THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

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

Discipline ou spécialité :

Systèmes automatiques

Présentée et soutenue par :
Alireza NAKHAEI
le : jeudi 24 septembre 2009

Titre :
Motion planning and perception : integration on humanoid robots
Planification de mouvement, modélisation et perception : intégration sur un robot humanoïde

---

JURY

Prof. Seth HUTCHINSON Rapporteur
Dr. François PIERROT Rapporteur
Dr. Florent LAMIRAUX Directeur de Thèse
Dr. Jean-Paul LAUMOND Président du jury

Ecole doctorale :
Systèmes (EDSYS)
Unité de recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes
Directeur(s) de Thèse :

Dr. Florent LAMIRAUX

Rapporteurs :
Prof. Seth HUTCHINSON
Dr. François PIERROT

Abstract

This thesis starts by proposing a new framework for motion planning using stochastic maps, such as occupancy-grid maps. In autonomous robotics applications, the robot's map of the environment is typically constructed online, using techniques from SLAM. These methods can construct a dense map of the environment, or a sparse map that contains a set of identifiable landmarks. In this situation, path planning would be performed using the dense map, and the path would be executed in a sensor-based fashion, using feedback control to track the reference path based on sensor information regarding landmark position. Maximum-likelihood estimation techniques are used to model the sensing process as well as to estimate the most likely nominal path that will be followed by the robot during execution of the plan. The proposed approach is potentially a practical way to plan under the specific sorts of uncertainty confronted by a humanoid robot.

The next chapter, presents methods for constructing free paths in dynamic environments. The chapter begins with a comprehensive review of past methods, ranging from modifying sampling-based methods for the dynamic obstacle problem, to methods that were specifically designed for this problem. The thesis proposes to adapt a method reported originally by Leven *et al.*. so that it can be used to plan paths for humanoid robots in dynamic environments. The basic idea of this method is to construct a mapping from voxels in a discretized representation of the workspace to vertices and arcs in a configuration space network built using sampling-based planning methods. When an obstacle intersects a voxel in the workspace, the corresponding nodes and arcs in the configuration space roadmap are marked as invalid. The part of the network that remains comprises the set of valid candidate paths. The specific approach described here extends previous work by imposing a two-level hierarchical structure on the representation of the workspace.

The methods described in Chapters 2 and 3 essentially deal with low-dimensional

problems (e.g., moving a bounding box). The reduction in dimensionality is essential, since the path planning problem confronted in these chapters is complicated by uncertainty and dynamic obstacles, respectively. Chapter 4 addresses the problem of planning the full motion of a humanoid robot (whole-body task planning). The approach presented here is essentially a four-step approach. First, multiple viable goal configurations are generated using a local task solver, and these are used in a classical path planning approach with one initial condition and multiple goals. This classical problem is solved using an RRT-based method. Once a path is found, optimization methods are applied to the goal posture. Finally, classic path optimization algorithms are applied to the solution path and posture optimization.

The fifth chapter describes algorithms for building a representation of the environment using stereo vision as the sensing modality. Such algorithms are necessary components of the autonomous system proposed in the first chapter of the thesis. A simple occupancy-grid based method is proposed, in which each voxel in the grid is assigned a number indicating the probability that it is occupied. The representation is updated during execution based on values received from the sensing system. The sensor model used is a simple Gaussian observation model in which measured distance is assumed to be true distance plus additive Gaussian noise. Sequential Bayes updating is then used to incrementally update occupancy values as new measurements are received.

Finally, chapter 6 provides some details about the overall system architecture, and in particular, about those components of the architecture that have been taken from existing software (and therefore, do not themselves represent contributions of the thesis). Several software systems are described, including GIK, WorldModelGrid3D, HppDynamicObstacle, and GenoM.

Résumé

Le chapitre 1 est pour l'essentiel une brève introduction générale qui donne le contexte générale de la planification et présente l'organisation du document dans son ensemble et quelques uns des points clés retenus : robot humanoïde, environnement non statique, perception par vision artificielle, et représentation de cet environnement par grilles d'occupation.

Dans le chapitre 2, après une revue de littérature bien menée, l'auteur propose de considérer les points de repère de l'environnement dès la phase de planification de chemin afin de rendre plus robuste l'exécution des déplacements en cas d'évolution de l'environnement entre le moment où la planification est menée et celui où le robot se déplace (*évolution* étant entendu comme liée à une amélioration de la connaissance par mise à jour, ou due à un changement de l'environnement lui-même). Le concept est décrit et une formalisation proposée.

Le chapitre 3 s'intéresse en détail à la planification dans le cas d'environnements dynamiques. Les méthodes existantes, nombreuses, sont tout d'abord analysées et bien présentées. Le choix est fait ici de décrire l'environnement comme étant décomposé en *cellules*, regroupant elles-mêmes des *voxels*, éléments atomiques de la représentation. L'environnement étant changeant, l'auteur propose de réévaluer le plan préétabli à partir d'une bonne détection de la zone qui a pu se trouver modifiée dans l'environnement. L'approche est validée expérimentalement en utilisant une des plateformes robotiques du LAAS qui dispose de bonnes capacités de localisation : le manipulateur mobile Jido étant à ce jour plus performant sur ce plan que l'humanoïde HRP2, c'est lui qui a été utilisé. Ces expérimentations donnent des indications concordantes sur l'efficacité de l'approche retenue. Notons également que la planification s'appuie sur une *boite englobante* de l'humanoïde, et non pas sur une représentation plus riche (multi-degré-de-liberté). En revanche, c'est bien de planification pour l'humanoïde considéré dans toute sa complexité qu'il s'agit au chapitre 4 : on s'intéresse ici à tous les degrés de liberté du robot. L'auteur propose des évolutions de méthodes existantes et en particulier sur la manière de tirer profit de la redondance cinématique. L'approche est bien décrite et permet d'inclure une phase d'optimisation de la posture globale du robot. Des exemples illustrent le propos et sont l'occasion de comparaison avec d'autres méthodes.

Le chapitre 5 s'intéresse à la manière de modéliser l'environnement, sachant qu'on s'intéresse ici au cas d'une perception par vision artificielle, et précisément au cas de l'humanoïde, robot d'assurer lui-même cette perception au fur et à mesure de son avancée dans l'environnement. On est donc dans le cadre de la recherche de la *meilleure vue suivante* qui doit permettre d'enrichir au mieux la connaissance qu'a le robot de son environnement. L'approche retenue fait à nouveau appel à la *boite englobante* de l'humanoïde et non à sa représentation complète ; il sera intéressant de voir dans le futur ce que pourrait apporter la prise en compte des degrés de liberté de la tête ou du torse à la résolution de ce problème.

Le chapitre 6 décrit la phase d'intégration de tous ces travaux sur la plateforme HRP2 du LAAS-CNRS, partie importante de tout travail de roboticien.

Acknowledgements

Early in the process of completing this project, it became quite clear to me that a researcher cannot complete a Ph.D. thesis alone. Although the list of individuals I wish to thank extends beyond the limits of this format, I would like to thank the following persons for their dedication, prayers, and support:

First of all, I would like to thank Florent Lamiraux, the best advisor I could have wished for, and without whose efforts for giving clear and simple explanations, his knowledge, criticism and humor it would have been a much more difficult, if not impossible, task. His advises even from long distance helped me a lot during these three years.

I would like to thank Jean-Paul Laumond who supported me during my research provided the facilities for conducting this thesis. I will always be thankful for his wisdom, knowledge, and deep concern.

I wish to thank Eiichi Yoshida for his help, advise and enthusiasm. My thanks to Anthony Mallet who helped me a lot for integrating my work on HRP2 and always taking the time to answer my questions.

I wish to express my sincere gratitude to Olivier Stasse, for his valuable suggestions for the improvement of this work.

Thanks to the people with whom I had the chance to collaborate during these years, Mathieu Poirier, Oussama Kanoun, Tran Minh Tuan, Jesse Himmelstein, David Flavigne, Manish Sreenivasa, Brice Burger, for always taking the time to answer my questions and give good ideas. And also thanks to the other Gepettistes, Tan Viet Anh Truong, Francisco Montecillo ...

A very special thanks to Sébastien Dalibard for a close collaboration on a research project which continues after the thesis.

I would like to thank the various people who have provided their valuable assistance, a helpful or encouraging word during these years.

Thanks to my office colleagues and friends for putting up with me all this time. I am indebted to my many colleagues for providing a stimulating and fun

environment, as well as for all their help and friendship and all those who I might be forgetting.

My deepest thanks to Sepanta Sekhavat, for encouraging me to continue my education in Gepetto group in LAAS.

It is difficult to overstate my gratitude to my family, specially to my parents, my sister Atefeh and my brother Amidreza, if I am here it is certainly because of them. Thanks.

Contents

. . . . iv
. . . . vi

Contents **i**

1 General Introduction **1**
1.1 Robotics 1
1.2 Humanoid robots 2
1.3 Motivation 3
1.4 Contribution 4
1.5 Document organization 5
1.6 Publications associated to this work 6

2 Motion planning in stochastic maps **7**
2.1 Introduction 7
2.2 Motion planning concepts 11
2.3 Motion planning in stochastic maps 13
2.4 Definitions 14
2.5 Description of the method 17
    2.5.1 Input data 17
    2.5.2 Localization 17
    2.5.3 Feedback control law 18
    2.5.4 Path planing 18
    2.5.5 Distance to obstacles 19
    2.5.6 Calculating the probability of collision 20
2.6 Example 22
2.7 Conclusion 23

3 Motion planning in changing environments **25**

3.1 Introduction 25
3.2 General idea of the planner 37
3.3 Work space cell-decomposition 39
3.4 Updating cells 41
3.5 Constructing roadmap 41
3.6 Updating roadmap 42
3.7 Examples 42
  3.7.1 Experiment 1 44
  3.7.2 Experiment 2 46
  3.7.3 Experiment 3 46
3.8 Conclusion 47

4 Whole-body task planning 51
4.1 Introduction 51
  4.1.1 Whole-Body task motion planning 51
  4.1.2 Randomized motion planning 52
  4.1.3 Contribution 52
4.2 Preliminaries 53
  4.2.1 RRT-Connect 53
  4.2.2 Local path planning under task constraints 54
4.3 Randomized task planning for a humanoid robot 55
  4.3.1 Task constrained extension 55
  4.3.2 Goal configuration generation 57
  4.3.3 Posture optimization 58
  4.3.4 General architecture 59
4.4 Work space analysis 59
4.5 Examples and comparison with previous methods 61
  4.5.1 Dual support ”Table” scenario 64
  4.5.2 Single support “Torus” scenario 64
  4.5.3 When to deal with obstacle avoidance 65
4.6 Conclusion 66

5 Environment modeling 67
5.1 Introduction 67
5.2 Environment modeling 69
  5.2.1 Occupancy grid map. 69
  5.2.2 Sensor model 70
  5.2.3 Updating 3D model 70
5.3 Next best view 72

5.3.1 Constraints on the next best view 73
5.3.2 Evaluating the next best view 73
5.3.3 Computing the next best view 73
5.4 Conclusion 75

6 Integration 77
6.1 Introduction 77
6.2 Integration 78
6.2.1 Whole body motion 80
6.2.2 vision 80
6.2.3 Navigation 81
6.2.4 Data flows 82
6.3 Experimental results 82
6.4 Conclusion 83

7 Conclusion 85

Bibliography 87

List of Figures 95

List of Tables 100

Chapter 1

General Introduction

1.1 Robotics

Robotics is a branch of engineering that involves the conception, design, manufacture and operation of robots. This field overlaps with mechanics, electronics, computer science, artificial intelligence, mechatronics, nanotechnology and bioengineering.

The notion of robots can be traced back to medieval times. Although people of that era did not have a term to describe what we would eventually call a robot they were nevertheless imagining mechanisms that could perform human-like tasks. While the concept of a robot has been around for a very long time, it was not until the 1940s that the modern day robot was born, with the arrival of computers.

The robot really became a popular concept during the late 1950s and early 1960s. With the automotive industry in full expansion at that time, industrial robots were employed to help factory operators. Industrial robots do not have the imaginative, human-like appearance that we have been dreaming of throughout the ages. They are computer-controlled manipulators, like arms and hands, which can weld or spray paint cars as they roll down an assembly line.

Today, commercial and industrial robots are in widespread use performing jobs more cheaply or with greater accuracy and reliability than humans. They are employed for jobs which are too dirty, dangerous, or dull to be suitable for humans.

The structure of a robot is mostly mechanical and can be called a kinematic chain (its functionality being similar to the skeleton of the human body). The chain is formed of links (its bones), actuators (its muscles), and joints which can allow one or more degrees of freedom.

The mechanical structure of a robot must be controlled to perform tasks. The control of a robot involves three distinct phases: perception, processing, and action.

1.2 Humanoid robots

A humanoid robot is a robot with its overall appearance based on that of the human body, allowing interaction with made-for-human tools or environments. In general humanoid robots have a torso with a head, two arms and two legs, although some forms of humanoid robots may model only part of the body, for example, from the waist up.

The creation of humanoid robots has been motivated by the idea of having a device capable of operating in environments made by and for humans with minimal change to those environments.

These machines are expected to perform autonomously most of the functions a person is capable of. These include climbing stairs, reaching for objects, etc.

Like other mechanical robots, humanoids refer to the following basic components too: Sensing, Actuating, Planning and Control. Since they try to simulate the human structure and behaviors and they are autonomous systems, most of the times they are more complex than other kinds of robots.

Humanoid robots are used as a research tool in several scientific areas. Researchers need to understand the human body structure and behaviors and the way that it interact with its environment.

Besides the research, humanoid robots are being developed to perform human tasks like personal assistance, where they should be able to assist the sick and elderly, and dirty or dangerous jobs.

Based on these requirements, various humanoid platforms have been introduced to research labs during the last years.

Kawada industries have developed a series of humanoid robots in the Humanoid Robotics Project. This project is mainly supported by the National Institute of Advanced Industrial Science and Technology (AIST) in Japan. HRP2 is a successful platform which is developed in this project. HRP2 family is made of 15 clones which are mostly used in the research lab. HRP3 family is introduced to the research lab later. Apart from some structural different with its ancestor, its main mechanical and structural components are designed to prevent the penetration of dust or spray.

The last HRP platform is HRP4 which was designed to look like an average

Japanese woman. It was presented to public in 2009. Figure 1.1 illustrates the three platforms of the Humanoid Robotics Project.

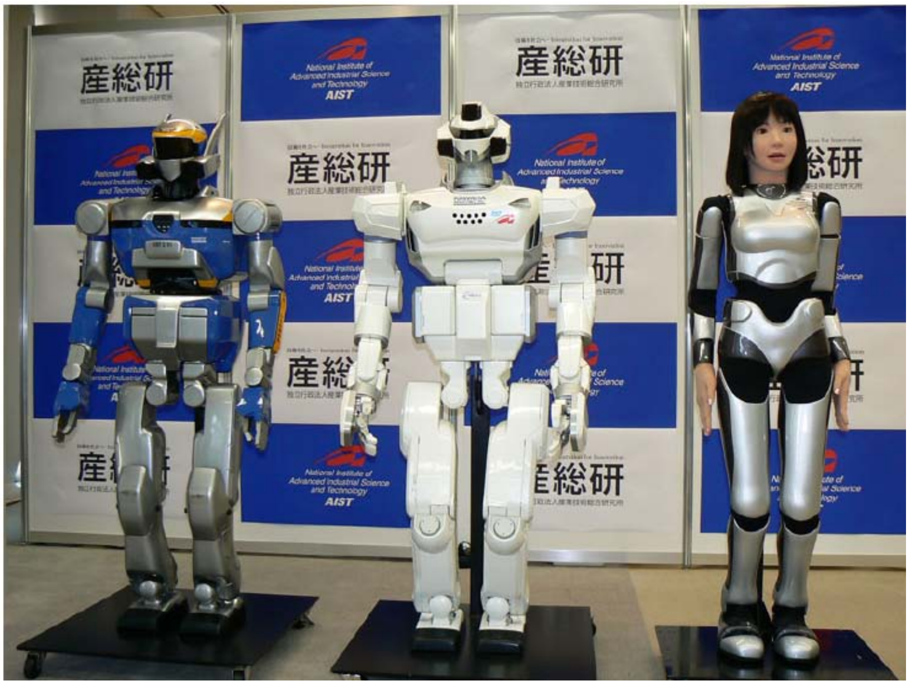


Figure 1.1: HRP platforms. Right to left: HRP2, HRP3 and the last humanoid HRP4.

1.3 Motivation

During the recent years, autonomy in robots has been in the centre of attention. An autonomous robot must be able to interact with workspace both in modelling environments and planning collision free path.

Therefore, to have an autonomous robot in environment, the robot should be able to capture information from its environment by sensors, analyze sensor data, generate a model of the workspace and then plan a collision free path in this model. As it was explained in the previous section, the main phases are as follows:

1. **Perception** is capturing the required information and data from environments. This information varies from the position of other objects to their colors. Various types of sensors provide the required information for modeling environments. Cameras, sonic and ultrasonic sensors, laser sensors capture information from the environment.

2. **Environment modeling** is the process of developing a mathematical representation of environments. Based on the captured information from environments, a model should be generated to be used later in other algorithms such as motion planning.

3. **Motion planning** is to produce a continuous path that connects a start configuration $\mathbf{q}_S$ and a goal configuration $\mathbf{q}_G$, while avoiding collision with known obstacles. The robot and obstacle geometry is described in a 2D or 3D workspace, while the path is represented in configuration space.

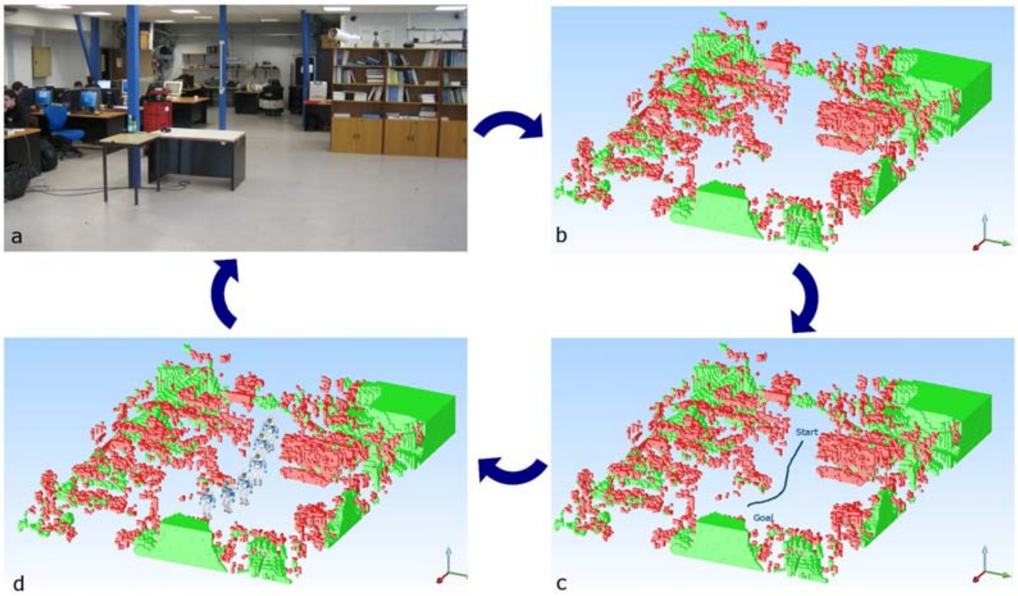
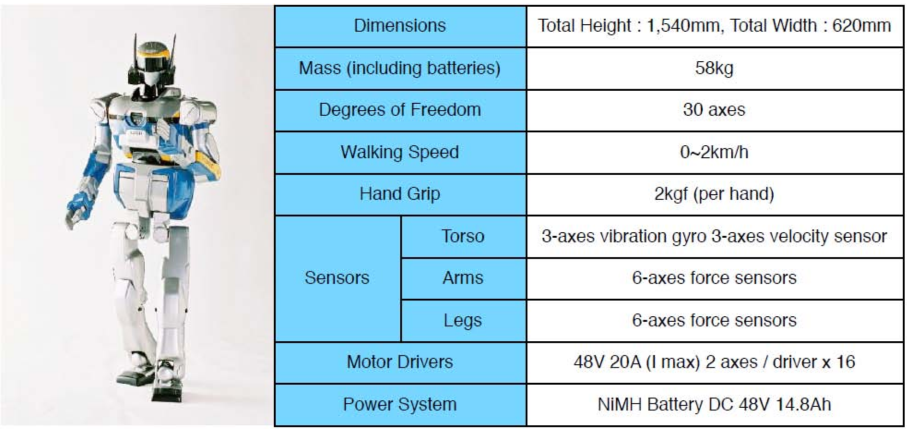


Figure 1.2: Autonomous robot in an environment: a- Perception, b- Modeling, c- Motion Planning, d- Execution.

Our motivation is having an autonomous humanoid robot. Therefore, we are going to link the required methods for perception, environment modeling and motion planning. Based on our access to the HRP2 platform in LAAS-CNRS, we integrate these methods on the robot. It enables the robot to interact autonomously with its environment.

1.4 Contribution

The context of this work is vision-based motion planning for humanoid robots in unknown environments. We present an efficient combination of on-line 3D environment modeling and motion planning methods for humanoid robots for



Dimensions		Total Height : 1,540mm, Total Width : 620mm
Mass (including batteries)		58kg
Degrees of Freedom		30 axes
Walking Speed		0~2km/h
Hand Grip		2kgf (per hand)
Sensors	Torso	3-axes vibration gyro 3-axes velocity sensor
	Arms	6-axes force sensors
	Legs	6-axes force sensors
Motor Drivers		48V 20A (I max) 2 axes / driver x 16
Power System		NiMH Battery DC 48V 14.8Ah

Figure 1.3: The humanoid robot HRP2

navigation in non-static environment. Our work also enables humanoid robots to plan whole-body collision free path in a 3D model.

For modeling environments, we rely on 3D occupancy grid method. The robot is supposed to capture the required information by two on board cameras. At the same time, by analyzing the captured images, HRP2 uses stereo vision data to update the occupancy grid model of the environment.

As the robot navigates in the environment, it receives updated information through its cameras and refreshes the 3D model. Conventionally, the model can be composed of up to thousands of voxels. Based on any new information from environment, the model changes.

As the model is not static, we deal with motion planning problem in changing environments to accelerate path planning for navigation. Our approach is a descendant of an existing method which deals with changing environments.

In order to plan whole body paths, we propose to use local Jacobian based method within randomized motion planning. This approach enables the robot to plan stable paths to fulfill the required tasks.

1.5 Document organization

This document is divided in 6 chapters in the following way:

In chapter 2, a new framework for planning motion in probabilistic environments is proposed. The method is developed and at the end, a simple example illustrates the functionality of the method in an uncertain map.

Chapter 3 presents our work on motion planning in changing environments. The algorithms are explained and finally, the experimental results are used to evaluate the method.

Chapter 4 describes the idea of whole body task planning in 3D workspace. The algorithms are used for HRP2 model to plan a path for user defined tasks. Moreover, in some examples, the method is compared with existing approaches.

Chapter 5 is a brief review of a 3D occupancy grid approach which is implemented on the robot to generate the 3D model of its environment. Moreover, we present our algorithms for finding the next best view for exploring unknown environments.

The corresponding experimental tests on the robot are described in chapter 6. We conducted experiments to evaluate our approach in a real environment with HRP2. The robot is supposed to navigate in a 3D occupancy grid model generated by vision. A brief description of all the modules used in the experiments can be found in this chapter. Finally, chapter 7 presents the conclusion of our work.

1.6 Publications associated to this work

1. S. Dalibard, A. Nakhaei, F. Lamiraux, J.P. Laumond. Whole-Body Task Planning for a Humanoid Robot: a Way to Integrate Collision Avoidance. In *9th IEEE International Conference on Humanoids Robots (HUMANOIDS 2009)*, Paris , France, 2009, (Submitted).

2. A. Nakhaei, F. Lamiraux. Motion planning for humanoid robots in environments modeled by vision. In *8th IEEE International Conference on Humanoids Robots (HUMANOIDS 2008)*, Daejon, Korea , 2008.

3. A. Nakhaei, F. Lamiraux. A framework for planning motions in stochastic maps, *10th International Conference on Control, Automation, Robotics and Vision (ICARCV 2008)*, Hanoi, Vietnam, 2008.

4. J. Himmelstein, G. Ginioux , E. Ferre , A. Nakhaei , F. Lamiraux , J.P. Laumond. Efficient architecture for collision detection between heterogeneous data structures. Application for vision-guided robots. *10th International Conference on Control, Automation, Robotics and Vision (ICARCV 2008)*, Hanoi, Vietnam, 2008.

Chapter 2

Motion planning in stochastic maps

2.1 Introduction

During the last decades, many researchers became interested in solving the problem of moving an object between two position in the environment by avoiding collision with obstacles in the workspace. A famous example is the Piano Mover's which is stated by Schwartz and Sharir [58]. This pioneer work has led to the development of the Motion Planning research field described extensively in [11, 36, 37, 39]. The problem can be stated as follows: Given an environment with obstacles and a piano, is it possible to move the piano from one position and orientation, to another without colliding with the obstacles. Figure 2.2 shows an instance of the Piano Mover's Problem.

The concept of configuration space was introduced in 80's by Lozano-Perez [46]. Configuration space $\mathcal{C}$ is the set of all the possible configurations that a mechanism can attain. This definition has been a key concept in motion planning for it allows to translate the problem of moving a body in a space $\mathcal{W} \subset \mathbb{R}^2$ or $\mathbb{R}^3$ into the problem of moving a point in another space.

In this context, a motion planning problem is re-stated as the problem of computing $\mathcal{C}_{obstacle}$ and finding a continuous curve or $Path : [0\ 1] \rightarrow \mathcal{C}$ that connects an initial configuration $Path(0) = \mathbf{q}_{init}$ to a final configuration $Path(1) = \mathbf{q}_{end}$. The obstacle region $\mathcal{C}_{obstacle}$ in the configuration space corresponds to the configurations which are forbidden.

Based on this translation, several planners which construct an explicit and exact representation of $\mathcal{C}_{obstacle}$ were proposed such as [25, 9]. Although these ap-

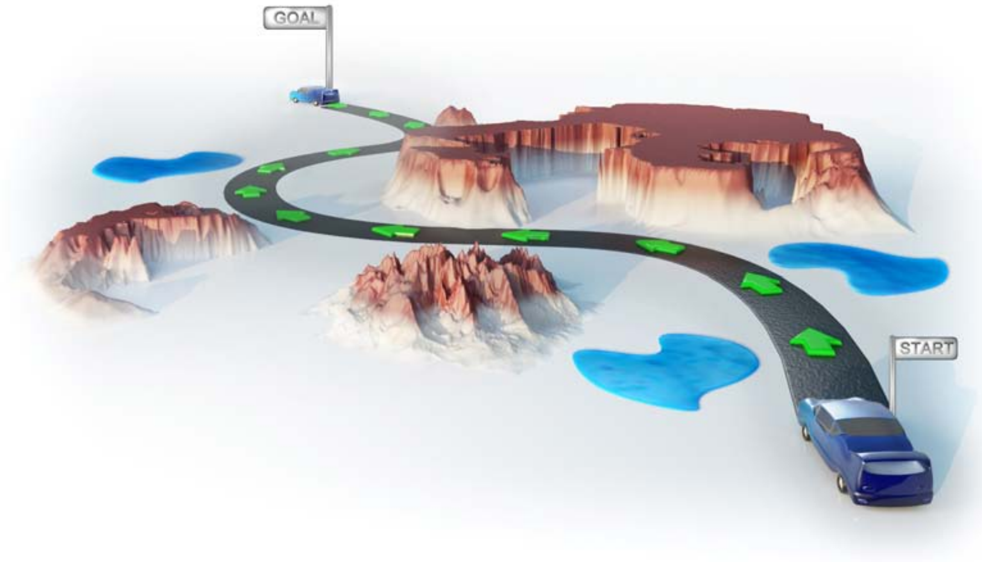


Figure 2.1: Motion planning concept: Finding a collision free path between an initial and final configurations.

proaches have desirable properties such as completeness, in some cases, the combinatorial complexity makes even simple problems computationally intractable.

In order to provide a practical resolution for the planning problem, sampling-based methods have been developed over the last fifteen years. These methods have proven their efficiency for solving difficult problems in high-dimensional spaces.

The main idea of these methods is to use the topology of $\mathcal{C}_{free}$ in a roadmap $\mathcal{R}$ without computing explicitly $\mathcal{C}_{obstacle}$. This roadmap is used to find a collision-free path that connects $\mathbf{q}_{init}$ to $\mathbf{q}_{end}$. A roadmap can be obtained mainly by using two types of algorithms: sampling and diffusion. These methods are said to be probabilistic complete, which means that the probability of finding a solution, if one exists, converges to 1 as the computing time tends to infinity.

These two kinds of sampling methods have been investigated with success:

1. The sampling approach, first introduced in [32] as probabilistic roadmaps (PRM), consists in computing a graph, or a roadmap, whose nodes are collision free configurations, sampled at random in the free space and whose edges reflect the existence of a collision free elementary path between two configurations. It aims at capturing the topology of the free space in a learning phase in order to handle multiple queries in a solving phase.

2. The diffusion approach, introduced in both [23] and [35], which includes

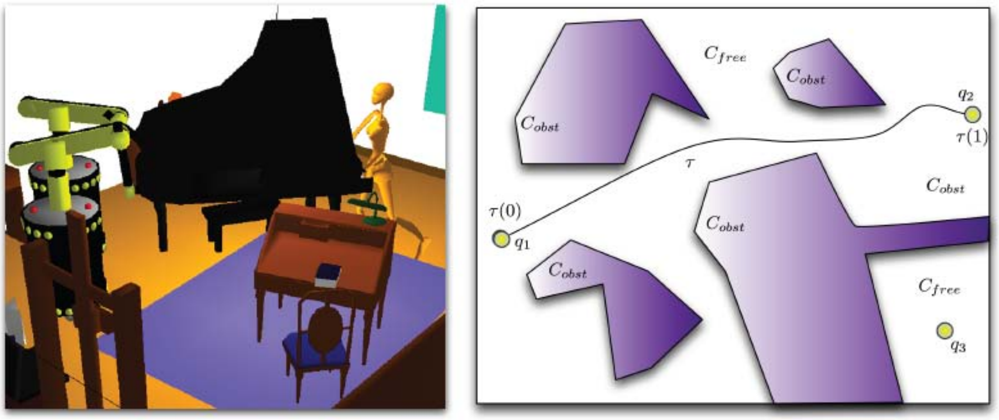


Figure 2.2: From [17]: a) A Piano Mover's Problem. Two robots and a virtual character cooperate to transport a piano in a workspace $\mathcal{W} \subset \mathbb{R}^3$. b) The configuration space is divided in $\mathcal{C}_{free}$ (light areas) and $\mathcal{C}_{obstacle}$ (dark areas). A path can be found between $\mathbf{q}_1$ and $\mathbf{q}_2$ because they lie in the same connected component of $\mathcal{C}_{free}$, which is not the case for $\mathbf{q}_3$.

RRT planners, consists in solving single queries by growing a tree rooted at the start configuration towards the goal configuration to be reached.

The Probabilistic Roadmap (PRM) family of planners first shoot random configurations and add the collision free configurations to a roadmap.The initial and goal configurations are also added to the roadmap. Then, a local search algorithm attempts to connect nearby pairs of configurations (nodes), and if a free path is found, an edge is added between the two nodes representing the local planners path between the two configurations. This forms a graph of $\mathcal{C}_{free}$ which can be searched using A*. Classical PRM can answer many queries by attempting to connect the initial and goal positions to an existing graph, making it a multi-shot planning algorithm. Figure 2.3 illustrates how these types of planner explore the free zone of the configuration space to connect the initial configuration to the goal configuration.

The second approach of the family of randomized planners is based on the Rapidly Exploring Random Tree (RRT). Configurations are sampled randomly from configuration space, and the nearest node in the current roadmap is extended a fixed distance toward the sampled point. The new edge and node are added only if the local path is collision free. The sampling and extension steps are repeated to grow a search tree from the initial position until it reaches a goal state. Figure 2.4 illustrates how RRT approaches deal with a motion planning problem.

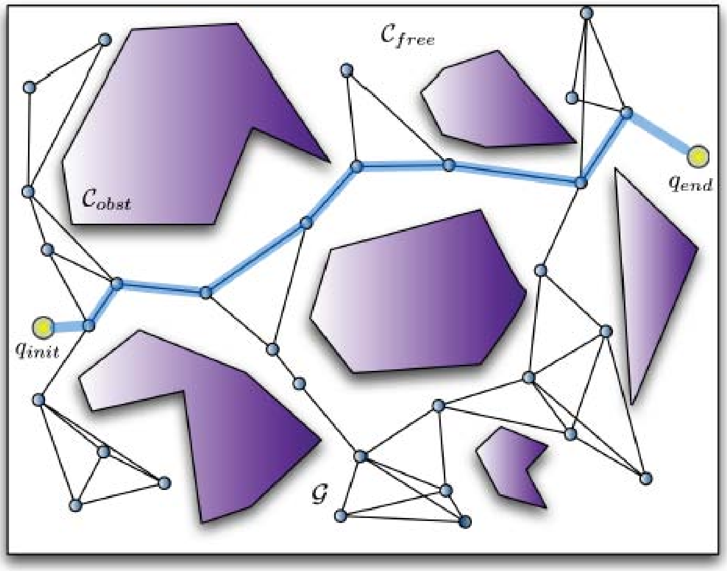


Figure 2.3: From [17]: In the PRM algorithm a roadmap $\mathcal{R}$ is build around the obstacles by drawing random collision-free configurations (nodes) and connecting them to their nearest neighbors using feasible robot motions (edges). The query is solved by connecting $\mathbf{q}_{init}$ and $\mathbf{q}_{end}$ to the graph and then finding the shortest path on it (thick lines)

These methods have been proven to be efficient and suitable for a large class of motion planning problems.

In the following years, various approaches have been proposed based on the sampling method. For example, in 2000, Simeon *et al.* [60] proposed Visibility PRM. Figure 2.5 illustrates how visibility PRM deals with motion planning problems.

The main idea of this approach is, as in the basic PRM approach, to capture the topology of $\mathcal{C}_{free}$ into a roadmap. The main difference is that not every collision-free drawn node is integrated into roadmap, but only the nodes with certain visibility or connection characteristics are added to the roadmap. The generated roadmap using the Visibility PRM algorithm is more compact that the one obtained using PRM alone.

Most of the recent planners are good enough to solve motion planning problems. However, they need a complete and accurate model of environments. Therefore, how can we deal with the uncertainties in the model of environments to reduce the probabilities of collision?

Considering uncertainties in the model, we reformulate the path planning problem in a stochastic map and then propose a way to modify classical path

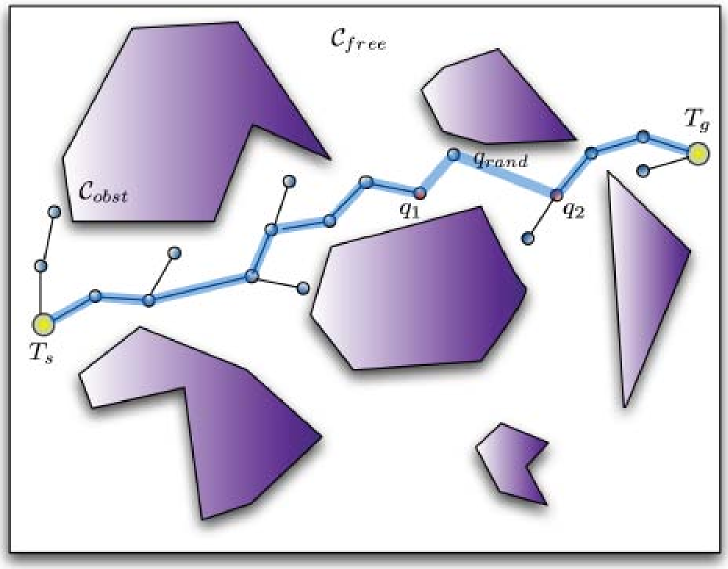


Figure 2.4: From [17]: With the RRT algorithm, two trees rooted on $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ can be grown simultaneously. Each tree is extended until leaf configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ from each tree can be connected by a randomly drawn configuration $\mathbf{q}_{rand}$.

planning methods in order to fit into this new framework. In our framework, sensors and landmarks should be taken into account. The core computations lie in the evaluation of the probability of robot collision with its environment.

2.2 Motion planning concepts

In this section, we will present and define several important motion planning concepts:

Workspace $\mathcal{W}$ is the robot environment. All the movements of robots and the obstacles are realized in workspace. Generally, robots and obstacle geometry is described in a 2D or 3D workspace.

Configuration Space $\mathcal{C}$ is the set of all possible configurations. A configuration $\mathbf{q}$ describes the robot pose. For example:

1. If the robot is a single point (zero-sized) moving in a 2-dimensional plane (the workspace), $\mathcal{C}$ is a plane, and a configuration can be represented using two parameters (x, y).

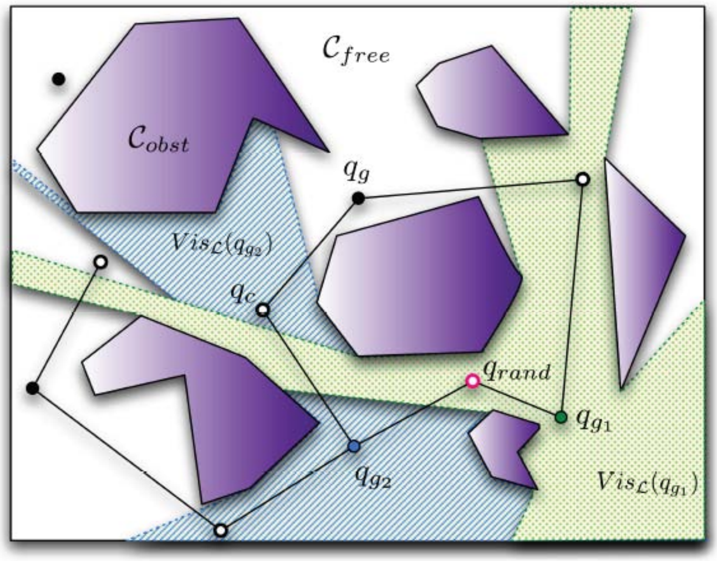


Figure 2.5: From [17]: The Visibility-PRM algorithm produces a compact roadmap around obstacles with only a few nodes: guards $\mathbf{q}_g$ (dark circles) and connectors $\mathbf{q}_c$ (in white). A new randomly drawn configuration $\mathbf{q}_{rand}$ is added as a guard if it covers a previously unseen visibility domain with a given L or as a connector if it serves as liaison between at least two connected components.

2. If the robot is a $2D$ shape that can translate and rotate, the workspace is still 2-dimensional. However, the elements of $\mathcal{C}$ can be represented using 3 parameters (x, y, θ).

3. If the robot is a solid $3D$ shape that can translate and rotate, the workspace is 3-dimensional, but a configuration requires 6 parameters: (x, y, z) for translation, and Euler angles (α, β, γ).

4. If the robot is a manipulator with n joints, the workspace is 3-dimensional and configuration space $\mathcal{C}$ is n-dimensional.

Free Space $\mathcal{C}_{free}$ is a set of configurations that avoids collision with obstacles and self collision. The complement of $\mathcal{C}_{free}$ in $\mathcal{C}$ is called the forbidden region $\mathcal{C}_{obstacle}$.

Often, it is prohibitively difficult to explicitly compute the shape of $\mathcal{C}_{free}$. However, testing whether a given configuration is in $\mathcal{C}_{free}$ is efficient. First, forward kinematics determines the position of the robot's geometry, and collision detection tests if the robot's geometry collides with the environment's geometry.

Collision detection involves algorithms for collision checking in physical simulations, video games and motion planner etc. These algorithms detect collisions of two or more geometrical objects.

Path is a function as $Path : [0\ 1] \rightarrow \mathcal{C}$ that connects two configurations. Each movement of a robot in $\mathcal{W}$ corresponds a movement in $\mathcal{C}$.

Admissible Path is a path whose alongside configurations are collision free.

2.3 Motion planning in stochastic maps

To perform a task autonomously, a robot should be able to build a map of its environment and to plan motions in this map. In this respect, motion planning and SLAM (simultaneous localization and map building) have been active research fields for the past decades. However, these fields have evolved separately. Many robots are able to navigate in indoor 2D maps built by sensors [27, 64, 1].

However, for robots with high dimensional configuration spaces like robotic arms or humanoid robots, performing autonomous motion is still a very difficult problem. Several reasons explain this state of fact.

1. 3D SLAM techniques usually build sparse maps of landmarks containing no obstacles [14, 18, 41, 42] and planning motions for high-dimensional robots requires a 3D-map of obstacles.

2. Motion planning algorithms for high-dimensional systems are based on the assumption that the map of the world is exact and static. These structures are mainly hierarchies of bounding boxes around obstacles [44] and roadmaps of local paths in the free configuration space [32, 35].

Therefore, these algorithms do not allow robust operations for high-dimensional robots interacting with real world with many uncertainties. In fact, we need to deal with uncertainties in models of the environment evolving along time.

Our objective is to propose preliminary ideas that enables path planning techniques for high dimensional systems to deal with models built by sensors. In fact, the concept of autonomy means generating the model of the environment with sensors and implements the techniques of motion planning in the generated model. As such models are not accurate, we are going to propose a new framework for motion planning to deal with these uncertainties.

The contribution of this chapter is to propose a new framework to plan motions in uncertain environments represented by stochastic maps as built by SLAM techniques. We do not address yet the non static aspect of the environment.

We assume that we have a stochastic Gaussian map composed of landmarks the position of which is represented by a mean vector and a covariance matrix. To make the reasoning simpler, we assume that the landmarks are also obstacles. However, this assumption could be easily relaxed.

Our work shares ideas with previous work in this area [48] but follows a different approach. In [48], Missiuro and Roy address sensor uncertainty in the context of complete motion planning. The approach adaptively samples configuration space using a distribution based upon the certainty of the sampled configurations. Paths are found in the resulting roadmap using A* search and an uncertainty heuristic. The main difference comes from the localization notion that is taken into account in the path planning step. The idea is to minimize the probability of the collision as the robot follows the planned path. Our approach significantly increases the complexity of the problem but we strongly believe that landmarks should be taken into account right at the path planning step.

We would like to emphasize that the contribution of the chapter is mainly conceptual. This work is very preliminary and we do not have impressive experimental results yet. However, we truly believe this preliminary work is worth reporting in a chapter.

2.4 Definitions

Suppose that a model of an environment and desired initial and goal configurations are given to a planner. The result of this motion planning problem is a path and the robot is supposed to execute this reference path in a real environment. During navigation in the environment, the robot usually:

1. Performs localization on landmarks of the map.
2. Tries to make the result of localization converge toward the configuration along the reference path.

To make the reasoning simpler, let us assume that the closed loop control law of the robot is powerful enough to maintain equality between the result of the localization and the reference configuration:

$$\mathbf{q}^{loc}(\mathbf{q}, M) = \mathbf{q}^{ref} \tag{2.1}$$

where $\mathbf{q}^{loc}$ is the result of localization that depends on the actual position of the robot $\mathbf{q}$ and the actual position of the landmarks M (the map). This equation therefore defines a relation between $\mathbf{q}$, $\mathbf{q}^{ref}$ and M that we can inverse to get an equality of the form:

$$\mathbf{q} = g(\mathbf{q}^{ref}, M) \tag{2.2}$$

This equation can be understood as: The executed path is the result of both the reference path expressed in a landmark-based frame and the actual position of the landmarks. This statement is very intuitive. As a simple example, a planner plans a path for a robot and at the result, the robot is supposed to follow a path alongside a wall. If the robot performs localization on the wall, the actual path depends on the actual position of the wall also.

If the map does not correspond exactly to the actual position of the landmarks, the actual path will be different from the reference path. As the map is represented by random variables, the path followed by the robot becomes a random variable. In result, the admissibility of the path becomes a random event. This idea is the basis of our work.

We now give precise definitions for the different notions we will use in our reasoning.

Stochastic map: A stochastic map is the random vector M, defining the position of several landmarks $\{\mathbf{l}_1, ..., \mathbf{l}_p\}$ in environments. Each landmark is represented by a vector $\mathbf{l}_i$ of dimension m_i, $i \in \{1, ..., p\}$.

$$M = \begin{pmatrix} \mathbf{l}_1 \\ \vdots \\ \mathbf{l}_p \end{pmatrix} \in \mathbb{R}^m \tag{2.3}$$

where $m = \sum_{i=1}^{p} m_i$.

Sensor: A sensor maps the positions of landmarks (the relative position of landmarks) to the image space of the sensor:

$$im_i = im_i(\mathbf{q}, \mathbf{l}_i) \in \mathbb{R}^{p_i}, \;\; i \in \{1, ..., l\} \tag{2.4}$$

where im_i is the image of $\mathbf{l}_i$ in the associated sensor. For instance, considering a camera as the sensor, a 3D-point ($\mathbb{R}^3$) is projected to the image space as a point of dimension 2. Also, a vertical plane is projected in the image plane of an horizontal laser range scanner as a straight-line.

If k landmarks are perceived simultaneously, we can put the above equations in a vector as follow:

$$IM = f(\mathbf{q}, M), IM \in \mathbb{R}^r, \mathbf{q} \in \mathcal{C} \tag{2.5}$$

where,

$$IM = \begin{pmatrix} im_1 \\ \vdots \\ im_k \end{pmatrix} \quad r = \sum_{i=1}^{k} p_i. \tag{2.6}$$

Localization: Localization consists in computing the maximum likelihood configuration, given a perception and a stochastic map:

$$\begin{aligned} \mathbf{q}^{loc} &= f_1(IM, M) && (2.7) \\ &= f_2(\mathbf{q}, M) && (2.8) \end{aligned}$$

after substituting (2.5).

Closed-loop path tracking: To track a reference path, a mobile robot usually implements a closed loop control feedback. The effect of this control process is to make the localization $\mathbf{q}^{loc}$ converge toward the reference path.

For simplification purposes, we assume that the closed-loop control law is good enough to maintain equality between the reference configuration and the localization:

$$\mathbf{q}^{loc} = \mathbf{q}^{ref} \tag{2.9}$$

Using (2.7), this equality defines a relation between the $\mathbf{q}$, $\mathbf{q}^{ref}$ and the map:

$$\mathbf{q} = f_3(\mathbf{q}^{ref}, M) \tag{2.10}$$

As a conclusion, given a reference path $\mathbf{q}^{ref}(s), s \in [0,1]$ planned in a map of the environment, the path $\mathbf{q}(s), s \in [0,1]$ actually followed by the robot depends on the stochastic map M. This path is therefore a random variable and its collision is a random event.

In the classical formulation of path planning, a path is said to be admissible (or collision-free) if any configuration along this path is collision-free.

In our framework,

definition: *A reference path* $\boldsymbol{q}^{ref}(s), s \in [0,1]$ *is said to be admissible if the probability of collision of the path* $\boldsymbol{q}(s), s \in [0,1]$ *actually followed by the robot is less than a tolerance* $\varepsilon > 0$.

The goal of our work is to propose a method for planning admissible paths according to the above definition.

2.5 Description of the method

In this section, we develop the computations necessary to our method and we introduce some approximations or simplifications.

2.5.1 Input data

The data given as input to our planner are:

1. A Gaussian stochastic map built beforehand by SLAM techniques represented by a mean vector $\bar{M}$ and a covariance matrix Σ_M.
2. An initial configuration $\mathbf{q}_{init}$.
3. A goal configuration $\mathbf{q}_{goal}$.

2.5.2 Localization

Assuming that the actual configuration of the robot keeps close to the reference configuration, we perform localization by linearizing the relation (2.5) between the actual configuration and images seen in the sensors about the reference configuration:

$$IM^{ref} + \frac{\partial f}{\partial \mathbf{q}}(\mathbf{q} - \mathbf{q}^{ref}) + \frac{\partial f}{\partial M}(M - \bar{M}) = IM \tag{2.11}$$

IM^{ref} is the expected image, that is the image seen from reference configuration when $M = \bar{M}$ and IM is the image actually perceived.

To find the best estimator $\mathbf{q}^{loc}$ of $\mathbf{q}$, we use Gauss-Markov theorem as follows.

Gauss Markov theorem: *Let X and Y be Gaussian random vectors such that*

$$Y = H.X + b \tag{2.12}$$

where H is a matrix and b a centered random vector of covariance matrix identity. The optimal estimator $\hat{X}$ of X given an observation of Y is given by:

$$\hat{X} = (H^T H)^{-1} H^T Y \tag{2.13}$$

We consider $\frac{\partial f}{\partial M}(M - \bar{M})$ as a Gaussian noise in (2.11). The covariance matrix Σ_b of this noise is:

$$\Sigma_b = \frac{\partial f}{\partial M} \Sigma_M \frac{\partial f^T}{\partial M} \tag{2.14}$$

Based on Gauss-Markov theorem, the best estimator of **q** is:

$$\mathbf{q}^{loc} = \mathbf{q}^{ref} + (\frac{\partial f^T}{\partial \mathbf{q}} \Sigma_b^{-1} \frac{\partial f}{\partial \mathbf{q}})^{-1} \frac{\partial f^T}{\partial \mathbf{q}} \Sigma_b^{-1} (IM - IM^{ref}) \tag{2.15}$$

2.5.3 Feedback control law

As explained earlier, the robot is subjected to a closed loop control law which tends to make $\mathbf{q}^{loc}$ converge toward $\mathbf{q}^{ref}$. As an approximation, we assume that these two configurations are constantly equal, condition equivalent to:

$$\frac{\partial f^T}{\partial \mathbf{q}} \Sigma_b^{-1} (IM - IM^{ref}) = 0 \tag{2.16}$$

On the other hand, according to (2.11), the image IM in the sensor depends on the configuration of the robot and the map. Therefore, substituting this later expression into (2.16), the result will be:

$$\frac{\partial f^T}{\partial \mathbf{q}} \Sigma_b^{-1} (\frac{\partial f}{\partial \mathbf{q}} (\mathbf{q} - \mathbf{q}^{ref}) + \frac{\partial f}{\partial M} (M - \bar{M})) = 0 \tag{2.17}$$

which can be inverted as:

$$\mathbf{q} = \mathbf{q}^{ref} - (\frac{\partial f^T}{\partial \mathbf{q}} \Sigma_b^{-1} (\frac{\partial f}{\partial \mathbf{q}})^{-1} \frac{\partial f^T}{\partial \mathbf{q}} \Sigma_b^{-1} \frac{\partial f}{\partial M} (M - \bar{M}) \tag{2.18}$$

This latter equation can be understood in two ways:

1. Given a reference configuration $\mathbf{q}^{ref}$ and the actual position of the landmark M, we can predict the position to which the robot will converge when trying to reach $\mathbf{q}^{ref}$ in the environment.

2. Given a reference configuration $\mathbf{q}^{ref}$ and the stochastic map, we can express the position of the robot that converges to a random variable depending on the map.

2.5.4 Path planing

In section 2.4, the definition of admissible paths in a stochastic map was presented. In this section, we are going to explain how we plan admissible paths in a meaning close to this definition. To make computations simpler, we indeed consider that a path $\mathbf{q}^{ref}(s), s \in [0, 1]$ is admissible if and only if

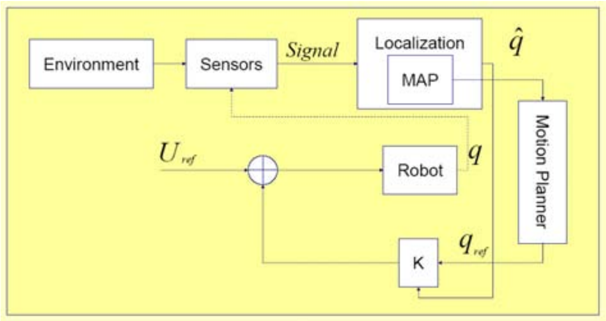


Figure 2.6: A brief diagram illustrates the planning date flow.

$$\forall s \in [0,1], P(\mathbf{q}^{ref}(s) \text{ is in collision}) < \varepsilon$$

We then modify existing random motion planning methods such as RRT or PRM. In the classical path planning frameworks, roadmap based path planning methods pick random configurations, test collision and then build paths between these configurations. We replace the collision detection algorithm by the algorithm of calculating the collision probability for random configurations as defined by (2.18).

To approximate the collision probability of a configuration with the uncertain obstacles of the map, we approximate the vector of random distances of the robot to the obstacles by a Gaussian vector and compute the probability that one of these distances is less than 0.

A brief description of our method is illustrated in figure 2.6.

2.5.5 Distance to obstacles

As the robot configuration and the positions of the obstacles (or landmark) are considered as Gaussian variables, the distance between the robot and each obstacle is also approximated by a Gaussian random variable. We build a random vector $d_i(\mathbf{q}, \mathbf{l}_i)$ by gathering these distances as follows:

$$d(\mathbf{q}, M) = \begin{pmatrix} d(\mathbf{q}, \mathbf{l}_1) \\ d(\mathbf{q}, \mathbf{l}_2) \\ \vdots \\ d(\mathbf{q}, \mathbf{l}_p) \end{pmatrix} \tag{2.19}$$

To approximate this vector by a Gaussian vector, we linearize the distance function as:

$$d(\mathbf{q}, M) - d(\mathbf{q}^{ref}, \bar{M}) = \frac{\partial d}{\partial M}(M - \bar{M}) + \frac{\partial d}{\partial \mathbf{q}}(\mathbf{q} - \mathbf{q}^{ref}) \tag{2.20}$$

Replacing **q** by expression (2.18), we get a linear relation between $d(\mathbf{q}, M) - d(\mathbf{q}^{ref}, \bar{M})$ and $(M - \bar{M})$ that we do not express in this work for clarity. All the computations can however be done by considering:

1. The expression of the distance between the robot and landmarks in a given configuration.
2. The expression of the images in sensors with respect to the configuration of the robot.

Point 1 requires the kinematic model of the robot and point 2 requires in addition the model of the sensors.

The last equation will result in a Gaussian approximation of vector $d(\mathbf{q}, M)$ with mean value $d(\mathbf{q}^{ref}, \bar{M})$ and variance-covariance matrix Σ_d.

After approximating the distance vector, the probability of collision for an instance configuration should be calculated.

2.5.6 Calculating the probability of collision

In an uncertain world, the actual positions of obstacles are unknown. It is not straightforward to validate a configuration. We propose a collision probability parameter for making decision on accepting or rejecting a configuration.

Therefore, based on the calculated mean value of distance $\bar{d}$ to obstacles and its variance-covariance matrix Σ_d, the probability of collision for a proposed configuration will be calculated as follow:

$$P(collision) = 1 - P(coll\bar{i}sion) = 1 - P(d(\mathbf{q}, l) > 0) \tag{2.21}$$

where inequality between two vectors is equivalent to inequality between each component. The probability of non-collision will be as follows:

$$P(coll\bar{i}sion) = \int_{\mathbb{R}} 1_{d>0}\, dp(d) \tag{2.22}$$

$$P(coll\bar{i}sion) = \\ \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{det(\Sigma_d)}} \int_{(\mathbb{R}+)^n} exp(-\tfrac{1}{2}(x - \bar{d})^T \Sigma_d^{-1}(x - \bar{d}))dx \tag{2.23}$$

loa 1: Monte-Carlo Algorithm

```
Input: Number of test($N_{max}$)
Initialization: Probability ($p = 0$)
For $i$=1 to $N_{max}$
    Generating a Gaussian vector $v$ with mean value of $P\bar{d}$ and matrix variance-covariance of identity.
    IF $v$ is inside cone $P[(\mathbb{R}^+)^n]$
        $p++$
    End
End
$p \leftarrow \frac{p}{N_{max}}$
Output: Probability ($p$)
```

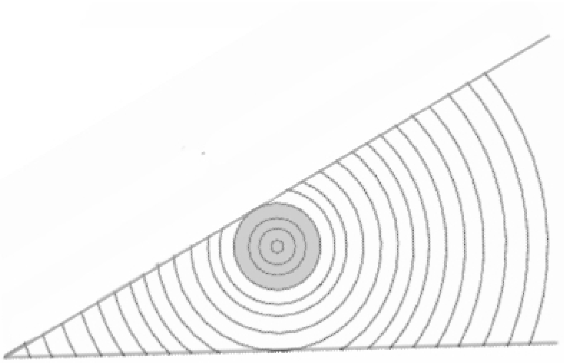


Figure 2.7: The integral is computed over a ball included in the cone defining non-collision.

As matrix Σ_d^{-1} is a symmetric positive definite matrix, there is a P such that $\Sigma_d^{-1} = P^T.P$. Let $y = P.x$, The change of variables theorem gives $dy = det(P).dx$. Thus, the probability becomes:

$$P(\overline{collision}) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{det(\Sigma_d)det(P)}} \int_{P[(\mathbb{R}^+)^n]} exp(-\frac{1}{2}(y - P.\bar{d})^T\Sigma_d^{-1}(y - P.\bar{d}))dy \tag{2.24}$$

we propose 2 methods for calculating this integral: Monte−Carlo and direct approximation.

Monte-Carlo method: The integral that we want to calculate is the probability of a Gaussian variable with a mean value of $P\bar{d}$ and matrix variance-covariance identity. We can use n independent Gaussian variables in the algorithm 1 to approximate the probability.

Direct approximation method: In this approach, we find a lower bound for this integral which is much faster than the Monte-Carlo method. Actually, we reduce the space of integration from a cone to a hyper-spheres which centered on $P\bar{d}$ having the greatest radius (Figure 2.7). The radius of this hyper-sphere will be equal or minimum distances of $P\bar{d}$ to various hyper-plane which limits the cone. We call this radius and the hyperbole of this radius centered on $P\bar{d}$ respectively R and $B_n(R)$. So, if the point $P\bar{d}$ is inside the cone, there will be a negative distance to one of the obstacles. Therefore, the probability calculations stop and the result will be:

$$P(collision) = 1 \tag{2.25}$$

As the iso-surfaces of our integral are hyper-spheres, we can easily have a hyper-sphere change of variable as below:

$$\begin{array}{l} \int_{P[(\mathbb{R}^+)^n]} exp(-\frac{1}{2}(y - P.\bar{d})^T(y - P.\bar{d}))dy > \\ \int_{B_n(R)} exp(-\frac{1}{2}y^T y)dy > \\ const(n-1)\int_0^{\mathbb{R}} r^{n-1}exp(-0.5r^2)dr \end{array} \tag{2.26}$$

as,

If n is even, then: $cons(n-1) = \frac{2pi^{p-1}}{(p-1)!}, n = 2p$

If n is even, then: $cons(n-1) = \frac{2^{2p+1}pi^p p!}{(2p)!}, n = 2p+1$

So, the only thing that should be calculated is $\int_0^{\mathbb{R}} r^{n-1}exp(-\frac{1}{2}r^2)dr$ which can be calculated by recursive method.

So, we can calculate a lower bound for the non collision probability or an upper band for the collision probability as follow:

$$\begin{array}{l} p(collision) < \\ 1 - \frac{1}{(2P)^{\frac{n}{2}}}const(n-1)\int_0^{\mathbb{R}} r^{n-1}exp(-0.5r^2)dr \end{array} \tag{2.27}$$

2.6 Example

To evaluate the approach, we consider 42 obstacles in the environment which are considered as landmarks also. Figure 2.8 shows the model of the environment which the obstacles are located in their mean value of their positions.

Given the map (the mean vector of obstacle positions and its variance-covariance matrix), robots size, and the start and goal configurations, the planner aims to produce a valid path between start and goal.

The robot is supposed to find a path between the initial point $I.P$ and final point $F.P$ in the map and tracks the planned path based on the captured images

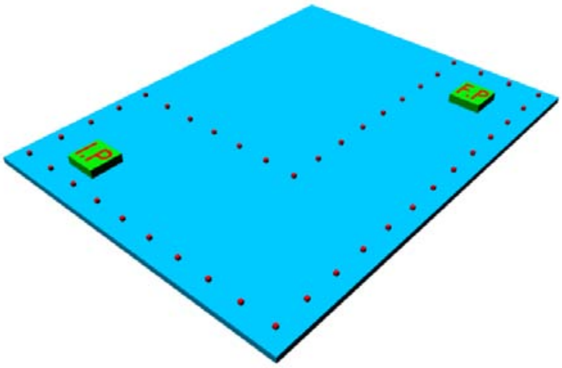


Figure 2.8: The mean value of the obstacle configurations and the robot initial (I.P) and final (F.P) configuration.

from the environment. The robot is a two wheeled mobile robot and is equipped with a 2D laser scanner. As it was explained in the previous sections, the planner uses the mean values of positions of obstacles (landmark). The planner builds a roadmap based on the proposed probabilistic method and finds the appropriate path between $I.P$ and $F.P$. The probability threshold is set to 30% in this example.

At the step of executing the generated path in the real environment, as the robot navigates in the environment and captures image of its environment, it gets the real positions of the landmarks which would be different from the mean values.

Figure 2.9 shows the planned path in the model. Red dots represent the mean values the position of obstacles (landmarks). The red line represents the path planned based on this stochastic map. Blue dots are the actual positions of the obstacles and the blue line represents the path that is followed by the robot by performing localization on the landmarks.

2.7 Conclusion

In this chapter, we presented a new framework to take into account uncertainties of a map in path planning. We believe that this framework is more complete than previously published work on this topic.

The main originality of our framework resides in the prevision of the path actually followed by the robot given actual position of landmarks (simulation of localization and motion control process), and in the estimation of collision probability in a stochastic framework.

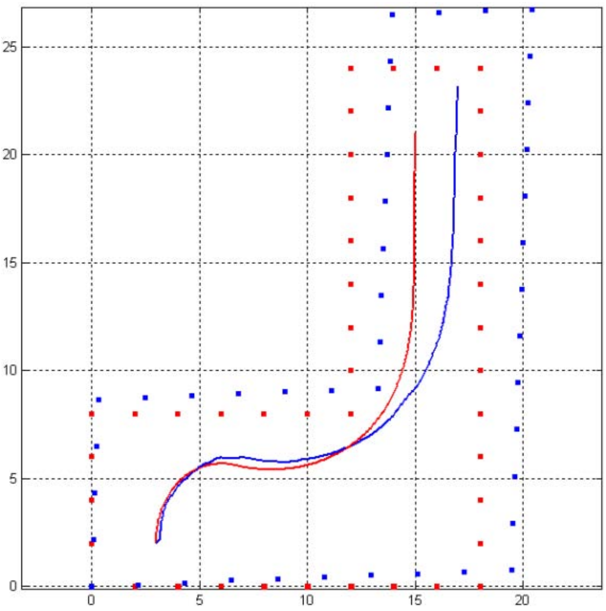


Figure 2.9: Red dots represent the mean values in the stochastic map. The red line represents the planned path. Blue dots represent the actual positions of the landmarks and the blue line represents the path which is followed by the robot by performing localization on the landmarks.

Let us notice that the input of the path planning algorithm is the output of the SLAM process. We then proposed an algorithm to plan a path in stochastic maps. A simple example illustrates this preliminary work on the way to binding path planning with SLAM.

Chapter 3

Motion planning in changing environments

3.1 Introduction

Computation of a collision-free path for a movable object among obstacles is an important problem in the field of robotics. The problem has been studied and several approaches were proposed for solving the problem in static environments. However, most of real life environments are non-static and for path planning in such environments, the key issue is how to deal with the changes in such models. Considering a cost for time and memory usage during planning, it becomes a challenging issue to use efficiently the existing information in each state of the environment for solving a motion planning problem in the newly updated environments.

Dynamic changes in environment are very common in various applications such as motion planning in industrial environments [47] and navigating in real word [19] or virtual word [55]. Although the probabilistic method such as RRT and PRM are efficient enough to deal with motion planning problems, they are focussed on static environments.

To deal with changing environments, some approaches have been implemented and address the problem:

M. Cherif and M. Vidal used a roadmap based planner for planning in changing industrial plants [10]. Their idea is to find a collision-free trajectory for a set of mobile robots for handling a group of movable objects in a cluttered industrial plant. Their planner copes with constraints due to changing of the workspace. The planner decomposes the problem into building a single roadmap and then planning coordinated collision-free paths for these systems. It includes manipulation of the roadmap to guarantee the incorporation of the workspace changing. The basic idea of their approach consists in pre-processing the configuration spaces of all the moving systems (i.e., robots, and robot-object systems) and build a roadmap. They manipulate this roadmap for accounting the changes of the workspace. Their planner uses this roadmap to find collision free paths for the individual systems and to coordinate their motions safely.

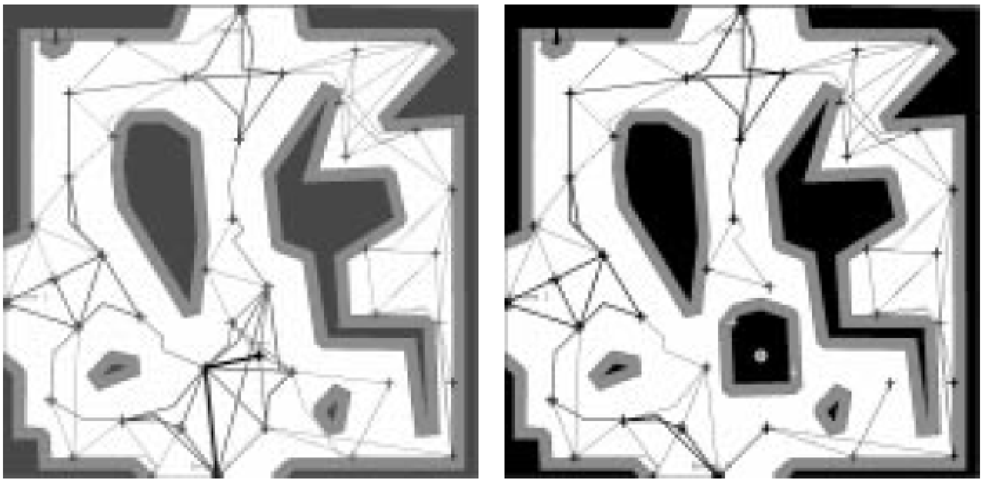


Figure 3.1: From [10]: Left: the planned path. Right: the resulting updated roadmap after changing the workspace

O. Brock and O. Khatib used a strip framework to solve the problem [7, 8]. In the first contribution, they present the elastic strip framework, which addresses the problem by integrating global motion planning methods with a reactive motion execution approach. To maintain a collision free trajectory, a given motion plan is incrementally modified to react to changes in the environment. This modification can be performed without suspending task behaviors. The elastic strip framework is computationally efficient and can be applied to robots with many degrees of freedom. In this approach, a trajectory can be imagined as elastic material filling the volume swept by the robot along the trajectory.

The elastic strip framework is very similar in spirit to the elastic band framework [56]. In the elastic band framework a previously planned robot motion is modeled as elastic material. A path between an initial and a final configuration can be imagined as a rubber band spanning the gap between two points in space. Obstacles exert a repulsive force on the trajectory, resulting in an incremental modification. This can be imagined as a moving obstacle pushing and deform-

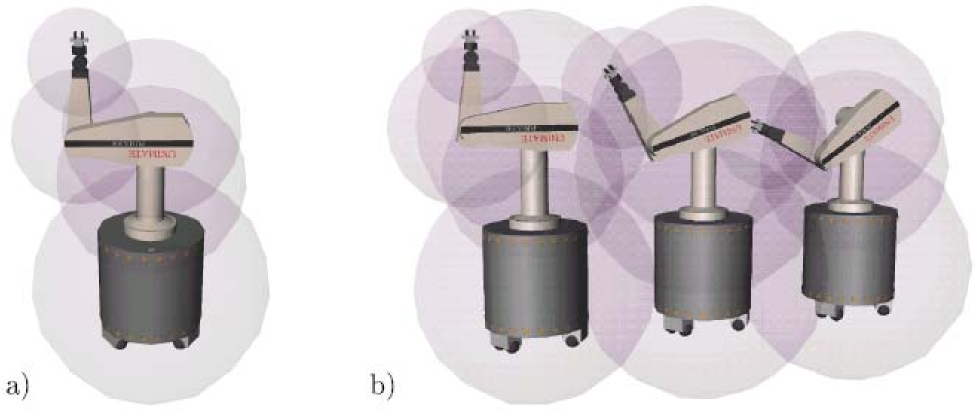


Figure 3.2: From [7]: a) A protective hull around the Stanford Mobile Manipulator. b)An elastic tunnel formed by several overlapping protective hulls.

ing the rubber band. When the obstacle is removed, the trajectory will return to its initial configuration, just as the rubber band would. An elastic band is represented as a one-dimensional curve in configuration space. This leads to high computational complexity for high-dimensional configuration spaces.

By avoiding configuration space computation, the framework becomes applicable to robots with many degrees of freedom. In the elastic band framework, the trajectory and the task are both described in workspace. In the elastic strip framework a trajectory can be imagined as elastic material filling the volume swept by the robot along the trajectory. This strip of elastic material deforms when obstacles approach and regains its shape as they retract.

In the second contribution, a planning operation generates a path. The path is augmented by a set of paths homotopic to it. This set is represented implicitly by a volume of free space in the work space. Effectively, this corresponds to delaying part of the planning operation for the homotopic paths until motion execution. During execution reactive control algorithms are used to select a valid path from the set of homotopic paths, using proximity to the environment as a simple and effective heuristic and thereby significantly pruning the search in the configuration space. The underlying idea is to represent a set of homotopic paths by the workspace volume a robot would sweep out along them. This can be done without considering the kinematic properties of the robot, i.e. without exploring the configuration space. The prerequisite is the existence of a valid path, called candidate path. Assume a planner has generated such a candidate path and it lies entirely in free space. The free space around the candidate path must contain the volume swept by the robot along paths homotopic to the candi-

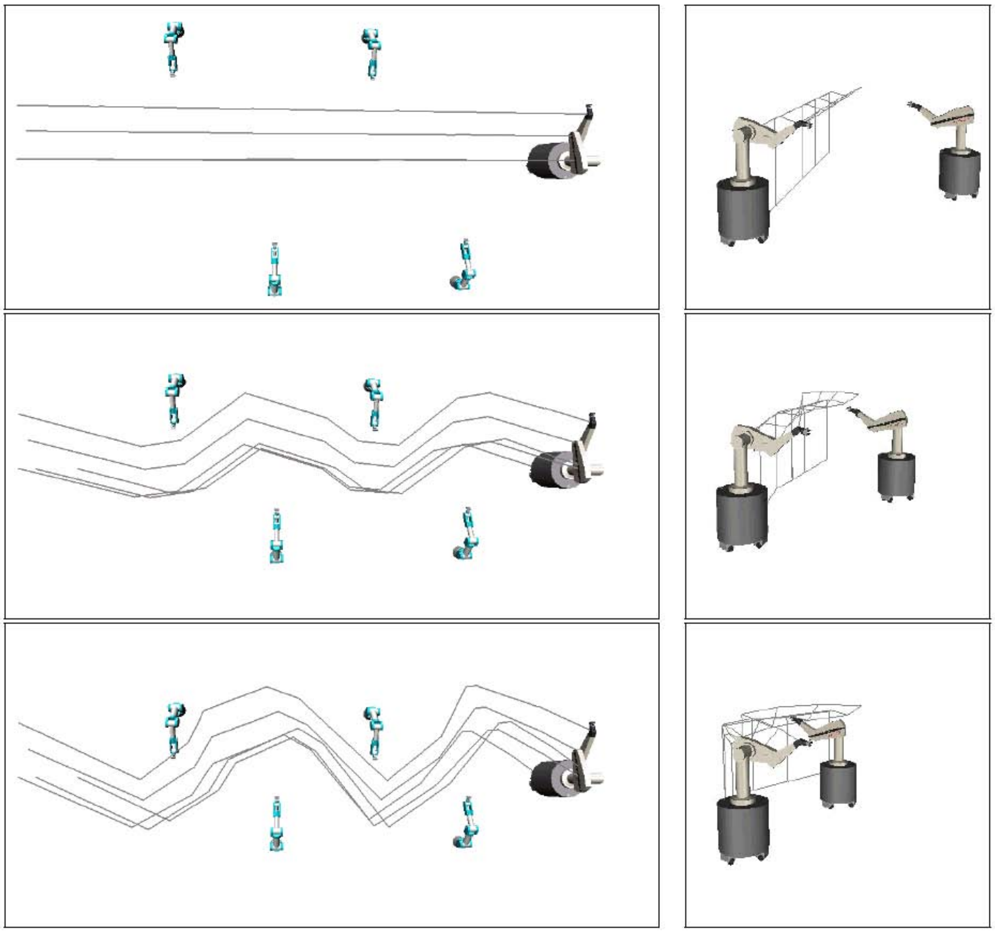


Figure 3.3: From [8]: On the left: Four manipulator arms move into the path of the Stanford Mobile Manipulator; replanning using elastic strips is performed in real-time to maintain a valid path. The path is indicated by lines connecting points on conscutive configurations along the elastic strip. It is converted into a smooth trajectory during execution. On the right: A Stanford Mobile Manipulator moves into the path of another one. The path is updated in real-time to avoid a collision.

date path itself. Those paths are represented implicitly by an approximation of the free space around the candidate path, rather than computing them explicitly. During execution this set of homotopic paths can be searched very efficiently for an alternate valid path, should the candidate path be invalidated by changes in the environment.

Using such a representation, planning and control can be integrated very tightly: The path generated by a motion planner is transformed into a more general representation by augmenting every path with an implicit representation of paths homotopic to it. Control algorithms can then be used during execution to efficiently search that space to find a valid path, should the original one become invalid due to changes in the environment.

O. Brock and L. Kavraki used decomposition based approach proposed in [6]. They decompose the original planning problem into simpler subproblems, whose successive solution empirically results in a large reduction of the overall complexity. To achieve real-time motion planning for robots with many degrees of freedom, a motion planning paradigm based on problem decomposition was proposed. The paradigm addresses planning problems in which a minimum clearance to obstacles can be guaranteed along the solution path.

The overall planning problem is decomposed into two planning subtasks: capturing relevant connectivity information about the free space in a low-dimensional space and planning for the degrees of freedom of the robot in its high-dimensional configuration space. The solution to the lower-dimensional problem is computed in such a manner that it can be used as a guide to efficiently solve the original planning problem. This allows decomposition-based planning to achieve real-time performance for robots with many degrees of freedom.

David Hsu *et al.* explore the *configuration* × *time* space for moving obstacles along known trajectories [22]. They presents a novel randomized motion planner for robots that must achieve a specified goal under kinematics and/or dynamic motion constraints while avoiding collision with moving obstacles with known trajectories. Their planner encodes the motion constraints on the robot with a control system and samples the robot's *state* × *time* space by picking control inputs at random and integrating its equations of motion.

Their algorithm constructs a roadmap of sampled milestones in the *state* × *time* space of a robot. It samples new milestones by first picking at random a point in the space of admissible control functions and then mapping the point into the state space by integrating the equations of motion. Thus the motion constraints are naturally enforced during the construction of the roadmap. They evaluated the performance of the algorithm through both theoretical analysis and extensive experiments. The result is a probabilistic roadmap of sampled *state* × *time* points, connected by short admissible trajectories. For each planning query, their planner generates a new roadmap to connect an initial and a goal state time point. The application of this approach is limited because of requiring information about obstacle trajectories.

L. Jaillet and T. Siméon proposed a PRM-based motion planner for dynamically changing environments [26]. The proposed planner is based on probabilistic path planning techniques and it combines techniques originally designed for solving multiple query and single query problems. The planner first starts

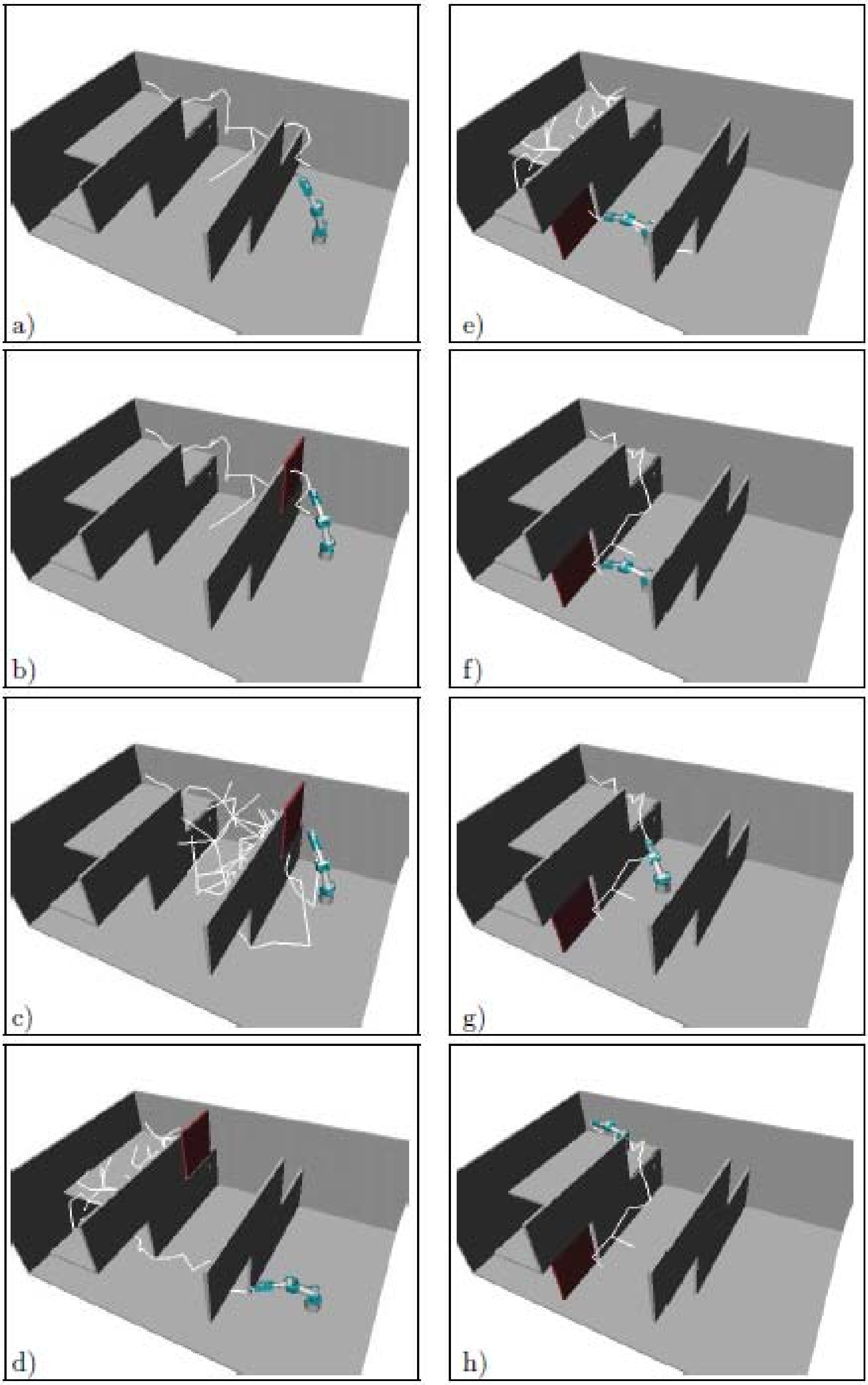


Figure 3.4: From [5]: Real-time planning in a dynamic environment.

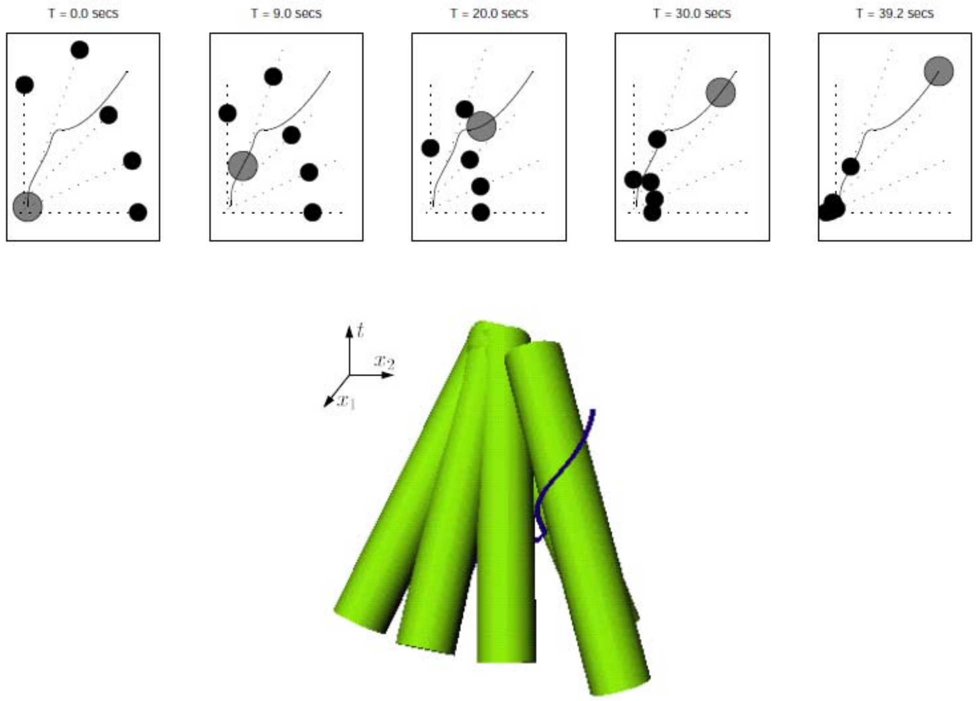


Figure 3.5: From [22]: Top) Computed example for the air-cushined robot. bottom) *configuration* × *time* representation.

with a preprocessing stage that construct a roadmap of valid paths with respect to the static obstacles. It then uses lazy-evaluation mechanisms combined with a single query technique as local planner in order to rapidly update the roadmap according to the dynamic changes. This allows to answer queries quickly when the moving obstacles have little impact on the free space connectivity. When the solution can not be found in the updated roadmap, the planner initiates a reinforcement stage that possibly results into the creation of cycles representing alternative paths that were not already stored in the roadmap.

Because the environment is only partially modified between each of the queries, they use a two-stage method. First, they compute a roadmap for the robot and the static obstacles (Figure 3.7.1), without considering the presence of the moving obstacles. Then to solve the path query, portions of the roadmap are updated by checking if existing edges are collision-free with respect to the current position of the moving obstacles. Colliding edges are labeled as blocked in the roadmap. If this labeling permits to obtain a path which does not contain any blocked edge, then a solution is found (Figure 3.7.2). When the solution path contains blocked edges, a mechanism of local reconnection based on RRT tech-

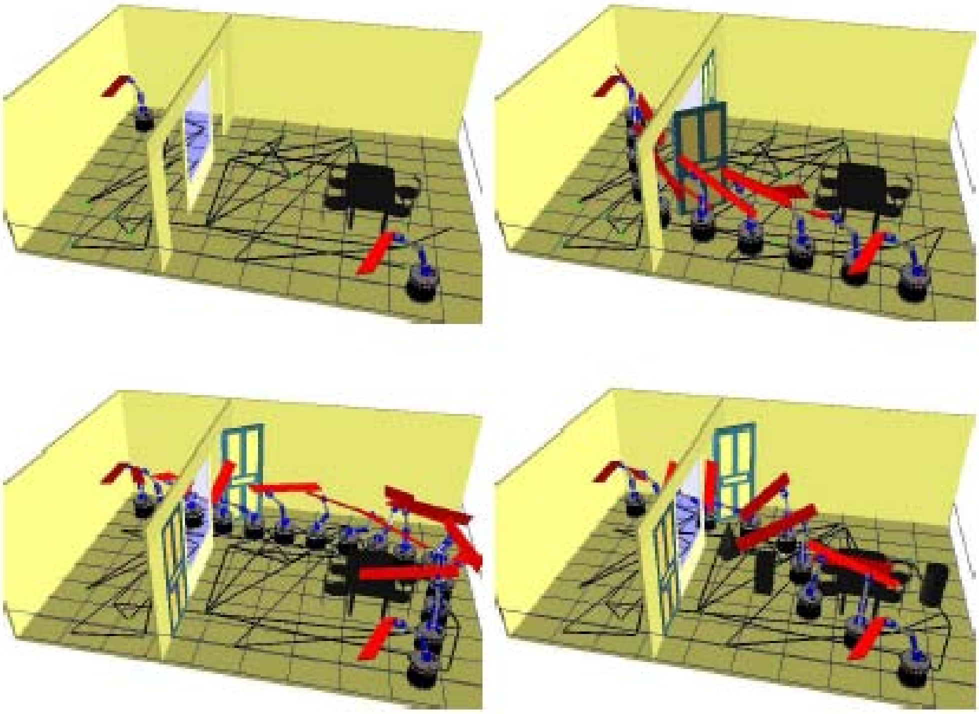


Figure 3.6: From [26]: Example of a 3D scene with moving obstacles: start and goal configurations of the 9dof mobile manipulator carrying a board and three path solutions obtained for several settings of the environment (doors closed/open and three other moving obstacles placed around the table)

niques is used. It tries to reconnect the two extremities of those blocked edges (Figure 3.7.3). Finally, if the existing roadmap does not allow to find a solution, new nodes are inserted and the roadmap is reinforced with cycle creation (Figure 3.7.4). In practice, this general approach is combined with several methods avoiding unnecessary collision tests, which makes it more efficient.

J. Berg, D. Ferguson and J. Kuffner in [66] use probabilistic sampling to create a robust roadmap encoding the planning space and then plan and re-plan over this graph in an anytime, deterministic fashion. The resulting algorithm is able to generate and repair solutions very efficiently, and improve the quality of these solutions as deliberation time allows.

In the first, a roadmap is built representing the static portion of the planning space. Next, an initial trajectory is planned over this roadmap in state-time space, taking into account any known dynamic obstacles. This trajectory is planned in an anytime fashion and is continually improved until the time for deliberation is exhausted. Further, while the agent executes its traverse, its trajectory continues

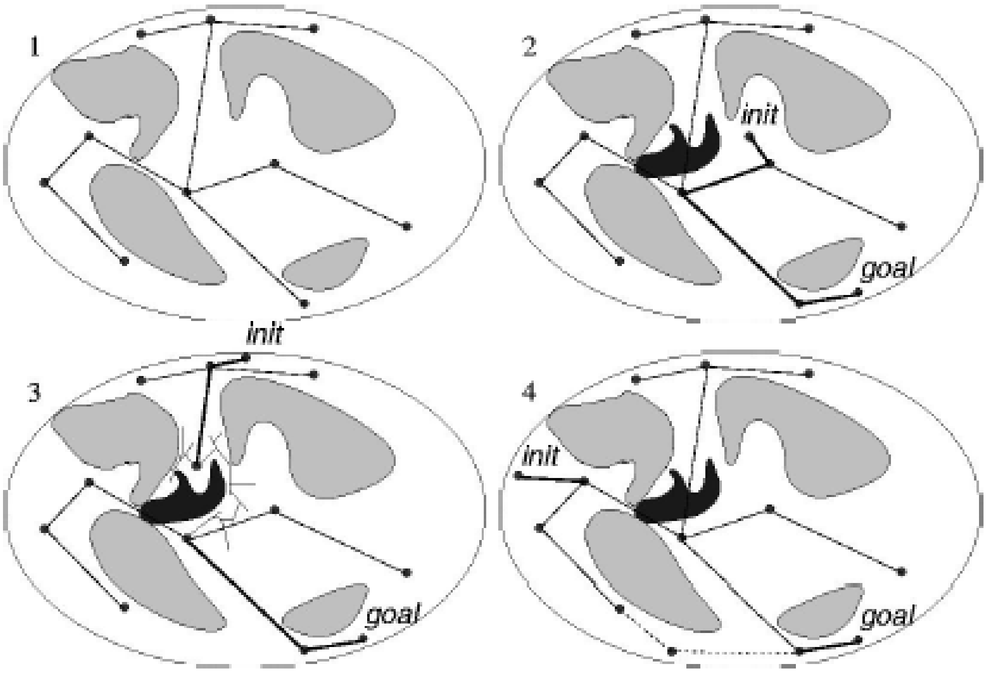


Figure 3.7: From [26]: A static roadmap is first computed in the configuration space of the robot (1). During queries, a solution path can be found directly inside this roadmap (2) or via a RRT like technique to reconnect edges broken by dynamic obstacles (3). If the existing roadmap does not permit to find a solution, new nodes are inserted and the roadmap is reinforced with cycle creation (4).

to be improved upon. Finally, when changes are observed, either to the static or dynamic elements of the environment, the current trajectory is repaired to reflect these changes. Again, this is done in an anytime fashion, so that at any time a solution can be extracted.

There are a lot of attention to use mapping between workspace and configuration space to update roadmaps in changing environment. These approaches are adopted to update the roadmap online rather than to construct it once again when environments change. Dynamic Roadmap Method (DRM) [43, 29], preserves a mapping from workspace to configuration space ($\mathcal{W}$-$\mathcal{C}$ mapping) as a mechanism to indicate invalid nodes and edges in a roadmap. DRM is a kind of variation of PRM to solve path planning problems in changing environments. The idea behind DRM is to represent the relationship between workspace and a constructed roadmap in configuration space so that the roadmap can be updated accordingly when obstacles move in workspace. DRM performs well when obstacles move in workspace and environment changes.

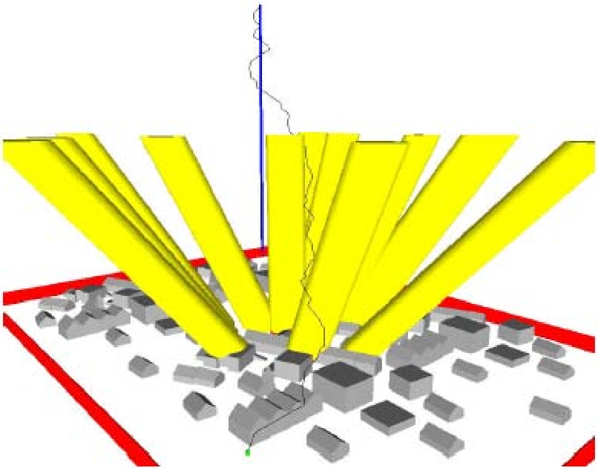


Figure 3.8: From [66]: An example path (in black) planned through state-time space from the initial robot position (in green) to the goal (shown as a vertical blue line extending from the goal location upwards through time). Also shown are the extrapolated trajectories of the dynamic obstacles (in yellow). The underlying PRM has been left out for clarity but can be seen in Right image.

In [43], P. Leven and S. Hutchinson begin by constructing a graph that represents a roadmap for obstacle free environment in the configuration space. Nodes are generated by shooting sample configurations and these nodes are then connected to form a the roadmap by using straight-line planner. In this phase, they consider obstacle free workspace to generate the initial roadmap. The only constraint in this step is self collision for the robot. They prohibited the configuration in which the robot is in self collision for generating the roadmap. Then, they encode a mapping from work space to nodes and edges. $\mathcal{W}$-$\mathcal{C}$ mapping is made up of $\mathcal{W}$-$\mathcal{C}$ nodes mapping and $\mathcal{W}$-$\mathcal{C}$ edges mapping, which map every basic cell in workspace to nodes and edges in roadmap, respectively.

During the planning, in case of changing in the environment, they detect the cells which became obstacles. Thanks to the $\mathcal{W}$-$\mathcal{C}$, they detect the concerned nodes and edges to detected cells. Then, they erase the concerning nodes and edges to a modified cell of the workspace which will result in a roadmap whose nodes and edges are valid. Their planner uses this roadmap to plan a motion between initial and goal configurations. The planning is reduced to connecting initial and goal configuration to the roadmap. In case of failure, their planner addresses more nodes to enrich the roadmap and solve the problem.

In [29] M. Kallman and M. Mataric evaluate the tradeoffs between using DRM and applying RRT directly in changing environments which requires no preprocessing or maintenance. They ground the analysis in several benchmarks

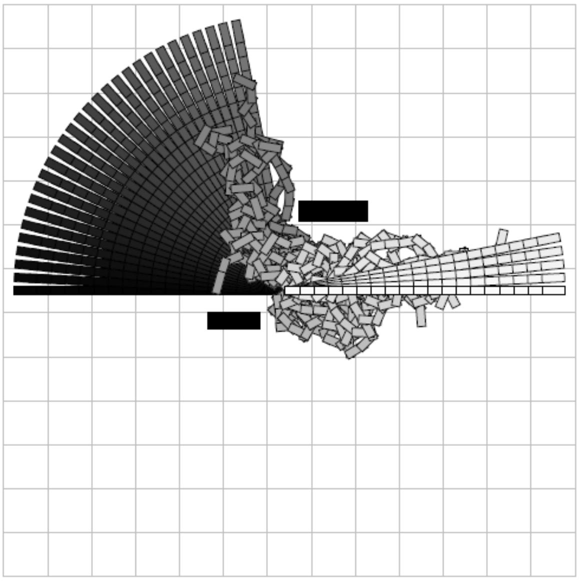


Figure 3.9: From [43]: A plan for a 19-joint robot passing through a relatively narrow corridor. The dark blocks are the obstacles.

in virtual environments with randomly moving obstacles. Different robotics structures are used, including a 17 degrees of freedom model of NASA's Robonaut humanoid.

For implementing the method, two data structures are computed off line: the roadmap $\mathcal{R}$ and a grid-based cell decomposition of the workspace $\mathcal{W}$. The roadmap is computed only considering the robot and the static obstacles in $\mathcal{W}$. The grid G stores in each cell $c \in G$, all nodes and edges of $\mathcal{R}$ that are affected by c. They call this process cell localization and, coupling G with $\mathcal{R}$, they obtain a Dynamic Roadmap (DRM). During run time, dynamic obstacles are tracked and each time an obstacle appears, disappears, or moves, the affected nodes and edges of $\mathcal{R}$ are updated accordingly. Their results show that dynamic roadmaps can be both faster and more capable for planning difficult motions than using on-line planning alone. The evaluation results recommend that it is worth constructing $\mathcal{W}$-$\mathcal{C}$ mapping and maintaining a roadmap that could be updated dynamically.

H. Liu *et al.* also used the approach of mapping between workspace and configuration space [45]. $\mathcal{W}$-$\mathcal{C}$ nodes mapping coupled with lazy edges evaluation is used to ensure a generated path containing only valid nodes and edges when constructed probabilistic roadmap becomes partially invalid in changing environments. They made an interesting observation and coupled the $\mathcal{W}$-$\mathcal{C}$ approach

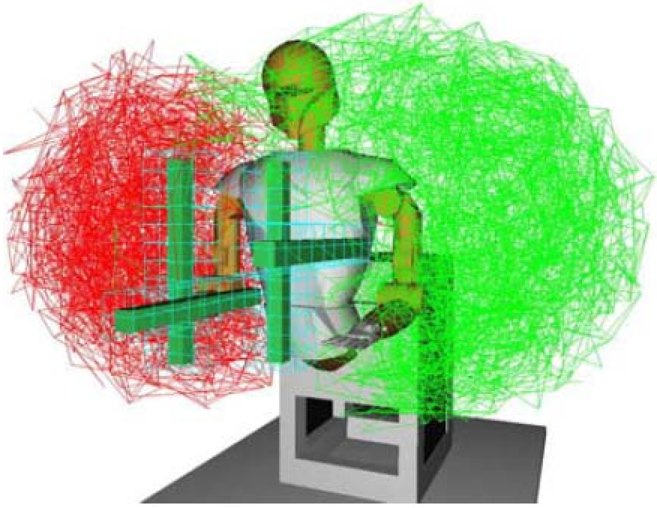


Figure 3.10: From [29]: Portions of the roadmap are dynamically invalidated according to the obstacles inserted in Robonaut's workspace.

with lazy PRM to accelerate the process. They believe that it takes only a little time to build $\mathcal{W}$-$\mathcal{C}$ nodes mapping and most time is spent for the computation of $\mathcal{W}$-$\mathcal{C}$ edges mapping. Besides, another observation is that $\mathcal{W}$-$\mathcal{C}$ nodes mapping is more important than $\mathcal{W}$-$\mathcal{C}$ edges mapping. If a node is invalid, a collision-free path will not contain all its adjacent edges and thus these edges should be invalid implicitly. An invalid edge needs to be marked explicitly only if its two adjacent nodes are valid, and this situation appears only when the regions of configuration space obstacles are very small or narrow.

In their approach, sampled nodes of a roadmap can be regarded as milestones in configuration space. Therefore, for probabilistic path planner, $\mathcal{W}$-$\mathcal{C}$ nodes mapping is enough to preserve major information representing relationship between workspace and configuration space. In their approach, $\mathcal{W}$-$\mathcal{C}$ nodes mapping is preserved and constructed within a little time while skipping $\mathcal{W}$-$\mathcal{C}$ edges mapping. In order to ensure the validity of found paths, lazy edges evaluation, which is successfully used in Lazy PRM method [4], is integrated in their method. Lazy edges evaluation is used to ensure all edges valid along a found path. They use a dual-manipulator system to evaluate the efficiency of their planner in changing environments.

The approach we propose in this work to plan motions in changing environment is also a $\mathcal{W}$-$\mathcal{C}$ method and is an extension of the approach which was presented in [43]. Their method is efficient in dealing with changes in environments. However, we tried to optimize their approach for integrating it in the

Figure 3.11: From [45]: Path planning for a dual-manipulators system with 12 DOFs in changing environments.

problem of motion planning for humanoid robot in a big environment which is explored by vision.

In the approach we describe in this chapter, we use the same basic idea of workspace cell-decomposition, but we use two levels of decompositions and we test again for collision the roadmap features lying in cells where obstacles have appeared.

3.2 General idea of the planner

A robot is supposed to navigate in an environment. The robot has a perfect model of the environment at the planning moment. After planning a path, it receives updated information from the sensors regarding workspace. The positions of some obstacles are changed. Some obstacles disappear and some new obstacles enter to the environment. The robot is supposed to plan another path to a goal configuration in the updated environment.

As it was explained in the previous section, considering a cost for time and memory, how to deal with changes in environments is an important issue . Using a roadmap based planner, the basic idea is looking for an algorithm to use efficiently the existing solution for the previous problem to solve the new problem.

As the information concerning the free accessible zone $\mathcal{C}_{free}$ is stored in a roadmap, it is a good idea to use this information in the next step. But the environment is modified and some information in the roadmap is not valid anymore.

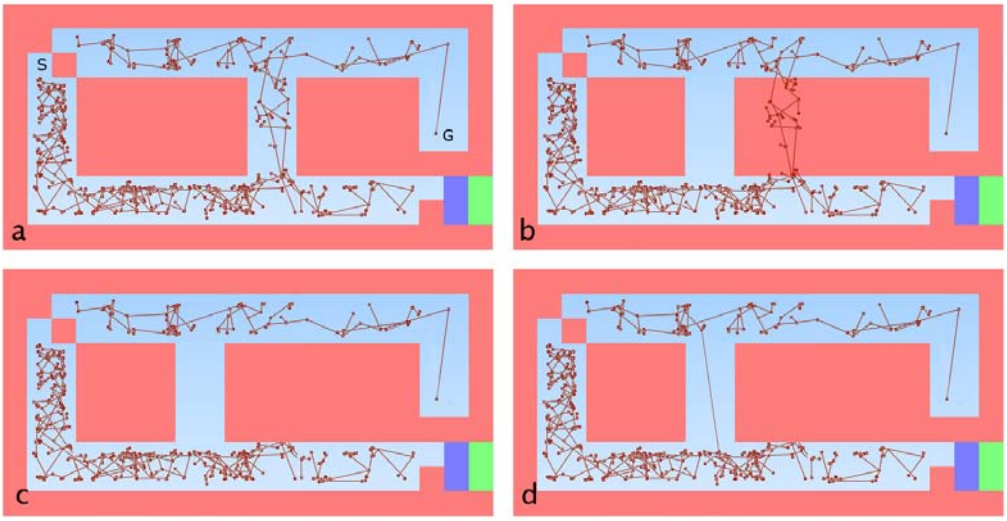


Figure 3.12: The planner updates a roadmap based on the modification in the environment. Figure a illustrates simple 2D model of the environment with a roadmap that connect initial and final configuration. Figure b shows the recent environment which is modified partially and in result, some nodes and edges of the roadmap are in collision based on the new model. In c, the invalid nodes and edges are erased and finally, in d, the updated roadmap is used to find the path in new environment.

Erasing the invalid nodes and edges in the roadmap will result in a collision free roadmap. This roadmap can be used as an initial estimate of accessible free zone in the modified workspace.

Now, the question is how to optimize the algorithm to update an existing roadmap. Logically, checking the validity of all nodes and edges is not an efficient method in case of simple modification in workspaces.

We use an algorithm to detect the modified zone of the workspace. Then, we extract the nodes and the edges which have intersection with the modified zone. By re-validating these limited number of nodes and edges, the planner detects the invalid nodes and edges rapidly and erase them. This algorithm will result in a valid roadmap faster than checking all nodes and edges.

Figure 3.12, illustrates how the algorithm works. A roadmap is modified based on the changes of the workspace. The nodes and edges that became invalid are detected an erased.

3.3 Work space cell-decomposition

We use a two-level workspace decomposition to deal with the problem. In the first level, workspace is decomposed into a number of cells for which we store a list of nodes and edge. In the second level, each cell is decomposed into voxels which are the smallest volume of environment. Voxels can have the state of *occupied* , *free* or *unknown*. Based on the captured information from environment, the state of each voxel may change.

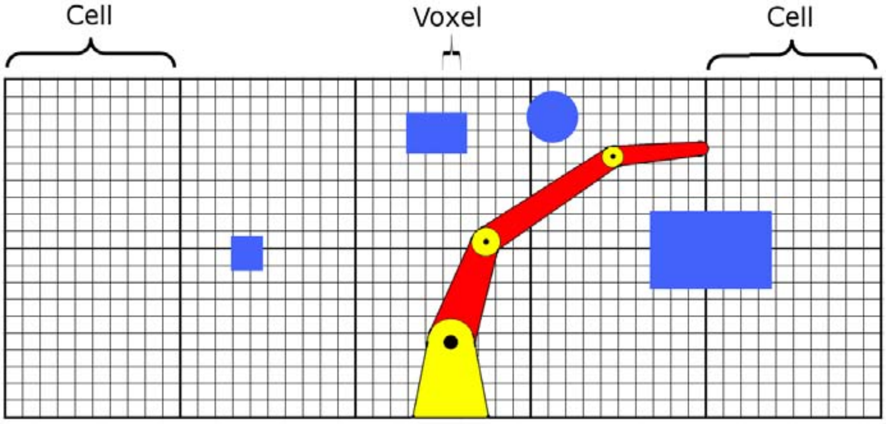


Figure 3.13: 2D cell decomposition for a robotic arm.

$$\mathcal{W} = \bigcup_{i=1}^{m} C_i, \forall i, j \in [1, m], C_i \cap C_j = \emptyset \tag{3.1}$$

C_i presents a cell in the model and $\mathcal{W}$ is the workspace. In other hand,

$$C_i = \bigcup_{i=p}^{q} V_i, \forall i, j \in [p, g], V_i \cap V_j = \emptyset \tag{3.2}$$

It should be mentioned that in this approach, cell-decomposition is not necessarily uniform in the workspace. Therefore, based on any preliminary information about the environment, the shape and size of cells would be different in some regions to optimize calculations.

Moreover, as in [43], a mapping between configuration space $\mathcal{C}$ and work space $\mathcal{W}$ is built to associate any configuration to a list of cells.

$$\phi(\mathbf{q}) = \{C_i \mid A(\mathbf{q}) \cap C_i \neq \emptyset\} \tag{3.3}$$

where $\mathbf{q}$ is configuration and $A(\mathbf{q})$ is a volume of environment which is occupied by robot in $\mathbf{q}$. In fact, thanks to this mapping, each $\mathbf{q}$ is associated to a list of cells

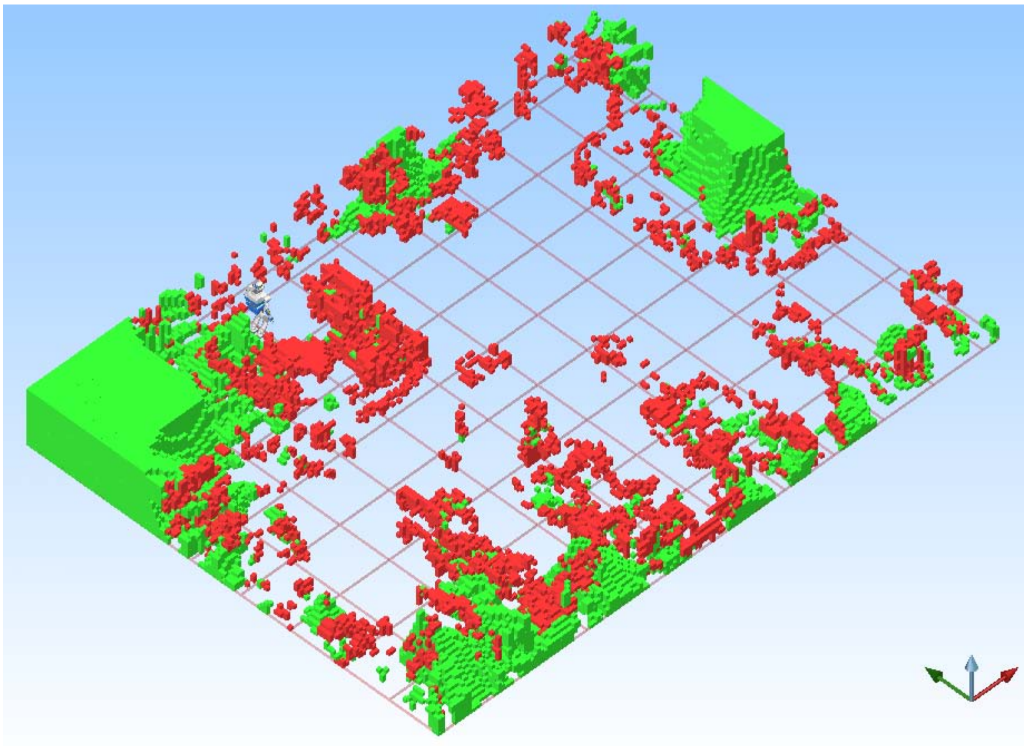


Figure 3.14: Motion planning in 3D model of robotics lab in LAAS-CNRS. The size of workspace is 17m ×12.5m×1.7m. The workspace is modeled with 10cm×10cm×10cm 3D *Voxels*. Also, workspace is decomposed to the 1.5m×1.5×1.5m *Cells* to implement the approach.

which are in collision with the robot at this configuration. By using (3.3), each node and each edge are associated to one or several cells. To be more precise, each node is a representation of a **q** and each edge is a mapping from interval $[0, T]$ to $\mathcal{C}$.

Therefore, (3.3) maps each node and edge to one or several cells.

On other hand, based on this approach, a list of nodes and edges will be assigned to each cell. Therefore:

$$C_i = \{L_i^N, L_i^E\} \tag{3.4}$$

where, L_i^N is a list of nodes which the robot in their configuration will be in collision with C_i. Also, L_i^E is a list of edges for which the robot is in collision with C_i if it tracks that edges.

Figure 3.13 illustrates the concept of cells and voxels in a 2D workspace.

3.4 Updating cells

L_i^N and L_i^E are updated as the planner finds a path between initial and final configurations. To optimize the calculation time, cell lists are updated during execution of the generated path. Therefore, by using the updated roadmap, the concerned nodes and edges should be added to the associated lists of each cells C_i.

The figure 3.15 shows the process sequence in encountering a change in the workspace.

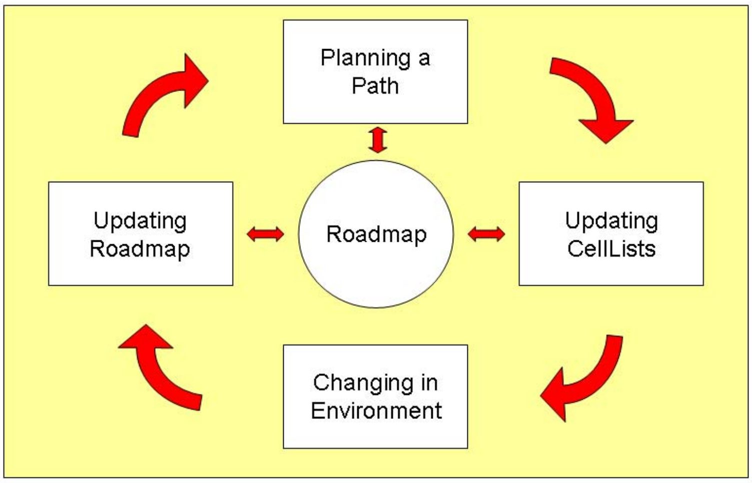


Figure 3.15: The sequence of process in encountering a change in an environment.

For updating L_i^N with a new node, the robot will be considered at the concerned **q** and the node will be added to L^N of cells which are in collision with robot. Updating L^E is more complicated and time consuming. In fact, it consists in a mapping between an edge and a set of cells which are in collision with robot if the robot sweeps the concerned path. To update L_i^E, the same algorithm as in [43] was used.

Figure 3.14 illustrates a 3D model which is decomposed into 1.5 m cell to implement the approach.

3.5 Constructing roadmap

Having an instant model of environment, we begin by creating a roadmap in the environment. Our planner uses all the internal constraints such as degree of freedom and kinematic limitations to create a roadmap in the free zone of the workspace. Nodes are generated by shooting random configuration in $\mathcal{C}$ and valid nodes are added to the roadmap by valid edge. We denote the roadmap as $\mathcal{R}(N, E)$ where N and E are the list of nodes and edges respectively.

However, as the environment is not static, the free zone is modified as the robot received new information. So some nodes in N and some edges in E may become invalid in the next instance of the environment model. After updating the roadmap based on the modification in environment, the planner uses this roadmap to find a path between initial and final configuration. In case of failure, planner shoots random configurations and enriches the updated roadmap to solve the problem.

3.6 Updating roadmap

Our approach for updating roadmaps is an extension of the approach presented in [43]. Our planner can detect the cells which are modified. It detect the cells whose voxels status changed to *occupied*.

After detecting the modified cells, instead of erasing their nodes and edges , our planner re-validates their features. The non valid features are erased and the valid nodes and edges are used for the next path planning.

3.7 Examples

In order to validate our approach for motion planning in changing environment, we needed to build a large model of environment. For building the model of workspaces, we rely on an occupancy grid approach. As it is presented in chapter 5, localization is a critical issue in generating a model. However, we do not have yet a precise localization module on HRP2 robot. We therefore built the map using another robot *Jido* [*] equipped with an accurate localization module. HRP2 then planned motions in this model.

Jido, figure 3.17, is a mobile manipulator which is equipped with a stereo camera and also a laser scanner for localization. *Jido* was used in our experiments for taking several photos. These photos were used to generate the 3D occupancy model of the environment. The experiment was conducted in the robotic laboratory of LAAS-CNRS as a large workspace.

The allocated size of the environment in these models is 17 m×12.5 m×1.7 m. The size of each voxel in the these 3D occupancy grid map was 10 cm×10 cm ×10 cm. The model is composed of 361250 voxels.

---

[*] http://www.laas.fr/robots/jido/data/en/jido.php

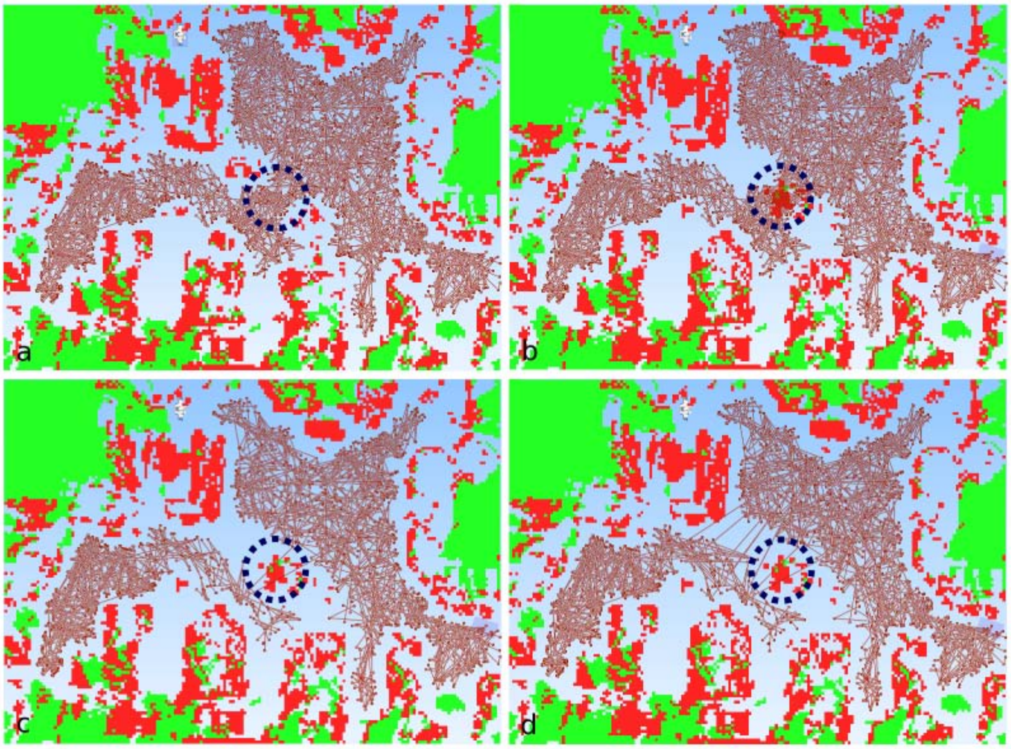


Figure 3.16: Illustration of roadmap management during updates of workspace. Red and green voxels represents *occupied* and *unknown* zone respectively. The circle shows a *free* part of environment which becomes *occupied* later: a) A roadmap is illustrated in the top view of a 3D model of an environment which is generated by vision. b) The model is updated based on new information from camera. c) The roadmap is modified based on the updates in the 3D model by using our cell decomposition algorithm. Some nodes and edges are erased from the roadmap during the modifications. d) The modified roadmap is used for a motion planning process. Some nodes and edges are added to the roadmap.

We used a path planning algorithm for the HRP2 bounding box. The generated path for the bounding box is later used to generate a whole body path for the robot. Although we simplified the problem of motion planning from a robot with 40 degrees of freedom to a box with 3 degree of freedom, the approach is valid in case of using all the degree of freedom in the stage of planning.

We conducted 3 tests in the generated models to illustrate the efficiency of our planner. The presented times in each experiment are the average of computation times for 10 tests. All the simulations were performed on a 2.13 GHz Intel Core 2 Duo PC with 2 GB RAM.

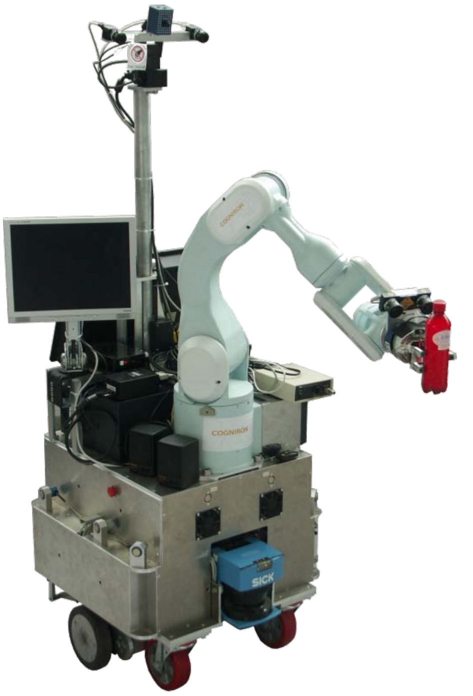


Figure 3.17: The mobile robot Jido which is used in the experiment for taking the required photos.

3.7.1 Experiment 1

Two models of the lab were generated. The main difference between these 2 models is a table at the middle of the lab which was displaced. Displacement of this table cause some *free* regions to become *occupied* and also some *occupied* regions to become *free*. Although the only difference in the 2 states of environment is the displacement of the table, there are much more differences between the two 3D models. In fact, based on any error of localization, some parts of the model were modified. There are 7848 *occupied* voxel in the first model. There are 1810 voxels which have been *free* or *unknown* in the first model and became *occupied* in the second model.

Figure 3.18 illustrates the 2 generated models which are used to test our approach in changing environments. Although environment modeling was not done on-line with HRP2, it does not disturb the efficiency of the method.

In this scenario, we implement 3 tests to evaluate the efficiency of our algorithm in terms of time. The goal is measuring the path planning time between an initial and goal configurations in the 2 models.

In the test A, we do not use our algorithm and the roadmap is erased after planning a path in the first environment. So, the time of planning a motion

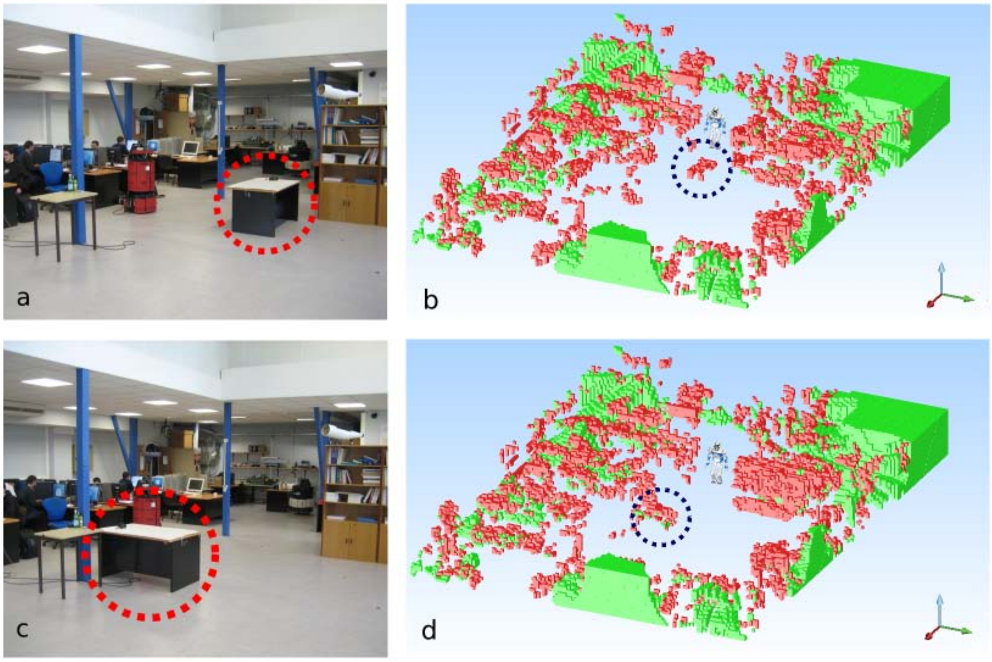


Figure 3.18: Changes in the environment. HRP2 uses the cell decomposition method to validate some of the nodes and edges in the existing roadmap for planning next motions

between 2 configurations was measured separately in the 2 models. In the test B, we keep the roadmap after solving the motion planning problem for the first model. However, after loading the second model, we validate all the nodes and edges of the roadmap with the new obstacles in the second model.

Finally, in test C, we use our approach to optimize the computation time. We use our approach and the planner validates locally just some of the nodes and edges of the roadmap. The size of cells in the last types of the tests was set to 1.7 m × 1.7 m × 1.7 m and each cell is composed of 4913 voxels. Table 3.1 compares the average of computation time in each stage of the conducted tests.

Table 3.1: Time performance.

Test	**Solving problem 1 [ms]**	**Updating Roadmap [ms]**	**Solving problem 2 [ms]**
A	399563	0	314452
B	399563	5447	18716
C	399563	1936	18716

3.7.2 Experiment 2

To analyze the effects of the cell size on the computation time and the memory usage, we conduct 6 tests. We use the same 3D models as the previous experiment. The first model of the environment is loaded in the planner. Then, the planner generates a roadmap and find a path between the initial and final configurations. The generated roadmap is kept and modified after loading the second model.

In the first test, we use the smallest possible size for each cell. The size of each cell is 0.1m × 0.1m × 0.1m and each cell is composed of just one voxel. We increase the size of the cells in the next successive tests. In the sixth test, the size of each cell is 8 m × 8 m × 8 m and each cell is composed of 512000 voxels. The generated roadmap in the model is composed of 2999 nodes and 5996 edges. Table 3.2 illustrates the effects of cell size on the computation time and the memory usage.

Table 3.2: Memory performance.

Cell Size [m]	**Re-validated Nodes**	**Re-validated Edges**	**Revalidation Time [ms]**	**Memory [kB]**
0.1	107	242	71	301656
0.5	161	296	93	135004
1.0	234	580	118	131148
2.0	234	580	119	130644
4.0	447	1006	169	130076
8.0	1258	2636	355	129872

It is clear that by using bigger cells, we can save memory. On other hand, by increasing the cell size, the number of nodes and edges in each cell increases. So, the planner consumes more time to re-validates the components (nodes and edges) of the modified cells.

Based on our experiments, we believe that the efficiency of cell size depends on the problem. There are various parameters which affect this efficiency such as average number of modified voxels, robot size and voxel size. Therefore, we can not propose a cell size which optimizes the time and memory usage for all environments.

3.7.3 Experiment 3

we modified partially a model of the environment to simulate some changes in the environment. A model of LAAS-CNRS robotic laboratory is used in this experiment. Modification of 124 voxels in a model represents the displacement

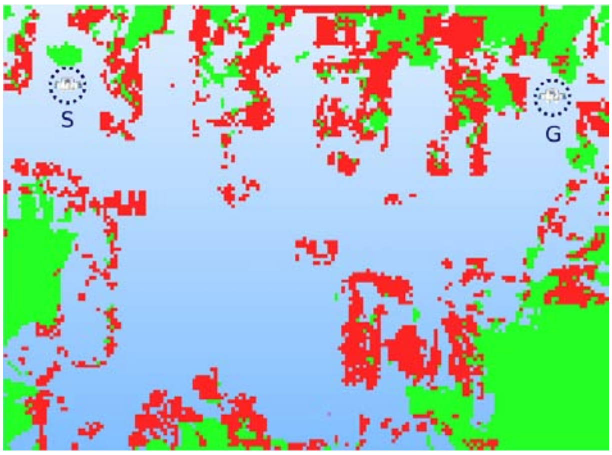


Figure 3.19: S and G represent the initial and final configuration of HRP2 in the model. Red and Green voxels represent the *occupied* and *unknown* region in the model.

of a table in the workspace. Figure 3.19 and 3.20 illustrate the initial and goal configuration and also the modifications in the model.

The models consist of 32787 *unknown* voxels, 7919 *occupied* voxels and 320544 *free* voxels. The size of each voxel is 0.1m × 0.1 m × 0.1 m. Also, the size of the cells that are used in this experiment is 2.0 m × 2.0 m × 2.0 m.

Table 3.3 presents the computation time for solving the problem by generating a roadmap in the model to connect initial and final configurations.

Table 3.4 illustrates the efficiency of the approach by presenting the required time for updating an existing roadmap in the environment and path planning by using the modified roadmap. In the first line, the planner uses the roadmap which is generated for solving the problem in the model B and updates it to plan a path in the model A. The second line represents the results of the inverse scenario.The planner updates the roadmap of model A based on the differences between the two models and then it plans a path in the model B.

Based on the results presented in table 3.4, the computation time for path planning is dramatically reduced by applying our approach.

3.8 Conclusion

We have presented an approach to deal with motion planning in changing environments. We extended an existing approach and we improved its limitations in terms of memory to adapt it to our problem. Although various approaches have been presented to deal with changing environments in various types of problem,

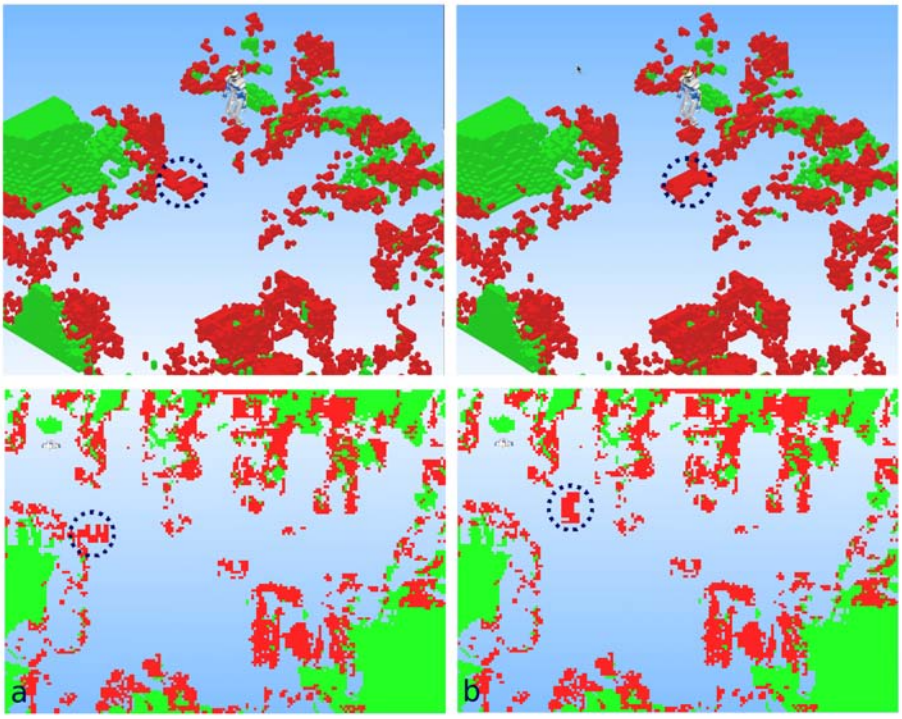


Figure 3.20: Modification in the status of 124 voxels represent displacement of a table in the model. Red and Green voxels represent the *occupied* and *unknown* region in the model.

Table 3.3: Path planning

Workspace	**Path Planning [ms]**	**Nodes**
A	198075	173
B	74401	74

Table 3.4: Path planning by applying the approach

Workspace	**Updating Roadmap [ms]**	**Re-validated Nodes**	**Path Planning [ms]**
B $\rightarrow A$	625	7	31739
A $\rightarrow B$	718	14	34603

we believe that our approach is highly efficient for navigating in large workspace. This approach is adapted for planning a path for the bounding box of the robot. Moreover, using two levels of discretization for workspace enable us to combine the algorithm with occupancy model which is used for modeling the environment(chapter 5).

Furthermore, the results illustrate the robustness of the approach in interacting with a large environment composed of thousands of voxels.

Chapter 4

Whole-body task planning

4.1 Introduction

Humanoid robots are highly redundant and complex systems. Planning collision-free stable motions for these systems is challenging for several reasons. First, most of them have a high number of degrees of freedom, which leads to the exploration of a highly dimensioned configuration space to find a collision-free path. Second, the motion we try to generate must satisfy many constraints: stability, physical capabilities, or even task constraints. Real-world task for a humanoid robot can include reaching to an object with a hand, as well as opening a door or a drawer.

This chapter presents a novel way to use random motion planning algorithms coupled with local inverse kinematic techniques.

4.1.1 Whole-Body task motion planning

The problem of inverse kinematics for a humanoid robot, or any articulated structure, is to compute a joint motion to achieve a goal task. As the robots we deal with are redundant, it is natural to take advantage of this redundancy by specifying multiple tasks, potentially with different priorities. This problem has been widely studied in robotics planning and control literature, and many jacobian-based solutions have been proposed, among which [2, 33, 50, 59]. Obstacle avoidance can be taken into account with similar local methods. To do so, one has to include the obstacles as other task constraints to be satisfied. A recent contribution on this subject is [30].

In our work, we have chosen to use the prioritized pseudo-inverse technique without taking into account the obstacles. The collision avoidance is externalized from the task set, following the paradigm of randomized motion planning, where the collision detection is used as a black box to validate sampled configurations.

4.1.2 Randomized motion planning

The other toolkit at our disposal is randomized motion planning. In the past fifteen years, several ways of randomly exploring the configuration space $\mathcal{C}$ have been successfully proposed [11, 32, 35, 36, 39]. These methods have been shown to be efficient for planning collision-free paths for highly dimensioned systems.

Their application to humanoid robots, however, is not straightforward. [34] proposes a whole-body motion planning method that deals with obstacle avoidance and dynamic balance constraints. It explores a set of pre-computed statically stable postures with a RRT-like algorithm and then filters the configuration space path into a dynamically stable trajectory. [61] deals with global motion planning under task constraints. The method is also an extension of the RRT algorithm, where the sampled configurations are projected on the sub-manifold of $\mathcal{C}$ that solves a task. The limit of this work is that it only considers very specific constraints, basically restrictions on a robot end-effector motion relatively to some world fixed frame. This is not sufficient to solve problems such as static stability, which can be seen as positioning an abstract end-effector -the center of mass of the robot- within a given region. A similar drawback is that it can not deal with multiple and prioritized constraints, whereas the corresponding local jacobian methods exist in the literature.

Finally, the problem we address is related with motion planning for closed kinematics chains. Some random techniques have been proposed to solve these problems [13, 40]. It differs from our work by the fact that the task is solved by random techniques rather than jacobian based optimization.

4.1.3 Contribution

The work presented in this chapter proposes a new way to use local jacobian based methods within randomized motion planning. The local methods are more powerful than the ones in [61], which makes our algorithm usable for solving humanoid whole-body motion planning. It differs from [30] and [31] by the way we treat obstacle avoidance, i.e. not in the optimization loop. The experimental section will show that it is a relevant choice.

Before going further and detailing our method, note that we only address the problem of finding statically stable *paths* for humanoid robots. To transform

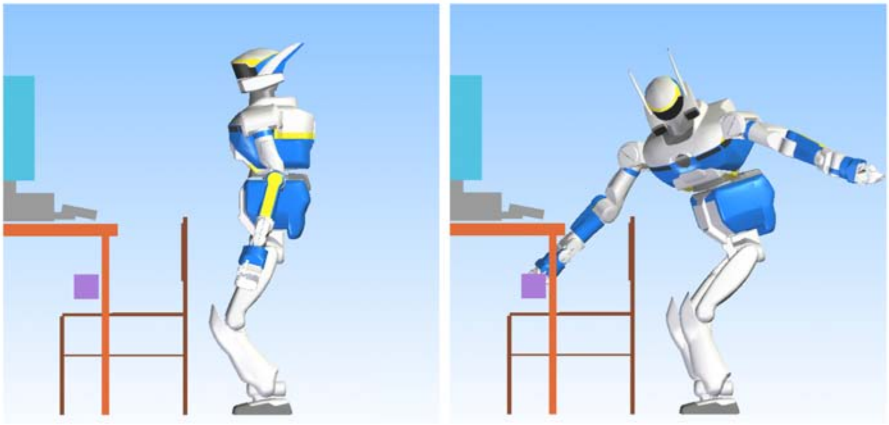


Figure 4.1: Reaching for an object in a constrained area

these paths into dynamically stable *trajectories*, we would need to use an other optimization stage, as it is presented in [34] and [31] for instance. As this second stage is independent from the first one, we chose not to deal with it here.

4.2 Preliminaries

4.2.1 RRT-Connect

This work considers humanoid whole-body motion planning problems. Because of the stability constraints and of the way the goal is expressed (as a task and not as a final configuration to achieve), we can not use classic generic motion planning algorithms. However, we did use an architecture similar to random diffusion methods, and more precisely to what is known in literature as RRT-Connect [35]. Let us remind briefly the structure of that algorithm.

Algorithm 1 shows the pseudo-code of the RRT algorithm. It takes as input an initial configuration q_0 and grows a tree $\mathcal{R}$ in $\mathcal{C}$. At each step of diffusion, it samples a random configuration q_{rand}, finds the nearest configuration to q_{rand} in $\mathcal{R}$: q_{near}, and extends $\mathcal{R}$ from q_{near} in the direction of q_{rand}. The original version of RRT-Connect depends on a step parameter ε. Each step of extension adds as many collision-free nodes to $\mathcal{R}$ as possible in direction of q_{rand}, each node being at a distance ε from the previous one. Fig. 4.2 shows one step of extension of the RRT-Connect algorithm.

4.2.2 Local path planning under task constraints

The tasks we consider for the robot can be expressed as a goal value for some function T of the robot configuration q: $T(q) = 0$. Assume the current configuration has a value $T(q) = c$. By computing the jacobian $J = \frac{\partial T}{\partial q}(q)$, one can calculate velocities $\dot{q}$ that tend to achieve the task. $\dot{q}$ is solution of the linear system:

$$J\dot{q} = -\lambda c \tag{4.1}$$

where λ is a positive real.

If the robot is redundant enough, i.e. if that linear system is under constrained, we can solve other lower prioritized tasks at the same time, by keeping $\dot{q}$ in the affine space solution of eq. (4.1). Iteratively changing q according to (4.1) is known as the Newton-Raphson method. Fig. 4.3 shows a 1-D example of a zero approximation by this algorithm.

The jacobian based engine for solving tasks will be referred to as the "local task solver". The precise implementation of this engine is described in [69]. During global planning, we will consider several types of tasks for various reasons:

- Static stability: the center of mass of the robot should stay at the vertical of the support polygon center; if we are planning a dual support motion, the two feet should remain at the same position and orientation. These tasks should always be achieved.

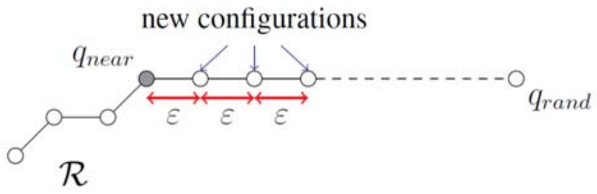


Figure 4.2: One step of extension of the RRT-Connect algorithm

loa 2: RRT-Connect(q_0)

```
$\mathcal{R}$.Init($q_0$)
for $i$ = 1 to $K$ do
    $q_{rand} \leftarrow$ Rand($CS$)
    $q_{near} \leftarrow$ Nearest($q_{rand}, \mathcal{R}$)
    Connect($q_{near}, q_{rand}$)
end for
```

- Goal achievement: the robot should at some point not only explore $\mathcal{C}$ but achieve some particular goal task as well. The ones we will present in the experimental section are reaching a particular point with the hand and open a door ; the hand should then stay on a given arc of a circle.
- Configuration space exploration: as we will explain in more details in the next section, we use configuration space defined tasks to explore $\mathcal{C}$.

4.3 Randomized task planning for a humanoid robot

This section describes the global whole-body motion planner. The core of it is a RRT-Connect algorithm. Some tasks have to be achieved for every configuration in the growing tree. This is the case for instance for stability tasks as well as for the position of the hand of the robot during a door opening motion. We will refer to these tasks as constraints. Next paragraph explains how we change the Connect function of the RRT algorithm to comply with these constraints.

4.3.1 Task constrained extension

Starting from a configuration q_{near} that respects a set of constraints, we want to go as far as possible in an other configuration (q_{rand}) direction, while keeping the constraints verified. To do so, we add a task to the local task solver, whose value is the distance, in $\mathcal{C}$, to a given configuration. It is referred to as $ConfigurationTask$ in algorithm 3. This task is added with the lowest priority. The configurations we try to reach are successively the ones a classical RRT would have added to the tree. Fig.4.4 shows one step of constrained extension.

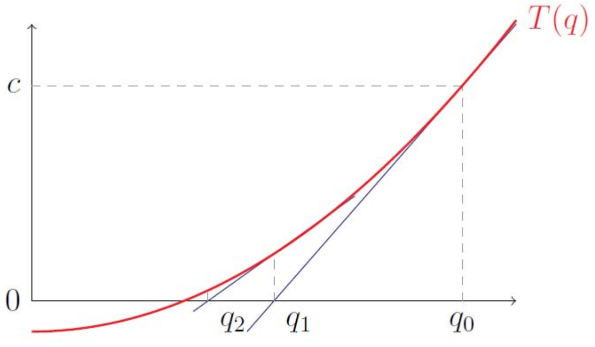


Figure 4.3: Root finding using Newton-Raphson method

loa 3: Constrained-Connect($\mathcal{R}, q_{near}, q_{rand}$)

```
$Tasks$.Initiate($Constraints$)
$Tasks$.Add($ConfigurationTask$)
$\Delta_q \leftarrow \varepsilon.(q_{rand} - q_{near})/||(q_{rand} - q_{near})||$
$q_{target} \leftarrow q_{near} + \Delta_q$
$q_{current} \leftarrow q_{near}$
$State \leftarrow$ Progressing
while $State =$ Progressing do
    $ConfigurationTask$.SetTarget($q_{target}$)
    $q_{new} \leftarrow$ LocalPlannerPerformOneStep($Tasks, q_{current}$)
    if $q_{new} \neq q_{current}$
    and CollisionCheck($q_{new}$) = OK
    and CollisionCheckEdge($q_{current}, q_{new}$) = OK
    and ConstraintsCheck($q_{new}$) = OK then
        $\mathcal{R}$.AddNode($q_{new}$)
        $\mathcal{R}$.AddEdge($q_{current}, q_{new}$)
        $q_{current} \leftarrow q_{new}$
        if $q_{target} \neq q_{rand}$ then
            $q_{target} \leftarrow q_{target} + \Delta_q$
        else
            $State \leftarrow$ Reached
        end if
    else
        $State \leftarrow$ Trapped
    end if
end while
```

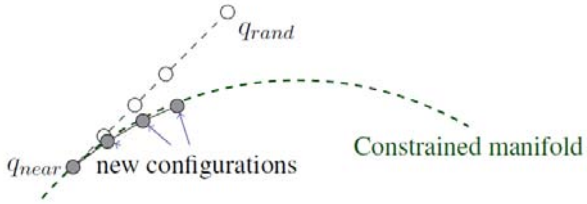


Figure 4.4: One step of constrained extension

After calling the local task solver, we check that the configurations respect indeed the constraints, that they are collision free, and that the edges linking them together are collision free as well. Algorithm 3 shows pseudo-code for one call to the Constrained-Connect function.

4.3.2 Goal configuration generation

One main difference between the problems we consider and classic motion planning is that the goal configuration is not defined explicitly. Instead, the input of the planner is a given task in the workspace. The output of the planner should therefore be a statically stable, collision-free path between the initial configuration and a configuration that solves the goal task. There are many ways to adapt motion planners to do so. One could be to grow a tree in $\mathcal{C}$ and try from time to time to apply the goal task to a newly generated configuration, hoping that the local task solver will return a collision-free path that solves the problem. This option is available in our planner, simply by changing the task to solve in the Constrained-Connect function. However, we found it more efficient to generate random goal configurations, using the local task solver, and then express the problem as a classic motion planning one, with one initial configuration and several goal configurations. The reason is that once we have defined explicit goal configurations, we can root random trees at those configurations. The idea of growing a tree rooted at the final configuration was first proposed in [35].

The way we generate a goal configuration is the following:

1. Shoot a random configuration in $\mathcal{C}$.
2. Call the local task solver on this configuration, with the static stability constraints and the goal task.
3. Check that all of those tasks are achieved.
4. Check for collisions.

Fig. 4.5 shows different random goal configurations respecting stability and goal constraints.

Probabilistic completeness Guaranteeing probabilistic completeness is not as straight forward for task planning as for classical configuration to configuration planning. The reason is that we do not know explicitly the shape of the solution manifold. Some solutions may not be in the same connected component as the initial configuration while some other are. If we choose to first shoot random goal configurations, we must ensure that any neighborhood of the solution manifold has a positive probability of being reached. This is the case with our way of shooting the goals configurations. Reciprocally, if a collision-free, statically stable solution path exists, we have a positive probability of shooting a goal configuration in the neighborhood of that path's final configuration, then

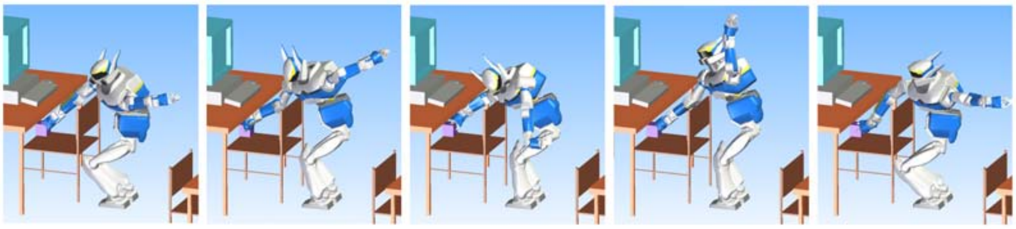


Figure 4.5: Random goal configurations solving the reaching task. All the configurations are stable and collision-free.

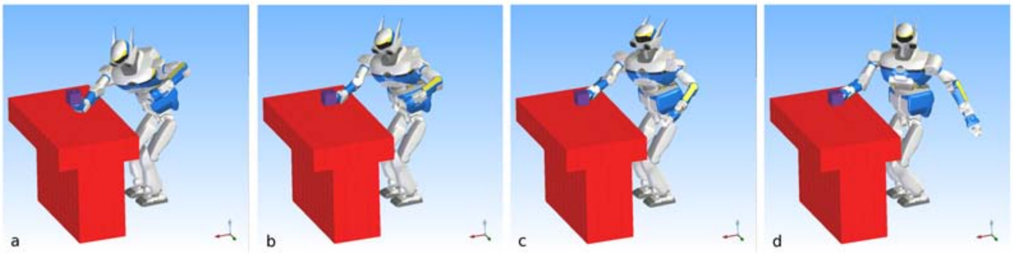


Figure 4.6: Different steps of a posture optimization. All the configurations respect the stability and goal constraints, and the configurations are more and more natural.

a positive probability of growing a branch of the tree in a neighborhood of the path. This makes our algorithm probabilistically complete.

4.3.3 Posture optimization

As in classic motion planning, once we have found a random solution path, it is important to optimize it. The criteria to optimize depend on the system. In humanoid robotics, the motion should look “realistic” or “natural”, for any good definition of these concepts. As the goal configuration we have reached as the end of the solution path was generated randomly, we need to optimize it as well. Note that we only optimize it after having found a solution path because we do not want to optimize useless goal configurations.

To optimize the posture, we use a random gradient descent algorithm, starting from the configuration to optimize. The local task solver ensures that all the constraints (stability and goal task) are achieved and is repetitively called to generate collision-free random displacements that minimize a certain “natural” cost function. In the presented examples, that function was the $\mathcal{C}$ distance to a reference posture. The new solution path is the concatenation of the previous solution and the path resulting from the posture optimization.

Fig. 4.6 shows the different steps of a goal posture optimization.

4.3.4 General architecture

Now that we have all the tools at our disposal, the architecture of the planner can be explained.

1. First we generate several goal configurations.
2. We search for a path between the initial configuration and one of the goal configurations with a RRT algorithm. We can either grow a single tree rooted at the initial configuration or several other trees rooted at each of the goal configurations.
3. If a path is found, we optimize the goal posture that has been reached.
4. We optimize the concatenation of the solution path and the posture optimization with classic random motion planning path optimization methods.

During the RRT search and the final path optimization, we use the task constrained extension detailed earlier.

4.4 Work space analysis

Goal configurations are generated in two steps. In the first step, we shoot a random position and orientation for the robot and then we generate the whole body configurations that satisfy the desired goal tasks.

Selecting an appropriate position for the robot is an important issue. It reduces the process of goal configuration generation and also results in configurations which look “realistic” or “natural”.

Figure 4.7 illustrates HRP2 in two different configurations for grasping a ball by the right hand. Although, the both configurations satisfy the required task as well as the stability constraints, configuration (a) looks more “natural”.

These configurations are generated by placing the robot in two different positions. Then, the robot is initialized in a random configuration and the inverse kinematics model is solved to satisfy the task of grasping by the right hand.

Moreover, for some tasks such as grasping an object, when there is no priority for choosing the hand, choosing the proper hand is an important issue. For grasping an object, choosing the active hand for grasping optimizes the goal configuration and also increases the accessible workspace.

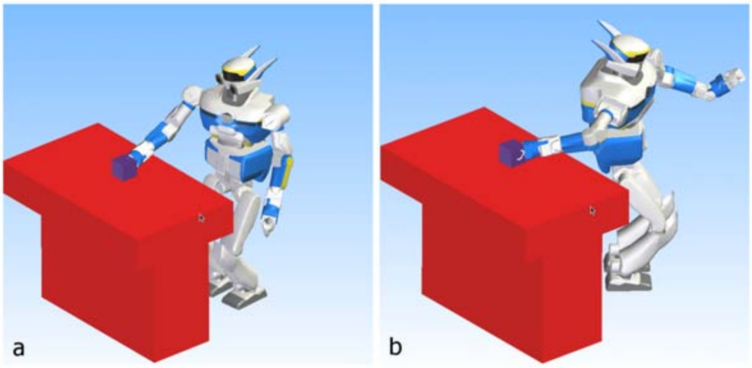


Figure 4.7: HRP2 in two configurations: The both configurations satisfy the task of grasping an object with the right hand. Configuration (a) looks more “natural”.

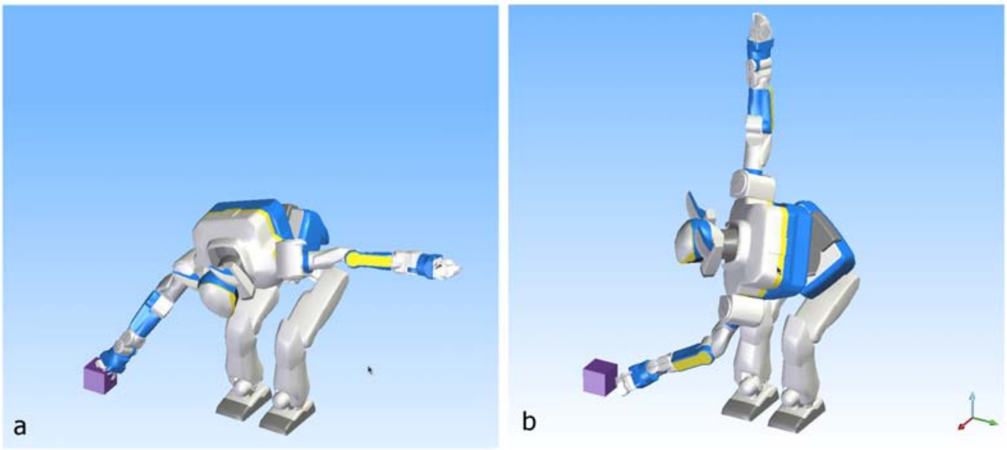


Figure 4.8: By considering the constraint of fixed foot position, the target is not accessible by the left hand.

Figure 4.8 illustrates a simple situation where the target for grasping is not accessible for the left hand.

For implementing our goal generator, we did some analyses on whole body configurations for the task of grasping an object.

We generated a number of random configurations which satisfy the stability constraints while keeping the feet at the same position. On the other hand, the workspace is discretized and the position of each hand is noted for each configuration and stored in the corresponding cell. This grid stores the information of hand position density in each cell for a number of random stable configurations. This data can be used for choosing an efficient position for the robot feet or selecting the active hand for grasping an object.

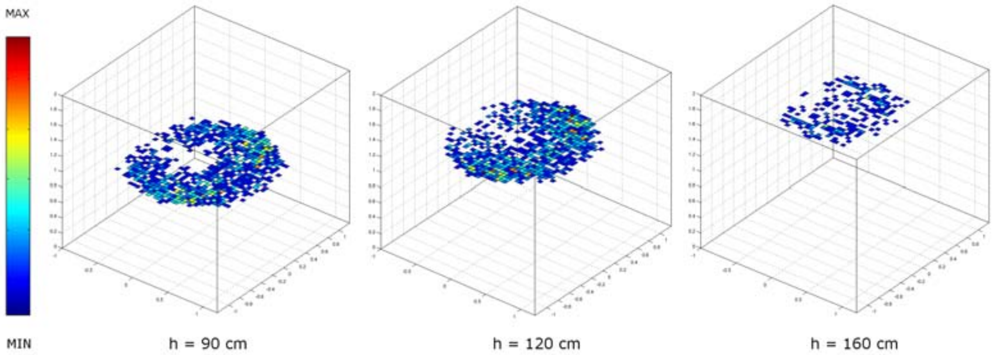


Figure 4.9: The density of hand positions for a number of random configurations.

Figure 4.9 illustrates the density of hand positions in the conducted survey. Moreover, the accessible volume of workspace for grasping is shown in this figure.

Figure 4.10 illustrates the same information in 2D. The density of HRP2 hand positions is illustrated for each layer along the Z axis.

Such analysis can be conducted for other required tasks in order to develop a specific goal generator.

4.5 Examples and comparison with previous methods

This section presents experimental results of our whole-body motion planner on simulations on the robot HRP2. We compared it thoroughly with the local collision avoidance method presented in [30] and [31] on two different examples of hand reaching motion. One is a dual support motion, where the humanoid robot has to reach an object under a table, and the other one is a more complicated single support motion, where the robot should reach a point inside a large torus shaped obstacle. These comparisons were made to evaluate the relevancy of avoiding obstacles at a global scale rather than inside the optimization loop.

In [30] and [31], the robot and its environment are modeled by non-strictly convex polyhedra. A local collision avoidance method for non-strictly convex polyhedra with continuous velocities is used to solve motion planning problems. The problem of the continuous interaction generation between polyhedra is reduced to the continuous constraints generation between polygonal faces. A collision-free motion is obtained by solving an optimization problem defined by

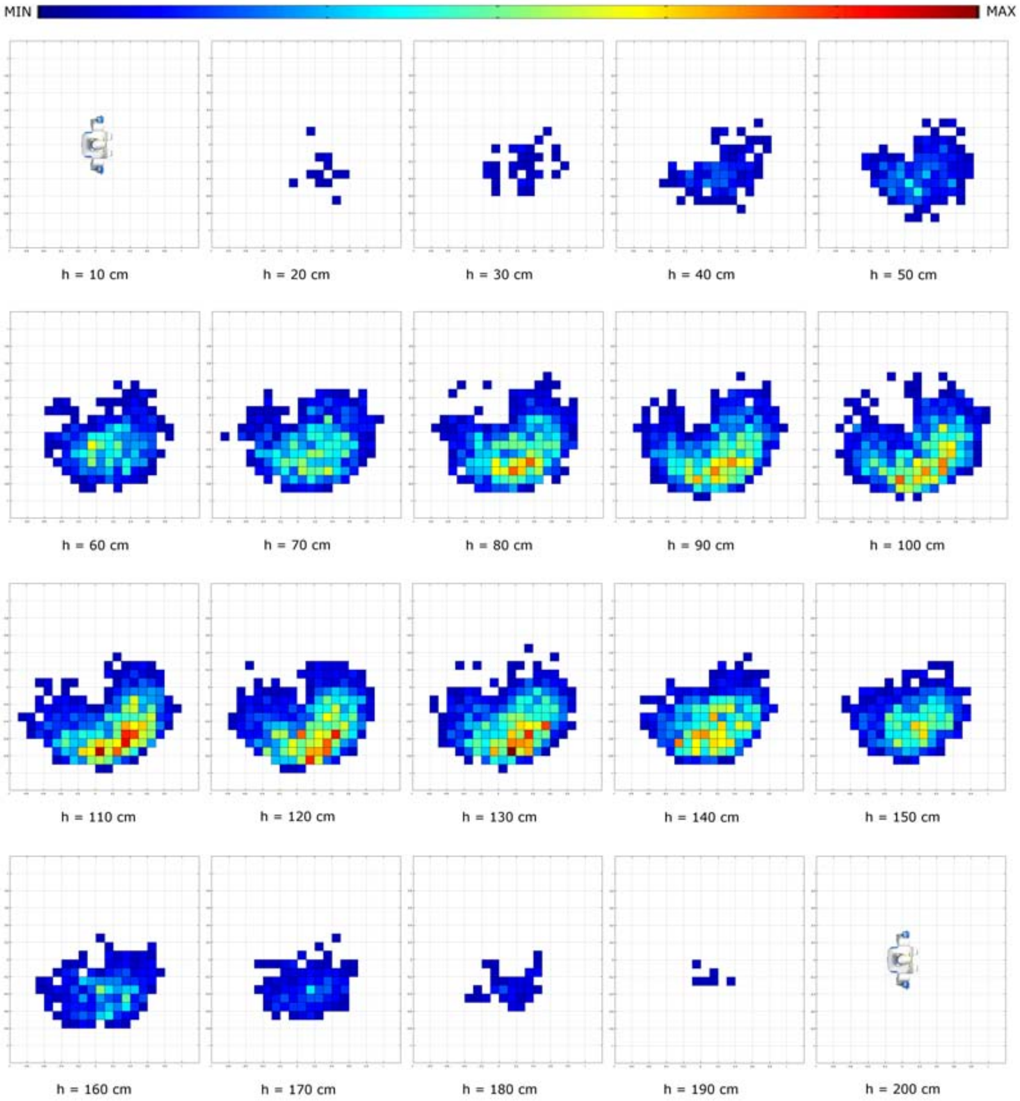


Figure 4.10: The density of hand positions for a number of random configurations. The density is illustrated for each layer along the Z axis.

Table 4.1: Computational time [ms] for the presented scenarios.

	Goal Generation	Path Planning	Posture Optimization	Path Optimization	Local Planner	Collision Detector
Table	281	4,017	16,181	5,096	1.25 %	86 %
Torus	1,689	27,965	40,479	23,851	9.19 %	84 %

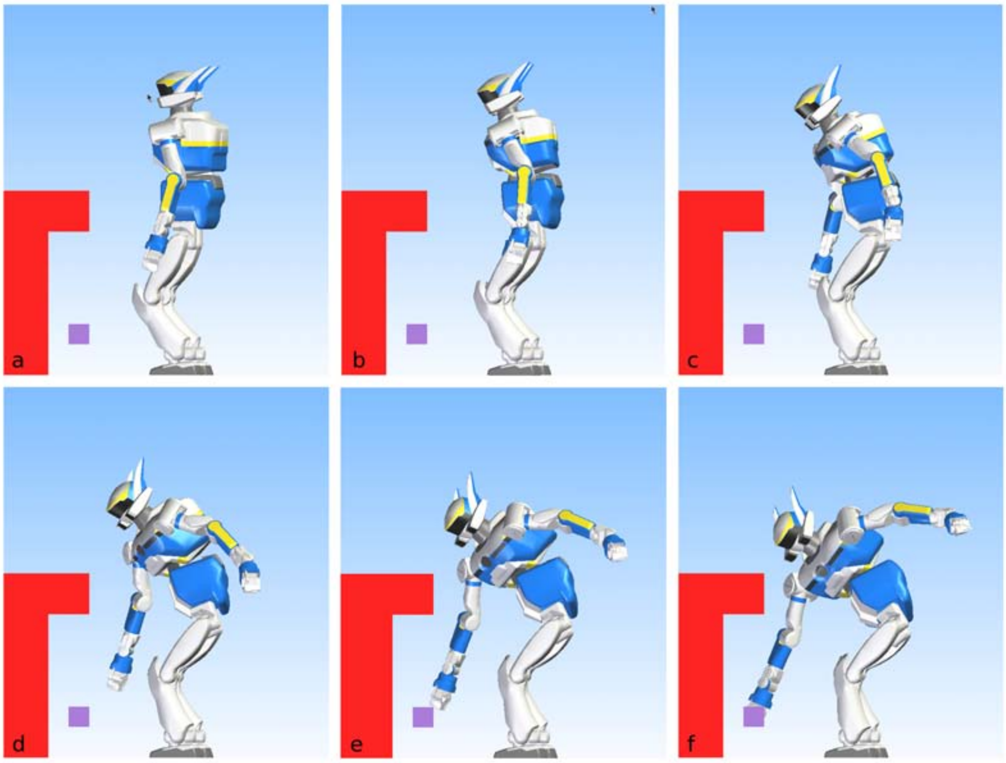


Figure 4.11: Solution path for “Table” scenario.

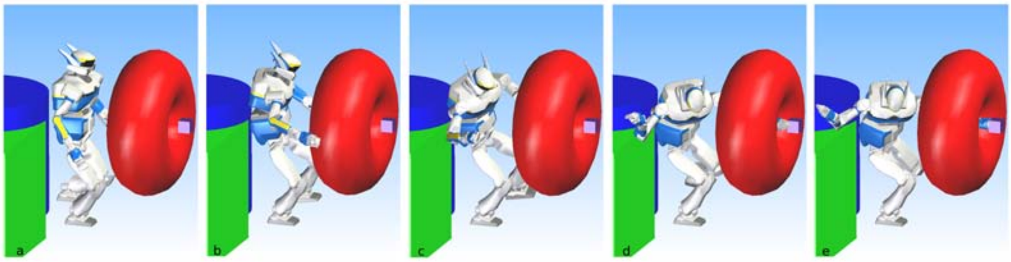


Figure 4.12: Solution path for “Torus” scenario.

an objective function which describes a task and linear inequality constraints which do geometrical constraints to avoid collisions.

The implementation of our algorithm uses KineoWorks$^{\text{TM}}$[38] implementation of random diffusion algorithms and collision checking. All the simulations were performed on a 2.13 GHz Intel Core 2 Duo PC with 2 GB RAM. Note that the figures taken from [30] and [31] were obtained on a similar PC, so time comparisons are relevant here. For each problem, we ran the planner ten times: this includes goal configuration generation, path planning, posture optimization and whole-body path optimization. We indicated the cost of each of these computations, as well as the relative costs of the collision detector and the local task solver. The last costs are expressed as percentages of the total computational time. They do not add up to 100%. The rest of the time is spent in roadmap and environment management.

4.5.1 Dual support ”Table” scenario

Fig. 4.11 shows the solution path for a problem of reaching an object under a table. A similar experiment was presented in [30]. The local method presented in [30] computes one step of optimization in about 100 ms for this problem, which means that it can run at half the speed required for real time when selecting a time step of 50 ms. The trajectory lasts 14 s so the planning time is about 28 s.

Table 4.1 shows the time taken by our planner. The average total time, including both planning and optimization, is 25.6 s. Note that most of the time is spent with posture and path optimization. Problem solving itself (goal generation and path planning) only lasts 4.3 s. However, optimization is mandatory since we are using random path planning method. It is indeed the total time that should be compared to the 28 s found in [30].

4.5.2 Single support “Torus” scenario

Fig. 4.12 shows the solution path for a more complex problem. The robot is on one foot and has to reach a point inside a large torus shaped obstacle.Table 4.1 shows the time taken by our planner. This problem was presented in [31]. It is a very difficult problem for a local method dealing with obstacle avoidance because of the complexity and proximity of the obstacles. When computing this motion with the precise models of both the robot and the torus, the local method for obstacle avoidance takes 1.47 s per step. The reason is that there are 3460 constraints induced by the collision avoidance. The article presents simplified models of the robot and the torus, which leads to a computational time of 50 ms per step (375 constraints). The generated trajectory lasts 77 s (before dynamical

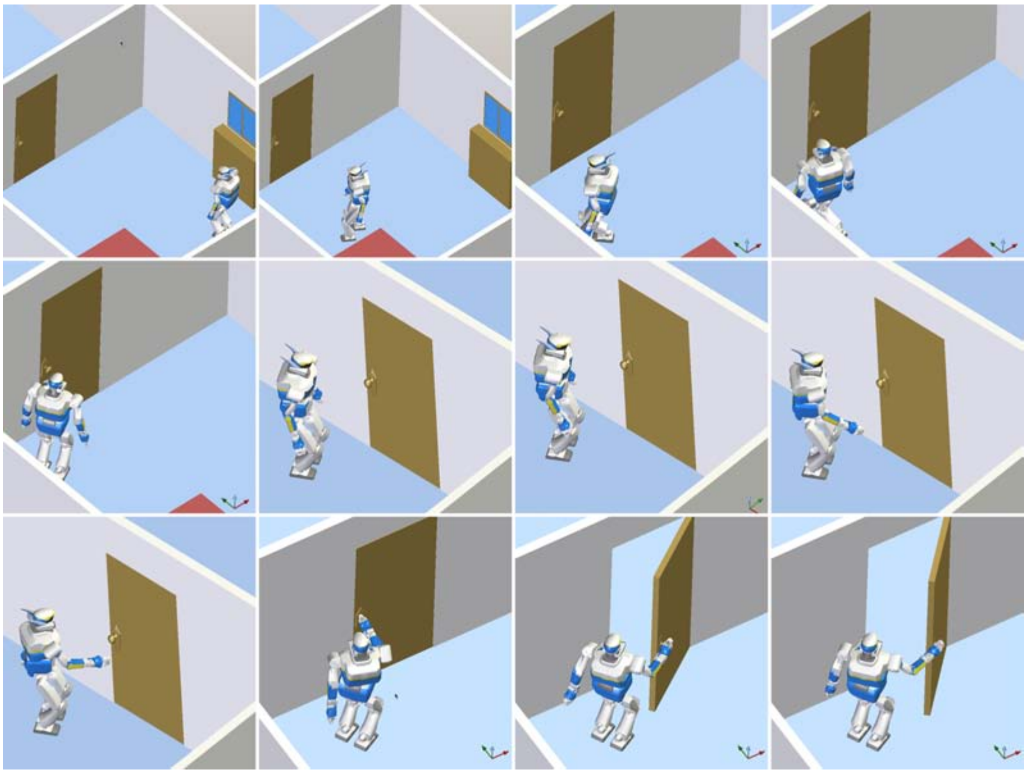


Figure 4.13: Solution path for a complete resolution of a door opening problem. The walk motion at start was generated separately with a walk planner. We then successively solved a hand reaching problem to grab the handle and a specific door constraint problem to keep the hand on a given arc of a circle.

optimization). The planning time is then 77 s for the simplified models and 1540 s for the precise models.

Table 4.1 shows the time taken by our planner. The average total time is 94.0 s and the problem solving itself is 29.7 s. Once again, most of the time is spent optimizing the results. Comparing to the previous scenario, the local task solver is comparatively more costly for this problem, one explanation might be that the stability constraint is more difficult to compute for single foot support.

4.5.3 When to deal with obstacle avoidance

It is difficult to compare our planner with a local collision avoidance method because they are not intended for the same use. The local collision avoidance method can be used as a controller, while our work is really about offline planning. However, what we can say is that the cost of collision avoidance constraints make the local method unusable on complex examples. On the other hand, our probabilistic planner can be used to compute -in a reasonable time- a statically stable, collision-free path that could be executed afterwards by a controller. Note that it does not mean that local collision avoidance methods are hopeless, but for now, they can only be used on simplified or dedicated models. For instance, if the robot and the obstacles were modeled by spheres or smooth curves (with analytical formulas for distance computation) rather than triangles, the number of constraints induced by collision avoidance would decrease and the computational time would be drastically reduced.

4.6 Conclusion

In this chapter, we have presented a novel whole-body planning method for humanoid robots. It uses a local task solver to generate valid configurations within a random diffusion algorithm framework. Its advantages are the generosity of the local task solver, that makes the planner usable for a large variety of problems -Fig. 4.13 shows a complete example of a door opening problem, and the efficiency of random diffusion algorithms for highly dimensioned problems. In this example, the intermediate robot feet position for opening the door was chosen automatically. This position was computed deterministically based on the geometry of the door. We have compared our approach with a local method for collision avoidance. The results show that for complex scenarios, the planning time can be more than an order of magnitude lower with random planning than with local obstacle avoidance.

Chapter 5

Environment modeling

5.1 Introduction

As explained in the previous chapters, the context of this work is vision-based motion planning for humanoid robots in an unknown environment. We presented an efficient method of motion planning in non-static environments and whole body task planning.

In this chapter, we present an implementation of stereo vision environment modeling and also our method for finding the next best view in each instance of the model.

Robotic mapping has been an active area in AI for a few decades. It addresses the problem of acquiring a spatial model of the workspace through available sensors on a robot. We can distinguish two main approaches addressing slightly different problems :

1. Sparse 3D mapping aims at building a geometrically coherent sparse map of characteristic features and localizing the robot in this map. Recent work in this area include [14, 41].

2. Dense SLAM aims at building a dense map of the environment integrating obstacles. These techniques are usually based on occupancy grid methods. For more references and details about SLAM, please refer to [65].

Pioneering work on occupancy grids is described in [16, 49]. Later studies define occupancy grid in a probabilistic framework [12, 63]. Moreover, [62] proposed a very impressive metrical approach.

An occupancy Grid Map maps the environment as an array of voxels. Each voxel holds a probability value that the voxel is occupied. Occupancy map is useful for combining different sensor scans, and even different sensor modalities such as sonar, laser, IR, etc.

In this chapter, a simple occupancy grid approach is presented to generate a model of the environment. The humanoid HRP2 is supposed to generate a model of the environment based on stereo vision in order to plan collision free motions in workspace. As the robot navigates in the environment, it receives updated information through its on-board cameras and refreshes the 3D model of the environment incrementally. HRP2 is supposed to plan and drive to multiple positions to efficiently construct the model.

Finding the next best view (NBV) that improves the current model is another important issue. In literature, finding the NBV also addresses the problem of object reconstruction where the exterior of the objects can be seen.

The traditional NBV algorithms assume that the sensor head can freely move around some object [3]. But in reality, various constraints may limit the movements of the sensor. Also, the sensor is inside the scene and the accuracy of the pose is limited.

The calculation of viewpoints, i.e., the placement of the 3D sensor to generate a model of the environment, is similar to the art gallery problem. The problem states given a map of a building, what is the optimum placement of guards such that the whole building is supervised [54]? Various approaches have been proposed for finding the NBV.

J.M Sanchiz *et al.* present an approach to full 3D scene recovery by a range sensor mounted on a mobile robot. The recovered scene is modeled by a voxel map [57].

Andreas Nuchter *et al.* presented an approach for planning the next scan pose as well as the robot motion. Their approach calculates a collision free trajectory in presence of complicated objects [51].

T. Foissotte *et al.* implemented an algorithm for generating the next best posture for a humanoid robot. This algorithm aims at building autonomously visual models of unknown objects, using a humanoid robot [21].

In this chapter, after presenting the approach of occupancy grid methods, we are going to present our algorithm for finding the next best position for HRP2. Based on the limits of stability and whole body collision free motion, we rely on a simple goal based algorithm which returns the next efficient 2D position of HRP2 (x, y , θ) as a collision free reachable NBV.

5.2 Environment modeling

5.2.1 Occupancy grid map.

In our approach, a 3D occupancy grid map represents the environment. The occupancy grid representation employs a multidimensional tessellation of workspace into voxels, where each voxel stores a probabilistic estimate of its state. The environment is discretized into uniform cubes which are the smallest volume of our 3D model. We use the existing approach of modeling the environment to generate the required model for planning [16].

The size of the cubes is selectable based on our desired resolution in the model. Based on the volume of the 3D model, it can be composed of millions of voxels.

$$\mathcal{W} = \bigcup_{i=1}^{n} V_i, \forall i, j \in [1, n], V_i \cap V_j = \emptyset \tag{5.1}$$

where $\mathcal{W}$ is the workspace and V_i presents the voxel i in the model. The state variable associated with a voxel V_i is defined as a discrete random variable with three states: *unknown*, *occupied* and *free*.

$P[S(V_i) = Occ]$ is the probability of voxel i to be *occupied*. This probability has a range from 0 to 1. A probability of 0 means that the voxel is *free* and a probability of 1 shows the highest possibility of being *occupied*. Logically, probability of 0.5 is the lowest information about a voxel and such a voxel is classified as *unknown*.

The robot obtains information from its environment through its cameras and uses a stereo vision method to get a range value r for a 3D point in the environment. A stochastic sensor model defined by a probability density function $p(r|z)$ is used to relate the reading r to a true parameter space range value z.

Based on the recent probability density, $P[s(V_i) = Occ]$ is modeled for each voxel.

For certain applications such as motion planning, it is necessary to assign a specific state to each voxel of the occupancy map. We used two optimum thresholds to assign a state to each voxel. Based on these thresholds, probability in a range around 0.5 represents the *unknown* zone. For a probability more than the upper threshold, the voxel is considered as *occupied* and as *free* zone for a probability less than the lower bound.

For initializing the model, all voxels are considered as *unknown*. Therefore, the grid map is initialized with a probability equal to 0.5. The model is updated incrementally as the robot explores the unknown environment.

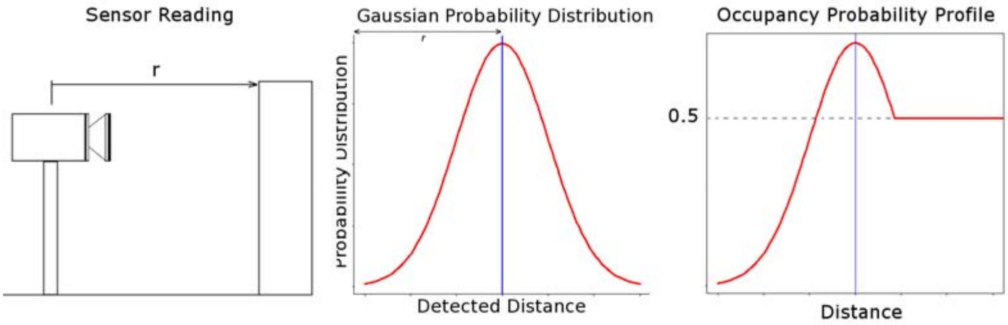


Figure 5.1: Estimating the occupancy probability profile.

5.2.2 Sensor model

As explained earlier, a sensor model is used to relate the reading r to a true parameter space range value z and finally, to obtain the concerned probability for each voxel. In fact, in this approach, the cameras are modeled with simple Gaussian uncertainties in depth as follow:

$$p(r_j|z) = \frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-(r_j - z)}{2\sigma^2}) \tag{5.2}$$

where j is the index of each 3D point of the stereo-image. Figure 5.1 illustrates a simple model which is used to obtain the probabilities.

5.2.3 Updating 3D model

To allow the incremental composition of sensory information, the sequential updating formulation of Bayes' theorem is used to determine the probability of each voxel [15]. Moreover, using these algorithms preserves the dynamic characteristics of the model. In cases where an obstacle enters into or exits from the environment, the probabilities of the concerned voxels change rapidly and their states are modified. Extending the approach of [16] to the model, and considering the current probabilities of V_i as $P_i(s(V_i) = Occ \mid r_t)$ based on observations $\{r\}_t = \{r_1, r_2, ..., r_t\}$ and given a new observation r_{t+1}, the updated estimate will be given as follows:

$$\begin{aligned} & P[s(V_i) = Occ \mid \{r\}_{t+1}] = \\ & \frac{P[r_{t+1} \mid s(V_i) = Occ].P[s(V_i) = Occ \mid \{r\}_t]}{\sum\limits_{s(V_i)} P[r_{t+1} \mid s(V_i)].P[s(V_i) \mid \{r\}_t]} \end{aligned} \tag{5.3}$$

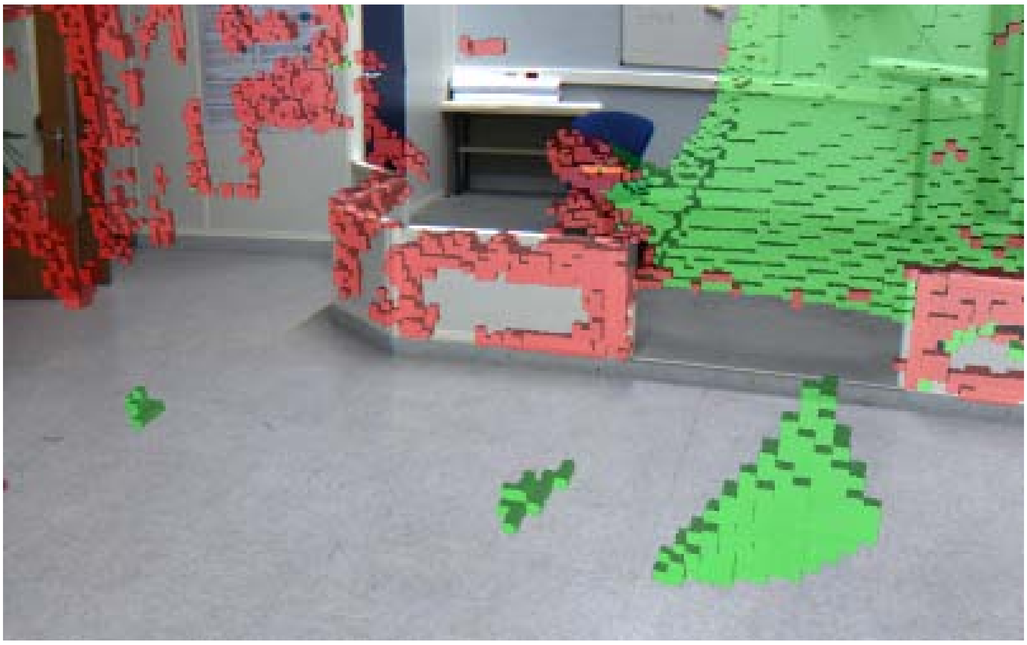


Figure 5.2: The robotic lab in LAAS-CNRS which is modeled based on this approach. Red and green voxels (cube) represent *occupied* and *unknown* volumes respectively in the model. The work space is modeled with 3D voxels of 5cm×5cm×5cm. Notice that this model is the result of several stereo-images taken from different points.

At each step of updating the model, the current probabilities are obtained from the occupancy grid map. Moreover, at the end of calculations, results are stored in the voxels. These stored probabilities are used as the initial probabilities for the next update. Figure 5.2 illustrates an environment which is modeled by stereo vision.

It should be mentioned that localization plays a critical role in environment modeling. In fact, as the data obtained from sensors are read in the sensor coordinate frame, localization of sensors in the model is important for having a precise model.

At the following step, the generated environment model is used for motion planning. The robot moves in the free zone and incrementally improves the occupancy grid map and explore the unknown area.

Figure 5.3 illustrates the incremental exploration of an unknown environment.

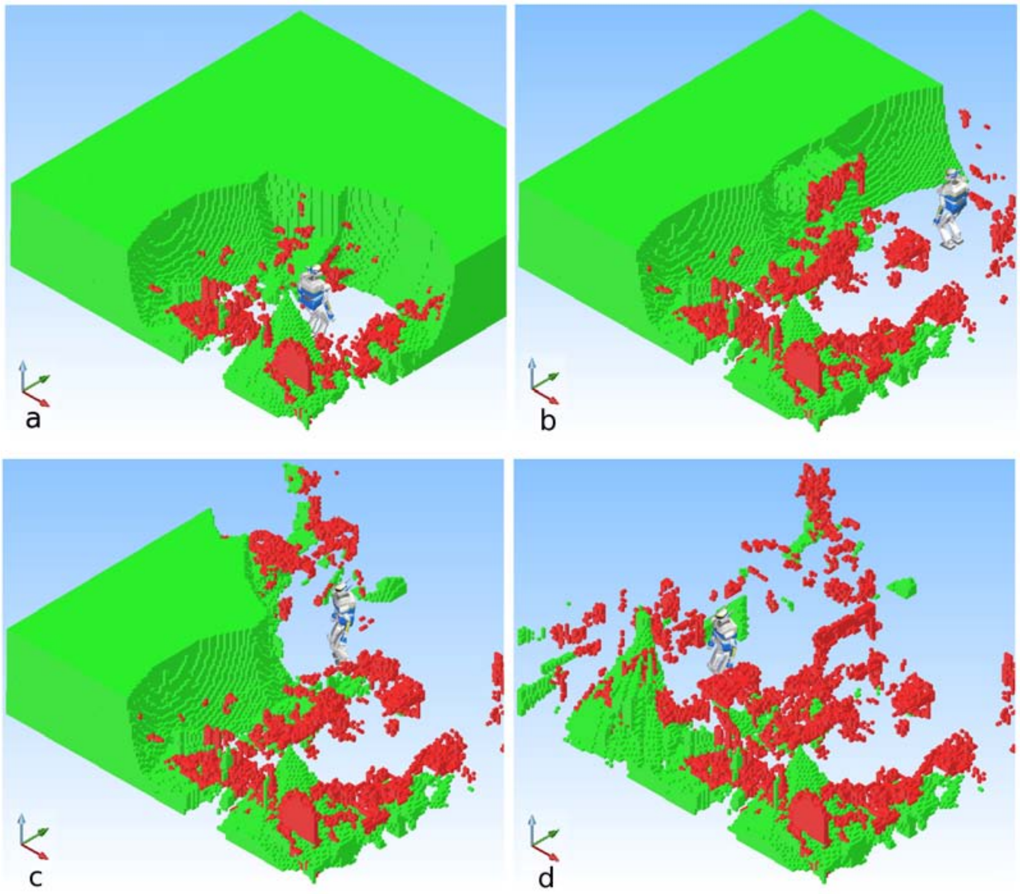


Figure 5.3: Exploring an unknown environment. Red and green voxels represent *occupied* and *unknown* zones in the model. The robot plans a path in *free* zone.

5.3 Next best view

Automatic map building is an important problem in robotics. Research in this area has traditionally focused on developing techniques to extract environmental features and currently on simultaneous localization and mapping (SLAM). But an important question at the time of constructing a model is "where should the robot move next to observe the unexplored regions?"

The answer involves the computation of successive sensing locations by iteratively solving the NBV problem. At each location, the robot must not only observe large unexplored areas of the environment, but also a portion of the known environment to allow the robot to connect to the known environment. NBV is complementary to SLAM. A SLAM algorithm builds a map by making the best use of the available sensor data, whereas a NBV algorithm guides the robot through locations that provide the best possible sensor data.

NBV is an on-line version of the sensor placement problem, where the 2-D map of the environment is unknown initially and only revealed incrementally as new sensor data are acquired.

5.3.1 Constraints on the next best view

Various algorithms were proposed and implemented to find the NBV. But in our case, we should consider some constraints before integrating an algorithm on HRP2.

The first constraint is that the robot must not collide with known obstacles or unknown area. The second is the humanoid robots stability. Stability constraints are important limits in choosing a next best view. Also, generating stable motion to reach the next best configuration limits the space of results.

Moreover, generating one connected *free* area instead of various smaller *free* zones is important. Otherwise, based on the limitations of our planner, the robot is obliged to stay in one *free* connected zone. Our planner plans a path for the bounding box of the robot. The bounding box slides in the workspace and it can not jump over the unknown zone.

5.3.2 Evaluating the next best view

Suppose that at the time t, the robot is positioned at $\mathbf{q}_t$ and the free configuration space is $\mathcal{C}_{free}$. The goal is to compute the future best configuration of the robot to explore the unknown zone or arrive at a goal configuration. The first fact is that the unexplored areas of the environment can only be revealed from a configuration in the *free* zone.

Moreover, this best configuration is logically located somewhere around the boundary of the known *free* zone. This boundary has a high potential visibility gain compared to the middle of the *free* zone.

Also, in case of having a geometrical target in our model, the NBV must favor an approach towards this goal location.

The computation of the NBV must also have a factor in the cost of motion, which is weighed against the potential visibility gain.

5.3.3 Computing the next best view

At this point, the only remaining issue is to search for the NBV. Based on our limits and problem constraints, the problem is simplified. Similar to our planner

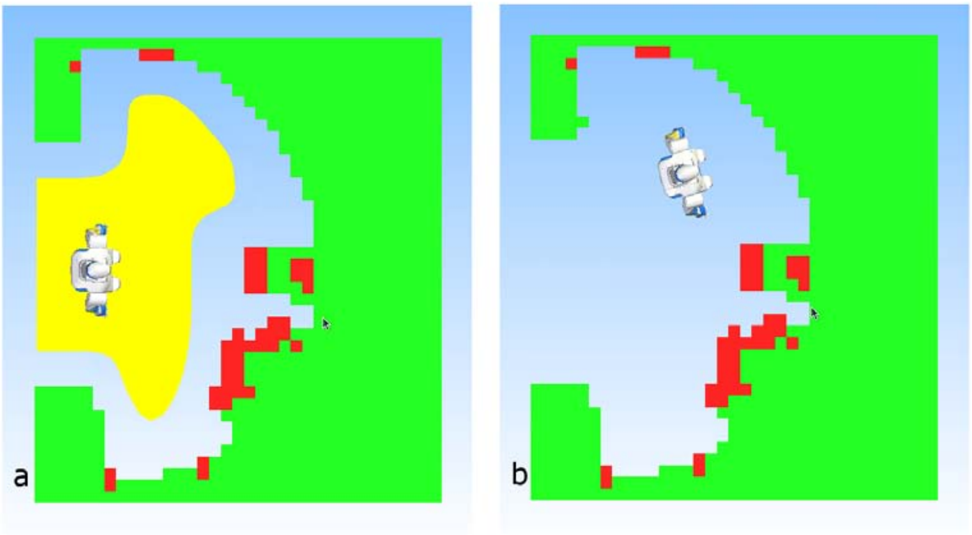


Figure 5.4: The safe zone in which we look for the NBV (Schematic).

in navigation, we work with the bounding box of the robot instead of considering all the degrees of freedom. To find non-collision configurations inside the *free* zone, we use a random-sampling method to generate a number of accessible configurations for the bounding box (x, y, θ) of the robot. The planner generates a roadmap in the *free* zone (safe zone) with a predefined minimum number of nodes.

Moreover, a set of head configurations are pre-registered to be executed for taking photos and updating the model at each position of the bounding box. In other words, a number of predefined head configurations are assigned to each bounding box position and will be executed automatically after each displacement of the bounding box. These configurations and the whole body motion between each of them are generated online by a generalized inverse kinematics (GIK) package that guarantees the stability of the robot [71]. This approach addresses the question of stability and also accelerates the process.

Based on the set of head configurations and camera properties, it is simple to find a distance threshold between the robot bounding box and unknown zone to guarantee an overlap between the consecutive explorations. This threshold defines a zone inside the *free* area which addresses the question of having overlap between the explorations. Figure 5.4 illustrates our safe zone in which we look for the next best view.

Based on the camera properties and the set of head configurations, an algorithm was developed to estimate the potential visibility gain. This potential

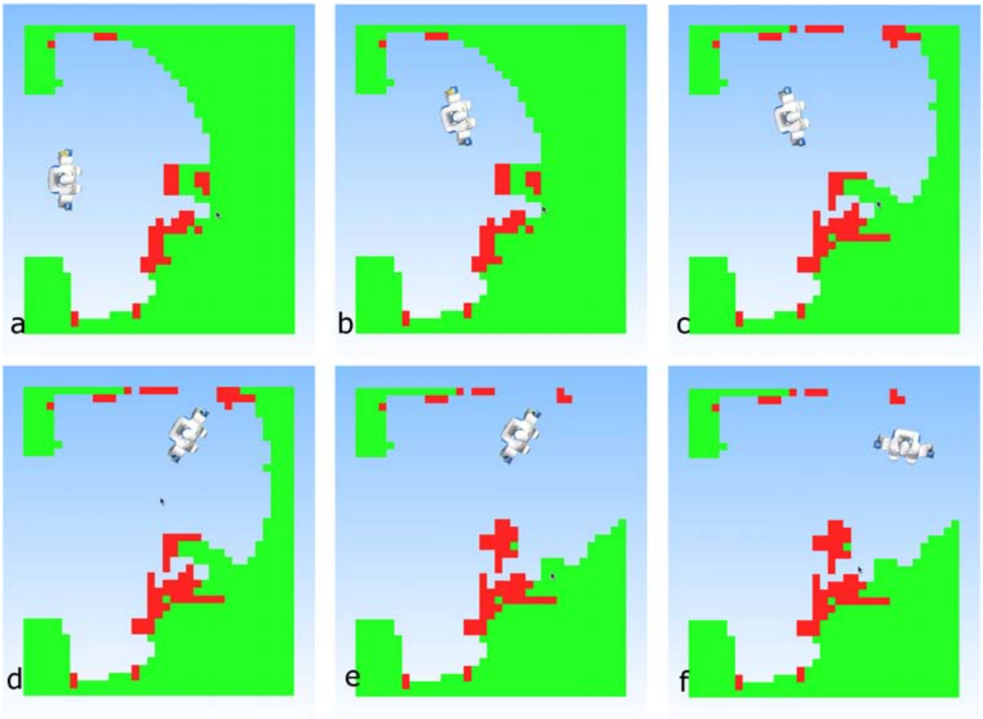


Figure 5.5: The consecutive NBV in exploring the environment.

visibility gain is roughly the number of *unknown* voxels which are located in the visible zone. Also, the voxels which are located in the target zone are favored.

Therefore, this algorithm returns a random accessible configuration for the bounding box with a higher potential visibility gain that guarantees the overlap of the exploration.

5.4 Conclusion

We presented the integration of an algorithms for environment modeling. The approach is an implementation of an existing method. We do not considered this part of our work as a contribution, but, for having an autonomous robot, we developed and integrated the method to link it to our planner to deal with unknown and changing environments.

We also developed a simple algorithm which returns a NBV that fulfills our requirements to explore the model and navigate toward a goal zone. The experiments for generating the model and looking for the NBV are presented in detail in the following chapter.

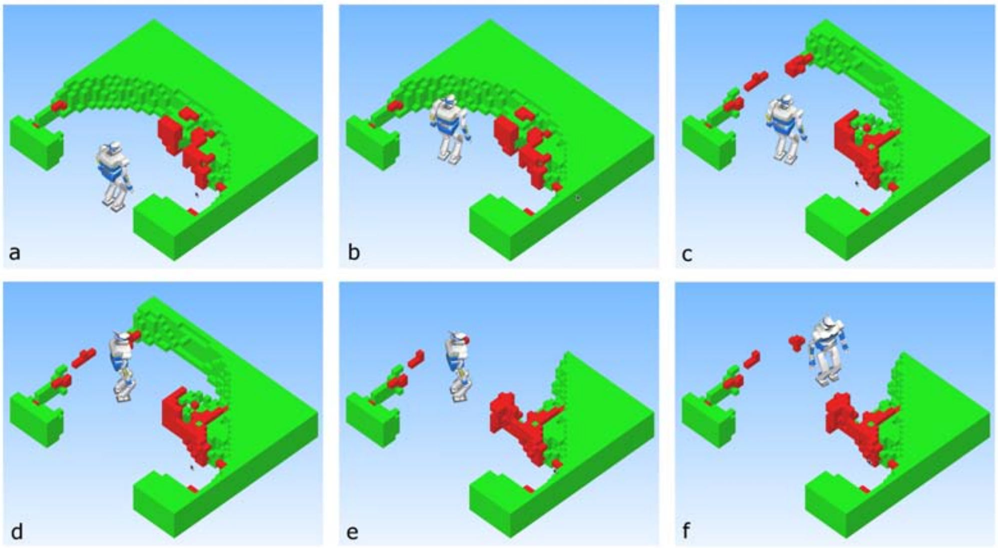


Figure 5.6: The consecutive NBV in exploring the environment.

Chapter 6

Integration

6.1 Introduction

Thanks to the recent developments in mechatronics, humanoid robots are today certainly the most challenging platforms we may expect to support fundamental research on robot autonomy. They are challenging because of their anthropomorphic mechanical structures. Humanoid robots may perform a lot of tasks nearing the complexity and dexterity of human tasks. Such platforms are also challenging due to their higher number of degrees of freedom and their stability issues.

To evaluate our work, we have integrated our approach on the platform of humanoid robot HRP2 N.14 (Figure 6.1). The HRP2 robot family is made of 15 clones. Several of them are at the research facilities at the University of Tokyo [24, 52, 53] and AIST [67, 68].

This chapter presents our experiment on integrating vision, navigation and motion planning on HRP2. This chapter does not focus on a specific technical topic in our contribution. We are going to present the organization of various packages which are used for an autonomous exploration of an environment. This chapter presents an extension to the previous integration on the robot in [70]. The objective is navigating in an unknown environment.

The robot is supposed to navigate in a 3D occupancy grid model of the environment generated by vision. This extension can be added to the "purple ball" demo [70] and enables the robot to find a collision free path in the real environment. The robot is supposed to localize the next best view and explore the unknown environment to fulfill a task such as "Give me the purple ball!".

As mentioned in chapter 5, localization is an important issue for generating a

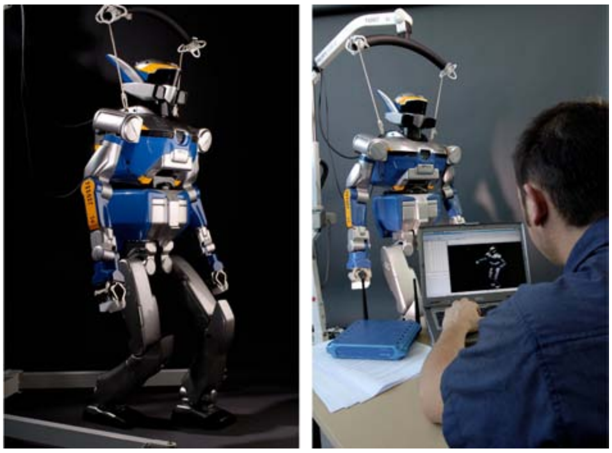


Figure 6.1: The humanoid robot HRP2

precise model of environments. Localization errors become a critical issue when generating models of a large environment or when the robot has a lot of displacement in the environment. So, we allocate a small volume of the environment for exploration by HRP2.

The model of the environment is generated incrementally. At each step, the robot is supposed to find the NBV and plan a collision free motion to the point. At the same time, for updating the model, HRP2 is required to look around and take several photos. Then, the robot uses the photos to update the model of the environment.

Moreover, an algorithm is provided an option to favors the NBV in way to go toward a predetermined location. For example, it can be the position for grasping an objet.

Figure 6.2 illustrates the evolution of a 3D model of an unknown environment along with the planning motion in the model.

6.2 Integration

As it was presented in [70], there are various packages which are integrated on HRP2 to provide the robot with some autonomy in interacting with the workspace. We are going to briefly present some of the existing packages which are presented in detail in [70] and also our recent work to interact with an unknown environment.

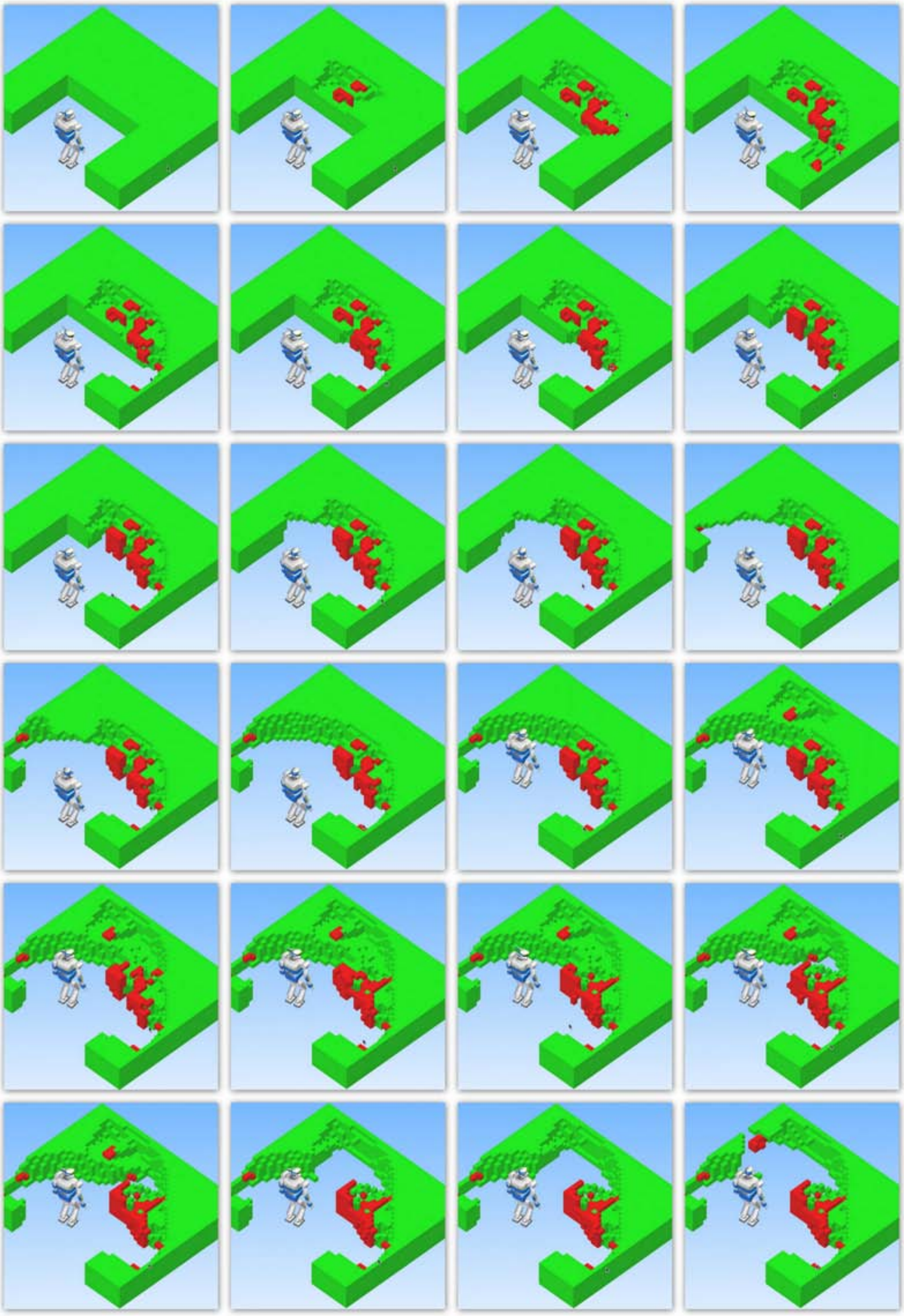


Figure 6.2: Exploration of an unknown environment. Red and green voxels represent *occupied* and *unknown* volume. The robot plans a path to the NBV after updating the model.

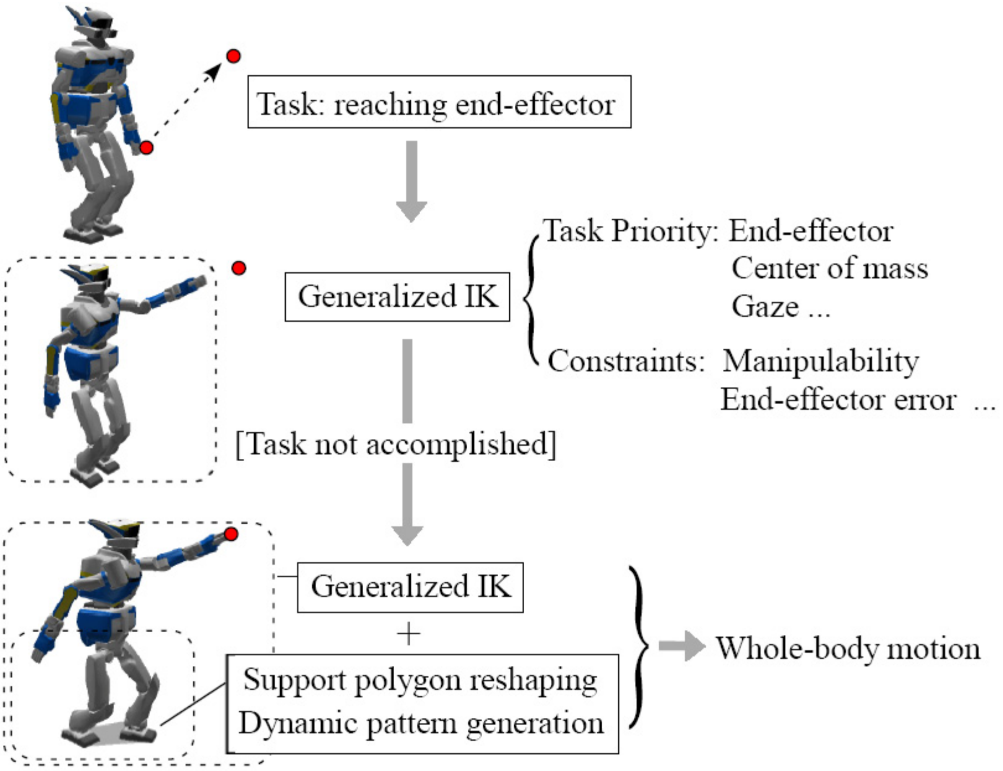

Task: reaching end-effector
Generalized IK
Task Priority: End-effector
Center of mass
Gaze ...
Constraints: Manipulability
End-effector error ...
[Task not accomplished]
Generalized IK
+
Support polygon reshaping
Dynamic pattern generation
Whole-body motion

Figure 6.3: From [70]: GIK: A general framework for task-driven whole-body motion including support polygon reshaping.

6.2.1 Whole body motion

For generating whole body motions such as turning the head around for taking photos, a generalized inverse kinematic (IK) model was used as presented in [69]. Based on this method, tasks such as gaze control and hand motion are considered and whole-body joint angles are computed to execute the motion. Meanwhile, several criteria such as manipulability, stability or joint limits are monitored and finally, the motion is executed. The package GIK is in charge of all the required calculation. A general framework of whole-body motion generation [69] including support polygon reshaping is adapted in GIK.

Figure 6.3 shows an overview of the method which is used in the package GIK.

6.2.2 vision

HRP2 is equipped with two pairs of firewire digital color cameras, configured on two independent stereo-vision benches. Different lenses on the two benches

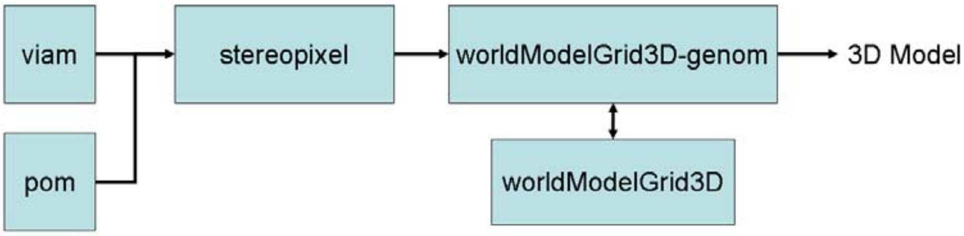


Figure 6.4: Data flow for generating the model of the environment.

allow the selection of the appropriate bench to deal with either narrow images and close objects or global scenes.

For constructing the model of the environment, we implementd the algorithm of occupancy grid in the package WorldModelGrid3D. WorldModelGrid3D receives the required information from the package which is in charge of computing dense 3D images StereoPixel. Also, WorldModelGrid3D requires the position of the cameras for updating the 3D Occupancy map. The package Pom calculates the position of the robot and reports the position of each joint.

Based on the algorithms which are presented in chapter 5, the package WorldModelGrid3D generate a 3D grid map. Our planner uses this model for planning collision free motion in the environment.

Figure 6.4 illustrates the data flow for generating a model of environment.

6.2.3 Navigation

After localizing a next best view, the robot is supposed to plan a free collision path and execute it. A sampling-based method is used for motion planning. The planner is a combination of a linear planner and that used in [70]. It favors the non-holonomic vehicle motion and in case of failure, it uses linear motion to connect the sampled configurations. We apply this planning method to the bounding box of the humanoid robot.

Converting the generated path to a locomotion path is done as presented in [70]. After converting the path into footsteps, a walking pattern generator is applied to generate a dynamically stable walking motion using a method proposed by Kajita *et al.* [28] based on a preview controller for zero moment point (ZMP).

We linked this planner to the package which is in charge of updating the roadmap in a changing environment. The package HppDynamicObstacle deals

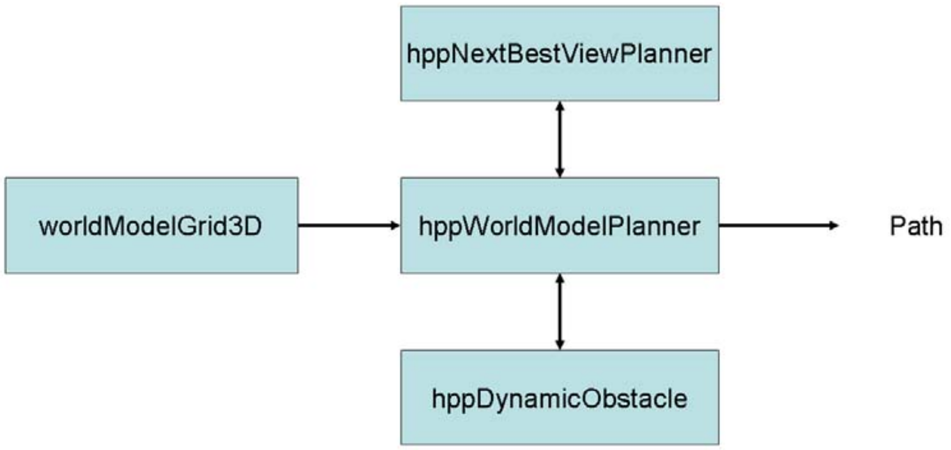


Figure 6.5: Data flow for navigation in environments.

with the changes of the model of the environment and updates the existing roadmap in each instant. The package HppWorldModelPlanner plans the path and also is in charge of finding the NBV after receiving the modification of environment from the package WorldModelGrid3D.

Figure 6.5 illustrates the data flow for navigation in environments.

6.2.4 Data flows

Data like photos, models of environment, NBV and path flow through the components by using the standard mechanism of posters defined by GenoM [20]. Posters are basically shared data structures that can be written only by one producer (the owner) and read asynchronously by several readers without requiring code execution in the owner process context.

6.3 Experimental results

The experiment scenario is exploring an unknown environment towards a predetermined location. HRP2 generates a model of the workspace incrementally by vision. The experiment is conducted in an environment of $3m \times 3m$. The robot generates a 3D occupancy model with voxels of $10cm$. The robot finds NBV's after receiving new information from the environment and updating the model. Then, by updating the existing roadmap, it plans a motion toward the NBV. This loop is repeated until it arrives at the predetermined location.

As it was mentioned in chapter 4, localization plays a critical role in modeling the environment. As we do not have yet a precise localization module on HRP2 robot, after some steps of executing a path and modeling, we loose the precision of robot position.

6.4 Conclusion

We have presented the integration of our work and the existing packages on HRP2. we combining motion planning problems with a standard occupancy grid modeling of the environment. This integration enables HRP2 to conduct the experiment of exploration of an unknown environment in a real situation.

Such a feature supposes an important effort in terms of software development coordination. The targeted robustness has been reached in a small workspace thanks to the use of experienced methods integrated in carefully defined software architecture.

However, based on the weakness of our localization, the integration is not robust in a large environment. We believe that these approaches in terms of integrating the modeling and motion planning would be a step toward allowing autonomous humanoid robots work in real environments.

Chapter 7

Conclusion

In this document we presented a strategy to plan motions for humanoid robots an environment modeled by vision. In particular, we tackled the problem of motion planning in stochastic maps, motion planning in non-static environments, whole-body task planning and generating the model of an environment.

A new framework for motion planning in stochastic maps is introduced in the second chapter. We took into account the uncertainties of the map and linked the resultant path to the landmarks. We redefined the definition of admissible path and reformulate the motion planning problem according to this new definition. Our planner estimates the probability of collision with the environment and uses this probability to accept or reject random configurations. These configurations are then used to construct a roadmap and find a path between the initial and final configurations. We presented a simple example to illustrate the functionality of this approach. We emphasize that our contribution in the second chapter is mainly conceptual and our work is preliminary in nature. However, we believe that the problem raised by us and our proposed solution are worth reporting.

For motion planning in non-static environments, we adapted an existing approach to the problem of motion planning for the bounding box of a humanoid robot. The 3d model of this non-static environment, is composed of thousands of voxels and their status changes based on updated information. In our adaptation, we first decomposed the environment into cells, each of which was composed of a number of voxels. Based on any modification in the status of the voxels, we updated the existing roadmap and we used this modified roadmap in the next motion planning. Although various approaches have been presented to deal with changing environments, we show that our approach is especially suited for navigating in large workspaces.

In chapter 4, we presented a novel way to plan whole-body tasks for humanoid robots. We used a local task solver to generate valid configurations within a random diffusion algorithm framework. Our approach is general and addresses various types of task planning problems for humanoid robots. The presented examples in the chapter illustrate the efficiency of our random diffusion algorithms for highly dimensioned problems. Additionally, regarding the computation time required, the presented example proves that our planner is faster than the existing local method for collision detection.

For generating the model of an environment, we relied on a 3D occupancy grid method. The environment was discretized to thousands of voxels and a probability of occupancy was assigned to each voxel. Stereo vision provided the required information for updating these probabilities. Based on these probabilities, we assigned a status to each voxel. This enables the robot to generate a model of its environment incrementally. Moreover, we presented a simple method that allows the robot to find the next best view for updating the map. The critical issue concerning environment modeling is localization. It plays an important role in generating an efficient model of the environment.

Finally, we integrated our work on the platform of HRP2 to enable the robot to explore unknown environments. The practical implementation of the various algorithms and methods described in the previous chapters, on the robot, required extensive effort in term of software integration.

From our experiments we observe that the target robustness of the algorithms was achieved in small workspaces. However, the desired functionality could not be completely satisfied for large scale environments. This was mainly due to technical problems of localization which were not an issue of this thesis.

The work detailed in this thesis, implements and integrates novel approaches to environment modeling and motion and task planning. We believe our contribution is a small step towards allowing autonomous humanoid robots to work in real environments.

Bibliography

[1] R. Alami, M. Herrb, B. Morisset, R. Chatila, F. Ingrand, P. Moutarlier, S. Fleury, M. Khatib, and T. Simeon. Around the lab in 40 days [indoor robot navigation]. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA'00*, volume 1, 2000. [cited at p. 13]

[2] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998. Proceedings.*, volume 1, 1998. [cited at p. 51]

[3] J.E. Banta, Y. Zhien, X.Z. Wang, G. Zhang, M.T. Smith, and M.A. Abidi. A best-next-view algorithm for three-dimensional scene reconstruction using range images. In *Proc. SPIE*, volume 2588, pages 418–429, 1995. [cited at p. 68]

[4] R. Bohlin and LE Kavraki. Path planning using lazy PRM. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA'00*, volume 1, 2000. [cited at p. 36]

[5] O. Brock and L.E. Kavraki. Decomposition-based motion planning: Towards real-time planning for robots with many degrees of freedom. 2000. [cited at p. 30, 96]

[6] O. Brock and L.E. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA*, volume 2, 2001. [cited at p. 29]

[7] O. Brock and O. Khatib. Elastic strips: A framework for integrated planning and execution. *Lecture notes in control and information sciences*, pages 329–338, 1999. [cited at p. 26, 27, 96]

[8] O. Brock and O. Khatib. Real-time re-planning in high-dimensional configuration spacesusing sets of homotopic paths. In *IEEE International Confer-*

ence on Robotics and Automation, 2000. Proceedings. ICRA'00, volume 1, 2000. [cited at p. 26, 28, 96]

[9] J. Canny. *The complexity of robot motion planning*. MIT press, 1988. [cited at p. 7]

[10] M. Cherif and M. Vidal. Planning handling operations in changing industrial plants. In *1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings*, volume 1, 1998. [cited at p. 25, 26, 96]

[11] H.M. Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT Press, 2005. [cited at p. 7, 52]

[12] D. Cole and P. Newman. Using laser range data for 3d slam in outdoor environments. In *IEEE International Conference on Robotics and Automation*, pages 1556–1563, Florida, May 2006. [cited at p. 67]

[13] J. Cortes, T. Simeon, and J.P. Laumond. A random loop generator for planning the motions of closed kinematic chains using PRM methods. *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 2, 2002. [cited at p. 52]

[14] A. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transaction an pattern analysis and machine intelligence*, 29(6), June 2007. [cited at p. 13, 67]

[15] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University, 1989. [cited at p. 70]

[16] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. [cited at p. 67, 69, 70]

[17] C. Esteves Jaramillo. *Motion planning: from digital actors to humanoid robots*. PhD thesis, Institut National Polytechnique, Toulouse, 100p., 2007. Doctorat. [cited at p. 9, 10, 11, 12, 95, 96]

[18] C. Estrada, J. Neira, and J.D. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005. [cited at p. 13]

[19] E. Feron, E. Frazzoli, and M. Dahleh. Real-time motion planning for agile autonomous vehicles. In *AIAA Conference on Guidance, Navigation and Control*, 2000. [cited at p. 25]

[20] S. Fleury, M. Herrb, and R. Chatila. Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 1997. [cited at p. 82]

[21] T. Foissotte, O. Stasse, A. Escande, P.B. Wieber, and A. Kheddar. A two-steps next-best-view algorithm for autonomous 3d object modeling by a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1159–1164, 2009. [cited at p. 68]

[22] D. Hsu, R. Kindel, J.C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233, 2002. [cited at p. 29, 31, 96]

[23] D. Hsu, J.C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Computational Geometry & Applications*, 9(4-5):495–512, 1999. [cited at p. 8]

[24] T. Inamura, K. Okada, S. Tokutsu, N. Hatao, M. Inaba, and H. Inoue. HRP-2W: A humanoid platform for research on support behavior in daily life environments. *Robotics and Autonomous Systems*, 2008. [cited at p. 77]

[25] P. Indyk and J. Matousek. Low-distortion embeddings of finite metric spaces, Handbook of Discrete and Computational Geometry, 2004. [cited at p. 7]

[26] L. Jaillet and T. Simeon. A PRM-based motion planner for dynamically changing environments. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings*, volume 2. [cited at p. 29, 32, 33, 96, 97]

[27] B. Jensen, G. Froidevaux, X. Greppin, A. Lorotte, L. Mayor, M. Meisser, G. Ramel, and R. Siegwart. The interactive autonomous mobile system RoboX. In *IEEE/RSJ International Conference on Intelligent Robots and System, 2002*, volume 2, 2002. [cited at p. 13]

[28] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, 2, 2003. [cited at p. 81]

[29] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, volume 5. [cited at p. 33, 34, 36, 97]

[30] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and JP Laumond. A Local Collision Avoidance Method for Non-strictly Convex Polyhedra. In *2008 Robotics: Science and Systems Conference*, 2008. [cited at p. 51, 52, 61, 64]

[31] F. Kanehiro, W. Suleiman, F. Lamiraux, E. Yoshida, and J.P. Laumond. Integrating Dynamics into Motion Planning for Humanoid Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pages 660–667, 2008. [cited at p. 52, 53, 61, 64]

[32] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996. [cited at p. 8, 13, 52]

[33] O. Khatib, L. Sentis, J. Park, and J. Warren. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43, 2004. [cited at p. 51]

[34] J.J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002. [cited at p. 52, 53]

[35] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA'00*, volume 2, 2000. [cited at p. 8, 13, 52, 53, 57]

[36] J.C. Latombe. *Robot motion planning*. Kluwer academic publishers, 1991. [cited at p. 7, 52]

[37] J.P. Laumond. Robot motion planning and control. *Lecture notes in control and information sciences*. [cited at p. 7]

[38] J.P. Laumond. Kineo CAM: a success story of motion planning algorithms. *Robotics & Automation Magazine, IEEE*, 13(2):90–93, 2006. [cited at p. 64]

[39] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. [cited at p. 7, 52]

[40] S.M. LaValle, J.H. Yakey, and L.E. Kavraki. A probabilistic roadmap approach for systems with closed kinematicchains. In *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*, volume 3, 1999. [cited at p. 52]

[41] T. Lemaire, S. Lacroix, and J. Solá. A practical 3d bearing-only slam algorithm. Edmonton, Canada, August 2005. IEEE/RSJ. [cited at p. 13, 67]

[42] J.L. Leonard, R.N. Carpenter, and H.J.S. Feder. Stochastic mapping using forward look sonar. *Robotica*, 19(05):467–480, 2001. [cited at p. 13]

[43] P. Leven and S. Hutchinson. Toward Real-Time Path Planning in Changing Environments. *Algorithmic and Computational Robotics: New Directions: the Fourth Workshop on the Algorithmic Foundations of Robotics*, 2001. [cited at p. 33, 34, 35, 36, 39, 41, 42, 97]

[44] M. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision detection: Algorithms and applications. In *Algorithms for Robotics Motion and Manipulation (Proc. of 1996 Workshop on the Algorithmic Foundations of Robotics)*, pages 129–142, 1996. [cited at p. 13]

[45] H. Liu, X. Deng, H. Zha, and D. Ding. A Path Planner in Changing Environments by Using WC Nodes Mapping Coupled with Lazy Edges Evaluation. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4078–4083, 2006. [cited at p. 35, 37, 97]

[46] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE transactions on computers*, 1983. [cited at p. 7]

[47] A. McLean and I. Mazon. Incremental roadmaps and global path planning in evolvingindustrial environments. In *1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings.*, volume 1, 1996. [cited at p. 25]

[48] P.E. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1261–1267, 2006. [cited at p. 14]

[49] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, 2, 1985. [cited at p. 67]

[50] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986. [cited at p. 51]

[51] A. Nuchter, H. Surmann, J. Hertzberg, and S. Birlinghoven. Planning robot motion for 3d digitalization of indoor environments. In *Proc. of the 11th International Conference on Advanced Robotics (ICAR)*, 2003. [cited at p. 68]

[52] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba. Humanoid motion generation system on HRP2-JSK for daily life environment. *planning*, 6:7. [cited at p. 77]

[53] K. Okada, T. Ogura, A. Haneda, D. Kousaka, H. Nakai, M. Inaba, and H. Inoue. Integrated system software for HRP2 humanoid. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, volume 4, 2004. [cited at p. 77]

[54] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press New York, 1987. [cited at p. 68]

[55] M.H. Overmars. Algorithms for motion and navigation in virtual environments and games. *Algorithmic Foundations of Robotics V*, page 1, 2003. [cited at p. 25]

[56] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *IEEE International Conference on Robotics and Automation*, pages 802–802. Citeseer, 1993. [cited at p. 26]

[57] J.M. Sanchiz and R.B. Fisher. A next-best-view algorithm for 3D scene recovery with 5 degrees of freedom. In *Proc. British Machine Vision Conference BMVC99, Nottingham*, pages 163–172, 1999. [cited at p. 68]

[58] J.T. Schwartz and M. Sharir. On the piano mover's problem: III. *Planning, Geometry, and Complexity of Robot Motion, edited by JT Schwartz, M. Sharir, and J. Hopcroft, Ablex, New York*, 1987. [cited at p. 7]

[59] B. Siciliano and J.J.E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1211–1216, 1991. [cited at p. 51]

[60] T. Simeon, J.P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–494, 2000. [cited at p. 10]

[61] M. Stilman. Task constrained motion planning in robot joint space. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 3074–3081. Citeseer, 2007. [cited at p. 52]

[62] S. Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, 2003. [cited at p. 67]

[63] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *IJRR*, 19(11):pp 972–999, 2000. [cited at p. 67]

[64] S. Thrun, M. Bennewitz, W. Burgard, AB Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, et al. MINERVA: A second-generation museum tour-guide robot. In *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*, volume 3, 1999. [cited at p. 13]

[65] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005. [cited at p. 67]

[66] J. van den Berg, D. Ferguson, and J.J. Kuffner. Anytime path planning and replanning in dynamic environments. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2366–2371, 2006. [cited at p. 32, 34, 97]

[67] K. Yokoi, E.E.S. Neo, H. Arisumi, E. Yoshida, O. Stasse, Y. Kawai, S. Kajita, F. Kanehiro, et al. Humanoid robot hrp-2 no. 10 with human supervision. *Nippon Kikai Gakkai Robotikusu, Mekatoronikusu Koenkai Koen Ronbunshu*, 2005, 2005. [cited at p. 77]

[68] K. Yokoi, N.E. Sian, T. Sakaguchi, O. Stasse, Y. Kawai, and K.I. Maruyama. Humanoid robot HRP-2 with human supervision. In *Experimental Robotics*, volume 39, pages 513–522. Springer. [cited at p. 77]

[69] E. Yoshida, O. Kanoun, C. Esteves, and J.P. Laumond. Task-driven support polygon reshaping for humanoids. *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 208–213, 2006. [cited at p. 54, 80]

[70] E. Yoshida, A. Mallet, F. Lamiraux, O. Kanoun, O. Stasse, M. Poirier, P.F. Dominey, J.P. Laumond, and K. Yokoi. Give me the purple ball – he said to hrp-2 n.14. In *IEEE RAS/RSJ Conference on Humanoids Robots, Pittsburg, USA, 30 Nov. - 2 Dec.*, page Oral presentation, 2007. [cited at p. 77, 78, 80, 81, 99]

[71] E. Yoshida, A. Mallet, F. Lamiraux, O. Kanoun, O. Stasse, M. Poirier, P.F. Dominey, J.P. Laumond, and K. Yokoi. "give me the purple ball" he said to hrp-2 n. 14. In *Proceedings of the IEEE-RAS Conf. on Humanoids*, 2007. [cited at p. 74]

List of Figures

1.1 HRP platforms. Right to left: HRP2, HRP3 and the last humanoid HRP4. 3

1.2 Autonomous robot in an environment: a- Perception, b- Modeling, c- Motion Planning, d- Execution. 4

1.3 The humanoid robot HRP2 5

2.1 Motion planning concept: Finding a collision free path between an initial and final configurations. 8

2.2 From [17]: a) A Piano Mover's Problem. Two robots and a virtual character cooperate to transport a piano in a workspace $\mathcal{W} \subset \mathbb{R}^3$. b) The configuration space is divided in $\mathcal{C}_{free}$ (light areas) and $\mathcal{C}_{obstacle}$ (dark areas). A path can be found between $\mathbf{q}_1$ and $\mathbf{q}_2$ because they lie in the same connected component of $\mathcal{C}_{free}$, which is not the case for $\mathbf{q}_3$. 9

2.3 From [17]: In the PRM algorithm a roadmap $\mathcal{R}$ is build around the obstacles by drawing random collision-free configurations (nodes) and connecting them to their nearest neighbors using feasible robot motions (edges). The query is solved by connecting $\mathbf{q}_{init}$ and $\mathbf{q}_{end}$ to the graph and then finding the shortest path on it (thick lines) 10

2.4 From [17]: With the RRT algorithm, two trees rooted on $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ can be grown simultaneously. Each tree is extended until leaf configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ from each tree can be connected by a randomly drawn configuration $\mathbf{q}_{rand}$. 11

2.5 From [17]: The Visibility-PRM algorithm produces a compact roadmap around obstacles with only a few nodes: guards $\mathbf{q}_g$ (dark circles) and connectors $\mathbf{q}_c$ (in white). A new randomly drawn configuration $\mathbf{q}_{rand}$ is added as a guard if it covers a previously unseen visibility domain with a given L or as a connector if it serves as liaison between at least two connected components. 12

2.6 A brief diagram illustrates the planning date flow. 19

2.7 The integral is computed over a ball included in the cone defining non-collision. 21

2.8 The mean value of the obstacle configurations and the robot initial (I.P) and final (F.P) configuration. 23

2.9 Red dots represent the mean values in the stochastic map. The red line represents the planned path. Blue dots represent the actual positions of the landmarks and the blue line represents the path which is followed by the robot by performing localization on the landmarks. 24

3.1 From [10]: Left: the planned path. Right: the resulting updated roadmap after changing the workspace 26

3.2 From [7]: a) A protective hull around the Stanford Mobile Manipulator. b)An elastic tunnel formed by several overlapping protective hulls. 27

3.3 From [8]: On the left: Four manipulator arms move into the path of the Stanford Mobile Manipulator; replanning using elastic strips is performed in real-time to maintain a valid path. The path is indicated by lines connecting points on conscutive configurations along the elastic strip. It is converted into a smooth trajectory during execution. On the right: A Stanford Mobile Manipulator moves into the path of another one. The path is updated in real-time to avoid a collision. 28

3.4 From [5]: Real-time planning in a dynamic environment. 30

3.5 From [22]: Top) Computed example for the air-cushined robot. bottom) *configuration* × *time* representation. 31

3.6 From [26]: Example of a 3D scene with moving obstacles: start and goal configurations of the 9dof mobile manipulator carrying a board and three path solutions obtained for several settings of the environment (doors closed/open and three other moving obstacles placed around the table) 32

3.7 From [26]: A static roadmap is first computed in the configuration space of the robot (1). During queries, a solution path can be found directly inside this roadmap (2) or via a RRT like technique to reconnect edges broken by dynamic obstacles (3). If the existing roadmap does not permit to find a solution, new nodes are inserted and the roadmap is reinforced with cycle creation (4). 33

3.8 From [66]: An example path (in black) planned through state-time space from the initial robot position (in green) to the goal (shown as a vertical blue line extending from the goal location upwards through time). Also shown are the extrapolated trajectories of the dynamic obstacles (in yellow). The underlying PRM has been left out for clarity but can be seen in Right image. 34

3.9 From [43]: A plan for a 19-joint robot passing through a relatively narrow corridor. The dark blocks are the obstacles. 35

3.10 From [29]: Portions of the roadmap are dynamically invalidated according to the obstacles inserted in Robonaut's workspace. 36

3.11 From [45]: Path planning for a dual-manipulators system with 12 DOFs in changing environments. 37

3.12 The planner updates a roadmap based on the modification in the environment. Figure a illustrates simple 2D model of the environment with a roadmap that connect initial and final configuration. Figure b shows the recent environment which is modified partially and in result, some nodes and edges of the roadmap are in collision based on the new model. In c, the invalid nodes and edges are erased and finally, in d, the updated roadmap is used to find the path in new environment. 38

3.13 2D cell decomposition for a robotic arm. 39

3.14 Motion planning in 3D model of robotics lab in LAAS-CNRS. The size of workspace is 17m ×12.5m×1.7m. The workspace is modeled with 10cm×10cm×10cm 3D *Voxels*. Also, workspace is decomposed to the 1.5m×1.5×1.5m *Cells* to implement the approach. 40

3.15 The sequence of process in encountering a change in an environment. 41

3.16 Illustration of roadmap management during updates of workspace. Red and green voxels represents *occupied* and *unknown* zone respectively. The circle shows a *free* part of environment which becomes *occupied* later: a) A roadmap is illustrated in the top view of a 3D model of an environment which is generated by vision. b) The model is updated based on new information from camera. c) The roadmap is modified based on the updates in the 3D model by using our cell decomposition algorithm. Some nodes and edges are erased from the roadmap during the modifications. d) The modified roadmap is used for a motion planning process. Some nodes and edges are added to the roadmap. 43
3.17 The mobile robot Jido which is used in the experiment for taking the required photos. 44
3.18 Changes in the environment. HRP2 uses the cell decomposition method to validate some of the nodes and edges in the existing roadmap for planning next motions 45
3.19 S and G represent the initial and final configuration of HRP2 in the model. Red and Green voxels represent the *occupied* and *unknown* region in the model. 47
3.20 Modification in the status of 124 voxels represent displacement of a table in the model. Red and Green voxels represent the *occupied* and *unknown* region in the model. 48

4.1 Reaching for an object in a constrained area 53
4.2 One step of extension of the RRT-Connect algorithm 54
4.3 Root finding using Newton-Raphson method 55
4.4 One step of constrained extension 56
4.5 Random goal configurations solving the reaching task. All the configurations are stable and collision-free. 58
4.6 Different steps of a posture optimization. All the configurations respect the stability and goal constraints, and the configurations are more and more natural. 58
4.7 HRP2 in two configurations: The both configurations satisfy the task of grasping an object with the right hand. Configuration (a) looks more “natural”. 60
4.8 By considering the constraint of fixed foot position, the target is not accessible by the left hand. 60
4.9 The density of hand positions for a number of random configurations. 61

4.10 The density of hand positions for a number of random configurations. The density is illustrated for each layer along the Z axis. . . . 62
4.11 Solution path for “Table” scenario. 63
4.12 Solution path for “Torus” scenario. 63
4.13 Solution path for a complete resolution of a door opening problem. The walk motion at start was generated separately with a walk planner. We then successively solved a hand reaching problem to grab the handle and a specific door constraint problem to keep the hand on a given arc of a circle. 65

5.1 Estimating the occupancy probability profile. 70
5.2 The robotic lab in LAAS-CNRS which is modeled based on this approach. Red and green voxels (cube) represent *occupied* and *unknown* volumes respectively in the model. The work space is modeled with 3D voxels of 5cm×5cm×5cm. Notice that this model is the result of several stereo-images taken from different points. . . . 71
5.3 Exploring an unknown environment. Red and green voxels represent *occupied* and *unknown* zones in the model. The robot plans a path in *free* zone. 72
5.4 The safe zone in which we look for the NBV (Schematic). 74
5.5 The consecutive NBV in exploring the environment. 75
5.6 The consecutive NBV in exploring the environment. 76

6.1 The humanoid robot HRP2 78
6.2 Exploration of an unknown environment. Red and green voxels represent *occupied* and *unknown* volume. The robot plans a path to the NBV after updating the model. 79
6.3 From [70]: GIK: A general framework for task-driven whole-body motion including support polygon reshaping. 80
6.4 Data flow for generating the model of the environment. 81
6.5 Data flow for navigation in environments. 82

List of Tables

3.1 Time performance. 45
3.2 Memory performance. 46
3.3 Path planning 48
3.4 Path planning by applying the approach 48

4.1 Computational time [ms] for the presented scenarios. 63