SIMULATEUR TUTORIEL INTELLIGENT POUR LES OPÉRATIONS ROBOTISÉES

APPLICATION AU BRAS CANADIEN SUR LA STATION SPATIALE INTERNATIONALE

par

Khaled Belghith

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 19 juillet 2010

Your file Votre référence
ISBN: 978-0-494-70601-5
Our file Notre référence
ISBN: 978-0-494-70601-5

Le 22 juillet 2010,

le jury a accepté la thèse de Monsieur Khaled Belghith
dans sa version finale.

Membres du jury

Professeur Froduald Kabanza
Directeur de recherche
Département d'informatique

Professeur Roger Nkambou
Membre
Département Informatique
Université du Québec à Montréal

Docteur Leo Hartman
Membre externe
Agence spatiale canadienne

Professeur Richard St-Denis
Président rapporteur
Département d'informatique

Sommaire

Cette thèse a pour objectif de développer un simulateur tutoriel intelligent pour l'apprentissage de manipulations robotisées, applicable au bras robot canadien sur la station spatiale internationale. Le simulateur appelé Roman Tutor est une preuve de concept de simulateur d'apprentissage autonome et continu pour des manipulations robotisées complexes. Un tel concept est notamment pertinent pour les futures missions spatiales sur Mars ou sur la Lune, et ce en dépit de l'inadéquation du bras canadien pour de telles missions en raison de sa trop grande complexité. Le fait de démontrer la possibilité de conception d'un simulateur capable, dans une certaine mesure, de donner des rétroactions similaires à celles d'un enseignant humain, pourrait inspirer de nouvelles idées pour des concepts similaires, applicables à des robots plus simples, qui seraient utilisés dans les prochaines missions spatiales. Afin de réaliser ce prototype, il est question de développer et d'intégrer trois composantes originales : premièrement, un planificateur de trajectoires pour des environnements dynamiques présentant des contraintes dures et flexibles ; deuxièmement, un générateur automatique de démonstrations de tâches, lequel fait appel au planificateur de trajectoires pour trouver une trajectoire solution à une tâche de déplacement du bras robot et à des techniques de planification des animations pour filmer la solution obtenue ; et troisièmement, un modèle pédagogique implémentant des stratégies d'intervention pour donner de l'aide à un opérateur manipulant le SSRMS. L'assistance apportée à un opérateur sur Roman Tutor fait appel d'une part à des démonstrations de tâches générées par le générateur automatique de démonstrations, et d'autre part au planificateur de trajectoires pour suivre la progression de l'opérateur sur sa tâche, lui fournir de l'aide et le corriger au besoin.

Remerciements

À mon directeur de thèse, monsieur Froduald Kabanza, professeur au département d'informatique de l'Université de Sherbrooke. Je vous remercie de m'avoir accueilli dans votre laboratoire et de m'avoir confié ce projet qui me tenait particulièrement à coeur. Puissent ces quelques mots vous témoigner de ma profonde gratitude, pour votre encadrement exemplaire, votre soutien, votre confiance, ainsi que pour vos précieux apports scientifiques tout au long de cette thèse.

À tous les membres du jury de cette thèse. Je vous remercie vivement pour l'honneur que vous me faites en acceptant d'évaluer ce travail de recherche.

À monsieur Roger Nkambou, professeur à l'Université du Québec À Montréal. Un grand Merci pour votre disponibilité et pour vos idées pertinentes et vos conseils congrus, m'ayant été d'une grande aide dans la réalisation du troisième chapitre de cette thèse.

À monsieur Leo Hartman, chercheur à l'Agence spatiale canadienne. Je tiens à vous remercier pour tous vos conseils, et tout le soutien que vous m'avez apporté tout au long de la rédaction de la thèse.

Je tiens également à remercier tous les membres des laboratoires GDAC (UQAM) et Planiart (Université de Sherbrooke) ayant contribué de près ou de loin à la réalisation du projet Roman Tutor. Merci pour toutes ces collaborations ayant aidé à réaliser cette thèse. Un remerciement particulier à Philipe Bellefeuille, Mahie Khan et Benjamin Auder.

Enfin, à ma mère, à mon père, sans qui cette thèse n'aurait pas vu le jour. À mon frère, à toute ma famille et à toutes les personnes qui me sont proches, un grand merci pour vos encouragements, votre affection et pour avoir cru en moi. Un tendre merci à Raoudha Trabelsi, ma future femme, pour sa patience et son soutien constant tout au long de la rédaction de cette thèse.

Abréviations

ATDG Automatic Task Demonstration Generator

FADPRM Flexible Anytime Dynamic PRM

LTL Logique temporelle linéaire

PRM Probabilistic Roadmap Methods

Roman Tutor RObot MANipulation Tutor

SSI Station spatiale internationale

SSRMS Space Station Remote Manipulator System (le bras canadien)

STI Simulateur (système) tutoriel intelligent

Table des matières

Table des figures

Introduction

La manipulation du bras robot canadien ou SSRMS (pour "Space Station Remote Manipulator System") sur la station spatiale internationale (SSI) est une tâche fortement complexe. Elle nécessite la maîtrise complète de la géométrie du robot et des différents modes de manipulation, ainsi qu'une connaissance approfondie de l'architecture de la station avec ses différents composants. La figure 1 montre une vue réelle de la SSI et du SSRMS. Le SSRMS est utilisé pour déplacer une charge d'un endroit à un autre, réparer un des éléments de la station et inspecter la station à l'aide d'une caméra montée à son extrémité. Toutes ces manipulations sont accomplies avec la plus grande précision de sorte que le SSRMS ne doit, en aucun cas, entrer en collision avec un élément de la station.

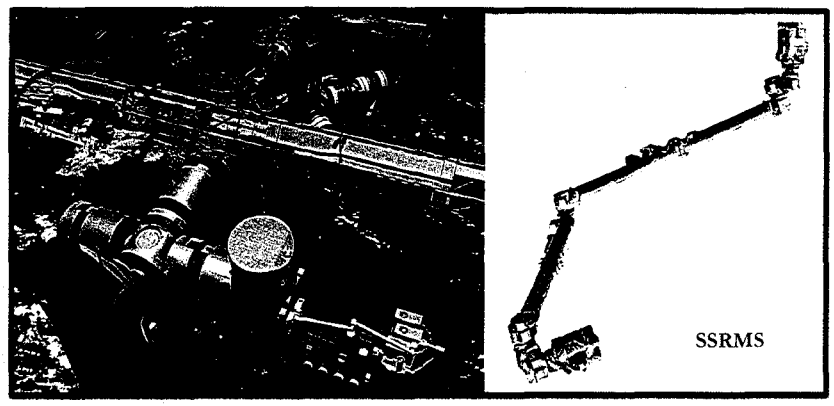


figure 1 – SSRMS sur la station spatiale internationale

La manipulation du SSRMS se fait à distance, à partir d'un ordinateur situé à l'intérieur d'un des compartiments de la SSI. Cet ordinateur comprend trois moniteurs reliés chacun à une caméra placée à un endroit particulier de la SSI. En tout, les caméras sont au nombre de quatorze, montées à différents endroits de la SSI. Lors de la manipulation du SSRMS,

un bon choix de caméras pour chacun des trois moniteurs s'impose. L'astronaute choisit celles qui lui offrent une meilleure visibilité tout en gardant une bonne appréciation de son évolution dans la tâche. La difficulté de la manipulation du SSRMS provient surtout du fait que chacun des moniteurs ne présente qu'une vue partielle de la station. Nous utilisons l'appellation "manipulations robotisées" pour référer aux manipulations effectuées à distance par les astronautes à travers l'ordinateur pour déplacer le SSRMS.

Cette thèse a pour objectif de développer un système, dans notre cas un simulateur, tutoriel intelligent (STI) pour l'apprentissage de manipulations robotisées, applicable au SSRMS sur la SSI. Le STI appelé Roman Tutor (voir la figure 2), pour RObot MANipulation Tutor, est une preuve du concept de simulateur d'apprentissage autonome et continue pour des manipulations robotisées complexes. Un tel concept est pertinent pour les futures missions spatiales sur Mars ou sur la Lune. Même si le SSRMS ne sera pas impliqué dans de telles missions à cause de sa trop grande complexité, le fait de démontrer qu'on peut concevoir un simulateur capable de donner des rétroactions similaires à celles d'un enseignant humain peut fournir des idées pour des concepts similaires applicables aux robots plus simples qui seront utilisées dans les prochaines missions sur Mars ou la Lune.

figure 2 – Roman tutor interface

Au-delà de prouver le concept d'apprentissage autonome des manipulations robotisées,

INTRODUCTION

Roman Tutor constitue un outil de validation des opérations robotisées utilisable par les équipes terrestres en support aux astronautes sur le SSI. En effet, la réalisation d'une opération impliquant le SSRMS sur le SSI nécessite au préalable la validation par une équipe sur terre en liaison avec les astronautes. L'équipe sur terre utilise un simulateur du SSRMS pour valider les opérations impliquant le SSRMS. Cependant, le simulateur actuel est essentiellement un outil de visualisation, sans aucune intelligence. Il ne peut pas comparer différents scénarios possibles ni donner des rétroactions pédagogiques aux opérateurs (par exemple, expliquer pourquoi une manoeuvre est préférable à une autre). En fait, il revient aux opérateurs de décider du déplacement du SSRMS dans le simulateur et comparer les différentes alternatives.

Contrairement à un simulateur normal, Roman Tutor a la capacité de comprendre, dans une certaine mesure, les manipulations du SSRMS pour pouvoir valider les manipulations faites par un opérateur, les critiquer et suggérer de meilleures approches. Une des composantes permettant au STI cette capacité de comprendre les manipulations est un planificateur des déplacements du SSRMS. Étant donné une configuration initiale du SSRMS et une configuration cible, le planificateur est en mesure de déterminer une trajectoire menant de la configuration initiale à la configuration finale, tenant compte des obstacles et des contraintes de visibilité. Ainsi, étant donné une trajectoire faite par un opérateur, Roman Tutor peut la critiquer par rapport à sa propre trajectoire et fournir des rétroactions au besoin.

La planification de trajectoires pour des robots articulés est un problème fortement étudié en robotique et en intelligence artificielle, auquel plusieurs solutions ont été proposées [36, 11, 22, 57, 44]. Dans sa forme classique, le problème consiste à amener un robot d'une position à une autre en évitant la collision avec les obstacles présents dans le milieu. Une des choses qui rendent le problème particulier dans Roman Tutor est son aspect fortement dynamique résultant du fait qu'il doit suivre les déplacements du SSRMS effectués par un opérateur pour les valider. Pendant que l'opérateur déplace le SSRMS, le rôle du planificateur est de vérifier, à chaque mouvement, l'existence d'une trajectoire possible entre la configuration courante du robot et une configuration finale donnée comme cible à atteindre. Roman Tutor ne peut pas prévoir à l'avance les mouvements de l'opérateur ; il doit recalculer dynamiquement la trajectoire vers la cible à partir de la configuration courante. Cela doit se faire efficacement en exploitant les calculs réalisés dans les étapes précédentes.

Idéalement, ceci nécessite que le planificateur de trajectoire ait la propriété "anytime" [20], c'est-à-dire, qu'il soit capable de donner une solution approximative dans un délai limité ; de sorte que plus le délai alloué est élevé, plus la solution est proche de l'optimalité.

Une autre particularité du problème de planification de trajectoires dans Roman Tutor concerne les contraintes de visibilité. Le planificateur doit calculer des trajectoires visibles au travers des caméras montées sur la SSI et que l'opérateur utilise pour voir l'extérieur de la SSI. Bien entendu, la contrainte de déplacer le SSRMS d'une position initiale vers une position finale en évitant les collisions demeure la contrainte la plus critique. Il s'agit d'une contrainte dite dure parce que les collisions doivent être évitées à tout prix. Par contre, différentes régions de la SSI peuvent être plus ou moins visibles selon les caméras choisies et selon les conditions de l'environnement. Par exemple, l'orbite de la SSI peut faire en sorte qu'à différents moments des caméras deviennent exposées au soleil, rendant difficile la visibilité des régions couvertes par ces dernières. Selon ce degré de visibilité, une trajectoire passant à travers certaines régions doit être évitée le plus possible, mais peut être acceptée s'il s'avère impossible de trouver de meilleures solutions. Les contraintes de visibilité sont donc des contraintes souples ou flexibles. Elles expriment des préférences dans la navigation du robot.

Pour donner des rétroactions, Roman Tutor utilise un générateur automatique de démonstrations de tâches ATDG pour "Automatic Task Demonstration Genrator". Étant donné la description d'une tâche (par exemple, déplacer une charge d'un endroit à un autre), le ATDG génère une animation 3D interactive expliquant comment effectuer une telle tâche. La conception du ATDG est inspirée des techniques de planification automatique des animations 3D. Les techniques usuelles présentes dans la littérature sont de deux types : les approches par satisfaction de contraintes et les approches par idiomes (chapitre 2). Les approches par satisfaction de contraintes travaillent au niveau de chaque image constituant le film et utilisent des méthodes numériques fastidieuses pour s'assurer de l'exactitude et de la cohérence de leur contenu. Les approches par idiomes s'inspirent du monde cinématographique et utilisent différents concepts leurs permettant de filmer une animation comme le fait un metteur en scène. Vu le caractère très complexe de notre application, les deux familles d'approche s'avèrent être inapplicables directement à notre problématique.

En résumé, cette thèse a pour objectif le développement d'un prototype simulateur tutoriel intelligent pour l'apprentissage d'opérations robotisées, appelé Roman Tutor. Pour

réaliser ce prototype, nous nous proposons de développer et d'intégrer trois composantes originales.

1. Un planificateur de trajectoires pour des environnements dynamiques présentant des contraintes dures et flexibles.
2. Un générateur automatique de démonstrations de tâches (ATDG). Ce dernier fait appel au planificateur de trajectoires pour trouver une trajectoire solution à une tâche de déplacement du SSRMS et à des techniques de planification des animations pour filmer la solution obtenue.
3. Un modèle pédagogique implémentant des stratégies d'intervention pour donner de l'aide à un opérateur manipulant le SSRMS.

Le premier chapitre de cette thèse introduit la nouvelle approche de planification de trajectoires avec préférences que nous avons développée. Cette nouvelle approche étend les approches de planification de trajectoires par échantillonnage actuelles par l'ajout de la caractéristique flexible permettant de tenir compte des contraintes souples dans l'environnement, et de là tenir compte des contraintes de visibilité sur la SSI. Les approches par échantillonnage sont les plus efficaces dans des environnements à forte complexité [30]. Nous montrons dans ce chapitre que cette notion de flexibilité permet d'améliorer grandement la qualité des trajectoires générées, un des plus importants handicaps des approches par échantillonnage. De plus, ce nouveau planificateur adapte les caractéristiques "anytime" et dynamique au sein des approches de planification par échantillonnage.

Le deuxième chapitre est consacré à l'étude de l'ATDG, le générateur automatique de démonstrations de tâches. ATDG utilise la logique temporelle linéaire (LTL) [3] pour exprimer les principes cinématographiques et les préférences de filmage, un langage plus expressif et avec une sémantique plus simple que les anciens langages de planification de caméras. ATDG implémente l'algorithme sous-jacent TLPlan pour la planification de caméras. Tout d'abord, la trajectoire du robot est générée en utilisant le planificateur de trajectoires. Le planificateur de caméras TLPlan est ensuite invoqué pour trouver la meilleure séquence de configurations de caméras filmant le robot sur sa trajectoire.

Dans le troisième et dernier chapitre de cette thèse, nous exposons le modèle pédagogique implémenté au sein de Roman Tutor pour supporter et encadrer l'apprentissage de manipulations robotisées. L'aide à un opérateur fait appel d'une part à des démonstrations

de tâches générées automatiquement par le ATDG, et d'autre part au planificateur de trajectoires pour suivre la progression de l'opérateur sur sa tâche, lui fournir de l'aide et le corriger au besoin.

Chapitre 1

Planification de trajectoires avec préférences dans des environnements dynamiques très complexes

Résumé

Les approches de planification de trajectoires présentes dans la littérature peuvent être classées en deux principales catégories : les approches combinatoires et les approches par échantillonnage [14, 44]. Les approches par échantillonnage ou PRMs (pour "Probabilistic Roadmap Methods") sont les plus efficaces dans des environnements à forte complexité [30]. Leur caractéristique probabiliste leur permet d'éviter la représentation explicite et complète de l'espace. Par échantillonnage probabiliste, elles construisent un roadmap ou graphe qui n'en est qu'une représentation simplifiée. Vu la complexité accrue des environnements dans lesquels oeuvrent les robots propres à notre problématique, il semble évident que la nouvelle approche de planification de trajectoires que nous présentons dans ce chapitre soit alignée à cette catégorie de planificateurs.

Dans sa formulation traditionnelle, le problème de planification de trajectoires consiste à amener un robot d'une position à une autre en évitant la collision avec les obstacles présents dans l'environnement. Dans certaines applications complexes du monde réel, toutefois, en plus des obstacles qui doivent être

évités, il peut y avoir des zones dangereuses qui doivent être évitées autant que possible. La notion de danger est pertinente par exemple dans des applications militaires [51, 59]. Inversement, il peut être souhaitable pour un chemin de rester à proximité de certaines zones autant que possible. C'est notamment le cas dans la manipulation du SSRMS sur la SSI, où les préférences sont liées à des champs de visions de caméras, qui changent de façon dynamique, en partie à cause des changements dans l'orbite de la SSI.

Nous présentons dans les pages qui suivent un article qui introduit notre nouvelle approche de planification de trajectoires avec préférences. Cette nouvelle approche est intitulée FADPRM pour "Flexible Anytime Dynamic PRM". Le planificateur FADPRM va étendre les approches de planification PRMs actuelles par l'ajout de la caractéristique flexible permettant de tenir compte des contraintes souples dans l'environnement. Les contraintes souples (ou flexibles) sont dictées par la présence de zones avec différents degrés de désirabilité dans l'environnement. FADPRM prévilégie des trajectoires solutions qui évitent les zones non désirables et qui font passer le robot le plus possible à travers les zones désirables.

Même quand un problème de planification de trajectoires ne nécessite pas l'ajout de la notion de désirabilité de manière explicite, l'introduire permet de fournir un moyen efficace pour contrôler la qualité des trajectoires générées par une méthode de planification par échatillonnage. En effet, dans les approches par échantillonnage, les trajectoires sont obtenues en reliant des configurations qui sont choisies au hasard dans l'espace des configurations libres. Ceci conduit à des trajectoires solutions complexes et de mauvaise qualité, nécessitant de opérations heuristiques de post-traitement pour les lisser [58]. Dans l'article suivant, nous démontrons qu'il est possible d'influencer la stratégie d'échantillonnage pour générer des trajectoires plus lisses juste en spécifiant des zones avec degrés de désirabilité dans l'environnement. Nous montrons également que la notion de désirabilité permet un meilleur contrôle de l'exploration de l'espace de recherche pour garantir une planification plus efficace.

Avec FADPRM, l'exploration par échantillonnge de l'espace des configurations libres s'inspire de différentes stratégies d'exploration dynamiques et "anytime" présentes dans le littérature [47, 48, 39]. Un planificateur de trajectoires

peut s'adapter aux changements dynamiques qui surviennent dans l'environnement (changement dans les obstacles, dans les zones et dans leurs degrés de désirabilité ou dans les configurations initiale et finale) en recalculant une nouvelle trajectoire à chaque fois. Une stratégie dynamique réutilise les résultats obtenus des recherches antérieures pour garantir de meilleures performances dans la planification. Une stratégie "anytime" [20] procède de manière incrémentale, en commençant par un plan de degré de désirabilité bas, puis l'améliore progressivement. De cette façon, et à tout moment, le planificateur dispose d'un plan d'un certain degré de satisfaction qui est amélioré si plus de temps de planification est disponible.

Commentaires

Une première version du planificateur FADPRM a été présentée à la conférence "IEEE International Conference on Robotics and Automation" en 2006 [7]. Dans cette dernière, FADPRM utilisait la stratégie d'exploration AD* pour "Anytime Dynamic A*" [48]. Dans sa nouvelle version présentée dans les pages qui suivent, FADPRM implémente une version "anytime" de la stratégie plus récente GAA* pour "Genralized Adaptive A*" [61]. Nous utilisons maintenant GAA* à la place de AD* parce que les deux algorithmes ont des performances comparables et que GAA* a une structure plus simple et plus facile à comprendre et à implémenter.

L'article présenté dans ce chapitre a été soumis au journal "International Journal of Robotics Research". Une version plus courte de ce travail va paraître dans les actes de "International Conference on Autonomous and Intelligent Systems" [6]. Dans la version courte, seulement l'adaptation du planificateur GAA* dans FADPRM sans intégrer de la stratégie "anytime" a été présentée. Ces travaux ont été réalisés, validés et rédigés sous la supervision du Professeur Froduald Kabanza (Université de Sherbrooke) et avec les conseils du Professeur Leo Hartman (Agence spatiale canadienne).

Randomized Path Planning with Preferences in Highly Complex Dynamic Environments

Khaled Belghith, Froduald Kabanza
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
khaled.belghith@usherbrooke.ca, kabanza@usherbrooke.ca
Leo Hartman
Canadian Space Agency,
John H. Chapman Space Centre, 6767 Route de l'Aéroport
Saint-Hubert, Québec J3Y 8Y9
leo.hartman@asc-csa.gc.ca

Abstract

In this paper, we consider the problem of planning paths for articulated bodies operating in workplaces containing obstacles and regions with preferences expressed as degrees of desirability. Degrees of desirability could specify danger zones and desire zones. A planned path should not collide with the obstacles and should maximize the degrees of desirability. Region desirability can also convey search-control strategies guiding the exploration of the search space. To handle desirability specifications, we introduce the notion of flexible probabilistic roadmap (flexible PRM) as an extension of the traditional PRM. Each edge in a flexible PRM is assigned a desirability degree. We show that flexible PRM planning can be achieved very efficiently with a simple sampling strategy of the configuration space defined as a trade-off between a traditional sampling oriented towards coverage of the configuration space and a heuristic optimization of the path desirability degree. For path planning problems in dynamic environments, where obstacles and region desirability can change in real-time, we use dynamic and anytime search exploration strategies. The dynamic strategy allows the planner to re-plan efficiently by exploiting results from previous planning phases. The anytime strategy starts with a quickly computed path with a potentially low desirability degree which is then incrementally improved depending on the available planning time.

1.1 Introduction

In its traditional form, the path planning problem is to plan a path for a moving body (typically a robot) from a given start configuration to a given goal configuration in a workspace containing a set of obstacles. The basic constraint on solution paths is to avoid collision with obstacles, which we call hereby a hard constraint. There exist numerous approaches for path planning under this constraint [36, 11, 22, 57, 44].

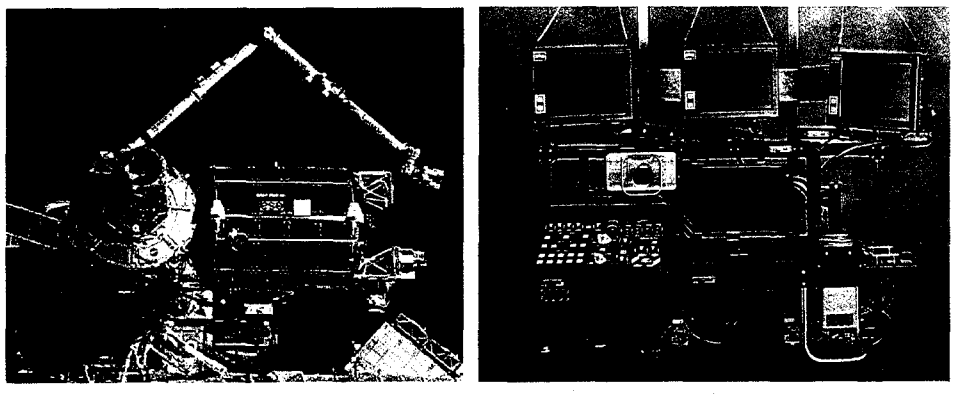
In many complex applications, however, in addition to obstacles that must be avoided, we may have dangerous areas that must be avoided as much as possible. That is, a path going through these areas is not highly desirable, but would be acceptable if no better path exists or can be computed efficiently. The danger concept is relevant, for example, in military applications. Some path planning techniques that deal with it have been proposed, including [59, 51]. Conversely, it may be desirable for a path to stay close to certain areas as much as possible. Even when a path planning problem has no explicit notion of region desirability, introducing the notion provides a way to control the quality of a path generated by a randomized path planning method. Indeed, paths are obtained by connecting milestones that are randomly sampled in the free workspace and this tends to yield awkward paths, requiring heuristic post-processing operations to smooth them. In this paper, we demonstrate that one can influence the sampling strategy to generate less awkward paths by specifying zones the path is preferred to go through. We also show that region desirability specifications can also help control the exploration of the sampled search space and make the path-planner more efficient.

Our path planning approach builds flexible roadmaps by extending existing sampling techniques including delayed collision checking, single query, bi-direction and adaptive sampling [58]. Desirable and undesirable workspace regions are soft constraints on the robot path, whereas obstacles are hard constraints. The soft constraints convey preferences for rating solutions paths which must avoid obstacles. The more a path avoids undesirable zones and goes through desirable zones, the better it is.

The exploration of the sampled configuration space is done using dynamic and anytime space exploration methods [47, 48, 39]. In dynamic environments, a path planner can adapt a previously computed path to dynamic changes in obstacle configurations, goals or region desirability, by computing a new path. Dynamic state space exploration strategies reuse

the results obtained from previous searches to achieve better performance compared to re-searching from scratch. In addition, an anytime search strategy proceeds incrementally, starting with a path having a low desirability degree and then improving it incrementally. In this way at "any time", the planner has a plan with some degree of satisfaction that is improved as more planning time is spent.

Our testbed is a simulation of the Space Station Remote Manipulator System (SSRMS) deployed on the International Space Station (ISS). The SSRMS is a 17 meter long articulated robot manipulator, having a translation joint, seven rotational joints (each with a range of 270 degrees) and two latching end-effectors which can be moved to various fixtures, giving it the capability to "walk" from one grappling fixture to next on the exterior of the ISS [19]. Astronauts operate SSRMS using the robot control station located inside one of the ISS compartments (Figure 1.1). A robot control station has an interface with three monitors, each connected to a camera placed at a strategic location of the ISS. There are many cameras covering different parts of the ISS structure and three of them are selected and mapped to the three monitors.



(a) View from one of the monitors (b) Robot control station

figure 1.1 – ISS path-planning domain and robot control station

Most of the SSRMS tasks on the ISS involve moving the robot from one configuration to another in order, for example, to move a payload from the shuttle or inspect a region of the ISS exterior using a camera mounted on the end effector. A judicious choice of the camera on each of the three monitors along different segments of a robot path ensures that the operator is appropriately aware of the robot motion. Computed paths must go as

much as possible through camera fields of view to enable a good appreciation of the robot motion. In other words, the camera fields of view convey preferences for regions through which the robot path should remain as much as possible while avoiding collisions with the ISS structure.

In the next section we give the background and discuss some of the related work. We then present our path planning approach to handling path preferences in the robot workspace. We follow with experiments in the ISS environment and in a car repair domain, showing the capability of the new planning approach to handle path preferences and search control specifications that are expressed by assigning desirability degrees to workspace regions.

1.2 Background and Related Work

A configuration q of an articulated robot with n degrees of freedom (dof) is an n element vector of the robot joint positions. Since the robot moves by changing its joint rotations or translations, a path between two points is a sequence of configurations sufficiently close together connecting the two points. A path is collision-free or in the space of collision-free configurations C_{free}, if the robot does not collide with any obstacle in the workspace in any of the configurations on the path. Computing a path is seen as making a query (to the path-planner) with the input of the start and goal configurations. Two very commonly used approaches to path planning are the combinatorial and randomized approaches.

Combinatorial approaches, also called decomposition or exact approaches, proceed by searching through a geometric representation of C_{free}. Given a 2D or 3D model of obstacles in the workspace, a 2D or 3D model of the robot, the configuration space is decomposed into an occupancy grid of cells, also called a roadmap. A path from a start cell to a goal cell is then found by searching a sequence of moves between adjacent free cells, connecting the start configuration to the goal [38, 49, 22, 44]. These moves correspond to possible edges in a graph with nodes corresponding to free cells in the grid. Graph-search algorithms such as A* search [29, 54] or AD* [48] can be used to compute a path between the start and goal configurations.

Randomized approaches, also known as sampling-based approaches, proceed by sampling the space of the robot configurations. Given a 2D or 3D model of obstacles in the

workspace and a 2D or 3D model of the robot, a randomized planner builds a graph of nodes corresponding to configurations in C_{free} by picking configurations randomly and checking that they are collision-free. It uses a fast collision detection checker (called a local planner) to check that an edge between two adjacent nodes is also collision-free ; each time the local planner succeeds, the corresponding edge (i.e., local path or path segment) is inserted into the graph. The graph built that way is called a probabilistic roadmap (PRM) [36] or a rapidly-exploring random tree (RRT) [44] and is a simplified representation of C_{free}. Here too graph-search algorithms such as A* search [54] or AD* [48] can be used to explore the graph to find a collision-free path linking the start to the goal configuration.

Therefore combinatorial as well as randomized approaches have in common the discretization problem to build an intermediate graph structure (the roadmap) and search through it. The key difference lies in what the graph represents and how it is built. With combinatorial approaches the graph is meant to be an exact representation of C_{free} and its construction takes into account the geometry of the workspace and the robot. With sampling approaches, the graph represents samples of C_{free}. It is not an exact representation of C_{free}. Given that the configuration space is randomly sampled, randomized approaches do not guarantee a full coverage of free space and they are not complete and do not guarantee optimality. In fact, they are probabilistically complete, meaning that the more samples are made the closer the probability of guaranteeing the absence or presence of a solution gets to 1 [58]. Combinatorial approaches guarantee completeness and optimality by using a sufficiently small discritization step. In practice, this results in large search spaces, making the approaches generally intractable for high dimensional configuration spaces [30].

A heuristic method for grid decompositions is to plan using a coarse discretization space. If no solution is found or to improve the solution found so far, a new planning iteration is made with finer discretization pace. The process can be iterated as more planning time is invested or until a satisfactory solution is found. Another exploration strategy for the occupancy grid maybe to use random search [45]. While this may help coping with the complexity of the configuration space, in very large configuration spaces the planner spends a large amount of time generating the occupancy grid [30]. Sampling-based methods generally offer better performance than exact methods for domains with high-dimensional spaces [30, 14, 44].

1.2.1 Probabilistic Roadmap Approach

Our randomized implementation follows a PRM approach [58]. However, it can be easily adapted to an RRT approach given an RRT approach fundamentally corresponds to a single-query PRM with on the fly search of the sampled roadmap combined with on-the-fly collision detection [45]. Our implementation includes various configuration modes that allow the planner to run in a single or multiple query mode, with on-the-fly search of the roadmap or not and with collision detection on-the-fly or delayed.

A PRM is an undirected graph $G = (N, V)$ with N the nodes of the graph and V the arcs. The nodes are sampled configurations in C_{free}, also called milestones. The arcs represent links or segments v connecting two configurations. Algorithm 1 shows a basic PRM path planning algorithm. It starts by initializing the roadmap G with the start and goal configurations n_{start} and n_{goal}. Then, a new node n is sampled randomly, with a probability measure π, in C_{free} and added to the roadmap. A set of nodes in G and in the neighborhood of n called V_n is selected. Using a collision checker (local planner), we look for a node n' in V_n such that the link (n, n') is free of collisions and then add it to G. The process is repeated until a path connecting n_{start} and n_{goal} is found.

Algorithm 1 Basic PRM Algorithm

```
01. Initialize the roadmap G with two nodes, n_start and n_goal
02. Repeat
03.     Sample a configuration n from C_free with probability measure π
04.     if (n ∈ C_free) then add n as a new node of G
05.     for some nodes n' in a neighborhood V_n of n in G such that n' ≠ n
06.         if collisionFree(n, n') then add v = (n, n') as a new edge of G
07. until n_start and n_goal are in the same connected component of G or G contains N + 2 nodes
08. if n_start and n_goal are in the same connected component of G then
09.     return a path between them
10. else
11.     return No Path
```

The above algorithm follows a single-query on-the-fly collision detection approach. The samples of C_{free} corresponding to the nodes in the graph G are generated while searching G and detecting collision on the fly. On each query, the graph is reconstructed. It is conceivable to generate G, store it and then search it each time we have a query. In this case, a sufficiently large G needs to be generated to cover potential queries. This is known as a multiple-query approach because several queries can be made on the same roadmap. A

delayed collision-checking approach would avoid checking collisions (Step 6) until a whole path has been found. If a segment on the path turns out to be colliding, the algorithm would backtrack to search for a new path. A delayed collision-checking method can outperform a non-delayed one on some planning domains [11, 58].

A PRM planner selects a node to expand in the free configuration space according to some given sampling measure. The efficiency of PRM approaches significantly depends on this measure [30]. A naive sampling measure will likely lose efficiency when the free space C_{free} contains narrow passages. A narrow passage is a small region in C_{free} where the sampling probability becomes very low. Some approaches exploit the geometry of obstacles in the workspace to adapt the sampling measure accordingly [9, 40, 41]. Other methods use machine learning techniques to adapt the sampling strategy dynamically during the construction of the probabilistic roadmap [41, 13, 31].

1.2.2 Path Planning with Preferences

In addition to collision avoidance, the concept of dangerous areas that have to be avoided as much as possible has been addressed in some path planning approaches [51, 59]. Herein we generalize the concept to preferences among regions in the C_{free} space. Different regions can be assigned different degrees of desirability, meaning that we would like the path planner to compute a path which not only avoids obstacles but also maximizes the degree of desirability for the path. Since path quality criterion may also depend on other metrics such as the distance along the path, we define the overall path quality as a trade-off between region desirability and distance. The trade-off is conveyed by a parameter weighing the contribution of each of these criteria to the global path-quality criterion. As a means to convey preferences among collision-free paths, region desirability provides a way to specify search control information for a path planner. It can be used to determine how the search process chooses the next node to expand.

1.2.3 Anytime Path Planning

In real-time applications involving the computation of an optimal solution, it is often desirable to have an incremental algorithm that computes its solution as a sequence of intermediate useful, but suboptimal solutions, converging towards an optimal solution. Dean

and Boddy called these anytime algorithms [20]. Such an algorithm guarantees a useful approximate solution at any time, which gets improved incrementally if more planning time is allowed. With a PRM planner, an anytime capability can be integrated into the search algorithm exploring the sampled roadmap. In particular, using the A* heuristic graph-search algorithm, one can implement a twofold anytime capability.

If given a transition function covering the entire search space and an admissible heuristic, A* guarantees finding an optimal solution. In our case, the search space is the sampled roadmap and a heuristic function $h(n)$ is a function taking a configuration n as input and returning the estimated distance from a configuration n to the goal configuration. It is admissible if it never overestimates the actual distance $h^*(n)$. We use the Euclidean distance as admissible heuristic. Obviously, the larger the search space, the more time it may take to find an optimal solution, even though A* does not have to exhaust the search space to guarantee optimality. To mitigate the combinatorial size of the state space, one can run A* on a smaller portion of the search space (producing an approximate path), then expand the search space and compute a new solution path for it and so on. This gives a sequence of solutions, converging to the optimal when the entire search space is covered. Given a deadline for computing a solution, the search space will be iteratively expanded accordingly ; a solution for each chunk of expansion is then computed. This implements an anytime capability through state expansion. The search process can be stopped at any time and give a solution (more precisely anytime after the time necessary for a first solution) and the more time it is given, the better the solution will be.

Another interesting property of A* is that, if given a non admissible heuristic $h(n) = h^*(n) + \epsilon$, then the cost of the path computed by A^* minus the cost of the optimal distance is less or equal than ϵ. In other words, ϵ is an upper bound on the error for the cost of the solution compared to the optimal solution. Moreover, A* tends to return a solution, possibly suboptimal, faster with inadmissible heuristics than with admissible heuristics. Based on these two observations and given an admissible heuristic h (e.g., the Euclidean distance), another way to implement an anytime A* search would be to compute a path using $h(n)+\epsilon_1$, then another solution using $h(n)+\epsilon_2$ and so on. In other words, a sequence of solutions using a decreasing error bound ϵ_i on the admissible heuristic is computed, with $\epsilon_{i+1} < \epsilon_i$. Given a deadline for computing a solution path, the inflating factor will be decreased iteratively, computing a solution for each decrease. This is the anytime capability

through heuristic improvement.

Both previous methods for implementing anytime capabilities with A* are complementary and can be combined as is the case in the Anytime Repairing A* (ARA*) algorithm [49]. We use a similar approach to explore a randomized flexible roadmap.

1.2.4 Dynamic Path Planning

In the ISS environment, most of the structure is fixed, only the robots can move. However, region desirability degrees can change as well as the goal. Regions of desirability depend on the task and the involved camera views. Depending on the orbit of the ISS, a camera may have its view towards the sun, making it undesirable. From the roadmap perspective, this means that the cost of a segment between two nodes can change dynamically. Such changes may invalidate a previously calculated path, either because it is no longer optimal or simply because it now leads to a dead-end. Re-planning is necessary in such cases.

Dynamic path planners adapt dynamically to change happening around the robot by repairing incrementally their representation of the environment. Different approaches exist that are extensions of the A* algorithm, including D* Lite [38], Anytime Dynamic A* (AD*) [48] and Generalized Adaptive A* (GAA*) [61]. These algorithms extends A* search to solve dynamic search problems faster by updating heuristics on nodes using knowledge acquired from previous searches.

1.3 FADPRM Path Planner

Combining region preferences, anytime search and dynamic re-planning, we obtain a flexible anytime dynamic probabilistic roadmap planner (FADPRM). The general idea is to keep track of milestones in an optimal solution to the goal. When changes are noticed, edges costs are updated and a new roadmap is re-computed fast, starting from the goal, taking into account previous traces of the path-calculation. This brings us back to a method in between the multiple query approach and the single query approach. The difference with a multiple query approach is that we are now only concerned with the roadmap to the current goal the robot is trying to reach in a dynamic environment.

FADPRM uses GAA* to explore the roadmap. The cost of an edge between two configurations depends on the actual distance between the configurations and the desirability degrees of the configurations along that edge. In a preliminary version of FADPRM [7], we have used AD* instead of GAA*. We now use GAA* instead of AD* because they have comparable performances, yet GAA* has a simpler description. Note that the contribution of FADPRM does not just amount to using GAA* to explore a sampled roadmap. The integration of preferences and their use to control both the path quality as well as the search-process for computing such a path are the key contributions.

1.3.1 Algorithm Sketch

FADPRM works with C_{free} segmented into zones, each zone being assigned a degree of desirability (dd), that is, a real number in the interval [0 1]. The closer is dd to 1, the more desirable the zone is. Every configuration in the roadmap is assigned a dd equal to the average of dd of zones overlapping with it. The dd of a path is an average of dd of configurations in the path. An optimal path is one having the highest dd.

The input for FADPRM is thus : a start configuration, a goal configuration, a 3D model of obstacles in the workspace, a 3D specification of zones with corresponding dd and a 3D model of the robot. Given this input :

1. To find a path connecting the input and goal configuration, we search backward from the goal towards the start (current) robot configuration. Backward instead of forward search is done because the robot moves ; we want to re-compute a path to the same goal but from the current position whenever the environment changes before the goal is reached.
2. A probabilistic priority queue $OPEN$ contains nodes on the frontier of the current roadmap (i.e., nodes that need to be expanded because they have no predecessor yet ; or nodes that have been previously expanded but are not being updated anymore) and a list $CLOSED$ contains non frontier nodes (i.e., nodes already expanded)
3. Search consists of repeatedly picking a node from $OPEN$, generating its predecessors and putting the new ones and the ones not yet updated in $OPEN$.
   (a) Every node n in $OPEN$ has a key priority proportional to the node's density and best estimate to the goal. The density of a node n, $density(n)$, reflects the

density of nodes around n and is the number of nodes in the roadmap with configurations that are a short distance away. The estimate to the goal, $f(n)$, takes into account the node's dd and the Euclidean distance to the goal configuration as explained below. Nodes in $OPEN$ are selected for expansion in decreasing priority. With these definitions, a node n in $OPEN$ is selected for expansion with priority proportional to

$$(1-\beta)/density(n) + \beta * f(n),$$

β is the inflation factor with $0 \leq \beta \leq 1$.

(b) To increase the resolution of the roadmap, a new predecessor is randomly generated within a short neighborhood radius (the radius is fixed empirically based on the density of obstacles in the workspace) and added to the list of predecessors in the roadmap generated so far ; then the entire list of predecessors is returned.

(c) Collision is delayed : detection of collisions on the edges between the current node and its predecessors is delayed until a candidate solution is found ; if colliding, we backtrack and rearrange the roadmap by eliminating nodes involved in this collision.

4. The robot may start executing the first path found.
5. Concurrently, the path continues to be improved.
6. Changes in the environment (moving obstacles and zones or changes in dd for zones) cause updating of the roadmap and replanning.

With β equal to 0, the selection of a node to expand is totally blind to zone degrees of desirability and to edges costs (Euclidian distance). Assuming $OPEN$ is the entire roadmap, this case corresponds to a normal PRM and the algorithm probabilistically converges towards an optimal solution as is the case for a normal PRM [58]. With $\beta = 1$, the selection of a node is a best-first strategy and by adopting an A*-like $f(n)$ implementation, we can guarantee finding an optimal solution within the resolution of the roadmap sampled so far. Therefore the expression $(1-\beta)/density(n) + \beta * f(n)$ implements a balance between fast-solution search and best-solution search by choosing different values for β.

Values of β closer to 1 give better solutions, but take more time. An initial path is generated fast assuming a value close to 0, then β is increased by a small quantity, a new path is computed again and so on. At each step, we have a higher probability of getting a better path (probability 1 when β reaches 1). This is the key in the anytime capability of our algorithm.

The heuristic estimate is separated into two components $g(n)$ (the quality of the best path so far from n to the goal configuration) and $h(n)$ (estimate of the quality of the path from n to the start configuration), that is, $f(n) = (g(n) + h(n))/2$; we divide by 2 to normalize $f(n)$ to values between $[0, 1]$. This definition of $f(n)$ is as in normal A* except that :

- We do backward search, hence $g(n)$ and $h(n)$ are reversed.
- The quality of a path is a combination of its dd and its cost in terms of distance traveled by the robot. Given $pathCost(n, n')$, the cost between two nodes, $g(n)$ is defined as follows :

  $$g(n) = pathdd(n_{goal}, n)/(1 + \gamma pathCost(n_{goal}, n))$$

  with $0 \leq \gamma \leq 1$.
- The heuristic $h(n)$ is expressed in the same way as $g(n)$ and estimates the cost of the path remaining to reach n_{start} :

  $$h(n) = pathdd(n, n_{start})/(1 + \gamma pathCost(n, n_{start}))$$

The factor γ determines the influence of the dd on $g(n)$ and on $h(n)$. With $\gamma = 0$, nodes with high dd are privileged, whereas with $\gamma = 1$ and with the dd of all nodes equal to 1, nodes with least cost to the goal are privileged. In the last case, if the cost between two nodes $pathCost(n, n'))$ is chosen to be the Euclidean distance, then we have an admissible heuristic and the algorithm is guaranteed to converge to the optimal solution. When dd's are involved and since zones can have arbitrary configurations, it is difficult to define admissible heuristics. The algorithm guarantees improvement of the solution, but it's impossible to verify optimality. Since the dd measures the quality of the path, the idea is to run the algorithm until a satisfactory dd is reached. The functions $pathdd$ and $pathCost$ are implemented by attaching these values to nodes and updating them on every expansion or when dynamic changes are observed in the environment.

1.3.2 Algorithm Details

The detailed structure of the FADPRM path planner is presented in Algorithm 2. Since FADPRM proceeds backwards, it updates h-values with respect to the start configuration of all expanded nodes n after every search as follows :

$$h(n) = g(n_{start}) - g(n).$$

Following GAA*, FADPRM does not initialize all g-values and h-values up front. Instead, it uses the variables *counter*, *search(n)* and *pathcost(x)* to decide when to initialize and update them by calling *UpdateState()* :

- The value of *counter* is x in the xth execution of $ComputeOrImprovePath$, that is, the xth call for GAA* on the roadmap.
- $search(n)$ stores the number of the last search that generated node n. FADPRM initializes these values to 0 for new nodes in the roadmap.
- $pathcost(x)$ stores the cost for the best path found on the roadmap by the xth search. More precisely, the formula for $pathcost(x)$ is :

$$pathcost(x) = g(n_{start}) = pathdd(n_{goal}, n_{start})/(1 + \gamma pathCost(n_{goal}, n_{start}))$$

Nodes in $OPEN$ are expanded in decreasing priority to update their g-values and their predecessors' g and h-values. The ordering of nodes in $OPEN$ is based on a node priority $key(n)$, which is a pair $[k_1(n), k_2(n)]$ defined as follows :

$$key(n) = [(1 - \beta)/density(n) + \beta f(n),\ h(n)],$$

with $f(n) = [(g(n) + h(n)]/2$ and $key(n) \leq key(n')$ if $k_1(n) \leq k_1(n')$ or $(k_1(n) = k_1(n')$ and $k_2(n) \leq k_2(n'))$. During the update on nodes, FADPRM initializes the g-value of nodes not yet generated by an already performed search, nodes with $search(n) = 0$, to zero.

In the function $ComputeorImprovePath()$, when a node n with maximum key is extracted from $OPEN$, we first try to connect it to n_{start} using a fast local planner as in SBL [58]. If it succeeds, a path is then returned (line 16). The expansion on a node n with maximum key from the $OPEN$ (line 18) consists of sampling a new collision-free node in the neighborhood of n [58] and then the sampled node is added in the set $Pred(n)$. After increasing the connectivity of the roadmap by adding a new node, FADPRM executes an

Algorithm 2 FADPRM Algorithm

```
01. KEY(n)
02.   f(n) = [g(n) + h(n)]/2;
03.   return [(1 − β)/density(n) + β.f(n); h(n)];

04. UPDATESTATE(n)
05.   if ( (search(n) ≠ 0) AND (search(n) ≠ counter) )
06.     if (g(n) + h(n) < pathcost(search(n))
07.       h(n) = pathcost(search(n)) − g(n);
08.       g(n) = 0;
09.   else if (search(n) = 0)
10.     g(n) = 0;
11.   search(n) = counter;

12. COMPUTEORIMPROVEPATH()
13.   while (NoPathfound)
14.     remove n with max key from OPEN;
15.     if (Connect(n, n_start))
16.       return β-suboptimal path;
17.     else
18.       ExpandNode(n);
19.       For all n' ∈ Pred(n)
20.         UPDATESTATE(n');
21.         g(n') = g(n) + c(n, n');
22.         insert n' into OPEN;
23.       insert n into CLOSED;

24. MAIN()
25.     counter = 1;
26.     g(n_start) = g(n_goal) = 0;
27.     search(n_start) = search(n_goal) = 0;
28.     β = β_0;
29.     OPEN = CLOSED = ∅;
30.     UPDATESTATE(n_start);
31.     UPDATESTATE(n_goal);
32.     insert n_goal into OPEN with key(n_goal);
33.   while (Not collision-free Path)
34.     Rearrange Tree;
35.     ComputeorImprovePath();
36.     counter = counter + 1;
37.     if (OPEN = ∅)
38.       pathcost(search(n)) = 0;
39.     else
40.       pathcost(search(n)) = g(n_start);
41.   publish current β_0−suboptimal solution;
42.   while (n_start not in neighborhood of n_goal)
43.     if (n_start changed)
44.       if (addtoTree(n_start))
45.         publish current solution;
46.     if changes in edge costs are detected
47.       for all changed edges (u, v)
48.         Update the edge cost c(u, v);
49.         UpdateState(u);
50.       Update the priorities for all
51.         n ∈ OPEN according to Key(n);
52.       CONSISTENCYPROCEDURE();
53.       decrease β or replan from scratch;
54.     if (β < 1)
55.       increase β;
56.     CLOSED = ∅;
57.     while (Not collision-free Path)
58.       Rearrange Tree;
59.       ComputeorImprovePath();
60.       counter = counter + 1;
61.       if (OPEN = ∅)
62.         pathcost(search(n)) = 0;
63.       else
64.         pathcost(search(n)) = g(n_start);
65.     publish current β−suboptimal solution;
66.     if (β = 1)
67.       wait for changes in edges cost;
```

update of the heuristics of all nodes in $Pred(n)$ in order to make them more informed and then allow for later more focused searches.

FADPRM updates the h-values of node n (line 7) if different conditions are satisfied :

- The node has not yet been generated by the current search ($search(n) \neq counter$)
- The node was generated by a previous search ($search(n) \neq 0$)
- The node was expanded by the search that generated it last ($g(n) + h(n) < pathcost(counter)$)

FADPRM sets $h(n)$ (line 7) to the difference between $g(n_{start})$ that is the cost of the path from n_{start} to n_{goal} during the last search that expanded n and $g(n)$ that remained the same since the same search. Dynamic changes in the environment affect (increase or decrease) edge costs. Such changes are handled by a consistency procedure, adapted from GAA*

and described in Algorithm 3. This procedure invoked at line 52 of the main algorithm whenever a cost decrease is observed. When invoked, it updates the h-values with respect to the start node.

Algorithm 3 Consistency Procedure

```
01. CONSISTENCYPROCEDURE()
02.   update the increased and decreased action costs (if any);
03.   OPEN = ∅;
04.   for all edges (n, n')
05.   with (n ≠ n_start) and edge cost c(s, s') decreased
06.     UPDATESTATE(n);
07.     UPDATESTATE(n');
08.     if (h(n) > c(n, n') + h(n'))
09.       h(n) = c(n, n') + h(n')
10.       if n ∈ OPEN
11.         delete n from OPEN;
12.       insert n in OPEN with key-value KEY(n);
13.     while (OPEN ≠ ∅);
14.       delete n' with smallest key-value from OPEN;
15.       for all states n ≠ n_start with succ(n) = n'
16.         UPDATESTATE(n);
17.         if (h(n) > c(n, n') + h(n'))
18.             h(n) = c(n, n') + h(succ(n'))
19.             if n ∈ OPEN
20.                 delete n from OPEN;
21.             insert n in OPEN with key-value KEY(n);
```

The Main procedure in FADPRM first sets the inflation factor β to a low value β_0, so that a suboptimal plan can be generated quickly (line 41). Then if no change in edge costs is detected, β is increased to improve the quality of its solution (lines 54-55). This will continue until the maximum of optimality is reached with $\beta = 1$ (lines 66-67).

FADPRM follows also the concept of lazy collision checking. Every time a β-suboptimal path is returned by ComputeorImprovePath(), it is checked for collision. If a collision is detected on one of the edges constituting the path, a rearrangement of the roadmap is then needed to eliminate nodes involved in this collision (lines 34, 58). FADPRM also handles the case of a floating starting configuration (lines 43-44).

1.4 Experimental Results

In a first set of experiments, we illustrate and validate the re-planning and anytime capabilities of FADPRM in dealing with highly complex environments with preferences. In a second set of experiments, we illustrate the search control capability of FADPRM and

show how region desirability specifications can help control the exploration of the sampled search space and make path-planning more efficient.

Experiments are made in two different environments : a simulation of the SSRMS on the ISS and a Puma robot operating on a car. The SSRMS is the most complex environment : 7 degrees of freedom, 75 obstacles modeled with 85 000 triangles. The Puma robot has 6 degrees of freedom and its environment is modeled by approximately 7 000 triangles.

All experiments were run on a 2,86 GHZ Core 2 Processor with 2GB of RAM. We consider paths with a dd of $0,5$ to be neutral, below $0,5$ to be dangerous and above to be desirable. More specifically, dangerous zones are given a dd of $0,2$ and desirable ones a dd of $0,8$. A free configuration of the robot not having any contact with zones is assigned a dd of $0,5$. We use *path-dd* as a measure for path quality. For all experiments, PRM refers to an implementation of SBL [58] for Single-query Bidirectional PRM-planner with Lazy collision detection.

1.4.1 Fast-Replanning Capability

In the SSRMS application, the concept of dangerous and desirable zones is motivated by a real-world application dealing with teaching astronauts to operate the SSRMS in order to move payloads or inspect the ISS using a camera mounted at the end effector. Astronauts have to move the SSRMS remotely, within safe corridors of operations. The definition of a safe corridor is that it must of course avoid obstacles (hard constraints), but also go as much as possible within regions visible through cameras mounted on the ISS exterior (so the astronaut can see the manipulations through a monitor on which the cameras are mapped). Hence, safe corridors depend on view angles and lighting conditions for cameras mounted on the ISS, which change dynamically with the orbit of the ISS by modifying their exposure to direct sunlight. As safe corridors are more complex to illustrate on paper, we just picked conical zones approximating cameras view regions and polygonal zones at arbitrary locations. Figure 1.2 illustrates a trajectory of the SSRMS carrying a load and going through three cameras fields of view (purple, green and blue cones) and avoiding an undesirable zone with very limited lighting conditions (rectangular zone in pale red).

The first experiment illustrates the situation in which a human operator is learning to manipulate the SSRMS from a given start configuration to a given goal configuration. To

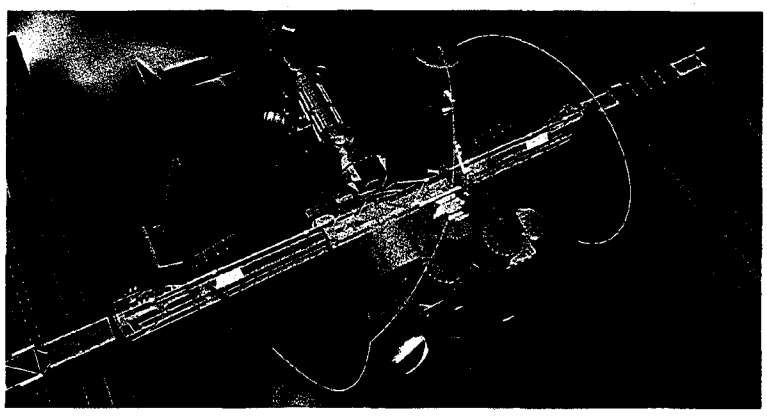


figure 1.2 – SSRMS going through three different cameras fields of view (purple, green and blue cones) and avoiding an undesirable zone (rectangular zone in pale red)

provide feedback on whether he is on the right track, from every current configuration, we call the FADPRM planner to calculate a path with a high *dd* to the goal. If no such a path exists, we notify the learner that he is moving the SSRMS to a dead end. Although paths are computed to confirm the learner is on the right track, they are not displayed to him. Hence while the learner is making suitable progress toward the goal, they are solving the problem on their own.

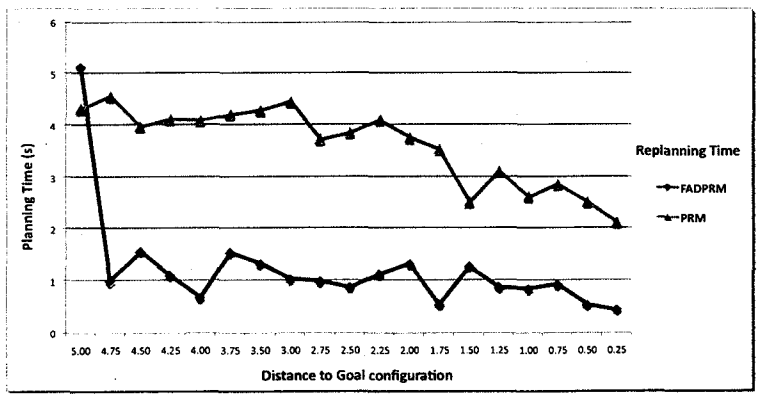


figure 1.3 – FADPRM versus PRM in replanning

Figure 1.3 shows the time taken for replanning while the human operator is moving the robot toward a goal configuration in the scenario of Figure 1.2. We conducted the experiment three times with the operator doing exactly the same manipulations to reach the goal

from the start configuration and each time using FADPRM (with $\beta = 0$) and the normal PRM. Except for the first few iterations, FADPRM take less re-planning time than PRM. For FADPRM and in the first few iterations, the overhead incurred by the GAA*-based exploration dominates the planning time. In later iterations, it is outweighed by the savings gained by re-planning from the previous roadmap.

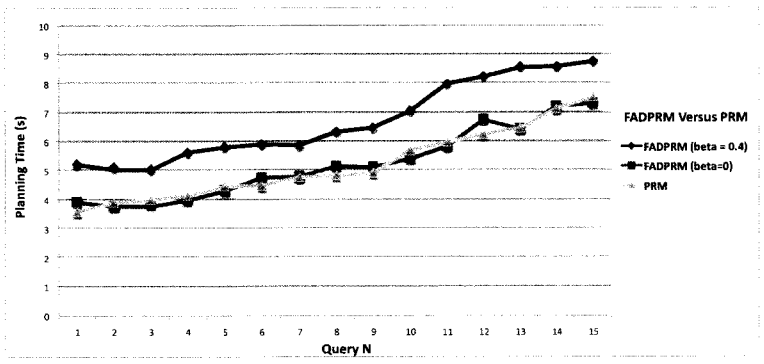


figure 1.4 – FADPRM versus PRM in planning time

In Figure 1.4, we compare the time needed for FADPRM and PRM to find a solution for 15 arbitrary queries in the ISS environment. Since the time (and path quality) for finding path is a random variable given the random sampling of the free workspace, for each query we ran each of the planners ten times and reported the average planning time. In this case, FADPRM is used in a mode that does not store the roadmap between successive runs. Before displaying the results, we sorted the PRM setting in increasing order of complexity, starting with queries taking less time to solve.

For FADPRM, we show results with $\beta = 0$ and $\beta = 0,4$. With $\beta = 0$, FADPRM behaves exactly like the normal PRM. With $\beta = 0,4$, planning takes more time for both planners. This validates our previous analysis about FADPRM : with $\beta = 0$, an FADPRM planner behaves in way very similar to a normal PRM, but as soon as we start seeking optimality (in our case with $\beta = 0,4$), the time for planning will increase proportionally.

On an other hand, Figure 1.5 shows that $\beta = 0,4$ yields higher quality paths than $\beta = 0$. This validates another previous analysis : higher β values yield better paths, but take more time to compute.

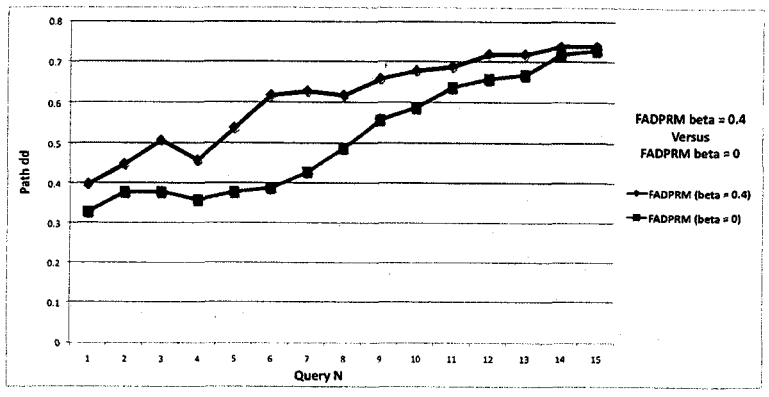
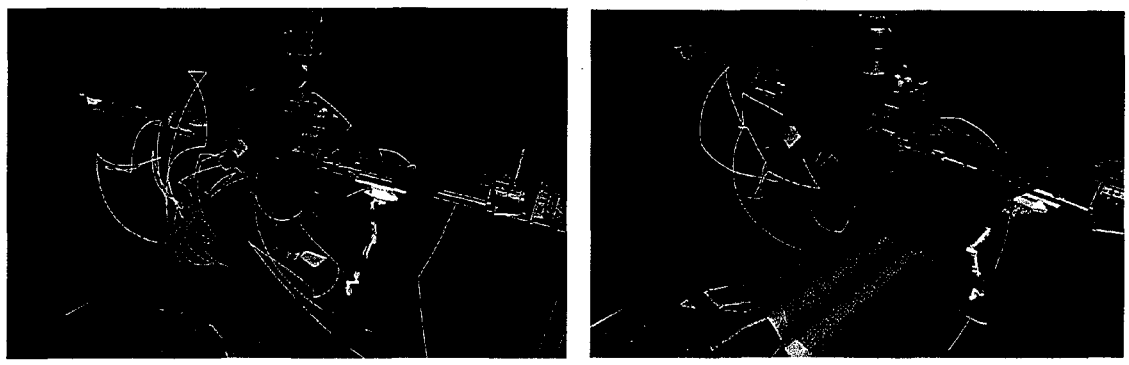


figure 1.5 – FADPRM versus PRM in path quality (Path-dd)

1.4.2 Search Control Capability

Specifying zone degrees of desirability provides a means to accelerate the computation of a path. Since FADPRM explores configurations in the order of specified preferences, it is possible to control the search process, via the specification of regions with suitable degrees of desirability, so it reaches a solution very quickly on average.



(a) Scenario 1 with one desirable zone

(b) Scenario 2 with two desirable zones and one danger zone

figure 1.6 – Not smoothed paths with FADPRM ($\beta = 0,7$) in the ISS environment

To illustrate the search control capability in the ISS environment, we establish a planning scenario where the SSRMS has to carry a load to a space shuttle docked to the ISS. Path Planning is further made more complex in this scenario with the final configuration placed in a narrow passage : near the shuttle and surrounded by a number of modules as

shown on Figure 1.6. Normal PRMs start losing efficiency in such areas since the sampling probability becomes very low. In the first experiment (Figure 1.6(a)), we plan for a path with a wide desirable zone on the left of the shuttle. In the second experiment (Figure 1.6(b)), we specify an undesirable zone on the right of the shuttle and two wide desirable zones on the front and on the left of the shuttle. By adding zones with appropriate degrees of desirability, we wanted to influence the sampling of the free workspace to yield better paths. Here, planning is done without any call to a post-processing smoothing step as is usually done with normal PRM planners [58]. This explains why we have awkward trajectories on both figures (Figure 1.6).

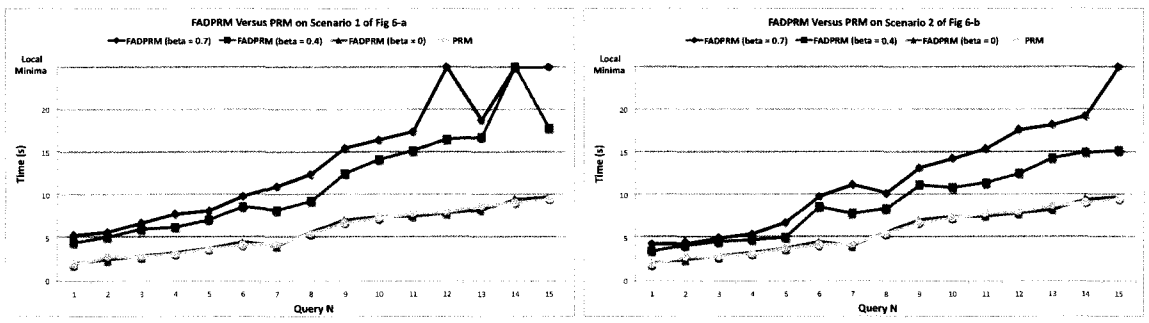


figure 1.7 – FADPRM ($\beta = 0,4$ and $\beta = 0,7$) versus PRM in planning time

In Figure 1.7, we compare the time needed for FADPRM and PRM to find a solution for 15 different queries within both scenarios of (Figure 1.6(a)) and (Figure 1.6(b)). For all queries, the goal configuration with the carried module inside the shuttle remains the same. Start configurations are picked randomly at different locations around the shuttle. For each query we ran each of the planners ten times and reported the average planning time. Before displaying the results, we sorted the PRM setting in increasing order of complexity, starting with queries taking less time to solve. For FADPRM we show the results with $\beta = 0$, $\beta = 0,4$ and $\beta = 0,7$. In these experiments, The bias factor γ that determines the influence of the dd on the cost of edges within the roadmap is equal to $0,5$.

With $\beta = 0$ FADPRM behaves exactly like a normal PRM in both scenarios yielding very complex awkward paths requiring an approximately equivalent time to compute. As soon as we start seeking optimality with $\beta = 0,4$ and $\beta = 0,7$, the time for planning in the two scenarios increases. In both scenarios, FADPRM with $\beta = 0,7$ yield better paths than FADPRM with $\beta = 0,4$ but takes more computing time. Figure 1.8 confirms this

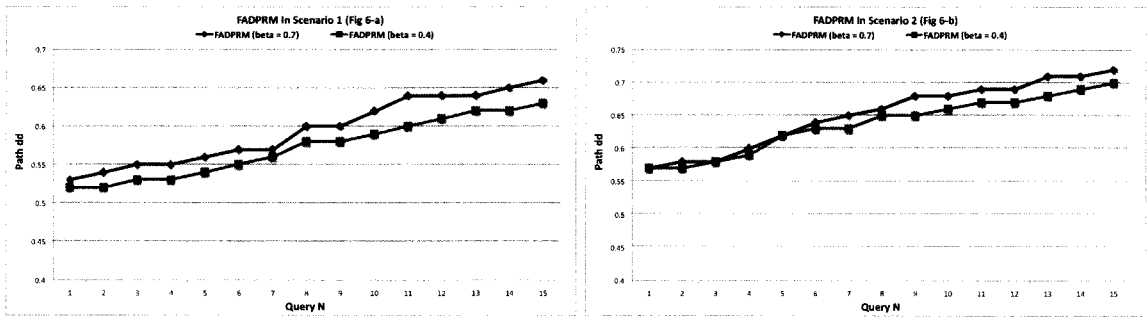


figure 1.8 – Path quality with FADPRM ($\beta = 0,4$ and $\beta = 0,7$)

and shows better quality (i.e., better dd's) for paths computed with FADPRM ($\beta = 0,7$) compared to paths found by FADPRM ($\beta = 0,4$).

If we compare the time taken for planning with the same version of FADPRM ($\beta = 0,4$ or $\beta = 0,7$) within the two different scenarios of Figure 1.6, we notice that more planning time is needed in the scenario of Figure 1.6(a). With larger values of β, the sampling with FADPRM is pushed into areas with high values of dd. If the workspace is not covered with enough desirable zones, the planner may remain stuck sampling within one desirable zone of the workspace. This explains the problem of local minima we see on Figure 1.7 with FADPRM in scenario of Figure 1.6(a). The more β is increased, the more the sampling is pushed within high dd zones. This explains why the local minima problem is more frequent with FADPRM with $\beta = 0,7$. By increasing the coverage of the workspace with more desirable and undesirable zones like in Figure 1.6(b), we significantly improve the planning time needed for finding an optimal solution and considerably reduce the probability of having the local minima problem (Figure 1.7).

The factor γ determines the influence of the dd on $g(n)$ and on $h(n)$. With $\gamma = 0$, nodes with high dd are privileged, whereas with $\gamma = 1$ and with the dd of all nodes equal to 1, nodes with least cost to the goal are privileged. In the following experiment, we test the influence of different values of γ on the planning time with FADPRM ($\beta = 0,7$) in the scenario of Figure 1.6(b) with a well constrained workspace. Seeking optimality in the robot path takes more time and can lead to local minima problems. With FADPRM, the local minima problem can have two reasons :

1. $distance$-minima problem : the planner remains stuck sampling without success in a narrow passage around a configuration too close to the $goal$-configuration.

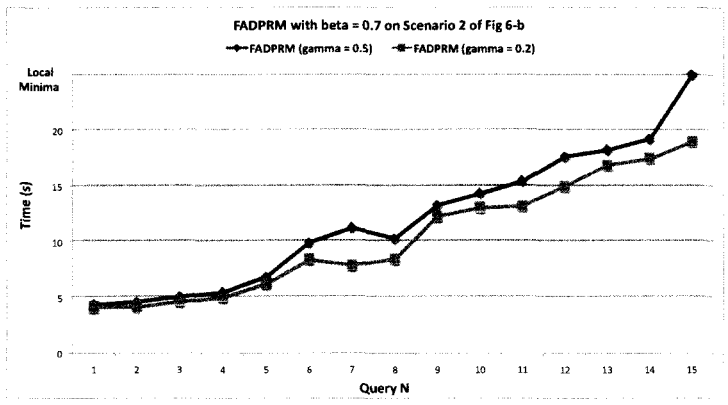


figure 1.9 – Planning time with FADPRM ($\beta = 0,7$) with $\gamma = 0,5$ and $\gamma = 0,2$

2. dd-minima problem : the planner remains stuck sampling without success within a tiny desirable zone of the workspace like in the scenario of Figure 1.6(a).

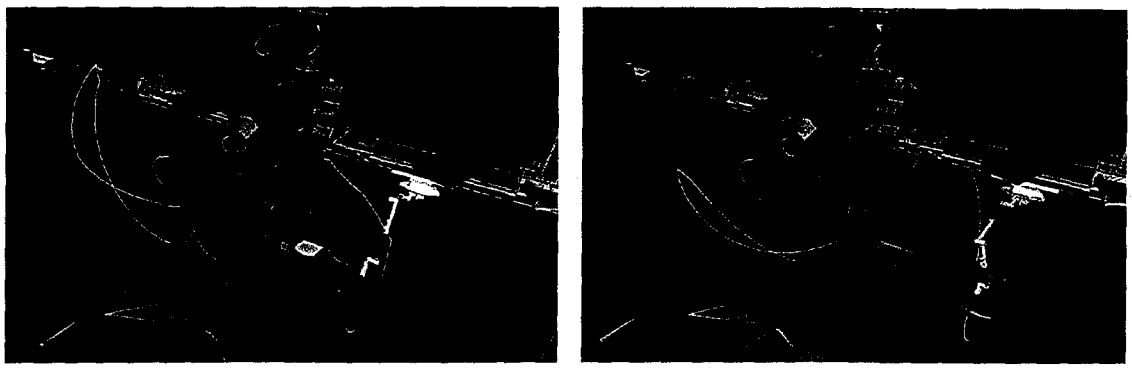
By increasing the influence of dd on the cost of nodes, we reduce the probability of having the $distance$-minima problem. By increasing the coverage of the workspace with desirable and undesirable zones, we reduce the probability of having the dd-minima problem. This explains why in Figure 1.9, FADPRM with $\gamma = 0,2$ take less time for planning compared to FADPRM with $\gamma = 0,5$. More importantly, no occurrence of the local minima problem is observed with FADPRM ($\gamma = 0,2$).

1.4.3 Path-Quality Control Capability

With randomized path planners, paths are obtained by connecting milestones that are randomly sampled in the free workspace and this tends to yield awkward paths, requiring heuristic post-processing operations to smooth them. With FADPRM, it is possible to influence the sampling strategy and generate less awkward paths by specifying zones we prefer them to go through or zones we want them to avoid.

In Figure 1.6, we notice an improvement in the smoothness of the path generated with FADPRM on the scenario of Figure 1.6(b) compared to the path on Figure 1.6(a). In Figure 1.11, we measure the time needed for smoothing the paths (as shown on Figure 1.10) found on the 15 queries of Figure 1.7.

Compared to normal PRM, FADPRM (with $\beta = 0,4$ or $\beta = 0,7$) always produces



(a) with one desirable zone (b) with two desirable zones and one danger zone

figure 1.10 – Smoothed paths with FADPRM in the ISS environment ($\beta = 0,7$)

paths that need less post-processing smoothing time. In both scenarios, FADPRM with $\beta = 0,7$ needs less time for smoothing than FADPRM with $\beta = 0,4$. Here, we notice that the more we seek optimality in the robot path, the less awkward solution paths are, which explains why they require less time to smooth.

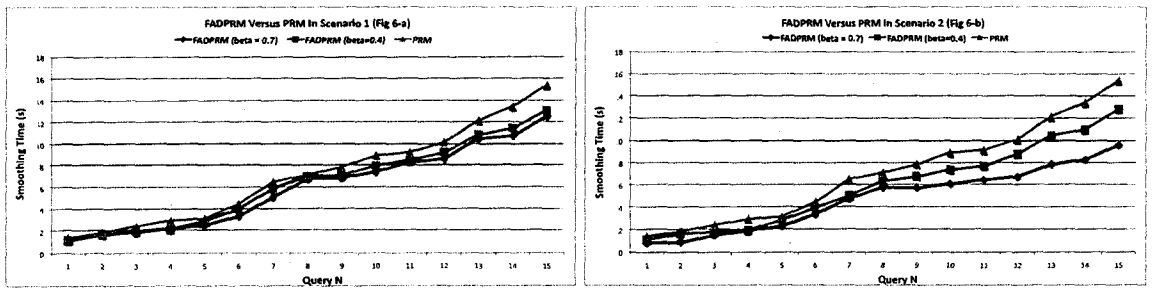


figure 1.11 – FADPRM versus PRM in smoothing time

We also notice that for both versions of FADPRM (with $\beta = 0,4$ or $\beta = 0,7$), more smoothing time is required in the first scenario on Figure 1.6(a). The more the path is constrained with desirable and undesirable zones within the workspace, the more quality and efficiency we guarantee in the solution path. That is, the path is smoother and requires less time to smooth.

Figure 1.12 confirms the same results found with not smoothed solution paths on Figure 1.8. Increasing β into FADPRM yields smoothed paths with better quality in terms of degrees of desirability. Also the more we cover the workspace with desirability zones, the

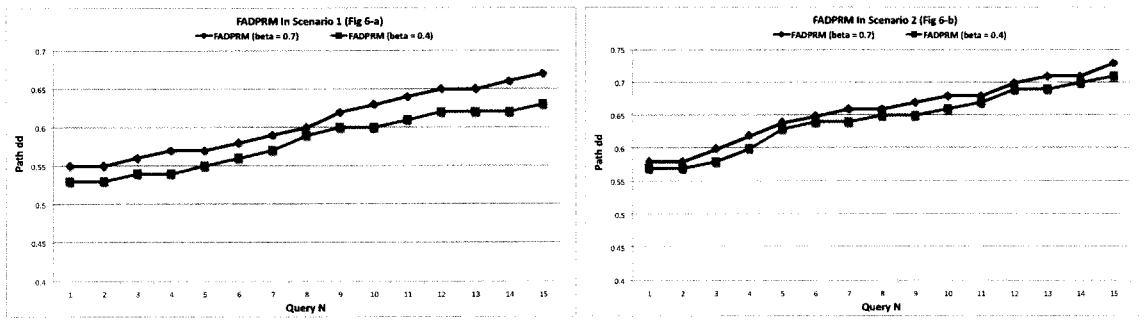


figure 1.12 – FADPRM versus PRM in path quality of smoothed paths

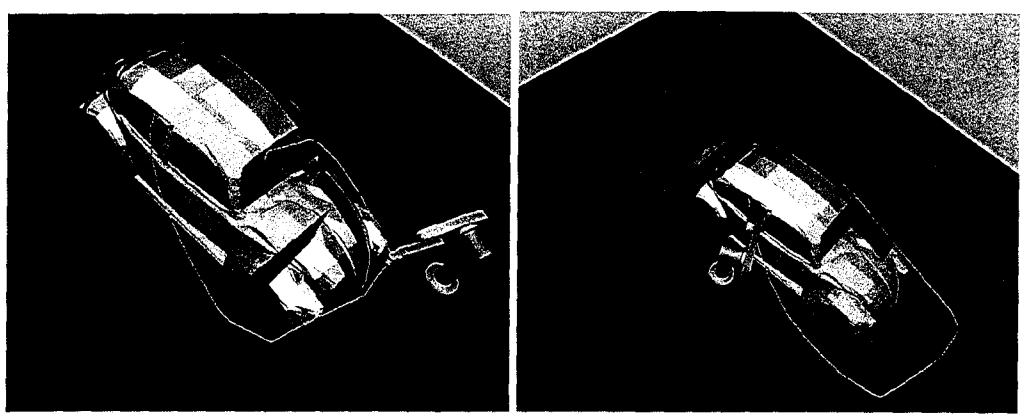
more this path quality is enhanced. Zones with degrees of desirability provide a mean to specify a sampling strategy that controls the search process to generate better paths (better dd's and better smoothness) by simply annotating the 3D workspace with the regions' degrees of desirability.

1.4.4 Generality of the Results

By running the previous experiments on several randomly chosen samples and reporting the average results, we somewhat try to verify that FADPRM features are not dependent on one specific scenario. Obviously, the ISS domain has a specific structure. Runs on different domains are necessary to increase confidence in the generality of the observed performances of FADPRM. In this regard, we did numerous experiments on a simulated Puma robot operating on a car. The obtained results confirm those in the SSRMS domain.

For instance, to evaluate the search-control capability of FADPRM, in one of the experiments we specified a small desirable zone (see Figure 1.13(a)) and in another, we specified a wider desirable zone (in front of the car) and a wide undesirable zone (in the back) (see Figure 1.13(b)). In both set of experiments, we wanted to influence the sampling of the free workspace to yield paths that move the robot in front of the car (from the left side, to the front, then to the right side).

By specifying a desirable zone on the right front of the car as shown in Figure 1.13(a) and running FADPRM planner many times on the same query (input/goal configuration), they yielded better paths, on average, than PRM. On the other hand, by enlarging the size and coverage of the desirable zone and adding an undesirable zone (right, on the back of



(a) (b)

figure 1.13 – PUMA robot around a car

the car), as shown in Figure 1.13(b), we noticed that the quality of paths increased by a percentage of 50% over 100 trials. The second experiment succeeds more often because the path is more constrained ; a wider desirable zone on the front of the car together with an undesirable zone on the back of the car, make the probability of sampling a configuration along the desirable region higher than in the first set-up.

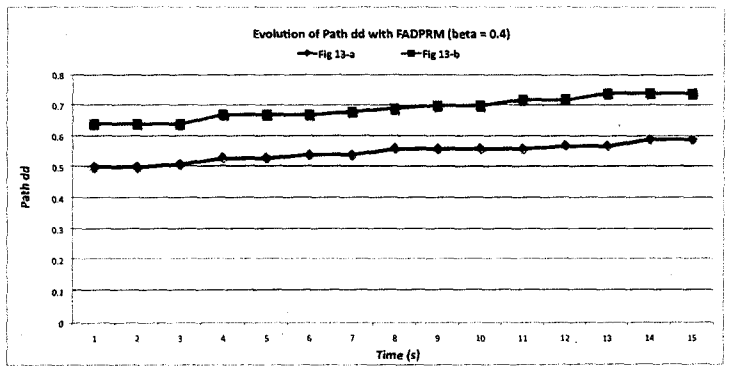


figure 1.14 – Path quality evolution with FADPRM

Figure 1.14 shows the anytime capability of FADPRM (with $\beta = 0,4$) on these two experiments. We notice the continuous improvement of the path quality ($path - dd$) for the two settings. The more time it is given, the better the path provided by FADPRM will be. The results here confirm the observations noticed in the first experiment with the SSRMS. Handling zones with degrees of desirability provides FADPRM with a powerful sampling

strategy that helps generate better quality paths. And the better is the coverage of the workspace with preference zones, the more optimal (in terms of degrees of desirability), the solution path to which FADPRM converges will be.

1.5 Conclusion

In many real world path planning applications, in addition to obstacles that must be avoided, we may have areas that the body is preferred to avoid (or, conversely, to go through) as much as possible. This is the case in the ISS domain where preferences are tied to camera views which change dynamically, in part because of varying environmental conditions throughout the orbit.

In this paper, we presented a new randomized approach for robot path planning which extends the PRM framework to handle a workspace containing regions with degrees of desirability. Our approach integrates dynamic and anytime search exploration strategies to deal with problems in dynamic environments where obstacles and region desirability can change in real-time. The dynamic strategy allows the planner to re-plan efficiently by exploiting results from previous planning iterations. The anytime strategy starts with a quickly computed path with a potentially low degree of desirability which is then incrementally improved if more planning time is allowed.

The experiments validated the different features of FADPRM on two particular path-planning domains. Although we obtain good path quality and better re-planning time than normal PRM approaches, there remains potential for improvement on both dimensions. Paths still need to be smoothed in post-processing and for real-time applications we still want a planning algorithm that is as fast as possible. We will therefore continue to explore ways to improve our approach and look for alternatives.

FADPRM is a component in a large simulation prototype for training astronauts on the SSRMS. It is invoked by an intelligent tutoring system (ITS) to monitor robot operations carried out by a student and provide feedback on how to control the arm. For instance, given the task of moving the SSRMS from one configuration to another, the ITS can try computing a path from the current configuration to the goal and advise the student when the current configuration seems to be a dead-end. The student can then backtrack to a previous point from which better paths to the goal are available.

1.6 Acknowledgement

The work presented herein was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Chapitre 2

Planification automatique de caméras pour filmer des opérations robotisées

Résumé

Dans le monde du cinéma, les metteurs en scène ont élaboré des principes et des règles cinématographiques pour montrer la façon dont différentes catégories de scènes impliquant l'interaction entre personnages doivent être filmées [2, 50]. Inspirées par ces méthodes, différentes approches pour le contrôle automatique de la caméra ont émergé dans le domaine de l'infographie et dans quelques domaines connexes comme les jeux vidéo, l'animation par ordinateur et la réalité virtuelle [5, 25, 33]. Ces approches permettent la génération automatique d'animations 3D avec haute précision sans ou avec un minimum de support des animateurs et des programmeurs.

Dans l'article qui suit, nous présentons une nouvelle approche de planification de caméras qui utilise la logique temporelle linéaire (LTL) [3] pour spécifier les principes et règles cinématographiques. Par rapport aux approches existantes dans la littérature et avec l'utilisation de LTL, nous obtenons un langage avec une sémantique plus claire et une syntaxe plus intuitive pour exprimer les règles de composition cinématographique. L'algorithme de planification sous jacent TLPlan [3] est également plus efficace que les planificateurs utilisés dans les

approches précédentes. En effet, TLPlan implémente une stratégie de contrôle de la recherche lui permettant de restreindre la planification seulement sur les états compatibles avec les formules LTL objectifs instaurées, qui dans notre cas transmettent les règles de composition cinématographiques.

Le nouveau planificateur de caméras est intégré dans ATDG, un système de planification automatisée pour la production de démonstrations de tâches 3D impliquant la manipulation du SSRMS sur la station spatiale internationale (SSI). Une tâche de manipulation typique consiste à déplacer le bras robot d'une configuration à une autre dans l'environnement de la SSI. Le principal défi est de planifier automatiquement les meilleures configurations de caméras pour filmer le bras de la manière la plus compréhensible par l'utilisateur. La trajectoire du robot est générée en utilisant le planificateur de trajectoires FADPRM (voir chapitre précédent). Le planificateur de caméras TLPlan est alors invoqué pour trouver la meilleure séquence de configurations de caméras filmant le robot sur sa trajectoire.

Commentaires

L'article présenté dans ce chapitre a été soumis au journal "International Journal of Knowledge-Based and Intelligent Engineering Systems". Une version plus courte de ce travail a été publiée dans les actes de "International Conference on Automated Planning and Scheduling" [34]. Ces travaux ont été effectués en collaboration avec Philipe Bellefeuille dans le cadre de sa maîtrise en informatique à l'Université de Sherbrooke. L'adaptation de TLPlan pour filmer des opérations robotisées a été implémentée par Philipe. L'ensemble de ces travaux ont été réalisés, validés et rédigés sous la supervision du Professeur Froduald Kabanza (Université de Sherbrooke) et avec les conseils du Professeur Leo Hartman (Agence spatiale canadienne).

Automated Camera Planning To Film Robot Operations

Khaled Belghith, Froduald Kabanza, Philipe Bellefeuille
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
khaled.belghith@usherbrooke.ca, kabanza@usherbrooke.ca,
philipe.bellefeuille@usherbrooke.ca

Leo Hartman
Canadian Space Agency,
John H. Chapman Space Centre, 6767 Route de l'Aéroport
Saint-Hubert, Québec J3Y 8Y9
leo.hartman@asc-csa.gc.ca

Abstract

Automatic 3D animation generation techniques are becoming increasingly popular in different areas related to computer graphics such as video games and animated movies. They help automate the filmmaking process even by non professionals without or with minimal intervention of animators and computer graphics programmers. Based on specified cinematographic principles and filming rules, they plan the sequence of virtual cameras that the best render a 3D scene. In this paper, we present an approach for automatic movie generation using linear temporal logic to express these filming and cinematography rules. We consider the filming of a 3D scene as a sequence of shots satisfying given filming rules, conveying constraints on the desirable configuration (position, orientation, and zoom) of virtual cameras. The selection of camera configurations at different points in time is understood as a camera plan, which is computed using a temporal-logic based planning system (TLPlan) to obtain a 3D movie. The camera planner is used within an automated planning application for generating 3D tasks demonstrations involving a teleoperated robot arm on the the International Space Station (ISS). A typical task demonstration involves moving the robot arm from one configuration to another. The main challenge is to automatically plan the configurations of virtual cameras to film the arm in a manner that conveys the

best awareness of the robot trajectory to the user. The robot trajectory is generated using a path-planner. The camera planner is then invoked to find a sequence of configurations of virtual cameras to film the trajectory.

2.1 Introduction

Filmmakers have developed different cinematography principles and rules stating how different categories of scenes evolving interacting characters should be filmed [2, 50]. Inspired by these methods, different methodologies for automatic camera control have emerged in computer graphics and related areas such as computer games, computer animation and virtual reality [5, 25, 33]. They allow the automatic generation of accurate 3D animations without or with minimal support from computer graphics professionals. An example of application is the automatic generation of movies showing a replay in a game such as soccer, hockey or basketball. In this paper, we present a camera planning approach that uses Linear Temporal Logic (LTL) [3] to specify cinematographic principles and filming rules. This is a more detailed account of the work presented in [34]. Compared to existing planning approaches, by using LTL we get a language with clear semantics for expressing camera planning rules and with an intuitive syntax. The underlying TLPlan planning algorithm [3] is also more efficient than the planners underlying previous approaches, because of the intrinsic ability of TLPlan to prune the search space by removing state trajectories inconsistent with the input LTL goal formula. In our case the LTL goal formula specifies composition rules.

Camera planning approaches found in the literature rely on filming primitives that capture and convey cinematographic principles specific to the domain of actors. These primitives are not relevant for filming a robot arm (since the points of interests and types of actions are different), but they can be adapted. As part of this adaptaion, in particular, we had to develop an ontology of meaningful elementary movements for robot arms relating to task descriptions.

The camera planner is integrated into an Automatic Task Demonstration Generator or ATDG, a fully automated system that we developed to generate 3D task demonstrations of the manipulations made by an astronaut with a simulated Space Station Remote Manipulator (SSRMS). The SSRMS is a 17-meter long articulated robot arm on the International

Space Station (ISS). It has a complex geometry, with seven rotational joints, each with a range of ± 270 degrees. Astronauts operate the SSRMS through a workstation located inside one of the ISS compartments. Tasks consist of manipulating the SSRMS from one configuration to another in order, for example, to move a payload or inspect a region of the ISS exterior using a camera mounted on the end effector. The current ATDG prototype is integrated within a proof-of-concept simulator called ROMAN Tutor (RObot MANipulation Tutor) for the command and control of the SSRMS that we have also developed [35]. Figure 2.1 shows a snapshot of the simulator.

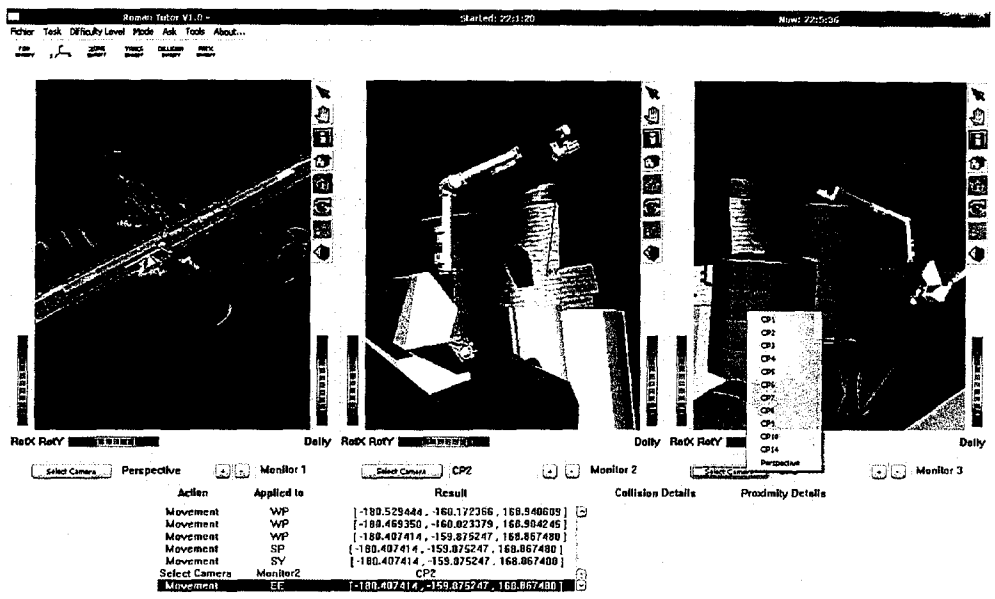


Figure 2.1: Roman tutor interface

One motivation for this application is to support ground operators in planning SSRMS manipulation tasks for the ISS. Given a new task, or given changes to a task previously planned, ATDG automatically and efficiently generates 3D demonstrations of the task without or with minimal intervention of computer graphics programmers. Another motivation is to use automatically generated demonstrations in a 3D training simulator to provide feedback to student astronauts learning to manipulate the robot arm.

In the next section, we give the most relevant background. We then describe the Automatic Task Demonstration Generator, the system application that motivated our research. After a detailed study of our temporal camera planning approach in a next paragraph, we

illustrate with some experiments that shows some of its merits.

2.2 Background

2.2.1 Virtual Camera and Virtual Camera Planning

An animation is a rapid display of a sequence of images. To create the illusion of movement, an animation displays from 24 to 30 images per second (frames per second or fps). On each frame, a 3D virtual camera has at least seven degrees of freedom for position, orientation and field of view (Figure. 2.2). Planning camera configurations consists of generating a series of camera configurations, each corresponding to one frame of the desired final movie. The search space for possible solutions is then very large.

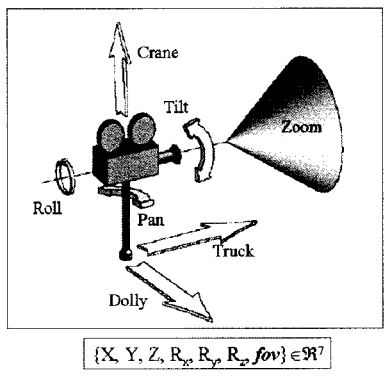


Figure 2.2: Degrees of freedom of a virtual camera

2.2.2 Scenes, Shots and Idioms

To help simplify the calculation of the frame sequences composing a film, camera planning approaches borrow some concepts from the world of cinematography to abstract the camera configuration's search space and reduce its combinatorial complexity. Specifically, a movie is hierarchically decomposed into *scenes* and *shots* (Figure 2.3), and the concept of *idiom*.

A scene is a distinct narrative unit usually characterized by unity of location or unity of time. For the case of robotic manipulations, we consider a scene as a distinctive movement

of the robotic arm (e.g., translation along the mobile base, or moving the end effector towards a given position). A scene is defined as a sequence of shots. A shot in turn is defined as a continuous succession of images taken by one camera (each generally lasts a few seconds). Figure 2.3 illustrates this film hierarchy. [1]

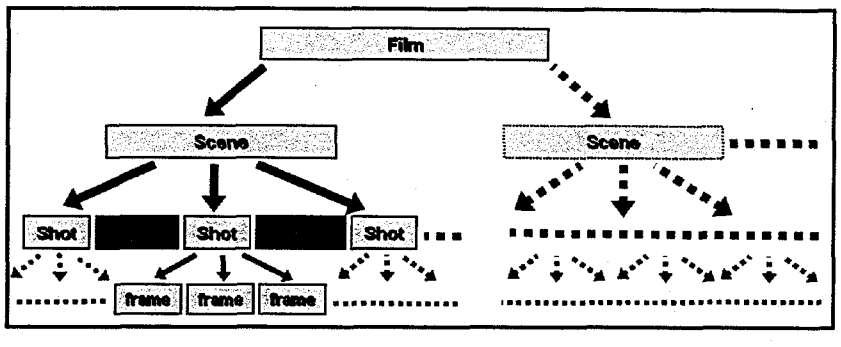


Figure 2.3: Film abstraction hierarchy

An idiom describes how a scene can be filmed, that is, how the different shots composing a scene can be taken. For instance, an idiom for a conversation between two people will specify the number of shots and how the camera for each shot is placed and configured. In general we have different ways of filming a scene. Hence we have different idioms for a scene. But in a film, for each scene only one specific idiom will be selected and applied. Similarly, for each scene illustrating a distinctive movement of the SSRMS, different idioms for filming the movement can be applied.

2.2.3 Camera Planning Approaches

Given the previous hierarchical movie decomposition, the problem of automatically generating a movie essentially becomes one of determining what shot to make at different segments of the scene. This is often understood as a camera planning problem, that is, determining the sequences of shots given a goal corresponding to what meaning should be conveyed by the movie (e.g., in our application a looming collision or the arm going too fast ; in other applications a person running, a happy person or a conversation among a

1. A more general film hierarchy considers two additional granularity levels, where a film consists of acts, an act comprises sequences, and a sequence contains one or more scenes. Our hierarchy stops at the scene level.

group of people).

Previous automated camera planning approaches in the literature can be grouped into two main categories: constraint approaches and cinematographic approaches. Constraint approaches film a scene (i.e., its objects or background, static or in motion) based only on the geometry of the scene. Constraints on the camera (position, view angle, zoom, covered objects or regions) are essentially geometric (e.g., filming a moving character during 20 minutes, without any occlusion of the camera). There are no explicit constraints associated with actions involving objects in the scene. For instance, in previous approaches we cannot express a constraint such as "if the person is moving then his legs must remain in the camera view angle during the motion".

One of the earliest attempt to put these concepts into practice was made by Blinn [10]. His algorithm could find the required camera configuration (position and orientation) starting from some constraints on the content of the images to produce. Later, more sophisticated constraint camera planning systems were proposed. The system developed in [8] uses camera motions constraints expressed using a system of polynomial equations. The Cinema system [21] has a procedural interface for specifying camera movements relative to objects, events, and the general state of an environment. The Virtual Cameraman [32, 42] provides the user with camera movements satisfying user defined constraints specified in the image space and/or constraints on the objects of the scene. Bares et al. developed ConstraintCam, a real-time camera engine for dynamic 3D environments [4]. Three classes of constraints are implemented within ConstraintCam: viewing angle, viewing distance and occlusion avoidance. CamPlan [28], a camera planning subsystem for polygonal graphics, uses a genetic algorithm to optimize the camera with respect to a set of image objectives.

The approach in [53] could also be seen as a constraint or geometric approach, although not explicitly introduced that way. Camera placements are seen as configurations in a probabilistic roadmap; a camera plan (for example, to navigate in a museum from one point to another) becomes a trajectory in the roadmap, obtained using probabilistic roadmap algorithms normally used for articulated bodies (e.g., robots) combined with some post processing to smooth the view angles and the zoom.

Constraint approaches operate at the frame level and are not appropriate for filming a complex task (e.g., a moving robot) during which the camera has to constantly adapt to the context in which the task is being carried out. In other words, they cannot dynamically take

into account the semantics associated with actions in the task. In fact, for a dynamic handling of the action semantics, we need to use cinematographic camera planning approaches which operate at the shot level.

At the shot level, rules for composing (or sequencing) shots depending on the type of action being filmed are specified. This way, a scene can be filmed while taking into account constraints on actions in addition to geometric constraints. For example, we may want to specify a rule such as if you are filming a sequence of shots showing one person talking to another, the faces of both people should remain in the camera view angle. Or, if the end effector of the SSRMS is aligned with some visual cue on the ISS then we would like to film it with the cue in view (e.g., near the Canadian flag for example) and thereby make the astronaut more aware of the current configuration of the robot.

The Camera Planning System (CPS) of [15] is an example of a cinematographic camera planner. It automatically generates camera positions based on idioms specified in a declarative camera control language (DCCL). The language provides four primitive concepts, namely, fragments, views, placements, and movement endpoints. By camera placement we mean a continuous sequence of camera configurations over a period of time (or equivalently, from one point of the scene to another). These quite intuitive primitives are then combined to specify higher-level constructs such as shots and idioms. An example of a fragment is a go-by shot which represents a static camera overseeing a given region for a given period. Using these DCCL primitives, the user can specify idioms indicating how to film objects of interest. For example, we can specify how to film two people approaching each other: the camera could show one person advancing at some time; then switch to a close position where we see both of them, and finally slightly moving away from the target. The generic aspect of the system comes from the fact that the same scene can be filmed the same way regardless of the involved characters. Also, one can change the filming of the scene by specifying a different idiom, without going into the details of the computer graphics, which are filled in automatically through the primitive operators. The approach in [5] is analogous, with the difference that users can specify visual preferences about how objects should be filmed. Given such preferences, the system UCAM selects the best camera placements that satisfy those preferences.

The approach in [27] combines constraint (frame level) and cinematographic (shot level) approaches to generate more refined films. The algorithm takes as input a geo-

metric description of the scene (e.g., the object in the scene one would like to see) and plans the camera positions accordingly, while taking into account cinematographic principles regarding camera placements. Geometric frame-level constraints are solved using a contraint-solving technique, in which constraints can relate to the height angle, distance to the target, and orientation of the camera.

Tomlinson *et al.* [62] implemented an autonomous cinematography system based on the autonomous character design work of the Synthetic Characters Group at the MIT Media Lab. Their system used expressive and user-controlled characters to drive the on-the-fly selection of shots in a virtual collaborative 3D environment. Friedman *et al.* [25, 26] implemented a knowledge-based system that allowed users to express cinematography preferences in a specific language and produce automatic animated 3D movies accordingly. More recently, Jhala and Young [33] developed a camera planning system that mirrors the film production pipeline in narrative-oriented virtual worlds. In more of finding the best framing for shots to determine geometrically smooth transitions between them, their system exploits the narrative structure and the causal relationships between shots and scene segments to find better camera configurations.

2.3 Needs for a Camera Planning Approach for Viewing Robot Manipulation Tasks

All the previous approaches were developed for domains with predefined story scripts and in which subjects are characters and objects as normally found in movies or video games. They assume a finite number of primitive filming operators which convey cinematographic principles proper to the domain of actors. To illustrate, "Extreme Close Up" is a primitive consisting of using a camera zoom on the face of a person being filmed. There are a small number of such primitives [15, 5, 26].

Filming an articulated robot arm introduces additional challenges not explicitly envisioned in these approaches. First of all, the cinematographic principles for filming actors are different from those of filming a robot task given that the points of interests and types of actions are different. A specific ontology for filming robot tasks is necessary and it must include types of meaningful elementary movements for robot arms, relations of those

elements to tasks and to filming idioms. We initiate this quest for such ontological definition of camera planning in the domain of articulated robot arms by proposing one for the SSRMS.

Another particularity of filming robot tasks is that the trajectory, motion and actions of the robot being filmed are not known in advance. In other words, tasks and underlying trajectories in our domain are not scripted in advance. As we mentioned, given a task, we generate a trajectory of the robot accomplishing the task by using a path planner. This trajectory is automatically decomposed into meaningful segments (shots) according to some heuristics.

Once we have an ontology for the right filming primitives and an on-the-fly decomposition of a robot trajectory into shots, in principle the previous approaches (like DCCL) could be easily adapted to film the robot. However, here we opt rather for using LTL [3] as the language for specifying shot composition rules. The advantage of LTL over the languages used in previous approaches, such as DCCL, is that LTL has well defined semantics. With LTL, we can express arbitrary temporal conditions about the order in which objects should be filmed, which objects should remain in the background until some condition become true as well as more complex constraints. Another advantage of adopting LTL is that the underlying TLPlan planning algorithm is far more efficient than the planning systems underlying previous cinematographic approaches. For instance, DCCL was handled using a variant of UCPOP [15]. TLPlan was shown to outperform such planning systems by several orders of magnitude [3].

2.4 The Automatic Task Demonstration Generator

The automatic task demonstration generator (ATDG) takes as input start and goal configurations of the SSRMS. Using those two configurations, the ATDG generates a movie demonstration of the required manipulations in order to bring the SSRMS from its start configuration to its goal configuration. The top figure in Figure 2.4 illustrates the internal architecture of the ATDG. The bottom one shows the different steps the data go through in order to transform the two given configurations into a complete movie demonstration.

First, the ATDG uses a path planning algorithm (i.e., the Flexible Anytime Dynamic Probabilistic Roadmap (FADPRM) planner [7]), which takes the two given configurations

2.4. THE AUTOMATIC TASK DEMONSTRATION GENERATOR

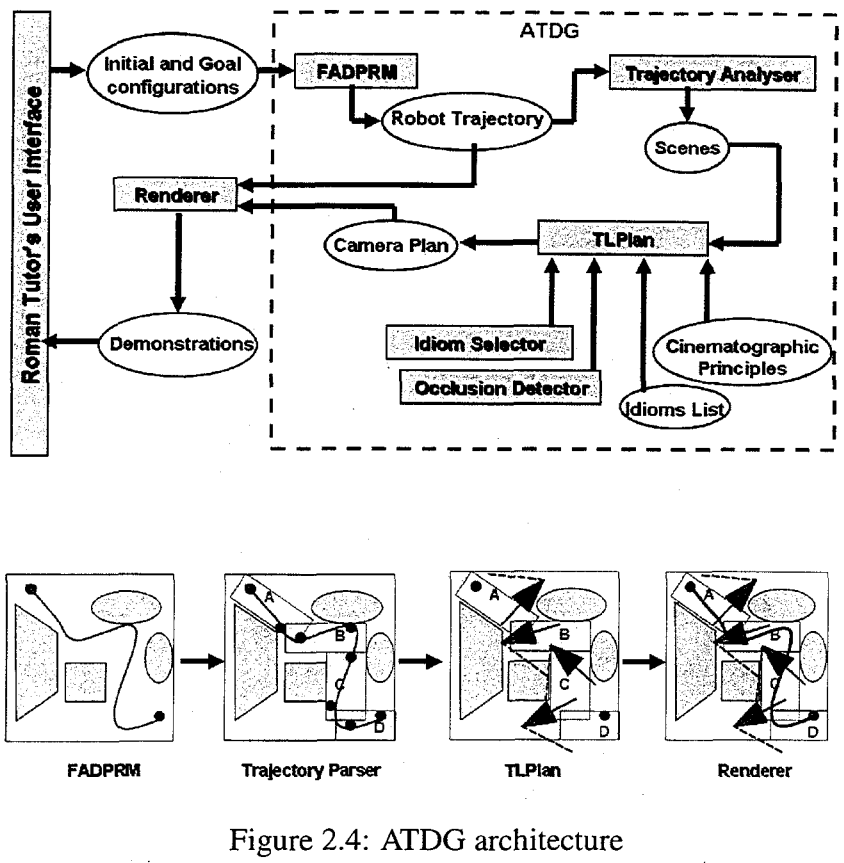


Figure 2.4: ATDG architecture

and generates a collision free path between them. This path is then given to the trajectory parser which separates it into categorized segments. This transforms the continuous trajectory into a succession of scenes, where each scene can be filmed by a specific group of idioms. The parser looks for uniformity in the movements of the SSRMS. This process is described in greater detail in the next section.

Once the path is parsed, the camera planner uses TLPlan to find the shots in order to best convey each scene, while making sure that the whole is pleasing and comprehensive. To do this, TLPlan uses an idiom database to help it find the best way to film each scene. In addition to the idiom database, TLPlan applies a set of LTL shot composition rules to generate a movie that is visually appealing and coherent. TLPlan further applies an occlusion detector to make sure the SSRMS is visible all the time. Once TLPlan is done, we are left with a list of shots that is used by the rendering system to create the animation. The renderer uses both the shots given by TLPlan and the SSRMS trajectory in order to

position the cameras in relation with the SSRMS, generating the final task demonstration.

2.5 Camera Planning Approach

2.5.1 Segmenting the Robot Trajectory into Scenes

In order to divide the animation into short sequences as a human would, we must study the robotic arm's trajectory. Indeed, it is the only factor that can be used to split the movie since everything else in the scene is stationary. We implemented two different complementary ways to proceed with this partitioning task.

Dividing according to elementary movements

The idea here is to define some elementary motions of the robotic arm that the segmentation algorithm is able to recognize in the trajectory. Presently we use the following elementary motions, based on an intuitive analysis of the different movements of an arm one wants to recognize.

- *Vertical elevation:* The arm moves up, due to the elbow joint or the pitch shoulder joint. This movement occurs when we need to elevate a load to avoid an obstacle, for example.
- *Lateral rotating motion:* The movement of the yaw shoulder joint dominates and causes the arm to move laterally, possibly to transfer to another local region of the station when no obstacles lay between.
- *Static rotation:* This movement corresponds to a rotation of the shoulder-elbow segment, controlled by the roll shoulder joint.
- *Wrist movement:* As the name indicates, here only the wrists joints are moving significantly.
- *Rail translation:* The arm translates from one point to another along the rectilinear rail on the ISS. This movement is used when the arm needs to change to a different work area.

The algorithm used to detect these movements consists of calculating the elementary variations along each of the robotic arm's 7 degrees of freedom frame by frame and cutting

the trajectory when the nature of the movement changes. We can then film each segment by selecting and then applying an idiom well suited to the elementary motion.

Dividing according to objects and regions of interest

We also segment the trajectory according to its variation with respect to given obstacles or visual cues. When the variation reaches a fixed threshold, we have a transition to a new segment. The lower level implementation invokes the graphics Proximity Query Package (PQP) [43] to calculate the distances from the arm to given obstacles or cues. In this way we can also define segmentations of the arm depending on whether the entire arm or selected parts of the arm move from one zone of interest to another.

2.5.2 Specifying Shots and Idioms

A shot is specified by five components: shot type, camera placement mode, camera zooming mode, side of the line of interest and length.

- *Shot Types:* Five shot types are currently defined in the ATDG System: Static, GoBy, Pan, Track and POV. A Static shot is done from a static camera when the robot is in a constant position or moving slowly. A GoBy shot has the camera in a static position showing the robot in movement. For a Pan shot, the camera is in a static position but doing incremental rotations following the movement of the robot. A Track shot has the camera following the robot and keeping a constant position relative to it. Finally, the POV shot has the camera placed directly on the SSRMS, moving with the robot.
- *Camera Placements:* For each shot type, other than POV, the camera can be placed in five different ways according to some given line of interest: External, Parallel, Internal, Apex and External II (Figure 2.5). In the current implementation, the line of interest is the trajectory along which the center of gravity of the robot is moving; this is sufficient for filming many typical manoeuvres. POV is treated separately. Since the camera is directly on the SSRMS, the previously described camera placements are inapplicable. This attribute is instead used to specify where on the robot arm the camera is placed (such as on the end effector, on some joint, in the middle of a segment, etc.).

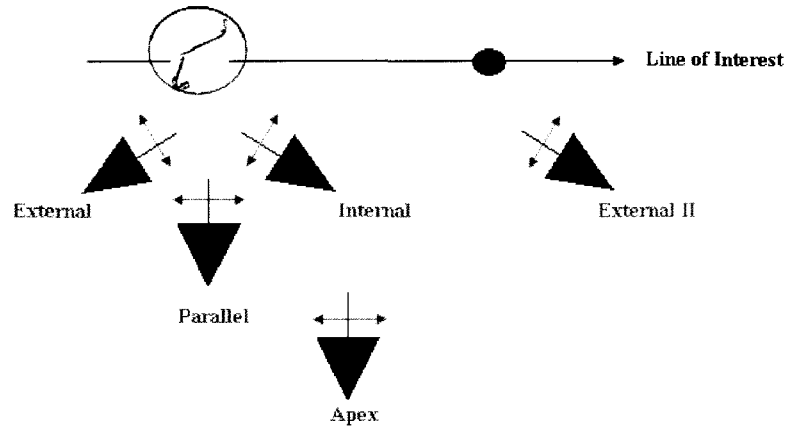


Figure 2.5: Camera placements

- *Zoom modes:* For each shot type and camera placement, the zoom of the camera can be in five different modes: Extreme Close Up, Close Up, Medium View, Full View and Long View.
- *Side:* Each shot type other than POV can be taken from either side of the line of interest or from above. In the case of POV, this attribute is used to tell whether the camera is forwards or backwards of the SSRMS. Shots from above help achieve smooth transitions between a shot from one side to a shot from the other side of the line of interest or can be used when the robot is flanked on both sides by obstacles.
- *Length:* The length is a fraction of the scene occupied by the shot. The total length for all shots in a scene must be 1. For instance, if the first shot has a length of 0,25, the second, a length of 0,5 and the last a length of 0,25, while the scene lasts 2 seconds, then the first shot will end after half a second, the second will then start and end at 1,5 seconds, and so on.

More shot types, camera placements and zoom modes will be added in the future to specify a greater variety of shots. We are now in position to explain how idioms are formalized. An idiom is specified by describing the sequence of shots composing it. Figure 2.6 shows three examples.

Each idiom consists of an identifier, the scene to which it is applicable, the number of shots and then the shots. Thus the first idiom is applicable to a translation of the SSRMS along its mobile base and it contains three shots. The idiom states that a translation can

```
(idiom1 translation 3
        (go-by external medium-view 0,25 left)
        (go-by parallel full-view 0,5 left)
        (pan internal full-view 0,25 left))

(idiom2 translation 3
       (static external medium-view 0,25 right)
       (static apex medium-view 0,67 right))
       (static parallel closeup-view 0,67 right))

(idiom3 effector 3
       (pan parallel full-view 0,33 left)
       (static parallel closeup-view 0,67 left))
```

Figure 2.6: Examples of idioms

be filmed by first placing the camera external to the robot using a medium view zoom, following the SSRMS for a quarter of the whole movement, then changing to a full view zoom while still following the robot on a parallel course for half the scene and then stopping and using a pan shot with the camera rotating to follow the robot for the rest of the way, still at a full view zoom. This is just one of the many ways such a scene could be filmed. There are other idioms specifying the alternatives.

For instance, idiom2 illustrates another way of filming a translation of the SSRMS along its mobile base to anchor a new module on the ISS. In this case, we start with a static shot showing the robot approaching the destination module of the ISS using an External camera placement to have a good visualization of the destination module; the zoom is medium to have a good view of the approach. The second shot is also static but with an Apex placement to show the robot approaching the destination module. The last shot switches to a Close Up Static view to give a good appreciation of the anchoring operation.

The third idiom describes a sequence of shots for filming the SSRMS's end effector fixing a new component on the ISS. The first shot is a pan shot following the rotation of the robot to the anchor point. The last shot is a static shot focusing on the joint at the extremity of the robot while fixing the new component on the desired target.

Thus for each SSRMS movement type (or scene), we have several idioms (from six to ten in the current implementation) and each idiom is defined by taking into account the complexity of the movement, the geometry of the ISS, the visual cues on the ISS and subjective expectations of the viewer. For example, if the SSRMS is moving along its mobile base, it is important that the camera shows not only the entire arm but also some visual cues on the ISS in order to provide situational awareness of the relocation of the

mobile base. Consequently, the idioms for this manipulation involve shots with a Full or Long View zoom. In contrast, manipulations involving the end effector require a high precision, so an Extreme Close Up zoom will be involved.

2.5.3 Specifying Shot Composition Rules

The description of idioms is based on considerations that are local to a scene. Shots within an idiom are in principle sequenced coherently with respect to the scene, but this does not guarantee that the sequencing of scenes is coherent too. The role of shot composition rules is to ensure that selected idioms generate a continuous animation with smooth transitions between scenes and with some global constraints that must be respected across the entire film. Such global shot composition rules are expressed in LTL.

As mentioned before, LTL is the language used by TLPlan to specify temporally extended goals [3]. LTL formulas are interpreted over state sequences [3]. In our case, a state conveys some properties about the current shot, hence LTL formulas are indirectly interpreted over sequences of shots. In the LTL language, one uses the temporal modalities `next`, `always`, `eventual`, `until`, combined with the standard first order connectives, to express temporal statements. For instance, `(next f)`, where `f` is a formula means that `f` is true in the next state; `(always f)` means that `f` holds on the entire state sequence; similarly for the other modalities, they have the intuitive semantics suggested by their names. Given a planning domain, an initial state and an LTL temporally extended goal, TLPlan computes a plan, as a sequence of actions, such that the underlying sequence of states satisfies the goal.

```
;; line of interest
(always
  (and
    (forall (?t0 ?p0 ?z0 ?l0)
      (current-shot ?t0 ?p0 ?z0 ?l0 right)
         (next (not (exists (?t1 ?p1 ?z1 ?l1)
                (current-shot ?t1 ?p1 ?z1 ?l1 left)))))
    (forall (?t0 ?p0 ?z0 ?l0)
       (current-shot ?t0 ?p0 ?z0 ?l0 left)
         (next (not (exists (?t1 ?p1 ?z1 ?l1)
              (current-shot ?t1 ?p1 ?z1 ?l1 right)))))))
```

Figure 2.7: A shot composition rule

Figure 2.7 illustrates an LTL shot composition rule forbidding the selection of two dif-

ferent sides for two successive shots. This implements a cinematography rule that prevents crossing the line of interest because it could induce a misunderstanding of the manipulation performed. This idiom will require TLPlan to insert an intermediate shot in order to satisfy the requirement.

2.5.4 Planning the Cameras

When searching for a sequence of shots to satisfy a goal, shots are evaluated using the *Occlusion Detector* function according to their degree of occlusion. Specifically this function measures the degree of visibility of the robot within the shot. This is done by examining each image in the shot and evaluating the number of joints present in the image and the zoom made on the robot. The quality measure on each image is heuristically defined by :

$$\alpha_{shot} = (Nbr_{JV}/Nbr_{JT} + S_R/S_T)/2,$$

with :
- Nbr_{JV} : Number of joints visible in the image
- Nbr_{JT} : Total Number of joints in the robot
- S_R : Surface covered by the robot on the image
- S_T : Total Surface of the image

TLPlan calls the *Occlusion Detector* not only to compute the quality measure on each shot but also to compute the quality measure on every idiom. The quality measure of an idiom α_{idiom} is the average of the quality measures on the shots composing it.

The planning domain is specified using two different kinds of planning operators: Idiom-Selection and Shot-Specification. The first type of operators conveys idiom-level constraints. The second conveys shot-level constraints. More specifically, Idiom-Selection operators select idioms for shots, and Shot-Specification operators select attributes for each shot composing an idiom. Since the attributes are already specified by the idioms, the role of this operator is essential to ensure that the shot follows the defined cinematographic rules and to allow the *Occlusion Detector* to verify its quality.

Figure 2.9 illustrates an idiom-selection operator. The operator checks whether the scene already has an idiom associated to it (i.e., *(not-planned ?scene)*). If no idiom has

```
;; Shot selection operator
(add-adl-op
 :name '(apply-shot ?scene ?shot-type ?shot-place
                    ?shot-zoom ?shot-length ?shot-side)
 :pre (adl-pre :var-gens '((?scene ?idiom)
                           (planned ?scene ?idiom)
                           (?nextShot)
                           (next-shot ?scene ?nextShot)
                           (?shot-type)
                           (next-shot-type ?idiom
                                           ?nextShot)
                           (?shot-place)
                           (next-shot-place ?idiom
                                            ?nextShot)
                           (?shot-zoom)
                           (next-shot-zoom ?idiom
                                           ?nextShot)
                           (?shot-length)
                           (next-shot-length ?idiom
                                             ?nextShot)
                           (?shot-side)
                           (next-shot-side ?idiom
                                           ?nextShot)
                           (?nbrShot) (nb-shot ?scene
                                               ?nbrShot)))
:add (adl-add (adl-cond :form '(= ?nextShot
                                  (- ?nbrShot 1))
                        :lit '(done-plan ?scene))
              (adl-cond :form '(not(= ?nextShot
                                      (- ?nbrShot 1)))
                        :lit '(next-shot ?scene
                                         (+ ?nextShot 1)))
              (adl-cond :lit '(last-shot ?shot-type
                                ?shot-place ?shot-zoom
                                ?shot-length
                                ?shot-side)))
:del (adl-del (adl-cond :var-gens '((?nextShot)
                                    (next-shot ?scene
                                               ?nextShot))
                        :lit '(next-shot ?scene ?
                                         nextShot))
              (adl-cond :var-gens '((?t ?p ?z ?l ?s)
                                    (last-shot ?t ?p
                                               ?z ?l ?s))
                        :lit '(last-shot ?t ?p ?z ?l
                                         ?s))))
```

Figure 2.8: Shot specification operator

been planned for the scene, the operator will update the current state by adding an idiom for the scene, updating the number of shots to be planned for this scene (as specified by the chosen idiom) and will update the next shot to be planned to be the first shot of the idiom. Figure 2.8 illustrates a shot-specification operator.

During search, the "current" world state in this domain consists of:

1. The "current" scene from a given scene list.
2. The "current" idiom being tested as a candidate for the current scene.
3. The "current" shot in the idiom currently being tested.

```
;; Idiom selection operator
 (add-adl-op
 :name '(apply-idiom ?scene ?idiom)
 :pre (adl-pre
        :var-gens '((?scene) (not-planned ?scene)
                    (?type) (sc-type ?scene ?type)
                    (?idiom) (idioms))
        :form '(eq? ?type (id-type ?idiom)))
 :add (adl-add (adl-cond
           :var-gens '((?nbrShot)
                       (gen-nb-shot ?idiom))
           :lit '(nb-shot ?scene ?nbrShot))
         (adl-cond :lit '(planned ?scene ?idiom))
         (adl-cond :lit '(next-shot ?scene 0)))
:del (adl-del (adl-cond :lit '(not-planned ?scene))))
```

Figure 2.9: Idiom selection operator

Intuitively, the search process underlying TLPlan explores the world state space as follows. On each iteration, TLPlan takes the current scene from the list of scenes and checks whether an idiom has already been selected to be tested as best candidate for it. If not, it calls the Idiom-Selection operator and selects an idiom from the list of idioms associated to the corresponding category of scene. When a current idiom is selected in the current state, TLPlan takes the list of shots composing it and finds the next unplanned shot (if all the shots have been planned, then the scene is completed and TLPlan can now move to the next scene). Then it calls the Shot-Specification operators on the current shot which calls the *Occlusion Detector*. If the shot is accepted, then it is added to the list of planned shots.

2.5.5 Discussion

We explicitly specify the sequence of shots composing an idiom. Given that an LTL formula describes a sequence of states (that is, the sequence of states satisfying it), we could have adopted LTL formulas not just for specifying shot composition rules but also for describing idioms. For instance, filming a translation scene using idiom1 or idiom2 in Figure 2.6 could be specified as described in Figure 2.10.

```
(define control-idiom-translation
 (always
    (exists (?scene ?type) (and (current-sc ?scene) (sc-type ?scene ?type))
      (implies (?type translation)
        (or
          (and (not bad-idiom Id0 ?scene)
               (current-idiom ?scene Id0)
               (next (current-shot go-by external medium-view 0,25 left))
               (next (next (current-shot go-by parallel full-view 0,5 left)))
               (next (next (next (and (current-shot pan internal full-view 0,25 left)
                                      (planned ?scene))))))
          (and (not bad-idiom Id1 ?scene)
               (current-idiom ?scene Id1)
               (next (current-shot pan external medium-view 0,33 right))
               (next (next (and (current-shot static parallel closeup-view 0,67 right)
                                (planned ?scene)))))))))))
```

Figure 2.10: Idiom-description LTL formula used to film a translation of the SSRMS

This approach would allow a richer idiom specification language since LTL can express more general sequences than listing a series of shots. However, with this approach the size of the search space becomes larger and sophisticated search control would have to be involved. Indeed, for each scene, the search process would have to consider all possible shots sequences satisfying the idiom formula. In contrast, with the current approach, search is only limited to the shot sequences in the idiom specifications. More intelligent ways of using LTL formulas to specify idioms remain a topic for future research.

We acquired knowledge on the SSRMS through discussions with experts (including instructors of the SSRMS) and sitting in actual training courses of the SSRMS. The current idioms take into account only visibility requirements, but there are other constraints that will have to be integrated to complete the tool, including the various modes of operating the arm, which involve, among many things, switching between different frames of references during a manipulation.

2.6 Experiments

We used a publicly available Scheme version of TLPlan within the ATDG. The Robot Simulator is in C++. The communication between TLPlan and other components in the ATDG is done using sockets and through reading and writing in text files. For example, TLPlan communicates by sockets with the *Occlusion Detector* to compute at each iteration the quality measure on shots and idioms. The camera plan is passed to the renderer in text format.

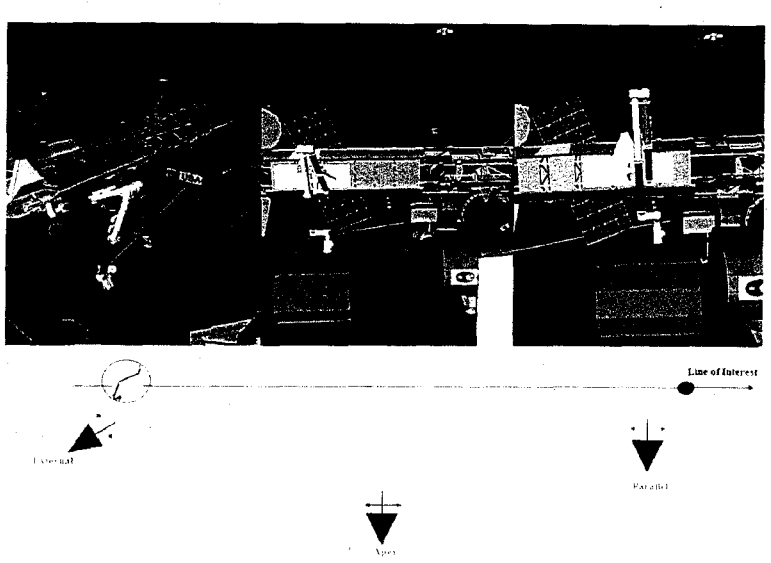


Figure 2.11: Idiom to film the SSRMS anchoring a new module on the ISS

We implemented two different variations of the camera planner. The first version (V1) delays the check of the quality on shots until a complete sequence of shots composing the whole film is found; the other version (V2) makes the check on the fly on each idiom as has been understood so far. In each case, the metrics for the quality of an animation are the absence of occlusion. Checking occlusions takes time, hence the motivation to verify whether there is any gain by delaying it. The experiments also include a comparison with a simplified implementation of a constraint-based approach as in ConstraintCam (CC) [4].

Figure 2.11 shows snapshots generated by ATDG illustrating the anchoring operation of idiom2 in Figure 2.6. The ISS workspace is specified by almost 85 000 triangles. This is moderately complex by computer graphics standards. The experiments were performed

on a Pentium Dual Core, 2,8 GHZ, with 2G of RAM.

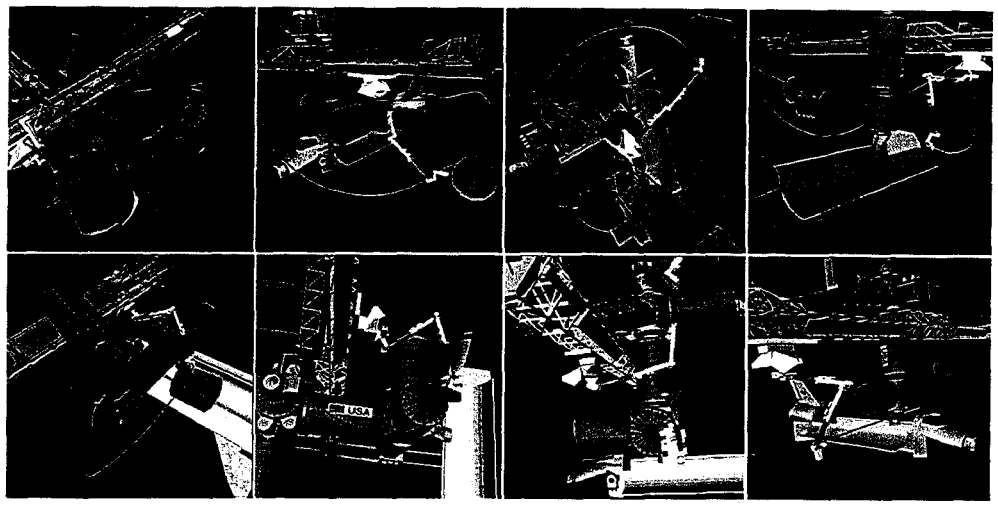


Figure 2.12: Scenarios

The trajectories for the eight scenarios on which we made our experiments are given in Figure 2.12. Figure 2.13 shows the performance data on the scenarios. The first three colomns in Figure 2.13 indicate, respectively: the scenario, the number of shots composing the film and the duration of the film in seconds. The next three columns express the quality of the movie for each of the methods V1, V2 and CC, in terms of the proportion of shots without any occlusion of the camera and with all selected elements of the arm visible. This occlusion-based quality measure could be refined by counting the number of images in occlusion or by taking into account the proportion of the shot that is occluded. The last three columns give the planning time for each of the methods.

Scene Properties			Visual Quality			Planning Time (s)		
Sc N	Nbr Shots	Length (s)	V1	V2	CC	V1	V2	CC
1	2	5.34	0.61	0.64	0.4	0.65	0.87	0.54
2	3	13.42	0.54	0.57	0.37	0.66	0.73	0.45
3	4	7.45	0.71	0.76	0.5	1.04	1.52	1.12
4	8	10.41	0.56	0.62	0.23	1.35	2.12	1.21
5	6	20.21	0.7	0.76	0.43	1.53	1.87	1.3
6	5	8.73	0.81	0.87	0.62	1.42	1.65	0.98
7	11	24.1	0.68	0.73	0.24	2.86	3.76	1.57
8	9	17.2	0.8	0.82	0.45	2.03	2.54	1.24

Figure 2.13: Performance data

As the experiments show, the quality of the demonstrations generated by ATDG are

very good in terms of the number of shots that are not occluded and this is an important property that we would like our application to have. We also noticed that the film is very smooth considering it is generated automatically. As it turns out, both versions of ATDG generate movies of similar qualities, but they differ in the planning time. Delaying occlusion checking turns out to be worthwhile for this experiment.

The results also show that the quality of the path filmed by ATDG was always better than CC. This is due to the fact that TLPlan works at the level of the idiom, a level higher than that of a frame (the level to which ConstraintCam applies) and this always ensures a higher level of quality. We also believe that with a C++ implementation of TLPlan, our approach would become more efficient. One of the key aspects of our solution is the use of LTL to specify shot composition rules and generally they are more easily understood than frame-level constraints. It is important to remember, however, that this is a comparison with a simplified implementation of the original ConstraintCam.

2.7 Conclusion and Future Work

In this paper, we presented a temporal-logic camera planning approach that brings improvements to automatic 3D animation generation techniques along two main dimensions. First, it uses temporal logics to express cinematographic principles and filming preferences, a language more expressive and with simpler semantics than previous camera planning languages. Second, it implements the TLPlan algorithm, a planner more powerful than previous camera planners in the literature, because with TLPlan, LTL composition rules enable search pruning.

The new camera planner is implemented within an application of automated planning for the generation of 3D task demonstrations for an articulated robot arm. Our application currently concerns the SSRMS deployed on the international space station but the results are transferable to other telemanipulated robots and to other domains.

So far we have obtained promising results using very simple metrics for the quality of movies. Adding visual cues and regional constraints is quite straightforward and will be done in the near future. Before the tool becomes useable in practice, additional metrics characterizing the different manoeuvres in terms of task and space awareness will have to be brought in. The definition of these metrics will involve human factor experts and

instructors.

Besides the intended future use of our system to support ground operators, future inquiry will also concern the integration of the system into a training simulator to provide feedback to students by showing them how to accomplish the task. This opens several interesting research opportunities, including making the generated animation interactive rather than a continuous video as is currently the case.

Finally, using TLPlan also opens up interesting opportunities for developing efficient search control knowledge for this particular application domain and for learning such knowledge. As mentioned above, it would be also interesting to extend the use of LTL formulas to the specification of idioms.

2.8 Acknowledgement

The work presented herein was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Chapitre 3

Modèle pédagogique dans un simulateur intelligent pour les opérations robotisées

Résumé

L'apprentissage par simulation en environnements virtuels est seulement possible s'il existe un espace problème bien explicite associé aux différentes tâches à réaliser par l'apprenant [1, 63]. Un tel espace permet de bien suivre l'apprenant au cours de son apprentissage et de le guider si nécessaire. Au fait, le suivi de la connaissance et de son acquisition par l'apprenant pendant l'apprentissage ne sont possibles qu'en satisfaisant parfaitement à cette condition [18]. Cependant, dans des environnements très complexes, il n'est pas toujours possible de construire une structure de tâche bien explicite. Par exemple, les environnements traitant de connaissances spatiales ont une multitude de possibilités pour résoudre une tâche donnée. Ceci est le cas dans Roman Tutor où la tâche de manipulation du SSRMS pour l'emmener d'un endroit à un autre peut s'effectuer selon une infinité de trajectoires.

Dans l'article que nous présentons dans ce chapitre, nous montrons que par l'utilisation du planificateur de trajectoires FADPRM (voir chapitre 1) comme expert du domaine, il n'est plus nécessaire de créer un graphe de tâches complexe et bien explicite. Nous présentons ensuite l'architecture du modèle pédagogique

instauré dans Roman Tutor pour supporter l'apprentissage d'un apprenant sur le simulateur. Nous montrons comment les différentes composantes intégrées, principalement FADPRM et ATDG, sont mises à contribution pour assurer un suivi et un encadrement efficaces de l'apprenant.

Pour illustrer, pendant que l'apprenant manipule le robot pour l'exécution de différents tâches de manipulations, Roman Tutor fait appel à FADPRM en arrière plan pour surveiller d'une manière continue l'évolution de l'apprenant dans son apprentissage : Est-t-il en train de suivre les zones désirables ou non ? S'approche-t-il de la configuration finale ou pas ? Est-ce qu'il existe une famille de solutions meilleures que celle qu'il est en train de réaliser ? La caractéristique dynamique aide FADPRM à s'ajuster automatiquement aux changements continuels dans l'environnement (déplacements du robot, déplacements des zones). La caractéristique anytime va lui permettre de trouver des diagnostics en temps réel pour un encadrement plus efficace. De plus, Roman Tutor fait appel à ATDG pour guider l'apprenant s'il se trouve bloqué ou s'il commence à effectuer des erreurs pour lui présenter presque instantanément des vidéos correctives ou illustrant les trajectoires solutions qui permettent de compléter la tâche.

Commentaires

L'article présenté dans ce chapitre a été soumis au journal "IEEE Transactions on Learning Technology". Une version plus courte de ce travail a été publiée dans les actes de "International Conference on Intelligent Tutoring Systems" [56]. Ces travaux ont été réalisés, validés et rédigés sous la supervision du Professeur Roger Nkambou (Université du Québec à Montréal), du Professeur Froduald Kabanza (Université de Sherbrooke) et avec les conseils du Professeur Leo Hartman (Agence spatiale canadienne).

An Intelligent Simulator for Tele-robotics Training

Khaled Belghith, Froduald Kabanza
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
khaled.belghith@usherbrooke.ca, kabanza@usherbrooke.ca
Roger Nkambou
Université du Québec à Montréal
Montréal, Québec H3C 3P8, Canada
nkambou.roger@uqam.ca
Leo Hartman
Canadian Space Agency,
John H. Chapman Space Centre, 6767 Route de l'Aéroport
Saint-Hubert, Québec J3Y 8Y9
leo.hartman@asc-csa.gc.ca

Abstract

Roman Tutor is a tutoring system that uses sophisticated domain knowledge to monitor the progress of students and advise them while they are learning how to operate a space tele-robotic system. It is intended to help train operators of the Space Station Remote Manipulator System (SSRMS) including astronauts, operators involved in ground-based control of SSRMS and technical support staff. Currently there is only a single training facility for SSRMS operations and it is heavily scheduled. The training staff time is in heavy demand for teaching students, planning training tasks, developing teaching material and new teaching tools. For example, all SSRMS simulation exercises are developed by hand and this process requires a lot of staff time. Once in orbit ISS astronauts currently have only simple web-based material for skill development and maintenance. For long duration space flight astronauts will require sophisticated simulation tools to maintain skills. Roman Tutor addresses these challenges by providing a sophisticated portable training tool. It incorporates a model of the system operations curriculum, a kinematic simulation of the robotics equipment and the ISS, a high performance path planner and an automatic task demonstration generator. For

each element of the curriculum that the student is supposed to master, Roman Tutor generates example tasks for the student to accomplish within the simulation environment and then monitors its progression to provide relevant feedback when needed. Although motivated by the SSRMS application, Roman Tutor remains applicable to any tele-robotics system application.

F

3.1 Introduction

Roman Tutor (RObot MANipulation Tutor) is a simulation-based tutoring system to support astronauts in learning how to operate the SSRMS, an articulated robot arm mounted on the International Space Station (ISS). Figure 3.1 includes an image of the SSRMS on the ISS. Astronauts operate the SSRMS from a robotic workstation located inside one of the ISS compartments. Figure 3.1 also shows the workstation which has an interface with three monitors, each of which can be connected to any of the 14 cameras placed at strategic locations on the exterior of the ISS. Roman Tutor's user interface in Figure 3.2 includes the most important features of the robotic workstation.

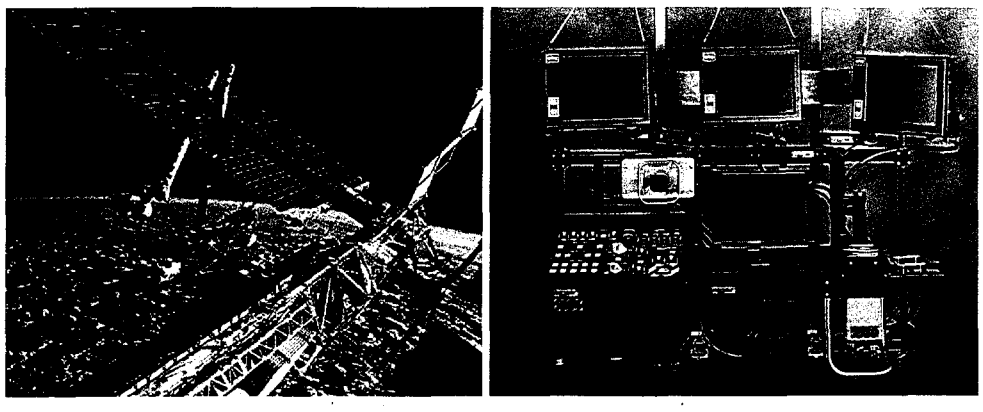


Figure 3.1: SSRMS on the ISS (left) and the robotic workstation (right)

The SSRMS is a key component of the ISS and is used in the assembly, maintenance and repair of the station, and also for moving payloads from visiting shuttles. Operators manipulating the SSRMS on orbit receive support from ground operations. Part of this

support consists of visualizing and validating maneuvers before they are actually carried out on the ISS. Operators have in principle rehearsed the maneuvers many times on the ground prior to the mission, but unexpected changes are frequent during the mission. In such cases, ground operators may have to generate 3D animations for the new maneuvers and upload them to the operator on the station. So far, the generation of these 3D animations are done manually by computer graphic programmers and thus are very time consuming.

SSRMS can be involved in various tasks on the ISS, including moving a load from one place of the station to another, inspecting the ISS structure (using a camera on the arm's end effector) and making repairs. These tasks must be carried out very carefully to avoid collisions with the ISS structure and to maintain safety-operating constraints on the SSRMS (such as avoiding self-collisions and singularities). At different phases of a given manipulation, the astronaut must choose a setting of cameras that provides him with the best visibility while maintaining awareness of his progress on the task. Thus astronauts are trained not only to operate the arm itself, but also to recognize visual cues on the station that are crucial in mentally reconstructing the actual working environment from the partial and restricted views provided by the three monitors, and to select cameras depending on the task and other parameters.

One challenge in developing a good training simulator is, of course, to build it so that one can reason about it. This is even more important when the simulator is built for training purposes [23]. Until now, simulation-based tutoring is possible only if there is an explicit model or representation of the problem space associated with training tasks. The explicit representation is required in order to track student actions, to identify if these actions are still on a path to a solution and to generate relevant tutoring feedback [1, 63]. Knowledge and model tracing are only possible in these conditions [18]. It is not always possible to develop an explicit comprehensive task structure in complex domains, especially in domains where spatial knowledge is used, as there are many possible ways to solve a given problem. The robot manipulation that Roman Tutor focuses on is an example of such a domain. For each robot manipulation task, there is a combinatorial explosion of possible solutions for moving SSRMS from one place to another in the ISS environment. Such domains has been identified as "ill-structured" [60, 24].

Conventional tutoring approaches such as model-tracing [37] or constraint-based modeling [52] are very limited when applied on "ill-structured" domains. A model-tracing

approach consists of comparing a predefined task model with a student's solution. In the context of robot manipulations, because of the infinity of solutions we have associated with each task, designing a task model by hand becomes practically infeasible. Applying a constraint-based modeling approach in the context of robot manipulations will also face the same kind of limitations. Here, identifying the constraints associated with robot manipulation tasks can be difficult and very time consuming. Since a huge number of constraints is required to achieve an adequate level of tutoring assistance [24], the approach becomes impractical.

To overcome these limitations, we propose a solution to this issue by integrating a sophisticated path planner FADPRM [7] as a domain expert system to support spatial reasoning within the simulator and make model tracing tutoring possible without any explicit task structure.

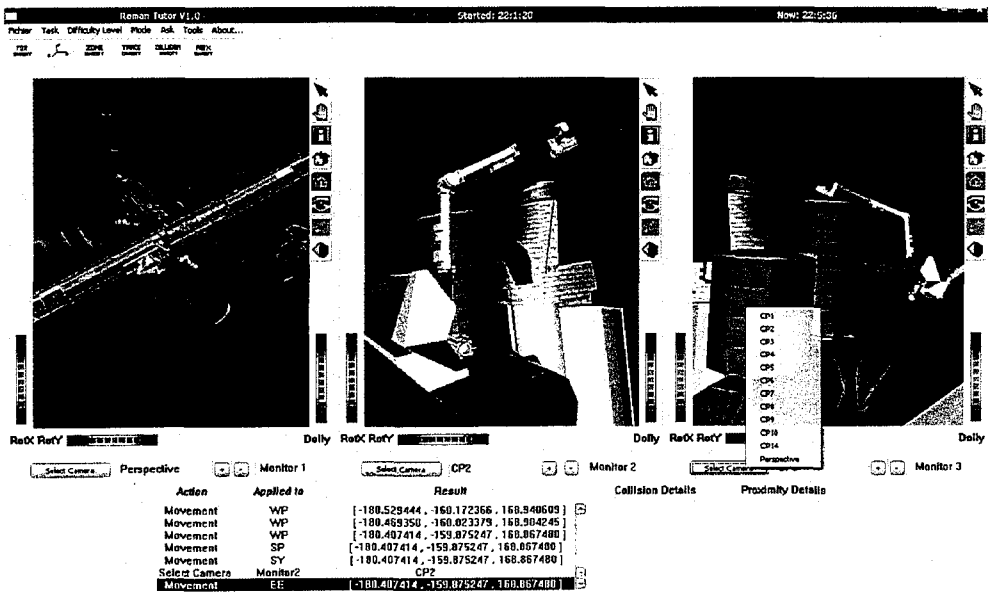


Figure 3.2: Roman tutor interface

FADPRM (Flexible Anytime Dynamic PRM path planner) is an extension to the PRM planning framework [36] to handle regions to which we assign preferences within complex workspaces. By being flexible in this way, FADPRM not only computes collision free paths but is also capable of taking into account the placement of cameras on the ISS, the lighting conditions and other safety constraints on operating the SSRMS. This allows the generation

of collision-free trajectories in which the robot stays within regions visible through cameras and in which the manipulation is, therefore, safer and easier. FADPRM also implements a dynamic strategy to adapt efficiently to dynamic changes in the environment and re-plan on the fly by exploiting results from previous planning phases. FADPRM also implements an anytime strategy to provide a correct but likely suboptimal solution very quickly and then incrementally improve the quality of this solution if more planning time is allowed.

Roman Tutor uses the different capabilities implemented within the FADPRM path planner to provide useful feedback to a student operating the SSRMS simulation. To illustrate, when a student is learning to move a payload with the robot, Roman Tutor invokes the FADPRM path-planner periodically to check whether there is a path from the current configuration to the target and provides feedback accordingly. By using FADPRM as a robot manipulation domain expert, we follow an "expert system approach" to support the tutoring process within Roman Tutor. This approach has proven successful and has been used within different well-known intelligent tutoring systems such as SOPHIE I [12] and GUIDON [16]. But in our case, we are applying it in the context of robot manipulations, an "ill-structured" domain.

We also developed within Roman Tutor an automatic task demonstration generator ATDG [34], which generates 3D animations that demonstrate how to perform a given task with the SSRMS. The ATDG is integrated with the FADPRM path planner and can contribute to ground support of SSRMS operations by generating useful task demonstrations on the fly that help the student carry out his tasks. ATDG includes a component based on TLPlan [3] for camera planning and uses Linear Temporal Logic (LTL) as the language for specifying cinematographic principles and filming preferences. A robot trajectory is first generated by FADPRM and TLPlan is then called to find the best sequence of camera shots following the robot on its path.

In the next section, we start by presenting FADPRM and ATDG in detail. We then describe Roman Tutor's internal architecture and outline its basic functionalities. After enumerating the tasks on which a student is trained within Roman Tutor, we describe the approaches followed to provide the tutoring assistance. In a following section, we show how the use of FADPRM as a domain expert within the simulator helped in providing very relevant tutoring feedback to the student. We finally conclude with a discussion on related work.

3.2 FADPRM Path Planner

In its traditional form, the path planning problem is to plan a path for a moving body (typically a robot) from a given start configuration to a given goal configuration in a workspace containing a set of obstacles. The basic constraint on solution paths is to avoid collision with obstacles, which we call hereby a hard constraint. There exist numerous approaches for path planning under this constraint [36, 14, 47, 57, 44]. In order to take into account the visibility constraints we have in the SSRMS environment, we developed a new class of flexible path planners FADPRM [7] able to express and take into account preferences in the navigation of the robot within very complex environments. In addition to the obstacles the robot must avoid, our approach takes account of desirable and undesirable (or dangerous) zones. This will make it possible to take into account the disposition of cameras on the station. Thus, our planner will try to keep the robot in zones offering the best possible visibility of progress on the task while trying to avoid zones with reduced visibility.

The robot free workspace is segmented into zones with each zone having an associated degree of desirability (dd), that is, a real number in the interval [0 1], depending on the task, visual cue positions, camera positions and lighting conditions. The closer the dd is to 1, the more the zone is desirable. Safe corridors are zones with dd near to 1, whereas unsafe corridors are those with dd in the neighborhood of 0. A zone covering the field of view of a camera will be assigned a high dd and will have a cone shape ; whereas a zone with very limited lighting conditions will be considered as an undesirable zone with a dd near 0 and will take an arbitrary polygonal shape. Figure 3.3 illustrates a trajectory of the SSRMS going through three cameras fields of view (three cones) and avoiding an undesirable zone (rectangular zone in pale red).

For efficient path planning, we pre-process the robot workspace into a roadmap of collision-free robot motions in regions with highest desirability degree. More precisely, the roadmap is a graph such that every node n in the graph is labeled with its corresponding robot configuration n.q. and its degree of desirability n.dd, which is the average of dd of zones overlapping with n.q. An edge (n,n') connecting two nodes is also assigned a dd equal to the average of dd of configurations in the path-segment (n.q,n'.q). The dd of a path (i.e., a sequence of nodes) is an average of dd of its edges.

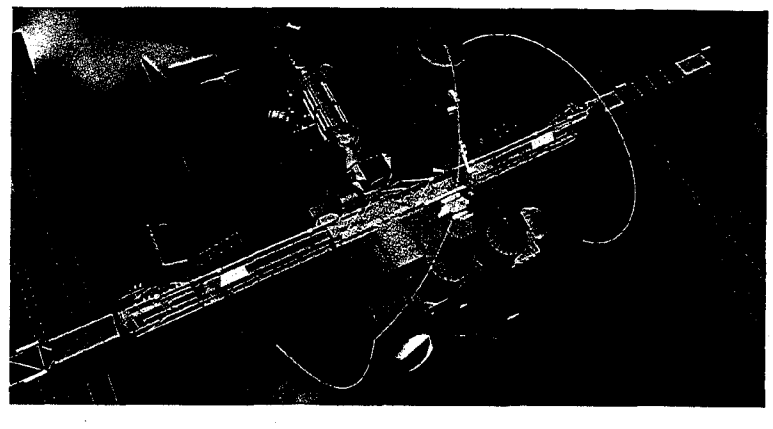


Figure 3.3: SSRMS going through three different cameras fields of view (purple, green and blue cones) and avoiding an undesirable zone (rectangular zone in pale red)

Following probabilistic roadmap methods (PRM) [58], we build the roadmap by picking robot configurations probabilistically, with a probability that is biased by the density of obstacles. A path is then a sequence of collision free edges in the roadmap, connecting the initial and goal configuration. Following the Anytime Dynamic A* (AD*) approach [48], to get new paths when the conditions defining safe zones have dynamically changed, we can quickly re-plan by exploiting the previous roadmap. On the other hand, paths are computed through incremental improvements so the planner can be stopped at anytime to provide a collision-free path (i.e., anytime after the first such path has been found) and the more time it is given, the more the path is optimized to move through desirable zones.

FADPRM works as follows. The input is an initial configuration, a goal configuration, a 3D model of obstacles in the workspace, a 3D specification of zones with corresponding dd and a 3D model of the robot. Given this input:

- To find a path connecting the input and goal configuration, we search backward from the goal towards the initial (current) robot configuration. Backward search instead of forward search is done because the robot moves and, hence, its current configuration is not in general the initial configuration; we want to re-compute a path to the same goal when the environment changes before the goal is reached.
- A probabilistic queue OPEN contains nodes of the frontier of the current roadmap (i.e., nodes are expanded because they are new or because they have previously been expanded but are no longer up to date w.r.t. to the desirable path) and a list CLOSED

contains non frontier nodes (i.e., nodes already expanded).

- Search consists of repeatedly picking a node from OPEN, generating its predecessors and putting the new ones or out of date ones in OPEN.
- The density of a node is the number of nodes in the roadmap with configurations that are a short distance away (proximity being an empirically set parameter, taking into account the obstacles in an application domain). The distance estimate to the goal takes into account the node's dd and the Euclidean distance to the goal. A node n in OPEN is selected for expansion with probability proportional to :

$$(1-\beta)/density(n) + \beta * goal - distance - estimate(n)$$

$with\ 0 \leq \beta \leq 1.$

This equation implements a balance between fast-solution search and best-solution search by choosing different values for β. With β near to 0, the choice of a node to be expanded from OPEN depends only on the density around it. That is, nodes with lower density will be chosen first, which is the heuristic used in traditional PRM approaches to guarantee the diffusion of nodes and to accelerate the search for a path [58]. As β approaches 1, the choice of a node to be expanded from OPEN will rather depend on its estimated distance to the goal. In this case, we are seeking optimality rather than the speed of finding a solution.

- To increase the resolution of the roadmap, a new predecessor is randomly generated within a small neighborhood radius (that is, the radius is fixed empirically based on the density of obstacles in the workspace) and added to the list of successors in the roadmap generated so far. The entire list predecessors is returned.
- Collision is delayed: detection of collisions on the edges between the current node and its predecessors is delayed until a candidate solution is found; if there is a collision, we backtrack. Collisions that have already been detected are stored in the roadmap to avoid doing them again.
- The robot may start executing the first path found.
- Concurrently, the path continues being improved by replanning with an increased value of β.
- Changes in the environment (moving obstacles or changes in dd for zones) cause updates of the roadmap and replanning.

The calculation of a configuration dd and a path dd is a straightforward extension of collision checking for configurations and path segments. For this, we customized the Proximity Query Package (PQP) [43]. The 3D models for the ISS, the SSRMS and zones are implemented using a customization of Silicon Graphics' Open Inventor. The robot is modeled using Motion Planning Kit (MPK), that is, an implementation of Sanchez and Latombe's PRM planner [58].

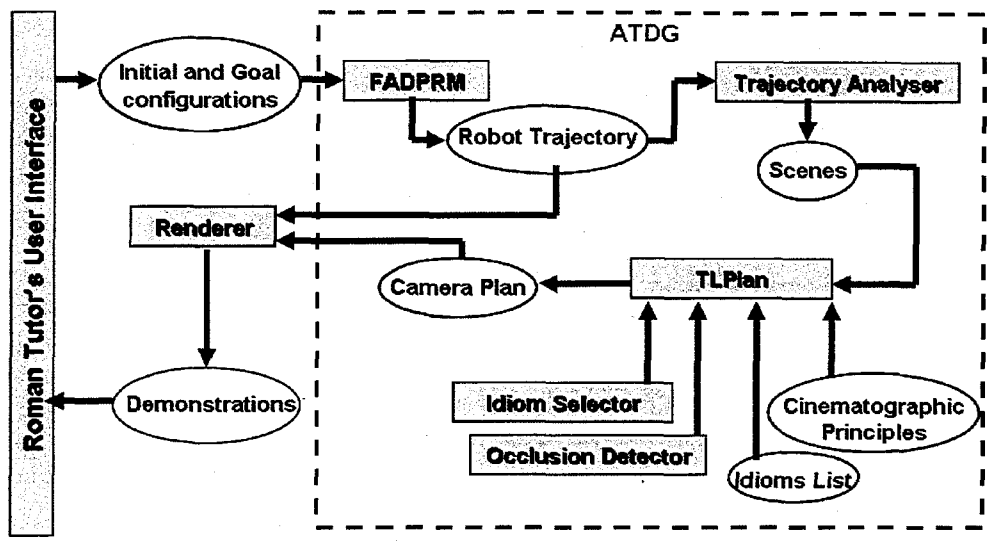
3.3 The Automatic Task Demonstration Generator

The automatic task demonstration generator (ATDG) [34] takes as input a start and a goal configuration of the SSRMS. ATDG will generate a movie demonstration of the required manipulations that bring the SSRMS from the start configuration to the goal configuration. The top figure in Figure 3.4 shows the internal architecture of the ATDG. The bottom one shows the different steps the data go through in order to transform the two given configurations into a complete movie demonstration.

First, ATDG calls the FADPRM path planner to generate a collision free path between the two given configurations. The path is then passed to the trajectory parser which separates it into categorized segments. This will turn the continuous trajectory into a succession of scenes, where each scene can be filmed by a specific group of idioms. An idiom is a succession of shots that represents a stereotypical way to film a scene category. The parser looks for uniformity in the movements of the SSRMS to detect and recognize the category of scenes. Once the path is parsed, a call is made to the camera planner TLPlan to find the best shots that best convey each scene, while making sure the whole is pleasing and comprehensive.

The use of TLPlan as a camera planner within ATDG provides two advantages. First LTL, the language used by TLPlan is more expressive, yet with a simpler defined semantics, than previous camera planning languages such as DCCL [15]. For instance, we can express arbitrary temporal conditions about the order in which objects should be filmed, which objects should remain in the background until some condition become true and more complex constraints that the LTL language can express. Secondly, TLPlan is more powerful than other camera planners presented in the literature such as [15, 5, 26, 33] because with TLPlan, LTL shot composition rules provide a search pruning capability. In ATDG,

3.3. THE AUTOMATIC TASK DEMONSTRATION GENERATOR



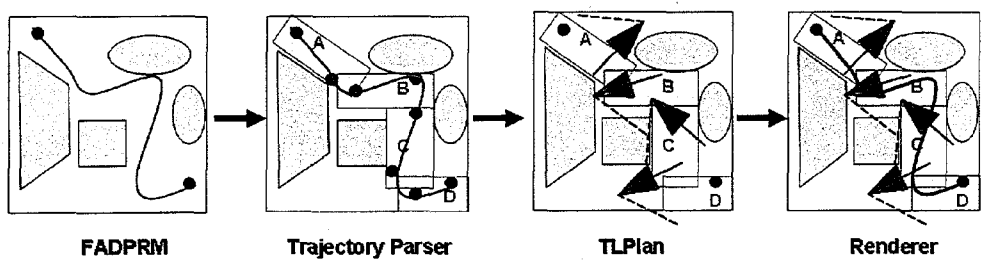


Figure 3.4: ATDG architecture

each shot in the idiom is distinguished by three key attributes: shot type, camera placement mode, camera zooming mode.

- Shot Types: five shot types are currently defined in the ATDG System: Static, GoBy, Pan, Track and Pov. A Static shot for example is done from a static camera when the robot is in a constant position or moving slowly. Whereas in a Track shot, a camera follows the robot and keeps a constant distance from it.
- Camera Placements: for each shot type, the camera can be placed in five different ways according to some given line of interest: External, Parallel, Internal, Apex and External II. Currently, we take the trajectory of the robot's center of gravity as the line of interest which allows filming of a number of many typical manoeuvres. For

larger coverage of manoeuvres, additional lines of interest will be added later.

- Zoom modes: for each shot type and camera placement, the zoom of the camera can be in five different modes: Extreme Close up, Close up, Medium View, Full View and Long View.

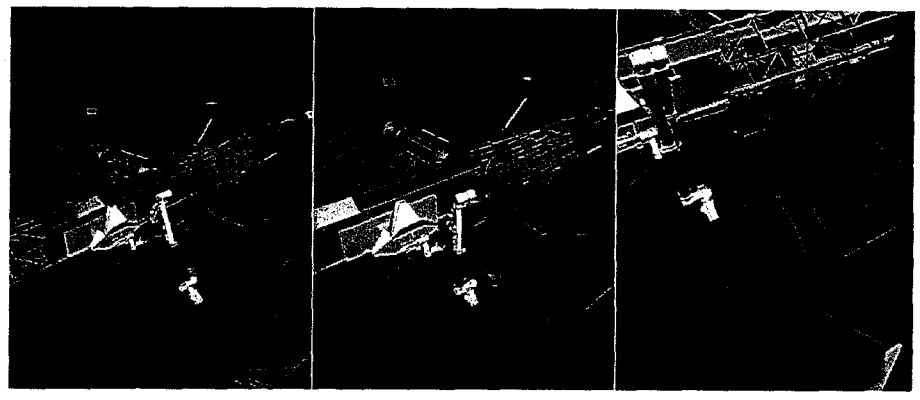


Figure 3.5: Idiom to film the SSRMS anchoring a new component on the ISS

Figure 3.5 shows an idiom illustrating the anchoring of a new component on the ISS. It starts with a Track shot following the robot while moving on the truss. Then, another Track shot follows that shows the rotation of one joint on the robot to align with the ISS structure. And finally there is a Static shot focusing on the anchoring operation. In TLPlan, idioms are specified in the Planning Definition Language (PDDL 3.0). Intuitively, a PDDL operator specifies preferences about shot types in time and in space depending on the robot manoeuver. Parsing the trajectory of the robot with the successive scenes performed, TLPlan will try to find a succession of shots that captures the best possible idioms. TLPlan also takes into account cinematic principles to ensure consistency of the resulting movie. Idioms and cinematic principles are in fact encoded in the form of temporal logic formulas within the planner. TLPlan uses also an occlusion detector to make sure the SSRMS is visible all the time. Once TLPlan is done, we are left with a list of shots that is passed to the rendering system to create the animation. The renderer uses both the shots given by TLPlan and the SSRMS trajectory in order to position the cameras in relation with the SSRMS, generating the final task demonstration.

For each SSRMS movement type (or scene), we have several idioms (from six to ten in the current implementation) and each idiom is defined by taking into account the com-

plexity of the movement, the geometry of the ISS, the visual cues on the ISS and subjective expectations of the viewer. For example, if the SSRMS and the mobile base are moving along the main truss of the ISS, it is important that the camera show not only the entire arm but also some visual cues on the ISS so the operator can get a sense of situational awareness for the relocation of the base of the arm. Consequently, the idioms for this manipulation will involve shots with a Full or Long View zoom. In contrast, movements involving the end effector require a high precision, so an Extreme Close Up zoom will be involved.

3.4 Roman Tutor Architecture and Basic Functionalities

3.4.1 Architecture and Main Components

Roman Tutor works with any robot manipulator provided a 3D model of the robot and its workspace are specified. Roman Tutor's architecture includes the following components (Figure 3.6): the graphic user interface, the State Reflector, the FADPRM path planner, the automatic task demonstration generator ATDG, the Tutoring Module and the Simulator Core with several third-party libraries: Proximity Query Package (PQP) [43], Open Inventor from Silicon Graphics and Motion Planning Kit (MPK) [58].

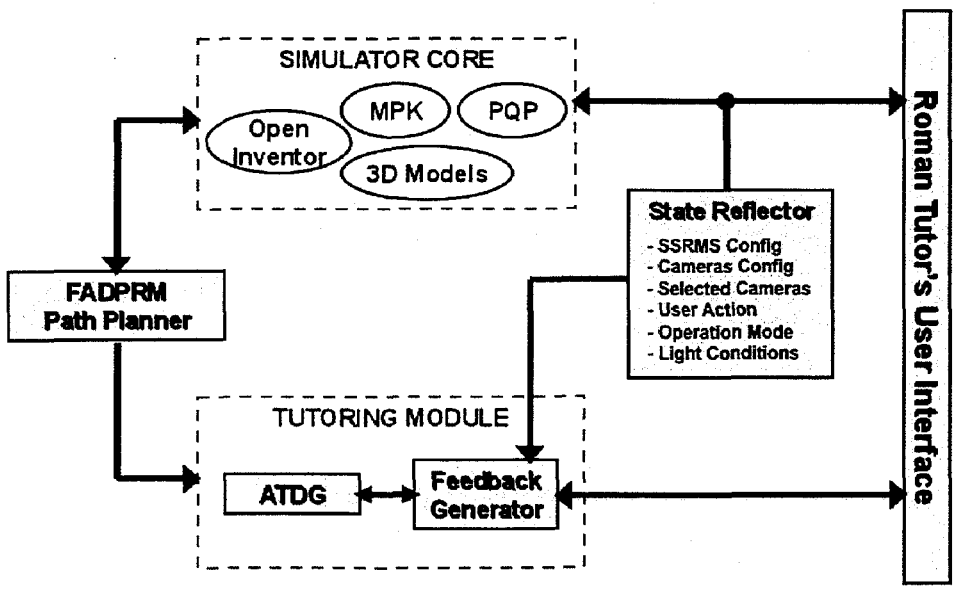


Figure 3.6: Roman Tutor architecture

As shown in Figure 3.2, Roman Tutor's user interface has three screens (for the three monitors). The keyboard is used to operate the robot (the SSRMS in our case). In command mode, one controls the joints directly; in automatic mode, one moves the end-effector, small increments at a time, relying on inverse kinematics to calculate the joint rotations. In Figure 3.2, different cameras are selected, displaying the same robot configuration from different viewpoints. The perspective camera (on the left) can inspect the entire ISS 3D model. It is used in training tasks aimed at helping a student to develop a mental 3D model of the ISS even though there is no such camera on the ISS. Normal training uses small physical models of the ISS for the same purpose.

In Roman Tutor students could carry out several kinds of training tasks that we describe more formally in the next section. The State Reflector periodically updates the student's actions (i.e., keyboard inputs) and their effects on the ISS environment (robot configuration, cameras mapped to the monitors, their view angles and the operation mode). It also monitors lighting conditions.

3.4.2 Training Tasks

Training tasks can be classified as open, recognition, localization, or robot manipulation. Open tasks are those in which the student interacts with the simulator, without any formally set goal, with minimal assistance configured in the system's preferences (e.g., collision warning and avoidance). Recognition tasks train to recognize the different elements in the workspace. An example is to show a picture of an element of the ISS and ask the student to name it and describe its function. Localization tasks train to locate visual cues or ISS elements and to relate them spatially to other elements. An example is to show a picture of a visual cue and ask the student to make it visible on the screen using an appropriate selection of cameras; or we can ask to name elements that are above another element shown on the screen.

Robot operation tasks deal with moving the manipulator (avoiding collision and singularities, using the appropriate speed, switching cameras as appropriate and using the right operation mode at different stages), berthing, or mating. An illustration is to move the arm from one position to another, with or without a payload. Another example is to inspect an indicated component of the ISS using a camera on the end-effector. These tasks require

the student to be able to define a corridor in a free workspace for a safe operation of the robot and follow it. The student must do this based on the task, the location of cameras and visual cues and the current lighting conditions. Therefore localization and navigation are important in robot operations. Robot operation tasks are made more or less unpredictable by dynamically changing the lighting conditions, thus requiring the revalidation of safe corridors.

3.4.3 Tutoring Approaches in Roman Tutor

The Feedback Generator inside the Tutoring Module (Figure 3.6) periodically checks the current state to trigger feedback to the student, using rules that are preconditioned on the current state information and the current goal. These are "teaching" expert rules and can be as efficient as the available teaching expertise allows. Feedback rules take into account how long the student has been trying on a subtask and how good or bad he is progressing on it.

In the context of open, recognition and localization tasks, providing tutoring assistance seems straight forward. The domain knowledge is well defined: what element or cue of the ISS to recognize or to localize? what camera to choose and when? etc. Here, we follow a model-tracing approach and define for each category of tasks a well structured task model to support the tutoring process. Task models are designed by hand starting from recommendations provided by human experts and are structured in the form of a graph encoding if-then rules. The Feedback Generator uses the predefined task graphs to validate student actions, identify gaps and provide feedback accordingly.

As we stated previously in an early section, the domain of robot manipulations is an "ill-structured" domain where classical tutoring approaches start to loose efficiency and show limitations. To overcome these limitations, we choose to follow an "expert system approach" and use the FADPRM path-planner as a domain expert in our system to support the tutoring process. In the context of robot manipulation tasks, the Feedback Generator evaluates student actions by comparing it to the optimal solutions found by FADPRM and provides useful feedback accordingly. The tutoring process that uses FADPRM as an expert of the domain knowledge is described in more details in the next section.

One of the very important early results in intelligent tutoring research is the importance

of the cognitive fidelity of the domain knowledge module. That is, it is important for the tutor to reason about the problem in the same way that humans do [17]. Approaches for modeling a domain expert within intelligent tutoring systems can be grouped into three main categories: black box models, glass box models and cognitive models [55]. The main difference between these models lies in the cognitive fidelity with which each model represents the expert domain knowledge.

A black box model describes problem states differently than the student. The classic example of such a system is SOPHIE I [12]. SOPHIE I is a tutor for electronic troubleshooting that used its expert system to evaluate the measurements students were making in troubleshooting a circuit. The expert system made its decisions only by solving sets of equations. A glass box model is an intermediate model that reasons in terms of the same domain constructs as the human expert. However, the model reasons with a different control structure than the human expert. A classic example of such a system is GUIDON [16], a tutoring system for medical diagnosis. This system was built around MYCIN, an expert system for the treatment of bacterial infections. A cognitive approach, on the other hand, aims to develop a cognitive model of the domain knowledge that captures the way knowledge is represented in the human mind in order to make the tutor respond to problem-solving situations in a way very similar to humans. This approach, in contrast to the other approaches, has as an objective to support cognitively plausible reasoning [55]. A good example for such a tutoring system is SHERLOCK [46], another practice environment for electronics troubleshooting. SHERLOCK used a procedural domain knowledge representation based on a cognitive analysis of human skill acquisition.

By taking into account the disposition of cameras on the ISS, FADPRM reasons about actions in a way very similar to students. Thus, the use of FADPRM as a domain expert in Roman Tutor results in a tutoring approach that lies in between a glass box approach and a cognitive approach. Even if we are applying it in the context of an "ill-structured" domain, we believe that this will guarantee good quality of the tutoring provided to the student, at least at the same level as the one provided by a glass box model like GUIDON. In the next section, we describe and evaluate the tutoring provided using FADPRM as an expert of the domain to a student working on robot manipulation tasks.

3.5 FADPRM as a Domain Expert in Roman Tutor

Roman Tutor initiates a robot manipulation task and monitors the student's progress towards accomplishing it. Students begin the task and can ask Roman Tutor for help or for a recommendation about what to do next. Students can ask Roman Tutor about how to avoid a collision with a nearby obstacle, how to go to a desirable location in the workspace or how to go through a desirable zone. In this situation, the Feedback Generator calls the ATDG (which calls the FADPRM planner) to compute and show a movie illustrating how to complete the manipulation task. If the objective is to give the operator a sense of the task as he will be seeing it from the command and control workstation, then virtual camera positions will be selected from the 14 cameras on the exterior of the ISS. But if the objective is to convey some cognitive awareness of the task, then virtual cameras are selected to best help the operator gain a maximal cognitive awareness.

Using the real time dynamic capability of the FADPRM path planner, the Feedback Generator monitors the student's activity in the State Reflector to validate incrementally student's action or sequence of actions, give information about the next relevant action or sequence of actions. The Feedback Generator regularly evaluates whether the task can be completed from the current configuration of the manipulator and whether it can be completed efficiently. At the point at which it discovers that the student would have to backtrack from the current position or that achieving the task takes more than the time planned for it, the Feedback Generator will intervene and begin to show the student a more efficient trajectory. Once a better initial trajectory has been demonstrated, the student can take control and resume the task. This error-prompted turn taking repeats until the task is completed (Figure 3.7). We see here the importance of having FADPRM as a planner in our system to guide the operations by the student. By taking into account the disposition of the cameras on the station, we are assured that the plan shown to the student passes through zones that are visible from cameras placed in the ISS environment and can then be followed by the student.

To evaluate the tutoring mechanics we implemented to support a student working on robot manipulation tasks, we compare the types of feedback we provide in our application to those provided by a classic intelligent tutoring system SHERLOCK [46] that is known to be efficient. SHERLOCK is a practice environment for electronics troubleshooting and

3.5. FADPRM AS A DOMAIN EXPERT IN ROMAN TUTOR

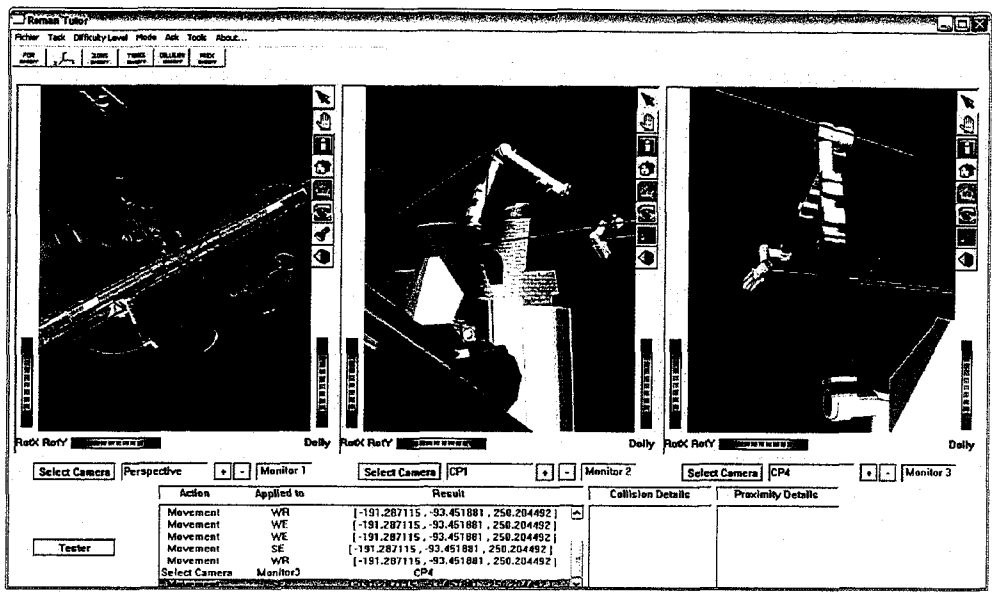


Figure 3.7: Roman Tutor showing a robot trajectory to the student

provides advice on problem solving steps upon student request. Four types of feedback are available [17]: (1) advice on what test action to carry out and how, (2) advice on how to read the outcome of the test, (3) advice on what conclusion can be drawn from the test and (4) advice on what option to pursue next.

As described earlier, our "FADPRM as a domain expert" tutoring approach provides feedback not only upon request but also intervenes automatically when it detects errors or difficulties experienced by the student. Different types of feedback are also available : (1) advice on what action (or manipulation) to execute and how by showing at each step a valid path to the goal or by showing a movie computed with ATDG, (2) advice on how to avoid errors while progressing on a task by showing paths that avoid a nearby obstacle or by showing movies recorded from the most useful cameras, (3) advice on what conclusion can be drawn from the errors made by detecting incorrect choices made by the student and by proposing the right path to follow and (4) advice on what action or sequence of actions to pursue next in order to reach the goal.

The types of feedback provided by our tutoring approach are at a level of expressiveness very similar to those provided by SHERLOCK. By using FADPRM as a domain expert within Roman Tutor and despite the fact that we are working in an "ill-structured" domain,

we succeeded in achieving a level of quality for the tutoring similar to the one provided by an ITS with a cognitive representation for the domain expert.

3.6 Conclusion

In this paper, we presented a real-time flexible approach for robot path planning called FADPRM and showed how it can be used efficiently to provide very helpful feedback to a student on a robot manipulation training simulator. FADPRM supports spatial reasoning and makes model tracing tutoring possible without any explicit task structure. By using FADPRM as a domain expert within the simulator, we showed how to achieve a high quality level for the tutoring assistance without planning in advance what feedback to give to the student and without creating a complex task graph to support the tutoring process.

We also detailed the architecture of the intelligent training simulator Roman Tutor in which FADPRM is integrated. Among other components, Roman Tutor contains an automatic task demonstration generator ATDG used for the on the fly generation of useful task demonstrations that help the student carry on his manipulation tasks on the simulator.

Roman Tutor's benefits to future training strategies are (1) the simulation of complex tasks at a low cost (e.g., using inexpensive simulation equipment and with no risk of injuries or equipment damage) and (2) the installation anywhere and anytime to provide "just in time" training. Crew members would be able to use it onboard the ISS, for example, to study complex maintenance or repair operations. For very long missions, they would be able to use it to train regularly in order to maintain their skills. In particular Roman Tutor is able to generate as many training examples as the student wants. This capacity provides important learning challenges and opportunities that are not possible with the current system based on a fixed set of manually generated examples. Although motivated by the SSRMS application, Roman Tutor with its innovative components (FADPRM and ATDG) remains applicable to any other tele-robotics system application.

3.7 Acknowledgement

The work presented herein was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Conclusion

Cette thèse avait pour objectif le développement d'un simulateur tutoriel intelligent pour les opérations robotisées, appelé Roman Tutor, applicable à la manipulation du bras canadien (SSRMS) sur la station spatiale internationale (SSI). Ce simulateur intègre trois composantes originales. La première consiste en un nouveau planificateur de trajectoires appelé FADPRM pour des environnements dynamiques présentant des contraintes dures et flexibles. En plus de déterminer une trajectoire sans collisions pour le robot à travers les obstacles, FADPRM permet d'ajouter des préférences quant à la navigation de ce dernier dans l'environnement de manipulation. La deuxième composante est un générateur automatique de démonstrations de tâches ATDG. Ce dernier fait appel à FADPRM pour trouver une trajectoire solution pour une tâche de déplacement du robot et à de nouvelles techniques de planification de la caméra du rendu pour filmer la solution obtenue. Enfin, la troisième composante intégrée au sein de Roman Tutor consiste en une modélisation pédagogique implémentant des stratégies d'intervention pour donner de l'aide à un opérateur manipulant le SSRMS sur Roman Tutor. L'aide à l'apprentissage fait appel d'une part à des démonstrations de tâches générées automatiquement par le ATDG, et d'autre part au planificateur de trajectoires pour suivre la progression de l'opérateur sur sa tâche, lui fournir de l'aide et le corriger au besoin.

Dans le premier chapitre de cette thèse, nous avons présenté un article qui introduit la nouvelle approche de planification de trajectoires FADPRM que nous avons développée. Cet article a été soumis au journal "International Journal of Robotics Research". La nouvelle approche FADPRM pour " Flexible Anytime Dynamic PRM" étend les approches PRM de base pour s'appliquer dans des environnements contenant des régions avec des degrés de désirabilité. Cette nouvelle approche intègre les stratégies de recherche dynamique et "anytime" pour faire face à des environnements dynamiques, où les obstacles et

les zones avec degré de désirabilité peuvent changer en temps réel. La stratégie dynamique permet au planificateur de replanifier efficacement en exploitant les résultats trouvés dans les itérations précédentes. La stratégie "anytime" commence par une trajectoire rapidement calculée avec un degré de désirabilité potentiellement faible qui est ensuite progressivement augmenté si plus de temps de planification est permis.

Dans le cadre de la manipulation du SSRMS sur la SSI, la caractéristique flexible permet à FADPRM de prendre en considération la disposition des caméras sur la station pour générer des trajectoires solutions en tout point visibles par un opérateur sur Roman Tutor. Les expériences conduites dans l'article ont validé les différentes caractéristiques de FADPRM. Cependant, bien que nous obtenions une meilleure qualité pour les trajectoires générées avec un meilleur temps de replanification en comparaison avec les approches PRM de base, il persiste toutefois un potentiel d'amélioration sur ces deux dimensions. En effet, les trajectoires nécessitent toujours une étape de post-traitement pour les lisser, et dans les applications en temps réel, nous cherchons toujours à avoir un algorithme de planification le plus rapide possible. Comme travail futur, nous continuerons donc à explorer des moyens pour améliorer notre approche en adoptant et en testant de nouvelles alternatives.

L'article introduit dans le deuxième chapitre présente le générateur automatique de démonstrations de tâche ATDG. Cet article a été soumis au journal "International Journal of Knowledge-Based and Intelligent Engineering Systems". ATDG est implémenté au sein de Roman Tutor pour générer des démonstrations de tâches de manipulation du SSRMS sur la SSI. ATDG intègre une nouvelle approche de planification des caméras qui apporte des améliorations aux techniques de génération automatique d'animations suivant deux dimensions principales. Premièrement, elle utilise la logique temporelle pour exprimer les principes cinématographiques, un langage plus expressif et avec une sémantique plus simple que les anciens langages de planification de caméras. Deuxièmement, elle utilise l'algorithme TLPlan, un planificateur plus puissant que les planificateurs précédents, puisque avec TLPlan, les règles de composition LTL permettent un contrôle efficace de la recherche.

Nous avons obtenu des résultats très prometteurs en utilisant des mesures simples pour la qualité des films générés en comparant avec d'anciennes approches de planification de caméras usuelles. L'ajout de repères visuels comme contraintes dans la planification est assez simple et se fera dans le cadre d'un travail futur. Dans la version actuelle du ATDG,

nous utilisons les formules de logique temporelle dans TLPlan seulement pour encoder les lois de compositions cinématographiques. Dans un travail futur, nous planifions d'étendre l'utilisation des formules LTL à la description des idiomes. Cette approche permettra de bénéficier d'un langage plus riche pour la spécification des idiomes.

Le troisième et dernier article de cette thèse concerne le modèle pédagogique implémenté au sein de Roman Tutor pour encadrer l'apprentissage de manipulations robotisées. Cet article a été soumis au journal "IEEE Transactions on Learning Technology". Nous y avons montré comment le planificateur de trajectoires FADPRM peut être efficacement utilisé pour fournir des rétroactions incroyablement utiles à un apprenant sur Roman Tutor. En utilisant FADPRM comme un expert du domaine dans le simulateur, et ATDG comme générateur automatique de démonstrations, nous avons montré qu'on pouvait atteindre un haut niveau de qualité dans l'assistance tutorielle sans pour autant planifier à l'avance quelles rétroactions fournir, ni créer au préalable et explicitement un graphe de tâches complexe pour appuyer et guider le processus de tutorat.

Une fois en orbite, les astronautes sur la SSI ne disposent actuellement que de simple matériel basé Web pour le maintien et le développement de leurs compétences relatives à la manipulation du SSRMS. Pour les missions spatiales de longue durée, les astronautes auront besoin d'outils de simulation plus sophistiqués pour maintenir leurs compétences. Roman Tutor répond à ces besoins en fournissant un outil de formation intelligent et portable. Il intègre, entre autres, un environnement de simulation réaliste du SSRMS sur la SSI, un planificateur de trajectoires efficace qui tient compte des contraintes de visibilité sur la station, et un générateur automatique de démonstrations de tâches. Roman Tutor génère des exemples de tâches à accomplir par un apprenant dans l'environnement de simulation et surveille sa progression, pour lui fournir des informations pertinentes en cas de besoin. Bien que motivé par l'application du bras canadien sur la SSI, le concept de simulateur intelligent Roman Tutor avec ses différentes composantes innovatrices dont FADPRM et ATDG, reste applicable et transférable à tout autre application d'opérations robotisées.

Bibliographie

[1] R. ANGROS, W.L. JOHNSON, J. RICKEL et A. SCHOLER. « Learning Domain Knowledge for Teaching Procedural Skills ». Dans Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 1372–1378, 2002.

[2] D. ARIJON. Grammar of the Film Language. Communication Arts Books, Hastings House Publichers, New York, 1976.

[3] F. BACCHUS et F. KABANZA. « Using Temporal Logics to Express Search Control Knowledge for Planning ». Artificial Intelligence, 116(1-2) :123–191, 2000.

[4] W.H. BARES, J.P. GREGOIRE et J.C. LESTER. « Real-time Constraint-Based Cinematography for Complex Interactive 3D Worlds ». Dans Proceedings of 15th National Conference on Artificial Intelligence (AAAI/IAAI), pages 1101–1106, 1998.

[5] W.H. BARES, L.S. ZETTLEMOYER, D.W. RODRIGUEZ et J.C. LESTER. « Task-sensitive Cinematography Interfaces for Interactive 3D Learning Environments ». Dans Proceedings of Intelligent User Interfaces, pages 81–88, 1998.

[6] K. BELGHITH, F. KABANZA et L. HARTMAN. « Using a Randomized Path Planner to Generate 3D Task Demonstrations of Robot Operations ». Dans Proceedings of the 1st International Conference on Autonomous and Intelligent Systems (AIS), 2010.

[7] K. BELGHITH, F. KABANZA, L. HARTMAN et R. NKAMBOU. « Anytime Dynamic Path-planning with Flexible Probabilistic Roadmaps ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 2372–2377, 2006.

[8] F. BENHAMOU, F. GOUALARD, E. LANGUENOU et M. CHRISTIE. « Interval Constraint Solving for Camera Control and Motion Planning ». Journal of ACM Transactions on Computational Logic, 5(4) :732–767, 2004.

[9] J. P. Van Den BERG et M. H. OVERMARS. « Using Workspace Information as a Guide to Non-Uniform Sampling in Probabilistic Roadmap Planners ». International Journal on Robotics Research, 24(12) :1055–1071, 2005.

[10] J. BLINN. « Where Am I ? What Am I Looking At ? ». Journal IEEE Computer Graphics and Applications, 8(4) :76–81, 1988.

[11] R. BOHLIN et L. KAVRAKI. « Path Planning Using Lazy PRM ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 521–528, 2000.

[12] J.S. BROWN, R. BURTON et F. ZDYBEL. « A Model-Driven Question-Answering System for Mixed Initiative Computer-Assisted Instruction ». Journal of IEEE Transactions on Systems, Man and Cybernetics, 3(3) :248–257, 1973.

[13] B. BURNS et O. BROCK. « Sampling-Based Motion Planning Using Predictive Models ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 3120–3125, 2005.

[14] H. CHOSET, K.M. LYNCH, S. HUTCHINSON, G.A. KANTOR, W. BURGARD, L.E. KAVRAKI et S. THRUN. Principles of Robot Motion : Theory, Algorithms, and Implementations. MIT Press, Cambridge, 2005.

[15] D.B. CHRISTIANSON, S.E. ANDERSON, L. HE, D.H. SALESIN, D.S. WELD et Cohen M.F.. « Declarative Camera Control for Automatic Cinematography ». Dans Proceedings of the 13th National Conference on Artificial Intelligence (AAAI/IAAI), pages 148–155, 1996.

[16] W.J. CLANCEY. Tutoring Rules for Guiding a Case Method Dialogue. In D. Sleeman and J. Brown (Eds.) Intelligent tutoring systems. New York : Academic Press, 1982.

[17] A.T. CORBETT, K.R. KOEDINGER et J.R. ANDERSON. Intelligent Tutoring Systems (Chapter 37). In : Handbook of Human-Computer Interaction, Second, Completely Revised Edition in M. Helander, T. K. Landauer, P. Prabhu (Eds). Elsevier Science, 1997.

[18] R.S. CROWLEY, E. LEGOWSKI, O. MEDVEDEVA, E. TSEYTLIN, E. ROH et D. JUKIC. « An ITS for Medical Classification Problem-Solving : Effects of Tutoring and Representations ». Dans Proceedings of the 12th International Conference on Artificial Intelligence in Education, pages 192–199, 2005.

[19] N. CURRIE et B. PEACOCK. « International Space Station Robotic Systems Operations : A Human Factors Perspective ». Dans Proceedings of Human Factors and Ergonomics Society Annual Meeting, Aerospace Systems, pages 26–30. Human Factors and Ergonomics Society, 2002.

[20] T. DEAN et M. BODDY. « An Analysis of Time-Dependent Planning ». Dans Proceedings of the 14th National Conference on Artificial Intelligence (AAAI), 1988.

[21] S.M. DRUCKER, T.A. GALYEAN et D. ZELTZER. « CINEMA : A System for Procedural Camera Movements ». Dans Proceedings of Symposium on Interactive 3D Graphics (SI3D), pages 67–70, 1992.

[22] D. FERGUSON, M. LIKHACHEV et A. STENTZ. « A Guide to Heuristic-based Path-Planning ». Dans Proceedings of ICAPS Workshop on Planning under uncertainty for Autonomous Systems, pages 9–18, 2005.

[23] K.D. FORBUS. « Articulate Software for Science and Engineering Education ». Smart Machines in Education : The Coming Revolution in Educational Technology, pages 235–267, 2001.

[24] P. FOURNIER-VIGER, R. NKAMBOU et E. Mephu NGUIFO. Supporting Tutoring Services in Ill-Defined Domains. In : Nkambou et al. (eds.) Advances in Intelligent Tutoring Systems. Springer, 2010.

[25] D.A. FRIEDMAN et Y.A. FELDMAN. « Knowledge-Based Cinematography and its Applications ». Dans Proceedings of the 13th European Conference on Artificial Intelligence (ECAI), pages 256–262, 2004.

[26] D.A. FRIEDMAN et Y.A. FELDMAN. « Automated Cinematic Reasoning about Camera Behavior ». Journal of Expert Systems with Applications, 30(4) :694–704, 2006.

[27] N. HALPER, R. HELBING et T. STROTHOTTE. « A Camera Engine for Computer Games : Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence ». Journal of Computer Graphics Forum, 20(3), 2001.

[28] N. HALPER et P. OLIVIER. « CAMPLAN : A Camera Planning Agent ». Dans Proceedings of Smart Graphics, AAAI Spring Symposium, pages 92–100, 2000.

[29] P. HART, N. NILSSON et B. RAFAEL. « A Formal Basis for the Heuristic Determination of Minimum Cost Paths ». Journal of IEEE Transactions on Systems, Science and Cybernetics, SSC-4(2) :100–107, 1968.

[30] D. HSU, J.C. LATOMBE et H. KURNIAWATI. « On the Probabilistic Foundations of Probabilistic Roadmap Planning ». International Journal of Robotics Research, 25(7) :627–643, 2006.

[31] D. HSU, G. SANCHEZ-ANTE et Z. SUN. « Hybrid PRM Sampling with a Cost-Sensitive Adaptive Strategy ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 3885–3891, 2005.

[32] F. JARDILLIER et E. LANGUENOU. « Screen-Space Constraints for Camera Movements : the Virtual Cameraman ». Journal of Computer Graphics Forum, 17(3) :175–186, 1988.

[33] A. JHALA et R.M. YOUNG. « A Discourse Planning Approach to Cinematic Camera Control for Narratives in Virtual Environments ». Dans Proceedings of the 20th National Conference on Artificial Intelligence (AAAI/IAAI), pages 307–312, 2005.

[34] F. KABANZA, K. BELGHITH, P. BELLEFEUILLE, B. AUDER et L. HARTMAN. « Planning 3D Task Demonstrations of a Teleoperated Space Robot Arm ». Dans Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS), pages 164–173, 2008.

[35] F. KABANZA, R. NKAMBOU et K. BELGHITH. « Path-Planning for Autonomous Training on Robot Manipulators in Space ». Dans Proceedings of the 19th International Joint Conference In Artificial Intelligence (IJCAI), pages 1729–1731, 2005.

[36] L. KAVRAKI, P. SVESTKA, J. C. LATOMBE et M. OVERMARS. « Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces ». IEEE Transactions on Robotics and Automation, 12(4) :566–580, 1996.

[37] K.R. KOEDINGER, J.R. ANDERSON, W.H. HADLEY et M.A. MARK. « Intelligent Tutoring Goes to School in the Big City ». Intenational Journal of Artificial Intelligence in Education, 8(9) :30–43, 1997.

[38] S. KOENIG et M. LIKHACHEV. « D*Lite ». Dans Proceedings of the 18th National Conference on Artificial Intelligence (AAAI/IAAI), pages 476–483, 2002.

[39] S. KOENIG et M. LIKHACHEV. « Adaptive A* ». Dans Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 1311–1312, 2005.

[40] H. KURNIAWATI et D. HSU. « Workspace Importance Sampling for Probabilistic Roadmap Planning ». Dans Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1618–1623, 2004.

[41] H. KURNIAWATI et D. HSU. Workspace-Based Connectivity Oracle : An Adaptive Sampling Strategy for PRM Planning. Dans S. AKELLA et OTHERS, éditeurs, Proceedings of the 7th International Workshop on the Algorithmic Foundations of Robotics (WAFR). Springer, 2006.

[42] E. LANGUENOU et L. GRANVILLIERS. « Modelling Camera Control with Constrained Hypertubes ». Dans Proccedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP), pages 618–632, 2002.

[43] E. LARSEN, S. GOTTSHALK, M.C LIN et D. MANOCHA. « Fast Proximity Queries with Swept Sphere Volumes ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 3719–3726, 2000.

[44] S. M. LAVALLE. Planning Algorithms. Cambridge University Press, 2006.

[45] S.M. LAVALLE, M.S. BRANICKY et S.R. LINDEMANN. « On the Relationship between Classical Grid Search and Probabilistic Roadmaps ». International Journal of Robotics Research, 23(7-8) :673–692, 2004.

[46] A. LESGOLD, G. EGGAN, S. KATZ et G. RAO. Possibilities for Assessment Using Computer-Based Apprenticeship Environments. In J. Regian and V. Shute (Eds.) Cognitive Approaches to Automated Instruction. Hilisdale, NJ : Lawrence Eribaum Associates, 1992.

[47] M. LIKHACHEV, D. FERGUSON, G. J. GORDON, A. STENTZ et S. THRUN. « Anytime Search in Dynamic Graphs ». Artificial Intelligence, 172(14) :1613–1643, 2008.

[48] M. LIKHACHEV, D. I. FERGUSON, G. J. GORDON, A. STENTZ et S. THRUN. « Anytime Dynamic A* : An Anytime, Replanning Algorithm ». Dans Proceedings of

the 15th International Conference on Automated Planning and Scheduling (ICAPS), pages 262–271, 2005.

[49] M. LIKHACHEV, G. GORDON et S. THRUN. « ARA* : Anytime A* Search with Provable Bounds on Sub-Optimality ». Dans Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS), 2003.

[50] C. LUCAS. Directing for Film and Television. Anchor Press/Doubleday, Garden City NY, 1985.

[51] P. MELCHIOR, B. ORSONI, O. LAVIALLE, A. POTY et A. OUSTALOUP. « Consideration of Obstacle Danger Level in Path Planning using A^* and Fast-Marching Optimisation : Comparative Study ». Journal of Signal Processing, 83(11) :2387–2396, 2003.

[52] A. MITROVIC, M. MAYO, P. SURAWEERA et B. MARTIN. « Contraint-Based Tutors : a Success Story ». Dans Proceedings of the 14th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE), pages 931–940, 2001.

[53] D. NIEUWENHUISEN et M.H. OVERMARS. « Motion Planning for Camera Movements in Virtual Environments ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 3870– 3876, 2004.

[54] N. NILSSON. Principles of Artificial Intelligence. Tioga Publishing Company, 1980.

[55] R. NKAMBOU. Modeling the Domain : An Introduction to the Expert Module. In : Nkambou et al. (eds.) Advances in intelligent tutoring systems. Springer, 2010.

[56] R. NKAMBOU, K BELGHITH et F. KABANZA. « An Approach to Intelligent Training on a Robotic Simulator Using an Innovative Path-Planner ». Dans Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS), pages 645–654, 2006.

[57] M. SAHA, J.C. LATOMBE, Y. CHANG et F. PRINZ. « Finding Narrow Passages with Probabilistic Roadmaps : The Small-Step Retraction Method ». Journal of Autonomous Robots, 19(3) :301–319, 2005.

[58] G. SANCHEZ et J.C. LATOMBE. « A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking ». Dans Proceedings of the 10th International Symposium on Robotics Research (ISRR), pages 403–417, 2001.

[59] D. SENT et M. OVERMARS. « Motion Planning in Environments with Dangerzones ». Dans Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 1488–1493, 2001.

[60] H.A. SIMON. « The Structure of Ill Structured Problems ». Artificial Intelligence, 4(3) :181–201, 1973.

[61] X. SUN, S. KOENIG et W. YEOH. « Generalized Adaptive A* ». Dans Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 469–476, 2008.

[62] B. TOMLINSON, B. BLUMEBERG et D. NAIN. « Expressive Autonomous Cinematography for Interactive Virtual Environments ». Dans Proceedings of the 5th International Conference on Autonomous Agents, pages 317–324, 2000.

[63] K. VANLEHN. « The Advantages of Explicity Representing Problem Spaces ». Dans Proceedings of 9th International Conference on User Modeling (UM), page 3, 2003.