Apport des méthodes de planification automatique dans les simulations interactives d'industrialisation et de maintenance en réalité virtuelle

Nicolas Ladeveze

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Systèmes Automatiques

Présentée et soutenue par :
Nicolas LADEVEZE

le : lundi 19 avril 2010

Titre :

Apport des méthodes de planification automatique dans les simulations interactives d'industrialisation et de maintenance en réalité virtuelle

---

JURY
Sabine COQUILLART - Directrice de Recherche INRIA
René ZAPATA - Professeur des Universités LIRMM
Xavier FISCHER - Enseignant Chercheur ESTIA
Jean Paul LAUMOND - Directeur de Recherche LAAS-CNRS
Michel TAIX - Maître de conférences LAAS-CNRS

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :

Equipe Production Automatisée du Laboratoire Génie de Production (LGP-ENIT)

Directeur(s) de Thèse :

M. Jean -Yves Fourquet - Professeur des Universités ENIT-LGP

Rapporteurs :
René ZAPATA - Professeur des Universités LIRMM
Xavier FISCHER - Enseignant Chercheur ESTIA

Apports des méthodes de planification automatique dans les simulations interactives d'industrialisation et de maintenance en réalité virtuelle

Nicolas Ladeveze

27 mai 2010

Remerciements

Je tiens tout d'abord à remercier chaleureusement mon directeur de thèse, Monsieur Jean Yves Fourquet, professeur des Universités au LGP-ENIT, pour m'avoir encadré et guidé autour de ce sujet passionnant. Je tiens à souligner la grande qualité de son encadrement, sa patience et sa capacité à transmettre sa passion pour la recherche scientifique. Je tiens également à remercier mon directeur industriel Bernard Puel, Manager Conception et Industrialisation sur le site d'Alstom Tarbes, pour son appui ainsi que l'apport important de sa vision pragmatique de mon projet de thèse.

J'aimerais remercier ensuite les deux rapporteurs de cette thèse, Monsieur René Zapata, Professeur des Universités au LIRMM et Xavier Fisher, Enseignant Chercheur à l'ESTIA. Leurs remarques et encouragements m'ont particulièrement aidé à franchir le dernier cap de la thèse.

Je remercie également les examinateurs du jury de thèse :

Monsieur Jean Paul Laumont, Directeur de Recherche au LAAS-CNRS, pour avoir bien voulu présider le jury de thèse, Madame Sabine Coquillart, Directrice de Recherche à L'INRIA, et Monsieur Michel Taix, maître de conférences au LAAS-CNRS, qui m'a fait l'honneur de m'encadrer durant mon master de recherche et qui à continué à partager ses idées avec moi tout au long de cette thèse.

Je tiens également à remercier particulièrement, Benoit Reculeau, dont les travaux de master ont profondément guidé ma réflexion sur la partie applicative de cette thèse, Adel Sghaier, qui m'a énormément aidé sur toute la partie traitement de maillage et qui m'a permis de développer mes connaissances sur les formats 3d, ainsi qu' Alexandre Delan pour son aide, sa disponibilité et sa patience à répondre à toutes mes questions sur son forum spécialisé sur Virtools.

J'aimerais aussi remercier mes compagnons et collègues doctorants du LGP-ENIT, Arnaud, Fabien, Valentin, Marina, Mickael, Vincent, Benjamin, Mathieu, Clélia, Adrien, et tous les autres, pour leur soutient et leur amitié.

Enfin, je remercie et j'embrasse ma famille, pour leur amour et leurs encouragements.

Table des matières

Liste des symboles

Acronymes

AABB Axis Aligned Bounding Box, boite englobante d'un objet dont les normales des faces sont constitués par les axes principaux du monde virtuel.

DFQ Design For Quality - Cycle en V adapté à la production de produits dans la société Alstom Transport S.A.

PLM Product Life cycle Management
Approche qui tend à mettre en cohérence le cycle de vie d'un produit depuis la création du concept jusqu'à sa mise en service et son démantèlement en passant par les phases de conception et de fabrication.
Par extension, ensemble de logiciels permettant la gestion des données numériques relatives aux produits.

EPFL Ecole Polytechnique Fédérale de Lausanne

HMD Head Mounted Display

IPP Interactive Physic Pack

IPSI Interactive Physics Simulation Interface

IVH Interface Visuo Haptique

LOD Level Of Detail

PMI Product Manufacturing Information

UQO Université du Québec en Outaouais

VPS Voxmap Point Shell

VRLab Virtual Reality Lab

VRPN Virtual Reality Peripheral Network

Chapitre 1

Introduction Générale

Sommaire

1.1 Motivations et contexte des travaux

Les travaux présentés dans ce mémoire ont été réalisés en partenariat entre le Laboratoire Génie Production (ENIT-LGP) et la société ALSTOM Transport S.A. Le sujet "Apports des méthodes de planification automatique dans les simulations interactives d'industrialisation et de maintenance en réalité virtuelle" concerne l'introduction des techniques et des outils de simulation interactive dans le domaine de la conception des convertisseurs de puissance.

La conception des systèmes de haute technologie induit une complexité directement liée au nombre, toujours croissant, des composants mis en présence dans un volume toujours plus réduit. Cette complexité purement géométrique s'accompagne d'une complexité liée aux différentes contraintes physiques imposées par le comportement actif de certains de ces composants. De plus, la conception de tels ensembles constitue un réel challenge car elle doit également prendre en compte toutes les étapes de la vie du produit : son assemblage, sa valorisation, son fonctionnement, sa maintenance, et son démantèlement. Dans ce contexte, le concepteur doit assurer l'optimisation d'un système multi-composants, par une approche multi-métiers, multi-physique,

et multi-processus. ll doit également être en mesure de fournir les éléments pour la certification, d'assurer la traçabilité des essais et des tests réalisés. Enfin, sa démarche doit pouvoir être ré-utilisable par morceaux - être modulaire - afin d'optimiser la déclinaison en gamme des produits et des processus en termes de coût direct ou de capitalisation de l'expérience.

La démarche de conception s'appuie traditionnellement sur l'application de cycles en V de conception. Celle-ci intègre l'utilisation de moyens de simulation numériques mais demeure fortement dépendante des études réalisées sur prototype physique. Néanmoins, les progrès réalisés dans le domaine de la simulation et le fait que la plupart des composants sont définis à l'aide d'outils de CAO permettent d'envisager de réaliser de nombreuses validations de processus directement à partir des définitions numériques d'un équipement. Cette approche requiert, en phase de définition, une description la plus complète possible des sous ensembles mis en jeu et son intégration dans une démarche de type PLM, et, à l'usage, la capacité d'accéder de façon sélective aux données requises par les outils de simulation. En effet, la définition d'une pièce ou d'un système pour la conception intégrée ne se limite plus à la mise en plan 3D. Il faut associer à l'objet différentes fonctions, différentes représentations pour permettre une description efficace des processus à simuler.

La numérisation de la démarche de conception intégrée - débutée avec les premiers outils de CAO dans les années 1970, donne ainsi naissance à de nombreuses questions. Quels processus peut-on numériser ? A quel coût d'achat et au prix de quel investissement humain ? Avec quelle pertinence et pour quel public ? Quelle cohérence entre le prototypage virtuel et les expérimentations ? Ces questions se posent à la fois du point de vue scientifique, en termes de faisabilité, mais aussi dans le champ industriel, en termes d'efficacité et d'intégration à un schéma de développement à moyen terme.
Du point de vue industriel, un des objectifs prioritaires est la diminution du nombre et du rôle des prototypes physiques. Il faut donc déterminer quelles fonctions peuvent être testées virtuellement :

- en termes de coût : coût d'achat ou de développement des outils, coût de développement des fonctions, qualifications requises pour ces développements, coût de maintenance d'un système cohérent
- en termes de pertinence : le virtuel dit-il l'essentiel du réel ? Autant que le prototype physique ? Quel niveau de validation, de certification atteint-on ?
- sur le plan humain : quels seront les impacts humains des modifications du process de conception existant ?

Et plus particulièrement, partout où le processus requiert l'action de l'homme, quels sont les processus humains qu'il est illusoire, au moins à moyen terme, de vouloir formaliser ou reproduire ?
D'un point de vue scientifique et technologique, les problèmes concernent à la fois la structure des données manipulées et les traitements associés : quelle part doit être automatisée ? Comment définir la modularité ? Comment organiser le partage des informations et des décisions entre les différents acteurs humains et les tâches automatisables ? Comment associer le virtuel au réel ?

Une première catégorie d'outils de simulation permet de caractériser le comportement physique d'un système soumis à un contexte donné. Ces outils de calcul permettent d'approximer en simulation les différents phénomènes physiques (mécanique, thermique, électrique) qu'un équipement peut connaître. L'humain n'agit véritablement dans ce processus qu'au moment de la définition des hypothèses de calcul, de manière algorithmique. Ensuite, il en analyse les résultats. Il n'est pas "dans la boucle". D'autres outils, ceux

du PLM, décrivent des processus pour lesquels l'humain modifie l'expérience, se trouve dans la boucle de la simulation. Il en est ainsi notamment de tous les processus qui décrivent l'action physique d'un opérateur humain sur un équipement. C'est le cas également de tous les processus de création de scénario par les concepteurs. Une première approche consiste à qualifier et quantifier les capacités de l'humain, opérateur et concepteur, et à prédire ses comportements afin de simuler des scénarios plausibles de manière automatique. C'est la démarche suivie notamment lorsque l'on cherche à prédire des mouvements d'opérateurs sur postes de travail, ou bien encore des stratégies de déplacement en milieu encombré. Cette approche bute sur l'incapacité à comprendre et donc à reproduire l'humain mais aussi sur la lourdeur des interfaces nécessaires pour établir la description des scénarios, dans un cadre algorithmique. Elle se prive de toute l'expérience non formelle. L'autre approche consiste à utiliser toutes les capacités psycho-motrices de l'humain pour lui permettre, de manière intuitive, de créer des scénarios ou de produire des mouvements, en environnement numérique. Dans ce cadre, il est nécessaire de forger les outils logiciels qui rendront intuitive et efficace la création interactive de scénarios qui tirent parti de l'expérience humaine.

L'opération de manipulation constitue une opération fondamentale dans la définition des processus industriels car elle est instanciée dans différentes étapes du cycle en V de conception que ce soit dans les phases amont, d'étude de la complexité du montage ou de la maintenabilité, ou dans des phases plus en aval de définition des processus de fabrication ou encore de formation du personnel. Cette opération est typique de la difficulté à automatiser (à formaliser) les tâches complexes qui requièrent l'expérience et la dextérité de l'humain. En effet, ces deux caractéristiques humaines ouvrent le champ à des scénarios d'une grande variabilité, entre individus et pour chaque individu, ou l'adaptabilité, naturelle et résultant de l'apprentissage, de l'homme a rendu inutile une formalisation pas à pas des processus. On peut d'ailleurs observer que si les outils logiciels inclus aux suites logicielles PLM permettent de réaliser la planification de tâches lorsque les ressources de manipulation sont des systèmes robotisés, ces mêmes outils n'apportent aujourd'hui qu'une réponse partielle à la planification de tâches humaines.

Les travaux présentés dans cette thèse sont centrés sur la simulation des processus qui induisent le mouvement ou la manipulation d'objets, composants ou systèmes. Ils induisent d'une manière ou d'une autre l'action motrice de l'humain - du concepteur ou de l'opérateur - vis à vis de la maquette numérique. Les caractéristiques fondamentales de ces processus sont déclinés principalement sur des processus d'assemblage à des fins de test de faisabilité ou de formation. Ce type de problème est présent dans un grand nombre des processus qu'on souhaite simuler. Il est fondamental en ce sens qu'il pose naturellement la question de l'interaction temps-réel multi-sensorielle et de son intégration aux systèmes d'information disponibles dans l'industrie. Il pose la question de l'intégration et de la performance des techniques de planification automatique. Enfin, il est révélateur de la nécessité de développer des architectures qui permettent la collaboration temps réel et multi-modale entre algorithmes et humains, à différents niveaux de formalisation de l'activité humaine.

Du point de vue scientifique, cette problématique ouvre de nouveaux défis tout en nécessitant un effort d'intégration et une re-visite de concepts. Elle possède également une dimension technologique pour tout ce qui concerne les composants logiciels et matériels, les formats et protocoles, la compatibilité et l'interopérabilité des différents modules et outils. Dans cette thèse, la préoccupation industrielle de "trouver

une solution efficace dans le PLM pour décrire des processus d'assemblage" rencontre le défi scientifique qui consiste à mettre en oeuvre un schéma de coopération qui permette à un opérateur immergé en réalité virtuelle d'échanger des informations sensori-motrices avec un planificateur automatique de chemins. La mise en oeuvre de cette solution coopérative nécessite la définition d'un environnement, d'une architecture logicielle ouverte. Ces deux aspects font l'objet du présent mémoire.

1.2 Processus industriels et cycle en V

Lors de la conception d'un équipement, la complexité induite par la prise en compte simultanée des contraintes nécessite la mise en oeuvre de stratégies permettant de mesurer l'état d'avancement d'une conception par la validation de différentes étapes du projet. Dans le cadre du développement de produits industriels, l'une des approches les plus utilisées repose sur l'utilisation d'un cycle en V de conception (cf figure 1.1). Cette approche permet, après une discrétisation du projet en différentes étapes stratégiques, d'une part à un comité de pilotage de surveiller l'état d'avancement de la conception d'un produit, mais aussi de limiter l'impact, en termes de coût et de temps, des défauts constatés au fur et à mesure de la validation d'une conception depuis le concept jusqu'au produit fini.

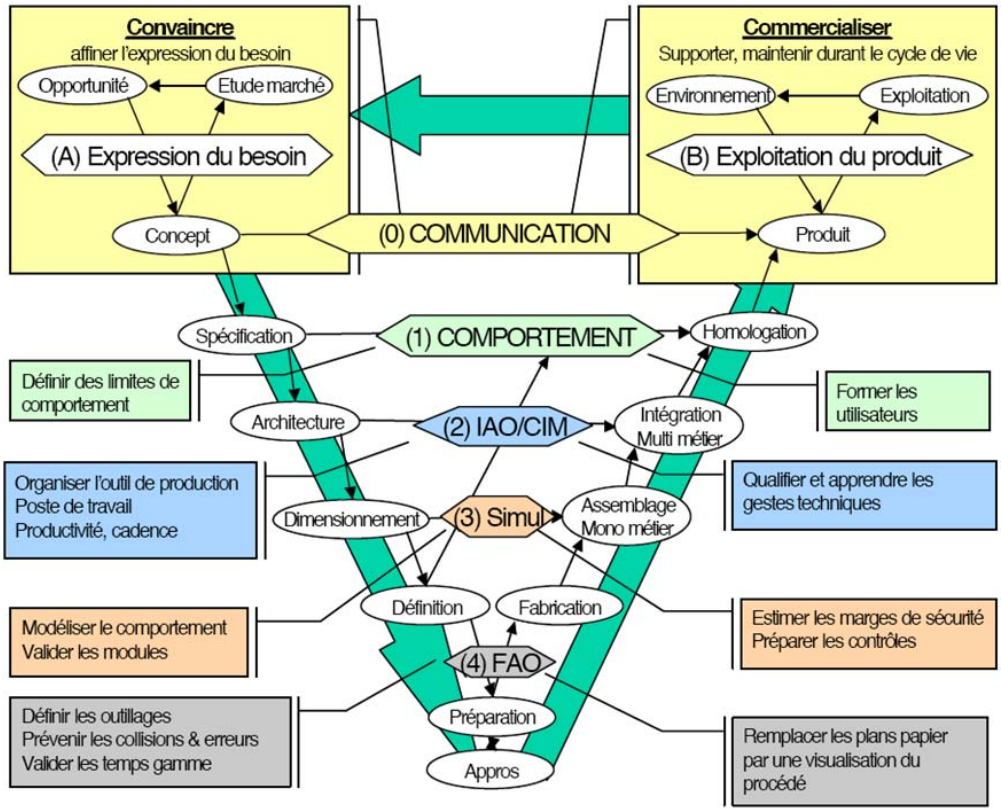


FIGURE 1.1 – Cycle en V de conception d'un produit (Source PERF-RV)

La conception d'un ensemble complexe tel qu'un convertisseur de puissance nécessite de mener des études faisant appel à des spécialistes dans des domaines de compétences divers pour aboutir à la définition d'un

produit fini répondant aux impératifs du cahier des charges. Elle implique donc le travail organisé de plusieurs ingénieurs sur différents modèles numériques afin d'aboutir à une solution. Dans la société Alstom transport, ces activités sont menées en suivant une méthode de conception standardisée appelée cycle "Design For Quality" (DFQ) analogue à un cycle en V et illustré par la figure 1.2.

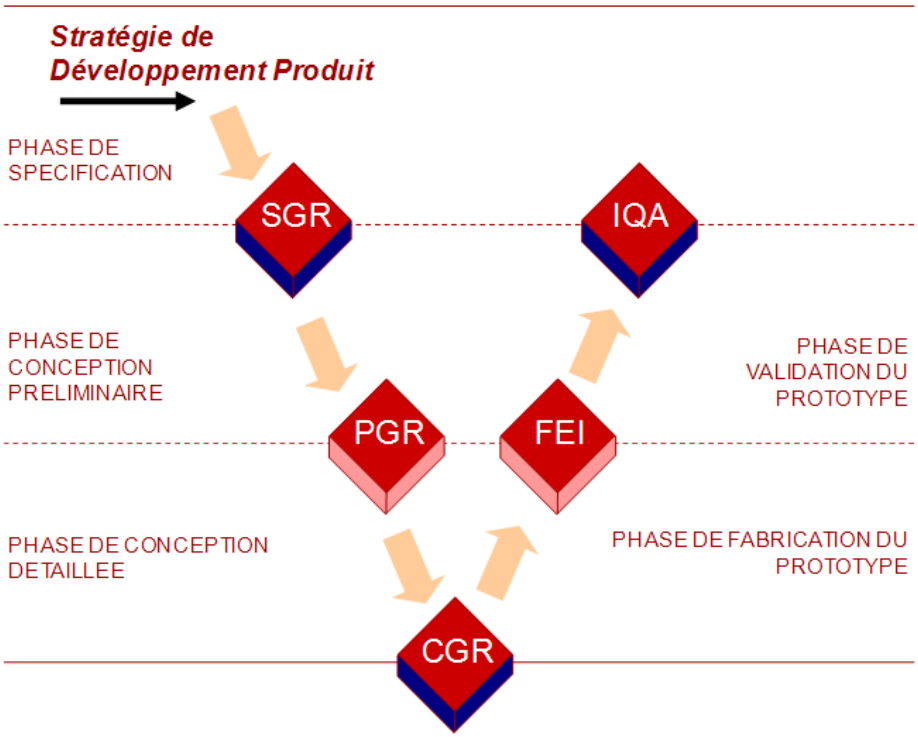


FIGURE 1.2 – Cycle en V appliqué chez Alstom Transport : Design For Quality

Ce mode de conception permet à partir de la spécification d'un cahier des charges de jalonner les différentes étapes de la conception d'un produit. Chaque étape du cycle aboutit à une "réunion de conception" permettant de valider l'avancement dans le processus de conception. Lors de ces réunions aussi appelées "Gate Review", différents aspects de la conception sont évalués afin de valider le passage vers l'étape suivante :

- La "Specification Gate Review" notée SGR (ou réunion de révision des spécifications) : elle permet de vérifier si toute les spécifications sont définies et acceptées par tous les participants et conformes à toutes les exigences contractuelles pour lancer concrètement la conception préliminaire.
- La "Preliminary Gate Review" (ou réunion de revue de conception préliminaire) : elle permet de vérifier que la conception préliminaire est à un niveau de validation tel que l'équipe projet est convaincue que les objectifs du projet peuvent être atteints et que la conception détaillée peut être lancée.
- La "Critical Gate Review" (ou réunion de revue de conception détaillée) : elle permet de valider que la conception détaillée permet d'atteindre les objectifs en respectant le budget fixé afin de lancer la phase de définition de l'industrialisation et la création d'un premier prototype.
- La "First Equipment Inspection" (ou réunion de revue de prototype) : elle permet de valider le processus de production en série par un bilan de la fabrication du ou des prototype(s) représentatif(s). A l'issue de cette réunion, on procède au lancement de la phase de test d'intégration grâce au prototype physique.

- La “Initial Quality Approval” (ou réunion de revue d’industrialisation) : elle permet de valider la conception définitive du produit et du processus de production en série.

Cependant, le cycle en V prévoit des retours possibles dans des phases amont lorsque la définition courante d’un produit ne permet pas d’avancer dans le processus de conception. Lors d’un retour dans les phases amont, la géométrie du produit est souvent rectifiée afin d’éviter le problème rencontré et les différentes revues de design sont re-franchies. Ces retours en phase amont sont souvent très coûteux, surtout lorsqu’il relèvent de la détection d’un défaut sur le prototype physique.

1.3 Outils PLM et cycle en V

Le PLM (Product Lifecycle Management) est l’approche qui tend à mettre en cohérence le cycle de vie d’un produit depuis la création du concept jusqu’à sa mise en service et son démantèlement en passant par les phases de conception et de fabrication. Le PLM a pour vocation d’intégrer les personnes, les données, les processus technologiques et financiers dans une gestion commune des informations.

Cette approche s’appuie sur des suites logicielles qui permettent notamment de réaliser un suivi des modifications apportées à la maquette numérique d’un produit du fait de nombreuses rectifications de conception effectuées au cours de développement ou de prendre en compte la gestion des différentes versions d’un même produit au cours des différentes phases de re-conception qu’il subit. En termes de processus, ces suites logicielles permettent également de décrire l’engagement des ressources sur les produits, de décrire par exemple une séquence d’assemblage robotisé ou l’analyse posturale d’un mannequin humain.

Parmi les objectifs de cette approche, on retrouve notamment la mise à jour (semi-)automatisée des différents éléments liés aux définitions :

- **de sa conception :**
  on retrouvera dans cette partie :
  - les différents *modèles 3D des composants* réalisés grâce à un logiciel de CAO. Ceux-ci sont soit directement disponibles au bureau d’étude de l’entreprise soit fournis par un fournisseur ou un sous traitant.
  - les *modèles de simulation numérique* utilisés pour la validation du produit du point de vue de ses caractéristiques physiques : notamment les modèles et résultats d’études de la résistance aux contraintes mécaniques, thermiques, et électromagnétiques mises en présence lors du fonctionnement du produit.
- **de sa fabrication :**
  la fabrication du produit est ordonnée en différentes phases d’assemblage. Suivant ce modèle, le PLM spécifie pour chaque phase d’assemblage du produit la liste exhaustive des *sous ensembles* et des *outils* utilisés, la définition du *poste de travail*, les *compétences* nécessaires à sa manipulation ainsi que les *notices de montages* associées.
- **de sa maintenance :**
  A partir de la définition des différents modules ou sous ensembles remplaçables, le PLM permet de définir chaque opération de maintenance du produit. Suivant une organisation similaire à la définition de son assemblage, il permet de discrétiser chaque opération de maintenance en différentes phases de manipulation

comprenant chacune une liste de sous ensembles manipulés, les outils et les procédures utilisées pour les remplacer.

L'un des enjeux réside donc dans la façon de créer, de stocker et d'échanger l'information relative aux processus mis en jeu dans ces différentes étapes du cycle de vie du produit. Que numérise-t-on ? Quels outils utilise-t-on pour créer efficacement de l'information relative aux processus ?
Alstom Transport, en s'appuyant sur le logiciel "UGS Siemens PLM Software" tend à améliorer la gestion de données numériques relative à ses produits et à augmenter la cohérence de sa chaîne d'information.

1.4 De nouveaux besoins qui découlent de l'application du PLM

Le développement de nouveaux convertisseurs de puissance implique une intégration de l'ensemble des constituants dans des volumes de plus en plus petits tout en maintenant le respect des spécifications. Cette intégration implique une augmentation de l'encombrement géométrique à l'intérieur du convertisseur qui génère des problématiques de montage et de démontage plus complexe à évaluer pour les concepteurs. Dans ce contexte, l'utilisation de la méthode de conception classique "DFQ", engendre des phases de re-conception successives après des défauts constatés sur les différents prototypes physiques réalisés notamment dans la phase FEI (cf. figure 1.2).

Ces défauts de conception du prototype sont souvent liés à des problématiques d'intégration, d'ergonomie de montage et de démontage du fait du manque d'outils permettant de les détecter plus tôt dans le processus de conception. Dans cette optique, Alstom Transport est intéressé par un outil de pré-conception permettant de réaliser un maximum de tests sur prototype virtuel afin de prendre en compte dans la phase de conception détaillée les problématiques de montage, de maintenance et d'intégration.

Le développement de ce genre d'outil nécessite la conception d'un environnement, de structures de stockage et de traitements de l'information compatibles avec les formats de sortie du PLM. Cependant, du fait du caractère multi-métiers de cette démarche, la conception d'un tel environnement nécessite de prévoir différents scénarios d'interaction avec des données communes, en fonction des préférences et des besoins de chaque intervenant au fur et à mesure de la validation.
Alstom Transport souhaite en particulier développer des outils de PLM interactifs permettant à un utilisateur néophyte du point de vue de la réalité virtuelle de :

- Diriger des revues de conception en 3d,
- Simuler de manière interactive et immersive l'assemblage et le démontage lors de phases de montage ou de maintenance,
- Former les intervenants au geste technique en environnement immersif,

1.4.1 Diriger des revues de conception en 3d

Lors des actuelles revues de conception, différents ingénieurs collaborent au travers de différentes présentations et discussions sur des supports hétérogènes suivant les goûts de chacun mais aussi suivant les habitudes

de chaque métier. En effet, un dessinateur en mécanique utilisera par exemple une capture d'écran tirée du logiciel *Catia*® qu'il illustrera par des plans tirés de coupes transverses pour illustrer ses propos. Un électrotechnicien préférera utiliser des schémas ou des résultats de calcul tirés de *Maxwell*®, tandis qu'un thermicien présentera des résultats de simulation tirés par exemple du logiciel *Fluent*®. Cette hétérogénéité des supports rend souvent difficile la communication d'informations au sein d'une équipe de conception, ainsi que l'analyse des couplages entre phénomènes électromagnétiques, thermiques et mécaniques inhérents à la conception d'un convertisseur de puissance.

L'une des principales attentes d'Alstom Transport est de pouvoir disposer d'un moyen interactif de simulation sur un support de discussions unique permettant une communication efficace entre des ingénieurs possédant des compétences dans des disciplines différentes. De plus, ce support doit permettre une analyse multi-échelle et multi-physique des produits afin de localiser et d'analyser les phénomènes de couplage entre les différentes physiques.

1.4.2 Simuler de manière interactive et immersive l'assemblage et le démontage

Lors de la conception, les phases de montage et de maintenance sont évaluées par les concepteurs grâce à leur "expérience métier" puis vérifiées lors de tests de montage, de démontage et de l'intégration au moyen d'un prototype physique. Ceci entraîne parfois, lors de la constatation de défauts, des retards et des sur-coûts de conception. Afin de valider rapidement les choix de conception des produits et des processus de montage et de démontage, la simulation interactive doit permettre une interaction en temps réel avec les modèles numériques reproduisant au mieux la manipulation d'un prototype physique.

1.4.3 Former les intervenants au geste technique en environnement immersif

La formation est un volet important de l'intégration des données. En effet, il est nécessaire bien souvent de faire de la formation sans arrêter la production, et même parfois avant la mise à disposition physique des composants. Il est donc nécessaire de disposer du modèle numérique des composants et d'outils permettant de simuler efficacement des scénarios induisant du mouvement. Les trajectoires ainsi que les séquences de montage et de démontage des différents sous ensembles doivent pouvoir être enregistrées afin de permettre une évaluation et une formation rapide au geste technique pour les futurs techniciens intervenants dans le montage ou la maintenance du produit. L'environnement doit également permettre une certaine autonomie de mouvement de l'opérateur tout en tirant parti des informations visuo-haptiques que peut fournir un environnement immersif.

1.5 Un composant essentiel : l'assistance au mouvement

Un grand nombre des processus qui requièrent habituellement un prototype physique vont impliquer le mouvement de composants les uns par rapport aux autres mais aussi le mouvement des ressources (humains, robots, transitique) nécessaire à leur déplacement. La modélisation de ces processus est un défi pour

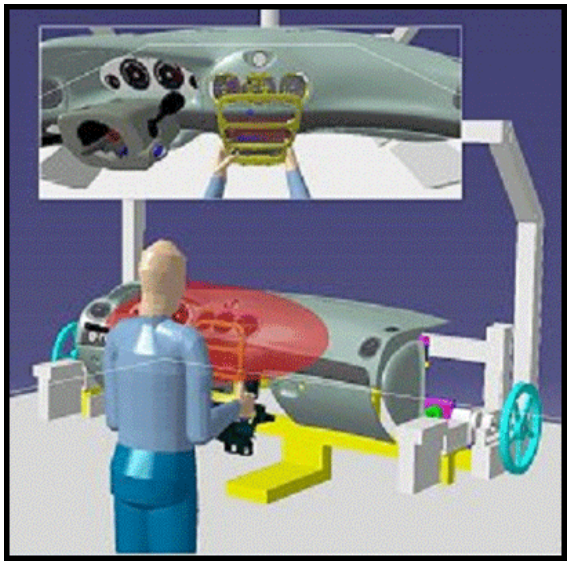
l'intégration dans une démarche PLM dans la mesure où ils sont apparemment sous-contraints et que leur exécution peut-être déclinée à l'infini. Une tâche confiée à un opérateur est décrite en "langage naturel". La codifier de manière unique revient à faire un choix d'exécution a priori. Inclure la description de ces processus dans les outils PLM revient à poser la question suivante : doit-on s'efforcer de coder une solution de référence unique ou doit-on créer l'environnement permettant de réaliser tous les scénarios possibles ?

L'intervention de l'humain à différentes étapes est essentielle. En particulier, le choix du point de vue, des conditions initiales et finales, de la façon de décrire la tâche requièrent des moyens d'interaction les plus intuitifs possibles.

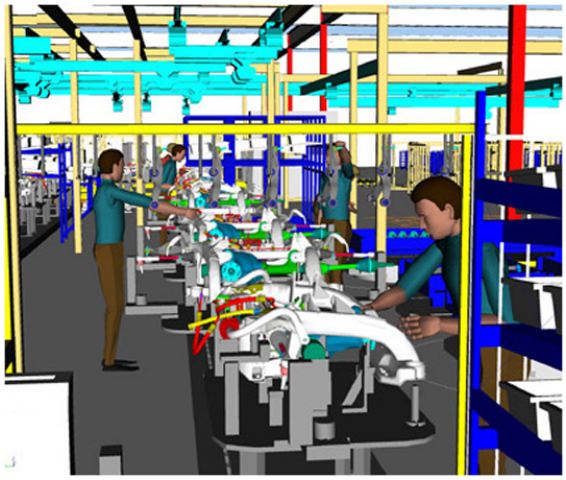
Une fois la tâche définie, elle se ramène de manière générique à un problème de génération de mouvement en environnement encombré. D'un point de vue strictement géométrique, augmenter l'intégration revient à augmenter l'encombrement à l'intérieur du convertisseur. Chaque opération de montage, de démontage ou d'inspection, par exemple, va être rendue plus difficile à planifier ou à analyser. On pourrait résumer ce type de problème à *la recherche d'un déplacement compte tenu de la définition d'une origine, d'un but, d'un ensemble de contraintes (de type égalité et inégalité, statique ou dynamique) et d'un ensemble de critères à optimiser.*

Il existe une littérature abondante issue du domaine de la robotique essentiellement pour les problèmes où ces contraintes sont géométriques. Cette problématique est alors résumée à la mise en place de méthodes de planification automatique de solutions [1]. Les méthodes disponibles offrent une planification efficace d'un point de vue probabiliste. Elles souffrent néanmoins d'un manque de généricité dans la mesure où la qualité du résultat est très dépendante du contexte.
Ces techniques, couplant planification de chemins et détection de collisions, se retrouvent dans certaines suites PLM utilisées dans l'industrie, tels que *Delmia*® (Dassault Systèmes) ou *Technomatix*® (UGS-Siemens) illustrés en figure 1.3. Elles permettent de réaliser des simulations de mouvement des pièces et des ressources dans la maquette numérique. Ces outils sont particulièrement efficaces lorsque les opérations simulées relèvent de manipulations robotisées ou lorsque la modélisation de l'action des opérateurs humains demeure, soit relativement sommaire (tests posturaux d'accessibilité simples, illustration de la présence d'opérateurs dans l'environnement), soit très spécialisée (posture de pilotage de véhicules par exemple). Ces outils permettent de simuler dans le PLM un certain nombre de processus de mouvement qui peuvent être intégrés à des phases de test ou de formation, par exemple. Il présentent l'avantage de préserver la continuité et l'intégrité des données avec la chaîne CAO. Au delà des questions de coûts, ces logiciels comportent certaines lacunes qui freinent leur dissémination. La plus importante est certainement la capacité à considérer les stratégies de mouvement humain.
En termes de prédiction de mouvement d'abord : bien souvent, l'action de l'humain est "illustrée" plus que "simulée" ; l'humanoïde est mis en situation dans le système de production ou encore en phase de maintenance, par exemple. Les aspects dynamiques sont éludés. Même d'un point de vue statique, la programmation de leur interaction avec l'environnement virtuel et la mise en posture réaliste restent longues et délicates, notamment du fait de l' absence de modèles de coordination du mouvement humain.
En termes d'interaction ensuite : ces outils fournissent une trajectoire de référence que l'on peut rejouer

mais avec laquelle il n'est pas - ou peu - possible d'interagir directement. On ne dispose pas de moyens de pilotage de la déformation de la référence qui permettraient une utilisation aussi bien en phase de test de faisabilité que de formation.



(a) Simulation avec le module Human *Delmia*© de Dassault Systèmes



(b) Simulation avec le module Jack de *Technomatix*© de UGS-Siemens [2]

FIGURE 1.3 – Évaluation des possibilités de montage et de démontage à l'aide de simulations numériques de processus

De manière plus générale, ce type d'approche ne fournit qu'une information *géométrique* alors que la tâche de mouvement est par nature *dynamique* et qu'il est nécessaire pour la simuler de rendre compte des *efforts*, notamment de contact.

Les techniques d'immersion en réalité virtuelle offrent une alternative aux simulateurs de processus classiques grâce à l'utilisation de périphériques de manipulation haptique ou pseudo-haptique ([3]). Elles permettent de faire entrer l'opérateur dans la boucle «perception-décision-action» principalement par le biais d'interactions temps réel visuelles et haptiques. On a ainsi accès à la dynamique du mouvement, en termes de déplacement et d'effort dans un contexte . Ces techniques, par nature intuitives et préservant un certain degré d'ergonomie, semblent donc particulièrement adaptées à la simulation des tâches de mouvement :

- dans la phase de définition de la tâche : l'opérateur n'a pas besoin de formaliser algorithmiquement ce qu'il voit ; les différents réglages de condition initale ou de points de vue sont intuitifs.
- lors de l'exécution de celle-ci : l'opérateur peut trouver sa solution sur la base non formelle de son expérience.

Néanmoins, les simulations immersives introduisent des contraintes de temps réel très fortes. C'est particulièrement vrai pour les calculs du rendu haptique (rafraîchissement de l'ordre du kHz). Les modèles 3D

utilisés doivent donc être pré-traités afin d'obtenir les performances attendues, et notamment un bon compromis entre précision de manipulation et ergonomie de la manipulation. De plus, ces outils ne sont que faiblement intégrés aux outils PLM et, de ce fait la continuité et l'intégrité des données ne sont pas assurées. Enfin, l'opérateur fait souvent des choix rapides et pertinents dans des environnements qu'il perçoit comme familiers. Il est par contre souvent démuni devant des environnements complexes en 3 dimensions ou des phénomènes d'influence mutuelle. Dans certains cas de scènes complexes, de par la forme, le nombre et la densité d'objets mis en situation, l'utilisateur peut ne pas aisément trouver de chemin sans collisions et peut requérir de l'aide afin de mener à bien une tâche de montage ou de démontage. Enfin, l'utilisation de détecteurs de collision adaptés à l'utilisation d'interfaces haptiques permet rarement à l'utilisateur de placer les pièces manipulées en réelle configuration de contact avec les obstacles. En effet, la discrétisation réalisée par ces programmes pour répondre à la contrainte de temps réel, fixe une précision de calcul du retour haptique qui permet difficilement de déplacer des objets en état de collision (à proximité immédiate de la configuration finale par exemple).

Dans le but d'assister l'utilisateur lors de ces opérations de montage et de démontage et d'améliorer l'ergonomie de manipulation, il peut être utile d'utiliser les méthodes et les outils de planification automatique de trajectoire issus du domaine de la robotique. Ces méthodes de planification présentent l'avantage de garantir la découverte en temps fini d'une trajectoire solution si elle existe mais ne sont pas compatibles en l'état avec une utilisation en temps réel car elles sont souvent conçues pour une utilisation hors ligne sans limite de temps de calcul. De plus, lors de leur exécution dans des environnements complexes, le temps nécessaire à la découverte d'un passage solution peut être très dépendant du contexte : de la taille des objets, de leur forme ou encore de la densité d'obstacles dans l'espace de travail. En revanche, dans certains cas, leur vitesse de convergence est compatible avec la vitesse de manipulation d'un utilisateur immergé. Il est donc intéressant de coupler une planification "humaine" transcrite par la manipulation haptique à des méthodes de planification automatique de trajectoire afin, à la fois, de réduire le temps de calcul nécessaire au planificateur automatique de trajectoire, mais aussi, de guider l'utilisateur par une planification automatique lorsque celui ci rencontre des difficultés. Cette approche de "planification interactive" doit permettre de découvrir les différentes trajectoires de mouvement, en particulier de montage et de démontage, des sous ensembles manipulés en un temps réduit de manière indépendante du contexte.

Ainsi, les deux approches, planification automatique et immersion de l'opérateur, apparaissent complémentaires. Les techniques issues de la robotique fournissent des planificateurs, des schémas de commande locale en environnement encombré. Celles issues de la réalité virtuelle proposent aujourd'hui des outils permettant de capter et de produire des efforts ainsi que des déplacements ou des manifestations visuelles.

Le premier objectif de ces travaux est donc de proposer des méthodes permettant de conjuguer les avantages des techniques de planification ou de génération de consigne automatique, issues de la robotique, à ceux liés à la présence de l'opérateur "dans la boucle", favorisée par les techniques et moyens de la réalité virtuelle. Il peut se résumer à : *fournir une solution de simulation interactive :*

1. *qui permet de simuler en temps réel les différents scénarii de déplacement, assemblage, démontage de pièces en tenant compte des collisions*

2. *qui fournit à l'opérateur un protocole d'assistance et d'information dans la réalisation de sa tâche par l'utilisation de techniques de planification ;*

Il requiert notamment :

- la définition d'une structure de données permettant la planification hors ligne et en ligne et d'un protocole de partage du contrôle du scénario entre l'opérateur immergé et le planificateur.
- la définition d'une structure de données permettant la création et la destruction interactive de hiérarchies et de séquences et la définition des fonctions d'interaction visuelles et haptiques.

Pour répondre aux besoins réels de l'entreprise et envisager différents traitements à partir des données CAO de l'entreprise, notamment en termes d'intégration dans la chaîne numérique, ce projet requiert également la définition d'une architecture unifiée de données, de traitements et de modes d'interactions, à partir de travaux développés dans la communauté de la RV.

A l'heure actuelle, il n'existe pas à notre connaissance d'environnement remplissant cet objectif.

1.6 Contributions

Le travail décrit dans ce document a donné lieu aux publications suivantes :

1. N. Ladeveze, J.Y. Fourquet, M. Taix, "Interactive Motion Planning in Virtual Reality Environments", Virtual Reality International Conference 2008 (VRIC 08) - Avril 2008 - Full paper & Poster
2. Adel Sghaier, Nicolas Ladevèze, Jean Yves Fourquet, "Model Structuring for Virtual Prototype Redesign", Conférence IDMME - Virtual Concept 08 - Mai 2008 - Full paper & Talk
3. Nicolas Ladeveze, Adel Sghaier, Jean Yves Fourquet, "A Virtual Reality System Framework for Industrial Product Design Validation", 2^{nd} Mediterranean Conference on Intelligent Systems and Automation (CISA 08) - Mars 2009 - Full paper & Talk
4. Adel Sghaier, Nicolas Ladevèze, Jean Yves Fourquet, "An Integrated Virtual Reality Solution for Product Evaluation", Conférence Imagina 09 - Février 2009 - Full paper & Talk
5. N. Ladeveze, J. Y. Fourquet, B. Puel, M. Taix, "Haptic Assembly and Disassembly Task Assistance using Interactive Path Planning", Conférence IEEE Virtual Reality 2009 (IEEE-VR09) - Mars 2009 - Full paper & Talk
6. N. Ladeveze, Jean-Yves Fourquet, Bernard Puel, "Interactive Path Planning for Haptic Assembly Task Assistance", ELSEVIER - "Computers & Graphics" - Volume 34 - Special Section IEEE VR 09 - Février 2010

1.7 Plan du mémoire

Ce mémoire ne décrit pas l'intégralité des travaux réalisés dans le cadre de cette thèse pour la mise en place des techniques de RV au sein de la collaboration entre le LGP et Alstom Transport. Il est principalement

focalisé sur la contribution théorique à la collaboration entre planification automatique et recherche interactive. Les aspects liés à l'intégration des différents outils et méthodes au sein d'une plateforme unique sont décrits plus succinctement.

Dans le chapitre suivant, nous présentons les résultats utiles à notre démarche dans le domaine de la réalité virtuelle, des environnements de simulation en temps réel d'une part, en matière de planification de trajectoire d'autre part. Par la suite, le troisième chapitre décrit l'architecture de données et de traitements retenue pour la création d'un environnement répondant aux besoins industriels spécifiés. Le quatrième chapitre est consacré à la conception de techniques de mouvement interactif assisté par la planification et présente différentes situations d'expérimentation de nos travaux. Enfin, dans un dernier chapitre, la conclusion s'accompagne de la présentation succincte des perspectives ouvertes par ces travaux.

Chapitre 2

État de l'art

Sommaire

2.1 Réalité Virtuelle

2.1.1 Première approche et définitions

2.1.1.1 Définition de la réalité virtuelle

"Il est naturel pour l'homme de s'échapper de la réalité pour différentes raisons (artistiques, culturelles ou professionnelles). L'évolution des techniques aidant, l'homme a pu satisfaire ce besoin par des représentations principalement visuelles ou sonores mais figées du monde. En effet, l'utilisateur n'est que spectateur, que ce soit lorsqu'il regarde une peinture, une photographie ou un film. La réalité virtuelle lui offre une

dimension supplémentaire en lui procurant un environnement virtuel avec lequel il va pouvoir interagir. La réalité virtuelle oscille, dans l'esprit du grand public, entre fantasme et technologie, entre rêve et réalité."

Pour plus de clarté, cinq volumes précisent les fondements, les buts et les méthodes usuelles de la réalité virtuelle [4, 5, 6, 7, 8] en la définissant comme un "domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3d, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudo-naturelle par l'intermédiaire de canaux sensori-moteurs".

Une personne en immersion et en interaction dans un environnement virtuel perçoit par ses sens naturels, décide et agit sur le monde virtuel ce qui peut être schématisé par une approche "perception, décision, action" illustrée par la Figure 2.1.

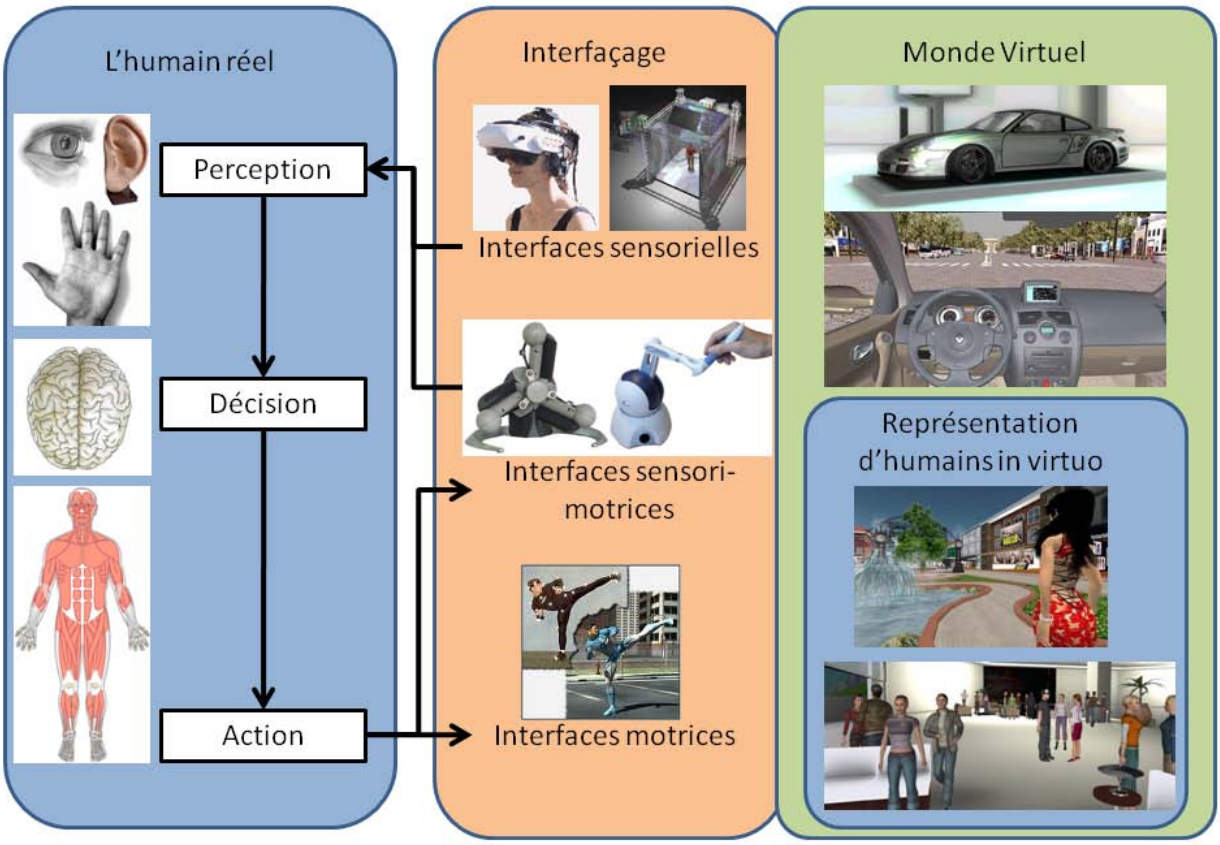


FIGURE 2.1 – Boucle de "Perception, Décision, Action" situant les trois grandes problématiques de la réalité virtuelle : "L'analyse et la modélisation de l'activité humaine en environnement réel et en environnement virtuel" (en bleu, à gauche et à droite), "l'analyse et modélisation de l'interfaçage du sujet pour l'immersion et l'interaction dans l'environnement virtuel" (en orange, au milieu) et "la modélisation et réalisation de l'environnement virtuel" (en vert, à droite)

La réflexion générale de ce domaine s'articule donc autour de trois grandes problématiques :

2.1.1.1.1 L'analyse et la modélisation de l'activité humaine en environnement réel et en environnement virtuel

Cette problématique concerne essentiellement la représentation des êtres humains dans les simulations, que ce soit sous la forme d'avatar ou d'humain virtuel. Elle concerne également l'étude du comportement de sujets humains lors de mise en situations grâce à des simulations interactives comme illustré en bleu sur le schéma en figure 2.1. On retrouve notamment dans cette problématique :

- les études de simulation du comportement humain menée par l'équipe Bunraku [9] parmis lesquelles on peut noter par exemple l'analyse du mouvement humain [10] ou la conception d'environnements [11] et de loi de commande pour l'animation réaliste d'humain virtuels [12, 13].
- les travaux réalisés au laboratoire VRLab de l'EPFL [14] portant sur la synthèse de mouvements pour les humains virtuels [15, 16, 17] ainsi que des études sur les interactions entre humains réels et humains virtuels[18, 19],
- les projets liés à l'étude du comportement humain en état d'immersion comme par exemple les interactions entre humains distants aux moyen d'environnements virtuels [20], la mesure des réactions d'un utilisateur face à un stress en immersion pour l'étude de traitements de phobies du laboratoire de Cyberpsychologie de l'UQO [21] comme l'anxiété, le vertige, ou l'arachnophobie [22, 23].

2.1.1.1.2 L'analyse et la modélisation de l'interfaçage du sujet pour l'immersion et l'interaction

Cette problématique, illustrée en orange sur la figure 2.1, débouche, premièrement, sur la conception et la réalisation d'interfaces sensorielles et/ou motrices [24] telles que les systèmes haptiques [25], les outils de capture de mouvements, ou les systèmes de visualisation immersifs. On retrouve comme exemple dans cette problématique les travaux précurseurs, dérivés de la téléopération en robotique nucléaire, réalisés par le CEA-List [26] concrétisés par le biais de la société Haption [27]. Ces périphériques sont alors utilisés pour la synthèse de retours haptique actif, haptique passif [28](basé sur l'utilisation de "props''), pseudo-haptique [29], dans le but de caractériser l'information à fournir à l'utilisateur la plus adaptée en fonction de la tâche qu'il doit réaliser.

Deuxièmement, cette problématique s'articule autour de la conception et de l'implémentation de métaphores, d'algorithmes permettant une augmentation de l'ergonomie et de l'immersion de l'opérateur dans les simulations [30, 31, 32] ainsi que l'études de nouveaux paradigmes et métaphores d'interaction pour réaliser des opérations de visualisation [33], de navigation, de saisie, de manipulation d'objets seul [34, 35] ou en coopération [36, 37], rigides ou déformables [38, 39].

2.1.1.1.3 La modélisation et la réalisation de l'environnement virtuel

Cette problématique, illustrée en vert sur la figure 2.1, concerne les choix à effectuer en terme de fonctionnalités et de rendu pour garantir une immersion répondant aux besoins spécifiés par l'opérateur dans le monde virtuel. Elle se concentre donc sur la création de métaphores et d'algorithmes permettant de garantir une interaction temps réel avec un monde virtuel adapté à une tâche dans les limites fixées par la technologie actuelle. Ces environnements peuvent être symboliques, fortement interractifs et centré sur l'apprentissage [40], ou encore réalistes car centrés par exemple sur une analyse visuelle de l'utilisateur [41, 42, 43].

2.1.1.2 Structure générale d'un système de réalité virtuelle

De manière générale [4], un système de réalité virtuelle doit répondre à cinq grandes caractéristiques :

2.1.1.2.1 Être basé sur l'informatique

Pour interfacer un monde virtuel interactif avec un utilisateur, il faut exploiter les potentialités matérielles et logicielles de l'informatique pour créer une simulation dynamique réaliste. On retrouve par conséquent dans les techniques de la réalité virtuelle certaines techniques déjà utilisées en imagerie de synthèse, en animation et en simulation.

2.1.1.2.2 Utiliser des interfaces comportementales

Elles sont composées d'interfaces sensorielles informant l'utilisateur par ses sens de l'évolution du monde virtuel et d'interfaces motrices permettant à celui-ci d'agir de la manière la plus naturelle possible sur le monde virtuel (cf. figure 2.2).

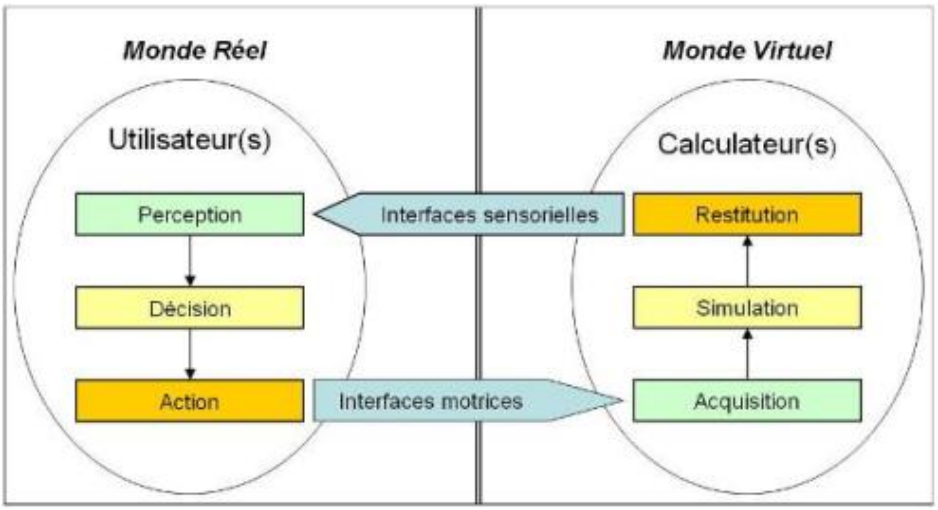


FIGURE 2.2 – Les interfaces comportementales : interfaces sensorielles et motrices (extrait de [4])

2.1.1.2.3 Comporter un monde virtuel

La création du monde virtuel est une des grande problématiques de la réalité virtuelle, il doit satisfaire au mieux les attentes de l'utilisateur en matière d'interactivité et de réalisme. Cette problématique s'articule autour de la modélisation, la numérisation, et le traitement informatique du monde virtuel. La conception d'un tel monde est souvent le résultat d'un équilibre judicieux entre le coût d'élaboration et les besoins exprimés par l'utilisateur. Il doit cependant chercher à maximiser l'efficacité de l'utilisateur dans la tâche qu'il décrit.

2.1.1.2.4 Garantir une interaction temps réel

Une interaction en temps réel permet de placer, du point de vue des sens de l'utilisateur, la simulation

au rang de réalité. Cette caractéristique est essentielle pour garantir une interaction naturelle de l'utilisateur lui permettant d'utiliser pleinement ses facultés mais c'est aussi l'une des contraintes les plus difficiles à satisfaire.

2.1.1.2.5 Placer le ou les utilisateurs en état d'immersion pseudo naturelle

La finalité de la réalité virtuelle est de permettre à une ou plusieurs personnes, au moyen de matériels et de techniques, d'avoir une activité sensori-motrice dans un monde artificiel en temps réel au point d'atteindre un état d'immersion pseudo-naturelle. Cet état est graduel et définit, de manière différente pour chacun, le niveau de réalisme de la simulation. On considère l'état d'immersion totale atteint lorsque l'utilisateur perçoit le monde simulé de la même manière que le monde réel et réagit naturellement aux stimuli virtuels qui lui sont appliqués.

2.1.2 Le matériel pour l'interaction

2.1.2.1 Système de visualisation immersive

La visualisation stéréoscopique permet de donner à l'utilisateur l'illusion que l'objet affiché est en 3 dimensions dans l'espace réel. La perception de la profondeur est due principalement à un décalage de point de vue entre l'image captée par l'oeil gauche et l'image captée par l'oeil droit de l'observateur. Grâce à ce décalage, le cerveau triangule la position de chacun des points perçus dans l'espace (cf. figure 2.3). En créant de manière artificielle ce décalage, c'est-à-dire en captant deux images d'une même scène selon deux points de vue légèrement différents et les envoyant à chaque oeil, on incite le cerveau de l'observateur à recomposer la composante de profondeur et ainsi à percevoir l'affichage comme étant naturellement tridimensionnel.

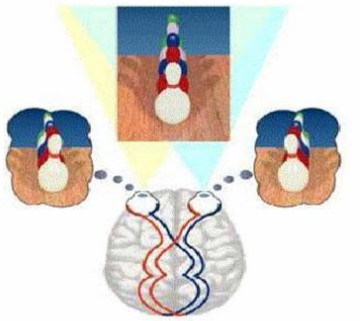
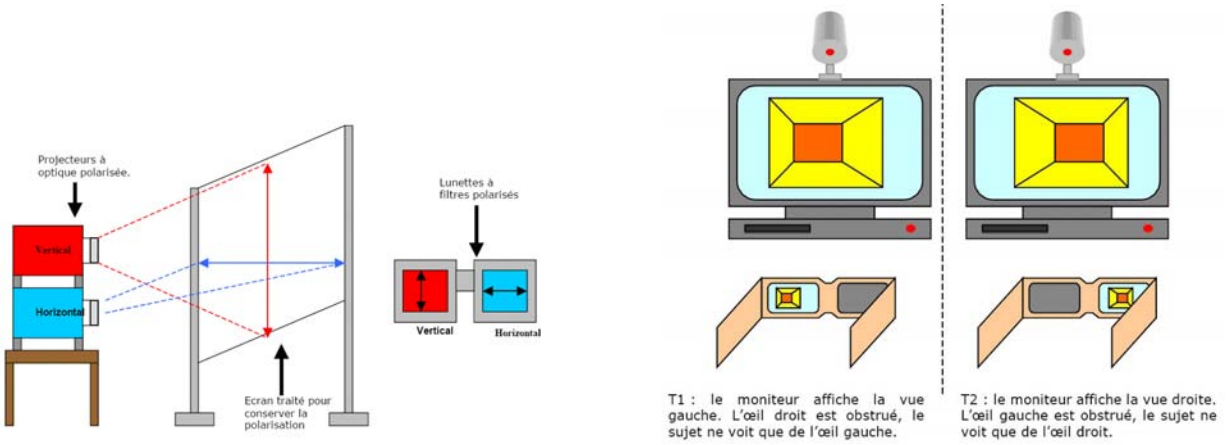


FIGURE 2.3 – Illustration du principe de stéréoscopie

Dans le cas de systèmes de projection fixes, l'utilisateur regarde avec ses deux yeux la même surface sur lequel sont affichés les deux points de vues (oeil gauche et oeil droit). Pour assigner à chaque oeil l'image qui lui est destinée, il existe plusieurs technologies :

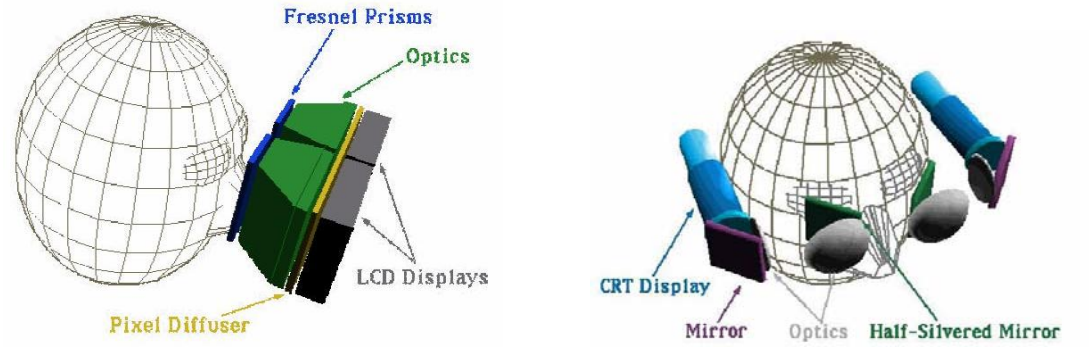
- la stéréoscopie passive, où les deux images sont rétro-projetées simultanément sur la même surface mais polarisées différemment (Figure 2.4a). Le plus souvent on utilise deux rétro-projecteurs munis de filtres polarisants affichant chacun l'une des deux images. L'utilisateur porte des lunettes munies de deux filtres polarisés qui ne laissent percevoir à chaque oeil de l'utilisateur que l'image qui lui est destinée. Cette technologie permet une bonne qualité d'affichage par le maintien de hautes fréquences d'affichage de chaque projecteur. Néanmoins, il peut apparaître, en pratique, des effets de "ghost" résultant d'une légère perception de l'image gauche sur l'oeil droit et inversement de l'image droite sur l'oeil gauche.



(a) Stéréoscopie Passive (b) Stéréoscopie Active

FIGURE 2.4 – Schémas de principe de la stéréoscopie passive et active

- la stéréoscopie active, où les images gauches et droites sont rétro-projetées successivement sur la même surface (cf. figure 2.4b). L'utilisateur porte des lunettes synchronisées avec le système de projection qui réalisent l'obturation séquentielle des lunettes gauche et droite permettant ainsi la séparation des images au niveau des yeux de l'utilisateur. Cette technologie permet d'éviter le phénomène de "ghost" parfois perceptible lors de l'utilisation d'un système de stéréoscopie passive mais peut souffrir en pratique de problèmes de défauts de synchronisation qui produisent des obturations décalées ou redondantes des lunettes (effet d'oculaire noire).
- les systèmes d'affichage portatifs (HMD : Head Mounted Display), où les images sont affichées au moyen d'écrans séparés placés devant chaque oeil de l'utilisateur. Ces systèmes utilisent les technologies d'affichage sur écrans LCD ou CRT (cf. figures 2.5 a et b). Ces systèmes présentent l'avantage de permettre une très bonne immersion de l'utilisateur mais sont dans le cas des HMD CRT encore très chers et lourds, et dans le cas des HMD LCD de faible résolution. De plus ces technologies ne sont adaptées qu'à une visualisation pour un utilisateur unique, ce qui multiplie le coût du système de visualisation lors de leur utilisation en réunion immersive.



(a) HMD LCD (b) HMD-CRT

FIGURE 2.5 – Systèmes immersifs portatifs : Head Mounted Display (HMD)

2.1.2.2 Capture de mouvements

2.1.2.2.1 Capture avec retour d'effort

Les périphériques à retour d'effort sont utilisés pour exploiter le sens du toucher de l'utilisateur dans le monde virtuel. Ces systèmes sont utilisés pour capturer la position et l'orientation de la main de l'opérateur dans l'espace et renvoyer un effort ou un couple pouvant ainsi traduire une raideur virtuelle ou une trajectoire imposée. En effet, ce type d'interface permet de situer dans l'espace l'organe terminal saisi par la main de l'utilisateur mais aussi d'exercer un torseur sur celui-ci grâce à une commande des différents moteurs. Ces périphériques constituent donc des interfaces sensori-motrices, d'entrée et de sortie, privilégiées pour la manipulation de modèles 3d dans les environnements virtuels. On peut classer ces systèmes en deux catégories : les bras haptiques et les systèmes faiblement intrusifs à câbles.

- les bras haptiques sont des interfaces à structure articulée s'apparentant à des robots 6 axes comme par exemple le Virtuose 6D 35-45 de la société Haption [27] (Figure 2.6a), ou des robots à structure parallèle comme le Virtuose 6D desktop (Figure 2.6c). Ces interfaces haptiques ont comme inconvénient un volume de travail faible par rapport aux capacités de mouvement d'un bras humain (Exemple : une sphère de 120 mm en translation et une rotation de 35° dans chaque direction au centre de la sphère pour le Virtuose 6d desktop) mais permettent d'exercer des efforts relativement élevés (35 N de force et 3.1 Nm de couple au maximum pour le virtuose 6D 35-45), suffisants pour transcrire des contacts de manière précise.
- les systèmes faiblement intrusifs à câbles comme l'Inca 6D ou le Spidar (figure 2.6d) permettent de manipuler les modèles 3d en co-localisation avec l'organe terminal. Dans ce cas, la configuration de l'objet virtuel est perçue par l'utilisateur comme confondu avec la configuration réelle de l'organe de manipulation ce qui permet une manipulation plus naturelle que la téléopération. Le principal intérêt de ce genre de système repose sur le déport des systèmes de motorisation hors du champ de vision de l'utilisateur lors de leur utilisation avec des systèmes de projection sur grand écran ce qui les rend moins intrusifs que les bras haptiques.

Plus récents, ils disposent d'une raideur comparable aux systèmes articulés classiques (Forces de 35 N et couples de 3 Nm pour l'Inca6d). De plus, il permettent une plus grande liberté de mouvements grâce à un volume de travail plus grand.

(a) Bras haptique Virtuose 6D 35-45 [27]

(b) Bras haptique de bureau Virtuose 6D Desktop [27]

(c) Bras haptique de bureau PHANTOM Desktop

(d) Système à câble Inca 6D

FIGURE 2.6 – Capture de mouvements avec retour d'effort

Les versions 3d de ces systèmes tels que, par exemple, le Phantom Desktop [44] (cf. figure 2.6c) permettent d'obtenir un retour d'effort sur 3 axes mais sont peu ergonomiques pour des manipulations d'objets en milieu encombré car elles ne permettent pas de ressentir de manière naturelle les contacts. Les versions de bras haptiques "6D" et l'Inca6D [27] permettent un retour d'effort à la fois sur les 3 translations et les 3 rotations ce qui permet d'obtenir un ressenti proche de la manipulation réelle d'objets.

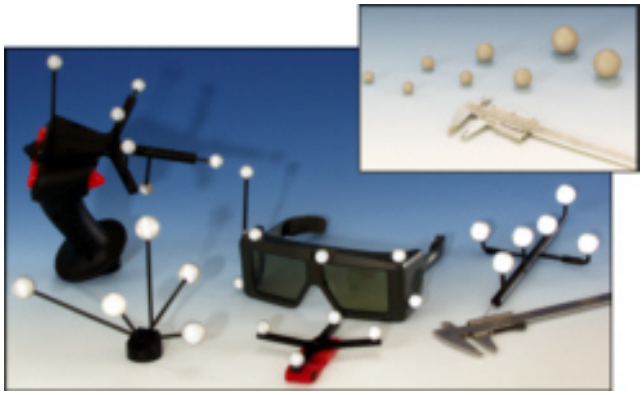
2.1.2.2.2 Capture sans retour d'effort

La position et l'orientation de différentes parties du corps de l'opérateur réel peuvent être transcrits dans

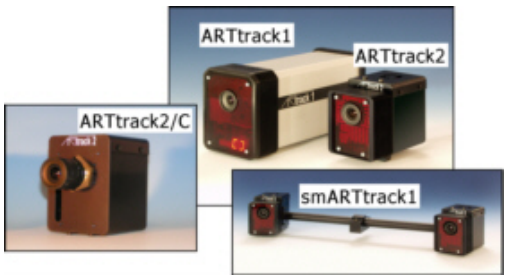
le monde virtuel grâce à des systèmes de capture de mouvement. De tels systèmes exploitent diverses technologies permettant la localisation de corps dans l'espace :

- la capture optique utilise des systèmes de plusieurs caméras ainsi qu'un logiciel de triangulation permettant la localisation de marqueurs passifs (réfléchissant une onde lumineuse) ou actifs (éméttant un signal lumineux) dans l'espace,
- la capture électromagnétique nécessite un émetteur de champ magnétique artificiel et des capteurs capables, de se repérer en temps réel dans ce champ, et de transmettre leur position dans l'espace,
- la capture par accéléromètres, permet le calcul de la position d'un corps par la double intégration de son accélération détectée par exemple par une centrale inertielle.

L'un des systèmes de capture optique largement utilisé est le système "ARTrack" illustré en figure 2.7. Ce dispositif, dit à "capture optique passive", est composé d'au minimum 2 caméras (figure 2.7b) qui émettent des flashs infrarouges renvoyés par des sphères réfléchissantes aussi appelées marqueurs (figure 2.7a). Chaque caméra du système capture alors une image de la position des différents marqueurs selon un point de vue distinct. Ces différentes images sont alors transmises à un PC qui réalise une opération de triangulation afin de positionner en 3 dimensions chaque marqueur. Grâce à la détection de distances invariantes entre des marqueurs 3d dans l'espace, reconnues lors de l'étape de triangulation, le système identifie des corps (systèmes de plusieurs marqueurs) ce qui permet de connaître la position et l'orientation de certaines parties du corps de l'utilisateur comme par exemple les lunettes et donc la tête de l'opérateur, ses bras ou encore ses mains.



(a) Marqueurs et Corps



(b) Caméras infrarouge ART

FIGURE 2.7 – Système ARTrack pour la capture de mouvement sans retour d'effort

Ce genre de système à l'avantage d'avoir un très grand volume de travail (un cube d'environ 4 m de coté par exemple pour le système ARTrack 2) par rapport aux interfaces haptiques précédemment évoquées, ce qui permet une action libre et directe des mouvements de l'utilisateur sur le monde virtuel. Il est généralement utilisé dans les simulations interactives pour la commande directe d'un avatar humanoïde par l'utilisateur. Ils sont en revanche peu adaptés dans des cadre de manipulation au contact car l'utilisateur n'a pas de ressenti haptique et doit se contenter d'un retour pseudo haptique [3], c'est à dire des informations visuelles et / ou sonores permettant de transcrire un contact. Ce type de système est tributaire de phénomènes d' *occlusions* de certains marqueurs : l'un des marqueurs est masqué par une partie du corps de l'opérateur dans une ou

plusieurs des images prises par les caméras ce qui empêche la triangulation [45]. Ce type de problème peut être résolu par une augmentation du nombre de caméras ou par leur positionnement stratégique par rapport à l'utilisateur [46, 47].

Pour capturer sans retour d'effort la configuration des doigts de la main de l'opérateur on peut également utiliser un gant de données qui vient généralement compléter un système de capture de mouvement. Ces gants sont généralement composés de fibres optique ou de jauges de contrainte pour déterminer la flexion de chaque doigt et permettent de représenter la configuration de la main de l'utilisateur dans le monde virtuel.

2.1.3 Les outils logiciels

Le développement d'une application de réalité virtuelle peut être effectuée par la programmation totale d'un environnement dédié reposant sur l'utilisation de bibliothèques graphiques telles que OpenGL ou DirectX. Cependant le temps nécessaire à la conception peut être réduit par l'utilisation d'une plateforme de développement dédiée aux applications temps réel. Dans les deux cas, la plupart de ces plateformes reposent sur l'utilisation de briques logicielles optimisées pour des applications interactives telles que des détecteurs de collision, des systèmes de gestion de la complexité d'affichage, des outils de communication... Nous détaillerons dans cette partie les différents outils logiciels utilisés ou utilisables dans le cadre de notre projet.

2.1.3.1 Plateforme de développement : Virtools Dev.

Le domaine de la réalité virtuelle ayant des applications très diverses, il existe de nombreux environnements de développement pouvant servir de plateforme de simulation tels que, par exemple, Open Mask [48, 49], ARéVI [50, 51] ou encore EVI3d [52, 53]. Néanmoins, nous ne détaillerons dans cette partie que la plateforme de développement acquise par le laboratoire : *Virtools Dev*® de Dassault Systèmes [54]. Notre choix à été motivé par la volonté d'Alstom transport d'utiliser une plateforme de développement aboutie, ayant déja fait ses preuves lors d'applications dans le milieu industriel et facilement maintenable.

2.1.3.1.1 Vue Générale

L'environnement de réalité virtuelle Virtools Dev est utilisé pour construire des compositions d'éléments 3d et définir des interactions entre ces éléments et avec l'utilisateur. Virtools est employé pour décrire le comportement des éléments qui composent l'environnement virtuel. Ce comportement est décrit par une programmation de haut niveau d'abstraction basée sur l'utilisation de modules pré-programmés analogues à ceux employés, par exemple, dans Matlab Simulink (cf. figue 2.8).

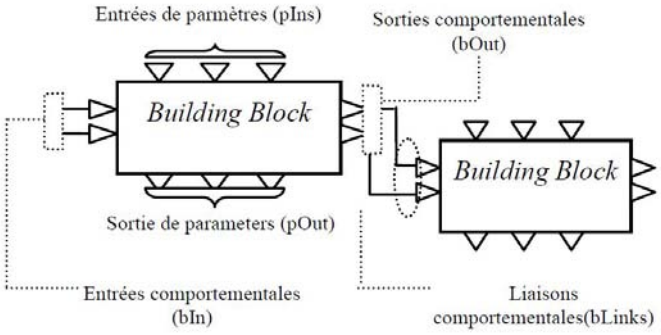


FIGURE 2.8 – Architecture pour la définition des comportement utilisée par Virtools Dev.

Virtools Dev contient une collection riche de modules comportementaux prédéfinis. Le comportement d'un élément est implémenté par le moteur comportemental en association avec des moteurs externes comme les moteurs spécialisés dans le son par exemple. En outre, le moteur comportemental Virtools est équipé d'un moteur de rendu. Pour afficher une application 3d, on peut employer le moteur de rendu en association avec Virtools Rasterizers (qui fournit un support pour DirectX et OpenGL) ou un moteur externe grâce au SDK (kit de développement de logiciel).

La création d'une composition (une application de réalité virtuelle) avec *Virtools Dev*® passe tout d'abord par l'importation de modèles 3d issus de différentes sources depuis les logiciels de CAO jusqu'aux logiciels de création de contenu digital tels que *3D Studio Max*®. Dans une deuxième étape, on doit décrire le comportement de ces modèles face aux différentes interactions possibles en utilisant les différents modules fournis par Virtools. Enfin, le moteur de rendu de Virtools est employé pour afficher l'application de réalité virtuelle grâce à l'application VR Player.

Virtools emploie l'approche orientée objet pour la construction de compositions. Chaque élément est considéré comme une instance de la classe constante (CKClass) et peut être importé dans la simulation, soit de manière statique, lors de la programmation de l'environnement dans l'interface dev, soit de façon dynamique, c'est à dire pendant l'exécution de la simulation. Dès lors, le comportement de ces objets est programmé au moyen de *scripts de comportement*, ce qui permet de spécifier leurs réactions par rapport à des stimuli provenant d'une action de l'utilisateur ou d'une réaction au comportement d'un objet. Ces scripts de comportement sont des graphes d'actions élémentaires, représentées par des *building blocks*, permettant un codage graphique intuitif pour le concepteur de l'environnement virtuel.

2.1.3.1.2 Boucle d'exécution de Virtools

Virtools Dev est un moteur temps réel de simulation. En effet, chaque composition s'exécute dans un seul processus sur la machine cible. La composition prend donc en charge l'ordonnancement des tâches qui la composent. L'exécution d'une composition se traduit par le parcours d'une boucle infinie qui met à jour les processus les uns après les autres suivant leur ordre de priorité. Comme dans la plupart des systèmes temps

réel, le temps de parcours d'une boucle est considéré comme étant le plus petit retard qu'il peut y avoir entre l'action d'un utilisateur et la réaction du système. Les phases qui composent un cycle de rafraîchissement typique sont données dans la figure 2.9.

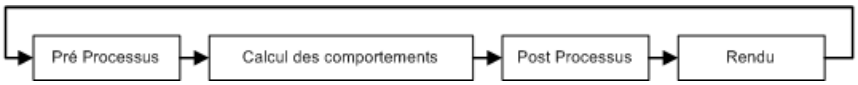


FIGURE 2.9 – Boucle d'exécution de Virtools

2.1.3.1.3 Programmation des comportements

Les comportements des objets virtuels dans l'environnement Virtools Dev sont définis à l'aide de graphes de comportements constitués de séquences de blocs comportementaux ou BB représentés sur la figure 2.10. Ces graphes de comportements sont encapsulés dans des "scripts" servant non seulement à définir les actions et réactions du système mais aussi à les exécuter lors des requêtes du moteur d'exécution. Chaque script est attaché à un Objet Virtools qui est spécifié dans son en tête (Annotation "1" sur la figure 2.10) et définit son comportement face au différents stimuli qu'il reçoit. Lors de l'activation d'un script, un stimulus d'initialisation est émis de manière à activer le ou les graphes depuis le "Start" (Annotation "3" sur la figure 2.10).

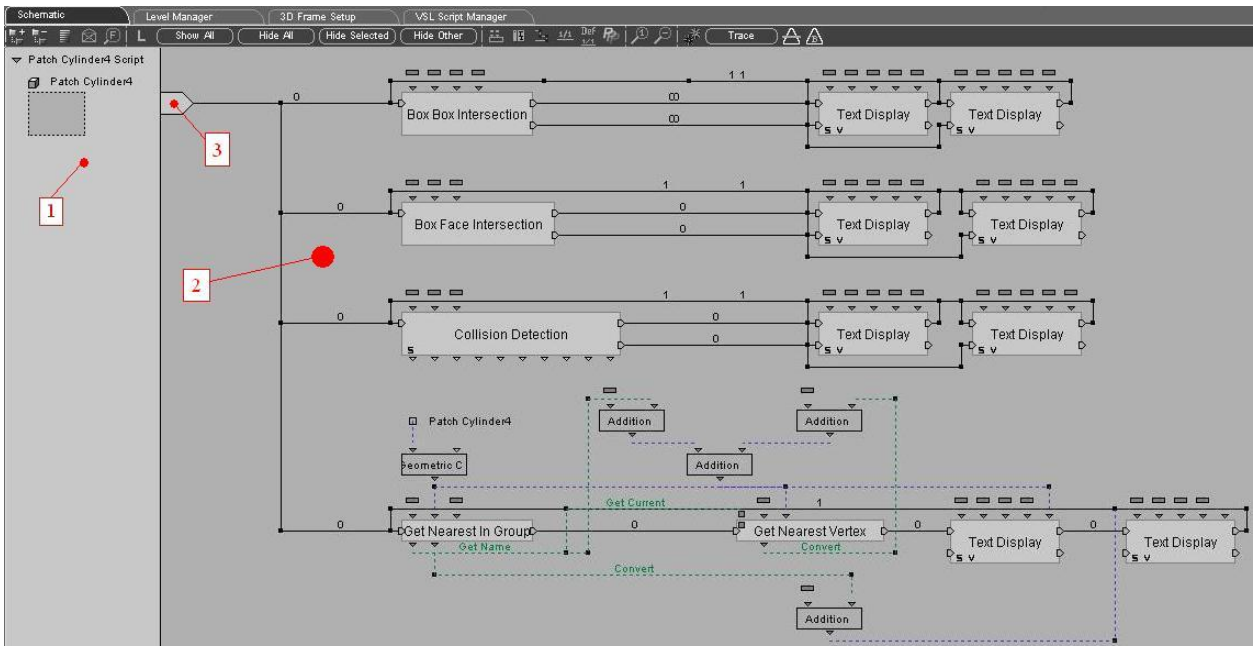


FIGURE 2.10 – Illustration d'un Script constitué de Building Blocks : 1. En tête spécifiant l'objet ciblé, 2. Graphes de comportement, 3. Start

Chaque bloc comportemental est représenté par une boîte noire qui comporte des entrées, des sorties et des paramètres. Les entrées et les sorties servent à transmettre des stimuli d'activation entre les différents blocs. Les paramètres servent, quant à eux, à transférer des valeurs (variables ou constantes) entre les différents BB.

2.1.3.1.4 Programmation de nouveaux BB Virtools dev propose au concepteur une large gamme de BB pré-programmés permettant la réalisation d'actions simples sur les objets chargés dans l'environnement virtuel comme par exemple des translations, des rotations, des changements hiérarchiques ... Néanmoins, il est parfois nécessaire d'en créer de nouveaux afin d'une part, de réduire la complexité et d'optimiser le temps d'exécution des graphes de comportements, et d'autre part, de représenter de nouveaux comportements impossibles à synthétiser à partir des BB existants. Dans cette optique, le concepteur peut utiliser le language de programmation VSL proposé dans l'interface dev ou encore utiliser le SDK fourni par Virtools pour programmer de nouveaux BB en C++.

Le Virtools Scripting Language (VSL)

est un moyen simple mais limité de créer un comportement à partir d'un bloc de base appelé "Run VSL" (Figure 2.11). Le nombre, le type et les dénominations des variables et des paramètres d'entrée et de sortie sont personnalisables et la plupart des classes d'objets du SDK de Virtools dev sont accessibles. Une fois ce BB placé dans la fenêtre de script, il est possible de l'éditer pour programmer un comportement grâce à une interface de programmation directement accessible dans Virtools dev (Figure 2.12). Le langage utilisé est très proche du C++, le débogage et la compilation de ce bloc sont réalisés par Virtools dev. Cependant, les facilités de programmation apportées par le VSL ne vont pas sans quelques restrictions : on ne peut ni utiliser l'allocation dynamique, ni créer de classes, ni utiliser des pointeurs, ni lire ou écrire sur le disque et la taille du programme est limitée en nombre de lignes.

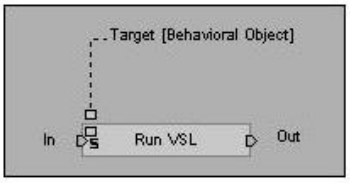


FIGURE 2.11 – Bloc Run VSL

FIGURE 2.12 – Fenêtre de programmation VSL

Le C++ SDK

permet d'accéder à toutes les fonctionnalités du C++ : l'intégration de librairie externes, la création de nouvelles classes, et l'utilisation de tous attributs et méthodes des objets de Virtools. Cependant, pour utiliser

cette fonctionnalité, il faut programmer sous Visual C++ .NET 2005 sur la base d'un module de paramétrage fourni par Virtools. Ce module permet de faciliter le travail du développeur en structurant le programme de telle sorte qu'une fois compilé sous forme de bibliothèque dynamique (dll), il soit disponible dans l'interface dev et intégrable sous forme de BB dans des scripts de comportement des objets.

Chaque BB peut être créé de manière indépendante ou être lié à un "manager". Dans ce dernier cas, une catégorie de BB peut partager des données et constituer un ensemble de fonctionnalités cohérentes (cf. figure 2.13).

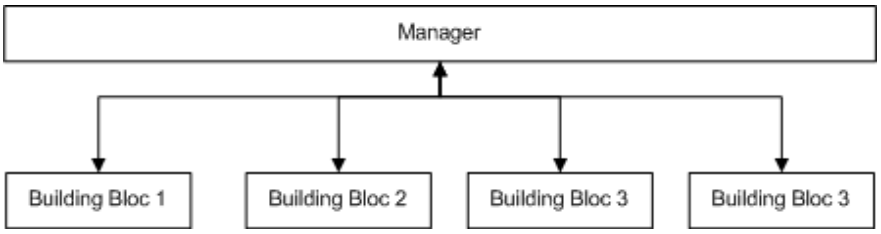


FIGURE 2.13 – Illustration de dépendances et des flux de données entre une catégorie de BB et un manager

Le principal inconvénient de ce type de programmation est que lorsque l'on compile avec succès un bloc avec Visual C++, il n'est pas garanti qu'il fonctionne dans Virtools. Il faut donc surveiller l'évolution du programme grâce à la console de Virtools ou en fixant des point d'arrêt dans l'interface de Visual C++ de façon à savoir exactement ou se situent les éventuels problèmes au cours du développement.

2.1.3.1.5 Les différents modules et bibliothèques additionnelles

Packs Virtools dev

Virtools dev propose en option certaines bibliothèques additionnelles qui se décomposent en une série de managers et de building blocks spécialisés. On y trouve notamment :

- la *communication avec des interfaces de réalité virtuelle* (VR Pack)
  Ce pack comporte une série d'outils logiciels permettant notamment de recevoir et d'émettre des données au format VRPN ou encore de paramétrer simplement un système de visualisation stéréoscopique. Il constitue un outil indispensable pour l'utilisation de Virtools en environnement immersif.
- l'*intelligence artificielle* (IA Pack)
  Cet ensemble de fonctionnalités s'adresse principalement aux concepteurs de jeux vidéos. Il permet de munir une entité 3d ou un personnage d'une "intelligence" limitée qui lui est propre au travers d'un script de comportement. On retrouve dans cette catégorie des fonctionnalités d'évitement de zone ainsi que des outils de cartographie 2d associé à des outils de calcul de chemin (cf. figure 2.14).

FIGURE 2.14 – Illustration de fonctions de calcul de chemin disponibles dans le pack d'intelligence artificielle

- la *simulation physique* (Physics Pack)
  Le Physics pack regroupe un ensemble d'outils permettant essentiellement de rendre les scènes virtuelles plus réalistes par l'introduction de comportement de physique newtonienne au niveau des objets. Il permet de définir des articulations entre différents objets afin de les rendre réactifs à des stimuli en provenance d'autre objets dotés de propriétés physiques ou de l'utilisateur lui même.

Interactive Physic Pack (IPP)

IPP est un outil logiciel développé par la société Haption pour permettre l'utilisation d'une interface haptique de type Haption avec la plateforme Virtools dev.

Il est composé principalement d'une série de BB (Figure 2.15) permettant la liaison entrée / sortie entre la simulation et des outils de manipulation (Virtuose 6D, space mouse ou ARTrack). Il contient également un module de gestion de collision avancé nommé IPSI basé sur le détecteur de collision VPS, décrit en partie 2.1.3.2.1, suffisamment rapide pour calculer le retour haptique résultant d'un contact dans l'environnement. Il comprend également une bibliothèque de BB permettant de limiter les degrés de liberté lors de la manipulation.

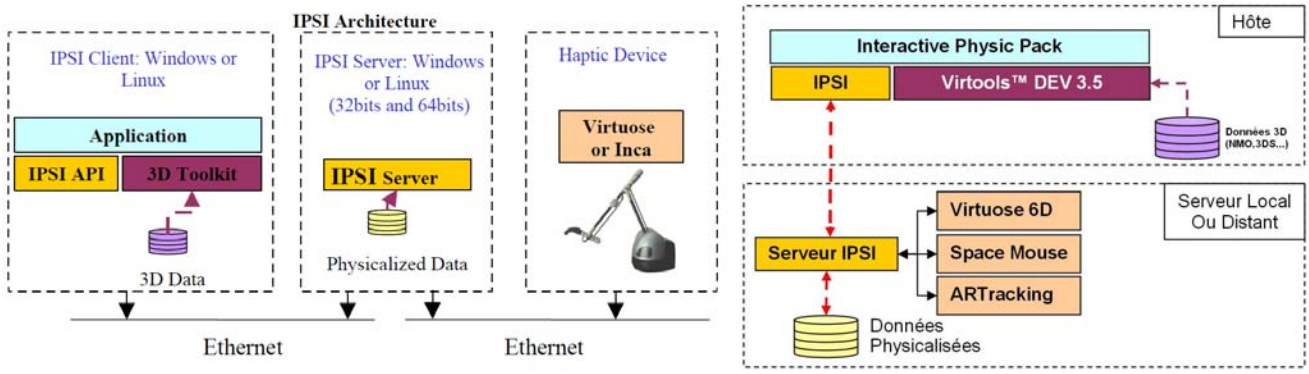


FIGURE 2.15 – Architecture d'IPSI et du plug in IPP pour Virtools dev

2.1.3.1.6 Pourquoi utiliser Virtools ?

Virtools dev intègre une série d'outils logiciels permettant une mise en oeuvre rapide de simulations

interactives. De plus, il permet grâce au VR Pack et à sa gestion du protocole VRPN une utilisation aisée des interfaces de capture de mouvement de type ARTrack. Il permet une importation de données 3d sous format VRML (largement utilisé) mais également 3dxml ce qui renforce sa compatibilité avec les outils Dassault Systèmes tels que Catia V5, largement utilisé dans le milieu industriel. Il comporte une bibliothèque de BB permettant de créer rapidement des comportements simples. Enfin, Il permet une programmation avancée de comportements, permettant l'intégration de bibliothèques externes, adaptée à l'implémentation de nouvelles métaphores d'interaction.

Virtools comporte, dans sa version de base (version 4.1), certaines lacunes, notamment au niveau de son détecteur de collisions, qui peuvent être corrigées par l'utilisation de bibliothèques externes grâce à une programmation de BB personnalisés. Néanmoins, n'offrant pas une ouverture totale de son code, les modifications effectuées sont restreintes à la définition de comportement ou à la modification de la technique de calcul de rendu mais ne peuvent influer sur la structure générale de la boucle d'exécution.

2.1.3.2 Bibliothèques logicielles et Algorithmes utilisés en réalité virtuelle

2.1.3.2.1 Les détecteurs de collisions

Il existe un grand nombre de bibliothèques et algorithmes permettant de détecter les collisions dans un environnement virtuel [55, 56] et ainsi de restituer à l'utilisateur des sensations qui augmentent son degré d'immersion en utilisant un retour haptique ou des effets visuels dits pseudo-haptiques. D'après [57], les méthodes de détection de collision peuvent être classées en trois grandes catégories :

- Les *méthodes discrètes* qui échantillonnent la trajectoire estimée d'un objet et ne détectent que les interpénétrations de l'objet en déplacement avec les obstacles de la scène à chaque échantillon. Le pas choisi pour la discrétisation de la trajectoire conditionne directement la précision de ce genre de méthode qui dans certains cas peuvent "oublier" des collisions (« tunneling effect »). Elles réalisent généralement une estimation de l'instant de collision en procédant par dichotomie et en utilisant un pas de discrétisation de plus en plus petit. Elles demeurent les plus répandues car elles sont simples à programmer mais sont très coûteuses en temps de calcul.
- Les *méthodes pseudo-continues* se basent sur le calcul du volume généré par le déplacement de l'objet le long de sa trajectoire ("Swept Volume"). Elles permettent de détecter facilement la non collision d'un objet avec les obstacles de la scène mais faisant abstraction de la composante temporelle du problème lors de la génération du volume, elles rendent difficile la définition de l'instant de collision.
- Les *méthodes continues* calculent l'instant du premier contact entre objets pendant la détection de collision. Ce calcul est partie intégrante de l'algorithme [58].

Ces méthodes sont implémentées et disponibles sous la forme de bibliothèques logicielles parmi lesquelles on peut citer VPS, V-Collide, IMMPACT, ou encore CONTACT Toolkit qui sont présentées ci-après.

VPS (Voxmap Point Shell) [59]

Ce détecteur de collisions, développé par Boeing, est spécialement adapté à une utilisation avec des interfaces haptiques en réalité virtuelle. Il repose sur une décomposition spatiale hors ligne qui permet de

détecter les contacts et de gérer les collisions à des fréquences adaptées aux interfaces haptiques (environ 1 kHz). Il est très utilisé dans les systèmes de réalité virtuelle utilisant des données C.A.O. car il comporte des qualités essentielles :

- il détecte tous les contacts de surfaces de manière anticipée et minimise l'interpénétration.
- il calcule la force et le couple résultant du contact.
- il peut maintenir un rafraîchissement de 1 kHz avec des géométries complexes (>100000 triangles).
- il possède un système de contrôle évitant de transmettre les instabilités du calcul du retour d'effort au niveau de la main de l'utilisateur.

Cependant VPS comporte des limitations en terme de nombre de pièces *physicalisables,* c'est à dire prises en compte dans le calcul du retour haptique que se soient des objets manipulables ou des obstacles fixes. De plus, toutes les pièces physicalisées doivent être précisées à l'initialisation ce qui bride les possibilités d'interaction temps réel de l'utilisateur dans le monde virtuel.

Le système VPS repose sur 3 éléments de base :

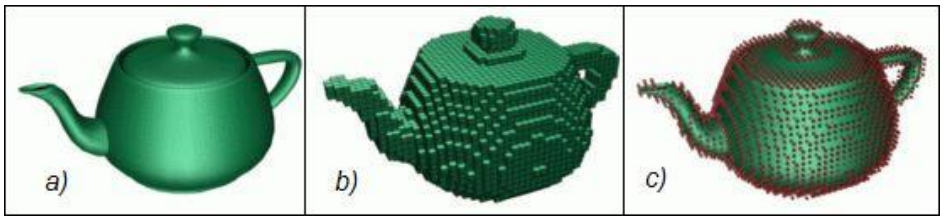


FIGURE 2.16 – Les structures de données utilisées dans VPS : a) Modèle polygonal , b) Modèle décomposé en voxels , c) Modèle "Point-Shell"

1. Un arbre de profondeur fixe appelé "Point shell" qui représente l'objet dynamique (objet manipulé). Cet arbre est constitué de points appartenant à la surface de l'objet ainsi que des normales à la surface associée (cf. figure 2.16, c) .
2. Une carte de voxels (cubes) qui représente la scène avec tous les objets statiques (cf. figure 2.16, b). Cette représentation permet de simplifier l'environnement dans lequel l'objet dynamique va évoluer de façon à effectuer des calculs rapides. La carte de voxels ("VoxMap") est réalisée grâce à une décomposition spatiale proche d'un octree appelée 3N-tree. Les différents voxels qui la composent sont indexés d'un indicateur de distance : 1 pour l'intérieur, 2 pour la surface, 3 à N pour l'extérieur proche de la surface, 0 pour l'espace totalement libre, comme illustré par la figure 2.17.
3. Un système de forces de pénalité : "Tangent Plane Force Model". Ce système permet de calculer les interactions entre l'objet dynamique représenté par son modèle "Point-Shell" et la partie statique du monde virtuel représenté par la "VoxMap". Lorsque l'un des points du modèle "point shell" de l'objet dynamique pénètre dans un voxel de la "VoxMap", la distance "d" entre le plan tangent et ce point est calculée (cf. figure 2.18a). Le plan tangent est un plan passant par le centre de gravité du voxel et orienté par la normale attachée au point qui a pénétré le voxel. La force exercée sur l'objet dynamique en chaque point de sa "point shell" est calculée proportionnellement à la distance de pénétration. En sommant et en rapportant toutes ces forces au repère local de manipulation, on obtient le torseur

appliqué au niveau du bras haptique. Ce mode de calcul est très simple et provoque de nombreuses instabilités. Cependant un modèle dynamique basé sur l'ajout d'une viscoélasticité $[k_r, k_t, b_t]$ entre le retour d'effort calculé par le système de forces de pénalité et celui appliqué au niveau de l'interface haptique permet de les éliminer (cf. figure 2.18b).

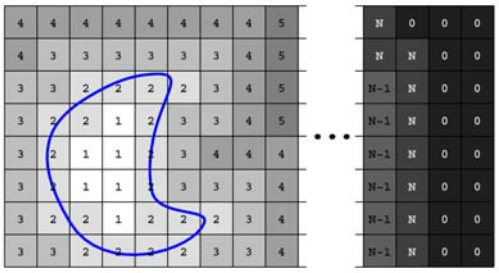
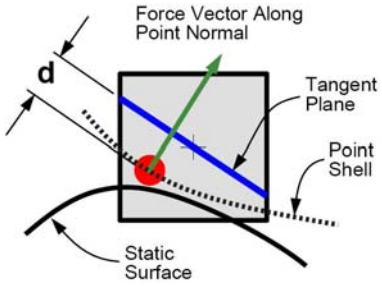
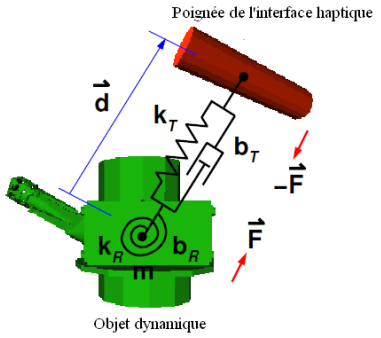


FIGURE 2.17 – Carte de voxels (VoxMap) synthétisée par VPS



(a) Calcul de la distance "d" d'interpénétration de l'objet dynamique dans les obstacles statiques : "Tangent Plane Force Model"



(b) Modèle dynamique de calcul du retour d'effort

FIGURE 2.18 – Calcul du retour d'effort dans l'algorithme VPS

V-Collide [55]

V-Collide est un algorithme de détection de collision reposant sur une unification des algorithmes I-Collide et RAPID [60].

Au premier niveau, cet algorithme réalise le calcul de boîtes de type "Axis-Aligned Bounding Boxes" (AABBs) englobant chaque objet de la scène. A chaque rafraîchissement, les bornes maximales et minimales de ces boîtes englobantes sont organisées dans des listes ce qui permet de sélectionner rapidement

des paires d'objets potentiellement en collision (cf. figure 2.19). Dans un deuxième niveau de détection, l'algorithme construit un graphe de boîtes englobantes orientées ("Oriented Bounding Boxes" ou OBBs) pour décomposer chaque entité 3d de la scène. Lorsque une paire d'objets est détectée comme étant potentiellement en collision au premier niveau de l'algorithme, le deuxième niveau, permet par tests successifs réalisés sur les graphes d'OBBs de préciser les zones de ces objets potentiellement en contact. Enfin, le contact est confirmé par un test de collision exact "triangles vs triangles".

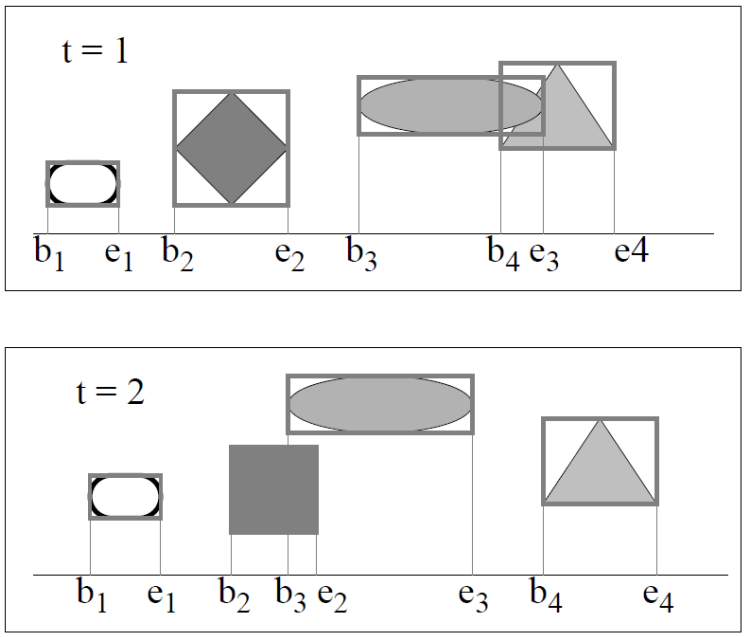


FIGURE 2.19 – Selection des objets potentiellement en collision par calcul d'intersection d'AABB extrait de [61]

Interactive Massive Model Proximity and Collision Tester IMMPACT

Ce détecteur de collision, développé au "Department of Computer Science" de l'université de Caroline du Nord (USA) [62, 63] est dédié au calcul de distances et à la détection d'intersections entre modèles lourds, tels que les modèles 3d issus de la CAO, dans les simulations interactives. L'algorithme de détection de collision [64] repose sur la création d'un graphe de chevauchement de boîtes englobantes (de type AABB). Les chevauchements détectés entre boîtes englobantes constituent alors les arcs d'un graphe symbolisant les contacts potentiels. Les noeuds de ce graphe comportant le plus grand nombre d'arcs et désignés comme "high-valence nodes" sont traités en priorité afin de réduire le plus rapidement possible la connectivité du graphe de contacts potentiels. Enfin, ce graphe est décomposé en sous graphes dimensionnés en fonction de la taille du cache alloué à la détection de collision. De cette manière, les tests de détection de collision peuvent être effectués sur des modèles lourds en minimisant le nombre d'accès au disque et en bornant la taille de la mémoire utilisée.

CONTACT toolkit

« Contact toolkit » est une bibliothèque en C++ réalisée par l'équipe i3d de l'INRIA [65]. Elle synthétise le travail réalisé à l'INRIA sur la mise au point d'une méthode de détection de collision en continu [66]. La plupart des méthodes de détection de collision sont dites « discrètes » : elle discrétisent le mouvement de l'objet testé et détectent l'interpénétration de celui-ci avec le ou les obstacles. Mais ces méthodes peuvent oublier des collisions (« tunneling effect »). Pour corriger ce problème, on utilise des pas variables ou des méthodes prédictives dans les applications hors ligne mais ce n'est pas souhaitable, car très coûteux en temps de calcul, pour des applications temps réel. De plus les méthodes discrètes sont obligées d'utiliser des méthodes de type « Backtracking » pour calculer le premier point d'impact dont les temps de calcul dépendent fortement de la complexité des objets. On parle ici de détection de collision continue, c'est-à-dire une méthode permettant de calculer exactement le premier point de contact durant la phase de détection de collision. Ce calcul de point de contact directement intégré au programme de détection de collision le rend plus robuste pour être utilisé avec des interfaces haptiques car il interdit les interpénétrations d'objets.

2.1.3.2.2 Technique de simplification de l'affichage

Afin de respecter la contrainte de temps réel dans les simulations de réalité virtuelle, il est parfois nécessaire de recourir à des techniques de simplification des modèles. Ces techniques permettent de réduire la complexité (en terme de nombre de triangles) de la scène et donc permettent de garantir un calcul de rendu en temps réel. Elles sont particulièrement utiles dans les cas d'affichage de données lourdes telles que les scènes de CAO.

Level Of Detail (LOD)[67, 68, 69]

Le concept de LOD repose sur une simplification des modèles perçus par l'utilisateur comme étant d'un intérêt secondaire dans le contexte de la scène affichée. On trouve dans ce cas les modèles proches de faible dimension mais également les géométries distantes qui ne sont pas des points d'intérêt de la scène. Ce principe permet essentiellement de trouver un équilibre entre la fidélité du rendu et le maintien de la contrainte d'affichage temps réel. Différentes versions de ce principe sont utilisées :

- Le LOD Discret [67]

  Dans ce cas, différentes versions, de plus en plus dégradées, d'un même maillage sont préparées "hors ligne" puis sont utilisées suivant un critère de distance entre l'observateur et l'objet, ou encore de position de l'objet affiché dans le champ de vision de l'utilisateur (cf. figure 2.20). Cette technique est relativement simple et permet un calcul de rendu accéléré au niveau des cartes graphique modernes. Cependant cette technique peut produire des effets indésirables comme par exemple des effets dits de "popping" résultant de la perception par l'utilisateur d'un changement brutal de résolution d'un modèle lors d'un changement de position dans le monde virtuel. On constate également des problèmes de rendu avec cette méthode lors de l'affichage de modèles composés d'un grand nombre d'objets : c'est notamment le cas des scènes de CAO. Dans ce cas de figure, la réduction de la complexité des différents maillages unitaires composant un assemblage entraîne une perte d'intégrité fortement perceptible.
- Le LOD continu [67]

Afin de pallier les lacunes du LOD discret, le LOD continu se réalise par la création d'une structure de donnée des objets constituant la scène et une méthode de calcul en ligne d'un niveau de détail désiré. Par exemple le "View dependent LOD" permet une simplification "en ligne" adaptative sur un même maillage qui dépend du point de vue de l'utilisateur (cf. figure 2.21). On notera que cette méthode nécessite une définition particulière des maillages utilisés dans la scène, comme par exemple les maillages progressifs, permettant une extraction d'un LOD en temps réel.

- le LOD Hiérarchique [70, 71]

  Cette dernière méthode tend à résoudre les problèmes de visualisation de modèles composés d'un grand nombre d'objets hiérarchisés. Elle est notamment d'un grand intéret dans les cas d'utilisation de modèles CAO qui sont dès leur création très hiérarchisés. Cette méthode, généralement utilisée de manière complémentaire avec une méthode de LOD classique, permet de définir plusieurs entités initialement indépendantes en un seul maillage (cf. figure 2.22).

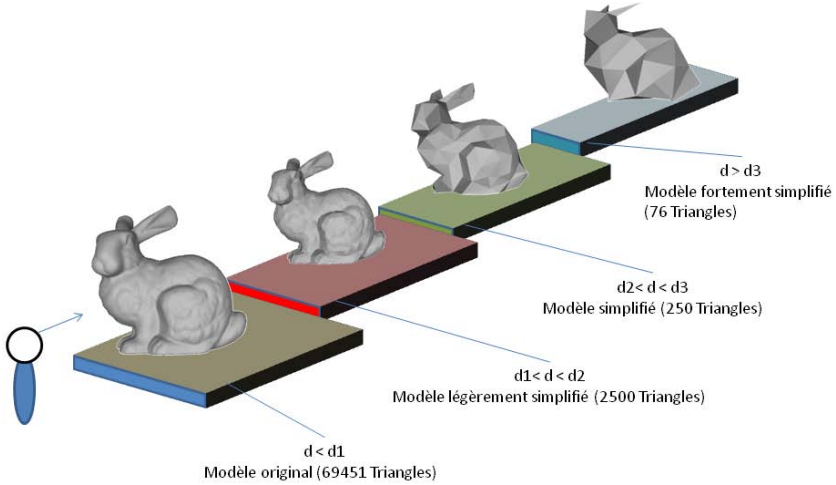
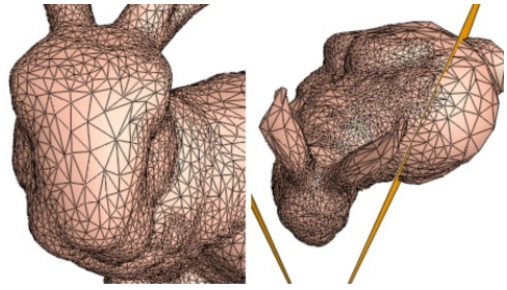
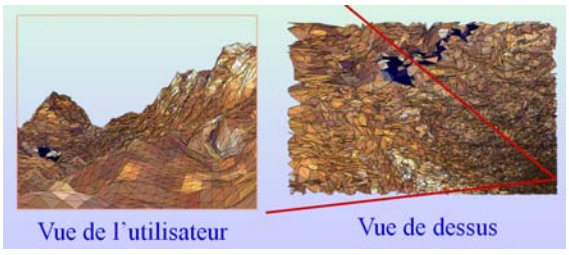


FIGURE 2.20 – Illustration du principe d'un LOD discret

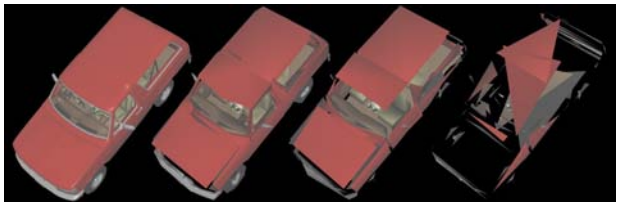


(a) Simplification adaptative en fonction de la position de l'objet dans le champ de vision de l'utilisateur

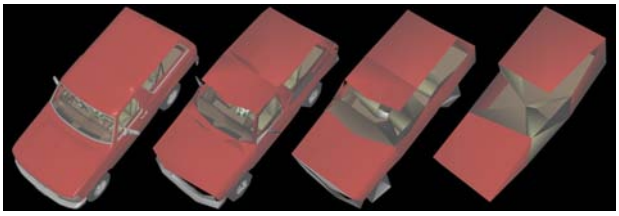


(b) Simplification adaptative en fonction de la distance

FIGURE 2.21 – "View Dependent LOD" [71]



(a) Utilisation d'un LOD Classique provoque une perte de l'intégrité du modèle, de gauche à droite 74308 faces (modèle original), 1,357 faces, 341 faces et 108 faces.



(b) Utilisation d'un LOD Hiérarchique (HLOD) permet de conserver l'intégrité d'un modèle initialement hiérarchique, de gauche à droite 74,308 faces, 1,357 faces, 338 faces et 80 faces

FIGURE 2.22 – Comparatif entre LOD classique et LOD Hiérarchique sur une scène constitué d'objets hiérarchisés [70, 71]

Maillages progressifs et "patch meshes"

Ces méthodes de codage des maillages permettent un réglage "en ligne" du niveau de fidélité du maillage :

- Les maillages progressifs utilisables dans Virtools et introduits par Hugues Hoppe en 1996 [72] reposent sur une reconstruction itérative d'un maillage préalablement réduit. Dans une étape de pré-calcul, le maillage original M_n est simplifié itérativement : à chaque étape, un maillage M_i est défini à l'aide d'une liste d'opérations de simplifications effectuées par une suppression de vertex ou de segments du maillage M_{i-1}. Les différentes étapes de simplification sont poursuivies jusqu'à arriver au niveau de simplification maximum M_0. Ce format de maillage permet à partir d'un modèle très simplifié M_0 de restituer une à une les simplifications effectuées jusqu'à la reconstruction totale du maillage original M_n comme illustré en figure 2.23. Cette définition particulière de maillage est très adaptée lors de l'utilisation de LODs ou la transmission de modèles en streaming [73]. Le réglage de ce mode de représentation s'effectue principalement par le choix de la méthode d'optimisation utilisée lors des étapes de simplification réalisées entre deux itérations d'un même maillage M_{i-1}et M_i.
- Les "patch meshes", utilisés dans Virtools dev [54], sont codés selon le principe de surfaces de Bezier : par un tableau de points-clefs du modèle où sont définies des tangentes au maillage. Ce format de stockage présente des avantages évidents pour une génération en ligne de maillage à base de triangles suivant

un paramètre précision spécifié mais peut devenir très lourd et difficilement exprimable dans le cas de modèles complexes.

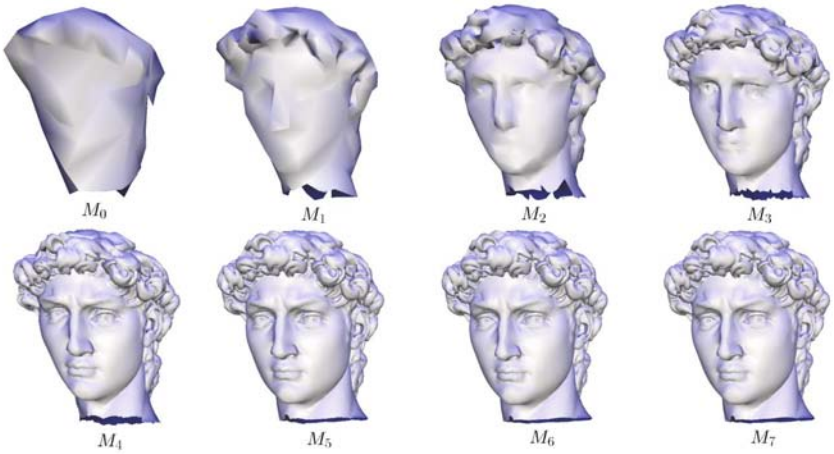


FIGURE 2.23 – Restitution d'un maillage progressifs [74] depuis le maillage le plus simplifié M_0 :154 vertices jusqu'au maillage original M_7 : 24085 vertices

2.1.3.3 Les formats de données 3d

Il existe une très grande quantité de formats de données 3d souvent liés à un logiciel propriétaire et adaptés à des utilisations spécifiques. Dans cette partie, nous détaillerons simplement dans cette partie les principaux formats de données 3d conçus spécifiquement pour une utilisation dans les environnements interactifs.

2.1.3.3.1 Le format NMO

Le format NMO est le format natif utilisé pour les objets 3d dans Virtools [54] ; il constitue donc un format d'échange privilégié avec les simulations interactives programmées sous Virtools dev. Il contient la définition d'un ensemble d'objets 3d munis d'un ou plusieurs maillages, matériaux et textures ainsi que des scripts permettant de coder leurs comportements. Le principal défaut de ce format est le manque de structuration des maillages : chaque maillage est défini à l'aide d'une liste chaînée non ordonnée de triangles. Ce type d'organisation rend très difficile l'exécution d'algorithmes de diffusion sur le maillage Virtools.

2.1.3.3.2 Le format VRML (Virtual Reality Modeling Language)

Le VRML (Virtual Reality Modeling Langage) est un langage de modélisation de scènes 3d principalement destiné à être exploité en environnement interactif et sur internet [75, 76]. Il donne la possibilité de décrire des environnements virtuels dans un fichier texte sans avoir à programmer la manière dont cette description est ensuite utilisée pour générer les images sur l'écran. Tout le niveau de mise en oeuvre est à

la charge du logiciel de visualisation 3d. Tandis que la version 1.0 du langage ne permettait que la description d'environnements statiques, le format VRML 97 (petite mise à jour du format VRML 2.0) permet en plus de décrire dans une certaine mesure des comportements dynamiques (animations) et plus d'interaction entre l'utilisateur et la scène. Les programmes VRML peuvent décrire des formes simples (points, lignes, polygones) ou complexes (sphères, cubes, cônes, cylindres...), du texte, des images, des animations, des éclairages, des sons, des hyperliens, ainsi que leur agencement dans l'espace, leurs textures, leurs couleurs et leurs matériaux. Ce format constitue par sa nature libre et gratuite un format d'échange de modèles 3d et de simulations interactives privilégié pour de larges diffusions sur internet.

2.1.3.3.3 Le format 3dXML

Depuis 2005, est apparu un format développé par Dassault Systèmes permettant de stocker la géométrie et les données PLM d'un produit. Ce format léger (compressé) est commun à toute la gamme de logiciels Dassault Systèmes (Catia V5R14 et les versions suivantes, Delmia, Solid Works, Virtools ...). S'appuyant sur du XML standard, ce format autorise tout logiciel à lire, écrire et enrichir les contenus des fichiers 3dXML à l'aide d'outils standards, ce qui facilite la conversion depuis les formats 3d existants. Ce format permet non seulement de définir la géométrie d'un objet mais aussi une liste de comportements associés.

Depuis la sortie de Virtools 4.0, le format 3dXML est accepté pour l'importation et l'exportation de données. Il constitue donc une passerelle privilégiée pour exploiter des données CAO issues de CATIA dans des environnements programmés sous 3DVIA VIRTOOLS. Ce format est prévu pour permettre une mise en situation du produit à partir de sa définition numérique 3d. En effet, grâce à la géométrie et aux comportements, les objets 3dXML peuvent être importés dans Virtools sous forme d'objets 3d munis de scripts déjà programmés. Cette passerelle permet ainsi de créer rapidement des simulations interactives permettant aux futurs clients de manipuler de manière réaliste le modèle numérique du produit fini en utilisant un logiciel gratuit de lecture de composition Virtools (3dVIAPlayer ou 3dLifePlayer) ou le visualisateur de fichier 3dXML (3dXML Player) tout deux fournis par *Dassault Sytèmes*®.

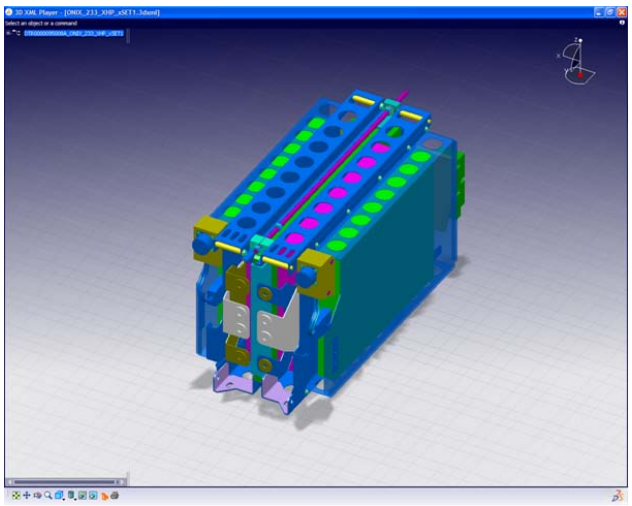


FIGURE 2.24 – Modèle 3d encodé en 3DXML visualisé en utilisant le 3DXML Player

2.1.3.3.4 Le format JT

Depuis le lancement du “JT Open Program” en 2003, le format “JT” s’est étendu et en 2005 a été publié et annoncé comme un nouveau format de référence pour la diffusion de données 3d. Le programme “JT Open” lancé par la société Siemens [2] permet un accès en lecture et en écriture aux fichier sous format JT. Ce format constitue donc une passerelle d’échange privilégiée avec le logiciel de management de la vie du produit “UGS Siemens PLM” utilisé par Alstom transport.

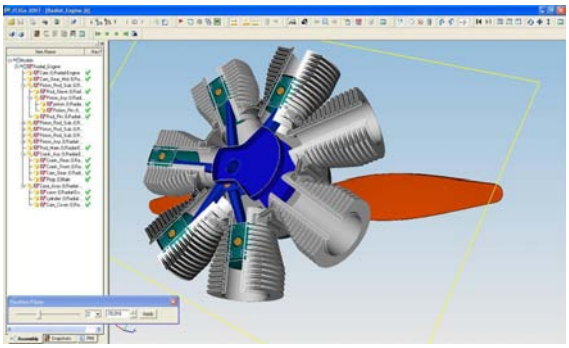
Le format JT est conçu pour permettre une visualisation temps réel et une analyse des modèles CAO lors des réunions de projet éventuellement effectuées à distance. Cette visualisation peut être réalisée en utilisant le visualisateur mono poste gratuit JT2Go ou le logiciel propriétaire UGS Teamcenter Visualisation Mockup.

Il permet de stocker :

- les différents assemblages de sous ensembles et de pièces constituant le produit organisés suivant une arborescence de conception illustrée sur la figure 2.25b,
- les représentations B-Rep aux formats standard Parasolid XT [77, 78](version 19.0),
- la définition analytique de la géométrie (Définition mathématique des surfaces des composants),
- les représentations facettisées des produits sur 3 degrés de précision différents (L.O.D.) pour une optimisation de l’affichage des modèles,
- les informations liées à l’industrialisation du produit et de ses composants (P.M.I. : Product Manufacturing Information),
- une représentation hiérarchique à base de boîtes et de sphères englobantes,
- les attributs de visualisation (matériaux, textures, éclairages et shaders).

Bien entendu, ces différentes données ne sont accessibles que dans la mesure où elles sont précisées lors de la construction du modèle CAO et si elles sont autorisées à l’exportation lors de sa sauvegarde du modèle sous format JT suivant le niveau de confidentialité désiré. En revanche, contrairement au format 3DXML de Dassault Systèmes, le format JT ne prévoit pas de décrire le comportement des objets stockés.

(a) Visualisation d’un modèle CAO au format JT



(b) Visualisation de l’architecture d’un modèle CAO au format JT

FIGURE 2.25 – Exemple de données CAO au format JT visualisée grâce au logiciel JT2Go

2.2 Planification automatique de trajectoire

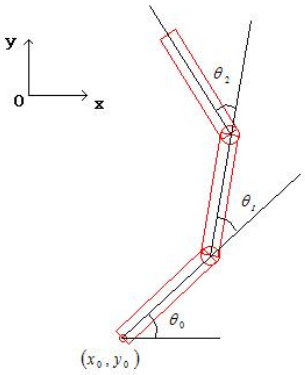
2.2.1 Introduction et Classification

2.2.1.1 Définition du problème

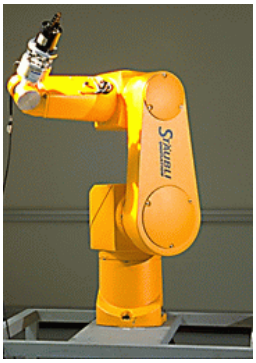
A l'origine, la planification de trajectoire en robotique est définie à partir de la définition du problème dit du "déménageur de piano" illustré en figure 2.26 que l'on peut formuler de la façon suivante : "Soit un piano (ou un robot) et des obstacles ; étant donné deux positions permises du piano, existe-t-il un mouvement de ce piano entre ces deux positions sans collision avec les obstacles, et si oui, donner un tel mouvement".

FIGURE 2.26 – Problème du "déménageur de piano" : trouver un mouvement sans collision entre une configuration initiale (verte) et une configuration finale (rouge) dans un espace encombré.

Dans le cas général, l'objet à déplacer, un robot par exemple, n'est pas nécessairement un corps rigide, mais peut être constitué d'éléments articulés et possède ainsi un certain nombre n_{ddl} de degrés de liberté (cf. figure 2.27). Ce nombre de degrés de liberté (ddl) détermine le degré de complexité du problème de planification de trajectoire du robot considéré.



(a) Robot à 3 ddl



(b) Robot Industriel à 6 ddl

(c) Robot Humanoïde HRP2 à 32 ddl

FIGURE 2.27 – Les degrés de liberté (ddl) en robotique

2.2.1.2 Notion d'espace des configurations

L'espace des configurations noté $\mathbb{C}$ représente les différentes positions et orientations possibles que le "robot" peut occuper dans l'espace. Cet espace comporte autant de dimensions que le nombre de degrés de liberté *ddl* du robot. La configuration du robot mobile notée q est alors définit tel que $q \in \mathbb{C}$ avec $dim(q) = ddl$.

Dans l'exemple en deux dimensions d'un corps rigide libre, illustré en figure 2.28, sachant que le carré peut se mouvoir en 3 dimensions (x, y, θ) et connaissant la géométrie de l'environnement disponible, on peut générer un espace des configurations *libre* d'obstacles noté $\mathbb{C}_{libre}$ de dimension 3.

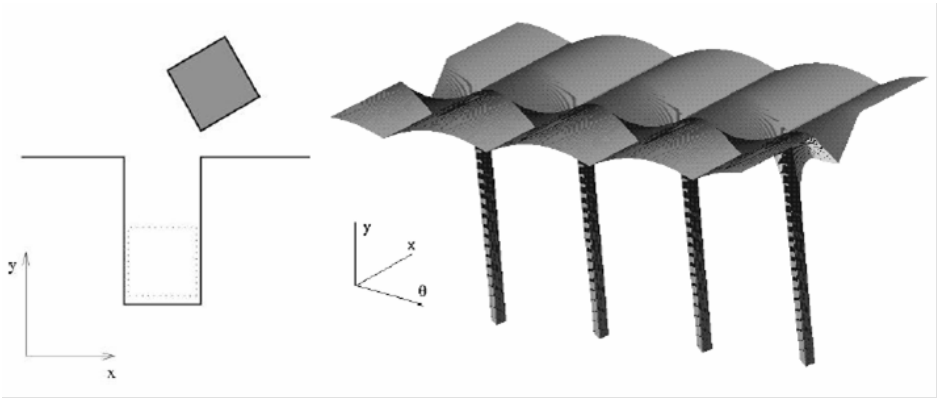


FIGURE 2.28 – Exemple de génération d'un espace des configurations, cas du "robot" carré et d'un obstacle formant un trou carré.

Dans l'exemple, l'espace des configurations est de dimension 3, l'espace des obstacles est la traduction en 3d de l'obstacle avec le trou dans lequel peut descendre le robot (le carré dans la figure 2.28) à condition que ses cotés soient alignés avec les bords de l'obstacle (d'où la répétition de la colonne suivant l'axe y pour les abscisses $\theta = k \times \pi/2$, $k \in \mathbb{N}$ dans l'espace libre). Dans le cas d'une structure articulée comme pour les exemples illustrés en figure 2.27 comportant de nombreux ddl, l'expression de l'espace des configurations est plus complexe car comportant un très grand nombre de dimensions, ce qui complexifie le calcul de $\mathbb{C}_{libre}$ ou encore la détermination d'une trajectoire libre dans celui-ci.

Cette notion d'espace des configurations permet de transformer le problème de planification de la trajectoire d'un "robot" en problème de trajectoire d'un point dans l'espace des configurations depuis la configuration initiale $q_{initiale}$ jusqu'à la configuration q_{finale}. Le problème se limite donc, soit à déterminer $\mathbb{C}_{libre}$ puis à choisir une trajectoire optimale dans cet espace, soit, plus simplement, à déterminer une trajectoire sans contact dans $\mathbb{C}$, non unique et sans garantie d'optimalité. Les méthodes de planification automatique utilisées en robotique se consacrent plus généralement à résoudre un problème par la découverte d'une fonction Γ telle que :

$$\Gamma : [0, 1] \rightarrow \mathbb{C}_{libre}$$

$$\text{avec } \Gamma(0) = q_{initiale} \text{ et } \Gamma(1) = q_{finale}$$

2.2.1.3 Catégorisation des méthodes

Traditionnellement [79] , la planification de mouvements est réalisée au moyen de :

- *méthodes globales* fournissant la description exacte ou approchée de l'espace des configurations libres,
- méthodes heuristiques : les *méthodes locales* et les *méthodes probabilistes* ne capturant qu'une connectivité de l'espace des configurations libres.

La complexité des *méthodes globales* rend leur utilisation difficile dès que le système comporte un nombre de degrés de liberté élevé du fait de la difficulté de transcription de la définition des obstacles tridimensionnels dans l'espace des configurations du robot mais garantissent une *complétude déterministe*, c'est à dire la garantie de trouver la solution optimale du problème posé si il en existe au moins une.

Les *méthodes locales,* quant à elles, souffrent des inconvénients liés à la non complétude : elle ne permettent pas de garantir la découverte d'une solution notamment dans le cas d'environnements complexes se traduisant par une définition de $\mathbb{C}_{libre}$ mettant en échec l'heuristique utilisée (par exemple : des minima locaux, pour la "méthode du potentiel"). Elles présentent néanmoins l'avantage d'être moins coûteuses en temps de calcul que les méthodes globales.

Plus récemment, de nombreuses approches non déterministes sont apparues, elles ne reposent plus sur une construction préalable d'une représentation exacte ou approchée de $\mathbb{C}$ mais se basent plutôt sur l'exploration aléatoire d'un espace des configurations admissibles et à la capture de sa connectivité. Ces nouvelles méthodes de planification dites «méthodes probabilistes», permettent de faire face à la complexité des problèmes de planification pour des systèmes comportant un grand nombre de ddl. Ces méthodes vérifient une propriété de *complétude probabiliste*, c'est à dire garantissant en probabilité de trouver une solution en temps fini, lorsqu'il en existe une. Certes, cette propriété est moins forte que la complétude déterministe vérifiée par les méthodes exactes mais ces nouvelles approches sont désormais moins sensibles à la dimension de l'espace de recherche et s'avèrent très efficaces en pratique pour des espaces de forte dimension (nombre de dimensions supérieur à 4).

2.2.2 Méthodes globales

Les méthodes *globales* sont des méthodes de calcul basées sur la géométrie et qui l'exploitent de manière à créer un maillage permettant d'établir différents chemins depuis la configuration initiale jusqu'à la configuration finale et ainsi de capturer la connectivité complète ou approchée de l'espace des configurations libres.

Il existe deux grands types de méthodes globales : les méthodes "squelette" et les méthodes de "décomposition en cellules".

2.2.2.1 Méthodes squelette

Cette approche cherche à construire une représentation de l'espace des configurations à partir de portions d'espace de dimension inférieure à celle de l'espace des configurations. Les propriétés de connectivité sont gardées dans une nouvelle représentation appelée squelette. En effet, dans cette approche une représentation explicite des espaces libres connexes est utilisée. L'ensemble des trajectoires dans l'espace libre est contracté ou réduit dans un graphe. Ce graphe représente toutes les régions de l'espace libre. De cette manière, toute trajectoire dans l'espace libre est représentée par un ensemble d'arcs et de sommets. Pour chercher une trajectoire d'une configuration initiale à une configuration finale, il suffit de projeter les configurations initiale et finale sur deux sommets du graphe. Ensuite, une méthode de parcours de graphe est utilisée pour trouver l'ensemble de sommets et d'arcs décrivant la trajectoire optimale. Les approches existantes pour la construction d'une contraction de l'espace libre sont :

- les *graphes de visibilité*, consistent à capturer la connectivité de l'espace des configurations libres du robot grâce à une représentation par un réseau de courbes unidimensionnelles [79] (Figure 2.29, au centre) ; ils utilisent les sommets des obstacles.
- les *diagrammes de Voronoï*, consistent à construire une contraction de l'espace libre à partir des points équidistants entre les obstacles [79] (Figure 2.29, à droite) ce qui permet de générer des chemins passant le plus loin possible des obstacles.
- La méthode *silhouette,* décrite dans [1], utilise les extrema d'une fonction, permettant de discerner l'espace libre de l'espace des obstacles, définie dans des dimensions ajoutées à l'espace des configurations appelé plan de coupe (aussi appelée "tranche" ou "slice") et noté Q_λ. Le paramètre λ caractérise le plan de coupe et permet un balayage de l'espace des configurations. Pour chaque valeur de λ, les points critiques de la fonction sont déterminés. Une fois la totalité de l'espace des configurations balayé, l'algorithme crée alors une carte à partir des points critiques de chaque plan de coupe.

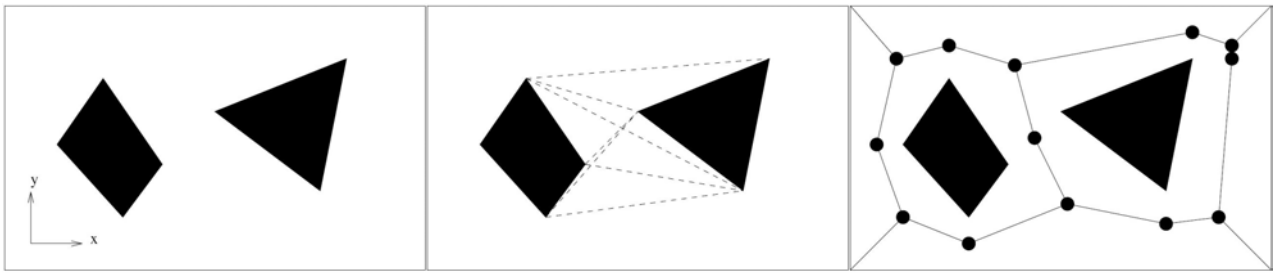


FIGURE 2.29 – Exemples de décompositions en chemins pré-calculés dans les cartes métriques : graphe de visibilité et diagramme de Voronoï

2.2.2.2 Méthodes de décomposition en cellules

Cette approche consiste à diviser l'espace de recherche en fractions de l'espace des configurations appelées cellules. La recherche d'une trajectoire est alors la construction d'un chemin entre deux cellules. Chaque cellule libre d'obstacles représente un sommet. Deux sommets sont connectés entre eux si les cellules sont

adjacentes [1, 79]. Les approches de décomposition en cellules peuvent être exactes ou approximatives :

- pour les approches exactes, il s'agit d'obtenir une représentation exacte de toutes les cellules complètement libres d'obstacles (Figure 2.30, Décomposition en cellules exactes),
- pour les approches approchées, la connaissance approchée de l'espace libre peut être représentée par exemple, sous la forme de quadtree (Figure 2.30, Décomposition en quadtree). Pour ce type de représentation, trois types de cellules sont considérés : les cellules libres (en blanc sur la figure 2.30) représentant une zone sans obstacle, les cellules obstacles (en noir sur la figure 2.30) qui représentent des cellules entièrement interdites et les cellules mixtes (en gris sur la figure) représentant des cellules contenant à la fois une partie libre et une partie obstacle. Les cellules mixtes peuvent être redécoupées en cellules libres, obstacles et mixtes de façon à obtenir une cartographie plus précise. On trouve également d'autres types de décompositions en cellules comme la décomposition en cellules rectangulaires ou régulières (Figure 2.30).

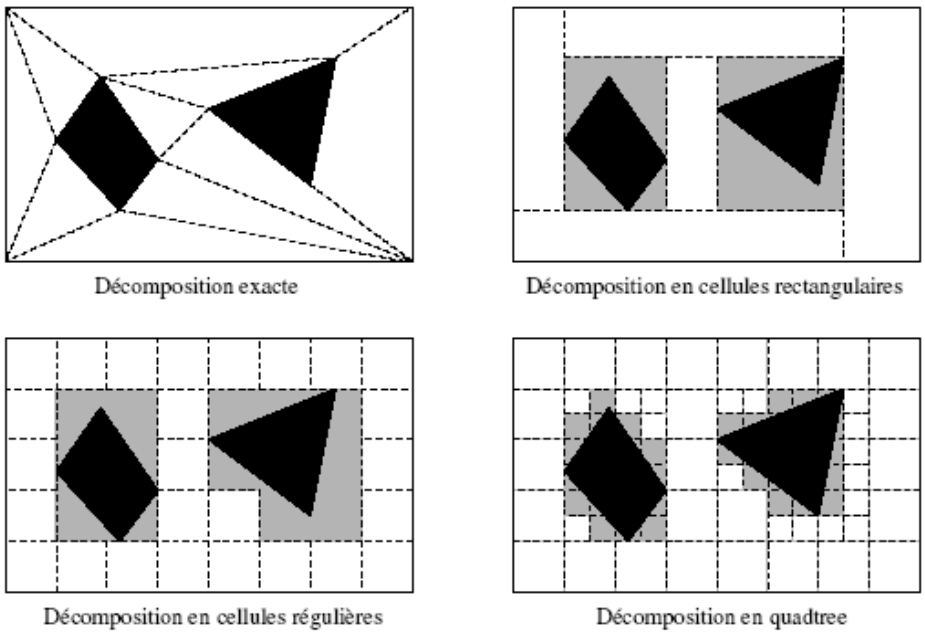


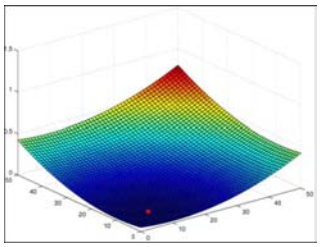
FIGURE 2.30 – Exemples de décompositions : Cellules exactes, cellules rectangulaires, cellules régulières, quadtree.

2.2.3 Méthodes locales

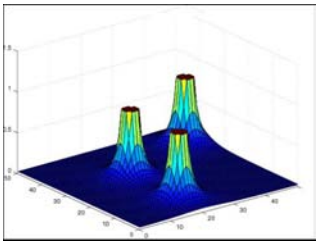
2.2.3.1 Méthode du champ de potentiel

L'idée de cette approche [80] consiste à considérer le robot comme une particule évoluant dans un champ de potentiels exercant sur celle ci des forces d'attraction et de répulsion. Ce champ de potentiel est calculé grâce à la somme de deux champs : le premier, illustré par la figure 2.31a, est calculé de manière à créer un potentiel attractif entre les configurations initiales et finale, le deuxième est généré par des fonctions décrivant des potentiels répulsifs à proximité des obstacles, illustré par la figure 2.31b. La somme de ces

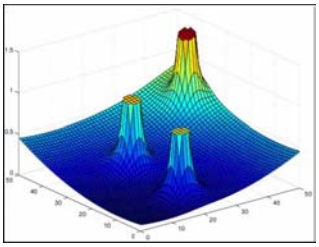
deux champs peut être représentée par un champ de courbes équipotentielles illustré par la figure 2.31c [79, 1]. Considérant que la particule est munie de l'énergie potentielle relative à sa configuration dans le champ de potentiel, la minimisation de l'énergie portée par la particule permet de décrire une trajectoire libre.



(a) Champ attractif vers la position finale



(b) Champ répulsif au niveau des obstacles



(c) Champ total

FIGURE 2.31 – Construction du champs de potentiels

Une fonction potentiel est une fonction dérivable $U : \mathbb{R}^m \mapsto \mathbb{R}$. Pour diriger le robot dans le champ de potentiel, on suit la descente du gradient de ce potentiel :

$$\dot{q}(t) = -\nabla U(q(t))$$

$$\text{avec} : \nabla U(q(t)) = \left[\frac{\partial U(q(t))}{\partial q_1}, \frac{\partial U(q(t))}{\partial q_2}, \ldots, \frac{\partial U(q(t))}{\partial q_m}\right],$$

$\dot{q}(t)$ et $q(t)$ étant les vecteurs vitesse et position du robot dans l'espace des configurations.

On suit ainsi l'évolution du gradient localement et on tend vers un minimum : $\nabla U(q*) = 0$. L'inconvénient de cette technique est que le robot peut être bloqué dans un minimum local de la fonction potentiel, différent du minimum global. Il est possible d'imposer que le potentiel calculé soit une fonction harmonique [81], ce qui garantit qu'il n'ait pas d'autre minimum que le minimum traduisant la position finale, mais cela complexifie beaucoup son calcul.

2.2.3.2 Méthode des contraintes

Dans le cadre d'une approche locale, Faverjon et Tournassoud proposent une méthode, dite des contraintes, qui aborde le problème de l'évitement d'obstacles en modélisant localement chacun de ceux-ci par l'ensemble de ses plans tangents [82]. La génération des mouvements du robot est obtenue par la minimisation d'un critère quadratique établi en fonction de la tâche à effectuer (le plus souvent, un critère de distance) en présence de contraintes linéaires associées aux équations des plans tangents aux obstacles. A la différence

de la méthode du potentiel, les obstacles n'agissent sur le robot, pendant le processus de minimisation, que lorsque celui ci est très proche et a tendance à y pénétrer. Toutefois, la méthode reste sensible à la présence de minima locaux. Ce problème est abordé dans [82] en combinant la méthode locale avec une technique d'apprentissage.

2.2.4 Méthodes Probabilistes

2.2.4.1 Méthode PRM, "Probabilistic Roadmap Method" :

Cette méthode a été introduite par Kavraki et Latombe sous le nom de PRM (Probabilistic Roadmap Methods) [1]. Depuis, plusieurs variantes de cette méthode ont été proposées dans la littérature mais toutes se basent sur le même concept. La technique développée se décompose en trois phases : la phase «d'apprentissage » de la topologie et les phases de recherche et de lissage d'une solution.

- Phase d'apprentissage (Figure 2.32, a) : pendant cette phase, l'algorithme crée une "roadmap" grâce à des jets de configurations dans l'espace des configurations libres $\mathbb{C}_{\text{libre}}$. Le programme teste ensuite grâce à une méthode locale si le passage entre ce nouveau point et les points précédents est sauf (ne rencontre pas d'obstacle), et construit ainsi la "roadmap". Les configurations sont "jetées" de manière probabilistes, ce qui permet un réglage de la méthode en fonction du problème.
- Phase de recherche (Figure 2.32, b) : une fois la roadmap créée, le programme choisit le passage optimal parmi les différents passages possibles.
- Phase d'optimisation et lissage (Figure 2.32, c) : cette phase n'est pas obligatoire mais permet de lisser le chemin choisi et ainsi la trajectoire du robot.

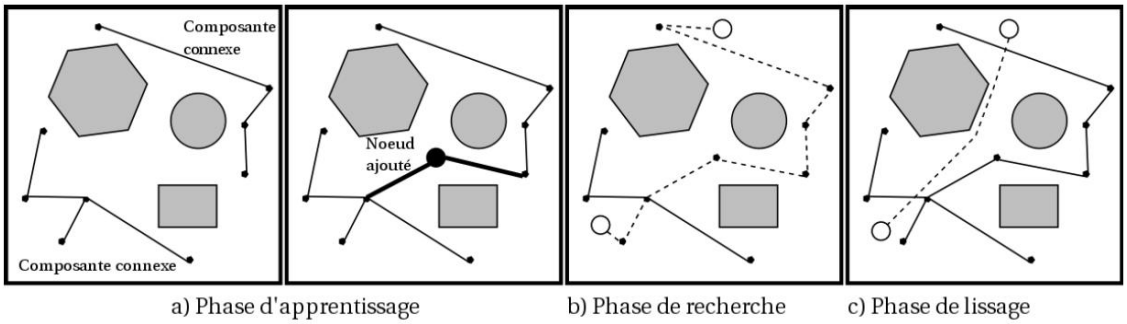


FIGURE 2.32 – Illustration des 3 phases de la méthode PRM

Cette méthode est très populaire dans le milieu de la robotique car elle permet de résoudre des problèmes ayant un nombre important de degrés de liberté en un temps réduit [83]. Cependant, elle comporte encore quelques problèmes : durant la phase d'apprentissage, la méthode de jet de points utilisée ("sampling") doit être adaptée au problème pour que la complétude probabiliste soit admissible en pratique, en particulier lorsque l'environnement comporte des passages étroits. En vue d'améliorer cette méthode dans tout les cas de figure, c'est à dire améliorer la répartition des points jetés, beaucoup de publications exposent des répartitions de points gaussiennes, adaptatives (en fonction de l'environnement) ou satisfaisant au mieux un critère d'optimisation [84, 85].

2.2.4.2 Méthodes par diffusion

Les méthodes par diffusion reposent sur un principe similaire aux méthodes probabilistes classiques (PRM et ses dérivées). Cependant, les phases d'apprentissage et de recherche sont confondues pour trouver une solution propre à chaque paire de configuration initiale et finale. Ainsi la "roadmap" construite dépend des configurations initiale et finale du problème. Pour le même environnement, le même robot et la même méthode locale, ces méthodes ne peuvent souvent pas réutiliser efficacement une roadmap construite lors d'une requête préalable pour résoudre un autre problème de planification.

2.2.4.2.1 La méthode Rapidly-exploring Dense Tree (RDT)

La méthode RRT introduite par Steven LaValle [86] consiste en l'utilisation de structures de données aléatoires appelées "Rapidly-exploring Random Tree" (RRT). Ces structures peuvent être assimilées à des arbres construits par diffusion (Figure 2.33). Dans le principe, la structure est construite grâce à des ajouts successifs de nouveaux points, aléatoirement positionnés mais toujours raccordés aux points de l'arbre les plus proches. Les "branches", reliant deux points de l'arbre, sont déterminées en respectant les contraintes de trajectoire imposées par le système commandé. Par exemple, dans le cas de planification de trajectoire pour des robots non holonomes les branches générées doivent respecter des contraintes de chemin à courbure continue. Ce mode de propagation permet d'explorer progressivement l'espace des configuration en testant des trajectoires possibles depuis une configuration donnée. Le réglage de la méthode est essentiellement réalisé grâce à la définition de contraintes de trajectoire et à la méthode utilisée pour le jet de nouveaux points.

Les algorithmes de classe RDT découlent historiquement de la généralisation de l'utilisation des methodes RRT. Leur construction est réalisée de manière itérative [86], soit de manière totalement aléatoire (RRT) ou en imposant des contraintes lors des phases de jet de points.

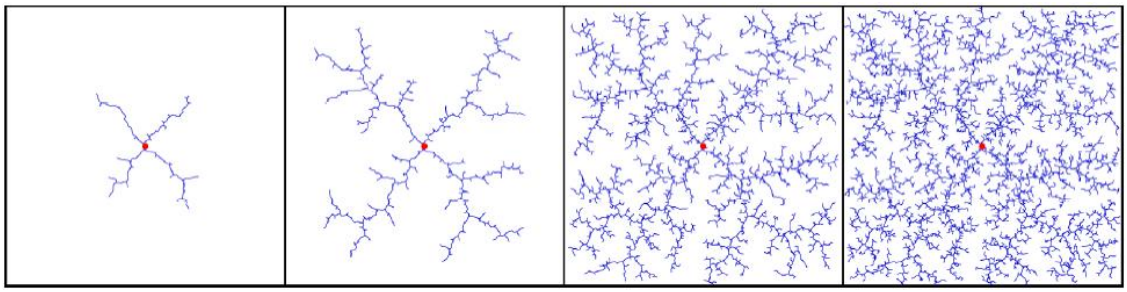


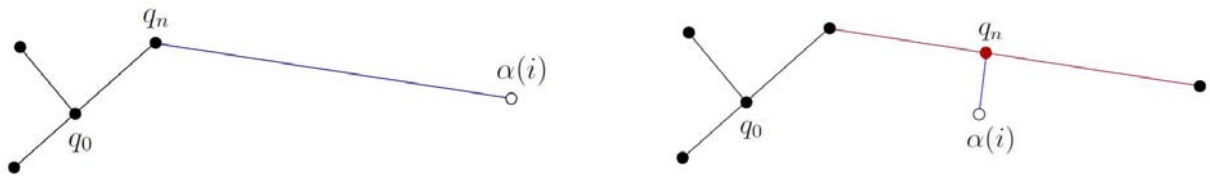
FIGURE 2.33 – Illustration du principe de diffusion : exemple de diffusion pour une méthode RRT

Au début de l'exécution de la méthode décrite par l'algorithme 2.1, l'arbre RDT ne contient que la configuration de départ. Chaque opération de croissance de l'arbre est réalisée grâce au jet d'une configuration α_i. Pour chaque configuration α_i, la configuration la plus proche q_n est repérée dans l'arbre existant, que cette configuration soit un noeud de l'arbre (Figure 2.34a) ou parmi les points d'un segment (Figure 2.34b). Une opération de détection de collision permet de tester la viabilité du segment $[q_n, \alpha_i]$. Dans le cas de la détection d'une collision sur le segment $[q_n, \alpha_i]$, celui ci est modifié en fonction de la dernière configuration libre

Algorithme 2.1 Algorithme de croissance d'un arbre RDT dans un environnement encombré

```
RDT(q_0)
1  G.init(q_0);
2  for i = 1 to k do
3      q_n ← NEAREST(S, α(i));
4      q_s ← STOPPING-CONFIGURATION(q_n, α(i));
5      if q_s ≠ q_n then
6          G.add_vertex(q_s);
7          G.add_edge(q_n, q_s);
```

détectée q_s comme illustré par la figure 2.35. Le segment $[q_n, q_s]$ est ensuite ajouté comme une nouvelle "branche" de l'arbre RDT.



(a) Cas où q_n est un noeud de l'arbre

(b) Cas où q_n appartient à un segment

FIGURE 2.34 – Détermination de la configuration q_n de l'arbre RDT la plus proche de la configuration α_i jettée

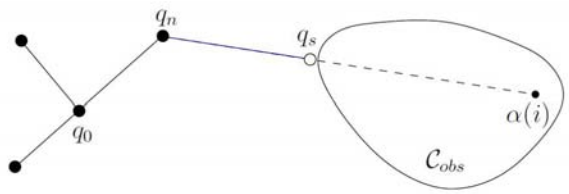


FIGURE 2.35 – Modification d'un segment lors de la détection d'une collision pendant la phase de test de viabilité

L'opération de croissance est arrêtée par une limitation K du nombre de jets de configuration ou lorsque la configuration finale peut être connectée à l'arbre RDT.

En pratique, on utilise un algorithme RDT bi-directionnel pour la recherche d'une trajectoire libre entre les configurations initiale $q_{initiale}$ et finale q_{finale} décrit par l'algorithme 2.2. Dans ce cas, deux arbres sont utilisés : l'un constituant l'ensemble des chemins issus de la configuration initiale, l'autre les chemins issus de la configuration finale. Les opérations de croissance sont réalisées par le jet d'une configuration α_i pour les deux arbres. L'algorithme est arrêté lors de la connection des deux arbres ou lorsque le nombre maximal de jets de points K est atteint.

La méthode RRT dite originale classique a donné lieu à de nombreuses déclinaisons que l'on peut classer en 2 grandes catégories :

Algorithme 2.2 Algorithme RRT Bi directionnel

```
RDT_BALANCED_BIDIRECTIONAL(q_I, q_G)
1   T_a.init(q_I); T_b.init(q_G);
2   for i = 1 to K do
3       q_n ← NEAREST(S_a, α(i));
4       q_s ← STOPPING-CONFIGURATION(q_n,α(i));
5       if q_s ≠ q_n then
6           T_a.add_vertex(q_s);
7           T_a.add_edge(q_n, q_s);
8           q'_n ← NEAREST(S_b, q_s);
9           q'_s ← STOPPING-CONFIGURATION(q'_n,q_s);
10          if q'_s ≠ q'_n then
11              T_b.add_vertex(q'_s);
12              T_b.add_edge(q'_n, q'_s);
13          if q'_s = q_s then return SOLUTION;
14      if |T_b| > |T_a| then SWAP(T_a, T_b);
15  return FAILURE
```

Les dérivées classiques (RRT-Like Algorithms)

Certaines de ces déclinaisons se sont destinées à la problématique de planification de mouvement pour les systèmes comportant un très grand nombre de degrés de liberté tels que les robots anthropomorphes, la robotique médicale ou encore les simulations moléculaires [87]. Ces déclinaisons conservent une composante purement aléatoire dans la diffusion de jet de points mais introduisent des pondérations dans la priorité de mouvement de certaines articulations de la chaîne cinématique, des méthodes particulières de calcul de norme (m-RRT) utilisées pour sélectionner le plus proche voisin d'une configuration donnée, ou encore des heuristiques de choix de configuration de raccordement parmi les plus proches points de l'arbre (k-RRT) ou de suppression de noeud de l'arbre ne permettant pas d'expansion (l-RRT).

Les possibilités d'amélioration des RDTs

Différentes méthodes permettent d'accélérer la vitesse de convergence des méthodes RDT vers une solution. L'une des méthodes consiste en une restriction de l'espace de travail de la méthode [88, 89] afin d'obtenir une vitesse de convergence moins dépendante du contexte. Dans cette optique, les différentes configurations α_i jetées lors de la croissance des arbres sont choisies par l'application d'heuristiques dans des zones de l'espace cartésien stratégiquement avantageuses déterminées par des études préliminaires ou dynamiques de l'espace de travail [84]. D'autre méthodes permettent de fiabiliser le temps de convergence en triant les branches crées suivant leur utilité [90].

2.2.4.2.2 Algorithme Fil d'ariane

L'algorithme « Fil d'Ariane », décrit dans [91], utilise une méthode d'optimisation des plans, semblable à des algorithmes génétiques, pour résoudre des problèmes de planification de trajectoire. Les auteurs ne se placent pas, comme dans la plupart des planificateurs, dans l'espace des configurations mais plutôt dans l'espace des commandes. Ils définissent un plan comme un ensemble de commandes pour les actionneurs. Pour certain robots, par exemple des robots non holonomes, il peut être délicat de trouver une méthode

locale qui permet de définir un mouvement entre deux configurations aléatoires qui respecte la cinématique du robot. Si on se place dans l'espace des commandes, on ne cherche pas forcément à générer une portion de trajectoire vers une configuration précise. La configuration finale sera calculée par un enchaînement de commandes qui, dans le cas de robots non holonomes, est un calcul beaucoup plus simple car il est réalisé sur des paramètres indépendants. Ainsi l'algorithme du fil d'Ariane remplace l'utilisation d'une méthode locale par une optimisation sur les plans possibles pour trouver une configuration finale proche de la configuration finale désirée. Pour résoudre le problème des minima locaux de l'optimisation des plans, les auteurs construisent un arbre de plans permettant l'exploration de tout l'espace libre. Cet arbre est obtenu en posant des balises dans l'espace accessible du robot. Du point de vue de l'implémentation, l'algorithme « Fil d'Ariane » est composé de deux sous algorithmes SEARCH et EXPLORE.

Le premier, SEARCH, collecte des informations sur l'espace accessible depuis la position initiale en posant des balises dans l'espace de recherche et en mémorisant les chemins entre ces balises et la position initiale.

Le second essaie de placer de nouvelles balises aussi loin que possible de celles déjà placées. La figure 2.36 illustre l´évolution des balises placées par l'algorithme. EXPLORE fait un pavage progressif de l'espace des configurations assurant la complétude de l'algorithme.

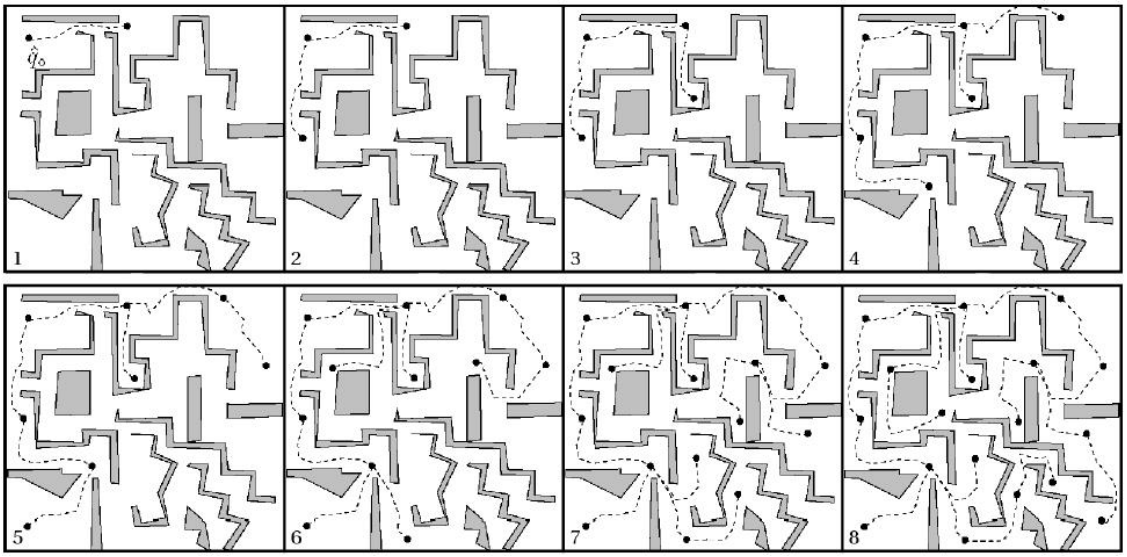


FIGURE 2.36 – Évolution des balises placées par l'algorithme SEARCH de la méthode du "Fil d'Ariane"

2.3 Projets et prototypes proches de notre contexte

Par le biais de projets tels que Perf-RV 1 et 2 [92], de nombreux démonstrateurs et prototypes ont été développés pour promouvoir le développement et l'utilisation de la réalité virtuelle dans le domaine industriel. Ces partenariats regroupant de nombreux laboratoires tels que l'INRIA, le CEA/LIST, l'École des Mines de Paris, l'Institut Image/ENSAM, le LABRI , le Laboratoire de Robotique de Versailles (LRV) ou encore le LIMSI, ainsi que de grandes sociétés européenes tels que EADS, Renault, PSA, Dassault aviation, GIAT-industries ou encore EDF, ont débouché sur des avancées scientifiques permettant une première mise en oeuvre pratique de technologies liées à la réalité virtuelle dans un contexte industriel proche du notre parmi lesquelles :

- la simulation d'assemblage et de démontage,
- la formation au geste technique.

2.3.1 Simulation d'assemblage ou de démontage en environnement immersif

Ce thème, exploré par le projet PerfRV 1, a donné lieu à la création de plusieurs démonstrateurs mêlant à la fois des interfaces haptiques et une visualisation immersive dans un couplage visuo-haptique illustré en figure 2.37a, le développement d'outils pour la conception et la revue de projet immersives illustré en figure 2.37b, tout ceci dans le but d'intégrer ces techniques dans le processus de conception et de développement de produits industriels.

On retrouve dans ce thème l'utilisation et l'intégration des technologies développés pour l'utilisation d'interfaces haptiques et des systèmes de visualisation afin de réaliser des simulations d'assemblage et de démontage pour l'ingénierie notamment pour l'aéronautique et de l'automobile.

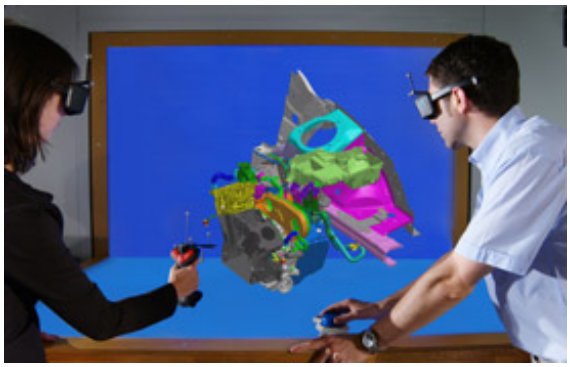
Plus largement, on notera dans ce thème les travaux réalisés au :

- CEA-LIST [26] sur le moteur de calcul de physique "Interactive Physics Simulation Interface" (IPSI) pour les simulations haptiques utilisant des périphériques de type Virtuose ou Inca accessible au moyen de l' "Interactive Physic Pack" (IPP) sous Virtools dev illustré en figure 2.38a et IFC Core sous Catia V5 illustré en figure 2.38b. On notera également le projet d'évaluation de la maintenabilité à partir de modèles CAO dans des simulations immersives EMM3d [93]
- VRCIM [94] à la Washington State University sur le projet Virtual Assembly Design Environment (VADE) [95, 96] permettant la conception et l'évaluation de procédures de montage et de démontage de composants CAO en environnement virtuel [97, 98].

2.3.2 Formation des personnels

On distingue deux types de formations interactives dans les développements à vocation industrielle actuels en réalité virtuelle :

(a) Couplage Haptique et visualisation immersive

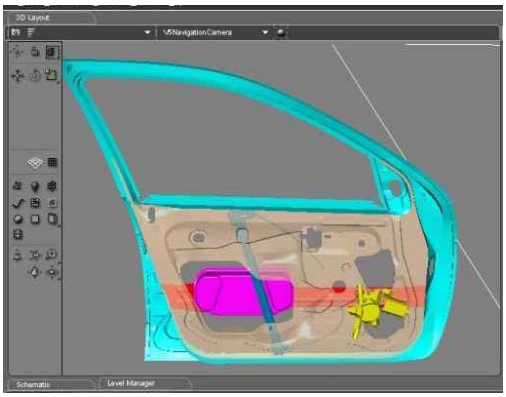


(b) Revue de projet immersive

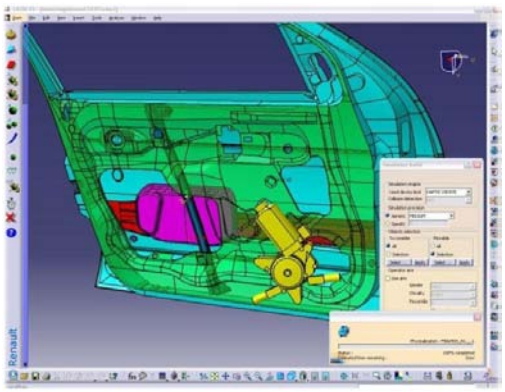
FIGURE 2.37 – Illustration de l'utilisation d'interfaces haptiques et de visualisations immersives de modèles 3d industriels

2.3.2.0.3 Les simulations interactives centrées sur l'apprentissage des procédures de montage et de démontage Dans cette catégorie, on trouve notamment le simulateur Generic Virtual Training [99]. Ces programmes permettent à un ou plusieurs apprenants de tester progressivement la mémorisation de procédures d'intervention (cf. figure 2.39a). Ce type de simulation s'avère très utile notamment dans le domaine militaire où le matériel nécessaire dans le cadre de formations classiques demeure coûteux et dangereux (cf. figure 2.39b).

2.3.2.0.4 Les simulations permettant une formation des personnels au geste technique Ces approches visent à former des intervenant techniques à des tâches nécessitant une dextérité particulière difficilement explicitable sur des supports classiques de formation. Elles permettent, par exemple, dans le cas du projet Samira lancé par EADS et illustré en figure 2.40a, d'une part, de prouver la faisabilité et d'évaluer la difficulté d'un geste de montage dans des environnements encombrés et d'autre part de conserver et transmettre les savoir-faire liés à la manipulations d'objets comme par exemple des astuces de montage dits "en aveugle". D'autres approches, comme le projet Virtual Painting développé par PSA et illustré en figure



(a) Interactive Physic Pack (IPP) pour Virtools dev



(b) IFC Core sous Catia V5

FIGURE 2.38 – Utilisation de la bibliothèque logicielle IPSI sous virtools dev et Catia V5

(a) Formation aux procédures de montage en environnement immersif collaboratif

(b) Formation sur matériel militaire ou dangereux

FIGURE 2.39 – Generic Virtual Training (GVT) - Formation interactive

2.40b, se consacrent davantage à une évaluation d'apprenants sur des gestes techniques spécifiques dans des simulations visuo-haptiques réalistes.

2.3.3 Planification "interactive" de trajectoire

Finalement, peu d'équipes ou de projets se sont intéressés à une planification réellement interactive de trajectoire. On citera néanmoins les travaux réalisés par :

- O. Bayazit au Department of Computer Science de la Texas A&M University [100] qui a exploré les possibilités d'interaction entre un utilisateur humain et une méthode de planification de type PRM au moyen d'une interface haptique.
- C. Vázquez et J. Rosell à l'Institute of Industrial and Control Engineering (IOC-UPC) de Barcelone[101] qui ont réalisé une méthode de planification basée sur l'utilisation des fonctions harmoniques (analogue

(a) Illustration d'évaluation et de formation au démontage de composants aéronautique dans le cadre du projet Samira (EADS)

(b) Projet d'évaluation et de formation à la peinture en milieu automobile "Virtual Painting" développé par PSA

FIGURE 2.40 – Simulations interactives pour la formation des personnels intervenants au geste technique

à une méthode du potentiel) pour le guidage haptique de l'utilisateur lors de tâche de téléopération en robotique.

- H. Xuejian au Department of Mechanical Engineering à l'université de Hong Kong [102] qui étudie l'influence de l'intervention humaine sur la vitesse de convergence d'un planificateur de type PRM lors de tâches de planification de trajectoire en robotique grâce à la spécification au moyen d'une interface haptique de configurations particulières de passage pour le robot manipulateur.

2.4 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art dans des domaines de la réalité virtuelle et de la planification de trajectoire. Ce travail de synthèse de l'existant permet de mesurer les avancées réalisées dans ces deux domaines.

Le domaine de la réalité virtuelle offre une large gamme de matériels et de méthodes permettant de plonger l'utilisateur en état d'immersion pseudo-naturelle dans un environement virtuel. Ces différents outils permettent de réaliser des simulations interactives réalistes directement à partir de données industrielles extraites des formats CAO. Cependant le traitement nécéssaire de ces données reste en grande partie manuel et nécessite l'intervention de personnels experts. De plus les prototypes et méthodes développées étant souvent utilisés comme démonstrateurs de faisabilité, leur mise en oeuvre reste difficile dans le cas de conception de simulations génériques. Il apparait donc nécessaire de regrouper les études et prototypes réalisées dans une seule et même simulation interactive afin de permettre une utilisation plus conviviale de ces différents outils dans un contexte industriel.

Le domaine de la planification de trajectoire regroupe une large gamme de méthodes permettant de définir des trajectoires sans collision dans des environnements encombrés. Cependant, ces méthodes, largement

étudiées et appliquées en robotique, ont pour la plupart été conçues pour des utilisations hors ligne. De plus, l'inventaire réalisé dans ce chapitre ne laissent pas apparaître de méthode "parfaite" permettant de garantir la découverte rapide d'une trajectoire sans collision dans des environnements a priori inconnus. Chaque méthode décrite dans ce chapitre possède ses propres avantages et au moins un cas particulier permettant de démontrer ses faiblesses. L'une des pistes possibles permettant de réduire la sensibilité des méthodes automatiques vis à vis du contexte de planification, c'est à dire à la fois la nature de l'objet déplacé et la complexité de l'environnement dans lequel il est amené à évoluer, consiste à intégrer le point de vue d'un utilisateur humain dans le processus de planification. Ce nouveau type de méthode dite de "planification interactive" permet de combiner la caractéristique intuitive du raisonnement humain aux performances des méthodes automatiques existantes.

Les recherches bibliographiques confirment l'originalité du sujet de cette thèse par le faible nombre de projets traitant de "planification interactive". En effet, l'utilisation de méthodes de planification dans des environements interactifs introduit une nouvelle contrainte dite de "temps réel" qui n'est souvent pas prévue dans la conception initiale des algorithmes automatiques. Ce contexte d'utilisation implique d'une part, d'adapter les méthodes de planification existantes afin de les utiliser de manière interative et, d'autre part, de définir des structures de données et des modalités d'interaction avec l'utilisateur permettant leur exploitation dans des simulations de réalité virtuelle.

Chapitre 3

Architecture de simulation interactive

Sommaire

3.1 Introduction

Les entreprises industrielles montrent de plus en plus d'intérêt pour la réalité virtuelle [103, 104, 105, 7, 20]. Beaucoup de simulations interactives ont été mises au point pour résoudre des problèmes industriels tels que le montage et le démontage de produit [106, 107], la maîtrise du coût de formation du personnel ou encore pour des études d'ergonomie [108] ainsi que les supports d'analyse associés [109] dont nous avons décrit quelques exemples en partie 2.3.

La plupart des solutions qui sont appliqués à l'heure actuelle en industrie représentent des solutions propriétaires dédiées à chaque métier. En effet, dans la majorité des cas, les industriels font appels à des solutions

logicielles pour analyser et résoudre les problèmes rencontrés au cours du developpement du produit. Ils utilisent par exemple des outils logiciels d'études d'assemblage par calcul de trajectoire automatique diponibles dans *Catia*® V5, ou encore des solutions pour l'étude de process comme *Delmia*®. Ces différents outils sont généralement intégrés dans un système de PLM très lourd et faiblement réactif du point de vue du ou des utilisateurs.

Le but de notre approche est de fournir un outil intégré qui permet à des utilisateurs, n'ayant pas de connaissance importantes en infographie ou en réalité virtuelle, d'évaluer la validité de la conception d'équipements dans différents contextes de leur cycle de vie de manière naturelle et rapide. Dans ce but, l'outil développé dans ce travail regroupe dans une seule application un grand nombre de fonctionnalités permettant la manipulation de prototypes virtuels. Cet outil de préconception sera également utilisé pour corriger rapidement les géométries en tenant compte de l'environnement du produit très tôt dans le processus de conception.

3.2 Plateforme expérimentale

La plateforme de réalité virtuelle acquise par le laboratoire ENIT-LGP et illustrée par la figure 3.1 est constituée :

- d'un système de visualisation en stéréoscopie passive sur un écran de 3 mètres de large et de 2.25 mètres de haut permettant une visualisation en groupe de modèles numériques en 3d à l'échelle "1",
- d'un système de capture de mouvement optique ARTrack 2 composé de quatre caméras permettant à la fois le pilotage direct d'avatars, la manipulation de modèles 3d, ainsi que le pilotage du point de vue dans la simulation,
- d'un bras haptique Virtuose 6D 35-45 qui permet une manipulation d'éléments 3d avec retour d'effort.

Nous utilisons une architecture logicielle permettant l'utilisation simultanée de l'ensemble du matériel du laboratoire illustrée par la figure 3.2 et reposant sur l'utilisation des logiciels suivants :

- Virtools Dev v 4.1 [54] utilisé comme plateforme pour la simulation interactive munie du VRPack pour une gestion centralisée des données de simulation et la gestion des interfaces de réalité virtuelle.
- Le logiciel IPP v1.0 (Interactive Physic Pack) est utilisé pour la communication entre Virtools Dev et le bras haptique Virtuose 6D 35-45.
- Le logiciel DTrack pour le calcul et la communication au format VRPN des données capturées par le système de caméras ART.

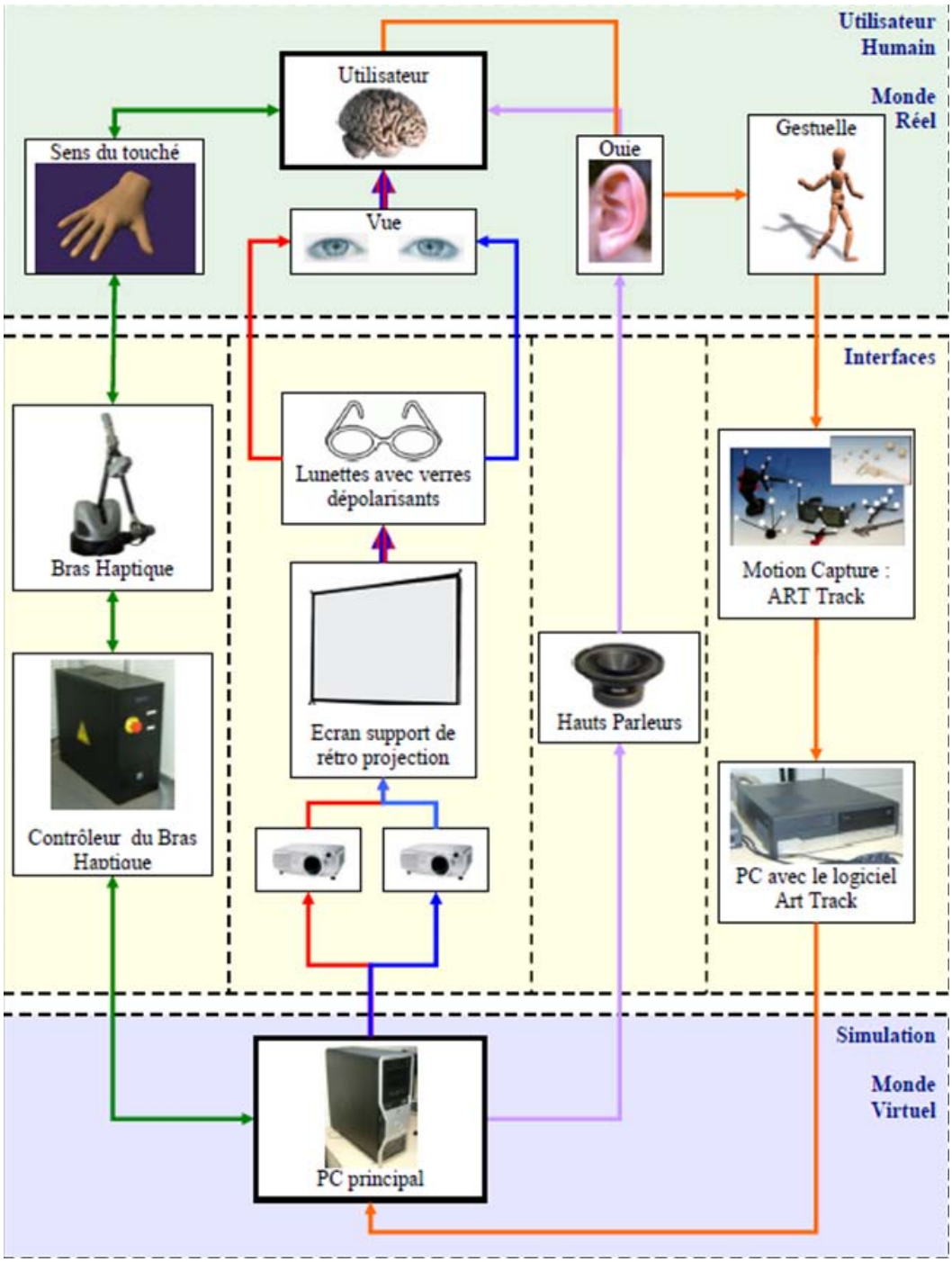


FIGURE 3.1 – Système de réalité virtuelle du LGP-ENIT

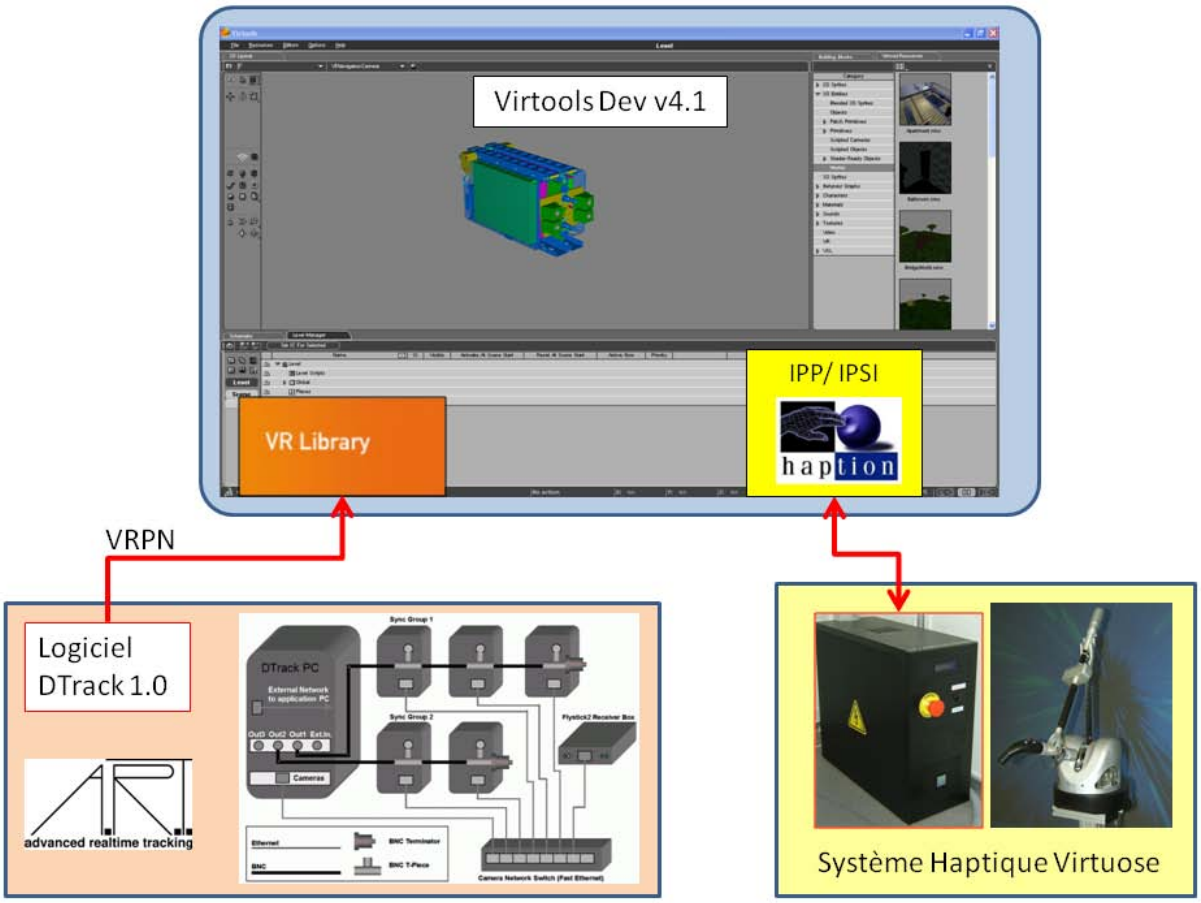


FIGURE 3.2 – Architecture logicielle

3.3 Architecture générale

3.3.1 Schéma général

La diversité des besoins spécifiés par Alstom Transport nécessite de créer une architecture unifiée alliant des capacités de visualisation interactive à de grandes possibilités d'interaction. Après analyse du cahier des charges, le flux d'informations et de commandes peut être schématisé comme illustré en figure 3.3.

3.3.2 Problématique liée au formatage des données

De par leur nature, les modèles 3d issus de la CAO sont difficilement exploitables dans les simulations interactives de RV [98]. Ils doivent donc être traités avant d'être chargés dans un environnement interactif.

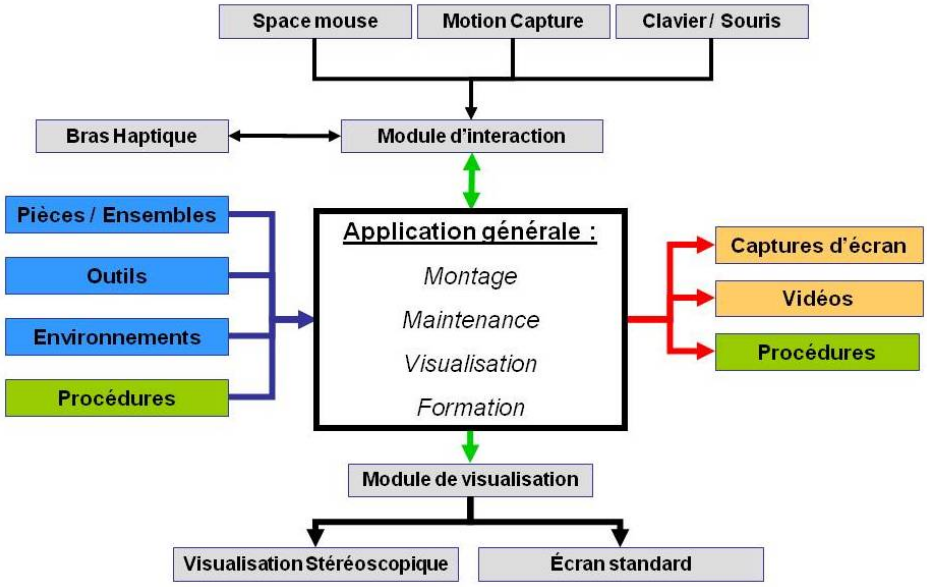


FIGURE 3.3 – Schéma flux de données dans l'architecture

3.3.2.1 Le format CAO des données d'entrée

La création de modèles 3d dans les applications CAO, illustré en figure 3.4, est réalisé premièrement par le dessin d'une esquisse 2D. Dans un deuxième temps, cette esquisse est projetée dans l'espace 3d par une opération d'extrusion ou de révolution formant ainsi un solide. Enfin des opérations booléennes sont réalisées entre ces solides de manière à aboutir à la forme finale d'une pièce.

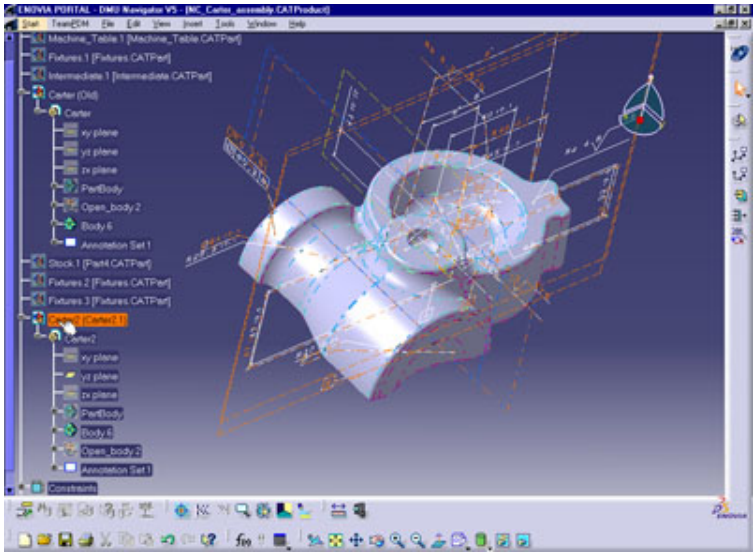


FIGURE 3.4 – Réalisation de modèles 3d avec *Catia*®V5

Ce mode de création de solide 3d repose sur deux méthodes de modélisation géométrique complémentaires :

1. la B-Rep, “Boundary Representation” : cette représentation des modèles repose sur une description surfacique par morceaux de tout ou partie du modèle 3d. Les différentes surfaces sont décrites par des NURBS (“Non-Uniform Rational Basis Splines”).
2. la CSG, “Constructive Solid Geometry” : cette représentation des modèles repose sur une successions d’opérations booléennes entre des primitives géométriques comme par exemple des parallélépipèdes rectangles, des cylindres, des sphères, des cônes, des tores, illustrée par l’arbre binaire de la figure 3.5.

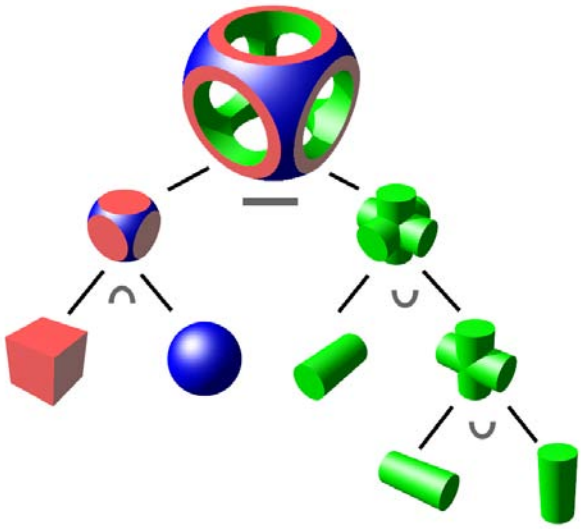
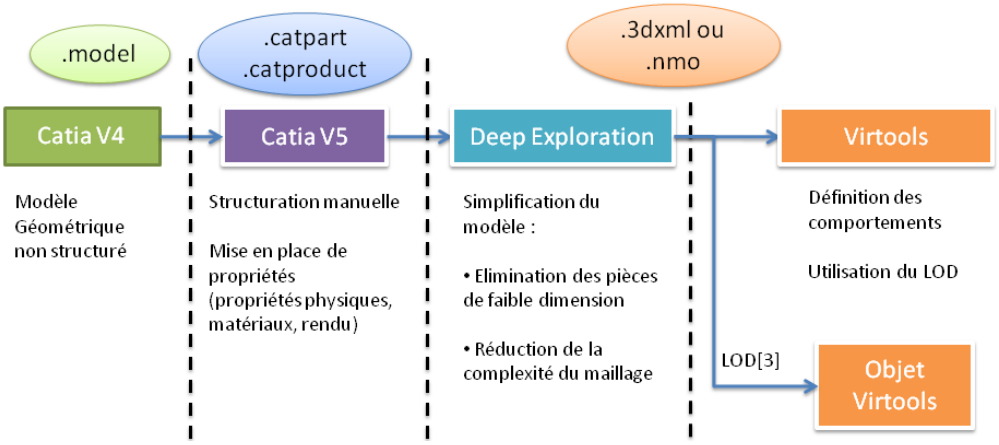


FIGURE 3.5 – Illustration d’un arbre CSG

3.3.2.2 Simplification des données CAO

Les modèles exportés à partir d’un logiciel CAO ne peuvent pas être directement pris en charge par la plateforme du fait de leur structure et de leur taille. Les modèles utilisés pour la simulation interactive doivent passer par des étapes de simplification et de réduction de maillage afin de pouvoir garantir une interaction en temps réel sur les systèmes actuel de visualisation. Une fois les modèles simplifiés et convertis, ils sont chargés dans la plateforme de réalité virtuelle. Initialement, cette opération est réalisée hors ligne, illustré en figure 3.6, grâce à l’utilisation du logiciel *Deep Exploration*® (Right Hemisphere) qui permet :

- d’une part de lire des formats CAO classiques (*Catia*®*V*4, *Catia*®*V*5, Pro Eng, *SolidWorks*® ...)
- de réparer les différentes erreurs présentes dans les modèles
- de réduire la complexité et donc la taille des maillages représentant ces modèles
- d’exporter le modèle ainsi préparé dans le format NMO (format natif des objets Virtools)



(a) Process de préparation des modèles “hors ligne”

(b) Opération de conversion et de simplification dans le logiciel Deep Exploration

FIGURE 3.6 – Préparation hors ligne des modèles en vue de leur utilisation dans une scène virtuelle

Cette méthode de préparation des modèles est très efficace et permet de réaliser plusieurs versions dégradées des objets qui doivent être utilisés dans l’environnement virtuel permettant ainsi une utilisation d’un système de LOD discret. Cependant, cette méthode de préparation est longue et nécessite de bonnes connaissances des problématiques liées à la gestion de la complexité d’une scène de réalité virtuelle.

Bien que le chargement des modèles puisse s’effectuer hors-ligne dans la plateforme de réalité virtuelle, le moyen le plus ergonomique pour l’utilisateur est de charger les modèles en ligne d’une manière interactive. Dans ce but, nous avons développé un module de chargement de modèles permettant à l’utilisateur de charger une pièce au format CAO dans le monde virtuel. Pour cela, nous définissons les différentes étapes à réaliser dans le processus de chargement de modèles :

3.3.2.2.1 Lecture du fichier

Cette opération nécessite l’acquisition de lecteurs de formats propriétaires CAO (*.CADPart et *.CADProduct pour CATIA V5) tels que ceux utilisés par le plug-in “CADPack” de Deep Exploration. Ces lecteurs communément appelés “parseurs” permettent de décoder ces fichiers binaires et de les transcrire dans des structures de données explicites rendant ainsi les informations qu’ils contiennent accessibles.

3.3.2.2.2 Structuration des données pour une utilisation temps réel

La description du modèle lue à l'étape précédente doit être interprétée afin d'être convertie en un ou plusieurs maillages de triangles. En effet, les modèles décris dans les fichiers CAO sont pour la plupart constitués de B-Rep et de CSG non pris en charge par les systèmes de calculs de rendu de Virtools Dev. Les matériaux et textures éventuellement décrits en temps que propriété avancée lors de la création du modèle dans le logiciel de CAO sont restitués sur les maillages calculés. En revanche, les données d'ingénierie tels que les esquisses et les propriétés liées à l'industrialisation ne seront pas restituées.

3.3.2.2.3 Réparation des erreurs de représentation du modèle original

Lors de l'étape de structuration des données, les différentes surfaces générées à partir de la B-Rep et de l'arbre CSG sont des approximations des surfaces théoriques et peuvent comporter des erreurs de positionnement de sommets liées au niveau de précision utilisé lors de la méthode de triangulation. De plus lorsqu'un modèle est décrit par morceaux à l'aide de méthodes de modélisation différentes, il peut apparaître des discontinuités ou des ouvertures dans le maillage généré. Cette dernière possibilité ne pose pas de problèmes pour le calcul de rendu visuel ou haptique mais rend les traitements de maillage plus difficiles. Pour répondre à ce problème, différents algorithmes de réparation peuvent être utilisés. Dans le cas du logiciel Deep Exploration, une option "Mesh Repair" permet de souder les surfaces constituant un même maillage ainsi que l'élimination des triangles de surface nulle ou négligeable.

3.3.2.2.4 Simplification du modèle

Afin d'utiliser l'option de LOD discret disponible dans Virtools, plusieurs versions d'un même modèle peuvent être générées. Ces différentes versions sont simplifiées en séquence par une diminution du nombre de triangles les composant. Lors de cette étape, des algorithmes de simplification tel que ceux décrit dans [110] sont utilisés pour atteindre un compromis entre la vitesse et la fidélité de l'affichage.

3.3.2.3 Les formats d'échange de modèles 3d pour la visualisation temps réel (formats 3d allégés)

Depuis quelques années, les logiciel de CAO ou de PLM permettent une exportation simplifiée des modèles 3d vers des applications interactives. Ces possibilités d'exportation ont été accompagnées de la création de formats propriétaires spécifiques comme le format JT ou 3DXml décrits en partie 2.1.3.3. Ceux-ci permettent une description exhaustive des caractéristiques techniques des modèles industriels, ce qui n'est pas le cas des formats libres, comme par exemple le format VRML, comme on peut le voir dans le tableau comparatif 3.1.

Notre choix s'est porté, après analyse et concertation avec notre partenaire industriel, sur le format JT comme format préférentiel d'importation de modèles CAO dans la simulation. En effet, il permet notamment le stockage d'un LOD discret dès l'exportation depuis le système PLM UGS ce qui rend possible

Format	Géométrie (Nombre de maillages)	Données d'industrialisation	Géométrie Exacte
3Dxml	1 (Simplifié à l'exportation)	ajoutées sur demande	non
JT	4 (LOD)	oui	oui (version 9)
VRML	1	non	non

TABLE 3.1 – Tableau comparatif des contenus sauvegardables en fonction des format d'exportation des modèles 3d

l'utilisation de modèles lourds dans la simulation sans nécessiter une connaissance approfondie du traitement de modèles pour l'utilisateur. De plus, il permet une visualisation temps réel et une utilisation des données d'industrialisation du produit dans la simulation.

3.3.3 Stratégies de conception de la simulation pour l'interaction en temps réel

3.3.3.1 Organisation en différentes scènes

Afin de garantir une interaction temps réel dans l'environnement Virtools Dev, la simulation doit être divisée en différentes "scènes". En effet, cette séparation permet de limiter le nombre de fonctionnalités accessibles par l'utilisateur, le nombre et le type d'objets affichés ainsi que leur mode de représentation. Cette décomposition permet de garantir l'interaction temps réel entre l'utilisateur et la simulation ainsi qu'une architecture modulaire permettant un ajout aisé de nouvelles fonctionnalités. Afin d'améliorer l'ergonomie, chaque scène est lié à un contexte d'utilisation des modèles choisi par l'utilisateur comme illustré en figure 3.7.

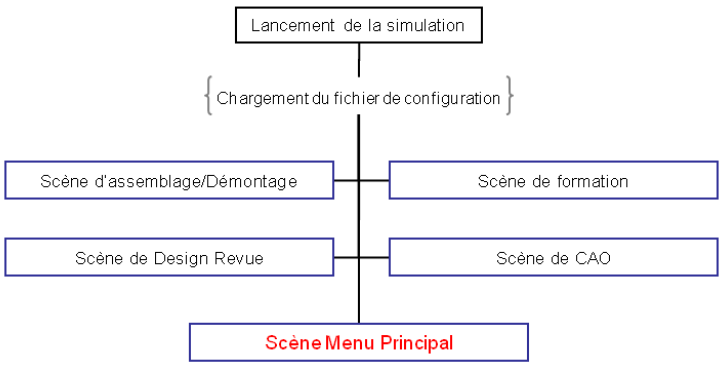


FIGURE 3.7 – Architecture de simulation : Organisation à base de scènes virtuelles

3.3.3.2 Calcul de rendu adaptatif : Level Of Detail

Malgré l'économie de charge graphique réalisée par le choix d'une architecture organisée en différentes scènes, les données issues de la CAO ne peuvent être affichées dans leur niveau de détail maximum. Nous choisissons donc d'utiliser le système de LOD discret disponible dans Virtools afin de garantir une bonne

qualité de rendu des modèles proches de l'utilisateur, c'est à dire suscitant un intérêt, et une réduction de la qualité du rendu pour les éléments plus lointains, d'un intérêt secondaire.

3.4 Implémentation : Projet VREMA

L'implémentation de ce projet est réalisée en C++ en utilisant le SDK fourni par Virtools dev. La totalité des fonctions présentées ci-après et dans le chapitre suivant sont implémentées de manière unifiée dans un projet de développement appelé VREMA (Virtuel Reality Environment for Manufacturing Applications). Ce mode de programmation présente l'avantage du point de vue du concepteur de répondre de manière unifiée aux attentes d'Alstom Transport dans le cadre de cette thèse mais également de donner une base de travail pour de futurs développements. Une fois compilé en dll (Dynamic Link Library) et directement intégré dans Virtools dev, ce projet permet d'importer sous forme de BB les différents développements réalisés et peut ainsi être intégré aux futures compositions des utilisateurs.

3.4.1 Description du Manager Principal

3.4.1.1 Conception générale

Nous nous appuyons sur la représentation "Manager - Buildings Blocks" du logiciel Virtools pour la gestion des données et des fonctions nécessaires à la création et à l'utilisation du monde virtuel. Pour répondre aux attentes, le démonstrateur sera composé d'un programme principal permettant le management général de l'application grâce à quatre modules de fonction dits « principaux » :

1. Un module d'*interface* permettant le management des périphériques : ce module permet la gestion des entrées/sorties de l'application. L'utilisateur doit pouvoir choisir ses périphériques de manipulation et de visualisation selon la configuration du système qu'il utilise et ses préférences : la personnalisation de la configuration du système est assurée par un outil de gestion des ressources géré par Virtools (VRNR) paramétré au lancement de l'application. Ce système permet de préciser au programme la configuration matérielle et donc les possibilités d'interfaçage avec l'utilisateur surtout en ce qui concerne le système de visualisation. La personnalisation des interfaces de manipulation est chargée et reconfigurable dynamiquement.
2. Un module d'*importation* de données dans l'application : ce module permet l'importation de modèles 3d Alstom issues de la CAO ou du PLM qui seront manipulées en temps que pièce ou outil dans la simulation. Différents fichiers tels que des procédures de montage ou de démontage pourront également être importés dans l'environnement. Ce module servira également au chargement des contextes de manipulation sous forme de décor.
3. Un module d'*exportation* : ce module permet de sauvegarder ou d'exporter les modèles 3d modifiés dans l'environnement de simulation. Il permettra également la création et la sauvegarde d'images ou de vidéos pour illustrer les documentations de montage ou de maintenance générées.

4. Un module de *navigation* : ce module permet à l'utilisateur de déplacer son point de vue dans l'environnement au moyen de l'interface de son choix.

Du point de vue de l'utilisateur, le manager principal sera relié à des sous-systèmes secondaires spécialisés qui seront utilisés de manière transparente pour l'utilisateur suivant les besoins spécifiques aux différentes scènes de la simulation illustrée en figure 3.7 :

- un sous système de revue de design,
- un sous système de montage,
- un sous système de maintenance,
- un sous système de formation haptique au geste technique.

Cette séparation volontaire permet d'optimiser le fonctionnement de la simulation suivant le cas d'utilisation par une limitation des données et fonctions accessibles. Cette séparation permet également de préconfigurer les comportements des objets chargés en fonction de leur cadre d'utilisation.

3.4.1.2 Centralisation des données et des fonctions

L'ensemble des données importées, capturées ou crées dans la simulation est stocké dans le manager principal par le biais une structure d'objets C++ illustrée en figure 3.8. Cette organisation permet une gestion de la hiérarchie entre les différents objets parallèle à la relation parent/enfant proposée par Virtools dev. De cette façon, les différents objets peuvent être organisés d'une part dans la structure de VREMA selon la hiérarchie "industrielle", présente dans le fichier CAO original, et selon une organisation adaptée à un contexte de manipulation variable dans le gestionnaire hiérarchique de l'environnement de simulation Virtools.

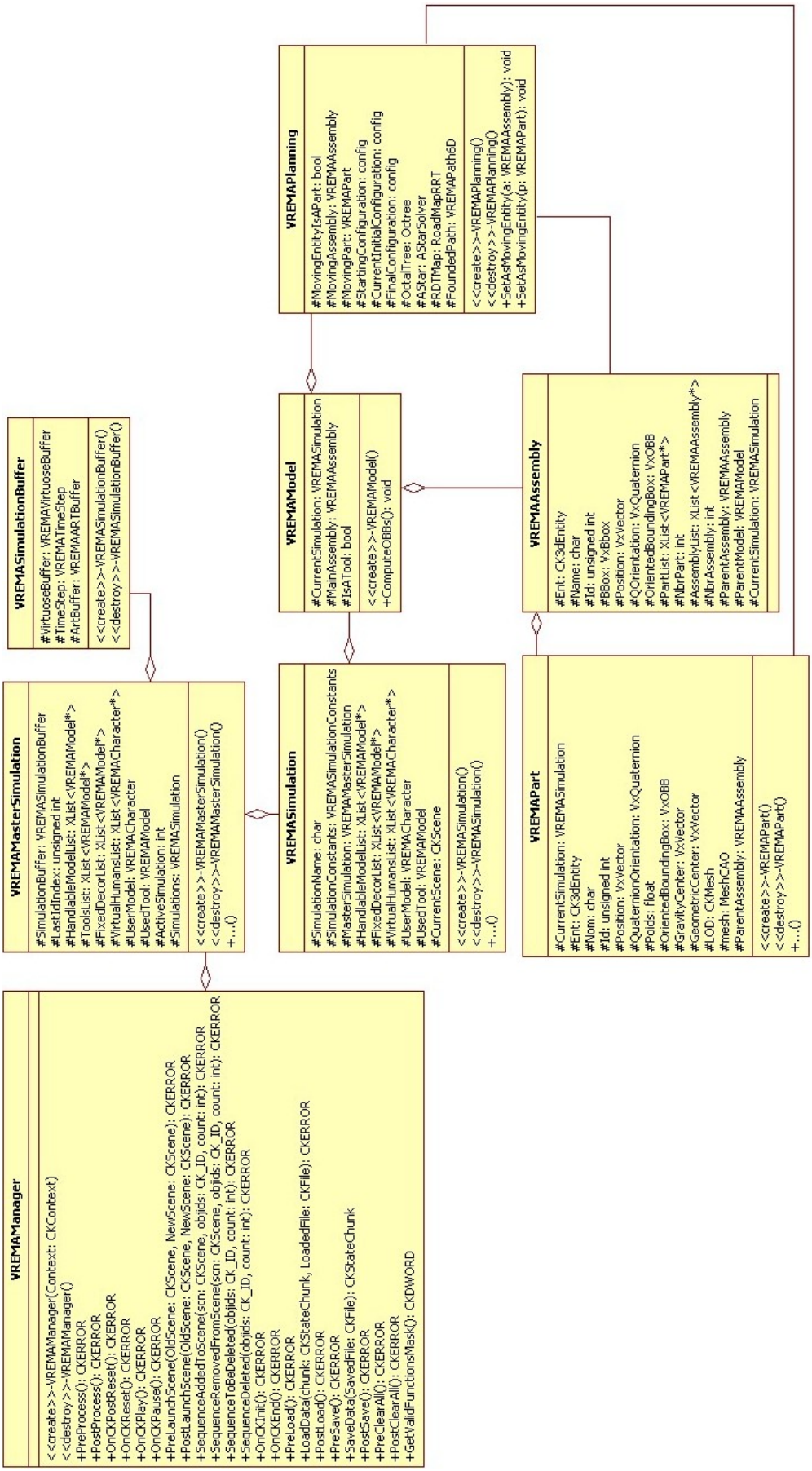


FIGURE 3.8 – Architecture centralisée des données de simulation dans le manager VREMA

3.4.2 Flux de données

3.4.2.1 Gestion des données

Les différentes données chargées et crées dans la simulation sont stockées en utilisant les attributs d'un manager instancié au début de chaque lancement de la simulation (VREMA Manager). Ce manager, au travers d'une architecture orientée objet permet d'accéder aux différentes fonctionnalités attendues par Alstom Transport. Ces fonctionnalités, implémentées dans VREMA, peuvent être appelées au niveau des scripts de la simulation au moyen de BB réalisant des opérations élémentaires telles que le chargement d'un modèle, sa sélection, sa saisie, ou encore sa destruction. Cette architecture unifiée permet un ajout aisé de nouvelles fonctionnalités et une gestion centralisée des données.

3.4.2.2 Importation de données

Au lancement de la simulation, l'environnement est vide. L'utilisateur doit donc choisir un *contexte de simulation* et l'importer dans la simulation. Ce contexte, précédemment réalisé sous Virtools Dev et sauvegardé à échelle 1 sous format NMO ou 3dxml, est une représentation de l'environnement statique conditionnant directement la manipulation des modèles comme par exemple la description géométrique de la ligne d'assemblage dans le cas d'une étude de montage. Une fois le contexte choisi, l'utilisateur peut importer différents modèles de pièces ou d'ensemble de pièces CAO. Ces modèles peuvent provenir de format d'échange 3d tels que le format JT ou 3dxml mais aussi VRML ou encore NMO.

L'importation des données est réalisée aux moyen de lecteurs spécialisés suivant le format d'importation :

- dans le cas du format NMO, VRML, ou encore 3dxml, directement intégrables dans virtools, le lecteur utilisé est la fonction "Load Object" disponible dans le SDK de Virtools Dev.
- dans le cas du format JT, un parseur externe est nécessaire afin de capturer l'ensemble des informations stockées dans le fichier. Dans ce contexte, nous utilisons le SDK "Data Kit JT Reader" réalisant une lecture et une interprétation des données contenues dans les fichiers au format JT sous la forme d'une structure en C++.

Une fois importé, chaque objet est traité et stocké dans les attributs du manager VREMA suivant son type d'utilisation dans une des trois catégories suivantes :

- "Modèle manipulable", dans ce cas, l'objet peut être saisi par l'utilisateur au moyen de l'interface dédié à la manipulation et bénéficie d'un accès à des fonctionnalités de planification de trajectoire, de lecture de données relatives à l'industrialisation et d'enregistrement de données (pour la rédaction de notices),
- "Décor", dans ce cas, l'objet chargé est considéré comme statique dans l'environnement car symbolisant le contexte de la manipulation. Il est également muni d'attributs de collision statique permettant sa prise en charge lors des opérations de manipulation haptique ou de planification de trajectoire des "modèles manipulables".
- "Outil" , dans ce dernier cas, l'objet chargé est manipulable mais ne servira qu'à des phases de validation brèves permettant le renseignement de notices de montage ou de démontage des modèles manipulables,

sans donner lieu à une rédaction de notice propre à l'outil. L'outil est alors défini comme un pièce manipulable indépendante du modèle mais servant à valider et illustrer un processus.

3.4.2.3 Exportations de données

Plusieurs types de données peuvent être exportées à l'issue de l'analyse réalisée dans la simulation :

- les modèles CAO analysés et annotés des différentes modifications effectuées (suppression d'éléments, déplacement de sous ensembles, modification de maillages ...) par des codes de couleurs et un fichier texte ;
- des procédures de montage ou de démontage, constituées de la séquence de sélection d'objets ou d'outils réalisées au cours de la simulation ainsi que les différentes trajectoires suivies, stockées sous forme de fichier texte ;
- des prises d'écran au format jpg ou des vidéos au format mpeg, directement capturées à partir d'une caméra virtuelle placée par l'utilisateur pour illustrer un process de montage ou de démontage. Ces illustrations seront par la suite ajoutées aux procédures sauvegardées afin de réaliser une documentation de manière anticipée ;
- des sauvegardes complète de scènes, comprenant les différentes entités 3d chargées dans la scène ainsi que leur configuration. Ces sauvegardes sont effectuées dans un fichier binaire.

3.4.3 Formatage des données d'entrée

3.4.3.1 Chargement de modèles

Les différents modèles chargés dans la simulation sont préparés dès leur importation afin de répondre aux principales contraintes de la simulation, à savoir, la possibilité de modification des maillages et leur visualisation en temps réel. Dans ce contexte, le format de structuration proposé par Virtools étant insuffisant, il est nécessaire de définir une structure personnalisée de maillage répondant aux contraintes spécifiées ainsi qu'un processus de traitement adapté illustré en figure 3.9.

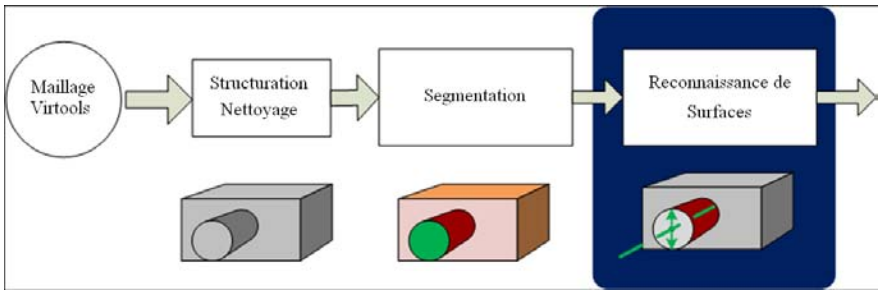


FIGURE 3.9 – Processus de chargement automatisé des modèles 3d dès leur importation dans la simulation

3.4.3.1.1 Modèles issus de la CAO

Afin d'obtenir une processus de chargement compatible avec le PLM, les modèles à évaluer seront direc-

tement importés dans la simulation au format JT. Afin de permettre cette opération, une bibliothèque de lecture de fichiers JT, fournie par la société Data Kit, est acquise par le laboratoire. Cette bibliothèque présente l'avantage de réaliser l'interprétation de modèles 3d codés par des arbres CSG et des B-Rep décrits dans le fichier JT afin de les convertir en un maillage à base de triangles. Elle permet de charger intégralement l'ensemble des données contenues dans le format JT (si elles sont décrites et exportées depuis le logiciel de CAO) y compris les données liés à l'industrialisation du produit comme illustré sur la figure 3.10. Une classe d'objet du projet VREMA (classe VREMAModel) permet ensuite d'intégrer dynamiquement dans Virtools les modèles JT décodés par Data Kit et de les utiliser dans la simulation.

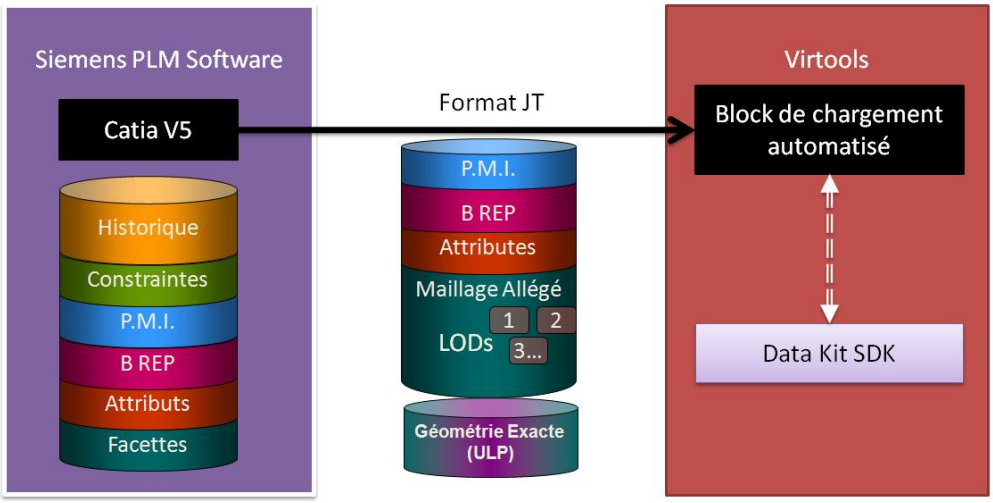


FIGURE 3.10 – Importation des modèles CAO dans une des scènes virtuelles interactives de la simulation sous format JT

3.4.3.1.2 Modèles 3d des décors

Dans certaines situations où le contexte est essentiel à l'évaluation de la qualité de conception d'un produit, comme par exemple les tests de montage ou de maintenance, l'intégration de décors appropriés est essentielle. Ces décors sont assemblés dans l'environnement Virtools Dev à partir de géométries importées depuis un logiciel de CAO, d'infographie ou encore directement depuis une bibliothèque d'objet disponible dans Virtools. Par la suite, cet ensemble d'objets décrivant le décor est exporté sous le format NMO ce qui facilite son importation et son utilisation dans la simulation principale. Lors de leur chargement, ils sont pris en compte en temps que "décors" pour limiter l'espace de travail disponible est ainsi rendre la manipulation plus réaliste.

3.4.3.2 Structuration et réparation

Lors de leur chargement dans l'environnement Virtools les maillages représentant les objets sont très faiblement structurés comme illustré en figure 3.11 et décrit par l'algorithme 3.1. Ils sont décrit par une classe

“CKMesh” ayant comme attributs une liste de sommets décrit chacun par un vecteur position et un index, et une liste de de triangles créés à partir d’un ensemble de trois indices de sommet. Une fois le maillage de l’objet 3d instancié sous forme d’objet de type CKMesh, les données qu’il contient deviennent accessibles pendant la simulation mais uniquement au travers des fonctions de recherche naïves très coûteuses en temps de calcul.

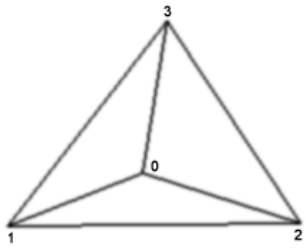


FIGURE 3.11 – Illustration d’un maillage “Exemple” dans l’environnement Virtools

Algorithme 3.1 Création du maillage “Exemple” illustré en figure 3.11

```
FONCTION Créer_Maillage_Exemple (contexte : CKContext) : CKMesh*
DÉBUT
// Création de l'instance de maillage
CKMesh * mesh = (CKMesh *) contexte→CreateObject(CKCID_MESH,"Exemple") ;
// Vertices
mesh->SetVertexCount(4) ; // Spécification de la dimension de la liste de vertices
VxVector v ;
v.Set(0,0.0,0) ; mesh→SetVertexPosition(0,&v) ;
v.Set(1.0,0.0,0) ; mesh→SetVertexPosition(1,&v) ;
v.Set(1.0,0.0,1.0) ; mesh→SetVertexPosition(2,&v) ;
v.Set(0.5,1.0,0.5) ; mesh→SetVertexPosition(3,&v) ;
// Faces
mesh→SetFacesCount(4) ; // Spécification de la dimension de la liste de faces
mesh→SetFaceVertexIndex(0,0,1,2) ; // face gauche
mesh→SetFaceVertexIndex(1,3,2,1) ; // face avant
mesh→>SetFaceVertexIndex(2,3,0,2) ; // face droite
mesh→SetFaceVertexIndex(3,3,1,0) ; // face gauche
RETOURNE mesh
FIN
```

Les modèles doivent posséder des maillages fermés et structurés pour garantir une interaction en temps réel dans les différents cas d’utilisation prévus, notamment lors de l’exécution de méthodes d’exploration de maillage utilisés lors de déformations de celui ci, ainsi que lors de l’utilisation d’algorithmes de détection de collision. Dans ce but, nous implémentons une classe d’objets en C++ permettant de décrire, de manière plus appropriée à notre cas, le maillage d’un objet 3d. Une classe de maillage (Mesh CAO décrite sur la figure 3.12) incluse dans le projet VREMA permet de stocker un maillage avec une structuration facilitant l’exploration.

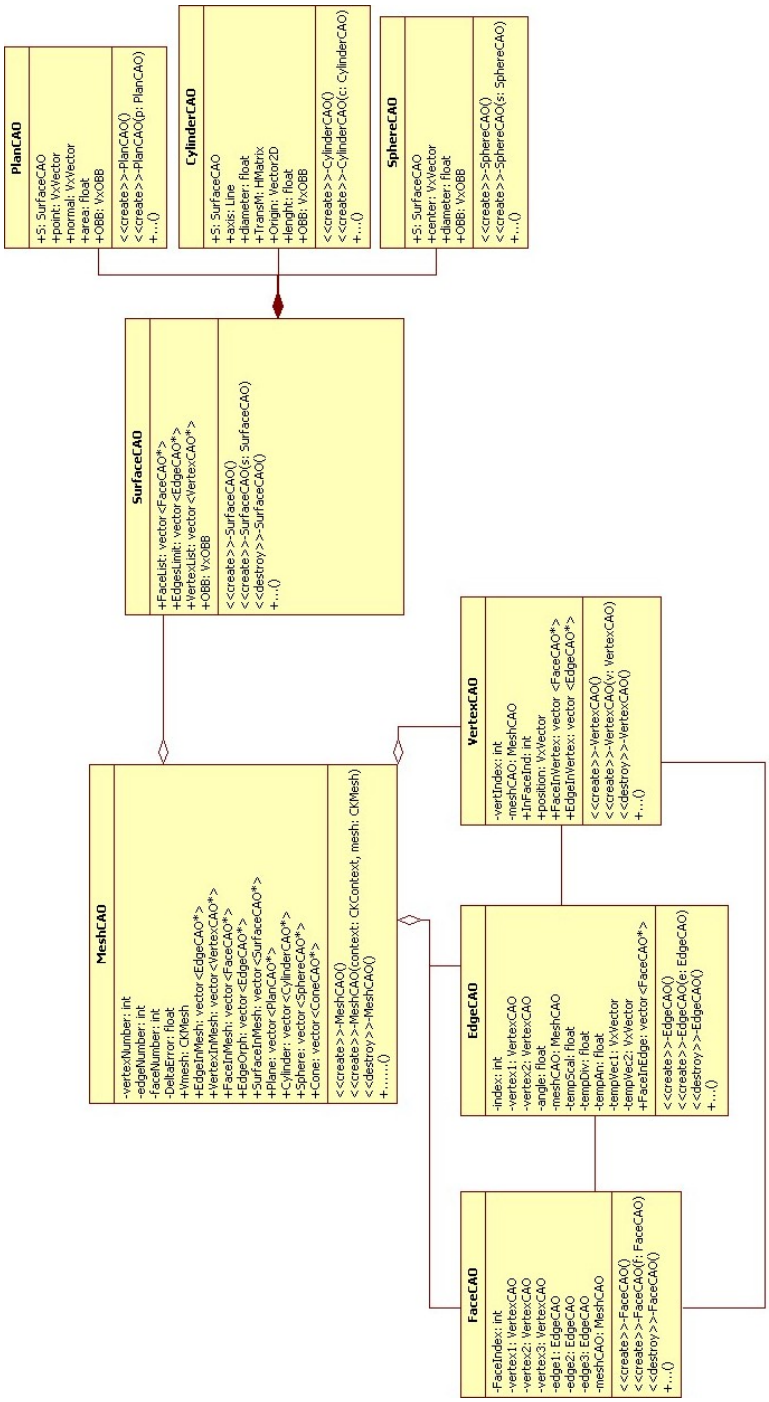


FIGURE 3.12 – Schéma de la classe Mesh CAO

Après cette structuration, l'algorithme 3.2 de "nettoyage" de maillage est exécuté afin que l'ensemble des maillages des objets réponde aux même contraintes. En effet, afin de faciliter l'exécution de méthodes d'exploration, de calcul et de modification, les maillages sont nécessairement fermés et ne comportent pas de triangles d'aire nulle.

Algorithme 3.2 Algorithme de “Nettoyage” de maillage

```
FONCTION Nettoyage_de_maillage ( mesh : MeshCAO ) : MeshCAO
DÉBUT
POUR itérateur<Facette> itf DE mesh.ListeFacette.Debut A mesh.ListeFacette.Fin
 SI (itf.Aire = 0)  mesh.Retirer ( itf )
Liste<edge> leo ← mesh.Liste_edges_orphelines () ) ; // Repérage des Edges n'appartenant qu'à une seul facette
Liste<edge> l2
TAN QUE (leo non vide) FAIRE
 Liste<edge> lf ←extraire_boucle ( leo )
 SI ( lf.taille = 2 )
  mesh.soudure_edges_simple ( lf )
 SINON
  SI (lf.taille > 2)
   mesh.soudure_edges_en_cycle ( lf )
  SINON
   l2.ajouter( lf )
SI (l2 vide)
 RETOURNER mesh //Maillage fermé
SINON
 RETOURNER 'Erreur' //Maillage ouvert non résolu
FIN
```

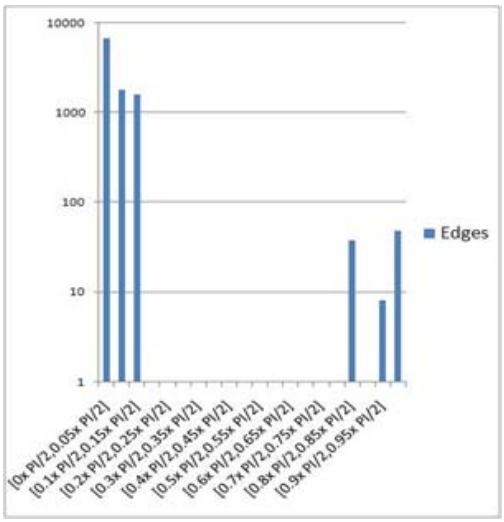
3.4.3.3 Segmentation

Cette étape permet de séparer en différentes surfaces un même maillage afin de repérer les différentes surfaces caractéristiques qui le constituent. Lors de cette étape, nous utilisons l'algorithme 3.3 inspiré de méthodes de segmentation par expansion de régions ainsi que d'analyse de courbure décrit dans [111, 112, 113]. Celui ci permet de découper un maillage fermé à base de triangles par la définition d'un angle maximum entre deux triangles caractérisant la frontière entre deux surfaces.

Dans une première étape, une analyse du modèle à segmenter est réalisée afin de capturer un classement des angles entre facettes comme illustré en figure 3.13. Cette analyse permet d'estimer le nombre d'itérations entre segmentation et identification nécessaire à la décomposition du modèle ainsi que les différentes valeurs de limite d'angle à spécifier.

Ensuite, plusieurs étapes de segmentation utilisant des expansions de régions sont réalisées successivement en utilisant de manière décroissante les limitations de la valeur maximale de l'angle entre surfaces déterminées à l'étape précédente.

L'algorithme 3.3 permet une segmentation efficace de modèles industriels “classiques” issues de la CAO comme illustré en figure 3.14.



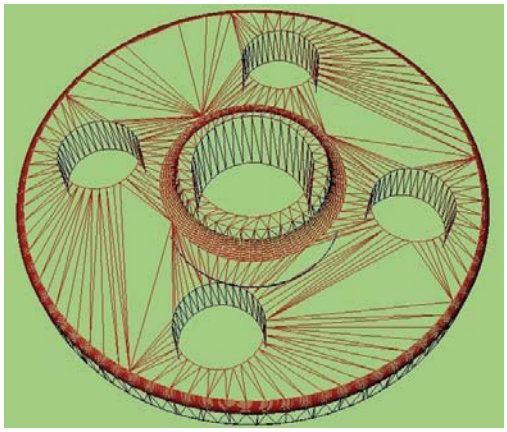


FIGURE 3.13 – Classement des angles entre facettes dans le maillage

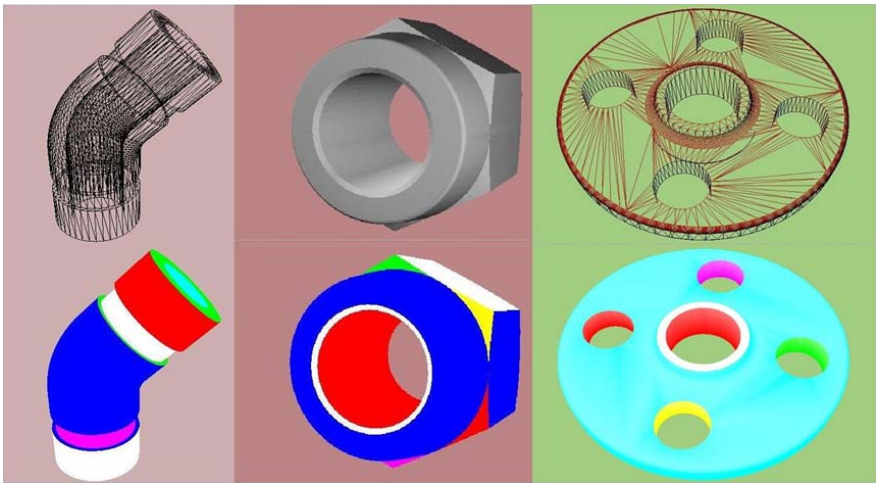


FIGURE 3.14 – Exemple de modèles 3d après la phase de segmentation

3.4.3.4 Reconnaissance de surfaces canoniques

Dans les pièces réalisées dans les logiciels de CAO, les surfaces sont très souvent composées de portions de surfaces canoniques simples telles que des plans, des cylindres, des sphères, des cônes ou encore des tores. Afin de faciliter les opérations de modification de maillage et d'analyse des modèles, nous implémentons des algorithmes de reconnaissance de surfaces afin de s'assurer de découper intelligemment le modèle. Dans ce sens, nous implémentons l'algorithme 3.4 permettant de caractériser une surface issue de l'opération de segmentation et d'en calculer les constantes. Les caractéristiques de chaque surface à identifier sont estimées à partir de calculs géométriques simples sur un nombre limité de triangles, détaillés en annexe 5.2, puis validés sur l'ensemble des sommets de la surface. Dans le cas où l'algorithme 3.4 ne permet pas de trouver une caractéristique de surface simple, l'algorithme de segmentation 3.3 est exécuté de nouveau sur la surface avec un paramètre de limite d'angle plus faible afin de la re-décomposer comme illustré par la figure 3.15.

Algorithme 3.3 Algorithme de Segmentation par expansion de régions

```
FONCTION Segmentation ( m : MeshCAO , limit : Réel ) : MeshCAO
DEBUT
Liste<Facette> l ← m.liste_de_facettes
Liste <Edge> perimètre
FAIRE
  Surface S ← nouvelle Surface
  Facette f1 ← l[0] ;
  Liste<Facette> Buffer
  Buffer.Ajouter ( f1 )
  S.Ajout ( f1 )
  l.retirer( f1 )
  FAIRE
    Facette ft ←Buffer[0]
    POUR CHAQUE Facette voisin_de_ft
      SI(angle ( ft, voisin_de_ft ) < limit)
        Buffer.Ajout ( voisin_de_ft )
        S.Liste_facettes.Ajout ( voisin_de_ft )
        l.Retirer (voisin_de_ft )
      SINON S.Périmètre.Ajout ( edge( ft, voisin_de_ft ) )
    Buffer.Retirer ( ft )
   TAN QUE (Buffer non vide)
m.Ajout_surface( S )
TAN QUE (l non vide) ;
RETOURNE m
FIN
```

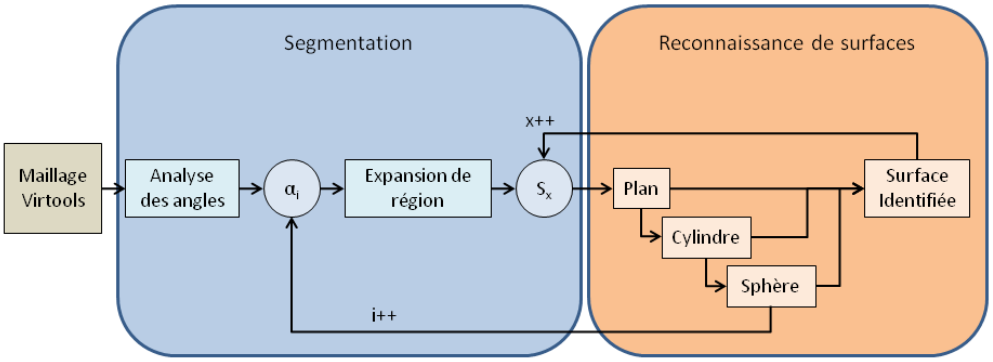


FIGURE 3.15 – Itérations entre Segmentation et Identification

3.4.3.5 Perspective ouvertes

L'architecture définie dans cette partie permet une utilisation simplifiée des technologies de la réalité virtuelle dans le cadre d'applications industrielles. La structure de données présentée permet d'envisager de nouveaux développements notamment dans le domaine de la rectification des géométries visualisées ainsi

Algorithme 3.4 Algorithme de caractérisation de surface canonique simple

```
FONCTION (ls : Liste<Facette>) : Face
DÉBUT
Facette f0 ← ls.debut()
Plan p(f0) //Création d'un plan à partir de la première facette
Face f ←p
POUR itérateur<Facette> it DE ls.debut()+1 A ls.fin()
  SI (p.Comprend ( it ) = false ) ALORS
  Facette f1 ←it
  Cylindre c(LS[0],f1)
  f ←c
  SI (c.Comprend ( it ) = false ) ALORS
    Facette f2 ←it ;
    Sphere s(LS[0],f1)
    f ←s
    //On peut Ajouter ici une recherche de cône puis de tore
    //Suivant la même logique
  SINON
    f ←0
RETOURNE f
FIN
```

que la conception de détecteurs de collisions plus performants s'appuyant sur des définitions exactes des surfaces.

3.4.4 Flux de consignes

Afin de permettre l'interaction entre l'utilisateur et le monde virtuel, les différentes consignes émanant des périphériques mis à disposition doivent être interprétées par la simulation. Ces données sont collectées et mises à disposition au travers de différents conteneurs du sous système d'interface de VREMA. Néanmoins, certains périphériques comme par exemple l'interface haptique sont gérés de manière indépendante de la simulation Virtools. Il imposent au concepteur de la simulation Virtools de prendre en compte leur mode de fonctionnement et leurs limitations dans la programmation, suivant les besoins énoncés par l'utilisateur.

3.4.4.1 Le problème de la manipulation haptique

Le plug in IPP (v1.0), utilisé pour la manipulation haptique dans l'environnement Virtools Dev, présente un certain nombre d'inconvénients imposant des limitations dans la liberté de manipulation des modèles :

- il existe une limitation du nombre d'objets "physicalisables",
- le réglage de la précision de définition du contact (erreur de corde ou SAG) est réalisée à l'initialisation de la manipulation haptique,
- pour des questions de stabilité, la vitesse de manipulation est limitée en fonction de la précision de manipulation choisie.

3.4.4.1.1 Limitation du nombre d'objets physicalisable

Cette limitation est imposée par le détecteur de collision VPS utilisé par IPP afin de garantir un calcul de

détection de collision entre objets suffisamment rapide pour assurer une perception haptique réaliste. Compte tenu du nombre d'objets manipulables qui est a priori non déterminé, la gestion du calcul de détection de collision pour l'interface haptique est gérée par l'utilisateur. Il doit pour cela sélectionner un maximum de 9 objets à manipuler, le dixième étant l'ensemble des autres objets et décors pris en compte pour le calcul de retour d'effort.

Lorsque l'utilisateur souhaite changer de phase d'assemblage ou manipuler un objet non contenu dans le groupe initialement choisi, le manager de détection de collision d'IPP est réinitialisé avec un groupe d'objets à physicaliser différent. L'étape de physicalisation des objets peut parfois, suivant leur nombre, leur complexité, ainsi que la précision spécifiée, consommer une grande quantité de mémoire ainsi qu'une durée importante de chargement pendant laquelle la simulation est inutilisable.

3.4.4.1.2 Réglage de la précision

Le réglage de la précision du contact (SAG ou erreur de corde) est également un facteur clef car il est utilisé, lors de l'étape de physicalisation, pour régler la définition des cartographies réalisées par l'algorithme VPS notamment lors du calcul des VoxMaps. Cette variable conditionne d'une part le temps nécessaire aux calculs d'initialisation mais aussi la qualité du rendu haptique pendant la manipulation.

Dans le cas d'une initialisation de cette variable trop importante, le rendu haptique obtenu n'est pas représentatif du contact observé. En revanche, lors de son initialisation à une valeur trop faible, le temps de calcul entre chaque rafraîchissement (au sens haptique) devient très important. Le programme I.P.P. limite alors la vitesse de déplacement dans la scène pour garantir la stabilité du retour d'effort calculé ce qui rend la manipulation des objets difficile voire impossible.

Pour trouver le bon réglage de ce paramètre permettant à la fois le respect de la contrainte d'interaction en temps réel ainsi qu'une qualité de rendu haptique satisfaisant l'utilisateur, une série d'expériences de manipulation à l'échelle 1 sont menées sur une gamme de réglages potentiels.

3.4.4.2 Capture des consignes ART

Les données capturées par le PC supportant le logiciel DTrack (illustré en figure 3.1), sont transmises par trame réseau au PC hébergeant la simulation. Ces différentes données sont reçues par Virtools en utilisant le protocole VRPN et interprétées de deux manière différentes suivant leur nature.

3.4.4.2.1 Position et orientation des marqueurs

La position et l'orientation des différents corps disponibles sont reçues en utilisant les BB de communication VRPN disponibles dans le VR Pack. Cette manipulation permet de les rendre utilisables sous forme de variables dans l'environnement de programmation de Virtools. Un script de comportement permet d'asservir la position de repères 3d sur la position et l'orientation des corps capturés afin d'utiliser simplement ces données dans l'environnement virtuel. De plus, un BB implémenté dans la bibliothèque VREMA permet par la suite d'importer ces valeurs pour qu'elles soient également disponibles dans la structure de données.

3.4.4.2.2 Signaux du flystick

Dans le cas du flystick (figure 3.16), les données transmises ne se limitent pas à la capture de la position et de l'orientation mais intègrent également l'état des différents boutons et joysticks qu'il comprend. Dans ce cas, des blocs spécifiques du VR Pack permettent de transcrire dans la simulation des signaux d'activation correspondant aux appuis sur les différents boutons ainsi que des états binaires pour les boutons ou des pourcentages pour le ou les joysticks.

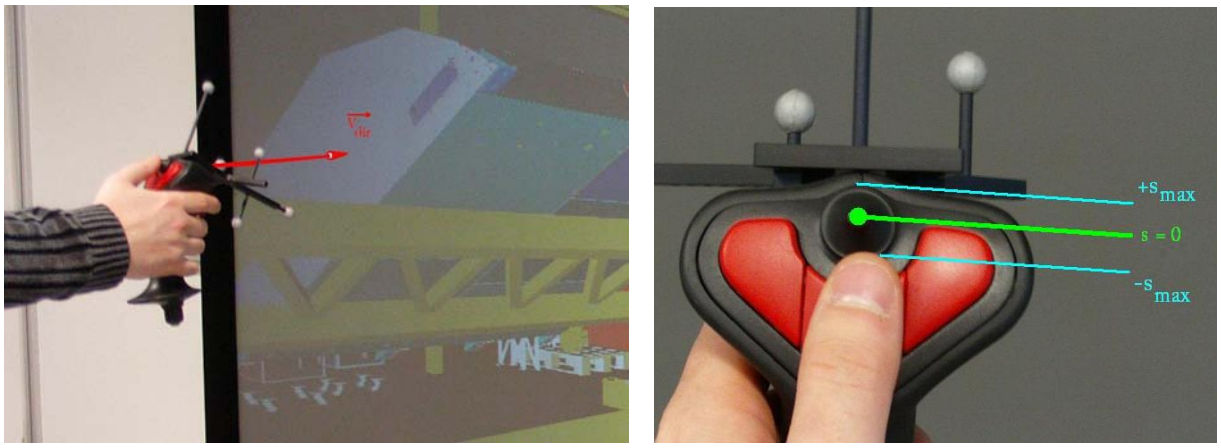
3.4.4.3 Commande de navigation à partir du flux de consignes

Afin de rendre l'interaction entre l'utilisateur et le système de visualisation la plus naturelle possible, nous asservissons la caméra principale de la scène sur le point de vue de l'utilisateur calculée d'après une mesure de la position et de l'orientation du marqueur des lunettes qu'il porte. Les données de capture étant exprimées par rapport à un repère de référence fixe noté R_{ART}, il est nécessaire de réaliser des mesures du système physique. Afin de paramétrer la configuration de l'affichage stéréoscopique, il est nécessaire de connaître les dimensions de l'écran utilisé (largeur et hauteur) ainsi que la position de celui ci par rapport à R_{ART}. Ces dimensions permettent de définir le plan de projection de la vue stéréoscopique.

L'utilisateur peut dans un premier temps faire évoluer son point de vue par un déplacement physique de sa position par rapport à l'écran. Néanmoins, étant donné les limitations dues aux dimensions du système physique, ce mode d'interaction est privilégié pour l'*inspection* d'un modèle 3d. Pour couvrir de plus grandes distances ou pour changer radicalement de point de vue, il est nécessaire de déplacer la représentation virtuelle de R_{ART} communément dénommée repère de navigation R_{nav} dans la scène virtuelle.

3.4.4.3.1 Navigation par manipulation du repère de navigation Le mode le plus simple de navigation consiste en un déplacement manuel de R_{nav} de manière analogue à un modèle 3d manipulable. L'utilisateur peut alors par mouvements successifs choisir une position et une orientation de la modélisation virtuelle de son système de visualisation par rapport aux éléments qu'il souhaite observer. Ce mode s'avère très efficace pour le choix d'un nouveau point de vue lors de l'inspection d'un modèle mais peu approprié pour le déplacement sur de grande distances dans le monde virtuel.

3.4.4.3.2 Navigation sur un vecteur capturé Un deuxième mode de navigation consiste en la capture d'un vecteur directeur $\overrightarrow{V_{dir}}$ et d'une composante de vitesse de déplacement s. Généralement, ce mode d'interaction est utilisé par le biais d'un flystick qui permet de regrouper d'une part l'indication de direction $\overrightarrow{V_{dir}}$ et d'autre part, par l'utilisation du joystick, la définition de la vitesse de déplacement. Une fois ces informations capturées, la commande de navigation en position $[\dot{x}\,\dot{y}\,\dot{z}]$ peut être synthétisée directement comme illustré par la figure 3.16.



(a) Définition du vecteur de navigation (b) Capture de la composante de vitesse

FIGURE 3.16 – Navigation sur un vecteur capturé correspondant à la configuration du flystick

3.4.4.3.3 Navigation par capture de mouvements Dans le cadre d'un déplacement sur un plan, la commande de navigation peut être réduite au calcul de 3 composantes $\left[\dot{x}\,\dot{z}\,\dot{\theta}_y\right]$. Dans ce cas, la commande de navigation peut également être composée directement par capture des mouvements (cf. Figure 3.17) :

$$\begin{cases} \dot{x} = k_x \times \Delta x \\ \dot{z} = k_z \times \Delta z \\ \dot{\theta}_y = k_y \times \Delta \theta_y \end{cases}$$

Ce mode de commande s'avère très intuitif pour la navigation simple dans des mondes virtuels de grande taille comme des représentations virtuelles de lignes de productions à échelle 1.

3.5 Réalisations pratiques

3.5.1 Visualisation de modèles 3D et analyse de la géométrie

Grâce à l'architecture décrite précédemment et notamment au système de LOD discret disponible dans Virtools, la plateforme du laboratoire nous permet de visualiser et de naviguer en temps réel dans des modèles de plus de 1 500 000 de triangles. Cette capacité de visualisation permet de répondre aux demandes de visualisation énoncées par Alstom Transport (cf. Figures 3.18, 3.19 et 3.20.

Cependant, en pratique, ces modèles sont pour l'instant créés sous Catia V4, une étape de conversion sous Catia V5 et de préparation manuelle reste nécessaire afin qu'ils puissent être chargés dans la simulation selon le processus prévu dans le cahier des charges.

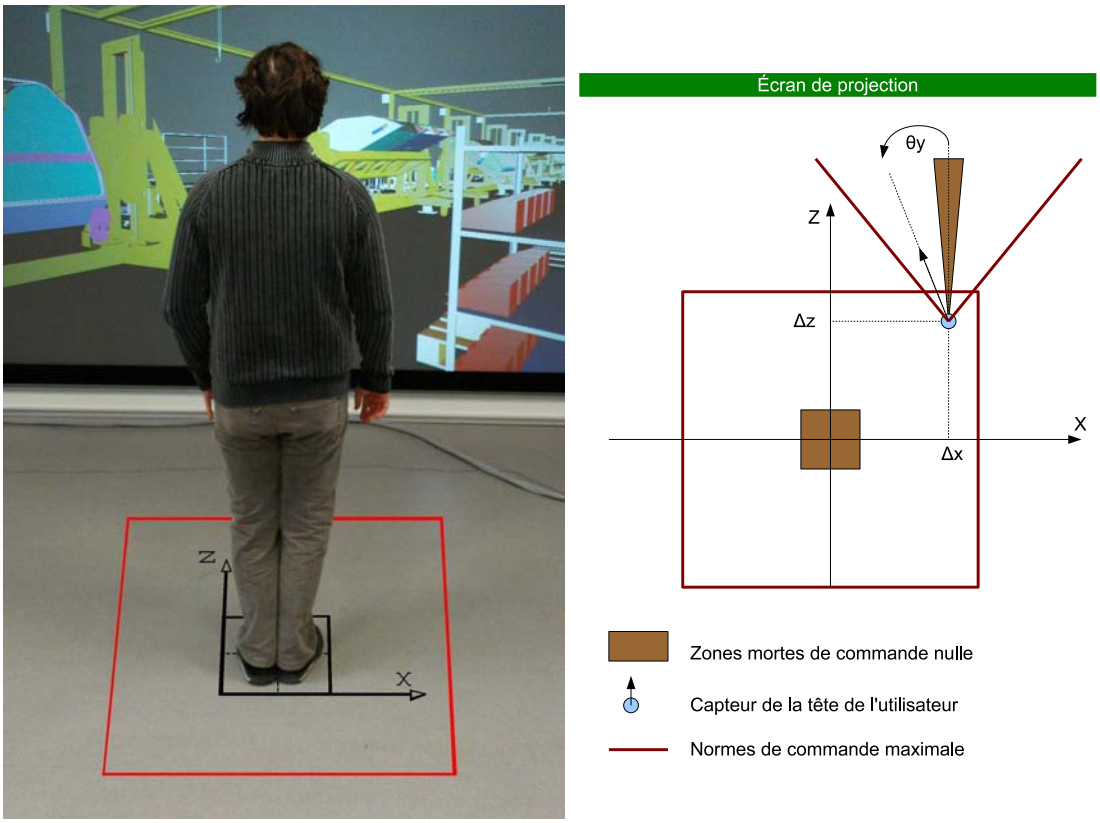


FIGURE 3.17 – Navigation par capture de mouvement

3.5.2 Test de montage et de démontage par manipulation haptique

A partir d'un chargement de modèles, réalisés initialement sous Catia V5, les procédures de montage et de démontage de convertisseurs de puissance peuvent être établies puis testées en utilisant une scène d'interaction visuo haptique comme illustré en figure 3.21. Cependant, du fait de l'utilisation d'IPP basé sur le détecteur de collision discret VPS, certaines pièces manipulées ne peuvent être placées dans leur position finale. Dans ce cas, l'utilisateur doit s'en approcher en manipulant la pièce grâce à l'interface haptique puis, lorsque celle ci est suffisamment proche, elle est lâchée automatiquement. Dès lors, la finalisation de son insertion est réalisée en utilisant une détection de collision plus précise, ce qui permet sa mise en position finale.

Dans certains cas d'insertions fortement contraintes ou difficilement résolvables, l'utilisateur ne peut rejoindre une configuration suffisamment proche de la configuration finale de la pièce manipulée afin de valider une trajectoire. Il peut, dans ce cas, faire appel à des fonctionnalités de planification interactive décrites dans le chapitre 4.

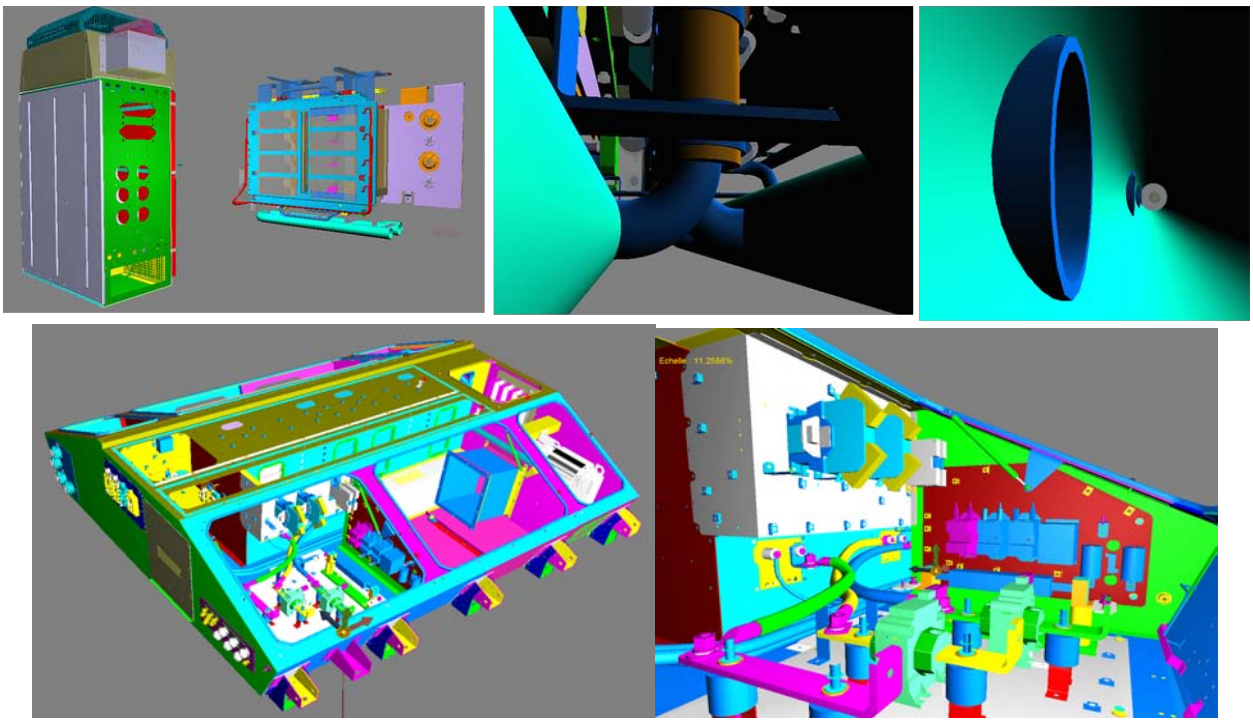


FIGURE 3.18 – Visualisation de sous ensembles issus de la CAO lors de revues de design

3.6 Conclusion

Dans ce chapitre, nous avons décrit la structure de notre plateforme expérimentale et les stratégies développées pour mettre en oeuvre les flux de données et de consignes. Un accent particulier à été mis sur la problématique de l'importation et du traitement des modèles 3d issus de la CAO dans les environnements virtuels pour la réalisation de taches de visualisation, d'analyse et de manipulation haptique d'une part, et sur les stratégies de génération de flux de consigne d'autre part. Ces différents traitements et fonctions nécessaires à la visualisation et la manipulation de modèles 3d industriels ont été intégrés dans le projet VREMA.

Celui ci est basé sur une architecture orientée utilisateur qui induit une séparation en scènes "métier". Ceci permet de limiter simplement l'accès aux différentes fonctions et périphériques. Cette approche rend la programmation de nouveaux comportements plus accessible au concepteur et l'utilisation plus familière aux différents utilisateurs. Cette structure a permi de réaliser de nombreuses démonstrations permettant de mettre en évidence les capacités de la plateforme expérimentale acquise par le laboratoire en matière de visualisation de maquettes numériques pour la revue de projet ainsi que de manipulation haptique pour la conception de scénarii de montage et de démontage.

La structure du projet VREMA permet d'envisager de nombreuses extensions de manière modulaire et favorise la réutilisabilité des fonctions développées. Elle sert notamment de base d'implémentation aux travaux developpés au chapitre suivant dans le domaine de l'intégration de techniques de planification en situation immersive.

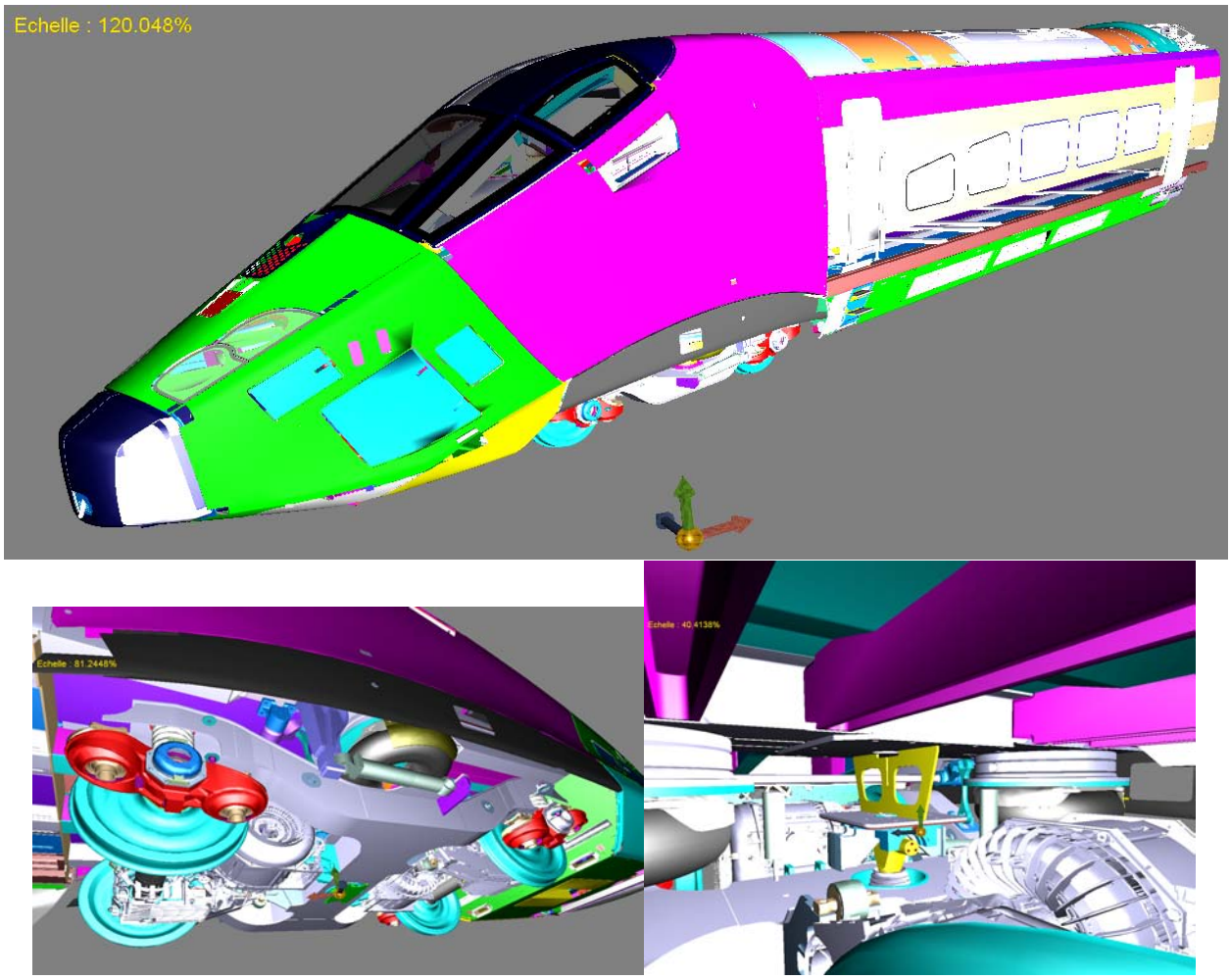


FIGURE 3.19 – Navigation à l'intérieur d'un modèle CAO lourds (AGV)

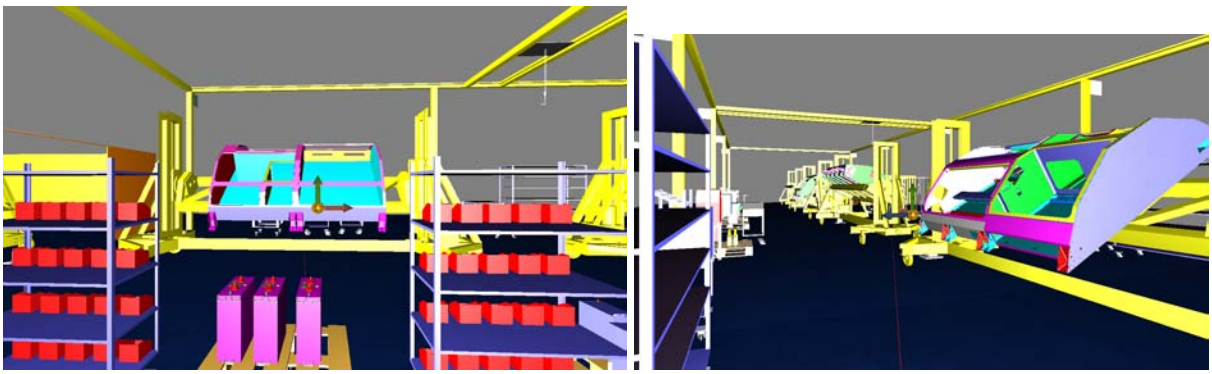


FIGURE 3.20 – Visualisation d'une ligne d'assemblage de coffres de traction

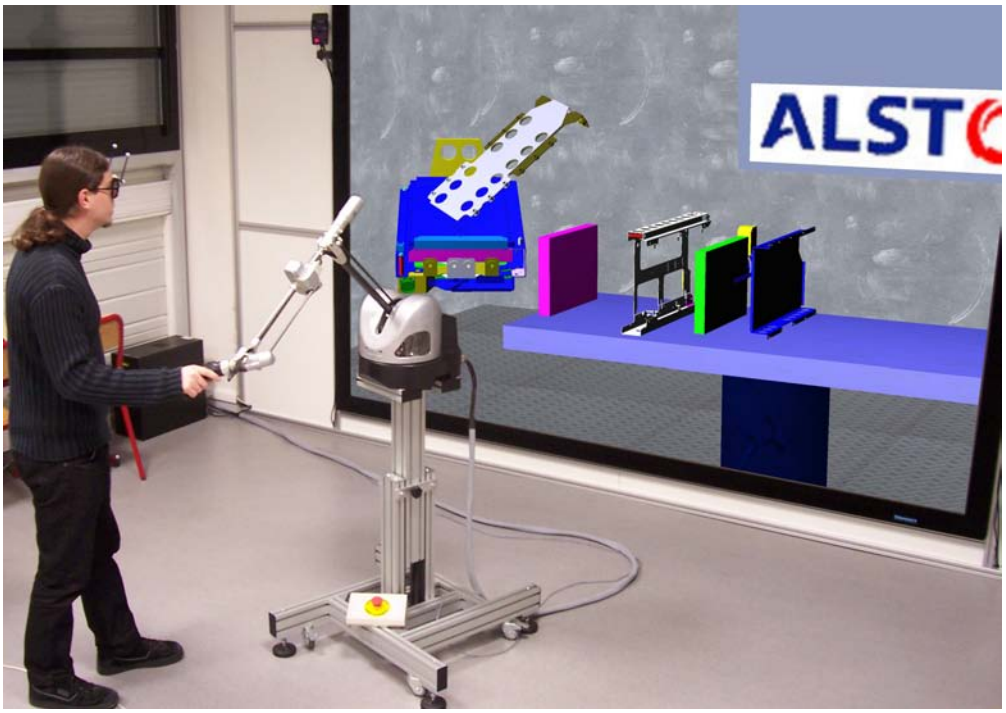


FIGURE 3.21 – Manipulation haptique de composants d'un convertisseur de puissance

Chapitre 4

Planification interactive de trajectoire

Sommaire

4.1 Introduction et motivations

4.1.1 Introduction et problématique

Dans la simulation présentée en fin de chapitre 3, un utilisateur humain est amené à tester des séquences de montage et de démontage de pièces mécaniques en environnement virtuel. Il dispose pour cela d'une interface haptique lui permettant de manipuler une pièce mécanique tout en ressentant les efforts résultant des contacts avec le reste de l'environnement et d'un système de visualisation immersive lui procurant une vision réaliste du problème à résoudre. Lors de chaque phase d'un assemblage, le cerveau de l'utilisateur est

amené à résoudre un problème qui peut être assimilé à un problème de planification de la trajectoire d'un corps rigide mobile dans un espace comportant des obstacles fixes.

Nous avons remarqué, au cours de la mise en place d'un tel système, que dans certains cas de scènes complexes, l'utilisateur peut ne pas aisément trouver de chemin sans collision et avoir besoin d'aide :

Dans un cas classique d'assemblage mécanique, deux éléments sont juxtaposés et liés par une ou plusieurs liaisons complètes. Dans ce contexte, l'utilisateur est amené à manipuler séquentiellement chaque pièce à assembler de manière à l'amener dans une configuration particulière permettant l'établissement des liaisons requises pour son maintien. Lors de l'approche de la configuration requise, la pièce manipulée est très souvent au contact d'autres pièces assimilables à des obstacles. De plus, dans le cas de pièces insérées, le guidage vers la configuration finale est réalisée grâce à un suivi du contact contre l'obstacle. L'utilisation de détecteurs de collision discrets, comme VPS [59], avec des interfaces haptiques ne permet pas à l'utilisateur de déplacer les pièces manipulées dans des configurations où elles sont en contact réel avec les obstacles. Etant donné que les configurations proches de la configuration finale sont souvent détectées comme étant en état de collision leur ralliement est donc interdit par la procédure de calcul de retour d'effort précitée.

Dans le but d'assister l'utilisateur lors de ces opérations de montage et de démontage et d'améliorer l'ergonomie de manipulation, il est utile de lui fournir des informations issues d'un planificateur automatique de mouvements.

4.1.2 Études préliminaires sur la planification de mouvements

La première partie de ce travail est le choix du type et de l'architecture du planificateur de trajectoire. Nous avons décrit dans [114] que dans une application de réalité virtuelle, le maintien d'une interaction temps réel entre l'utilisateur et le monde virtuel est primordial, ce qui induit une contrainte forte au niveau des programmes utilisés. De plus, dans le contexte de la manipulation de composants, la définition d'une solution optimale d'un point de vue d'un planificateur n'est pas nécessairement souhaitable d'un point de vue du manipulateur. En effet, l'utilisateur ne souhaite pas forcément faire appel à un planificateur dans des cas où son immersion lui permet de résoudre simplement son problème, mais il peut en revanche solliciter une aide dans des cas précis et pouvoir agir sur la proposition de trajectoire qui lui est présentée.

En prenant en compte cette contrainte de "temps réel" dans le choix du planificateur, nous avons conclu que l'utilisation d'un planificateur de trajectoire « classique » analogue à ceux présentés en partie 2.2 ne pouvait répondre à nos besoins :

- Les planificateurs globaux consomment beaucoup de ressources et, bien que très efficaces théoriquement, sont en pratique trop lents pour une utilisation dans des applications en temps réel.
- Les planificateurs locaux ne sont pas satisfaisants dans un espace de dimension 6 car ils sont confrontés à des problèmes liés aux minima locaux.
- Les planificateurs probabilistes sont peu fiables car, convergeant de manière probabiliste, ils ne peuvent garantir un résultat en temps fini. De plus, dans un contexte où l'espace de travail comporte d'importantes zones libres, ce type de planificateur peut consommer beaucoup de temps de calcul avant de converger.

Dans notre contexte, l'immersion de l'utilisateur dans une scène virtuelle lui donne un point de vue général ce qui lui permet généralement de trouver la meilleure façon d'aborder le problème comme expliqué dans [100] et ainsi de focaliser le planificateur sur un espace de recherche moins étendu, accélérant ainsi sa vitesse de convergence et influençant fortement le résultat final. En ce sens, nous pouvons dire que le cerveau humain et un planificateur automatique de trajectoire peuvent coopérer de manière à réduire le temps nécessaire à l'obtention d'un chemin solution. Dans cette optique, nous avons décidé d'implémenter une solution qui permet à un planificateur et à un utilisateur de dialoguer en temps réel et de s'influencer mutuellement.

Le développement d'un tel planificateur capable d'interagir avec un utilisateur humain diffère quelque peu de celui d'un planificateur classique de par les contraintes fortes imposées par son utilisation en simulation interactive.

Si on prend en compte le nombre de dimensions ainsi que la nature irrégulière de l'environnement, les planificateurs de type probabiliste semblent les plus adaptés. Ils comportent néanmoins une composante aléatoire donnant des résultats peu fiables au niveau du temps de calcul nécessaire à la convergence vers un chemin solution. Ce défaut caractéristique des méthodes probabilistes est connu pour être accentué dans le cas d'espaces de travail comportant des passages étroits ou de très grandes portions d'espace libre. Or, c'est typiquement le cas rencontré lors de tâches de montage (ou de démontage). En effet, on y observe généralement une configuration initiale (finale) entourée d'un très grand espace libre d'obstacle et une configuration finale (initiale) fortement contrainte. Pour remédier à ce problème, certaines méthodes introduisent un biais dans la méthode probabiliste à l'aide d'études partielles locales ou globales [89, 90].

4.2 Architecture de planification interactive

La complexité des opérations de montage ou démontage dans un environnement virtuel peut varier à la fois par rapport à l'expérience de l'utilisateur mais aussi par rapport à l'opération demandée. Dans certains cas très simples, la mise en oeuvre d'un système de guidage très simple basé sur une approche locale peut suffire à guider l'utilisateur. Par contre, dans les cas complexes de passages étroits difficiles à appréhender, une mise en oeuvre plus lourde peut être nécessaire car l'utilisateur n'arrive pas à trouver de solution par lui même.

Partant de ce principe, une architecture de planification interactive ne doit pas reposer sur une méthode unique mais sur une gamme de méthodes afin que l'utilisateur puisse faire un choix judicieux pour trouver un compromis entre son gain en charge cognitive et le surcroît de charge pour le planificateur. L'utilisateur peut alors choisir un mode de guidage plus ou moins fort, en s'appuyant sur son interprétation du degré de complexité du problème et sur sa capacité à le résoudre, comme illustré par la figure 4.1.

Une architecture de planification interactive doit donc prévoir des flux d'informations entre l'utilisateur et le planificateur. Ce flux d'information doit être relayé par les interfaces comportementales de la plateforme

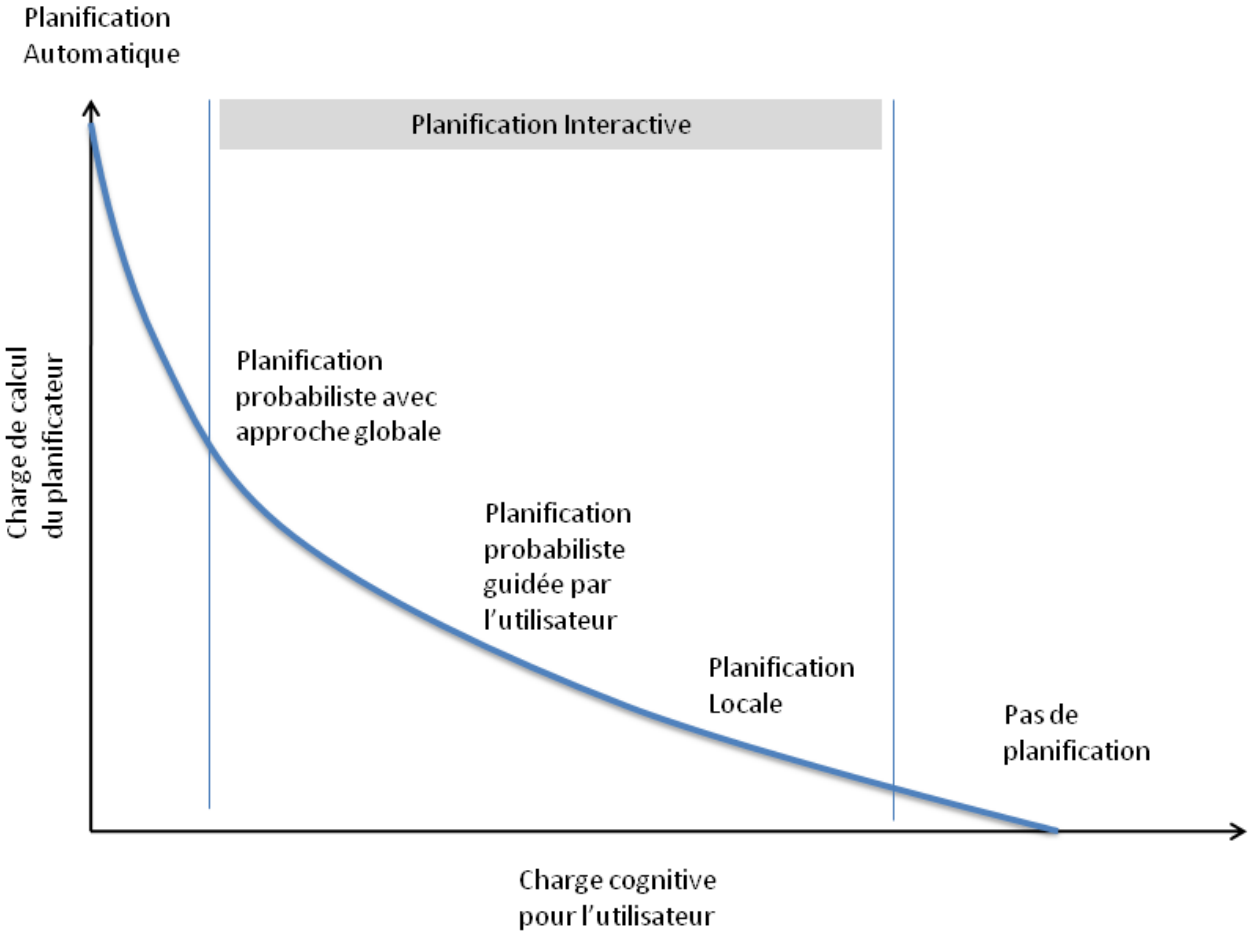


FIGURE 4.1 – Schéma de principe de l'évolution de la charge de calcul et du type de planificateur requis par rapport à la charge cognitive demandée à l'utilisateur pour la résolution interactive d'un problème donné de planification de trajectoire.

de réalité virtuelle. Dans notre cas, l'interface haptique constitue un périphérique privilégié du fait de sa capacité à transmettre simultanément des flux d'entrée et de sortie entre l'utilisateur et la simulation. Diverses informations comme la configuration courante de l'entité manipulée, son évolution dans le temps, les couples et forces résistants imposés par l'utilisateur peuvent fournir des informations importantes qui traduisent l'attitude de l'utilisateur et qui peuvent être utilisées pour renseigner le planificateur.

Dans l'architecture que nous proposons, illustrée par la figure 4.2, différents planificateurs de trajectoire et l'utilisateur interagissent par un échange de données au travers d'une interface visuo-haptique (IVH). Une fois sélectionné par l'utilisateur, le planificateur de trajectoire reçoit une consigne composée des configurations initiale et finale de la pièce à manipuler ainsi que d'un contexte de manipulation. Le planificateur calcule alors des commandes permettant de guider l'utilisateur entre sa configuration courante et la configuration finale. Ces commandes sont ensuite transmises à l'utilisateur par l'IVH.

Cette interface comprend :

- Un bloc de commande *haptique* permettant de transformer une trajectoire 6d en effort à imposer sur la

main de l'utilisateur. Il permet également de mesurer l'attitude de l'utilisateur face à cet effort grâce à des relevés de distance entre la configuration courante et la trajectoire ainsi que l'évolution de cette distance.

- Un bloc d'*affichage stéréoscopique et de capture de mouvements* permettant d'afficher la scène de test ainsi que des informations liées à la tâche comme la représentation de la configuration finale, des points de passage ou encore la totalité du passage trouvé... Il permet également de capturer les mouvements de l'utilisateur qu'il transmet au bloc d'analyse de comportement afin par exemple d'adapter le point de vue dans la simulation.
- Le bloc d'*entrées digitales* permet de lire les entrées discrètes (appuis sur les boutons) traduisant une formulation explicite des choix de l'utilisateur et de les communiquer au bloc d'analyse de comportement.
- Le bloc d'*analyse de comportement* permet, à partir des données récoltées par le système de capture de mouvement, des entrées digitales ainsi que du contexte de manipulation courant, de déduire l'attitude de l'utilisateur afin de renseigner le planificateur utilisé.

Ces différents blocs partagent les informations relatives aux objets (configuration courante, attributs de collision) et les informations déduites de l'attitude de l'utilisateur (Attributs de suivi, d'éloignement de la trajectoire ou d'attente) par l'intermédiaire d'une *structure de données de simulation.*

L'utilisateur perçoit de son coté un effort de guidage ainsi que des informations visuelles lui permettant de juger de la qualité du chemin généré par le planificateur. Il peut alors choisir de se laisser guider par l'effort de guidage. Dans le cas contraire, il peut soit explicitement spécifier une nouvelle requête de planification, soit implicitement engendrer des phases de re-planification par l'intermédiaire de son comportement vis à vis de la trajectoire calculée.

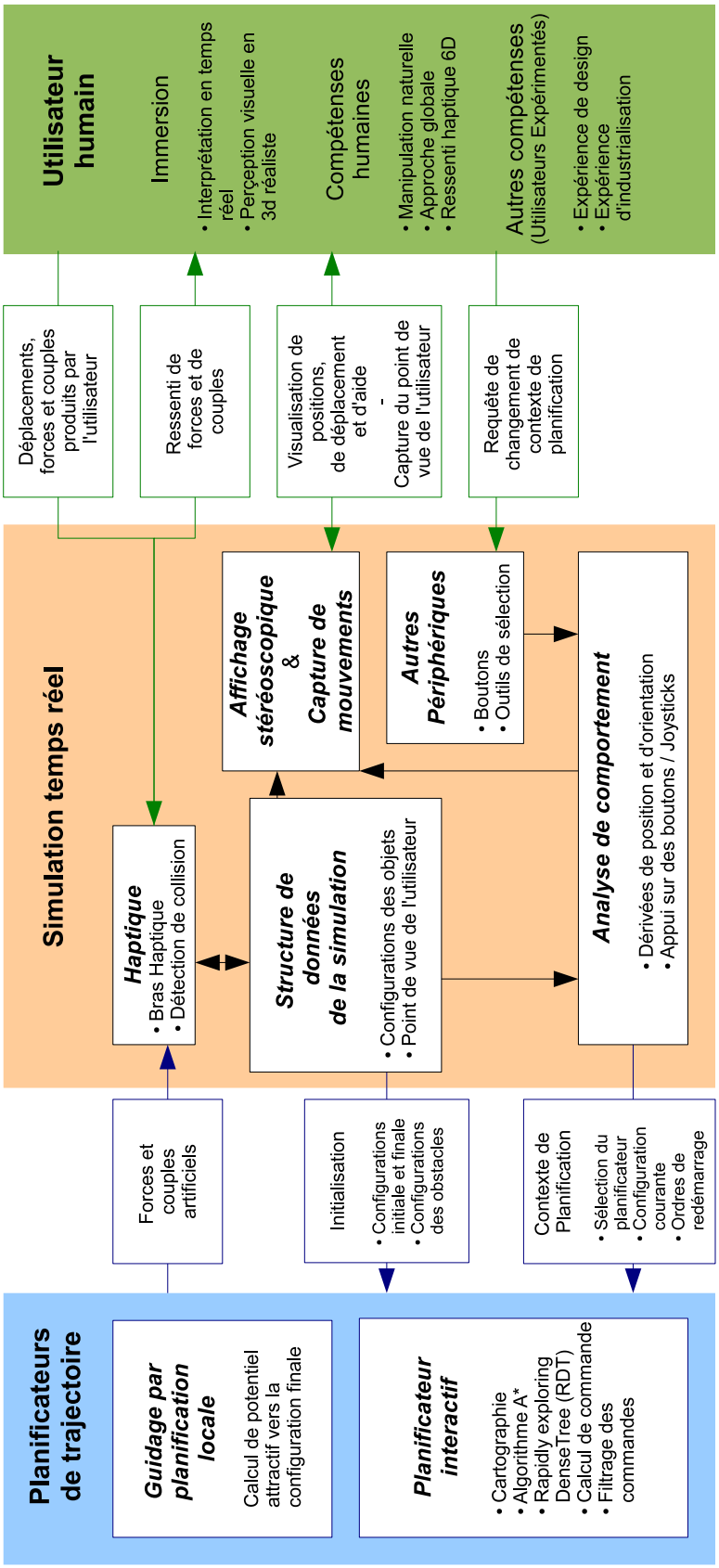


FIGURE 4.2 – Architecture et flux de données dans une simulation visuo-haptique de planification interactive

4.2.1 Choix d'un planificateur suivant le besoin de l'utilisateur

4.2.1.1 Determination du contexte de planification

L'architecture présentée en partie 4.2, permet l'utilisation de différents types de planificateurs suivant les besoins spécifiés par l'utilisateur. Des requêtes asynchrones permettent de changer de planificateur au cours de la manipulation offrant un choix entre une manipulation libre, une assistance faiblement guidée, et un guidage global dans les cas les plus complexes. Ce choix est laissé libre à l'utilisateur qui, en fonction de son appréciation personnelle, choisit le planificateur le plus approprié au problème.

4.2.1.1.1 Une assistance à la manipulation par un guidage faible issu d'une planification locale

Dans les cas les plus simples, l'utilisateur peut utiliser un système de guidage simple permettant notamment des opérations prohibés par un calcul de forces naturelles basé sur un détecteur de collision discret. En effet, dans les cas de manipulation proche du contact, comme des passages étroits, la qualité de manipulation est fortement dépendante de la qualité du retour d'effort. Lors de l'utilisation d'un détecteur de collision discret, certaines opérations d'insertions peuvent sembler interdites à l'utilisateur car elle nécessitent un maintien parfait de l'alignement de la pièce manipulée avec un ou plusieurs obstacles. Dans ce contexte, des composantes de guidage induisant des efforts permettant de maintenir cet alignement améliorent la manipulabilité.

Ce type de guidage permet une définition plus rapide et fortement interactive des trajectoires de montage et de démontage. De plus, dans un cadre de formation, ce type d'assistance peut être utilisé pour former des utilisateurs au geste technique dans des phases d'évaluation où seule la configuration finale du composant déplacé est donnée.

4.2.1.1.2 Un guidage issu d'une approche globale dans un environnement complexe

Dans les cas plus complexes induisant une forte charge cognitive, un autre type de planificateur peut, en tenant compte de l'environnement, fournir à l'utilisateur des pistes lui permettant d'éviter les obstacles proches ou encore lui proposer un chemin complet. Ce type de guidage peut être utilisé dans des cas de cinématique de montage ou de démontage complexe, lorsque la manipulation nécessite de fréquents changements de point de vue ou en cas de manipulation en aveugle.

L'utilisation en temps réel de tels planificateurs requiert tout d'abord une fragmentation des différentes phases de l'opération de calcul de chemin. Cette fragmentation doit être réalisée de manière à privilégier les échanges rapides de données entre l'utilisateur et le planificateur. Lors de la découverte d'un chemin possible, celui ci est proposé à l'utilisateur par le biais des interfaces sensorielles. L'utilisateur peut alors valider tout ou partie de ce chemin calculé en respectant les contraintes de guidage ou encore communiquer au planificateur un refus du suivi de la trajectoire.

4.2.1.2 Planificateurs utilisables en environnement interactif

4.2.1.2.1 Construction interactive du chemin : guidage par potentiels

Ce mode de guidage est analogue au fonctionnement d'un planificateur local de type "Méthode du champ de potentiels" décrit en partie 2.2.3.1. Une force de guidage dite "artificielle" est calculée en temps réel attirant la main de l'utilisateur vers la configuration finale de l'entité manipulée agissant alors de manière analogue à un champ attractif. Lors de la détection de collisions avec un ou plusieurs obstacles de la scène, le moteur de rendu haptique exerce une force dite "naturelle" traduisant le contact analogue à un potentiel répulsif. Le couple et la force ressentis par l'utilisateur au niveau de l'interface haptique sont alors calculés comme la somme de ces deux composantes. Lorsque ce mode de planification rencontre un minimum local, c'est à dire lorsque la somme des forces naturelles et de la force artificielle est nulle, l'utilisateur peut imposer sur le bras haptique un effort supplémentaire qui permet d'éviter ce point d'équilibre.

4.2.1.2.2 Guidage sur un chemin calculé

Différentes méthodes probablistes peuvent être utilisées pour planifier une trajectoire en 6d comme par exemple les méthodes PRM, RRT ou RDT détaillés dans la section 2.2.4. Cependant après différents essais de fragmentation de ces méthodes, les méthodes RRT et RDT semblent les plus adaptées pour le calcul interactif de trajectoire dans les cas de montage et de démontage. En effet, dans ces deux cas, la concentration de contraintes présentes au niveau des positions initiale et/ou finale et la forte variabilité de la densité d'obstacle dans l'espace de travail, engendrent des difficultés de convergence pour les méthodes PRM. Dans ce cas de figure, les méthodes de planification par diffusion, permettant une exploration progressive de l'espace des configurations, semblent plus adaptées. Néanmoins, le temps d'exécution requis pour la planification de chemin par une méthode de type RRT, reste très aléatoire et fortement dépendante du contexte. Cependant, ce défaut peut être minimisé dans le cas de l'utilisation d'une méthode RDT utilisant des heuristiques de jet de points adaptées.

En prenant en compte la contrainte de temps réel, nous avons décidé de diviser ce planificateur interactif en 2 parties principales correspondant à des temps de calcul différents comme illustré par la figure 4.3.

1. Tout d'abord, dans une phase d'initialisation précédant la manipulation et donc sans contrainte de temps réel, nous créons une représentation 3d discrète de l'espace de travail en utilisant une structure en arbre octal déséquilibré dans le but de collecter le maximum d'informations sur l'organisation de l'espace.
2. Ensuite, dans une phase interactive et pour chaque pièce manipulée, nous générons un volume libre 3d continu entre la position initiale et finale au moyen d'un algorithme A* qui sera utilisé pour limiter la position (x, y, z) des configurations jetées par un planificateur RDT (Rapidly-exploring Deterministic Tree) bi-directionnel. Après convergence du RDT, nous obtenons une trajectoire libre en 6d.

Une fois un passage libre calculé, nous procédons à un asservissement de la main de l'utilisateur sur ce passage en pilotant l'interface haptique. L'utilisateur peut ensuite décider soit de suivre le chemin qui lui

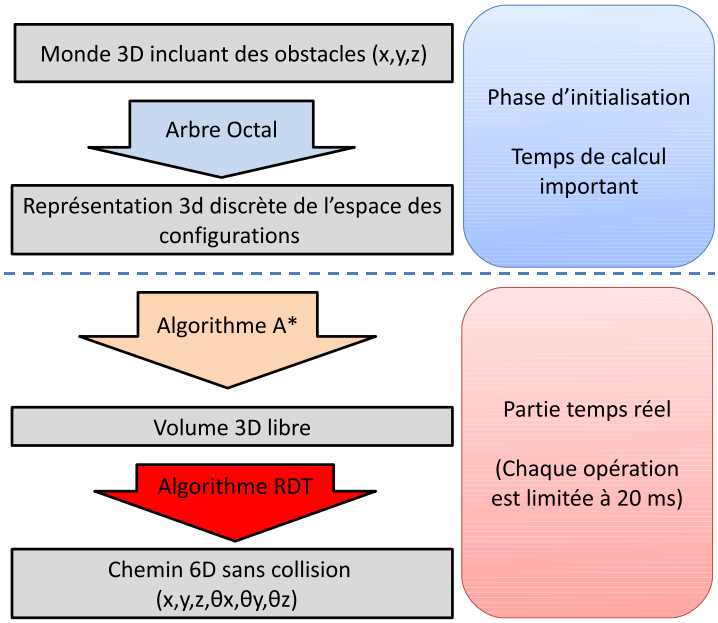


FIGURE 4.3 – Architecture générale du planificateur

est suggéré en se laissant guider soit de lutter contre cette force de guidage. Ce deuxième cas illustrant une volonté de l'utilisateur de quitter le passage proposé, la partie temps réel du planificateur est redémarrée. Lors de ce nouveau démarrage, la position de départ de l'algorithme A* est réactualisée ce qui provoque la rectification de la zone de jets de points du RDT. Les deux arbres utilisés par l'algorithme RDT sont alors enrichis de nouveau jets de points afin de calculer une nouvelle trajectoire en accord avec la volonté de l'utilisateur.

4.2.2 Planificateur interactif basé sur une méthode RDT

4.2.2.1 Phase d'initialisation : Cartographie préliminaire

4.2.2.1.1 Cartographie par discrétisation spatiale

La première étape consiste à créer une représentation 3d discrète de l'espace de travail. Cette étape représente la partie principale du temps de calcul total nécessaire au planificateur et peut être considérée comme une initialisation car elle est réalisée avant le lancement du programme principal. Pour réaliser cette discrétisation, nous utilisons un arbre octal déséquilibré à profondeur réglable. La profondeur de l'arbre sera choisie en fonction de la taille des pièces présentes dans la scène afin de minimiser le temps de calcul nécessaire et l'espace mémoire requis. La structure de données ainsi construite est utilisée pour tous les problèmes de planification des différents sous ensembles à assembler ou à démonter présents dans la scène. Chaque feuille de l'octree contient des informations importantes, comme les pièces avec lesquelles elle est en collision et le vecteur normal au contact, qui seront utilisées plus tard pour réduire le temps de calcul des parties suivantes.

4.2.2.1.2 Mise à jour de la cartographie

Dans notre cas d'assemblage séquentiel, l'arbre octal doit être mis à jour pour chaque phase de l'assemblage. La pièce déposée est considérée comme un obstacle et la nouvelle pièce saisie devient l'objet mobile. Cette mise à jour est réalisée de manière locale pour minimiser le temps de calcul : dans un premier temps, les noeuds de l'arbre sont actualisés de manière à éliminer leur éventuelle précédente liaison avec l'objet qui a été déplacé. Dans le cas où l'objet est le seul en collision avec un noeud, ce noeud devient libre, ce qui peut entraîner la modification de l'état de ses ascendants. Enfin, les noeuds en état de collision avec la boite englobante de l'objet (AABB) à sa configuration finale voient leur état ré-évalués.

4.2.2.2 Planification temps réel

4.2.2.2.1 Algorithme A*

Cette partie du programme permet la construction d'un passage 3d volumique assimilable à un "tunnel" reliant la position initiale et la position finale. Le but de cette partie est d'accélérer la convergence du RDT en définissant des limites à la zone de jet de points. Pour construire ce "tunnel", nous utilisons un algorithme A* (détaillé par l'algorithme 4.1) recherchant un passage libre dans l'arbre octal déséquilibré réalisé lors de la phase d'initialisation. Cet algorithme progresse en minimisant un fonction de coût définie par $f(n) = g(n) + h(n)$, où $g(n)$ est la distance parcourue depuis la position initiale jusqu'au centre du cube d'indice n et $h(n)$ la distance euclidienne reliant le centre du cube d'indice n courant à la position finale.

Algorithme 4.1 Algorithme A* standard

DEBUT
$ouverts \leftarrow$ {état initial}
$fermés \leftarrow$ vide
$succès \leftarrow$ faux
TANT QUE (($ouverts \neq \oslash$) ET ($succès = faux$)) FAIRE
  Choisir $n \in ouverts$ tel que $f(n)$ est minimum
  SI est_final(n) ALORS $succès \leftarrow$ vrai
  SINON $ouverts \leftarrow ouverts$ privé de n
  POUR chaque successeurs s de n FAIRE
    SI (($s \notin ouverts$) ET ($s \notin fermés$)) ALORS
      | $ouverts = ouverts + s$
      | $père(s) = n$
      | $g(s) = g(n) + coût(n, s)$
    SINON
      | SI ($g(s) > g(n) + coût(n, s)$) ALORS
      | $père(s) = n$
      | $g(s) = g(n) + coût(n, s)$
      | SI ($s \in fermés$) ALORS
      | $fermés = fermés - s$
      | $ouverts = ouverts + s$
FIN

Les résultats observés lors de l'implémentation de l'algorithme original montrent qu'il permet de converger vers une solution minimisant le fonction $f(n)$. Cependant, l'algorithme A* original présente dans notre contexte d'utilisation des inconvénients majeurs :

- Il est conçu pour une utilisation dans des structures équilibrées, pour un cube de départ et un cube d'arrivée libres et doit être initialisé avec un niveau (ou une taille) de cube fixe avant de s'exécuter. Dans certains cas, il peut ne pas converger vers une solution du fait de l'organisation des obstacles dans l'espace ou du niveau de décomposition choisi. Dans ce cas, une recherche plus poussée doit être envisagée à un niveau de décomposition supérieur permettant le passage par un ou plusieurs cubes de taille plus faible.
- La trajectoire générée, constituée d'une liste de cubes à traverser, est calculée grâce la minimisation de la longueur du chemin parcouru ce qui se traduit généralement par une solution tangente aux obstacles. Dans son contexte d'utilisation, le chemin calculé n'est pas satisfaisant car il induit des tests de détection de collisions plus complexes à calculer, et rend parfois la convergence de l'algorithme RDT impossible car contraignant le jet de configuration dans une portion de $\mathbb{C}_{obs}$ [1].
- Enfin, cet algorithme ne peut pas être exécuté lorsque les cubes initial et/ou final ne sont pas libres.

A partir de cette liste d'inconvénients, l'algorithme A* est modifié afin de mieux répondre aux besoins :

- L'algorithme A* est adapté afin de permettre une planification dans des structures déséquilibrées et initialisé grâce à une limite maximale l_{max} de niveau de cube correspondant à la dimension minimale de la boîte englobante orientée (OBB) de l'objet déplacé. Dans le cas où cette initialisation ne permet pas la convergence, cette limite est incrémentée $l_{max} \leftarrow l_{max} + 1$ ce qui peut engendrer de nouvelles divisions de certains cubes de l'arbre octal. Cependant, pour des raisons de gestion de la complexité, une limite L_{MAX} est fixée au niveau du nombre d'incrémentations successives admissibles.
- L'algorithme est également ajusté dans les cas où le ou les cubes englobant les configurations initiale et/ou finale sont occupés. Dans ces cas, le cube occupé est alors redécomposé itérativement jusqu'à l'obtention d'un cube libre. Le niveau maximal de cube l_{max}, correspondant au cube le plus petit admissible, est alors recalculé localement lorsque les re-décompositions précitées ont induit un calcul local de cubes de niveau supérieur à L_{MAX}.
- L'heuristique initiale $f(n) = g(n) + h(n)$ est revue de manière à trouver un passage solution équilibrant la contrainte d'un passage optimisé en longueur mais aussi relativement éloigné des obstacles. Dans cette optique, une pénalité liée à la taille des cubes traversés est intégrée à cette heuristique :

$$f(n) = g_1(n) + \frac{h(n)}{a(n)} \text{ avec } g_1(n) = \frac{d(n, n+1)}{s(n+1)}$$

Cette nouvelle formulation de l'heuristique fait intervenir la taille du cube enfant évalué $s(n+1)$, la distance entre les centres des cubes courant et évalués $d(n, n+1)$ ainsi que la longueur $a(n)$ calculée comme la norme maximale du vecteur $\overrightarrow{h(n)}$ projeté sur les axes $(\vec{x}, \vec{y}, \vec{z})$ de la scène, c'est à dire la valeur maximale des composantes du vecteur $\overrightarrow{h(n)}$, illustrée sur la figure 4.4.

1. $\mathbb{C}_{obs}$: Partie non libre de l'espace des configurations

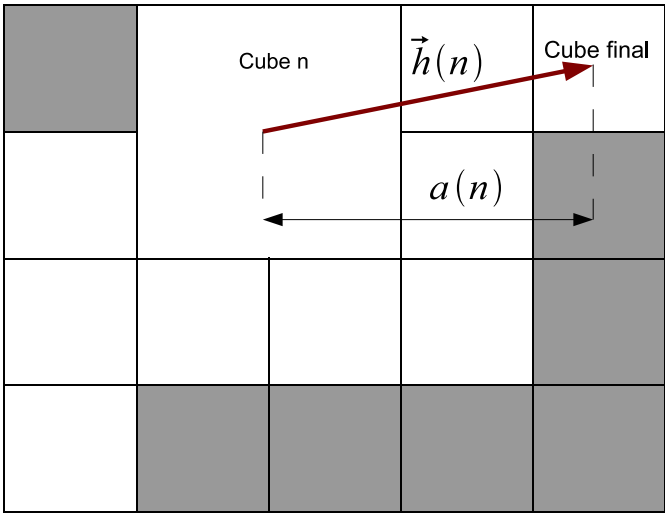


FIGURE 4.4 – Illustration du calcul de la fonction $a(n)$

En incluant, cette variante de l'heuristique, la trajectoire calculée maximise le volume du passage tout en maintenant une optimisation de la longueur du chemin parcouru comme illustré sur la figure 4.5. De cette manière, lorsque ce type de solution est utilisé pour contraindre la méthode de jet de points de l'algorithme RDT, les tests de collision nécessaires aux calculs des possibilités d'expansion des branches des arbres sont moins coûteux car les configurations jetées ont une plus grande probabilité d'appartenir à $\mathbb{C}_{libre}$. La maximisation du volume 3d permet de réduire les contraintes en position des jets de configurations 6d réalisé par l'algorithme RDT afin d'augmenter la probabilité de jet dans $\mathbb{C}_{libre}$ et ainsi d'obtenir une solution.

4.2.2.2.2 Méthode RDT

Cette dernière partie du planificateur a pour but de trouver un passage en 6d pour l'objet manipulé en utilisant les données préparées lors des deux précédentes étapes (Arbre octal et A*). Pour réaliser cette tâche nous avons implémenté un RDT (Rapidly-exploring Dense Tree : RRT avec un jet de point déterministe) bi-directionnel présenté dans [86, 115]. La notion de distance utilisée dans la fonction de recherche de plus proche voisin du RDT est implémentée en 6d en utilisant la norme pondérée nommée Norm6D entre deux configurations [2] q_1 et q_2 définie dans [116] par :

$$norm6D(q_1, q_2) = \sqrt{\mu_{pos} \times (\Delta x^2 + \Delta y^2 + \Delta z^2) + \mu_{rot} \times (\Delta_{\theta x}^2 + \Delta_{\theta y}^2 + \Delta_{\theta z}^2)}$$

2. q_i : Configuration d'une entitée caractérisée en un état i donné par la définition de sa position $[x\,y\,z]$ par rapport à un repère de scène et de son orientation par les angles d'euler $[\theta_x\,\theta_y\,\theta_z]$

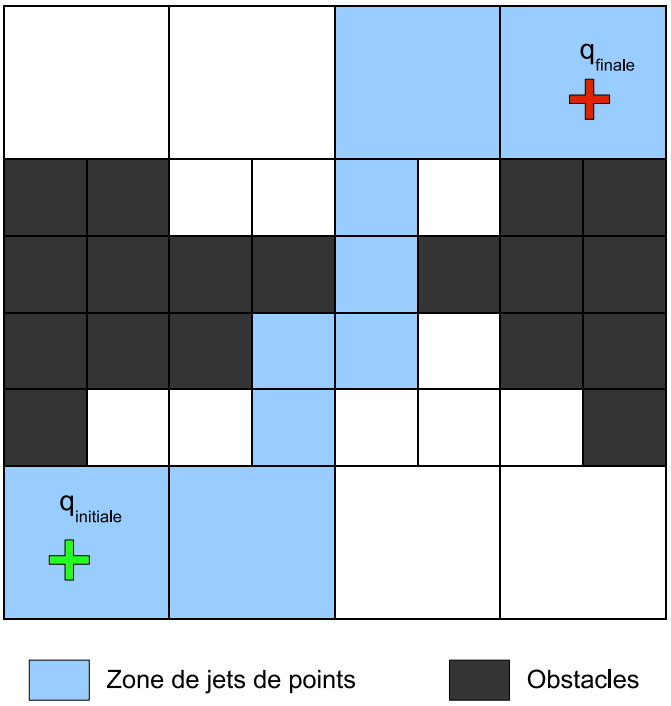


FIGURE 4.5 – Illustration de la solution de l'algorithme A* modifié dans une structure 2D déséquilibrée permttant de définir une zone de jets de points pour le RDT

$$\text{avec} \begin{cases} \mu_{pos} & \text{la pondération alouée à la composante de position} \\ \mu_{rot} & \text{la pondération alouée à la composante de orientation} \\ \Delta i & = q_2 \cdot i - q_1 \cdot i \text{ avec } i = x, y, z, \theta_x, \theta_y, \theta_z \end{cases}$$

L'algorithme RDT original se propage en utilisant une limite de diffusion qui permet l'exploration progressive de l'espace. Ceci permet habituellement d'obtenir de bons résultats mais nécessite un grand nombre de jets de points ce qui augmente d'autant plus la taille utilisée en mémoire et le nombre d'itérations nécessaires pour la convergence. Dans notre cas où le temps de calcul et les ressources sont limités, nous devons minimiser le nombre de jets de points. L'approche globale concrétisée par la solution de l'algorithme A* nous permettant de jeter des points ayant de plus grandes chances d'être utiles à la résolution du problème, nous avons donc décidé d'implémenter l'algorithme RDT sans limite de diffusion. Nous avons utilisé la méthode RDT avec une fonction de jet de points limitée au volume délimité par l'union des cubes formant la solution de l'algorithme A* (notée $SolA*$) comme illustré sur la figure 4.5.

La méthode de jet de points utilisée permet de choisir une position aléatoire inclue dans la zone de restriction définie par la solution de l'algorithme A*. Nous utilisons donc une méthode naïve s'appuyant sur une loi uniforme pour la définition de la position $[x\,y\,z] \in SolA*$ et une loi uniforme sur $]-\pi, \pi]$ pour le choix des angles d'Euler codant l'orientation.

Cette méthode est améliorée par une heuristique permettant le dégagement rapide des obstacles lors de la croissance des arbres de diffusion. Lors des premiers jets de points utilisés pour la croissance de chacun des deux arbres, nous utilisons des heuristiques basées sur la définition de configurations de dégagement. Dans le cas d'assemblage ou de démontages classique, l'approche de la configuration finale est dans la majoritée des cas une opération de translation suivant les axes principaux de l'entitée déplacée. Cette configuration de dégagement peut alors être choisie comme appartenant à l'un des axes principaux de la boîte englobante orientée de l'objet. Pour trouver le vecteur de dégagement parmi les 6 possibilités définies par les axes principaux, nous utilisons le vecteur normal au contact $\overrightarrow{V_{nc}}$ stocké dans le cube incluant la configuration étudiée, illustré sur la figure 4.6. Le vecteur de dégagement $\overrightarrow{V_d}$, illustré sur la figure 4.7, est choisi parmi les directions principales de la boîte englobante orientée de manière à maximiser le critère $\overrightarrow{V_d} \cdot \overrightarrow{V_{nc}}$.

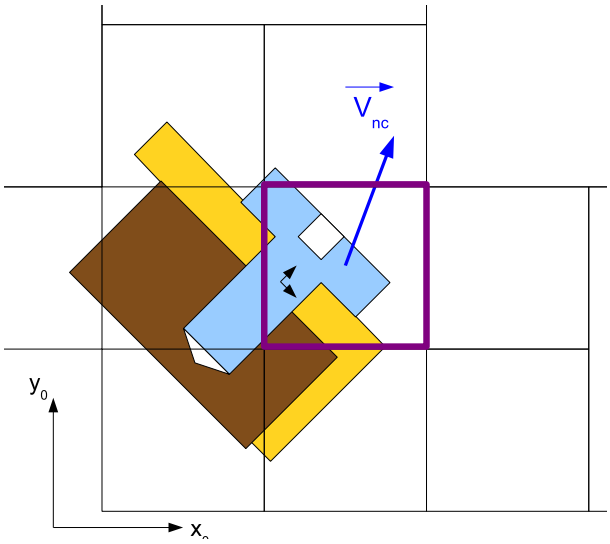


FIGURE 4.6 – Schéma 2d illustrant le vecteur $\overrightarrow{V_{nc}}$ (calculé à partir de la détection d'intersection entre une feuille de l'arbre octal et les obstacles de la scène) stocké dans la feuille de l'arbre octal contenant la configuration de départ de l'opération de dégagement. Dans cet exemple, l'objet déplacé est une vis en position finale (en bleu sur la figure) qui n'est pas considérée comme un obstacle.

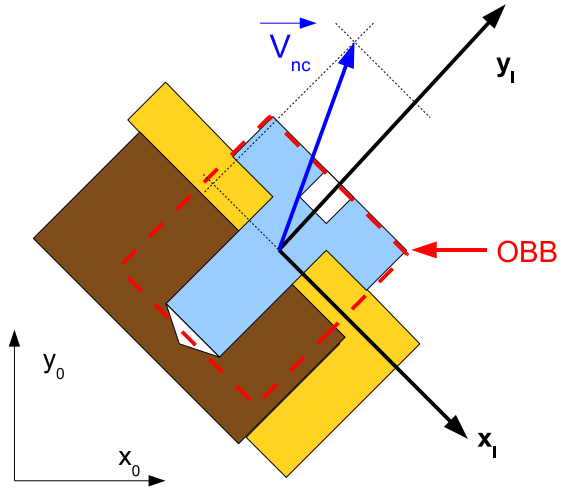


FIGURE 4.7 – Shéma 2d illustrant la détermination d'un vecteur de dégagement $\overrightarrow{V_d}$ parmi les 4 vecteurs $\overrightarrow{+x_l}$, $\overrightarrow{-x_l}$, $\overrightarrow{+y_l}$, $\overrightarrow{-y_l}$, à partir de la connaissance de $\overrightarrow{V_{nc}}$. Dans cet exemple, le vecteur de dégagement retenu $\overrightarrow{V_d} = \overrightarrow{+y_l}$ car il maximise le produit scalaire $\overrightarrow{V_{nc}}.\overrightarrow{V_d}$.

4.2.3 Génération de commandes et description de l'interaction

4.2.3.1 Présentation de l'interaction

Le passage trouvé par le planificateur est suggéré à l'utilisateur au moyen de l'interface haptique. Cette interface est prévue, dans une manipulation standard, pour retourner un effort dit "naturel" caractérisant les contacts de la pièce manipulée avec les obstacles de l'environnement. Pour communiquer le passage trouvé par le planificateur à l'utilisateur, nous ajoutons une force artificielle qualifiée de « force de guidage » qui pousse l'utilisateur à suivre le chemin solution sans pour autant restreindre totalement ses déplacements comme illustré sur la figure 4.8.

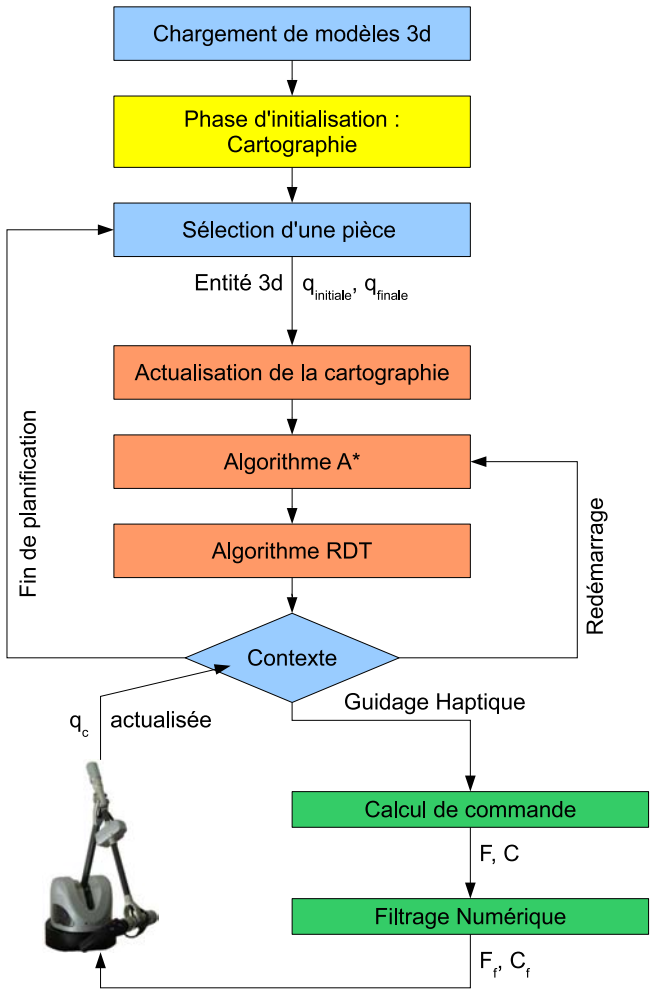


FIGURE 4.8 – Illustration de l'interaction entre le planificateur et l'utilisateur

Cette force de guidage est appliquée de deux manières différentes suivant la configuration courante de la pièce manipulée. Dans les cas où le guidage ne démarre qu'après la découverte d'un premier chemin calculé

par la première itération de planification, l'utilisateur est libre de se déplacer et l'objet manipulé peut être dans une configuration très différente de la configuration initiale au démarrage de l'asservissement. Pour prendre en compte ce cas, nous définissons une valeur limite pour la norme 6d spécifiée par l'utilisateur, notée *DL*, représentant un volume qualifié de "zone de suivi", illustré sur la figure 4.9, autour de la trajectoire à suivre.

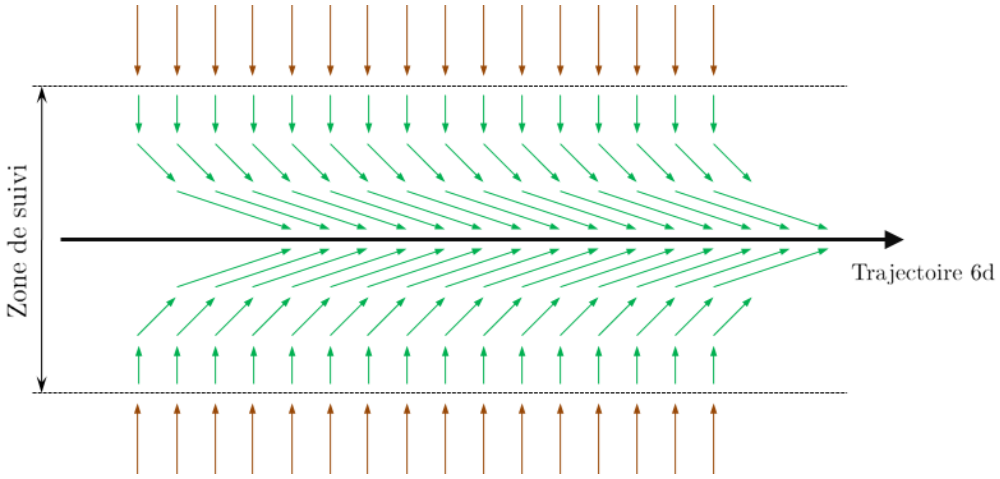


FIGURE 4.9 – Illustration discrétisée en 2d des deux modes de calcul de la force de guidage. Le champ vectoriel Rouge repésente la phase d'approche, le champ vert représente la phase de suivi. La norme *DL* permet de définir la zone de suivi (ici $2 \times DL$) séparant ces deux types de commande.

La définition de cette zone permet de réaliser deux modes de guidage différents selon que l'utilisateur doit rejoindre ou suivre la trajectoire. Nous utiliserons également *DL* pour détecter la volonté de l'utilisateur de quitter la trajectoire afin de rechercher une autre solution.

4.2.3.2 Guidage de l'utilisateur par retour d'effort

Afin de guider l'utilisateur sur la trajectoire trouvée par le planificateur, nous réalisons un asservissement en configuration de l'interface haptique sur le chemin 6d calculé. Dans un premier temps, un générateur de consigne détermine la configuration q_a du chemin la plus proche de la configuration courante q_c qui constitue une configuration attractive de référence comme illustré sur la figure 4.10. Cette configuration attractive évolue au cours du guidage le long de la trajectoire à suivre. Cependant, le générateur de consigne interdit tout retour en arrière sur la trajectoire afin d'éviter des instabilités sur un passage non lissé.

4.2.3.2.1 Calcul d'une commande simulant une configuration attractive

Pour modéliser cette force, le programme détecte à chaque rafraîchissement d'image (60 Hz) une confi-

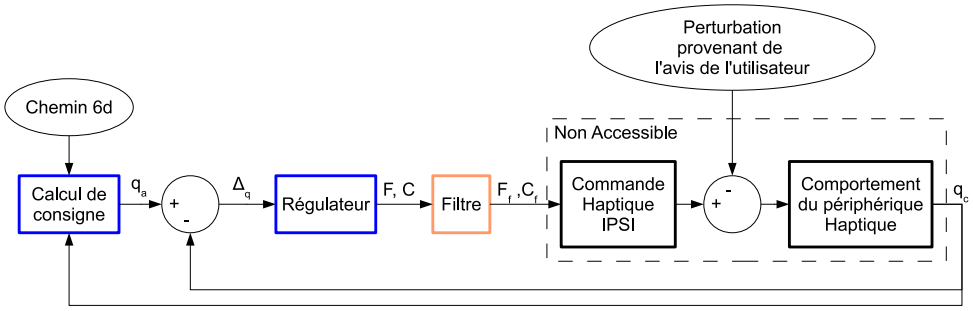


FIGURE 4.10 – Schéma de l'asservissement du bras haptique sur une trajectoire planifiée avec :
q_c : Configuration courante de l'entité manipulée
q_a : Configuration attractive
Δq : Vecteur différenciel $\left[\Delta x\, \Delta y\, \Delta z\, \Delta\theta_x\, \Delta\theta_y\, \Delta\theta_z\right]$
F, C : Force et de Couple à appliquer
F_f, C_f : Force et couple filtrés

guration attractive située sur le chemin solution calculé à partir de la position courante. Une commande constituée d'une force F et d'un couple C est calculée afin de rejoindre cette configuration attractive.

Pour calculer cette commande, nous devons transformer une différence de configuration Δq entre une configuration courante q_c et une configuration attractive q_a en une force F et un couple C à appliquer au niveau du bras haptique. Nous dimensionnons chaque composante de F et de C en fonction des composantes du vecteur Δq de manière à éviter les instabilités et les blocages au niveau du bras haptique en utilisant les fonctions :

$$F_i = \frac{\Delta q \cdot i}{|\Delta q \cdot i|} \times \sqrt{K_F \times \frac{|\Delta q \cdot i|}{\left\|\left[\begin{array}{ccc} \Delta q \cdot x & \Delta q \cdot y & \Delta q \cdot z \end{array}\right]\right\|}} \times F_M,\ i = [x, y, z]$$

$$C_i = \frac{\Delta q \cdot i}{|\Delta q \cdot i|} \times \sqrt{K_C \times \frac{|\Delta q \cdot i|}{\pi}} \times C_M,\ i = \left[\theta_x, \theta_y, \theta_z\right]$$

où F_M est la valeur maximale de la force allouée au guidage et C_M la valeur maximale du couple alloué au guidage. Les arguments K_F et K_C sont définis par l'utilisateur en fonction de ses préférences ergonomiques. Lors des tests réalisés, nous avons choisi $K_F = 3$ et $K_C = 5$ empiriquement car permetant d'obtenir un guidage fluide, assez naturel du point de vue de l'utilisateur.

4.2.3.2.2 Problèmes de stabilité et filtrage

La trajectoire calculée par l'algorithme RDT est constituée d'une succession de segments libres reliant des configurations libres depuis la configuration initiale jusqu'à la configuration finale. Ces segments sont réalisés par des interpolations linéaires. Dans une première étape, et pour faciliter le calcul des consignes de force et de couple, cette trajectoire est discrétisée en une succession de configurations de passage. Du fait

de l'état non lissé de la trajectoire, la force et le couple calculés pour le suivi sont discontinus. Pour garantir une stabilité des commandes envoyées au bras haptique et de ce fait une bonne ergonomie de manipulation pour l'utilisateur, nous filtrons les commandes en utilisant la formule suivante :

$$T = \frac{2}{(N+1) \times N} \times \sum_{i=0}^{N-1} [(N-i) \times T_{(t-i)}]$$

Ce filtre est implémenté de manière à ce que chaque composante du torseur T (force et couple) soit calculé en fonction d'une moyenne pondérée des $(N-1)$ commandes précédentes $T_{(t-i)}$. Ce type de filtre permet de lisser les commandes mais introduit un retard qui sera compensé par une expression adaptée des commandes et l'utilisation de la dynamique de l'utilisateur.

4.2.3.2.3 Génération de commande pour le suivi de trajectoire

A l'extérieur de la "zone de suivi"

Lorsque l'utilisateur est à l'extérieur de la "zone de suivi", il doit tout d'abord rejoindre le plus vite possible cette zone afin d'engager la seconde phase de suivi. Dans ce but, la commande appliquée au bras haptique est calculée en considérant comme configuration attractive la configuration du chemin la plus proche de la configuration courante, illustrée par des flèches rouges sur la figure 4.9. Cette commande est utilisée dans les cas où le temps de calcul d'une trajectoire laisse à l'utilisateur le temps de s'écarter de la configuration initiale.

A l'intérieur de la "zone de suivi"

Une fois que la "zone de suivi" est atteinte, les commandes envoyées aux bras haptiques sont générées au moyen d'une commande illustrée par la figure 4.11 et représentée par des flèches vertes sur la figure 4.9. A partir de l'expression de la configuration la plus proche de la configuration courante q_n, nous exprimons des efforts F_t calculés comme des efforts attractifs vers les configurations q_{n+t-1}. Ces différents efforts sont ensuite utilisés pour calculer une commande permettant de compenser le retard qui sera introduit par le filtrage. Pour cela chaque argument du torseur T, à appliquer au bras haptique, est calculé de la manière suivante :

$$T = \frac{2}{(N+1) \times N} \times \sum_{i=0}^{N-1} [(N-i) \times T_{(t+i)}]$$

Pour équilibrer le retard introduit par l'application du filtre de commande explicité précédemment, N est choisi de manière identique pour le calcul de commande et le dimensionnement du filtre.

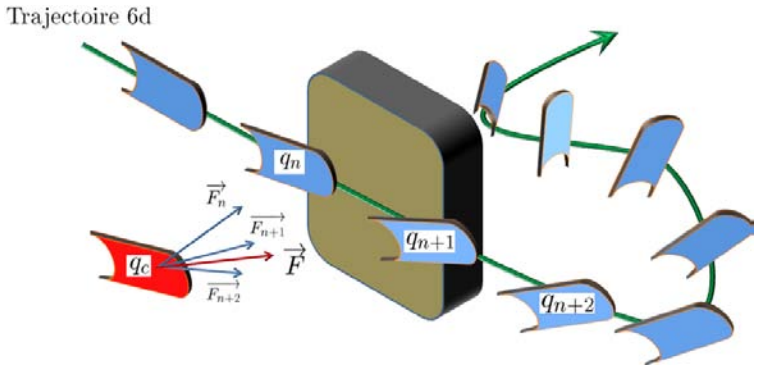


FIGURE 4.11 – Illustration du calcul de commande prédictive pour le suivi de trajectoire

4.2.3.3 Guidage de la planification par l'utilisateur

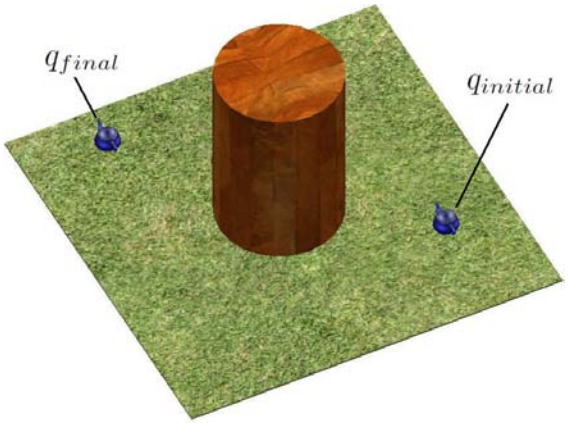
Lorsque l'utilisateur n'est pas satisfait par le chemin qui lui est proposé par le biais de l'interface haptique, il peut tout naturellement lutter contre l'effort qui le pousse à suivre la trajectoire calculée. Le fait de suivre un chemin différent de celui proposé implique que la position courante de l'entité manipulée est détectée comme sortant de la "zone de suivi". L'asservissement sur la trajectoire est alors neutralisé et la partie temps réel du planificateur est redémarrée avec de nouvelles conditions initiales pour l'algorithme A*. Ces nouvelles conditions initiales provoquent la découverte d'une zone de jet de point parfois différente. La cartographie du RDT est alors augmentée par de nouveaux jets de points jusqu'à la découverte d'une nouvelle trajectoire à suivre.

4.2.3.3.1 Informations visuelles mises à la disposition de l'utilisateur

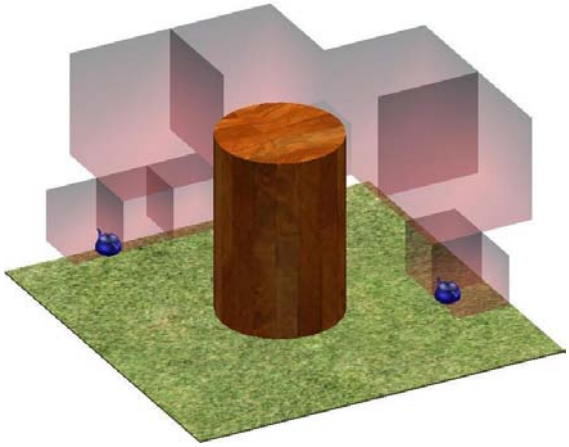
Afin d'améliorer la communication entre le planificateur et l'utilisateur, il est possible de symboliser la progression du planificateur à l'aide de métaphores visuelles. Dans un premier temps, la configuration finale de l'objet manipulé est représentée à l'aide d'une réplique de l'objet manipulé afin de communiquer à l'utilisateur le but de l'opération de calcul de chemin (Figure 4.12a). L'affichage du résultat de la recherche A* permet de communiquer à l'utilisateur la zone de recherche afin qu'il puisse interpréter l'approche globale utilisée par le planificateur (Figure 4.12b). La croissance des arbres issus des configurations initiale et finale est symbolisée en 3d par une série de segments et de noeuds informant l'utilisateur de la répartition dans l'espace des jets de points réalisés, de l'avancée de l'opération de planification, ainsi que de ses difficultés éventuelles (Figures 4.12c et 4.12d). Enfin, lorsqu'un chemin solution est découvert, la trajectoire à suivre est affichée par une répétition de l'objet déplacé dans différentes configurations (Figure 4.12e).

L'affichage de l'ensemble des informations du planificateur permet de bien visualiser l'approche globale adoptée par le programme mais surcharge le champ de vision de l'utilisateur. En pratique, l'affichage de la représentation 3d des arbres RDT s'avère trop complexe pour être réellement interprétable par un utilisateur. En revanche, l'affichage du résultat de la méthode A* permet à l'utilisateur de vérifier son influence sur l'approche globale choisie par le planificateur. De plus, l'affichage de la trajectoire finale permet à l'utilisateur de mieux interpréter les changements de configuration à effectuer et ainsi de mieux appréhender le guidage de manière à lisser la trajectoire. Néanmoins, cette dernière pouvant être ressentie directement par guidage

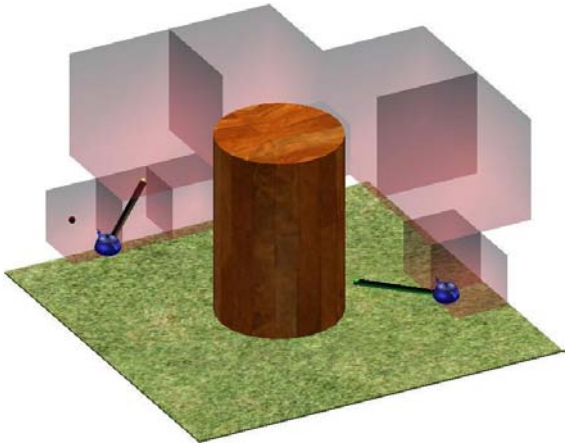
au travers de l'interface haptique, son affichage peut constituer une redondance d'information superflue dans des cas simples.



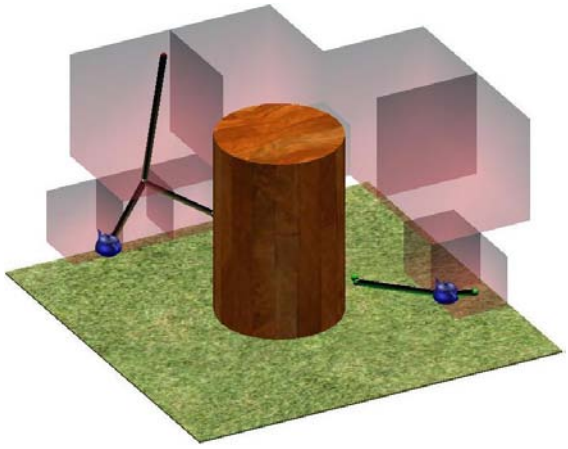
(a) Scène de test très simple, composée d'un seul obstacle



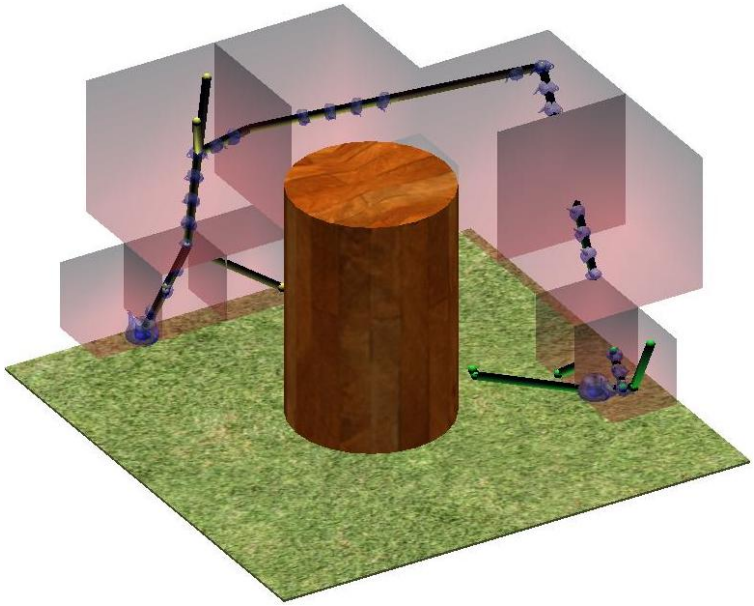
(b) Visualisation du volume servant de limitation à la méthode de jet de la méthode RDT



(c) Visualisation du démarrage de la méthode RDT par l'affichage des arbres de diffusion



(d) Visualisation de la progression de la méthode RDT



(e) Visualisation du chemin solution calculée

FIGURE 4.12 – Informations visuelles transmises par le planificateur à l'utilisateur

4.2.3.3.2 Interprétation de l'écartement de l'utilisateur de la trajectoire proposée

L'analyse du comportement de l'utilisateur pendant la phase de suivi de la trajectoire permet de détecter sa volonté de suivre un chemin différent de celui sur lequel il est guidé (cf figure 4.13). Cette détection est réalisée par la surveillance de la norme 6d entre la configuration courante q_c de l'entité qu'il manipule et la configuration q_a la plus proche de q_c appartenant au chemin suivi. Lorsque cette norme est détectée comme supérieure à la limite DL fixée par l'utilisateur, l'asservissement haptique sur la trajectoire est désactivé et une nouvelle demande de calcul de chemin est envoyée au planificateur comme illustré sur la figure 4.14.

Lors du redémarrage de la partie temps réel du planificateur, une nouvelle zone de jet est calculée par l'algorithme A* entre un cube libre contenant la configuration courante q_c et le cube final (cf figure 4.14). L'algorithme RDT contraint alors les nouveaux jets de points dans cette nouvelle zone en réutilisant les arbres existants pour calculer une solution différente (cf figure 4.15).

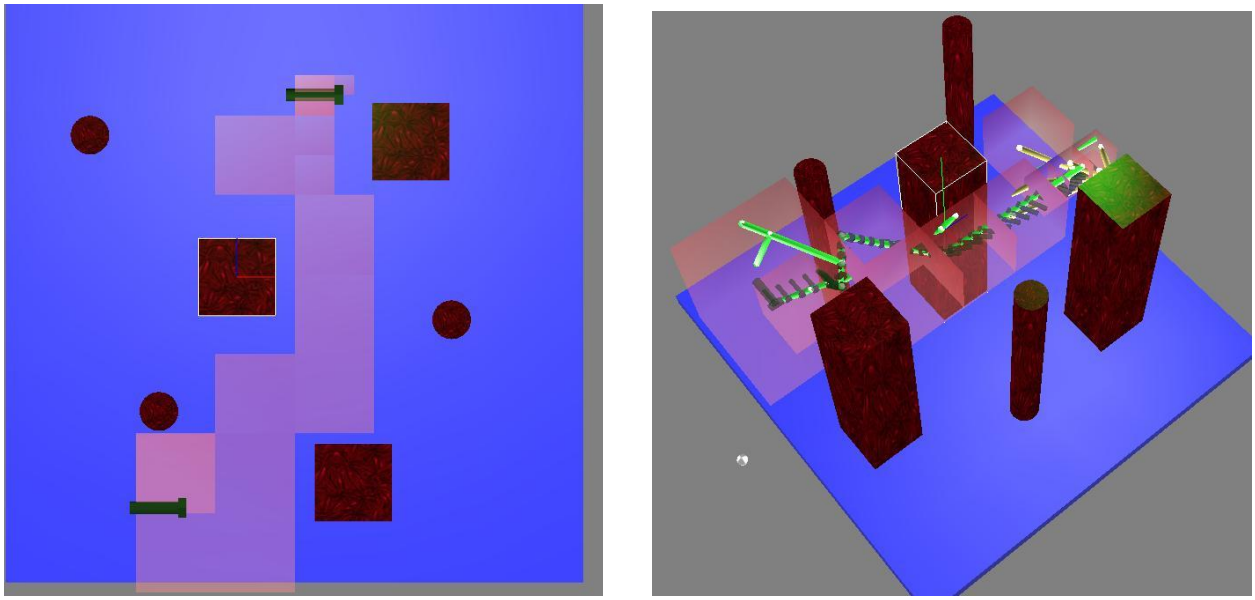


FIGURE 4.13 – Premier chemin solution proposé ne convenant pas à l'utilisateur

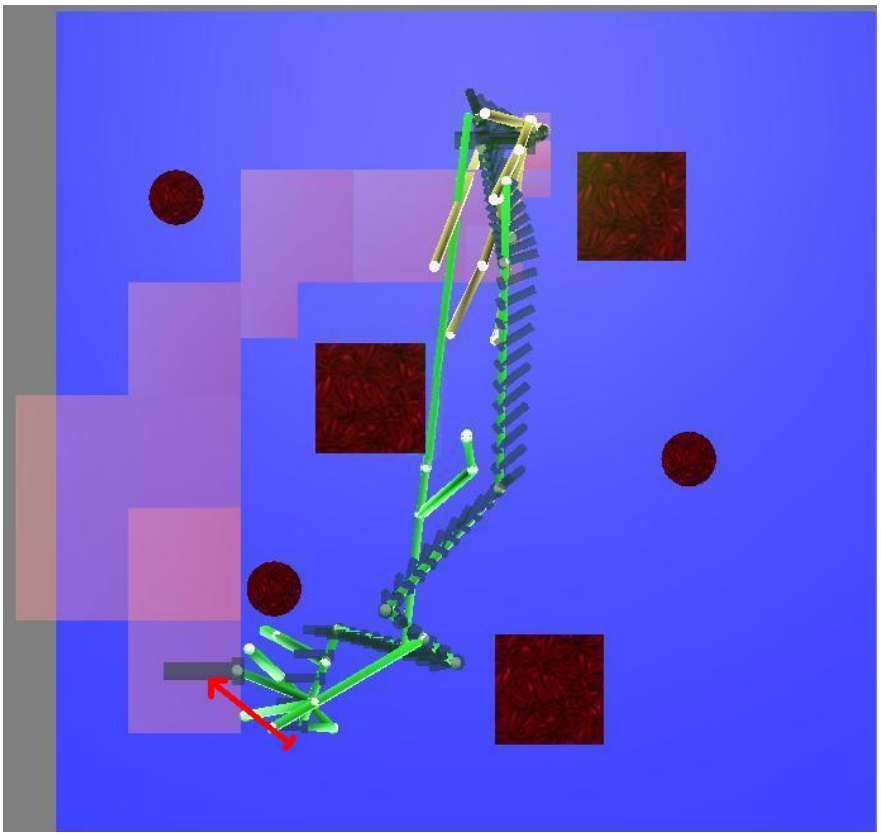


FIGURE 4.14 – Décrochage volontaire de l'utilisateur, dont la trajectoire suivie est illustrée par une flèche rouge, entraînant un redémarrage de l'algorithme A* et la découverte d'une nouvelle solution.

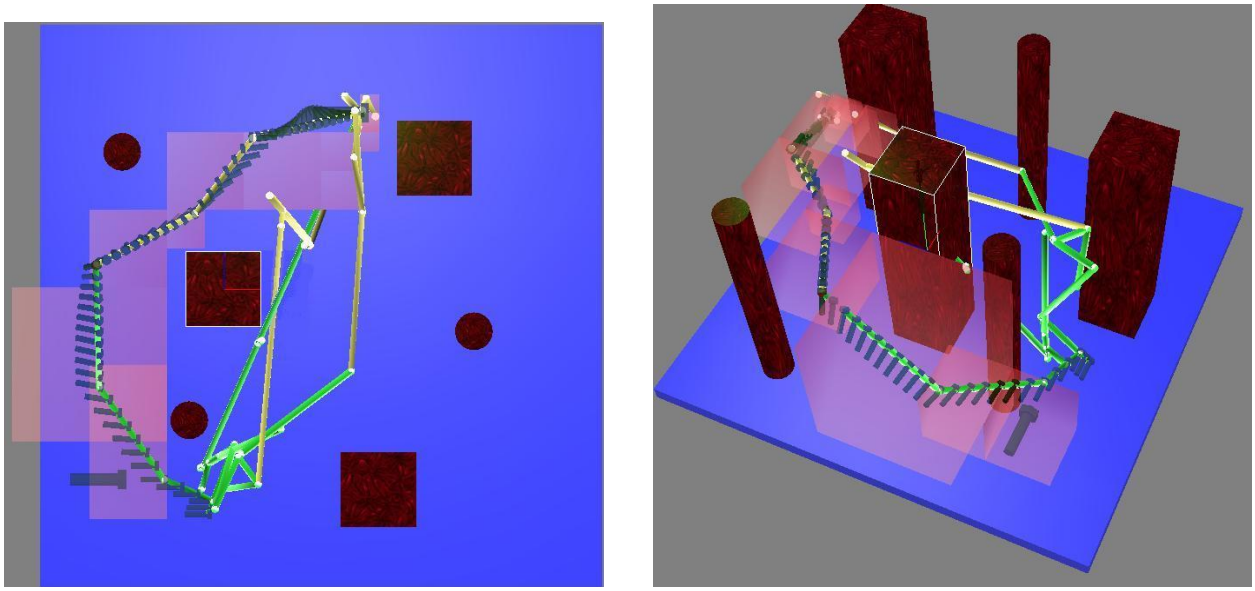


FIGURE 4.15 – Nouveau chemin solution proposé à l'utilisateur à la suite du redémarrage

Cette procédure de redémarrage est utilisée dans deux cas différents :

Recherche d'un chemin totalement différent. Dans ce premier cas, l'utilisateur n'est en effet pas satisfait de la trajectoire proposée et désire une approche globale différente. Dans ce cas, il désire lors du redémarrage de la partie temps réel du planificateur, que l'algorithme A* traduise son intention par la découverte d'une solution différente de celle qui est actuellement proposée. Pour cela, il doit modifier les données d'entrée de l'algorithme A* par une modification de la configuration courante de l'entité déplacée (cf figure 4.14).

Cependant, lors de la découverte d'une nouvelle zone de jet de point pour l'algorithme RDT, notre algorithme peut effectuer des jets permettant la convergence de l'algorithme dans des zones ne permettant pas de modifications notables de la trajectoire proposée. Par exemple, lorsque de nouveaux jets de points sont concentrés près de configuration finale, ils ne permettent pas de faire croître les arbres de diffusion en suivant l'approche globale proposée par la solution de l'algorithme A*. Dans ce cas, l'utilisateur doit réaliser des phases de décrochage successives jusqu'à l'obtention d'un chemin plus adapté.

Cas d'un Faux positif Dans un deuxième cas, l'utilisateur est simplement en train de réaliser un lissage de la trajectoire proposée par le planificateur. Dans ce cas, lors du redémarage de la partie temps réel du planificateur, le nouveau chemin solution de l'algorithme A* est inclu dans la solution précédente. La trajectoire nouvellement synthétisée par l'algorithme RDT est alors très proche du chemin précédent.

Afin de caractériser cette opération de lissage, la configuration de l'entité, capturée à l'instant du décrochage de la trajectoire, est ajoutée par un jet de cette configuration dans l'arbre initial. Ce nouveau jet permet d'inclure les décrochages successifs comme points de passage dans l'arbre initial ce qui permet de construire des branches indépendamment de la méthode de jet classique.

4.3 Implémentation en C++

4.3.1 Organisation

Nous choisissons de diviser le planificateur en plusieurs BBs afin de garantir une modularité et une absence de latence dans la simulation à la fois lors de l'étape de planification et lors de la phase de guidage de la main de l'utilisateur. Les blocs illustrés en Figure 4.16 permettent de commander les fonctions d'un manager général dit de planification.

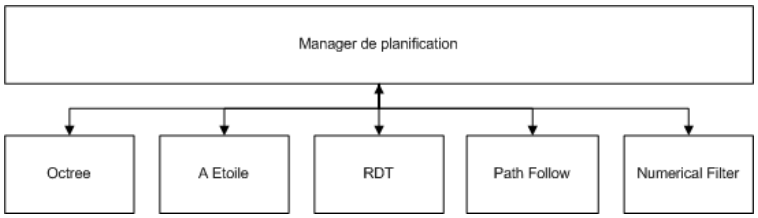


FIGURE 4.16 – Dépendance entre manager de planification et les BB

4.3.2 Planification de trajectoire

4.3.2.1 Manager de planification

Le manager de planification concentre toute les données et fonctions principales liées à la planification et exploite la structure de données du manager principal “VREMA Manager” décrit en partie 3. Pour plus de simplicité, il est intégré à la bibliothèque VREMA sous la forme d’un objet lié à une phase assemblage de pièces ou de sous ensembles illustré en figure 4.17. Cet objet est instancié au début d’une phase d’assemblage, ce qui implique la construction d’un arbre octal (classe “Octree”). Pour chaque trajectoire d’objet à planifier, un planificateur A* noté “AStarSolver” et une cartographie vide “RoadMapRRT” sont instanciés. Lorsqu’une trajectoire 6D est disponible dans la cartographie RDT, une classe “VREMAPath6D” est implémentée pour la stocker. Cet objet est alors utilisé pour calculer un effort de guidage de l’utilisateur vers celle ci.

4.3.2.2 Octree

L’implémentation de l’octree est réalisée au moyen de 2 différentes classes :

- la première, “Octree”, concentre les données et les fonctions principales dédiées à des manipulations globales sur l’arbre octal.
- la deuxième, “OctreeNode”, permet le stockage d’informations locales et des opérations unitaires sur les noeuds de l’arbre.

Cette organisation permet une implémentation aisée des BB permettant la création et le rafraîchissement d’un arbre octal. Afin d’optimiser les fonctions de recherche et de manipulation des noeuds de l’arbre, un système de référencement [117] est ajouté aux attributs de la classe “OctreeNode”. De plus, pour faciliter la fonction de rafraîchissement à la suite du déplacement d’une pièce, une liste des entités en collision avec le noeud est ajoutée à ses attributs. Ces diverses fonctions sont implémentées dans deux BBs :

- le BB “Create Octree” qui permet de lancer la phase de pré-calcul initiale pour une phase d’assemblage,
- le BB “Refresh Octree” qui permet d’effectuer le rafraîchissement de l’arbre octal en spécifiant la pièce mise en position finale et la nouvelle pièce à manipuler.

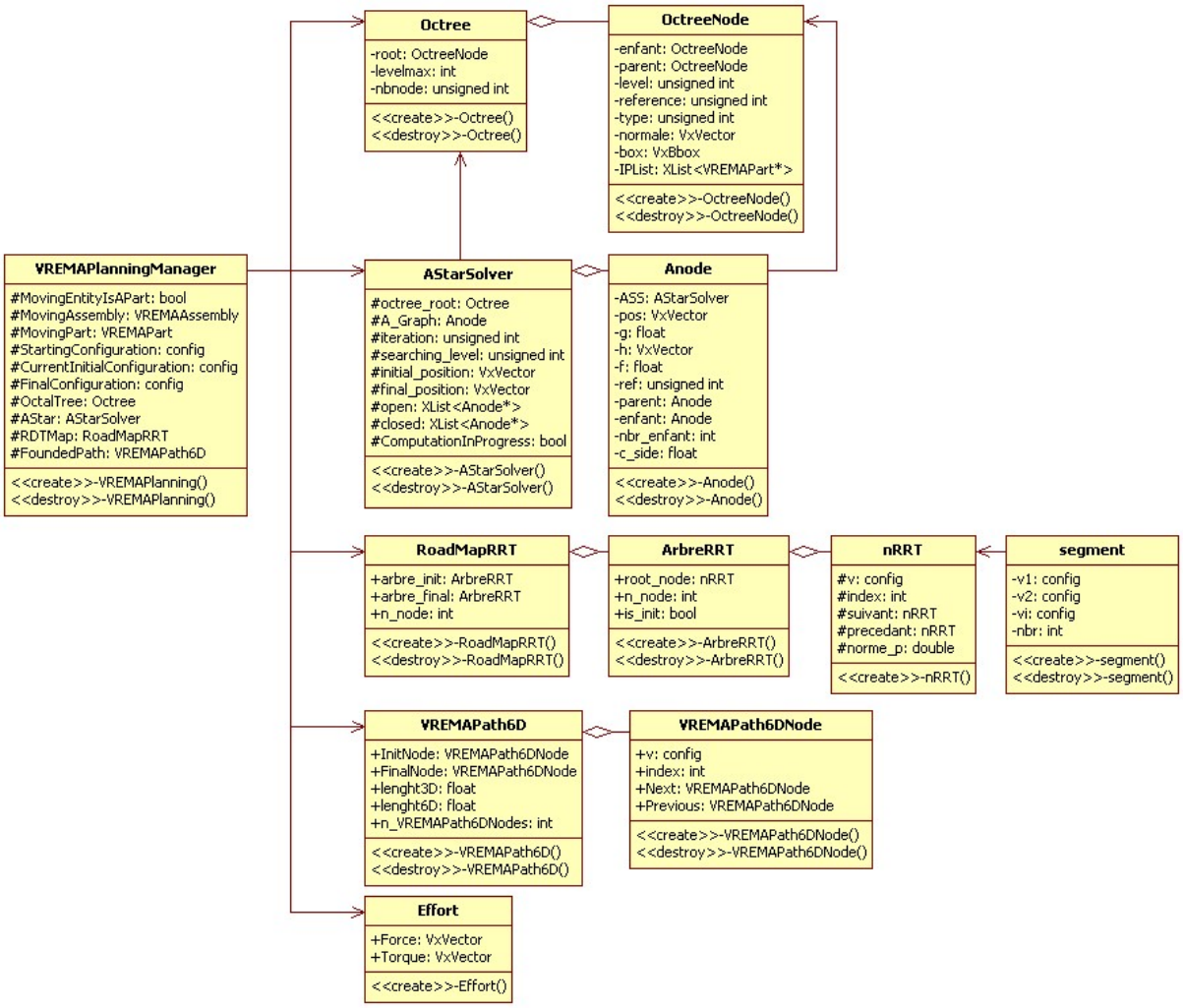


FIGURE 4.17 – Structure orientée objet regroupant les données et les fonctions pour la planification interactive d'une trajectoire

4.3.2.3 A*

L'implémentation de la méthode A* est réalisée suivant le même modèle que l'octree :

- Une première classe, "AStarSolver", permet de conserver les données et fonctions globales liées à la planification A*
- Une classe, "ANode", permet de décrire les noeuds du graphe ainsi que les fonctions de manipulation de ses noeuds. Étant lié aux cubes de l'arbre octal dans le processus de calcul de l'algorithme A*, chaque noeud ANode est lié à un noeud de type OctreeNode.

Le graphe stocké dans la classe A* est renseigné au fur et à mesure de l'avancée de ses itérations de manière à éviter de stocker trop d'informations. Dans la même optique, les noeuds de ce graphe sont conçus pour contenir uniquement les informations propres à l'algorithme A* telles que les caractéristiques : $f(n)$, $g(n)$ et $a(n)$. De manière à minimiser l'occupation de mémoire, une liaison par pointeur est réalisée entre les

objets de classe “ANode” et “OctreeNode” qu’ils représentent. Cette liaison permet de calculer rapidement la norme $d(n, n+1)$ et la taille $s(n)$ d’un cube n considéré. De plus, l’utilisation des références standardisées dans la construction de l’arbre octal permet une construction rapide de la liste des cubes adjacents au cube n considéré. Les fonctions de l’objet “AStarSolver” instancié dans le manager de planification sont utilisées par l’intermédiaire d’un BB nommé “A Star”. Ce BB permet le lancement d’un calcul rapide du volume englobant 3D préalable à une planification de trajectoire 6D.

4.3.2.4 RDT

Toujours suivant ce même principe de décomposition, l’implémentation du RDT est réalisée par l’utilisation de 4 classes :

- la classe “RDT_Node” permettant de créer des noeuds d’arbre RDT,
- la classe “RDT_Segment” permettant de créer des segments d’arbre RDT réalisant l’interpolation entre deux noeuds,
- la classe “RDT_Tree” permettant de créer un arbre constitué de noeuds RDT,
- la classe “RDT_RoadMap” permettant de stocker les deux arbres du problème RDT bi-directionnel.

Les noeuds de l’arbre RDT permettent de stocker chaque configuration clef d’une trajectoire. Les objets de type “RDT_Segment” sont instanciés uniquement lors de tests de validation de la possibilité de raccordement entre 2 noeuds d’un arbre, de façon à minimiser la quantité de mémoire utilisée pour la cartographie. Afin de faciliter l’utilisation des fonctionnalités du RDT dans les BB de planification, des fonctions orientées utilisateur sont implémentées dans la classe “RDT_RoadMap”. Ces fonctions sont directement utilisées dans le BB “RDT Planner”. Ce BB permet le lancement d’une planification de trajectoire 6D consécutive à l’exécution du BB “A Star”. Cette planification est réalisée de manière itérative en ajoutant successivement des noeuds à la cartographie “RDT_RoadMap” jusqu’à la découverte d’une solution. Dans une dernière étape, la trajectoire est convertie depuis la cartographie dans une instance d’objet de classe “VREMAPath6D”.

4.3.3 Génération de commande d’efforts artificiels pour l’interface haptique

Pour commander en forces artificielles l’interface haptique, deux BB, illustrés sur la figure 4.18, sont implémentés : “Path Follow” et “Numerical Filter”. De manière analogue aux BB précités, ils dépendent du manager de planification.

4.3.3.0.1 Le BB “Path Follow”

Ce BB calcule à partir de la configuration courante de la pièce manipulée par l’utilisateur et d’une trajectoire (“VREMAPath6D”) une commande de guidage. Le calcul de cette commande de guidage peut être réglée dans le script Virtools par des paramètres locaux des BB fixant les valeurs des constantes K_F, K_c, F_M et C_M. Il permet également de détecter si l’utilisateur est en phase d’approche ou de suivi de la trajectoire au moyen d’un drapeau de contexte et le cas échéant si l’utilisateur cherche à suivre un chemin différent de celui qui lui est proposé. Le torseur à exercer est alors calculé en fonction du contexte détecté à l’aide des formules établies en partie 4.2.3.2.3.

4.3.3.0.2 Le BB “Numerical Filter”

Ce BB réalise le filtrage des commande fournies à chaque rafraîchissement d’image par le BB décrit précédemment. Le filtrage est réalisé en utilisant une file FILO à longueur variable implémentée sur le modèle d’une liste chaînée stockée dans le manager de planification. A chaque exécution du BB, un décalage est effectué dans la file FILO permettant le stockage du torseur calculé au moyen des formules décrites en partie 4.2.3.2.2.

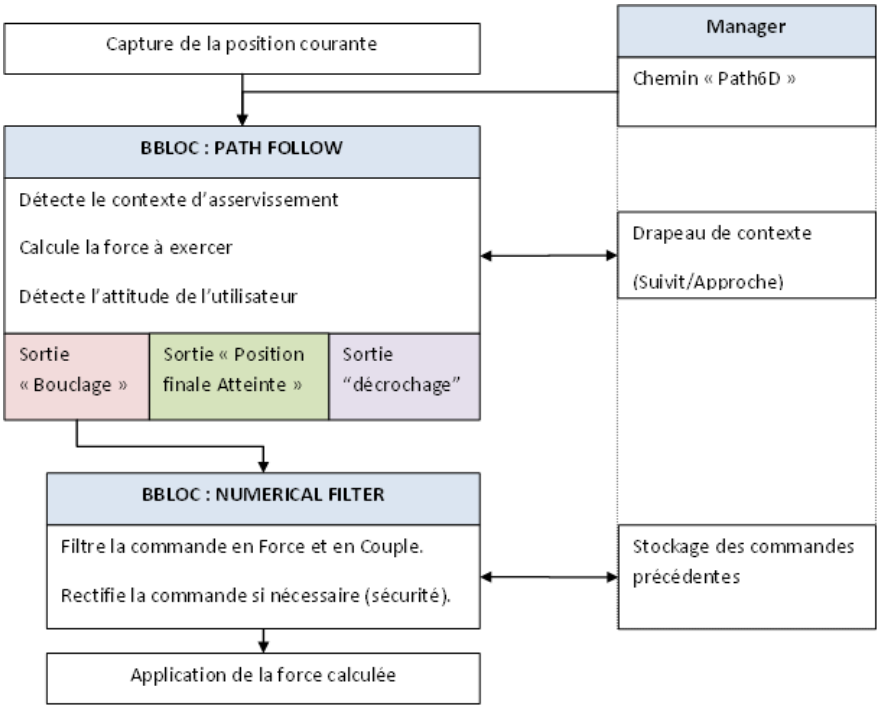


FIGURE 4.18 – Schéma fonctionnel des BBs de guidage

4.3.3.0.3 Scripting du guidage dans Virtools

La force de guidage calculée par le BB “Numerical Filter” est alors appliquée au niveau du bras haptique pendant la manipulation en tant que force artificielle en utilisant le BB “Add Force To Objet” disponible dans le plug in IPP. La programmation dans les scripts Virtools du calcul de la commande et de son application à travers plusieurs BB permet une alternance régulière entre le calcul et l’affichage ce qui garantit une absence de latence et donc une interaction en temps réel.

4.4 Tests comparatifs

Du fait du caractère interactif de la méthode que nous proposons, il est difficile de comparer l’efficacité, en terme de temps de calcul, de la planification interactive par rapport à la planification classique. Dans

cette partie, nous nous proposons de comparer les vitesses de convergence de notre planificateur et d'un planificateur de type RRT afin de valider le fait que les modifications effectuées le rendent utilisable en environnement interactif.

4.4.1 Scènes de test utilisées

Après l'implémentation, une série de tests comparatifs sont réalisés sur différentes scènes illustrant des cas réalistes de manipulation haptique pour le montage et le démontage de modèles CAO. Ces tests sont réalisés sur le PC supportant le monde virtuel : équipé de deux processeurs 3.73 GHz et de 2 Go de RAM. Les résultats obtenus sont alors calculés à partir de moyennes réalisés sur 50 simulations.

Deux scènes de test sont présentées ici :

- La première permet de tester la vitesse de convergence dans le cas d'une scène d'assemblage sur poste de travail. Ce type de contexte comporte comme difficulté principale de larges zones d'espace libre. La scène contient 10 objets issus de la CAO représentant des pièces constituant un convertisseur de puissance Alstom constituant un total de 231000 facettes (cf 4.19).
- La seconde permet de simuler un cas de maintenance d'un convertisseur de puissance. Ce type de simulation permet de réaliser des opérations de planification dans un environnement comportant un grand nombre d'objets (16 816 objets soit 3 878 806 facettes) et des passages étroits (cf figure 4.20).

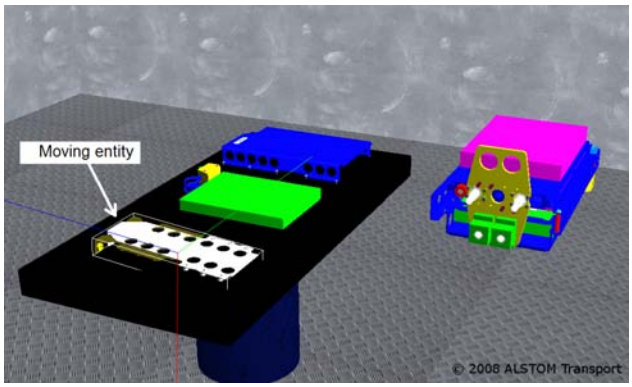


FIGURE 4.19 – Scène d'assemblage

4.4.2 Résultats obtenus

4.4.2.1 Scène de montage

Dans cette scène, les calculs de détection de collision sont coûteux en temps de calcul lorsque l'objet testé est très proche des obstacles. L'algorithme RRT classique utilisant une méthode de jet de configurations

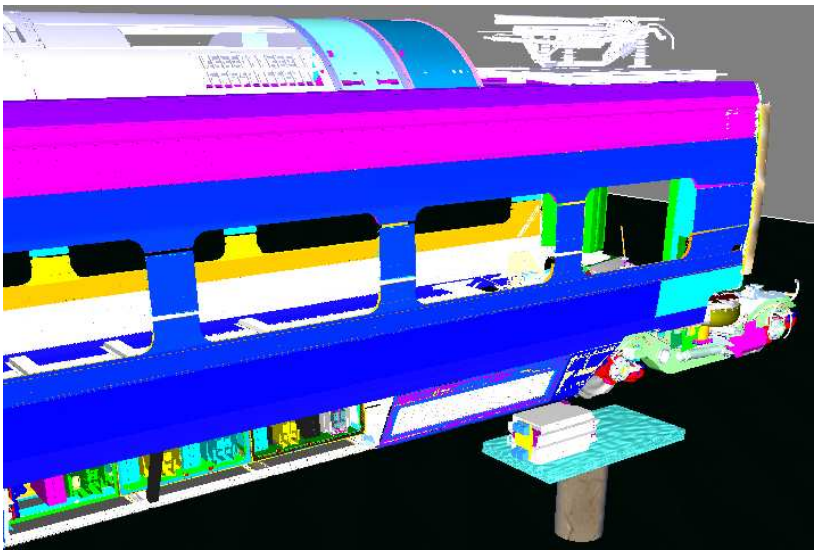


FIGURE 4.20 – Scène de maintenance

aléatoire [86], le temps de calcul nécessaire aux opérations de détection de collision réalisées lors de chaque expansion d'arbre est imprévisible car il dépend de la distance aux obstacles comme illustré par le large écart type du tableau 4.1.

	Nombre de Jets	Temps de calcul
Moyenne	5.1	3375 ms
Écart Type	3.7	3456 ms

TABLE 4.1 – Simulation d'assemblage : temps de calcul et nombre de jets de points requis par l'algorithme RRT

D'après le temps de calcul décrit dans le tableau 4.2, la méthode A* implémentée permet en moyenne de calculer un passage en 3d dans un intervalle de temps compatible avec une simulation en temps réel c'est à dire inférieur à 17 ms (60 Hz).

	Temps de calcul
Moyenne	15 ms
Écart Type	3.5 ms

TABLE 4.2 – Simulation d'assemblage : temps de calcul de l'algorithme A*

En utilisant le résultat de la méthode A* comme contrainte pour la méthode de jet de points de l'algorithme RDT, le temps d'exécution de la méthode et le nombre de jets de points requis sont bien plus faibles et réguliers que dans le cas de l'algorithme RRT comme illustré par le tableau 4.3.

	Nombre de Jets	Temps de calcul
Moyenne	1.6	82 ms
Écart Type	0.9	28 ms

TABLE 4.3 – Simulation d'assemblage : temps de calcul et nombre de jets de points requis par l'algorithme RDT

Dans ce dernier cas de figure, la contrainte du jet de configuration appliquée au RDT permet de réaliser des tests de collision dans des zones éloignées des obstacles ce qui réduit le temps de calcul. La trajectoire trouvée présente également l'avantage de permettre un guidage de l'utilisateur dans des zones faiblement contraintes, ce qui facilite la manipulation haptique. Néanmoins, les trajectoires générées par ce planificateur ne sont pas lisses du fait de leur mode de calcul comme illustré sur la figure 4.21. Ce problème peut être résolu par une étape supplémentaire de lissage ou encore par une modification des termes de pondération μ_{pos} et μ_{rot} utilisés pour le calcul de la norme 6d dans l'algorithme RDT.

On remarque également que, lors du guidage de la manipulation, l'utilisateur ajoute, tout en restant dans les limites fixées par la définition de la zone de suivi, une force légèrement résistante au guidage. Il réalise ainsi une opération de lissage qui permet d'éviter les point anguleux de la trajectoire proposée. La combinaison de ces deux forces permet alors de produire une trajectoire finale plus lisse et donc plus naturelle que celle proposée par l'algorithme RDT.

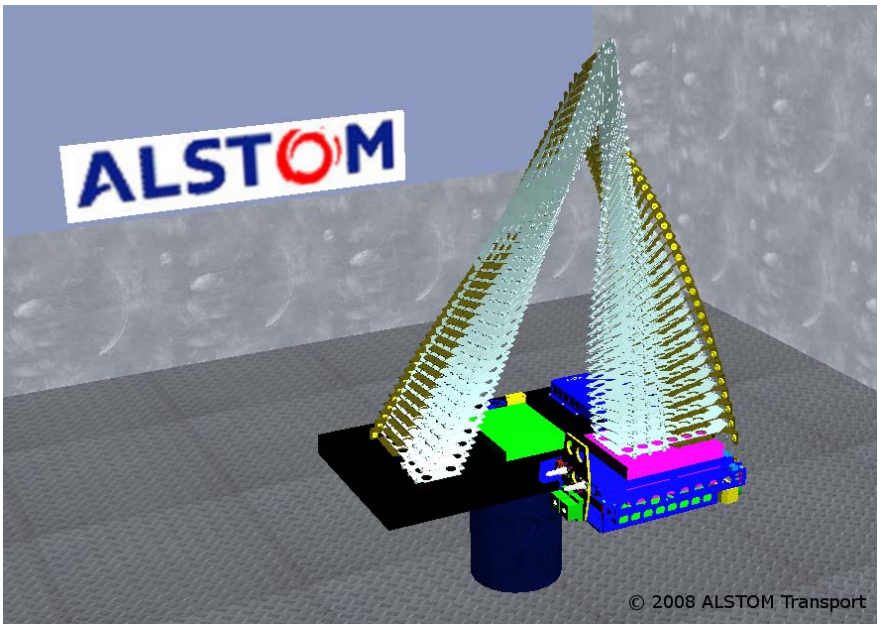


FIGURE 4.21 – Simulation de montage : exemple de trajectoire calculée par l'algorithme RDT en utilisant la solution de l'algorithme A*

4.4.2.2 Scène de maintenance

Tout comme dans la scène de montage, l'algorithme RRT nécessite un grand nombre de jets de configuration afin de trouver une trajectoire solution. On remarque également sur le tableau 4.4 un temps de calcul très élevé du au temps nécessaire aux opérations de détection de collision dans les passages fortement contraints de cet environnement.

	Nombre de Jets	Temps de calcul
Moyenne	593	12 470 ms
Écart Type	270	5895 ms

TABLE 4.4 – Simulation de maintenance : temps de calcul et nombre de jets de points requis par l'algorithme RRT

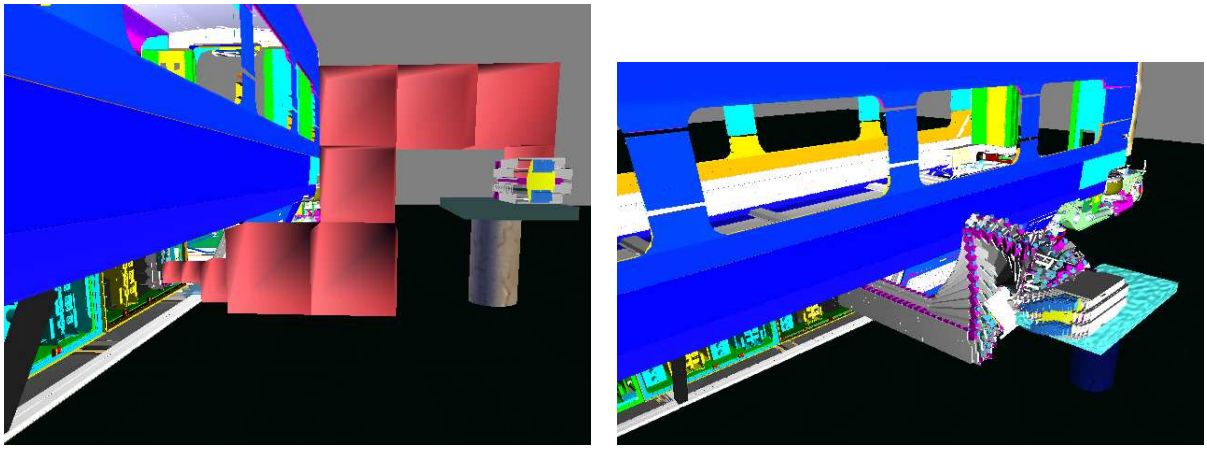
Dans ce cas complexe, l'algorithme A* réalise des opérations de subdivision supplémentaires (explicité en section 4.2.2.2) afin de satisfaire la contrainte de position finale libre permettant son démarrage. Ces opérations supplémentaires s'ajoutent au temps de calcul ce qui le rend plus lent que dans le cas précédent comme illustré sur le tableau 4.5. Grâce à cette opération supplémentaire, l'algorithme permet de calculer une zone de jet de configurations optimisée pour la convergence de l'algorithme RDT (cf Figure 4.22a).

Néanmoins, ces opérations pratiquant des modifications locales au niveau de l'arbre octal ne sont effectuées que lors de la première exécution de l'algorithme A*. Lors d'opérations de redémarrage déclenchées par l'utilisateur, la charge de calcul de l'algorithme A* est alors réduite à la simple opération de planification dans un espace discret. Son temps d'exécution est alors considérablement réduit, ce qui nous permet d'obtenir des résultats comparables au cas de la scène de montage.

	Temps de calcul
Moyenne	1020 ms
Écart Type	240 ms

TABLE 4.5 – Simulation de maintenance : temps de calcul requis par la première exécution de l'algorithme A*

L'utilisation du résultat de l'algorithme A* permet à l'algorithme RDT de converger beaucoup plus vite qu'un RRT standard, ce que l'on peut constater par la comparaison des résultats obtenus indiqués dans les tableaux 4.6 et 4.4 : on observe une réduction de l'écart type du temps de calcul ainsi que du nombre de jet de configurations nécessaires à la convergence vers un chemin solution.



(a) Simulation de maintenance : exemple de résultat obtenu par l'algorithme A*

(b) Simulation de maintenance : exemple de résultat obtenu par l'algorithme RDT en utilisant le résultat calculé par l'algorithme A*

FIGURE 4.22 – Simulation de maintenance

	Nombre de Jets	Temps de calcul
Moyenne	19	720 ms
Écart Type	23	1300 ms

TABLE 4.6 – Simulation de maintenance : temps de calcul et nombre de jets de points requis par l'algorithme RDT

La trajectoire générée, illustrée en figure 4.22b, n'est pas optimale en terme de longueur de chemin, mais permet grâce à un asservissement haptique de transmettre à l'utilisateur des couples et des forces artificielle le guidant dans un espace libre d'obstacles ce qui améliore l'ergonomie de manipulation. De plus, comme dans le cas précédent, la trajectoire finale suivie par l'utilisateur est lissée naturellement grâce à une composition entre la force de guidage et une résultante exercée par l'utilisateur.

4.5 Conclusion

Notre motivation à coupler des planificateurs automatiques et un utilisateur humain dans une simulation de réalité virtuelle dédiée au montage et au démontage de modèles 3D issus de la CAO nous a permis de définir une architecture de communication basée sur un interfaçage visuo-haptique. Cette architecture permet à un utilisateur d'être assisté par un planificateur automatique de son choix pendant la réalisation de sa tâche. Suivant la difficulté du problème rencontré, il peut choisir entre une assistance à la manipulation fourni par un guidage faible issu d'une planification locale, ou un planificateur plus élaboré fournissant un guidage issu d'une approche globale dans un environnement complexe.

Ce dernier est exécuté en deux temps : au préalable, une étude de l'espace de travail est réalisée par la

création d'une cartographie basée sur un Octree déséquilibré. Ensuite, un algorithme A* adapté permet de calculer un volume 3d servant à limiter la méthode de jet de configurations d'un algorithme RDT. Cette approche permettant l'utilisation d'informations globales dans l'exécution d'un algorithme probabiliste permet de réduire la complexité théorique du problème de planification de trajectoire. Néanmoins, la réduction de cette complexité, effectuée par une réduction automatisée de l'espace de recherche ne permet pas de garantir la convergence probabiliste de l'algorithme. En pratique, ce défaut, fréquemment rencontré dans les environnement 3d fortement encombrés, est alors contourné par l'utilisation de la décision de l'utilisateur dans le processus de planification.

Le résultat (ou le non résultat) de l'opération de planification est transmise à l'utilisateur sous la forme d'une force artificielle de guidage vers, puis sur la trajectoire 6d calculée. L'asservissement réalisé laisse alors une certaine liberté à l'utilisateur qui peut choisir de le respecter en se laissant guider par l'effort de guidage ou, au contraire, de résister à cet effort en s'éloignant de la trajectoire proposée. En effet, le chemin solution calculé par le planificateur permet de résoudre le problème théorique mais ne satisfait pas toujours l'utilisateur. Ce dernier peut alors, par l'intermédiaire de son attitude dans la manipulation de l'objet qu'il déplace, déclencher un redémarrage rapide du planificateur afin de définir automatiquement un espace de recherche permettant la convergence du RDT vers un chemin solution lui paraissant plus adapté. Dans le cadre de cette interaction, le chemin solution de ce planificateur "interactif" est créé par le déplacement de l'entitée manipulée au moyen de l'interface haptique. Ce déplacement provient de la somme d'un effort de guidage calculé à partir d'un chemin théorique trouvé par le planificateur automatique et d'un effort résistant exercé par l'utilisateur. De plus, l'utilisation d'efforts artificiels pour le guidage ainsi que la détection géométrique de l'intention de l'utilisateur permettent une interaction très "naturelle" entre l'humain et le planificateur et ainsi une efficacité accrue dans les tâches de montage et de démontage. Cependant, cette impression reste à quantifier en terme d'économie de temps ce qui peut nous amener à améliorer ou tout simplement personnaliser la méthode de détection d'intention.

Après une série de tests comparatifs entre notre solution et un planificateur RRT standard, il est apparu que notre approche permettait de réduire les temps de calcul nécessaires pour la découverte d'une trajectoire 6D solution dans les environnement virtuels. De plus, cette nouvelle approche permet de fiabiliser le planificateur en réduisant également les écarts types du temps de calcul ainsi que du nombre de jets de points nécessaires à la convergence. Cependant, le temps d'exécution reste encore très dépendant de la complexité de la scène virtuelle. On notera que dans le cadre de l'utilisation d'objets CAO très complexes sur notre plateforme, une latence due au calcul de trajectoire est perçue par l'utilisateur, ce qui détériore d'une part la qualité de l'interaction entre utilisateur et planificateur et d'autre part l'immersion de l'utilisateur dans le monde virtuel.

L'analyse de la répartition du temps de calcul utilisé par l'algorithme RDT indique que cette latence est essentiellement concentrée au niveau des opérations de détection de collision dans des zones fortement encombrées d'objets très complexes. Nous pouvons donc conclure que cette latence peut être réduite voire éliminée par l'utilisation d'un détecteur de collision plus performant que celui proposé par Virtools. Une autre piste pour la résolution de ce problème consiste en une segmentation des opérations de détection de collision permettant ainsi de répartir le temps de calcul nécessaire entre deux raffraichissements d'image de

la simulation.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion générale

Le travail synthétisé dans ce document s'intègre dans la démarche d'Alstom transport qui tend à expérimenter virtuellement différents processus du PLM et fait appel à des composants logiciels permettant d'interfacer, dans un contexte industriel, les outils de la réalité virtuelle avec des techniques de planification automatique de trajectoire.

Il s'intéresse plus particulièrement à la possibilité d'assister un opérateur immergé grâce à des moyens de calcul automatique de chemin dans des tâches de manipulation récurrentes lors de la simulation de processus industriels.

La réalisation de ces objectifs se traduit par deux contributions :

- la définition d'une architecture (VREMA) qui permet d'intégrer simplement différents scénarios interactifs du PLM,
- la conception et la réalisation d'un protocole qui permet de coordonner des techniques de planification automatique et des capacité d'interaction temps réel dans des simulations visuo-haptiques.

Grâce à une architecture modulaire basée sur une bibliothèque orientée objet nommée Virtual Reality Environment for Manufacturing Applications (VREMA), l'ensemble des fonctionnalités implémentées est accessible par une programmation de BBs et leur intégration dans des graphes de comportement. Ces BBs, modifiables si besoin, sont alors intégrables dans la simulation sous la forme de graphes de comportement. Dans sa version actuelle, cette bibliothèque permet, grâce à une série de BBs déjà implémentés, la création rapide de simulations industrielles répondant aux exigences de la société Alstom Transport S.A. Ces différents BBs permettent l'intégration directe de composants issus de la CAO au format 3dxml ou JT dans l'environnement, ainsi que leur restructuration en vue d'une utilisation dans un environnement fortement interactif.

Dans le cadre de simulations de manipulations industrielles pour l'étude de process d'assemblage ou de maintenance, cette simulation prévoit l'utilisation d'une assistance par le biais d'une interface visuo-haptique.

Dans ce contexte, diverses métaphores ont été crées pour permettre une assistance personnalisable suivant les difficultés rencontrées par l'utilisateur. Dans le cadre d'un montage simple, un système de guidage basé sur une interaction entre l'utilisateur est un planificateur local utilisant une méthode de potentiel permet de guider l'utilisateur vers la position finale. Dans des cas plus complexes, un planificateur probabiliste interactif est implémenté de manière à guider l'utilisateur selon une trajectoire calculée automatiquement. Grâce à un système d'évaluation du comportement de l'utilisateur vis à vis de la trajectoire proposée, le planificateur probabiliste permet des phases de redémarrage partiel et ainsi une construction interactive du chemin solution. Les différents tests réalisés pour tester l'efficacité de ce type de planificateur dans des environnements interactifs ont montré de bons résultats dans le cadre de scènes de montage de modèles issus de la CAO. Néanmoins, les résultats démontrent également des cas de latence lors de manipulations dans des environnements complexes comportant un grand nombre d'objets et de facettes.

5.2 Perspectives

Le travail réalisé au cours de ce projet pose les bases de concepts utilisables dans le cadre du "PLM interactif". Il révèle également un certain nombre de challenges technologiques ou conceptuels. Nous avons souligné dans ce projet que dans le cadre d'application interactives, le respect d'une interaction en temps réel demeure la contrainte principale. Pour s'assurer du respect de celle ci à l'avenir, plusieurs aspects sont à considérer et différentes pistes restent à explorer :

Un challenge technologique : Le traitement paralléle

Parallélisation du code du projet VREMA

L'un des différents moyens de diminuer le temps de calcul nécessaire à la planification consiste à exploiter l'architecture multi coeurs des machines modernes par une parallélisation du programme. Dans cette optique, la bibliothèque C++ Boost [118] est d'ores et déjà disponible et incluse à la bibliothèque VREMA. Cependant, cette parallélisation peut s'avérer difficile en pratique du fait de l'intégration du code VREMA dans Virtools dev qui, lui, reste pour l'instant non parallélisé. De plus les gains de cette opération éventuelle seront limités du fait du grand nombre de processus déjà supportés par la machine hébergeant la simulation.

Utilisation de plusieurs machines et parallélisation des processus

L'utilisation simultanée de système de capture de mouvements, d'interface haptique, de gants de donnés, de programme de planification ou encore de simulation physique pose le problème de la parallélisation des processus. Ce dernier peut être résolu par l'utilisation de ces différentes fonctionnalités sur une grappe de machines organisées en réseau local. Ce type d'organisation, déja implémenté dans le cadre des SAS Cubes et de plus en plus couramment utilisé dans la plupart des plateformes de réalité virtuelle doit être envisagé pour les futures améliorations de la plateforme du laboratoire.

L'évolution constante des formats 3d

Les choix des formats 3d pour l'importation d'objets dans la simulation doivent être remis en cause du fait de leur constante évolution. Le format JT prévoit, dans les prochaines version, un stockage de surfaces exactes permettant de décrire les objets 3d. Ce type d'extension déja prévu dans la structure de donnée de VREMA doit être actualisé afin de pouvoir intégrer de plus en plus d'informations issues du PLM comme par exemple des données d'industrialisation comme des tolérances de montage. A chaque évolution des formats, ces informations supplémentaires doivent être importées dans la structure de données et utilisées afin de rendre la simulation interactive la plus réaliste possible.

Challenges scientifiques

Amélioration des modifications apportées au planificateur

Afin de garantir une absence de latence dans la simulation, les différentes opérations d'expansion d'arbre réalisées dans la phase de construction de la cartographie RDT peuvent être bornées de manière à garantir une restitution des ressources entre chaque calcul de comportement. Dans cette optique, des fonctions d'estimation du coût, en terme de temps de calcul, des opérations de détection de collision ainsi que des possibilités d'interruption de la fonction d'expansion d'arbre doivent être intégrées au programme déjà existant afin de cadencer le calcul de planification de trajectoire avec la fréquence de rafraîchissement d'image de la simulation de manière à garantir une absence totale de latence. Cependant ce type de méthode ne permet pas de résoudre d'éventuels problèmes de latence lorsqu'un test de détection de collision dépasse le temps imparti au calcul de comportement. Dans un contexte où les modèles CAO chargés dans les simulations interactives sont de plus en plus complexes, en terme de nombre de triangles, ce type de problème implique une reprogrammation du détecteur de collision actuellement utilisé ou l'intégration d'un détecteur plus performant.

Utilisation de surfaces exactes

L'utilisation de ces surfaces permet d'améliorer les fonctions de détection de collision disponible dans Virtools en exploitant la structure de donnée disponible dans VREMA. Ce nouveau détecteur de collision devra exploiter la hiérarchie d'objets présente dans Virtools, mais aussi la segmentation de maillage et l'identification de surfaces exactes réalisée dès l'opération de chargement de modèle dans l'environnement virtuel. Cette amélioration permet de réduire le temps de calcul de la détection de collision utilisé dans l'algorithme de planification mais surtout rend ce temps indépendant de la résolution du maillage utilisé pour l'affichage.

De plus, grâce à la définition d'un nouveau détecteur de collisions, l'algorithme de planification peut effectuer une analyse plus fine des contacts entre objets de manière à optimiser les heuristiques de dégagement par la détection des degrés de liberté de l'objet manipulé dans des positions contraintes. En effet, grâce à cette analyse des contacts détectés dans les positions fortement contraintes, il est possible de déterminer avec précision les cinématiques de démontage complexes et ainsi accélèrer la convergence de l'algorithme de planification de chemin.

Modification des stratégies de planification

La stratégie d'accélération de la méthode de planification présentée dans ce document repose sur une décomposition spatiale de l'espace libre. Cependant, d'autre stratégies de recherche peuvent être envisagées et ajoutée à la méthode existante comme par exemple le suivi de surfaces de contact par glissement ou déformation, très fréquemment utilisé dans les cas d'insertion d'une pièce mécanique dans sa configuration finale.

Planification de mouvements d'humains virtuels

Dans un premier temps, le planificateur décrit au chapitre 4 peut être utilisé pour la planification de la trajectoire d'une trajectoire en 6D en ajoutant à l'objet mobile des corps rigides représentant la position des mains de l'utilisateur. Cette idée peut ensuite être étendue par une représentation articulée des bras d'un humain virtuel manipulant l'objet mobile et donc l'implémentation à un plus haut niveau d'un planificateur de mouvements. Cette nouvelle contrainte augmente cependant le nombre de dimensions de l'espace de recherche du planificateur RDT et donc la complexité du problème de planification. Néanmoins, la stratégie de réduction de l'espace de recherche d'un planificateur probabiliste pour une utilisation en environnement interactif présentée dans ce document peut être étendue à la planification de mouvements d'humanoïdes.

De plus, pour simuler une manipulation réaliste, le planificateur de mouvements doit également permettre une re-configuration des points d'attache des mains de l'utilisateur sur la pièce manipulée. Cette extension ouvre ainsi la porte à des études d'ergonomie de montage et de démontage basé sur l'analyse du comportement d'un humanoïde virtuel réalisant en temps réel l'opération de manipulation d'une pièce dans un environnement virtuel sur une trajectoire 6D calculée de manière interactive.

Bibliographie

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion : Theory, Algorithms, and Implementations*. Cambridge, MA : MIT Press, June 2005. 1.5, 2.2.2.1, 2.2.2.2, 2.2.3.1, 2.2.4.1

[2] [Online]. Available : www.siemens.com/plm 1.3b, 2.1.3.3.4

[3] A. Lecuyer, "Contribution á l´ étude des retours haptique et pseudo-haptique et de leur impact sur les simulations d´ opérations de montage / démontage en aéronautique," Ph.D. dissertation, Université Paris XI, réalisée en collaboration avec EADS CCR, INRIA et LRP, 2001. 1.5, 2.1.2.2.2

[4] P. Fuchs, G. Moreau, B. Arnaldi, and P. Guitton, *Le traité de la Réalité Virtuelle, Volume 1 : L´ homme et l'environnement virtuel*, ser. Collection sciences mathematiques et informatiques. Les Presses de l'Ecole des Mines, 2006, vol. 1. [Online]. Available : http://www.ensmp.fr/Presses/consultation.php?livreplus=67--col1 2.1.1.1, 2.1.1.2, 2.2

[5] P. Fuchs, G. Moreau, and B. Arnaldi, *Le traité de la Réalité Virtuelle, Volume 2 : L´ interfaçage, l´ immersion et l´ interaction en environnement virtuel*, ser. Collection sciences mathematiques et informatiques. Les Presses de l'Ecole des Mines, 2006, vol. 2. [Online]. Available : http://www.ensmp.fr/Presses/consultation.php?livreplus=68--col1 2.1.1.1

[6] P. Fuchs, G. Moreau, B. Arnaldi, P. Guitton, M.-P. Cani, F. Neyret, M. Parenthoen, and J. Tisseau, *Le traité de la Réalité Virtuelle, Volume 3 : outils et modeles informatiques des environnements naturels*, 3rd ed., ser. Collection sciences mathematiques et informatiques. Les Presses de l'Ecole des Mines, 2006, vol. 3. [Online]. Available : http://www.ensmp.fr/Presses/consultation.php?livreplus=70--col1 2.1.1.1

[7] P. Fuchs, G. Moreau, B. Arnaldi, P. Guitton, J. Bukhardt, D. Lourdeaux, and D. Mellet-d'Huart, *Le traité de la Réalité Virtuelle, Volume 4 : Les applications de la Réalité Virtuelle*, third edition ed., ser. Collection sciences mathematiques et informatiques. Les Presses de l'Ecole des Mines, 2006, vol. 4, no. ISBN :2-911762-65-7. [Online]. Available : http://www-evasion.imag.fr/Publications/2006/CNPT06 2.1.1.1, 3.1

[8] P. Fuchs, G. Moreau, B. Arnaldi, and P. Guitton, *Le traité de la Réalité Virtuelle, Volume 5 : Les humains virtuels*, 3rd ed., ser. collection sciences mathematiques et informatiques. Les Presses de l'Ecole des Mines, 2009, vol. 3. [Online]. Available : http://www.ensmp.fr/Presses/consultation.php?livreplus=123--col1 2.1.1.1

[9] [Online]. Available : http://www.irisa.fr/bunraku/home_html 2.1.1.1.1

[10] S. Gibet and P.-F. Marteau, "Analysis of human motion, based on the reduction of multidimensional captured data — application to hand gesture compression, segmentation and synthesis," in *AMDO '08 : Proceedings of the 5th international conference on Articulated Motion and Deformable Objects*. Berlin, Heidelberg : Springer-Verlag, 2008, pp. 72–81. 2.1.1.1.1

[11] F. Multon, R. Kulpa, and B. Bideau, "Mkm : A global framework for animating humans in virtual reality applications," *Presence : Teleoperators & Virtual Environments*, vol. 17, no. 1, pp. 17–28, 2008. [Online]. Available : http://www.mitpressjournals.org/doi/abs/10.1162/pres.17.1.17 2.1.1.1.1

[12] N. Pronost, "Définition et réalisation d'outils de modélisation et de calcul de mouvement pour des humanoïdes virtuels," Ph.D. dissertation, University of Rennes 1, Dec. 2006. 2.1.1.1.1

[13] N. Pronost and G. Dumont, "Dynamics-based analysis and synthesis of human locomotion," *The Visual Computer*, vol. 23, no. 7, pp. 513–522, Jul. 2007. 2.1.1.1.1

[14] [Online]. Available : http://vrlab.epfl.ch/ 2.1.1.1.1

[15] P. Glardon, "On-line locomotion synthesis for virtual humans," Ph.D. dissertation, Lausanne, 2005. [Online]. Available : http://library.epfl.ch/theses/?nr=3431 2.1.1.1.1

[16] T. Abaci, "Object manipulation and grasping for virtual humans," Ph.D. dissertation, Lausanne, 2006. [Online]. Available : http://library.epfl.ch/theses/?nr=3474 2.1.1.1.1

[17] E. Arbabi, "Contact modeling and collision detection in human joints," Ph.D. dissertation, Lausanne, 2009. [Online]. Available : http://library.epfl.ch/theses/?nr=4421 2.1.1.1.1

[18] J. Ciger, "Collaboration with agents in vr environments," Ph.D. dissertation, Lausanne, 2005. [Online]. Available : http://library.epfl.ch/theses/?nr=3350 2.1.1.1.1

[19] H. Grillon, "Simulating interactions with virtual characters for the treatment of social phobia," Ph.D. dissertation, Lausanne, 2009. [Online]. Available : http://library.epfl.ch/theses/?nr=4466 2.1.1.1.1

[20] N. Pallamin, B. Pavard, M. C. Zamberlan, and V. Santos, Eds., *Interactions professionnelles en univers virtuel*, ser. Ergonomie de produits et des services. http ://www.puf.com : Presses Universitaires de France (PUF), 2009. 2.1.1.1.1, 3.1

[21] [Online]. Available : http://w3.uqo.ca/cyberpsy/ 2.1.1.1.1

[22] M. Dugas, K. Francis, and S. Bouchard, "Cognitive behavioural therapy and applied relaxation for generalized anxiety disorder : A time series analysis of change in worry and somatic anxiety," *Cognitive Behaviour Therapy*, vol. 38, no. 1, pp. 29–41. [Online]. Available : http://dx.doi.org/10.1080/16506070802473221 2.1.1.1.1

[23] D. H. Goh, R. P. Ang, and H. C. Tan, "Strategies for designing effective psychotherapeutic gaming interventions for children and adolescents," *Comput. Hum. Behav.*, vol. 24, no. 5, pp. 2217–2235, 2008. 2.1.1.1.1

[24] D. A. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa, and W. Stuerzlinger, "3d user interfaces : New directions and perspectives," *IEEE Comput. Graph. Appl.*, vol. 28, no. 6, pp. 20–36, 2008. 2.1.1.1.2

[25] F. Gosselin, C. Andriot, J. Savall, and J. Martín, "Large workspace haptic devices for human-scale interaction : A survey," in *EuroHaptics*, ser. Lecture Notes in Computer Science, M. Ferre, Ed., vol. 5024. Springer, 2008, pp. 523–528. 2.1.1.1.2

[26] [Online]. Available : http://www-list.cea.fr/ 2.1.1.1.2, 2.3.1

[27] [Online]. Available : http://www.haption.com/site/index.html 2.1.1.1.2, 2.1.2.2.1, 2.6a, 2.6b, 2.1.2.2.1

[28] M. Ortega and S. Coquillart, "Prop-based haptic interaction with co-location and immersion : an automotive application," Tech. Rep. cs.HC/0601025, Jan 2006. 2.1.1.1.2

[29] A. Lécuyer, S. Coquillart, I. Rocquencourt, A. Kheddar, P. Coiffet, and P. Richard, "Pseudo-haptic feedback : Can isometric input devices simulate force feedback ?" 2000. 2.1.1.1.2

[30] J. Sreng, A. Lécuyer, C. Mégard, and C. Andriot, "Using visual cues of contact to improve interactive manipulation of virtual objects in industrial assembly/maintenance simulations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1013–1020, 2006. 2.1.1.1.2

[31] J. Sreng, A. Lécuyer, and C. Andriot, "Using vibration patterns to provide impact position information in haptic manipulation of virtual objects," in *Proceedings of Eurohaptics 2008*, ser. Lecture Notes in Computer Science, M. Ferre, Ed., vol. 5024. Springer-Verlag, 2008, pp. 589–598. 2.1.1.1.2

[32] J. Sreng, "Contribution to the study of visual, auditory and haptic rendering of information of contact in virtual environments," Ph.D. dissertation, 2008. 2.1.1.1.2

[33] F. Schramm, F. Geffard, G. Morel, and A. Micaelli, "Calibration free image point path planning simultaneously ensuring visibility and controlling camera path," in *ICRA*. IEEE, 2007, pp. 2074–2079. 2.1.1.1.2

[34] M. Gautier, J. Sreng, and C. Andriot, "Influence of event-based haptic on the manipulation of rigid objects in degraded virtual environments," in *VRST*, S. N. Spencer, Y. Kitamura, H. Takemura, K. Kiyokawa, B. Lok, and D. Thalmann, Eds. ACM, 2009, pp. 249–250. 2.1.1.1.2

[35] M. Collet, F. Philippe, M. Chevaldonne, D. Rigaudiere, F. Merienne, F. Guillaume, and N. Chevassus, "Manipulation intuitive dóbjets virtuels à l áide dúne interface à deux mains," in *Virtual Concept*, 2002. 2.1.1.1.2

[36] M. Gautier and C. Andriot, "6dof haptic cooperation over large latency network with wave variables for virtual prototyping," in *ICRA*. IEEE, 2009, pp. 1086–1091. 2.1.1.1.2

[37] H. Hrimech and F. Merienne, "Evaluation of interaction metaphors for collaborative work in virtual immersion," in *Journées de l'Association Francaise de Réalité Virtuelle*, October 2007 2007, p. 198. 2.1.1.1.2

[38] C. Duriez, C. Andriot, and A. Kheddar, "A multi-threaded approach for deformable/rigid contacts with haptic feedback," in *HAPTICS*. IEEE Computer Society, 2004, pp. 272–279. 2.1.1.1.2

[39] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *CoRR*, vol. abs/0804.0561, 2008. 2.1.1.1.2

[40] D. Paillot, F. Rivet, F. Merienne, and M. Bonnet, "Training simulator to the handling of fire extinguisher," in *Virtual Concept*, November 2006 2006. 2.1.1.1.3

[41] Y. Bisheng, L. Qingquan, and L. Deren, "Building modeling for 3d city model," *Geo-Spatial Information Science*, vol. 2, pp. 109–114, 1999-12-01. 2.1.1.1.3

[42] K. Lee, "3d urban modeling and rendering with high resolution remote sensing imagery on mobile 3d and web 3d environments ; system architecture and prototype implementation," april 2007, pp. 1 –5. 2.1.1.1.3

[43] T. Muller, O. Prat, J. Roger, and J.-M. Sanchez, "Illumination photoréaliste interactive en environnement distant," in *Virtual Retrospect*, November 2003 2003, pp. 23–25. 2.1.1.1.3

[44] [Online]. Available : http://www.sensable.com/products-haptic-devices.htm 2.1.2.2.1

[45] C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, L. Wietzke, and G. Sommer, *Articulated Motion and Deformable Objects*, ser. Lecture Notes in Computer Science. Springer Berlin, 2008, ch. Dealing with Self-occlusion in Region Based Motion Capture by Means of Internal Regions, pp. 102 – 111. [Online]. Available : http://www.springerlink.com/content/9mu1142k6161461m 2.1.2.2.2

[46] X. Chen and D. James, "Camera placement considering occlusion for robust motion capture," Computer Graphics Laboratory, Stanford University, Tech. Rep., 2000. 2.1.2.2.2

[47] Y. Iwashita, R. Kurazume, T. Hasegawa, and K. Hara, "Robust motion capture system against target occlusion using fast level set method," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, may 2006, pp. 168 –174. 2.1.2.2.2

[48] [Online]. Available : http://www.irisa.fr/bunraku/OpenMASK/ 2.1.3.1

[49] D. Margery, B. Arnaldi, A. Chauffaut, S. Donikian, and T. Duval, "Openmask : Multi-threaded, modular animation and simulation kernel, kit : un bref survol," in *Virtual Reality International Conference VRIC02, Laval, France*, 2002. 2.1.3.1

[50] T. Duval, S. Morvan, P. Reignier, F. Harrouet, and J. Tisseau, "Arévi : une boîte à outils 3d pour des applications coopératives," *Calculateurs Paralléles*, vol. Numéro spécial sur la coopération, 1997. 2.1.3.1

[51] P. Reigner, F. Harrouet, S. Morvan, J. Tisseau, and T. Duval, "Arévi : A virtual reality multiagent platform," in *Lecture notes in computer science*, Springer, Ed., 1998. 2.1.3.1

[52] L. Arnal and P. Bourdot, "EVserveur : serveur hiérarchique dínformations pour environnements virtuels," LIMSI, Notes et Documents LIMSI n° 99-25, Décembre 1999. 2.1.3.1

[53] T. Damien, "Interaction naturelle en environnements immersifs - démonstrateur multimodal et validation sur des applications scientifiques," Ph.D. dissertation, Université Paris XI Orsay, 2003. 2.1.3.1

[54] [Online]. Available : http://a2.media.3ds.com/products/3dvia/3dvia-virtools/welcome/ 2.1.3.1, 2.1.3.2.2, 2.1.3.3.1, 3.2

[55] M. C. Lin, "Efficient collision detection for animation and robotics," Tech. Rep., 1993. 2.1.3.2.1, 2.1.3.2.1

[56] P. Jiménez, F. Thomas, and C. Torras, "3d collision detection : A survey," *Computers and Graphics*, vol. 25, pp. 269–285, 2000. 2.1.3.2.1

[57] S. Redon, "Algorithmes de simulation dynamique interactive d'objets rigides," Ph.D. dissertation, Université d'Evry-Val d'Essonne, 10 2002, Numero d'ordre EVRY : 02EVRY0009. [Online]. Available : http://tel.archives-ouvertes.fr/tel-00003580/en/ 2.1.3.2.1

[58] Z. Xinyu, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using Taylor models and temporal culling," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007, special Issue : Proceedings of the 2007 SIGGRAPH conference. 2.1.3.2.1

[59] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using voxel sampling," in *SIGGRAPH '05 : ACM SIGGRAPH 2005 Courses*. New York, NY, USA : ACM, 2005, p. 42. 2.1.3.2.1, 4.1.1

[60] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree : A hierarchical structure for rapid interference detection," *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 171–180, 1996. [Online]. Available : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.7796 2.1.3.2.1

[61] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi, "I-collide : An interactive and exact collision detection system for large-scale environments," in *In Proc. of ACM Interactive 3D Graphics Conference*, 1995, pp. 189–196. 2.19

[62] A. Wilson, E. Larsen, D. Manocha, and M. C. Lin, "Immpact : A system for interactive proximity queries on massive models," Tech. Rep., 1998. 2.1.3.2.1

[63] S. A. Gottschalk, "Collision queries using oriented bounding boxes," Ph.D. dissertation, 2000, director-Manocha, Dinesh and Director-Lin, Ming C. 2.1.3.2.1

[64] A. Wilson, E. Larsen, D. Manocha, and M. C. Lin, "Immpact : Partitioning and handling massive models for interactive collision detection," in *Eurographics*. Blackwell Publishers, 1999, pp. 319–329. 2.1.3.2.1

[65] S. Redon, A. Kheddar, and S. Coquillart, "Contact : arbitrary in-between motions for collision detection," in *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, 2001, pp. 106–111. 2.1.3.2.1

[66] S. Redon, "Continuous collision detection," in *Haptic Rendering : Foundations, Algorithms and Applications*, M. C. Lin and M. Otaduy, Eds. A. K. Peters, Ltd, Nov. 2008. 2.1.3.2.1

[67] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney, *Level of Detail for 3D Graphics*. New York, NY, USA : Elsevier Science Inc., 2002. 2.1.3.2.2

[68] T. K. Heok and D. Daman, "A review on level of detail," *Computer Graphics, Imaging and Visualization, International Conference on*, vol. 0, pp. 70–75, 2004. 2.1.3.2.2

[69] S. Gao, Y. Qi, X. Shen, and Y. Hu, "A realtime rendering framework of large dataset environment based on precomputed hlod," in *Digital Media and its Application in Museum & Heritages, Second Workshop on*, Dec. 2007, pp. 212–217. 2.1.3.2.2

[70] C. Erikson and D. Manocha, "Hierarchical levels of detail for fast display of large static and dynamic environments," Tech. Rep., 2000. 2.1.3.2.2, 2.22

[71] L. Hu, P. V. Sander, and H. Hoppe, "Parallel view-dependent refinement of progressive meshes," in *I3D '09 : Proceedings of the 2009 symposium on Interactive 3D graphics and games*. New York, NY, USA : ACM, 2009, pp. 169–176. [Online]. Available : http://dx.doi.org/10.1145/1507149.1507177 2.1.3.2.2, 2.21, 2.22

[72] H. Hoppe, "Progressive meshes," in *SIGGRAPH '96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1996, pp. 99–108. 2.1.3.2.2

[73] R. Pacanowski, M. Raynaud, X. Granier, P. Reuter, C. Schlick, and P. Poulin, "Efficient streaming of 3d scenes with complex geometry and complex lighting," in *Web3D '08 : Proceedings of the 13th international symposium on 3D web technology*. New York, NY, USA : ACM, 2008, pp. 11–17. 2.1.3.2.2

[74] O. Engolz, R. Goldenthal, D. Lischinski, and D. Cohen-Or, "An algebraic multi-grid approach for high-pass quantization," in *Proceedings of The 5th Korea-Israel Bi-National Conference on Geometric Modeling and Computer Graphics*, 2004, pp. 57–62. 2.23

[75] S. Ressler, Q. Wang, S. Bodarky, C. Sheppard, and G. Seidman, "Using vrml to access manufacturing data," in *VRML*, 1997, pp. 109–. 2.1.3.3.2

[76] G. S. Carson, R. F. Puk, and R. Carey, "Developing the vrml 97 international standard," *IEEE Comput. Graph. Appl.*, vol. 19, no. 2, pp. 52–58, 1999. 2.1.3.3.2

[77] [Online]. Available : http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml 2.1.3.3.4

[78] [Online]. Available : http://www.jtopen.com/docs/xt_format_Oct06.pdf 2.1.3.3.4

[79] J.-C. Latombe, *Robot Motion Planning*, T. S. I. S. in Engineering and C. Science, Eds. Springer, December 1990. 2.2.1.3, 2.2.2.1, 2.2.2.2, 2.2.3.1

[80] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.*, vol. 5, no. 1, pp. 90–98, 1986. 2.2.3.1

[81] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using laplace's equation," in *In Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, 1990, pp. 2102–2106. 2.2.3.1

[82] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," INRIA, Tech. Rep. RR-0621, 1987. 2.2.3.2

[83] K. Gupta and A. P. Pobil, *Practical Motion Planning in Robotics : Current Approaches and Future Directions*. New York, NY, USA : John Wiley & Sons, Inc., 1998. 2.2.4.1

[84] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *The International Journal of Robotics Research*, vol. 24, no. 12, pp. 1055–1071, 2005. [Online]. Available : http://ijr.sagepub.com/cgi/content/abstract/24/12/1055 2.2.4.1, 2.2.4.2.1

[85] A. Sanchez, J. A. Arenas, and R. Zapata, "Non-holonomic path planning using a quasi-random prm approach," in *IROS 2002 : international conference on intelligent robots and systems*, Septembre 2002. 2.2.4.1

[86] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, May 2006. 2.2.4.2.1, 4.2.2.2.2, 4.4.2.1

[87] J. Cortes, L. Jaillet, and T. Simeon, "Molecular disassembly with rrt-like algorithms," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 3301–3306. 2.2.4.2.1

[88] A. Yershova and S. M. LaValle, "Deterministic sampling methods for spheres and so(3)." in *ICRA*. IEEE, 2004, pp. 3974–3980. 2.2.4.2.1

[89] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain rrts : Efficient exploration by controlling the sampling domain." in *ICRA*. IEEE, 2005, pp. 3856–3861. 2.2.4.2.1, 4.1.2

[90] S. R. Lindemann and S. M. LaValle, "Steps toward derandomizing RRTs," in *IEEE Fourth International Workshop on Robot Motion and Control*, 2004. 2.2.4.2.1, 4.1.2

[91] J. A. Larios, "Le fil d'ariane : une méthode de planification générale. application à la planification automatique de trajectoires," Ph.D. dissertation, Institut National Polytechnique de Grenoble, 1994. 2.2.4.2.2

[92] [Online]. Available : http://www.perfrv.org/ 2.3

[93] C. Chabal, C. Megard, and L. Sibile, "Emm-3d : a virtual environment for evaluating maintenability from cad models," in *7th International Conference on Virtual Reality (VRIC 2005)*, 2005. 2.3.1

[94] [Online]. Available : http://www.vrcim.wsu.edu/ 2.3.1

[95] S. Jayaram, Y. Wang, U. Jayaram, K. Lyons, and P. Hart, "Vade : A virtual assembly design environment," in *VR '99 : Proceedings of the IEEE Virtual Reality*. Washington, DC, USA : IEEE Computer Society, 1999, p. 172. 2.3.1

[96] H. Joshi, S. Jayaram, and U. Jayaramand, "An open architecture for embedding vr-based mechanical tools into cad applications," in *Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2008. [Online]. Available : http://www.vrcim.wsu.edu/pages/cad/vr-based-function.html 2.3.1

[97] Y. Wang, U. Jayaram, S. Jayaram, and S. Imtiyaz, "Methods and algorithms for constraint-based virtual assembly," *Virtual Real.*, vol. 6, no. 4, pp. 229–243, 2003. 2.3.1

[98] U. Jayaram, Y. Kim, S. Jayaram, V. K. Jandhyala, and T. Mitsui, "Reorganizing cad assembly models (as-designed) for manufacturing simulations and planning (as-built)," *Journal of Computing and Information Science in Engineering*, vol. 4, no. 2, pp. 98–108, 2004. [Online]. Available : http://link.aip.org/link/?CIS/4/98/1 2.3.1, 3.3.2

[99] S. Gerbaud, N. Mollet, F. Ganier, B. Arnaldi, and J. Tisseau, "Gvt : A platform to create virtual environments for procedural learning." in *Actes du colloque IEEE Virtual Reality 2008 'VR 2008'*, Jan. 2008. 2.3.2.0.3

[100] O. B. Bayazit, G. Song, and N. M. Amato, "Enhancing randomized motion planners : Exploring with haptic hints," *Auton. Robots*, vol. 10, no. 2, pp. 163–174, 2001. 2.3.3, 4.1.2

[101] J. Rosell, C. Vázquez, A. Pérez, and P. I niguez, "Motion planning for haptic guidance," *J. Intell. Robotics Syst.*, vol. 53, no. 3, pp. 223–245, 2008. 2.3.3

[102] X. He and Y. Chen, "Haptic-aided robot path planning based on virtual tele-operation," *Robot. Comput.-Integr. Manuf.*, vol. 25, no. 4-5, pp. 792–803, 2009. 2.3.3

[103] D. C. Sherrard and M. Narayanan, "The aid of virtual reality in the industry," in *Northcon 95. I EEE Technical Applications Conference and Workshops Northcon95*, Oct 1995, pp. 28–. 3.1

[104] M. Göbel, "Industrial applications of ves," *IEEE Computer Graphics and Applications*, vol. 16, no. 1, pp. 10–13, 1996. 3.1

[105] S. Jayaram, J. Vance, R. Gadh, U. Jayaram, and H. Srinivasan, "Assessment of vr technology and its applications to engineering problems," *J. Comput. Info. Sci. Eng.*, vol. 1, no. 1, pp. 72–83, 2001. 3.1

[106] M. Kerttula, M. Salmela, and M. Heikkinen, "Virtual reality prototyping - a framework for the development of electronics and telecommunication products," in *RSP '97 : Proceedings of the 8th International Workshop on Rapid System Prototyping (RSP '97) Shortening the Path from Specification to Prototype*. Washington, DC, USA : IEEE Computer Society, 1997, p. 2. 3.1

[107] Jayaram, Sankar, Jayaram, Uma, Kim, Young, Dechenne, Charles, Lyons, Kevin, Palmer, Craig, Mitsui, and Tatsuki, "Industry case studies in the use of immersive virtual assembly," *Virtual Reality*, vol. 11, no. 4, pp. 217–228, October 2007. [Online]. Available : http://dx.doi.org/10.1007/s10055-007-0070-x 3.1

[108] N. I. Badler, C. A. Erignac, and Y. Liu, "Virtual humans for validating maintenance procedures," *Commun. ACM*, vol. 45, no. 7, pp. 56–63, 2002. 3.1

[109] D. Paillot, C. Pere, and F. Merienne, "Revue de projet industriel en immersion virtuelle," in *AIP Primeca*, March, 2003 2003. 3.1

[110] M. KRUS, "Connexions et facettisation : Gestion adaptative de sc/'enes virtuelles, application à la navigation dans des installations industrielles," Ph.D. dissertation, Universit/'e Paris XI - Sud U.F.R. de sciences, Juin 1999. 3.3.2.2.4

[111] G. Papaioannou, E.-A. Karabassi, and T. Theoharis, "Segmentation and surface characterization of arbitrary 3d meshes for object reconstruction and recognition," *Pattern Recognition, International Conference on*, vol. 1, p. 1734, 2000. 3.4.3.3

[112] Y. Zhang, J. Paik, A. Koschan, and M. A. Abidi, "A simple and efficient algorithm for part decomposition of 3-d triangulated models based on curvature analysis," in *in Proceedings of the International Conference on Image Processing, III*, 2002, pp. 273–276. 3.4.3.3

[113] W. B. Thompson, J. C. Owen, H. James, H. J. de St. Germain, S. R. Stark, T. C. Henderson, S. Member, and S. Member, "Feature-based reverse engineering of mechanical parts," in *In Proc. ARPA Image Understanding Workshop*, 1999, pp. 1115–1124. 3.4.3.3

[114] N. Ladeveze, J. Y. Fourquet, and M. Taix, "Enchaînement de mouvements en environnements complexes : méthodes alliant la planification à la prise de décision en situation immersive," Master's thesis, Ecole Nationale d'Ingénieurs de tarbes, 2006. 4.1.2

[115] J. Kuffner, J.J. and S. LaValle, "Rrt-connect : An efficient approach to single-query path planning," vol. 2, 2000, pp. 995–1001 vol.2. 4.2.2.2.2

[116] A. Yershova and S. M. LaValle, "Improving motion-planning algorithms by efficient nearest-neighbor searching." *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 151–157, 2007. 4.2.2.2.2

[117] S. F. Frisken and R. N. Perry, "Simple and efficient traversal methods for quadtrees and octrees," *Journal of Graphics Tools*, vol. 7, no. 7, p. 2002, 2002. 4.3.2.2

[118] [Online]. Available : http://www.boost.org/ 5.2

Annexes

Solutions de calcul retenues pour la caractérisation approximative rapide de surfaces canoniques

5.2.0.2.1 Calcul d'un plan L'identification d'un plan est réalisée par la spécification d'un point et d'un vecteur normal. Dans ce contexte, une seule facette permet de fixer rapidement les caractéristiques au moyen de l'algorithme 5.1.

Algorithme 5.1 Caractérisation d'un plan

```
FONCTION Créer_Plan ( f : Facette ) : PlanCAO
DÉBUT
PlanCAO P ;
P.point ←f.vertex0
P.normale←Produit Vectoriel ( ( f.vertex1 - f.vertex0 , f.vertex2 - f.vertex0 )
RETOURNE P
FIN
```

La condition d'appartenance d'un vertex V au plan $P(O, \overrightarrow{n})$ est alors exprimée par rapport à la définition d'une distance minimale ε :

$$\left|\overrightarrow{OV} \cdot \overrightarrow{n}\right| < \varepsilon$$

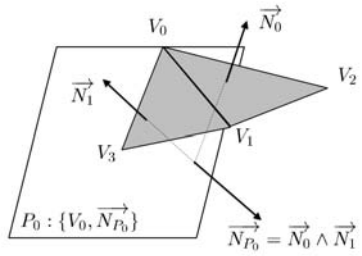
5.2.0.2.2 Calcul d'un cylindre L'identification d'un cylindre est réalisée par la spécification d'un axe (décrit par un point et un vecteur directeur) et un rayon. Dans ce cas, l'expression de ces 3 paramètres requiert deux facettes non coplanaires $f1$ et $f2$. La direction de l'axe n_{P_0} est fixée par le produit vectoriel des normales au facettes $f1$ et $f2$ comme illustrée sur la figure 5.1a.

Grâce au calcul de cette direction et des coordonnées d'un vertex, on peut définir un plan $P_0(V_0, \overrightarrow{n_{P_0}})$ sur lequel on projette les vertices v_0, v_1, v_2, v_3 en v'_0, v'_1, v'_2, v'_3. On définit un repère local $R_l(v'_0, \overrightarrow{i}, \overrightarrow{j}, \overrightarrow{n_{P_0}})$ avec $\overrightarrow{i} = \overrightarrow{V_0V'_1}$ et $\overrightarrow{j} = -\overrightarrow{i} \wedge \overrightarrow{n_{P_0}}$ ainsi qu'une matrice de changement de repère $M_{R_m \mapsto R_l}$ permettant l'expression des coordonnées d'un vertex depuis sa définition dans le repère du maillage R_m dans R_l. On calcule ensuite la position de du centre$C$ du cercle passant par 3 des 4 points v'_0, v'_1, v'_2, v'_3 non alignés ainsi que son rayon r comme illustré sur la figure 5.1b.

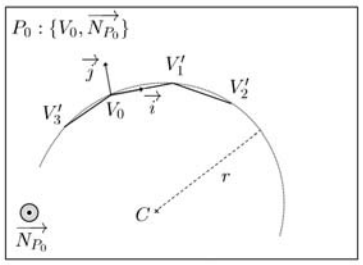
L'appartenance d'un vertex V au Cylindre $(P_0, M_{R_m \mapsto R_l}, C, r)$ est alors conditionnée par rapport à une distance minimale ε :

$$\left|(V_l \cdot \overrightarrow{i} - C \cdot \overrightarrow{i})^2 + (V_l \cdot \overrightarrow{j} - C \cdot \overrightarrow{j})^2 + (V_l \cdot \overrightarrow{n_{P_0}} - C \cdot \overrightarrow{n_{P_0}})^2 - r\right| < \varepsilon\,, \text{ avec } V_l = M_{R_m \mapsto R_l} \times V$$

5.2.0.2.3 Calcul d'une sphère L'identification d'une sphère par la détermination de son centre C et de son rayon r peut être réalisée grâce à une méthode de calcul itérative à partir de 4 points dans l'espace mais



(a) Calcul de la direction de l'axe du cylindre et d'un plan de projection P_0 servant au calcul de la position du centre et du rayon



(b) Caractéristiques d'un cylindre déduite à partir de deux facettes tangentes non coplanaires $f_0 : \{V_0, V_1, V_2\}$ et $f1 : \{V_0, V_3, V_1\}$par projection des vertices sur le plan P_0

FIGURE 5.1 – Calcul des caractéristiques d'un cylindre

représente un temps de calcul très coûteux. Il est possible d'estimer ses caractéristiques à partir de trois facettes $f_0 : (v_{01}, v_{02}, v_{03})$, $f_1 : (v_{11}, v_{12}, v_{13})$ et $f_2 : (v_{21}, v_{22}, v_{23})$ non coplanaires deux à deux à partir de la méthode de calcul des caractéristiques d'un cylindre. Grâce à deux paire de facettes (f_0, f_1) et (f_0, f_2), deux caractéristiques de cylindre $(P_0, M_{R_m \mapsto R_{l0}}, C_0, r_0)$ et $(P_1, M_{R_m \mapsto R_{l1}}, C_1, r_1)$ sont calculées. Dans le cas où les deux rayons r_0 et r_1 sont égaux ou proches, une approximation du rayon de sphère $R = (r_0 + r_1)/2$ permet de calculer la position du centre C_s à partir de deux des trois facettes grâce à une relation (développée en Annexe) issues de la relation $C_s v_{01} = C_s v_{02} = C_s v_{03} = R$.

Algorithme 5.2 Caractérisation d'une sphère

```
FONCTION Créer_Sphère ( f1 , f2 , f3 ) : Sphère
DÉBUT
Cylindre C1←Cylindre(f1,f2)
Cylindre C2←Cylindre(f1,f3)
R ←(C1.Rayon + C2.Rayon)/2
Sphère S ←Sphère( f1, f2, R)
FIN
```

Liste des algorithmes