la cible à atteindre, le point en bas à droite. 1.6b : Utilisation d'un RRT-Connect. Un arbre de recherche est attaché à la position courante de l'agent tandis qu'un second est rattaché à la cible à atteindre. Les deux arbres de recherches explorent l'environnement de manière alternative.

prendre en compte ces changements dans l'espace des configurations afin de mettre à jour les structures de données ? Et, enfin, comment est-il possible de remettre à jour un chemin planifié si une obstruction qui n'était pas présente précédemment est subitement détectée ? Les méthodes qui ont été introduites dans cette section ne présentent pas de solutions permettant de gérer ces différents aspects. Toutefois, la majeur partie des environnements considérés vont être composés d'une large composante statique. En partant de cette observation il va donc être possible de s'appuyer sur des méthodes de planification prévues pour des environnements statiques puis de les adapter afin de pouvoir gérer des environnements présentant des aspects dynamiques. Nous nous intéresserons dans la section suivante à présenter les algorithmes proposés pour gérer la planification de chemin au sein de tels environnements.

3 Planification de chemin en environnements statiques avec des éléments dynamiques

Afin de pouvoir gérer une plus grande variété de problèmes, certaines méthodes se sont adressées à la représentation d'environnement pouvant être peuplés par des éléments dynamiques. Ces éléments dynamiques vont augmenter la complexité du problème de planification de chemin en lui ajoutant un aspect temporel qui n'était pas présent précédemment. En effet, en considérant l'environnement à des temps différents il est possible que celui-ci ait évolué du fait de l'ajout, la suppression ou le déplacement

de certains de ses éléments. L'espace des configurations qui lui est associé doit donc être mis à jour en conséquence afin de maintenir une représentation de l'environnement permettant la planification de chemin. Alors que les pré-calculs étaient possibles pour appréhender la totalité des environnements statiques vus précédemment, ils sont plus limités dans le cas présent puisque la configuration de l'environnement va évoluer au cours de la simulation. Toutefois, les environnements représentés sont généralement composés d'une grande partie statique dans laquelle sont insérés des éléments dynamiques. Différentes approches ont donc été proposées pour permettre la planification de chemins au sein de tels environnements. Certaines proposent de calculer des chemins dans une représentation de l'environnement et de replanifier ensuite ces chemins lorsqu'ils sont invalidés par la présence d'objets dynamiques. D'autres approches proposent de planifier une trajectoire qui intègre directement au cours de la planification l'aspect temporel de l'environnement. Au cours de cette thèse, une trajectoire sert à désigner un chemin dans l'espace spatio-temporel. Une trajectoire, contrairement à un chemin, va donc chercher à anticiper les déplacements des objets dynamiques dans le domaine spatio-temporel pour proposer une solution de planification prenant l'aspect temporel en considération. Nous verrons ainsi dans un premier temps, les méthodes proposant de mettre à jour la représentation de l'environnement seulement au moment de la planification. Un chemin est ensuite planifié dans cette structure mise à jour et une nouvelle planification permet ensuite la mise à jour de ce chemin si une collision avec un élément de l'environnement est détectée lors de la navigation de l'agent. Dans un second temps, d'autres approches proposent de maintenir la représentation de l'espace des configurations à jour lorsque les changements sont détectés permettant ainsi d'avoir directement la structure adéquate lorsqu'une requête de planification de chemin est effectuée. Enfin, une troisième famille de méthodes propose de planifier directement dans l'espace spatio-temporel. Alors que les méthodes précédentes se focalisent sur le calcul d'un chemin dans l'environnement à un temps donné, celles-ci cherchent à planifier directement une trajectoire dans le domaine espace-temps en anticipant du mieux possible les déplacements des éléments dynamiques.

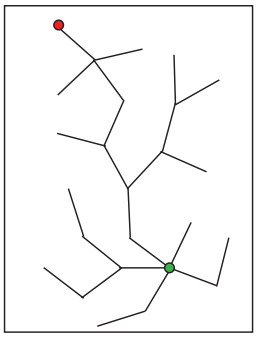
3.1 Adaptation ponctuelle de la représentation de l'environnement pour la planification de chemin

Les Rapidly-exploring Random Trees (RRTs), initialement utilisés dans le cadre des environnements statiques, présentent l'avantage de ne pas nécessiter de pré-calculs de l'environnement. Cette propriété les rend très attrayants pour les environnements présentant des aspects dynamiques dont les pré-calculs sont compliqués par une topologie constamment changeante et généralement non connue à l'avance. De nombreuses extensions des RRTs ont ainsi été proposées afin de gérer la navigation d'agents dans des environnements comportant des éléments dynamiques.

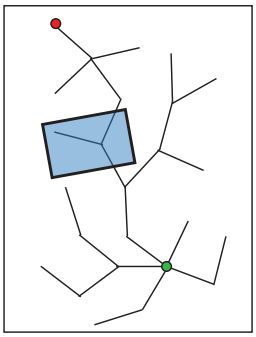
Les RRTs permettent d'explorer une partie de l'environnement lorsque des requêtes de planification de chemin sont formulées. Au sein d'environnements dynamiques, l'agent va être amené à devoir replanifier son chemin lorsque des obstacles sont détectés le long du chemin originellement défini. Une solution simple est d'utiliser un

RRT afin de planifier de nouveau le déplacement de l'agent. Toutefois, dans ce cas, l'information apprise lors de l'exploration de l'environnement par le RRT ayant permis de calculer le premier chemin n'est pas conservée et donc pas réutilisée lors de la replanification. Ainsi, lorsqu'un chemin calculé par un RRT est altéré par la présence d'un obstacle, plusieurs méthodes partent de l'hypothèse qu'un chemin alternatif peut être retrouvé dans un voisinage proche du chemin original. En partant de cette observation, ces méthodes proposent donc de réutiliser le RRT calculé précédemment afin d'orienter la nouvelle planification de chemin en biaisant le nouvel échantillonnage des configurations pour la construction du nouvel arbre de recherche. L'utilisation des informations contenues dans l'arbre de recherche va ainsi permettre d'optimiser la recherche en utilisant le RRT précédent comme un heuristique lors de la recherche. Parmi ces méthodes on peut notamment distinguer les Execution Extented RRT (ERRT) [BV02] ainsi que les Dynamic RRT (DRRT) [FKS06]. Lors de la génération d'un trajet valide, les ERRT proposent de sauvegarder les configurations appartenant au chemin. Lorsqu'une mise à jour de la trajectoire est nécessaire, les points appartenant au chemin original ou le but visé par l'agent ont alors une certaine probabilité d'être utilisés comme échantillons lors de la construction du nouveau RRT (voir Fig. 1.7e). En biaisant ainsi l'échantillonnage des configurations, les ERRT permettent d'obtenir rapidement une nouvelle solution restant proche de la planification précédente. Les DRRT se basent sur le principe de la continuité spatiale et temporelle et proposent de vérifier la validité des branches présentes dans le RRT précédent (voir Fig. 1.7c). Les branches valides et connectées à la racine sont donc conservées. Cette méthode, bien qu'ayant un léger surcoût à l'initialisation du nouveau RRT, permet également de trouver une solution généralement assez proche de la solution d'origine. De plus, en plaçant la racine des DRRT au niveau du but et non de la position courante de l'agent, cette méthode permet également d'éviter de recalculer entièrement l'arbre de recherche lorsque l'agent s'est déplacé. Enfin, les Multipartite-RRTs utilisent un principe similaire afin de guider la nouvelle planification. Dans un premier temps, le chemin de l'agent est planifié à l'aide de RRTs. Lorsque des replanifications sont nécessaires, l'échantillonnage des nouvelles configurations est alors biaisé en utilisant les nœuds des sous-arbres encore valides dans le RRT précédent comme des candidats à l'extension du nouvel arbre d'exploration [ZKB07] (voir Fig. 1.7d).

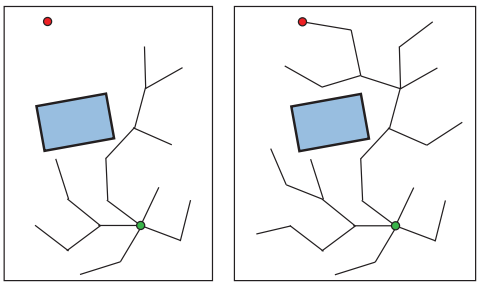
Réutiliser les RRTs calculés précédemment lorsqu'il est nécessaire de replanifier un chemin permet d'accélérer les nouvelles planification de l'agent lorsqu'un obstacle est détecté dans le chemin. Afin d'étendre cette idée, certaines méthodes proposent de conserver au sein d'une structure l'ensemble des RRTs déjà calculés pour pouvoir s'en servir ultérieurement lorsque de nouvelles requêtes de planification sont formulées. Alors que ces RRTs étaient précédemment conservés afin de biaiser la nouvelle planification, ces méthodes proposent de les stocker dans une structure et de construire ainsi une "forêt" de RRT permettant de créer, au fil des planifications successives, une représentation de plus en plus complète de l'espace des configurations. Lors des planifications de chemin, une solution est donc d'abord recherchée dans la forêt à disposition et si elle n'est pas trouvé un nouvel RRT est en général calculé et ajouté à la forêt afin de compléter la connaissance de l'environnement. Puisque nous nous plaçons dans le contexte



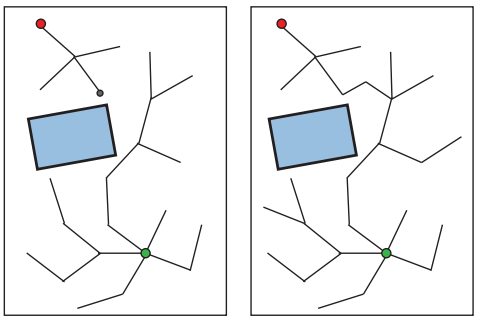
(a) Planification utilisant un RRT



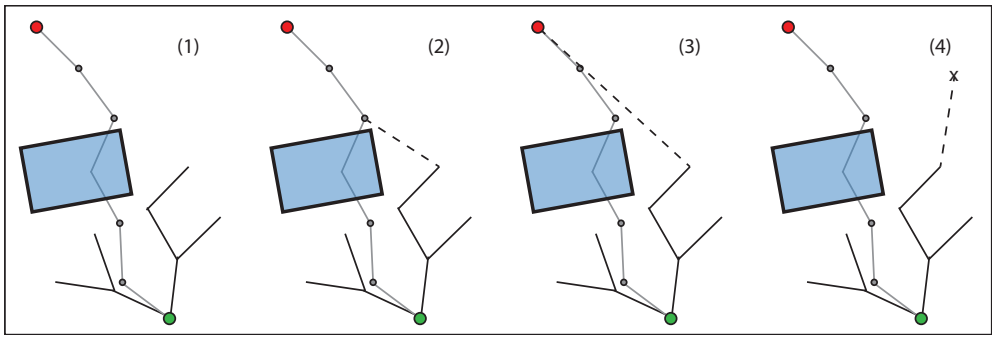
(b) Détection d'une obstruction



(c) DRRT [FKS06]



(d) Multipartite-RRT [ZKB07]



(e) ERRT [BV02] : Le nouvel échantillonnage a une probabilité de tirer soit une configuration de l'ancienne planification (2), soit la cible de la planification (3), soit une nouvelle configuration (4).

Figure 1.7 – Lorsqu'une obstruction est détectée dans un chemin calculé à l'aide d'un RRT (1.7b), les DRRTs (1.7c) proposent de conserver la partie valide du RRT en supprimant les sous-arbres invalidés alors que les Multipartite-RRTs (1.7d) conservent tous les sous-arbres valides qui peuvent être issus du RRT original. Les ERRTs (1.7e) de leur côté biaisent le nouvel échantillonnage en utilisant des points du chemin original.

d'environnements dynamiques où les éléments peuvent être ajoutés, déplacés ou enlevés, la configuration de l'environnement va être amenée à évoluer. L'utilisation de RRTs permet donc d'explorer rapidement les parties de l'espace des configurations qui ont été libérées mais également de supprimer de la "forêt" les arbres qui ont été invalidés par l'ajout d'un obstacle, ce qui permet de prendre en compte toutes les modifications observées. Les Reconfigurable Random Forest (RRF) permettent ainsi de conserver les arbres calculés lors des premières planifications et de simplifier de temps en temps la structure, en supprimant les nœuds redondants pour ne garder qu'une représentation compacte de l'environnement [Li02]. Lors de chaque planification de chemin, une solution est d'abord recherchée dans la forêt de RRT existante et, si elle n'est pas trouvée, de nouveaux arbres sont alors calculés et ajoutés à la représentation. Le concept introduit par les RRFs est ensuite étendu dans l'utilisation des Lazy Reconfigurable Forest (LRFs) [GKX07]. La forêt est alors construite en explorant l'environnement à l'aide de plusieurs RRTs simultanément et en tentant de les inter-connecter. L'agent de son côté navigue dans l'environnement en adoptant trois comportements selon sa situation courante. Si une solution pour atteindre le but est trouvée dans les LRFs, alors l'agent suit en priorité ce chemin. Si ce chemin n'existe pas ou qu'il est rendu invalide par la présence d'obstacles dynamiques, l'agent avance sur un chemin qui le rapproche du but, même s'il ne l'atteint pas. Pendant ce temps, les RRTs continuent d'explorer l'environnement jusqu'à pouvoir fournir une nouvelle solution à la planification. Enfin, s'il n'existe pas de chemin atteignant ou se rapprochant du but de la planification, l'entité peut également simplement attendre. Cette boucle perception/action permet notamment à l'agent de commencer à se déplacer localement vers son but avant même qu'une solution globale ne soit proposée.

Plutôt que de construire la représentation de l'environnement au fil de la simulation en explorant progressivement l'environnement à l'aide de RRT, certaines approches proposent de pré-calculer une représentation de l'environnement. Celle-ci est calculée en considérant l'environnement comme statique et en replanifiant un chemin si nécessaire afin de gérer les obstacles détectés. Ces méthodes se basent notamment sur le fait que la majeure partie de l'environnement reste généralement statique même en présence d'éléments dynamiques. Une carte de cheminement couvrant l'ensemble de l'environnement est donc générée dans un premier temps en utilisant en général des techniques de type PRM, puis cette représentation est raffinée au cours de la planification. Les lazy-PRMs sont par exemple basées sur les PRMs et sont construites sans tester si les configurations appartiennent ou non à C-Free, reportant cette validation des arcs et des nœuds au moment de la planification de chemin [BK00]. Ceux qui sont à ce moment détectés en collision avec C-Obstacle sont supprimés de la PRM. Cette mesure permet de limiter les détections de collision aux chemins et configurations potentiellement utilisés lors de la navigation de l'agent. D'autre part, afin de compléter la connaissance de l'espace des configurations au cours de la simulation, des budgets temps sont alloués au cours de la simulation afin d'échantillonner et d'ajouter de nouvelles configurations dans la PRM. Cette mesure permet de gérer ainsi des environnements où les objets sont déplacés en maintenant ainsi à jour une représentation de l'espace des configurations. Toutefois, si un obstacle est placé dans le chemin de l'agent, celui-ci devra de nouveau trouver

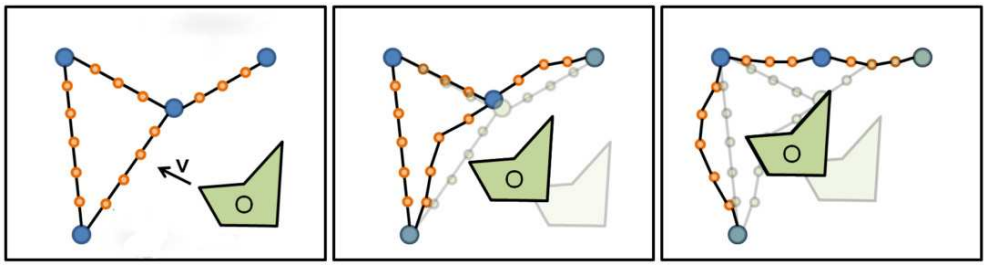
un chemin valide au sein de la PRM. La présence d'obstacles dynamiques peut créer de nombreuses obstructions au sein d'une PRM et créer également des déconnexions dans la carte de cheminement. Afin de pallier ce défaut et d'optimiser les performances de la planification au sein de la carte de cheminement, de nombreuses méthodes proposent d'utiliser simultanément plusieurs algorithmes de planification. Un RRT peut, par exemple, être utilisé conjointement à une PRM afin de compléter la représentation créée par cette dernière si aucune solution ne peut y être trouvée [JS04]. Il est également possible de coupler une PRM avec une subdivision en grille régulière, comme c'est le cas avec les Dynamic Roadmaps (DRM) [KM04]. Les nœuds et les arcs de la PRM sont alors mis en relation avec la cellule de la grille régulière couvrant la même partie de l'espace de configuration. Lorsque la configuration de l'environnement évolue, par le déplacement d'un obstacle par exemple, les cellules concernées de la grille régulière sont facilement identifiées et les éléments de la PRM associés sont annotés comme potentiellement obstrués. Lors de la planification de chemin, la détection de collision avec les obstacles est donc allégée et se focalise seulement sur les arcs étant annotés comme potentiellement obstrués, réduisant ainsi le coût de la planification. De plus, cette structure est également complétée par un RRT-Connect qui permet de calculer rapidement un chemin lorsqu'aucune solution n'est trouvée dans la PRM.

3.2 Adaptation dynamique de la représentation de l'environnement pour la planification de chemin

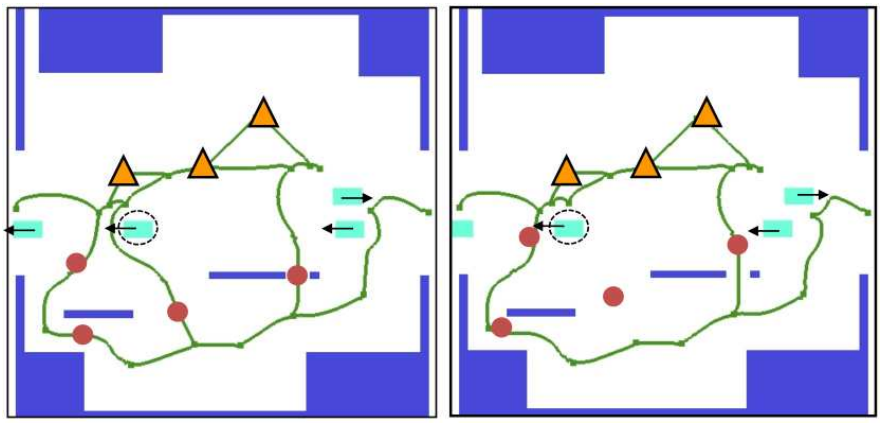
Les techniques présentées dans la partie précédente utilisent une représentation statique de l'environnement et cherchent ensuite à y planifier un chemin. Les obstacles dynamiques sont évités dans un second temps en planifiant, de nouveau, un chemin lorsqu'une obstruction est détectée. D'autres méthodes [GSLM07, SGA+07, SAC+07, SAC+08] s'intéressent à tenir compte de la dynamique de l'environnement directement au niveau de la représentation de l'environnement. Les cartes de cheminement générées par ces méthodes sont donc dynamiques et sont modifiées directement au fil de la simulation par le déplacement des éléments de l'environnement. Grâce aux mises à jour permanentes, la représentation de l'environnement reste ainsi correcte au cours de temps et les chemins planifiés sont également adaptés aux contraintes créées par la présence des obstacles dynamiques. En effet, la modification de la carte de cheminement va permettre d'adapter directement le chemin assigné à un agent en le déformant de la même manière. Il sera toutefois nécessaire de replanifier parfois un nouveau chemin lorsque des liens de la carte de cheminement seront, par exemple, rompus à cause de la proximité dangereuse d'un obstacle.

La dynamique des obstacles entraine des changements constants dans la configuration de l'environnement. Afin de gérer ces changements, certaines méthodes proposent de mettre à jour, au cours, de la simulation les cartes de cheminement associées à l'environnement. Celles-ci sont généralement qualifiées de cartes de cheminement "déformables" car elles réagissent aux déplacements des obstacles de l'environnement, repoussant les nœuds de ces cartes de cheminement et déformant les arcs existants. Ces cartes de cheminement sont généralement construites de manière à compter un nombre de nœuds

restreint, ce qui permet une mise à jour plus rapide de la structure. D'autre part, ces représentations sont généralement associées à des systèmes de particules. L'avantage d'un tel système est que chaque élément de l'environnement, ainsi que chaque nœud de la carte, peut générer des forces d'attraction, ou de répulsion, sur la carte de cheminement. Les forces d'attraction permettent en général de représenter les forces internes entre les éléments de la carte de cheminement, ce qui permet de conserver la connectivité entre les nœuds. Les forces de répulsion, à l'opposé, permettent de représenter les forces externes exercées par les objets dynamiques, ou les autres agents, traduisant ainsi la volonté de trouver un chemin loin de potentiels obstacles. De plus, les liens de la carte peuvent également être rompus lorsque les forces externes exercées sont trop fortes, ce qui peut créer des déconnexions. Les Reactive Deforming Roadmaps (RDR) sont, par exemple, composées de nœuds, appelés "balises", et de liens qui sont tous deux gérés par un système de particule [GSLM07] (voir Fig.1.8b). Cette carte est déformée en fonction des forces internes et externes qu'elle subit. Les liens sont ensuite cassés lorsque des conditions maximales d'élasticité sont atteintes. De nouveaux liens ou nœuds sont échantillonnés au besoin pendant la simulation, afin de garder une bonne représentation de l'environnement. L'utilisation de ce système permet de gérer la planification de chemin au sein d'un environnement avec des obstacles dynamiques, sans qu'aucune connaissance a priori sur les mouvements des obstacles ne soit nécessaire.



(a) Reactive Deforming Roadmaps (RDR) [GSLM07]

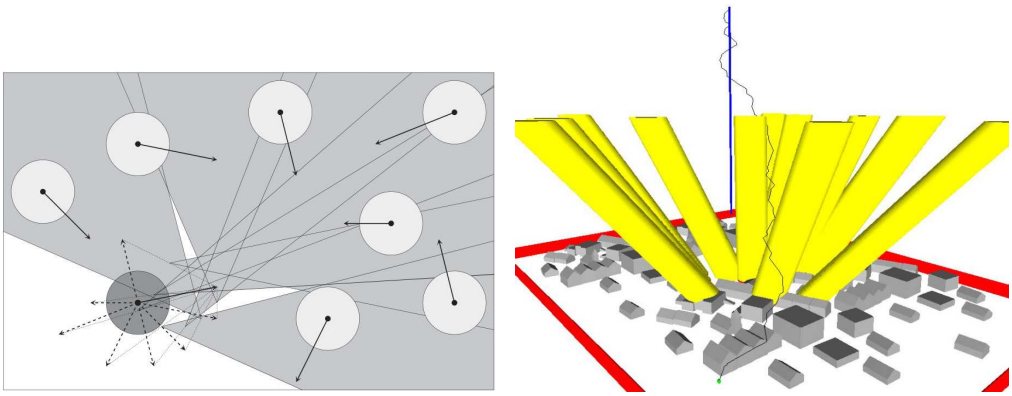


(b) Adaptative Elastic ROadmaps (AERO) [SGA+07]

Figure 1.8 – Les Reactive Deforming Roadmaps 1.8a ainsi que les Adaptative Elastic ROadmaps 1.8b sont des cartes de cheminement dont la structure est mise à jour au fil de la simulation par le déplacement des obstacles.

Une seconde alternative pour gérer les obstacles dynamiques d'un environnement est de créer une carte de cheminement basée sur des diagrammes de Voronoï généralisés. Celle-ci est ensuite remise à jour lorsque des obstacles sont détectés. Les diagrammes de Voronoï généralisés permettent de déterminer les points les plus éloignés des obstacles. Une carte de cheminement mise à jour de manière dynamique en utilisant ces diagrammes permet donc de déterminer directement des chemins présentant une marge de sécurité maximale vis-à-vis des obstacles de l'environnement. Plusieurs techniques telles que les Adaptative Elastic ROadmaps (AERO) [SGA+07] ou encore les graphes de navigation multi-agents (MaNG) [SAC+07, SAC+08] se basent sur ces techniques pour gérer la navigation de foules d'agents au sein d'environnements dynamiques. AERO propose d'utiliser ainsi des couloirs de navigation associés aux cartes de cheminement élastiques, afin de relâcher les contraintes sur la navigation des agents, et leur permettre d'exploiter au maximum l'espace libre à leur disposition dans l'environnement (voir Fig.1.8a). MaNG utilise des rendus sur cartes graphiques afin de calculer les diagrammes de Voronoï du premier et du second ordre et d'utiliser l'union de ces deux diagrammes pour la navigation. Cette méthode utilise également des forces d'attraction / répulsion entre les agents durant la navigation afin de créer des comportements de groupe lors des déplacements de ces derniers.

3.3 Planification de trajectoire



(a) Velocity Obstacles [FS98, vdBM08]

(b) Extrapolation des obstacles dans le domaines spatio-temporel [vdBFK06]

Figure 1.9 – 1.9a : La représentation des obstacles à l'aide de Velocity Obstacles permet d'identifier les différentes vitesses et directions de l'agent permettant de naviguer sans collisions dans l'environnement. 1.9b : Les déplacements des obstacles peuvent être extrapolés dans le domaine spatio-temporel (jaune) ce qui permet de planifier une trajectoire valide (noire) afin de les éviter.

Les techniques présentées jusqu'alors se focalisent sur le calcul de chemin dans l'environnement et sur des méthodes de replanification rapide lorsqu'un obstacle est détecté, ou sur une mise à jour au fil de la simulation de la carte de cheminement. Ces ap-

proches, toutefois, subissent la dynamique de l'environnement et s'y adaptent pour proposer des solutions de navigation. Certaines méthodes proposent également de gérer la dynamique de l'environnement en planifiant dans le domaine spatio-temporel. Augmenter la représentation géométrique de l'environnement par la dimension temporelle permet notamment d'anticiper le déplacement des éléments de l'environnement pour pouvoir en tenir compte lors de la planification. En considérant une représentation spatio-temporelle, la planification permet de calculer une trajectoire pour l'agent puisque l'information apportée par la dimension temporelle sera également incluse dans la planification. De plus cette planification permet de mieux prendre en compte l'aspect dynamique de l'environnement, dès la recherche de chemin, et de réduire ainsi les besoins de replanification a posteriori durant la navigation.

Certaines techniques [LK06] se basent ainsi sur un arbre de recherche, afin d'explorer l'environnement. Chaque nœud de l'arbre correspond à la représentation, à intervalles réguliers, de configurations valides de l'agent dans l'environnement. L'identification d'une séquence de configuration, lors de la planification, permet donc de déterminer la trajectoire que l'agent doit emprunter. L'utilisation de cet arbre de recherche permet également d'identifier, à court terme, les manœuvres à effectuer afin d'éviter les obstacles dynamiques présents. Ces techniques sont notamment basées sur l'utilisation des *Velocity Obstacles* [FS98] ou de leur extension, les *Reciprocal Velocity Obstacles* [vdBM08]. L'agent et les obstacles sont alors représentés dans l'espace de configuration par des cercles possédant un centre, un encombrement et une vitesse de déplacement. En regardant les vitesses relatives entre l'agent et les obstacles dynamiques, des cônes de vitesse sont extraits. Ces cônes de vitesse permettent de déterminer si il y aura une collision entre l'agent et l'obstacle en vérifiant si l'extrémité du vecteur de déplacement de l'agent arrive dans le cône associé à l'obstacle. Ces cônes sont appelés les *Velocity Obstacles* (voir Fig. 1.9a). En utilisant des heuristiques de recherche dans l'arbre, la trajectoire calculée satisfait une liste d'objectifs prioritaires tels qu'atteindre le but, maximiser la vitesse de déplacement ou conserver la structure de la trajectoire initiale. De plus, la vitesse et la direction de déplacement de l'agent sont également adaptées afin de rester en dehors de tous les *Velocity Obstacles* et donc d'éviter les collisions avec les obstacles environnant. Dans le cadre des *Reciprocal Velocity Obstacles*, les agents font l'hypothèse supplémentaire que les autres agents naviguent également de façon réactive dans l'environnement, ce qui permet de réduire les oscillations et de créer des trajectoires plus lisses.

Une alternative existe au calcul de configurations finies de l'agent dans son environnement par la construction d'un arbre de recherche. Cette solution consiste à utiliser des cartes de cheminement et à anticiper les déplacements des obstacles durant la planification. Certaines solutions, utilisant cette approche, permettent notamment de simplifier le problème, en considérant que les déplacements des obstacles sont entièrement connus a priori. Certaines techniques utilisent une planification à deux étages [LK05, LK06]. Cela permet, dans un premier temps, d'identifier des sous-buts locaux appartenant à la trajectoire globale puis, ensuite, de calculer les mouvements de l'agent permettant de naviguer entre ces sous-buts en s'adaptant aux contraintes dynamiques de l'envi-

ronnement. La planification locale est ensuite effectuée également à l'aide d'un arbre de recherche. Toutefois, cette solution a ses limites. Si les changements dans l'environnement local arrivent de manière imprévue une fois que cette planification locale a eu lieu l'agent ne peut pas adapter sa trajectoire à ces changements inopinés.

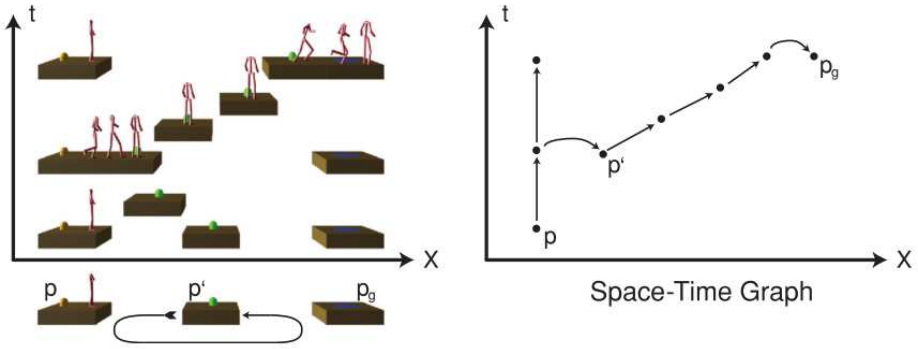
Afin de gérer la planification de trajectoire pour un agent au sein d'environnements dynamiques où les mouvements des obstacles peuvent être définis, il est possible d'utiliser des intervalles de sécurité afin de déterminer des configurations où l'agent peut attendre durant sa navigation [PL11]. Une autre alternative consiste à représenter les obstacles de l'environnement sous la forme de volumes spatio-temporels [vdBFK06]. La représentation de ces obstacles va être définie en fonction des connaissances disponibles sur leurs déplacements. Ainsi, un obstacle, dont la trajectoire est entièrement connue a priori, sera représenté par une extrusion de sa géométrie le long de sa trajectoire au cours du temps. À l'inverse, un obstacle dont la trajectoire est inconnue aura un volume spatio-temporel qui sera borné par son déplacement maximum dans le temps. La méthode proposée utilise une PRM, afin de capturer la topologie de la composante statique de l'environnement. La planification est effectuée dans cette PRM en utilisant les volumes spatio-temporels afin de calculer une trajectoire évitant les obstacles dynamiques (voir Fig. 1.9b). Lors de la navigation, l'information concernant les trajectoires des obstacles dynamiques se raffine et la trajectoire de l'agent est donc mise à jour en utilisant un algorithme de type D* [FS06].

Comme nous venons de le voir, différentes techniques permettent la planification de chemin ou de trajectoire au sein d'environnements dynamiques. La dynamique est générée soit en proposant des structures permettant de replanifier rapidement le chemin lorsque des obstructions sont détectées, soit en modifiant directement la carte de cheminement en fonction des déplacements des obstacles, soit finalement en planifiant une trajectoire évitant les obstructions dans le domaine spatio-temporel. Toutefois, la dynamique des environnements considérés par ces différentes techniques se restreint à des obstacles qui peuvent se déplacer dans l'environnement, ou être déplacés, ajoutés ou supprimés par l'intervention d'un utilisateur extérieur. De nombreux environnements virtuels, tels que ceux qui sont utilisés dans les jeux vidéo, présentent des aspects plus complexes en proposant au joueur de naviguer, par exemple, sur des plateformes en mouvement. De telles possibilités de navigation ne sont donc pas prises en compte par des algorithmes considérant les éléments dynamiques seulement comme des obstacles à la navigation. De nouvelles méthodes, permettant notamment la navigation sur des éléments dynamiques, doivent donc être proposées.

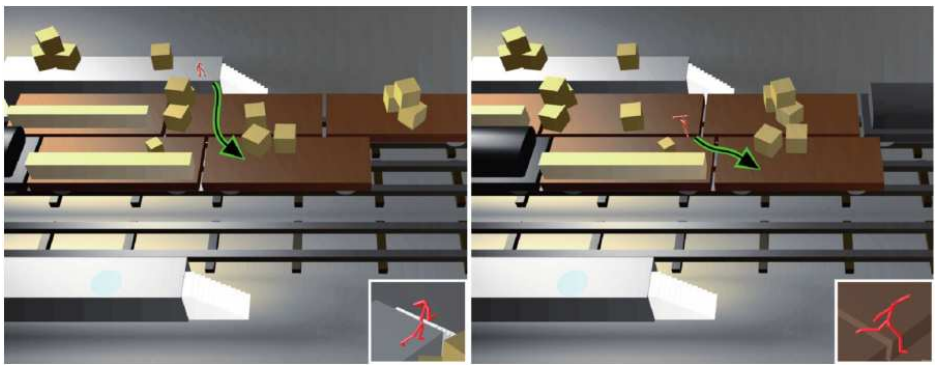
4 Planification de chemin en environnements complexes dynamiques

Les solutions présentées jusqu'ici s'intéressent à la planification de chemins en environnements statiques potentiellement peuplés par des obstacles dynamiques. Toutefois, de nombreux domaines utilisant actuellement les environnements virtuels cherchent à

les représenter de manière plus réaliste. Le réalisme de ces environnements va être le fruit de plusieurs facteurs. Il peut tout d'abord s'agir du rendu de l'environnement, mais également d'une simulation physique du monde, permettant de gérer entre autre la gravité. Ces environnements vont ainsi présenter de nombreux éléments dynamiques sans, toutefois, que ces objets ne se comportent que comme des obstacles. Une plateforme mobile peut par exemple être utilisée afin de connecter différentes zones de l'environnement. De manière similaire, un agent peut potentiellement naviguer sur une caisse, ou une planche tombée sous l'action d'un moteur physique. On peut ainsi imaginer que l'agent va être capable d'utiliser de nombreux objets différents afin d'accroître ses possibilités de déplacement, utilisant la caisse pour accéder à des endroits surélevés ou la planche comme un pont entre des surfaces trop éloignées. La prise en compte de ces éléments permet donc d'augmenter la navigabilité d'un environnement. Mais le problème de planification est également rendu plus complexe. En effet, les accès entre les éléments dynamiques doivent être détectés et caractérisés, augmentant ainsi l'espace de recherche lors de la planification.



(a) Planification dans le domaine spatio-temporel [LLKP11]



(b) Planification de trajectoire à travers des éléments mobiles

Figure 1.10 – 1.10a : La connaissance a priori des déplacements des éléments permet de détecter les futures connexions et d'en tenir compte dans la construction de l'arbre de recherche. 1.10b : Exemples de trajectoires traversant différents éléments mobiles navigables.

Très peu de recherches se sont pour le moment intéressées à la navigation dans de tels environnements. Toutefois, les développements récents apportés aux environnements virtuels dénotent un besoin, de plus en plus présent, d'avoir des solutions de planification au sein d'environnement présentant une grande dynamique. De plus, l'agent doit être capable de trouver des solutions de déplacement complexes dans des simulations interactives. Une première solution pour résoudre un problème de ce type a été proposée récemment par Levine et al. [LLKP11]. Lévine est le premier à proposer une solution à la planification de chemin pour des environnements dynamiques complexes où les surfaces des éléments dynamiques sont utilisées pendant la phase de planification. Afin de pouvoir naviguer sur ces différents éléments, l'environnement est tout d'abord précalculé hors-ligne avant la simulation. Un ensemble de configuration est associé à chaque surface de l'environnement caractérisée comme étant navigable. Ces configurations sont désignées comme des points de passage où l'agent peut accéder au cours de sa navigation. Cette méthode se base sur l'hypothèse forte que les mouvements des objets peuvent être connus a priori. Cela permet ainsi de définir totalement l'état de l'environnement à un temps donné. Lors de la planification, une trajectoire est donc recherchée dans le domaine spatio-temporel en anticipant les déplacements des obstacles, ainsi que des éléments navigables (voir Fig.1.10a). Un arbre de recherche explore le domaine spatio-temporel lors de la planification, afin de tenter de connecter les différents points de passage à l'aide de séquence de mouvements associés à l'agent. Lorsqu'un mouvement permet de connecter deux points, l'arc est alors ajouté dans l'arbre de recherche. Lorsque l'un des points considéré est mobile, un critère de proximité est utilisé avec les autres points de l'environnement, afin de limiter les tests de connexions. Puisque les déplacements des éléments mobiles sont entièrement connus a priori, il est possible de déterminer ainsi des trajectoires de l'agent entre des surfaces qui sont déconnectées dans l'espace mais aussi au cours du temps. La connexion des différents points de passage de l'environnement est faite en testant, par essai-erreur, différents contrôleurs de mouvement. Cela permet de vérifier si le mouvement en question peut être utilisé, en regardant notamment sa validité vis-à-vis des obstacles alentours. La trajectoire identifiée dans l'arbre de recherche permet de définir la séquence de contrôleur de mouvement à utiliser, ainsi que les délais d'attente entre ces différents mouvements pour pouvoir atteindre le but fixé.

Cette méthode de planification permet la navigation de l'agent au sein d'environnements dynamiques complets et complexes : les éléments dynamiques ne sont plus de simples obstacles mais également des surfaces où il est possible pour l'agent de naviguer (voir Fig.1.10b). Cette approche permet d'envisager de nouvelles possibilités de navigation dans les environnements virtuels. L'hypothèse, faite sur la connaissance a priori de l'évolution des éléments dynamiques au cours du temps, limite toutefois les champs d'application de cette méthode. En effet, de nombreux environnements ne répondent pas à ce pré-requis. En considérant, par exemple, des environnements utilisant un moteur physique, les mouvements des éléments ne vont généralement pas pouvoir être prédits à l'avance. D'autre part, le besoin de pré-calculer les points de passage associés à chaque objet limite également l'application à des éléments dont la configuration n'est pas amenée à changer au cours de la simulation. Cela n'est encore une

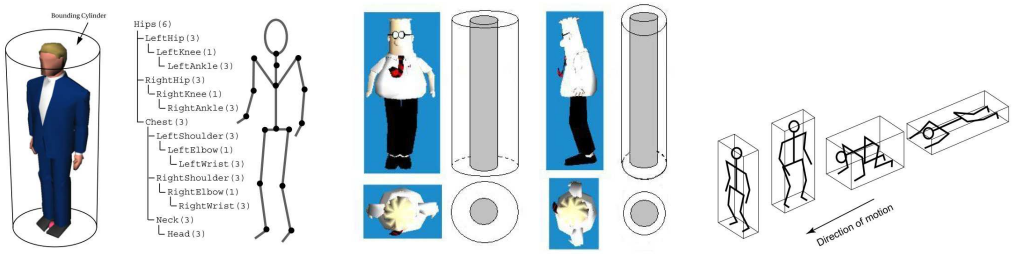
fois pas le cas dans des environnements physiques. Enfin, l'approche de Lévine utilise des tests, par essai-erreur, afin de trouver une trajectoire valide dans l'environnement. L'accessibilité entre les éléments n'est donc jamais explicitement formulée, et repose sur les mouvements testés lors de la planification. La topologie globale de l'environnement n'est donc pas connue de l'agent naviguant, mais testée au fur et à mesure de la planification. Cette technique ouvre donc vers de nouvelles possibilités mais présente encore des limitations qui contraignent grandement ses champs d'applications. Afin d'élargir le champ d'application à des environnements plus complexes, par exemple à des environnements physiques, il va falloir être capable de planifier des trajectoires au milieu d'éléments dynamiques dont les déplacements sont imprévus. Prendre en compte de tels environnements va toutefois augmenter la complexité du problème de planification. Une solution peut-être alors de réduire la complexité associée au mouvement de l'agent afin de diminuer l'espace de recherche.

5 Planification de chemin et mouvements de l'agent

La planification de chemin permet de trouver où l'agent doit se diriger afin d'atteindre un but qui lui a été fixé. Toutefois, afin d'atteindre ce but, l'agent va être amené à se déplacer le long de ce chemin en utilisant ses différentes capacités de mouvement. La planification de chemin et les mouvements utilisés par l'agent pour se déplacer sont les deux composantes du même problème, et il est donc impossible de s'intéresser au problème de planification sans prendre en compte les mouvements de l'agent. En effet, considérons par exemple un environnement simple où l'agent doit passer au-dessous d'une poutre placée en hauteur. Si cette poutre est basse, l'agent ne pourra pas passer en-dessous sans se baisser. S'il est capable de se baisser, alors cette tâche ne lui pose pas de problème. En imaginant que l'agent ne soit pas capable de faire cette adaptation alors la planification de chemin doit en prendre compte afin de déterminer un autre chemin ne possédant pas ce cas de figure. Toutefois, plus la représentation faite de l'agent est précise, plus il va posséder de degrés de liberté. Cela va augmenter de manière conséquente la dimension de l'espace des configurations à considérer, et la complexité de la tâche de planification. De nombreuses méthodes se sont donc intéressées à réduire la complexité du problème de planification en proposant différentes représentations de mouvements ou de postures associées à un agent.

Afin de simplifier le problème de la planification et de réduire le nombre de degrés de liberté certaines méthodes proposent de découpler les phases de planification de chemin et de génération de mouvements le long de ce chemin. Pour séparer ces deux phases, de nombreuses méthodes utilisent des volumes englobants. Ces volumes sont généralement des formes géométriques simples utilisées afin d'abstraire la représentation de l'agent au cours de la planification. Ceux-ci sont définis pour contenir à tout instant de la simulation l'ensemble de la géométrie de l'agent associée à un mouvement. Cette approximation permet tout d'abord de réduire le coût des détections de collision entre l'agent et son environnement en simplifiant la géométrie à considérer. En effet, le corps de l'agent étant borné par le volume englobant, il suffit que ce volume ne soit pas en

collision avec l'environnement pour être certain qu'il en est de même pour l'agent indépendamment de sa position à l'intérieur de ce volume. La tâche de planification est alors ramenée au déplacement du volume englobant dans l'environnement, tandis que l'animation est générée a posteriori lorsque l'agent se déplace sur sa trajectoire. Certaines méthodes proposent d'utiliser des polyèdres comme volumes englobants [SYN01] (cf. Fig.1.11c) mais la plupart du temps l'agent est représenté à l'aide d'un cylindre englobant [Kuf98, PLS03a, Li04, LK06] (cf. Fig.1.11a et Fig.1.11b). En représentant l'agent par un cylindre de centre C et de rayon R, on peut élargir la taille des obstacles de R et ramener ainsi le problème de navigation du cylindre à la planification de chemin du point C dans l'environnement dont les objets ont été élargis d'un rayon R.



(a) Kuffner 1998 [Kuf98] **(b)** Li 2004 [Li04] **(c)** Shiller 2001 [SYN01]

Figure 1.11 – Présentation de différents volumes englobants utilisés dans la littérature pour représenter un agent. Kuffner utilise un simple cylindre (a) tandis que Li en utilise plusieurs en fonction de la position de l'agent (b). Shiller utilise lui des polyèdres pour englober les postures de l'agent (c).

Lorsque l'animation de l'agent varie grandement entre plusieurs mouvements, l'utilisation d'un volume englobant unique pour le représenter apporte une trop grande approximation lors de la planification, et va donc créer des erreurs. Afin de gérer l'utilisation de mouvements différents au cours de la planification, plusieurs volumes englobants peuvent être associés à un même agent. Chaque volume représente alors une aptitude de déplacement de l'agent. La transition entre différents mouvements se traduira donc par un changement de volume englobant au cours de la planification et chaque volume permettra donc d'approcher au mieux un mouvement particulier. Certaines représentations [SYN01] proposent d'associer à chaque volume englobant une représentation associée de l'environnement (cf. Fig.1.11c). Ces diverses représentations sont ensuite reliées les unes aux autres. Cela permet des transitions entre les mouvements de l'agent et une adaptation de posture aux contraintes environnementales. L'animation est toutefois générée seulement au cours de la navigation de l'agent, ce qui permet de réduire le coût de la planification de chemin. D'autres méthodes utilisent conjointement un cylindre englobant et une représentation partielle de l'agent. Ceci permet d'obtenir une planification de chemin au coût raisonnable mais restant précise, notamment pour les mouvements des bras [PLS03a, PLS03b]. La partie inférieure de l'agent est ainsi ramenée à un cylindre possédant 3 degrés et la planification de chemin est effectuée pour ce cylindre. La trajectoire est ensuite lissée puis la posture du haut

de l'entité (bras, buste) est déformée le long de la trajectoire, afin de répondre aux contraintes externes, et d'éviter les obstacles.

Une fois un chemin trouvé dans l'environnement, différentes méthodes peuvent être utilisées afin de générer l'animation de l'agent lors de son déplacement. De nombreuses solutions utilisent les graphes de mouvements (*Motion Graphs*) [KGP02]. Ces graphes représentent une base de données de mouvements capturés sur des personnes réelles et dont les transitions entre les différents mouvements ont été calculés pour obtenir des animations réalistes. Trouver une animation pour l'agent le long du chemin va donc revenir à trouver une séquence de mouvements dans le graphe permettant de coller au maximum au chemin issu de la planification, c'est à dire de minimiser une fonction d'erreur entre le chemin calculer et l'animation générée. Les difficultés de ces techniques reposent notamment sur l'identification des connexions entre les différents mouvements, mais également sur l'adaptation de posture lorsqu'un agent rencontre une situation qui n'est pas dans le graphe de mouvement. Afin de gérer ces adaptations, certaines méthodes se sont donc intéressées à la possibilité de mélanger différents mouvements de la base de données ensemble, afin de pouvoir générer de nouveaux comportements [PLS03a, PLS03b, SH07]. D'un autre côté, certaines techniques utilisent un nombre restreint de mouvements et permettent ensuite une adaptation des postures de l'agent aux contraintes externes apportées par l'environnement [Lam09]. Ces techniques utilisent la cinématique inverse afin de calculer précisément les adaptations de postures tout en conservant les propriétés des mouvements initialement capturés.

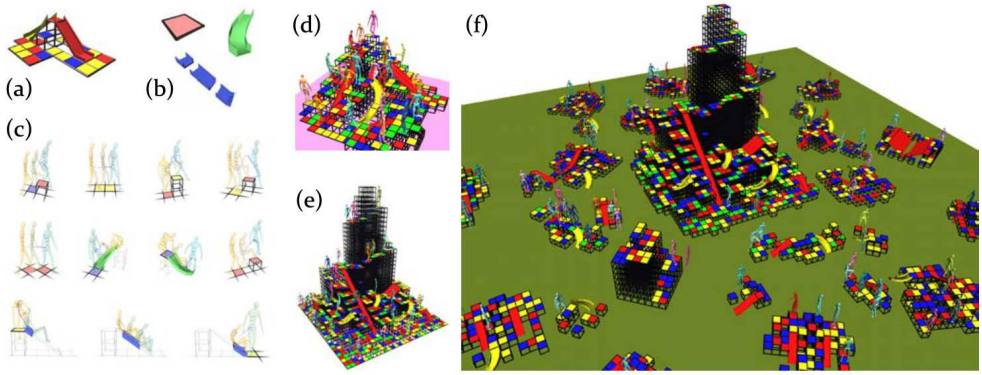


Figure 1.12 – Motion Patches [LCL06] (a) : Environnement source utilisé pour la capture de mouvements. (b) : Élements unitaires calculés à partir de l'environnement source. (c) : Motions Patches calculés (d)(e)(f) : environnements de démonstrations construits à partir de ces éléments.

Plutôt que de séparer le mouvement généré pour l'agent de la planification de chemin certaines méthodes lient étroitement les deux aspects. Une approche singulière, nommée *Motion Patchs*, propose de construire l'environnement à l'aide d'éléments, appelées des *patchs* ou tuiles, contenant directement les mouvements que l'agent peut effectuer lorsqu'il y navigue [LCL06]. L'information concernant les mouvements possibles pour traverser ces tuiles est issue de mouvements capturés sur des personnes réelles. Dans ce

cas particulier, l'espace des configurations n'est plus représenté explicitement puisque C-Free sera représenté par les tuiles utilisées et que les chemins possibles seront associés à ces tuiles. La planification de chemin va donc consister à identifier une séquence de tuiles que l'agent va devoir traverser. L'animation sera alors obtenue en jouant les mouvements associés aux différentes tuiles (voir Fig.1.12). Tout comme pour les Motion Graphs, l'une des principales difficultés consiste à connecter les différents mouvements ensemble afin d'obtenir des animations fluides lors de la navigation de l'agent. Mais en fonction des captures de mouvements effectuées, les mouvements synthétisés lors de la navigation peuvent être assez complexes comme la montée d'un escalier ou encore la descente d'un toboggan. Toutefois, la construction est limitée à l'utilisation d'objets pour lesquels une capture de mouvements a été effectuée. De plus, ces objets sont relativement simples afin de faciliter leur assemblage et se limitent à des tuiles carrées. Enfin, l'utilisation de captures de mouvements rattachées au éléments de construction de l'environnement empêche l'adaptation de l'entité à des situations imprévues telles que des obstacles apparaissant au milieu de son chemin.

Afin de lier planification et génération du mouvement, une autre approche consiste à enregistrer des informations concernant les mouvements dans la structure de planification de chemin. Il est ainsi possible d'utiliser une carte de cheminement ou un arbre de recherche ne caractérisant pas seulement des configurations de l'espace de recherche, mais contenant également des informations concernant les mouvements utilisés. Il est ainsi possible de construire une carte de cheminement de type PRM en échantillonnant directement des empreintes dans l'environnement, plutôt que des configurations [CLS03]. Ces empreintes, constituées d'un couple position/orientation, définissent les nœuds de la carte de cheminement. Une base de donnée constituée de mouvements capturés permet ensuite de connecter deux empreintes de la carte de cheminement à l'aide d'un mouvement existant. Afin d'augmenter les chances de connexions, les empreintes peuvent être légèrement déformées afin de mieux s'adapter aux mouvements considérés. La recherche de chemin va ainsi se traduire par l'identification d'une séquence de mouvement permettant de relier la position de départ à la position cible dans l'environnement. Une autre méthode permettant de générer directement une planification à partir d'une base de mouvement consiste à construire un arbre de recherche (*Precomputed Search Trees)* à l'aide de ces différents mouvements [LK05, LK06]. Un arbre de recherche est tout d'abord precalculé en prenant comme racine la position de l'agent (voir Fig. 1.13). Cet arbre permet de caractériser l'ensemble des positions atteignables par l'agent dans son environnement local. Celui-ci est construit en utilisant un automate d'état fini représentant les mouvements et transitions possibles par l'agent en tenant compte de ses capacités de déplacement. Lorsqu'une requête de planification est effectuée, un chemin global, déterminant des sous-buts sur le chemin, est tout d'abord déterminé. Une planification locale est ensuite effectuée en insérant l'arbre de recherche dans l'environnement à la position courante de l'agent. Les branches représentant des déplacements invalidés par la présence d'obstacles sont supprimées. Un chemin local est déterminé vers le sous-but suivant. Le but est ensuite atteint en répétant l'opération autant de fois que nécessaire. Cet arbre de recherche couplé à la connaissance des déplacements des obstacles permet à l'agent d'adapter sa trajectoire aux contraintes

dynamiques de l'environnement, tout en conservant des mouvements réalistes.

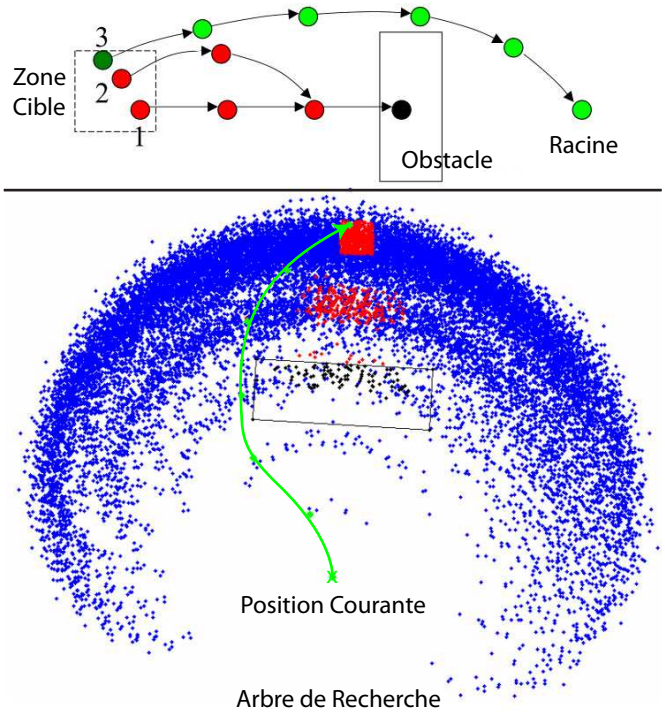


Figure 1.13 – Precomputed Search Trees [LK06] : Un arbre de recherche est précalculé en utilisant les mouvements potentiels de l'agent. La recherche est faite en identifiant les points appartenant à la zone ciblée puis en cherchant un chemin valide, sans collisions avec les obstacles, permettant d'y accéder.

Lévine [LLKP11] utilise des extensions de ces arbres de recherche afin d'identifier des trajectoires au sein d'environnements dynamiques. Afin d'identifier une trajectoire dans l'environnement, il construit en effet un arbre d'exploration en testant les différentes possibilités de navigation créées dans l'environnement. Il prend également en compte les déplacements connus des éléments dynamiques. Toutefois il permet en plus de détecter des accessibilités entre les éléments dynamiques de l'environnement en prévoyant à l'avance les connexions qui vont avoir lieu et en testant, par essai-erreur, les différents mouvements pouvant permettre d'effectuer la transition entre les surfaces de ces éléments.

Différentes approches ont donc été envisagées afin de gérer l'animation de l'agent au cours de sa navigation. Certaines méthodes, utilisant par exemple les volumes englobants, essaient de décorréler au maximum la planification de chemin des mouvements que l'agent doit adopter alors que d'autres, telles que les *Precomputed Search Trees* [LK05, LK06] ou les *Motion Patchs* [LCL06], reposent entièrement sur les mouvements de l'agent afin d'identifier un chemin. Les méthodes reposant principalement sur les mouvements de l'agent permettent généralement de produire de très bonnes animations lors de la navigation et d'obtenir une animation très proche de la réalité. Ce réalisme

est notamment du à l'utilisation de données qui ont été capturées sur des humains réels afin d'être ensuite rejouées sur des agents virtuels. L'utilisation de telles données va cependant avoir un coût important. En effet, afin de créer une animation et de gérer de manière précise les transitions entre les différents mouvements de la base de nombreux précalculs vont être nécessaires. La recherche dans la base de donnée sera elle aussi coûteuse afin de trouver une bonne animation libre de collision. Les volumes englobants permettent d'accélérer notamment à la planification en s'abstrayant des mouvements de l'agent. La planification de chemin s'en trouve accélérée, mais les chemins calculés sont moins précis, puisque les positions exactes de l'agent ne sont pas considérées. L'animation de l'agent lors de la planification de chemin repose donc principalement sur un compromis, entre la qualité et la précision désirée de ses mouvements d'un côté, et les contraintes temporelles déterminant quel délai peut-être alloué à la génération de l'animation. Des applications interactives privilégieront donc des solutions demandant un temps de calcul faible au détriment de la qualité de l'animation générée tandis que des applications n'ayant pas de contraintes temporelles vont utiliser des solutions plus coûteuses en temps de calcul mais permettant des animations plus réalistes.

6 Conclusion

De nombreuses solutions ont été proposées pour la planification de chemin au sein d'environnements virtuels. Ce domaine de recherche est très actif dans le contexte des mondes virtuels et de l'animation comportementale dont le but consiste à automatiser le peuplement des environnements virtuels à l'aide d'agents virtuels autonomes. Toutefois, les solutions proposées s'adressent chacune à des problèmes de planification de chemin particuliers pouvant être classés en différentes catégories. Les premières techniques proposées se sont tout d'abord concentrées sur la navigation au sein d'environnements statiques. En réduisant le problème à des situations statiques, il est ainsi possible de pré-traiter les environnements afin notamment de simplifier l'exploration de l'espace des configurations lors de requêtes de planification. Certaines méthodes proposent de représenter l'environnement à l'aide de subdivision exactes ou approchées de l'environnement. Ces méthodes permettent d'obtenir de bonnes représentations de C-Free et de C-Obstacle, mais conviennent principalement à des environnements de faible dimensions. Les méthodes basées sur les cartes de cheminement permettent d'un autre côté d'explorer l'environnement pour en construire une représentation explorant de manière plus performante les espaces de configurations de plus grandes dimensions. Plus récemment, la problématique de la planification de chemin s'est étendue à des environnements peuplés d'obstacles dynamiques. Au sein de tels environnements, l'agent doit être capable de détecter les obstacles qui se sont déplacés, et d'adapter son chemin en fonction. Cette approche rend le problème de navigation plus complexe, mais également plus intéressant, en élargissant les possibilités d'applications. Certaines techniques se sont basées sur une représentation de l'environnement statique couplée à des méthodes de replanification. Cela permet de réparer les chemins des agents invalidés lors de déplacements d'obstacles. D'autres proposent d'adapter la carte de cheminement en la déformant en réaction aux mouvements des obstacles. Une troisième approche consiste

à ne plus planifier un chemin à un temps donné, mais à anticiper les déplacements dans le domaine spatio-temporel, et de calculer ainsi une trajectoire de l'agent dans ce domaine en prenant le temps comme l'un des paramètres de la planification. Néanmoins, les éléments dynamiques gérés au sein de ces environnements sont considérés seulement en temps qu'obstacles.

Les éléments dynamiques présents dans un environnement peuvent présenter des caractéristiques qui autorisent un agent à naviguer dessus. Par la même occasion, cela augmente les possibilités de navigation offerte à l'agent en question. Un container-poubelle, un canapé ou encore une caisse peuvent par exemple être déplacés et utilisés afin d'accéder à des endroits élevés. De la même manière, un agent pourrait utiliser des plateformes mobiles afin de pouvoir naviguer entre des zones de navigation déconnectées. Ces diverses situations n'ont été abordées que récemment. Une seule solution propose actuellement de gérer un agent naviguant dans un environnement dynamique présentant cette complexité. Toutefois, cette solution utilise une hypothèse forte en admettant que le déplacement des objets dynamiques est entièrement défini a priori et que l'agent le connait lors de sa planification. Cette hypothèse restreint les domaines d'application et ne permet pas de gérer une navigation au sein d'environnement imprévisibles où les déplacements des objets ne peuvent être connus à l'avance. De tels environnements sont toutefois de plus en plus présents, grâce aux développements des moteurs physiques fréquemment intégrés dans les environnements pour en augmenter le réalisme.

Dans le cadre de cette thèse, nous nous sommes intéressés à la planification de chemin au sein d'environnements complexes entièrement dynamiques. Ces environnements peuvent ainsi être peuplés d'objets devant être considérés d'une part comme des obstacles dynamiques, mais également comme de potentielles zones navigables pouvant aider l'agent durant ses déplacements. De plus, contrairement à l'approche de Lévine [LLKP11], nous nous intéressons à des environnements imprévisibles. Au sein de tels environnements, le déplacement des éléments dynamiques n'est pas connu a priori mais seulement observé par l'agent durant la simulation. En considérant un monde virtuel augmenté d'un moteur physique, comme c'est le cas par exemple de nombreux jeux actuels, les changements dans l'environnement vont résulter des actions des différents agents ainsi que de la gravité et ne vont donc pas être prévisibles. De tels environnements vont donc demander à l'agent de s'adapter rapidement aux éléments dynamiques dans l'environnement, afin de pouvoir y naviguer de manière autonome. Pour doter ces agents de la capacité de naviguer au sein de tels environnements, il va donc falloir proposer une solution de navigation permettant de caractériser efficacement les obstructions, ainsi que les accès crées dans le monde virtuel, sans avoir de connaissances a priori sur ces éléments dynamiques.

Représentation de l'agent et des objets de l'environnement

2

Table des matières

1 Représentation de l'agent **42**
1.1 Découpler animation de l'agent et planification de son chemin 42
1.2 Caractérisation de l'agent 43
2 Caractérisation des objets **45**
2.1 Préambule 46
2.2 Les Volumes d'Interaction : Caractérisation de l'impact topologique d'un objet 47
2.3 Mouvements des objets et Volumes d'Interaction 57
3 Conclusion **60**

Les environnements dynamiques sont généralement peuplés par un grand nombre d'éléments mobiles qui, par leur ajout, leur suppression ou leur déplacement, modifient la configuration du monde. Au cours de la simulation, un agent est donc amené à caractériser la topologie existante afin de pouvoir se frayer un chemin. Toutefois, la représentation d'un tel environnement et des interactions existantes est coûteuse à cause des changements fréquents apportés à la topologie au cours du temps. Dans le cadre d'applications interactives, ces temps de calcul sont un facteur important, et les performances des méthodes proposées sont donc importantes. Afin de représenter l'impact d'un objet sur l'environnement, nous proposons de prétraiter les éléments de manière à définir une représentation de leur impact local. Cette représentation va tout d'abord se traduire par une définition de l'agent et de ses capacités de déplacement afin de comprendre l'étendue de ses possibilités de navigation. Dans un second temps, l'accent est mis sur la représentation des objets géométriques de l'environnement. La mise en commun de l'agent, de ses capacités de déplacement et de la géométrie d'un élément, nous permet de définir une représentation explicite de l'interaction agent-objet existante, et de caractériser ainsi les possibilités de navigation de l'agent vis-à-vis de l'objet en question. La représentation duale des objets géométriques que nous avons ainsi définie permet de caractériser l'impact topologique d'un objet en terme d'accessibilité et d'obstruction sur son environnement local. Cette représentation des propriétés topologiques associées à un objet est faite à l'aide des Volumes d'Interaction qui permettent de caractériser des sous-ensembles de l'espace des configurations.

1 Représentation de l'agent

La représentation de l'agent est essentielle à la définition du problème de planification de chemin. En effet, sa forme géométrique ainsi que les degrés de liberté qui lui sont associés vont permettre de définir le problème de planification de chemin tandis que ses capacités de déplacement vont définir les outils permettant d'y trouver une solution. La définition de l'agent est donc une tâche préalable nécessaire à tout problème de planification.

1.1 Découpler animation de l'agent et planification de son chemin

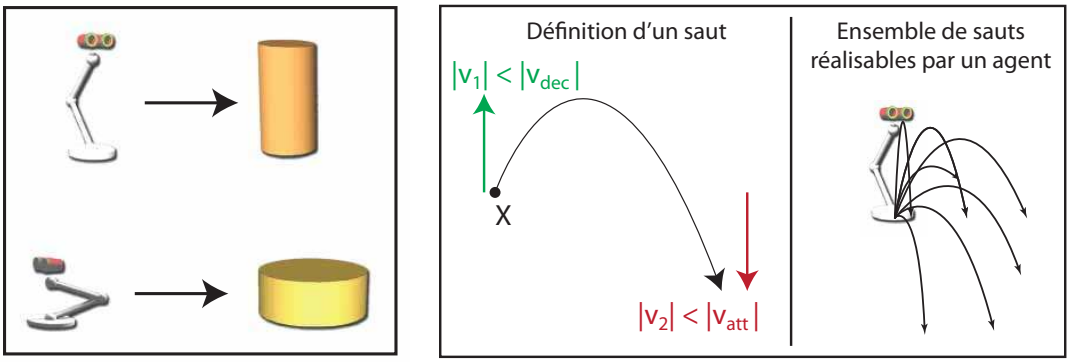
La représentation de l'agent va être différente selon les problèmes de planification à adresser. Afin de réduire les temps de calcul, les méthodes proposées doivent chercher à réduire la complexité du problème. Une représentation de l'agent avec l'ensemble de ses degrés de liberté permettra une planification de chemin précise dans l'environnement [JS04]. Mais cela se fera au prix d'une recherche dans un espace de grande dimension et donc d'un coût important en terme de temps de calcul.

La planification de chemin en environnements dynamique est une problématique complexe puisque l'évolution de l'environnement, qui n'est pas forcément connue a priori, doit être prise en compte lors de la recherche de chemin. Afin de proposer une solution de planification compatible avec une application interactive, nous réduisons la complexité de la planification en simplifiant la représentation associée à l'agent, à l'aide de volumes englobants [Kuf98, PLS03a, Li04, LK06]. Ces volumes permettent d'une part de réduire le nombre de degrés de liberté de l'agent mais également de décorréler la phase de planification de chemin de la phase d'animation de l'agent lors de sa navigation. Comme nous avons pu le voir au cours de la Section 5 de la bibliographie, ces volumes englobants permettent de représenter l'agent à l'aide d'une géométrie simple bornant ses mouvements. La dimension de l'espace de recherche est ainsi réduite, ce qui diminue également la complexité du problème de planification. Toutefois, la posture de l'agent pouvant varier largement entre différents mouvements, l'utilisation d'un volume englobant unique entraîne des erreurs d'approximation importantes. D'autre part, chaque capacité de mouvement va posséder des propriétés particulières et, la représentation de l'ensemble des capacités de déplacement par un volume englobant unique, entraîne la perte de ces propriétés particulières. Un agent utilisant un mouvement de marche accroupi peut par exemple accéder à des positions où un mouvement de marche normal ne permet pas l'accès. Afin de gérer ces différentes capacités, il est possible de borner chacun des mouvements par un volume englobant différent [SYN01, Li04]. La transition entre différents mouvements de l'agent va ainsi se traduire par un changement de volume englobant lors de la planification. Nous associons ainsi à chaque capacité de déplacement de l'agent un cylindre englobant. La formulation du problème de planification de chemin pour l'agent considéré se ramène ainsi à une planification de chemin pour le centre de la base du cylindre. Le centre de la base de chaque cylindre étant un

point de coordonnées (x, y, z), l'espace des configurations est ainsi ramené à un espace tridimensionnel. Cette réduction de la dimension de l'espace des configurations permet également de représenter l'environnement et l'espace des configurations dans des bases de même dimension, simplifiant la correspondance entre les deux espaces.

Comme présenté précédemment, l'utilisation de volumes englobants nous permet de découpler l'animation de l'agent de la planification de chemin. Cette propriété permet de réduire la complexité du problème de navigation de l'agent. Cela offre également l'avantage de pouvoir abstraire la méthode de planification proposée de la nature de l'agent. En effet, que l'agent virtuel soit un humanoïde, une lampe animée ou un chien, la méthode présentée pour la planification de chemin sera la même, à condition de pouvoir représenter l'agent à l'aide de cylindres englobants. Les différences entre ces agents se feront alors au niveau de l'animation générée pendant leurs déplacements.

1.2 Caractérisation de l'agent



(a) Des cylindres englobants sont associés à l'agent

(b) Définition d'un saut et exemple de sauts possibles

Figure 2.1 – 2.1a : Un cylindre englobant l'agent est associé à chaque capacité de déplacement. 2.1b : Caractérisation d'un mouvement de saut.

Un agent est donc doté de plusieurs capacités de déplacement M_i lui permettant de naviguer dans son environnement. Il lui est par exemple possible de marcher mais également de s'accroupir, courir ou sauter. Ces capacités de déplacement vont pouvoir être décomposées en deux familles. D'un côté, nous identifions les capacités de déplacements contraints au sol telles que la marche par exemple, tandis que, d'un autre côté, nous avons des capacités de déplacement permettant de se détacher de cette contrainte telles que des mouvements de sauts. Chaque capacité de déplacement possède des caractéristiques propres permettant de définir le mouvement associé. Afin de représenter ces caractéristiques, nous associons à chaque capacité de déplacement M_i plusieurs paramètres :

– Un cylindre englobant C_i bornant les postures de l'agent

- Une pente navigable maximale p_{nav}
- Une vitesse minimale de déplacement v_{min}
- Une vitesse maximale de déplacement v_{max}

La figure 2.1a présente ainsi deux exemples de capacités de déplacement auxquels sont associés des cylindres englobants. L'ensemble des cylindres englobants C_i associés à un agent est stocké dans un ensemble $\mathcal{C}$. La pente navigable p_{nav} permet à l'agent de ne pas limiter sa navigation à des environnements plats mais d'utiliser ses capacités de déplacements pour la navigation sur des sols en pente [RSM+09]. Les vitesses v_{min} et v_{max} permettent de leur côté de connaître les vitesses de déplacement de l'agent au cours de sa navigation.

Les capacités de déplacement permettant à l'agent de se détacher du sol vont lui permettre d'accéder à de nouvelles zones déconnectées de l'environnement. Comme le montre la figure 2.1b, nous représentons ces mouvements à l'aide d'une trajectoire parabolique. La première partie de la parabole caractérise une phase de décollage permettant à l'agent de quitter le sol et la seconde partie est une phase d'atterrissage. Afin de caractériser ces mouvements nous leur associons deux paramètres supplémentaires :

- Une vitesse verticale maximale de décollage v_{dec}
- Une vitesse verticale maximale d'atterrissage v_{att}

La vitesse de décollage v_{dec} permet de définir l'impulsion de l'agent lors de ses sauts. De plus, bien que l'agent ne soit pas concrètement limité au niveau de la vitesse d'atterrissage, v_{att} nous permet de définir une vitesse verticale d'atterrissage de confort pour l'agent. En utilisant une vitesse d'atterrissage supérieure à v_{att}, l'agent pourrait être dans des situations risquées et être potentiellement endommagé par la chute.

Lors de la planification de chemin dans l'environnement, l'agent peut être amené à modifier sa posture afin de changer de capacité de déplacement lors de sa navigation. Plusieurs solutions ont été proposées dans la littérature pour gérer cette transition entre capacités de déplacement. On peut tout d'abord choisir de jouer un nouveau mouvement de transition connu entre les deux postures à l'aide par exemple d'un graphe de mouvement [KGP02] représentant les mouvements possibles de l'agent. Une autre solution consiste à mélanger les deux mouvements afin d'interpoler la transition entre les deux [KG03]. Dans les deux cas, il est possible de calculer un nouveau cylindre englobant $T_{i,j}$ pour cette transition. Ce cylindre évite ainsi que l'agent n'entre en collision avec son environnement au cours d'une transition d'un mouvement M_i à un mouvement M_j. Ces cylindres de transition sont conservés dans l'ensemble $\mathcal{T}$. Une transition entre deux capacités de déplacement est possible lorsque celles-ci partagent des propriétés communes. En considérant par exemple des mouvements contraints au sol, on s'intéressera aux intervalles de vitesses $[v_{min}, v_{max}]$ qui leur sont associés afin de déterminer si il existe un sous-ensemble commun de vitesses permettant la transition. Comme le montre la figure 2.2, la mise en correspondance de ces capacités de déplacement revient à construire un graphe de transition identifiant les liens entre les différentes capacités.

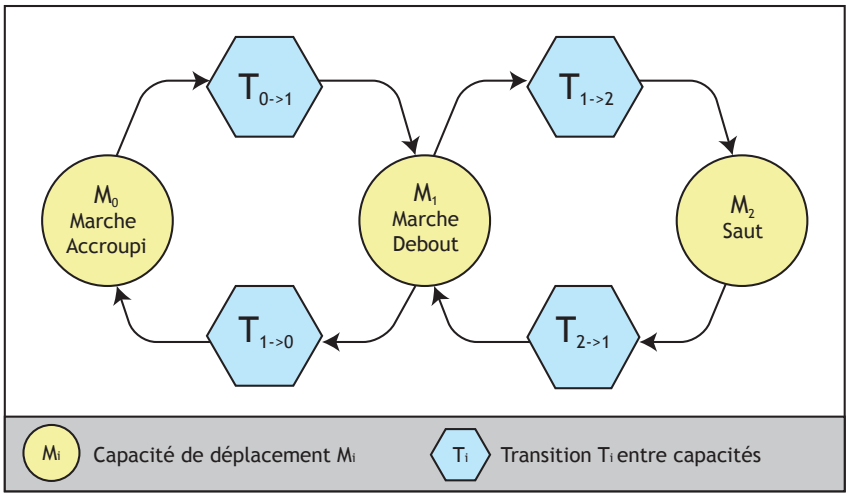


Figure 2.2 – Un graphe de transition permet d'identifier les transitions possibles entre les différentes capacités de déplacement.

En conclusion, nous représentons l'agent au sein de l'environnement par un ensemble de capacités de déplacement M_i. Un agent possédant généralement plusieurs capacités de déplacement, nous définissons également un ensemble de transitions $T_{i,j}$ afin de représenter les mouvements de transitions entre deux capacités de déplacements. En associant, de plus, des informations supplémentaires à chaque capacité de déplacement, nous permettons d'anticiper la navigation de l'agent sur des sols pentus mais également les transitions que l'agent va pouvoir effectuer. Enfin, nous caractérisons également des mouvements permettant à l'agent de se déconnecter du sol et de changer ainsi de zone de navigation. En nous basant sur cette représentation de l'agent, nous allons maintenant définir la représentation des interactions entre cet agent et les objets de l'environnement que nous avons introduits au cours de cette thèse.

2 Caractérisation des objets

Un environnement est composé de multiples éléments sur lesquels il est possible de naviguer. Une chaise ou un escabeau peuvent par exemple être utilisés afin d'accéder à d'autres endroits. De la même manière, une planche va pouvoir être utilisée comme un pont reliant deux zones de navigations déconnectées. La représentation de telles relations au sein d'environnements dynamiques est complexe à cause des changements constants intervenant dans la configuration des éléments. Nous définissons ici une représentation duale des objets géométriques permettant de caractériser les interactions possibles entre un objet quelconque de l'environnement et l'agent. Cette caractérisation repose sur la mise en relation des capacités de déplacement de l'agent et de la représentation géométrique de l'objet. Cette représentation géométrique est généralement le maillage triangulaire permettant de représenter l'objet dans un environnement virtuel

en 3D. Notre objectif ici est donc de proposer une représentation duale d'un tel objet décrivant l'impact de sa géométrie sur la navigation de l'agent, soit en terme d'obstruction, ou d'accessibilité. L'analyse de la géométrie couplée aux capacités de déplacement de l'agent permet d'identifier les possibilités de navigation offertes par cet objet et de représenter dans l'espace des configurations. Cette représentation duale dans l'espace des configurations est appelée Volumes d'Interaction.

2.1 Préambule

Observations

Nous observons généralement que les objets présents au sein d'un environnement ont une influence directe sur la navigation des agents. Cette influence dépend des caractéristiques de l'agent considéré et de ses capacités de déplacement. L'obstruction d'une zone par un objet dépend ainsi de l'occupation spatiale de l'agent considéré. De la même manière, la navigation entre différentes surfaces de l'environnement va reposer sur les capacités physiques de l'agent. Un agent sportif aura plus de facilité à sauter de longues distances comparé à un agent fatigué ou peu entraîné. À l'inverse, un agent de petite taille aura la possibilité d'accéder à des espaces réduits en se baissant. La définition de l'accessibilité ou d'une obstruction est donc dépendante de l'agent et va varier en fonction de ses capacités. Il nous a donc paru important, au regard de ces observations, de conserver cette propriété et de caractériser l'impact topologique d'un objet en se basant sur les capacités de déplacement de l'agent naviguant.

Notations

Afin de clarifier notre discours, nous introduisons dans cette section quelques notations qui seront utilisées dans les sections suivantes. L'agent est immergé dans un environnement virtuel tridimensionnel (3D) peuplé d'objets statiques et dynamiques au milieu desquels il va naviguer. Cet environnement va être désigné comme l'espace de travail E_t. L'agent est représenté par ses capacités de déplacements et par les cylindres englobants associés. La position $p(x, y, z)$ du cylindre englobant va désigner la position de l'agent dans E_t, cette position $p(x, y, z)$ correspondant au centre de la base du cylindre. L'ensemble de l'espace de travail E_t et des éléments le composant est également représenté dans l'espace des configurations, noté E_c, où la planification de chemin sera effectuée. Le nombre de degrés de liberté de l'agent définit la dimension de l'espace des configurations. L'agent que nous considérons étant représenté à l'aide de cylindres englobants, le problème de planification se ramène à une planification de chemin pour le centre du cylindre. L'espace des configurations associé est donc tridimensionnel. Cette correspondance entre l'espace de travail E_t et l'espace des configurations E_c est une propriété essentielle de notre représentation et permet de faciliter la représentation des éléments de E_t dans E_c. Cette représentation met en effet directement en correspondance chaque configuration $c(x, y, z)$ de E_c à une position $p(x, y, z)$ de E_t pour la planification.

2.2 Les Volumes d'Interaction : Caractérisation de l'impact topologique d'un objet

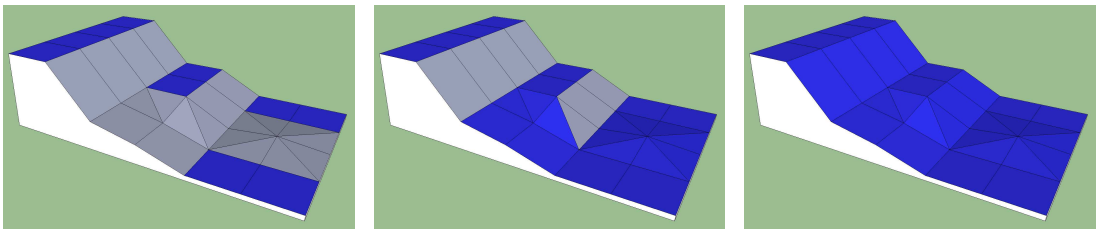
Afin de représenter l'impact topologique des objets présents dans l'environnement nous introduisons les Volumes d'Interaction. La définition de ces Volumes d'Interaction associe à chaque objet O_i présent dans l'espace de travail E_t une représentation duale dans l'espace des configurations E_c. Cette représentation explicite caractérise l'impact topologique de l'objet O_i sur son voisinage dans l'espace des configurations local à l'objet noté $E_c(O_i)$. Les différentes interactions possibles entre un agent et un objet O_i nous permettent de caractériser plusieurs sous-espace de $E_c(O_i)$. Un agent pouvant posséder plusieurs capacités de déplacement M_k, les Volumes d'Interaction sont donc définis pour chaque couple (O_i, M_k) existant. Au cours de l'état de l'art, nous avons vu que l'espace des configurations était généralement divisé en deux sous-espaces : *C-free* et *C-obstacle*. *C-obstacle* représente la partie de l'espace de configuration interdite à l'agent et *C-free* la partie à travers laquelle l'agent peut naviguer. Pour mieux caractériser l'impact d'un objet sur la navigation de l'agent, nous raffinons cette représentation et divisons l'espace des configurations E_c en quatre sous-ensembles :

- *C-obstacle*, définit comme précédemment l'ensemble des configurations où l'agent est en collision avec l'objet
- *C-navigable*, caractérise l'ensemble des positions appartenant à l'objet où l'agent va être capable de se tenir et de naviguer
- *C-accessible*, représente les configurations de l'objet et autour de l'objet où l'agent va pouvoir accéder à partir de cet objet
- *C-out*, englobant l'ensemble des configurations restantes ne représentant pas une interaction possible de l'agent avec l'objet

Les trois premiers sous-ensembles permettent de caractériser l'impact topologique d'un objet et vont être représentés explicitement dans l'espace des configurations à l'aide des Volumes d'Interaction. *C-obstacle* va ainsi être représenté à l'aide des Volumes Interdits, *C-navigable* à l'aide des Surfaces Navigables et enfin *C-accessible* à l'aide des Volumes d'Accessibilité. Nous nous focaliserons, pour plus de clarté dans le discours, à la représentation des Volumes d'Interaction pour une seule capacité de déplacement M_k.

Les Surfaces Navigables : représentation de l'espace de navigation

Certains objets présentent des parties de leur géométrie permettant à un agent de s'y tenir et d'y naviguer. C'est par exemple le cas d'une chaise ou d'une table sur lesquelles un agent pourrait monter. La prise en compte de cette propriété permet d'étendre le domaine navigable de l'agent non seulement au sol de l'environnement, mais également aux objets présents. Afin qu'un agent puisse utiliser ces éléments lors de sa navigation, nous définissons dans un premier temps une identification, et une représentation de ces zones navigables dans l'espace des configurations.



(a) Pente navigable maximale = 0°

(b) Pente navigable maximale = 20°

(c) Pente navigable maximale = 45°

Figure 2.3 – Les zones bleues indiquent les Surfaces Navigables de l'élément définies en se basant sur la pente maximale navigable par l'agent.

Afin de capturer cette navigabilité, nous associons à chaque objet O_i une Surface Navigable, annotée $Ns(O_i, M_k)$. Cette surface représente l'ensemble des positions de l'objet O_i où un agent est capable de se tenir lorsqu'il utilise sa capacité de déplacement M_k. Comme nous l'avons décrit précédemment, une pente navigable seuil est associée à chaque capacité M_k. Cette pente navigable nous permet de déterminer dans quelle mesure l'agent est capable de naviguer sur les zones inclinées d'un élément. La définition des Surfaces Navigables se base sur la mise en relation des capacités de déplacement de l'agent et de l'objet géométrique considéré. Les objets géométriques de l'environnement étant construits à l'aide d'un maillage triangulaire, l'analyse des triangles nous permet de déterminer l'ensemble des surfaces dont la pente est navigable. Si l'angle entre l'axe vertical de l'espace de travail et la normale à un triangle du maillage de l'objet est compris dans l'intervalle de pente navigable associé à M_k, alors ce triangle est navigable du point de vue de l'agent. Dans le cas contraire, la pente est trop inclinée pour que l'agent puisse y naviguer. La figure 2.3 montre un exemple de définitions de Surfaces Navigables pour un objet O_i en utilisant différentes valeurs de pentes navigables (angles de 0°, 30°, 45°). L'utilisation de différentes valeurs mène à la création de Surfaces Navigable différentes. Comme nous pouvons le voir, plus le critère est relâché, plus les possibilités de navigation offertes à l'agent sont étendues.

L'identification des triangles navigables permet de caractériser l'ensemble des configurations $c(x, y, z)$ dans E_c associées à l'ensemble des positions $p(x, y, z)$ où l'agent est capable de naviguer dans l'espace de travail E_t. La figure 2.4 met en correspondance l'espace de travail E_t où est placé une caisse avec l'espace des configurations E_c associé. Les cylindres bleus placés dans l'espace de travail représentent des positions où l'agent est capable de se tenir sur la caisse. L'ensemble de ces positions dans l'espace de travail est capturé et représenté par la Surface Navigable dans l'espace des configurations, à savoir ici, la surface supérieure de la caisse. L'ensemble des configurations identifiées par la Surface Navigable est un ensemble fermé, c'est à dire que les limites de cet ensemble font également partie intégrante de l'ensemble. Il faut également tenir compte des obstructions qu'un élément va créer vis-à-vis de lui-même. Un objet, tel que le canapé de la figure 2.5, va pouvoir présenter des parties verticales ou inclinées comme le dossier du canapé. Ces éléments de la géométrie vont donc gêner la navigation de l'agent. Afin

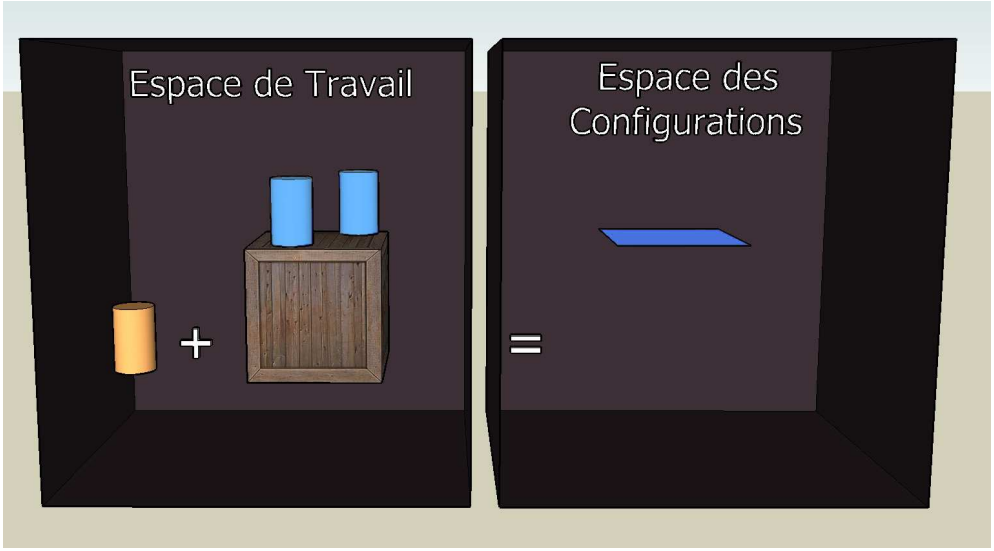


Figure 2.4 – Construction de la Surface Navigable associée à une caisse. La Surface Navigable extraite représente l'ensemble des configurations où l'agent peut se tenir et naviguer.

d'en tenir compte, les configurations des Surfaces Navigables en collision avec l'objet sont donc supprimés de la surface. Les Surfaces Navigables du canapé présentées dans la figure 2.5 prennent en compte ces configurations obstruées et on peut voir que la Surface Navigable sur les sièges est réduite. L'ensemble des configurations contenues dans une Surface Navigable définit le sous-ensemble de C-free, *C-Navigable*.

Les Volumes Interdits : représentation des obstructions

Quel que soit l'objet considéré, sa présence dans l'environnement va créer une gêne à la navigation, une zone d'obstruction du point de vue de l'agent. Afin d'identifier un chemin libre de toute collision, la caractérisation de ces obstructions est essentielle. Nous définissons un sous-espace de E_c appelé *C-obstacle* contenant l'ensemble des configurations $c(x, y, z)$ en interaction avec l'objet O_i considéré. Les configurations identifiées dans E_c vont correspondre dans l'espace de travail E_t aux positions $p(x, y, z)$ où le cylindre C_k représentant l'agent est en collision avec l'objet O_i et où la navigation lui est donc interdite. Ce sous-ensemble de l'espace des configurations définit le Volume Interdit, noté $V_f(O_i, M_k)$, associé à l'objet considéré.

Tout comme pour la Surface Navigable, la définition du Volume Interdit se base sur la géométrie de l'objet. Le Volume Interdit identifie ainsi les configurations dans E_c correspondant à ses positions en collision dans E_t en se basant sur le maillage. D'après la géométrie de l'objet, le Volume Interdit associé à un objet est construit de la manière suivante :

1. L'ensemble de l'élément géométrique est tout d'abord extrudé selon l'axe vertical d'une distance égale à la hauteur du cylindre C_k représentant l'agent. Cette

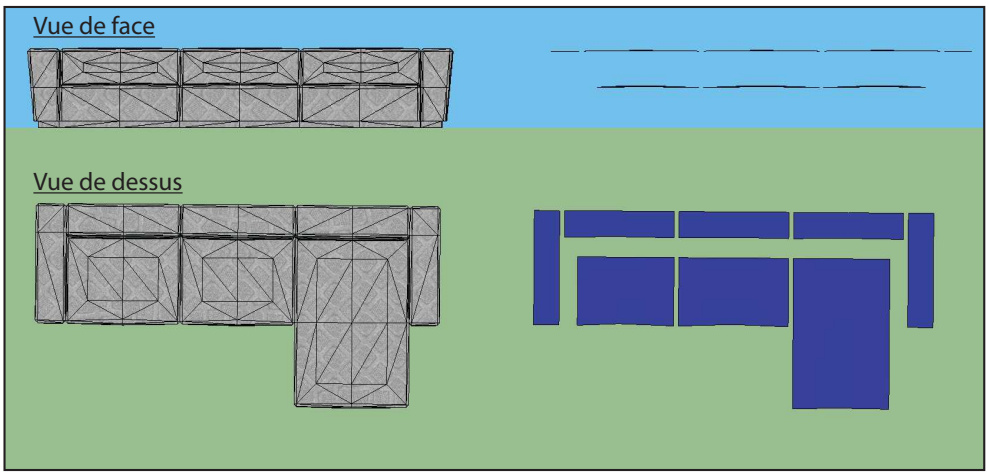


Figure 2.5 – Représentation de la Surface Navigable associée à un canapé. La partie supérieure de l'image montre que l'inclinaison des Surfaces Navigables respecte le critère des pentes navigables tandis que la partie inférieure montre que les configurations trop proche du dossier, et donc où l'agent serait en collision avec l'objet, sont retirées ultérieurement du Volume d'Interaction.

extrusion correspond aux configurations de E_c où l'agent entre en collision avec l'objet dans E_t en essayant de passer en-dessous de celui-ci.

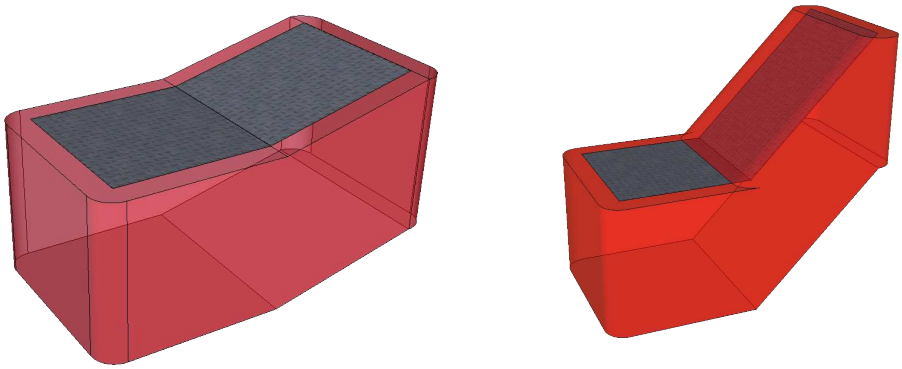
2. (a) Les triangles navigables identifiés précédemment sont récupérés. En supprimant ces triangles de la géométrie de l'objet, nous obtenons l'ensemble des parties non-navigables de cet objet. Cette géométrie non navigable va jouer un rôle d'obstacle pendant la navigation de l'agent.

(b) Afin de représenter les obstructions dues à la géométrie non navigable identifiée en **2.***(a)*, cette géométrie est dilatée dans le plan horizontal d'une distance R. Cette distance correspond au rayon du cylindre C_k. L'extrusion du volume de cette largeur permet de caractériser l'ensemble des positions où l'agent est en collision avec les bords de l'objet dans le plan horizontal.

3. Le Volume Interdit complet correspond à l'union des volumes définis en **1.** et **2.**.

L'identification des triangles navigables, effectuée en **2.***(a)* nous permet d'éviter de considérer une pente présente sur un objet comme un obstacle à la navigation. La figure 2.6 présente les différences de construction dans le cas d'une pente navigable ou non. Dans le schéma 2.6a, les deux plans sont navigables et seules les bordures de l'objet sont extrudées vers l'extérieur, afin d'empêcher un agent d'entrer en collision avec l'objet. Dans le schéma 2.6b, la surface de droite n'est pas navigable pour l'agent et joue donc un rôle assimilable à celui d'un mur. La géométrie de cette surface est donc extrudée pendant l'étape **2.***(b)*, afin d'empêcher l'agent d'entrer en collision avec le "mur" lors de sa navigation.

Contrairement aux Surfaces Navigables, l'ensemble des configurations identifié par le Volume Interdit est un ensemble ouvert, les configurations comprises aux limites



(a) La surface inclinée est navigable et n'est donc pas considérée comme un obstacle

(b) La surface inclinée n'est ici pas navigable et est donc considérée comme un obstacle

Figure 2.6 – En fonction de l'inclinaison des surfaces d'un objet, celles-ci peuvent être navigable (Fig. 2.6a) ou se comporter comme des murs, obstruant le passage (Fig. 2.6b).

de cet ensemble n'en faisant pas partie. Puisque la construction de la Surface Navigable et du Volume Interdit se basent sur la même géométrie de l'objet, la définition de deux ensembles fermés impliquerait que les configurations de la Surface Navigable soient incluses par construction dans le Volume Interdit. Cette propriété ferait que les configurations de la Surface Navigable seraient également interdites à la navigation. La définition d'un ensemble de configuration ouvert pour le Volume Interdit et fermé pour la Surface Navigable permet ainsi que les configurations de la Surface Navigable ne soient donc pas comprises dans le Volume Interdit de l'objet.

La figure 2.7 montre un exemple de Volumes Interdits. Des positions du cylindre représentant l'agent sont montrées dans l'espace de travail E_t. Ces positions sont en collision avec l'objet et il est donc impossible pour l'agent d'y accéder au cours de sa navigation. Le Volume Interdit associé à cette caisse est ensuite représenté dans l'espace des configurations E_c. Il contient l'ensemble des configurations qui sont à l'intérieur de l'objet, ainsi que celles qui sont en dessous ou trop près des bords de celui-ci. Un second exemple est également montré pour un canapé possédant une géométrie plus complexe (voir Fig. 2.8). Nous voyons en haut de la figure que le volume est extrait sur la hauteur afin d'interdire le passage à l'agent en-dessous de l'objet avec le mouvement considéré. La seconde vue est une vue de 3/4 sur laquelle nous observons que le volume de l'objet a été augmenté dans le plan horizontal du rayon R de l'agent afin de prévenir toute collision entre celui-ci et les bords de l'objet.

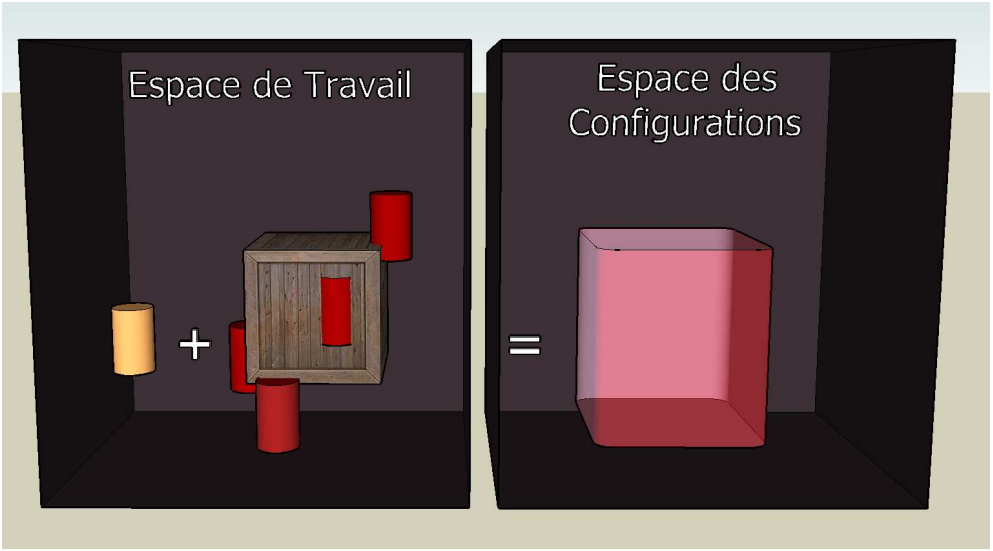


Figure 2.7 – Construction du Volume Interdit associé à une caisse.

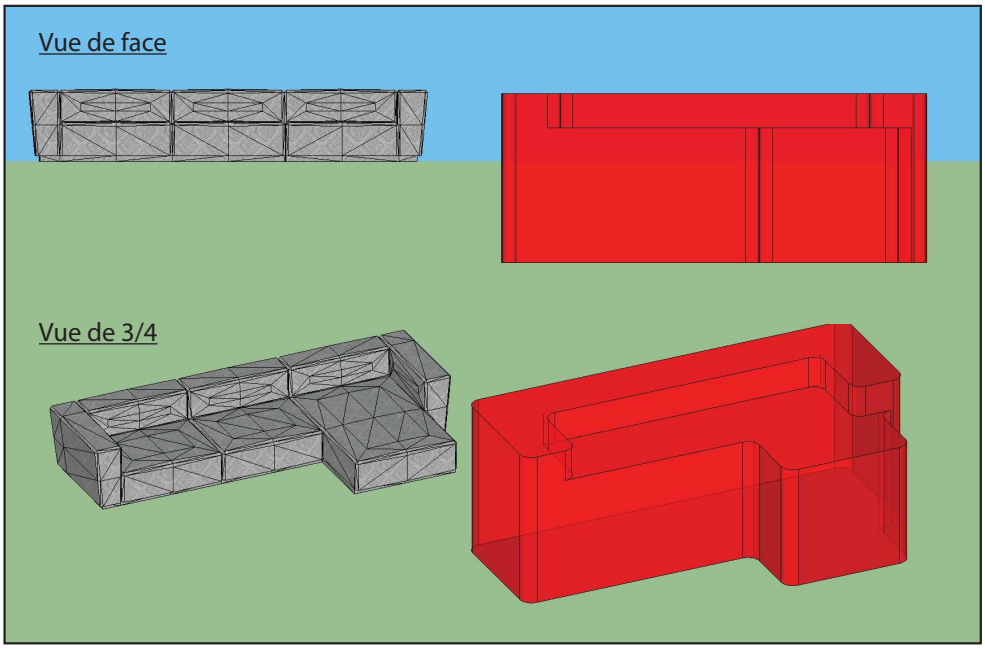


Figure 2.8 – Représentation du Volume Interdit associé à un canapé.

Les Volumes d'Accessibilité : augmentation de l'espace de navigation

La notion d'accessibilité employée dans le contexte d'environnements avec des surfaces déconnectées va dénoter la possibilité pour un agent d'accéder à une surface à partir d'une zone de navigation appartenant à un autre élément. Cette notion repose sur les capacités de déplacement de l'agent afin de vérifier s'il est capable de passer d'une zone à l'autre. Cette définition est à opposer à la notion d'accessibilité en environnement connexe, qui dénote la capacité de l'agent à pouvoir atteindre une cible en évitant des obstacles. En d'autres termes, l'accessibilité entre deux composantes correspond ici à la capacité que possède un agent à naviguer entre des zones déconnectées de l'environnement. Peu de méthodes se sont pour le moment intéressées à caractériser ces accès entre des surfaces déconnectées au sein d'environnements navigables [CLS03, Li04, SH07, LLKP11]. Cette prise en compte de l'accessibilité entre objets est au cœur de notre représentation. Elle permet d'augmenter l'espace de navigation à disposition de l'agent en prenant en compte toutes les possibilités de navigation apportées par la présence d'éléments navigables au sein de cet environnement.

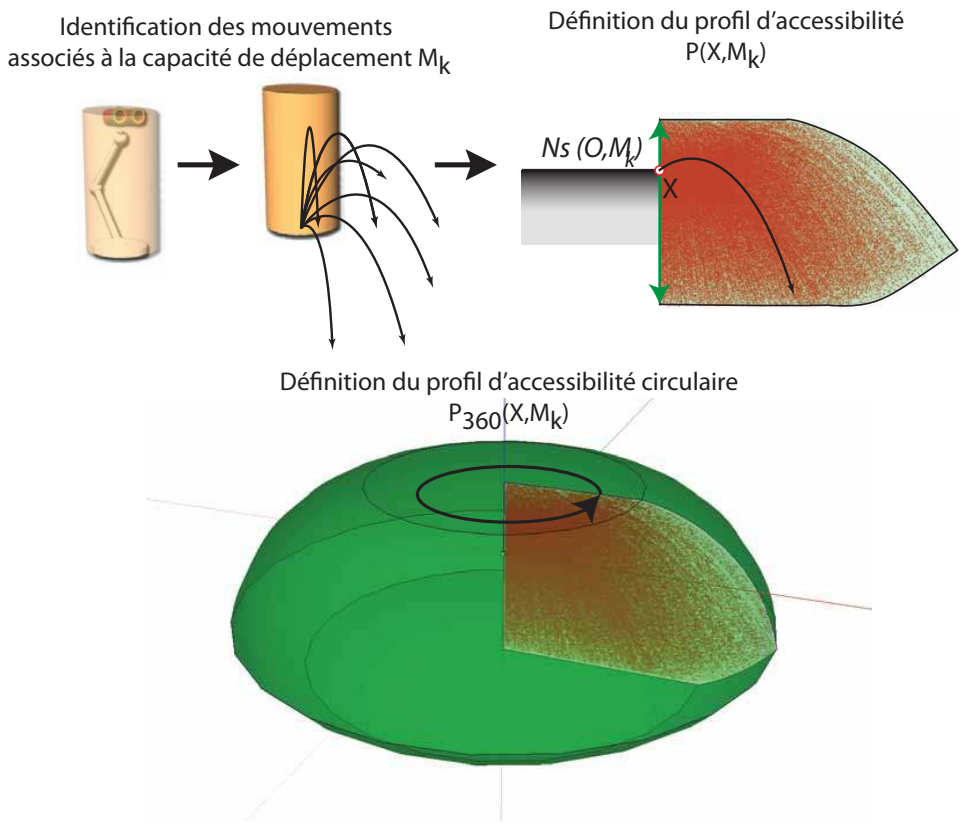


Figure 2.9 – Définition du profil d'accessibilité. En échantillonnant les trajectoires associées à la capacité de déplacement M_k, nous définissons le profil d'accessibilité en calculant l'enveloppe convexe des trajectoires échantillonnées. Le profil circulaire est ensuite extrait à partir de ce profil.

Afin de représenter cette accessibilité depuis un objet vers d'autres configurations de l'environnement, nous définissons les Volumes d'Accessibilité. Ces volumes représentent de manière explicite l'ensemble des configurations de E_c accessibles à l'agent depuis un objet géométrique de E_t. Le Volume d'Accessibilité associé à la capacité de déplacement M_k et à un objet O_i est annoté $V_a(O_i, M_k)$. Les mouvements M_k auxquels sont associés des Volumes d'Accessibilité sont les mouvements tels que les sauts qui permettent de se détacher du sol afin d'accéder à des configurations déconnectées de l'objet. La figure 2.9 présente différentes étapes de la construction d'un Volume d'Accessibilité. Comme nous avons pu le voir lors de la caractérisation de l'agent, les mouvements de saut sont assimilés à des trajectoires paraboliques et nous leur associons des vitesses maximales de décollage et d'atterrissage, appelées respectivement v_{dec} et v_{att}. L'agent suivant une trajectoire balistique lors de son saut, cette trajectoire va rester par définition dans un plan vertical. En considérant que l'agent se tient à une configuration X dans un environnement, nous déterminons ainsi l'ensemble des configurations que l'agent peut atteindre dans E_c à l'aide d'une capacité de saut M_k en respectant les contraintes apportées par v_{dec} et v_{att}. L'agent est donc capable de sauter sur place, vers l'avant mais également vers le bas afin d'atteindre une position moins élevée. Nous construisons donc un profil d'accessibilité, noté $P(X, M_k)$, représentant l'ensemble des configurations atteignables (dans un plan vertical) par l'agent depuis une configuration X de départ. Ce profil, comme le montre la figure 2.9, est construit par échantillonnage de trajectoires de saut. L'enveloppe convexe des trajectoires échantillonnées est ensuite récupérée pour définir le profil d'accessibilité $P(X, M_k)$. Toutefois, ce profil représente les configurations accessibles selon une seule direction. Afin de caractériser les configurations accessibles quel que soit l'orientation de l'agent, un profil circulaire, noté $P_{360}(X, M_k)$, est défini en effectuant une révolution du profil d'accessibilité $P(X, M_k)$ autour de l'axe vertical. Ce profil circulaire caractérise ainsi l'ensemble des configurations auxquelles l'agent peut accéder à partir d'une configuration X en utilisant la capacité de saut M_k.

L'ensemble des configurations où peut se placer un agent sur un objet O_i a précédemment été identifié par la Surface Navigable $Ns(O_i, M_k)$ associée à cet objet. L'identification de l'ensemble des positions accessibles depuis l'objet O_i revient donc à calculer le profil circulaire $P_{360}(X, M_k)$ pour chaque configuration X où l'agent va pouvoir se tenir sur O_i, en d'autres termes, pour chaque configuration X appartenant à la Surface Navigable $Ns(O_i, M_k)$. Toutefois cette construction doit également tenir compte des configurations obstruées par l'objet lui-même. Les configurations appartenant au Volume Interdit de l'objet sont donc supprimées du Volume d'Accessibilité. Le Volume d'Accessibilité associé à un objet O_i est donc défini par :

$$V_a(O_i, M_k) = \left(\bigcup_{X \in Ns(O_i, M_k)} P_{360}(X, M_k) \right) - V_f(O_i, M_k)$$

La construction des Volumes d'Accessibilité est finalement illustrée sur deux exemples. Le premier cas d'exemple, présenté dans la figure 2.10, est une simple caisse dont la Surface Navigable est la face supérieure. Les configurations représentées en vert

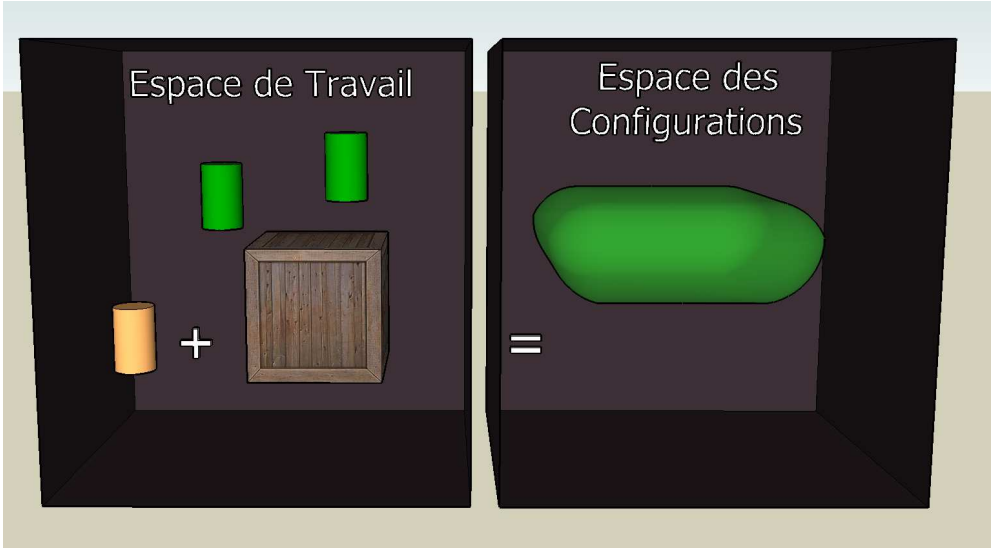


Figure 2.10 – Construction du Volume d'Accessibilité pour une caisse. Le volume construit dans l'espace des configurations représente l'ensemble des configurations accessibles par l'agent en utilisant la capacité de mouvement associé à son cylindre englobant.

dans l'espace de travail E_t identifient les positions accessibles en utilisant la capacité de déplacement M_k. L'identification de toutes les configurations permet la définition du Volume d'Accessibilité dans l'espace des configurations E_c associé. Le second exemple présente un canapé possédant plusieurs Surfaces Navigables déconnectées. Le Volume d'Accessibilité est donc l'union des Volumes d'Accessibilité associés à chacune de ces surfaces comme le montre la figure 2.11. La définition de ce Volume d'Accessibilité permet d'ailleurs d'identifier directement un accès existant entre les Surfaces Navigables du canapé et donc la possibilité de naviguer entre le dossier et l'assise du canapé.

La définition de ce Volume d'Accessibilité permet d'avoir une représentation 3D concrète des accessibilités offertes par un élément de l'environnement. Ces Volumes d'Accessibilité permettent de définir un nouveau sous-espace dans l'espace des configurations. Ce sous-espace, appelé C-accessible, est un sous-ensemble de C-free. Nous avons proposé de représenter les trajectoires de saut de l'agent à l'aide d'une parabole et de paramètres de vitesses au décollage, v_{dec}, et à l'atterrissage, v_{att}. Ces conditions ont été posées afin de pouvoir limiter les sauts de l'agent dans un intervalle de vitesses de sécurité. Afin de gérer de nouvelles transitions entre les éléments de l'environnement, il est possible de définir de nouvelles capacités de déplacement à associer à l'agent.

Familles de Volumes d'Interaction

Comme nous venons de le voir, trois Volumes d'Interaction vont pouvoir être extraits de la mise en correspondance d'un objet O_i et d'une capacité de déplacement M_k. Ces trois volumes vont définir la *famille de Volumes d'Interaction* associée à la capacité de déplacement M_k. En considérant un agent possédant plusieurs capacités de

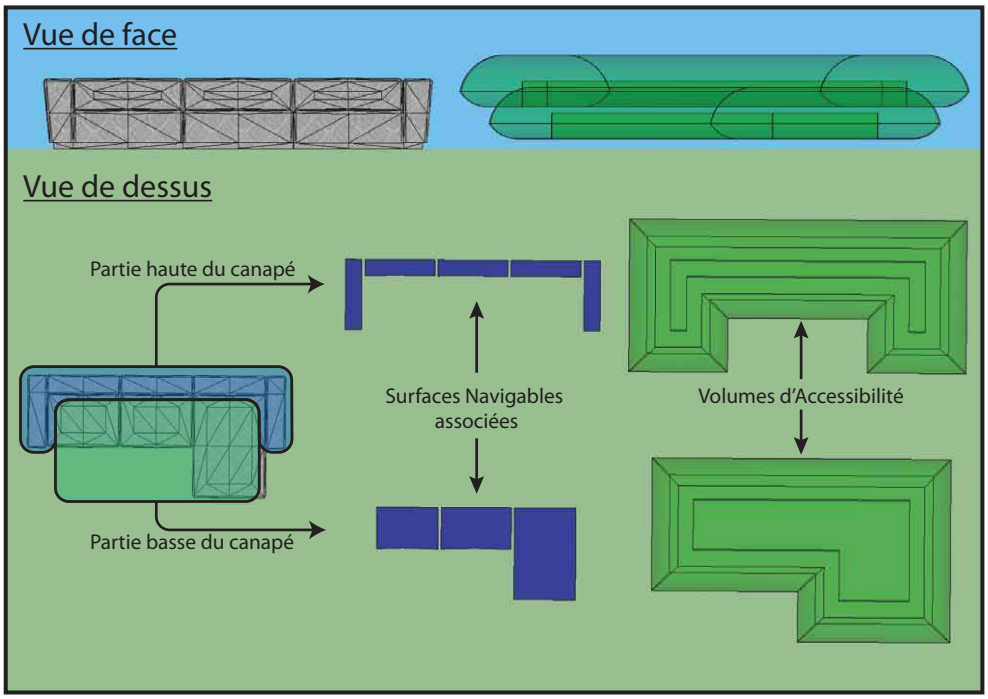


Figure 2.11 – Représentation du Volume d'Accessibilité associé à un canapé. Afin de simplifier la visualisation, la partie basse de l'image décompose le Volume d'Accessibilité en deux parties. La première partie représente la partie du Volume associée à la partie haute du canapé (les appuie-têtes) tandis que la seconde représente celle associée à la partie basse (les coussins). L'union de ces parties définit le Volume d'Accessibilité associé à l'objet (partie supérieure de l'image).

déplacement, une famille de Volumes d'Interaction sera calculée pour chacune de ses capacités. Les familles de Volumes d'Interaction associées aux mouvements contraints au sol sont cependant différentes, puisque dans un mouvement contraint au sol l'agent ne possède pas la capacité de se détacher de la Surface Navigable. Aussi, une famille de Volume d'Interaction rattachée à un mouvement de ce type possède seulement deux Volumes d'Interaction : une Surface Navigable et un Volume Interdit.

2.3 Mouvements des objets et Volumes d'Interaction

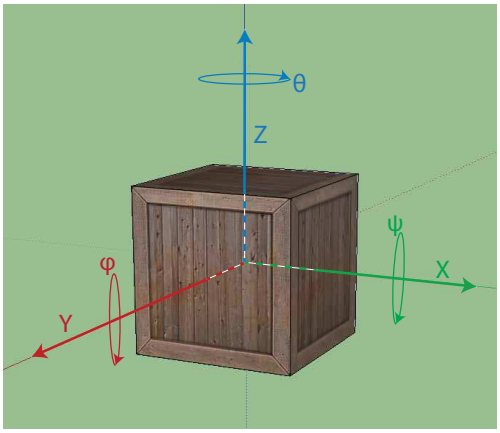


Figure 2.12 – Chaque élément de l'environnement possède 6 degrés de liberté : une translation sur chacun des axes X, Y et Z de l'environnement, et 3 rotations d'angles ψ, ϕ et θ autour de chacun de ces axes.

La prise en compte d'environnements dynamiques amène à considérer des situations où la configuration des éléments évolue au cours du temps. En regardant ces mondes virtuels, les évolutions vont pouvoir être totalement incontrôlées et imprévisibles pour l'agent. Un moteur physique ou un moteur d'animation peut par exemple faire tomber des objets dans l'environnement, un pont fait de cordes et de planches de bois peut également bouger sous l'action de forces extérieures ou une plateforme peut effectuer des aller-retour entre différentes zones. La représentation des objets dynamiques à l'aide des Volumes d'Interaction nous permet d'identifier l'impact de ces éléments sous forme d'obstacles et de surfaces de navigation. Toutefois, afin de gérer la dynamique de ces éléments, la définition des Volumes d'Interaction va parfois devoir être remise à jour.

La définition des Volumes d'Interaction s'appuie sur l'hypothèse implicite que la géométrie de l'objet n'est pas amenée à changer au cours de la simulation, ce qui est le cas généralement avec des éléments indéformables. Toutefois, ces objets sont susceptibles de se déplacer ou d'être déplacés dans l'environnement. Leur orientation peut donc changer, remettant en cause les Volumes d'Interaction précédemment définis. Prenons pour illustrer ce fait l'exemple d'une caisse posée au sol. Comme présenté précédemment, trois Volumes d'Interactions seront associés à cette caisse : un Volume Interdit, une Surface Navigable et un Volume d'Accessibilité. En considérant un environnement

dynamique, le positionnement de cette caisse va pouvoir être modifié selon 6 degrés de libertés différents illustrés dans la figure 2.12 : trois translations (selon X, Y et Z) et trois rotations (ψ selon X, ϕ selon Y et θ selon Z). Ces modifications vont être gérées de trois manières différentes :

1. Les trois translations vont pouvoir être effectuées selon les 3 axes X, Y et Z de l'environnement. Ces translations vont seulement modifier la position de l'objet : la définition des Surfaces Navigables ainsi que des Volumes Interdits et d'Accessibilité associés à l'objet n'est donc pas remises en cause par de telles transitions. Maintenir à jour les Volumes d'Interaction associés à l'objet lors d'une translation va donc consister à effectuer la même transformation des Volumes d'Interaction dans l'espace des configurations.

2. Les rotations d'angle θ autour de l'axe vertical vont modifier l'orientation de l'objet, mais ne vont pas modifier l'inclinaison des surfaces. Puisque cette inclinaison reste identique, les Surfaces Navigable et les Volumes d'Accessibilité demeurent inchangés. De même, l'extraction du Volume Interdit étant faite selon l'axe vertical, ce volume demeure également valide puisque l'axe vertical de l'objet n'est pas modifié. La mise à jour des Volumes d'Interaction pour un objet subissant une rotation θ consistera simplement à appliquer la même transformation à l'objet, et aux Volumes d'Interaction qui lui sont associés.

3. Les rotations ψ et ϕ vont par contre totalement modifier la configuration de l'objet. En effet, ces rotations vont directement changer l'axe vertical des éléments et par conséquent agir sur les normales des Surfaces Navigables des objets et sur l'extraction du Volume Interdit. La figure 2.13 montre l'exemple d'une caisse basculant sur le côté. La figure de gauche montre la caisse et les Volumes d'Interaction qui lui sont associés à l'origine. La seconde figure montre la caisse effectuant une rotation ψ ou ϕ de 45° puis la rotation de 90° est montrée sur la dernière figure. On peut ici s'apercevoir que les Volumes d'Interaction ne représentent plus l'impact réel de l'objet sur son environnement. Ils fournissent de mauvaises représentations, tant au niveau des accessibilités que des obstructions.

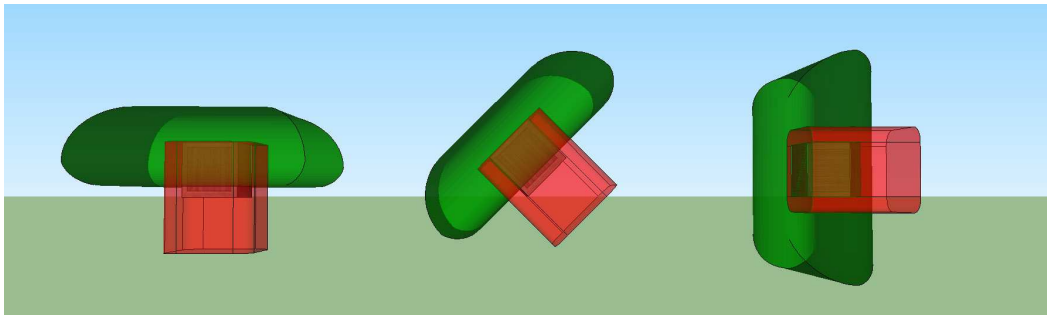


Figure 2.13 – Exemple de rotation d'un objet autour de l'axe X ou Y, ici une caisse, sans remise à jour des Volumes d'Interaction. On peut observer que tous les Volumes sont faussés à cause de la rotation de l'élément.

Dans le cas des rotations ψ et ϕ, selon les axes X et Y, nous recalculons donc les différents Volumes d'Interaction afin de conserver une représentation valide des éléments. Toutefois, afin d'éviter d'engendrer des coûts de calculs trop importants par la création / suppression d'éléments à chaque pas de temps, nous proposons plusieurs optimisations concernant la mise à jour de ces Volumes d'Interaction. Ces optimisations se basent sur les observations suivantes :

- Dans le cas d'une simulation dynamique, un objet en rotation va tourner d'un angle infime entre deux pas de simulation.
- D'autre part, un objet en rotation va rarement être navigable pour un agent virtuel pendant sa rotation. Un tel élément va en effet présenter des espaces navigables quand il sera posé au sol et stable. Mais, dès sa mise en rotation, ces zones navigables seront invalidées. Il devient alors un obstacle et non une aide à la navigation pour l'agent.

En se basant sur ces deux observations, nous commençons par associer un paramètre δ et une valeur d'angle seuil Δ à chaque objet. Le paramètre δ représente l'angle entre l'axe vertical associé à la position actuelle de l'élément, et l'axe vertical du monde au moment où ces volumes ont été calculés. La valeur de δ est mise à jour à chaque pas de simulation. La valeur seuil Δ est utilisée pour déterminer si les Volumes d'Interaction doivent être ou non recalculés. Deux cas se présentent :

- $\delta < \Delta$: la rotation actuelle est inférieure au seuil. Elle n'est donc pas suffisante pour provoquer le calcul de nouveaux Volumes d'Interaction. Ces volumes sont donc simplement mis à jour en leur appliquant la rotation δ.
- $\delta \geq \Delta$: l'inclinaison de l'objet est importante et la représentation créée par les Volumes d'Interaction est donc éloignée de la réalité. Les Volumes d'Interaction sont donc invalidés.

Lorsque les volumes sont invalidés par une rotation, ils sont supprimés et doivent être recalculés. Deux cas sont alors possibles :

1. L'objet est trop dynamique pour que l'agent puisse y naviguer.

2. L'objet a une position stable dans l'environnement et l'agent peut de nouveau y naviguer.

Dans le cas **1.** , l'objet va principalement être considéré comme un obstacle pour l'agent puisque ses rotations ne vont pas permettre à un agent d'y naviguer. Dans ce cas, nous associons à l'objet seulement un Volume Interdit. Nous construisons ce Volume Interdit en récupérant la géométrie de l'objet et en considérant l'ensemble des surfaces comme non-navigables. Nous réalisons ensuite l'extraction du volume à partir de cette géométrie comme nous l'avons précédemment défini. Dans le cas **2.**, l'objet a atteint une position stable et la géométrie de l'objet est de nouveau parcourue afin d'extraire de nouveaux Volumes d'Interaction capturant la nouvelle topologie de l'élément. L'utilisation de ces optimisations est montrée dans la figure 2.14. La caisse est tout d'abord statique dans la position gauche, puis sa rotation est amorcée, la mettant en position instable où son Volume Interdit est recalculé. Finalement, la rotation se termine et les nouveaux Volumes d'Interaction sont recalculés, menant à une représentation quasiment similaire à la situation originale, à cause des symétries dans la géométrie de

l'objet de base.

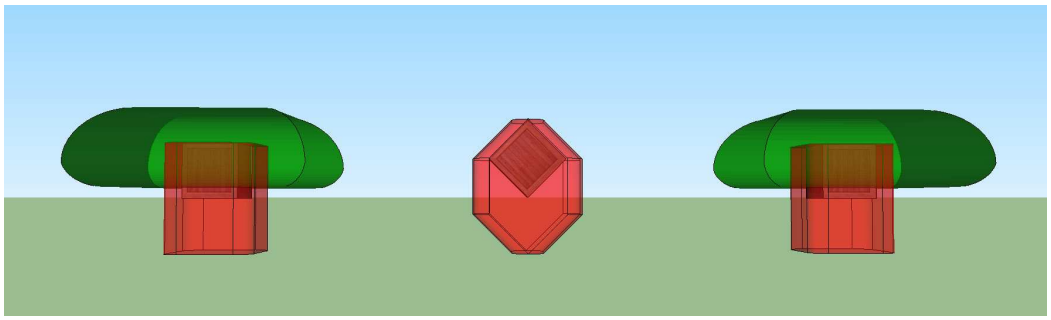


Figure 2.14 – Les Volumes d'Interaction sont remis à jour lors de la rotation de l'élément. Pendant son mouvement (figure centrale) l'élément est représenté simplement par un Volume Interdit. Lorsque l'objet se stabilise dans l'environnement, l'ensemble des Volumes d'Interaction sont de nouveaux recalculés à la volée.

Ces différentes optimisations permettent d'améliorer les performances de mise à jour des Volumes d'Interaction dans des mondes dynamiques. L'introduction du seuil Δ permet d'éviter de recalculer les Volumes d'Interaction à chaque pas de temps lors d'une rotation. Toutefois, l'utilisation de ce paramètre mène à une approximation des Volumes d'Interaction réels. Il faut donc choisir entre une valeur de Δ faible entraînant des mises à jour plus nombreuses, ou une valeur plus élevée amenant de plus grandes approximations des Volumes pendant la simulation. La représentation d'un objet en mouvement sous la forme d'un simple Volume Interdit est aussi une approximation. Elle est toutefois introduite de manière seulement temporaire afin d'assurer à l'agent une navigation en toute sécurité.

3 Conclusion

Nous avons tout d'abord défini dans cette partie notre représentation de l'agent. Cet agent peut posséder différentes capacités de déplacement qui sont classées en deux catégories : les déplacements contraints au sol, tels que la marche, et les déplacements permettant de se déconnecter du sol tels que les sauts. Des cylindres englobants sont ensuite associés à chaque capacité de navigation afin de borner les mouvements de l'agent. Cette représentation nous permet de réduire la complexité du problème de planification en ramenant ce problème à la planification de chemin d'un cylindre à trois degrés de liberté quelles que soient la nature et la complexité géométrique de l'agent original. La représentation de l'agent dans un espace tridimensionnel nous permet de plus d'avoir accès à l'ensemble de l'algorithmique 3D pour notre méthode. De plus, cette représentation nous permet de décorréler la planification de chemin de l'animation à jouer lors du déplacement de l'agent.

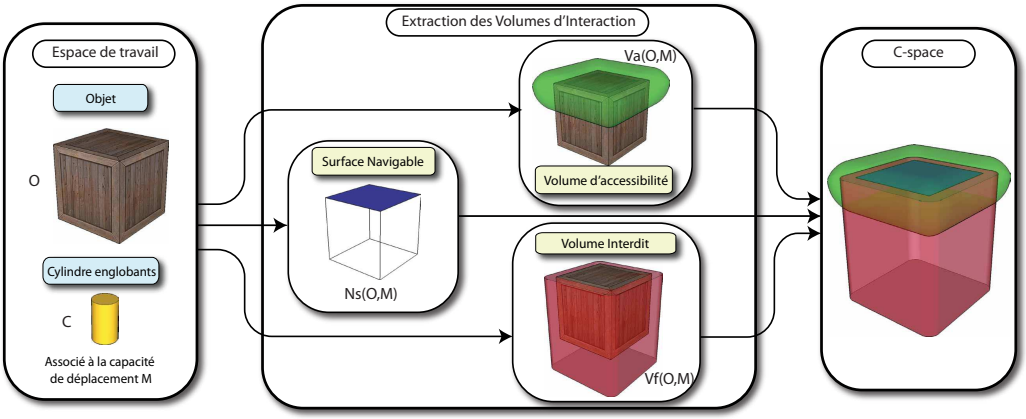


Figure 2.15 – Trois Volumes d'Interaction sont définis à partir de la géométrie d'un objet et d'une capacité de déplacement. Ces trois Volumes d'Interaction définissent des sous-ensembles de l'espace des configurations.

Nous avons défini, dans un second temps, une représentation duale des éléments géométriques composant l'environnement : les Volumes d'Interaction. Comme le résume la figure 2.15, ces Volumes d'Interaction sont une représentation explicite de l'impact des objets géométriques de l'environnement de travail E_t dans l'espace des configurations E_c. Cet impact topologique d'un objet sur son environnement dépend en grande partie de l'agent navigant, aussi la forme de ces Volumes d'Interaction est calculée en considérant les capacités de déplacement de cet agent, ainsi que son occupation spatiale. En considérant une capacité de déplacement de l'agent M_k et un élément de l'environnement O_i, nous définissons donc trois Volumes d'Interaction :

- Une Surface Navigable, notée $Ns(O_i, M_k)$, permettant de définir les zones de l'objet où l'agent est capable de naviguer et définissant un sous-ensemble de l'espace des configurations appelé *C-navigable*.
- Un Volume Interdit, noté $V_f(O_i, M_k)$, permettant de caractériser l'objet en tant qu'obstacle et de détecter les zones de l'environnement qu'il obstrue. Ce volume caractérise le sous ensemble de l'espace des configurations *C-obstacle*.
- Un Volume d'Accessibilité, noté $V_a(O_i, M_k)$, permettant de représenter les positions déconnectées mais accessibles par l'agent depuis la Surface Navigable. Ce volume délimite le sous-ensemble de l'espace des configurations *C-accessible*.

L'ensemble des Volumes d'Interaction associé à un couple objet - capacité de déplacement, autrement dit à un couple (O_i, M_k), définit une famille de volumes. Un agent possédant N capacités de navigation différentes possèdera donc N familles de volumes afin de représenter l'impact topologique d'un objet O_i dans sa globalité. Toutefois, dans le cas de capacités de déplacement contraintes au sol, telles que la marche, nous ne définissons pas de Volume d'Accessibilité puisque l'agent ne peut se détacher de la Surface Navigable. De plus, certains objets peuvent par exemple être trop fragiles ou trop dangereux pour permettre à l'agent d'y naviguer. Cette propriété est facilement intégrée dans le modèle en ne considérant pas l'aspect navigable de l'objet, et en le représentant seulement à l'aide d'un Volume Interdit.

Cette représentation locale de l'influence des objets sur leur environnement à l'aide des Volumes d'Interaction permet dans un premier temps de s'abstraire de la représentation globale de l'environnement. En rattachant les Volumes d'Interaction de manière individuelle aux objets de l'environnement, nous proposons un lien direct entre chaque objet géométrique O_i de l'espace de travail E_t, et sa représentation duale dans l'espace des configurations E_c. Cette mise en correspondance permet de traduire rapidement les mises à jour de l'espace de travail vers l'espace des configurations.

Ces Volumes d'Interaction peuvent être précalculés lorsque la géométrie des objets est déterminée à l'avance. Toutefois, dans des environnements présentant une grande dynamique, tels que des mondes soumis à un moteur physique, certains objets vont subir de grandes modifications. Nous avons donc proposé dans ce chapitre une méthode de mise à jour dynamique de ces Volumes d'Interaction. Cette mise à jour permet de conserver tout au long de la simulation une représentation valide de l'espace des configurations. De plus, cette mise à jour est suffisamment performante pour conserver une représentation de l'environnement compatible avec des applications interactives, telles que des environnements physiques ou des jeux vidéo.

La suite de la discussion va maintenant s'intéresser à considérer ces différents objets et leur représentations associées de manière simultanée afin de caractériser les relations topologiques dans l'environnement global. Cette prise en compte des différents éléments va permettre de localiser précisément, dans l'espace des configurations, les accès et les obstructions créés dans l'environnement et influant sur la navigation de l'agent.

Représentation dynamique de la topologie

3

Table des matières

1 Caractérisation des relations entre deux objets de l'environnement . 64
1.1 Caractérisation des accessibilités et obstructions entre deux objets . . 64
1.2 Détection des interactions entre objets 67
2 Représentation de la topologie globale et de son évolution au cours du temps 69
2.1 Le Graphe Topologique 69
2.2 Représentation de la topologie et de son évolution au cours du temps 75
2.3 Suivi des modifications de Volumes d'Interaction 80
3 Conclusion 81

La planification de chemin en environnement dynamique est une tâche rendue difficile par l'évolution constante de la topologie de l'environnement. Une représentation et un suivi constant de la topologie globale sont donc nécessaires afin de proposer des solutions de navigation valides à un agent. Pour cela, nous devons identifier les relations topologiques d'obstruction et d'accessibilité existant dans l'environnement et suivre l'évolution et les modifications de ces relations au fil du temps. En nous basant sur les Volumes d'Interaction définis dans le chapitre précédent, nous sommes capables de définir l'impact local d'un objet sur la topologie en identifiant les configurations obstruées, navigables et accessibles. En observant les interactions entre ces volumes, nous pouvons donc caractériser les relations d'accessibilité et d'obstruction entre les éléments associés. En se basant sur ces observations et leur évolution au fil de la simulation nous construisons donc une représentation dynamique de la topologie globale de l'environnement. En ajoutant à cette représentation des informations temporelles, nous proposons d'identifier et de caractériser, sans connaissance a priori, les connexions dynamiques existantes dans l'environnement. Ces informations temporelles nous permettent notamment d'identifier des accessibilités entre des éléments créées de manière périodique. Une plateforme mobile va par exemple pouvoir se déplacer périodiquement et permettre l'accès à certains moments à un premier élément de l'environnement et à d'autres moments à un second élément. En plus d'identifier cette connexion périodique, l'utilisation d'informations temporelles permet de détecter l'accès entre les deux éléments à l'aide de la plateforme mobile bien qu'aucune liaison physique n'existe simultanément entre les deux éléments. Sans injecter de connaissance à priori sur l'évolution

de l'environnement, l'agent va ainsi être capable de détecter ces liaisons temporelles et d'utiliser les mouvements propres des éléments dynamiques de l'environnement afin de pouvoir naviguer dans l'environnement. Nous allons voir au cours de ce chapitre comment l'utilisation des Volumes d'Interaction nous permet de caractériser les interactions existantes entre deux objets. Dans un second temps, nous verrons comment cette caractérisation est utilisée pour construire une représentation dynamique et temporelle de la topologie de l'environnement.

1 Caractérisation des relations entre deux objets de l'environnement

Nous allons nous focaliser dans cette section sur la définition des relations topologiques liant deux éléments de l'environnement. Les Volumes d'Interaction, définis dans le chapitre précédent, nous permettent de définir ces relations topologiques entre les objets en détectant les interactions existant dans l'espace des configurations entre leurs volumes associés. Dans cette partie, nous aborderons dans un premier temps la caractérisation d'une accessibilité et d'une obstruction entre deux éléments de l'environnement. Puis nous verrons comment détecter ces relations dans l'espace des configurations pendant le déroulement de la simulation.

1.1 Caractérisation des accessibilités et obstructions entre deux objets

La notion d'obstruction ou d'accessibilité au sein d'un environnement va dépendre de l'agent considéré. Les capacités de déplacement d'un agent, ainsi que son occupation spatiale, vont en effet avoir une influence sur les accessibilités qu'il peut se créer, ou sur les éléments qui vont se comporter en obstacle de son point de vue. La caractérisation des relations topologiques va donc reposer sur ces capacités de déplacement afin de représenter au mieux les possibilités offertes à l'agent.

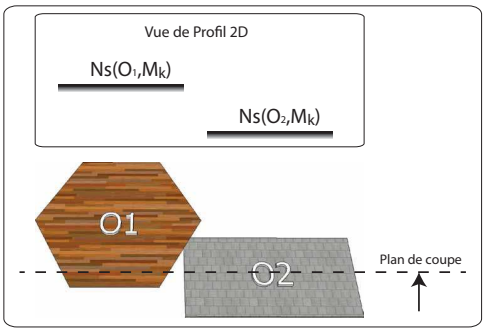
Caractérisation d'une accessibilité

Afin de se déplacer l'agent peut être amené à naviguer sur différents éléments. Comme le montre la figure 3.1a, ces éléments vont généralement être déconnectés et l'agent sera capable d'y naviguer si une relation d'accessibilité est identifiée. Comme nous l'avons vu dans le chapitre 2, les configurations accessibles depuis la Surface Navigable d'un objet sont représentées par les Volumes d'Accessibilité. Considérant une capacité M_k ainsi que deux objets distincts O_i et O_j, une relation d'accessibilité depuis O_i vers O_j est notée $A(O_i, O_j, M_k)$. Cette relation est définie par :

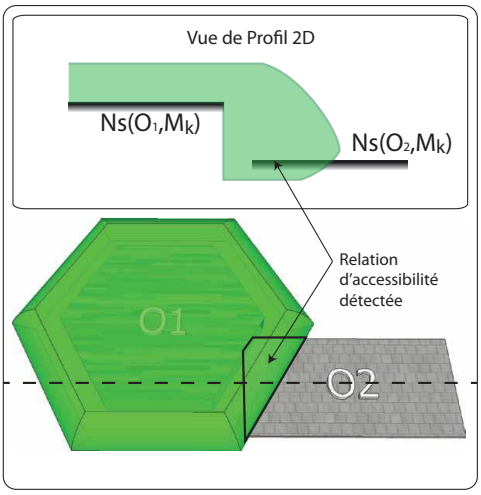
$$A(O_i, O_j, M_k) \Leftrightarrow V_a(O_i, M_k) \cap Ns(O_j, M_k) \neq \emptyset$$

Cette équation définit les configurations accessibles depuis O_i, c'est à dire les configurations qui appartiennent à la fois au Volume d'Accessibilité de O_i et à la Surface Navigable de O_j. Par définition, la capacité de déplacement M_k permet donc à l'agent de trouver une transition depuis O_i vers O_j. Comme montré avec l'accessibilité détectée

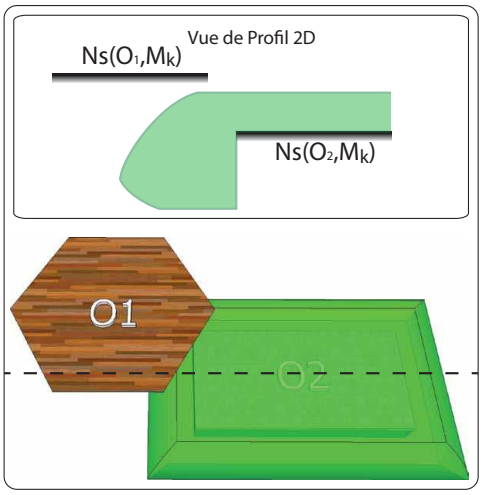
dans la figure 3.1b, cette relation se traduit par une intersection des Volumes d'Interactions dans l'espace des configurations. Dans le cas où il n'y a pas d'intersection entre les volumes, comme le montre la figure 3.1c, l'agent ne possède pas la capacité d'accéder à l'objet en question. D'autre part, comme le montre la figure 3.1, l'accès d'un objet O_i à un objet O_j n'est pas bijectif.



(a) Configurations des éléments O_1 à O_2



(b) Accessibilité détectée de O_1 à O_2



(c) Pas d'accessibilité de O_2 à O_1

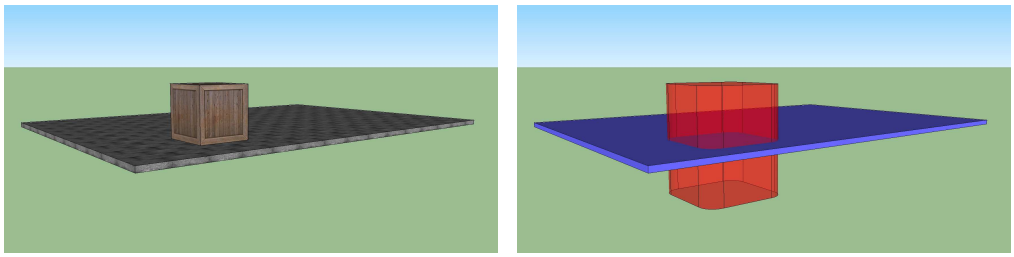
Figure 3.1 – Vues 2D (en haut) et 3D (en bas) des objets O_1 et O_2. 3.1b : Caractérisation d'une accessibilité de O_1 à O_2, la relation $V_a(O_1, M_k) \cap Ns(O_2, M_k) \neq \emptyset$ est vérifiée 3.1c : la relation n'est pas bijective, il n'existe pas d'accessibilité de O_2 à O_1.

Caractérisation d'une obstruction

D'autre part, un objet crée par sa présence différentes obstructions vis-à-vis des autres objets. Au regard des capacités de déplacement utilisées par un agent, une relation d'obstruction entre deux éléments d'un environnement va être différente selon la capacité considérée. La relation d'obstruction d'un objet O_i sur un objet O_j va donc reposer sur la capacité de déplacement M_k utilisée. Elle sera notée $O(O_i, O_j, M_k)$. Deux types d'obstructions vont pouvoir être détectées dans l'espace des configurations :

– obstruction d'un Volume Interdit sur une Surface Navigable

– obstruction d'un Volume Interdit sur un Volume d'Accessibilité



(a) Vue de l'espace de travail composé d'une caisse sur un sol

(b) Représentation dans l'espace des configurations

Figure 3.2 – 3.2a : Un environnement simple est composé d'une caisse posée sur un sol. 3.1b : Dans l'espace des configurations le Volume Interdit de la caisse (rouge) est en interaction avec la Surface Navigable (verte) caractérisant une obstruction.

Dans un premier temps, un élément peut donc bloquer une partie de la Surface Navigable d'un objet O_i. Dans ce cas, les configurations placées directement sous celui-ci deviennent inaccessibles à l'agent naviguant. Cette relation est détectée lorsque le Volume Interdit de O_i va être en interaction avec la Surface Navigable de O_j et sera donc caractérisée par :

$$O(O_i, O_j, M_k) \Leftrightarrow V_f(O_i, M_k) \cap Ns(O_j, M_k) \neq \emptyset$$

L'exemple d'une situation de ce genre est présenté dans la figure 3.2a où un objet O_i est placé au milieu d'une surface d'un second objet O_j. Cette équation traduit le fait que les obstructions créées dans l'espace des configurations sont à la fois contenues dans le Volume Interdit de O_i et dans la Surface Navigable de O_j en considérant la capacité de déplacement M_k.

Une obstruction va pouvoir également bloquer des transitions entre des objets sans pour autant que la navigation ne soit bloquée à la surface de l'un ou de l'autre des objets. Ainsi, un saut d'un élément à un autre peut être obstrué par un plafond trop bas, ou par la présence d'une poutre en hauteur, alors que ces éléments ne gênent en rien la navigation sur chacune des surfaces. Une relation de ce type est détectée lorsque le Volume Interdit d'un objet est en interaction avec le Volume d'Accessibilité d'un autre. Cette relation est donc définie par :

$$O(O_i, O_j, M_k) \Leftrightarrow V_f(O_i, M_k) \cap V_a(O_j, M_k) \neq \emptyset$$

L'ensemble des configurations ainsi désignées est à la fois contenu dans le Volume Interdit de O_i et dans le Volume d'Accessibilité de O_j en considérant la capacité de déplacement M_k. La figure 3.3 présente cette situation en montrant une poutre O_2 placée en hauteur et ne gênant pas la navigation sur l'objet voisin O_1. La relation ainsi caractérisée va dénoter le fait que l'obstruction n'influe pas forcément sur la navigation sur la surface de O_j mais qu'elle empêche certaines transitions caractérisées par le Volume d'Accessibilité de O_j.

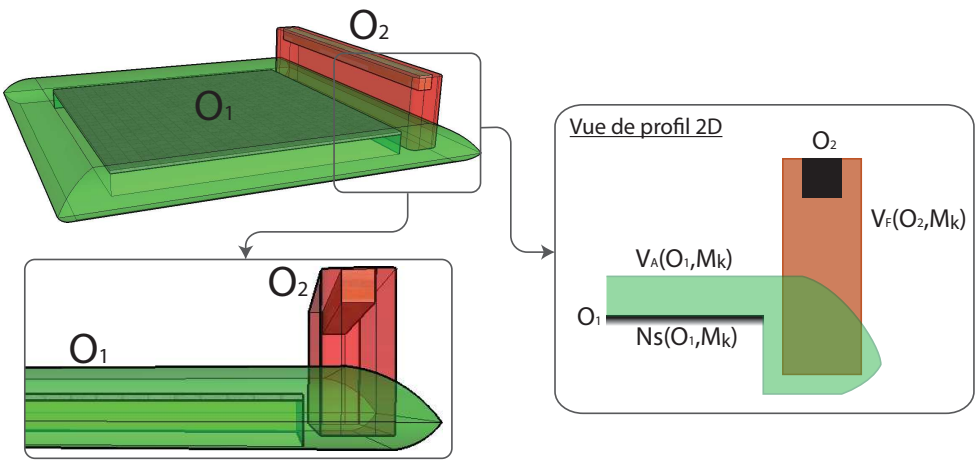


Figure 3.3 – L'objet O_2 est une poutre située en altitude. Son Volume Interdit (rouge) est en interaction avec le Volume d'Accessibilité de O_1 (vert) ce qui interdit de potentiels accès de O_1 vers d'autres éléments de l'environnement même si O_2 n'est pas un obstacle à la navigation sur la Surface navigable de O_1.

La totalité des configurations obstruées va donc être définie par l'équation :

$$O(O_i, O_j, M_k) \Leftrightarrow V_f(O_i, M_k) \cap (\ Ns(O_j, M_k) \cup V_a(O_j, M_k)\ ) \neq \emptyset$$

Cette équation permet de définir l'ensemble des obstructions pouvant intervenir entre deux éléments. Il convient de remarquer également qu'une obstruction sur la Surface Navigable d'un élément implique systématiquement l'obstruction du Volume d'Accessibilité de cet élément par définition des Volumes d'Interaction.

1.2 Détection des interactions entre objets

Pour caractériser une relation topologique, nous devons donc identifier les intersections existantes entre les Volumes d'Interaction. Comme nous l'avons vu au cours du chapitre précédent, l'utilisation de ces volumes ramène l'espace des configurations à un espace tridimensionnel ce qui nous donne accès à l'ensemble de l'algorithmique 3D. Afin de localiser précisément ces relations, autrement dit ces intersections entre les volumes, nous utilisons un moteur de détection de collision que nous associons à l'espace des configurations. La définition des Volumes d'Interaction nous permet une construction et une représentation intuitive des objets dans le moteur. Grâce aux nombreuses méthodes proposées et à un domaine de recherche très actif depuis quelques dizaines d'années, les moteurs de détection de collisions présentent aujourd'hui l'avantage d'offrir des algorithmes très perfectionnés et optimisés. Ceci permet de gérer des environnements contenant plusieurs centaines d'objets en temps réel [LG98, TKH+05, KHI+07, Avr11].

Les moteurs de détection de collision offrent plusieurs optimisation afin d'accélérer les calculs du moteur. Il est ainsi possible de paramétrer des filtres afin de considérer seulement les collisions entre certains types d'objets. Il va également être possible de

paramétrer les types d'informations qui sont désirés en retour comme le calcul ou non d'une profondeur d'interpénétration, ou des forces de répulsions créées par la collision de deux objets. Afin d'augmenter les performances de ces moteurs, la détection de collision est, de plus, généralement composées de deux parties. La *broad phase* va tout d'abord effectuer une rapide identification des paires d'objets étant potentiellement en collision. Les paires d'éléments conservées sont alors testées plus finement à l'aide de la *narrow phase* qui détermine alors de manière précise s'il y a interpénétration ou non entre les deux éléments. Le fonctionnement global de la détection de collision que nous utilisons est représenté dans la figure 3.4. En nous basant sur les propriétés de ces moteurs de détection de collision et afin de réduire le nombre d'éléments à prendre en compte au cours de la *narrow phase*, nous filtrons les éléments présents dans l'espace des configurations en nous basant sur les propriétés suivantes :

1. Une relation peut exister entre deux Volumes d'Interactions *ssi* ceux-ci sont associés à la même capacité de déplacement M_k ;
2. Une relation d'accessibilité est définie par :
   $A(O_i, O_j, M_k) \Leftrightarrow V_a(O_i, M_k) \cap Ns(O_j, M_k) \neq \emptyset$;
3. Une relation d'obstruction est définie par :
   $O(O_i, O_j, M_k) \Leftrightarrow V_f(O_i, M_k) \cap (\ Ns(O_j, M_k) \cup V_a(O_j, M_k)\ ) \neq \emptyset$.

Ces trois propriétés nous permettent donc de remarquer que nous sommes intéressés par seulement 3 types d'interactions qui sont définies par les intersections suivantes :

- $V_a(O_i, M_k) \cap Ns(O_j, M_k)$
- $V_f(O_i, M_k) \cap Ns(O_j, M_k)$
- $V_f(O_i, M_k) \cap V_a(O_j, M_k)$

Les autres cas, tels que les interactions entre des volumes associés à des capacités de déplacement M_k et M_l différentes, ou à des Volumes d'Interaction de même nature sont donc filtrés par ce moteur. Ce filtrage allège le nombre d'éléments à prendre en compte lors de la *narrow phase*. D'autre part, nous configurons cette *narrow phase* pour qu'elle s'arrête dès qu'une paire de triangle est détectée en collision lors des tests. Cette collision est alors rapportée, et nous permet de caractériser une relation topologique. Cette propriété nous permet également d'alléger cette *narrow phase* en évitant des calculs trop coûteux tels que la distance d'interpénétration, qui ne nous sont pas utiles dans le cas des relations topologiques.

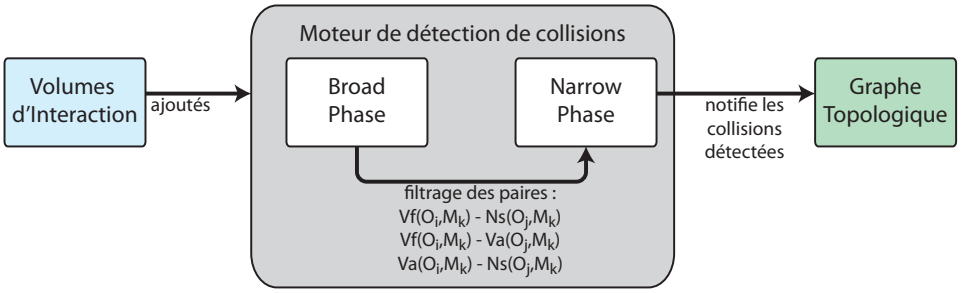


Figure 3.4 – Mise à jour du Graphe Topologique par le moteur de détection de collision

De plus, les moteurs de détection de collisions présentent également l'avantage d'être de plus en plus intégrés aujourd'hui dans les mondes virtuels. De nombreux environnements disposent, par exemple, d'un moteur physique afin d'immerger un utilisateur et son avatar dans une simulation plus réaliste. Notre méthode va donc pouvoir utiliser ces moteurs physiques afin de tirer parti des informations calculées par ce moteur. Celles-ci sont détournées et réutilisées afin d'optimiser l'identification des relations topologiques entre les Volumes d'Interaction. Il est, par exemple, possible de se baser sur la *broad phase* du moteur physique afin de filtrer des paires d'éléments proches dans l'environnement. Ainsi, nous limitons les calculs d'interaction entre les Volumes d'Interaction concernés. L'utilisation des Volumes d'Interaction et la détection des relations à l'aide d'un moteur de détection de collisions vont tout de même surcharger les calculs à effectuer au cours de la simulation. Mais cette surcharge est toutefois limitée par les différents filtrages utilisés et par l'utilisation d'un outil déjà très présent dans les environnements virtuels considérés. Nous allons maintenant rassembler l'ensemble de ces informations au sein d'une structure globale nous permettant de planifier des chemins au sein de l'environnement global.

2 Représentation de la topologie globale et de son évolution au cours du temps

Dans la section précédente, nous nous sommes focalisés sur la détection de relations d'accessibilité et d'obstruction entre deux objets de l'environnement. Nous cherchons maintenant à obtenir une représentation de la topologie globale de l'environnement. Cette représentation va devoir organiser l'ensemble des informations issues de l'identification des relations topologiques, et y ajouter des informations temporelles issues de l'environnement dynamique. Nous introduisons dans cette section le Graphe Topologique dont l'objectif est de représenter cette topologie dynamique. De plus, en exploitant les informations temporelles, il sera possible à travers ce graphe de déduire des propriétés concernant les relations temporelles identifiées.

2.1 Le Graphe Topologique

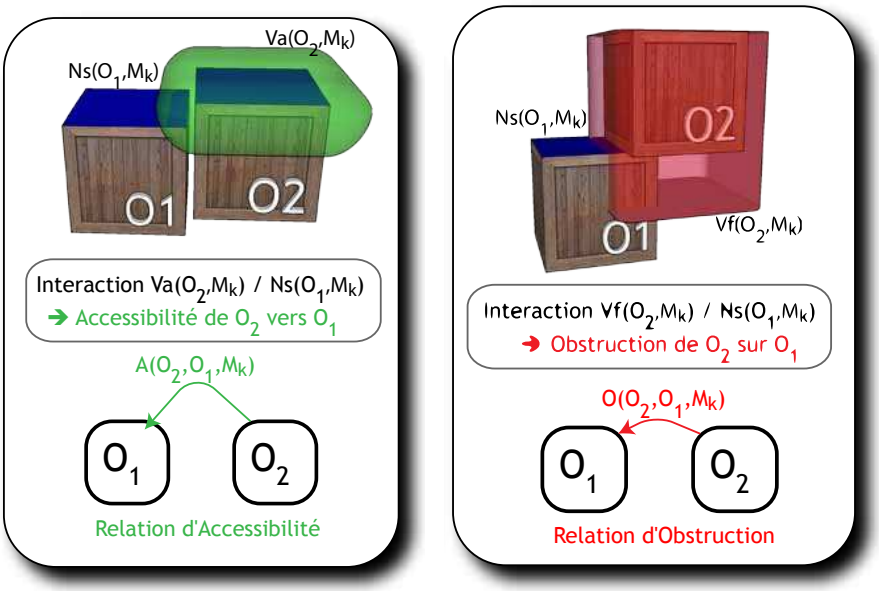
La détection des accessibilités et des obstructions entre paires d'objets nous donne un aperçu à un temps donné de la topologie. Le Graphe Topologique est mis en place pour fournir une vue globale de la topologie de l'environnement en structurant l'ensemble des relations identifiées. Comme le montre le schéma 3.4, la détection de collisions entre les Volumes d'Interaction permet de rapporter les informations concernant les relations identifiées au Graphe Topologique et de maintenir à jour ce dernier. Cette partie présente tout d'abord comment est défini ce Graphe Topologique puis dans un second temps nous nous appuierons sur un exemple afin de montrer sa construction.

2.1.1 Construction du Graphe Topologique

Le Graphe Topologique permet de représenter les relations topologiques de l'environnement et ainsi la configuration de l'environnement dans sa globalité. Ce Graphe Topologique est défini de la manière suivante :

- Un nœud du graphe est associé à chaque objet O_i de l'environnement.
- Un arc orienté reliant deux nœuds représente une relation topologique entre les objets O_i et O_j associés. L'orientation de ces arcs est importante puisque les relations d'accessibilité et d'obstruction ne sont pas bijectives. Deux types d'arcs sont distingués :
  - les arcs représentant les relations d'accessibilité
  - les arcs représentant les relations d'obstruction

Une illustration du lien entre la détection des interactions dans l'environnement et la construction du Graphe Topologique est proposée dans la figure 3.5. Chaque relation topologique est donc caractérisée dans la topologie globale à travers cette représentation. De plus, l'absence d'arc entre deux nœuds permet de caractériser l'absence de relation topologique entre des paires d'éléments. La mise à jour de ce Graphe Topologique est présentée dans la figure 3.6.



(a) Détection et création d'une relation d'accessibilité

(b) Détection et création d'une relation d'obstruction

Figure 3.5 – Création du Graphe Topologique et caractérisation des relations topologiques entre les objets. 3.5a : Caractérisation d'une accessibilité et création de l'arc associé. 3.5b : Détection d'une obstruction et création de l'arc associé.

Le Graphe Topologique montre une représentation globale de la topologie de l'environnement. Toutefois, celle-ci est un sur-ensemble de la topologie réelle de l'environne-

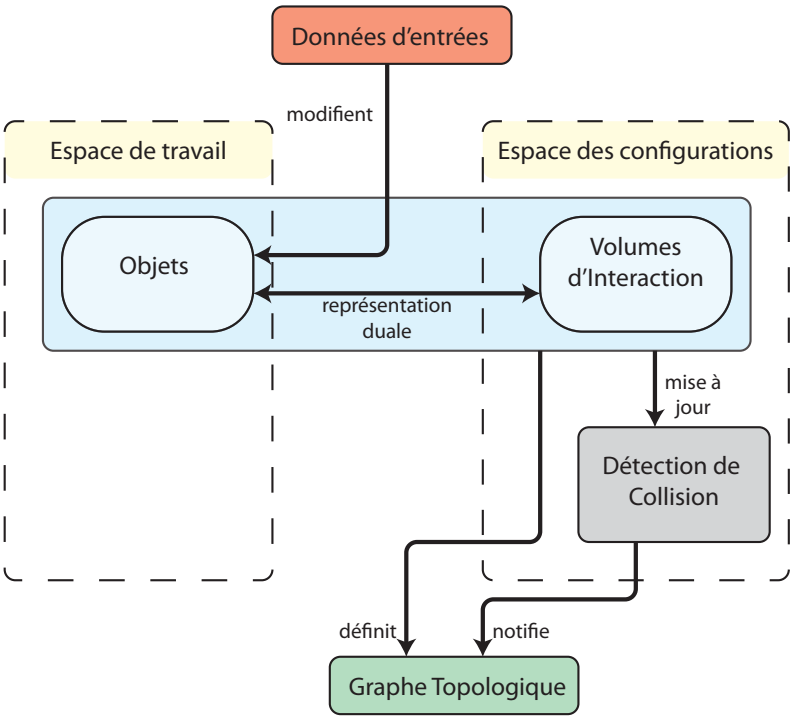


Figure 3.6 – Mise à jour du Graphe Topologique par le moteur de détection de collision

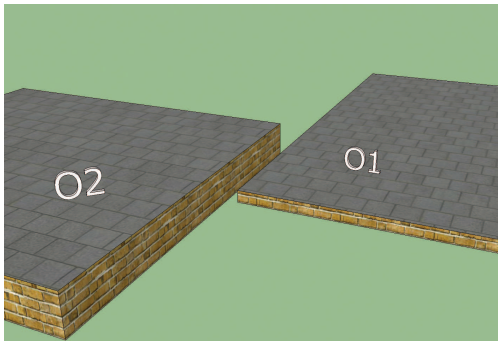
ment. En effet, certains doutes peuvent apparaître à la vue des accessibilités détectées dans le graphe :

1. Un objet va tout d'abord pouvoir couper l'accès entre deux parties d'une Surface Navigable en obstruant celle-ci de part en part.
2. Un objet va également pouvoir couper une accessibilité détectée entre deux élément navigables.

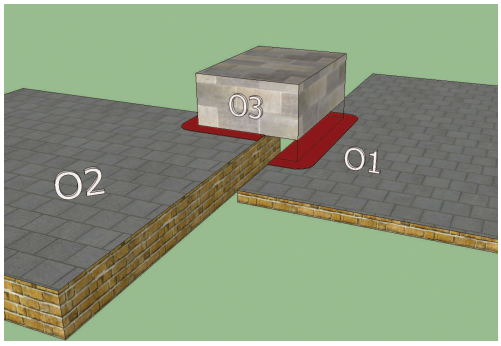
Les relations d'accessibilité identifiées dans le Graphe Topologique vont donc devoir être raffinées afin d'être validées. Ces doutes sont dus à plusieurs facteurs. Dans le cas **1.**, l'obstruction est détectée dans le graphe, mais son impact réel sur la Surface Navigable de l'objet n'a pas été pour le moment raffinée, et n'est donc pas connu précisément. Dans le second cas, cette accessibilité détectée entre deux éléments est rendue invalide à cause de la présence d'un troisième élément car le Graphe Topologique n'identifie que des relations entre des paires d'éléments et ne caractérise pas les relations entre un triplé d'objets.

Les figures 3.7b et 3.7c illustrent une situation où la présence d'un objet va obstruer l'accessibilité entre deux surfaces. Dans le cas de la figure 3.7b, la présence de O_3 contraint peu l'accès de O_1 à O_2 et cet accès est toujours possible. À l'opposé, dans l'exemple de la figure 3.7c, l'objet O_3 bloque totalement l'accès de O_1 à O_2 et crée donc une relation dite de "fausse accessibilité" entre ces deux éléments. Dans les deux cas, une accessibilité est tout d'abord rapportée par la détection de collisions depuis

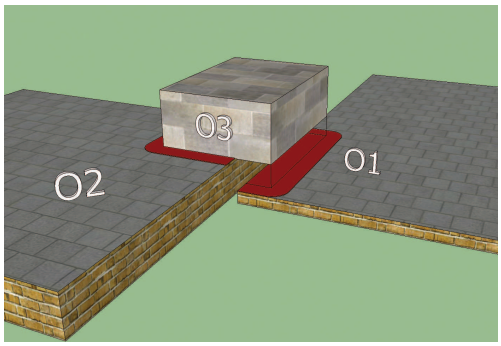
l'objet O_1 jusqu'à O_2 ($A(O_1, O_2, M_k)$). Deux relations d'obstructions sont également caractérisées : $O(O_3, O_1, M_k)$ et $O(O_3, O_2, M_k)$. L'accessibilité de O_1 vers O_2 est de ce fait remise en cause par la présence de l'objet O_3. Afin de représenter plus précisément la topologie, nous allons devoir raffiner ces accessibilités car le Graphe Topologique présente, dans les deux cas, la même configuration, comme on peut le voir figure 3.7d. Ce raffinement, expliqué dans le chapitre suivant, permettra de lever l'incertitude sur de tels cas.



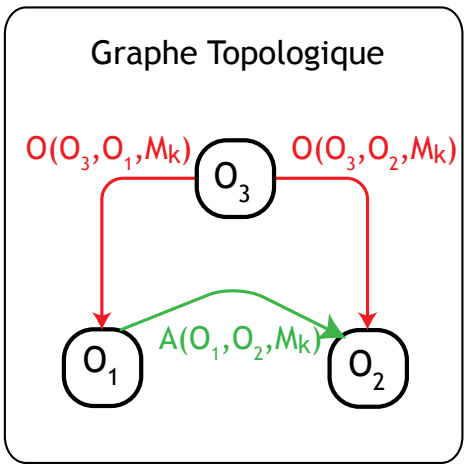
(a) Accessibilité réelle de O_1 à O_2



(b) Accessibilité de O_1 à O_2 et obstruction de O_3 qui ne gêne pas entièrement l'accessibilité



(c) Fausse accessibilité de O_1 à O_2 : le passage de O_1 à O_2 est totalement obstrué par la présence de O_3



(d) Graphe Topologique en présence de O_3

Figure 3.7 – 3.7a : L'accessibilité détectée entre les deux surfaces est valide. 3.7b : une accessibilité est détectée et la présence de l'obstacle O_3 ne gêne pas cette accessibilité. 3.7c : L'accessibilité est détectée or elle est maintenant invalidée par O_3 ce qui implique une fausse accessibilité. 3.7d : Graphe Topologique associé aux situations présentées en 3.7b et 3.7c. Le Graphe Topologique est le même dans les deux cas.

Nous avons défini au cours de cette section une représentation globale de la topologie. Cette représentation, appelée Graphe Topologique, se base sur l'observation des relations topologiques identifiées dans l'environnement pour proposer une vue ponctuelle de la configuration de l'environnement. Afin de pouvoir planifier la trajectoire d'un agent au fil du temps dans un environnement dynamique, il est donc nécessaire d'ajouter des informations temporelles à notre représentation de l'environnement. Cela nous permettra d'anticiper l'évolution de l'environnement et les relations topologiques à venir.

2.1.2 Exemple de construction d'un graphe Topologique

Nous allons maintenant illustrer la construction d'un Graphe Topologique à l'aide d'un environnement de démonstration présenté dans la figure 3.8. Celui-ci est composé d'un sol totalement navigable et d'éléments variés. Afin de simplifier cet exemple, nous allons considérer que l'agent ne possède qu'une seule capacité de déplacement M de type saut. Cette capacité lui permet donc, à l'aide de petits bons, de bouger sur une Surface Navigable mais également d'accéder à des Surfaces navigables appartenant à d'autres éléments.

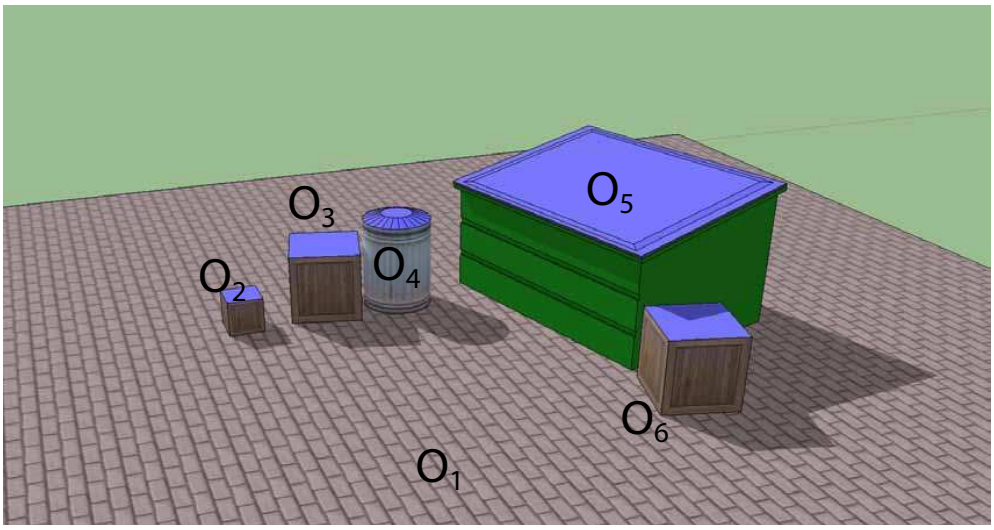


Figure 3.8 – L'environnement utilisé est constitué d'un sol entièrement navigable (O_1) et d'objets divers (annotés de O_2 à O_6) dont les Surfaces Navigables sont représentées en bleu.

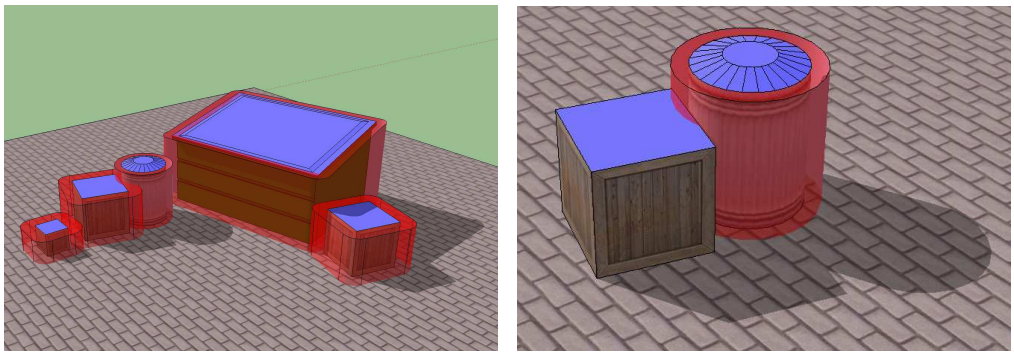
Nous commençons par précalculer les Volumes d'Interactions associés à chaque élément de l'environnement. L'agent possède une unique capacité de déplacement M. Un élément de l'environnement est donc caractérisé dans l'espace des configurations par trois Volumes d'Interactions, un de chaque type. Chaque élément O_i se voit donc associer une Surface Navigable $Ns(O_i, M)$, un Volume d'Interdiction $V_f(O_i, M)$ et un Volume d'Accessibilité $V_a(O_i, M)$. Les caisses en bois présentent des Surfaces Navigables horizontales, tandis que la poubelle ronde et le bac à ordures ont des surfaces

inclinées. Afin de visualiser les Surfaces Navigables des objets composant la scène, celles-ci ont été dessinées en bleu sur la figure 3.8. La figure 3.9a montre les Volumes Interdits définis pour les différents éléments. Enfin, la figure 3.10 montre quelques-uns des Volumes d'Accessibilités qui ont été définis.

L'ensemble des Volumes d'Interactions ainsi créé est ajouté dans l'espace des configurations, et dans le moteur de détection de collisions qui lui est associé. Comme il a été vu dans la section 1.2, le moteur de détection de collisions est configuré afin de rapporter certaines collisions :

- Volumes d'Accessibilité $V_a(O_i, M)$ / Surfaces Navigable $Ns(O_j, M)$
- Volumes Interdits $V_f(O_i, M)$ / Surfaces Navigable $Ns(O_j, M)$
- Volumes Interdits $V_f(O_i, M)$ / Volumes d'Accessibilité $V_a(O_j, M)$

L'annotation de ces volumes lors de leur insertion dans le moteur va permettre d'effectuer un premier filtrage lors de la détection de collision, afin d'alléger le coût de cette dernière en ne considérant que les Volumes d'Interaction pertinents. En d'autres termes, nous considérons seulement les Volumes d'Interactions pouvant potentiellement être à l'origine de relations topologiques par la suite.



(a) Détection des obstructions des objets sur le sol O_1

(b) Obstruction de l'objet O_4 sur l'objet O_3

Figure 3.9 – Détection des relations d'obstruction dans l'environnement. 3.9a Les Volumes Interdits (en rouge) des objets O_2, O_3, O_4, O_5 et O_6 sont tous en interaction avec la Surface Navigable du sol O_1. 3.9b Le Volume Interdit de O_4 est en interaction avec une partie de la Surface Navigable de O_3.

Pendant l'exécution de l'application, le moteur de détection de collision tourne en parallèle à la simulation, agissant en tâche de fond. À chaque fois qu'une paire d'objets en collision est détectée, le moteur notifie directement le Graphe Topologique pour lui signaler les éléments détectés en collision, et la nature des Volumes d'Interaction concernés. Dans l'exemple présent, le moteur va détecter différents types de collisions :

1. Collisions entre les Volumes Interdits des objets O_2 à O_6 avec la Surface Navigable du sol O_1, définies par $\forall i \in [2;6], V_f(O_i, M) \cap Ns(O_1, M) \neq \emptyset$
2. Collisions entre le Volume Interdit de l'objet O_4 avec la Surface Navigable de l'objet O_3, définie par $V_f(O_4, M) \cap Ns(O_3, M) \neq \emptyset$

3. Collisions entre les Volumes d'Accessibilité et les Surfaces Navigables suivantes :
- $V_a(O_1, M) \cap Ns(O_2, M) \neq \emptyset$
- $V_a(O_2, M) \cap Ns(O_1, M) \neq \emptyset$
- $V_a(O_2, M) \cap Ns(O_3, M) \neq \emptyset$
- $V_a(O_3, M) \cap Ns(O_2, M) \neq \emptyset$
- $V_a(O_3, M) \cap Ns(O_4, M) \neq \emptyset$
- $V_a(O_4, M) \cap Ns(O_3, M) \neq \emptyset$
- $V_a(O_4, M) \cap Ns(O_5, M) \neq \emptyset$
- $V_a(O_5, M) \cap Ns(O_4, M) \neq \emptyset$
- $V_a(O_5, M) \cap Ns(O_6, M) \neq \emptyset$

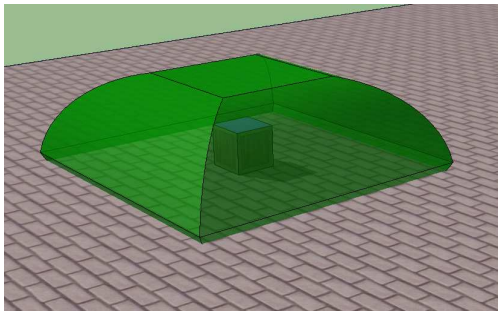
Les différentes figures fournies permettent d'illustrer ces collisions détectées dans l'environnement. La figure 3.9a montre que les Volumes Interdits de tous les objets de l'environnement sont en interaction avec le sol navigable. La figure 3.9b illustre la collision entre le Volume Interdit de O_4 et la Surface Navigable de O_3. La figure 3.10 présente certaines des interactions détectées entre les Surfaces Navigables et les Volumes d'Accessibilité dans l'espace des configurations.

Le moteur de détection de collision renvoie donc au Graphe Topologique l'ensemble des paires de Volumes d'Interaction qu'il détecte en interpénétration. Le Graphe Topologique identifie ensuite, à partir de ces paires d'éléments, les relations topologiques correspondantes. Ainsi, les interactions impliquant un Volume Interdit et une Surface Navigable vont créer des relations d'obstruction, tandis que les collisions impliquant un Volume d'Accessibilité et une Surface Navigable permettront de caractériser une accessibilité. Au regard des différentes collisions détectées précédemment, les collisions des cas **1.** et **2.** seront représentées par des relations d'obstructions dans le Graphe Topologique tandis que les obstructions du cas **3.** vont créer des relations d'accessibilités entre les objets concernés. Une vue du Graphe Topologique associé à cet environnement est présentée dans la figure 3.11, les relations d'obstructions sont les arcs notés de RO_1 à RO_6 tandis que les relations d'accessibilités détectées sont notées de RA_1 à RA_9. Ce Graphe Topologique est ensuite maintenu à jour, au fil de la simulation, par le moteur de détection de collision.

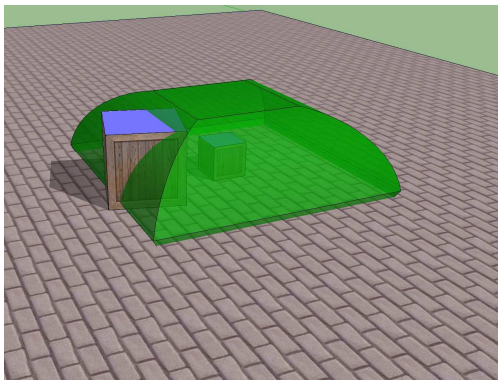
Le Graphe Topologique construit dans cet exemple nous donne une vue ponctuelle de la topologie de l'environnement. Toutefois, dans le cadre d'environnements dynamiques, cette vue ponctuelle va se révéler insuffisante. La prise en compte de la composante temporelle dans la Graphe Topologique va nous permettre de détecter des relations pouvant évoluer au cours de la simulation. Nous utiliserons les propriétés de ces relations pour détecter des relations périodiques ou encore des accès entre des zones physiquement déconnectées pendant la simulation, mais connectées par le biais d'un objet mobile.

2.2 Représentation de la topologie et de son évolution au cours du temps

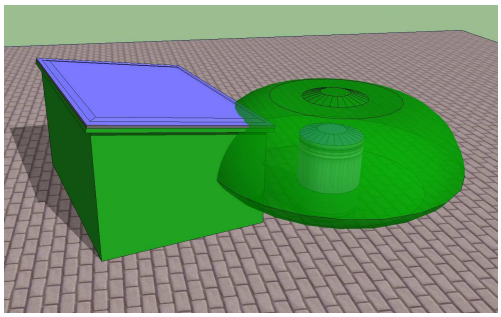
Nous nous sommes intéressés au cours de cette thèse à des environnements dynamiques dont l'évolution de la topologie au fil de la simulation n'est pas connue a priori.



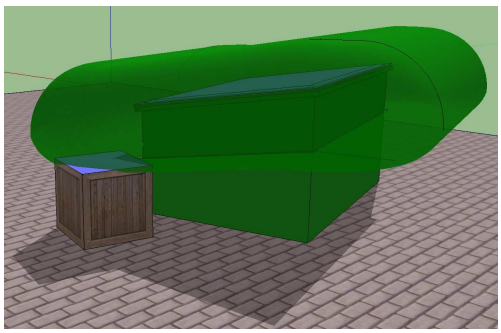
(a) Détection de l'accessibilité de la caisse O_2 au sol O_1



(b) Détection de l'accessibilité de la caisse O_2 à O_3



(c) Détection de l'accessibilité entre O_4 et O_5



(d) Détection de l'accessibilité de O_5 à O_6

Figure 3.10 – Représentation de quelques relations détectées entre différents éléments de l'environnement présenté Fig.3.8. 3.10a : Le Volume d'Accessibilité de la caisse O_2 est en collision avec le sol. 3.10b : Détection d'une accessibilité depuis la caisse O_2 vers O_3. 3.10c : Accessibilité depuis le sommet de la poubelle O_4 vers l'objet O_5. 3.10d : Accessibilité entre le Volume d'Accessibilité de O_5 et la Surface Navigable de O_6.

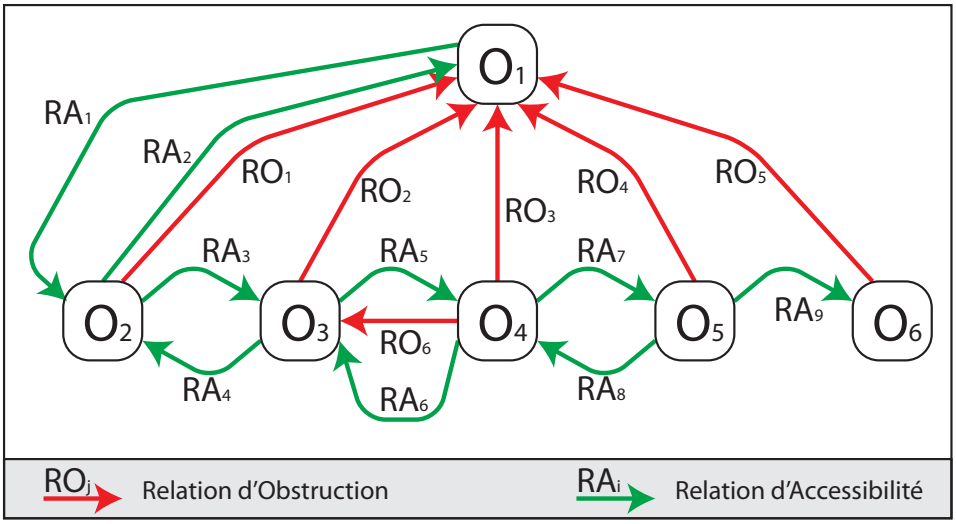


Figure 3.11 – Graphe Topologique associé à l'environnement de test. Chaque relation topologique détectée est représentée par un lien entre les objets affectés. Ce graphe permet d'avoir une représentation globale de la topologie entre les objets de l'environnement au moment de la détection.

Cette évolution va pouvoir être la conséquence de plusieurs facteurs. L'action d'un utilisateur externe, d'un scénario ou d'un moteur physique sont autant d'éléments pouvant modifier la configuration de l'environnement au fil du temps. De tels environnements sont donc imprévisibles au niveau des déplacements et des modifications de la topologie de l'environnement. Afin de planifier correctement des chemins pour l'agent au sein de tels environnements, l'évolution temporelle de la topologie globale doit être prise en compte au cours de la planification pour anticiper les déplacements des éléments, mais également pour les exploiter.

Prenons pour exemple une plateforme se déplaçant de façon périodique comme dans la situation présentée figure 3.12. Celle-ci va de temps à autre permettre d'accéder à une première surface et à d'autres moments permettre l'accès à une seconde surface. Toutefois, cette plateforme navigable n'est jamais connectée simultanément aux deux éléments. Aucun accès direct n'existe entre ces deux surfaces. Des images ponctuelles de la topologie l'environnement ne vont pas permettre d'identifier un chemin permettant de passer de l'un des éléments au second. La prise en compte de l'aspect temporel de cette simulation va nous permettre de représenter ces accessibilités périodiques et nous allons pouvoir identifier le chemin permettant de lier ces deux éléments en utilisant le déplacement de la plateforme. Le mouvement propre de l'objet va alors créer un pont temporel entre les deux autres éléments. L'intégration d'une composante temporelle dans le Graphe Topologique permet de détecter et d'exploiter de telles situations. L'agent sera alors capable de localiser ces ponts lors de la planification de chemin et de les utiliser lors de sa navigation.

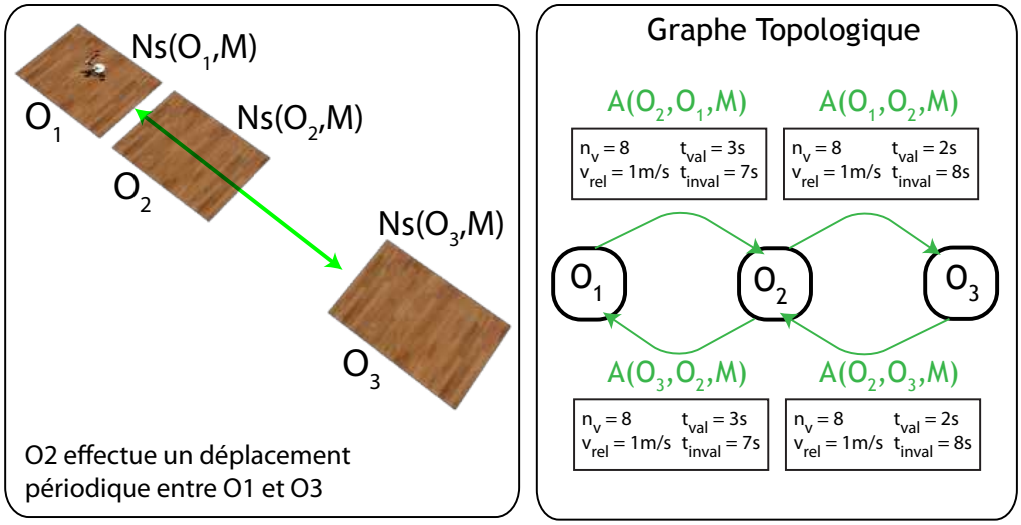


Figure 3.12 – L'objet O_2 agit comment un pont temporel en connectant les objets O_1 et O_3. Les objets ne subissent pas de rotations, pour l'ensemble des relations nous avons donc : $\theta_{min} = \theta_{max} = 0$

En fonction des objets présents dans l'environnement, les relations topologiques détectées vont posséder des propriétés temporelles différentes. Des objets statiques vont par exemple créer des liens permanents entre certains éléments. D'un autre côté, le déplacement de certains objets va engendrer la modification des liens topologiques associés à ces objets. Enfin, certaines relations peuvent présenter une certaine périodicité valides pendant certaines périodes, puis invalides ensuite, avant de redevenir valides par la suite.

En faisant l'hypothèse que l'évolution du monde n'est pas connue a priori, l'observation de l'environnement et de ses modifications au cours du temps va permettre d'apprendre comment celui-ci évolue. Du point de vue de l'agent, la représentation de l'environnement sera donc le regroupement d'un ensemble d'observations faites au fur et à mesure de la simulation. Puis, améliorant sa base de connaissances, une mémoire se construit au fil de la simulation. Le Graphe Topologique fournit une image ponctuelle de la topologie. À la manière d'un film, nous proposons donc d'associer ces images ponctuelles de la topologie afin d'en créer une représentation continue. Ces images ponctuelles vont ainsi nous permettre de détecter les modifications dans les relations topologiques.

L'observation des modifications dans le Graphe Topologique va nous permettre de déduire différentes propriétés sur les relations de l'environnement. Les caractéristiques suivantes sont donc relevées pour chaque relation qui apparaît dans le Graphe Topologique, c'est à dire qui a été observée à un instant donné :

- la validité actuelle de la relation, *valide* ou *invalide* ;
- le nombre de validation de la relation, n_v ;

- les temps moyen de validité et d'invalidité du lien, t_{val} et t_{inval} ;
- la vitesse relative des objets concernés, V_{rel} ;
- les inclinaisons minimale θ_{min} et maximale θ_{max} observées pour la surface si celle-ci présente des mouvements de roulis ou de tangage par exemple.

Ces éléments sont stockés au niveau de chaque relation du graphe. Lorsque des relations sont invalidées, alors qu'elles étaient précédemment valides, elles sont conservées dans le Graphe Topologique, et leurs caractéristiques sont mises à jour. Ces caractéristiques sont ainsi des statistiques construites et remises à jour au fil des détections de collisions. Ces statistiques vont donc changer en fonction de l'évolution de l'environnement et traduire ainsi les modifications observées au sein de l'environnement. L'utilisation de ces caractéristiques va nous permettre de déduire différentes propriétés temporelles concernant les relations considérées.

Tout d'abord, le *nombre de validités de la relation* n_v permet de distinguer trois types de relations :

- les relations statiques, détectées une première fois et qui sont restées valides depuis ce temps ;
- les relations ponctuelles, détectées une première fois, invalidées, puis qui n'ont jamais été de nouveau détectées ;
- les relations périodiques, régulièrement validées puis invalidées au fil de la simulation. Cela permet de déduire qu'elles ont de grandes chances d'apparaître de nouveau au cours de la simulation.

Dans le cas des relations périodiques, le *temps de validité moyen* t_{val} d'une relation nous permet de déduire la fiabilité de celle-ci et de déterminer si elle est assez sûre pour la navigation de l'agent. Dans le cas d'une relation d'accessibilité avec un temps de validité court par exemple, l'agent disposera de peu de temps pour effectuer sa transition. Il aura donc quelques risques à effectuer cette transition. Dans le cas où ce temps de validité est plus long, l'agent va disposer de plus de marge de manœuvre, afin d'anticiper les connexions et de réagir. La relation sera donc plus sûre pour la navigation. Le *temps moyen d'invalidité* t_{inval} va permettre de déduire un "temps d'attente" moyen avant la prochaine validité de la relation. La somme de ces deux temps moyens représente enfin la "période moyenne d'une relation" ou, en d'autres termes, le délai entre deux détections consécutives de la relation dans l'environnement.

Finalement, la *vitesse relative* V_{rel} des objets ainsi que les inclinaisons θ_{min} et θ_{max} sont des critères de sécurité pour la navigation de l'agent. En effet, un agent dans un environnement dont l'évolution n'est pas connue a priori va avoir des difficultés à anticiper une transition entre deux éléments dont les vitesses relatives sont très différentes. De la même manière, il sera difficile pour lui de pouvoir, et de vouloir naviguer sur une surface instable dont l'inclinaison change de manière dangereuse.

La figure 3.12 présente un environnement simple où une plateforme mobile effectue des aller-retour entre deux éléments de l'environnement. L'information temporelle

permet de représenter au sein du Graphe Topologique l'accessibilité créée entre les éléments O_1 et O_3 bien que cette accessibilité ne soit jamais valide simultanément entre O_1-O_2 et O_2-O_3. Nous pouvons détecter cette accessibilité grâce à la conservation des liens dans le Graphe Topologique même lorsqu'ils ont été invalidés et également grâce à l'ajout des informations temporelles. Ces informations nous permettent ainsi de déterminer, pour cet exemple, que la période moyenne du déplacement de la plateforme est de 10 secondes. De plus, la connexion entre O_1 et O_2 est valide en général pendant 3 secondes, tandis que celle entre O_2 et O_3 est généralement plus rapide, restant valide pendant 2 secondes. L'accessibilité identifiée ne repose pas seulement sur les capacités de l'agent mais également et surtout sur le déplacement propre de la plateforme entre les deux éléments. Grâce à cette représentation temporelle, l'agent est capable de détecter un chemin à travers des éléments mobiles mais surtout de se servir de ces objets dynamiques pour assister sa navigation.

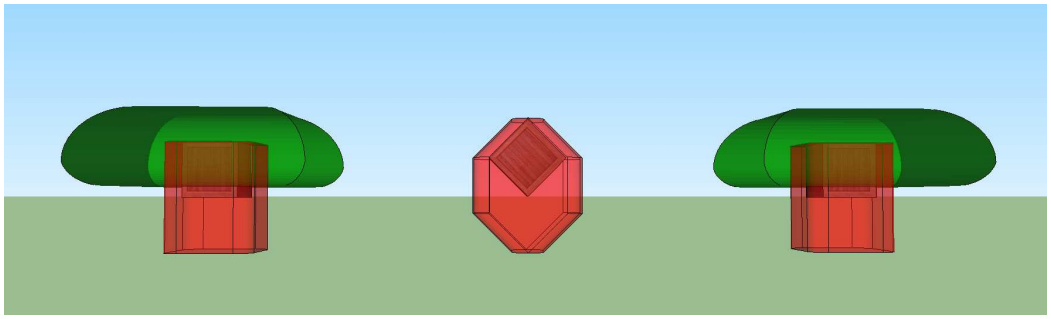


Figure 3.13 – Les Volumes d'Interaction sont remis à jour lors de la rotation de l'élément. L'élément est représenté simplement par un Volume Interdit construit à partir de la boite englobante de sa géométrie pendant son mouvement (figure centrale). Lorsque l'objet se stabilise dans l'environnement, les Volumes d'Interaction sont de nouveaux recalculés et associés à l'élément.

2.3 Suivi des modifications de Volumes d'Interaction

Nous avons vu au cours du chapitre précédent que la prise en compte d'environnements dynamiques implique de devoir maintenir à jour les Volumes d'Interaction en les recalculant si nécessaire. La modification des objets, et surtout leur rotation, va modifier leurs propriétés topologiques. Ces changements ont de plus un impact sur les relations topologiques détectées dans l'espace des configurations.

Comme le rappelle la figure 3.13, nous avons vu précédemment que la mise en rotation d'un élément autour d'un axe horizontal entraîne des modifications dans la définition des Volumes d'Interaction. Les Surfaces Navigables et les Volumes d'Accessibilité de l'objet sont supprimés et seul un Volume Interdit est conservé en attendant que l'élément se stabilise. La modification de ces Volumes d'Interaction va donc engendrer l'ajout et la suppression d'éléments dans le moteur de détection de collision associé à l'espace des configurations au cours de la simulation.

Malgré la modification des Volumes d'Interaction, la représentation proposée par le Graphe Topologique doit rester cohérente au fil du temps. Lorsqu'un élément de l'environnement se déplace, le nœud représentant cet élément dans le Graphe Topologique reste donc actif. Les relations topologiques associées sont également conservées. Lorsque le déplacement de l'objet n'implique pas de suppression des Volumes d'Interaction, la mise à jour du graphe est effectuée normalement. Dans le cas où les Volumes d'Interaction ont besoin d'être supprimés puis redéfinis, nous conservons les relations topologiques associés à ces Volumes, en les marquant toutefois comme invalides lors de la suppression des Volumes d'Interaction. Les Volumes d'Interaction recréés par la suite vont être alors associés au nœud déjà présent dans le Graphe Topologique pour représenter l'objet. Si les relations précédemment définies sont de nouveau détectées, elles sont alors réactivées dans le graphe ce qui permet de conserver toutes les informations disponibles concernant leur historique. Il est toutefois possible qu'un objet se retrouve éloigné de sa position d'origine et qu'il n'ait donc plus aucune interaction avec les objets identifiés par les relations topologiques. Il sera donc intéressant d'utiliser un critère utilisant la mémoire du Graphe Topologique afin d'identifier de tels cas et de les supprimer de la représentation.

En prenant en compte des environnements dynamiques, un utilisateur aura la possibilité d'ajouter ou supprimer des éléments dans l'environnement pendant l'exécution. En ce qui concerne les simulations physiques, les environnements sont majoritairement destructifs. Ils permettent rarement des retours en arrière pour retrouver des configurations précédentes. Nous pouvons facilement imaginer faire tomber un tas de barils et le faire exploser, détruisant des bâtiments au passage. Mais nous pouvons rarement replacer tous ces éléments a posteriori dans leur situation originelle. L'ajout, comme la suppression, d'éléments dans un monde virtuel dynamique vont donc devoir être pris en compte. La gestion de ces ajouts et suppressions est facilitée au niveau du Graphe Topologique puisqu'il suffit d'ajouter, ou respectivement de retirer, un nœud représentant l'objet. Lors de la suppression d'un nœud, les relations topologiques associées sont également retirées du graphe. Lors de l'ajout d'un objet, la définition et l'insertion dans l'espace des configurations des Volumes d'Interaction vont permettre de localiser automatiquement les relations topologiques. Dès son ajout dans l'environnement, les relations seront donc ajoutées au Graphe Topologique et l'objet sera donc intégré directement à la navigation de l'agent.

3 Conclusion

Nous avons présenté dans ce chapitre une représentation dynamique de la topologie à travers le Graphe Topologique. Cette représentation est construite par l'observation des relations topologiques et par le suivi de leur évolution au cours du temps. Cette observation nous permet ainsi d'extraire de la connaissance sur les relations topologiques présentes. La structure mise ensuite en place permet de représenter de manière homogène les relations topologiques statiques et dynamiques identifiées au sein d'un environnement non connu. L'utilisation des informations temporelles permet, sans in-

jecter de connaissance a priori, de détecter des ponts temporels ou encore d'estimer des délais d'attente ou des connexions à risque entre les éléments. Le Graphe Topologique va ainsi permettre de localiser des accessibilités entre des zones navigables déconnectées et mobiles dans l'environnement. Grâce à cette structure, un agent sera donc capable d'identifier au cours de sa navigation des chemins entre des zones déconnectées à la fois dans l'espace mais également dans le temps. De plus, cette représentation est maintenue à jour par l'utilisation du moteur de détection de collision associé à l'espace des configurations. Cela permet de conserver en tout temps une représentation valide de la topologie globale de l'environnement et d'en connaître instantanément les changements.

Toutefois, le Graphe Topologique est une sur-représentation de la topologie réelle de l'environnement. Il faut tout d'abord observer que les informations temporelles concernant les relations topologiques sont statistiques. Ces statistiques sont calculées en se basant sur les modifications observées au cours de l'évolution de l'environnement. Elles illustrent donc l'évolution de la topologie jusqu'à ce moment et permettent une estimation de l'évolution future de l'environnement. Elles ne sont toutefois pas infaillibles quant aux prédictions qui peuvent en être déduites. En effet, l'évolution de l'environnement reste imprévisible et non connue a priori. D'autre part, les relations topologiques sont détectées entre des paires d'éléments, la prise en compte d'un troisième d'élément pourra modifier cette relation. Ainsi, lorsqu'une obstruction intervient, certaines relations d'accessibilité peuvent être invalidées et devront donc être raffinées. Enfin, la configuration locale et précise des Surfaces Navigables doit également être raffinée afin de permettre une navigation précise de l'agent évitant les collisions avec les obstacles et utilisant les accessibilités détectées.

Le Graphe Topologique est donc une représentation globale de la topologie qui, en intégrant des informations temporelles liées aux relations topologiques, permet de construire une représentation dynamique de la configuration de l'environnement. De plus, en se basant sur les observations faites, cela offre l'opportunité d'anticiper certaines évolutions futures de l'environnement alors que cette évolution n'est pas connue a priori. Certaines relations nécessitent toutefois d'être raffinées afin d'être validées. Nous allons maintenant aborder la planification de chemin de l'agent. Celle-ci va se baser principalement sur la représentation dynamique de la topologie fournie par le Graphe Topologique. Toutefois, nous présentons dans un premier temps comment raffiner l'ensemble des informations apportées par le Graphe Topologique, afin de détecter les relations invalides et les zones obstruées de l'environnement.

Planification de trajectoire en environnement dynamique

4

Table des matières

1 Planification de la trajectoire locale . . . 84
1.1 Représentation de la topologie locale à un élément . . . 84
1.2 Anticipation de l'évolution des obstructions locales dans le domaine spatio-temporel . . . 87
1.3 Planification locale dans le domaine spatio-temporel . . . 90
1.4 Réactivité de l'agent pendant la navigation . . . 94
2 Planifier le déplacement de l'agent dans l'environnement dynamique 95
2.1 Raffinement des relations topologiques . . . 96
2.2 Calcul et optimisation du chemin topologique . . . 99
3 Planification hiérarchique . . . 103
3.1 Association de la planification globale et locale . . . 103
3.2 Gestion d'une obstruction de la Surface Navigable . . . 104
4 Conclusion . . . 108

Les environnement dynamiques évoluant au fil du temps, la recherche de chemin va devoir également prendre en compte l'aspect temporel afin de proposer des solutions de navigation valides à l'agent. Les obstructions et les accessibilités changeant dynamiquement doivent ainsi être prises en compte dans la planification de trajectoire. Cette tâche est rendue d'autant plus difficile que l'évolution de cette topologie n'est pas connue a priori et que les propriétés des relations topologiques ne sont donc pas connues parfaitement. Au cours de ce chapitre, nous proposons une structure de planification hiérarchique permettant de planifier un chemin au sein d'un tel environnement dynamique. Cette structure hiérarchique est composée de deux niveaux. La planification locale identifie tout d'abord une trajectoire au sein d'une surface navigable unique permettant à l'agent une navigation anticipant les mouvements des obstacles dynamiques locaux. Dans un second temps la planification globale va permettre d'identifier une séquence d'objets dans le temps et l'espace permettant à l'agent d'atteindre sa destination dans l'environnement. La planification hiérarchique va donc permettre d'identifier un chemin global des éléments que l'agent doit traverser puis de calculer une trajectoire locale sur chaque surface ainsi identifiée. L'information temporelle nécessaire à cette planification au sein d'un environnement dynamique est obtenue à partir de la représentation dynamique de la topologie que nous avons défini dans le chapitre précédent à travers le Graphe Topologique.

1 Planification de la trajectoire locale

Dans un premier temps, nous nous focalisons sur la problématique de la navigation de l'agent sur la surface d'un élément unique. Afin de planifier une trajectoire sur cet élément, nous devons identifier l'ensemble des configurations où l'agent est capable de se tenir durant sa navigation puis déterminer comment il peut naviguer entre ces configurations. Nous proposons ici de capturer la topologie locale d'un objet à l'aide de cartes de cheminement associées aux éléments. La planification locale a ensuite pour but de calculer une trajectoire permettant à l'agent de naviguer sur l'élément en anticipant et en évitant les collisions avec les différents obstacles statiques et dynamiques obstruant l'élément.

1.1 Représentation de la topologie locale à un élément

Notre premier objectif est de capturer la topologie navigable d'un objet au sein d'une structure. Cette structure doit permettre de gérer la planification de trajectoire pour l'agent mais également de prendre en compte ses adaptations de posture. L'ensemble des configurations où un agent est capable de se tenir est capturé par les différentes Surfaces Navigables associées à un élément. Nous cherchons donc à regrouper au sein d'une structure compacte unique l'ensemble de ces informations concernant la navigabilité de la surface et les capacités de déplacement utilisables par l'agent. Pour ce faire, nous définissons une carte de cheminement locale que nous associons à chaque objet. Nous regroupons ensuite dans ces cartes de cheminement les informations de navigabilité issues des différentes Surfaces Navigables. Ainsi, en ajoutant au sein de ces cartes de cheminement des informations sur les capacités de déplacement, nous permettons d'intégrer directement les adaptations de posture de l'agent à la planification locale de la trajectoire et de représenter l'ensemble de la topologie des Surfaces Navigables dans une structure unique.

Nous proposons ici une approche utilisant des cartes de cheminement probabilistes de type PRM. L'utilisation d'une carte de cheminement nous permet d'avoir une représentation unique et permanente de la navigabilité des surfaces. Comme nous l'avons vu au cours du chapitre 1, plusieurs méthodes ont été proposées afin de générer des cartes de cheminement. Le choix d'une méthode à requêtes multiple telle que la PRM présente l'avantage de conserver la même représentation de la topologie de l'objet au fil de la simulation et de permettre la caractérisation directe de l'impact des relations topologiques au niveau local.

Une PRM est généralement construite en échantillonnant aléatoirement des configurations de l'espace des configurations et en conservant celles qui appartiennent à C-free. Les configurations échantillonnées sont ensuite connectées les unes aux autres en interpolant une transition à l'intérieur de C-free permettant de relier deux configurations considérées. Afin de gérer les différentes capacités de déplacement de l'agent,

nous augmentons l'information stockée dans notre carte de cheminement afin de faire ressortir les différentes capacités de déplacement dans la structure et de pouvoir ensuite réutiliser cette information pour la planification. Notre carte de cheminement est donc définie comme suit :

- Les nœuds représentent une configuration où l'agent va pouvoir se tenir et chaque nœud est annoté de l'ensemble des capacités de déplacement qui sont réalisables à cette configuration.
- Les arcs représentent une transition possible entre deux nœuds. Cette transition est possible si les deux nœuds partagent au moins une capacité de déplacement commune et que l'utilisation de cette capacité le long de l'arc est valide, i.e. que cet arc n'est pas obstrué au regard de la capacité considérée.

Afin de construire notre carte de cheminement nous nous basons essentiellement sur les Surfaces Navigables $Ns(O, M_k)$ qui sont associée à l'objet O considéré. Ces Surfaces Navigables défissent l'ensemble des configurations où l'agent est capable de se tenir sur l'objet au regard de ses capacités de déplacement. En échantillonnant une configuration appartenant à une Surface Navigable $Ns(O, M_k)$, nous sommes donc certain que cette configuration appartient à C-free mais également qu'elle est réalisable en utilisant la capacité de déplacement M_k. En se basant sur cette définition, nous construisons donc la carte de cheminement associée à un objet O de la manière suivante :

1. Des configurations c sont tout d'abord tirées aléatoirement dans les Surfaces Navigables associées à l'objet. Nous avons donc par définition :

$$c \in \bigcup_k Ns(O, M_k)$$

2. Chaque configuration c échantillonnée est ensuite annotée de l'ensemble $\mathcal{M}(c)$ des capacités de déplacement C_m réalisables en c :

$$\mathcal{M}(c) \ = \{M_k \mid c \in Ns(O, M_k)\}$$

3. Deux configurations c_1 et c_2 sont ensuite connectées dans la PRM à l'aide d'un arc $e \ = \ (c_1, c_2)$ si :
- ces deux configurations partagent au moins une capacité de navigation M_k commune, c'est à dire ssi :

$$\mathcal{M}(c_1) \cap \mathcal{M}(c_2) \neq \emptyset$$

- un arc (c_1, c_2) reste dans l'une des Surfaces Navigables $Ns(O, M_k)$ commune à c_1 et c_2 et qu'il n'entre pas en collision avec le Volume Interdit associé à l'objet O et la capacité de déplacement M_k :

$$\exists M_k \in (\mathcal{M}(c_1) \cap \mathcal{M}(c_2)), (c_1, c_2) \in Ns(O, M_k) \ \wedge \ (c_1, c_2) \cap V_f(O, M_k) = \emptyset$$

La carte de cheminement ainsi définie va permettre à un agent d'avoir une représentation de l'espace de navigation local et de l'ensemble des possibilités de déplacement

offertes. Toutefois, certains arcs de la carte de cheminement vont, lors de la planification de trajectoire, permettre d'utiliser plusieurs capacités de déplacement. Afin de faire en choix entre ces différentes capacités de déplacement, nous définissons deux coûts associés aux déplacements :

- Un coût α_{M_k} est associé à chaque capacité de déplacement M_k pour permettre à l'agent de choisir les capacités de déplacement les moins coûteuses.
- Un coût de transition $\beta_{(M_k,M_j)}$ est associé à chaque transition d'une capacité M_k à M_j. Ce coût supplémentaire permet de limiter le nombre de changement de capacité de déplacement de l'agent au cours de sa navigation et d'obtenir ainsi un comportement plus constant.

Ces coûts sont définis pour un agent. Toutefois, afin de prendre en compte différentes surfaces, il peut être possible de définir différents coûts pour différentes surfaces de navigation. La prise en compte de ces coûts au cours de la planification sera expliquée plus tard dans le chapitre, lors du détail de l'algorithme de planification locale.

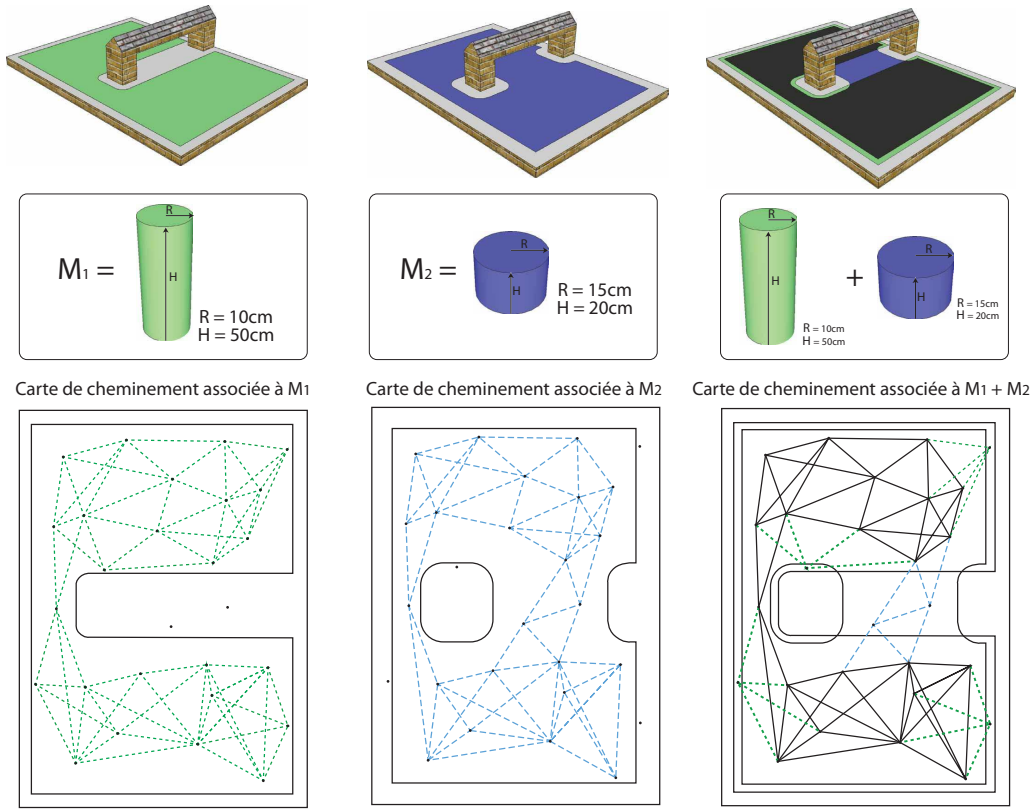


Figure 4.1 – Construction d'une carte de cheminement prenant en compte les deux capacités de déplacement M_1 et M_2 et en utilisant les Surfaces Navigables associées.

La figure 4.1 présente un exemple de construction de carte de cheminement se basant sur deux capacités de déplacement : une marche debout M_1 et une marche accroupie M_2. L'agent utilisant la capacité de marche debout est représenté à l'aide d'un cylindre haut (H=50cm) mais assez étroit (R=10cm) tandis que la marche accroupie

permet d'avoir une hauteur de l'agent limitée (H=20cm) mais occuper une zone plus large (R=15cm). Le mouvement debout permet donc de s'approcher plus des bords des objets mais est plus sensible aux contraintes de hauteur tandis que le déplacement accroupi permet de passer dans des endroits limités en hauteur. Afin de construire la carte de cheminement, des configurations, représentées par des points noirs sur les schémas, sont tout d'abord tirées aléatoirement dans $Ns(O, M_1) \cup Ns(O, M_2)$. Ces configurations sont ensuite connectées les unes aux autres à l'aide d'arcs. Ces arcs correspondent à une extrapolation linéaire entre les deux configurations choisies. La carte de cheminement verte, respectivement bleue, est associée à M_1, respectivement M_2. Elle représente l'ensemble des arcs échantillonnés permettant de relier deux configurations où l'agent peut se tenir à l'aide de la capacité de déplacement M_1, respectivement M_2. Afin de limiter la complexité de la carte de cheminement, chaque nœud est connecté à ses k plus proches voisins, ici $k = 4$. La carte de cheminement finale associée à l'objet correspond à la fusion des deux cartes de cheminement précédentes. Dans cette carte de cheminement, les arcs noirs sont navigables à l'aide des deux capacités de déplacement tandis que les verts sont navigables seulement en utilisant la capacité M_1 et les bleus seulement à l'aide de M_2. L'annotation des nœuds de la carte de cheminement permet ainsi de capturer l'ensemble de la topologie et de considérer les deux capacités de déplacement de l'agent dans une carte de cheminement unique.

Les environnements considérés étant dynamiques, la configuration topologique est amenée à évoluer au cours du temps et les relations topologiques associées, dont les obstructions, vont donc évoluer de la même manière. Il n'est donc pas possible, au moment de la création de la carte de cheminement, de déterminer de manière précise quels seront les arcs et nœuds de la PRM qui seront obstrués par des obstacles lors de la planification. De plus les obstacles dynamiques vont se déplacer au cours de la simulation et pendant la navigation de l'agent. Ces changements vont donc devoir être pris en compte durant la planification pour que l'agent puisse anticiper toutes ces modifications de la topologie. Nous allons maintenant voir comment, en utilisant les cartes de cheminement que nous venons de définir, nous permettons d'anticiper les modifications de la topologie pendant la planification de chemin de l'agent.

1.2 Anticipation de l'évolution des obstructions locales dans le domaine spatio-temporel

La planification locale a pour but de générer une trajectoire permettant à l'agent d'éviter les obstacles, statiques ou mobiles, présents autour de lui. De plus, cette planification doit également gérer l'adaptation de posture de l'agent face aux différentes contraintes environnementales en utilisant l'information contenue dans les cartes de cheminement. Afin d'anticiper le déplacement des éléments dynamiques dans l'environnement, nous effectuons cette planification locale dans le domaine spatio-temporel. La prise en compte de la dimension temporelle pendant la planification nous permet de générer une trajectoire anticipant le déplacement des éléments de l'environnement.

Les éléments créant de potentielles obstructions sur une Surface Navigable sont identifiés par des relations d'obstruction dans le Graphe Topologique. Afin d'éviter ces éléments au cours de la navigation, nous les représentons dans le domaine spatio-temporel en anticipant leurs déplacements. Différentes représentations des obstacles dynamiques peuvent être définies dans le domaine spatio-temporel [vdBFK06]. Afin de définir ces obstacles spatio-temporels, la position des éléments est extrapolée dans le temps en fonction de la connaissance dont dispose le système sur ces éléments. Une connaissance précise du mouvement d'un obstacle permettra une extrapolation précise de sa trajectoire tandis qu'un manque de connaissance va amener à devoir faire des hypothèses et des approximations lors de l'extrapolation de cette trajectoire. Différents exemples d'obstacles spatio-temporels sont montrés dans la figure 4.2. En fonction des connaissances à disposition, quatre types de volumes spatio-temporels peuvent être définis :

- Un élément statique est représenté par une extrapolation de sa géométrie le long de l'axe temporel comme le montre la figure 4.2(a).
- Un élément dynamique va pouvoir être représenté par différents volumes spatio-temporels :
  - Un cône bornant le déplacement maximum de l'obstacle, comme le montre la figure 4.2(b). L'objet est alors représenté par son cercle englobant et ce cercle englobant est extrapolé selon l'axe temporel et agrandi en fonction de la vitesse maximale de déplacement estimée, ou connue, de l'élément. Ce volume permet de représenter, en utilisant seulement une vitesse de déplacement maximale, une estimation au pire du déplacement de l'objet lorsque l'on ne dispose que de peu d'informations.
  - Une extrapolation linéaire de la géométrie de l'objet en utilisant la vitesse et la direction de déplacement de l'agent comme c'est le cas dans la figure 4.2(c). Cette approximation linéaire permet alors une meilleure anticipation du déplacement futur de l'objet à partir de sa configuration courante au moment de l'extrapolation.
  - Une extrapolation exacte dans le cas où la connaissance concernant le déplacement de l'objet est totale comme dans le cas de la figure 4.2(d). Le déplacement de l'objet est alors totalement défini a priori et peut donc être représenté finement dans le domaine spatio-temporel.

Dans notre contexte applicatif, nous n'avons pas accès à une information précise concernant les déplacements des obstacles. Toutefois, les déplacements des objets, bien que non connus a priori, sont observables au fil de la simulation.

L'observation de la vitesse et de la direction de déplacement des objets va nous permettre de classer ces obstacles en deux groupes :

- Les obstacles statiques, qui ont une vitesse nulle et ne se déplacent pas
- Les obstacles dynamiques, qui vont se déplacer pendant la navigation de l'agent

Pour les obstacles statiques, la construction du volume spatio-temporel revient à faire une extrapolation linéaire de la position de l'objet avec une vitesse de déplacement nulle. Concernant les obstacles dynamiques plusieurs choix sont possibles. Le manque de connaissance a priori sur le mouvement des objets ne nous permet pas d'utiliser une extrapolation exacte du déplacement de l'objet. De plus, une extrapolation *au*

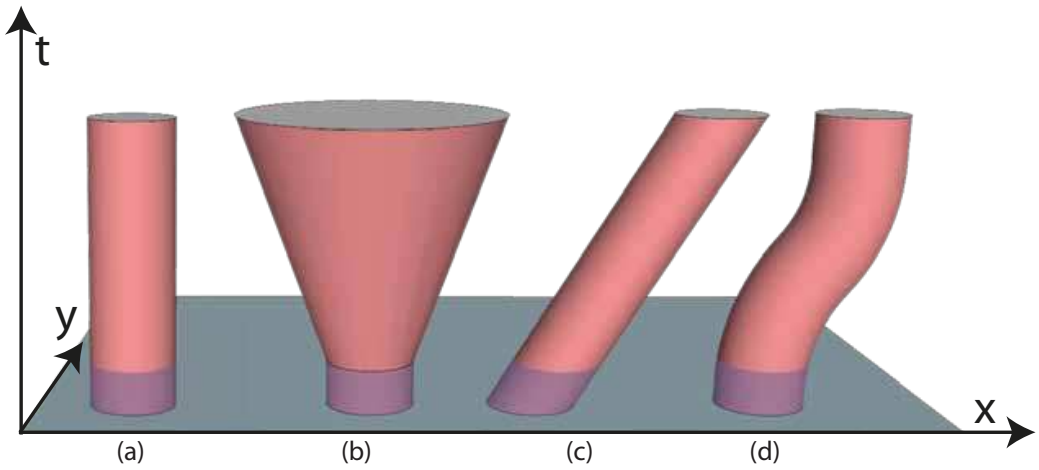


Figure 4.2 – Représentation des obstacles dans le domaine spatio-temporel. (a) : Obstacle statique. (b) : Obstacle dynamique dont la vitesse est bornée. (c) : Extrapolation linéaire de la trajectoire de l'obstacle. (d) : Obstacle dynamique dont les déplacements sont entièrement connus a priori.

pire bornée par une vitesse maximale va invalider de nombreuses configurations ce qui va sur-contraindre l'espace de recherche et potentiellement bloquer la planification même si des solutions viables existent. Nous représentons donc nos volumes spatio-temporels à l'aide d'une extrapolation linéaire. Cette extrapolation linéaire se base sur deux paramètres :

- Le vecteur vitesse de déplacement de l'objet, v_{ext}
- Un délai d'anticipation, t_{ext}

L'environnement étant observable, lors de la planification l'observation d'un objet va permettre de déterminer son vecteur vitesse de déplacement v_{ext}. Ce vecteur nous permet ainsi de faire l'extrapolation linéaire de la position de l'objet dans le domaine spatio-temporel. Cependant, cette extrapolation n'étant pas exacte, elle va engendrer une erreur d'approximation de plus en plus importante au fil du temps. En d'autres termes, l'erreur commise par l'extrapolation linéaire est d'autant plus importante que la période temporelle considérée est éloignée dans le futur. En effet, l'extrapolation linéaire dans un futur immédiat, d'1 ou 2 secondes par exemple, permet une approximation assez proche de la réalité tandis qu'une extrapolation linéaire pour un futur beaucoup plus éloigné, sur une période de 10 minutes par exemple, a plus de chances de se révéler fausse et d'introduire une erreur importante. Nous introduisons donc le délai d'anticipation t_{ext} afin de limiter cette erreur. Ce délai représente la capacité de l'agent à anticiper l'évolution de son environnement. Lors de la planification de chemin à un temps t, le déplacement des objets sera donc extrapolé pour la période $[t, t + t_{ext}]$. Au-delà de ce délai t_{ext}, nous considérons que l'erreur introduite par l'extrapolation linéaire est trop importante. Au delà du délai t_{ext}, nous considérons donc que nous ne possédons pas assez d'informations sur le déplacement de l'obstacle et donc que son déplacement ne peut pas être anticipé. Ne pouvant pas anticiper la position de l'obstacle dynamique celui-ci n'est donc pas pris en compte dans la planification pour un délai supérieur à t_{ext}. Ce choix introduit de nouvelles erreurs puisque l'on ne prend pas en compte un obstacle

présent lors de la planification. Toutefois, cette solution permet tout d'abord de refléter l'incapacité de l'agent à anticiper avec trop d'avance des déplacements d'obstacles, ce qui est généralement également le cas dans le monde réel. De plus, cette solution permet de réduire le nombre d'obstacles à prendre en compte dans la planification de trajectoire au-delà de t_{ext}. Cette propriété permet d'augmenter les performances de la planification en diminuant les tests de collision à effectuer. Ainsi, au-delà de t_{ext} seul l'évitement des obstacles statiques, dont la position est connue de manière précise, sera anticipée.

1.3 Planification locale dans le domaine spatio-temporel

Le rôle de la planification locale est de calculer une trajectoire pour l'agent à travers la Surface Navigable associée à un objet. Cette planification de trajectoire se base sur les cartes de cheminement locales définies dans la section 1.1 afin de rechercher des solutions de navigation. De plus, cette recherche est effectuée dans le domaine spatio-temporel afin d'anticiper le déplacement des obstacles au cours de la navigation. Deux cas de planifications locales vont se présenter :

- La cible globale de la planification appartient à l'objet en question et la planification de chemin locale va alors avoir pour but de déterminer une trajectoire de l'agent permettant de rejoindre cette cible.
- La cible globale est localisée dans la carte de cheminement d'un autre objet de l'environnement. La planification de chemin locale va alors avoir pour rôle d'identifier une trajectoire permettant à l'agent d'effectuer une transition vers un autre objet et se rapprocher ainsi de la cible.

Dans le premier cas, l'agent cherche à atteindre directement la cible globale à travers la carte de cheminement locale. Dans le second cas, comme nous le verrons dans la section 2, l'algorithme de planification globale va fournir à la planification locale plusieurs cibles intermédiaires permettant à l'agent d'effectuer une transition de l'objet courant vers l'objet suivant. La planification de chemin locale sera effectuée de manière similaire dans les deux cas, à la différence près du nombre de cibles que l'agent va chercher à atteindre.

Afin de déterminer une trajectoire libre pour l'agent, nous utilisons un algorithme de type A* pour explorer la carte de cheminement locale à la recherche d'une solution libre de toute collision dans le domaine spatio-temporel [HNR68]. Connaissant la position des sommets explorés et la position des cibles à atteindre, l'heuristique utilisé dans notre A* estime la distance minimale à parcourir pour atteindre la cible la plus proche. De plus, connaissant la vitesse maximale de déplacement associé à une capacité de l'agent, nous déterminons une estimation du délai nécessaire pour atteindre une cible. Lorsque la planification locale a pour but de trouver une trajectoire permettant d'atteindre une autre surface, plusieurs cibles intermédiaires peuvent être fournies à la planification locale. Afin de permettre une planification de chemin vers des cibles multiples, nous étendons cet algorithme de recherche en A* à cibles multiples.

Pendant l'exploration du A*, les nœuds visités sont annotés de plusieurs paramètres :

- la capacité de déplacement, M_e, que l'agent a utilisé pour accéder au nœud ;
- le temps de parcours, $t_{parcours}$, représentant le délai minimal nécessaire à l'agent pour atteindre au nœud depuis la source de l'exploration ;
- le coût, λ, du parcours de chemin depuis la source de la planification jusqu'au nœud et qui est proportionnel au temps de parcours ;
- l'heuristique, $t_{heuristique}$, qui est le temps estimé minimal pour atteindre l'une des cibles de la planification.

Lors de l'exploration, le candidat possédant le meilleur coût estimé ($\lambda + t_{heuristique}$) est choisi dans la file d'attente des nœuds ouverts et ses successeurs dans la carte de cheminement sont visités successivement.

Lorsqu'un nœud ouvert c_1 est visité par l'algorithme de planification, nous commençons par vérifier pour chaque successeur c_2 du nœud c_1 la validité de l'arc $e = (c_1, c_2)$. La validité de cet arc est vérifiée dans le domaine spatio-temporel afin de prendre en compte le déplacement des obstacles dynamiques. Au début de la planification locale, l'ensemble des Volumes Interdits se comportant en obstacles vis-à-vis de la Surface Navigable considéré est donc récupéré dans le Graphe Topologique et stocké dans un ensemble $\mathcal{E}_{obs}$. La validité de l'arc e est ensuite vérifiée de la manière suivante :

1. Le temps mis pour accéder à c_1, $t_{parcours}$, est récupéré ;
2. Pour chaque capacité de déplacement M_k permettant de naviguer sur l'arc, le temps de parcours de l'arc $t_{(e,M_k)}$ est calculé ;
3. Le temps d'arrivée en c_2 est alors estimé par $t_{c_2,M_k} = t_{parcours} + t_{(e,M_k)}$;
4. La validité de l'arc est ensuite testée pour chaque obstacle O_{M_k} dans $\mathcal{E}_{obs}$. Deux cas sont à distinguer :
   - *Si O_{M_k} est un obstacle statique :*
     - la position de O_{M_k} ne change pas au cours du temps. Un test de collision est donc effectué le long du segment $[c_1, c_2]$ à l'aide d'un lancer de rayon dans l'espace des configurations. Si aucune interaction n'est détectée entre ce rayon et O_{M_k}, alors l'arc est validé pour la capacité M_k. Dans le cas contraire, la navigation à l'aide de M_k est interdite car l'arc est obstrué par l'obstacle.
   - *Si O_{M_k} est un obstacle dynamique*, sa position va être modifiée pendant la navigation et deux cas vont se présenter :
     - $t_{parcours} > t_{ext}$: Le temps pour arriver à l'arc est supérieur au délai d'extrapolation fixé. L'obstacle O_{M_k} n'est donc pas pris en compte par la planification.
     - $t_{parcours} < t_{ext}$: La position de l'obstacle O_{M_k} est extrapolée afin de tester la validité de l'arc. La validation de l'arc peut être effectuée en extrapolant la position de O_{M_k} sur la période $[t_{parcours}\ ,\ t_{c_2,M_k}]$. Toutefois, afin d'alléger les calculs, nous effectuons l'opération inverse en projetant l'arc dans le repère local de l'obstacle O_{M_k} et d'origine $p(O_{M_k})$. Un exemple de cette projection est montré dans la figure 4.3. Les projetés c_1^{proj} et c_2^{proj} de c_1 et c_2 dans le repère local de O_{M_k} sont définis par :

$$c_1^{proj} = c_1 - v_{ext} * t_{parcours} - p(O_{M_k})$$

$$c_2^{proj} = c_2 - v_{ext} * t_{c_2,M_k} - p(O_{M_k})$$

Le projeté c_1^{proj} extrapole ainsi la position relative de c_1 par rapport à O_{M_k} au temps $t_{parcours}$. De la même manière, c_2^{proj} extrapole la position relative de c_2 vis-à-vis de O_{M_k} à t_{c_2,M_k}. La validité de l'arc est testée en lançant le rayon $[c_1^{proj}\ ,\ c_2^{proj}]$ dans le moteur de détection de collisions. Si une interaction est détectée, alors l'arc e est libre de toute collision vis-à-vis de O_{M_k} pendant la période $[t_{parcours}\ ,\ t_{c_2,M_k}]$. À l'inverse, si une collision est détectée, alors la navigation n'est pas possible sur e car l'arc est obstrué.

5. Dès qu'une interaction entre l'arc e et un obstacle O_{M_k} est détectée, l'arc est interdit à la navigation pour la capacité de navigation M_k et le reste des obstacles n'a pas besoin d'être testé. Si l'arc e est valide pour au moins une capacité M_k alors il est conservé et son coût estimé. Si aucune capacité de navigation ne permet de valider cet arc, alors il n'est pas utilisé pour la navigation. Toutefois cet arc est conservé puisqu'il pourra être de nouveau valide lors de futures planifications.

Lorsqu'un arc e est validé, un coût de parcours lui est alors associé. Le coût λ_e associé à cet arc e est calculé seulement pour les capacités de déplacement valides pour cet arc. Pour chaque capacité M_k valide, ce coût de parcours dépend de trois paramètres :

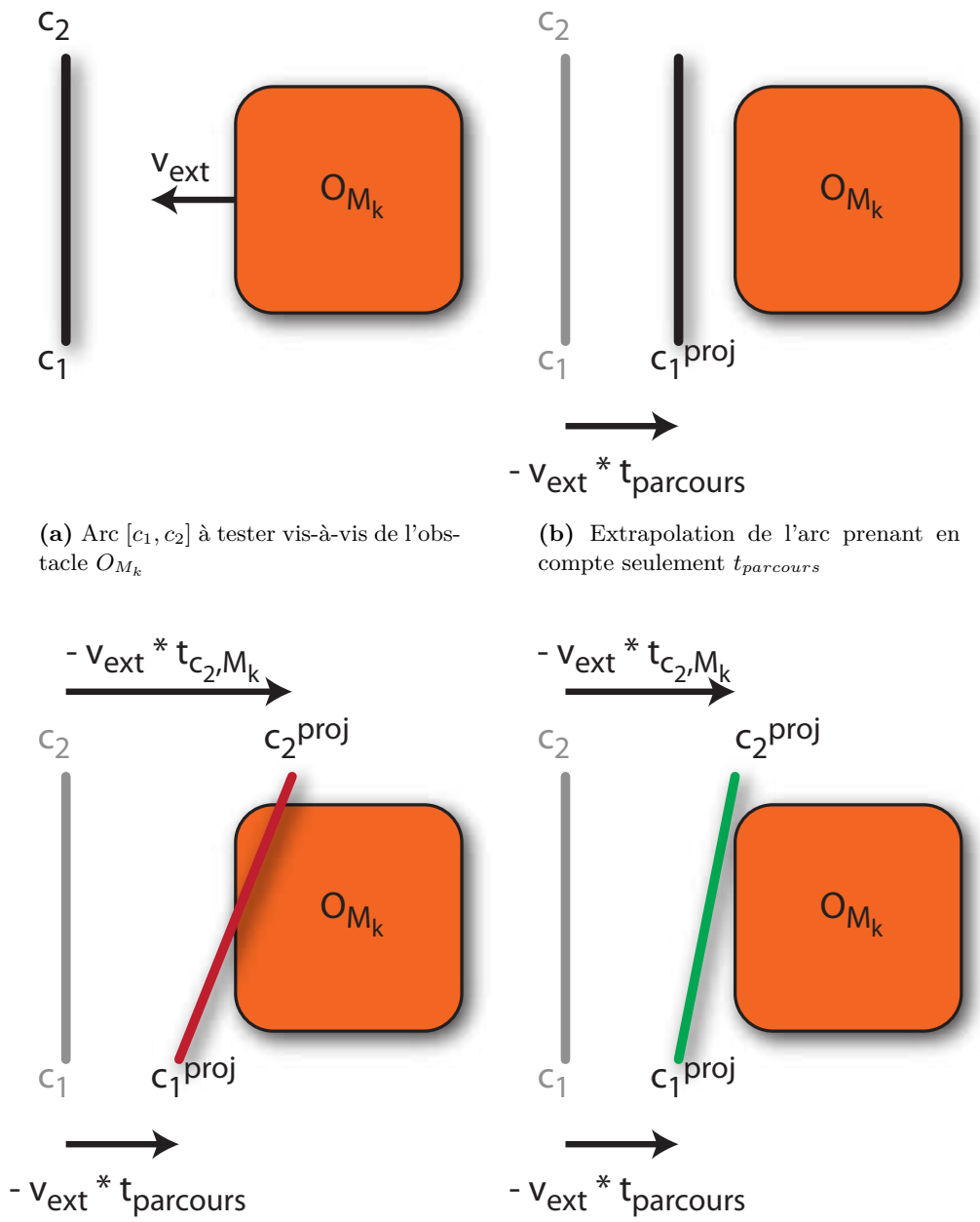
- Le temps de parcours de l'arc $t_{(e,M_k)}$ en utilisant la capacité de déplacement M_k ;
- Le coût α_{M_k} associé à la capacité de déplacement M_k utilisée par l'agent pour parcourir l'arc ;
- Le coût de transition $\beta_{(M_e,M_k)}$ représentant un changement de capacité de déplacement de M_e à M_k au début de la navigation sur l'arc.

Le temps de parcours de l'arc $t_{(e,M_k)}$ permet de rechercher le chemin le plus rapide permettant d'atteindre une cible. Le coût α_{M_k}, comme décrit précédemment, permet à l'agent de favoriser certaines capacités de déplacement vis-à-vis d'autres. Ce coût permet par exemple de favoriser des mouvements simples et facilement réalisables pendant la planification plutôt que des déplacements complexes. Enfin, le coût $\beta_{(M_e,M_k)}$ permet de limiter le nombre de changement de capacité de déplacement durant la navigation sur une Surface Navigable car un surcoût est engendré à chaque changement. Afin de favoriser un comportement constant, ce coût sera de zéro si $M_e = M_k$, c'est-à-dire si l'agent conserve le même mouvement. Autrement le mouvement minimisant le coût de changement de posture sera choisi. En se basant sur ces différents paramètres, le coût global λ_e associé à l'arc e est donc défini par :

$$\lambda_e = min_{M_k}(\ \alpha_{M_k} * t_{(e,M_k)} + \beta_{M_e,M_k}\ )$$

Une fois la validité de l'arc e vérifiée et son coût de parcours λ_e calculé, les paramètres du successeur c_2 sont alors mis à jour de la manière suivante :

- la capacité de déplacement, M_e, correspond à la capacité M_k permettant le déplacement de coût minimal λ_e ;
- le coût, $\lambda(c_2) = \lambda(c_1) + \lambda_e$;
- le temps de parcours, $t_{parcours}(c_2) = t_{parcours}(c_1) + t_{(e,M_e)}$;
- l'heuristique, $t_{heuristique}$ est calculée en estimant le temps d'accès à la cible la plus proche.



(a) Arc $[c_1, c_2]$ à tester vis-à-vis de l'obstacle O_{M_k}

(b) Extrapolation de l'arc prenant en compte seulement $t_{parcours}$

(c) Extrapolation de l'arc avec prise en compte du temps d'arrivée en c_2 : détection d'une collision

(d) Extrapolation de l'arc avec prise en compte du temps d'arrivée en c_2 : navigation sur l'arc validée

Figure 4.3 – Validation d'un arc de la carte de cheminement locale. La validité est testée en projetant l'arc dans le repère local de l'obstacle tout en extrapolant le déplacement de cet obstacle pendant le parcours de l'arc.

Ces valeurs sont alors calculées pour l'ensemble des successeurs de c_1 et ceux-ci sont ajoutés à leur tour à la file d'attente de l'exploration jusqu'à ce qu'une cible soit atteinte. Si la cible atteinte est la cible globale de la planification alors la tâche de planification est terminée et la trajectoire est renvoyée à l'agent. Si cette cible est une configuration permettant la transition vers la Surface Navigable d'un autre élément, l'agent va pouvoir changer de surface de navigation. Dans le cas de liaisons périodiques, il est possible que l'agent reste sur place en attendant de détecter la liaison comme étant de nouveau valide. Une fois cette transition trouvée, la planification globale continue et les planifications locales sont effectuées sur les objets suivants dans le chemin.

Discussion

Lors de la planification locale, nous explorons la carte de cheminement à l'aide d'un algorithme de type A*. Toutefois, les algorithmes A* ne proposent pas d'explorer le même nœud plusieurs fois. Il n'est donc pas possible pour un agent de repasser plusieurs fois par la même configuration afin, par exemple, d'éviter un obstacle dynamique. D'autre part, il n'est pas non plus possible à l'agent de s'arrêter à une position fixe au cours de sa planification locale. Cet arrêt impliquerait de devoir explorer le même nœud à des temps différents. Il est possible de gérer de tels cas, toutefois la possibilité de passer plusieurs fois au même endroit ou d'attendre à une position de l'environnement vont créer une situation de planification qui ne sera plus bornée. Dans certaines situations, il sera par exemple possible que l'agent attende à un endroit et qu'il y reste bloqué, sans que la planification ne puisse ni trouver un chemin valide, ni renvoyer un échec à la requête de planification.

1.4 Réactivité de l'agent pendant la navigation

Comme nous l'avons vu précédemment, nous anticipons les déplacements des obstacles pendant la planification en effectuant une extrapolation linéaire de leur position au fil du temps. Cette extrapolation est limitée dans le temps et au-delà du délai d'anticipation t_{ext} ces obstacles dynamiques ne sont plus pris en compte. Toutefois l'extrapolation linéaire ainsi que le délai d'anticipation utilisé vont créer une erreur lors du calcul de la trajectoire puisque les éléments ne vont pas forcément suivre une trajectoire linéaire et que les éléments situés hors de la fenêtre d'anticipation vont potentiellement avoir un impact sur la trajectoire de l'agent. Afin de maintenir une trajectoire valide pendant la navigation et d'éviter les obstacles dynamiques, la validité de la trajectoire est vérifiée en permanence. De la même manière que pour la planification de chemin, chaque segment $[c_1, c_2]$ de la trajectoire sera testé par un lancer de rayon dans l'espace des configurations. Ce test prend également en compte une fenêtre d'anticipation t_{ext} afin d'alléger les calculs. Si une collision est détectée entre la trajection et un obstacle, une nouvelle planification locale est alors effectuée depuis la position courante de l'agent vers la(les) cible(s) utilisée(s) précédemment afin de trouver une trajectoire alternative. Si une nouvelle trajectoire valide est trouvée, l'agent se met à suivre la nouvelle solution, dans le cas contraire, l'échec de planification est remonté au niveau de la planification globale qui aura pour tâche d'identifier un nouveau chemin.

L'utilisation d'une PRM a pour conséquence de générer une trajectoire discontinue et bruitée pour l'agent. En suivant exactement cette trajectoire l'agent aura donc une navigation saccadée, suivant des segments de droites et présentant des rotations brusques entre ces segments. Afin de proposer une navigation plus homogène de l'agent, la trajectoire est lissée pendant la navigation. Ainsi, lors de la validation de la trajectoire, des tests sont effectués entre la position courante de l'agent et les différents points de la trajectoire qu'il est en train de parcourir. L'agent va ainsi se diriger en priorité vers le point de la trajectoire le plus éloigné de sa position actuelle et tel que la ligne droite reliant ces deux positions n'entre pas en collision avec des obstacles de l'environnement. La trajectoire est donc ainsi lissée au fur et à mesure de la navigation et permet d'avoir une navigation beaucoup plus fluide de l'agent dans l'environnement.

L'agent ayant besoin de replanifier une trajectoire locale en fonction des obstructions détectées, le choix de l'algorithme de planification local est donc important. Les algorithmes de type A* proposent de planifier le chemin depuis la source vers une destination tandis que des algorithmes, comme le Dynamic-A* ou D* [Ste95], proposent d'effectuer la planification dans le sens inverse : depuis la destination vers la source. Cette modification permet, dans le cadre de replanifications, de pouvoir réutiliser une partie des calculs précédemment effectués afin d'accélérer la recherche. Ceci n'est pas possible avec un algorithme de type A* puisque la source de la planification est modifiée dès que l'agent bouge. Toutefois, dans notre contexte d'application dynamique, nous avons besoin de connaître pendant la planification les temps de parcours permettant d'arriver au nœud exploré afin de pouvoir anticiper le déplacement des obstacles. Dans ce contexte il n'est donc pas possible d'effectuer une planification depuis la destination vers la source puisqu'il est impossible de savoir a priori à quelle date la destination sera atteinte. Une alternative possible au A* que nous utilisons serait de prendre un algorithme de type AD* tel que cela a été fait par Van den Berg afin d'éviter des obstacles dans un environnement dynamique [vdBFK06].

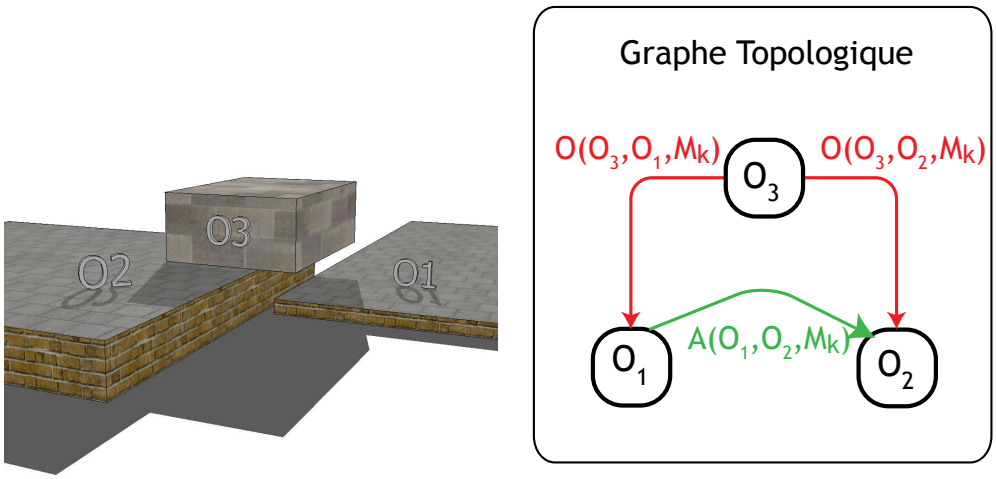
2 Planifier le déplacement de l'agent dans l'environnement dynamique

Nous allons nous focaliser maintenant sur les chemins nécessitant que l'agent navigue à travers plusieurs éléments. Le but de la planification globale est alors d'identifier la séquence d'éléments que l'agent va devoir traverser lorsque la source de la planification et la destination n'appartiennent pas au même élément. Le calcul de ce chemin global repose alors sur les accessibilités existant entre les objets de l'environnement. Le Graphe Topologique défini dans le chapitre précédent nous donne accès à une vision globale de la topologie dynamique de l'environnement. Ainsi, en utilisant cette représentation temporelle de l'environnement, nous déterminons le chemin global que va devoir emprunter l'agent au cours de sa navigation.

2.1 Raffinement des relations topologiques

Dans le but de déterminer la séquence d'objets à traverser pour atteindre une cible fixée, il est nécessaire de connaître les accès possibles entre les objets de l'environnement. Le Graphe Topologique nous permet d'avoir un accès direct à l'information d'accessibilité et d'identifier facilement les relations topologiques entre les éléments. Ce graphe permet également de planifier un chemin topologique en prenant en considération seulement les relations d'accessibilité. Nous désignons ici par chemin topologique la séquence d'objets à traverser pour atteindre une surface cible. Toutefois, comme nous avons remarqué au cours de la section 2.1.1 du chapitre précédent, certaines relations d'accessibilité représentées dans le Graphe Topologique peuvent se révéler être de "fausses accessibilités". L'information du Graphe Topologique doit donc être dans un premier temps raffinée afin de vérifier la validité réelle de ces relations.

Les arcs du Graphe Topologique représentant des relations entre des paires d'objets, l'impact des relations sur les autres relations n'est donc pas pris en compte. Il est ainsi possible que certaines relations d'accessibilité identifiées soient invalidées par des relations d'obstructions. L'exemple de la figure 4.4 illustre ce problème. Dans cet exemple, un objet O_3 obstrue totalement l'accès détecté entre les deux éléments O_1 et O_2. L'accessibilité $A(O_1, O_2, C_k)$ est tout de même rapportée dans le Graphe Topologique de même que les obstructions $O(O_3, O_1, C)$ et $O(O_3, O_2, C)$. Le raffinement de ces fausses accessibilités a donc pour objectif de lever les incertitudes sur ces cas de figure particuliers.

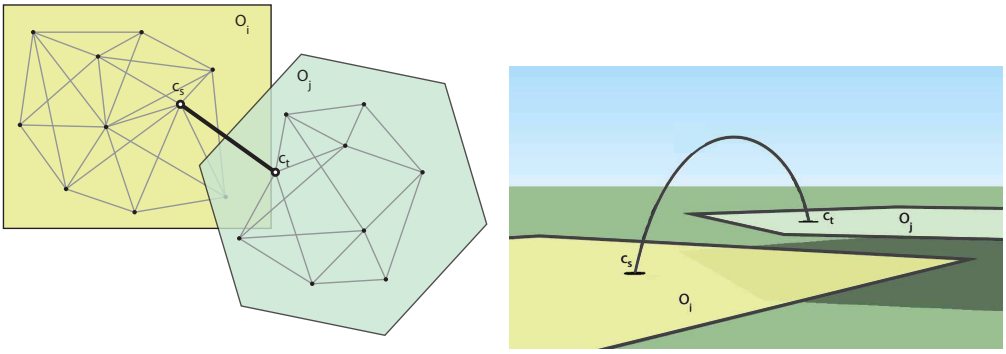


(a) Fausse accessibilité de O_1 à O_2

(b) Graphe Topologique en présence de O_3

Figure 4.4 – 4.4a : L'accessibilité détectée est invalidée par O_3 créant une fausse accessibilité. 4.4b : Graphe Topologique associé à l'environnement présenté en 4.4b.

Afin de valider les relations d'accessibilité, nous raffinons donc l'information apportée par le Graphe Topologique en échantillonnant des trajectoires d'accès entre les deux Surfaces Navigables considérées. Dans le cadre de notre exemple des trajectoires potentielles vont donc être échantillonnées vers O_2 depuis O_1. Si au moins l'une des trajectoires échantillonnées est valide, c'est-à-dire qu'elle n'entre en collision avec aucun des Volumes Interdit obstruant O_1 ou O_2, alors la relation d'accessibilité est validée. Dans le cas contraire la relation est invalidée du Graphe Topologique puisque qu'aucune trajectoire valide ne permet l'accès en question.



(a) Vue de dessus **(b)** Vue de trois-quart

Figure 4.5 – Un lien d'accessibilité est construit en échantillonnant une trajectoire de saut. Cette trajectoire est une trajectoire balistique ayant comme origine la configuration c_s et destination la configuration c_t. De plus elle est contrainte de passer par la configuration intermédiaire c_r appartenant à $V_a(O_i, M_k)$.

Par définition, les différentes capacités de déplacement permettant à l'agent de transiter entre différents objets sont associées, dans notre cas, à des mouvements non contraints au sol, tels que des sauts. Comme nous l'avons vu dans le chapitre 2 lors de la caractérisation de l'agent, ces mouvements sont représentables à l'aide de trajectoires balistiques. Au regard de la définition d'une accessibilité, un objet O_j est accessible depuis un objet O_i si et seulement si $V_a(O_i, M_k) \cap Ns(O_j, M_k) \neq \emptyset$ où M_k représente la capacité de déplacement correspondant à un saut. Afin de valider cette relation et d'éviter des trajectoires en collision avec des Volumes Interdits, nous échantillonnons un ensemble de sauts, et donc de trajectoires paraboliques, de O_i vers O_j. Un exemple d'échantillonnage est présenté dans la figure 4.5 entre les deux éléments O_i et O_j, O_j étant accessible depuis O_i. Cet échantillonnage est réalisé de la manière suivante :

1. Une configuration cible c_t appartenant à la carte de cheminement associée à $Ns(O_j, M_k)$ et à l'ensemble $V_a(O_i, M_k) \cap Ns(O_j, M_k)$ est sélectionnée.
2. Une configuration source c_s est sélectionnée dans la carte de cheminement associée la Surface Navigable $Ns(O_i, M_k)$. Afin de trouver un bon candidat pour une transition, la configuration c_s est sélectionnée en récupérant la configuration libre de collisions la plus proche de c_t.
3. Comme le montre la vue de dessus de la figure 4.5, les configurations c_s et c_t sont ensuite projetées dans le plan horizontal (X, Y) afin de calculer la distance

euclidienne 2D d_{XY} entre ces configurations. Nous posons dans un premier temps les notations suivantes :

$$c_s = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} \text{ et } c_t = \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix}$$

d_{XY} est donc défini par :

$$d_{XY} = \sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}$$

4. Nous cherchons maintenant à déterminer le polynôme de degré 2 correspondant à la trajectoire de l'agent permettant de lier les configurations c_s et c_t. Ce polynôme va définir une trajectoire restant dans un plan vertical. Partant de cette observation nous cherchons donc à caractériser un polynôme de la forme :

$$P(X) = \alpha.X^2 + \beta.X + \gamma$$

où α, β et γ sont les paramètres du polynôme qui doivent être défini, X correspond à l'abscisse de l'agent pendant la trajectoire et $P(X)$ défini l'altitude de l'agent le long de cette trajectoire. Nous définissons le saut de l'agent de la même manière que le lancer d'un projectile. Les seules forces extérieures que subit l'agent se résument à son poids $\overrightarrow{P}$ et les forces de frottements de l'air sont négligeables. D'après la seconde loi de Newton, l'accélération $\overrightarrow{a}$ que subit le projectile est donc égale à la gravité $\overrightarrow{g}$ qui se résume ici à une valeur de $-g$ dans le plan vertical. Nous obtenons donc la valeur suivante pour l'accélération : $a = -g$. En intégrant cette équation et en ajoutant les condition initiales, nous obtenons :

$$V_z(X) = -g.X + v_z^0$$

où v_z^0 est la vitesse initiale de l'agent selon l'axe vertical

Par une nouvelle intégration nous obtenons :

$$P(X) = -\frac{1}{2}g.X^2 + v_z^0.X + \gamma$$

L'identification des coefficients nous permet ainsi de déduire $\alpha = -\frac{1}{2}g$ et $\beta = v_z^0$. Nous posons de plus comme condition initiale que la configuration source c_s est l'origine de la trajectoire, de cette manière nous définissons $P(0) = z_s$. Il en découle donc :

$$P(0) = \gamma = z_s$$

La distance d_{XY} représentant la distance entre les configurations c_s et c_t, nous en déduisons : $P(d_{XY}) = z_t$. Il en découle :

$$P(d_{XY}) = -\frac{1}{2}g.(d_{XY})^2 + v_z^0.d_{XY} + z_s = z_t$$

$$\text{d'où } v_z^0 = \frac{z_t - z_s}{d_{XY}} + \frac{1}{2}g.d_{XY}$$

La trajectoire ainsi définie a finalement pour équation :

$$P(X) = -\frac{1}{2}g.X^2 + (\frac{z_t - z_s}{d_{XY}} + \frac{1}{2}g.d_{XY}).X + z_s$$

5. La validité de la trajectoire obtenue est ensuite testée. Cette trajectoire est valide si et seulement si :
   - les vitesses d'impulsion v_{dec} et d'atterrissage v_{att} sont en accord avec les propriétés du saut qui est associé à la capacité M_k. Cette vérification est faite à l'aide d'une comparaison avec la vitesse v_z^0 calculée et en calculant la vitesse d'atterrissage en $P(d_{XY})$
   - la trajectoire n'entre pas en collision avec l'un des Volumes Interdit obstruant O_i ou O_j pour la capacité M_k

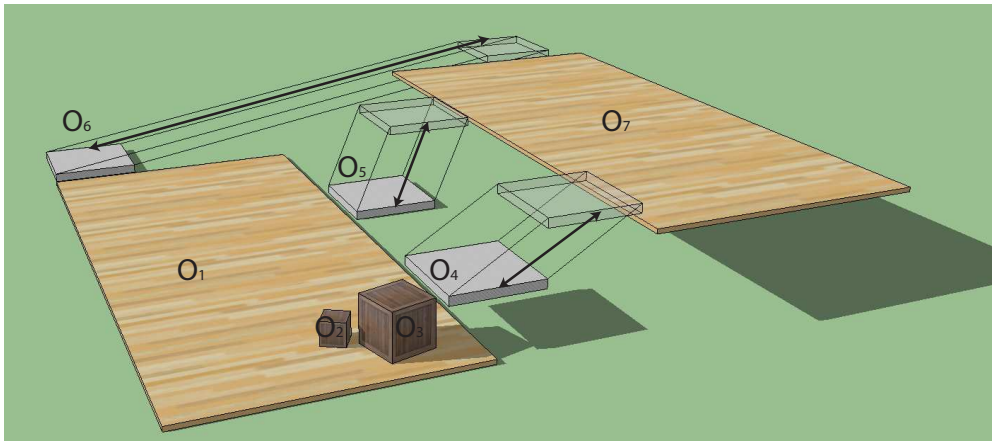
Si la trajectoire échantillonnée est validée, alors elle est stockée au niveau de la relation d'accessibilité dans le Graphe Topologique ce qui permettra de la récupérer lors des planifications locales. De plus, nous allouons un budget-temps à chaque pas de simulation afin d'échantillonner ces nouvelles trajectoires de transition. Ce budget-temps permet de limiter le coût introduit par l'échantillonnage de ces trajectoires en le répartissant sur plusieurs pas de temps. Ce budget-temps est ensuite réparti entre les différentes relations d'accessibilité valides au moment de la détection.

2.2 Calcul et optimisation du chemin topologique

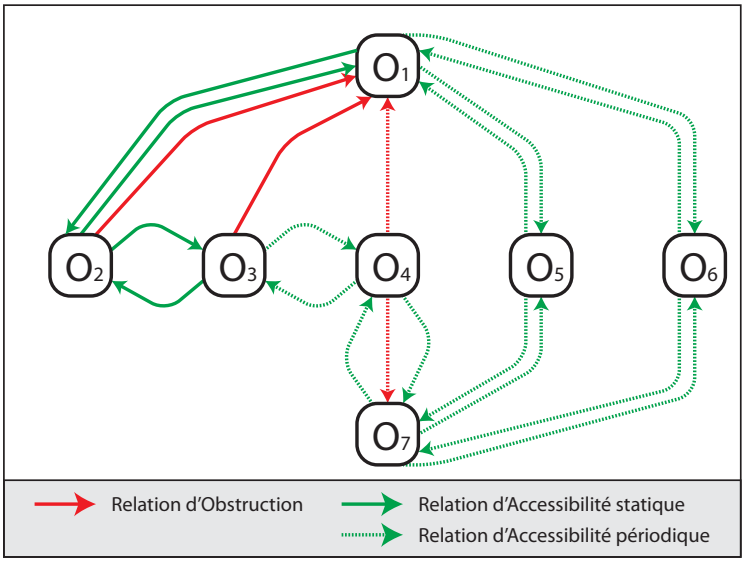
La planification globale a pour but d'identifier la séquence d'objets permettant à l'agent de rejoindre sa cible à partir de sa position courante. Afin d'identifier un chemin topologique différents critères peuvent être utilisés :

- minimiser le nombre d'objets dans le chemin topologique, ce qui favorise la recherche de chemins topologiques simples et les plus épurés possible en nombre d'éléments à traverser ;
- minimiser les délais d'attente de transition entre les objets dans le cas d'objets en mouvements en regardant les périodicités des relations identifiés ;
- minimiser les transitions à risques pour l'agent, ce qui va permettre à travers différents critères de déterminer le chemin le plus sûr pour la navigation de l'agent à travers l'environnement dynamique. La sécurité du chemin va pouvoir être déterminée par différents critères tels que la stabilité des relations topologiques ou leur périodicité.

Seules les relations d'accessibilité valides vont être utiles afin d'identifier un chemin topologique. Les relations d'obstruction, ainsi que les relations d'accessibilités invalidées par l'identification d'une fausse accessibilité dans le Graphe Topologique sont donc désactivées pendant la planification globale. La figure 4.6 montre un exemple d'environnement dynamique et de son Graphe Topologique associé. Dans cet environnement, les relations d'obstructions identifiées dans le graphe ne sont donc pas pris en compte dans l'identification d'un chemin global. Nous utilisons ensuite différents critères afin de filtrer plus précisément les relations du Graphe Topologique et de pondérer les arcs



(a) Environnement composé d'éléments statiques (O_1, O_2, O_3 et O_7) et de trois plateformes mobiles (O_4, O_5 et O_6).



(b) Graphe Topologique associé à l'environnement : les relations entre (O_1, O_2) et (O_2, O_3) sont statiques, les autres étant des liaisons périodiques

Figure 4.6 – Le but de la planification topologique est d'identifier une séquence d'objets à traverser afin d'atteindre une surface ciblée. L'environnement présenté en (a) et son Graphe Topologique associé (b) permettent d'identifier trois voies d'accès différentes entre O_1 et O_7. Le but de la planification topologique est de sélectionner le meilleur accès pour l'agent.

de ce graphe lors de l'exploration. Les différents paramètres que nous allons prendre en compte pour la navigation sont les suivants :

- Un coût fixe de passage sur un arc, ω ;
- La vitesse relative observées entre les objets concernés par la relation, V_{rel} ;
- La variation angulaire maximale observée pour la surface d'arrivée, $\theta = |\theta_{max} - \theta_{min}|$, où θ_{min} et θ_{max} sont les angles d'inclinaisons minimum et maximum qui ont été observés pour la surface ;
- Le temps moyen de validité de la relation d'accessibilité, t_{val} ;
- La périodicité de la relation détectée $\delta = t_{val} + t_{inval}$.

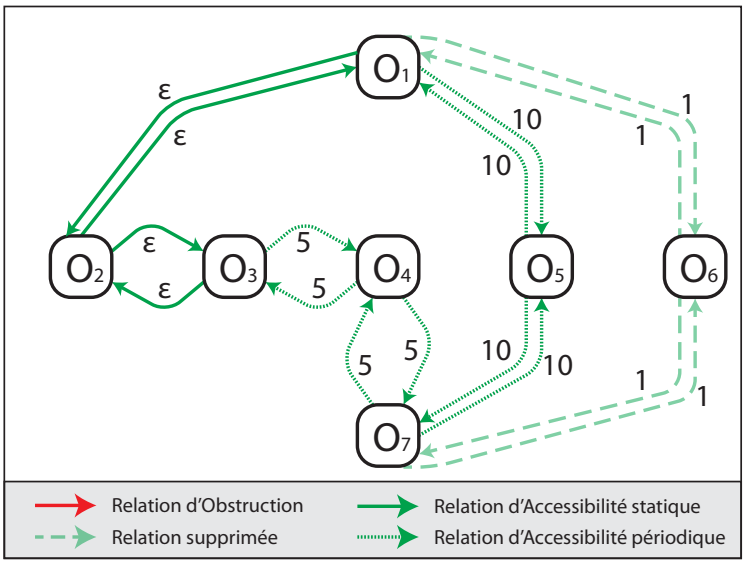


Figure 4.7 – Les relations d'accessibilités concernant deux objets dont les vitesses de déplacements relatives sont trop importantes sont enlevées. C'est le cas ici pour les relations entre (O_1, O_6) et (O_6, O_7).

Les paramètres de vitesse relative V_{rel} et de variation angulaire θ vont permettre de définir des conditions de sécurité pendant la navigation de l'agent. En effet, la transition entre deux objets de l'environnement va être d'autant plus dangereuse que la vitesse relative entre ces deux objets va être élevée, ou que le second objet va accélérer de façon brutale ou encore que cet objet va avoir une grande amplitude dans son inclinaison. La dangerosité créée par de telles relations vient du fait que des mouvements trop rapides vont être difficiles à anticiper pour l'agent. En essayant de naviguer entre deux Surfaces Navigables dont la vitesse relative V_{rel} est trop élevée, un agent pourra par exemple être amené à commencer son saut en anticipant la future relation topologique alors que la position de destination est en dehors de tout support au début de saut. De la même manière, un agent pourrait essayer de sauter vers une plateforme aux oscillations de roulis et de tangage amples et chaotiques. Dans ce cas, l'anticipation de l'atterrissage et des appuis sera difficile. Afin d'éviter de telles situations, des seuils

sont fixés pour déterminer des valeurs limite pour la vitesse relative V_{seuil}, l'accélération A_{seuil}, la vitesse angulaire θ_{seuil} et le temps moyen de validité de la relation t_{seuil} puis le Graphe Topologique est filtré une nouvelle fois. Les relations topologiques du graphe sont supprimées si au moins l'une de ces conditions de sécurité n'est pas respectée :

1. $V_{rel} > V_{seuil}$: la vitesse relative entre les objets est trop rapide pour que l'agent puisse naviguer entre les deux ;
2. $\theta > \theta_{seuil}$: l'inclinaison de la surface d'arrivée a une trop grande amplitude pour que l'agent puisse anticiper un atterrissage précis et sans risque ;
3. $t_{val} < t_{seuil}$: la période de validité est trop courte, l'agent n'aura pas le temps de l'anticiper convenablement et risque de sauter en retard et de ne pas atteindre la surface.

Ces seuils vont pouvoir être modifiés afin d'ajuster le comportement souhaité pour l'agent. Un agent acceptant les comportements à risque pourra par exemple avoir des seuils plus élevés qu'un agent préférant la sécurité des relations. Comme le montre la figure 4.7, ces différents critères nous permettent de filtrer de nouveau le Graphe Topologique en ôtant les relations qui ne vérifient les critères de sécurité de l'agent. Dans la figure d'exemple, les relations entre O_1 et O_6 et entre O_6 et O_7 sont supprimées du Graphe Topologique car la vitesse de déplacement de la plateforme navigable O_6 est trop élevée au regard des capacités de l'agent.

Une fois les relations filtrées, nous associons à chaque arc du Graphe Topologique un coût de passage. Ce coût va reposer sur les différents paramètres introduits précédemment. Ce coût sera ainsi défini comme suit :

$$\text{coût} = \omega + a_1.S + a_2.\delta$$

$$\text{où } S = e^{\alpha \frac{V_{rel}}{V_{seuil}} + \beta \frac{\theta}{\theta_{seuil}}}$$

Le calcul du coût prend donc en compte trois termes distincts. Le terme ω définit le coût associé à la transition entre deux éléments de l'environnement durant la navigation. Le second terme, S, est un critère de sécurité. Ce critère prend en compte la vitesse relative des éléments et l'oscillation des surfaces navigables afin d'évaluer si la transition entre les deux est stable ou dangereuse vis-à-vis de l'agent. Les paramètres de vitesse relative V_{rel} et d'oscillation θ sont normalisés à l'aide des valeurs seuils associées, V_{seuil} et θ_{seuil}. Les coefficients α et β sont ensuite définis pour représenter l'importance respective de la vitesse et de l'oscillation dans ce critère. Enfin, le terme δ représente, comme défini plus tôt, la périodicité de la relation. Une relation dont la périodicité sera plus courte sera donc favorisée par ce terme, afin que l'agent réduise ses temps d'attente. Les coefficients a_1 et a_2 permettent finalement de définir l'importance relative des critères de sécurité et de périodicité de la relation vis-à-vis de coût ω associé à la transition entre les éléments. Pour les relations statiques, les coefficients a_1 et a_2 seront fixés à 0 afin de ne pas prendre en compte ces paramètres caractérisant les relations dynamiques.

Le coût calculé est ensuite associé chaque arc. À la vue de ce coût et de nos différents critères, il est difficile de définir un heuristique admissible pour guider la recherche de chemin dans le Graphe Topologique. Ne possédant pas d'heuristique, nous utilisons donc un Dijkstra pour effectuer la recherche dans le Graphe. La prise en compte de la notion de sécurité et de la périodicité des relations permet toutefois de représenter différent comportement de l'agent. Un exemple de pondération des liens est montré par exemple dans la figure 4.7. Dans cet exemple, le coût fixe ω des relations a une valeur ϵ, a_1 a une valeur nulle et a_2 est fixé à une valeur de 1. Les relations dynamiques sont donc pondérées uniquement par leurs périodicités, ce qui va conduire l'agent à emprunter de préférence le chemin présentant des temps d'attentes moins important entre les surfaces où il sera amené à naviguer. En utilisant cette représentation, une planification effectuée sur l'environnement présenté dans la figure 4.6 va favoriser le chemin $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow O_4 \rightarrow O_7$ en préférant les relations statiques ou dont la périodicité est courte au détriment du chemin $O_1 \rightarrow O_5 \rightarrow O_7$ plus court en nombres d'éléments, mais plus long au regard de la périodicité des relations.

3 Planification hiérarchique

3.1 Association de la planification globale et locale

la planification globale dans un premier temps permet à l'agent de trouver un chemin entre des surfaces navigables déconnectées permettant ainsi la navigation dans un environnement complet et dynamique. Dans un second temps, la planification locale permet ensuite à l'agent d'identifier un chemin dans le domaine spatio-temporel au sein d'une surface navigable en anticipant les mouvements des obstacles afin de proposer une solution libre de collision et permettant les adaptations de posture de l'agent.

La planification hiérarchique repose sur ces deux planificateurs afin de proposer une solution de navigation complète à l'agent. Une requête globale est tout d'abord effectuée afin de déterminer le chemin topologique identifiant les surfaces navigables que l'agent doit traverser. Ce chemin topologique permet également de localiser les accessibilités que va utiliser l'agent durant sa navigation et de récupérer de ce fait les transitions échantillonnées pour chacune. Une planification de trajectoire locale est ensuite effectuée dans la surface navigable courante de l'agent. Les cibles locales de cette planification sont identifiées par les transitions échantillonnées vers la surface suivante du chemin topologique.

À la fin de sa navigation sur la surface locale, l'agent attend que la relation d'accessibilité soit valide avant d'effectuer sa transition. Une fois la validité de la relation avéré, l'agent utilise sa capacité de déplacement afin d'atteindre la surface de navigation suivante. Une nouvelle planification locale est alors effectuée afin de déterminer la nouvelle trajectoire à suivre. Cette planification hiérarchique permet ainsi de diviser le calcul du chemin en l'identification de multiples chemins locaux étalés dans le temps, lors de l'arrivée de l'agent sur la surface en question. Toutefois il est possible qu'un obstacle

empêche la navigation de l'agent sur une surface navigable en scindant par exemple la carte de cheminement locale en deux composantes connexes. L'échec de la planification locale doit donc être pris en compte au niveau de la planification hiérarchique afin de proposer une alternative de navigation.

3.2 Gestion d'une obstruction de la Surface Navigable

La planification hiérarchique permet de répondre à la majorité des requêtes de planification dans l'environnement. Toutefois, il est possible de se trouver dans une situation où la présence d'un élément va déconnecter la carte de cheminement d'un autre élément faisant potentiellement échouer la planification locale. La planification hiérarchique va donc devoir identifier ces cas afin de représenter ces différentes composantes connexes directement au niveau du Graphe Topologique et de les prendre en compte lors de la planification globale. L'exemple d'une telle situation est présenté dans la figure 4.8 où une planche posée sur le sol, bien qu'accessible, peut se comporter comme un obstacle pour la carte de cheminement associée au sol puisqu'elle obstrue des configurations sur le sol et sépare la carte de cheminement en deux composantes connexes.

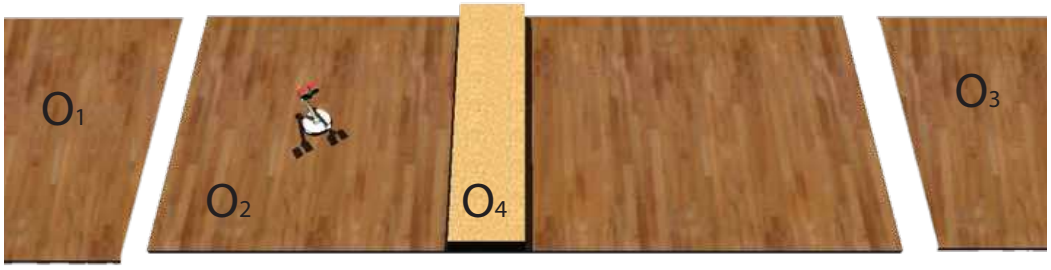


Figure 4.8 – Dans cet environnement, la surface Navigable de O_2 est obstruée par O_4 qui la sépare ainsi en deux composantes connexes. Afin que l'agent puisse naviguer vers la composante droite à partir de la composante gauche, nous utilisons la méthode de duplication.

Afin de pouvoir gérer ce cas particulier nous avons besoin de faire remonter l'information détectée localement à la planification globale pour faire ressortir cette propriété au niveau du Graphe Topologique. Le cas présent est susceptible d'apparaître lorsque qu'un ou plusieurs élément(s) présent(s) dans l'environnement obstruent une carte de cheminement locale et la séparent en différentes composantes connexes. Pour de meilleures performances, nous distinguons deux cas selon la nature statique ou dynamique de l'obstacle puisque l'impact sur la carte de cheminement locale sera différent. Dans le cas de l'obstacle statique, l'impact de l'objet est permanent tandis que dans celui de l'obstacle dynamique, cet impact n'est que temporaire. L'information concernant les éléments statiques peut donc être remontée de manière permanente dans le Graphe Topologique puisque cet impact ne sera pas remis en question par la suite. À l'opposé, l'impact d'un élément dynamique ne sera détecté qu'au moment de la planification et remonté au niveau du Graphe Topologique seulement à ce moment.

Que l'obstacle soit statique ou dynamique, la détection d'une déconnexion de la carte de cheminement entraîne une duplication de la surface navigable dans le Graphe Topologique. Afin d'effectuer cette duplication, cette déconnexion est tout d'abord notifiée au niveau du Graphe Topologique. Cette notification traduit que l'objet O_i représenté par un nœud N_i dans le Graphe Topologique est en fait constitué de plusieurs composantes connexes. La duplication consiste alors à dédoubler le nœud N_i en question en deux nœuds, N_i et N_i'. Cette duplication permet ainsi de traduire l'existence de deux composantes connexes en associant à chacune un nœud dans le Graphe Topologique. Cette duplication est ainsi effectuée pour chaque composante connexe détectée. Si une carte de cheminement possède n composantes connexes l'élément associé sera alors représenté par n nœuds dans le Graphe Topologique. Les relations topologiques initialement présentes dans le Graphe Topologique sont ensuite rattachées aux nœuds concernés par ces relations. Un exemple de duplication dans le Graphe Topologique est montré dans les figures 4.9 et 4.10. La figure 4.9 présente le Graphe Topologique original. Une duplication est effectuée sur O_2 lorsque la déconnexion de la carte de cheminement est détectée. La duplication de O_2 et la reconnexion des relations topologiques aux nœuds concernés amènent au nouveau Graphe Topologique présenté dans la figure 4.10.

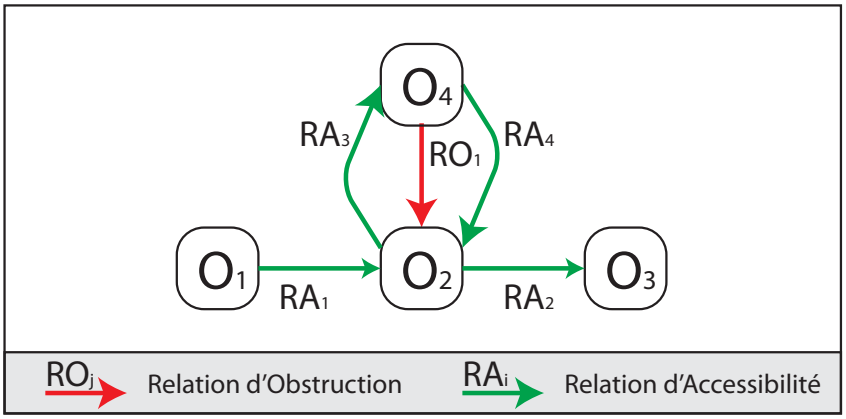


Figure 4.9 – Graphe Topologique associé à l'environnement présenté dans la figure 4.8 avant d'effectuer une duplication.

Bien que le principe soit similaire pour les objets statiques ou dynamiques, la duplication et le rattachement des arcs sont effectués à deux moments distincts. Dans le cas des objets statiques, la duplication de l'objet est effectuée dès l'insertion de l'objet obstruant tandis que dans le cas d'objets dynamiques cette duplication sera effectuée à la volée pendant la planification.

1. Les objets statiques sont caractérisés par une vitesse de déplacement nulle. Quand un objet statique est détecté dans l'environnement, il est intéressant d'identifier directement son impact sur l'environnement puisqu'il ne sera pas remis en question par la suite (sauf dans le cas où cet objet est supprimé ou si il devient

dynamique). Lorsqu'un élément statique est identifié, le Graphe Topologique permet de connaître les objets qu'il obstrue. Nous récupérons donc l'ensemble de ces objets afin de vérifier la validité de leur cartes de cheminement et de voir si l'ajout de cet élément entraîne ou non une déconnexion de la carte de cheminement. La représentation de ces éléments dans le Graphe Topologique est inchangée si il n'y a pas de déconnexion. Dans le cas contraire, une déconnexion de la carte de cheminement entraîne une duplication de l'élément auquel elle est associée. Cette méthode de détection implique de devoir calculer l'impact de l'objet au moment de son ajout ou au moment de sa stabilisation dans le cas d'un objet préalablement dynamique.

2. Les obstructions apportées par les objets dynamiques sur les Surfaces Navigables qu'ils obstruent vont évoluer en fonction des déplacements de ces objets. Nous proposons de détecter les obstructions crées par ces éléments seulement au moment de la planification de chemin. Dans ce cas précis, la planification de chemin va tout d'abord identifier un chemin topologique, puis la planification locale va soit identifier un chemin valide, si la cible et le point de départ appartiennent à la même composante connexe, soit échouer si les deux configurations appartiennent à des composantes connexes différentes de la carte de cheminement. Dans ce second cas, l'information de l'échec est remontée au niveau du Graphe Topologique qui duplique le nœud à l'origine de l'erreur avant d'effectuer une nouvelle planification de chemin.

Afin d'expliquer le principe de la duplication, nous nous appuierons sur l'exemple présenté dans la figure 4.8. Dans cet exemple un élément O_1 permet l'accès à O_2 pour un accès final vers O_3. L'objet O_4 est une simple planche mais obstrue les configurations de la carte de cheminement de O_2, empêchant ainsi de le traverser dans toute sa longueur. La figure 4.9 présente le Graphe Topologique associé à cet environnement sans l'utilisation de la duplication. Quatre relations d'accessibilité sont alors détectées : $RA_1(O_1, O_2)$, $RA_2(O_2, O_3)$, $RA_3(O_2, O_4)$ et $RA_4(O_4, O_2)$. Une seule relation d'obstruction est identifiée : $RO_1(O_4, O_2)$. Dans ce cas, lors de la planification d'un chemin topologique entre les objets O_1 et O_3, l'algorithme identifie dans un premier temps le chemin $O_1 \rightarrow O_2 \rightarrow O_3$ et la planification locale sur O_2 se solde par un échec de par la présence de la planche.

Si la planche est un élément statique, la division de la carte de cheminement de O_2 en deux composantes connexes est remarquée dès l'insertion de cette planche dans l'environnement et la duplication est aussitôt effectuée. Les deux composantes sont alors représentées par deux nœuds (O_2 et O'_2) dans le Graphe Topologique. Lors de la validation des relations topologiques, nous déterminons à quelle composante connexe de O_2 est affectée la transition, et donc la relation, échantillonnée. Le Graphe Topologique résultant de cette duplication est présenté dans la figure 4.10. Il n'existe plus de chemin $O_1 \rightarrow O_2 \rightarrow O_3$ puisque l'information sur la déconnexion de la carte de cheminement est remontée au niveau du Graphe Topologique. Cette nouvelle représentation nous permet donc d'identifier directement le chemin topologique $O_1 \rightarrow O_2 \rightarrow O_4 \rightarrow O'_2 \rightarrow O_3$ dans

le Graphe topologique permettant de prendre en compte la déconnexion de O_2.

En considérant maintenant la planche en tant qu'objet dynamique, la duplication va être effectuée différemment. En effet, la séparation de la carte de cheminement en deux composantes connexes n'est pas connue à l'avance. Cette déconnexion n'est donc observée qu'au moment où la requête de planification locale est formulée. Lorsque celle-ci est effectuée, la planification locale sur O_2 renvoie un échec. L'information est alors notifiée au Graphe Topologique et O_2 est dupliqué en deux nœuds : O_2 et O_2'. Afin de séparer les deux composantes connexes et d'éviter de trouver de nouveau le même chemin topologique, nous dupliquons les arcs topologiques associés à O_2 de façon différente en fonction de leur propriétés :

1. Les arcs du Graphe Topologique originellement rattachés à O_2 sont dupliqués et rattachés à O_2 et O_2' si ils ne sont pas présents dans le chemin topologique global précédemment calculé. Dans l'exemple courant, c'est le cas des arcs RA_3 et RA_4 entre O_2 et O_4.
2. L'échec de planification dans la carte de cheminement locale, nous permet de déduire qu'il n'existe pas de trajectoire possible au niveau local entre les arcs entrant et sortant de la carte de cheminement dans le chemin topologique. L'arc d'entrée et donc rattaché à l'un des nœuds dupliqué tandis que l'arc sortant et rattaché au second nœud afin de représenter cette déconnexion. Dans notre exemple, l'arc entrant est l'arc RA_1 qui est alors rattaché à O_2 et l'arc sortant est l'arc RA_2 qui est rattaché à O_2'. Cette méthode permet ainsi de déconnecter le point d'entrée et le point de sortie dans l'objet O_2 en rattachant les arcs d'entrée et de sortie à des composantes connexes différentes.

Le Graphe Topologique issu de cette duplication est présenté dans la figure 4.10. Il s'agit ici du même graphe que dans le cas de l'obstacle statique. Une fois la duplication finie, une nouvelle requête de planification est relancée dans le Graphe Topologique modifié et le chemin alternatif $O_1 \rightarrow O_2 \rightarrow O_4 \rightarrow O_2' \rightarrow O_3$ est trouvé. Cette représentation permet ainsi de prendre en compte l'obstruction de O_4 sur O_2 détectée précédemment et de trouver la solution alternative.

Les méthodes de duplication statique et dynamique conduisent donc à la construction d'un Graphe Topologique similaire. Cette propriété permet notamment une utilisation simultanée de ces deux méthodes sans effet de bord entre les deux. L'utilisation de la duplication statique va entraîner un surcoût lors de l'ajout de l'objet afin de vérifier la validité et la connectivité des cartes de cheminement concernées. Toutefois, une fois cet ajout effectué, la représentation de l'objet dans le Graphe Topologique restera ensuite la même et permettra de repérer ces différentes composantes connexes à la première tentative de planification. Pour sa part, la duplication dynamique va être effectuée au prix de plusieurs planifications locales. La configuration des composantes connexes est donc détectée au fur et à mesure des échecs de planification par une méthode d'essai-erreur. Lorsque plusieurs objets vont séparer la surface, comme cela est présenté dans la figure 4.11, plusieurs duplications sont alors nécessaires. Dans ce cas, le processus de duplication sera répété autant de fois qu'il sera nécessaire afin de lever toutes les

ambiguïtés présentes dans le Graphe Topologique et capturer précisément la topologie de l'environnement.

4 Conclusion

Nous avons présenté au cours de ce chapitre notre algorithme de planification hiérarchique. Cette planification hiérarchique permet dans un premier temps d'identifier un chemin topologique global puis de le raffiner en calculant des trajectoires locales. Cette structure à deux étages permet d'effectuer une planification allant du haut niveau vers le bas niveau, commençant par identifier tout d'abord un chemin d'éléments devant être traversés puis le raffinant ensuite en calculant une trajectoire locale lorsque l'agent arrive sur une nouvelle surface navigable.

Lors de la planification globale, l'observation de l'évolution des relations au cours du temps nous permet d'intégrer à la recherche des critères de sécurité afin de gérer différentes attitudes et différentes prises de risques de la part de l'agent au cours de sa navigation. De plus, ces comportements et ces décisions sont gérés automatiquement et permettent l'adaptation automatique de l'agent en fonction de la mémoire qu'il possède sur l'environnement et son évolution. Les changements observés au fil de la simulation sont ainsi susceptibles de modifier ses décisions au cours du temps.

La planification dans le Graphe Topologique permet de plus d'emprunter des chemins topologiques utilisant des relations d'accessibilité périodiques. Cette propriété nous permet l'utilisation de manière implicite de ponts temporels entre des éléments dynamiques déconnectés dans le temps et l'espace. De plus, cela permet à l'agent d'utiliser les mouvements propres des objets pour l'aider dans sa tâche de navigation et d'augmenter ainsi ses possibilités de navigation. Nous proposons donc à travers notre représentation une prise en compte de la dynamique de l'environnement sans que l'agent ne possède de connaissances a priori sur son évolution. Cette prise en compte permet ainsi de détecter mais surtout d'anticiper des relations temporelles dans l'environnement afin de générer des chemins complexes au travers de surfaces déconnectées.

La planification locale a alors pour but de déterminer une trajectoire dans les cartes de cheminement locales. Durant cette planification locale, l'ensemble des capacités de déplacement de l'agent est capturé dans une structure unique de planification. Grâce à cette structure, sous forme de carte de cheminement, l'adaptation de posture de l'agent est directement gérée lors de la planification locale. De plus, la dynamique de l'environnement, et notamment le déplacement des obstacles, est directement intégrée puisque cette planification est effectuée dans le domaine spatio-temporel. Enfin les tests de collision effectués au cours de cette planification sont optimisés par le Graphe Topologique en considérant seulement les obstructions pertinentes identifiées et en réduisant ce test à un simple lancer de rayon dans l'espace des configurations, grâce à la définition des Volumes d'Interaction.

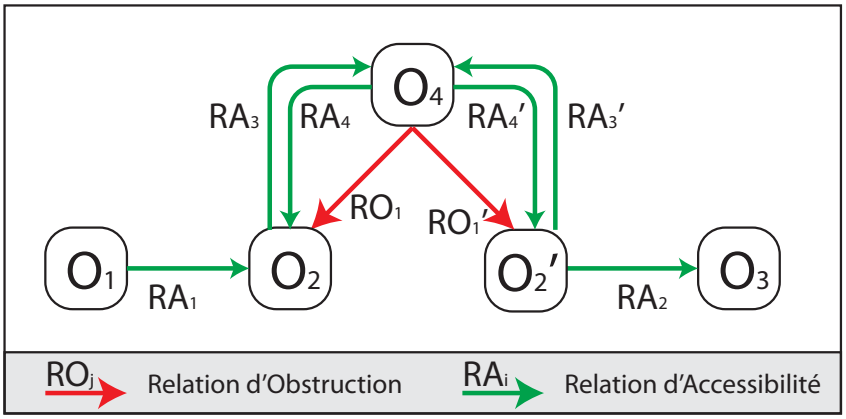
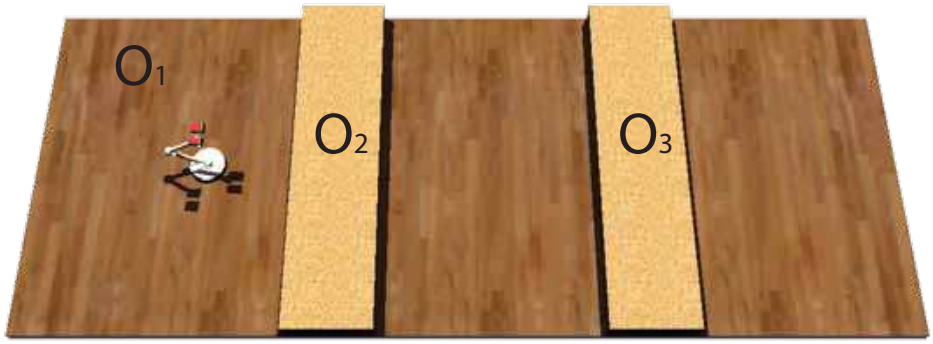
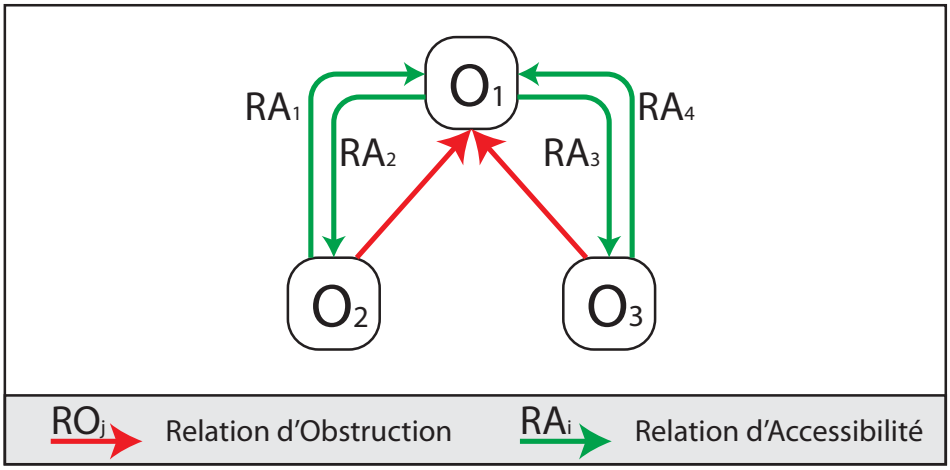


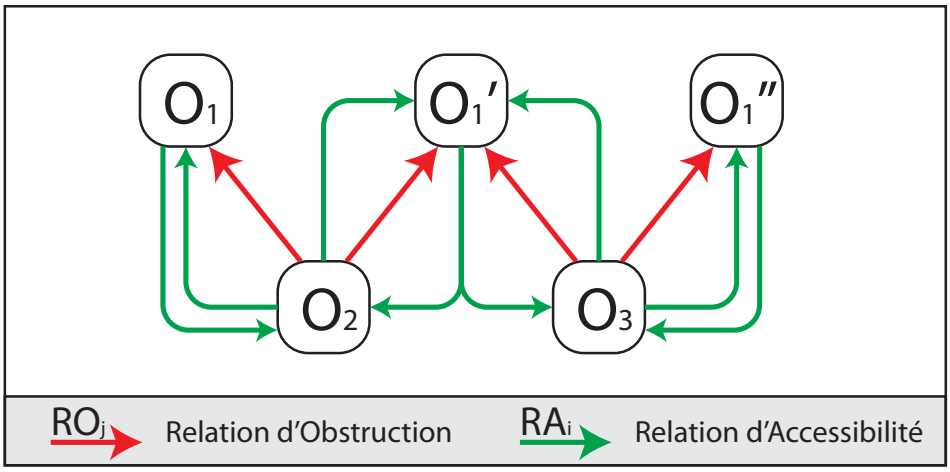
Figure 4.10 – Graphe Topologique associé à l'environnement présenté dans la figure 4.8 une fois que la duplication a été effectuée. On peut remarquer qu'il n'est plus possible de trouver un chemin topologique menant de O_1 à O_3 sans passer par O_4.



(a) Environnement avec deux planches créant des déconnexions



(b) Graphe Topologique associé sans duplication



(c) Graphe Topologique associé avec duplications

Figure 4.11 – Les deux planches O_2 et O_3 séparent la carte de cheminement de O_1 en trois composantes connexes. Cette séparation se retrouve au niveau du Graphe Topologique en utilisant les duplications.

Présentation des résultats obtenus 5

Après avoir présenté au cours des chapitres précédents les concepts théoriques sur lesquels repose notre méthode, nous allons maintenant présenter l'architecture globale de notre solution ainsi que son application pratique dans différents environnements de test. Nous allons voir dans un premier temps l'agencement des différents modules proposé. Dans un second temps nous présenterons le modèle utilisé pour représenter notre agent et les environnements dans lesquels nous avons effectués les tests. Dans une dernière partie, nous présenterons les résultats et les performances obtenus par notre méthode.

1 Architecture globale

Les différents concepts introduits au cours de cette thèse ont été implémentés et regroupés au sein d'une architecture globale. Cette architecture globale et l'agencement des différents modules est présentée dans la figure 5.1. Afin d'augmenter la dynamique et le caractère imprévisible de l'environnement, nous avons couplé notre environnement à un moteur physique. Au cours de la simulation, l'environnement peut donc être modifié soit par un scénario programmé, soit par l'utilisateur ajoutant ou retirant des éléments ou encore par un pas de simulation du moteur physique faisant évoluer la simulation dynamique. Ces modifications vont donc avoir un impact direct sur la scène et sur la configuration des objets remettant ainsi à jour le moteur physique et les Volumes d'Interaction associés dans l'espace des configurations.

Un second moteur de détection de collision tourne en tâche de fond de l'application et se charge de son côté de détecter et de notifier de façon permanente les interactions détectées entre les Volumes d'Interaction. Cette détection est rapportée au Graphe Topologique et permet de le maintenir constamment à jour. Lorsqu'une requête de planification est formulée, le Graphe Topologique offre donc une vision de la topologie valide et la planification hiérarchique se base sur ce graphe et sur les Volumes d'Interaction afin de fournir une trajectoire valide à l'agent.

Le moteur physique et le moteur de détection de collision ont été implémentés à l'aide de la librairie *Bullet Physics*. Comme nous le présentons dans l'architecture globale, l'intégration d'un moteur physique dans un environnement utilisant notre méthode est effectuée facilement. La séparation du moteur physique et du moteur de détection permet d'utiliser des moteurs différents dans les deux cas. Cette séparation permettra

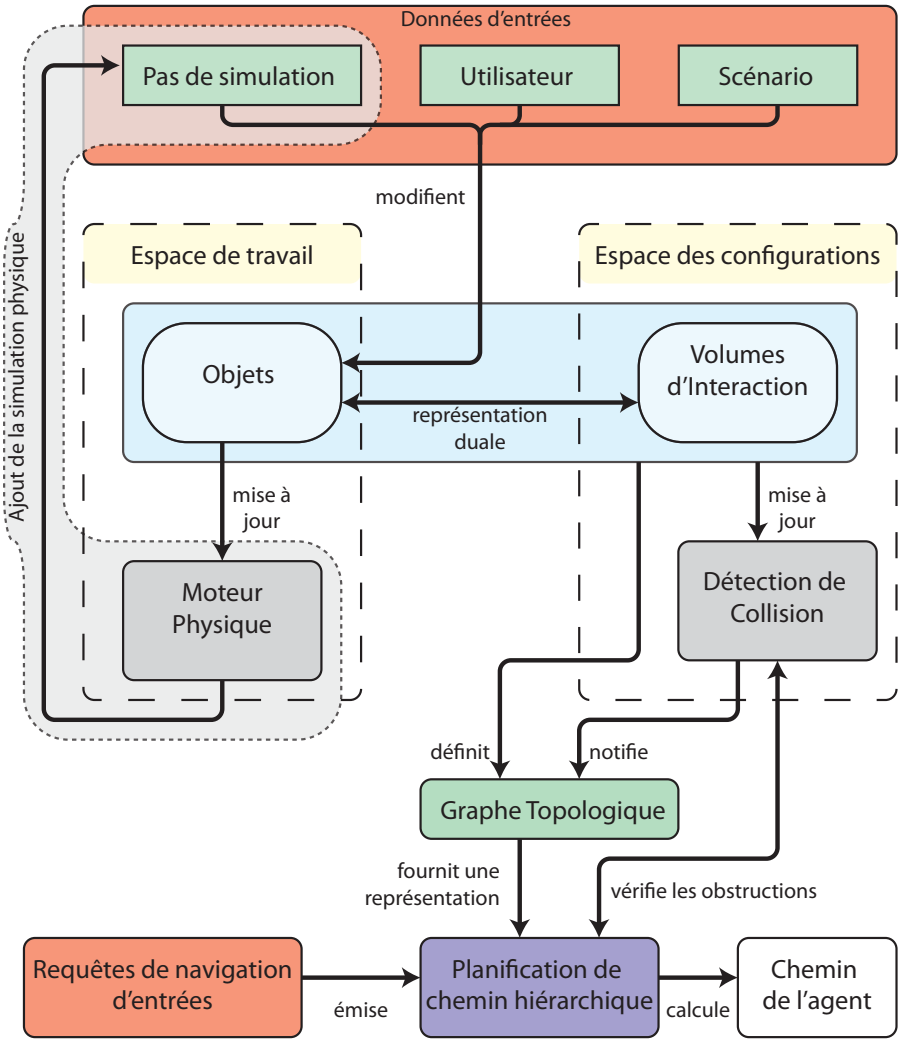


Figure 5.1 – Ce schéma représente l'architecture globale de notre méthode. La section grisée à gauche a été ajoutée pour intégrer le moteur physique au cœur de notre méthode. Cet ajout n'apporte pas de modification au reste de l'algorithme.

de spécialiser ultérieurement le moteur de détection de collision ou encore de paralléliser les deux moteurs en affectant chacun à un cœur différent dans les architectures multi-cœur.

Les différents tests ont été effectués dans un environnement interactif où l'agent peut intervenir en lançant par exemple des éléments physiques à travers l'environnement. Le suivi de la physique, à travers le moteur physique, et de la topologie, à travers le moteur de détection de collision, sont directement gérés en temps réel par ces moteurs qui tournent de manière parallèle à la simulation. De plus, lorsque des relations d'accessibilité sont validées, nous échantillonnons une dizaine de trajectoires de transition entre les surfaces. Cet échantillonnage n'est pas effectué à chaque pas de temps où la relation est détectée mais seulement si le dernier échantillonnage remonte à plus d'un dixième de seconde afin de les répartir au cours de la simulation. Enfin lorsque des objets physiques sont insérés dans l'environnement, le moteur physique possède un filtrage lui permettant de déterminer si un objet est actif ou au repos, nous récupérons donc cette information au cours de la simulation afin de déterminer les objets dynamiques de ceux qui se sont stabilisés au sol. Cette information nous permet donc de déterminer les moments où les Volumes d'Interaction doivent être remis à jour.

2 Définition des environnements

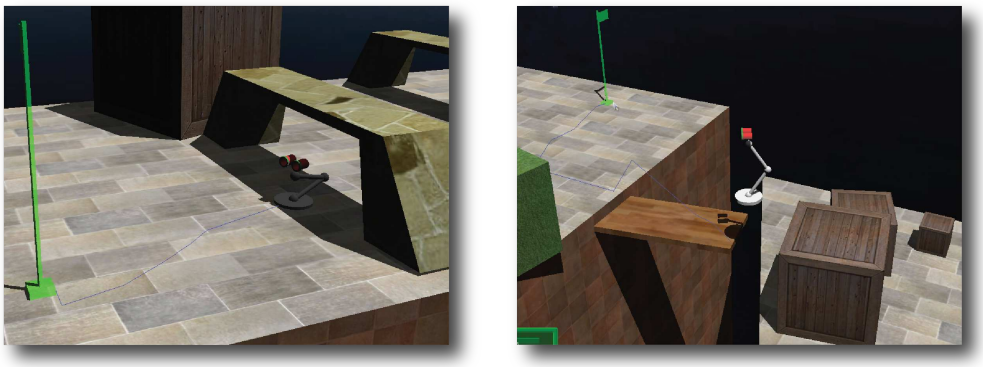
2.1 Représentation de l'agent

Dans nos différents cas de test l'agent que nous avons utilisé est une lampe animée. Cette lampe va permettre de mettre en avant les possibilités offertes par notre méthode grâce à ses différentes capacités de déplacement. Notre agent possède trois capacités de déplacement distinctes :

1. Une marche debout
2. Une marche accroupie
3. Un mouvement de saut

Les mouvements de marche sont en fait assimilés, dans le cas de notre lampe animée, à des translations de la lampe sur le sol de l'environnement. L'utilisation de deux mouvements de marche distincts permet de mettre en avant l'adaptation de posture de l'agent durant sa navigation. Le mouvement de saut enfin permet à notre agent de franchir des espaces vides et d'effectuer les transitions entre des surfaces de navigation déconnectées. Les capacités de marche accroupie et de saut sont illustrées dans la figure 5.2. Les cylindres englobants associés à ces capacités de navigation ont des hauteurs fixées à 50 cm pour les capacités **1** et **3** et à 15 cm pour la capacité de marche accroupie **2**. Le rayon de ces cylindres est de 20 cm pour les mouvements **1** et **3** et de 30 cm pour la capacité de déplacement **2**. Ainsi les capacités de déplacement de marche debout (**1**) et de saut (**3**) sont représentées par un cylindre unique tandis qu'un second cylindre est associé à la capacité de déplacement de marche accroupie. Les vitesses de déplacement pour les capacités **1** et **3** sont fixées à $v_{min} = 0\ m.s^{-1}$ et $v_{max} = 2\ m.s^{-1}$ tandis que

la vitesse maximale pour le mouvement **2** a été fixée à $v_{max} = 1\ m.s^{-1}$. Pour les trois capacités de déplacement, la pente navigable a été définie à $p_{nav} = 30°$. Enfin, pour le mouvement de saut, la vitesse d'impulsion maximale a été fixée à $v_{dec} = 2\ m.s^{-1}$ et la vitesse d'atterrissage à $v_{att} = 3\ m.s^{-1}$.



(a) Utilisation de la marche accroupie **(b)** Utilisation du saut

Figure 5.2 – L'agent possède plusieurs capacités de déplacement lui permettant d'augmenter ses possibilités de navigation dans l'environnement.

2.2 Représentation pratique des éléments de l'environnement virtuel

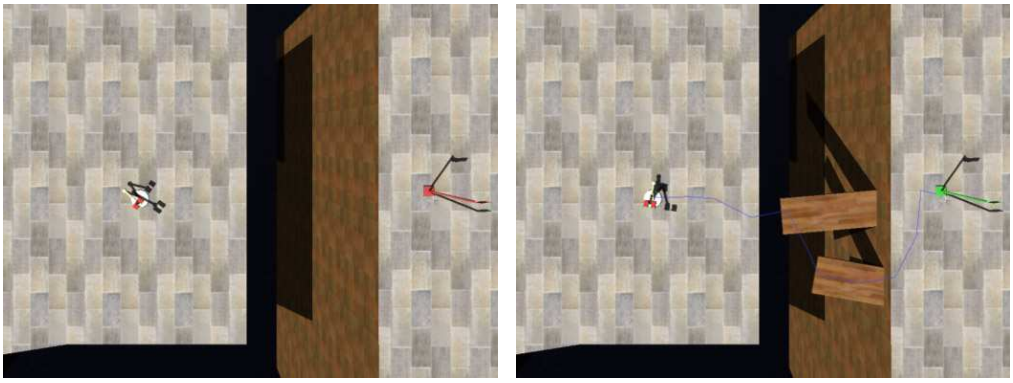
Nous avons testé notre méthode dans différents environnements virtuels. Au sein de nos environnements, chaque objet de l'espace de travail est représenté à l'aide de ses Volumes d'Interaction dans l'espace des configurations. Pour chaque objet, les Volumes d'Interaction sont définis de la façon suivante :

- La géométrie est analysée afin de déterminer l'ensemble des triangles navigables. L'utilisation d'une pente navigable $p_{nav} = 30°$ identique pour l'ensemble des capacités de déplacement permet d'avoir un unique ensemble de triangles navigables à identifier. Pour cet ensemble de triangles identifiés, nous définissons pour chaque capacité de déplacement M_k la Surface Navigable associée $Ns(O, M_k)$.
- La géométrie de l'objet est ensuite extrudée en se basant sur le cylindre englobant associé à chaque capacité de déplacement de l'agent. Ces extrusions permettent de définir le Volume Interdit associé à l'objet pour chaque capacité de déplacement utilisée. Toutefois, les capacités de marche debout et de saut étant associées à un même cylindre englobant, les Volumes Interdits calculés seront similaires. Afin de réduire le nombre d'éléments à prendre en compte dans le moteur de détection de collision un seul Volume Interdit est utilisé pour représenter les obstructions de ces deux capacités.
- Enfin, les vitesses de décollage et d'atterrissage associées à la capacité de saut permettent de définir le profil d'accessibilité représentant l'ensemble des configurations atteignables par l'agent. La mise en correspondance de ce profil avec la Surface Navigable associée au mouvement de saut permet de définir le Volume

d'Accessibilité de l'objet. Les mouvements de marche étant contraints au sol, il ne leur est donc pas associé de Volume d'Accessibilité.

Lors de la présentation des Volumes d'Interaction, au cours du chapitre 2, nous avons vu qu'un objet pouvait créer des obstructions vis-à-vis de lui-même, lorsqu'il présente par exemple des pans verticaux. Dans ce cas, nous avons proposé de supprimer ces configurations obstruées lors de la construction de la Surface Navigable. Toutefois, dans la pratique, la Surface Navigable est définie par l'ensemble des triangles navigables identifiés. Les configurations obstruées de la Surface Navigable seront en fait identifiées ultérieurement, lors de la construction de la carte de cheminement, en testant les configurations tirées pour construire la carte locale. Cette modification permet une construction plus efficace des Surfaces Navigables et des Volumes d'Accessibilité se basant directement sur la géométrie de l'objet.

3 Environnements de tests



(a) Les deux surfaces sont déconnectées

(b) L'ajout de planches crée un escalier automatiquement détecté par l'agent

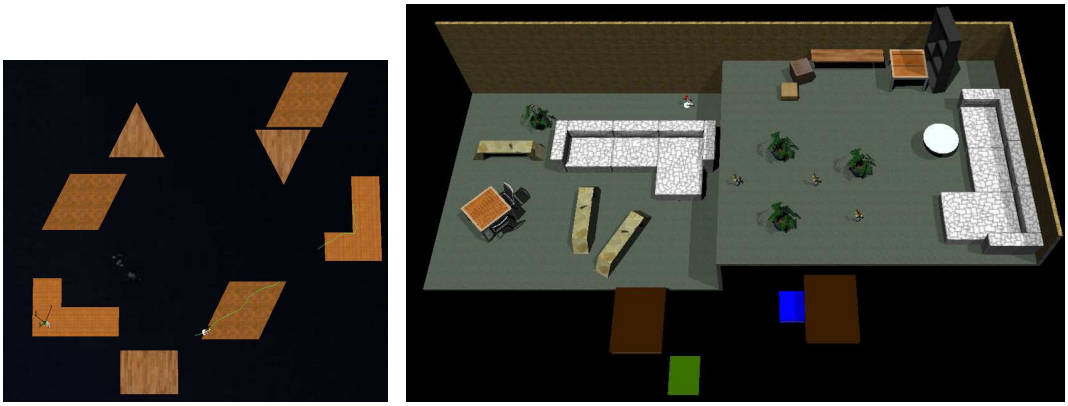
Figure 5.3 – L'agent détecte automatiquement les changements topologiques et les utilise dans sa tâche de navigation. L'ajout de deux planches dans l'exemple courant permet à l'agent d'identifier un chemin entre deux surfaces qui étaient à l'origine déconnectées.

Divers environnements de tests ont été utilisés au cours de cette thèse. Ces environnements ont pour but de mettre en avant différents aspects de notre méthode. Nous avons pu ainsi tester les propriétés suivantes :

- anticipation et évitement des obstacles dynamiques
- détection automatique à la volée d'accessibilité entre des surfaces déconnectées
- détection de chemin entre des éléments navigables mobiles et utilisation de leur déplacements propres pour la navigation de l'agent
- navigation au sein d'environnements composés d'éléments statiques mais également dynamiques (que les mouvements soient aléatoires ou déterminés par un scénario)

– navigation au sein d'un monde totalement dynamique utilisant un moteur physique

La figure 5.3 montre un exemple d'adaptation de l'agent aux modifications de l'environnement. L'agent détecte de manière autonome que l'ajout dynamique des planches dans l'environnement crée un escalier et lui offre de nouvelles opportunités de navigation. La figure 5.4 présentent deux autres exemples d'environnements utilisés. Dans la figure 5.4a, l'environnement se compose simplement de surfaces mobiles déconnectées. La tâche de l'agent consiste alors à identifier un chemin au sein de cet environnement en utilisant le déplacement propre des éléments pour trouver une séquence de surfaces mobiles le menant à sa cible. Cet environnement permet de tester la représentation des liens temporels introduite dans le Graphe Topologique. Le second environnement, présenté dans la figure 5.4b, permet de tester l'ensemble des capacités de l'agent. En effet, certains éléments présents vont contraindre son déplacement et le forcer à adapter sa posture, tandis que d'autres vont se comporter en simples obstacles, ou encore en plateformes navigables mobiles comme précédemment. Cet environnement nous permet de tester les capacités de notre méthode dans un environnement plus complet et plus contraint pour la navigation.



(a) Environnement composé de plateformes mobiles

(b) Living Room

Figure 5.4 – L'environnement 5.4a permet de tester la navigation sur des plateformes dynamiques en mouvement. L'agent doit trouver un chemin à l'aide de ces plateformes pour se déplacer dans l'environnement en utilisant les déplacements des objets. Dans l'environnement de la figure 5.4b, l'agent est amené à utiliser l'ensemble de ses capacités. Il va ainsi devoir se baisser pour passer sous certains éléments mais également anticiper le déplacement des obstacles mobiles et des éléments navigables.

4 Performances observées

Afin d'évaluer notre méthode, nous avons réalisé des mesures de performances dans différents environnements. Les résultats de ces mesures sont présentés dans le tableau 5.1. Les environnements *Disconnected*, *Living room* et *Physique 1* ont été présentés res-

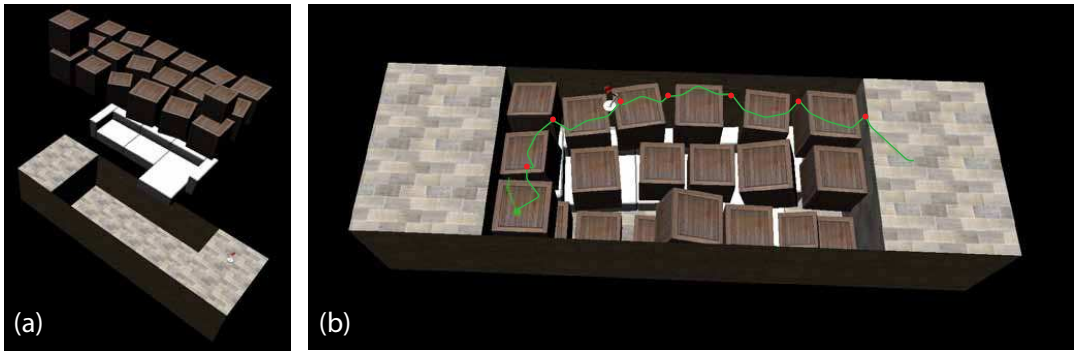


Figure 5.5 – Dans cet environnement nous utilisons un moteur physique. Les éléments sont placés dans la configuration (a) au départ, l'utilisateur peut ensuite ajouter de nouveaux éléments. L'agent est ensuite capable de trouver un chemin en utilisant les surfaces navigables des éléments comme montré par le schéma (b).

pectivement dans les figures 5.4a, 5.4b et 5.5. L'environnement *Physique 2* est composé d'une grande surface navigable et l'utilisateur lance des objets dans la scène pendant la navigation de l'agent. Au cours des mesures effectuées dans cet environnement *Physics 2*, il y avait généralement plus de 70 éléments dynamiques dans la scène. Ces mesures ont été effectuées sur un Intel(R) Core(TM)2 Extrem, CPU X7900, 2.80GHz. Comme nous l'avons dit précédemment, nous avons utilisé la bibliothèque *Bullet Physics* afin de gérer le moteur de détection de collision dans ces environnements. L'implémentation actuelle n'est pour l'instant pas parallélisée.

Le tableau 5.1 résume les temps moyens de : détection de collision entre les Volumes d'Interaction, de mise à jour du Graphe Topologique et enfin de planification de chemin. La dernière colonne présente le pourcentage de temps passé dans les tests d'obstructions lors de la validation des arcs de la carte de cheminement lors de la planification de chemin. Nos mesures montrent que la détection des interactions et la mise à jour du Graphe Topologique sont très peu coûteuses. On peut également observer, en regardant le premier environnement, qu'il est très simple d'identifier un chemin entre des surfaces mobiles déconnectées. Enfin la planification dans le dernier environnement est très coûteuse (plus de 300ms). Ce coût s'explique par la présence de nombreux obstacles dynamiques lors de la planification de chemin de l'agent, on peut également remarquer que 95 % de ce temps est d'ailleurs utilisé afin de vérifier la validité des arcs explorés. Il est donc à noter que les performances du moteur de détection de collision utilisé sont très importantes puisque ce moteur va jouer un rôle essentiel lors de la planification. On remarque ainsi que même pour des environnements plus simples, presque la moitié du temps de planification est utilisé pour la validation de ces arcs.

Enfin, des mesures supplémentaires ont été effectuées dans le dernier environnement physique comprenant plus de 70 objets. Ces mesures ont été effectuées afin de connaître les temps de mise à jour moyens des Volumes d'Interactions lorsque ceux-ci doivent être entièrement recalculés. Dans le cas où l'objet passe d'un statut stable à un statut

Table 5.1 – Benchmarks.

Nom de l'environnement	Détection de Collision	Mise à jour du Graphe Topologique	Temps moyen de planification	Tests d'obstructions (%planification)
Disconnected Env	0,48 ms	0,06 ms	7,8 ms	-
Living-room	14,12 ms	6,46 ms	93,98 ms	67,77 %
Physique 1	10,01 ms	0,71 ms	25 ms	37,5%
Physique 2	14,15 ms	1,25 ms	311 ms	94,6%

dynamique, alors seul un Volume d'Interdiction lui est associé et le temps moyen de construction de ce volume est de 1,25 ms. Dans le cas où l'ensemble des volumes doit être recalculé, on observe alors un temps de mise à jour de 4,6 ms. Ces temps sont donc tout à fait compatibles avec les applications interactives souhaitées.

5 Conclusion

Notre algorithme se base sur un moteur de détection de collision pour suivre les interactions entre les Volumes d'Interaction. Ce moteur physique est exécuté en tâche de fond lors de la simulation et permet donc d'identifier précisément tous les changements de topologie dès leur apparition. Les performances de notre méthode reposent donc en grande partie sur les performances de la librairie de détection de collision utilisée. Les différents résultats obtenus nous montrent que les temps de mise à jour du Graphe Topologique sont négligeables. Toutefois, dans des environnements complexes, tels que le salon, les performances de l'algorithme sont diminuées. Afin de réduire ces temps de calcul dans la détection de collision des géométries simplifiées des objets peuvent être utilisées.

Notre méthode propose de plus un algorithme de planification de chemin dynamique permettant l'identification de trajectoires à travers des surfaces navigables déconnectées et mobiles tout en évitant les collisions avec les obstacles dynamiques. Toutefois, la planification locale est le processus consommant le plus grand temps de calcul. Ce temps de calcul important est dû au fait que l'agent doit vérifier la validité de sa trajectoire lors de sa navigation afin de la maintenir à jour si des éléments dynamiques viennent subitement obstruer son chemin. Comme nous pouvons le voir, les tests de validité utilisent une grande partie du temps de calcul. Ce coût de la planification est encore augmenté lorsque la trajectoire de l'agent est invalidée et qu'elle doit être replanifiée à cause d'une nouvelle obstruction détectée. Afin d'augmenter les performances de la replanification, il serait intéressant de se diriger vers des algorithmes de type D* par exemple [**?**].

Conclusion

Les recherches effectuées au cours de cette thèse se sont intéressées à la planification de chemin d'un agent virtuel autonome et à son adaptation de posture au sein d'environnements virtuels dynamiques dont l'évolution n'est pas connue a priori. Les domaines d'applications utilisant de tels mondes virtuels sont aujourd'hui de plus en plus diversifiés. De nombreuses applications industrielles, telles que les gestions de cycles de vie des produits, utilisent maintenant des outils basés sur des environnements virtuels. Beaucoup de ces applications vont avoir un caractère interactif où l'utilisateur va être amené à interagir avec son environnement. Même dans le cadre d'applications de production, où le rendu final est produit hors ligne, un besoin de solutions performantes et rapides apparaît afin d'avoir des outils de prévisualisation efficaces des résultats. Le développement constant des outils informatiques permet de plus aujourd'hui de construire des environnements de plus en plus complexes et réalistes présentant de nombreuses composantes dynamiques. Nous avons proposé au cours de cette thèse de nouveaux outils de représentation et de planification permettant, à des agents virtuels, de trouver un chemin dans des environnements totalement dynamiques, tels que ceux faisant intervenir de la simulation physique. Ces nouveaux outils permettent de surmonter les limitations de nombreuses méthodes actuelles, dès que l'on s'intéresse à des environnements dynamiques dont l'évolution n'est pas connue a priori. Ces contributions se décomposent en trois parties distinctes. Utilisant ces trois contributions, nous avons défini une architecture globale de représentation et de suivi de l'évolution d'un environnement dynamique mais également d'identification de trajectoires de navigation pour les agents à l'intérieur de cet environnement.

Résumé des contributions

Les recherches effectuées au cours de cette thèse ont eu pour objectif global de proposer une solution de navigation dans des environnements dynamiques, dont l'évolution de la topologie n'est pas connue et n'est pas prévisible a priori. Cet objectif est lié à l'utilisation de plus en plus courante de tels mondes virtuels dans de nombreux domaines. Nous avons également proposé une nouvelle approche sur la manière de prendre en compte les objets présents dans un environnement en les caractérisant à la fois en temps qu'obstacles, mais surtout, comme des nouvelles zones pouvant aider l'agent au cours de sa navigation. Cette propriété permet d'obtenir des comportements plus complexes et intéressants de la part des agents mais n'a été que très peu exploitée jusqu'à maintenant. Enfin, nous avons voulu proposer une méthode performante et compatible

avec des temps de calculs interactifs. Cette contrainte forte découle notamment des applications envisagées, que ce soit pour des domaines demandant un temps de réaction très rapide, comme le jeu vidéo ou la réalité virtuelle, ou dans l'idée de proposer des outils de prévisualisation rapide dans le cadre d'application de production. La suite de la section résume l'ensemble des contributions que nous avons apporté au cours de cette thèse afin de répondre à ces objectifs.

Définition d'une représentation duale des éléments

Nous avons, dans un premier temps, proposé une nouvelle représentation des objets dans l'espace des configurations. Cette représentation duale permet, au regard des capacités de l'agent, de définir les interactions possibles entre un objet de l'environnement et cet agent, que ce soit en terme d'obstruction et de navigation. Notre définition des capacités de déplacement de l'agent permet d'introduire une caractérisation générique de ces interactions, et de s'abstraire de la représentation géométrique de l'agent considéré. Ces différentes capacités de navigation permettent, de plus, de gérer les adaptations de posture de l'agent. Enfin, cette représentation permet également de décorréler la phase de planification de chemin de la phase d'animation de l'agent pendant sa navigation, permettant ainsi de réduire la complexité du problème de planification. Trois types d'interactions agent-objet ont été définies :

- *L'obstruction*, permettant de définir les configurations où l'objet est une gêne pour l'agent.
- *La navigabilité*, permettant de caractériser les zones de l'objet où l'agent va pouvoir se tenir.
- *L'accessibilité*, permettant de représenter l'ensemble des configurations autour de l'objet où l'agent va être capable d'accéder en fonction de ses capacités de déplacement.

La définition de ces trois interactions repose sur la définition des Volumes d'Interaction qui permettent une identification explicite des propriétés associées à un objet, que ce soit en terme d'obstructions, mais également en terme de navigabilité et d'accessibilité. D'autre part, nous proposons une définition des interactions formulée dans un espace à trois dimensions et, permettons ainsi, d'utiliser l'ensemble de l'algorithmique 3D disponible, afin d'identifier facilement les propriétés de ces interactions. Pour utiliser cette représentation dans des environnements dynamiques, nous proposons également une méthode de mise à jour des Volumes d'Interaction lorsque des modifications de leur configuration sont détectées durant la simulation.

Construction de la topologie par l'observation

En se basant sur les représentations duales associées aux objets de l'environnement, nous définissons une représentation de la topologie globale de l'environnement. En localisant et caractérisant dans l'espace des configurations les interactions existantes entre les objets, nous identifions tout d'abord les relations topologiques d'accessibilité et d'obstruction existant dans l'environnement. Cette représentation offre ainsi un apport par rapport à la littérature qui propose en général une représentation des éléments

dynamiques sous la forme d'obstacles. La représentation globale de la topologie est ensuite construite par la mise en commun de l'ensemble des relations observées au cours de la simulation. De plus, grâce à l'observation et au suivi de l'évolution de ces relations, cette représentation est augmentée à l'aide d'informations temporelles. L'ajout de cette information temporelle peut être vu comme la mise en place d'une mémoire de l'évolution de la configuration des relations topologique au cours de la simulation. Cette information permet, sans connaissance a priori sur l'évolution de l'environnement, de caractériser les relations temporelles détectées dans l'environnement et d'anticiper, dans un futur proche, certaines évolutions topologiques possibles. Pour le moment, cette représentation globale de la topologie ne prend pas en compte de limite dans le temps ou dans l'espace et définit donc une représentation omnisciente de la topologie. Toutefois, il serait également possible de limiter la connaissance à disposition dans ce Graphe Topologique en lui associant, par exemple, une fenêtre d'observation soit temporelle, soit géographique, caractérisant la persistance de la mémoire de l'agent et son aptitude à percevoir des informations.

Planification de trajectoire hiérarchique

Nous proposons enfin une planification de chemin hiérarchique se basant sur notre représentation dynamique et temporelle de la topologie. Cette planification a pour rôle d'identifier tout d'abord une séquence de surfaces navigables que va devoir traverser l'agent, puis de calculer ensuite une trajectoire locale sur les zones de navigation identifiées. La représentation temporelle va permettre d'utiliser plusieurs critères afin d'optimiser la recherche de chemin en cherchant, par exemple, à minimiser les temps d'attente de l'agent, ou à définir des critères de sécurité au cours de la navigation. La structure que nous proposons permet de réduire l'espace de recherche de la planification, en effectuant un filtrage des éléments, grâce aux relations topologiques identifiées. De plus notre méthode permet de détecter des chemins existant entre des surfaces qui peuvent être déconnectées dans l'espace et dans le temps. L'information temporelle permet d'identifier des chemins utilisant les mouvements propres des objets dynamiques pour présenter de nouvelles opportunités de navigation à un agent. L'observation des modifications dans la topologie de l'environnement va maintenir la représentation de la topologie à jour, mais également permettre de modifier les choix de navigation de l'agent au fil de la simulation. Enfin, notre méthode de planification locale permet de générer des trajectoires plus fines anticipant et évitant les obstacles statiques et dynamiques, dans le domaine spatio-temporel, en se basant une fois encore sur les relations d'obstructions identifiées par le représentation dynamique de la topologie.

Nos travaux ont permis de mettre en place une structure permettant à un agent de naviguer au sein d'un environnement dynamique dont l'évolution n'est pas connue a priori. À la différence des approches actuelles, les éléments dynamiques de l'environnement sont considérés à la fois comme des obstacles, mais également comme des zones navigables de l'environnement. Cela permet la gestion de nouvelles situations et de nouveaux environnements. En utilisant les informations temporelles et les observations faites sur l'évolution, nous avons présenté une méthode permettant de déterminer et

d'exploiter des propriétés temporelles au sein d'un environnement non connu à l'origine. Notre représentation exploite l'ensemble des éléments navigables présents afin de résoudre au mieux un problème de planification donné. Elle permet, de ce fait, d'utiliser toutes les propriétés temporelles identifiées, ainsi que les objets dynamiques, afin de proposer une navigation opportuniste de l'agent au sein de son environnement. Enfin, notre méthode a été testée dans des applications interactives, permettant à l'utilisateur d'agir sur l'environnement pendant le déroulement de la simulation démontrant ainsi les performances apportées en temps de calcul.

Perspectives de recherche

Les recherches effectuées au cours de cette thèse présentent la navigation au sein d'environnements virtuels dynamiques sous un nouvel angle. De ce fait, de nombreuses extensions et applications nouvelles peuvent être envisagées afin de faire suite à ces travaux. Nous présentons ici quelques perspectives.

1- Passage à l'échelle et performances

La méthode actuelle a été mise en place pour un agent unique naviguant au sein d'environnements virtuels dynamiques et non-connus a priori. Il serait maintenant intéressant de passer à une échelle supérieure sur plusieurs points tels que la prise en charge d'environnements de plus grandes dimensions, ou la gestion de multiples agents aux capacités de mouvements différentes. Il serait aussi possible de créer des représentations pouvant évoluer au cours du temps permettant de développer les capacités au cours de la simulation, ou de les réduire, afin de gérer des paramètres tels que l'apprentissage ou la fatigue.

Toutefois, le passage à l'échelle va augmenter le nombre de Volumes d'Interaction à considérer, et à suivre, afin de maintenir une représentation valide de la topologie. De nombreuses méthodes sont aujourd'hui implémentées en multi-processus, afin de pouvoir paralléliser les calculs effectués, et d'augmenter ainsi les performances. Ces méthodes de parallélisation sur CPU et GPU ont notamment montré des gains importants de temps de calcul pour les algorithmes de détection de collision [Avr11, PKS10] mais également pour les algorithmes de planification de chemin [PLM10, DSC11]. L'utilisation de telles méthodes de parallélisation serait intéressante afin d'améliorer les performances de notre algorithme, notamment au niveau de la détection de collision, afin de conserver des performances compatibles avec des applications interactives lors de notre passage à l'échelle.

Il sera également possible de s'intéresser à la définition des Volumes d'Interaction. En effet, la prise en compte de nouvelles capacités de déplacement plus complexes pourra amener à la construction de nouveaux volumes. Il sera donc intéressant de proposer des structures d'optimisation afin d'améliorer la détection des interactions comme, par

exemple, la création de sur-volumes qui englobent un ensemble de Volumes d'Interaction et permettent un pré-filtrage des relations topologiques.

2- Couplage de notre méthode avec des méthodes existantes

Notre méthode permet la détection, sans connaissance a priori, de relations topologiques temporelles dans l'environnement dynamique. Toutefois, l'absence totale de connaissances a priori sur l'évolution de l'environnement permet d'utiliser seulement les relations périodiques lors de la planification de chemin. Il serait intéressant d'enrichir notre modèle à l'aide d'hypothèses sur le mouvement des objets présents. Dans le cas d'une voiture circulant sur une route, il est par exemple possible d'émettre l'hypothèse que la voiture va continuer son mouvement le long de cette route et d'utiliser cette hypothèse durant la planification de chemin.

Afin de tester les différents modules de notre méthode, nous envisageons de comparer différentes types de cartes de cheminement locales à associer aux Surfaces Navigables des éléments. Comparer différentes méthodes de représentation de la topologie locale offre plusieurs avantages. Cela nous aidera à déterminer leurs performances respectives, mais également les domaines d'applications dans lesquels elles seront les plus efficaces. De plus, il serait également intéressant de considérer des méthodes d'animation, de type Motion Graphs ou Motion Controllers lors de la connexion de surfaces navigables afin de gérer la transition de l'agent entre différentes surfaces [LLKP11]. En anticipant les connexions dans l'environnement, il pourrait ainsi être possible de rechercher la meilleure transition possible pour l'agent à travers une banque de mouvements.

3- Raisonnement spatial

Certains algorithmes de planification se sont récemment intéressés au raisonnement spatial et à la possibilité que l'agent puisse lui-même interagir avec son environnement. Il peut ainsi se construire un chemin au sein d'un environnement très contraint où de nombreux obstacles obstruent l'espace navigable [SK04]. Un développement de notre méthode dans ce sens pourrait permettre de gérer de nouvelles situations. En effet, un agent immergé au sein d'un environnement, et connaissant les opportunités offertes par les éléments présents, serait alors capable d'agencer de manière autonome son environnement en créant notamment des accessibilités dans l'environnement lui permettant d'augmenter ses possibilités de navigation. Les modalités de représentation proposées au cours de cette thèse pourraient aider à cette construction, et amener ainsi à résoudre de nouveaux types de puzzles spatiaux.

Publication et communications

Publications

[P1] Thomas LOPEZ et Fabrice LAMARCHE. "ToD & DyP, navigation en temps réel en environnement dynamique." Actes des journées de l'Association Française d'Infomatique graphique, 2009

[P2] Thomas LOPEZ, Fabrice LAMARCHE et Tsai-Yen LI. "A planning solution for efficient navigation in changing environments" dans *Workshop on the Algorithmic Foundations of Robotics*, 2010 *Poster*

[P3] Thomas LOPEZ, Fabrice LAMARCHE et Tsai-Yen LI. "Space-time planning in dynamic environments with unknown evolution" dans *4th International Conference on Motion In Game*, pages 317-329, 2011

[P4] Thomas LOPEZ, Fabrice LAMARCHE et Tsai-Yen LI. "Space-time planning in changing environments : using dynamic objects for accessibility" dans *Computer Animation and Virtual Worlds*, *to appear*, *Journal*

Communications

[C1] Thomas LOPEZ. Planification de chemin et adaptation de posture en environnements dynamiques pour le *Centre d'Initiation à l'Enseignement Supérieur (CIES) du Grand Ouest*, 2009

[C2] Thomas LOPEZ, Fabrice LAMARCHE et Romain THOMAS. ToD & DyP *Amplitel*, Mai 2011

Bibliographie

[APD11] R. Alterovitz, S. Patil, and A. Derbakova. Rapidly-exploring roadmaps : Weighing exploration vs. refinement in optimal motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference*, pages 3706–3712. IEEE, 2011. 20

[Avr11] Quentin Avril. *Détection de Collision sur Environnements Large Échelle : Modèle Unifié et Adaptatif sur Architectures Multi-coeur et Multi-GPU*. Computer science, INSA de Rennes, September 2011. 67, 122

[BA02] O. Burchan Bayazit and N.M. Amato. Roadmap-based flocking for complex environments. *10th Pacific Conference on Computer Graphics and Applications 2002*, pages 104–113, 2002. 17

[BK00] R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA'00, IEEE International Conference*, volume 1, pages 521–528. Ieee, 2000. 25

[BV02] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference*, volume 3, pages 2383–2388. Ieee, 2002. 23, 24

[Can88] John Canny. Some algebraic and geometric computations in PSPACE, 1988. 4, 11

[CLH+05] H. Choset, KM Lynch, S. Hutchinson, G. Kantor, W. Burgard, LE Kavraki, and S. Thrun. *Principles of robot motion : theory, algorithms, and implementation*. the MIT Press, 2005. 10

[CLS03] Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2) :182–203, April 2003. 36, 53

[DSC11] D. Devaurs, T. Siméon, and J. Cortés. Parallelizing rrt on distributed-memory architectures. In *Robotics and Automation (ICRA), 2011 IEEE International Conference*, pages 2261–2266. IEEE, 2011. 122

[FKS06] D. Ferguson, N. Kalra, and A. Stentz. Replanning with rrts. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference*, pages 1243–1248. IEEE, 2006. 23, 24

[FS98] P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, 17(7) :760–772, July 1998. 28, 29

[FS06] D. Ferguson and A. Stentz. Using interpolation to improve path planning : The field d* algorithm. *Journal of Field Robotics*, 23(2) :79–101, 2006. 30

[GKX07] Russell Gayle, Kristopher R. Klingler, and Patrick G. Xavier. Lazy Reconfiguration Forest (LRF) - An Approach for Motion Planning with Multiple Tasks in Dynamic Environments. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, (April) :1316–1323, April 2007. 25

[GO07] R. Geraerts and M.H. Overmars. The corridor map method : Real-time high-quality path planning. In *Robotics and Automation, 2007 IEEE International Conference*, pages 1023–1028. IEEE, 2007. 17, 18

[GSLM07] Russell Gayle, Avneesh Sud, Ming C. Lin, and Dinesh Manocha. Reactive deformation roadmaps : motion planning of multiple robots in dynamic environments. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3777–3783, October 2007. 26, 27

[HKL+99] Kenneth E. Hoff, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, pages 277–286, 1999. 17, 18

[HNR68] P. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 2, 1968. 90

[JS04] L. Jaillet and T. Simeon. A PRM-based motion planner for dynamically changing environments. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1606–1611, 2004. 26, 42

[Kal10] M. Kallmann. Shortest paths with arbitrary clearance from navigation meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 159–168. Eurographics Association, 2010. 15

[KBT03] M. Kallmann, H. Bieri, and D. Thalmann. Fully dynamic constrained delaunay triangulations. *Geometric Modelling for Scientific Visualization*, 3, 2003. 15

[KG03] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 214–224. Eurographics Association, 2003. 44

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3) :1–10, July 2002. 35, 44

[KHI+07] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and Richard Rowe. Collision detection : A survey. *International Conf. on Systems, Man and Cybernetics*, 2007. 67

[KL94] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration for fast path planning. *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2138–2145, 1994. 17

[KL00] J.J. Kuffner and S.M. LaValle. RRT-connect : An efficient approach to single-query path planning. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, (Icra) :995–1001, 2000. 20

[KM04] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, (April) :4399–4404 Vol.5, 2004. 26

[KSLO96] LE Kavraki, P Svestka, JC Latombe, and M OverMars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996. 17

[Kuf98] J. Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. *Modelling and Motion Capture Techniques for Virtual Environments*, pages 171–186, 1998. 12, 34, 42

[Kuf04] J.J. Kuffner. Efficient optimal search of euclidean-cost grids and lattices. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4466–4471. Citeseer, 2004. 12

[KWP+11] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference*, pages 1478–1483. IEEE, 2011. 20

[Lam09] F. Lamarche. TopoPlan : a topological path planner for real time human navigation under floor and ceiling constraints. *Computer Graphics Forum*, 28(2) :649–658, April 2009. 15, 35

[Lat91] J. C. Latombe. *Robot Motion Planning*. Boston : Kluwer Academic Publishers, Boston, 1991. 10, 12

[LaV98] S.M. LaValle. Rapidly-exploring random trees : A new tool for path planning. (98-11), 1998. 20

[LaV06] Steven M. LaValle. Planning Algorithms. *Complexity*, 2006. 10, 14

[LC03] Tsai-yen Li and Pei-feng Chen. Motion planning for humanoid walking in a layered environment. *2003 IEEE International Conference on Robotics and Automation*, pages 3421–3427, 2003. 12

[LCL06] K.H. Lee, M.G. Choi, and J. Lee. Motion patches : building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics (TOG)*, 25(3) :898–906, 2006. 35, 37

[LD04] Fabrice Lamarche and Stephane Donikian. Crowd of Virtual Humans : a New Approach for Real Time Navigation in Complex and Structured Environments. *Computer Graphics Forum*, 23(3) :509–518, September 2004. 15

[LG98] M. C. Lin and S. Gottschalk. Collision detection between geometric models : a survey. In *Proceedings of the 8th Conference on the Mathematics of Surfaces*, pages 37–56, 1998. 67

[Li02] Tsai-yen Li. An incremental learning approach to motion planning with roadmap management. *Proceedings 2002 IEEE International Conference on Robotics and Automation*, pages 3411–3416, 2002. 25

[Li04] Tsai-yen Li. Planning humanoid motions with striding ability in a virtual environment. *IEEE International Conference on Robotics and Automation, ICRA 2004*, pages 3195–3200 Vol.4, 2004. 12, 34, 42, 53

[LK05] Manfred Lau and James J. Kuffner. Behavior planning for character animation. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05*, page 271, 2005. 29, 36, 37

[LK06] M. Lau and J.J. Kuffner. Precomputed search trees : planning for interactive goal-driven animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 299–308. Eurographics Association, 2006. 29, 34, 36, 37, 42

[LLKP11] Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. Space-time planning with parameterized locomotion controllers. *ACM Transactions on Graphics*, 30(3) :1–11, May 2011. 31, 32, 37, 39, 53, 123

[Lp83] Tomas Lozano-perez. Spatial Planning : A Configuration Space Approach. *IEEE Transactions on Computers*, C-32(2) :108–120, February 1983. 10

[OS94] M.H. Overmars and P. Svestka. A probablisitic learning approach to motion planning. *UU-CS*, (1994-03), 1994. 17

[PKS10] S. Pabst, A. Koch, and W. Straßer. Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. In *Computer Graphics Forum*, volume 29, pages 1605–1612. Wiley Online Library, 2010. 122

[PL11] M. Phillips and M. Likhachev. Sipp : Safe interval path planning for dynamic environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference*, pages 5628–5635. IEEE, 2011. 30

[PLM10] J. Pan, C. Lauterbach, and D. Manocha. g-planner : Real-time motion planning and global navigation using gpus. In *AAAI Conference on Artificial Intelligence*, 2010. 122

[PLS03a] J. Pettre, J. Laumond, and T. Simeon. 3D Collision Avoidance for Digital Actors Locomotion. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, pages 400–405, 2003. 34, 35, 42

[PLS03b] J. Pettré, J.P. Laumond, and T. Siméon. A 2-stages locomotion planner for digital actors. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 258–264. Eurographics Association, 2003. 34, 35

[PLT05] J. Pettre, J.P. Laumond, and D. Thalmann. A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *First International Workshop on Crowd Simulation*, pages 81–90. Citeseer, 2005. 14

[Rei79] John H. Reif. Complexity of the mover's problem and generalizations, October 1979. 4, 11

[RSM+09] Radu Bogdan Rusu, Aravind Sundaresan, Benoit Morisset, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe, and Michael Beetz. Leaving flatland : Efficient real-time 3d perception and motion planning. *Journal of Field Robotics (Special Issue on 3D Mapping)*, 2009. 44

[SAC+07] Avneesh Sud, Erik Andersen, Sean Curtis, Ming Lin, and Dinesh Manocha. Real-time Path Planning for Virtual Agents in Dynamic Environments. *2007 IEEE Virtual Reality Conference*, pages 91–98, March 2007. 26, 28

[SAC+08] Avneesh Sud, Erik Andersen, Sean Curtis, Ming C Lin, and Dinesh Manocha. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *IEEE transactions on visualization and computer graphics*, 14(3) :526–38, 2008. 26, 28

[SGA+07] Avneesh Sud, Russell Gayle, Erik Andersen, Stephen Guy, Ming Lin, and Dinesh Manocha. Real-time navigation of independent agents using adaptive roadmaps. *Proceedings of the 2007 ACM symposium on Virtual reality software and technology - VRST '07*, 1(212) :99, 2007. 26, 27, 28

[SGLM03] Brian Salomon, Maxim Garber, Ming C. Lin, and Dinesh Manocha. Interactive navigation in complex environments using path planning. *Proceedings of the 2003 symposium on Interactive 3D graphics - SI3D '03*, page 41, 2003. 19

[SH07] A. Safonova and J.K. Hodgins. Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 papers*, pages 106–es. ACM, 2007. 35, 53

[SK04] M. Stilman and J. Kuffner. Navigation among movable obstacles : Real-time reasoning in complex environments. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference*, volume 1, pages 322–341. IEEE, 2004. 123

[SLN00] T. Simeon, J.P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6) :477–493, 2000. 19

[Ste95] A. Stentz. The focussed d* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1652–1659. Citeseer, 1995. 95

[SYN01] Z. Shiller, K. Yamane, and Y. Nakamura. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, pages 1–8, 2001. 12, 34, 42

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, et al. Collision detection for deformable objects. In *CGF*, volume 24, 2005. 67

[vdBFK06] J. van den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, (May) :2366–2371, 2006. 28, 30, 88, 95

[vdBM08] Jur van den Berg and Dinesh Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935, May 2008. 28, 29

[WL08] K. Wong and C. Loscos. Hierarchical path planning for virtual crowds. *Motion in Games*, pages 43–50, 2008. 16

[ZKB07] Matt Zucker, James Kuffner, and Michael Branicky. Multipartite RRTs for Rapid Replanning in Dynamic Environments. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1603–1609, April 2007. 23, 24

Résumé

Les mondes virtuels sont aujourd'hui utilisés dans de nombreux domaines applicatifs. Ces mondes virtuels sont généralement peuplés à l'aide d'agents autonomes qui rendent ces environnements plus vivants. L'autonomie de ces agents va reposer en grande partie sur leur capacité à naviguer au sein de l'environnement virtuel. Cette capacité de navigation est cruciale puisqu'elle va permettre à un agent d'aller à la découverte de son environnement, et d'augmenter ses possibilités d'action au sein du monde.

La navigation des agents virtuels est donc un centre d'intérêt commun à de nombreux domaines utilisant des environnements virtuels. Dans le cadre d'applications interactives, telles que le jeu vidéo, l'accent est mis sur les performances de calcul de la méthode. À l'inverse, dans le cadre de méthodes de production, telles que le cinéma d'animation, le résultat final sera en général produit hors-ligne mais des méthodes de prévisualisation rapides sont utilisées afin d'obtenir à l'avance une idée du résultat final.

Dans le cadre de cette thèse, nous avons proposé une nouvelle solution de planification en environnements dynamiques. Ceux-ci possèdent une configuration qui va évoluer au cours du temps de manière non-connue a priori. Afin de répondre à cette problématique de planification de chemin, nous avons tout d'abord introduit une nouvelle représentation des objets définissant explicitement les interactions existantes entre un objet et un agent virtuel, au regard des capacités de celui-ci. Nous avons ainsi proposé une solution innovante en considérant ces objets dynamiques, à la fois comme des obstacles, mais également comme des éléments navigables qu'un agent pourra utiliser au cours de sa navigation. Nous avons également défini une structure de représentation et de suivi dynamique de la topologie. Cette structure permet de déterminer, sans connaissance a priori, les propriétés temporelles des accessibilités et des obstructions présentes dans l'environnement. Cela permet de construire une vision d'ensemble de la topologie temporelle de l'environnement. Nous proposons enfin un algorithme de planification de chemin, utilisant l'ensemble des informations temporelles issues de l'environnement, afin de proposer une solution de navigation à l'agent au sein de cet environnement virtuel. Notre méthode, à l'issue de cette thèse, permet ainsi à un agent navigant d'identifier de manière autonome un chemin au sein d'un environnement dynamique composé d'éléments navigables déconnectés à la fois dans le temps et dans l'espace. De plus, afin de répondre aux contraintes temporelles de nombreux domaines, nous proposons une solution performante et compatible avec des applications interactives.

Abstract

Nowadays, many applications are using virtual worlds. Those virtual worlds are generally populated with autonomous virtual characters which bring those environments to life. The autonomy of these virtual characters rely on their ability to navigate within a considered virtual environments. This navigation capability is essential as it will allow the character to discover its surrounding environment and to augment its potential activities within this environment.

The virtual characters navigation is a well-known problem in many application fields using virtual environments. On the one hand, when dealing with interactive applications such as video games, the focus is set on the computation time in order to have a fast result. On the other hand, in the context of production methods, such as the animation movies, the final rendering will be produced offline but fast preview methods are generally used to obtain a first idea of the final result while avoiding heavy computation costs.

In this thesis we proposed a new path planning solution for dynamic environments. The dynamic environments we consider have evolving configuration over time. This evolution is not known a priori which makes the navigation problem even harder. In order to address this problem, we first introduced a new representation of the objects which defines explicitly interactions between this object and a considered character, regarding its navigation capabilities. We have proposed an innovative solution by considering these dynamic objects both as obstacles and as a navigable element that a character can use during its navigation task. We then proposed a representation structure to catch entirely the topology of the global environment. We add to this representation temporal information concerning detected accessibilities and obstructions in the environment. We thus obtain , with no a priori knowledge, an anytime representation of the global topology which allows to deduce temporal properties of the environment. Finally, we present a path planning algorithm using all the temporal information available to compute a navigation solution for the virtual character in the dynamic environment. During this thesis, we designed a global method that allows a virtual character to autonomously identify a path within a dynamic environment populated of navigable elements disconnected both in space and time. Moreover, according to our computation time constraints, our method is efficient and compatible with interactive applications.

INSA RENNES

Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coësmes . CS 14315 . F-35043 Rennes Cedex
Tel : 02 23 23 82 00 – Fax : 02 23 23 83 96