GENERATING ROBOT MOTION: THE INTEGRATION OF PLANNING AND EXECUTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Oliver Brock
November 14, 1999

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Oussama Khatib
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jean-Claude Latombe

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John F. Canny

Approved for the University Committee on Graduate Studies:

Abstract

As the application of robotic technology reaches beyond the factory floor into unstructured, dynamic, and populated environments, powerful motion generation algorithms become essential for safety, versatility, dexterity, and speed of task execution. This thesis proposes the integration of motion planning and motion execution algorithms as a framework to address this challenge. Two motion generation methodologies are presented. The *global dynamic window approach* applies to mobile bases with two or three degrees of freedom; the *elastic strip approach* is a more general framework applicable to redundant manipulators with many degrees of freedom. Both approaches integrate motion planning and motion execution algorithms. This integration results in motion generation approaches that retain desirable global properties of planning methods, while maintaining the advantages associated with reactive execution approaches. Experimental results are presented that show both approaches to be highly effective in practice. The application of these approaches results in a significant increase of the robot's capability to move safely and robustly in dynamic environments. The computational complexity of the elastic strip framework is analyzed and compared with motion planning algorithms.

Acknowledgments

Looking back upon the years that ultimately lead to what you are holding in your hands right now, it becomes apparent that I owe gratitude to many people for helping me to turn a dream into reality. The first person that comes to mind, of course, is my advisor Professor Oussama Khatib. His knowledge, vision, and willingness to share them have provided guidance for me during this laborious, yet inspiring journey.

To the members of my reading committee, Professor Jean-Claude Latombe and Professor John Canny, I would like to express my gratitude for the stimulating discussions and the constructive criticism during the final phase of my time at Stanford.

I am very fortunate to have had the opportunity to interact and work with smart, talented, motivated, and – most importantly of all – genuinely nice people: Alan Bowling, Kyong-Sok "K.C." Chang, Bob Holmberg, Oscar Madrigal, and Diego Ruspini. I have learned a lot from them, about robotics and about many other things in life.

Towards my parents, Waltraud and Mario Brock, I feel the most wholehearted gratitude. I am profoundly moved by the realization that all I have accomplished I owe to them. The unwavering trust they have put in me has given me the confidence to accomplish my goals without fear of failure. Their unconditional support has given me the strength and knowledge required to succeed. I sincerely hope they will accept this accomplishment as a token of my appreciation and gratitude. I also thank my brother, Patrick Brock, for showing me the way!

There were many people that influenced my life beyond academia. I would like to thank Sven Bernecker, Tony Flannery, Eric Halpern, Johannes Pohle, and Tim Ryan for their friendship and advice.

Finally, I would like to thank Marein Mathieu, who has always been by my side. So close and yet so far away, she was source of inspiration, strength, courage, and happiness.

Oliver Brock
Stanford, September 1999

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robotic technology has found widespread application in many industrial tasks: the manufacturing of cars, integrated circuits, and numerous other high-volume products is performed by mechanical devices that cut, move, handle, and assemble parts in an automated fashion. Most applications, however, share the characteristic of being performed in structured, highly engineered and constrained environments. These restrictions reduce the complexity of the application to allow automation using existing technology.

An example of an application that is engineered to be suitable for the current state of the art in industrial robotics is the warehouse shown in Figure 1.1. The area in which goods can be stored is divided from the area in which robots can move. Electric wires in the ground guide the robot along the corridors. Any unpredictable change in the environment is avoided. Humans can enter this area only for maintenance and only if robots are not operational. Similar restrictions apply for car manufacturing plants and assembly lines.

These examples indicate that technological capabilities have a large impact and impose constraints on the design of the manufacturing process and environment. Less sophisticated and more reliable technology can be applied where operations have to be executed many hundreds of thousands of times, because the additional effort of carefully designing the manufacturing process is offset by economies of scale.

Restrictive but simplifying requirements of structured and controlled surroundings

Figure 1.1: This warehouse serves as an example for a highly engineered environment to allow the application of robotic technology: obstacles and free space are clearly divided, cables in the ground guide the robots along predetermined trajectories, and human intervention is minimized.

allow the reliable, repeated execution of a particular task according to a predetermined motion sequence. If it can be guaranteed that the conditions are identical each time, the same sequence of motions is assumed to yield the same outcome. More advanced robotic systems rely on event-driven controls to determine the correct timing for the execution of those motions. In a welding station of a car assembly line, for example, the welding operation is started only when optical sensors detect that the conveyor belt has moved the chassis into the correct position. If the chassis of a different car model is delivered by the conveyor belt, however, the welding operation would be executed without achieving the desired result. The robot has no way of detecting the model of car or the fact that a human just moved into its path. Its capabilities to react to the environment – and in particular to unforeseen events in the environment – are very limited.

There obviously are many tasks for which these capabilities are sufficient, mainly because it is feasible and economically sound to engineer the environment in order to guarantee consistent execution conditions for a sequence of motions each time. The warehouse from Figure 1.1 and the car assembly line are just two examples. Nevertheless, there is a significant interest in making those applications safer, more robust, and more versatile. Furthermore, there are many industrial or domestic applications for which it is simply unreasonable to make such strict assumptions about the environment. These applications exhibit at least one of the following properties:

1. For some tasks the environment cannot be controlled sufficiently well for this approach to be applicable. One example of such a task is depicted in Figure 1.2. On a construction site the environment changes in an unpredictable manner every minute: walls are build, materials are delivered, workers and machinery move around, etc. Similar problems arise for tasks in which robots and humans share the workspace, like office environments or hospitals, for example.

2. For many industrial tasks the effort of carefully designing the manufacturing process and environment might not be feasible due to economic considerations. For products with small batch sizes the cost of designing and building a tailored assembly line cannot be justified.

In particular, there are many applications emerging in the area of service and field robotics that fall into the first of the aforementioned categories. Typical tasks in these areas require motion in populated and dynamic environments, manipulation of a wide range of objects with varying physical properties, and interaction with humans. Current robotic technology has proven to have limited applicability for those applications and therefore new methodologies have to be developed to successfully address those challenges.

The work presented in this thesis aims at extending the motion capabilities of robotic systems in order to enable the execution of sophisticated tasks in dynamic and populated environments. To achieve this goal, a robot system is required to perceive changes the environment and to modify its motion behavior accordingly. In generating robot motion, the constraints given by the task and by the continuously changing environment need to be considered. This thesis presents algorithms incorporating

Figure 1.2: A construction site is a highly unstructured and dynamic environment, causing problems for the application of existing robotic automation technology. Obstacles are created and moved constantly, the workspace is shared with human and machinery.

these aspects to allow motion generation and execution in unstructured and dynamic environments in a task-driven and robust manner.

1.1 Motivation

An abundance of algorithms to generate motion commands for robots can be found in the robotics literature. Generally, these algorithms are divided into two main categories: motion planning (Latombe 1991) and motion execution algorithms. In this thesis this division is eliminated by integrating approaches form both categories into a single framework. During the remainder of this thesis it will become apparent that these categories need to be viewed as solutions to two different aspects of a single problem. Since the distinction is commonly made, however, it will be adopted here for the presentation of previous work.

Algorithms that use the state of the world at one given moment in time to determine a sequence of motions to achieve a task will be considered motion planning algorithms. Note that the state of the world can also contain a prediction of how the world is going to evolve over time. The main characteristic is, however, that the entire motion is determined prior to execution. Another way of looking at motion planning algorithms is to say that they correspond to memory-based motion execution: a sequence of motions is determined, committed to memory and then executed from memory.

The class of motion generation algorithms considered as motion execution algorithms, on the other hand, can be viewed as memoryless motion execution: every motion command is entirely dependent on the current state of the robot. This state information can include the current kinematic and dynamic properties of the robot and the current sensory information, as well as the specification of the goal location.

When one compares the advantages and drawbacks of motion planning and motion execution algorithms, fundamental differences become apparent. Motion planning algorithms consider a world model in its entirety. This allows to determine a motion that is guaranteed to accomplish a given task under the assumption that the world model remains accurate during the entire execution of the motion. This is a very restrictive assumption, as in many practical applications the environment changes unpredictably. These changes would oftentimes invalidate a given sequence of motions. Accomplishing the task can no longer guaranteed after such a change has occurred and a new sequence of motion has to be determined after updating the world model. This approach is not practical for most applications, as it might take several seconds or even minutes to complete a motion planning operation.

Motion execution algorithms, on the other hand, exhibit complementary properties. They generally are computationally efficient so that a motion command can be computed several hundred times per second. This low computational complexity results from the fact that only the current state of the robot and local information about the environment are considered to determine a motion command. Therefore, motion execution algorithms are well suited for dynamic and changing environments, as moving or unforeseen obstacles immediately affect the selection of the motion command. This advantage with respect to motion planning algorithms comes at the cost of not being able to guarantee that a given task will actually be accomplished. It is possible, and often happens in practice, that either global knowledge of the world or memory about previously executed commands is required to successfully attain a certain goal.

The realization of the complementary nature of the fundamental properties of motion planning and motion execution algorithms provides the motivation for the work presented here. By integrating motion planning and motion execution algorithms into one coherent framework it will be possible to maintain the desirable properties of both motion generation approaches. The motion generation algorithms resulting from this integration allow to react to changes in the environment without suspending the task execution, effectively uniting the advantages of motion planning and motion execution algorithms.

1.2 Contributions

This thesis introduces two novel frameworks for robot motion generation. Both significantly extend the motion capabilities of robots by integrating motion planning and execution methods into a single, unified approach to motion generation. The main contributions of each of those approaches are detailed below.

1.2.1 Global Dynamic Window Approach

The *global dynamic window approach* is a motion generation method for mobile robots with two or three degrees of freedom. It represents an extremely powerful motion primitive that results in highly robust, goal-directed navigation in dynamic, unknown environments. The input to the algorithm consists solely of the desired goal position relative to the robot's current position. Without any prior knowledge of the environment the global dynamic window approach directs the robot towards the goal at very high velocities. The algorithm is not susceptible to local minima. Without human intervention the environment is searched for possible paths to the desired goal configuration. The robot moves autonomously for extended periods of time, exploring the environment to find paths to the goal until the task is successfully accomplished or it has been determined that no path to the goal exists.

Previous approaches used a complicated architecture of sophisticated algorithms to generate robot motion. In addition, these approaches relied on prior knowledge of the environment. The global dynamic window approach addresses the issues of sensing, map-building, motion generation, and motion execution in one single, simple framework. Furthermore, it allows fast and robust motion in completely unknown environments without being susceptible to local minima. The experimental results presented in Chapter 5 show that the application of the global dynamic window approach results in robot behavior of previously unmatched speed, autonomy, and robustness. For a more detailed comparison with previous approaches found in the literature see Section 5.3. The global dynamic window approach is subject of the first part of this thesis.

1.2.2 Elastic Strip Framework

Whereas the global dynamic window approach is suited for mobile robot with few degrees of freedom, the *elastic strip framework* represents a motion generation framework applicable to redundant robotic manipulators with many degrees of freedom and allows the execution of motion plans in dynamic environments. This is achieved by real-time path modification (Steele and Starr 1988; Quinlan 1994b). The modification procedure incrementally modifies the trajectory in reaction to changes in the environment, while respecting constraints imposed by the task. The incremental modification integrates task-behavior with obstacle avoidance behavior during motion execution.

The most computationally expensive operation of global motion generation approaches is the computation of free space. Traditionally, these computations are performed in the configuration space (Latombe 1991). The dimensionality of this space is dependent on the number of degrees of freedom of the robot. Computing the free space becomes very costly in such a high dimensional space (Canny 1988). As a result, prior approaches are not capable of motion execution for robots with many degrees of freedom in dynamic and unstructured environments. To achieve this goal, real-time planning or real-time modification of the previously planned trajectory is required. The elastic strip framework solves the problem of real-time trajectory modification for robots with many degrees of freedom by introducing a novel approach to free space computation and representation.

The goal of free space computation is to determine connected regions of free space. The computational complexity of free space computation in the configuration space results from the fact that a high-dimensional space has to be explored. For a given discretization of the configuration space, the number of such configurations increases exponentially with the number of degrees of freedom of the robot. The main contribution of the elastic strip framework is a free space representation that can be computed without explicitly exploring the high-dimensional configuration space. The free space is described by a workspace volume in Euclidean space. As the workspace is of constant dimensionality, the complexity of free space computation is decoupled from the number of degrees of freedom of the robot.

In the elastic strip framework, the workspace volume representing the free space in the vicinity of a previously planned trajectory can be seen as an approximation to a set of homotopic paths (see Section 7.2 on page 87). Rather than exploring all configurations and possible paths in configuration space, the volume implicitly approximates a set of homotopic paths by the volume swept by the robot along them. The elastic strip framework then selects one particular path from this set of homotopic paths, exploiting information about the proximity to obstacles in the environment. This effectively prunes and postpones the exploration of the configuration space and renders it demand driven: If during execution of a particular plan proximity information indicates that the current trajectory needs to be modified in order to avoid a collision with an unforeseen or moving obstacle, the configuration space is searched locally, using proximity information as a heuristic. This results in a highly effective approach to motion generation. The global information is provided by a motion planner in from of an initial path. This path is used to generate an approximation of the workspace volume swept by the robot along a set of paths homotopic to the planned one. A reactive motion execution method is integrated by allowing local, incremental modifications of the initial plan exploiting proximity information.

The elastic strip framework introduces the concept of *protective hulls* and the notion of *elastic tunnel* as methods of computing and representing the free space around a robot configuration and around a robot trajectory directly in the workspace. This novel representation of free space can be computed very efficiently, independent of the dimensionality of the configuration space (see Section 8 on page 128).

Using the proposed representation of free space, the elastic strip framework implements real-time path modification for robots with many degrees of freedom. The modification behavior integrates task execution and obstacle avoidance, resulting in obstacle avoidance without suspension of task execution. The effectiveness and efficiency of the approach are experimentally and theoretically validated. Previous approaches to motion generation in unpredictably changing environments could address neither robots with many degrees of freedom, nor the integration of task execution and obstacle avoidance.

Another contribution of this thesis is concerned with the execution of trajectories that are constantly changing. When executing a trajectory with a manipulator arm or on a mobile manipulator, the trajectory is assumed to remain constant over time. Here, the execution error serves as an input to a feedback loop. This loop determines new motion commands at a high rate, minimizing execution error. If the trajectory is modified during execution, as it is the case in the elastic strip framework, those methods cannot be applied. This thesis presents a new method of trajectory execution that allows discrete changes of the trajectory, while continuously executing smooth robot motion.

The above contributions render the elastic strip framework a powerful approach to motion generation for robotic systems with many degrees of freedom in dynamic environments. This novel approach opens up many new areas of application, as the resulting motion capabilities allow the robust execution of complex robotic tasks in unstructured and changing environments. The elastic strip framework unifies gross and fine motion execution by integrating force control into a gross motion execution method. The second part of this thesis contains a detailed description of this framework, and experimental and theoretical results related to it.

Chapter 2

Collision-Free Robot Motion

The robotics literature presents many approaches to the generation of collision-free motion for robots. They are generally categorized according to their underlying methodology. These fundamental categories are introduced in this chapter to provide the reader with an overview of prior approaches. The basic concepts common to those approaches are also introduced in this chapter. For a detailed introduction the reader is referred to the standard robot motion planning text by Jean-Claude Latombe (Latombe 1991).

2.1 Basic Concepts

In attempting to reason about, generate, and represent the motion of a robot, the ability to describe a robot's current position, a set of allowable positions in which the robot does not collide with the environment, and a continuous sequence of allowable positions representing a motion from an initial to a final position are required. This section introduces formalizations of those basic concepts common to the robotics literature.

2.1.1 Spatial Description

The position and orientation of a rigid body can be described by six parameters (Goldstein 1980). Attaching a coordinate frame to the body, three parameters are used to specify the position of its origin relative to a global reference frame and three parameters for its orientation. A robot is typically represented as an articulated body, a sequence of rigid bodies, called *links*, connected by *joints*.

Joints impose kinematic constraints on the rigid bodies they connect. For two rigid bodies linked by a revolute or prismatic joint only four parameters are needed to describe the relative position with respect to each other, as the joint axis restricts their relative motion. The *joint variable* describing the current setting of the joint is the only parameter among those four that can change over time. Hence, for two rigid bodies connected by a joint only one variable is needed to describe their relative position.

Any set of parameters uniquely describing the position and orientation of every part of a robot is called *configuration*; joint variables are a very natural choice to describe a configuration. For simplicity the reference frame usually coincides with the frame of the first link of the robot. Given a robot with $n+1$ links and n revolute joints, $6n$ parameters could be used to describe the n rigid body transformation between each body. However, exploiting the kinematic constraints imposed by the joints, n parameters are sufficient to specify its configuration. The minimal set of parameters uniquely describing the position and orientation of every part of a robot are called *generalized coordinates* (Khatib 1998).

The kinematic constraints imposed by revolute and prismatic joints belong to the category of *holonomic constraints*. Intuitively, holonomic constraints reduce the number of generalized coordinates needed to describe the configuration. *Nonholonomic constraints* on the other hand do not have this property. A familiar example of a nonholonomic constraint is the motion of a car. Although a car can assume any position and orientation in the plane and hence three parameters are needed to describe its configuration, its motion is determined by only two parameters, the steering angle and its linear velocity. The imposed nonholonomic constraint does not reduce the number of generalized coordinates. In general, robots that move in the plane are referred to as *mobile robots*.

A robot with n prismatic or revolute joints has n *degrees of freedom*. If the end-effector has m degrees of freedom, $m \leq 6$, and $m < n$, the robot is said to be *redundant* and $n - m$ is the *degree of redundancy*. If a given task executed with the end-effector only requires m' degrees of freedom, $n - m'$ is called *degree of redundancy with respect to the task*. For example, a robot with six degrees of freedom in not redundant for positioning and orienting the end-effector. If the task requires only positioning, however, the degree of redundancy with respect to the task is three.

2.1.2 Configuration Space

Using the notion of generalized coordinates, the configuration of a robot can be seen as a point in n-dimensional Euclidean space, where n is the number of generalized coordinates and the coordinate axes represent the n joint variables. This space is called *configuration space* (Goldstein 1980) and is designated by $\mathcal{C}$; it is commonly used in robotics (Lozano-Pérez 1983), because it allows to represent the motion of a complicated articulated body as a one-dimensional curve in a high-dimensional space.

As in the real world rigid bodies cannot interpenetrate, points in configuration space are divided into those that correspond to physically feasible configurations and those in which the robot would penetrate an obstacle. The union of the former is called *free space* $\mathcal{C}_{free}$, whereas connected regions of the latter are called *configuration space obstacles* $\mathcal{C}_{obst}$, or $\mathcal{C}$-obstacles. Hence, the free space is the complement of the union of all configuration space obstacles.

$$\mathcal{C}_{obst} = \overline{\mathcal{C}_{free}} = \{\mathbf{q} \in \mathcal{C} \ : \ \mathcal{R} \cap \mathcal{O} \neq \emptyset\},$$

where $\mathcal{R}$ represents the space occupied by the robot in configuration $\mathbf{q}$ and $\mathcal{O}$ the space occupied by obstacles. $\mathcal{C}_{obst}$ is called *configuration space obstacle region* ($\mathcal{C}$-obstacle region). Depending on the task, configurations for which the robot is in contact with the environment are regarded as belonging to the free space or to a configuration space obstacle.

The notion of configuration space is opposed with the notion of *workspace*, designated by $\mathcal{W}$. The workspace of a robot is the physical space it operates in. Planar robots have a two-dimensional workspace and spatial robots a three-dimensional one. Note that in the workspace a configuration is represented by the volume of the rigid bodies comprising the robot and not by a point, as it was the case in configuration space. The dimensionality is bounded and independent of the number of degrees of freedom of the robot, but the representation of a configuration much more complex.

2.1.3 Path and Trajectory

The motion of a robot can be described by a smooth and continuous one-dimensional curve of finite length in configuration space. This curve is called *path*. The endpoints of the path are called *initial* and *final configuration*. A path can be represented as a function $\mathbf{c}(s)$ that maps the interval $[0,1]$ to the corresponding curve in configuration space: $\mathbf{c}(s) : [0,1] \rightarrow \mathcal{C}$, $\mathbf{c}(0) = \mathbf{q}_{init}$, $\mathbf{c}(1) = \mathbf{q}_{goal}$. If the parameterization of the function representing the path is interpreted as time, we call it a *trajectory*: $\mathbf{c}(s(t)) : [0,1] \Rightarrow \mathcal{C}$.

2.2 Fundamental Methods

The majority of algorithms that generate motion for robots can be divided into four categories that are introduced in this section. Roadmaps, cell decomposition, and potential fields address the *basic motion planning problem*, in which the robot is considered to be a single rigid object or an articulated body, the environment is known, and a path has to be determined connecting an initial and a final configuration of the robot. Reactive control, on the other hand, addresses the problem of *sensor-based motion execution*. This distinction will be discussed in Section 2.3. Methods to generate motion for mobile robots are treated in more detail in Chapter 3, those for articulated robots in Chapter 6.

2.2.1 Roadmaps

Roadmap algorithms capture the connectivity of the free space $\mathcal{C}_{free}$ in a graph of partially connected one-dimensional curves, called the *roadmap*. A particular path is generated by connecting the initial and final configuration of the robot to the roadmap and subsequently connecting the resulting configurations by a graph search of the roadmap. The computationally most expensive operation is the construction of the roadmap. Once obtained for a static environment it can be used to solve all instances of the motion planning problem.

The *visibility graph method* (Nilsson 1969) is one of the earliest roadmap methods. It generates roadmaps for two-dimensional configuration spaces by building a graph connecting all the vertices of the polygonal obstacles that can be connected by a line without crossing another obstacle.

The concept of a retraction function from topology can also be used to generate a roadmap (Ó'Dúnlaing and Yap 1982). Such a function defines a continuous mapping of $\mathcal{C}_{free}$ onto a one-dimensional subset of $\mathcal{C}_{free}$ that is then used as the roadmap. Intuitively, one can imagine a retraction as shrinking the boundary of the free space until only one-dimensional curves remain. Compared to the visibility graph methods, in which the segments of the roadmap may graze obstacles, retraction approaches have the desirable property of maintaining large clearance from obstacles.

When solving the motion planning problem for a polygonal robot among polygonal obstacles in the plane, a roadmap can be generated using the *freeway method* (Brooks 1983). A freeway results from expanding a line segment defined by two opposing edges of obstacles. This line is extended until a termination criterion is met. This criterion ensures that the freeway remains entirely in free space. Overlapping freeways define the roadmap that can be used for planning.

The *silhouette method* (Canny 1988) is a very general roadmap method. It is applicable to any problem that can be reduced to planning a path in a compact semi-algebraic set. In this method the roadmap of a configuration space of arbitrary dimensionality is constructed using a sweep algorithm recursively in lower-dimensional subspaces of the configuration space. For a detailed description of this involved method the reader is referred to the text by John F. Canny (Canny 1988).

Compared to the approaches presented so far, *probabilistic roadmaps* (Kavraki and Latombe 1994; Kavraki 1994) are constructed in a fundamentally different way. Whereas all other approaches use a description of the configuration space obstacles to determine the roadmap, the randomized approach only computes the configuration space obstacles implicitly. A randomly chosen configuration q of the robot is checked for collision with the environment. If there is no collision, $q \in \mathcal{C}_{free}$, otherwise $q \in \mathcal{C}_{obst}$ and q is discarded. If enough free space configurations have been generated the roadmap can be build by connecting those configurations for which a very simple motion plan exists. This efficient way of approximating $\mathcal{C}_{free}$ makes the approach applicable to robots with a high number of degrees of freedom. Probabilistic motion planning is discussed in more detain in Section 6.1.1 of Chapter 6.

2.2.2 Cell Decomposition

Cell decomposition methods divide the volume (or area in two-dimensional configuration spaces) of the free space into disjunct units of a simple shape, called *cells*. For *exact cell decomposition* (Schwartz and Sharir 1983) the union of all cells corresponds to the entire free space. Hence, whenever a path exist, it can be computed using this method. Cells have to be computed analytically and the required computations quickly become intractable for robots with many degrees of freedom.

Approximate cell decomposition (Brooks and Lozano-Pérez 1985), on the other hand, uses uniform cells, squares or cubes for example. Since the boundaries of configuration space obstacles are not necessarily aligned with the boundaries of cells, some cells might contain both, free space and a configuration space obstacle. Those cells cannot be used to find a path. Therefore, the union of all cells representing free space does not necessarily represent the true free space accurately and the method might fail to find a path even though one exists.

A path can be generated by searching the *connectivity graph* describing the adjacency of the cells in the exact and approximate cell decomposition methods. The shape of the cells is chosen in such a manner that motion generation within a cell is simple. A path then consists of a sequence of cells and points at which the transition from one cell to another occurs.

2.2.3 Potential Field

Using the notion of configuration space, the robot can be viewed as a point. A path between an initial and a final configuration can then be determined by searching the discretized configuration space for a path connecting an initial and a final configuration without crossing obstacles. Heuristics need to be employed to make this exploration of $\mathcal{C}$-space computationally more efficient.

The heuristic used in the *potential field approach* (Khatib 1980; Khatib 1986) is based on the simulation of physical potentials. Obstacles exert a repulsive force on the robot, whereas the goal configuration attracts the robot. Different potentials are merged into a single potential function that is used to determine the motion of the robot. At every point in the configuration space the gradient of this resulting potential function specifies the desired direction of motion. This corresponds to applying a *gradient descent technique* to the potential function. For an example of a potential function see Figure 2.1.

The potential field approach does not compute a path in advance; rather, the path is represented implicitly by the potential function. The advantage of this approach is that the computational complexity is very low and a motion command can be generated very efficiently by evaluating the potential function at the robot's current position. However, the potential field approach is susceptible to local minima. Those are configurations different from the goal, where the individual potential functions have a resultant force of zero and hence the robot will not move from that position. An illustration of local minima is given in Figure 2.2. The attractive force $\mathbf{f}_a$ and the repulsive force $\mathbf{f}_r$ are opposing each other and the robot will come to rest at the point where their absolute values are equal. Using this potential function no motion can be generated that will cause the robot to reach its goal configuration from its current location.

To overcome this problem *navigation functions* (Koditschek 1987; Rimon and Koditschek 1992) were introduced. They are virtually local minima-free potential

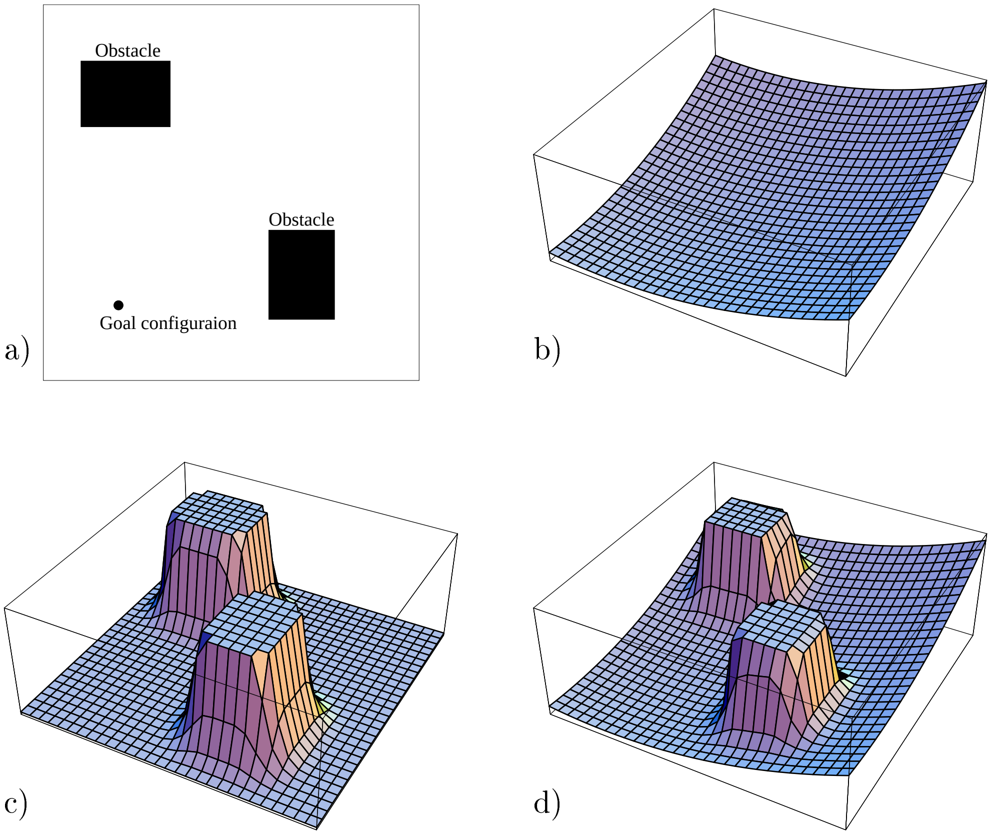


Figure 2.1: Example of a potential function: a) The environment contains to obstacles; the goal configuration is indicated. b) The distance to the goal is used to compute a potential function that attracts the robot to the goal. c) A repulsive potential is associated with the obstacles. d) After merging the attractive and repulsive potentials a gradient descent method can be used to find a path from any configuration to the goal configuration, while avoiding obstacles.

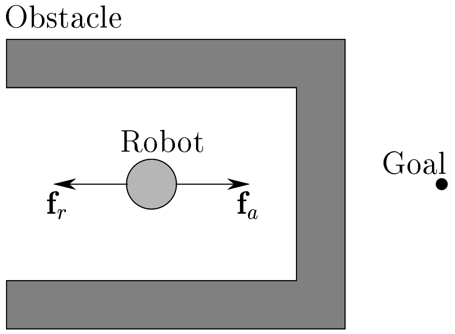


Figure 2.2: Example of a local minimum: The attractive and repulsive force are of equal magnitude but opposite direction and cancel out. As a result the robot does not move, despite the fact that is has not reached the goal yet.

functions that guarantee the robot to reach the goal, if a path exists. The computation of analytical navigation functions is computationally very complex. By discretizing the configuration space the computational requirements can be reduced significantly (Barraquand and Latombe 1991). A discretized navigation function can be computed by a wave-expansion algorithm and captures the connectivity of the entire free space. The details of this algorithm are given in Section 3.1.3.

2.2.4 Reactive Control

The three previous sections describe solutions to the basic motion planning problem. All of these approaches assume the environment to be known and static. This is a necessary requirement, since the path or the potential function are determined in advance and unknown obstacles or changes in the environment could invalidate them. The goal of *reactive control* is to integrate sensing into the motion generation process.

A reactively controlled robot uses sensory information to determine its motion. One such approach is the potential field method already introduced as a motion planning method in the previous section. The repulsive potential caused by obstacles can easily be derived from proximity information acquired through sensing. The attractive potential is still derived by the distance vector to the goal configuration. The problem of local minima mentioned in Section 2.2.3 still arises.

Boundary following is a reactive control scheme for motion in the plane that is immune to local minima (Lumelsky 1987). The robot is commanded to move on a straight line towards the goal. If an obstacles is blocking the path to the goal, the robot traces its contour, remembering the point closest to the goal. From this point the motion is continued on a straight line towards the goal. By repeating this process until the goal is reached convergence can be achieved and local minima are avoided.

2.3 Planning and Execution

In the previous section various methods for generating motion for robots were introduced. Reactive control or *motion execution* methods were distinguished from *motion planning* methods (roadmap, cell decomposition, potential field) by the fact that they can generate motion in dynamic environments, where the location of obstacles changes unpredictably. This section examines other criteria that can be used to distinguish between the presented motion generation algorithms.

A very common distinction labels the roadmap and the cell decomposition approaches as *global* and reactive execution approaches as *local*. These terms refer to the fact that the former compute the connectivity information for the entire configuration space, whereas reactive execution methods as described in Section 2.2.4 only use local sensor information. It is this locality that makes those algorithms susceptible to local minima. However, navigation functions (see Section 2.2.3) are computed considering connectivity information about the entire configuration space; they can eliminate local minima in the potential function. Hence, navigation functions must be regarded as global. Consequently, the potential field approach is considered global or local, depending on the kind of potential function it is applied to.

Despite the fact that the potential field approach cannot be assigned to one of those categories, it is possible to characterize planning methods as global and execution methods as local. Motion generation using potential fields with a purely distance based potential function would be considered execution, while potential fields applied to a globally computed navigation function would be considered planning.

Another criterion that is closely related to the global/local distinction is the characterization of configuration space obstacles. Local methods obviously only have a local representation, such as the sensory input measuring the proximity of obstacles. And almost all global methods represent the configuration space obstacles explicitly, deriving the free space directly from their boundary description. Probabilistic roadmaps represent configuration space obstacles only implicitly, by rejecting those configurations of the robot that are in collision with the environment. Since the configuration space obstacle computation is the most costly operation in planning, it is precisely this difference between explicit and implicit representation that makes probabilistic roadmaps such an efficient approach to motion planning and allows it to be applied to robots with many degrees of freedom.

The most natural distinction between motion generation algorithms, however, is temporal: planning algorithms can be said to be executed completely or partially *before* the execution of the planned motion begins, whereas execution algorithms *interleave* motion execution and the execution of algorithms to determine the next motion command. This temporal characteristic is directly related to the computational effort required for free space computation. Since computing the connectivity information of the configuration space is a computationally expensive operation, it cannot be performed in parallel with motion execution. Hence, the global methods presented in the robotics literature have exclusively been planning methods. By the same reasoning motion execution methods have been local.

The separation of motion planning and motion execution algorithms for the generation and execution of robot motion raises fundamental problems. Global methods are needed to be immune to local minima but only local methods can be interleaved with execution in order to accommodate changes in dynamic environments. Therefore, the applicability of automated motion generation and execution is restricted to either static or predictable environments, environments in which obstacles move on a known trajectory, or those environments that are inherently local minima-free. All of these assumptions cannot be maintained in almost every realistic application in robotics. The work presented in this thesis aims at integrating planning and execution algorithms to overcome this fundamental problem.

2.4 Integrating Planning and Execution

The integration of motion planning and motion execution algorithms has previously been applied to improve the performance or extend the applicability of motion generation algorithms (Faverjon and Tournassoud 1987; Choi and Latombe 1991; Quinlan 1994b). What is to be expected from such an integration? Can we expect to solve the problems that arise from the separation of global plan generation and local motion execution? This section outlines the goals pursued by integrating planning and execution into a single framework for motion generation for robotic applications.

In evaluating the problems and determining the goals of integration between planning and execution we will examine various system architectures for the generation of robot motion. These are graphically illustrated in Figures 2.3, 2.4, 2.5, and 2.6. In those figures, boxes on the left represent information about the environment at various levels of detail. One example is *sensing*, referring to unprocessed sensory information about the position of the robot, or distance information to obstacles detected by sensors. In Figure 2.4 a *world model* is used to represent global connectivity information about the free space. Boxes on the right symbolize the process or method that is used in order to generate robot motion. Boxes at the same height indicated information and processes at the same level of abstraction.

A purely reactive motion execution scheme determines motion commands by evaluating sensory information. Figure 2.3 illustrates this system architecture schematically. The control uses information obtained through sensing to influence the environment. This is usually done by moving the robot according to a motion or control command. As mentioned above, no global connectivity information about the free space is used in this process, causing this approach to be susceptible to local minima. As indicated by the arrows in the figure, the processes of sensing and motion execution form a loop, called *control loop* (Franklin, Powell, and Workman 1998).

Planning, on the other hand, avoids the problem of local minima by using a model of the environment to determine a plan. The assumption is made that the environment

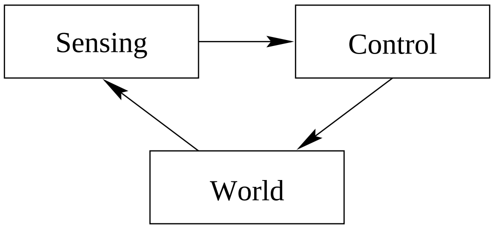


Figure 2.3: Reactive motion execution architecture: The perception of the world is directly converted into a motion control command. This can be done very efficiently, but only local information about the environment is used to determine the motion command and the approach is susceptible to local minima.

remains unchanged or that future changes are known at planning time. The resulting plan or trajectory is then guaranteed to move the robot from its initial configuration to the desired goal configuration. The assumption of predictable changes to the environment, however, is unrealistic in most robotic applications. If there should be an unforeseen change in the environment, the generated plan may be invalidated. The model of the world then has to be updated using sensing or human intervention and a new plan needs to be generated. Figure 2.4 shows the corresponding diagram. There is no loop in this diagram because planning is not executed iteratively. Instead, a plan is generated once and then executed blindly.

Both previously presented schemes can be combined. The result is shown in Figure 2.5. Here, a plan is generated once, using a model of the world. But instead of executing it without sensory feedback, a reactive scheme is used to avoid obstacles detected by sensors. An attractive potential drags the robot along the path while repulsive potentials cause it to evade unforeseen obstacles. While local minima that were present in the world model can be avoided this way, those that arise from changes in the environment may still prevent the robot from moving to the goal position. Examining Figure 2.5 it can be seen that the latter kind of local minima cannot be incorporated during the planning process, as the sensory information is not used to

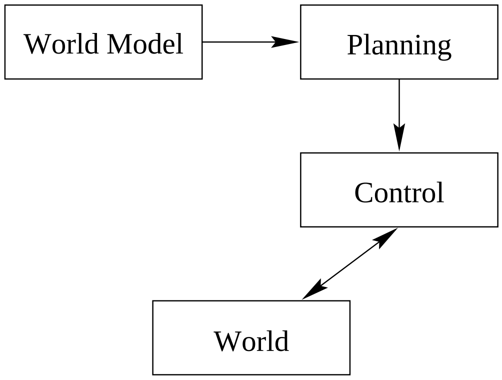


Figure 2.4: Motion planning architecture: A world model is used to determine a sequence of motion commands to achieve the desired goal configuration before the motion is executed. Once a plan has been determined it is executed by the control.

update the world model in order to generate a new plan.

It becomes obvious that the ideal system architecture closes a loop around planning, motion execution, sensing, and the world model. This ideal motion generation and execution architecture is shown in Figure 2.6. Each time the world model is updated with sensory data, a new plan is generated. This plan avoids all obstacles in the environment and consists of a sequence of motion commands that are guaranteed to navigate the robot to the goal configuration, avoiding all local minima.

Although in theory this idealized architecture solves the problems arising from the use of planning or execution alone or in sequence, it is not applicable to most practical problems. As will be discussed in more detail in Chapter 10, planning is a computationally demanding operation. For moderately complex planning problems even the fastest planners can take minutes to generate a plan. A control loop that issues a motion command once a minute would be as effective in moving a robot as the driver of a car would be, taking a glance at the street only once every half hour.

The dilemma is clear: Planning can guarantee that the robot will reach the goal configuration if it is at all possible, but the generation of a plan might take a long

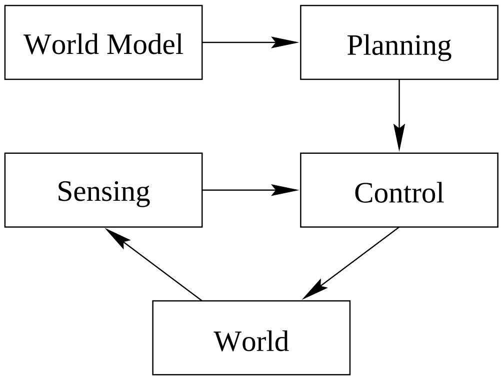


Figure 2.5: Motion planning architecture with reactive execution: This architecture differs from the pure motion planning architecture in that reactive methods are employed to execute the plan. It is intended to accommodate changes in the environment that occur after planning and during the execution of the plan.

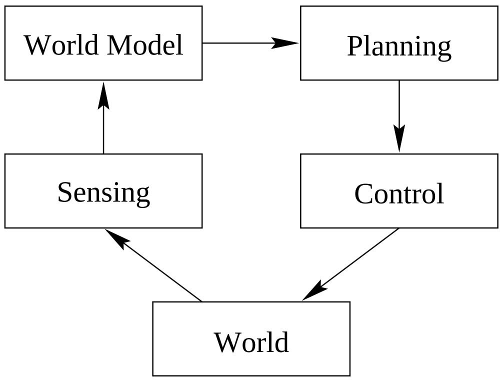


Figure 2.6: Ideal motion generation and execution architecture: During execution of the current plan the world model is updated using sensor information. The updated world model can now be used to determine the appropriate plan. This plan will always reflect the current state of the world. If the planning operation can be performed for each motion command, a globally optimal command can be chosen each time.

time. Too long for planning to be integrated into a control loop. The advantage of the control loop, on the other hand, is that it enables the robot motion to be changed in reaction to changes in the environment. Reactive motion execution, however, is susceptible to local minima and it can no longer be guaranteed that the robot will reach its goal. Both methods – planning and execution – exhibit desirable properties that complement each other. The goal of this thesis to combine the advantages of these approaches by integrating motion planning and motion execution. The effectiveness of this integration will show that the separation of motion generation into motion planning and motion execution algorithms might in fact be an artificial one and that these algorithms address two aspects of the same problem.

Part I

Navigation of a Mobile Base

Chapter 3

Motion Planning and Execution

This chapter introduces motion planning and motion execution algorithms for mobile robots. In this thesis, a robot is considered a mobile robot if it can be described by a single convex polygon moving in the plane. The dimensionality of the configuration space of such a robot cannot exceed three: two translational degrees of freedom and one rotational degree of freedom. Due to the low dimensionality, many approaches that would be computationally intractable for robots with many degrees of freedom are practical for mobile robots. This chapter introduces these planning and execution methods and discusses previous attempts of integrating them into one motion generation framework.

3.1 Planning Paradigms

The planning approaches described in Section 2.2 are applicable to mobile robots, once the configuration space representation of the obstacles has been computed. One method of configurations space obstacle computation for mobile robots is described in this section. The presented procedure assumes that the environment consists of convex, polygonal obstacles. Non-convex obstacles can be decomposed into convex regions. The planning approaches that are relevant to the framework introduced later in this part of the thesis will also be presented in this section.

3.1.1 Configuration Space Obstacles

The configuration space obstacle $\mathcal{CP}_\theta$ that corresponds to a convex polygon $\mathcal{P}$, given a robot $\mathcal{R}$ in fixed orientation θ, is defined as:

$$\mathcal{CP}_\theta = conv(vert(\mathcal{P}) \ominus vert(\mathcal{R}_{(0,0,\theta)})),$$

where $\mathcal{R}_{(0,0,\theta)}$ describes the robot $\mathcal{R}$ translated to position $(0,0)$ and rotated to orientation θ, $vert(\mathcal{P})$ designates the set of vertices of polygon $\mathcal{P}$, $conv(\cdot)$ denotes the convex hull of a set of points, and the $\ominus$ operator stands for the Minkowski difference of affine spaces A and B, given by $A \ominus B = \{x \mid x = a - b, a \in A, b \in B\}$. In the computational geometry literature various algorithms to compute the convex hull of a set of points have been presented (Preparata and Shamos 1985; Guibas 1994; Mulmuley 1994).

Intuitively, this definition describes a method of computing configurations space obstacles, proceeding by moving the robot along the boundary of the workspace obstacle in a fixed orientation and tracing the motion of its reference point. Figure 3.1 illustrates this for a triangular robot and a rectangular obstacle. The robot and its reference frame are shown on the left. The trace of the reference point of the robot during its motion around the workspace obstacle, shown in gray, is illustrated in the middle. On the right side of the figure the resulting configuration space obstacle is shown in black.

A slightly more efficient algorithm to compute the configuration space obstacle for a convex, polygonal robot with a fixed rotation and a convex, polygonal obstacle has been proposed in Lozano-Pérez 1983. This method can be extended to convex and locally non-convex (Latombe 1991) generalized polygons, i.e. regions bounded by circular arcs and line segments (Laumond 1987). A more general method also computes the configuration space obstacles for non-convex polygons at a higher computational expense (Avnaim and Boissonnat 1988).

Above procedure computes the configuration space obstacle for the robot $\mathcal{R}$ at a fixed orientation θ. By discretizing the range $(-\pi, \pi]$ of all possible orientations and by computing the configuration space obstacle region for each of the discrete

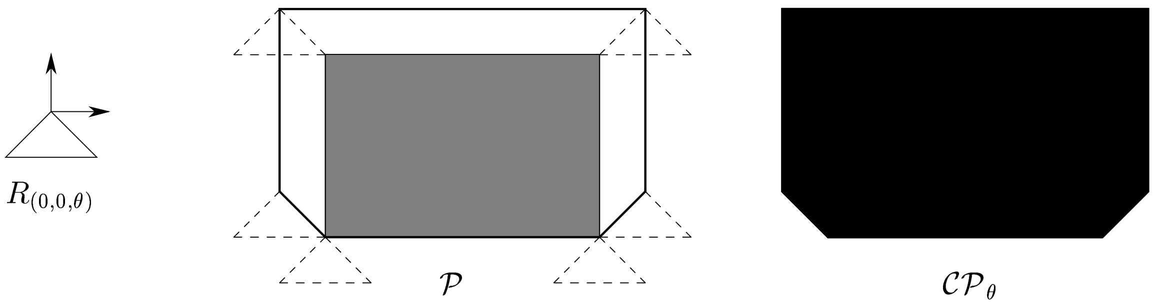


Figure 3.1: Generation of configuration space obstacles in two dimensions: On the left the robot and its attached reference frame are shown. By sliding the robot along the obstacle and tracing the origin of the reference frame, as shown in the middle, the configuration space obstacle can be computed. For the given orientation of the robot the configuration space obstacles is shown to the right.

orientations, the entire configuration space obstacle region can be computed in slices. Whether the robot at configuration (x, y, θ) is in collision with the environment or not can now be determined by performing a simple lookup operation in the resulting data structure.

With the configuration space obstacle region computed in this manner, the robot can now be regarded as a point moving in configuration space. A mobile robot that can be represented by a disc or has fixed orientation constitutes a special case. Here, the configuration space consists of only one slice and the transition from workspace to configuration space is referred to as *growing the obstacles*: the robot is shrunk to a point (the origin of its reference frame) and the obstacles are increased in size by the same amount. In this case, there exists a simple mapping between workspace and configuration space.

3.1.2 Configuration-Time Space

If the motion of all obstacles in the environment is known a priori, motion planning methods can be performed using the notion of *configuration-time space* (Reif and Sharir 1985). Configuration space is extended by another dimension representing time. For each value on the time axis, the other dimensions represent the configuration space obstacle regions for the obstacle configuration at that particular instant in time. For the generation of a valid motion plan it has to be ensured that the robot cannot move back in time along the negative time-axis. Configuration-time space obstacle regions can only be computed, if the trajectories for all moving obstacles are known.

3.1.3 Navigation Functions

In Section 2.2.3 it was shown that due to local minima the potential field approach may fail to move the robot to the desired goal configuration. In the presence of configuration space obstacles, the existence of a potential function that from every point in the configuration space will cause a motion of the robot towards the goal (Koditschek 1987) cannot be guaranteed. As a matter of fact, for every disjoint $\mathcal{C}$-obstacle this function must contain at least one saddle point. However, those saddle points have measure zero and any slight deviation will result in the desired behavior. A potential function containing saddle points but no local minima is called *almost global navigation function*, for simplicity *global navigation function*. It has been shown that such functions can be constructed for certain shapes of obstacles (Rimon and Koditschek 1989; Rimon and Koditschek 1990). More general approaches handle arbitrarily shaped obstacles in low-dimensional configuration spaces (Rimon and Koditschek 1992; LaValle 1998).

Despite the fact that the analytical construction of *analytical* navigation functions is rather involved, algorithms to compute *numerical* navigation functions are quite simple (Dorst and Trovato 1989; Barraquand and Latombe 1991). These functions represent local minima free potential functions and therefore eliminate the main drawback of the potential field approach. The complexity of their computation only allows an application to low-dimensional configuration spaces. Hence, they are well suited for mobile robots.

The numerical navigation functions presented here are grid-based. They can be constructed in configuration spaces of any dimension, but quickly become computationally intractable when the number of degrees of freedom of the robot exceeds three. Here, only the two-dimensional case will be presented. The algorithms can be easily extended to higher dimensions.

The simplest navigation function labels every cell of the configuration space grid with its L^1 distance to the goal. This distance refers to the shortest collision-free path from that cell to the goal configuration. This navigation function, called *NF1* (Barraquand and Latombe 1991), can easily be computed by a wavefront-expansion algorithm proceeding as follows: The cell corresponding to the goal configuration is labeled with the value zero. All unlabeled neighbors of that cell not contained within a configuration space obstacle are labeled with the next highest integer value. This process continues until every cell in the connected region of free space containing the goal configuration is labeled. By following the gradient of the resulting potential from any point in the connected region of free space containing the goal, the robot will reach the goal.

Since all cells are labeled with the distance of the shortest path to the goal, following the gradient will result in the shortest possible paths. Such a path will graze $\mathcal{C}$-obstacles. This is not desirable as execution uncertainties might lead to collisions. An improved numerical, grid-based navigation function, called *NF2* (Barraquand and Latombe 1991), causes the robot to have maximum clearance from obstacles. The resulting paths, however, will not be the shortest possible.

The computation of NF2 begins by extracting a one-dimensional subset of of the configuration space, called the *skeleton*. Every point on the skeleton has the property that at least the two closest obstacles are at the same distance from it. It is similar to the Voronoi diagram (Preparata and Shamos 1985). This skeleton can be computed with a wavefront-expansion similar to the one described above. The boundary of all configuration space obstacles is labeled with a value of zero. A wave is expanded from the boundary, labeling neighboring cells in free space with increasing values, until a labeled cell emanating from a different obstacle feature is encountered. Then that cell is labeled as belonging to the skeleton.

The goal configuration is connected to the skeleton by labeling every cell following the steepest ascent, starting form the goal, as belonging to the skeleton. Now another wavefront expansion is performed. Starting from the goal configuration, the wavefront expands only through cells of the skeleton. This wavefront leaves each cell of the skeleton labeled with its L^1 distance to the goal configuration. That distance only considers paths that lie entirely on the skeleton.

The final wavefront expansion creates the navigation function NF2. It expands from the skeleton and recursively labels every unlabeled cell in free space with increasing values. This results in a local minima-free potential function that will cause the point representing the robot to move from any point in the connected region of free space containing the goal to the skeleton, and then following the skeleton to the goal configuration. The robot will keep maximum clearance from obstacles along the path.

Due to the fact that the entire configuration space needs to be discretized and traversed, even multiple times for the computation of NF2, the application of numerical navigation functions to articulated robots with many degrees of freedom is not practical. For mobile robots they achieve very good results.

3.1.4 Randomized Planning

One way to solve the problem of local minima in the potential field approach is to create local minima-free potential functions, as described in Section 3.1.3. There are also extensions to the potential field approach solving this problem without changing the potential function.

By modifying the steepest descent procedure it is possible for the planner to evade local minima. At every step of the steepest descent process towards the goal, all unexplored neighbors of the current configuration are considered as candidates for the next move. They are searched in a best-first fashion (Ginsberg 1993). If no local minimum is encountered along the path, this algorithm will explore the same configurations as the original version of the potential field planner. Otherwise, the algorithm backtracks and all configurations inside the local minimum will eventually be considered, until exploration results in a path evading it. This method can be viewed as "filling up" the local minima and corresponds to an exhaustive search of all possible trajectories inside of it.

If the computational expense of exploring the entire local minimum is to be avoided, a randomized scheme can be employed to escape from it (Barraquand and Latombe 1990). This approach performs a sequence of random moves in the free space. The number of moves has to be chosen large enough for the search to escape the local minima, but small enough for the search not to invalidate too much of the previous plan. Special heuristics for this random motion include the reflection at configuration space obstacles (Horsch, Schwarz, and Tolle 1994).

3.1.5 Dynamic Environments

Special planning algorithms have been developed to address motion planning in dynamic environments. Environments are considered dynamic if obstacles move on predetermined trajectories. The knowledge of the trajectories is necessary because the planning algorithm needs to be able to anticipate their motion to generate a valid trajectory prior to motion execution. Please note that the meaning of the word *dynamic* differs here from the rest of the thesis, where an environment is considered dynamic if obstacles move on *unpredictable* trajectories.

Earliest approaches to motion planning in dynamic environments applied well-known planning techniques to configuration-time space (Reif and Sharir 1985; Erdmann and Lozano-Pérez 1986; Fujimura and Samet 1989; Fiorini and Shiller 1998). In this space the different motion planning paradigms, like visibility graph or cell decomposition, can be employed to generate paths. A path in configuration-time space inherently constitutes a trajectory, since it was generated in a time-parameterized space.

Planning in dynamic environments can also be performed by initially planning a path in a static environment (Kant and Zucker 1986; O'Donnell and Lozano-Pérez 1989). Subsequently, that path is time-parameterized such that all collisions are avoided. This might include a suspension of the robot's motion. This approach, however, might fail to find a solution to a planning problem even though one exists.

3.2 Execution Paradigms

According to the distinction made in Section 2.3 between motion planning algorithms and motion execution algorithms, motion execution algorithms incorporate the most recent sensor information to determine the motion command "on the fly". This results in *reactive* behavior: the motion of the robot "reacts" or is updated according to changes in the environment. Although avoiding collisions in a dynamic environment is a desirable property of motion generation algorithms, reaching a global goal is the main objective for most tasks. Only considering sensory data and not incorporating connectivity information about the free space usually cannot guarantee that this objective is met. This section introduces various execution paradigms; they all exhibit this shortcoming.

3.2.1 Potential Fields

The simplicity of the *artificial potential field approach* (Khatib 1980; Khatib 1986) described in Section 2.2.3 makes it a compelling paradigm for motion execution (Krogh 1984; Arkin 1987; Tilove 1990). It its most basic version, however, it is susceptible to local minima and under certain circumstances can result in undesirable behavior. Extensions of the basic approach aiming at solving those problems are presented here.

In the potential field approach the motion of a robot is generated by simulating the motion of a charged particle surrounded by obstacles of the same charge and a goal point with opposite charge. An entirely physically based simulation will lead to undesirable motion. When moving parallel in close proximity to an obstacle boundary towards the goal, the robot will react to a lateral repulsive force by moving away from the obstacle to increase the lateral clearance. Due to the direction of motion, however, this is not necessary: the robot's motion has no component towards the obstacle. The resulting behavior therefore seems unnatural.

To address this problem the *generalized potential field approach* was introduced (Krogh 1984). It takes into account the direction of motion to determine the effect of repulsive forces on the robot. This approach can also help to avoid oscillatory behavior if the robot moves in narrow passages between multiple obstacles and hence is exposed to repulsive forces of opposite directions.

To improve the trajectory resulting form the physically based simulation, the robot can be considered as a unit mass. The dynamics associated with this mass will result in a smoother trajectory (Tilove 1990). Other extensions with similar results incorporate rotation and task potentials to yield more natural trajectories (Khatib and Chatila 1995).

A more serious limitation of potential fields are local minima and much effort has been dedicated to eliminating this drawback. Since the chosen potential function is responsible for creating local minima, it is a natural candidate in the search for a solution to this problem. Navigation functions (Koditschek 1987; Rimon and Koditschek 1989; Rimon and Koditschek 1990) presented in Section 3.1.3 are one possible solution.

Another variation of the potential function are harmonic potentials (Kim and Khosla 1991; Feder and Slotine 1997). Harmonic potential functions model the flow of incompressible fluids to generate the motion of a robot. A sink is imagined at the goal configuration and incompressible fluid is flowing from the boundaries of the configuration space towards the sink. This results in a local minima-free, two-dimensional potential function that is free of local minima and from any point in free space will generate a motion to the goal position. Harmonic potentials can be considered physically inspired navigation functions.

Due to the computational complexity involved when simulating the flow of incompressible fluids, the dimensionality of this potential function has been limited to two in the literature. To generate motion for robots with many degrees of freedom, a number selected points are subjected to the potential field; these points are called *control points* (Khatib 1986). Using more that one control point can lead to *structural local minima* (Kim and Khosla 1991). Those are configurations in which the forces acting on the different control points of the robot cancel each other and do not result in motion. Hence, the problem of local minima remains for robots with more than two degrees of freedom.

Another attempt to remedy the problem of local minima resulted in the circulatory fields approach (Singh, Stephanou, and Wen 1996). Obstacles are surrounded by a magnetic field caused by a fictitious current flowing through their surface. The robot navigates around an obstacle by aligning itself with this field. It has been shown (Singh, Stephanou, and Wen 1996) by examples that global convergence is achieved. Collision avoidance behavior has not been proven formally, making this a preliminary result. For this method to be applicable, obstacles have to be represented in such a way that the magnetic field computation is possible.

3.2.2 Search Space Approaches

In the potential field approach, all information available about the environment is processed into a single motion command. Only the obstacles that exert repulsive forces onto the robot have an influence on this process. Much information about the state of the world is lost due to this restriction.

The *vector field histogram approach* (Borenstein and Koren 1991) incorporates sensor data at various levels of abstraction to use as much information about the local environment as possible. Based on an occupancy grid, obstacle densities are calculated for different steering angles. Low obstacle densities represent allowable steering angles. An allowable steering angle resulting in motion towards the goal is used as a motion command.

Extending this approach, *parameterized path families* (Feiten, Bauer, and Lawitzky 1994), or more specifically *steer angle fields*, take the nonholonomic kinematic constraints of the robot into account when choosing a motion. This reduces the search space and makes the approach more efficient. As opposed to just considering a steering angle, a kinematic simulation of the robot motion is performed to check for collisions with the environment. Again, a collision-free motion command is chosen that reduces the distance between the robot and the goal.

The *curvature-velocity method* (Simmons 1996) and the *dynamic window approach* (Fox, Burgard, and Thrun 1997) are based on the steer angle field approach. In addition to kinematic constrains these frameworks take into account dynamic constraints to reduce the search space even further. Both of these methods control velocity and acceleration of the robot in dependence of the local environment. The dynamic window approach has been integrated with a gross motion planner (Thrun et al. 1998) and was extended to incorporate a map in conjunction with sensory information to generate collision free motion (Fox et al. 1998). The dynamic window approach will be further developed in Chapter 4 and a more detailed presentation will be given in Section 4.1.

Although the above approaches yield good results for obstacle avoidance, the problem of local minima persists. These methods can fail to generate a motion that leads the robot to the desired goal configuration.

3.2.3 Other Approaches

For completeness some other approaches to motion execution are mentioned here. A Bayesian approach to obstacle avoidance was linked with global path planning (Hu and Brady 1994). Methods from machine learning have been applied to similar problems (Handley 1993). Lastly, neural networks have found broad application to the problem of generating motion commands from sensory data (Yang and Meng 1999).

3.3 Linking Planning and Execution

All motion planning algorithms presented here assume the environment to be known and static. This is necessary because the planning process is completed before the motion is executed. In most practical applications of robotics the environment is dynamic, however, and obstacles do not follow predictable trajectories. This implies that whenever such a change occurs in the environment, given that it can be detected and quantified, the planning process would have to be repeated (*replanning*) for the updated model. This constitutes a shortcoming of motion planning algorithms that limit their applicability to the robust generation of robot motion in dynamic environments. This limitation can partially be addressed by planning in configuration-time space, if the motion of all obstacles is known a priori.

The motion execution algorithms presented in Section 3.2 cannot completely fill this gap: They are either susceptible to local minima, as the regular potential field approach and all search space approaches presented in Section 3.2.2, or, like the potential field approach with harmonic potentials, they inherently depend on a rather complicated precomputed representation of the environment to generate a local minima-free potential function. Hence, they can either be applied to dynamic environment but fail to assure convergence to the goal, or can only be applied to static or predictable environments.

None of the presented algorithms completely solves the problems that motivated the work in this thesis and were described in Chapter 1. The following chapter will explore a novel approach to robot motion generation for mobile robots, combining the advantages of motion planning algorithms and motion execution algorithms by integrating them into one unified framework.

Approaches were presented in the robotics literature that link planning and execution paradigms to design robust algorithms for robot motion generation in completely or partially unknown and dynamic environments. They address certain issues of the general problem of motion planning in dynamic environments with unpredictably moving obstacles. Employing planning and execution methods in conjunction partially solves those issues. These approaches are presented in this section

3.3.1 Channels

A reactive method for planning and execution of robot motion with incomplete knowledge of the environment is based on the notion of *channels* (Choi and Latombe 1991). Approximate cell decomposition is used to determine a sequence of cells that connect the initial and the final configuration of the robot. The sequence of cells is called channel and represents a set of feasible paths. The channel has been obtained using a motion planning technique. The robot navigates through the channel using a potential field approach. Small, unforeseen deviations from the world model can thus be circumnavigated. To avoid local minima, intermediate goal configurations are defined at cell boundaries. Should the channel be obstructed, however, as would often be the case with moving obstacles of considerable size, replanning becomes necessary. This method is suited best for generation of robot motion in environments that contain small, unknown obstacles.

3.3.2 Subsumption Architecture

The *subsumption architecture* (Brooks 1986) provides a general framework for the integration of motion planning and motion execution algorithms. It does so by allowing various tasks at different levels of abstraction to interface to each other. On the lowest level obstacle avoidance algorithms keep the robot away from objects in the environment. Higher levels of abstraction *subsume* lower levels and determine the robot's behavior. This allows to execute a preplanned motion which is suspended to evade an unforeseen obstacle. During the evasion the planner can generate a new path, given the updated model of the world.

3.3.3 Elastic Bands

Elastic bands (Quinlan and Khatib 1993a; Quinlan 1994b) are a quite general framework for the execution of planned robot motion. They represent a path, previously generated by a planning algorithm, as a one-dimensional curve in configuration space with the properties of an elastic band. Obstacles exert a repulsive force on that band and deform it as they approach. Internal forces resulting from rubber-like properties cause the elastic band to contract as obstacles recede. Thus, a plan can be executed in dynamic environments with unpredictably moving obstacles. This framework is treated in greater detail in Section 6.3.1.

The elastic band approach has been extended to nonholonomic mobile robots and integrated with sensor-based navigation (Khatib 1996; Khatib et al. 1997; Jaouni et al. 1997).

3.3.4 Real-Time Planning

For the low-dimensional configuration spaces of mobile robots algorithms have been devised that allow real-time modification of an existing plan to reflect changes in a grid-based environment (Kim and Khosla 1991; Schmidt and Azarm 1993; Azarm and Schmidt 1994; Feder and Slotine 1997). These approaches are based on the simulation of physical effects such as the flow of liquids or the diffusion of gas. Computational complexity restricts their applicability to low-dimensional configuration spaces.

Chapter 4

Global Dynamic Window Approach

As discussed in previous chapters, algorithms that generate motion for mobile robots can be divided into motion planning algorithms and motion execution algorithms. Planning algorithms consider a model or map of the environment to compute a path from the robot's current position to the goal, prior to issuing motion commands. Motion execution algorithms, on the other hand, use current sensory information to determine a motion command for the next instant in time. The motion command is selected such that collision with obstacles in the environment are avoided.

For most applications in mobile robotics the environment is partially or completely unknown and changes with time. Under such circumstances the paths generated by planning algorithms can become invalidated and replanning is required to reach the goal. Since planning is too time-consuming to avoid collisions in real-time, motion commands for mobile robots are usually generated by computationally efficient motion execution approaches. Due to local minima, however, these approaches may not result in the behavior required to accomplish the robot's task.

In this chapter the *global dynamic window approach* (Brock and Khatib 1999b) is introduced. This novel approach combines motion planning algorithms based on numerical navigation functions and motion execution algorithms in one single framework. Using the global dynamic window approach, mobile robots can autonomously navigate safely and robustly at high velocities in unknown and dynamic environments. This section first presents the dynamic window approach. It is subsequently extended to the *holonomic* dynamic window approach and to the *global* dynamic window approach.

4.1 Dynamic Window Approach

The *dynamic window approach* (Fox, Burgard, and Thrun 1996; Fox, Burgard, and Thrun 1997) is a motion execution method. Based on perceived sensory data a motion command is determined that avoids collision with obstacles while making progress towards the goal configuration. The approach is applicable to *synchro-drive robots*, i.e. car-like robots with zero turning radius. To allow robot motion at high velocities, it is desirable to accurately predict the robot's behavior in response to a new motion command. This is achieved by performing a predictive dynamic simulation of all possible motion commands in a discretized search space and then selecting the one that maximizes a given quality function.

4.1.1 Search Space

When selecting possible motion commands for dynamic simulation from the discretized search space the kinematic constraints of the robot are taken into account by directly searching the velocity space of a synchro-drive robot, given actuator limitations. The search space is the set of tuples (v, ω) of translational velocities v and steering velocities ω that are achievable by the robot. Among all velocity tuples those are selected that, if selected and executed, would allow the robot to come to a stop before hitting an obstacle, given the current position, the current velocity, and the acceleration capabilities of the robot. These velocities are called *admissible velocities*.

The search is further restricted by a *dynamic window*, giving this method its name. It reflects the dynamic limitations of the robot. The dynamic window contains those admissible velocities that can be achieved by the robot, given its current velocity and its acceleration capabilities, within a given time interval. This time interval corresponds to a servo tick of the control loop. Figure 4.1 illustrates the subdivision of the search space in the dynamic window approach. The dynamic window is a rectangle, since acceleration capabilities for translation and steering are independent.

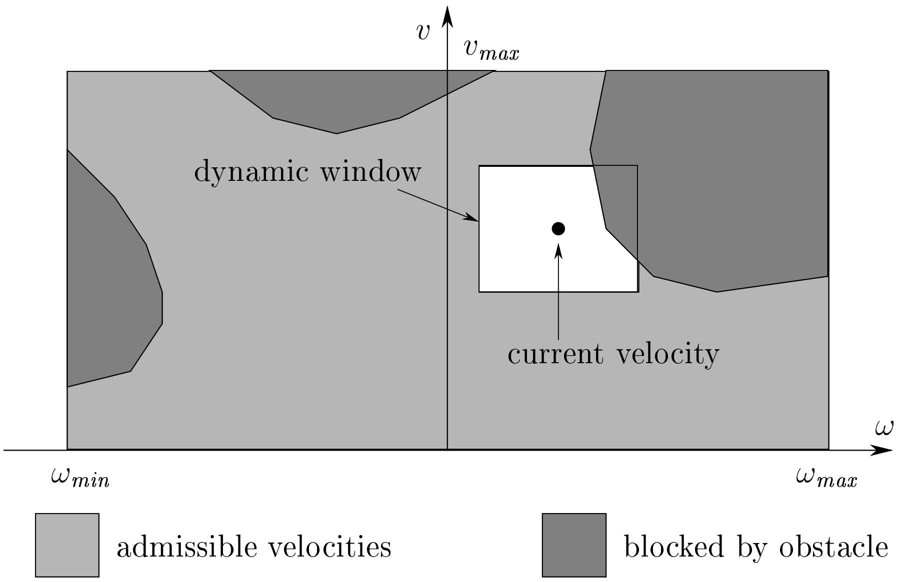


Figure 4.1: Search space in the dynamic window approach: The space of all possible velocity commands is divided into admissible and blocked regions. Only physically achievable velocity commands are considered for the next servo tick; they form the dynamic window around the current velocity.

4.1.2 Objective Function

To determine the next motion command all admissible velocities within the dynamic window are considered. Among those, a desired velocity $\vec{v} = (v_x, v_y) = (\dot{x}, \dot{y})$ and an acceleration $\vec{a} = (a_x, a_y) = (\ddot{x}, \ddot{y})$ are selected from the search space according to the objective function

$$\Omega(\vec{s}, \vec{v}, \vec{a}\,) = \alpha \cdot align(\vec{s}, \vec{v}\,) + \beta \cdot vel(\vec{v}\,) + \gamma \cdot dist(\vec{s}, \vec{v}, \vec{a}\,), \tag{4.1}$$

where $\vec{s} = (x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y})$ is the current state of the mobile base, describing its position, velocity, and acceleration. This objective function is a linear combination of three functions. The ranges of those functions are normalized to the interval $[0, 1]$.

To favor trajectories that are directed towards the goal, the function $align(\vec{s}, \vec{v}\,) = 1 - \frac{|\theta|}{\pi}$, where θ is the angle between the direction of motion and the goal heading, results in large values for good alignment with the goal heading. The goal heading is modified if the robot's lateral distance to an obstacle becomes too small.

The function $vel(\vec{v}\,)$ is defined as $\frac{||\vec{v}||}{v_{max}}$, where v_{max} the maximum velocity the robot can achieve. It will favor high velocities over low ones.

The value of the function $dist(\vec{s}, \vec{v}, \vec{a}\,)$ is determined by dynamically simulating the robot's motion response to the new motion command commanding velocity $\vec{v}$ using acceleration $\vec{a}$. The simulation determines the length of the trajectory from the robots current position executing the motion command until hitting an obstacle in the environment.

Using this approach robust obstacle avoidance behavior has been demonstrated at high velocities (Fox, Burgard, and Thrun 1997). However, since the dynamic window approach only considers goal heading and no connectivity information about the free space to determine the motion command, it is still susceptible to local minima.

4.2 Holonomic Dynamic Window Approach

Holonomic robots have several advantages over car-like and synchro-drive robots. Since they allow instantaneous acceleration in all directions they are much easier to control and have an increased maneuverability. The orientation of the robot can be controlled independently of its motion in the plane. In addition, the equations of motion have a simple closed-form solution. This provides the motivation for the extension of the dynamic window approach to holonomic robots, presented in this section. In Section 4.3 the holonomic dynamic window approach will be integrated with a global planning method to result in the global dynamic window approach.

4.2.1 Search Space

The most important difference between the dynamic window approach and the holonomic dynamic window approach is the overall search space. A holonomic robot has no limitations on the direction of instantaneous acceleration. However, it is impractical to search the entire space of possible velocity changes. Therefore, a subset has to be selected that exploits the kinematic advantages of holonomicity, while retaining computational feasibility.

For the holonomic dynamic window approach, the search space consists of all possible velocities in a global reference frame. It is discretized in polar coordinates, choosing a fixed set of directions and scalar velocities. Assuming isotropic acceleration and velocity capabilities, this results in a circular search space and a circular dynamic window, as depicted in Figure 4.2. The dynamic window is circular as a consequence of the caster-based drive system used in the experiments. By selecting a set of directions on the unit circle, the advantage of omnidirectional acceleration is maintained without incurring a high computational burden.

The use of a global reference frame allows the decoupling of the two translational axes, yielding the following equations of motion for constant acceleration $\vec{a} = (a_x, a_y)$ and velocity $\vec{v} = (v_x, v_y)$:

$$x(t_i) = x(0) + v_x t_i + \int_0^{t_i} a_x t \, dt = x(0) + v_x t_i + \frac{1}{2} a_x t_i^2$$

$$y(t_i) = y(0) + v_y t_i + \int_0^{t_i} a_y t \, dt = y(0) + v_y t_i + \frac{1}{2} a_y t_i^2.$$

These equations show that when accelerating from a constant velocity to achieve a given velocity command, the robot describes a quadratic curve until the desired velocity is attained. The curvature of those curves depends on the magnitude of the acceleration. In order to achieve curves with low curvature the two-dimensional search space shown in Figure 4.2 is searched for different accelerations. Low accelerations result in low curvature and allow to imitate car-like behavior. If the accelerations are chosen such that $a_{xy} = a_x = a_y$ the resulting overall search space is three-dimensional and a motion command is defined by $\vec{v} = (v_x, v_y)$ and $\vec{a} = (a_{xy}, a_{xy})$.

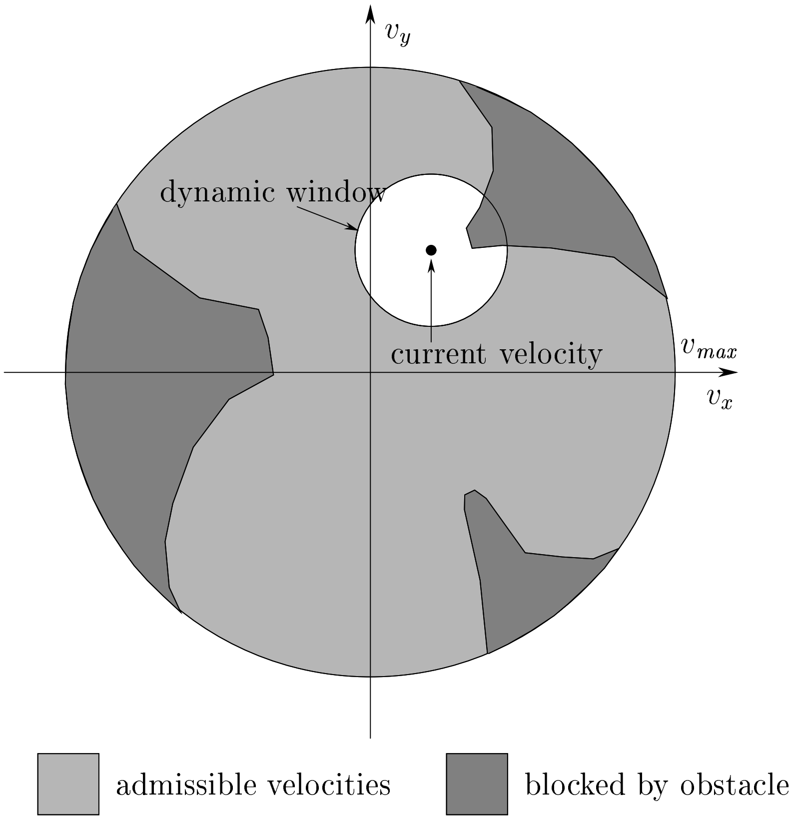


Figure 4.2: Search space in the holonomic dynamic window approach: For a holonomic, caster-driven base the space of possible velocities forms a circle; this is due to identical acceleration capabilities in all directions. Similarly to the dynamic window, the overall search space is divided into admissible and blocked velocities, based on whether they will result in collision or not. The dynamic window is circular, assuming that the robot has identical acceleration capabilities in all directions.

To determine if a motion command $(\vec{v}, \vec{a})$ is admissible, the length of the resulting trajectory has to be determined. Simulation of the base motion according to the equations of motion will determine the duration t_i of the trajectory until hitting an obstacle. The length $l(\vec{v}, \vec{a}, t_i)$ of the trajectory can then be computed analytically:

$$\begin{aligned} l(\vec{v}, \vec{a}, t_i) &= \int_0^{t_i} v_{t_i} \, dt \\ &= \int_0^{t_i} \sqrt{(v_x + a_x t)^2 + (v_y + a_y t)^2} \, dt. \end{aligned}$$

If the length of the trajectory permits the robot to come to a halt after moving for the duration of one servo tick and before hitting an obstacle, the motion command is considered admissible. When the environment is not known a priori only those motion commands are considered admissible that allow the robot to stop inside the explored region. In a more aggressive approach, the robot could consider unexplored regions empty; this is what humans do when turning around corners in hallways.

4.2.2 Objective Function

A desired velocity $\vec{v} = (v_x, v_y)$ and an acceleration $\vec{a} = (a_x, a_y)$ are selected from the search space according to the objective function

$$\Omega(\vec{s}, \vec{v}, \vec{a}) = \alpha \cdot \mathit{align}(\vec{s}, \vec{v}) + \beta \cdot \mathit{vel}(\vec{v}) + \gamma \cdot \mathit{dist}(\vec{s}, \vec{v}, \vec{a}) + \delta \cdot \mathit{goal}(\vec{s}, \vec{v}, \vec{a}), \tag{4.2}$$

where $\vec{s} = (x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y})$ is the current state of the mobile base, describing its position, velocity, and acceleration. This objective function is very similar to equation 4.1 on page 43: it is a linear combination of four functions that are normalized to the interval $[0, 1]$.

To favor trajectories that are directed towards the goal, the function $\mathit{align}(\vec{s}, \vec{v}) = 1 - |\theta|/\pi$ is slightly modified from before. While it still results in large values for velocities directed towards the goal, it can favor slightly modified headings when the robot's lateral distance to an obstacle becomes too small. This causes the robot to keep a safe distance from obstacles aligned with the goal heading. Oscillatory behavior can be avoided using hysteresis.

The function $vel(\vec{v})$ is defined as follows:

$$vel(\vec{v}) = \begin{cases} \frac{\|\vec{v}\|}{v_{max}} & \text{if robot is far from goal} \\ 1 - \frac{\|\vec{v}\|}{v_{max}} & \text{if robot is close to goal} \end{cases},$$

where v_{max} the maximum velocity the robot can achieve. It will favor high velocities if the robot is far from the goal and low velocities when it is close. This solves a shortcoming of the dynamic window approach that would overshoot a goal configuration as high velocities were always preferred.

The function $dist(\vec{s}, \vec{v}, \vec{a})$ is again defined as in the dynamic window approach, except that dynamic simulation can be performed much more efficiently, as the equations of motion have a simple closed form.

If the trajectory that results from the motion command $(\vec{v}, \vec{a})$ passes through the goal region, the value of the binary function $goal(\vec{s}, \vec{v}, \vec{a})$ is 1 , otherwise it is 0.

$$goal(\vec{s}, \vec{v}, \vec{a}) = \begin{cases} 1 & \text{if path goes through goal} \\ 0 & \text{otherwise} \end{cases}$$

This additional function causes short trajectories that lead through the goal to be favored over very long ones that pass by it.

When two motion commands achieve the same value of the objective function the one with higher acceleration is preferred. The parameters α, β, γ and δ can be adjusted to modify the behavior of the robot.

4.3 Global Dynamic Window Approach

The dynamic window approach and the holonomic dynamic window approach are both susceptible to local minima. The robot's motion with respect to the goal is only influenced by the goal heading. As discussed in Section 2.3, this limitation can be removed by incorporating information about the connectivity of the free space into the selection of a motion command.

The *global dynamic window approach* presented in this section extends the dynamic window approach (Fox, Burgard, and Thrun 1997) and the holonomic dynamic window approach presented in Section 4.2 by incorporating a simple and efficient motion planning algorithm. The global planning is efficient enough to be executed for each servo tick of the motion controller. This allows robot navigation in real-time in a globally goal-directed fashion.

The global dynamic window approach assumes no prior knowledge about the environment. It nevertheless achieves robust motion generation and execution in dynamic environments with unpredictably moving obstacles. If additional information about the environment was available, it could easily be incorporated into the motion selection algorithm.

4.3.1 Free Space Connectivity

To exploit information about the connectivity of the free space, a model of the environment is required. The model-based dynamic window approach (Fox, Burgard, Thrun, and Cremers 1998) incorporates information from sensory data and a map of the environment to determine collision-free motion. A similar technique could also be adapted for the global dynamic window approach. The work presented here, however, is restricted to the case where no a priori knowledge about the environment is available and hence global planning algorithms cannot be applied.

To collect information about the connectivity of the free space, sensory information is merged into a map. For the simplicity of presentation, it shall be assumed that the mobile robot can be modeled as a disc. The case of polygonal robots will be discussed later.

In order to achieve real-time performance for the overall algorithm, no preprocessing of the sensory data is performed. Sensory data acquired from range sensing is used as point data. This data is translated into configuration space obstacles that are represented in an occupancy grid. For a circular robot this is simply done by blocking a circular region of the two-dimensional configuration space of the diameter of the robot. Now the robot can be represented as a point.

This simple approach is motivated by the fact that only connectivity information about the free space is needed. The map will be constantly updated with current sensor information so that no absolute position information is necessary for safe navigation. Nevertheless, the mobile base used in the experiments presented in Chapter 5 has little slippage and resulting maps are very accurate.

If the mobile robot has the shape of a polygon, a three-dimensional configuration space is needed. In this space the computational complexity of generating configuration space obstacles is increased significantly. To solve this problem the robot could be represented by two concentric circles about the origin of the reference frame of the robot: the largest contained circle with radius r_1 and the smallest containing circle with radius r_2. Configuration space obstacles can be generated as before, assuming r_1 as the radius of the robot. A configuration is now free of collisions if it is at least r_2 away from obstacles. If the distance is between r_1 and r_2, special collision checking methods have to be evoked. For distances smaller than r_1, a collision is detected. Alternatively, a three-dimensional configuration space could be computed for the robot.

4.3.2 Navigation Function

Since the environment is represented by an occupancy grid, a grid-based navigation function is a natural and efficient choice for a global planning algorithm. The global dynamic window approach combines the dynamic window approach for reactive motion execution with the global, local minima-free numerical navigation function NF1 (Dorst and Trovato 1989; Barraquand and Latombe 1991; Latombe 1991). This function is computed using a wave-propagation technique starting at the goal. It labels cells in the occupancy grid with the L^1 distance to the goal, taking into account obstructions by obstacles. The result is a local minima-free potential function with a unique minimum at the goal. It is described in more detail in Section 3.1.3.

Employed with classical motion planning algorithms the navigation function NF1 has the disadvantage of producing trajectories that graze obstacles. Selecting motion commands using the global dynamic window approach eliminates this problem, since a minimum clearance from obstacles is maintained.

The classical motion planning algorithm (Barraquand and Latombe 1991; Latombe 1991) computes the NF1 for the entire occupancy grid. This is motivated by the fact that the same NF1 can be reused for every location of the robot, as long as the environment does not change. The global dynamic window approach recomputes the NF1 each time a motion command is selected, updating the world model as it is being explored, thereby allowing the robot to operate in unknown and dynamic environments. Hence, it is not necessary and not desirable to compute NF1 for the entire grid. Instead, the NF1 is computed in a rectangular region aligned with the goal heading. The width of this rectangular region is increased until the robot's current position is reached by the wave front. A navigation function computed for a subsection of the configuration space is referred to as *localized navigation function*.

Figure 4.3: Rectangular expansion of navigation function computation: The numerical navigation function is only computed for a narrow region of the free space connecting the goal and the current configuration. If this path is blocked by obstacles, as shown on the right, this region is widened incrementally until either the two configurations are connected or a failure is detected.

Figure 4.3 shows a narrow NF1 for an unobstructed path and a wider NF1 for an obstructed path. The NF1 is shown as gradient colors, the robot is the black dot at the bottom of the NF1 and the goal position is the gray dot in the darker region of the NF1, obstacles are shown in black.

When the width of the region in which the NF1 is computed is increased, partial recomputation may become necessary. This is due to the fact that the restriction in width may cut off the shortest path to a region of the configuration space. In fact, it is possible to construct a pathological case for which almost the entire region considered has to be recomputed each time the width is increased. Such an environment would have to contain many paths between the robot and the goal region and those paths would have to get shorter as the rectangular region widens. For those cases the heuristic of the rectangular region obviously performs very poorly. In practice, however, there are only few paths to the goal region and they tend to be longer if the rectangular region needs to be widened for them to be explored. Hence, for practical environments this heuristic greatly reduces the number of grid cell that need to be explored.

The computation of this *localized NF1* proceeds initially as described in Section 3.1.3: starting with the value zero at the goal configuration, neighboring cells are labeled with increasing numbers until the entire rectangle is filled or the current configuration of the robot is reached. With this extended algorithm it might be possible that the robot's current configuration cannot be reached by a path fully contained within the current rectangle. In this case, the width of the rectangle needs to be increased. To be able to resume the computation of the NF1, cells on the border of the rectangular region are remembered in the order they were encountered. These cells are then used to restart the computation for the widened rectangle.

To determine which of the cells have to be recomputed after the rectangle was widened, a time stamping mechanism is used. In addition to the value of the potential function, each cell in the grid representing the configuration space is assigned a time stamp. The time stamp used for the first rectangle is considered the beginning of the current *era* – the computation of the localized NF1 for the current environment. Each time the rectangle is widened, the value of the time stamp for cells that are being labeled is increased.

During the wavefront expansion algorithm, the time stamp is used in conjunction with the potential value of a cell to determine whether it needs to be labeled or not. Cells of a previous era are always labeled. Cells of the current era with a time stamp smaller than the current one and a potential value higher than the one it would get in the current labeling are relabeled. Those cells are the ones that could be reached by a detour in the narrower rectangle and now are reachable by a shorter path in the widened rectangle. Cells with the same time stamp are never relabeled. This mechanism allows the occupancy grid representing the configuration space to be used in a very efficient manner, as data can simply be outdated by increasing the time stamp or declaring a new era.

4.3.3 Objective Function

The objective function for the holonomic dynamic window approach as given in equation 4.2 in Section 4.2.2 can easily be modified to incorporate the navigation function described in Section 4.3.2. The function $align(\vec{s}, \vec{v})$ is replaced by the function $nf1(\vec{s}, \vec{v})$. This function's value increases if $\vec{v}$ is aligned with the gradient of the navigation function at the robot's location, rendering the global dynamic window approach immune to local minima, since NF1 is a local minima-free navigation function.

In addition, the function $\Delta nf1$ is added to the objective function. Its value indicates by how much a motion command is expected to reduce the value of the NF1 during the next servo tick. This favors motion commands that quickly reduce the distance to the goal. Note that the navigation function is not necessarily computed for the entire free space. If a trajectory would move the robot off the part of the map for which the navigation function has been computed, the smallest potential value along that trajectory is used. The objective functions Ω_g for the global dynamic window is then defined as:

$$\Omega_g(\vec{s}, \vec{v}, \vec{a}) = \alpha \cdot nf1(\vec{s}, \vec{v}) + \beta \cdot vel(\vec{v}) + \gamma \cdot dist(\vec{s}, \vec{v}, \vec{a}) + \delta \cdot goal(\vec{s}, \vec{v}, \vec{a}) + \epsilon \cdot \Delta nf1(\vec{s}, \vec{v}, \vec{a}).$$

The value of $nf1(\vec{v}, \vec{s})$ can be determined by examining the neighbors of the grid cell that corresponds to the robot's location. However, since NF1 is grid-based, its gradient can only be a multiple of 45°, resulting in unnatural behavior along passages that are not grid-aligned. By examining neighbors at a constant distance from the cell that corresponds to the robot's position, this behavior can be improved. In certain situations the numerical navigation function NF1 can cause to robot to move close to obstacles, following their boundaries. To avoid the grazing of obstacles typical for NF1, lateral clearance is ensured by changing the gradient of the localized NF1 in certain cases. Using hysteresis, oscillatory behavior is avoided.

4.4 Integration of Map-Building

The global dynamic window approach as described above only uses a map to represent the connectivity of the free space. For robust navigation the approach relies on the fact that the surroundings of the robot are sensed constantly and that the map is updated at a high rate. Using an accurate sensor, this implies that the map describing the area of the environment around the robot is very accurate and contains the same information as the current sensor readings. Regions further away correspond to older sensory information. Those regions will be explored again, as the robot enters them. Thus, even by employing only simplistic map building schemes, the robot can navigate robustly.

In the global dynamic window approach without sophisticated map-building techniques, the goal configuration is detected using the robot's odometry. The accuracy of the robot's integrated position information decreases with an increasing length of the trajectory. If the robot is navigated to several goal configurations subsequently without relocalizing, odometry can become a very poor means of inferring the actual position of the robot in the world. Once the robot's real position and the integrated position information differ by a large amount, the specification of goal coordinates in a global coordinate frame will be insufficient to attain that goal.

This approach, however, is not very general. The integration of more sophisticated map-building algorithms and relocalization schemes (Moutarlier and Chatila 1991; Shatkay and Kaelbling 1997; Thrun et al. 1998; Thrun et al. 1998) could reduce the impact of inaccurate odometry significantly. By building a qualitative map of the environment, the robot can be relocalized relative to recognizable landmarks in the environment and the accumulative errors in integrating the robot's configuration can be reduced or eliminated. If a complete or partial map of the environment is known, this information can also be integrated into the dynamic window approach (Fox, Burgard, Thrun, and Cremers 1998).

4.5 Multi-Resolution Global Dynamic Window

The size of the occupancy grid used to represent the map has a crucial impact on the computational complexity of the global dynamic window approach. To achieve high servo rates, it is desirable to use a coarse grid, resulting in fewer cells and faster execution. The representation of the environment at such a low resolution might fail to represent narrow passages the robot could pass through. Hence, there is a tradeoff between low resolution with inaccurate representation of the free space and low computational cost, and high resolution with accurate representation of free space and high computational cost.

The positive aspects of high and low resolution for environmental representation could be combined in a *multi-resolution global dynamic window approach*. This extension the the global dynamic window approach uses two maps at the same low resolution to represent the environment. A conservative one represents configuration space obstacles as before: a cell is labeled as blocked if the robot could be in collision with an obstacle for any configuration within that cell. In a second, less conservative representation, however, only those cells are labeled as blocked in which the robot would be in collision for any configuration contained by that cell.

When trajectories are checked for collision avoidance, the conservative world representation is used. If however, a trajectory should lead to collision in the conservative representation, but in the less conservative representation it is a trajectory with a very high value of the objective function, it becomes obvious that a desirable trajectory leads through a narrow passage in the environment. The lower resolution representation of the environment does not allow to verify if such a trajectory is free of collision or not. Locally, a global dynamic window at a higher resolution is generated. It can be used to navigate the robot through the narrow passage, if possible.

Switching between the multi-resolution dynamic windows requires additional computation. As the robot navigates through a narrow passage its speed will be reduced and a slightly reduced servo rate will still result in robust obstacle avoidance. Through the reduction of the servo rate the additional computation necessary for switching the resolution of the dynamic window can be accommodated.

Using a uniform resolution of $5 \times 5cm$, servo rates up to 40 Hz were achieved for small grid sizes (see Section 5). This implies that for many applications such a multi-resolution representation may not be necessary.

Chapter 5

Experimentation

5.1 Implementation

The global holonomic dynamic window approach introduced in the previous chapter has been implemented and tested on the Nomad XR4000 mobile base by Nomadic Technologies, Inc. shown in Figure 5.1. This base moves at omnidirectional translational velocities of up to $1.2\frac{m}{s}$ and accelerations of up to $1.5\frac{m}{s^2}$. It is equipped with a SICK laser range finder with a field of view of 180° and an accuracy of distance measurement of up to $1cm$. Holonomicity is achieved by a caster-based design of the drive system (Holmberg 1999; Holmberg and Khatib 1999).

The sensory data acquired using the laser range finder is collected in a map. The map is implemented using a simple binary occupancy grid. The navigation function NF1 can be computed efficiently for such a two-dimensional data structure. Using this map, a controller determines the next motion command, based on the current state of the robot and the goal configuration. The execution of the motion command is performed by the software library provided by Nomadic Technologies. The implemented algorithms relate very closely to the algorithms described in Chapter 4. To deal with the difficulties that arise when a theoretical idea is applied in practice and to overcome shortcomings of the hardware, some additional functionality was used to improve the performance. These will be briefly discussed, before presenting the experiments.

Figure 5.1: The Nomad XR4000 is a holonomic mobile base, driven by a caster-based drive system. It is equipped with a SICK laser range finder, measuring a field of view of $180°$ with an accuracy of up to $1cm$.

The dynamic window restricts the search space when determining the next motion command. If no command is found within the dynamic window, the robot is commanded to stop as fast as possible. This simple approach results in undesirable behavior in some situations. If the robot moves into a dead-end, for example, there might be valid motion commands within the dynamic window that explore this dead-end. The navigation function, however, already indicates that the desired motion leads out of the dead-end. Hence, the valid motion commands may reduce the distance to the goal and be free of collision, but do not reduce the value of the navigation function. It is desirable, however, to have the robot recognize that the only way out of the dead-end is to turn around. To achieve this behavior an extra heuristic was introduced in the global dynamic window approach. If the angle between the gradient of the navigation function and the selected motion command exceeds $90°$, the robot is commanded to stop. This corresponds to resetting the dynamic window to the origin of the search space. It will result in the consideration of all possible directions of motion. The result of this behavior is illustrated in Figure 5.3 in Section 5.2.

The robustness of the generated motion relies heavily on an accurate world model. Using a laser range finder measurements can be taken with very high accuracy. Those measurements are relative to the position of the sensor. As the base moves at high velocities, it becomes imperative to associate a time stamp with every taken measurement. Keeping track of the robot's position over time, the position of the sensor at the time the measurement was taken can be estimated. This significantly increases the accuracy of the measurement and therefore also the accuracy of the world model.

In the dynamic window approach, the selection of a motion command is based on dynamic simulation of a set of possible motion commands. The command that optimizes the objective function is chosen for execution (see Section 4.3.3). The value of the objective function is determined by simulating the effect of the motion command. During this simulation the position of the robot might change significantly. This would invalidate the previously computed simulation. To overcome this problem the duration of the entire simulation phase is estimated and the state of the robot is projected into the future. This state is then used to perform the simulations. Since the duration is only dependent on the portion of the map surrounding the robot, the value is not likely to change significantly between consecutive invocations of the algorithm. Therefore, the duration of the simulation for the previous step is used to estimate the duration for the current step.

5.2 Experiments

Using the on-board 450 MHz Pentium PC, servo rates of 5 to 40 Hz are achieved. The robot navigates reliably with very high velocities in tight, dynamic environments. In long but not necessarily wide open areas the base moves at velocities above $1.0\frac{m}{s}$.

Figure 5.2 shows two example executions of the global holonomic dynamic window approach. The two sets images show snapshots of the internal map representation, in which obstacles are shown in black. Obstacles are represented in configuration space; hence, the robot is represented as a point. The navigation function is shown as the gray shaded area. Darker values correspond to a lower value of the navigation function. The gray line indicates the trajectory of the robot.

In both examples velocities of above above $1.0\frac{m}{s}$ were achieved. The left four images represent a $6m \times 6m$ map, the right four images a $10m \times 10m$ map, both with grid cell sizes of $5cm \times 5cm$ for the occupancy grid. In both examples the robot started without any knowledge of the environment. The first image in both cases corresponds to the robot at rest; the obstacles shown are only those visible from its position. As the robot starts moving, obstacles are added to the map and the NF1 is recomputed correspondingly, until the robot reaches the goal.

The third image on the right side shows a situation in which sensory information indicates that the goal is obstructed. Hence, no NF1 can be computed and the global dynamic window approach reduces to the dynamic window approach. When updated sensory information shows that the goal is not obstructed, as can be seen in the fourth image, the NF1 is recomputed.

The execution of a more complicated example is shown in the eight images in Figure 5.3. Here, the black lines indicate equipotential lines of the navigation function. Notice that the navigation function is not smooth everywhere. This is an artifact of the computation procedure for the navigation function described in Section 4.3.2 and has no impact on the performance of the algorithm.

The image sequence shows how the plan initially represented by the navigation function is invalidated by the updated world model. In the second image of the sequence, the current plan leads through the narrow passage in the middle of the map. The next image indicates the path being blocked by obstacles. The new plan leads around the set of obstacles in the middle of the map to the goal configuration. Again, the robot starts out without any prior knowledge of the environment. It attempts to go to the goal configuration following the shortest possible path. When that path is discovered to be blocked, a new path is generated and executed without delay.

The experiment shown in Figure 5.4 demonstrates how the global dynamic window approach handles dynamic environments. Two groups of people are moving into the path of the robot. The navigation function is updated in real time and a collision can be avoided. The people move at about $1.0\frac{m}{s}$. As indicated by the trajectory, the robot moves on a smooth path towards the goal.

To determine the impact of the map size on the performance of the global dynamic window approach, another experiment was conducted with a large map. The map measures $30m \times 20m$ with a resolution of $5cm$. Using the conventional approach to computing the navigation function NF1 (Barraquand and Latombe 1991), up to one quarter of a million cells would have to be labeled during each servo tick. Using the approach described in Section 4.3.2 to compute the navigation function, no significant decrease in the servo rate of the controller is noticed. Only a small fraction of the map that needs to be explored to connect the goal configuration to the current configuration. Obviously, environments can be constructed that require the entire map to be explored and performance would degrade considerably.

Figure 5.5 shows the map and the trace of the robot at the end of this experiment. Some portion of the trajectory is covered by obstacles due to faulty sensor data. The map displays two parallel corridors connected by an open area in the left half of the image. The curvature of the upper corridor is a result of slippage of the base. The right portion of the open area is highly cluttered with chairs and tables. The bend in the trajectory is caused by the fact that using the laser range finder it cannot be detected immediately that the bottom right corner of the open area is closed off. The robot attempts to move there, detects the blockage, stops, and continues to move to the goal position, as indicated by the trajectory trace.

This experiment also demonstrates a limitation of the global dynamic window approach: the deformation of the straight corridors in Figure 5.5 is due to execution uncertainties, i.e. slippage of the base. Relying only on information from the encoders to determine the robot's current position is too limiting for long trajectories. To remedy this problem the global dynamic window approach can be integrated with relocalization schemes (Fox, Burgard, Thrun, and Cremers 1998).

In general, it is difficult to provide a precise and objective performance measure for the global dynamic window approach. One such measure is the servo rate that can be achieved. The duration of one servo tick, however, changes during an experiment. The computation of the navigation function, for example, can take a long time in a maze-like environment or be very efficient if the line of sight between the robot and the goal configuration is not obstructed by obstacles. Furthermore, the simulation of all possible motion commands can take a long time if the perimeter of the robot is free of obstacles, because the simulated trajectories are longer than in tight spaces. Hence, the servo rate of the global dynamic window approach is work space-dependent.The examples given above represent real-world environments and serve as a realistic test-bed for the global dynamic window approach. The servo rates achieved in all examples are between 5 and 40 Hz, sufficiently high to generate safe motion consistently, as the examples have shown.

The servo rate also depends to a large extent on parameters of the algorithm. The number of sensor readings entered into the map determine the duration of the map update. In addition, the resolution of the search space and the size of the dynamic window influence the performance of the global dynamic window significantly. Furthermore, the size of the map, the grid resolution, and the complexity of the environment, as mentioned above, have an impact on the servo rate. For all experiments reported in this chapter those parameters were instantiated with values that optimize the motion behavior and not the servo rate. Sensor readings were taken in intervals of one degree and a fine resolution of the search space was chosen to allow smooth motion. By changing those parameters much higher servo rates could be achieved, which might be desirable in environments with obstacles moving at high velocities.

The performance of the global dynamic window approach could also be measured subjectively by judging how natural and efficient the robot's motion is perceived to be. While this cannot be related in print, the global dynamic window approach causes the robot to move swiftly and navigate with great agility.

5.3 Comparison with Previous Approaches

The previous section presents experimental results for the global dynamic window approach. To emphasize the significance of the increase in motion capabilities obtained by the application of this new framework, this section presents some of the most prominent approaches found in the literature addressing similar problems, and compares their performance to the global dynamic window approach. This comparison is somewhat difficult, because most approaches address different sets of subproblems of the general motion generation task.

For the vector field histogram (Borenstein and Koren 1991), which is entirely sensor-based, very good experimental results have been presented. Moving down a straight corridor, the mobile robot CARMEL achieved a velocity of $0.78\frac{m}{s}$. As motion is generated based on sensor information, the approach is susceptible to local minima. Furthermore, in narrow passages oscillations between obstacles may occur. In comparison, the global dynamic window approach achieves velocities of above $1.0\frac{m}{s}$, limited by the actuator capabilities of the robot; the global dynamic window approach promises to be able to handle much higher velocities. The integration of global planning into the global dynamic window approach eliminates the susceptibility to local minima. Furthermore, as every trajectory is dynamically simulated, undesired oscillations do not occur, unless introduced by a periodically changing environment.

The comparison between the vector field histogram and the global dynamic window approach is representative for most sensor based methods (see Section 3.2). The vector field histogram was chosen as an example, because detailed experimental results have been published.

Much more recently, robotic systems giving tours of museum exhibits were developed. Because they are deployed in environments populated by humans, they have to display sophisticated motion capabilities. The experiences that were made with the robot Rhino (Thrun et al. 1998) lead to the development of Minerva (Burgard et al. 1998), a more advanced robotic museum tour guide. Minerva achieves velocities of up to $0.7\frac{m}{s}$. A map of the environment is provided beforehand. Using the global dynamic window approach faster motions were achieved without providing a map of the environment. It is a key advantage of the global dynamic window approach to be able to navigate fast and reliably in completely unknown environments.

The overall system architecture of Minerva includes many different components, like map building and learning, relocalization, local and global planning, and mission planning. Obviously, this is a much more sophisticated system architecture than the one for the global dynamic window approach, where very simple sensing, mapping, and planning methodologies are integrated into one single algorithm. Nevertheless, according the experimental data found in the literature (Burgard et al. 1998) the global dynamic window approach performs much better (compare Section 5.2). For motion over long distances and continuous motion over an extended period of time, a relocalization algorithm would have to be integrated into the global dynamic window approach to maintain this advantage.

Whereas the motion generation approaches for the museum robots require a map of the environment, dynamic path planning (Zelek 1995) addresses the problem of navigation in unknown environments. Harmonic potentials are used to generate a local minima-free potential function. As the robot moves and perceives the environment, the harmonic potential is updated. The velocities achieved with this method range between 0.05 to $0.10\frac{m}{s}$[1]. The implementation presented in Zelek 1995 suspends robot motion for planning.

5.4 Conclusion

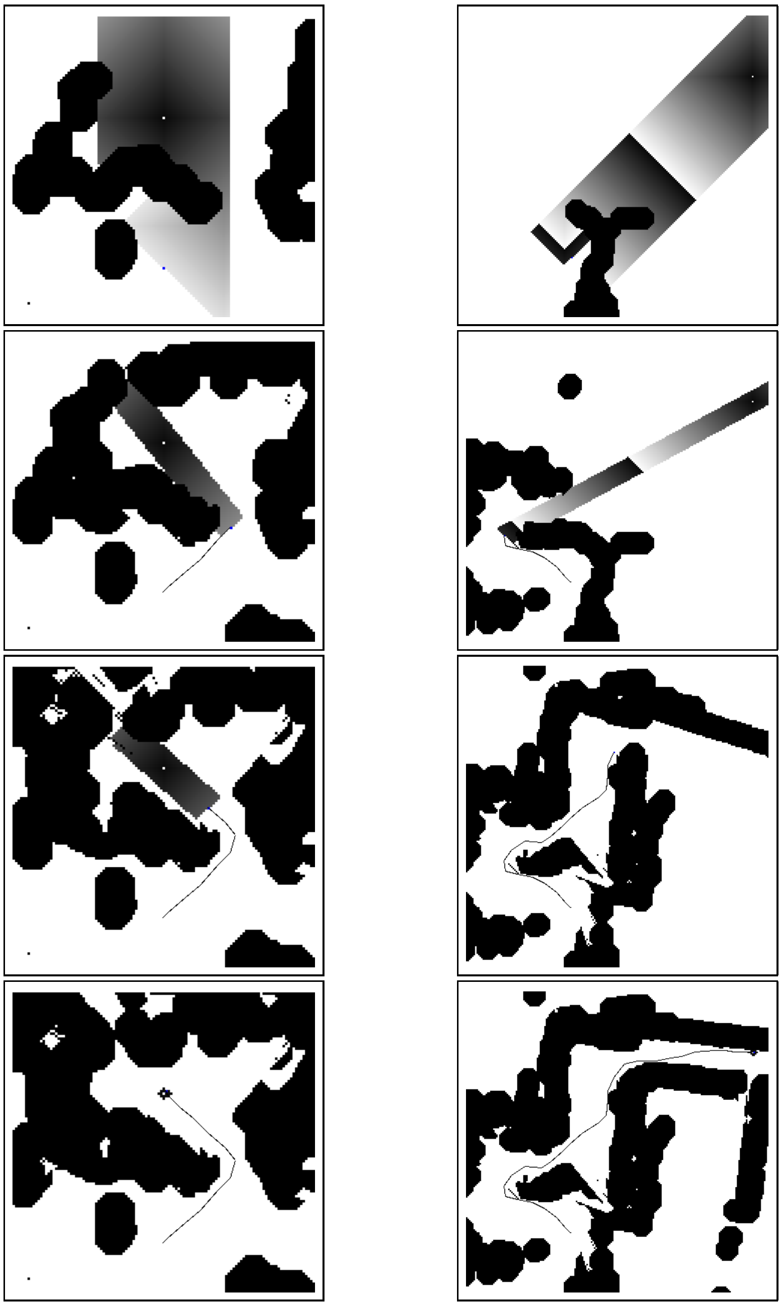
The global dynamic window approach has been shown to be a very robust and efficient framework for navigation of mobile robots at high velocities in unknown and dynamic environments. It enables the robot to perform complex motions without any prior knowledge of the environment by simply specifying a goal position relative to its current position. The robot safely navigates to that position, avoiding unknown stationary or moving obstacles. The experimental results are unmatched in the literature. Using very simple and efficient algorithms, complex behavior is achieved.

In Section 2.4 the integration of planning and execution methods was discussed. Several system architectures were presented and evaluated. The ideal system architecture according to the evaluation is depicted in Figure 2.6 of that section. It is called ideal because every motion command is determined taking into account all the available information about the environment, including global information about the connectivity of the free space. In contrast, planning takes into account all the information available at one point in time to determine all motion commands, and motion execution algorithms take into account local information to determine motion

[1]The velocity is estimated from a digital movie (http://www.cim.mcgill.ca/~zelek/SPOTT-ex1.mpg).

commands.

The effectiveness of the global dynamic window approach is derived from the fact that a motion command is chosen based on a complete motion plan, connecting the current position and the goal position. Effectively, this corresponds to the integration of the planning process into the servo loop. The resulting system architecture for is shown in Figure 5.6. It is identical to the ideal system architecture depicted in Figure 2.6 in Section 2.4. The integration of motion planning and motion execution algorithms results in a framework enabling mobile robots to move robustly in dynamic and unknown environments. The capabilities of existing approaches to robot navigation are extended significantly.



$6m \times 6m$ map $10m \times 10m$ map

Figure 5.2: The left four images show a motion sequence of the robot, initially located in the center of the lower part of the map, moving to a goal configuration, indicated by a white dot. It can be seen how the navigation function avoids the local minimum and is only computed for a small portion of the entire map. The four images to the right show a motion in a more complex environment. In the third image of this sequence the goal is temporarily believed to be obstructed by an obstacle. This is due to erroneous sensory data and the global dynamic window approach then defaults to the holonomic dynamic window approach.

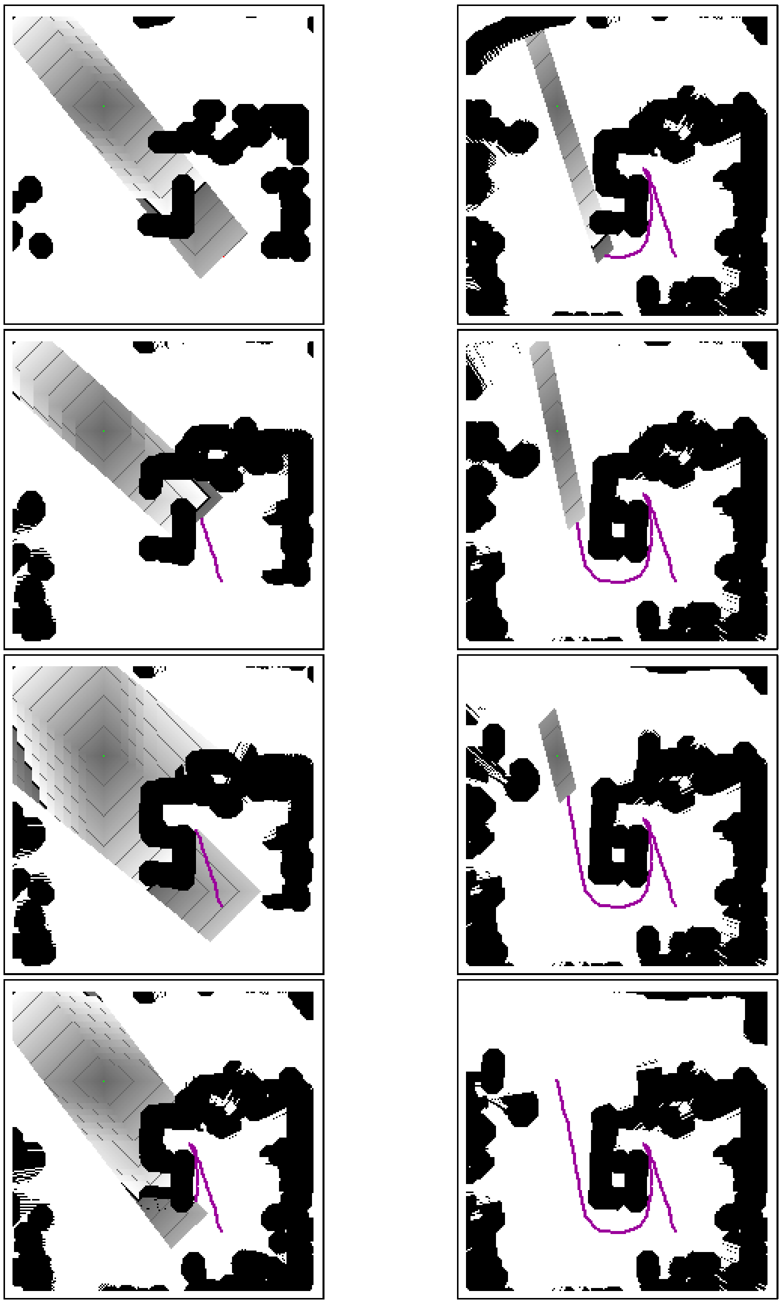


Figure 5.3: This sequence of eight images shows an experiment during which the exploration of the environment forced the robot to backtrack. Initially, the shortest path to the goal leads through an unexplored area. As sensory information indicates that the passage is blocked (third image from the top on the left) the navigation function changes discontinuously. Following the updated navigation function, the robot successfully reaches the goal configuration.

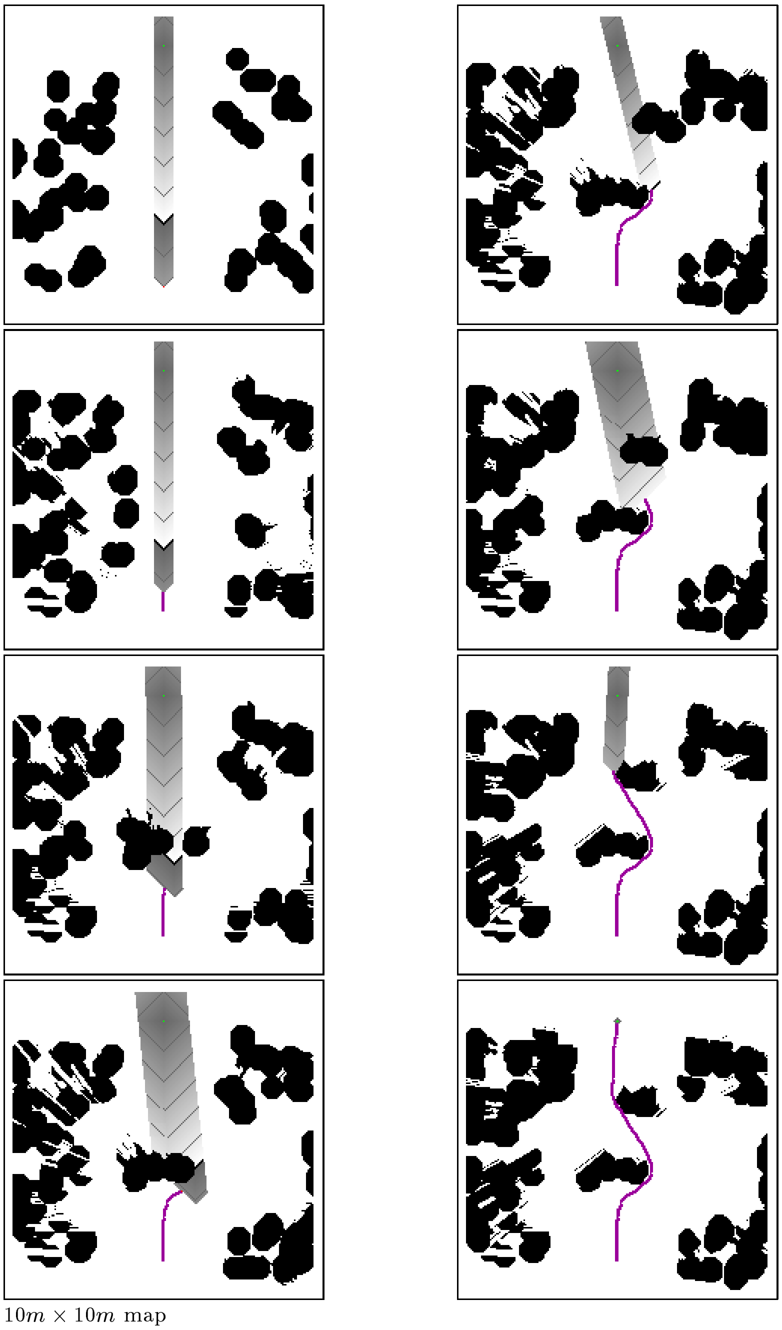


Figure 5.4: This experiment demonstrates the capability of the global dynamic window to navigate a robot in dynamic environments. The first four images indicate how a group of four people, perceived as circular obstacles, are gathering in the center of the map. As the robot circumnavigates them, two more people move into its trajectory in the top right part of the map. The robot evades and successfully reaches the goal.

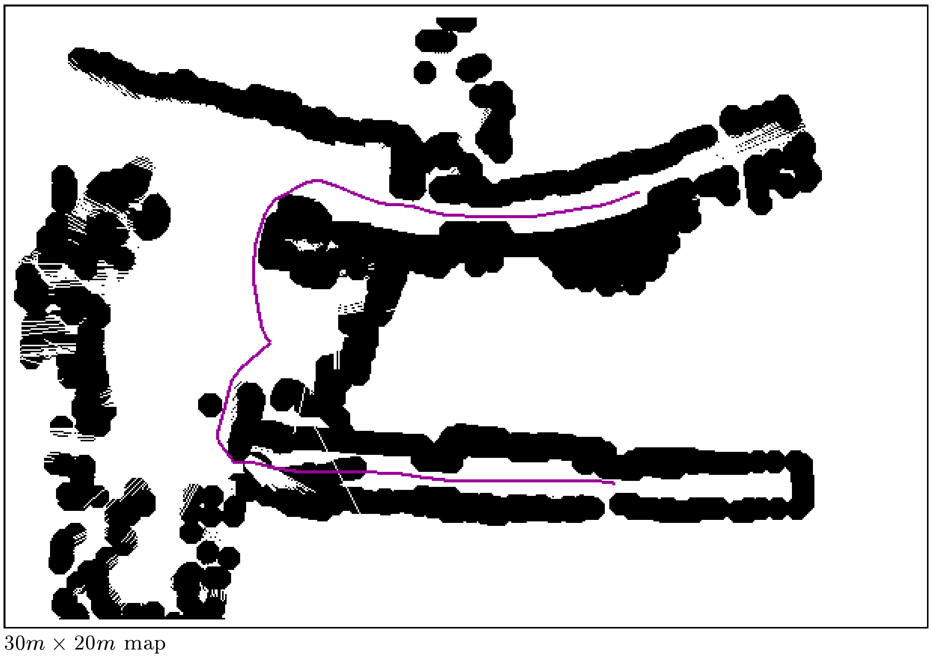


Figure 5.5: This map resulted from a motion over 40 meters in length. The curvature of the hallway in the upper part of the image is due to slippage on carpet.

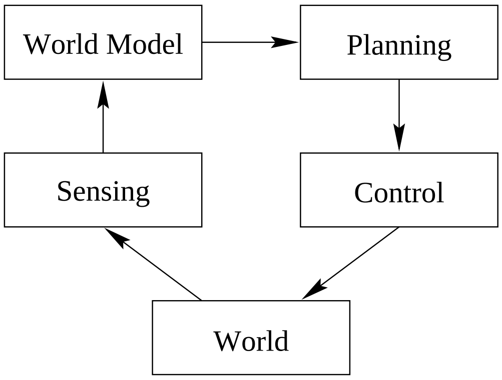


Figure 5.6: The system architecture of the global dynamic window achieves true integration of the planning process into the control loop. Each motion command is the result of a planning operation incorporating all the information about the state of the world available at that time.

Part II

Motion of an Articulated Robot

Chapter 6

Motion Planning and Execution

Motion planning and motion execution algorithms for mobile robots were introduced in Chapter 3. Due to the low dimensionality of the configuration spaces associated with mobile robots, techniques could be applied that either do not extend to higher dimensions or for which the extension results in computational requirements beyond practicality for real-world applications. In this chapter, planning and execution paradigms are presented that also apply to high-dimensional configuration spaces, i.e. robot with many degrees of freedom. The computational cost for those approaches is still dependent on the dimensionality of the configuration space associated with the problem at hand and performance degrades quickly as the number of dimensions increases. For a more detailed discussion of the computational complexity of planning algorithms see Chapter 10.

6.1 Planning Paradigms

Some of the planning paradigms introduced in Chapter 2 and in Section 3.1 of Chapter 3 can be applied to higher dimensions. A special version of the roadmap method, the *silhouette method* (Canny 1988), extends to n-dimensional spaces. Approximate cell decomposition can also be applied to higher-dimensional configuration spaces associated with articulated robots (Lozano-Pérez 1987). All these approaches become computationally intractable as the dimensionality of the configuration space increases.

To address the issue of computational complexity for planning in high-dimensional configuration spaces, simplifying assumptions can be introduced. For example, the dimensionality of the configuration space can be reduced by projecting the robot into a lower-dimensional subspace. This was implemented for multiple robots picking parts from a conveyor belt (Li and Latombe 1997), and more recently in computer graphics, for planning the path of an animated character in real time using rendering hardware functionality for speed (Kuffner 1998).

The potential field approach to motion planning can easily be extended to robots with many degrees of freedom (Faverjon and Tournassoud 1987; Faverjon and Tournassoud 1990). The likelihood of getting stuck in a local minimum and therefore not achieving the goal configuration, however, generally increases in high-dimensional configuration spaces. Random walks have been proposed as a method that can be integrated into the planner in order to escape local minima (Barraquand and Latombe 1990; Barraquand and Latombe 1991).

As mentioned in Section 2.3 of Chapter 2, probabilistic roadmaps differ from other planning approaches in that they represent configuration space obstacles only implicitly. While other approaches compute the boundary of configuration space obstacles from their workspace representation, probabilistic roadmaps compute an approximation from a random sampling of the configuration space. During this sampling process those configurations are rejected that are in collision with the environment. Because this approach has been shown to be an efficient paradigm for motion planning for robots with many degrees of freedom, the general idea will be presented below.

6.1.1 Probabilistic Motion Planning

Most planning methods presented in Section 2.2 in Chapter 2 construct the configuration free space by computing the configuration space obstacle region. This process is computationally very expensive, especially in configuration spaces of high dimensionality. *Probabilistic roadmaps* (Kavraki 1994; Kavraki, Švestka, Latombe, and Overmars 1996; Kavraki, Latombe, Motwani, and Raghavan 1998) approximate the connectivity of the free space using an implicit representation, avoiding the explicit computation of configuration space obstacles. The approximation can be computed at a significantly lower computational expense than the explicit representation. Therefore, this approach is particularly well suited for motion planning in high-dimensional configuration spaces.

Probabilistic motion planning proceeds in two phases: During the *preprocessing phase*, a probabilistic roadmap is generated. During the *query phase*, this roadmap is searched for a path. To this extent, probabilistic motion planning is identical to any other roadmap approach. The preprocessing phase that creates the roadmap, however, proceeds differently.

A probabilistic roadmap is a graph $G = (V, E)$ with a set V of vertices and a set E of edges connecting vertices $v_i, v_j \in V$. This graph is meant to capture the connectivity of the free space. The computation of a roadmap for a particular planning problem begins by determining the set V of vertices. Each vertex corresponds to a configuration of the robot. These configurations are generated by sampling uniformly at random the space of possible configurations. For each configuration q a *collision checker* is invoked to determine if the the robot in configuration q is completely in free space; in this case q is added to V as vertex v, otherwise it is discarded. The number of configurations V should contain to capture the connectivity of the free space accurately with high probability can be determined according to certain properties of the planning problem (Hsu, Latombe, and Motwani 1997).

If an efficient *local path planner* is able to generate a path between configurations q_i and q_j, the edge $e_{i,j}$ connecting vertices v_i and v_j is added to E. This simple local planner might fail to generate a path for more complicated planning problems, even though one exists. Hence, in order to generate mostly simple planning problems, the configurations q_i and q_j are chosen such that they are close to each other, according to some metric. This *network construction* process results in a first approximation of the connectivity of the free space: the randomly generated configurations in free space between which a plan can be easily generated are connected by edges in the probabilistic roadmap G.

The last step of the preprocessing phase attempts to improve the roadmap. Especially in complex or narrow environments the roadmap at this stage might contain more connected components than the actual free space. In order to reduce the number of connected components in G vertices from V are selected that are believed to lie in a difficult region or narrow passage of the free space (Kavraki 1994; Hsu, Kavraki, Latombe, Motwani, and Sorkin 1998). During the *expansion* of those configurations, additional configurations are generated at random in close proximity. Using the same or a slightly more general local planner as in the previous step, the probabilistic roadmap G is expanded with the attempt to merge connected components. By choosing the parameters of this process correctly, it can be guaranteed that with high probability this roadmap represents the actual free space accurately (Hsu, Latombe, and Motwani 1997).

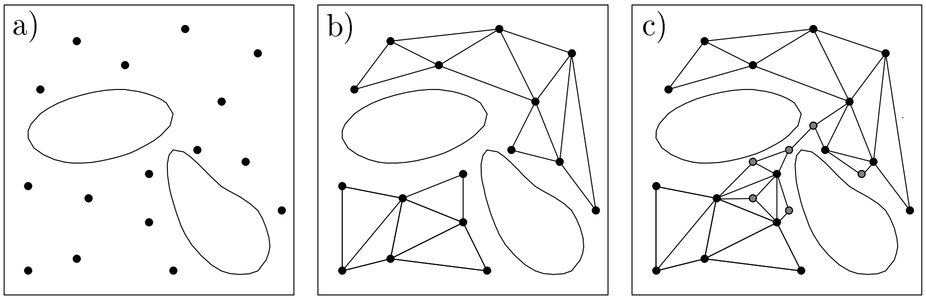


Figure 6.1: Construction of a probabilistic roadmap: a) Randomly generated configurations that are not in collision with the environment constitute the vertices of the roadmap. b) Using a simple and efficient planner vertices are connected to form the roadmap. c) The roadmap is refined in areas where unconnected vertices are spatially close.

The process of constructing a probabilistic roadmap is illustrated in Figure 6.1. Two configuration space obstacles in a two-dimensional configuration space are shown. The configurations resulting form the sampling phase are indicated by points. Configurations are connected by a line if a path can be found between them using the planner appropriate for the stage of the roadmap generation. Initially, random configurations are generated in the configuration space; this can be seen in Figure 6.1a).

According to the planning results of a simple local planner these configurations are connected. The resulting roadmap is shown in Figure 6.1b). The final roadmap is obtained by expansion of those nodes that are assumed to be in a difficult region of the free space. In Figure 6.1c) the nodes resulting form this expansion are shown in gray. Using a local planner to connect the new configurations to the roadmap, the number of connected components is reduced.

After this preprocessing phase, probabilistic motion planning proceeds like other roadmap methods: A plan connecting an initial configuration to a final configuration is generated by planning a path from those configurations to a node of the roadmap using the simple local planner. The roadmap is searched for a sequence of nodes connecting the resulting two configurations. By applying path smoothing techniques to the resulting path a smooth trajectory in configuration space can be obtained.

More recently, a randomized method was developed that takes kinematic and dynamic constraints into account (LaValle and Kuffner 1999). Expanding from the goal and the current configuration of the robot, branches of random incremental paths are generated. Once a branch originating from the goal meets a branch originating from the current configuration such that kinodynamic constraints are not violated, a path has been determined.

6.2 Execution Paradigms

In high-dimensional configuration spaces execution paradigms are mainly focussed on problems of feedback control (Franklin, Powell, and Workman 1998). Commanding an articulated body to move on a given trajectory is a challenging task. Next to the kinematic properties of a robot, the dynamic characteristics of rigid bodies and the coupling between them have to be taken into account to achieve high accuracy in execution. A discussion of this matter is beyond the scope of this thesis. The most important of those algorithms are briefly mentioned in this section.

6.2.1 Joint Space

One approach to executing a given trajectory with an articulated robot is to control each joint angle individually. This ignores dynamic coupling between links. In proportional control a torque proportional to the error is applied to the joint in order to achieve the desired joint configuration. A derivative and an integral term can be added to improve performance. By computing the desired joint torques using a dynamic model of the robot and subsequently applying one of the above control schemes, dynamic disturbances can be rejected. For an introduction to the control of dynamic systems please refer to the text by Franklin, Powell, and Workman. An introduction to those aspects of control that are relevant for the execution of trajectories can be found in Craig 1989. This text also discusses how trajectories can be computed, given a set of via points along the path.

6.2.2 Operational Space

In operational space the robot is not controlled in terms of joint angles, but in terms of end-effector position and orientation. The methods to generate a sequence of joint angles from a trajectory for the end-effector is described in Khatib 1998.

6.3 Linking Planning and Execution

Above we have distinguished between planning and execution methods. This distinction was previously introduced and discussed in Section 2.3. In Section 3.3 methods were presented that attempt to integrate the robot motion planning and execution phase for mobile robots in low-dimensional configuration spaces. Chapter 4 then introduced the global dynamic window approach, a framework in which this integration yields a robust method to generate and execute motion for mobile robots in unknown and dynamic environments. In this section the problem of integrating planning and execution is revisited from the perspective of robotic systems with many degrees of freedom. Many of the simplifying assumptions that helped to solve the problem for mobile robots in Part I of this thesis do not hold in this domain. Therefore, the integration problem becomes much harder, algorithmically and computationally.

6.3.1 Elastic Bands

The framework of *elastic bands* (Quinlan and Khatib 1993a; Quinlan and Khatib 1993b; Quinlan 1994b) will play a major role in Chapter 7, where one contribution of this thesis – the *elastic strip framework* – is presented. As already indicated by the name, there are many similarities between these two frameworks. For the understanding of elastic strips it is helpful to gain some insight into elastic bands. Hence, elastic bands are presented in this section in sufficient detail to familiarize the reader with the basic ideas common to both frameworks.

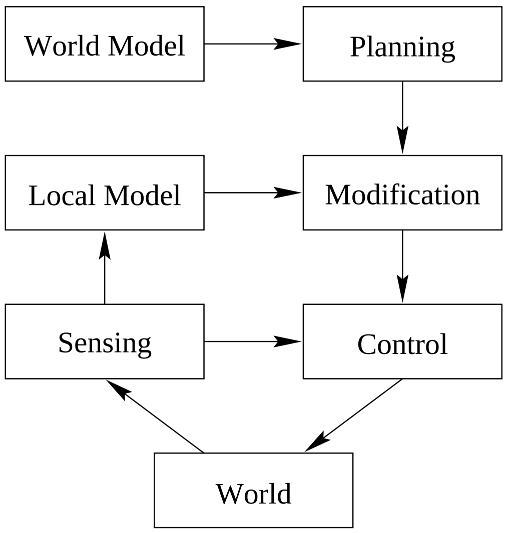


Figure 6.2: Motion planning and execution architecture using elastic bands: An intermediate representational layer is introduced. It represents the local obstacles in the vicinity of the trajectory. The trajectory is modified to avoid collisions.

The elastic band framework integrates robot motion planning with motion execution. The system architecture corresponding to the elastic band approach is shown in Figure 6.2 in a similar fashion as in Figures 2.3 to 2.6 in Section 2.4. This system architecture differs from the one for planning with reactive execution shown in Figure 2.5 on page 24 by the intermediate level, consisting of *Local Model* and *Modification*. The highest level – the planner – generates a path or trajectory. The local environment in the proximity of that path is represented on the intermediate level. This local model changes due to sensed changes in the environment. The path is modified incrementally to remain free of collisions. In the figure, this is indicated by *Modification*. Since the plan maintains its topological properties and continues to represent a path from the current configuration of the robot to the goal configuration, these modifications of the path do not introduce local minima. If the path becomes topologically infeasible, however, an entirely new plan has to be generated. This happens, for example, when a moving obstacle "pushes" the path against another obstacle. Here, only a topologically different plan can yield a solution to the posed planning problem.

Intuitively, the path resulting from the planning phase can be imagined as an elastic rubber band, stretched between the initial and the goal configuration. Its elasticity tends to shorten the rubber band, but obstacles impose constraints by resisting the contraction force of the rubber band. If a moving obstacle pushes against the rubber band representing the path, it will deform. The rubber band will always represent a collision-free path, because it cannot penetrate objects in the environment.

Moving back to robotics, an *elastic band* is a one-dimensional curve in configuration space of finite length, representing a path of a robot between an initial and a final configuration. This curve is modeled as an idealized elastic band. Internal forces shorten and straighten the elastic band; they correspond to the elasticity of a rubber band. External forces caused by obstacles prevent the elastic band from penetrating or touching obstacles and keep it at a safe distance, determined by the magnitude of the repulsive force. The incremental modification of a path for a point robot in a two-dimensional configuration space is illustrated in Figure 6.3. Note that in this simple example the dimensionality of the configuration space is identical to the one of the workspace. It can be seen how the moving obstacle, shown in gray, deforms the elastic band, indicated by the continuous line. The elastic band continues to represent a valid path between the start and the goal configuration.

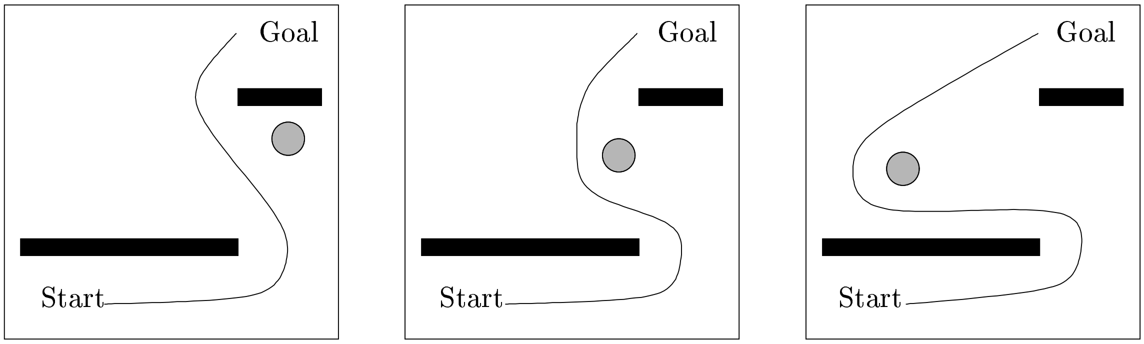


Figure 6.3: Illustration of elastic band framework: The circular obstacle moves from the right to the left, deforming the existing trajectory accordingly. Note that the topological properties of the trajectory do not change.

The elastic band system architecture, shown in Figure 6.2, closes a control loop on the intermediate level between a purely reactive system and the ideal system architecture that integrates planning into the loop (compare Figures 2.3 and 2.6 in Section 2.4). On this intermediate level most local minima can be avoided, as the elastic band represents a valid path to the goal as long as possible. However, the path can still be invalidated by a topological change in the environment, caused, for example, by an obstacle moving very close to another one. This situation could arise in Figure 6.3 if the moving obstacle would continue to move to the left, until the path for the robot is cut off between the wall and the obstacle.

In addition to invalidation of a path, the elastic band approach suffers from the problem of potentially producing suboptimal trajectories. The last part of Figure 6.3 illustrates this shortcoming of the elastic band approach: the moving obstacle continues to deform the elastic band, although a much shorter path would result from choosing to pass the obstacle on the other side. Both of the problems mentioned above can be solved by integrating replanning into the elastic band framework. In the elastic band approach, however,the local model is not used to update the world model in order to generate a new plan (see Figure 6.2).

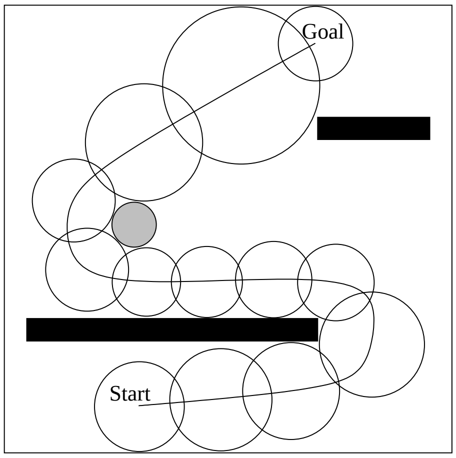


Figure 6.4: Bubbles covering a path: The trajectory from the previous examples is covered by overlapping bubbles of free space. As long as the trajectory is contained within the union of those bubbles, it is guaranteed to be in free space.

Another more fundamental shortcoming of the elastic band approach results form its free space computation and the resulting computational complexity.

The control loop of a motion generation and execution approach must be executed often enough to guarantee stable and smooth motion (Franklin, Powell, and Workman 1998). Due to the fact that the elastic band framework integrates the path modification into the control loop, this modification procedure must operate very efficiently. In the elastic band framework, the efficiency depends to a great extent on the representation of free space around the trajectory. This free space is captured by an overlapping set of convex regions of free space, centered on the trajectory. These regions are called *bubbles* (Quinlan and Khatib 1993a). In Figure 6.4 the free space around a path of a point robot moving in two dimensions is covered with bubbles. In this particular case, bubbles take the shape of circles. The bubbles overlap in such manner that their union completely contains the path. For a point robot moving in n dimensions, a bubble captures a spherical region of free space around a given point $\mathbf{p}$ and can be computed with just one distance computation. Let $d(\mathbf{p})$ be the function that computes the minimum distance from a point $\mathbf{p}$ to any obstacle. The *configuration space bubble* $\mathcal{B}(\mathbf{p})$ of free space around $\mathbf{p}$ for a point robot is defined as the set of points $\mathbf{q}$ that are closer to $\mathbf{p}$ than the closest obstacle:

$$\mathcal{B}(\mathbf{p}) = \{\, \mathbf{q} : \|\mathbf{p} - \mathbf{q}\| < d(\mathbf{p})\}.$$

When the elastic strip is modified by internal and external forces, the size of a bubble indicates the magnitude of the external force and also defines by how much the elastic band can be deformed at this location while still remaining free of collisions.

The computation of bubbles for articulated robots is slightly more involved. As this computation procedure inflicts severe limitations on the elastic band approach, it is presented here. These limitations will provide some of the motivation for devising the *elastic strip* approach, presented in Chapter 7.

Consider an articulated robot with n revolute joints. A configuration of this robot is represented by a vector $\mathbf{q} = \{q_1, \cdots, q_n\}$ of n joint variables q_i. Let d be the minimum distance between any point on the robot and the environment. For each joint axis the axis-aligned cylinder of minimum radius is determined such that all subsequent links are contained within that cylinder. The radii of the cylinders are designated by r_i. We can use those radii to limit the distance any point on the robot travels when moving from configuration $\mathbf{q}$ to configuration $\mathbf{q}'$. If we imagine each joint i to move in turn from q_i to q_i', each joint motion can cause a maximum displacement bounded by $r_i|q_i - q_i'|$. Hence, after all joints moved, no point on the robot can have moved more than

$$d_{motion} = \sum_{i=1}^{n} r_i \, |q_i - q_i'|.$$

This expression represents a weighted distance in the L^1 norm between $\mathbf{q}$ and $\mathbf{q}'$ with the radii r_i as the weights. A bubble around a configuration $\mathbf{q}$ is now defined as:

$$\mathcal{B}_{\mathbf{q}} = \{\mathbf{q}' \; : \; \sum_{i=1}^{n} r_i \, |q_i - q_i'| < d\}.$$

The computation of bubbles can be extended to incorporate joint limits and self collision (Quinlan 1994b).

Figure 6.5 illustrates the bubble computation using a planar manipulator with two degrees of freedom. The bounding cylinders become circles. If joint one is moved by an angle α, no point on the robot can move further than $\alpha\ r_1$. Consecutively, joint two is moved by an angle β. Now the total distance traveled by any point on the robot is bounded by $d_{motion} = \alpha\, r_1 + \beta\, r_2$. As long as $d_{motion} < d$ it is guaranteed that the motion does not lead to a collision with the obstacle. The configuration space bubble that results from this computation in two dimensions has the shape of a diamond, or more precisely a parallelogram. The ratio of width to height of the bubble is equal to the ratio of $\frac{1}{r_1}$ to $\frac{1}{r_2}$.

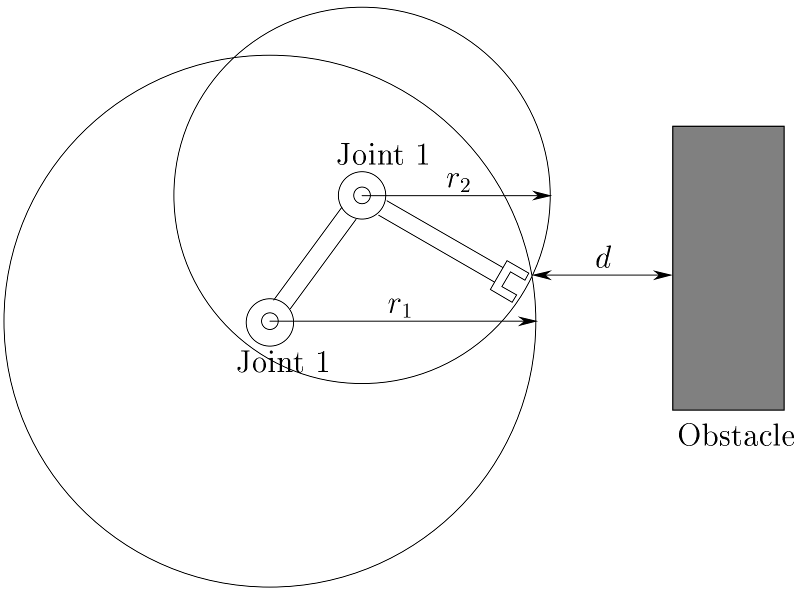


Figure 6.5: Computing a bubble for an articulated manipulator: The shortest distance d to any obstacle determines the maximum motion of every point on the robot. The size of the free space around a configuration is estimated by bounding the motion of a link using cylinders centered at the joint and containing the entire link.

The procedure of bubble computation becomes increasingly conservative as the number of degrees of freedom of the robot and the link length increases. This is due to the fact that the entire link length is used to bound the amount a point on the robot can move between two configurations. The direction of motion is constrained by the joint and the effect of the motion of subsequent joints might cancel out. This is ignored during free space computation, resulting in a significant underestimation of free space around a particular configuration. In Chapter 7 this limitation will be discussed in more detail.

The elastic band framework has been applied successfully to robots such as the six degree of freedom PUMA 560 (see Appendix A).

6.3.2 Other Approaches

In another approach to integrated planning and control a plan is converted into a trajectory using a path-based parameterization, rather than a time-based one (Tarn 1996). This allows the corresponding path-based controller to interrupt the execution of the trajectory, if an unforeseen obstacle is detected. Once the obstacle has been removed or an evasion maneuver has been specified by a human operator, the execution of the original trajectory is resumed. This method could potentially suspend execution forever, if the obstacle is never removed.

Chapter 7

Elastic Strips

This thesis proposes the integration of motion planning and motion execution algorithms as an approach to robust robot motion in dynamic environments. In Chapter 4, the *global dynamic window approach* was presented, achieving this goal for the motion of a mobile base. This chapter introduces the *elastic strip framework* with the intent of integrating planning and execution for robots with many degrees of freedom.

The elastic band approach (Quinlan 1994b) presented in Section 6.3.1 already accomplishes this integration to some extent. The following section discusses the shortcomings of the elastic band approach and motivates the development of the *elastic strip approach*. The remainder of this chapter then presents this new framework and its extensions.

7.1 Motivation

Before we set out to describe the main contribution of this thesis, the *elastic strip framework*, it might be helpful to lay out some guidelines according to which we can evaluate whether we succeed in solving the problem of integrating planning and execution or not. A solution should fulfill the following requirements:

Efficient for many degrees of freedom: The kind of task one would like to address with such a framework lies beyond pure automation of assembly in manufacturing. In those domains an integration is not necessary because the environment can be designed to suit the capabilities of the robot and dynamic changes of the world can be controlled very well. These assumptions, however, do not hold in all domains. In *personal robotics*, where robots assist humans in their everyday life, or *service/field robotics*, where robots and humans work together to accomplish a task in an unstructured environment, the environment changes constantly and unpredictably. To execute task successfully in those domains, robots need to have very versatile, dextrous capabilities. Therefore, redundant manipulators with many degrees of freedom are used commonly.

Since most planning algorithms quickly become computationally intractable as the number of degrees of freedom of the robot increases, one important requirement for a proposed solution is *scalable complexity*: there must be a reasonable, smooth degradation in performance as the number of degrees of freedom increases, the world model becomes more complex, or paths increase in length.

Intuitive task specification: Whereas in manufacturing environments a motion, once programmed, is executed over and over again, this motion generation paradigm fails when the task differs for each motion. For each motion a new motion plan needs to be generated and even for one given task multiple such motions need to be determined if changes in the environment have invalidated a previously determined motion. To facilitate planning in such a situation, an intuitive task specification becomes very important, especially if the motion should be generated with human intervention. Such a task representation is also applicable to tele-robotics with latencies, where only high-level commands can be transmitted to be executed autonomously.

Powerful system architecture: In Section 2.4 an ideal system architecture for motion generation was described (see Figure 2.6). It is designed for robust motion execution in dynamic environments with unpredictably moving obstacles. The global dynamic window approach introduced in Chapter 4 realized this architecture for mobile bases. In designing a framework that integrates motion planning and execution, we will attempt to implement this architecture as closely as possible.

The elastic *strip* framework presented below is based on the elastic *band* approach (Quinlan 1994b). To motivate the development of this novel framework, we evaluate the elastic band approach with respect to above guidelines.

Efficient for many degrees of freedom: The computation of free space is the computationally most expensive part of most global motion generation algorithms, such as the various planning approaches (see Sections 3 and 6), in particular when the free space is represented in configuration space and the number of degrees of freedom of the robot is large. The computational complexity of those algorithms grows exponentially in the number of dimensions of the configuration space (see also Section 10).

The free space representation used in the elastic band approach consists of configuration space bubbles (see Section 6.3.1). This choice of representation causes a significant performance degradation when the elastic band framework is applied to a robot with many degrees of freedom. The problems resulting from this representation are discussed in detail in Section 7.2.

Intuitive task specification: The trajectory executed by a robot is not necessarily always planned by a motion planner. As human interaction and intervention in planning and execution becomes more relevant in various domains of robotics, an interface is needed that is intuitive to humans.

In the elastic band framework a trajectory is specified and represented in configuration space. In contrast, most tasks are specified in workspace as they interact with and manipulate objects in workspace. The straight line motion in configuration space for an end-effector mounted on an arm with revolute joints, for example, results in a curved trajectory in workspace.

Powerful system architecture: The ideal system architecture presented in Chapter 2, illustrated schematically in Figure 2.6 on page 24, contains a control loop determining a new plan for each issued motion command. The architecture for elastic bands, shown in Figure 6.2 on page 77, closes the loop around the newly introduced intermediate level of representation of local free space around the trajectory. The motion plan is determined once and then modified incrementally. This can result in highly suboptimal trajectories, if a door opens, for example, and a shortcut between the current position of the robot and its goal becomes a feasible trajectory.

In devising the *elastic strip framework* presented in the remainder of this chapter, the attempt was made to eliminate the problems arising in the elastic band approach. In Section 9.4 these requirements will be the testbed for the novel framework.

7.2 Choice of Free Space Representation

In Section 7.1 scalable complexity is mentioned as a desirable property for a motion generation framework. It allows a framework to be applied to increasingly complex problems without experiencing a drastic loss of performance. As the computational complexity of global motion generation methods, i.e. motion planning, is mainly determined by the computation of the representation of free space, the choices pertaining to this aspect of motion generation will play a crucial role in the performance of a framework. In this section the classical choice of a configuration space representation will be revisited, taking into account the particular requirements of the elastic strip framework.

Traditionally, planning methods perform the computation and representation of free space in the configuration space of the robot (Latombe 1991). This allows the robot to be represented as a point and a path as a one-dimensional curve in that space, significantly simplifying the problem of generating and representing a path, once the free space has been computed. For robots with many degrees of freedom, however, the computational requirements for this procedure grow exponentially (Canny 1988).

Therefore, the traditional configuration space representation for free space does not seem appropriate in the context of the elastic strip framework, where all computations have to be performed in real time. In order to achieve the goal of real-time performance, one can exploit the fact that only the free space around a previously generated trajectory needs to be represented. This is exactly the approach taken in the elastic band framework (Quinlan 1994b).

Despite the ability to restrict the free space computation to the vicinity of the trajectory, the computational complexity of the elastic band framework degrades drastically with an increasing number of degrees of freedom of the robot. The reason lies in the method of computing regions of free space, called configuration space bubbles. This procedure was presented in Section 6.3.1. One distance computation results in the closest distance d between any point on the robot and any of the obstacles. The size of the bubble has to be determined such that for every configuration inside the bubble no point on the robot moves more than d. This method of free space computation becomes more and more conservative as the length of links and the number of consecutive links increases.

An illustration of the effect of the conservativeness of this procedure can be seen in Figure 7.1. Two images on the top and two images on the bottom of the figure illustrate the bubble for two different configurations of a planar two-link manipulator arm. The workspace and the configuration of the robot arm are illustrated in Figure 7.1 a) and c). The closest distance between the robot and the environment is indicated by a line. The images b) and d) show the configurations space obstacles for those situations in gray. The coordinate axes of the configuration space are the two joint variables of the manipulator arm q_1 and q_2. The diamond shape represents the configuration space bubble of free space. Figure 7.1 a) and b) show a situation in which the configuration space bubble closely approximates the local free space: it touches the configuration space obstacle region. A better result cannot be expected using only one distance computation. Images c) and d), on the other hand, show how the bubble computation procedure fails to capture the free space accurately. Despite the fact that both joints of the arm can move by a lot before hitting an obstacle, as can be seen in Figure 7.1c), the configuration space bubble is very small and does not

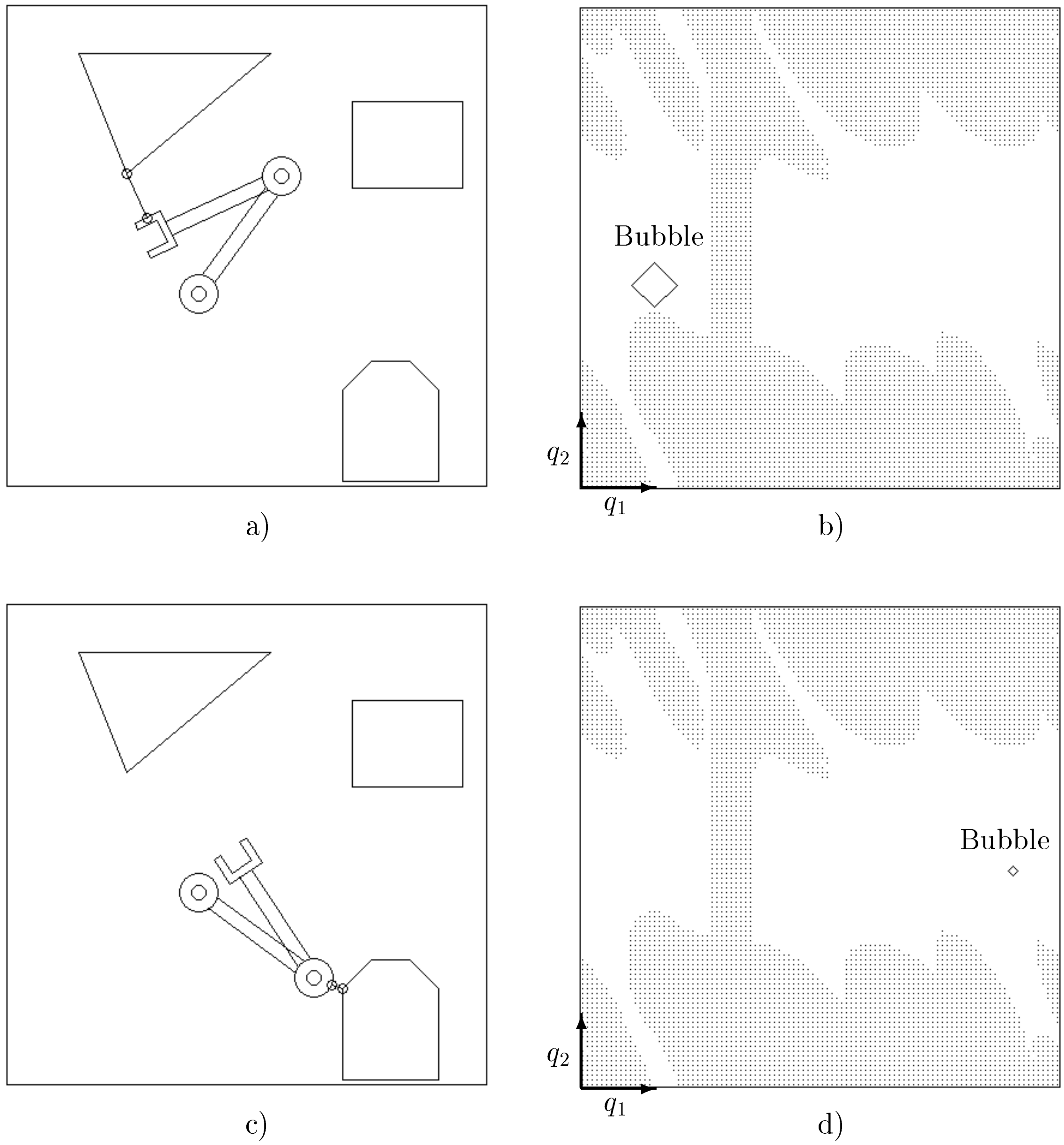


Figure 7.1: Two examples of configuration space bubbles: The bubble approximates the free space around a configuration a) fairly accurately, as it touches the configuration space obstacle in b). In c) and d) a different configuration is shown for which the free space is grossly underestimated.

touch the configurations space obstacle region. This is due to the fact that only one distance computation is used to represent the free space around the robot and that this one distance measurement does not take into account the kinematic constraints of the robot. In many situations configuration space bubbles grossly underestimate the local free space around a configuration of the robot. As a result, the number of bubbles required to cover the trajectory and therefore also the number of intermediate configurations used to represent it, are excessively high. This leads to the inefficiency of the approach, in particular for robots with many degrees of freedom, long links, or both.

Rather than using only one distance measurement to compute a region of configuration free space, many such computations could be used. Iterative or recursive methods can be used to get a more accurate approximation of free space around a certain configuration (Canny and Lin 1993); these methods numerically explore the bounds of a local area of free space. Obviously, the amount of computation will increase with the number of degrees of freedom of the robot because the extent of the free space region will have to explored in every dimension of the configuration space.

Comparing such a recursive method with the elastic band approach, the following observation can be made: There is an obvious trade-off between the complexity of the computations required to compute the free space and the accuracy of its representation. In the elastic band approach only one single distance computation is used to get a very coarse approximation of the free space. This approximation is overly conservative to the extend that it impacts the performance of the algorithm. Recursive methods, on the other hand, are on the other end of the trade-off. They invest significant computation in order to compute an accurate model of the free space. The computational complexity depends on the number of degrees of freedom of the robot and significant performance degradation will occur in high-dimensional configuration spaces.

The shortcoming of both free space computation methods presented above results in both cases from the fact that the kinematics of the manipulator are influencing the free space computation. In the elastic band approach the size of a bubble needs to be conservative enough to bound every point on the entire robot, avoiding the computation for different configurations of the robot by assuming the worst case. It is this worst case assumption that leads to performance degradation. Recursive methods compute the free space by exploring the high-dimensional space of configurations, which also leads to performance degradation for an increasing number of degrees of freedom.

To overcome these problems, in the elastic strip framework takes a different approach: Instead of using joint configurations to describe the free space, the free space is represented independently of the robot, simply by representing its workspace volume. Since we are only concerned with the free space in the vicinity of the predetermined trajectory, this can be imagined as a tunnel of free space containing the volume swept by the robot along a subset of all paths homotopic to the trajectory. Obviously, those homotopic paths are only represented implicitly. To compute them explicitly, one of the recursive methods mentioned above has to be used. But it is exactly this step of determining a representation for the path, describing joint variables as a function of time, that is computationally expensive. This poses the question: How can a particular path be selected from this tunnel of free space that implicitly represents a set of homotopic paths?

The basic idea behind the elastic strip framework is to start out with one path determined by a planner. This path is known in configuration space. The tunnel of free space surrounding it implicitly represents the workspace volume swept by the robot along a set of homotopic paths. All these paths represent alternative routes, in case the current path should be invalidated by a change in the environment. The selection of one of them is performed by incremental modification of the existing path. Repulsive forces exerted by obstacles guide this incremental modification and therefore also the selection of a particular path that is homotopic to the original one. In other words, rather than enumerating a large number of potential configurations in order to determine the free space during the planning process, as is done in the recursive methods, these configurations are enumerated only as needed during the execution. The search of all possible configurations is guided by the meaningful heuristic of proximity to obstacles in the form of repulsive forces. This enables the elastic strip framework to select a valid path from the set of homotopic paths without exploring the space of all configurations.

This concept of locally guided selection of particular configurations can also be used to simplify the planning task itself. When planning for a highly redundant robot it might not make sense to specify all degrees of freedom prior to execution. Using the elastic strip framework a partial plan for a subset of the degrees of freedom of the robot could be complemented automatically by the scheme described above. In a similar fashion a plan could be specified by simply specifying a task, such as the desired trajectory of a manipulated object in space. In conjunction with the current configuration of the robot, this defines a set of homotopic paths to the goal configuration. One of this paths is selected during execution by using external forces to guide the search of all homotopic paths.

There are many ways of representing the free space around a particular trajectory directly in the workspace. The representation chosen for the implementation of the elastic strip framework presented in this thesis is described in Chapter 8. For the understanding of the remainder of this chapter, however, the basic notions are introduced here.

The local free space around a given configuration is represented by a *protective hull*. It can be imagined as a hull of free space surrounding the robot and protecting it from collisions. A protective hull simply is a volume description of workspace volume containing the robot but no obstacles. An *elastic tunnel* is a set of overlapping protective hulls placed along a given trajectory such that at any point during the execution of the trajectory the robot is completely contained within the volume represented by the union of all protective hulls. Because this tunnel can deform in reaction to changes in the environment, it is called *elastic*. For illustrations of these concepts please refer to Chapter 8.

7.3 Discrete Model

Modeling the properties of elastic material is a well-studied problem in mechanics (Wempner 1973; Borg 1990). In mechanics the goal is to find a good model of the true physical laws governing elasticity. Other engineering disciplines have developed approximations to these mechanical models to either describe systems with similar behavior or to be able to simulate some aspect of elasticity without the requirement of complete physical accuracy.

The latter is the case in computer graphics, where only the visual effects of elasticity have to be duplicated. A comprehensive survey of deformable modeling approaches in computer graphics can be found in Gibson and Mirtich 1997. In computer vision, one-dimensional elastic curves were used to track image contours (Kass, Witkin, and Terzonpoulos 1988). In motion planning the model of an elastic curve was used to plan a path for a robot (Warren 1989; Warren 1990). Using a physically-based approximation, motion planning algorithms have been devised for thin sheets of elastic material (Holleman, Kavraki, and Warren 1998; Kavraki, Lamiraux, and Holleman 1998). And finally, as already mentioned in Section 6.3.1, a one-dimensional elastic curve has been applied to the modification of robot trajectories represented in configuration space (Quinlan 1994b).

This section introduces the model of elasticity used in the elastic strip framework. It roughly approximates the properties of elastic material. Since a simulation of the continuous properties of such material is computationally very complex, we develop a discrete model that approximates some aspects of elasticity while integrating characteristics needed for the application to motion generation for robots.

The general idea of the elastic strip is to represent the volume swept by the robot along its trajectory as elastic material. This volume can be deformed according to the laws of elasticity (Wempner 1973; Borg 1990). If the elastic material is imagined to be attached to the initial and the final configuration of the robot along the trajectory, it is obvious how the elastic representation of the trajectory can be deformed by forces exerted on it.

The correct physical simulation of a three-dimensional elastic body is computationally very complex. In devising the elastic strip framework, however, we are not interested in modeling precise physical behavior, but rather in applying some properties of elastic bodies to the generation of robot motion. We will take the liberty of ignoring constraints imposed in the real world to gain a computational advantage.

A significant reduction in complexity can be achieved by reducing the three-dimensional volume to a two-dimensional strip, hence the name *elastic strip*. The volume is then represented by associating a width with every point on the strip. The properties of thin two-dimensional elastic objects are computationally much simpler (Wempner 1973). Furthermore, while we want the trajectory to be able to "stretch" when it is deformed, the robot consists of rigid bodies that cannot change their physical dimensions. Therefore, the elastic strip will only exhibit properties of elastic material along one dimension, namely along the trajectory.

Elastic material also has an undesirable property for the application at hand. Imagine a rubber band spanned between to points. Pulling on the rubber band will deform it. This corresponds to the situation of an obstacle deforming a trajectory represented by an elastic band (see Section 6.3.1). As the rubber band gets elongated, it will resist further deformation with increasing force, i.e. an obstacle has to "push harder" to deform the elastic. As this is not desirable, the model of elasticity for the elastic strip will "add" material to the strip in an amount proportional to the increase in length.

7.3.1 Internal Forces

Whereas elastic material is homogeneous and its principal physical properties do not vary over its volume, this is not true for an elastic strip. In this framework an elastic strip can be seen as a two-dimensional grid of links and springs. Figure 7.2 illustrates that for the elastic strip $\mathcal{S} = (q_1, q_2, q_3)$ for an arm mounted on a mobile base. The virtual springs approximate elastic behavior along the trajectory. Since the forces resulting from elongation of those springs are directly related to the elasticity of the elastic strip, we call those forces *internal forces*. The virtual springs are attached to control points on subsequent configurations along the trajectory. Let p_j^i be the position vector of the control point attached to the jth joint of the robot in configuration q_i. The internal contraction force $F_{i,j}^{int}$ caused by the springs attached to joint j is defined as

$$F_{i,j}^{int} = k_c \left(\frac{d_j^{i-1}}{d_j^{i-1} + d_j^i} (p_j^{i+1} - p_j^{i-1}) - (p_j^i - p_j^{i-1}) \right),$$

where d_j^i is the distance $\|p_j^i - p_j^{i+1}\|$ in the initial, unmodified trajectory and k_c is a constant determining the contraction gain of the elastic strip.

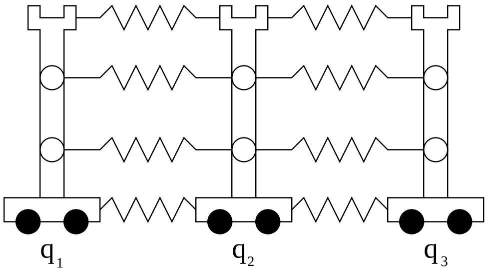


Figure 7.2: The internal forces in the elastic strip are caused by virtual springs attached to control points on consecutive configurations of the robot along the elastic strip.

These forces cause the elastic strip to contract. Effectively, the scaling factor

$$\frac{d_j^{i-1}}{d_j^{i-1} + d_j^i},$$

assures that the relative distance between adjacent configurations is maintained, as the strip is deformed. This avoids the migration of configurations along the elastic strip due to external forces (see Section 7.3.2). Without this scaling factor, those components of external forces along the elastic strip would cause the intermediate configurations along the elastic strip to move away form the source of the external force.

The magnitude of the force acting on the control points is proportional to the local curvature of the elastic strip and to the distance between two adjacent configurations. The dependency on the curvature causes trajectories with abrupt velocity changes to straighten faster than those with small velocity changes. The distance between adjacent configurations is an indication of the available free space around the trajectory. It is desirable to exert a larger force on a control point if the adjacent configurations are far away, as a large motion is likely to remain in free space. The scaling factor achieves this behavior.

The internal forces as defined above also have a positive side effect: When external forces originating form different obstacles are acting on control points, oscillatory behavior can arise. Internal forces effectively act as a damper to eliminate such behavior. For experimental results see Section 9.2.

7.3.2 External Forces

While internal forces provide a rough approximation of the behavior of elastic material, external forces ensure collision-avoidance. External forces are caused by a repulsive potential associated with the obstacles. Various potential functions have been suggested in the literature (Latombe 1991). A potential function used to repel the robot form obstacles should have a limited influence region; this prevents distant obstacles from influencing the robot's motion. Furthermore, as the robot approaches the obstacle the repulsive force should increase towards infinity; this will guarantee collision avoidance. A potential function that has those properties was introduced in Khatib 1986:

$$V_{ext}(p) = \begin{cases} \frac{1}{2}k_r(\frac{1}{d(p)} - \frac{1}{d_0})^2 & \text{if } d(p) < d_0 \\ 0 & \text{otherwise} \end{cases},$$

where $d(p)$ is the distance from p to the closest obstacle, d_0 defines the region of influence around obstacles, and k_r is the repulsion gain.

The nonlinear dependency of repulsive force and distance to the obstacle can cause problems if the modification of the position of the robot in reaction to those forces is not performed carefully. To avoid these problems, a simpler potential function is used in the elastic strip framework. It results in a linear relationship of forces and distances. For a point p it is defined as

$$V_{ext}(p) = \begin{cases} \frac{1}{2}k_r(d_0 - d(p))^2 & \text{if } d(p) < d_0 \\ 0 & \text{otherwise} \end{cases},$$

where $d(p)$, p, d_0, and k_r are defined as above.

The external force F_p^{ext} acting at point p is defined by the gradient of the potential function at that point:

$$F_p^{ext} = -\nabla V_{ext} = k_r(d_0 - d(p))\frac{d}{\|d\|},$$

where d is the vector between p and the closest point on the obstacle. No external forces are applied to the initial configuration q_1 and the final configuration q_n of the elastic strip; these configurations are specified by the user and represent the global task. The magnitude and direction of the external forces acting on a rigid body can be inferred from the distance computation required to compute the bubbles covering that body.

7.3.3 Modification

Let $\mathcal{S} = (q_1, q_2, q_3, \cdots, q_n)$ be an elastic strip. When $\mathcal{S}$ is subjected to the forces described in Sections 7.3.1 and 7.3.2, it is deformed by altering each of the configurations q_i in turn. To change a configuration according to the internal and external forces, these forces have to be mapped to joint torques.

For collision avoidance in the absence of a task requirement, we use the Jacobian J_p associated with the point p at which the force F_p is acting for this mapping. The joint torques Γ caused by F_p are given by

$$\Gamma = J_p^T F_p. \tag{7.1}$$

Joint limits can be avoided using a potential field function (Khatib 1986).

The dynamic model of the system is used to compute the joint displacements caused by the joint torques. The displacements for a configuration q_i define the new configuration q'_i, resulting in the modified elastic strip $\mathcal{S}' = (q_1, \cdots, q'_i, \cdots, q_n)$. $\mathcal{S}'$ represents a valid trajectory, only if the protective hulls $\mathcal{P}_{i-1}$, $\mathcal{P}'_i$, and $\mathcal{P}_{i+1}$ are *connected*. Two protective hulls are connected, if the robot can pass from one to the other without leaving the volume described by their union. This is verified using a procedure described in Section 8.5.

If $\mathcal{P}'_i$ and $\mathcal{P}_{i+1}$ are not connected, the elastic strip $\mathcal{S}'$ becomes invalid. This means that the trajectory represented by $\mathcal{S}'$ cannot be proven to be collision-free, using the representation of local free space associated with $\mathcal{S}'$. In order to reconnect $\mathcal{P}'_i$ and $\mathcal{P}_{i+1}$ intermediate protective hulls are inserted into the elastic strip. By imposing constraints on the transition from q_i to q'_i this procedure can be guaranteed to succeed.

As obstacles recede from the vicinity of the elastic strip, the protective hulls of configurations increase in volume and potentially move closer together. This can result in protective hulls $\mathcal{P}_{i-1}$ and $\mathcal{P}_{i+1}$ to be connected. In that case q_i is redundant and can be removed from $\mathcal{S}$.

The free space computation and elastic strip modification procedures can easily be parallelized to speed up computation. Parallelization relies on the concept of *critical regions* of the elastic strip, introduced below in Section 7.7.3. The parallelization scheme is discussed in Section 7.8.

7.4 Motion Behavior

The preferred change in posture of a robot in reaction to changes in the environment is mostly dependent on the task. This section introduces motion behavior integrating obstacle avoidance and task execution. The integration takes advantage of the motion capabilities of manipulators that are redundant with respect to the task.

7.4.1 Motion Control

The operational space equations of motion of a manipulator are (Khatib 1987)

$$\Lambda(x)\ddot{x} + \mu(x, \dot{x}) + p(x) = F, \tag{7.2}$$

where x is the vector of the m operational coordinates describing the position and orientation of the effector, and $\Lambda(x)$ is the $m \times m$ kinetic energy matrix associated with the operational space. $\mu(x, \dot{x})$, $p(x)$, and F are respectively the centrifugal and Coriolis force vector, gravity force vector, and generalized force vector acting in operational space.

These equations of motion describe the dynamic response of a manipulator to the application of an operational force F at the end effector. For non-redundant manipulators, the relationship between operational forces F and joint forces Γ is

$$\Gamma = J^T(q)F, \tag{7.3}$$

where $J(q)$ is the Jacobian matrix.

For redundant systems it can be shown that the relationship between joint torques and operational forces is

$$\Gamma = J^T(q)F + \left[I - J^T(q)\overline{J}^T(q)\right]\Gamma_0, \tag{7.4}$$

with

$$\overline{J}(q) = A^{-1}(q)J^T(q)\Lambda(q), \tag{7.5}$$

where $\overline{J}(q)$ is the dynamically consistent generalized inverse. This relationship provides a decomposition of joint forces into two dynamically decoupled control vectors: joint forces corresponding to forces acting at the end effector ($J^T F$) and joint forces that only affect internal motions, $\left([I - J^T(q)\overline{J}^T(q)]\mathbf{\Gamma}_0\right)$.

Using this decomposition, the end effector can be controlled by operational forces, whereas internal motions can be independently controlled by joint forces that are guaranteed not to alter the end effector's dynamic behavior.

The end-effector equations of motion for a redundant manipulator are obtained by the projection of the joint space equations of motion by the dynamically consistent generalized inverse $\overline{J}^T(q)$:

$$\overline{J}^T(q)\left[A(q)\ddot{q} + b(q, \dot{q}) + g(q) = \Gamma\right]$$

$$\Longrightarrow \quad \Lambda(q)\ddot{x} + \mu(q,\dot{q}) + p(q) = F \tag{7.6}$$

This property also applies to non-redundant manipulators, where the matrix $\overline{J}^T(q)$ reduces to $J^{-T}(q)$.

The idea behind the approach for the coordination of the base and the arm proposed in Khatib et al. 1996 is to treat them as a single redundant system. This can be applied to integrate task behavior with task-independent behavior, such as collision avoidance. In equation (7.4), F are the commanded forces at the end-effector to accomplish the task and Γ_0 are the joint forces mapped into the null space to implement task-independent behavior.

7.4.2 Obstacle Avoidance

Recall that the forces defined by the task F_{task} are translated into joint torques Γ_{task} using the Jacobian $J(q)$ of the manipulator.

$$\Gamma_{task} = J^T(q)F_{task}. \tag{7.7}$$

Operational space control of redundant manipulators can accommodate different kinds of motion behavior. In the simplest case a part has to be moved on any trajectory between two locations. To implement obstacle avoidance, the existing trajectory can be modified to accommodate unforeseen or moving obstacles, or to evade another robot sharing the same workspace. No particular motion behavior is required to accomplish the task and the joint torques $\Gamma_{task'}$ can be computed by adding the torques resulting from internal and external forces to equation 7.7:

$$\Gamma'_{task} = J^T(q)F_{task} + \sum_{p\in\mathcal{R}} J_p^T(q)F_p, \tag{7.8}$$

where F_p is the sum of internal and external forces action at point p and $J_p^T(q)$ is the Jacobian at that point in configuration q. Internal forces are caused by the potential function V_{int} associated with the virtual springs of the simulated elastic material and external forces can be derived from the potential function V_{ext} resulting from the proximity of obstacles: $F_p = -(\nabla V_{int} + \nabla V_{ext})$.

7.4.3 Task Execution

Redundancy of a robot with respect to its task can be exploited to integrate task behavior and obstacle avoidance behavior. Holding a subassembly, inserting a subassembly, or tracing the surface of an assembly requires the end-effector to remain stationary or to move along a given trajectory without deviation. This entails that obstacle avoidance cannot alter the motion of the end-effector.

Using the decomposition of equation 7.4, the end-effector can be controlled by a force F_{task} in operational space, whereas internal motions can be independently controlled by joint torques Γ_0 that do not alter the end-effector's dynamic behavior.

Using this framework, obstacle avoidance behavior is implemented by exploiting the redundant degrees of freedom of the robot without influencing end-effector motion. If kinematic constraints of the robot make obstacle avoidance impossible the task needs to be aborted or modified. This can occur when the robot reaches the border of its workspace, for example.

7.5 Motion Coordination

The previous sections described how a single robot can execute tasks in dynamic environments. In this section we will extend those concepts to multiple robots sharing the same workspace. The motion planning problem in this situation becomes much more complicated. The brute force approach consists of considering all robots as a single system and planning in the resulting high-dimensional combined configuration space. Due to the computational complexity of motion planning in such spaces, this approach can only be applied to rather simple problems.

To overcome this computational difficulty, a frequent approach to multiple-robot motion planning is to simplify the planning problem. One way to reduce the computational complexity of planning is to reduce the dimensionality of the configuration space, which can be achieved by projecting the robots into a subspace, most often a plane (Chu and ElMaraghy 1992; Li and Latombe 1997; Mirolo, Pagello, and Qian 1995). These algorithms can only be applied to tasks in which the robot's projection does not overlap with obstacles' projections. A different way to simplify the planning problem is to restrict robots to move on independently planned roadmaps (Lavalle and Hutchinson 1996). These simplifications, however, do not allow to tackle general problems for robots with many degrees of freedom.

The only planner that was used to control two physical robots sharing the same workspace in real-time draws upon a series of standard motion planning techniques (Li and Latombe 1997). Two planar robot arms with two degrees of freedom each are used to assemble parts that are delivered on a conveyor belt. The overall problem is divided into subproblems in lower dimensions that can be solved efficiently.

A potential field-based planner in configuration-time space was applied to the problem of coordinating the motion of multiple robots (Warren 1990). As with any planning approach in configuration-time space, the motion of the obstacles needs to be known a priori. This algorithm has been applied to two planar robot arms with two degrees of freedom.

Assuming static obstacles, the motion planning problem of multiple robots can be solved using optimization techniques (Çela and Haman 1992). Gradient descent methods are applied to a quality function that incorporates kinematic constraints of the robot and stationary obstacles in the environment. Those methods are generally too computationally expensive to allow real-time control of a robot.

The problem of integrating the two phases mentioned above, the motion planning phase and the motion execution phase, was addressed by a layered multi-robot planning and control system (Chu and ElMaraghy 1993). A planner generates high level tasks that are then executed robustly on a low level controller. This abstraction increases flexibility, because the precise sequence of motion commands is determined at execution time, taking into account the latest information about the environment. Moving obstacles, however, could invalidate a plan and therefore require frequent replanning.

Once trajectories for multiple robots have been determined independently, they can be modified to avoid collisions between them. The velocity tuning approach (Kant and Zucker 1986) is a decoupled approach (Latombe 1991) to motion coordination that modifies the velocity of a robot along a fixed path so that all moving obstacles are avoided. The motion of these obstacles must be known. Path coordination (O'Donnell and Lozano-Pérez 1989) is also a decoupled approach to motion coordination. It performs velocity tuning for two robots in conjunction. It also assumes static obstacles. Independently planned trajectories for both robots are time-parameterized to avoid collisions.

Decoupled approaches, such as velocity tuning and path coordination mentioned above, are inherently incomplete. Centralized approaches, on the other hand, have to plan in the combined configuration-time space of the robots, a task that quickly becomes intractable. These disadvantages can partially be eliminated by applying the *elastic strip* framework (Brock and Khatib 1997). The extensions to elastic strips presented in this section result in a framework particularly well suited for motion execution for robots with many degrees of freedom in a shared and dynamic environment, like a multi-robot workcell. The augmented elastic strip framework allows for implicit motion coordination of multiple robots sharing the same workspace, combining the advantages of centralized and decoupled approaches.

In a realistic setting for a flexible, multi-robot workcell few assumptions can be made about the motion of obstacles. This creates the need for a constantly updated representation of local free space around the robot and its trajectory. Motion coordination could be achieved by simply regarding other robots and their trajectories as obstacles that influence the local free space and therefore modify the elastic strip. This approach is called *trajectory modification*. In tight environments, however, independently planned trajectories of different robots might pass through the same narrow passage. Hence, this approach would lead to unavoidable collisions and require replanning.

7.5.1 Velocity Tuning

Given an elastic strip, the current dynamic state of the robot, and its dynamic limitations, an approximate time-parameterization of the trajectory can be computed. This temporal information about the trajectories of multiple robots can be exploited for motion coordination. Using *velocity tuning* (Kant and Zucker 1986; O'Donnell and Lozano-Pérez 1989) for multiple robots, collisions can be avoided by simply changing the time-parameterization of the elastic strip and the trajectory it represents.

Figure 7.3 shows an example of velocity tuning. Two independently planned trajectories are shown in part a). The time-parameterization is indicated by numbers. At time $t = 3$ a conflict is detected. An arbitrary or task-dependent prioritization of the trajectories is imposed to avoid circular dependencies. By changing the time parameterization of the robot with lower priority, as shown in Figure 7.3 b), the conflict is resolved.

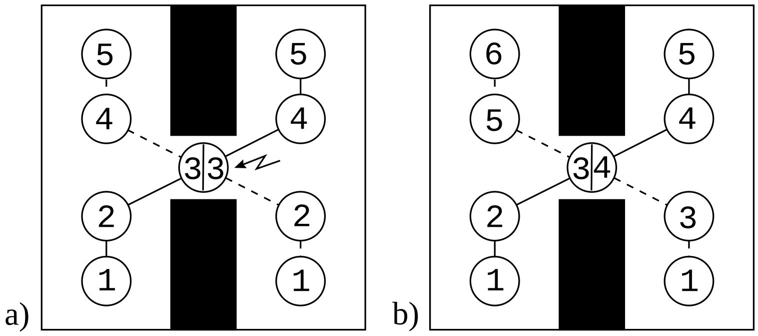


Figure 7.3: In part a) of this figure two trajectories are shown. The numbers indicate the time-parametrization of the trajectory. The two robots would cause a collision by attempting to pass through the same opening at the same time. By changing the velocity of the one of the robots, as shown in part b), the conflict can be resolved.

Collision checking between configurations of different robots is efficiently performed using the protective hull of a configuration. To determine a conflict only configurations that represent the robot's configuration at overlapping time intervals need to be considered for collision. Nevertheless, for n robots there are $O(tn^2)$ possible collisions that have to be considered, where t is the number of time intervals. As a result, this approach becomes computationally inefficient for a large number of robots. In a workspace shared by multiple robots, however, n is usually a small number and this algorithm remains practical.

The velocity tuning approach will fail, if two robots are attempting to switch positions or if two trajectories have the same priority, indicating that the task does not permit a change of time-parameterization.

7.5.2 Velocity/Trajectory Tuning

Velocity tuning and trajectory modification by themselves can fail to resolve a conflict. Velocity tuning might also result in a suboptimal conflict resolution when one robot has to wait for a long time for another robot to pass the region of conflict. These disadvantages can be eliminated by combining velocity tuning and trajectory modification.

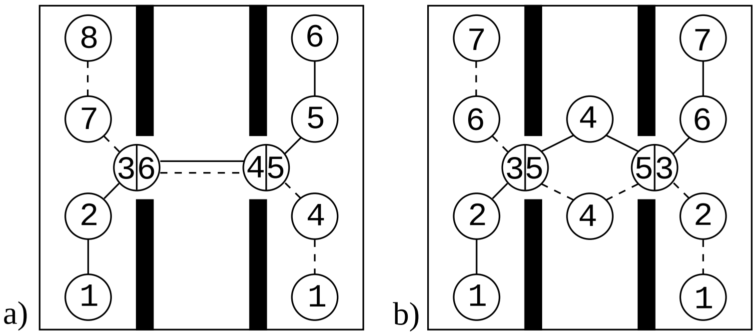


Figure 7.4: A collision can be avoided by velocity tuning, as shown in part a) of this figure. By modifying the trajectory, as shown in b), a more time efficient solution can be found.

In Figure 7.4 a) two independently planned trajectories are shown after the existing conflict has been resolved using velocity tuning only. One robot has wait for the other to pass through the room in the middle. Figure 7.4 b) shows the trajectories that result from trajectory modification.

Note that taking advantage of the time-parameterization of the elastic strip, the trajectory modification, like velocity tuning, only has to consider collisions between configurations during similar time intervals.

Depending on the tasks of the robots, velocity tuning or trajectory modification are the preferred approach to motion modification. If one approach should fail, however, both approaches can be applied in conjunction to attempt conflict resolution. Motion coordination is performed strictly based on task-specification. Potential motion conflicts and moving obstacles do not need to be considered during planning. The elastic band incorporates velocity tuning and trajectory modification to result in implicit motion coordination.

Velocity/trajectory tuning combines decoupled planning with centralized trajectory modification. This results in a motion coordination approach that avoids the computational expense of planning in the combined configuration-time space of multiple robot. However, it resolves motion conflicts beyond the capabilities of decoupled approaches. The trajectories for all the robots in the workcell can determined independently and are coordinated by the elastic strip framework.

The need for replanning, however, is not eliminated entirely. For two robots exchanging their position in a narrow corridor the planning algorithm has to generate a motion for both robots into an open area to perform the exchange. The need for replanning can be detected using equation 8.1. During the modification of the elastic strip external forces and the insertion of intermediate protective hulls assure a collision-free path. If equation 8.1 remains violated, the chosen trajectory is topologically not feasible and replanning is necessary.

7.6 Motion in Contact

Many tasks require the robot to be in contact with the environment. When sanding a surface or inserting a peg into a hole, for example, the end-effector or the manipulated object have to touch an obstacle. In such a situation the free space around the part of the robot in contact with the environment is difficult to represent, as it needs to capture the notion of a "desired collision". This section introduces high-level modifications to the world model that allow the application of the elastic strip framework to motion in contact with the environment. Using this extension, the elastic strip framework unifies execution methodologies for gross and fine motion.

7.6.1 Modified World Model

Let us assume that a given elastic strip represents a motion during which the end-effector of the robot is in contact with an obstacle in the environment. In the elastic strip framework as described above, this contact would be considered a collision and repulsive forces would push the end-effector away from the contact, thus interrupting task execution.

A task that requires the end-effector to be in contact with the environment can be specified by a contact force (Khatib 1998). This contact force is desired and necessary for the execution of the task. It implicitly specifies the spatial relation between the obstacle and the end-effector. Therefore, the contact cannot be considered a collision. To accommodate this task-dependent description of spatial relationship, the world model used in the elastic strip approach is modified. Whereas before every obstacle in the environment exerted a repulsive force on every rigid body of the robot, the modified world model will allow obstacles to become *transparent* with respect to a specific part of the robot.

The concept of transparent obstacles is illustrated in Figure 7.5. The task commands the end-effector of the manipulator to remain in contact with the obstacle. To allow the execution of such a task within the elastic strip framework, the obstacle with which the end-effector is in contact is considered *transparent* with respect to the end-effector. If an obstacle $\mathcal{O}^b_{transparent}$ is transparent with respect to rigid body b, it is not taken into account when computing the external forces acting on b. Hence, in Figure 7.5 the gray obstacle does not exert repulsive forces on the end-effector.

By introducing transparent obstacles the elastic strip framework can be extended to motion in contact. Notice that the obstacle and the rigid body in contact with the obstacle both need to be convex. If this condition does not hold, maintaining a contact does not exclude collisions between other parts of the rigid body and the obstacle.

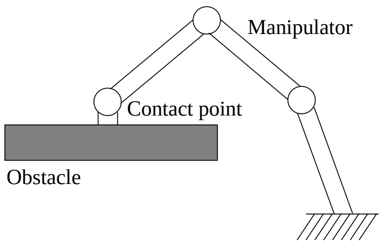


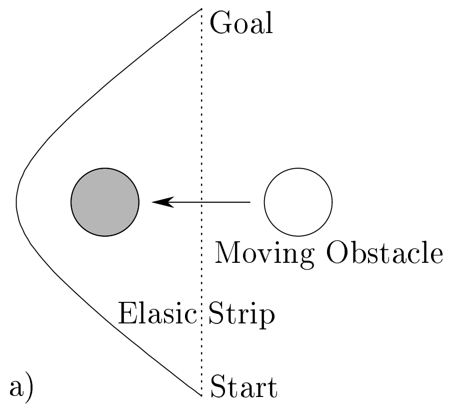
Figure 7.5: The free space around a contact between the end-effector and the environment cannot be modeled using bubbles. The object is declared *transparent* with respect to the part of the robot it is in contact with and is not taken into account for free space computation. It continues to have a repulsive effect on other parts of the robot.

7.6.2 Integration of Gross and Fine Motion Execution

By being able to integrate gross motion determined by a motion planner and represented as an elastic strip, with a task execution scheme based on moments and contact forces with respect to the environment, the elastic strip framework unifies the execution of gross and fine motion. Given a redundant manipulator, the end-effector is able to perform a fine motion task, using force control, while the redundant degrees of freedom are used to generate reactive motion behavior for obstacle avoidance. This capability allows the safe and robust manipulation of objects in dynamic and populated environments.

7.7 Replanning

An elastic strip represents a trajectory that was previously planned by a robot motion planner. This trajectory is modified in real-time to accommodate changes in the environment. Over time, the motion of obstacles can cause significant quantitative and qualitative changes, i.e. an obstacle can move by a lot, or a door can open, changing the connectivity of the free space. Due to those changes the trajectory might become highly suboptimal, or even invalid.



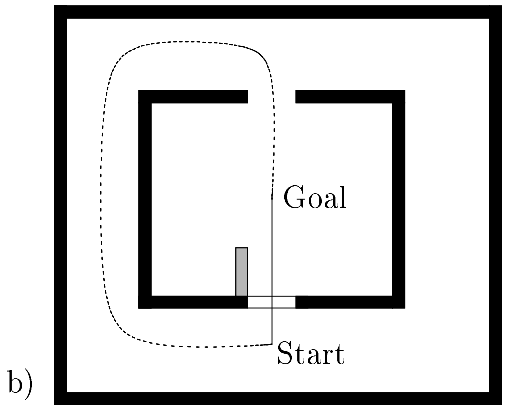


Figure 7.6: For the elastic strip to represent an optimal trajectory, replanning might be required. In part a) of this Figure it is illustrated how a single moving obstacle can make a path arbitrarily suboptimal. Part b) of this figure illustrates how changes in the environment might render an elastic strip suboptimal.

Two such cases are shown in Figure 7.6. Figure 7.6a) shows a *quantitative change* in the environment, where an obstacle has moved a long distance and deformed the elastic strip accordingly. The original trajectory is shown as a dotted line and the modified as a solid one. The modified trajectory represents a large detour and it would seem natural to alter the elastic strip as if the obstacle had "moved through" the trajectory. Such a behavior would result in the elastic strip to assume its original shape, shown as the dotted line. The position of the moving obstacle at planning time is given by its outline. The moving obstacle is shown in gray at its current location.

A *qualitative change* is illustrated in Figure 7.6b). The initial trajectory shown as a dotted line was generated under the assumption that the bottom door of the room was closed. Now that is is open, a much shorter trajectory has become possible; this trajectory is shown as a solid line.

Qualitative and quantitative changes in the environment differ fundamentally in that the former reflect a change in the connectivity of the free space, whereas the latter only reproportion the areas of free space and do not alter the topological properties of the free space. Quantitative changes are therefore a local phenomenon – only portions of the trajectory need to be modified to obtain an optimal or near-optimal trajectory – and qualitative may require a global reevaluation of the entire plan.

Those changes will occur frequently in dynamic environments. For the elastic strip framework to remain practical for motion modification and execution, the impact of qualitative and quantitative changes on its performance has to be minimized. Hence, replanning has to become an integral part of the elastic strip framework. In replanning, as opposed to planning, one has to strike a balance between optimality and efficiency, taking advantage of the information that is contained in the previously planned and potentially modified trajectory.

In this section various replanning approaches are presented, which can easily be integrated with elastic strips. They vary in their computational complexity and in their applicability to quantitative and qualitative changes of the environment. The difficulty in integrating a replanning approach with the elastic strip framework lies in the real-time requirement. Ideally, every time a motion command is issued, a new plan should be generated taking into account the latest sensory information about the environment. However, generating motion plans for most practical tasks requires computation time that does not allow real-time execution. The approaches presented below aim at finding a compromise between completing replanning fast for real-time execution and guaranteeing an optimal trajectory. Their practicality is highly dependent on the planning task at hand. Therefore, an algorithm tailored to a particular task is likely to perform best. The approaches given here can only be viewed as a set of building blocks for a solution tailored to the specific task.

7.7.1 Popping Through

In Figure 7.6 a) a qualitative change of the environment is shown that is likely to lead to a highly suboptimal trajectory, commanding the robot to meander around obstacles that once crossed the trajectory. Compared to qualitative changes, such a situation will occur quite frequently in environments with moving obstacles. It can also be argued that qualitative changes clearly pose a new planning problem that does not need to be handled by a motion execution paradigm. Hence, it is justified to devise a replanning solution that can only deal with qualitative changes.

Considering Figure 7.6a), it would appear quite natural for the moving obstacle to *"pop through"* the elastic strip. The strip would "open up", let the obstacle pass through, and then "close" again to take the shape of a straight line trajectory between the start configuration and the goal, indicated by the dotted line.

It turns out that such "popping" is quite easily implemented by a slight modification of the procedure inserting and deleting protective hulls into and from the elastic strip, described in Section 7.3.3. The original procedure was designed to ensure that the robot along its trajectory always remained within the elastic tunnel. When an obstacle approaches the strip and the size of protective hulls shrinks, intermediate configurations are inserted until the trajectory is contained within the elastic tunnel. This behavior will cause the path to remain covered until it is no longer physically possible, because the obstacle has pushed the robot against another object in the environment. For the obstacle to pop through the elastic strip, however, it is necessary to tolerate the trajectory to be temporarily uncovered. The trajectory separates into two disconnected pieces between which the obstacle can pass.

The modified procedure that allows popping through exploits a simple property of the elastic band representation. As an obstacle comes closer and closer to the robot, its repulsive force grows in magnitude until the component opposing the internal force between two adjacent configurations exceeds the internal force itself: the obstacle pushes the two configurations further apart. As opposed to the original procedure that inserts configurations to maintain coverage, the new procedure only does so until a certain relative distance measure between configurations is reached. By not inserting more configurations the trajectory is not valid any more, the obstacle is able to pass through, at which point the trajectory is restored automatically, due to internal forces.

Despite its efficiency and effectiveness in most situations, there are cases in which the modified procedure fails. Once the trajectory has been separated by external forces, the internal forces cause the elastic strip to close again after the obstacle has moved through. If the obstacle does not move far enough, internal forces might not be sufficient for the two ends of the trajectory to be joined again. This occurs, for example, if an obstacle comes to rest on a straight line trajectory between the start and the goal configuration. This situation is illustrated in Figure 7.7. The disconnected elastic strip is shown as a dashed line. The internal forces f_{i_1} and f_{i_2} contain no component that could cause a lateral motion at the ends of the trajectory. A local minimum of the total energy of the elastic strip is reached and causes the procedure to fail.

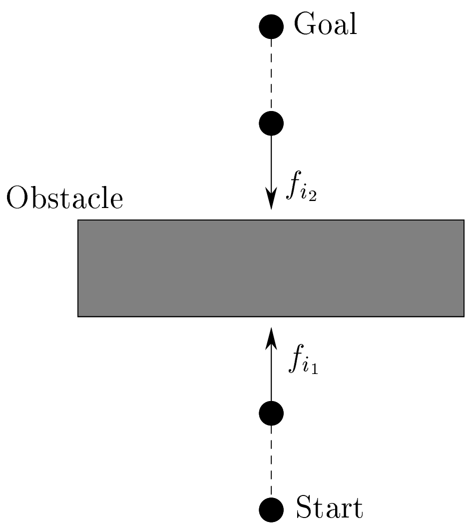


Figure 7.7: The process of an obstacle popping through the elastic fails when internal forces cannot reconnect the elastic strip. In such a situation traversal of the obstacle boundaries might restore connectivity of the elastic strip.

Such a failure is easily recognized, because the elastic strip remains disconnected. A local replanning method, such as wall-following (Lumelsky 1987), can be used to reconnect the elastic strip in an efficient manner. The time-parameterization of the trajectory has to be modified to accommodate disconnected trajectories and to delay motion execution in the disconnected region of the elastic strip until the local replanner has generated a new intermediate trajectory. In conjunction with local replanning, popping through is an efficient and effective way of solving the problem of qualitative changes.

7.7.2 Alternative Routes

Another approach of dealing with changes in dynamic environment consists of keeping track of a set of *alternative routes*. These can be considered as detours for the entire trajectory or just for parts of it. Since it is very efficient to verify that a given trajectory lies in free space, many alternative routes can be maintained as elastic strips. An obstructed trajectory can then be modified by replacing it partially or in its entirety with another one.

There are various ways of generating a set of alternative routes. The simplest approach consists of remembering the trajectory that was initially generated by the planner. When dealing with a quantitative change, this initial trajectory can be checked periodically. If it lies in free space, various criteria can be used to determine whether it should be preferred over the incrementally modified one. The length or total execution time could serve as such criteria. The dynamic state of the robot can also influence this choice.

In Figure 7.6 a) the dotted line represents this initial trajectory that is used as an alternative route. As the obstacle deforms the elastic strip further and further, the original trajectory lies in free space again and can be chosen over the deformed one to result in a plan with minimal length. In this rudimentary form, alternative routes can be used to accommodate quantitative changes.

A different approach, handling quantitative and qualitative changes puts much effort into the planning phase. Here, the planner creates many trajectories for the same planning task. Alternatively, the connectivity information of the free space can be used to generate a roadmap of trajectories. This roadmap can then be used as a lookup table for local replanning. In particular, the probabilistic motion planning approach presented in Section 6.1.1 is well suited for this task, as it allows different planning queries in the same environment to be answered efficiently.

Reconnecting a disconnected elastic strip can then be achieved by a graph search, combined with the simple local planning effort of connecting the elastic strip to the roadmap. Using this more involved version of alternative routes, the elastic strip can be updated and modified to represent a near-optimal trajectory in the presence of qualitative and quantitative changes. This methods is best suited for environments in which the basic connectivity of the free space remains unaltered, except for small moving obstacles and passages, like doors opening or closing. This ensures that the preplanned set of alternative routes remains relevant over time.

Due to the real-time requirement of motion execution, replanning must be interleaved with motion execution. Since the entire trajectory is being modified at all times, it is possible to predict the requirement of replanning. Once detected, the replanning process can be initiated and executed concurrently with the motion execution. As one would expect, it might be necessary to slow down or stop the motion of the robot to wait for the replanning process to be completed.

7.7.3 Concurrent Replanning

Ideally, one would like to generate an entire trajectory each time a motion command is issued. Only then it would be guaranteed that for a given situation in a dynamic environment the optimal trajectory was executed. Such a motion planning and execution paradigm would obviously be able to react appropriately to qualitative and quantitative changes. However, the planning task is too complex and computationally expensive to allow real-time performance. Results about the computational complexity of motion planning are surveyed in Section 10.1.

If an upper bound on the duration of the planning process is known, it can be used to predict the robot's position at the time the planning is completed. The path from that position to the goal configuration can be replanned concurrently during motion execution. The remaining part of the trajectory is updated once the planning process is completed. This can be done repeatedly until the goal configuration is reached. This approach can modify the trajectory to accommodate qualitative and quantitative changes. However, if the change occurs in the portion of the trajectory that will be executed during the concurrent planning phase, the trajectory will remain suboptimal and the robot might even have to come to a stop to wait for a new plan that connects the current and the goal configuration.

The shorter the duration of the planning process, the closer we can get to the ideal situation of generating a new plan for each issued motion command. Since planning the entire path for a robot with many degrees of freedom in a dynamic environment might take on the order of minutes, it seems desirable to replan subsections of the trajectory only. The planning process of a localized subsection can be accelerated by only considering a local region of the configuration space. This is a heuristic that might fail to find a trajectory even though one exists or that might generate a suboptimal trajectory. Nevertheless, the trajectory executed by the robot will always be at least as good as without replanning.

One method of selecting the portion of the trajectory to which replanning is applied chooses a random starting and ending location (Barraquand and Latombe 1991) on the current trajectory. Selecting locations that lie close together is likely to improve the trajectory in the presence of quantitative changes and locations further apart on the trajectory can remedy suboptimalities of the trajectory resulting from qualitative changes.

When configurations are chosen at random, planning time might frequently be waisted by replanning a trajectory that is already optimal. It is possible to improve this method by exploiting information about the elastic strip in order to reduce the number of times an optimal trajectory is replanned. This discussion assumes that the length of the trajectory is to be optimized. For the ease of presentation, we will discuss the case of a point robot in a two-dimensional workspace. The concepts presented below can easily be extended to articulated bodies moving in three dimensions.

Any deviation from the optimal straight line trajectory between two points must be caused by an obstacle in the environment. For a point robot moving in two dimensions this means that the second derivative of the function describing the trajectory is non-zero. Those regions of the trajectory for which that condition holds are called *critical regions*. The set of critical regions of a plan in two dimensions is shown in Figure 7.8 as a solid line. A dashed line indicates a straight-line portion of the plan.

The notion of critical regions can be exploited in choosing the portion of the plan

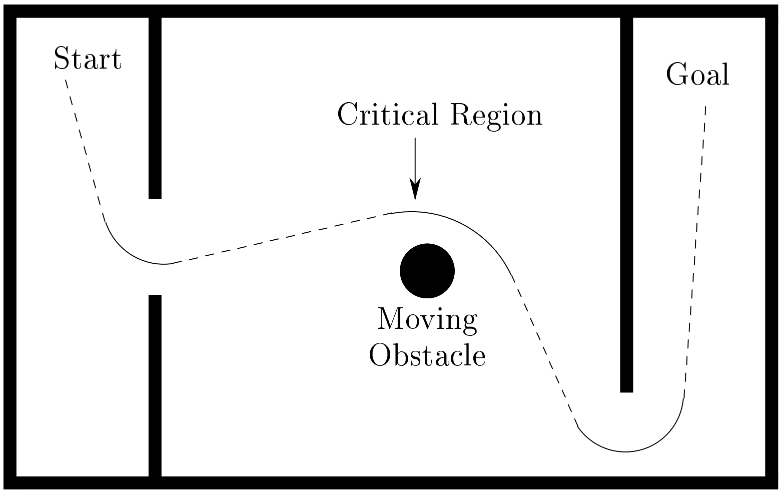


Figure 7.8: A critical region is considered an area on the elastic strip where proximity to obstacles in the environment and internal forces impose constraints on the motion of the configurations along the elastic strip.

to which replanning is applied. If start and end point of that region are chosen to lie on two non-consecutive critical regions, it is guaranteed that replanning can improve that portion. This is due to the fact that there must exist at least on obstacle that causes a deformation and hence might be circumnavigated in a more optimal manner. In Figure 7.8, for example, by picking the first and the last critical region of the elastic strip and replanning the middle part of the trajectory, the suboptimality of the path caused by the moving obstacle can be eliminated.

The probability distribution that determines which critical region is chosen can also be used to realize simple heuristics. For example, if the start point is frequently chosen in a critical region close to the current position of the robot and the end point is likely to be close to the start, suboptimalities of the trajectory due quantitative changes of the environment will be avoided. Replanning a larger portion of the trajectory to handle qualitative changes is performed with lower probability.

7.8 Parallelization

The modification of a trajectory in the elastic strip framework consists of two major steps: first, the local free space around the trajectory is computed, and subsequently the information about the local free space and the proximity of obstacles is used to modify the elastic strip. These two steps can be parallelized to increase the efficiency of the elastic strip framework. The problem of parallelization (Akl 1989) is to subdivide the overall task into subtasks that can be treated independently.

The problem of free space computation lends itself very easily to parallelization. The assumption that the elastic strip remains unchanged during this process makes every distance computation completely independent. If an infinite number of processors would be available, computations for each point on every configuration along the elastic strip could be performed in parallel. The computational complexity of the free space computation would be the computational complexity of one single distance computation.

Given a finite number n of processors, the computation of free space has to be discretized as described in Section 8 and a covering criterion described in Section 8.3 is used to compute the free space around rigid bodies. The use of those approximations causes the sequential algorithm to be more efficient, but they introduce a dependency between distance computations for the same rigid body. As one rigid body can be described by a set of spines, as described in Section 8.1, the expected speedup of the free space computation depends on the number of spines contained in the configurations used to describe the elastic strip. Assume that c configurations are used to represent the elastic strip and that the robot in each of those configurations is described by s spines. Furthermore, we will assume that the number of distance computations necessary to cover one spine is bounded by a constant. The speedup for the free space computation introduced by parallelization on n processors is then given by:

$$\text{speedup}_{\textit{freespace}}(n, c, s) = \begin{cases} n & \text{if } (c \cdot s) \geq n \\ c \cdot s & \text{otherwise} \end{cases}$$

From this equation it can be inferred that the free space computation for elastic strip is completely parallelizable: n processors will result in a speedup of n. This results from the fact that each spine can be covered completely independently.

A parallelization of the modification procedure for elastic strips has to divide the problem into independent subproblems. Since all configurations along the strip are connected by virtual springs, the modification of one configuration theoretically has an impact on its neighbors. In principle, this influence propagates along the entire strip. By exploiting the notion of *critical region* introduced in Section 7.7.3 it is possible to find those portions of the elastic strip that can be regarded as independent subproblems.

Intuitively speaking, critical regions are portions of the elastic strip in which an obstacle imposes constraints on the possible motion of the configurations in that region. These constraints reduce the impact that the modification of an adjacent configuration has on its neighbors. This insight allows parallelization of the computation procedures associated with elastic strips. The assumption that two adjacent configurations within the critical region will not influence each other and thus can be treated independently is made. Such a pair of configuration can be chosen arbitrarily in every critical region for each servo tick. At those configurations the elastic strip is divided into separate regions. For each of those *independent regions* the external and internal forces can be computed separately.

Given a finite number n of processors, assume r to be the number of critical regions along the elastic strip. Furthermore, let c_i denote the number of configurations in the independent region $i, 1 \leq i \leq r$ and let $\max(c_i)$ denote the maximum number of configurations for any independent region. The speedup introduced by parallelization of the modification process is governed by the independent region of the elastic strip with the most configurations:

$$\text{speedup}_{modification}(n, r, c, s) = \begin{cases} \frac{c}{\max(c_i)} & \text{if } r \leq n \\ \min\left(\frac{c}{\max(c_i)}, \frac{r}{\lceil \frac{r}{n} \rceil}\right) & \text{otherwise} \end{cases}$$

If there are more processors than there are independent regions of the elastic strip the most complex independent region determines the speedup of the modification procedure. Each region is assigned to a processor and the longest one determines the overall termination of the procedure. When there are fewer processors that independent regions, multiple regions have to be assigned to one processor. Now we have to distinguish between two cases:

1. If there is one very long independent region so that $\max(c_i) > \frac{c}{n}$ the modification of this region will determine the speedup. The situation is comparable to the situation when $r \leq n$.

2. When the computation time is not governed by one particular independent region, they can be scheduled to different processors to minimize computation time. An obvious upper bound for the speedup is $\frac{c}{n}$. This bound holds if every independent region contains a number of configurations that is a multiple of n. A lower bound can be found by considering critical regions of arbitrary sizes. Since case 1 does not hold, we can assume that $\max(c_i) \leq \frac{c}{n}$. For a worst case analysis we will assume $\max(c_i) = \frac{c}{n}$ for all r critical regions. Since no more than $\lceil \frac{r}{n} \rceil$ critical regions can be assigned to one processor, the speedup gained

by scheduling r critical regions to n processors is hence bounded by $\frac{r}{\lceil \frac{r}{n} \rceil}$.

The actual speedup results from the minimum of both cases. This analysis shows that the size and number of critical regions has a crucial impact on the speedup that can be expected by parallelizing the modification procedure. In practice it can be expected that obstacles in the environment divide the elastic strip into independent regions quite frequently.

During the modification procedure it might become necessary to insert intermediate configurations into the elastic strip to guarantee the connectivity criterion (see Section 8.5). For simplicity it is assumed that the number of inserted configurations is equal to the number of configurations that was removed so that the number of configurations within an independent region remains constant. The cost of inserting and removing intermediate configurations are ignored here.

The overall speedup of the elastic strip algorithms (free space computation and modification procedure) is bounded by the minimum of the two speedups, the speedup obtained for the modification procedure. The overall computational complexity of the elastic strips framework is discussed in Chapter 10.

7.9 Sensing

The modification procedure for elastic strips requires an approximate representation of the free space around the entire trajectory at all times. As this representation is derived from the world model that is updated using sensing, the workspace of the robot has to be monitored at all times along the entire elastic strip to guarantee that the trajectory remains valid. It is unrealistic to assume that sensing based on the robot, such as infrared, sonar, laser range scanner, or computer vision cameras, can accomplish this task. From most points along a realistic trajectory large portions of the workspace will be obstructed. If a robot is moving down an L-shaped hallway, for example, the trajectory can only be perceived in its entirety at the bend of the L. At any other location along the path some portion of the free space around the trajectory is obstructed and hence cannot not be update according to changes in the environment.

In controlled environments this problem can be solved by introducing sensing that is not based on the robot. Each room accessible to the robot can be equipped with multiple cameras. Their number and their mounting locations have to be chosen such that they are able to constantly perceive the entire room, even in the presence of obstructions by moving obstacles. In the computer vision literature using a large number of cameras to capture the motion of objects within a given workspace volume was first suggested in Neumann, Ulrich, and Fuchs 1993 and is referred to as using a *sea of cameras*. The information gathered by this redundant sensing system is transformed into a geometrical model and position information of obstacles, which can be used by the elastic strip modification procedure to compute a representation of the local free space around the trajectory. To facilitate this task, stationary obstacles can be identified and only moving or newly introduced obstacles need to be detected.

In order to infer the three-dimensional shape of objects in the environment from two-dimensional images captured by different cameras, depth information about the scene has to be extracted. The computer vision literature presents various approaches to accomplish this: depth from stereo, motion, focus, and defocus (Nalwa 1993; Trucco and Verri 1998; Raskar et al. 1998). Adequate sensing for the elastic strip framework, however, needs not only to compute the shape of objects in real time, but also over a large volume of space, integrating a large number of views into one coherent model. Similar problems have been addressed in the computer vision literature. A depth from stereo hardware architecture was designed, deriving depth information from the images captured by six cameras at a rate of 30 Hz (Kanade et al. 1995; Kanade et al. 1996). This architecture has recently been expanded to capturing the shape and motion of objects in a room using 49 cameras covering the walls and ceiling (Kanade and Vedula 1998).

To avoid the correspondence problem (Nalwa 1993) encountered in depth from stereo, an active depth extraction method has been developed (Nayar, Watanabe, and Noguchi 1995). In this approach a visual pattern is projected onto the scene. Matching the perceived pattern with the projected one, the shape and location of scene objects can be determined. This approach also achieves a frame rate of 30 Hz and depth information is computed with an accuracy of 0.3%.

These results from the computer vision literature indicate that the sensing requirements of the elastic strip framework can be fulfilled by state-of-the-art computer vision technology. These existing approaches, however, solve a problem much harder than the one required for the elastic strip framework: They attempt to determine the precise shape of objects in the scene. In the elastic strip framework we are not concerned with extracting the precise features of objects in the scene but rather with determining obstructed areas of free space. In the remainder of this section a new approach to this simpler task is outlined. This approach is efficient and robust with respect to occlusion, which reduces the number of cameras required for a certain environment.

The most difficult part to computing depth information in a scene from stereo or from motion is the correspondence problem. The location of corresponding image features has to be determined in a series of images, either taken from different camera angles or at different moments in time. Once the correspondence has been established, simple geometry can be used to infer the position of the feature in space. The method presented here avoids this computation entirely by taking a different approach to merging the information contained in images of the same scene taken from different camera positions. In a way, this method integrates depth from stereo and depth from motion to result in a shape form stereo and motion paradigm.

The basic assumption of the algorithm is that the relative position and orientation of all cameras in the scene with respect to each other is known. This can be accomplished very easily using an adequate procedure for camera calibration (Stein 1999). By placing uniquely identifiable landmarks at known locations the position and orientation of the cameras can be determined. The number of those landmarks has to be chosen such that the field of view of every camera contains enough of them to determine the camera's parameters.

The algorithm proceeds in two steps: First, for each camera in the scene a sequence of images is analyzed to yield the motion field (Nalwa 1993) for the scene. Using discontinuities in the motion field, image regions associated with objects in the scene can be extracted. These regions are likely to correspond to moving obstacles.

Assuming an accurate model of the static environment, we are interested in determining the space occupied by the moving obstacles. Static obstacles could be perceived using color discontinuities of the image. The motion field and color discontinuities can be used in conjunction to be able to determine the image region associated with a moving obstacle after it came to rest.

Consider an image region associated with a non-zero motion field. This region corresponds to the projection of a moving obstacle in the environment onto the image plane. Assuming that the outline of this region is described by a polygon in the image plane, rays emitted from the optical center of the camera through each point of the polygon describe a generalized cone. This cone describes a a volume within which the obstacle must be contained.

In the second step of the algorithm these generalized cones associated with image regions for each camera in the scene are merged into a single scene description. The correspondence problem can be ignored altogether: All cones are intersected and their intersections are candidate volumes containing obstacles. In some cases there might be intersections between cones that do not correspond to an obstacle. A simple consistency check with other cameras perceiving that particular portion of the work space can be used to eliminate such a candidate. Once this elimination process is completed the remaining candidate volumes are regarded as obstacle regions. For n cameras there could be as many as n^2 candidate volumes. By aligning the optical axes of the cameras outside a common plane and assuming a certain sparsity of moving obstacles, it can be assumed that only very few candidate regions are computed that do not correspond to actual obstacles.

As the computations required to determine the candidate volumes are simple, it can be assumed that an implementation of this algorithm can extract obstacle volumes in real time. Note that already two cameras perceiving one moving obstacle can give a meaningful approximation of its volume. Additional cameras improve the accuracy of the estimated volume. The efficiency of this approach results from the fact that only the outline of moving obstacles need to be determined for a given camera position. This also implies a limitation of this approach. The shape of a moving obstacle can only be inferred based on its projection. The cavity of a cup, for example, could not be extracted from its projections onto a plane.

In some robotic applications it might be impossible to equip the entire environment with sensors, such as vision cameras. In those environments, rather than constantly modifying the entire elastic strip, only the visible region can be updated according to the changing free space. The local free space around the current configuration of the robot can be perceived using sensor systems like sonars, infrared sensors, laser range finders, vision cameras, or sensor skins (Cheung and Lumelsky 1992). One possible approach of dealing with imperceivable areas of the environment is to assume they remain unchanged. If a previously unsensed area becomes visible and a change in the environment has invalidated the current trajectory, the replanning methods described in Section 7.7 of this Chapter can be employed to locally generate a new, valid trajectory. Depending on the transience of the environment, this strategy can perform very well and in many cases the effort and cost of sensing the entire environment might not be justified.

7.10 Limitations

When a planner generates a motion plan for a robot, it generally computes a single plan, succinctly describing the joint angles as functions of time. There usually is a large number of other paths, however, that exhibit identical topological properties; they form the set of homotopic paths (Latombe 1991). While this set is difficult to compute and represent, it could be a powerful tool for motion generation, especially when the plan is supposed to be executed in dynamic or partially unknown environments. If the current plan gets invalidated by changes in the environment, for example, the attempt could be made to select a different plan from the same set of homotopic paths. If the changes in the environment are small and incremental, this is likely to succeed. In this manner the information about the connectivity of the free space inherent in the previously generated plan and the corresponding set of homotopic paths could be exploited without expensive recomputation.

As already mentioned in Section 7.2, the elastic strip framework attempts to represent the set of homotopic paths indirectly by a volume in the workspace. The goal for the elastic strip is to select a valid path from the set of homotopic paths, as long as one exists. However, the volume used to implicitly represent the set of homotopic paths is depending on the current path and does not necessarily capture the workspace volume containing all homotopic paths. This means that the elastic strip framework only chooses paths from a subset of all homotopic paths. It is therefore possible for the elastic strip framework to fail to determine a path, although a valid homotopic path exists. This occurs when forces exerted by the obstacles onto the robot in the workspace do not result in the necessary displacement in configuration space for the robot to assume the posture required for the valid homotopic path. In most practical environments, however, this is not likely to occur.

Kinematic constraints and singularities of the robotic mechanism can potentially limit the effect that external forces have on the posture of the robot. This might also lead to failure to find a valid homotopic path, even though one exits. Similarly, approximating the free space, rather than computing its precise representation, results in some homotopic paths to be lost due to the representation. There is an obvious tradeoff between completeness and computational efficiency.

It might be considered an even more severe limitation that a section of the workspace volume around a particular path may contain paths that are not homotopic in configuration space. See Figure 7.9 for an illustration. The inability to differentiate between certain paths that are not homotopic could cause the plan to be changed in a way that does not maintain the homotopy of the original path. However, the internal forces of the elastic strip propagate the constraints that are implied by the homotopy. Therefore this situation cannot arise; the constraint propagation and the validity checking for the elastic strip ensure that the elastic strip always represents a path that homotopic to the one originally generated by the planner.

Given an initial path generated by a planner, if follows from above discussion that an elastic strip without replanning always represents a path that is homotopic the initial path. The elastic strip framework is not complete, however, in the sense that a valid, homotopic path may exist, but the modification procedure fails to find it. This is due to the fact that external forces are used to guide the modification of the elastic strip. This search is not exhaustive, resulting in computational efficiency at

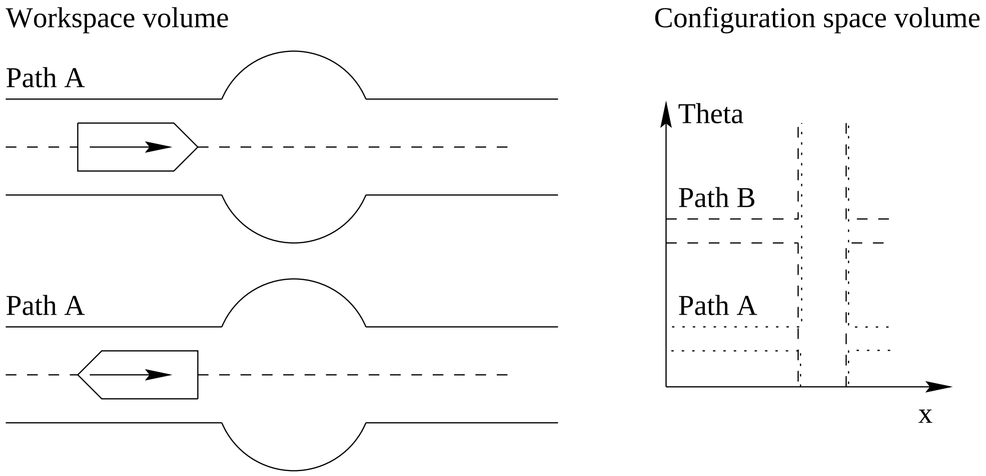


Figure 7.9: The workspace volume surrounding the path of the two robots on the left is limited by the two walls of a corridor. It is easy to see that independently of the orientation of the robot the workspace volumes are identical. In configuration space, however, due to the different orientation of the robot, the values of homotopic paths are different, but share a segment for those values of x where the corridor widens. Here, the robot can rotate and move from one homotopic family of paths to another one. While in configuration space this migration between homotopic sets is obvious, the representation of the work space volume swept by the robot along those paths obscures their difference.

the expense of completeness. In practice this situation will not occur very frequently, as most environments are not very cluttered. In very tight or cluttered environments, where the elastic strip framework is likely to fail, the use of a motion planner will yield significantly better results.

Obviously, whenever the entire set of homotopic paths becomes invalid, the elastic strip framework will fail and replanning will become necessary. In Section 7.7 several replanning schemes were discussed. In particular in environments that contain very tight passages, the elastic strip framework is likely to fail often. This could be remedied by integrating a fast, complete motion planner into the framework. The planner could be applied to locally difficult passages to complement the elastic strip framework. One choice of such a planner would be a probabilistic roadmap planner (Kavraki 1994).

Chapter 8

Free Space Representation

The elastic strip approach is based on the premise that it is more efficient to represent free space directly in the workspace, rather than in configuration space. To render the framework as efficient as possible such a representation should allow for very efficient computation, representation, and manipulation. This chapter introduces the particular choice for the implementation of the elastic strip framework presented in this thesis. Several approximations are made to trade off completeness for computational efficiency. The volume of rigid bodies, for example, is represented using a simple volume approximation. This approximation will be the basis of other algorithms presented later and was chosen mainly with computational efficiency in mind. The elastic strip framework itself is not restricted to these approximations, however, and can also be applied to more accurate and complex volumetric representations, at the expense of performance.

8.1 Rigid Body Description

The configuration, i.e. position and orientation of any articulated body can be represented as a point in the associated configuration space. To determine which configurations of an articulated body are in collision with the environment, configuration space obstacles need to be computed. It is this construction that for most motion planning approaches constitutes the computationally most expensive step. When a body is represented directly in the workspace this step can be avoided, but now we have to represent shape and spatial dimensions explicitly. Therefore, it is crucial to have a very efficient way of describing rigid bodies. This section introduces the *spine* approach to rigid body representation, used in the elastic strip framework.

The motivation for the spine approach is the observation that most robots consists of cylindrical rigid bodies, or at least rigid bodies that can be approximated well by cylinders. If a rigid body does not fit this category, multiple spines can be used to approximate it. Therefore, some form of generalized cylinder seems an appropriate choice for an approximate representation of the the volume of a rigid body.

To facilitate distance computation, we choose to represent rigid bodies by a line segment that is parameterized by a varying radius. The line segment is called the *spine* of the rigid body. The radius parameterization is a linear interpolation between two values r_1, r_2 at the endpoints p_1, p_2 of the line segment. This parameterization specifies a volume around the line segment fully containing the rigid body and describes the free space necessary to guarantee that the body is not in collision with the environment. The volume described by a spine and its radius parameterization is a symmetric generalized cylinder with spherical caps on its circular faces. We shall refer to this description as the *hull* of a rigid body. Figure 8.1 illustrates the parameters that specify the extent of a hull schematically. In Figure 8.2 the spine of a transparent PUMA 560 link is shown as the black line along its central axis; the volume associated with the spine and the radius parameterization, the hull is shown as a transparent shape enclosing the link.

This model might only be a very coarse approximation to some rigid bodies. However, it can be assumed that in most cases the robot will maintain a safe distance from obstacles. In those situations the computationally more efficient check against a coarse representation of its volume suffices to ensure collision avoidance. As a rigid body comes closer to obstacles, the model described above might result in incorrect collision detection, because the volume of the body is overestimated. To address this problem the representation of rigid bodies by spines can be generalized to describe rigid bodies at an arbitrary level of detail.

The basic idea of this generalization is to introduce more spines to cover the volume

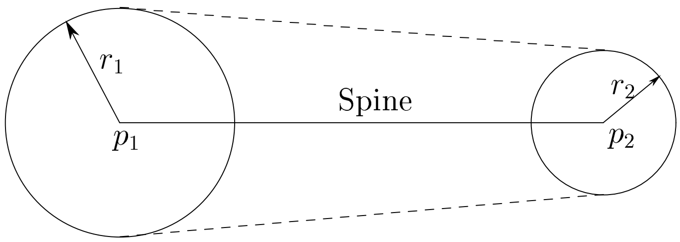


Figure 8.1: Schematic representation of a spine and corresponding hull: The spine is parameterized by a width. At point p_1 the width is r_1 and at point p_2 the width is r_2. The width is interpolated along the spine. The spine and its width parameterization describe a volume, called *hull*.

of the rigid body with increasing accuracy as the distance to obstacles decreases. These spines overlap in order to approximate a non-cylindrical shape with overlapping cylinders. This is illustrated in Figure 8.1 for the rectangular cross section of the PUMA 560 link from Figure 8.2. The cross section is shown in gray; the viewing direction is along the spine of the link. The three images in Figure 8.1 show how multiple spines can be used to represent this body at varying resolutions. On the left, the approximation by only one spine is shown, indicated by the circular cross section of the generalized cylinder. The resolution achieved by this representation is δ_1, indicated by the dotted line.

Figures 8.1 b) and c) show the same cross section represented with an increasing number of spines to achieve an increasing resolution. Using two and seven spines in images b) and c), a resolution of δ_2 and δ_3 can be achieved, respectively. These various levels of accuracy in representation can be viewed as a hierarchy. If the rigid body is close to obstacles and a collision cannot be ruled out with a simple volume description, the hierarchy can be traversed to retrieve representations of increasing level of detail until a collision is either detected or can be disproved. This hierarchy of volumetric models can be generated automatically (Martínez-Salvador, del Pobil, and Pérez-Francisco 1998). For simplicity, we assume in the remainder of this thesis

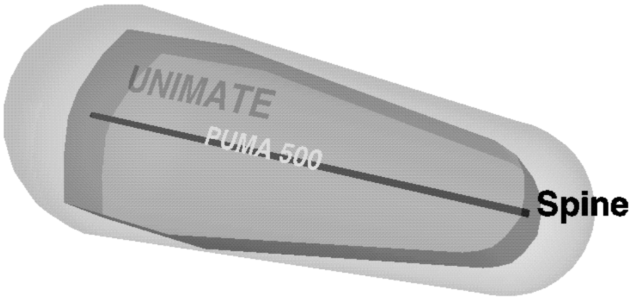


Figure 8.2: Illustration of a spine and corresponding hull: The spine is placed along the major axis of the rigid body. The width-parameterization is chosen such that is fits the rigid body tightly.

that the volume of rigid bodies is described by a single spine.

The representation of rigid bodies by a hull is always an approximation. There is a tradeoff between accuracy in representation of a hull and the computational effort required for its generation and manipulation. Since the volume of rigid bodies is overestimated, collisions will always be detected correctly. In some cases a non-existing collision might be detected according to the approximation of the body. Since for any given such occurrence the resolution of the representation could be adjusted accordingly, a motion generation approach using this approximation can be called *resolution complete* (Latombe 1991).

8.2 Protective Hull

In configuration space the free space around a robot with n degrees of freedom can be described by an n-dimensional convex region, called a *bubble* (Quinlan and Khatib 1993a). This approach to free space description is taken in the elastic band framework (see Section 6.3.1). Here, we are interested in a *workspace representation* of free space.

In choosing a workspace volume representation for free space the most important

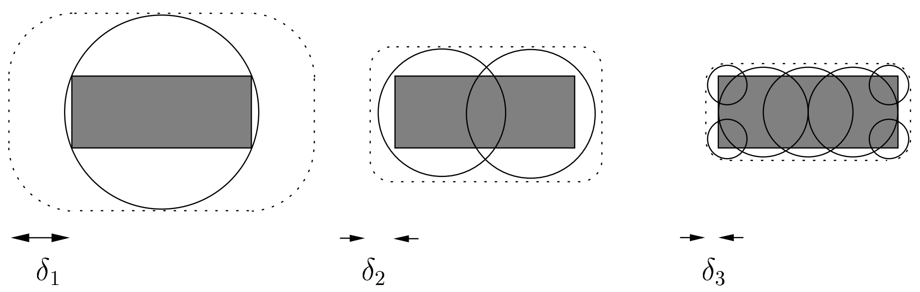


Figure 8.3: Covering a body with spines: Multiple spines can be used to describe a rigid body at various resolutions. The images show a rigid body from a view along the spine. For a coarse resolution up to distance δ_1, a single spine is suffices to cover the body, indicated by the black line. As the resolution increases more spines are needed to describe the volume with increasing accuracy.

criterion is computational efficiency. Hence, a sphere seems ideal: using only one distance computation between the environment and a point, a spherical region of free space can be computed. Its representation requires only four parameters, three for the point and one for the radius. Such a sphere is called a *workspace bubble*. Those workspace bubbles can be used to describe the free space independently of the number of degrees of freedom of the robot. This is an advantage over the elastic band approach, where the dimensionality of the bubble corresponds to the number of degrees of freedom of the robot. However, instead of a single *configuration space* bubble, many *workspace* bubbles might be necessary to describe the free space around a robot with sufficient accuracy. The remainder of this section formally introduces the primitives of free space representation used in the elastic strip framework.

A workspace bubble captures a spherical region of free space around a given point p. Let $d(p)$ be the function that computes the minimum distance from a point p to any obstacle. The *workspace bubble* of free space around p is defined as the set of points s that are closer to p than the closest obstacle in the environment:

$$\mathcal{B}(p) = \{\, s : \|p - s\| < d(p)\}.$$

In the remainder of this thesis we will refer to workspace bubbles simply as bubbles.

An approximation of the local free space around a rigid body b in configuration q can be computed by generating a set of overlapping workspace bubbles $\mathcal{B} = \{\mathcal{B}_1, \cdots, \mathcal{B}_n\}$ centered on the spine. This set of bubbles is called *protective hull* if it contains the workspace volume $\mathcal{V}_q^b$ of the rigid body b in configuration q and is defined as:

$$\mathcal{P}_q^b = \bigcup_{\mathcal{B}_i \in \mathcal{B}} \mathcal{B}_i \supset \mathcal{V}_q^b.$$

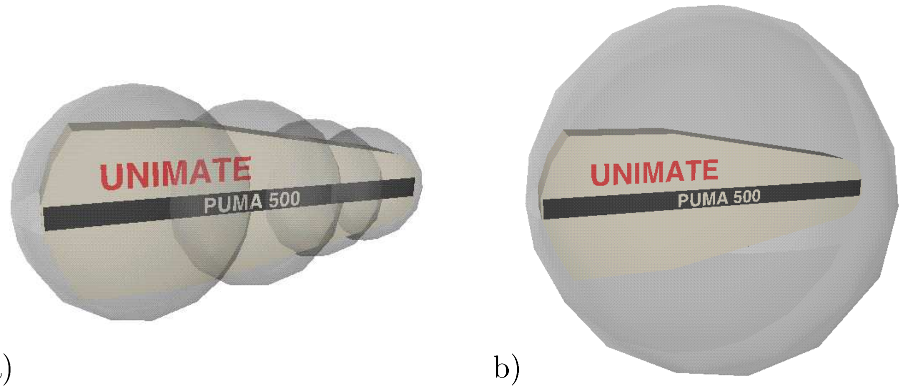
The protective hull of a rigid body can be easily computed using a divide and conquer technique along the length of the spine.

An illustration of a protective hull around a link from a PUMA 560 is shown in Figure 8.4. A set of four workspace bubbles centered on the spine of the link (see Section 8.1) fully contain the volume of the link. Because the bubbles represent free space, the link is guaranteed to lie in free space. Note that the protective hull in Figure 8.4 a) is a very tight representation of free space. The obstacles (not shown in the figure) are fairly close to the link. When obstacles are further away a protective hull with fewer bubbles would suffice to cover the body; this is shown in Figure 8.4 b).

A robot can consist of many rigid bodies. The union of the protective hulls of each rigid body b of a robot $\mathcal{R}$ can be used to describe the local free space at a configuration q. The protective hull $\mathcal{P}_q^{\mathcal{R}}$ for Robot $\mathcal{R}$ in configuration q is defined as:

$$\mathcal{P}_q^{\mathcal{R}} = \bigcup_{b \in \mathcal{R}} \mathcal{P}_q^b \supset \mathcal{V}_q^{\mathcal{R}},$$

where $\mathcal{V}_q^{\mathcal{R}}$ is the workspace volume of the robot $\mathcal{R}$ in configuration q. Figure 8.5 shows two protective hulls of the *Stanford Mobile Platform* in different configurations. The small, opaque spheres represent obstacle. The extent of each bubble is determined by the distance to the closest obstacle. Note that a single workspace bubble may contain



a) b)

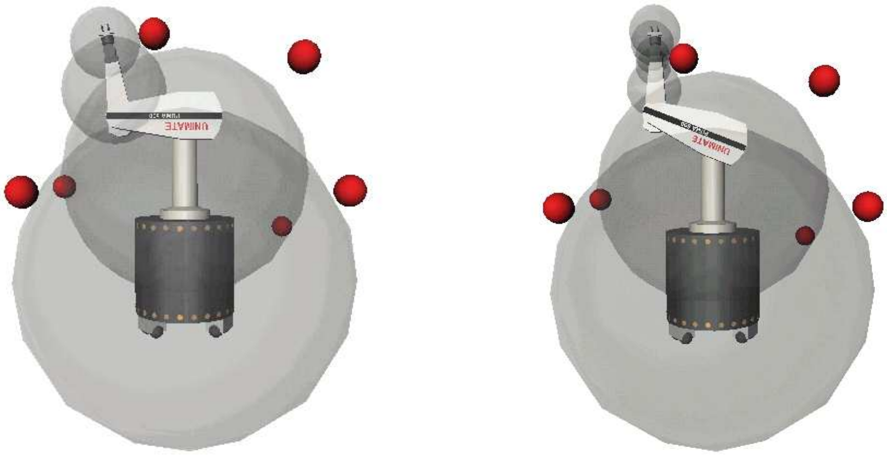
Figure 8.4: Protective hulls of a rigid body: If an obstacle is close to the rigid body, the protective hull needs to consist of many bubbles, as shown in a). If the free space around a body is large, a single bubble might suffice; this case is illustrated in part b) of this figure.

multiple rigid bodies or even the entire robot, implying that for large clearances a simple description of the local free space suffices. Figure 8.5 b) illustrates the effect on the protective hull as the end-effector approaches an obstacle: more bubbles with smaller radii are needed to verify that the robot lies completely in free space.

8.3 Covering Criterion

Given a set of bubbles $\{\mathcal{B}_1, \cdots, \mathcal{B}_n\}$ centered on the spine of a rigid body, it remains to be determined whether those bubbles form a protective hull or not, i.e. whether the rigid body is contained within their volumetric union or not. This is determined by the *covering criterion*. It verifies that the width of the union of bubbles along the spine exceeds the width of the hull at every point in order to guarantee that there is no collision, in other words that the protective hull *covers* the hull of the rigid body.

Please recall the schematic illustration of the hull of a rigid body in Figure 8.1 on page 130. To verify that a set of bubbles covers the hull two cases need to be distinguished:



a) b)

Figure 8.5: Illustrations of protective hulls: The small, opaque spheres represent obstacles. The radius of each bubbles is limited by such an obstacle. It can be seen that as a rigid body of the robot approaches an obstacle (b) more bubbles are required to describe the free space around that body.

1. The two bubbles closest to the end-points of the spine have to fulfill the following condition:
$$r_{\mathcal{B}} > r_e + ||c_{\mathcal{B}} - e||,$$
where $r_{\mathcal{B}}$ and $c_{\mathcal{B}}$ are the radius and the center point of the bubble and r_e is the radius of the spine at end-point e. This means that the radii of the first and the last bubble on the spine have to be large enough to contain the longest extension of the hull along the spine. If this condition holds, the end-portion of the spine beyond $c_{\mathcal{B}}$ is covered.

2. To simplify the computation for the remaining bubbles, constraints are imposed on the placement of bubbles on the spine. For adjacent bubbles $\mathcal{B}_1$ and $\mathcal{B}_2$ with

centers c_1 and c_2 and radii r_1 and r_2 three inequalities have to hold:

$$\begin{array}{ll} \text{(a)} & ||c_1 - c_2|| < r_1 + r_2 \\ \text{(b)} & ||c_1 - c_2|| + r_2 > r_1 \\ \text{(c)} & ||c_1 - c_2|| + r_1 > r_2 \end{array}$$

The first inequality guarantees that the spheres intersect and the following two ensure that a true intersection is present and one bubble is not contained within the other.

Due to the fact that both, the bubbles and the hull, are symmetric with respect to revolution about the axis described by the spine, we are able to perform the computations described below after projection into a plane containing the spine. We need to compute the width of the intersection of two circles C_1 and C_2 with centers c_1 and c_2 and radii r_1 and r_2. Referring to Figure 8.6 and using simple geometry the width w of the intersection can be computed:

$$a^2 + w^2 = r_1^2$$

$$(d - a)^2 + w^2 = r_2^2$$

can be solved for

$$a = \frac{d^2 + r_1^2 - r_2^2}{2d}$$

$$w = \sqrt{r_1^2 - a^2}.$$

The point c on the spine at which the protective hull has width w is computed by

$$c = c_1 + \frac{(c_2 - c_1)a}{d}.$$

If r_1, w, and r_2 exceed the width of the hull at points c_1, c, and c_2, respectively, the portion of the hull between c_1 and c_2 is covered.

These two cases can be used to verify in a piecewise manner whether a set of bubbles covers the hull or not and therefore constitutes a protective hull or not.

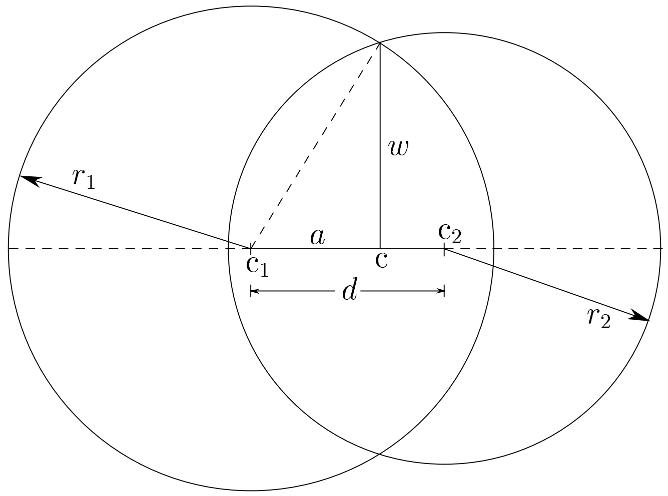


Figure 8.6: The intersection of two spheres is a circle. The computation of the center and radius of this circle is simple, after projecting the spheres into a plane through the two centers of the spheres. The circle lies in an orthogonal plane through point $\mathbf{c}$ and has radius w.

A protective hull $\mathcal{P}_b = \{\mathcal{B}_1, \cdots, \mathcal{B}_n\}$ is guaranteed to cover the hull of a rigid body b, but it is not necessarily a good representation of free space. If Figure 8.7a) the protective hull of a rigid body consists of two bubbles. The width of the resulting free space for the body is indicated by the arrows. By adding a bubble, as shown in Figure 8.7b) the width of the actual free space around the body can be represented more accurately. As a consequence of this observation, two adjacent bubbles of the protective hull are positioned such that the ratio between the width of the free space at the intersection and at the centers of the adjacent two bubbles remains close to 1; this guarantees that the intersection between two bubbles cannot be much narrower than the radius of the smaller bubble.

This heuristic and the above covering criterion lead to a recursive procedure to determine the protective hull for a given hull:

1. The endpoints of the spine are covered with bubbles. If this is not possible no protective hull can be constructed.

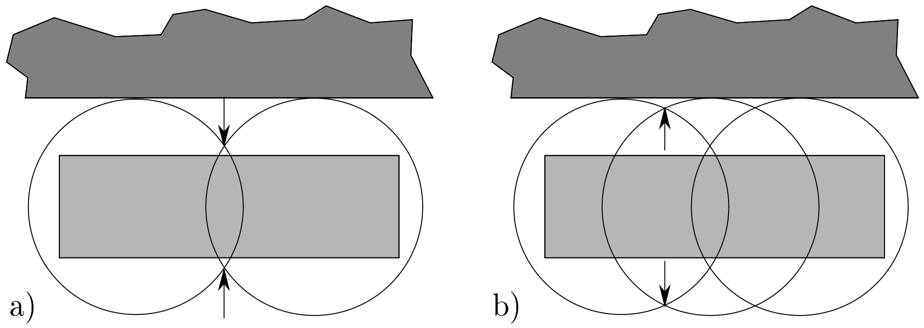


Figure 8.7: The accuracy of the free space description around a rigid body depends on the number of bubbles used to cover it. As illustrated in part a) of this figure, few bubble might cover the body but fail to describe the free space. Adding additional bubbles, as shown in b), a more accurate representation can be achieved.

2. The intersection of the two adjacent bubbles is computed.

3. Three cases are distinguished:

   (a) If the bubbles do not intersect a bubble is placed on the spine in the middle of the space separating the two bubbles. The two new adjacency relations result. For both of them the algorithm is invoked recursively at step 2.

   (b) If the ratio of the width of the intersection and the smaller radius and is smaller than $1 - \epsilon$, the body is covered, but the description of free space is not accurate enough. The algorithm is invoked recursively at step 2.

   (c) If the ratio of the width of the intersection and the smaller radius and is larger than $1 - \epsilon$ we are done.

The parameter ϵ determines how closely bubbles will be spaced along the spine. A smaller ϵ will result in a more accurate description of free space that requires more bubbles.

8.4 Elastic Tunnel

During the execution of a trajectory a robot sweeps out a volume in the workspace. The trajectory is free of collisions if this volume does not contain any obstacles. To warrant collision avoidance, the volume has to be computed using a model of the rigid bodies in motion and checked against obstacles in the environment.

Along the trajectory $\mathcal{U}$ a sequence of configurations $q_1, q_2, \cdots, q_n$ is chosen. This sequence is called an elastic strip $\mathcal{S}_{\mathcal{U}}^{\mathcal{R}}$ if the union of the protective hulls $\mathcal{P}_i^{\mathcal{R}}$ of the configurations $q_i, 1 \leq i \leq n$ fulfills the condition

$$\mathcal{V}_{\mathcal{U}}^{\mathcal{R}} \subseteq \mathcal{T}_{\mathcal{S}}^{\mathcal{R}} = \bigcup_{1 \leq i \leq n} \mathcal{P}_i^{\mathcal{R}}, \tag{8.1}$$

where $V_{\mathcal{U}}^{\mathcal{R}}$ denotes the workspace volume swept out by the robot $\mathcal{R}$ along the trajectory $\mathcal{U}$. If the condition of feasibility 8.1 holds, the union $\mathcal{T}_{\mathcal{S}}^{\mathcal{R}}$ of protective hulls is called *elastic tunnel*. It represents the local free space around the entire trajectory. Any point in the workspace that lies within the tunnel is guaranteed to be free of collisions by the protective hulls. As long as the volume $V_{\mathcal{U}}^{\mathcal{R}}$ swept by the robot remains within that tunnel, the robot will not collide with obstacles. An example of an elastic tunnel is shown in Figure 8.8. Three configurations represent snapshots of the robot's motion along the trajectory. The union of the protective hulls around those configurations form an elastic tunnel. It contains the volume swept by the robot along the trajectory. Another elastic tunnel is shown in Figure 8.9; here, an obstacle reduces the size of the tunnel. The robot's trajectory is modified to ensure that the volume swept by the robot along the trajectory is contained within the elastic tunnel.

This representation of free space is the key to the performance of the elastic strip framework. It can be computed very efficiently, while giving a good approximation of the the actual free space.

Figure 8.8: Along a trajectory multiple intermediate configurations are chosen such that the volume swept by the robot is contained within the union of the protective hulls associated with those configurations. The union of the protective hulls is called *elastic tunnel*.

8.5 Connectivity Criterion

An elastic strip $\mathcal{S}_{\mathcal{U}}^{\mathcal{R}}$ is a sequence of configurations $q_1, \cdots, q_n$. Associated with those configurations are the corresponding protective hulls $\mathcal{P}_1^{\mathcal{R}}, \cdots, \mathcal{P}_n^{\mathcal{R}}$. Each protective hull $\mathcal{P}_i^{\mathcal{R}}$ represents the local free space around configuration q_i along the trajectory $\mathcal{U}$ of the robot $\mathcal{R}$. Since each configuration q_i is guaranteed to be free of collisions by the protective hull $\mathcal{P}_{q_i}^{\mathcal{R}}$, it remains to be shown that the union of all protective hulls contains the volume $\mathcal{V}_{\mathcal{U}}^{\mathcal{R}}$ swept by the robot along the trajectory. The condition of feasibility of trajectory $\mathcal{U}$ given by $\mathcal{S}_{\mathcal{U}}^{\mathcal{R}}$ is

$$\mathcal{V}_{\mathcal{U}}^{\mathcal{R}} \subseteq \mathcal{T}_{\mathcal{S}}^{\mathcal{R}} = \bigcup_{1 \leq i \leq n} \mathcal{P}_i^{\mathcal{R}}. \tag{8.2}$$

It is sufficient to describe a procedure that verifies the existence of a path between

Figure 8.9: As the spherical obstacle approaches the elastic strip, the associated elastic tunnel deforms. The trajectory of the robot is modified to avoid the obstacle.

two consecutive configurations q_i and q_{i+1} inside the volume of the corresponding protective hulls $\mathcal{P}_i^{\mathcal{R}}$ and $\mathcal{P}_{i+1}^{\mathcal{R}}$. By applying this procedure repeatedly to adjacent configurations along the entire elastic strip, the condition of feasibility (8.2) can be ensured. This procedure is called *connectivity criterion* and will be described in this section.

We will make the simplifying assumption that every point on a rigid body b moves on a straight line as b transitions from q_i to q_{i+1}. This ignores the effect of rotation. This effect can be bounded and taken into account at a computational expense, when computing the protective hull of b. The justification for this assumption is that two adjacent configurations will be similar enough for this effect to be insignificant when the robot is close to an obstacle. This is a simplification but not an inherent limitation of the approach. Using this assumption the path of each rigid body b can be examined independently. If a trajectory between q_i and q_{i+1} exists for all rigid bodies $b \in \mathcal{R}$, one exists for $\mathcal{R}$.

The existence of a trajectory $\mathcal{U}_{i,i+1}$ for a rigid body b from configuration q_i to q_{i+1} is guaranteed if the volume $V^b_{\mathcal{U}_{i,i+1}}$ swept by b along $\mathcal{U}^b_{i,i+1}$ is contained within the protective hulls of the configurations q_i and q_{i+1},

$$V^b_{\mathcal{U}_{i,i+1}} \subseteq \left(\mathcal{P}^b_i \cup \mathcal{P}^b_{i+1}\right). \tag{8.3}$$

Let the protective hulls $\mathcal{P}^b_i$ and $\mathcal{P}^b_{i+1}$ be the sets of workspace bubbles $\{s_1, \cdots, s_m\}$ and $\{t_1, \cdots, t_n\}$, respectively. To verify condition (8.3), the union $U = \mathcal{P}^b_i \cup \mathcal{P}^b_{i+1}$ is examined. If b can pass through U on a straight line trajectory form q_i to q_{i+1} then $V^b_{\mathcal{U}_{i,i+1}}$ is contained within $\mathcal{P}^b_i \cup \mathcal{P}^b_{i+1}$. Hence, condition (8.3) holds and $\mathcal{U}_{i,i+1}$ must be collision-free.

Due to the convexity of bubbles, the locally narrowest passage along the trajectory is given by the intersection[1] I of three bubbles; see Figure 8.10. The intersection of the boundary of two bubbles, one from each spine, defines a circle. Intersecting that circle with the boundary of a third sphere results in two points. These points define a line segment. The length of this line segment can be checked against the width of the spine at the point that will pass through this passage. Repeating this procedure for all intersections of three bubbles verifies if the rigid body b can move from q_i to q_{i+1}, given the local free space $\mathcal{P}^b_i \cup \mathcal{P}^b_{i+1}$. If the widest point of a bubbles s_i with respect to p is contained within another bubble s_j, the narrowest passage is as wide as the radius of s_i and there is a trivial solution.

To verify that the entire rigid body b can move through the intersection of the two protective hulls, the intersections of all triplets of adjacent, intersecting bubbles need to be examined. Starting at an endpoint, both spines are traversed and the bubbles along them are intersected to determine the width of the intersection. Figure 8.11

[1]For the ease of presentation, the intersection of any two bubbles s_1 and s_2 is assumed to be non-empty and a true subset of each of the two bubbles: $I = s_1 \cup s_2 \neq \emptyset$, $I \subset s_1$, $I \subset s_2$. If the intersection is empty, no path is possible and if one bubble contains the other, a path must exist.

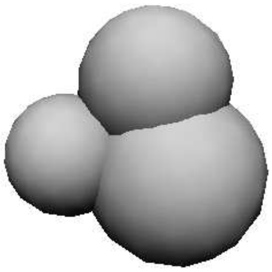


Figure 8.10: Consider the plane described by the centers of three bubbles. The height of the space described by those bubbles inside the triangle formed by the centers is smallest at the intersection of the three bubbles.

shows two pairs of spines to illustrate this traversal. In both parts of that figure the rigid body, shown as a dotted line, is supposed to move from the left to the right spine. The protective hulls of the spines are shown, in solid for the first configuration of the rigid body and dotted for the second. If for the intersection of all triplets is wider than the rigid body at the corresponding position, we say that the two protective hulls associated with the spines are *connected*.

The traversal of the bubbles on the spine proceeds as follows. Given a point on the spine, its motion can either lead to a bubble on the next spine, as shown in Figure 8.11 a), or to an adjacent bubble on the same spine, as shown in Figure 8.11 b). In the latter case, the motion can either be completed within the free space of the first spine, or a situation arises in which the former case applies. The computations involved in the latter case are simple and are omitted here. We will focus on the situation where motion from one protective hull to the next occurs.

Please refer to Figure 8.11 a). Assume that we are given two bubbles on different spines that intersect. If a rigid body is to move from the configuration described by one spine to the configuration described by the other spine without leaving the free space represented by the union of their protective hulls, a third bubble must exist on either spine that intersects both of the previous bubbles. This simple geometrical insight implies that if the sequential traversal of the bubbles along the spines fails,

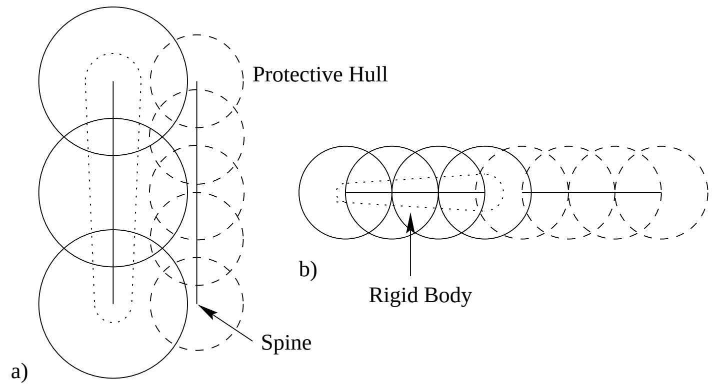


Figure 8.11: Two adjacent spines are traversed to compute the intersection of all sets of three intersecting bubbles. Depending on how these spines are aligned different intersection patterns arise.

the rigid body cannot pass through the union of the two corresponding protective hulls.

For the ease of presentation, the following description of the connectivity criterion ignores the case where the rigid body moves between bubbles on the same spine. This situation represents a special case and the algorithm can easily be extended to handle it. Given a situation as shown in Figure 8.11 a), the connectivity criterion is computed as follows:

1. Verify that the two bubble at one end of the spines intersect and that the width exceeds the width of the hull at that end point. If this fails, the protective hulls are not connected.

2. Select the next bubble from either spine that intersects the two existing ones and verify that the rigid body at the position that passed through the intersection is narrower than the intersection itself.

3. If this fails, we attempt to select the next bubble on the other spine. If that also fails the spines are not connected.

4. If the spines have not been traversed yet, discard the first bubble on the spine contributing two bubbles to the intersection and go to step 2, otherwise the spines are connected.

If for all rigid bodies $b \in \mathcal{R}$ the union of their protective hulls $\mathcal{P}_i^b \cup \mathcal{P}_{i+1}^b$ is large enough to allow a straight-line trajectory, we say that two consecutive protective hulls $\mathcal{P}_i^{\mathcal{R}}$ and $\mathcal{P}_{i+1}^{\mathcal{R}}$ are *connected*.

8.6 Distance Computation

The most costly operation during free space computation and modification of the elastic strip is the distance computation. For every bubble in every protective hull for every configuration along the strip several distance computations have to be performed. As a consequence, the efficiency of this part of the system has a large impact on the overall performance.

Many different approaches to distance computation and collision detection have been presented in the literature (Gilbert et al. 1988; Lin 1993; Quinlan 1994a; Gottschalk et al. 1996; Hubbard 1996; Cameron 1997; Mirtich 1997; Ong and Gilbert 1997; Xavier 1997). We chose the hierarchical bounding representation (Quinlan 1994a). A rigid body is represented by a binary tree in which each node represents a sphere. This tree has the following properties:

1. The root sphere completely contains the object.

2. The leaves of the tree are obtained by covering the boundary of the object with overlapping spheres.

3. Each internal node of the tree contains all parts of the rigid body that are contained within its children.

Distance computations can be performed efficiently by traversing this tree of spheres from root to leaf. Then, the computational complexity of a distance computation is determined by the tree traversal, which is $O(\log n)$, where n denotes the number of spheres necessary to cover the boundary of the rigid bodies. For the details of the algorithm the interested reader is referred to Quinlan 1994a.

Chapter 9

Experimentation

This chapter presents an implementation of the elastic strip framework, experimentally validating its applicability to motion generation for robots with many degrees of freedom in dynamic environments. The elastic strip framework was applied to the Stanford Mobile Manipulator, shown in Figure 9.1 (see also Appendix A), a six degree of freedom PUMA 560 manipulator arm mounted on a holonomic mobile base. It is equipped with 48 sonar and infra-red sensors, two on-board PCs, a radio-Ethernet connection, and battery power for many hours of untethered operation (Khatib et al. 1995; Khatib et al. 1996).

One of the challenges in applying the elastic strip framework is the conversion of the path represented by the configurations along the elastic strip into a time-parameterized trajectory. Since the elastic strip is constantly modified in reaction to changes in the environment, the trajectory needs to be re-parameterized continuously. This chapter introduces a novel approach to trajectory generation for changing paths. This new method is then applied in conjunction with the elastic strip framework to motion execution in dynamic environments.

9.1 Trajectory Generation and Execution

An elastic strip $\mathcal{S}$ is represented by a sequence of discrete configurations $q_0, \ldots, q_n$ along a path from q_0 to q_n. The conversion of such a sequence into a time-parameterized

Figure 9.1: The Stanford Mobile Manipulator consists of a PUMA 560 manipulator arm with six degrees of freedom mounted on a holonomic mobile base.

trajectory is a well-studied problem (Bobrow, Dubowsky, and Gibson 1985; Craig 1989). In the elastic strip framework, however, this sequence of configurations is changing in discrete steps as the elastic strip is modified in reaction to changes in the environment. These changes can render the state of the robot inconsistent with the state represented by elastic strip.

Let q_t and $\dot{q}_t$ be the vectors of joint positions and velocities of the robot at time t. They resulted from the execution of the trajectory described by the elastic strip $\mathcal{S}_{t-1}$ at time $t-1$. The modified elastic strip $\mathcal{S}_t$ will represent a different trajectory with a desired position of q'_t and a desired velocity of $\dot{q}'_t$ at time t and a potentially unattainable desired position q'_{t+1} and velocity $\dot{q}'_{t+1}$ at time $t+1$. Due to the rate at which the update of the elastic strip occurs, the difference between the current and desired position and velocity, $\|q_t - q'_t\|$ and $\|\dot{q}_t - \dot{q}'_t\|$, can be large. Hence, the application of conventional approaches to trajectory execution would not result in desirable behavior.

This problem, first recognized in the elastic band framework (Quinlan 1994b), may be solved by either constraining the initial portion of the elastic strip to remain constant or by limiting modification to those trajectories achievable, given the actuation capabilities of the robot.

These solutions have two disadvantages: For one, invalidating the path represented by the elastic strip requires a computationally expensive replanning operation, which should be avoided if at all possible. Hence, it is desirable for the elastic strip to represent a valid path, even if inconsistent with the current state of the robot. The local replanning problem of resolving this inconsistency is far simpler than a global planning operation. Secondly, when executing a motion on a mobile base, such as the Stanford Mobile Manipulator, execution errors accumulate for the mobile base due to slippage of the wheels. To reduce this error, various relocalization schemes can be employed. The error accumulated between different relocalizations can be expected to be large enough to invalidate a trajectory originally incorporating dynamic constraints imposed by the robot. Insufficient actuator capabilities could also be regarded as a source of error. Relocalization would then require a discrete trajectory change in any case. It would be, therefore, advisable to allow this inconsistency between the real robot and the elastic strip itself, rather than to invalidate the path.

To avoid these disadvantages, a novel approach to the execution of changing trajectories is presented in this section. The general idea is to maintain the elastic strip such that it always corresponds to a valid trajectory, ignoring the current state of the actual robot. The executable trajectory then results from merging the robot's current motion with the previously computed trajectory. These two steps are detailed in the two following subsections for a single degree of freedom. The extension to many degrees of freedom is trivial, as all joints can be treated independently.

9.1.1 Generating the Trajectory

Please note that so far most representations were workspace-based; the trajectory, on the other hand, will be specified in configuration space. The elastic strip represents a sequence of configurations $q_1, \ldots, q_n$ that are connected by straight-line segments in joint space. The discontinuous velocity change that can occur at a configuration q_i along the piecewise linear trajectory cannot be executed by the robot without coming to rest at q_i. As this is not desirable, an interpolation technique is applied to convert the piecewise linear trajectory into one with a continuous first derivative. As we can expect frequent velocity changes during the execution of a changing trajectory, this interpolation is performed with cubic polynomials, which also guarantee a continuous second derivative within a given segment of the trajectory.

When using a standard scheme for trajectory generation with cubic polynomials (Craig 1989), the trajectory passes through a set of via points, corresponding to the configurations q_i along the elastic strip. This may result in a large deviation from the straight-line trajectory between two adjacent configurations. Due to the free space description with protective hulls, however, this is the portion of the trajectory where the known free space is most narrow. It is hence desirable for the robot to follow this portion of the trajectory as closely as possible. To achieve this, the straight-line segments will be connected by *cubic turns*. The maximum allowed duration of a turn at configuration q_i can be inferred from the local free space and, along with actuator constraints, determines the velocity along the adjacent line segments from q_{i-1} to q_i to q_{i+1}.

The process of transforming the elastic strip into a trajectory is illustrated in Figure 9.2 for a single joint. An elastic strip is represented by four configurations $q_1, \cdots, q_4$. The desired velocity along the linear segments connecting the configurations is given by the slope of the line. The turning cubics are indicated by the dashed lines. As the robot is expected to be at rest before and after execution of the trajectory, starting and stopping cubics are necessary; they are shown as dotted lines.

A cubic polynomial for trajectory generation describes the values of the joint variable q as a function of time:

$$q(t) = c_0 + c_1\, t + c_2\, t^2 + c_3\, t^3.$$

The parameters c_0 and c_1 correspond to the initial value of the joint variable q_0 and the initial joint velocity $\dot{q}_0$, respectively. The second derivative $\ddot{q}(t)$ of $q(t)$ describes

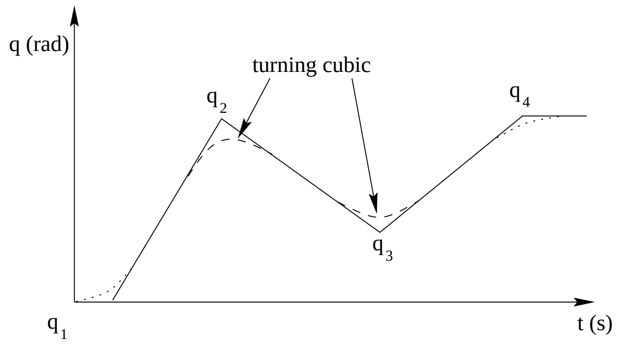


Figure 9.2: Cubic splines are used to connect the straight-line segments of the trajectory. They generate a smooth trajectory, interpolating joint values between consecutive configurations along the elastic strip.

a line. Hence, the acceleration $\ddot{q}(t)$ can be described by two parameters a_0 and a_1:

$$\ddot{q}(t) = a_1 + t\ a_2.$$

The parameters $q_0, \dot{q}_0, a_1$ and a_2 uniquely determine a cubic polynomial. The values of q_0 and $\dot{q}_0$ are given by the segment of the trajectory preceding the cubic and only a_1 and a_2 need to be determined. In the remainder of this section the process of determining a_1 and a_2 will be described.

To allow the execution of trajectories in tight spaces, a *turning cubic* will always begin by accelerating with the maximum acceleration $\ddot{q}_{max}$, given actuation limitations. Taking into account that the acceleration during a cubic can be described by a line, the duration d of the turning cubic is computed as follows:

$$\Delta\dot{q} = |\dot{q}_{i-1} - \dot{q}_i| = \int_0^d \ddot{q}(t)\ dt = \frac{d\ \ddot{q}_{max}}{2} \qquad \rightarrow \qquad d = \frac{2 \cdot |\Delta\dot{q}|}{\ddot{q}_{max}},$$

where $\dot{q}_i$ denotes the velocity along the line segment between q_i and q_{i+1}. The parameters for the line describing the acceleration for the duration of the turning cubic between joint values q_i and q_{i+1} can be computed as follows:

$$a_1 = \ddot{q}_{max} \cdot sign(q_{i+1} - qi)$$

$$a_2 = \frac{\ddot{q}_{max} \cdot (-sign(q_{i+1} - q_i))}{d},$$

where $sign(\cdot)$ is 1 for positive and -1 for negative arguments and d denotes the duration of the turning cubic; d remains to be computed.

The points at which the turning cubic connects with the adjacent line segments can be computed by equating the equations describing them:

$$q_i + (d_i - d_{\Delta i})\dot{q}_i + \Delta q = q_{i+1} + (d - d_{\Delta i})\dot{q}_{i+1},$$

where d_i denotes the duration of the line segment between q_i and q_{i+1} assuming constant velocity $\dot{q}_i$, and Δq stands for the change in joint position between beginning and ending of the turning cubic. Solving for $d_{\Delta i}$, the parameters of the turning cubic are computed. Its execution will begin at time $t_{i+1} - d_{\Delta i}$ and end at time $t_{i+1} + (d - d_{\Delta i})$, where t_i denotes the time at which execution of the ith segment begins. Using those values, a_2 can be determined and the starting and ending point of the turning cubic with respect to the line segments can be computed.

Having determined the parameters of the function describing the acceleration, the duration, and the starting and ending configuration of the cubic, the turning cubic is uniquely defined.

9.1.2 Merging the Robot's Motion with the Trajectory

The trajectory resulting from the method described in the previous subsection does not take into account the current velocity of the robot. Also, the robot's position might have changed due to the correction of accumulated execution error. To connect the robot to this trajectory, an iterative optimization algorithm could be used (Croft, Fenton, and Benhabib 1995). This algorithm retains the original time parameterization of the trajectory, which causes the robot to "catch up" with the trajectory.

In dynamic environments it is unreasonable to impose such a time constraint for the robot. Therefore, we will reconnect the robot's current state to the trajectory according to its dynamic capabilities and then adapt the time-parameterization of the remainder of the trajectory.

The cubic segment that merges the current state of the robot with a segment on the trajectory can be computed by equating the position and the velocity equations of the merging segment and the segment on the trajectory. When merging with a line segment the following equations result:

$$\begin{aligned} q_0 + d\,\dot{q}_0 &= q_r + \dot{q}_r\,d + a_1\,d^2 + a_2\,d^3 \\ \dot{q}_0 &= \dot{q}_r + 2\,a_1\,d + 3\,a_2\,d^2, \end{aligned}$$

where q_r and $\dot{q}_r$ are the position and the velocity of the robot, d is the duration of the *merging cubic*, and a_1, a_2 are its coefficients. These two equations can be solved symbolically for a_1 and a_2, the parameters determining the acceleration during the merging cubic. Since we know that at the beginning of the cubic the acceleration will be a_1, we can solve the symbolic equation for d, the duration of the cubic. Again, all parameters of the cubic have been derived and the cubic is determined uniquely.

When merging the current state of the robot with a cubic segment of the original trajectory, a solution can be found in a similar fashion. The duration d of the merger has to be determined such that the acceleration constraints of the robot are not violated.

9.2 Implementation

The Open Inventor software package (Wernecke 1994a; Wernecke 1994b) was chosen as the underlying modeling tool. It was extended to support a robot description language based on rigid bodies, revolute and prismatic joints. These extensions allows to define a simulated or model a real environment very easily. The environment consists of rigid or articulated bodies. A designated articulated body, called robot, is controlled using the elastic strip approach. The implementation allows for motion execution in simulated and in real environments.

During simulation the obstacles in the environment move on predetermined trajectories, specified in a scripting language. This allows to generate environments with complex dynamic interactions between obstacles. When controlling a real robot, the motion of the obstacles is perceived through sensing and the world model is updated accordingly. When conducting experiments in the physical world, rather than in simulation, the motion of obstacles needs to be monitored (see Section 7.9). To facilitate this task, only robots were allowed as moving obstacles for the experiments presented here. The position of the obstacle-robot could then be determined using the position information of the robot. The joint-level control of the real robot was performed using the *robotics library* (Chang 1998; Chang 1999).

9.3 Experimental Results

The elastic strip framework was implemented and tested on the Stanford Mobile Manipulator, a 9 degree-of-freedom robotic system, consisting of a PUMA 560 arm mounted on a holonomic mobile base (Brock and Khatib 1998b; Brock and Khatib 1998c; Brock and Khatib 1998a; Brock and Khatib 1999a; Khatib, Yokoi, Brock, Chang, and Casal 1999). We also applied the elastic framework to the simulation of a multi-robot workcell, consisting of two Mitsubishi PA-10 manipulator arms (Brock and Khatib 1999c). The various experiments conducted are presented in this section. For experiments involving real robots, the issue of sensing (see Section 7.9) was addressed in a very simplistic way: all moving obstacles in the environments are robots. Their encoder values and therefore also their current configuration is broadcasted so that the configurations of all robots are known at any time.

Figure 9.3 shows three snapshots of the elastic strip framework applied to the Stanford Mobile Manipulator. The initial and the final configuration along the presented path are indicated by the robots. The elastic strip is represented by the *frame* of the robot, consisting of line segments, connecting the origins of the joint frames. The images illustrate the modification of the elastic strip while an obstacle moves into the robot's trajectory. In the first snapshot the elastic strip is unaffected by the obstacle; internal forces cause it to be a straight line without posture change of the robot arm. As the obstacle moves closer to the trajectory, repulsive forces cause the elastic strip to deform in order to avoid the obstacle, shown in the second and third snapshot. The trajectory of the base, as well as the posture of the arm are modified to maintain a collision-free trajectory. After removal of the obstacle the elastic strip will again assume the original shape.

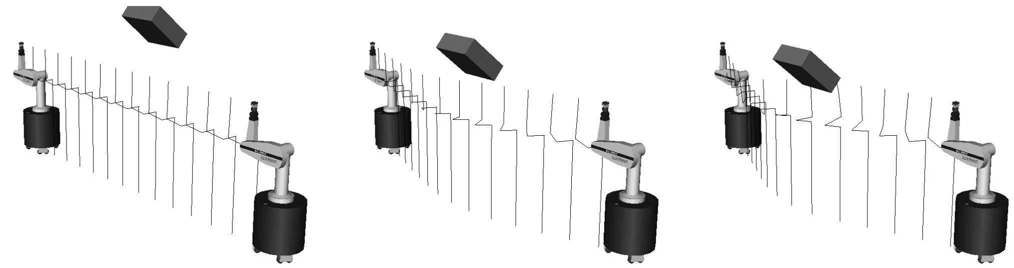


Figure 9.3: Initially, the elastic strip represents a straight-line trajectory between two configurations (left). As the obstacle moves into the path, the trajectory is modified to avoid a collision. This modification alters the trajectory of the arm and the base.

Figure 9.4 shows four snapshots of the elastic strip during execution of a motion. The first image shows the path that resulted from the planner. It avoids the two static obstacles by moving the base along an S-shaped path; the posture of the arm remains unchanged along the path. The second image shows the elastic strip after being shortened and smoothed by application of internal forces. Finally, the next two images show the modification of the elastic strip as a cylindrical obstacle moves into the path of the end-effector from above. The posture of the manipulator arm changes to avoid the obstacle.

In Figure 9.5 the application of the elastic strip framework to a multi-robot workcell is shown. The workcell consists of two Mitsubishi PA-10 robotic arms on gantries with nine degrees of freedom each, sharing the same workspace. The gantries are not explicitly shown; they enable the arms to move in the plane of the floor of the assembly cell. The gantry increases the redundancy of the robot with respect to the task

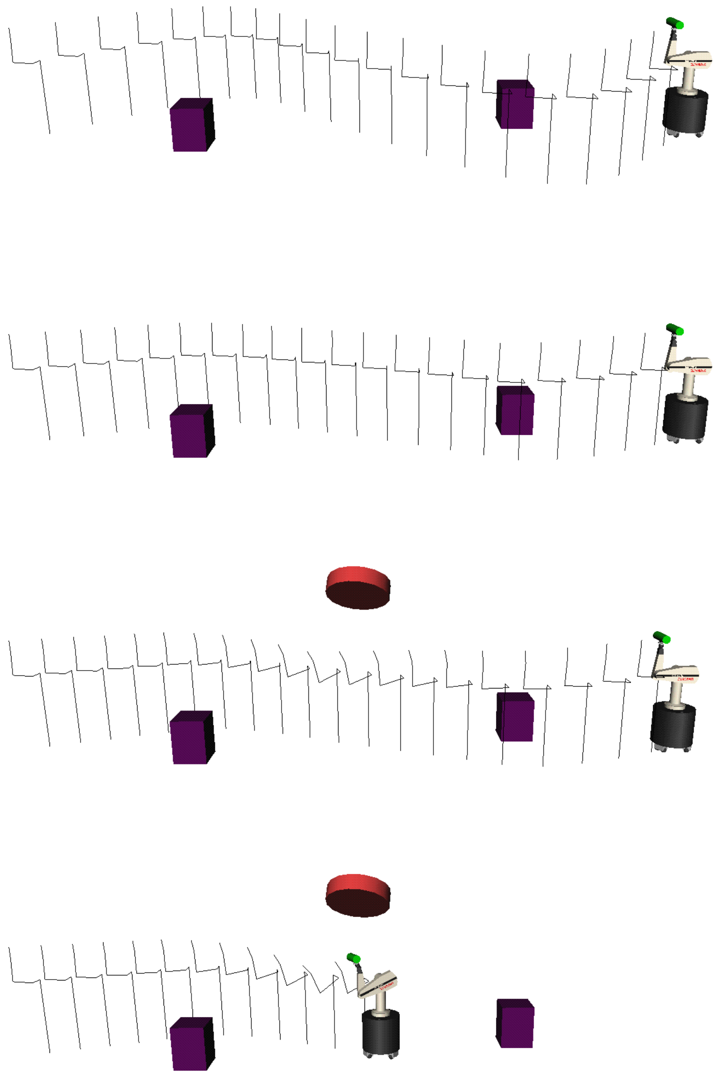


Figure 9.4: The first image shows the trajectory as resulting from a planning operation. Subsequently, internal forces are applied to shorten and smoothen the trajectory. An obstacle moves into the path of the arm and the arm's posture is modified to avoid collision. Finally, the motion is executed while avoiding the collision in real-time.

and hence allows better obstacle avoidance behavior. The elastic strip framework can equally well be applied to stationary manipulators.

The experiment in the multi-robot workcell, depicted in Figure 9.5, shows the incremental modification of the trajectory of a robot to avoid a collision with another manipulator moving into its path. The configurations along the trajectory represented by the elastic strip are again indicated by lines connecting the joint frames of the robot. Notice how more intermediate configurations are used to represent the trajectory where the robots are closer to each other. This is due to the decreasing size of protective hulls. The first image in Figure 9.5 shows the initial configuration of both robots. The next two images in Figure 9.5 show the modification of the elastic strip as a reaction to the robot on the right side of the figure raising its end-effector; the two configurations show the first robot just before and after passing the rising arm. The last image depicts the robot in the goal position defined by the elastic strip. The base of the obstacle-robot remains stationary throughout the entire experiment.

To demonstrate the elastic strip framework in more complex environments with many moving obstacles, the experiment shown in Figure 9.6 has been conducted. The trajectory of the Stanford Mobile Manipulator is represented as an elastic strip, indicated by traces of the origins of joint frames of the robot along the strip. Four Mitsubishi PA-10 arms moving on gantries are approaching and deforming the elastic strip. The rate of modification of the elastic strip decreases as the complexity of the distance computation operations increases. Exploiting spatial coherency this performance degradation can be reduced. Assuming constant-time distance computation, there is no inherent performance loss with the elastic strip framework, due to the complexity of the environment.

The representation of the elastic strip by intermediate configurations, as shown in most figures of this section, seems to indicate discontinuous velocity changes. Please keep in mind that for all the experiments presented in this section the trajectory computed from the elastic strip is smoothened before execution, according to the procedure described in Section 9.1.

In another experiment, the base of the Stanford Mobile Manipulator was commanded to move in a straight line along the x-axis of the global coordinate frame,

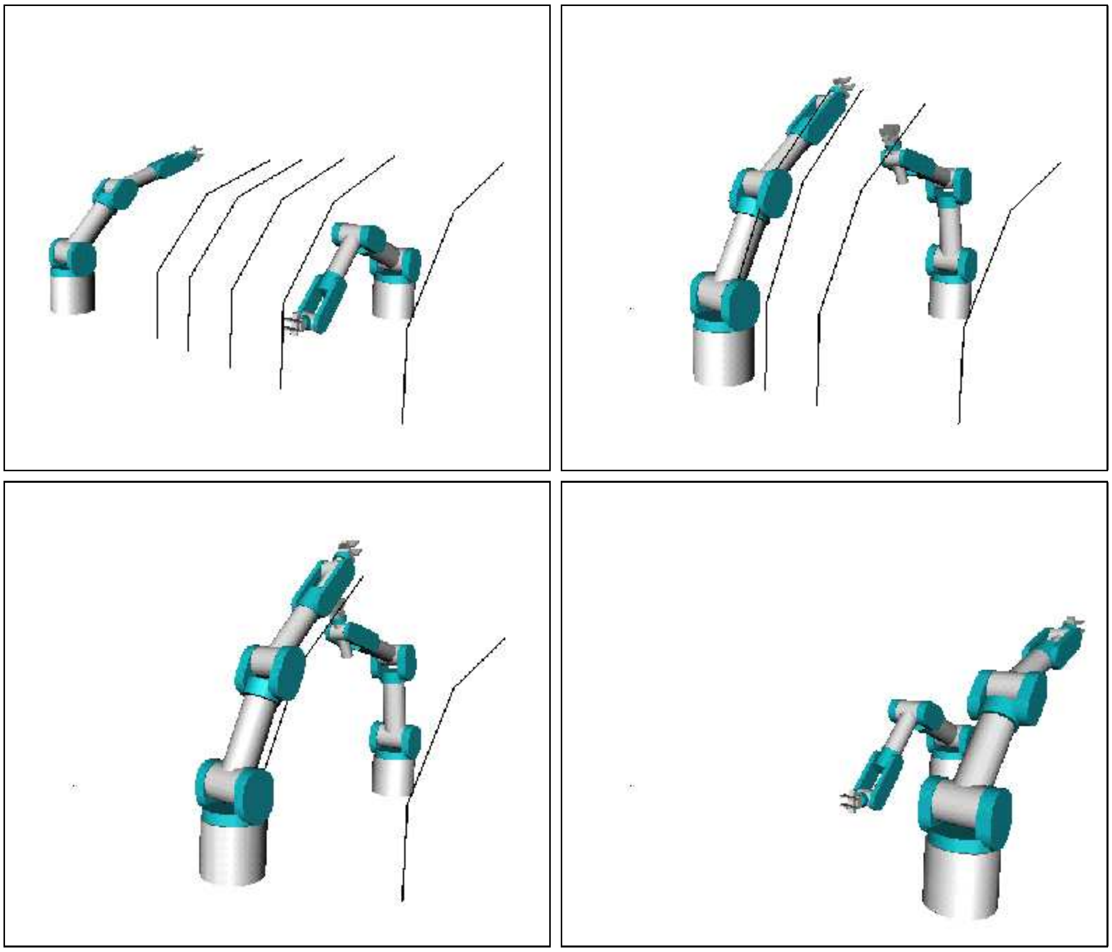


Figure 9.5: Two Mitsubishi PA-10 arms share the same work space in an assembly cell. One of the robots is mounted on a gantry that is not shown in the image. As it executes its trajectory, indicated by intermediate configurations, the second robots moves into its path. The collision is avoided by real-time path modification using elastic strips.

while maintaining the arm's posture. During the execution of this plan an unforeseen obstacle, another Stanford Mobile Manipulator, forces the first robot to deviate from the original plan. The trajectory modified using the elastic strip framework to avoid collision while achieving the desired goal configuration. Two different perspectives of the simulated modification of the trajectory are shown in Figure 9.7. A sequence of snapshots from the execution on the real robot can be seen in Figure 9.8. The plot of the base trajectory, as well as the trajectory for the first three joints of the PUMA 560 manipulator are shown in Figure 9.9 and 9.10. It can be seen that the elastic strip framework in conjunction with the trajectory execution scheme presented in Section 9.1 results in smooth trajectories, even when the path is modified during execution.

For all experiments presented above the elastic strip framework achieved real-time performance. Running on a 400 MHz PentiumII Personal Computer, update rates of the elastic strip vary between 5 Hz and 100 Hz. Experimentally, it can be observed that the servo rate decreases with the number of configurations represented along the elastic strip. To compute the protective hull for each of these configurations several distance computations are necessary. Obviously, as the number of configurations increases, more distance computations are required. As a result, the servo rate decreases with increasing length of the elastic strip and with decreasing clearance to obstacles. Even in tight and complex environment an update rate of 5 Hz could be achieved. The computational complexity of the elastic strip framework is analyzed more formally in Chapter 10. The experiments show, however, that the elastic strip framework yields real-time performance in realistic settings.

9.4 Conclusion

The elastic strip framework is an efficient method for motion execution for robots with many degrees of freedom in dynamic environments. It allows real-time obstacle avoidance behavior without the suspension of task execution. The effectiveness of the proposed algorithm is derived from an efficient integration of planning and execution methods. The validity of the algorithm was experimentally verified using the Stanford Mobile Manipulator, a nine degree-of-freedom robotic system, consisting of a mobile base and a manipulator arm and a multi-robot workcell with two 9 degree-of-freedom Mitsubishi manipulator arms mounted on gantries.

The ideal system architecture for the integration of motion planning and motion execution algorithms was introduced in Section 2.4 and is depicted in Figure 9.11. This architecture is considered ideal, because each motion command is determined based on all the information available about the environment, including the robot itself. A new motion plan is created from scratch, each time a change in the environment is perceived. If this re-evaluation happens often enough and incorporates dynamic constraints, the robot is able to navigate safely in dynamic environments.

The global dynamic window approach, presented in the first part of this thesis, realized this ideal architecture for the navigation of a mobile base. It takes advantage of the low dimensionality of the configuration space for such a robot. Knowing that the computational complexity of planning increases exponentially with the number of dimensions of the configuration space (see also Chapter 10), we cannot hope to achieve the same result for robots with many degrees of freedom. The planning operation is simply too computationally demanding to be integrated into the servo loop that determines the motion commands.

Intuitively speaking, planning is computationally complex because it considers the entire environment for possible trajectories. And this is necessary because it cannot be known in advance through which region of the free space the resulting path will lead. Once a path has been determined, however, it suffices to monitor the local free space around that path to update it in accordance with changes in the environment. This is the approach taken by the elastic strip approach. Instead of taking all free space into account, like planning algorithms, or just taking a local portion of the free space into account, like motion execution algorithms, it takes into account the free space around the path that resulted from the planning algorithm.

The basic system architecture of this approach is illustrated in Figure 9.12. A comparison it with the ideal architecture in Figure 9.11 reveals that an additional level is introduced. This level contains the local model of free space around the path represented by the elastic strip. The elastic strip is modified in accordance with sensed changes in the environment. By updating the world model a new planning process can be initiated. Once a new plan has been determined, the elastic strip can be updated accordingly. During the planning operation, the elastic strip framework is applied to the previously computed path. This enables the robot to move on a path towards the goal, while avoiding obstacle in real-time. An update of the elastic strip by the global motion planner is considered replanning (see Section 7.7).

The introduction of this intermediate level in the control architecture allows the integration of motion planning and motion execution algorithms for robot with many degrees of freedom. The general planning problem is too computationally demanding to be integrated into the motion control loop. The elastic strip reduces the planning problem to a much simpler one, namely, maintaining a collision-free trajectory by only monitoring the free space around the current path. While greatly reducing the computational complexity with respect to global planning, the ability to react to global changes in the environment that are affecting the current path is maintained. Thus, elastic strips enable robots with many degrees of freedom to safely and robustly navigate in dynamic environments. Such conditions are frequently encountered for applications in service or field robotics, where robots have to operate in unpredictable and populated environments.

In Section 7.1 three main objectives for the development of the elastic strip framework were specified. As the results of this section indicate, the elastic strip framework achieves these objectives: The elastic strip framework is efficient for motion execution robots with many degrees of freedom in dynamic environments. It also allows intuitive task specification directly in the task space (see Section 7.2 on page 92). Furthermore, it integrates task execution with obstacle avoidance. Thus, compared to planning and execution algorithms operating alone, the elastic strip can significantly extend the motion capabilities of robots in dynamic environments by integrating motion planning and motion execution paradigms.

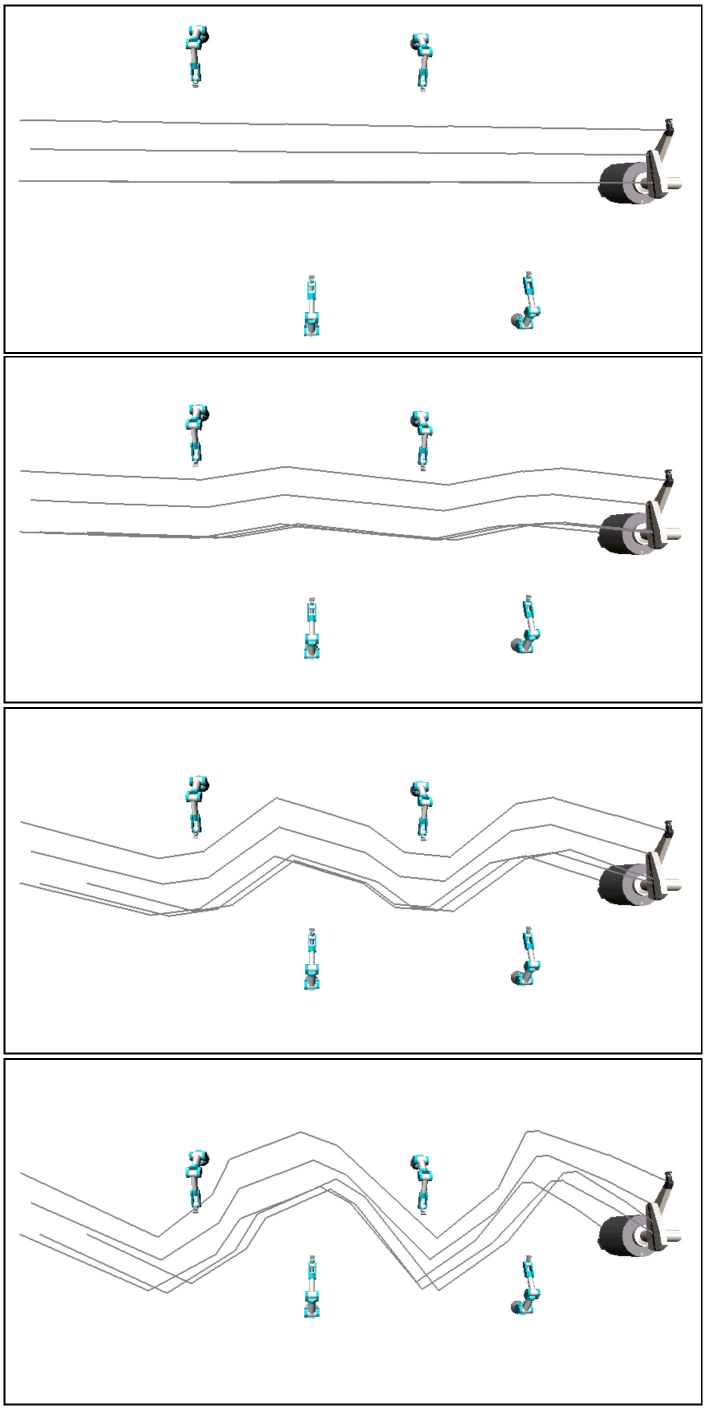


Figure 9.6: This experiment shows how an elastic strip is deformed in reaction to the motion of four robots, representing moving obstacles. The elastic strip is shown as line segments along the trajectory. During execution, the trajectory will be smoothened.

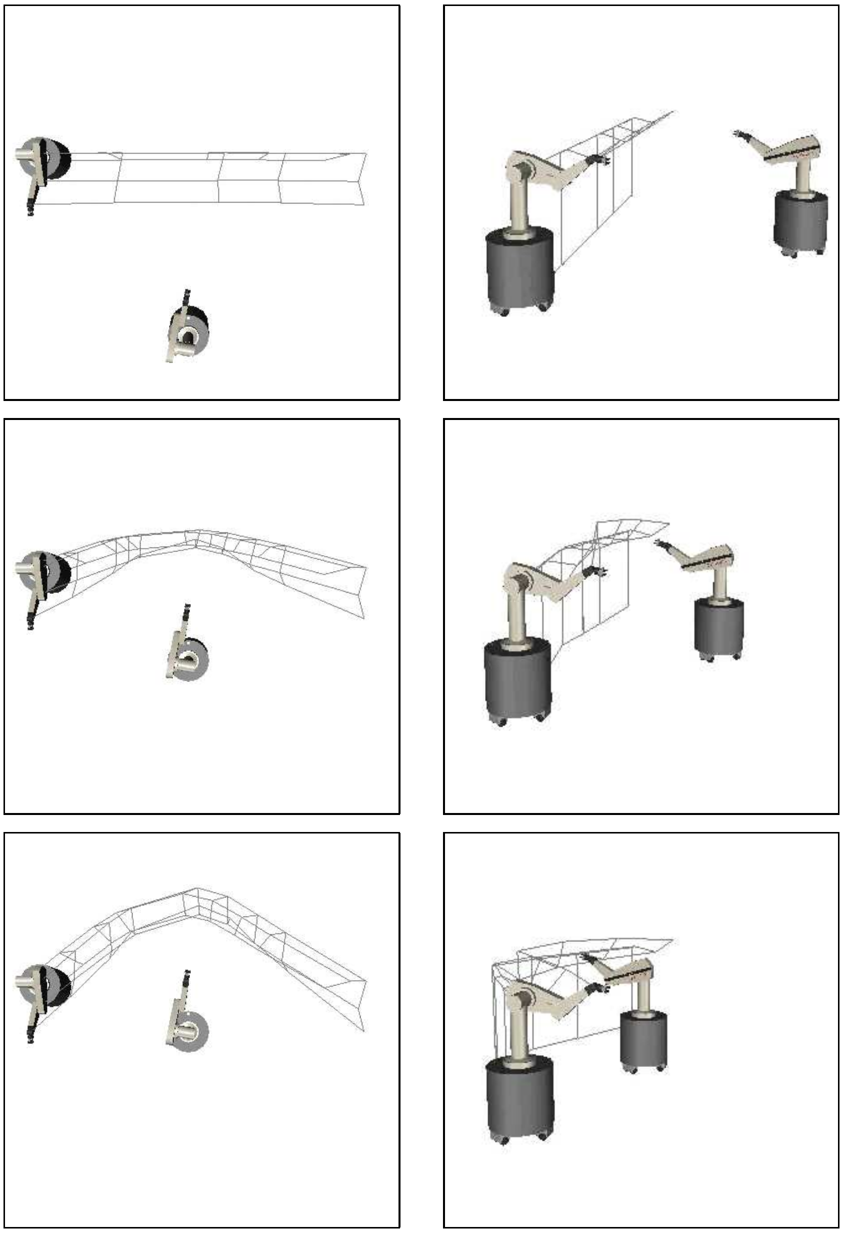


Figure 9.7: The elastic strip, shown as gray lines, is modified incrementally in order to maintain a valid path while avoiding a moving obstacle. Two different views of the same scene are shown for each of the three configurations.

Figure 9.8: Execution of a plan on a real robot using the elastic strip framework; the path is modified in real-time to avoid the obstacle. This image sequence corresponds to an execution of the example in Figure 9.7.

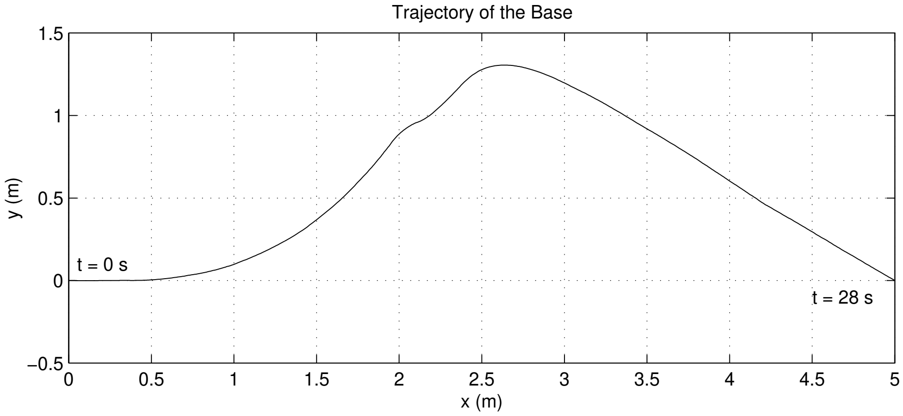


Figure 9.9: This graph shows the trajectory of the base in the x/y-plane for the experiment in Figure 9.8. The execution of the entire trajectory took 28 seconds.

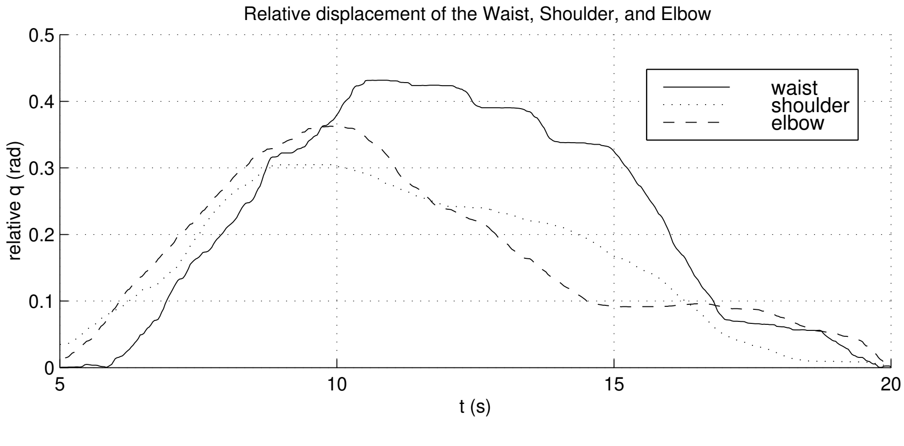


Figure 9.10: The relative displacement in radians of the waist, shoulder, and elbow of the PUMA 560 in the experiment from Figure 9.8 are shown. Note that only that portion of the trajectory is shown for which the arm deviates from its original posture.

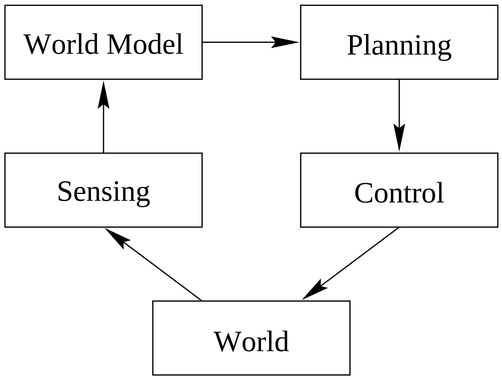


Figure 9.11: The ideal motion generation and execution architecture introduced in Section 2.4.

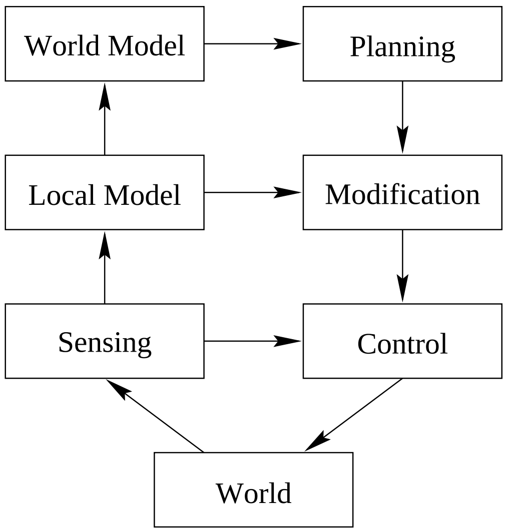


Figure 9.12: The motion planning and execution architecture with elastic strips introduces an intermediate representation between pure sensing and the world representation. It consists of the local free space around the trajectory. This intermediate representation allows for real-time path modification.

Chapter 10

Computational Complexity

This part of the thesis argues that reactive and incremental path modification with the elastic strip framework represents an efficient and effective solution for motion execution in dynamic, unstructured environments, because costly planning operations can be avoided in many situations. The experiments presented in Chapter 9 provide experimental validation. In this section arguments about the computational complexity of the elastic strip framework are laid out to confirm the above statement.

10.1 Motion Planning

The computational complexity of various instances of the motion planning problem has been examined in the literature (Reif 1979; Schwartz and Sharir 1983; Canny 1988; Latombe 1991). Three relevant results are given below:

Reif 1979: Planning a collision-free path for an articulated robot consisting of an arbitrary number rigid bodies among a finite set of polyhedral obstacles is a *PSPACE*-hard problem. This provides a lower bound (Hopcroft and Ullman 1979) on the complexity of motion planning in a polyhedral environment.

Schwartz and Sharir 1983: Given a configuration space of dimension n, where the free space is defined by m polynomial constraints of degree d, the computational complexity associated with the path planning problem is exponential in

n and polynomial in m and d. This results represents an upper bound to the complexity of motion planning.

Canny 1988: The complexity of the path planning problem in n dimensions using the silhouette method is singly-exponential in n. This method has the lowest computational complexity of planning methods devised today.

These results suggest that the complexity of motion planning increases exponentially with the dimension of the configuration space. This implies that as the number of degrees of freedom of a robotic system increases, the operation of planning a path for that system quickly becomes computationally infeasible. There are, however, various algorithms and heuristics that yield good results for practical problems in low-dimensional configuration spaces (Latombe 1991).

Realizing that the *explicit* representation of free space in high dimensions becomes increasingly computationally complex, probabilistic motion planning attempts represent the free space *implicitly* by randomized sampling (see Section 6.1.1). These algorithms perform very well, both in practice and when considering their computational complexity (Kavraki 1994). In particular in high-dimensional configuration spaces probabilistic methods perform much better than motion planning methods that rely on an explicit representation of the free space.

The computational complexity of planning algorithms that rely on an explicit representation of the free space is generally described in terms of the dimensionality of the configuration space and the number of features (vertices, edges, or faces) of the configuration space obstacles. This characterization of computational complexity can be considered *combinatorial*. Since probabilistic methods do not use an explicit representation of the configuration space obstacles, other means of describing their complexity have to be used. Several notions have been introduced to accomplish this. They aim at finding local properties of the configuration space that help to describe global characteristics. These properties can be used to determine a *statistical* or *probabilistic* complexity measure.

In order to capture the intrinsic complexity of the configuration free space the notion of *ϵ-goodness* was introduced (Kavraki, Latombe, Motwani, and Raghavan

1995). Recall from Section 6.1.1 that the free space is computed by sampling the configuration space uniformly at random. The complexity of the free space around a particular configuration in configuration space can be characterized by the number of adjacent configurations to which a motion plan can easily be computed. If for every point of the free space this set contains at least an ϵ-fraction of the entire free space, the configuration space is said to be *ϵ-good*. The set is called the visibility region of the sample.

Narrow passages in the configuration space cause the value ϵ to be very small, despite the fact that the planning problem at hand might not necessarily difficult (Hsu, Latombe, and Motwani 1997; Hsu, Kavraki, Latombe, Motwani, and Sorkin 1998). To address this problem a more powerful notion of characterizing configuration free space was introduced. *Expansive spaces* (Hsu, Latombe, and Motwani 1997) require each sample to be ϵ-good. In addition, a condition is imposed on how much points in the visibility region can contribute to extending the overall visibility region. Intuitively, this corresponds to looking around the corner in an L-shaped hall: standing at the end of the hall only parts of it are visible. But once the turn in the hall is reached the entire hall around the corner can be viewed also. Using these notions, the number of samples necessary to describe the free space accurately with high probability can be estimated. This number will govern the computational complexity of those algorithms.

10.2 Elastic Strips

In attempting to compare the computational complexity of the elastic strip framework to that of motion planning algorithms, several fundamental differences have to be noted. First, the elastic strip framework operates entirely in the workspace, a space of bounded dimensionality, whereas motion planning algorithms operate in a space whose dimensionality depends on the number of degrees of freedom of the particular robot at hand. Second, only the local free space around a given trajectory is considered. Those differences will have a significant impact on the tools that can be applied in evaluating the computational complexity of the elastic strip framework.

The notion of ϵ-goodness introduced to characterize configuration free space represents an entirely local property: the set of configurations visible from a particular configuration. As this notion does not capture the essence of the configuration space in sufficient detail, expansive spaces were introduced, slightly reducing the locality of the characterization. Locality is necessary, however, to be able to apply statistical methods to the determination of computational complexity. Therefore, it has to be concluded that in describing the complexity of free space a local measure can be employed. Attempting to use a uniform local description, however, characterizes the entire free space according to its most difficult region. The notions of ϵ-goodness and expansive spaces attempt to use a small set of parameters to capture a local property of the free space. This property is chosen such that it applies all over the free space. It is evident that such a representation overestimates the difficulty of the planning problem, as the most difficult passage prescribes the overall complexity.

The approach taken in probabilistic planning methods is nevertheless justified, as the entire free space needs to be considered and the free space is of dimensions where geometric intuition becomes increasingly difficult and even meaningless. Both of these arguments do not apply to the elastic strip framework, where free space is described directly in the workspace and only the portion of the free space about the given trajectory has to be considered. This motivates a geometric method of determining the computational complexity of the elastic strip framework.

The three major algorithmic procedures determining the computational complexity of the elastic strip framework are the covering criterion, the connectivity criterion, and the distance computation. They are examined in detail below. All other operations take $O(1)$ constant time.

10.2.1 Covering Criterion

The covering criterion described in Section 8.3 determines a set of bubbles that contain the hull (see Section 8.1) of a rigid body. In this section it will be determined how many bubbles are necessary to accomplish this task, given a minimum clearance of the hull to obstacles. We will make three simplifying assumptions:

1. The radius of the hull does not vary along the spine, i.e. the body to be covered is a cylinder with spherical caps.

2. The free space about the body is uniform, i.e. the free space is a cylinder whose axis coincides with the spine of the hull. The free space is parameterized by the difference of the radii of the two cylinders. This corresponds to a worst case environment: along the entire trajectory obstacles always are at the same distance to the robot.

3. We are only attempting to cover the hull, as opposed to generating an accurate model of the free space around the body. This implies that the intersection of two adjacent bubbles is allowed to intersect with the hull itself.

These assumptions only influence the complexity of the covering criterion up to a constant factor, which can be ignored.

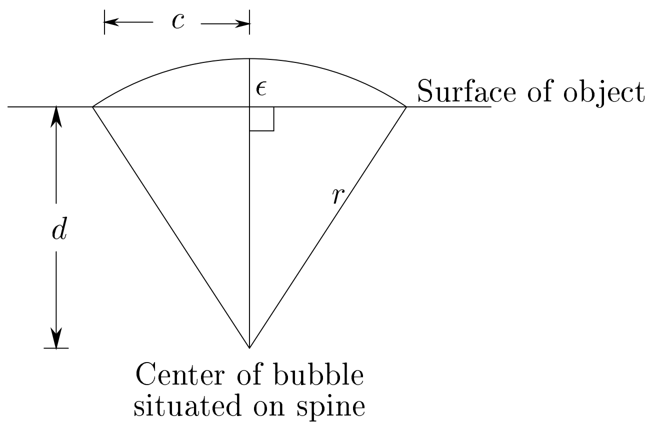


Figure 10.1: Schematic illustration of a bubble covering a rigid body: It is shown how a bubble approximates the linear surface of a body with a circular arc. The parameter $2 \cdot c$ describes how much of the the body was covered and depends on d, the width associated with the spine, and r, the radius of the bubble.

Consider the illustration in Figure 10.1. It shows how a single bubble is used to cover part of a rigid body. Only a portion of the bubble and the surface of the body are shown. The value r designates the radius of the bubble, its center is situated on the spine; d is the radius of the body around that spine. The value $2\ c$ represents the length of the body that is covered by the bubble. The two following equations relating these quantities follow directly:

$$r = d + \epsilon \qquad \text{and} \qquad c = \sqrt{r^2 - d^2},$$

which by substitution lead to

$$c = \sqrt{\epsilon^2 + 2\ \epsilon\ d}.$$

It is obvious that as ϵ approaches zero, c will approach zero and an infinite number of bubbles are needed to cover the hull of the body. This would be quite inefficient. But since this framework is designed for motion in free space and not in contact, we can assume that ϵ will be larger than zero. Now the question becomes: How much free space is needed around the body for this approach to be practical?

First, we attempt to derive an expression for the number of bubbles needed, given a certain amount of free space around the hull of the rigid body. This number can be obtained by dividing the length l of the body by $2\ c$, the length covered by one bubble:

$$n_b = \frac{l}{2\ c} = \frac{l}{2\ \sqrt{\epsilon^2 + 2\ \epsilon\ d}}$$

In order to normalize this equation and render it independent of the dimensions of the rigid body, the aspect ratio $a = l/d$ is used:

$$n_b' = \frac{n_b\ d}{l} = \frac{d\ l}{2\ l\ \sqrt{\epsilon^2 + 2\ \epsilon\ d}}.$$

We define the variable x to designate the ratio of free space and the value d, i.e. $x = \frac{\epsilon}{d}$. Substituting $\epsilon = x \cdot d$ yields:

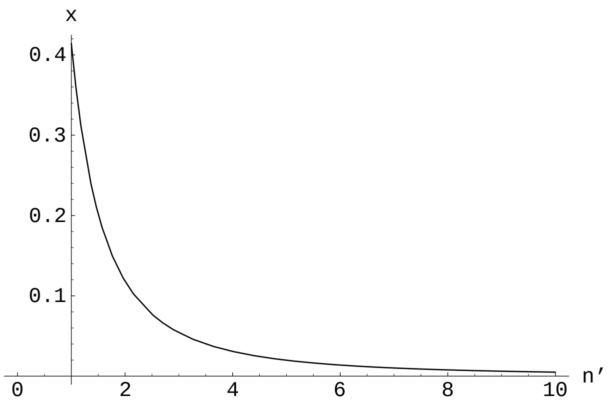
$$n_b' = \frac{1}{2\sqrt{x^2 + 2x}},$$

where n_b' now denotes the number of bubbles needed to cover a volume unit of a rigid body, independently of its width, height, and length, expressed in terms of how much free space is available around the object relative to the radius of its representation. A volume unit is defined as a portion of a cylindrical spine twice as wide as the radius of the spine. This is a natural way of splitting up the length of the spine, because it is the optimal volume of a cylinder inscribed into a single bubble.

Solving for x, we obtain an expression that relates the number of bubbles used to cover a unit volume of a rigid body to the amount of free space required for those bubbles to contain the body.

$$x = \frac{-n'_b + \sqrt{1 + {n'_b}^2}}{n'_b}$$

This equation converges quadratically, which implies that as the number of bubbles per unit volume is increased the required free space quickly becomes very small. A plot of the function is shown in Figure 10.2.



n'_b	$x = \epsilon/d$
1	0.41
2	0.12
3	0.054
4	0.03
5	0.02
10	0.005
100	0.00005

Figure 10.2: The graph relates the number of bubbles used to cover a unit volume of a rigid body to the achievable resolution of free space. For example, using five bubbles per unit volume, the body can be described with an accuracy of 2% of the radius associated with the spine or 1% of the width of the body.

If only one bubble is used per unit volume of rigid body, the radius of the cylinder of free space has to be 41% wider than the radius of the hull. This is to be expected, due to the radius of a sphere enclosing a cube. Allowing more than one bubble per unit volume of rigid body quickly reduces that requirement. When 10 bubbles are used, the radius of the free space is only required to be 0.5% larger than the radius of the hull of the body.

We will make the assumption that whenever the robot is in contact with the environment the resulting constraints are going to be used for collision avoidance (see Section 7.6). This implies that the free space around the hull of any rigid body will not be lower than a constant ϵ-fraction of the radius d of the rigid body's hull. Effectively, this corresponds to the notion of resolution-completeness in motion planning (Latombe 1991). Now the complexity of the covering criterion per volume unit can be bounded by a constant. This implies that the complexity of the covering criterion for a rigid body depends linearly on the aspect ratio of the object to be covered.

The computational complexity of the covering criterion for a body of aspect ratio a is $O(a)$ if there exists a constant $\epsilon > 0$ such that the value of the radius of the free space around the rigid body is bounded from below by $(1+\epsilon) \cdot d$. The value d denotes the radius of the rigid body's hull.

10.2.2 Connectivity Criterion

The computational complexity of the connectivity criterion described Section 8.5 is linear in the number of bubbles on the two spines: every bubble is considered at most twice and the intersection computation is of $O(1)$. Hence, for the case depicted in Figure 10.3, if m and n denote the number of bubbles on the two spines, the complexity of the connectivity criterion is $O(m+n) = O(n)$, assuming $O(m) = O(n)$.

At first glance, this linear bound does not seem to hold for the case depicted in Figure 10.3 b). As shown in the figure, there are $O(n)$ intersections that have to checked for each of the $O(n)$ bubbles. Hence, the naive algorithm is of complexity $O(n^2)$. A linear bound of $O(n)$ can be obtained by exploiting the properties of the hull: If a wider portion of the hull was able to perform the motion from one configuration to the next, subsequent narrower portions will also be able to do so. Conversely, a narrow part of the hull can be subsumed by a subsequent wider one. Using those

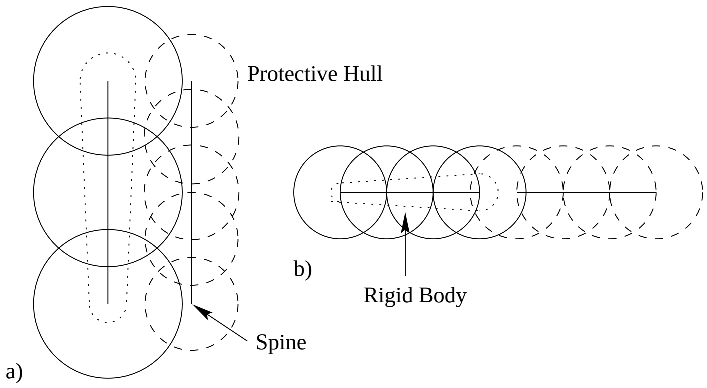


Figure 10.3: Depending on the relative positioning of consecutive spines, different arguments have to be made in support of the linear bound of the connectivity criterion.

optimizations, the linear bound of $O(n)$ can be achieved. Therefore, the connectivity criterion for two protective hulls can be computed in $O(n)$, where n is the number of bubbles in the protective hull.

10.2.3 Distance Computation

All distance computations in the elastic strip framework are distance computations between a given point and the environment. This is due to the representation of rigid bodies by spines. A distance computation can always be performed between a point on the spine an the reset of the environment. The computational complexity of such a distance computation depends on the complexity of the environment. Various algorithms have been proposed for distance computation (see Section 8.6). Most of them use bounding hierarchies to achieve a logarithmic complexity bound relative to the complexity of the environment. The approach used in the implementation of elastic strips presented in this thesis uses a hierarchy of bounding spheres (Quinlan 1994a). For this algorithm the logarithmic bound has been confirmed experimentally.

The environment to which the distance computation is applied is inherently dynamic. This imposes the requirement of having a bounding sphere hierarchy for every independently moving object. Assuming that there are n such objects and that the distance computation for each of them can be performed in $O(\log m)$, where m is the number of bounding spheres used to describe the object, the closest distance between a point and any object in the environment can be determined in $O(n \log m)$. In practice, exploiting spatial coherence by caching of the closest obstacle can eliminate the factor n in most cases.

10.2.4 Overall Complexity

To determine the overall computational complexity of the elastic strip framework, the algorithmic pieces examined the the sections above need to be put together. The goal is to determine the total number of bubbles needed to cover all configurations along the entire elastic strip. This will determine the number of distance computations for the covering criterion and bound the the number of intersection operations for the connectivity criterion. In this discussion, the computational complexity of the modification procedure for each intermediate configuration along the elastic strip is considered to be $O(1)$ (see Section 7.3). The matrix multiplication for the Jacobian J is bounded by a constant, assuming that the number of degrees of freedom is bounded by a constant. This matrix multiplication is the only operation that depends on the number of degrees of freedom of the robot.

It is apparent that the number of bubbles on the entire elastic strip depends on the number of bubbles per protective hull and on the number of protective hull along the elastic strip. For an individual protective hull a bound has been established in Section 10.2.1. The same result can be applied the entire elastic strip, as the volume swept by a rigid body can be regarded as a rigid body itself. For this rigid body the notion of aspect ratio is not very intuitive. It makes more sense to parameterize the complexity along the strip using the strip's length l and to absorb the constant from the narrowest body on the robot in the O-notation. As a result, the overall bubble count amounts to:

$$O(l) \cdot \sum_{b \in \mathcal{R}} O(a_b) = O(l \cdot w),$$

where b is a rigid body of the robot $\mathcal{R}$ and a_b is the aspect ratio of rigid body b. The sum over all rigid bodies of the robot can be replaced with a constant w that intuitively corresponds to the accumulative length of all links of the robot.

The computational complexity of $O(l \cdot w)$ can be interpreted as covering a two-dimensional surface – the elastic strip – with bubbles. Its length is determined by l and the width by w. In this interpretation of the algorithm has quadratic complexity. It is realistic, however, to assume that the expression of $O(w)$ is bound by a constant, as only robots of finite size are considered. This would result in a linear complexity in the length of the elastic strip, $O(l)$. Experimentally it can be observed that the duration of an update of the elastic strip is directly proportional to the number of intermediate configurations along the strip, supporting the linear bound (see Chapter 9).

Obviously, this ignores the cost of distance computation so that the resulting overall complexity of the elastic strip framework is

$$O(l \cdot n \log m),$$

where l represents the length of the elastic strip, n the number of independently moving objects, and m the number of bounding spheres used to cover the objects. The value of m depends on the surface area of the object and can be bounded by a constant, assuming a bounded object size. This result implies that reactively maintaining a previously planned path is a much cheaper operation as replanning.

The comparison of planning or replanning methods and the elastic strip framework is difficult. Motion planning considers the entire free space, whereas elastic strips only consider the local free space around the trajectory. The modification of the elastic strip does not incorporate topological changes of the trajectory. Planning methods, on the other hand, consider all topological possibilities during the computation of a plan. As the input to the elastic strip framework is a plan resulting from a motion planning algorithm, elastic strips do not replace motion planning, but complement it. It can be stated, however, that the elastic strip framework represents an attractive alternative to *re*planning.

To conclude the discussion of the computational complexity of the elastic strip framework, the assumptions leading to the $O(l \cdot n \log m)$ bound are listed below:

Resolution Completeness of Rigid Body Representation: The assumption of an ϵ-fraction of free space around each obstacle corresponds to the notion of resolution completeness in motion planning.

Bounded Size of Robot: A robot consists of a finite set of bounded rigid bodies. It is reasonable to assume that the length of all spines required to describe those bodies, weighted by the body's aspect ratio, can be bounded by a constant.

Bounded Size of Obstacles: In order to place a bound on the number of spheres used to cover a rigid body for distance computation, only bodies of a bounded size are permitted.

Simplifications for Covering Criterion: In section 10.2.1 three assumptions are made to help determine the computational complexity of the covering criterion. They influence the result only by a constant factor and can therefore be neglected.

10.3 Parallelized Elastic Strips

In Section 7.8, a parallelized version of the elastic strip framework was introduced. While such a parallelization will result in a significant speedup in practice, the computational complexity of the overall algorithm remains the same. The number of processors must be bounded by a constant and therefore can only achieve a speedup of a constant factor. Hence, they do not reduce the complexity of the elastic strip framework.

10.4 Conclusion

The elastic strip framework was designed to allow task-driven motion execution in dynamic environments. Rather than applying a replanning operation to a trajectory in order to reflect changes in the environment, an elastic strip reactively and incrementally modifies the existing trajectory. This section evaluates the efficiency of the elastic strip framework by comparing its computational complexity to the complexity of motion planning.

A direct comparison between the elastic strip framework and motion planning is difficult, as they address two fundamentally different problems. Motion planning represents the connectivity of the entire free space to determine a motion. Elastic strips use reactive methods to incrementally modify an existing motion and therefore have a much narrower applicability. Nevertheless, oftentimes planning is applied to incremental trajectory modification and the goal is to establish the elastic strip framework as a computationally more efficient alternative.

As discussed in Section 10.1, the computational complexity of motion planning grows exponentially in the dimension of the configuration space. For the elastic strip framework, a a sub-quadratic complexity bound as a function of the length of the trajectory was established in Section 10.2. It can therefore be concluded that within the range of problems to which the elastic strip framework is applicable, it represents a computationally more efficient alternative to planning algorithms.

Chapter 11

Conclusion

In this thesis a unified motion generation framework for the integration of motion planning and motion execution was presented. Two motion generation approaches were developed within this framework. The global dynamic window approach allows goal-directed robot navigation at high velocities in unknown and dynamic environments. The elastic strip framework is a more general approach, capable of real-time path modification and motion generation for robots with many degrees of freedom. The elastic strip framework links gross and fine motion planning and thereby offers a unified methodology to gross and fine motion execution. The motion capabilities resulting from both of these motion generation approaches demonstrate the effectiveness of the integration of planning and execution.

11.1 Global Dynamic Window Approach

The global dynamic window approach integrates planning and execution methods into a motion generation framework for mobile robots. It combines an efficient planner based on numerical navigation functions with an execution algorithm that takes the kinematic and dynamic constraints of the robot into account. This resulting motion generation framework allows a robot to navigate in an extremely robust manner in completely unknown and dynamic environments at very high velocities. During its motion, the robot relies entirely on sensory information.

The global dynamic window approach was implemented and tested on the Nomad XR4000, a holonomic, caster-driven mobile base. Various experiments were conducted demonstrating the effectiveness and efficiency of the approach. The robot moved autonomously, traveling at velocities of up to $1m/s$ over a distance of about 50 meters in an unknown environment, solely by specifying a desired goal configuration. As current sensory information invalidates a previously planned trajectory, the robot will search for a new, unobstructed route.

11.2 Extensions to the Global Dynamic Window Approach

The global dynamic window approach as introduced here represents a very powerful approach to motion generation for mobile robots. Comparing its behavior to the motion of a human in a dynamic environment, one shortcoming can be observed. The global dynamic window approach allows the robot to evade dynamic obstacles while moving towards the goal. The selection of a motion command, however, does not incorporate a predictive model of the obstacle's motion. Every motion command is determined purely based on the current static "snapshot" of the environment. It is not evident how such a trajectory estimation could be integrated into the approach. The efficiency of the global dynamic window approach seems to depend on this "snapshot"-property. Introducing configuration-time space is likely to render the approach computationally much more complex.

As the global dynamic window approach relies on odometry to determine if the goal configuration has been reached, its integration with relocalization and map-building techniques would allow to eliminate this limitation. Such an integration could significantly improve the performance for longer routes, where odometry error results in deviation from the desired goal configuration. The entire plan could be divided into sections that are executed sequentially, using the global dynamic window approach. Each time an intermediate goal has been reached, the next intermediate goal could be specified, after relocalizing. The relocalization step would reduce or eliminate the odometry error and thereby allow to reach distant goal configurations.

11.3 Elastic Strip Framework

The elastic strip framework integrates planning and execution by augmenting the path resulting form a motion planning operation with a reactive component. This reactive component allows the real-time modification of planned trajectories for robots with many degrees of freedom in reaction to changes in the environment occurring during the execution of a task. Those reactions are performed without suspending task execution. This allows mobile robots to robustly perform complex missions in dynamic environments. By integrating planning and low-level control within one framework, the elastic strip approach also unifies gross and fine motion execution.

The notions of protective hull and elastic tunnel were introduced as a method of describing the free space around a robot and its trajectory. By representing the free space directly in the workspace, the computational complexity of the elastic strip framework is rendered independent of the dimensionality of the configuration space. The efficiency of the elastic strip framework is based on this novel way of free space representation.

The effectiveness of the elastic strip framework was demonstrated experimentally. During the execution of a motion, the elastic strip has to be translated into a trajectory. Existing trajectory execution methods assume the trajectory to remain static throughout the entire motion. As the elastic strip changes during the execution of a motion in reaction to moving obstacles, a new trajectory execution methods was presented to address this problem. It allows discrete changes in the trajectory, while executing a motion with continuous velocities.

11.4 Extensions to the Elastic Strip Framework

The elastic strip framework is a very general approach to motion execution for robots with many degrees of freedom in dynamic environments. Not all areas of application or possible extensions are treated exhaustively in this thesis. Below we describe possible directions for further investigation.

Replanning: The subject of replanning is treated in some detail in Section 7.7. Various approaches to the integration of replanning into the elastic strip framework are discussed. The integration of those methods would be a natural extension to the current implementation.

Motion in Contact: As discussed in Section 7.6, it is possible to extend the elastic strip framework to motion in contact with the environment. Force-control algorithms can easily be integrated into the framework of reactive gross motion execution, effectively unifying the execution of gross and fine motion.

Parallelization: As the tasks to which the elastic strip framework is applied get more challenging, i.e. require longer trajectories in tight environments, processing power will become a relevant factor. The elastic strip framework lends itself very easily to parallelization and these limitations can be overcome by executing distance computation and update procedures in a multi-processor environment or on a parallel computer architecture.

Sensing: Sensing is of great importance to the elastic strip framework, as it relies on knowledge about the motion of all objects in the workspace. The monitoring of changes in the environment represents a challenging task. One could imagine a set of fixed, redundant, wall-mounted cameras monitoring the free space and updating the world model accordingly. More details and a novel sensing approach tailored to the requirements of the elastic strip framework were presented in Section 7.9.

The global dynamic window approach and the elastic strip framework were presented. They integrate motion planning and execution methods into a unified framework of motion generation for robots in dynamic environments. Experimental results were presented to validate their efficiency and effectiveness. The application of these framework results in a significant increase in motion capabilities for robots.

The global dynamic window approach allows mobile robots to navigate robustly and with previously unmatched velocities in dynamic and unknown environments. The elastic strip framework performs real-time trajectory modification of gross and fine motion plans, enabling the reliable execution of complex tasks in dynamic environments, using robots with many degrees of freedom. This thesis demonstrated that the integration of motion planning and motion execution algorithms incorporates the advantages of both approaches into a single framework of robot motion generation.

Appendix A

Robot Descriptions

A.1 Nomad XR 4000

The Nomad XR 4000, shown in Figure A.1, is a holonomic mobile base actuated by caster-based drive system. It is equipped with 48 sonars sensors, 48 infra-red sensors, a pressure sensitive skin, and a SICK laser range finder. It moves at velocities of up to $1.2\frac{m}{s}$ with accelerations of up to $5\frac{m}{s^2}$. The design and control of caster-based mobile robots is discussed in Holmberg 1999.

Figure A.1: The Nomad XR4000

A.2 Stanford Mobile Manipulator

The Stanford Mobile Manipulator, shown in Figure A.2, consists of a 6 degree of freedom PUMA 560 robot arm mounted on a holonomic mobile base. It has a total of 9 degrees of freedom. The drive-system of the base was designed by Oakridge National Laboratories and uses orthogonal wheels to achieve holonomicity. The body of the base was designed and build at Stanford University in collaboration with Nomadic Technologies. It is equipped with 48 sonar sensors and 48 infra-red sensors. The Denavit-Hartenberg parameters of this mobile manipulator are given in Table A.1.

Figure A.2: Stanford Mobile Manipulator

A.3 Mitsubishi PA-10

The Mitsubishi PA-10, shown in Figure A.3, is a 7 degree-of-freedom manipulator. Its kinematics are modeled after the human arm. The workspace of this robot is depicted in Figure A.4 and its Denavit-Hartenberg parameters are shown in Table A.2.

i	α_{i-1} degrees	a_{i-1} meters	d_i meters	θ_i degrees
1	-90	0	d_x	0
2	90	0	d_y	0
3	0	0	0	θ^{Base}
4	0	0	0	θ_1^{Puma}
5	-90	0	.2435	θ_2^{Puma}
6	0	.4318	-.0934	θ_3^{Puma}
7	90	-.0203	.4331	θ_4^{Puma}
8	-90	0	0	θ_5^{Puma}
9	90	0	0	θ_6^{Puma}

Table A.1: Denavit-Hartenberg parameters of the Stanford Mobile Manipulator

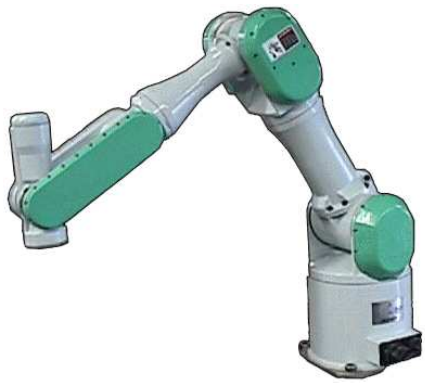


Figure A.3: Mitsubishi PA-10

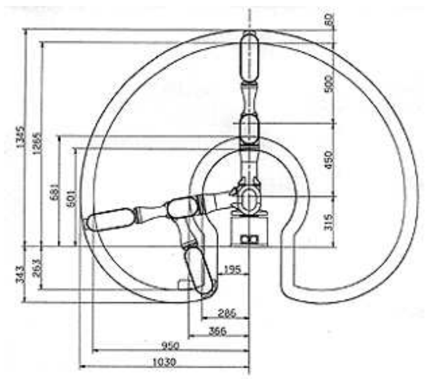


Figure A.4: Workspace of the Mitsubishi PA-10

i	α_{i-1} degrees	a_{i-1} meters	d_i meters	θ_i degrees
1	0	0	0	θ_1
2	-90	0	0	θ_2
3	90	-0.45	0	θ_3
4	-90	0	0	θ_4
5	90	-0.5	0	θ_5
6	-90	0	0	θ_6
7	90	0	0	θ_7

Table A.2: Denavit-Hartenberg parameters of the PA-10

Bibliography

Akl, S. G. (1989). *The Design and Analysis of Parallel Algorithms*. Prentice-Hall.

Arkin, R. C. (1987). Motor schema-based mobile robot navigation. *International Journal of Robotics Research 8*(4), 92–112.

Avnaim, F. and J. D. Boissonnat (1988). Polygon placement under translation and rotation. Technical Report 889, INRIA, Sophia-Antipolis, France.

Azarm, K. and G. Schmidt (1994). Integrated mobile robot motion planning and execution in changing indoor environments. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Volume 1, pp. 298–305.

Barraquand, J. and J.-C. Latombe (1990). A monte-carlo algorithm for path planning with many degrees of freedom. In *Proceedings of the International Conference on Robotics and Automation*, pp. 1712–1717.

Barraquand, J. and J.-C. Latombe (1991). Robot motion planning: A distributed representation approach. *International Journal of Robotics Research 10*(6), 628–649.

Bobrow, J. E., S. Dubowsky, and J. S. Gibson (1985). Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research 4*(3), 3–17.

Borenstein, J. and Y. Koren (1991). The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation 7*(3), 278–288.

Borg, S. F. (1990). *Fundamentals of Engineering Elasticity* (2nd ed.). World Scientific.

Brock, O. and O. Khatib (1997). Elastic strips: Real-time path modification for mobile manipulation. In *Proceedings of the International Symposium of Robotics Research*, pp. 117–122. Preprints.

Brock, O. and O. Khatib (1998a). Elastic strips: Real-time path modification for mobile manipulation. In Y. Shirai and S. Hirose (Eds.), *Robotics Research*, pp. 5–13. Springer Verlag.

Brock, O. and O. Khatib (1998b). Executing motion plans for robots with many degrees of freedom in dynamic environments. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 1–6.

Brock, O. and O. Khatib (1998c). Mobile manipulation: Collision-free path modification and motion coordination. In *Proceedings of the International Conference on Computational Engineering in Systems Applications*, Volume 4, pp. 839–845.

Brock, O. and O. Khatib (1999a). Elastic strips: A framework for integrated planning and execution. In P. Corke and J. Trevelyan (Eds.), *Proceedings of the International Symposium on Experimental Robotics*, Volume 250 of *Lecture Notes in Control and Information Sciences*, pp. 328–338. Springer Verlag.

Brock, O. and O. Khatib (1999b). High-speed navigation using the global dynamic window approach. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 341–346.

Brock, O. and O. Khatib (1999c). Real-time obstacle avoidance and motion coordination in a multi-robot workcell. In *Proceedings of the International Symposium on Assembly and Task Planning*, pp. 274–279.

Brooks, R. A. (1983). Solving the find-path problem by good representation of free-space. *IEEE Transactions on Systems, Man, and Cybernetics SMC-13*(3), 190–197.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *International Journal of Robotics and Automation RA-2*(1), 14–23.

Brooks, R. A. and T. Lozano-Pérez (1985). A subdivision algorithm in configuration space for findpath with rotation. *Proceedings of the International Conference on Systems, Man, and Cybernetics SMC-15*(2), 224–233.

Burgard, W., A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun (1998). The interactive museum tour-guide robot. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Volume 1, pp. 11–18.

Cameron, S. (1997). Enhancing gjk: Computing minimum and penetration distances between convex polyhedra. In *Proceedings of the International Conference on Robotics and Automation*, Volume 4, pp. 3112–3117.

Canny, J. F. (1988). *The Complexity of Robot Motion Planning.* MIT Press.

Canny, J. F. and M. C. Lin (1993). An opportunistic global path planner. *Algorithmica 10*, 102–120.

Çela, A. S. and Y. Haman (1992). Optimal motion planning of a multiple-robot system based on decomposition coordination. *IEEE Transactions on Robotics and Automation 8*(5), 585–596.

Chang, K.-S. (1998). *Robotics Library.* Stanford University. Dynamic control and simulation library.

Chang, K.-S. (1999). *Efficient Dynamic Control and Simulation of Highly Redundant Articulated Mechanisms.* Ph. D. thesis, Stanford University. To appear.

Cheung, E. and V. Lumelsky (1992). A sensitive skin system for motion control of robot arm manipulators. *Journal of Robotics and Autonomous Systems 10*, 9–32.

Choi, W. and J.-C. Latombe (1991). A reactive architecture for planning and executing robot motions with incomplete knowledge. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Volume 1, pp. 24–29.

Chu, H. and H. A. ElMaraghy (1992). Real-time multi-robot path planner based on a heuristic approach. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 475–480.

Chu, H. and H. A. ElMaraghy (1993). Integration of task planning and motion control in a multi-robot assembly workcell. *Robotics and Computer-Integrated Manufacturing 10*(3), 235–255.

Craig, J. J. (1989). *Introduction to Robotics: Mechanics and Control* (2nd ed.). Addison-Wesley.

Croft, E. A., R. G. Fenton, and B. Benhabib (1995). Time-optimal interception of objects moving along topologically varying paths. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, Volume 5, pp. 4089–4094.

Dorst, L. and K. Trovato (1989). Optimal path planning by cost wave propagation in metric configuration spa. In *Proceedings of the International Society for Optical Engineering – Mobile Robots III*, Volume 1007, pp. 186–197.

Erdmann, M. and T. Lozano-Pérez (1986). On multiple moving objects. *Algorithmica 2*(4), 477–521.

Faverjon, B. and P. Tournassoud (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings of the International Conference on Robotics and Automation*, Volume 2, pp. 1152–1159.

Faverjon, B. and P. Tournassoud (1990). A practical approach to motion-planning for manipulators with many degrees of freedom. In H. Miura (Ed.), *Robotics Research*, pp. 425–432.

Feder, H. J. S. and J.-J. E. Slotine (1997). Real-time path planning using harmonic potentials in dynamic environments. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 874–811.

Feiten, W., R. Bauer, and G. Lawitzky (1994). Robust obstacle avoidance in unknown and cramped environments. In *Proceedings of the International Conference on Robotics and Automation*, Volume 3, pp. 2412–2417.

Fiorini, P. and Z. Shiller (1998). Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research 17*(7), 760–772.

Fox, D., W. Burgard, and S. Thrun (1996). Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Volume 3, pp. 1280–1287.

Fox, D., W. Burgard, and S. Thrun (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine 4*(1), 23–33.

Fox, D., W. Burgard, S. Thrun, and A. B. Cremers (1998). A hybrid collision avoidance method for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, Volume 2, pp. 1238–1243.

Franklin, G. F., J. D. Powell, and M. L. Workman (1998). *Digital Control of Dynamic Systems* (3rd ed.). Addison-Wesley.

Fujimura, K. and H. Samet (1989). A hierarchical strategy for path planning among moving obstacles. *IEEE Transactions on Robotics and Automation 5*(1), 61–69.

Gibson, S. F. F. and B. Mirtich (1997). A survey of deformable modeling in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Laboratory (MERL).

Gilbert, E. G., D. W. Johnson, and S. S. Keerthi (1988). A fast procedure for computing the distance between complex objects in three-dimensional space. *International Journal of Robotics and Automation 4*(2), 193–203.

Ginsberg, M. L. (1993). *Essentials of Artificial Intelligence.* Morgan Kaufman.

Goldstein, H. (1980). *Classical Mechanics* (2nd ed.). Addison-Wesley.

Gottschalk, S., M. C. Lin, and D. Manocha (1996). Obb-tree: A hierarchical structure for rapid interference detection. In *SIGGRAPH*, pp. 171–180.

Guibas, L. J. (1994). CS327: Computational geometry. Stanford University class notes.

Handley, S. (1993). The genetic planner: The automatic generation of plans for a mobile robot via genetic programming. In *Proceedings of the International Symposium on Intelligent Control*, pp. 190–195.

Holleman, C., L. Kavraki, and J. Warren (1998). Planning paths for a flexible surface patch. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 21–26.

Holmberg, B. (1999). *Design and Development of a Holonomic Mobile Robot*. Ph. D. thesis, Stanford University. To appear.

Holmberg, B. and O. Khatib (1999). Development of a holonomic mobile robot for mobile manipulation tasks. In *Proceedings of the International Conference on Field and Service Robotics*, pp. 268–273.

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Horsch, T., F. Schwarz, and H. Tolle (1994). Motion planning with many degrees of freedom – random reflections at c-space obstacles. In *Proceedings of the International Conference on Robotics and Automation*, pp. 3318–3323.

Hsu, D., L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin (1998). On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pp. 141–154. A K Peters.

Hsu, D., J.-C. Latombe, and R. Motwani (1997). Path planning in expansive configuration spaces. In *Proceedings of the International Conference on Robotics and Automation*, Volume 3, pp. 2719–2726.

Hu, H. and M. Brady (1994). A bayesian approach to real-time obstacle avoidance for a mobile robot. *Autonomous Robots 1*, 69–92.

Hubbard, P. M. (1996). Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics 15*, 179–210.

Jaouni, H., M. Khatib, and J.-P. Laumond (1997). Dynamic path modification for car-like nonholonomic robots. In *Proceedings of the International Conference on Robotics and Automation*, Volume 4, pp. 2920–2925.

Kanade, T., H. Kano, S. Kimuara, A. Yoshida, and K. Oda (1995). Development of a video-rate stereo machine. In *Proceedings of the International Conference* *on Intelligent Robots and Systems*, Volume 3, pp. 95–100.

Kanade, T. and H. S. S. Vedula (1998). The 3d room: Digitizing time-varying 3d events by synchronized multiple video streams. Technical Report CMU-RI-TR-98-34, Carnegie Mellon University.

Kanade, T., A. Yoshida, K. Oda, H. Kano, and M. Tanaka (1996). A stereo machine for video-rate dense depth mapping and its new applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 196–202.

Kant, K. and S. W. Zucker (1986). Toward efficient trajectory planning: Path velocity decomposition. *International Journal of Robotics Research RA-2*(3), 135–145.

Kass, M., A. Witkin, and D. Terzonpoulos (1988). Snakes: Active contour models. *International Journal of Computer Vision 1*(4), 321–331.

Kavraki, L. (1994). *Random Networks in Configuration Space for Fast Motion Planning*. Ph. D. thesis, Stanford University.

Kavraki, L., F. Lamiraux, and C. Holleman (1998). Toward planning for elastic objects. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pp. 313–325. A K Peters.

Kavraki, L. and J.-C. Latombe (1994). Randomized preprocessing of configuration space for fast path planning. In *Proceedings of the International Conference on Robotics and Automation*, pp. 2138–2145.

Kavraki, L., J.-C. Latombe, R. Motwani, and P. Raghavan (1995). Randomized query processing in robot motion planning. *Journal of Computer and System Sciences 57*(1), 50–60.

Kavraki, L., J.-C. Latombe, R. Motwani, and P. Raghavan (1998). Randomized query processing in robot motion planning. *Journal of Computer and System Sciences 57*(1), 50–60.

Kavraki, L. E., P. Švestka, J.-C. Latombe, and M. H. Overmars (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE*

Transactions on Robotics and Automation 12(4), 566–580.

Khatib, M. (1996). *Sensor-Based Motion Control for Mobile Robots*. Ph. D. thesis, LAAS-CNRS, Toulouse, France.

Khatib, M. and R. Chatila (1995). An extended potential field approach for mobile robot sensor-based motions. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, pp. 490–496.

Khatib, M., H. Jaouni, R. Chatila, and J.-P. Laumond (1997). How to implement dynamic paths. In *Proceedings of the International Symposium on Experimental Robotics*, pp. 225–236. Preprints.

Khatib, O. (1980). *Command Dynamique dans l'Espace Opérationel des Robots Manipulateurs en Présence d'Obstacle*. Ph. D. thesis, Ecole Nationale Supérieur de l'Aéronautic et de l'Espace, Toulouse.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research 5*(1), 90–98.

Khatib, O. (1987). A unified approach to motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research 3*(1), 43–53.

Khatib, O. (1998). CS323: Advanced robotics. Stanford University class notes.

Khatib, O., K. Yokoi, O. Brock, K.-S. Chang, and A. Casal (1999). Robots in human environments: Basic autonomous capabilities. *International Journal of Robotics Research 18*(7), 684–696.

Khatib, O., K. Yokoi, K.-S. Chang, R. Holmberg, A. Casal, and A. Baader (1995). Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Proceedings of the International Symposium of Robotics Research*.

Khatib, O., K. Yokoi, K.-S. Chang, D. Ruspini, R. Holmberg, and A. Casal (1996). Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Volume 2, pp. 546–553.

Kim, J.-O. and P. Khosla (1991). Real-time obstacle avoidance using harmonic potential functions. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 790–796.

Koditschek, D. E. (1987). Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the International Conference on Robotics and Automation*, pp. 1–6.

Krogh, B. H. (1984). A generalized potential field approach to obstacle avoidance control. In *Robotics Research*, pp. MS84–484/1–15.

Kuffner, J. J. (1998). Goal-directed navigation for animated characters using real-time path planning and control. In *Proceedings of Captech'98*.

Latombe, J.-C. (1991). *Robot Motion Planning*. Boston: Kluwer Academic Publishers.

Laumond, J. P. (1987). Obstacle growing in a non-polygonal world. *Information Processing Letters 25*(1), 41–50.

LaValle, S. M. (1998). Numerical computation of optimal navigation functions. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pp. 339–350. A K Peters.

Lavalle, S. M. and S. A. Hutchinson (1996). Optimal motion planning for multiple robots having independent goals. In *Proceedings of the International Conference on Robotics and Automation*, Volume 3, pp. 2847–2852.

LaValle, S. M. and J. J. Kuffner (1999). Randomized kinodynamic planning. In *Proceedings of the International Conference on Robotics and Automation*. To appear.

Li, T.-Y. and J.-C. Latombe (1997). On-line manipulation planning for two robot arms in a dynamic environment. *International Journal of Robotics Research 16*(2), 144–167.

Lin, M. C. (1993). *Efficient Collision Detection for Animation and Robotics*. Ph. D. thesis, University of California at Berkeley.

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers C-32*(2), 108–120.

Lozano-Pérez, T. (1987). A simple motion planning algorithm for general robot manipulators. *IEEE Transactions on Robotics and Automation RA-3*(3), 224–238.

Lumelsky, V. (1987). Algorithmic issues of sensor-based robot motion planning. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 1796–1801.

Martínez-Salvador, B., A. P. del Pobil, and M. Pérez-Francisco (1998). Very fast collision detection for practical motion planning part i: The spatial representation. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 644–649.

Mirolo, C., E. Pagello, and W.-H. Qian (1995). Simplified motion planning strategies in flexible manufacturing. In *Proceedings of the International Symposium on Assembly and Task Planning*, pp. 394–399.

Mirtich, B. (1997). V-clip: Fast and robust polyhedral collision detection. Technical Report TR-97-05, Mitsubishi Electric Research Laboratory (MERL).

Moutarlier, P. and R. Chatila (1991). Incremental free-space modeling from uncertain data by an autonomous mobile robot. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Volume 2, pp. 1052–1058.

Mulmuley, K. (1994). *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice-Hall.

Nalwa, V. S. (1993). *A Guided Tour of Computer Vision*. Addison Wesley.

Nayar, S. K., M. Watanabe, and M. Noguchi (1995). Real-time focus range sensor. In *Proceedings of the International Conference on Computer Vision*, pp. 995–1001.

Neumann, Ulrich, and H. Fuchs (1993). A vision of telepresence for medical consultation and other applications. In *Robotics Research*, pp. 565–571. Springer Verlag.

Nilsson, N. J. (1969). A mobile automaton: An application of artificial intelligence techniques. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 509–520.

O'Donnell, P. A. and T. Lozano-Pérez (1989). Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings of the International Conference on Robotics and Automation*, pp. 484–489.

Ó'Dúnlaing, C. and C. K. Yap (1982). A retraction method for planning the motion of a disc. *Journal of Algorithms 1*(6), 104–111.

Ong, C. J. and E. G. Gilbert (1997). The Gilbert-Johnson-Keerthi distance algorithm: A fast version for incremental motions. In *Proceedings of the International Conference on Robotics and Automation*, Volume 2, pp. 1183–1189.

Preparata, F. P. and M. I. Shamos (1985). *Computational Geometry: An Introduction.* Springer Verlag.

Quinlan, S. (1994a). Efficient distance computation between non-convex objects. In *Proceedings of the International Conference on Robotics and Automation*, Volume 4, pp. 3324–3329.

Quinlan, S. (1994b). *Real-Time Modification of Collision-Free Paths.* Ph. D. thesis, Stanford University.

Quinlan, S. and O. Khatib (1993a). Elastic bands: Connecting path planning and control. In *Proceedings of the International Conference on Robotics and Automation*, Volume 2, pp. 802–807.

Quinlan, S. and O. Khatib (1993b). Towards real-time execution of motion tasks. In *Experimental Robotics 2.* Springer Verlag.

Raskar, R., G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs (1998). The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of SIGRAPH*, pp. 179–188.

Reif, J. H. (1979). Complexity of the mover's problem and generalizations. In *Proceedings of the Symposium on Foundations of Computer Science*, pp. 421–427.

Reif, J. H. and M. Sharir (1985). Motion planning in the presence of moving obstacles. In *Proceedings of the Symposium on Foundations of Computer Science*, pp. 144–154.

Rimon, E. and D. E. Koditschek (1989). The construction of analytic diffeomorphisms for exact robot navigation on star worlds. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 21–26.

Rimon, E. and D. E. Koditschek (1990). Exact robot navigation in geometrically complicated but topologically simple spaces. In *Proceedings of the International Conference on Robotics and Automation*, pp. 1937–1942.

Rimon, E. and D. E. Koditschek (1992). Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation 8*(5), 501–518.

Schmidt, G. K. and K. Azarm (1993). Mobile robot path planning and execution based on a diffusion equation strategy. *Advanced Robotics 7*(5), 4794–90.

Schwartz, J. T. and M. Sharir (1983). On the 'piano movers' problem: II. general techniques for computing topological properties of real algebraic manifolds. *Advanced in Applied Mathematics 1*(4), 293–351.

Shatkay, H. and L. P. Kaelbling (1997). Learning topological maps with weak local odometric information. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Volume 2, pp. 920–927.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *Proceedings of the International Conference on Robotics and Automation*, Volume 4, pp. 2275–2282.

Singh, L., H. Stephanou, and J. Wen (1996). Real-time robot motion control with circulatory fields. In *Proceedings of the International Conference on Robotics and Automation*, Volume 3, pp. 2737–2742.

Steele, J. P. H. and G. P. Starr (1988). Mobile robot path planning in dynamic environments. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, Volume 2, pp. 922–925.

Stein, G. P. (1999). Tracking from multiple view points: Self-calibration of space and time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Tarn, T.-J. (1996). Path-based approach to integrated planning and control for robotic systems. *Automatica 32*(12), 1675–1687.

Thrun, S., A. Brücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt (1998). Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. P. Bonasso, and R. Murphy (Eds.), *Artificial Intelligence and Mobile Robots*, Chapter 1, pp. 21–52. MIT/AAAI Press.

Thrun, S., D. Fox, and W. Burgard (1998). Probabilistic mapping of an environment by a mobile robot. In *Proceedings of the International Conference on Robotics and Automation*, Volume 2, pp. 1546–1551.

Tilove, R. B. (1990). Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 566–571.

Trucco, E. and A. Verri (1998). *Introductory Techniques for 3-D Computer Vision.* Prentice Hall.

Warren, C. W. (1989). Global path planning using artificial potential fields. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 316–321.

Warren, C. W. (1990). Multiple path coordination using artificial potential fields. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 500–505.

Wempner, G. A. (1973). *Mechanics of Solids.* McGraw-Hill.

Wernecke, J. (1994a). *The Inventor Mentor.* Addison-Wesley.

Wernecke, J. (1994b). *The Inventor Toolmaker.* Addison-Wesley.

Xavier, P. G. (1997). Fast swept-volume distance for robot collision detection.

In *Proceedings of the International Conference on Robotics and Automation*, Volume 2, pp. 1162–169.

Yang, X. and M. Meng (1999). A neural network approach to real-time collision-free navigation of 3-d.o.f. robots in 2d. In *Proceedings of the International Conference on Robotics and Automation*, Volume 1, pp. 23–28.

Zelek, J. S. (1995). Dynamic path planning. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, Volume 2, pp. 1285–1290.