UNIVERSITY OF CALIFORNIA

Los Angeles

Robot Motion Planning Among Moving Obstacles

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Mechanical Engineering

by

Paolo Fiorini

1995

1995

The dissertation of Paolo Fiorini is approved.

James S. Gibson

Jason L. Speyer

Paul K. C. Wang

Sriram Dasu

Antal K. Bejczy

Zvi Shiller, Committee Chair

University of California, Los Angeles

1995

DEDICATION

Questa dissertazione é dedicata

a

mia Madre e a mio Padre

in ringraziamento

per il loro costante

incoraggiamento, affetto e fiducia.

Senza il loro aiuto,

questa tesi non esisterebbe.

(I dedicate this dissertation,

to my

Mother and Father

as a small sign of gratitude

for their constant

encouragement, love and confidence.

Without their support,

this dissertation would not be.)

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I am deeply thankful to Prof. Zvi Shiller, my Thesis Advisor, for his help and patience during these years. He managed to give me the confidence and the tools necessary to complete this work. My thanks go also to my former Advisor Prof. Michel Melkanoff, and to Dr. Alfredo Inselberg for advising me during my first years at UCLA and for proposing the subject of this dissertation.

This work benefited greatly from the interaction with all the members of the (now defunct) Robotics and Automation Section of Caltech Jet Propulsion Laboratory. To each one of them, past and present colleagues, go my most sincere thanks. In particular I want to thank Dr. Antal Bejczy for having always provided food for my thoughts and on my table, and Dr. Edwin Kan, Mr. Daniel Kerrisk, Dr. Paul Schenker and Dr. Homayoun Seraji for having let me juggle professional and academic lives.

My friends Prof. Joel Burdick, Prof. Blake Hannaford, Dr. Marcos Salganicoff, Dr. Henry Stone, Mr. Satish Sundar and Dr. David Chichka were generous with good advice and fresh ideas, and my colleagues in the Man Machine Systems Group were always very helpful and supportive.

Finally I thank with all my heart Angela Cavani for having been my personal lightning rod during these trying years.

VITA

February 1st, 1953	Born in Verona (Italia) at lunch time on a Sunday
December, 1976	Laurea Degree in Electrical Engineering, University of Padova, Padova (Italia)
July 1977 to December, 1979	Microprocessor Systems Designer, Zeltron S.p.a, Udine (Italia)
January, 1980 to March 1981	System Analyst, Studio di Informatica, Milano (Italia)
April 1981 to June 1982	Teaching Assistant, University of California, Irvine (CA)
June 1982	Master of Science in Electrical Engineering, University of California, Irvine (CA)
July 1982 to January 1985	Senior Design Engineer, Parker Hannifin Corp. and Parker Bertea Corp., Irvine (CA)
February 1985 to -	Member of Technical Staff, Jet Propulsion Laboratory, Caltech, Pasadena (CA)

PUBLICATIONS

G. Buja and P. Fiorini. L'impiego dei microprocessori per il controllo dei sistemi di azionamento elettrico, *BIAS 1978*, Milano (Italy), Vol. 1, October 1978.

G. Buja and P. Fiorini. Controllo con microprocessore dei convertitori statici di energia elettrica, *Automazione e Strumentazione*, April 1979, Vol. 27, N. 4, pp. 269-274.

U. De Monte, P. Fiorini and others. A Software Structure for Appliance Control, *Euromicro Journal*, N. 6, pp. 17-19, 1980.

G. Buja and P. Fiorini. A Microcomputer Based Quasi Continuous Output controller for PWM Inverters, *IEEE International Conference on Industrial Electronics*, Philadelphia, PA, March 1980, pp.107-111.

G. Buja, P. Fiorini, and others. Improving the Performance of Microcomputer Based Controllers for PWM Inverters, *8th Powercom*, Dallas, TX, April 1982, pp. E1-3:1,E1-3:8.

G. Buja and P. Fiorini. Microcomputer Control of PWM Inverters, *IEEE Transactions on Industrial Electronics*, Vol. IE-29, N. 2, August 1982, pp. 212-216.

P. Fiorini and A. Stubberud. Models and Controls for Manipulators, *Robotic Intelligence and Productivity Conference*, Wayne State University, Detroit MI, November 1983, pp. 164-171.

P. Fiorini, B. Hannaford, B. Jau, E. Kan and A. Bejczy. Hand Trigger System for Bilateral Gripping Control in Teleoperation, *IEEE International Conference on Robotics and Automation*, Raleigh, NC, March 1987, pp. 586-592.

P. Fiorini. A Smart Hand for Manipulators, *IEEE Industrial Electronics Conference*, Cambridge, MA, November 1987, pp. 572-580.

B. Hannaford and P. Fiorini. A Detailed Model of Bilateral Teleoperation, *IEEE International Conference on Systems, Man and Cybernetics*, Beijin, China, August 1988, Vol. 2, pp. 117-122.

P. Fiorini and A. Inselberg. Parallel Coordinates : a New Representation for Robotics, *Proceedings of the NATO Advanced Research Workshop on Redundant Manipulators*, Salò (Italy), June 27 - July 1 1988.

P. Fiorini. A Versatile Hand for Manipulators, *IEEE Control Systems Magazine*, Vol. 8, N. 5, October 1988, pp. 20-24.

P. Fiorini and J. Chang. A Procedure Concept for Local Reflex Control of Grasping, *NASA Conference on Space Telerobotics*, Pasadena, CA, January 31 - February 2 1989, Vol. 4, pp. 81-90.

P. Fiorini and J. Chang. Autonomous Organization of Grasping Tasks, *SPIE Symposia on Aerospace Sensing, Artificial Intelligence VII*, Orlando, FL, March 27-31 1989, pp. 591-602.

P. Fiorini and A. Inselberg. Configuration Space Representation in Parallel Coordinates, *IEEE International Conference on Robotics and Automation*, Scottsdale AZ, May 14 - 19 1989, pp. 1215- 1220.

P. Fiorini. Task Dependent Control of Electric Motors, *IEEE Workshop on Microprocessor Control of Electric Motors*, Trieste (Italy), July 3 - 4 1989, pp. A6-1,A6-18. (Invited)

P. Fiorini and J. Chang. Architecture for Intelligent Control of Robotic tasks, *NASA Tech Briefs*, Vol. 15, N. 8, April 1991, pp. 28-29.

D. McAffee and P. Fiorini. Hand Controllers Design Requirements and Performance Issues in Telerobotics, *Fifth International Conference on Advanced Robotics (ICAR'91)*, Pisa (Italy), June 20-22 1991.

P. Fiorini, H. Das, H. Zak and A. Bejczy. Tele-operated Repair in Space: A System for Baseline Testing, *International Advanced Robotics Programme (IARP)*, The First Workshop on Robotics in Space, Pisa (Italy), June 17-18 1991.

P. Fiorini and J. Chang. More About Architecture for Intelligent Robotic Control, *NASA Tech Briefs*, Vol. 16, N. 6, June 1992, pp. 40-41.

P. Fiorini and A. Giancaspro. A Procedure for the Frequency Analysis of Telerobotics Tasks Data, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'92)*, Raleigh, NC, July 7-10 1992, pp. 873-880.

P. Fiorini, H. Das and S. Lee. Man-Machine Cooperation in Advanced Teleoperation, *6th Annual Space Operation, Application and Research Symposium (SOAR'92)*, Houston, TX, August 4-6 1992. (Invited)

P. Fiorini, A. Giancaspro, S. Losito and G. Pasquariello. Neural Networks for Off-Line Segmentation of Teleoperation Tasks, *International Symposium on Intelligent Control (ISIC-1992)*, Glasgow, UK, August 11-13 1992.

P. Fiorini, A. Bejczy and P. Schenker. Integrated Interface for Advanced Teleoperation, *IEEE International Conference on Systems, Man and Cybernetics (SMC'92)*, Chicago, IL, October 18-21 1992.

P. Fiorini, A. Giancaspro, S. Losito and G. Pasquariello. Real-Time Classification of Teleoperation Data with a Neural Network, *31st IEEE Conference on Decision and Control (CDC'92)*, Tucson, AZ, December 16-18 1992.

H. Das and P. Fiorini. Integrated Tools for Teleoperation, *International Symposium on Robotics and Manufacturing*, Santa Fè, NM, November 11-13 1992.

P. Fiorini, A. Giancaspro, S. Losito and G. Pasquariello. Neural Networks for the Segmentation of Teleoperation Tasks,
PRESENCE, Tele-operators and Virtual Environments, Vol. 2, N. 1, pp. 1-13, MIT Press, 1993.

P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments Using the Relative Velocity Paradigm, *IEEE International Conference on Robotics and Automation*, Atlanta GA, May 7-10 1993. (Finalist for the Anton Philips Best Student Paper Award).

P. Fiorini, H. Das, H. Zak and A.K. Bejczy. Tele-operated Repair in Space, a System for Baseline Testing, *NASA Tech Briefs*, Vol. 17, N. 5, May 1993, pp 43-44.

H. Das and P. Fiorini. Integrated Tools for Tele-operated Satellite Repair,
Automation in Construction, Vol. 2, N. 1, pp. 81-89, Elsevier Science Publisher, Amsterdam, The Netherlands, 1993.

P. Fiorini, A. Bejczy and P. Schenker. Integrated Interface for Advanced Teleoperation, *IEEE Control System Magazine*, Vol. 13, N. 5, October 1993.

P. Fiorini and A. Giancaspro. A Procedure for the Frequency Analysis of Telerobotic Task Data, *NASA Tech Briefs*, Vol. 18, N. 3, March 1994, pg. 120.

P. Fiorini, A. Giancaspro, S. Losito and G. Pasquariello. Real Time Classification of Teleoperation Data with a Neural Network, *NASA Tech Briefs*, Vol. 18, N. 4, April 1994, pg 78.

ABSTRACT OF THE DISSERTATION

Robot Motion Planning

Among Moving Obstacles

by

Paolo Fiorini

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 1995

Professor Zvi Shiller, Chair

This thesis presents a new approach for planning a robot's trajectory that avoids static and moving obstacles and minimizes motion time, subject to robot dynamics and actuator constraints. This approach consists of first computing a coarse trajectory that avoids the obstacles and satisfies an approximation of the actuator constraints. Then this trajectory is used as the initial guess of a dynamic optimization that satisfies obstacle avoidance, robot dynamics and the true actuator limits.

The first step of the approach is based on the concept of Velocity Obstacle (VO) that defines, at every instant in time, the set of colliding velocities between the robot and the obstacles. The VO set is computed using the relative velocity between the robot and each obstacle, and is instrumental in determining the set of robot velocities avoiding all obstacles and satisfying approximate system dynamics and

actuator limits. Within this set, the best avoidance maneuver is chosen heuristically so that the trajectory resulting from the sequence of maneuvers reaches the goal and minimizes motion time. The second step of the approach consists of refining the trajectory with a dynamic optimization that minimizes motion time, subject to the true robot's dynamics, actuator constraints, and time varying obstacle constraints. The dynamic optimization is based on Pontryagin's Minimum Principle and uses a gradient descent method.

These two steps lead to the implementation of a powerful and flexible planner that combines the advantages of heuristics with those of dynamic optimization. Heuristics in fact, captures the non-analytic aspects of motion planning, such as sequence of obstacle avoidance, conservative or aggressive maneuvers, and so on, thus characterizing the global structure of the trajectory. Dynamic optimization on the other hand, ensures feasibility and optimality of the trajectory thus guaranteeing its local correctness. This approach is demonstrated for planning the motions of an automated vehicle in an Intelligent Vehicle Highway System (IVHS) scenario, and of an articulated robot moving in a dynamic environment. This motion planner is suitable to a wide range of applications. The Velocity Obstacle method is very fast and, although approximate, it can be used for real time planning. The Dynamic Optimization is computationally intensive, but yields optimal solutions, which can be used when off-line planning is acceptable.

Lo duca e io per quel cammino ascoso
intrammo a ritornar nel chiaro mondo;
e sanza cura aver d'alcun riposo,
salimmo su, el primo e io secondo,
tanto ch'i' vidi de le cose belle
che porta 'l ciel, per un pertugio tondo.
E quindi uscimmo a riveder le stelle.

(My Guide and I crossed over and began
to mount that little known and lightless road
to ascend into the shining world again.
He first, I second, without thought of rest
we climbed the dark until we reached the point
where a round opening brought in sight the blest
and beauteous shining of the Heavenly cars.
And we walked out once more beneath the Stars.)

– La Divina Commedia, Inferno (XXXIV:136-143)
Dante Alighieri

Chapter 1

Introduction

1.1 Motivation

Many common environments can be described as dynamic, or time-varying, since they are characterized by the presence of several moving bodies negotiating each others. Typical examples are manufacturing tasks in which robot manipulators track and retrieve parts from moving conveyers, and air, sea, and land traffic, where aircraft, vessels and vehicles must avoid each other while moving towards their destination. A common feature of these examples is that the environment is unpredictable and often uncontrollable, and therefore it requires planning and executing appropriate trajectories, since the obstacles may not be expected to give way.

The computation of a correct motion plan in a dynamic environment is a difficult problem for a human, and a wrong solution can cause accidents and severe losses. It is therefore desirable that automatic planners be developed to autonomously compute, or at least assist the human in computing, the safe trajectories.

The main difficulty of dynamic motion planning is that it requires the simultaneous solution of the path planning and of the velocity planning problems. Path planning refers to the computation of a collision free path from start to goal without considering robot dynamics. Velocity planning consists instead of computing the velocity profile along a given path, that minimizes motion time or energy and satisfies system dynamics and actuator limits. In a static environment, if there exists a safe path, then a suitable velocity profile can always be found for the robot along that path. Clearly, in a dynamic environment this is not true, and path and velocity planning cannot be computed independently of each other, since obstacle avoidance depends on robot velocity and its dynamics. Furthermore, the uncertainty about the environment prevents the computation of a solution that is guaranteed to succeed. Each candidate trajectory must be tested to verify its correctness. Because of this, dynamic motion planning is an *intractable* problem that cannot be solved algorithmically, but requires the use of heuristics to manage its complexity [58], [14].

This thesis develops a method for first computing an initial guess of the trajectory, and then for refining this guess with a dynamic optimization. The initial trajectory guarantees the avoidance of both static and moving obstacles, and satisfies approximate dynamic constraints of the robot. The computation of the initial guess may be simplified by using heuristics, that would reduce the complexity of selecting the appropriate avoidance velocities. The dynamic optimization is used to satisfy the true dynamic constraints of the problem, and to minimize motion time. Thus, this method combines the advantages of heuristic motion planning with those of dynamic optimization.

The following sections of the introduction first survey the previous work in dynamic motion planning, and then formulate the motion planning problem as an optimization problem subject to state and control constraints, thus unifying the path and velocity planning problems. The last two sections summarize the method developed and present the content of the following chapters.

1.2 Previous Work

To date, motion planning in dynamic environments has not been widely addressed by researchers in Robotics. In the survey paper by Hwang [32], only eight references, out of the almost two hundred on the subject of gross motion planning, deal specifically with the problem of planning in time-varying environments. The earlier review book by Latombe [41] lists only thirteen references related to dynamic motion planning. Furthermore, most of this work is only partially related to this thesis, since Ó'Dúnlaing [50] alone considers the case of bounded robot's acceleration, and only Fujimura [27] and Canny [15] address the problem of minimum motion time.

Motion planning in dynamic environments benefited from the results of several research areas, such as geometric modeling, planning in static environments, and dynamic optimization. The relevant literature is henceforth reviewed in some detail.

1.2.1 Obstacles Representation

Several methods have been proposed to represent the obstacles in the environment in a way suitable to motion planning. The representation becomes an integral part of the motion planning algorithm, since it determines the collision test between the robot and the obstacles.

Set theory operations using basic shapes have been used by Cameron to model solid objects in motion [12]. His approach follows the ideas of constructive geometry by iteratively building occupancy maps consisting of the union and/or intersection of a few elementary shapes. This method has the drawback of not producing an analytical description of the obstacles and of their boundaries. This difficulty is overcome in [51], where O'Rourke uses maximally internal spheres to build generic solid shapes. This method is further refined by Featherstone who used a binary tree of minimally external spheres to describe both the shape of an object and its space occupancy during motion

[22]. This approach provides a recursive way for decomposing objects of any shape to the resolution required by a motion planning algorithm. A moving obstacle is in fact represented by a sphere enclosing the space swept by the obstacle during its trajectory. If a safe trajectory is not found with a single-sphere representation, the initial sphere can be replaced by two spheres in a finer decomposition, each enclosing half of the swept space, and the trajectory can be recomputed. This process is repeated until a solution is found or the object representation has reached the maximum resolution desired. Planar shapes can similarly be represented with hierarchies of minimally external circles.

1.2.2 Configuration Space

The original robot motion planning problem in Cartesian space has been converted by Lozano-Pérez to the equivalent problem of planning the motion of a point in the Configuration Space of the robot [46, 44, 45]. The robot is mapped into a point, and the obstacles are mapped into forbidden regions, representing collision configurations for the robot.

A general solution to the problem of finding a path in configuration space has been proposed by Canny in [13]. The *roadmap* algorithm maps the obstacles to the configuration space and finds a feasible path in polynomial time. The obstacle representation in configuration space is computed by testing the contact conditions between object and obstacles as described in [7], [61]. The roadmap is a graph describing the connectivity of the free regions of the configuration space. Although algorithms such as the roadmap or the cell decomposition of Schwartz [60] have been proven to compute the theoretically correct solution, in practice no realistic implementation has yet been possible because of their high computational complexity [17].

1.2.3 Dynamic Planning

For problems with moving objects, Erdmann added time as an additional variable to form an $n+1$ dimensional configuration-time space [20],[21]. The configuration space of the robot is computed at successive instants, with time added as an additional variable to the contact conditions. In this way the methods developed for planning in configuration space are directly applicable to this case, provided that the time axis is discretized with a suitable resolution.

In [38], [39], Kant decomposed the planning problem into two subproblems: the path planning and the velocity planning problems. The path planning problem consists of computing a path avoiding all static obstacles. The velocity planning problem determines the velocity along that path so that the robot avoids all the moving obstacles. This last step is carried out on a space-time plane in which the abscissa is the arc length of the path and the ordinate is time. A similar approach is described by Lee, [42], where the velocity profile is chosen to satisfy the additional goal of reaching the goal within a prescribed time limit. Approximate dynamic constraints are satisfied by limiting slope, direction and curvature of the trajectory in the space-time plane.

In [15], Canny presents an algorithm for computing a trajectory among fixed obstacles that satisfies approximate limits on the acceleration. The result is a polynomial time algorithm which computes a trajectory on a non-linear grid in the phase space. The trajectory is not guaranteed to be close to the true minimal-time trajectory, but the motion time is near-optimal. In [50], Ó'Dúnlaing presents an exact algorithm for computing the optimal velocity profile on a one-dimensional path subject to acceleration bounds.

In [58], Sanborn developed reaction rules for a robot moving in a Traffic World. The moving robot reacts to the environment by predicting the space-time intervals in which aggressive and indifferent obstacles would intersect its specified path. This approach takes into account the motion of the obstacles, but it neglects to consider the dynamics of the moving robot. In both [58] and [14] the authors show that dynamic motion planning is an intractable problem. In [58] this conclusion is reached by using the unpredictability of the environment as motivation, while in [14] it is formally proved using computational complexity theory.

In [27], [28], [26], approximate dynamic constraints are used by Fujimura to define the *collision front*, which represents the locus of collision points between two objects moving on arbitrary paths with piecewise constant velocities. However, this method is limited to robots that are faster than the obstacles.

The concept of *relative velocity* used in this thesis, has been previously applied by Inselberg to the computation of avoidance maneuvers for a circular object among circular moving obstacles in the context of air traffic control [33], [16].

1.2.4 Optimal Planning

A solution to the problem of minimum time planning between given end points for a manipulator was first proposed by Kahn in [37] where bang-bang control was computed for linearized robot dynamics. A bang-bang solution was also assumed by Scheinman, who used a given distance constraint to compute the control profiles and the placement of the manipulator [59].

The given distance problem was formulated by Weinreb as a Two Point Boundary Value Problem (TPBVP) [76], and solved iteratively using the Pontryagin Minimum Principle and a modified gradient algorithm [9]. This approach was extended by Meier to the case of variable initial conditions and bang-bang controls [48]. The initial guess for the gradient descent is given by the zero crossings of the controls computed by a preliminary optimization.

Another approach for computing the optimal solution to the point-to-point problem is presented by Sahar who uses dynamic programming to search a tessellated state space [57]. Parameter optimization was used by Rajan to first find the best trajectory among several candidates represented by cubic splines [55]. Each trajectory was then optimized using the method of [5].

Optimal trajectory planning on a constrained path has been first studied by Luh who modeled the path with polynomial functions, and assumed worst case constant bounds on velocity and acceleration [47]. A more realistic solution was proposed independently by Bobrow et al. [5] and by Shin and McKay [67] for computing the time-optimal trajectory of the manipulator tip along a specified path. Their results were extended by Prinz to compute the optimal starting point of the trajectory [54]. Shiller solved the point-to-point problem, subject to grasping force and pay-load acceleration constraints in [63], and showed the presence of singular points and arcs along the optimal path in [66]. Shiller also developed methods for computing the global optimal trajectories in [64] and [72].

Johnson solved the time-optimal obstacle avoidance by minimizing a penalty function of the distance between the manipulator and the obstacles [36]. More recently, Wang [75] and Shiller [62] have independently proposed methods for smoothing the control profiles by minimizing the energy consumed by the actuators.

In [73] and [74] Tominaga embedded his initial two-dimensional dynamic planning problem into a three-dimensional space, where penalty functions were used in a gradient descent to compute the optimal trajectory.

1.2.5 Missile Guidance

The problem of computing a control law for a missile aimed at a target is similar to the problem treated in this thesis. The control of missiles can be divided into strategic and tactical guidance. The first refers to ballistic missiles aimed at fixed targets on earth, whereas the latter refers to missiles aimed at moving targets. In strategic guidance, the location of the target is precisely known, and the trajectory to the target is computed during the boost phase. A tactical homing missile, on the other hand, is guided to the moving target during its entire flight until interception [79].

The tools used to compute a tactical guidance law are similar to those used in dynamic motion planning. In particular, the Proportional Navigation guidance law uses the homing triangle for computing the acceleration of a missile pursuing an evading target [79]. The homing triangle is defined by the pursuer, the target, and the point of interception. This control law makes the pursuer's acceleration normal to his path, and proportional to the rate of change of the line-of-sight vector to the target. This ensures that the angle between the velocity vector of the pursuer and the line-of-sight is constant. This in turn results in a homing triangle with fixed angles, that collapses to a point at interception. Missile guidance laws assume that the future trajectory of the target is completely defined [1], either analytically [30] or by a probabilistic model [71].

The method of the homing triangle is based on the concept of relative velocity used in this thesis. Moreover, the computation of the collision trajectory between a pursuer and an evader is the complementary problem to collision avoidance, and some of the techniques developed for missile guidance could be successfully applied to dynamic motion planning.

1.2.6 Differential Game Theory

A different perspective on dynamic planning is found in the literature on differential games. These results are only concerned with aggressive, or competitive, obstacles, as opposed to the current literature in motion planning, where the obstacles are indifferent, i.e. they do not change their trajectory, or cooperative, i.e. they help the avoidance maneuver by modifying their own trajectory.

The computation of the optimal trajectories for a pursuer and an evader has led Isaac to the development of the theory of differential games [34] and [3]. In its simplest formulation, a differential game consists of the computation of a trajectory for each of the two players, the pursuer and the evader, for optimally satisfying the respective goals. The pursuer's goal is to minimize its distance from the evader, while the evader goal is to maximize it [1]. This is an example of a competitive plan.

This formalism has been extended to several situations, including multiple target games [68] and robust control [56]. The problem of two players, each with its own target, is discussed in [68]. In [29], sufficient conditions are given for the game's termination. In [31] the qualitative solutions of a class of two-target games are computed in terms of regions of secured outcome. The quantitative solutions of other two-target games are given in [18] and [6] .

1.2.7 Conclusion

Dynamic motion planning is a relatively recent subject whose complexity has been often removed with simplifying assumptions. Some of the proposed solutions require that obstacles be slower than the robot, assume a fixed path, and yet other solutions do not satisfy the robot dynamics.

These simplifications generate trajectories that are not as efficient as those computed by simultaneously satisfying both kinematic and dynamic constraints. Thus the need for a better approach to dynamic motion planning.

1.3 Objective

This thesis focuses on the development of an approach for computing a *trajectory* of a robot moving between two given end-points that avoids moving obstacles, satisfies robot dynamics and actuator limits, and optimizes motion time. To make the problem computationally tractable, the following assumptions are made:

- the environment is restricted to the plane,
- the robot and the obstacles are modeled by circles,
- the information about the environment is complete.

With these assumptions, the approach can be extended to articulated robotic manipulators moving in a time varying environment by representing the manipulator as a point in its configuration space. Bounds on the controls become then state dependent constraints on the motion of that point. Circular objects can be considered as elementary parts of more complex planar shapes representable by a hierarchy of circles. Because of the assumption of complete information about the environment, the positions and velocities of the elements in the environment are either known a priori, or sensed in real time.

Mathematically, the planning problem consists of computing the control $u(t) \in \mathcal{U}$ in $t_0 \leq t \leq t_f$ that minimizes the performance index

$$J = \phi(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t)dt$$

subject to the two sets of constraints:

- **kinematic constraints**:
  - initial manifold:
    $$\Gamma(\mathbf{x}(t_0), t_0) = 0$$
  - final manifold:
    $$\Omega(\mathbf{x}(t_f), t_f) = 0$$

- obstacles:

$$\Psi : \bigcup_{i=1}^{n} [S_i(\mathbf{x}(t), t) = 0]$$

- **dynamic constraints**:

  - robot dynamics:

$$\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$

  - actuator constraints:

$$u_i(min) \leq u_i \leq u_i(max)$$

This is a difficult problem for which analytical solutions are not available. In this thesis a two-step approach is presented: first an approximate trajectory is computed, which is then refined with a dynamic optimization.

1.4 Outline of the Approach

By separating the constraints of the problem into:

1. *kinematic constraints*, imposed by the avoidance of the obstacles and by the given end points, and

2. *dynamic constraints*, imposed by robot dynamics and actuator constraints,

the complete problem of motion planning can be decomposed in the two separate problems: the *Kinematic* problem and the *Dynamic* problem. The kinematic problem consists of finding a trajectory that takes into account position and velocity of the obstacles as well as an approximation of the dynamic constraints of the robot. The dynamic problem consists of computing an optimal trajectory that satisfies the full set of kinematic and dynamic constraints, and is in a neighborhood of the solution to the kinematic problem.

Therefore, the approach to the solution of the complete motion planning problem consists of two steps. The first step computes a *kinematic* trajectory that solves the kinematic problem. The second step uses a dynamic optimization to optimize motion time subject to the dynamic constraints, using the kinematic trajectory as an initial guess.

The kinematic trajectory consists of a sequence of straight line segments computed using the concept of **Velocity Obstacle** (VO). The velocity obstacle is the set of velocities of the robot that will cause a future collision with an obstacle. Therefore, a safe trajectory can be computed by assuring that the velocity of the robot is outside the velocity obstacle at all times, if such velocities exist. Avoiding the obstacles guarantees the survival of the robot, but it contributes only indirectly towards achieving other goals and tasks assigned to the robot, such as reaching a target position, efficiently performing a task, and so on. The moving robot has thus two basic goals: the first is to survive, and the second is to achieve as many of its assigned tasks as possible. Since dynamic motion planning is an intractable problem [58], there is no guarantee that any of the goals can be achieved. Then, the robot can only organize its goals in a hierarchy and try to achieve each one of them according to their relative importance.

The velocity obstacle model provides a compact representation of the robot options for achieving its various goals. The survival goal is satisfied by choosing velocities outside VO, the target can be reached by choosing among the avoidance velocities those pointing towards the target, and the efficiency of the motion can be achieved by selecting the fastest velocities available. Furthermore, other heuristics can be embedded in the VO formalism, such as selecting the sequence with which the obstacles are to be avoided, and choosing the relative position of the robot with respect to the obstacles. For these reasons, the velocity obstacle represents a powerful tool for using heuristics in the solution of the dynamic planning problem.

The second step of the approach uses a **Dynamic Optimization** to minimize the motion time subject to robot's dynamics, actuator limits and state inequality constraints due to the moving obstacles. The initial guess of the controls required by the optimization is computed by smoothing the kinematic trajectory with a spline and using the inverse dynamics of the robot. The optimization itself is based on a steepest descent method [10], [9], modified to include the state inequality constraints due to the obstacles. The obstacle constraints are considered by transforming them into state-dependent control constraints. The numerical solution is shown to satisfy the set of necessary conditions derived from Pontryagin's Minimum Principle. Since the dynamic optimization can only find a *local minimum*, the quality of the solution depends heavily on the initial guess. A good choice of a kinematic trajectory is essential for the successful computation of the dynamic trajectory. Thus, the hierarchy of goals used for computing the kinematic trajectory ultimately determines the shape of the optimal solution.

In conclusion, the approach developed in this thesis is amenable to a wide range of applications, depending on the relative weight given to the solution of the kinematic and of the dynamic problem. A planner for real time applications, for instance, would need fast computations to adapt to the changing environment. Thus it would use heuristics to select the kinematic trajectory and would adjust the controls using the dynamic optimization to guarantee the feasibility of the trajectory over a short time horizon. An off-line planner, instead, could accept a long computation time and would use a global search over the velocity obstacles to find the trajectory with the most suitable structure. This optimal kinematic trajectory would then be refined by the dynamic optimization to minimize motion time and dynamic errors.

1.5 Organization

Chapter 2 presents the kinematic problem of avoiding moving obstacles. The tool used in this computation is the Velocity Obstacle, which represents the set of colliding velocities with the obstacles at some future time. The constraints and the objectives of the problem are mapped into the velocity obstacle, thus permitting the computation of an avoidance maneuver that satisfies multiple goals. The avoidance maneuver alone does not constitute the complete trajectory, but is its elementary component. The complete kinematic trajectory consists of a sequence of avoidance maneuvers selected according to the hierarchy of goals. This chapter concludes with examples of single avoidance maneuvers and complete trajectories computed using the heuristics and a global search over the velocity space.

Chapter 3 describes the dynamic optimization algorithm for the computation of the time optimal trajectory between given end points, subject to robot's dynamics and actuator limits. The algorithm iteratively computes a set of controls that satisfy the necessary optimality conditions stated by Pontryagin Minimum Principle. The state constraints given by the moving obstacles are satisfied by transforming them into control constraints for the dynamic optimization so that the robot only moves on the boundary of the obstacles. Several examples of optimal trajectories computed by the dynamic optimization are presented at the end of the chapter.

Chapter 4 presents several examples of trajectories in a time-varying environment. In this chapter, the kinematic trajectories for a few test cases are computed using different heuristic methods and then the trajectories are optimized with the method of Chapter 3.

Chapter 5 concludes this dissertation and presents recommendation for future research.

Chapter 2

The Kinematic Problem

2.1 Introduction

As mentioned earlier in Chapter 1, motion planning in time-varying environments is an intractable problem, and therefore a planner must rely on heuristics to compute the trajectory of the robot. The foundation of such heuristics is a hierarchy of goals which are satisfied in order of decreasing priority. The survival of the robot is the goal with the highest priority, followed, in general, by reaching the target, minimizing motion time, choosing the structure of the trajectory, and so on. Since hierarchical planning does not guarantee that a solution satisfies any of the goals, it is essential that candidate solutions be rapidly computed and tested in order to select the one satisfying the largest number of goals.

This chapter presents an efficient method for computing the trajectories satisfying the hierarchical goals. This method is based on the concept of **Velocity Obstacle** which is an extension of the Configuration Space Obstacles [46] to a time-varying environment. At each time instant, the velocity obstacle identifies the velocities of the robot that will cause a collision between the robot and the environment at some future time. The velocity obstacle represents geometrically some of the hierarchical goals that a planner may consider. For example, by choosing the velocity of the robot outside the velocity obstacle, the planner guarantees the survival of the robot. By selecting the velocity pointing towards the goal, the robot will reach the target, and by choosing the highest among these velocities, a minimum time trajectory can be achieved.

The velocity obstacle is computed by solving the *Kinematic Problem* associated with the given time-varying environment. In this thesis, the kinematic problem refers to the avoidance of static and moving obstacles, since the robot is required to avoid all obstacles, and to reach the target. In a time-varying environment, collision avoidance depends on the robot dynamics, and therefore the problem constraints should include the appropriate dynamic constraints. Here, to meet the speed requirement of a heuristic planner, dynamic constraints are approximated with fixed bounds on the acceleration.

This chapter is organized as follows. First, the concept of velocity obstacle is introduced by using it to compute the avoidance velocities of a single obstacle on the plane. Next, in Section 2.3, the velocity obstacle due to several moving obstacles is computed. Then, Section 2.4 describes the use of the velocity obstacle for choosing the structure of the trajectory, i.e. for computing the type of avoidance maneuvers and the possible contact points with the obstacles. Next, Section 2.5 presents the algorithm for computing the velocity obstacle and determines its computational complexity and completeness. The discussion of planar avoidance maneuvers is concluded in Section 2.6 with an example of avoidance of multiple moving obstacles. Section 2.7 briefly extends the concept of velocity obstacle to the three dimensional space, and gives an example of spatial collision avoidance. Section 2.8 presents examples of the solution of the kinematic problem, i.e. the computation of complete trajectories that avoid the obstacles and reach the goal. These trajectories are computed using heuristic and global search methods, and their relative characteristics are discussed.

Finally, the last section summarizes the main results of this chapter.

2.2 Planar Velocity Obstacles

The solution of the time-varying motion planning problem should be computed in the state space of the robot and of the obstacles. A compact representation of the state space is achieved by attaching the velocity vectors to the location of the corresponding element in the environment. This representation is often referred to as the *Velocity Space*, i.e. the *Tangent Bundle* $T\Re^n$ to the underlying Euclidean Space $\Re^n$ ($n = 2, 3$) [11]. The use of position and velocity information in the same representation allows the direct use of computational geometry tools for solving the motion planning problem.

A versatile tool for motion planning in the velocity space is the **Velocity Obstacle** (VO). For given states of the robot and of the obstacles, the VO identifies the set of robot velocities that, under certain conditions, will avoid all future collisions. It is the instantaneous map of all feasible avoidance maneuvers into a set of geometric shapes. Each obstacle in the environment corresponds to one VO. The VO of multiple obstacles is computed by combining the single VO's into a *Multiple* Velocity Obstacle. The computation of the VO assumes that all velocities are defined with respect to a common inertial frame, and that complete information is available about the time-varying environment.

Without loss of generality, robot and obstacles can be represented by circles. This representation is sufficient for obstacles with simple shapes or for gross motion planning among obstacles with complex shapes. Fine motion planning can be carried out by approximating the obstacles, at the desired resolution, with a hierarchy of circles [22], which are then treated as separate obstacles. Each object in the environment,

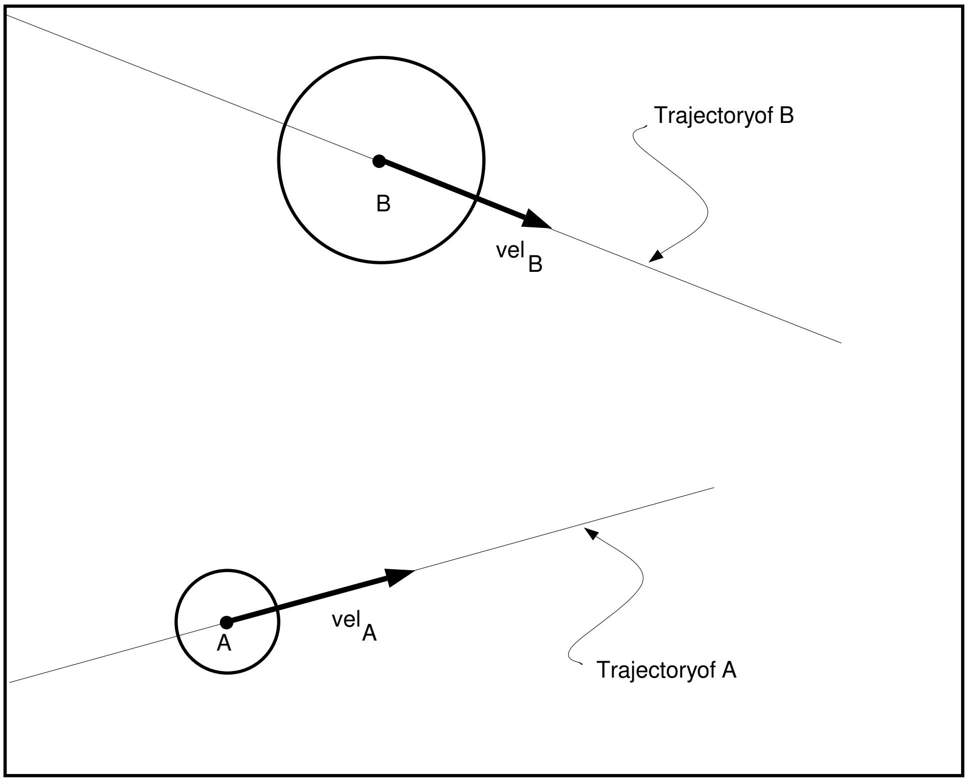

Figure 2.1: Scenario for the planar case.

either robot or obstacle, is then characterized by the triplet:

$$\mathbf{p}(t) = (\mathbf{x}_p(t), \mathbf{v}_p(t), r_p)$$

where $\mathbf{x}_p(t)$ is the origin of a reference frame fixed to the object, with the same orientation of the inertial reference, $\mathbf{v}_p(t)$ is the velocity of the object, and r_p is its radius. In the following, t_0 represents the time at which VO is computed and at which the avoidance maneuver is carried out.

The concept of VO is illustrated using the scenario of Figure 2.1, where two circular objects, **A** and **B**, are shown at time t_0 with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$. **A** represents the robot and **B** is the obstacle. The velocities and positions of **A** and **B** have been chosen so that **A** and **B** will collide at time $t_1 > t_0$, provided that their velocities do not change between t_0 and t_1. It is therefore necessary to modify the

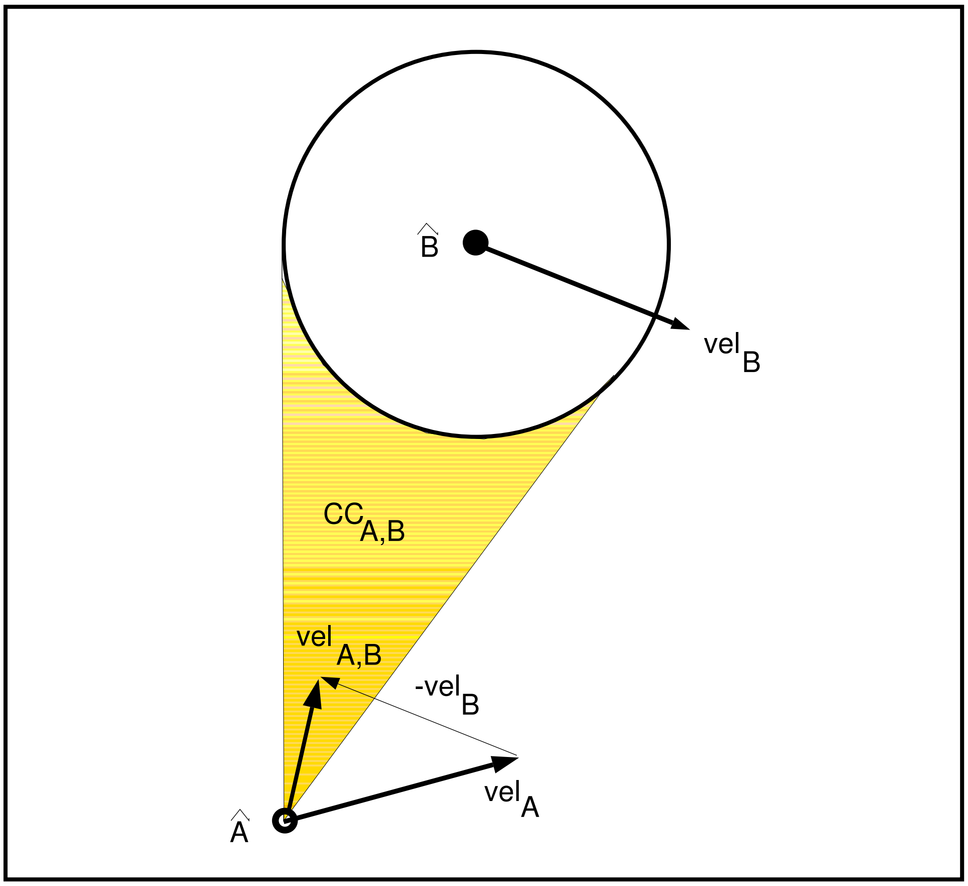

Figure 2.2: Geometry of the planar maneuver.

velocity of robot **A** so that it avoids **B**. The VO identifies the velocities of **A**, $\mathbf{v}_A$, that permit the desired avoidance.

The computation of VO consists of the following steps. First, **A** is reduced to the point $\widehat{\mathbf{A}}$ and **B** is enlarged by the radius of **A** to $\hat{\mathbf{B}}$, to produce the *Configuration Space Obstacle* of **A** due to **B** [44]. Then $\hat{\mathbf{B}}$ is considered stationary so that $\widehat{\mathbf{A}}$ moves with the relative velocity:

$$\mathbf{v}_{A,B} = \mathbf{v}_A - \mathbf{v}_B \tag{2.1}$$

on the relative trajectory, $trj_{A,B}$:

$$trj_{A,B} = \{(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \mid \mathbf{x}(t_0) = \mathbf{x}_0, \dot{\mathbf{x}}(t_0) = \mathbf{v}_{A,B}\} \tag{2.2}$$

It is convenient to use the relative trajectory, $trj_{A,B}$ for computing the collision between $\mathbf{A}$ and $\mathbf{B}$, since now the test for collision is reduced to testing the intersection of a line and a circle. Assuming that velocities $\mathbf{v}_A$ and $\mathbf{v}_B$ do not change, $\mathbf{A}$ and $\mathbf{B}$ will collide when:

$$trj_{A,B} \cap \hat{\mathbf{B}} \neq \emptyset \tag{2.3}$$

The relative velocity $\mathbf{v}_{A,B}$ transforms the time-varying planning problem into a static problem. In fact, the computation of a maneuver that avoids disk $\mathbf{B}$ is equivalent to the computation of the avoidance of the static obstacle $\hat{\mathbf{B}}$ by $\widehat{\mathbf{A}}$, moving at the relative velocity $\mathbf{v}_{A,B}$.

By using $\mathbf{v}_{A,B}$ as a parameter, condition (2.3) can be used to define the set of collision trajectories between $\widehat{\mathbf{A}}$ and $\hat{\mathbf{B}}$. This set is called the *Relative Collision Cone* $\mathbf{CC}_{A,B}$:

$$\mathbf{CC}_{A,B} = \{trj_{A,B} \mid trj_{A,B} \cap \hat{\mathbf{B}} \neq \emptyset\} \tag{2.4}$$

The cone is the planar sector with apex in $\widehat{\mathbf{A}}$, bounded by the two tangents from $\widehat{\mathbf{A}}$ to $\hat{\mathbf{B}}$, as shown in Figure 2.2. Since the underlying space is the Velocity Space, $\mathbf{CC}_{A,B}$ represents both trajectories $trj_{A,B}$ and relative velocities $\mathbf{v}_{A,B}$. Any relative velocity of $\mathbf{A}$, $\mathbf{v}_{A,B}$, remaining within $\mathbf{CC}_{A,B}$, is guaranteed, over time, to cause a collision between $\mathbf{A}$ and $\mathbf{B}$.

Equation (2.4) defines the collision condition for a single observation of the environment. It can be repeated when the environment is not predictable, thus replacing the velocity of $\mathbf{B}$ in the computation of VO, with the latest available measurement.

The relative collision cone defines *indirectly* the set of absolute colliding velocities of $\mathbf{A}$, $\mathbf{v}_A$. For the purpose of computing the VO, it is more convenient to represent this set directly, as a preliminary step to the computation of the VO of multiple obstacle, since relative velocities of different obstacles cannot be combined. A representation that is invariant of the number of obstacles is the *Absolute Collision Cone*, $\mathbf{CC}_B$,

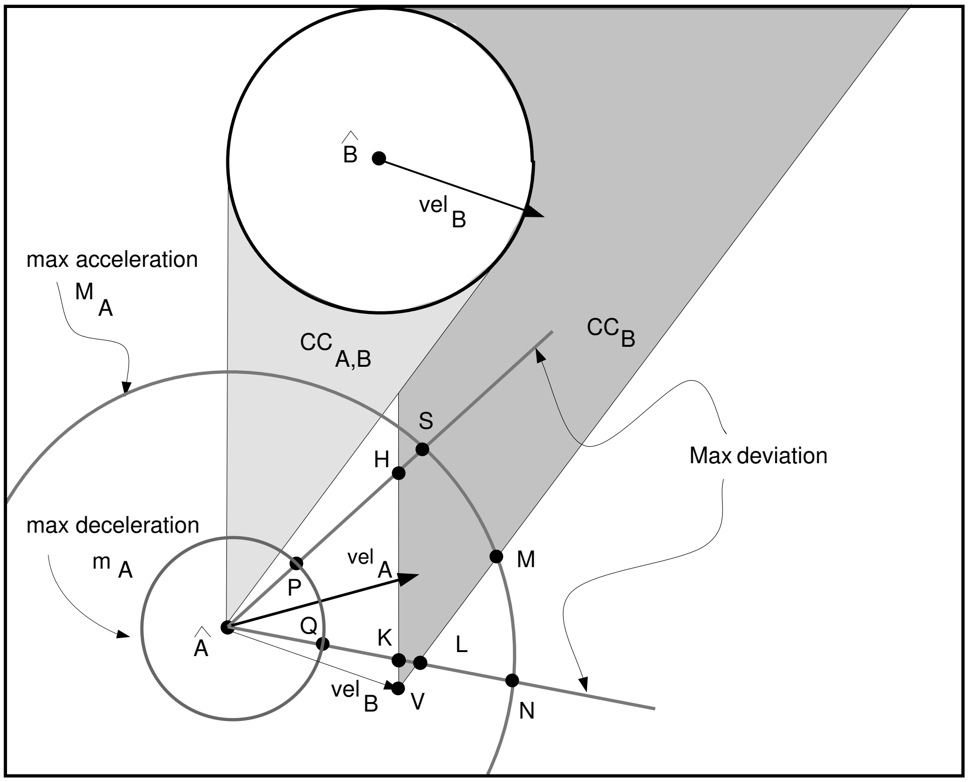

Figure 2.3: Construction of the velocity obstacle.

that identifies the set of absolute velocities, $\mathbf{v}_A$, causing a collision between **A** and **B**. The absolute collision cone is obtained by translating the relative collision cone $\mathbf{CC}_{A,B}$ from its initial position with apex in $\widehat{A}$, of the velocity of **B**, $\mathbf{v}_B$, to $\mathbf{CC}_B$ with apex in **V**, as shown in Figure 2.3 [23]. Thus **A** will collide with **B** if the tip of $\mathbf{v}_A$ is inside the cone $\mathbf{CC_B}$. For the case shown in Figure 2.3, **A** will collide with **B** since the tip of vector $\mathbf{v}_A$ is inside the velocity obstacle $\mathbf{CC}_B$. Note that the absolute velocity cone of a stationary obstacle is identical to its relative velocity cone.

At each point in the state space, i.e. for each pair $(\mathbf{x}_A(t_0), \mathbf{v}_A(t_0))$, the velocities achievable by **A** are determined by its dynamics and by the limits on its actuators. In solving the kinematic problem, these dynamic constraints are approximated by assuming that the achievable velocities are enclosed in a bounded set called the **Fea-**

sible Velocity set of **A**, $\mathbf{FV}_A(t_0)$, represented by the circular segment with vertices **PQNS** shown in Figure 2.3. This set is computed by imposing limits on the acceleration over a given interval Δt. Thus, the set $\mathbf{FV}_A(t_0)$ is bounded by two circles centered in **A**, with radius representing the maximum and minimum achievable velocity, and a cone representing the maximum achievable turning angle, over the interval Δt:

$$\begin{aligned} \mathbf{FV}_A(t_0) \quad = \quad & \{\mathbf{v}_A(t) \mid \\ & (\mathbf{x}(t_0) = \mathbf{x}_A, \mathbf{v}(t_0) = \mathbf{v}_A),\ \ddot{\mathbf{x}}_{min} \leq \ddot{\mathbf{x}}_A(t) \leq \ddot{\mathbf{x}}_{max},\ t_0 < t < t_0 + \Delta t\} \end{aligned} \tag{2.5}$$

In Figure 2.3, the set $\mathbf{FV}_A(t_0)$ is bounded by the circles $\mathbf{M}_A$ and $\mathbf{m}_A$, and the direction change is bounded by the lines $\widehat{\mathbf{AS}}$ and $\widehat{\mathbf{AN}}$. The interval Δt may be selected based on the knowledge of the environment and on the motions of the obstacles.

The Absolute Collision Cone partitions the set of all velocities of **A**, $\mathbf{v}_A$, into the two sets of collision and avoidance velocities. Only velocities that are physically achievable by **A**, i.e. included in $\mathbf{FV}_A$, have any significance in planning the trajectory.

Recalling the definition of the collision velocity set, $\mathbf{CC}_B$, and of the feasible velocity set, $\mathbf{FV}_A$, at time t_0, it follows that the **Velocity Obstacle** of **A** due to **B** at t_0, $\mathbf{VO}_B(t_0)$, is defined as the intersection of the absolute collision cone $\mathbf{CC}_B(t_0)$ with the feasible velocity set $\mathbf{FV}_A(t_0)$, and it consists of all feasible velocities causing a collision between **A** and **B**:

$$\mathbf{VO}_B(t_0) = \mathbf{CC}_B(t_0) \cap \mathbf{FV}_A(t_0) \tag{2.6}$$

The set difference between $\mathbf{FV}_A(t_0)$ and $\mathbf{VO}_B(t_0)$ is defined as the **Safe Velocity** set:

$$\mathbf{SV}_A(t_0) = \mathbf{FV}_A(t_0) \ominus \mathbf{VO}_B(t_0) \tag{2.7}$$

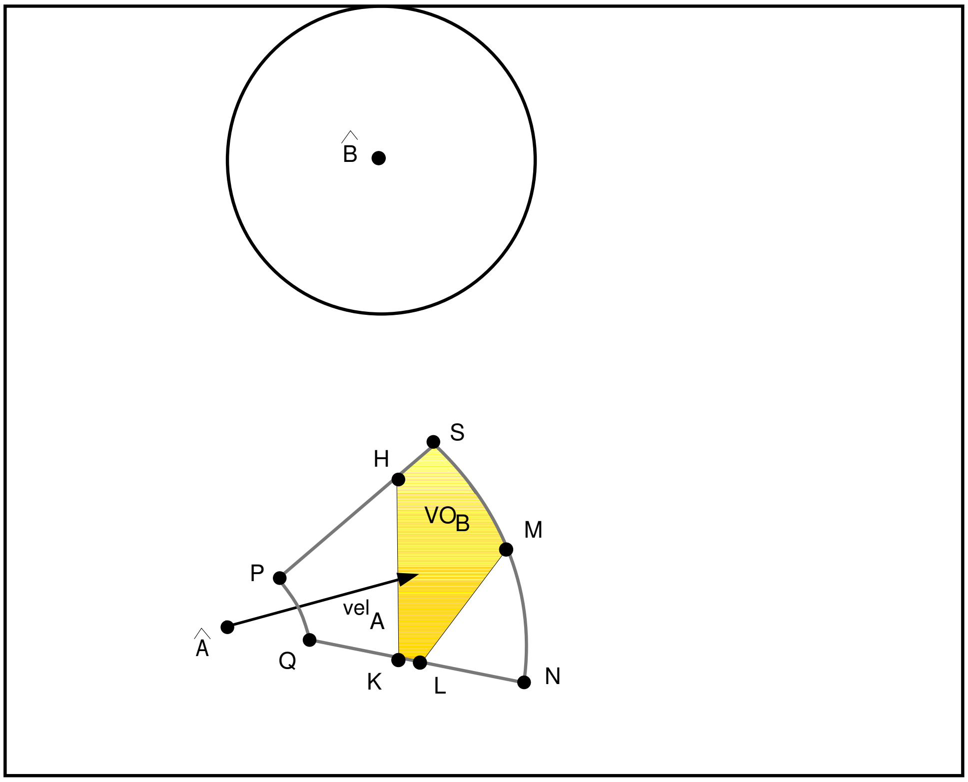

Figure 2.4: The safe velocity set $\mathbf{FV}_A$.

where $\ominus$ indicates the operation of set difference. This set consists of all feasible velocities avoiding the collisions with disk **B**. $\mathbf{SV}_A$ is shown as the clear areas in Figure 2.4, marked by vertices **PQKH** and **LMN**. The grey area in Figure 2.4, with vertices **KLMSH** is the velocity obstacle due to **B**, $\mathbf{VO}_B$.

Each velocity, $\mathbf{v}_A^*$, whose tip is inside the safe velocity set $\mathbf{SV}_A$ represents a collision avoiding maneuver at time t_0, or:

$$(\mathbf{B}(t) \cap \mathbf{A}(t) = \emptyset) \quad \text{if} \quad (\mathbf{v}_A^*(t_0) \in \mathbf{SV}_A(t_0)) \quad \text{for } t \geq t_0 \tag{2.8}$$

The avoidance maneuver is guaranteed to avoid obstacle **B** only if **B** maintains its current velocity between t_0 and t_1. Selecting the velocity $\mathbf{v}_A^*$ to be on the boundaries of the velocity obstacle $\mathbf{VO}_B(t_0)$ would result in **A** grazing **B**. Selecting $\mathbf{v}_A^*$ away from

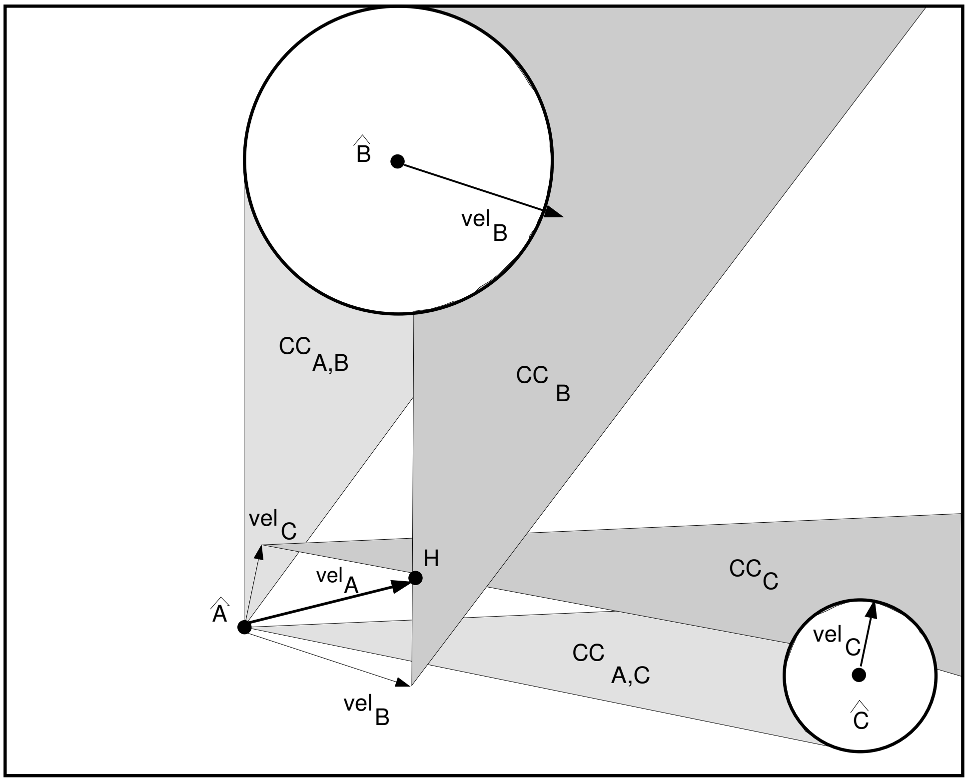

Figure 2.5: Absolute collision cones for multiple obstacles.

these boundaries would ensure some safety margin between **A** and **B**. The derivation of VO can be summarized in the following proposition:

Proposition 2.1: All velocities of **A**, $\mathbf{v}_A(t_0^+) \in \mathbf{SV}_A(t_0)$, defined by the Velocity Obstacle $\mathbf{VO}_A(t_0)$, are guaranteed to avoid **B** in $t_0 \leq t \leq \infty$, and to satisfy the dynamics of **A**, defined by the Feasible Velocity set $\mathbf{FV}_A(t_0)$. The Safe Velocity set, $\mathbf{SV}_A(t_0)$, is defined by the set difference:

$$\mathbf{SV}_A(t) = \mathbf{FV}_A(t) \ominus \mathbf{VO}_A(t)$$

$\square$

2.3 Multiple Velocity Obstacles

For the case of multiple obstacles, the absolute collision cones of each obstacle are combined into the Multiple Velocity Obstacle **MVO**. Then, the avoidance velocities $\mathbf{v}_A^*$ must be within the set difference of $\mathbf{FV}_A$ and **MVO**:

$$\begin{aligned} \{\mathbf{v}_A^*(t_0) &\in \mathbf{FV}_A(t_0) \ominus \mathbf{MVO}(t_0)\} \\ \mathbf{MVO}(t_0) &= \cup_{i=1}^m \mathbf{VO}_i(t_0) \end{aligned} \tag{2.9}$$

where m is the number of obstacles.

Figure 2.5 shows two obstacles, **B** and **C**, their relative collision cones, $\mathbf{CC}_{A,B}$ and $\mathbf{CC}_{A,C}$, and their absolute collision cones $\mathbf{CC}_B$ and $\mathbf{CC}_C$. The absolute velocity $\mathbf{v}_A^*$, represented by segment $\widehat{\mathbf{AH}}$, corresponds to a maneuver whose trajectory will be tangent to both obstacles, **B** and **C**, since the tip of $\mathbf{v}^*$ is on the boundary of the absolute cones of both obstacles.

Figure 2.6 shows the safe velocity set $\mathbf{SV}_A$ due to obstacles **B** and **C**, the multiple velocity obstacle **MVO**, and the absolute cones $\mathbf{CC}_B$ and $\mathbf{CC}_C$. The safe velocity set consists of four subsets:

$$\mathbf{SV}_A = \bigcup_{i=1}^{4} \mathbf{SV}_A^i$$

identified by points **BQDH**, **PNML**, **FGK**, and **CJE**.

2.4 Structure of the Avoidance Maneuver

The previous section has presented a geometric approach for computing the velocities of **A** that will avoid **B**. This section refines that approach by examining the structure of an avoidance maneuver and by deriving conditions that determine *how* **A** will avoid **B**.

The motivation for examining the details of an avoidance maneuver rests on the need to map the goals of the motion planning hierarchy described in Section 1.4, into

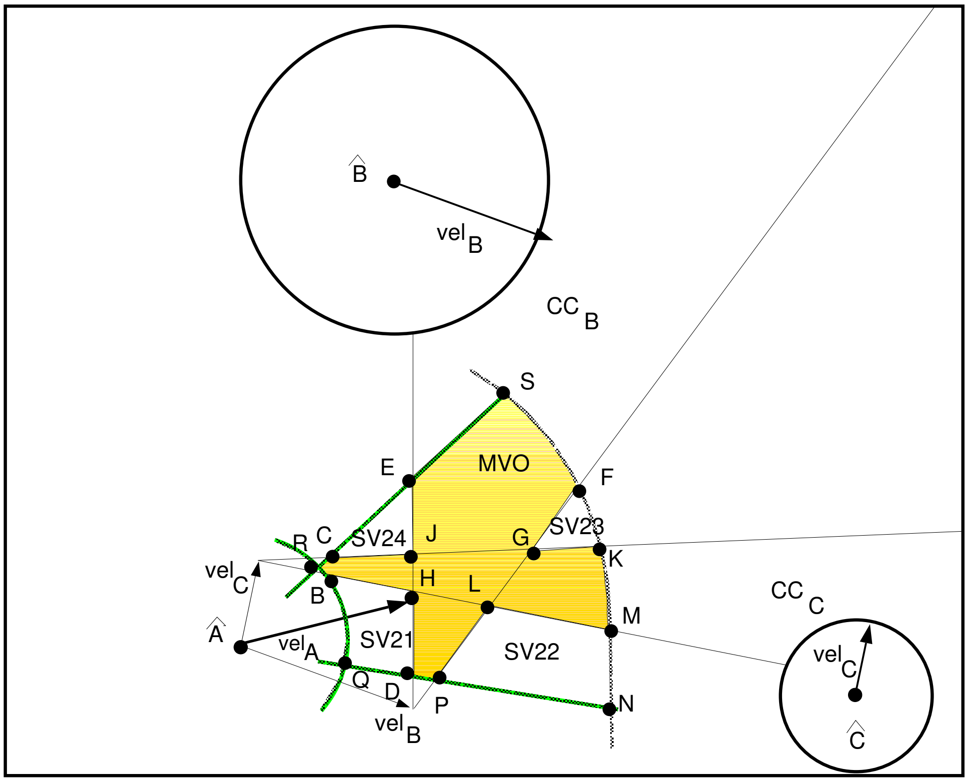

Figure 2.6: Safe velocity set with multiple avoidance.

well defined geometric conditions. The previous section was only concerned with the first, and most important goal of the hierarchy, namely the survival of the robot. If this goal is satisfied using the Velocity Obstacle approach, then it is natural to ask if the robot can do *better*, i.e. if it can satisfy lower priority goals. The secondary goals are achieved by giving the trajectory a particular structure, i.e. by requiring that the robot satisfies specific geometric relations with the obstacles. Once the structure has been identified, the planner can evaluate the feasibility of the secondary goals by assessing the risk of the maneuver and the uncertainty of the environment. In short, the structure of the avoidance maneuver is the key factor for developing the heuristics necessary for on-line motion planning in time-varying environments.

A brief example can help understand the process of mapping the secondary goals into the structure of an avoidance maneuver. When minimizing motion time is the second goal after survival, the robot will plan an *aggressive* trajectory that will attempt to pass in front of all the obstacles. This trajectory will consist of high velocity segments at which the robot will be close to its dynamic limits. The robot will reach rapidly its target, but it will take some risk and possibly miss the goal. In fact, if the obstacles move faster than anticipated, the robot may be unable to avoid the collision, since it is already using most of its maneuvering capability. A more conservative trajectory, on which the robot yields to some of the obstacles, may be more appropriate, although slower. In the time-varying environment represented by a highway segment, for example, an intelligent vehicle may decide whether to pass or give way to other vehicles, depending on their size, speed, and driving patterns.

In general then, the hierarchy of planning goals determines whether the robot should pass in front or give way to an obstacle. Then, each maneuver determines the range of avoidance velocities within the velocity obstacles representation. Finally, the secondary goals, such as minimizing motion time, reaching the target, or keeping a desired safety margin, select a particular avoidance velocity.

In the next sections, the velocity obstacle representation is used to implement this process by showing that each safe velocity set represents an avoidance maneuver passing in front or giving way to the obstacles. First the set of tangent velocities is computed, and then it is shown that the tangent velocity sets partition the safe velocity set $\mathbf{SV}_A$ into homogeneous subsets. Each subset generates only avoidance maneuvers of the same type of the tangency velocities forming its boundary.

2.4.1 Tangent Avoidance Maneuvers

To show that there exist a correspondence between the structure of the trajectory, i.e. the desired type of avoidance maneuvers, and the velocity obstacle, it is first

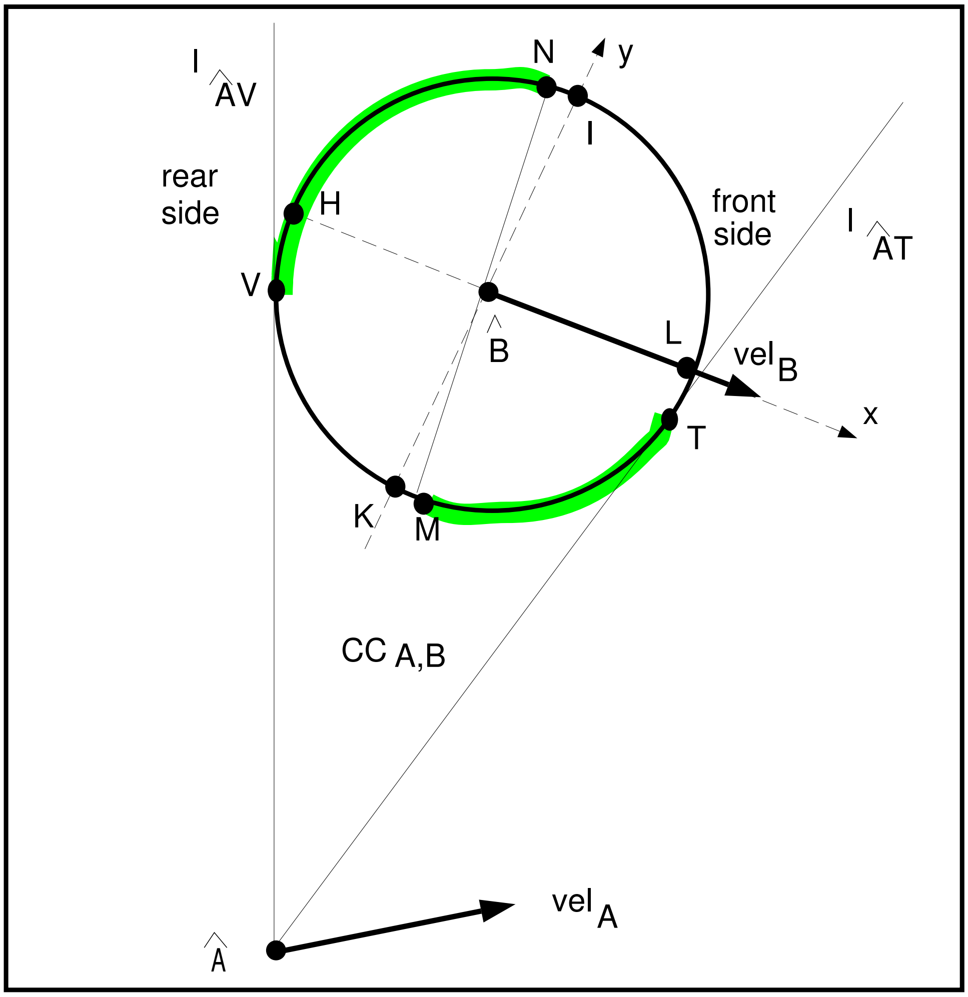

Figure 2.7: Grazing arcs in an avoidance maneuver.

necessary to define the set of avoidance maneuvers that are tangent to an obstacle. This is done by using the VO to compute the portions of the boundary of **B**, $\partial\mathbf{B}$, that are reachable by a tangent trajectory and their corresponding avoidance velocities.

The avoidance maneuvers of the robot have been described earlier as passing-in-front and giving-way. These maneuvers can be defined more rigorously by specifying the *front* and the *rear* sides of the obstacle. Figure 2.7 shows an example of a scenario consisting of robot **A** and obstacle **B**, each with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$, respectively. In order to partition $\partial\mathbf{B}$ into its front and rear semi-circles, a reference frame is defined on $\mathbf{B}$, with the X axis on the line supporting $\mathbf{v}_B$, origin on the center of $\mathbf{B}$ and positive X direction determined by $\mathbf{v}_B$. Then, the *front side* of $\mathbf{B}$, $\partial\mathbf{B}_f$, is defined as the semi-circle on the half-plane of positive X, and the *rear side* of $\mathbf{B}$, $\partial\mathbf{B}_r$, is the semi-circle on the half-plane of negative X. Of course $\partial\mathbf{B} = \partial\mathbf{B}_f \cup \partial\mathbf{B}_r$. In Figure 2.7, the front side of $\mathbf{B}$ is represented by the semi-circle $\mathbf{KLI}$, and the rear side of $\mathbf{B}$ is semi-circle $\mathbf{IHK}$, where $\mathbf{K}$ and $\mathbf{I}$ are the intersections of the Y axis with $\partial\mathbf{B}$, and $\mathbf{L}$ and $\mathbf{H}$ are the intersections of the X axis with $\partial\mathbf{B}$.

In Figure 2.7, the Relative Collision Cone between $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{CC}_{A,B}$ is bounded by the two lines $\mathbf{l}_{\widehat{AV}}$ and $\mathbf{l}_{\widehat{AT}}$. As mentioned earlier, the line $\mathbf{l}_{\widehat{AT}}$ represents the set of relative velocities of $\mathbf{A}$, $\mathbf{v}_{A,B}$, generating trajectories on which $\mathbf{A}$ is tangent to $\mathbf{B}$ at a point $\mathbf{P}^* \in \partial\mathbf{B}_f$, and represents the set of $\mathbf{v}_{A,B}$ taking $\mathbf{A}$ tangent to $\mathbf{B}$ at a point $\mathbf{P}^\# \in \partial\mathbf{B}_r$.

The objective now is to characterize the set of tangent velocities for the front side of $\mathbf{B}$, $\partial\mathbf{B}_f$, by computing the tangency points $\mathbf{P}^*$ and their corresponding tangent velocities $\mathbf{v}^*_{A,B}$:

$$\mathbf{v}^*_{A,B} \subset \mathbf{l}_{\widehat{AT}} \tag{2.10}$$

Clearly, the set of tangency points $\mathbf{P}^* \in \partial\mathbf{B}_f$ is a closed set $\mathcal{T}_f \subset \partial\mathbf{B}_f$, bounded by points $\mathbf{T}$ and $\mathbf{M}$. In fact, the set of relative tangent velocities $\mathbf{l}_{\widehat{AT}}$ is bounded by the two velocities with magnitude $|\mathbf{v}^0_{A,B}| = 0$ and $|\mathbf{v}^\infty_{A,B}| = \infty$, generating trajectories tangent to $\mathbf{B}$ at points $\mathbf{M}$ and $\mathbf{T}$, respectively. Then, since $\mathbf{l}_{\widehat{AT}}$ is compact, and tangency is a continuous map, it follows that $\mathcal{T}_f$ is also compact. Thus all velocities satisfying condition (2.10) generate trajectories that are tangent to $\mathbf{B}$ at a point $\mathbf{P}^* \in \mathbf{TM} \subset \partial\mathbf{B}_f$.

Similarly, velocities $\mathbf{v}^\#_{A,B} \in \mathbf{l}_{\widehat{AV}}$ generate trajectories that are tangent to $\mathbf{B}$ at a point $\mathbf{P}^\# \in \mathbf{VN}$ on the rear boundary of $\mathbf{B}$, $\partial\mathbf{B}_r$.

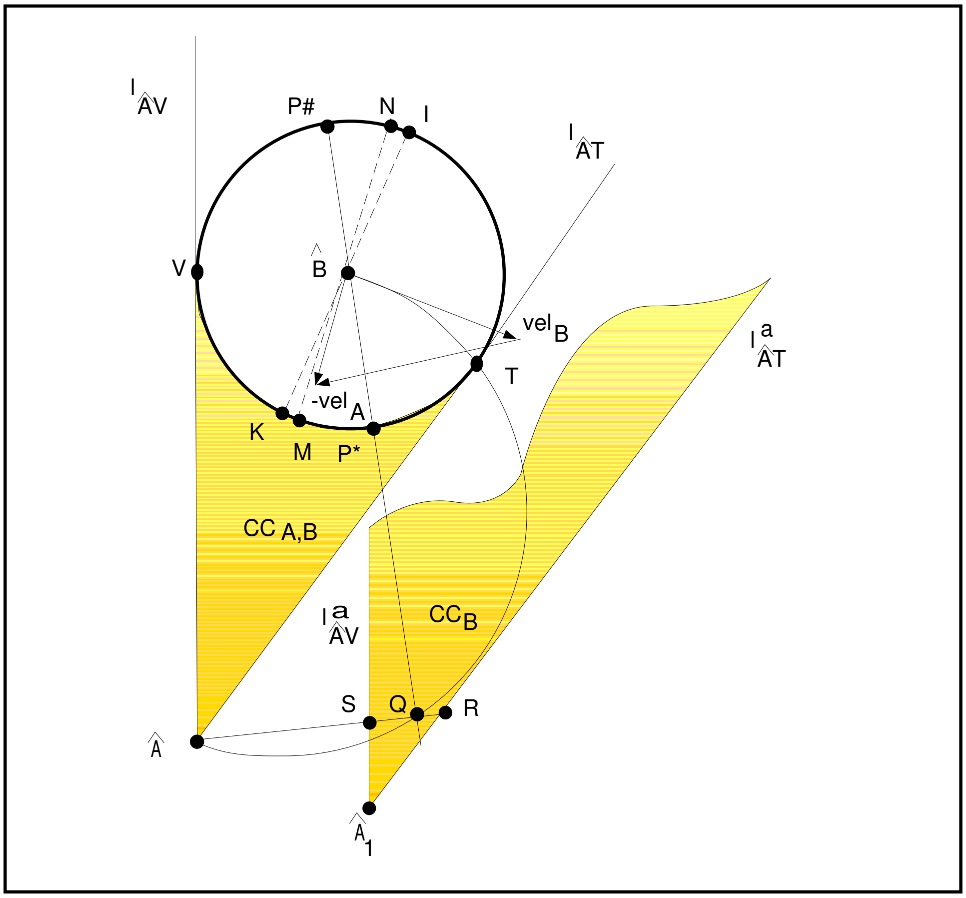

Figure 2.8: Contact points of tangent trajectories.

The position of any other tangency point $\mathbf{P}^* \neq \{\mathbf{T}, \mathbf{M}\}$ for a given relative velocity $\mathbf{v}^*_{A,B}$ can be computed with the geometric procedure that uses the robot $\mathbf{A}$, the obstacle $\mathbf{B}$, the Relative Collision Cone $\mathbf{l}_{\widehat{A}V}\widehat{\mathbf{A}}\mathbf{l}_{\widehat{A}T}$, and the Absolute Collision Cone $\mathbf{l}^a_{\widehat{A}V}\widehat{\mathbf{A}}_1 l^a_{\widehat{A}T}$, as shown in Figure 2.8.

The following derivation requires the use of the absolute velocity $\mathbf{v}_A$, from which the relative velocity follows as $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$. For a given point $\mathbf{P}^*$, the objective is to find the absolute velocity of $\mathbf{A}$, $\mathbf{v}^*_A$, the generates a trajectory tangent to $\mathbf{B}$ in $\mathbf{P}^*$. The direction of $\mathbf{v}_A$ is represented by $\angle \mathbf{v}_A$ and its magnitude by $|\mathbf{v}_A|$.

Since $\mathbf{v}_A$ is a tangent velocity, its direction is given by the line tangent to $\mathbf{B}$ at

$\mathbf{P}^*$. This direction can be mapped to $\widehat{\mathbf{A}}$ as follows. Since segment $\hat{\mathbf{B}}\mathbf{Q}$ is normal to the line tangent to $\mathbf{B}$ at $\mathbf{P}^*$, and angle $\widehat{\mathbf{A}}\mathbf{Q}\hat{\mathbf{B}}$ is a right angle, being inscribed in the semi-circle $\widehat{\mathbf{A}}\mathbf{T}\hat{\mathbf{B}}$, then the segment $\widehat{\mathbf{A}}\mathbf{Q}$ is parallel to the tangent to $\mathbf{B}$ in $\mathbf{P}^*$, i.e. $\angle \mathbf{v}_A^* = \angle \widehat{\mathbf{A}}\mathbf{Q}$.

The direction of the limit velocities $\mathbf{v}_{A,B}^{\infty}$ is given, as expected, by the segment $\widehat{\mathbf{A}}\mathbf{T}$, corresponding to the case of point $\mathbf{Q}$ coinciding with $\mathbf{T}$, i.e. $\mathbf{Q} \equiv \mathbf{T}$. Conversely, the direction of $\mathbf{v}_{A,B}^0$ is not specified when $\widehat{\mathbf{A}} \equiv \mathbf{Q}$, but the absolute velocity is clearly $\mathbf{v}_A^0 = \mathbf{v}_B$.

The magnitude of $\mathbf{v}_A^*$ is found using the fact that a velocity tangent to $\mathbf{B}$ must have its tip on $\mathbf{l}^a_{\hat{A}T}$, the front side of the absolute cone $\mathbf{l}^a_{\hat{A}V}\widehat{\mathbf{A}}\mathbf{l}^a_{\hat{A}T}$. Thus $|\mathbf{v}_A^*| = \widehat{\mathbf{A}}\mathbf{R}$.

The tangent velocity $\mathbf{v}_A^*$, producing a trajectory that is tangent to $\mathbf{B}$ at a point $\mathbf{P}^* \in \partial\mathbf{B}_f$, is:

$$\mathbf{v}_A^* = \widehat{\mathbf{A}}\mathbf{R} \tag{2.11}$$

Similarly, the velocity $\mathbf{v}_A^{\#}$ generating a trajectory tangent to $\mathbf{B}$ at $\mathbf{P}^{\#} \in \partial\mathbf{B}_r$ is:

$$\mathbf{v}_A^{\#} = \widehat{\mathbf{A}}\mathbf{S} \tag{2.12}$$

This discussion leads to the following lemma:

Lemma 2.1: The lines $\mathbf{l}^a_{\hat{A}V}$ and $\mathbf{l}^a_{\hat{A}T}$, boundary of the Absolute Cone $\mathbf{CC}_B$, represent the set of velocities tangent to the obstacle $\mathbf{B}$. In particular,

$$\begin{aligned} \mathbf{v}_A^* &\in \mathbf{l}^a_{\hat{A}T} \text{ is a } \textit{front} \text{ avoidance} \\ \mathbf{v}_A^{\#} &\in \mathbf{l}^a_{\hat{A}V} \text{ is a } \textit{rear} \text{ avoidance} \end{aligned}$$

$\square$

The trajectories grazing $\mathbf{B}$ on its front boundary are called *front avoidance maneuvers* and the trajectories grazing $\mathbf{B}$ on its rear boundary are called *rear avoidance*

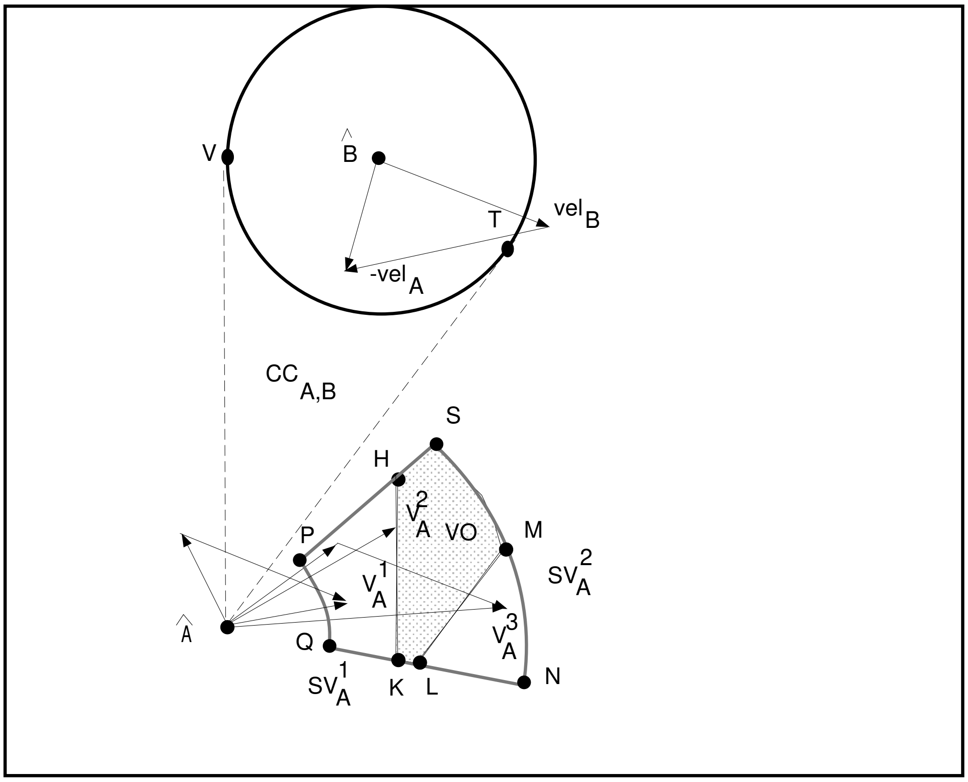

Figure 2.9: Front and rear avoidance in the Safe Velocity set.

maneuvers. From the lemma, it follows that the sides of the velocity obstacle that are in common with the boundary of the absolute cone $\mathbf{CC}_B$ represent a set of tangent avoidance velocities, such as segments **KH** and **LM**, shown in Figure 2.9.

2.4.2 Front and Rear Avoidance Maneuvers

The trajectories tangent to obstacle **B** are only a subset of the avoidance maneuvers identified by the safe velocity set $\mathbf{SV}_A$. The purpose of this section is to classify those velocities in $\mathbf{SV}_A$ that are not grazing **B**, as front or rear avoidance maneuvers.

Figure 2.9 shows the Safe Velocity set $\mathbf{SV}_A = \mathbf{SV}_A^1 \bigcup \mathbf{SV}_A^2$ for robot **A** due to obstacle **B**. The other sets shown are the Feasible Velocity set $\mathbf{FV}_A$, represented by the circular segment **PQNS**, and the Velocity Obstacle $\mathbf{VO}_B$, represented by

KLMSH. In general, the safe velocity set consists of disjoint subsets, as in this case $\mathbf{SV}_A^1$ and $\mathbf{SV}_A^2$ represented by **PQKH** and **LNM**.

Each subset of avoidance velocities $\mathbf{SV}_A^i$ $(i = 1, 2)$, includes on its boundary a segment of the boundary of the velocity obstacle. In this case, $\mathbf{KH} \subset \partial(\mathbf{SV}_A^1)$ represents the velocities tangent to $\partial\mathbf{B}_r$ and $\mathbf{LM} \subset \partial(\mathbf{SV}_A^2)$ represents the velocities tangent to $\partial\mathbf{B}_f$. As discussed in the previous section, the boundary of the velocity obstacle identifies the velocities that are tangent to **B**, such as $\mathbf{v}_A^2$ representing a rear avoidance maneuver. The segments of the boundary of $\mathbf{SV}_A$ in common with **VO**, i.e. **KH** and **LM**, separate the collision velocities from the avoidance velocities. The velocities $\mathbf{v}_A \in \mathbf{SV}_A^i$ $(i = 1, 2)$, whose tip is away from segments **KH** and **LM**, generate trajectories that will avoid **B** with a front or a rear maneuver, depending on whether the boundary of the respective $\mathbf{SV}_A^i$ $(i = 1, 2)$ includes segment **KH** or **LM**. For example, $\mathbf{v}_A^1 \in \mathbf{SV}_A^1$ represents a rear avoidance maneuver, since $\mathbf{KH} \subset \partial(\mathbf{SV}_A^1)$, and $\mathbf{v}_A^3 \in \mathbf{SV}_A^2$ represents a front avoidance maneuver, since $\mathbf{LM} \subset \partial(\mathbf{SV}_A^2)$.

Each safe velocity subset $\mathbf{SV}_A^i$ can then be labeled using the tangent velocity sets included in its boundary. In the example shown in Figure 2.9, $\mathbf{SV}_A^1$ can be classified as the set of rear avoidance maneuvers, and $\mathbf{SV}_A^2$ as the front avoidance set.

There are exceptions to this classification when the safe velocity set does not consist of disconnected convex sets. The velocity obstacle $\mathbf{VO}_B$ may intersect the feasible velocity set $\mathbf{FV}_A$ as shown in Figure 2.10, creating a single safe velocity set, equivalent to the rear avoidance set $\mathbf{SV}_A^1$ of Figure 2.9. Thus, in this case only rear avoidance maneuvers of **B** are feasible. Figure 2.11 shows a case in which $\mathbf{FV}_A$ is effectively divided into three sets: $\widehat{\mathbf{A}}\mathbf{KH}$ representing the rear avoidance maneuvers, **KLM**, representing the front avoidance maneuvers, and $\widehat{\mathbf{A}}\mathbf{KLN}$ representing velocity generating trajectories diverging from **B**. A typical scenario producing the safe velocity sets shown in Figure 2.11 consists of two vehicles, **A** and **B**, moving with

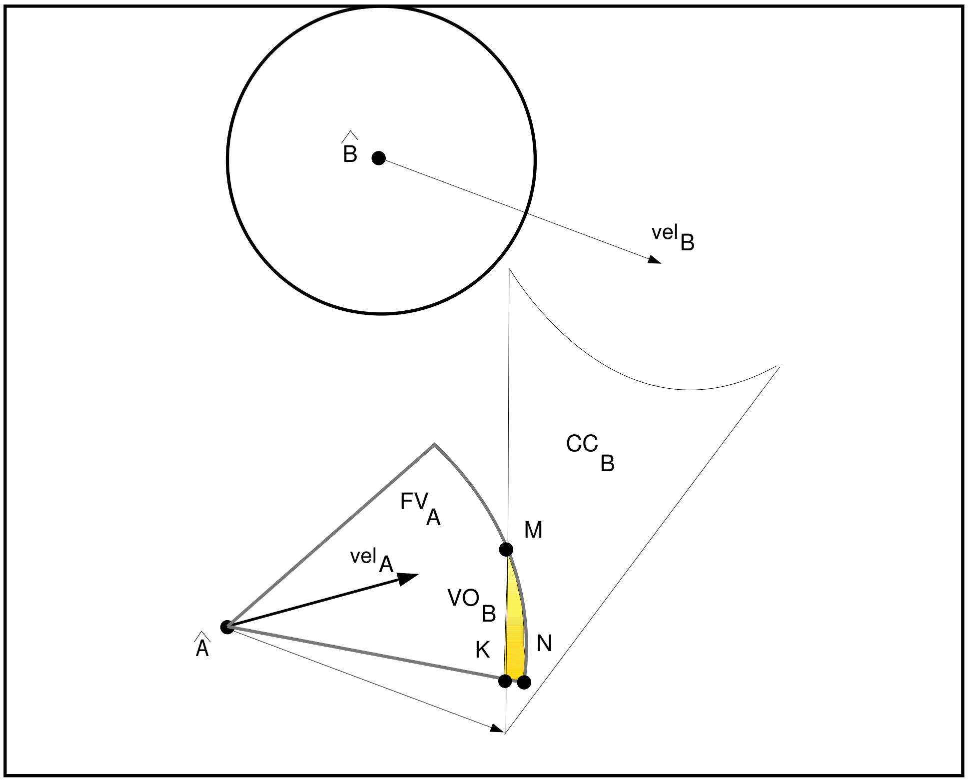

Figure 2.10: Example of single avoidance maneuver.

parallel trajectories on adjacent lanes of a freeway,

Another important case of non-convex safe velocity set is when **A** and **B** are heading towards each other or when **A** is overtaking **B**. In the first case, only a front avoidance maneuver is possible, in the second case, only a rear avoidance maneuver is possible. In both cases the trajectories can be further classified into *left* and *right* avoidance maneuvers, identified similarly to the front and rear maneuvers.

The classification described in this section can be summarized in the following lemma:

Lemma 2.2: The velocity obstacle $\mathbf{VO}_B$ partitions the set of feasible velocities of **A**, $\mathbf{FV}_A$, into one or more Safe Velocity subsets, $\mathbf{SV}_A^i$, defined by equation (2.7), each containing avoidance velocities $\mathbf{v}_A^i$ of a single type. These velocities generate

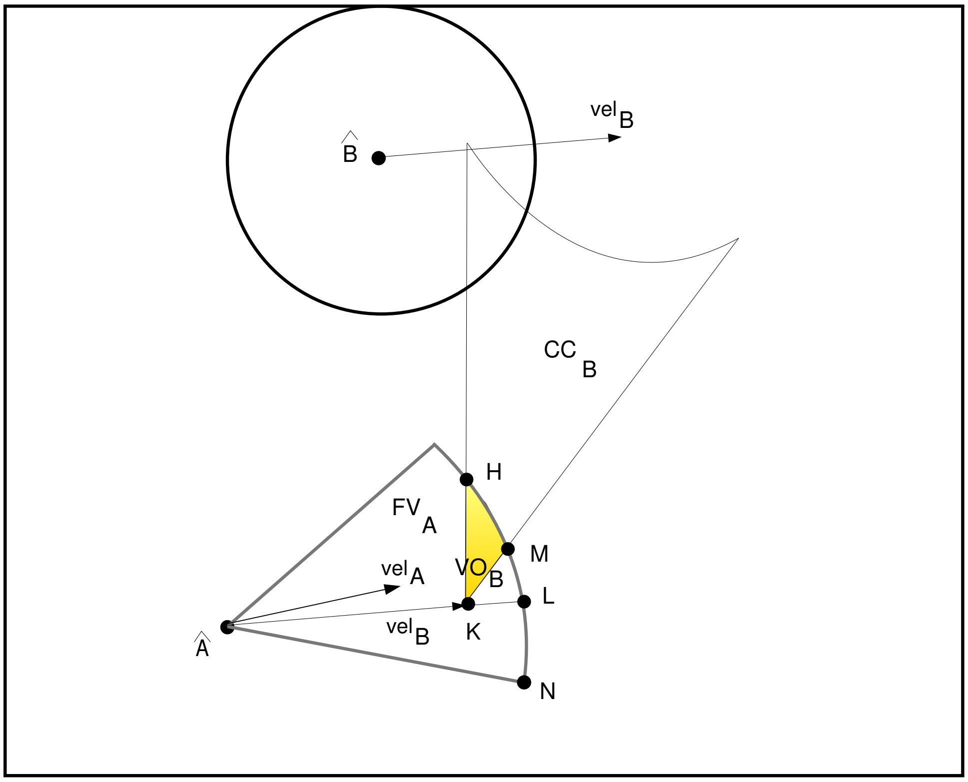

Figure 2.11: Example of non convex safety velocity set.

front or rear avoidance of obstacle $\mathbf{B}$, depending on whether the boundary of each $\mathbf{SV}_A^i$ includes a segment of the front or of the rear side of the absolute cone $\mathbf{CC}_B$. □

This classification is a powerful tool to identify the desired structure of the trajectory, in the case of a single obstacle. In the following section, the classification is extended to the avoidance of multiple obstacles.

2.4.3 Front and Rear Avoidance of Multiple Obstacles

The advantages of the classification just presented are more evident when planning the avoidance of m moving obstacles. In this case, the boundary of each subset $\mathbf{SV}_A^i$ consists of segments $\mathbf{S}_i$ from the boundaries of one or more velocity obstacles:

$$\partial(\mathbf{SV}_A^i) \supset \mathbf{S}_i \cup \mathbf{S}_{i+1} \cup \ldots \cup \mathbf{S}_l \;\; \text{with } 1 \leq i, l \leq m \tag{2.13}$$

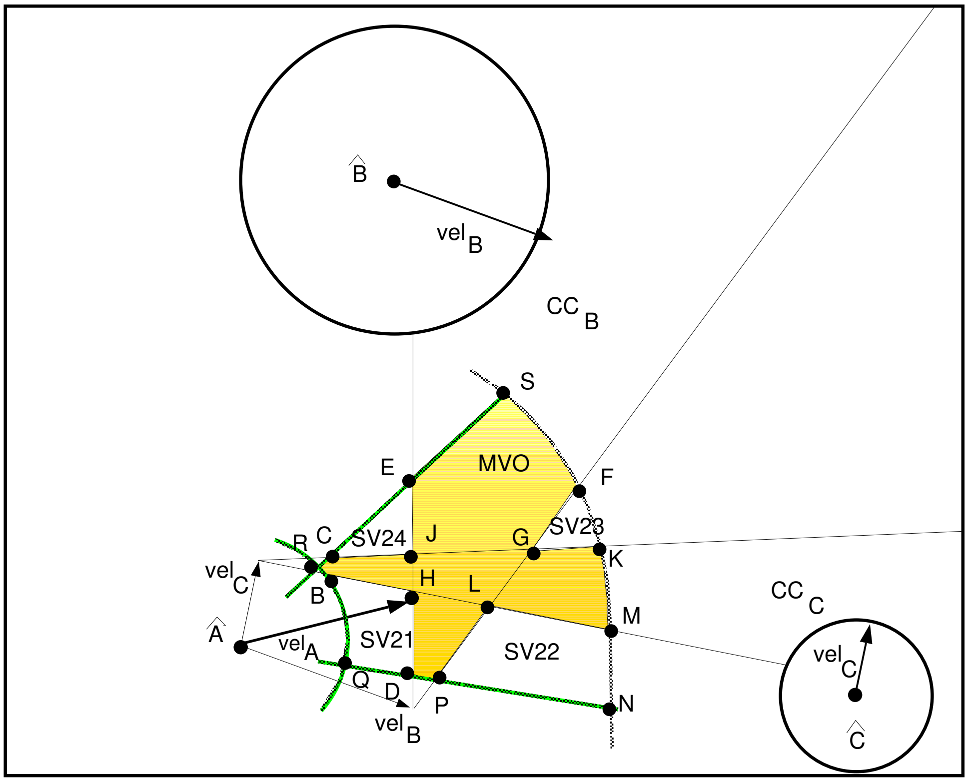

Figure 2.12: Safe velocity set with multiple avoidance.

with $\mathbf{S}_i \subset \partial(\mathbf{VO}^i)\;\; 1 \leq i \leq m$

Therefore, velocities within each $\mathbf{SV}_A^i$ will avoid the obstacles whose $\mathbf{VO}_i$ contributes to $\partial(\mathbf{SV}_A^i)$ with a front or a rear maneuver determined by the side of $\mathbf{VO}_i$ included in $\partial(\mathbf{SV}_A^i)$. Obstacles that are not contributing to the boundary of a $\mathbf{SV}_A^i$ are not on a collision trajectory with $\mathbf{A}$.

To illustrate the classification of the safe velocity sets, the example shown in Figure 2.5 and Figure 2.6 is duplicated in Figure 2.12. The safe set $\mathbf{SV}_A^1$ includes in its boundary segments $\mathbf{BH} \subset \partial(\mathbf{VO}_C)$ and $\mathbf{HD} \subset \partial(\mathbf{VO}_B)$. Furthermore, segment $\mathbf{BH}$ belongs to the rear side of $\mathbf{CC}_C$ and $\mathbf{HD}$ belongs to the rear side of $\mathbf{CC}_B$. Then, it follows from Lemma 2.2 that all velocities $\mathbf{v}_A \in \mathbf{SV}_A^1$ will generate rear avoidance maneuvers for both obstacles $\mathbf{B}$ and $\mathbf{C}$.

This and the previous two sections lead to the following theorem:

Theorem 2.1: Given robot $\mathbf{A}$ and any number of moving obstacles $\mathbf{B}_i$, $i = 1, \ldots, m$, the Velocity Obstacle $\mathbf{MVO}$ classifies all velocities $\mathbf{v}_A \in \mathbf{SV}_A$ as front or rear avoidance of each obstacle $\mathbf{B}_i$ whose $\mathbf{VO}_i$ contributes to the boundary of $\mathbf{SV}_A$. □

Proof: From the definition of multiple velocity obstacle, $\mathbf{MVO}$, the safety velocity set, $\mathbf{SV}_A$, due to several obstacles $\mathbf{B}_i$ $(i = 1, \ldots, m)$, consists of n subsets $\mathbf{SV}^l_A$, $l = 1, \ldots, n$. In general, the boundary of each subset, $\partial(\mathbf{SV}^l_A)$, includes segments from the boundary of the feasible velocity set, $\partial(\mathbf{FV}_A)$, and from the boundaries of some velocity obstacles $\partial(\mathbf{VO}_{B_j})$, $(j = h, \ldots, k;\ 1 \leq h, k \leq m)$.

From Lemma 2.1, if the boundary of a safe velocity subset $\mathbf{SV}^l_A$ contains a segment from $\partial(\mathbf{VO}_{B_j})$, then $\mathbf{A}$ could collide with $\mathbf{B}_j$. The collision can be avoided with at least a tangent trajectory if $\mathbf{A}$ chooses an avoidance velocity $\mathbf{v}_A \in \mathbf{SV}^l_A$. From Lemma 2.2, all velocities in the safe velocity subset $\mathbf{SV}^l_A$ avoid obstacle $\mathbf{B}_j$ with the same type of maneuver. Thus, for example, if segment $\mathbf{S}_{l,j} = \partial(\mathbf{SV}^l_A) \cap \partial(\mathbf{VO}_{B_j})$ is from the side of the collision cone $\mathbf{CC}_{B_j}$ corresponding to a rear avoidance maneuver of obstacle $\mathbf{B}_j$, then all velocities in $\mathbf{SV}^l_A$ will avoid $\mathbf{B}_j$ with a rear maneuver.

Lemma 2.2 can be extended to a safe velocity set $\mathbf{SV}^l_A$ whose boundary includes segments from r different velocity obstacles $\mathbf{VO}_{B_j}$, $(j = 1, \ldots, r;\ r \leq m)$. Then, it follows that $\mathbf{SV}^l_A$ contains velocities that avoid each obstacle $\mathbf{B}_j$, $(j = 1, \ldots, r;\ r \leq m)$. The segments of the boundary of the collision cones $\mathbf{CC}_{B_j}$ included in $\partial(\mathbf{SV}^l_A)$ determine the type of the avoidance maneuver of obstacles $\mathbf{B}_j$. Therefore, $\partial(\mathbf{SV}^l_A)$ determines the obstacles with the most critical avoidance and the type of maneuver available. **QED**

2.5 Computing the Safe Velocity Set

The algorithm for computing the safe velocity set of a robot **A** moving in a planar environment among m obstacles $\mathbf{B}_j$ $(j = 1, \ldots, m)$ consists of three main steps. First, the feasible velocity set of **A**, $\mathbf{FV}_A$, is computed, then the velocity obstacles, $\mathbf{VO}_j$, due to each single obstacle $\mathbf{B}_j$ are computed independently of each other, and finally, each $\mathbf{VO}_{B_j}$ is subtracted from $\mathbf{FV}_A$ to find the final safe velocity set $\mathbf{SV}_A^m$ due to the m obstacles.

Since $\mathbf{FV}_A$ is a circular sector and the $\mathbf{VO}_{B_j}$ are the intersections between $\mathbf{FV}_A$ and the absolute cones $\mathbf{CC}_{B_j}$, the first two steps of the algorithm can be carried out with standard methods for computing the intersection of convex shapes [53]. The third step is the most complex, and therefore it determines the performance of the algorithm.

To illustrate the computation of the safe velocity set $\mathbf{SV}_A^m$, the example presented in Figure 2.5 of Section 2.3 is decomposed into the steps needed for finding the $\mathbf{SV}_A$ due to obstacles **B** and **C**. Part (1) of Figure 2.13 shows the intersection of $\mathbf{CC}_B$ with $\mathbf{FV}_A$, part (2) shows the computation of $\mathbf{VO}_B$ and of $\mathbf{SV}_A^1$, part (3) shows the intersection of $\mathbf{SV}_A^1$ with $\mathbf{CC}_C$, and part (4) shows the computation of **MVO** and $\mathbf{SV}_A^2$.

The main data structure used to represent each area is an ordered list of vertices. Each vertex is specified by its position and by its intersecting lines. The vertices in each list are ordered counter-clockwise, so that the interior of the area is on the left of its boundary. With this ordering, two adjacent vertices have one of the sides of the boundary in common.

In Figure 2.13, the initial safe velocity set $\mathbf{SV}_A^0$ consists of the complete $\mathbf{FV}_A$ set represented by **RQNS**. The velocity obstacles for **B** and **C** of Figure 2.5 are

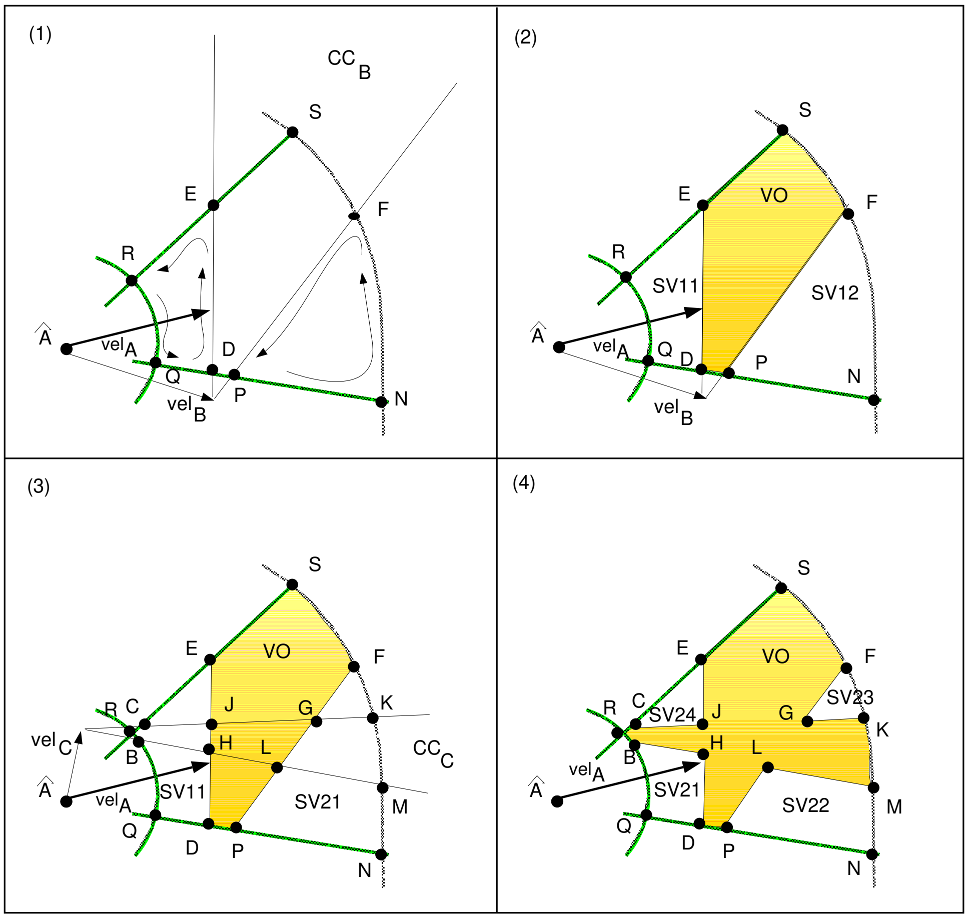

Figure 2.13: The computation of the Safe Velocity Set $\mathbf{SV}_A$.

represented by **DPFSE** for $\mathbf{VO}_B$, as shown in Figure 2.13-1, and by **BMKCR** for $\mathbf{VO}_C$, as shown in Figure 2.13-3.

The general approach for computing the difference between $\mathbf{VO}_j$ and $\mathbf{SV}_A^{j-1}$ is to represent the portions of $\mathbf{SV}_A^{j-1}$ not covered by $\mathbf{VO}_j$ with the lists of their surrounding vertices. These lists are computed by marching along the boundaries of $\mathbf{SV}_A^{j-1}$ and $\mathbf{VO}_j$ so that the free area of $\mathbf{SV}_A^{j-1}$ is always kept to the left of the marching direction. In Figure 2.13, if the first vertex of the list representing $\mathbf{SV}_A^0$ is **R**, then a counter-clock wise march is started on the boundary of $\mathbf{SV}_A^0$, $\partial(\mathbf{SV}_A^0)$, until vertex **Q** is reached, since segment **QN** intersects $\mathbf{VO}_B$ in **D**.

In general, a side of $\mathbf{SV}_A^{j-1}$ may support several vertices of both $\mathbf{SV}_A^{j-1}$ and $\mathbf{VO}_j$, as segment **QDPN** in Figure 2.13-1, where **Q** and **N** are vertices of $\mathbf{SV}_A^0$ and **D**, **P** of $\mathbf{VO}_B$. If the vertex belonging to $\mathbf{VO}_j$ is the closest to the current vertex, then the march stops following $\partial(\mathbf{SV}_A^{j-1})$ and starts moving along the boundary of $\mathbf{VO}_j$, $\partial(\mathbf{VO}_j)$. In Figure 2.13-1, **D** is closer to **Q** than **P** or **N**, and therefore segment **DE** is included in the list representing $\mathbf{SV}_A^1$. This vertex of $\mathbf{VO}_j$ becomes the starting point of a march on $\partial(\mathbf{VO}_j)$. At vertex **E** of Figure 2.13-1, the march abandons $\partial(\mathbf{VO}_B)$ and returns to $\partial(\mathbf{SV}_A^0)$ in order to keep the free area of $\mathbf{SV}_A^0$ to its left. The following vertex is **R**, which has already been examined and therefore this portion of $\mathbf{SV}_A^0$ is completed. The construction of the list representing $\mathbf{SV}_A^1$ continues until all vertices of $\mathbf{SV}_A^0$ have been assigned to a list or have been found internal to $\mathbf{VO}_1$. The resulting safe velocity set $\mathbf{SV}_A^1$ consists in general of more than one subset, such as $\mathbf{SV}_A^{1,1}$ and $\mathbf{SV}_A^{1,2}$ in Figure 2.13-2, represented by **RQDE** and **PNF**, respectively.

Parts (2) and (3) of Figure 2.13 show the intersection of $\mathbf{VO}_C$ with $\mathbf{SV}_A^1$ and the computation of $\mathbf{SV}_A^2$. In Figure 2.13-3, vertex **R** is now internal to $\mathbf{VO}_C$ and the march starts from vertex **Q**. When the march reaches vertex **D** it switches again to $\partial(\mathbf{VO}_B)$, but at **H** it moves to the boundary of $\mathbf{VO}_C$ towards **B**.

When all the vertices of $\mathbf{SV}_A^1$ have been examined and linked together, the safe velocity set $\mathbf{SV}_A^2$ consists of the following four subsets: $\mathbf{SV}_A^{2,1} = \{\mathbf{QDHB}\}$, $\mathbf{SV}_A^{2,2} = \{\mathbf{PNML}\}$, $\mathbf{SV}_A^{2,3} = \{\mathbf{FGK}\}$, and $\mathbf{SV}_A^{2,4} = \{\mathbf{ECJ}\}$.

In summary, the boundary of each subset consists of three chains: the first belonging to $\partial(\mathbf{SV}_A^{j-1})$, the second to $\partial(\mathbf{VO}_j)$, and the third part again to $\partial(\mathbf{SV}_A^{j-1})$, since only one $\mathbf{VO}_j$ is subtracted at each iteration. The switch between $\partial(\mathbf{SV}_A^{j-1})$ and $\partial(\mathbf{VO}_j)$ is executed when the closest vertex found on the current side of $\mathbf{SV}_A^{j-1}$ belongs to $\mathbf{VO}_j$. While marching on $\partial(\mathbf{SV}_A^{j-1})$, the march is counter-clockwise, while on $\partial(\mathbf{VO}_j)$ the march is clockwise: this guarantees that the set difference, $\mathbf{SV}_A^j =$

```
begin advance (Q)
        retrieve supporting line of Q, l_Q
        retrieve points P_i on l_Q
        find P^c ∈ P_i closest to Q
        if (P^c is multiple){
           retrieve points P_j on P^c
           retrieve supporting lines l_{P_j}
           find line l_{P_j} closest to l_Q
           P^c = P_j
           }
        return(P^c)
end
```

Figure 2.14: The algorithm *advance*.

$\mathbf{SV}_A^{j-1} \ominus \mathbf{VO}_j$, is computed correctly. Some care must be taken when more than two segments intersect on the same point. In this case, the intersection point is marked as *multiple*, and the march continues on the segment making the sharpest left angle with the current segment.

The selection of the vertices contributing to $\partial(\mathbf{SV}_A^j)$ is carried out by the procedure *advance*, as shown in Figure 2.14. In this algorithm, $\mathbf{Q}$ is the current point on $\partial(\mathbf{SV}_A^{i-1})$, $\mathbf{l}_Q$ is the line emanating from $\mathbf{Q}$, $\mathbf{P}_i$ are the points on $\mathbf{l}_Q$, and $\mathbf{l}_{P_j}$ are the lines centered at a multiple point $\mathbf{P}_j$.

Key to the correct execution of *advance* is the availability of the sorted list of vertices of each $\mathbf{VO}_j$. This list must include all the intersections between the sides of $\mathbf{VO}_j$ and the sides of the $\mathbf{VO}_k$, with $k < j$. For example, the velocity obstacle for $\mathbf{C}$, $\mathbf{VO}_C$, shown in Figure 2.13-3 is the area limited by vertices **CRBMK**. Before computing the **MVO** due to **B** and **C**, the list representing $\mathbf{VO}_C$ must be updated to include the intersections between the sides of $\mathbf{VO}_C$ and those of $\mathbf{VO}_B$, i.e. points **HLGJ**, so that *advance* can compute the correct safe sets $\mathbf{SV}_A^{2,i}$ $i = 1, \ldots, 4$.

In general, the lists describing each safe velocity set $\partial(\mathbf{SV}^i)$ $i = 1, \ldots, m$ include

```
begin update (VO_i)
        for (j = i + 1, ..., m) {
            intersect sides of VO_i with sides of VO_j
            add intersections to vertex list of VO_j
            sort vertices of VO_j
        }
end
```

Figure 2.15: The algorithm *update*.

segments from $\partial(\mathbf{VO}_i)$ $i = 1, \ldots, m$. Therefore, the boundary of $\partial(\mathbf{VO}_j)$, $j = i+1, \ldots, m$ must be updated to include the intersections with the sides of $\partial\mathbf{SV}^i$ that were not part of $\partial\mathbf{SV}^{i-1}$. The update function is carried out by the procedure *update*, shown in Figure 2.15, where $\mathbf{VO}_i$ is the current velocity obstacle, and m is the number of obstacles.

The algorithm for computing the **VO** of m obstacles is summarized by the procedure *safe_vel*, as shown in Figure 2.16. In this algorithm, $\mathbf{P}_j$ is a point on $\partial(\mathbf{SV}_A^{i-1})$, $\mathbf{Q}_j$ is a point on $\partial(\mathbf{SV}_A^i)$ m, is the number of obstacles, n_i is the number of vertices in $\partial(\mathbf{SV}_A^{i-1})$, $\mathbf{P}_j$ is a vertex in $\partial(\mathbf{SV}_A^{i-1})$, l is the index of the subsets in $\mathbf{SV}_A^i$.

The correctness of this algorithm can be proven by showing that the algorithm terminates and that it finds the difference of the two sets, $\mathbf{SV}_A^{i-1}$ and $\mathbf{VO}_i$, if they intersect. The termination of the algorithm is guaranteed since it examines only finite sets: the vertices of $\partial\mathbf{SV}_A^{i-1}$ and those of $\partial\mathbf{VO}_i$.

The computation of the difference between the two sets is assured by the selection of the boundary vertices performed by *advance*, and by the update of the vertices of $\mathbf{VO}_j$ $j = i, \ldots, m$ after the computation of $\mathbf{SV}_i$, performed by *update*. To illustrate the computation of the set difference, the two cases of intersecting and non intersecting $\mathbf{SV}_A^i$ and $\mathbf{VO}_{i+1}$ are shown in Figure 2.17. The figure shows a safe velocity set $\mathbf{SV}_A^i$ represented by points $\widehat{\mathbf{AKH}}$, the multiple velocity obstacle **MVO** due to obstacles

begin *safe_vel* ($\mathbf{SV}^0$)
    for ($i = 1, \ldots, m$) {
        for ($j = 1, \ldots, n_i$) {
            if ($\mathbf{P}_j \notin \partial(\mathbf{VO}_i)$)
              then {
                  $\mathbf{Q}_{l,1} = \mathbf{P}_j$
                  while ($\mathbf{Q}_{k+1}$ not stored) {
                      k=k+1
                      $\mathbf{Q}_k$ = advance ($\partial(\mathbf{SV}_A^{i-1})$, $\partial(\mathbf{VO}_i)$)
                      if($\mathbf{Q}_k$ stored)
                            then start subset $\mathbf{SV}_A^{i,l+1}$
                            else add $\mathbf{Q}_k$ to $\mathbf{SV}_A^{i,l}$
                  }
              }
        }
        for $h = i + 1, \ldots, m$ update ($\mathbf{VO}_h$)
    }
end

Figure 2.16: The algorithm *safe_vel*.

$\mathbf{B}_j$ $j = 1, \ldots, i$, represented by points **KNSH**, and three possible $\mathbf{VO}_{B_{i+1}}$. The **VO** marked with (1) is represented by points **BCD** and intersects $\mathbf{SV}_A^i$ only, the **VO** marked with (2) is represented by points **PQNR** and does not intersect $\mathbf{SV}_A^i$, and the **VO** marked by **XYV** intersects both $\mathbf{SV}_A^i$ and **VO**. Segment **KH** is the portion of the boundary of $\mathbf{SV}_A^i$ that was not present in $\mathbf{SV}_A^{i-1}$, added in the last iteration of the algorithm.

In general, a velocity obstacle $\mathbf{VO}_{B_{i+1}}$ has at least two points in common with the $\partial(\mathbf{SV}_A^i)$, such as, for example, points **C** and **B** marked by (1) in Figure 2.17. These two points are part of the initial list of vertices representing $\mathbf{VO}_{B_{i+1}}$, and therefore, the procedure *advance* is able to compute the difference between $\mathbf{VO}_{i+1}$ and $\mathbf{SV}_A^i$. The $\mathbf{VO}_{B_{i+1}}$, marked in (2) by points **PQNR**, does not intersect $\mathbf{SV}_A^i$. When *advance* reaches point **K** during its march on $\partial(\mathbf{SV}_A^i)$, it does not consider vertex **R**, and it leaves $\mathbf{SV}_A^i$ unchanged. The $\mathbf{VO}_{B_{i+1}}$ marked by (3) intersects segment **KH** that has

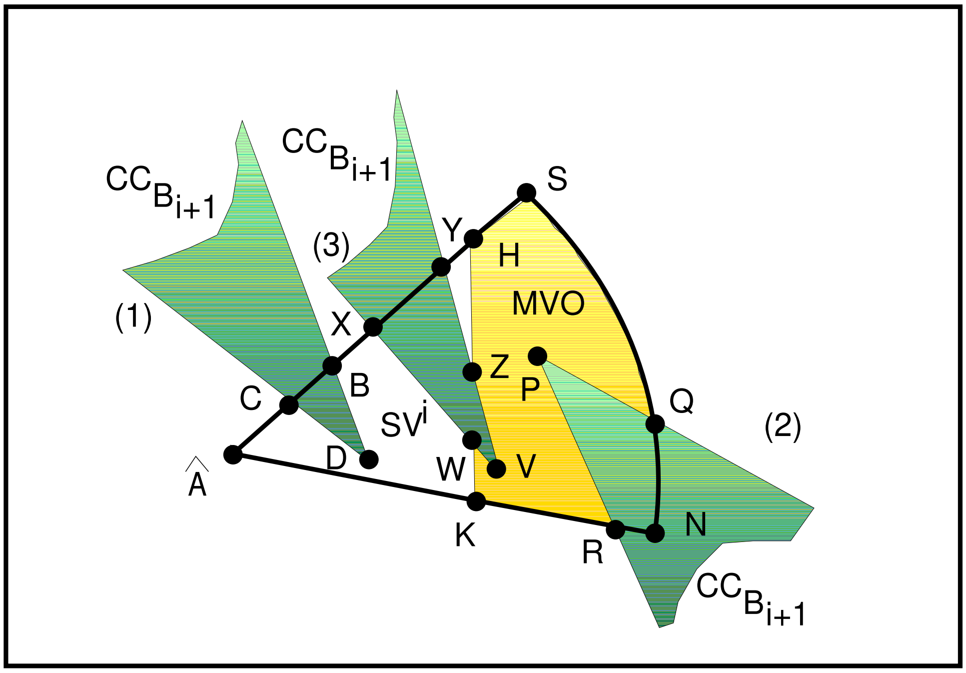

Figure 2.17: Example of intersecting and non-intersecting sets.

been added to the boundary of $\mathbf{SV}_A^i$ by $\mathbf{VO}_{B_i}$. In this case, the procedure *update* has included in the list of vertices representing $\mathbf{VO}_{B_{i+1}}$ the two new vertices $\mathbf{Z}$ and $\mathbf{W}$. Then *advance* is able to identify the two subsets $\widehat{\mathbf{A}}\mathbf{KWX}$ and $\mathbf{ZHY}$ forming the safe set $\mathbf{SV}_A^{i+1}$.

2.5.1 Complexity and Completeness of the Algorithm

The performance of the algorithm depends on two steps: the computation of the velocity obstacles, $\mathbf{VO}_{B_j}$ $j = 1, \ldots, m$, and of the safe velocity set $\mathbf{SV}_A^{j-1}$.

Each single $\mathbf{VO}_{B_j}$ is the intersection between the feasible set, $\mathbf{FV}_A$ and the absolute cone, $\mathbf{CC}_{B_j}$, of obstacle $\mathbf{B}_j$. Thus, it is the intersection of the convex area $\mathbf{FV}_A$ with the two sides of the absolute cone. Since the intersection of two convex polygons can be computed in $O(h+l)$ time [53], then the computation of a $\mathbf{VO}$ has the same time complexity, where the function $O(\cdot)$ represents the upper bound, and $h = 3$

and $l = 2$ are the number of sides of $\mathbf{FV}_A$ and $\mathbf{CC}_{B_j}$, respectively. Furthermore, the overall complexity of computing the intersection with m obstacles in the environment is $O(m)$, since h and l are small and constant. It is also important to note that the maximum number of vertices of each $\mathbf{VO}_{B_j}$ is $(h + l)$ [53].

The computation of $\mathbf{SV}_A^m$ consists of finding the difference between $\mathbf{SV}_A^{j-1}$, which may be non-convex, and a $\mathbf{VO}_{B_j}$ $(j = 1, \ldots, m)$, which is convex. During the computation of $\mathbf{SV}_A^1$, the algorithm examines the h vertices of $\mathbf{SV}_A^0 = \mathbf{FV}_A$ and the $(h + l)$ vertices of $\mathbf{VO}_{B_1}$.

In the computation of the following safe sets, the number of vertices of $\mathbf{SV}_A^j$ is at most increased by the number of vertices of $\mathbf{VO}_{B_j}$, which is at most equal to $(h + l)$. Thus, neglecting for the moment the procedure *update*, the complexity of computing the complete $\mathbf{SV}$ due to m obstacles is proportional to the number of vertices in each $\mathbf{SV}_A^j$, i.e. the number of times that the procedure *advance* is invoked. The lower bound of the complexity, indicated by the function $\Omega(\cdot)$, is:

$$\Omega(h + (m - 1) * (h + l)) = \Omega(m) \tag{2.14}$$

Only the lower bound can be defined, since the $\mathbf{SV}_A^j$ may be non-convex and may require a larger computation time. If all the $\mathbf{SV}_A^j$ are convex, then the computational complexity of this step is $\theta(m)$, where $\theta(\cdot)$ indicates the optimal time complexity [53]. This case occurs when all the apices of the absolute collision cones are outside of the feasible velocity set $\mathbf{FV}$.

The procedure *advance* presented earlier, does not affect the overall complexity of the computation of $\mathbf{SV}_i$, since tables are used to find the next vertex in the $\partial(\mathbf{SV})_A^j)$, and therefore the procedure takes constant time. In fact, the lines intersecting at a vertex are associated to a sorted list containing all vertices on that line, and therefore no search is necessary. Similarly, multiple vertices point to the sorted list of the lines intersecting at that vertex.

Once $\mathbf{SV}_A^j$ has been computed, the procedure *update* must be used to include the intersection of the new sides of $\mathbf{SV}_A^j$ in the lists representing the remaining $(m-j)$ $\mathbf{VO}_{B_k}$ $k = j+1, \ldots, m$. The new sides of $\mathbf{SV}_A^j$ are the sides due to $\mathbf{VO}_{B_j}$ that were not part of $\partial(\mathbf{SV}_A^{j-1})$. As mentioned earlier, these new sides are portions of the boundary of the absolute collision cone $\mathbf{CC}_{B_j}$. Therefore, the update of the $(m-j)$ boundaries of the velocity obstacles $\mathbf{VO}_{B_j}$, consists of computing the intersection of the two sides of $\mathbf{CC}_{B_j}$, with the remaining $(m-i)$ $\partial(\mathbf{VO}_{B_j})$, each with at most $(h+l)$ sides. Since the maximum number of sides of $\mathbf{VO}_{B_j}$ is fixed, the counter-clockwise ordering of the set of boundary points add a constant time to the overall complexity. The generic update step has then complexity of:

$$O((m-j)*(h+l+l)) = O(m) \tag{2.15}$$

The derivation of the complexity of the various procedures involved in the computation of $\mathbf{SV}_A^m$ can be summarized by the following proposition:

Proposition 2.2: The Safe Velocity set $\mathbf{SV}_A^m$ due to the m obstacles $\mathbf{B}_j$ $j = 1, \ldots, m$ can be computed in time $\Omega(m)$, which becomes $\theta(m)$ if all $\mathbf{VO}_{B_j}$ are convex, after $O(m^2)$ preprocessing time due to the procedure *update*. □

Since the time complexity of *update* is the dominating factor in the performance of the *safe_vel* algorithm, it is interesting to suggest an alternate approach for computing the lists representing the $\partial(\mathbf{VO}_{B_j})$ and including the intersections with the sides of all the absolute collision cones. The $(4*m)$ intersections between $\partial(\mathbf{SV}_A^0)$ and the absolute collision cones are computed initially, and inserted in a sorted circular list. Each element in the list has a link pointing to the other intersection of the same line with $\partial(\mathbf{SV}_A^0)$. With this data structure, sides of different velocity obstacles can intersect each other within $\mathbf{SV}_A^0$, only when they have alternating intersections with

$\partial(\mathbf{SV}_A^0)$. The complexity of this preprocessing step would then consists of $O(m)$, to compute all the intersections with the initial set $\mathbf{SV}_A^0$, $O(m \log m)$ due to the insertion in the sorted list, and $O(m)$ for scanning the circular list and computing the intersections of the overlapping segments. With this approach, the time complexity of the preprocessing step of update is reduced to $O(m \log m)$.

The completeness of the algorithm can only be stated if a number of hypothesis are made about the behavior of the obstacles. It is obvious that in the absence of any information about the intentions and performance of the obstacles, there is no guarantee that a planner can find a solution, even if one might exist [58].

If it can be assumed that changes in obstacles trajectory are slow with respect to the planning cycle, i.e. sensing the environment and updating the plan, then it can be stated that the planner is complete with respect to a single avoidance maneuver. In fact, the **VO** shows whether a single avoidance maneuver is possible or no solution exists.

Completeness with respect to the entire trajectory cannot be guaranteed, even in the case of known obstacle trajectories. This is due to the fact that the maneuvers computed using the **VO** avoid all obstacles in the environment irrespectively of their expected collision time. In this way, the algorithm computes a trajectory whose maneuvers are affected by the avoidance of collisions that may be a long time away. These trajectories are very conservative, since each segment of the computed trajectory could itself be a safe trajectory if extended to any time $t > t_0$. Conversely, safe trajectories may include segments with velocity violating the **VO** of the obstacles whose collision is not imminent, and satisfying the **VO** only of the obstacles within a reasonable time horizon. These solutions cannot be found using the current VO approach, and therefore no completeness with respect to the entire trajectory can be guaranteed.

2.6 An Example of Avoidance Maneuver in the Plane

An example of a planar avoidance maneuver is shown in Figure 2.18, in which a robot is moving in an environment containing four obstacles. The velocity vector of each object is attached to its center and is represented by the bars shown in the figure. The objects have the following parameters:

robot: $(x = 5.0,\ y = 5.0)$, $(v_x = 8.0,\ v_y = 5.0)$, $r = 5.0$,

obstacle 1: $(x = 90,\ y = 40)$, $(v_x = -12.0,\ v_y = -1.0)$, $r = 5.0$,

obstacle 2: $(x = 60,\ y = -5)$, $(v_x = -5.0,\ v_y = 5.5)$, $r = 5.0$,

obstacle 3: $(x = -30,\ y = -20)$, $(v_x = 5.0,\ v_y = 3.0)$, $r = 5.0$,

obstacle 4: $(x = -10,\ y = 40)$, $(v_x = 4.0,\ v_y = -1.0)$, $r = 5.0$.

These positions and velocities have been chosen primarily to generate clear displays of the velocity obstacle sets. To focus on the use of the Velocity Obstacle method, the robot has no final target and its goal is only to avoid the collision with the obstacles. The motion of the robot and of the obstacles with their original velocities is shown in Figure 2.19, where each object is drawn at 1 s intervals. Obstacle 2 is on a collision course with the robot with which it will collide at approximately $t = 5\ s$. It is necessary then to modify the velocity of the robot to avoid this collision.

The various steps of the computation of the avoidance maneuver are shown in the following figures. Figure 2.20 shows the feasible velocity set **FV** of the robot. For display purposes, the minimum velocity has been assumed to be equal to zero, and the maximum velocity to be equal to twice the original velocity of the robot. In this way, the **FV** set is clearly shown as the shaded area in Figure 2.20.

Figure 2.21 shows the absolute collision cones, $\mathbf{CC}_{B_i}$, due to the four obstacles as the grey triangular areas. The collision cones are intersected with the feasible set $\mathbf{FV}$, to generate the multiple velocity obstacle $\mathbf{MVO}$, shown as the grey area in Figure 2.22, and the safe velocity set $\mathbf{SV}^4$ shown as the clear areas in Figure 2.22. The safe velocity set has the index 4 since it is due to four obstacles, and it consists of the five subsets $\mathbf{SV}^{4,i}$ $i = 1, \ldots, 5$. Also shown in Figure 2.22 are the initial velocity of the robot, $\mathbf{v}_0$ and two avoidance velocities $\mathbf{v}_1$ and $\mathbf{v}_2$.

Following the development of Section 2.4, it is easy to identify the type of maneuver represented by each safe velocity subset. For example, $\mathbf{SV}^{4,2}$, represents maneuvers that could make the robot graze obstacles 2, 3 and 4. In fact, the boundary of $\mathbf{SV}^{4,2}$ includes segments from the collision cones $\mathbf{CC}_{B_i}$ $i = 2, 3, 4$. In particular, obstacle 2 can be avoided with a rear maneuver, since a segment of the rear boundary of $\mathbf{CC}_{B_2}$ is in $\partial(\mathbf{SV}^{4,2})$, and obstacle 4 can be avoided with a front maneuver, since a segment of the front boundary of $\mathbf{CC}_{B_4}$ is in $\partial(\mathbf{SV}^{4,2})$. The avoidance of obstacle 3 can be better defined as a *left* or *right* avoidance maneuver, since it moves directly behind the robot. The segment of $\mathbf{CC}_{B_3}$ included in $\partial(\mathbf{SV}^{4,2})$ corresponds to a right avoidance, while the segment of $\mathbf{CC}_{B_3}$ included in $\partial(\mathbf{SV}^{4,5})$ corresponds to a left avoidance maneuver.

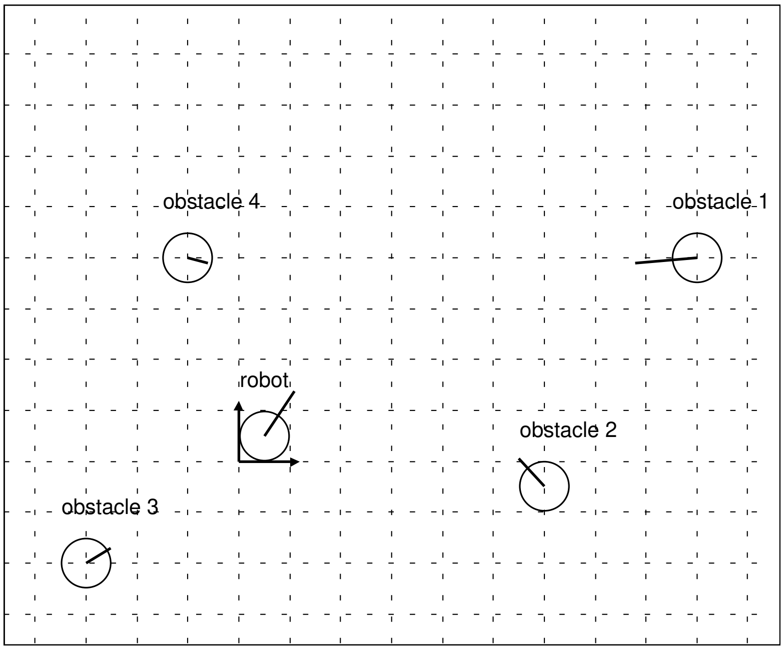

Figure 2.18: Initial configuration of the robot and the obstacles.

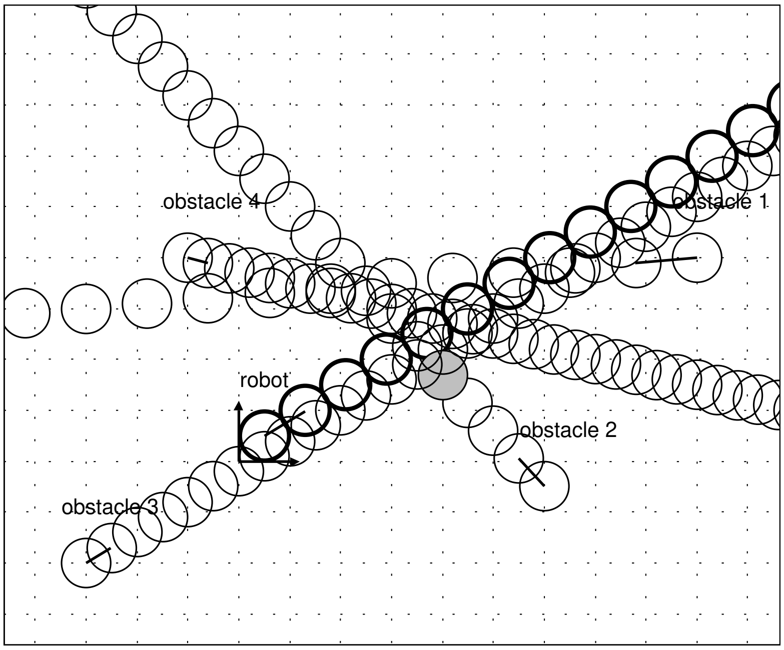

Figure 2.19: Simulation with the initial velocities.

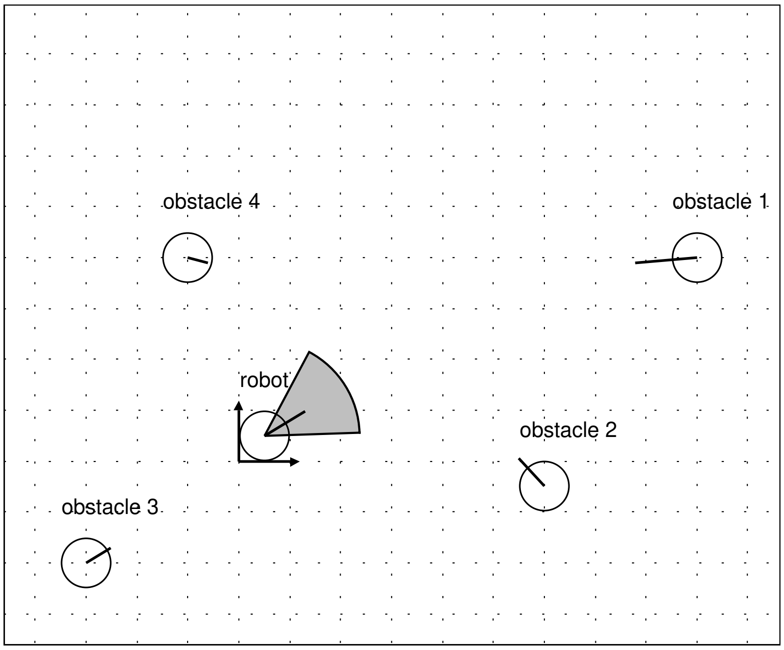

Figure 2.20: Feasible velocity set of the robot.

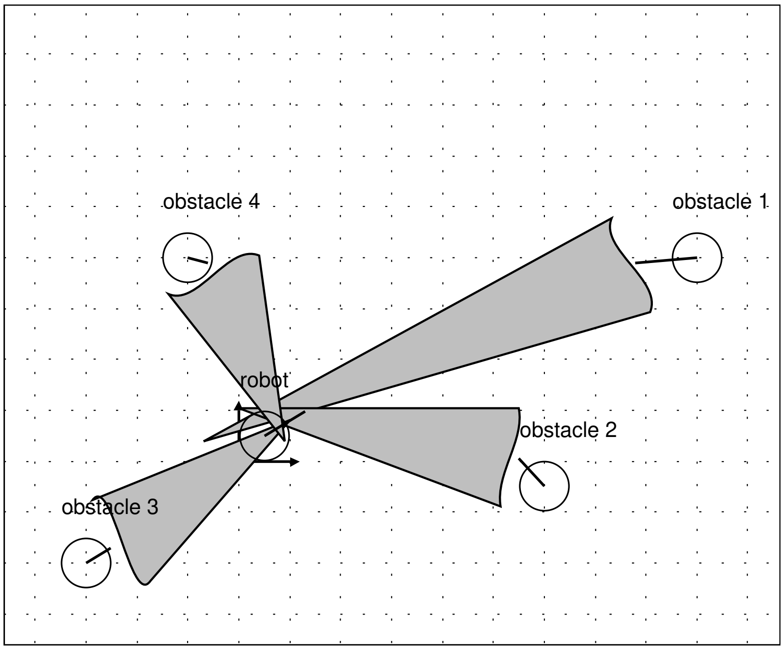

Figure 2.21: Absolute collision cones of the obstacles.

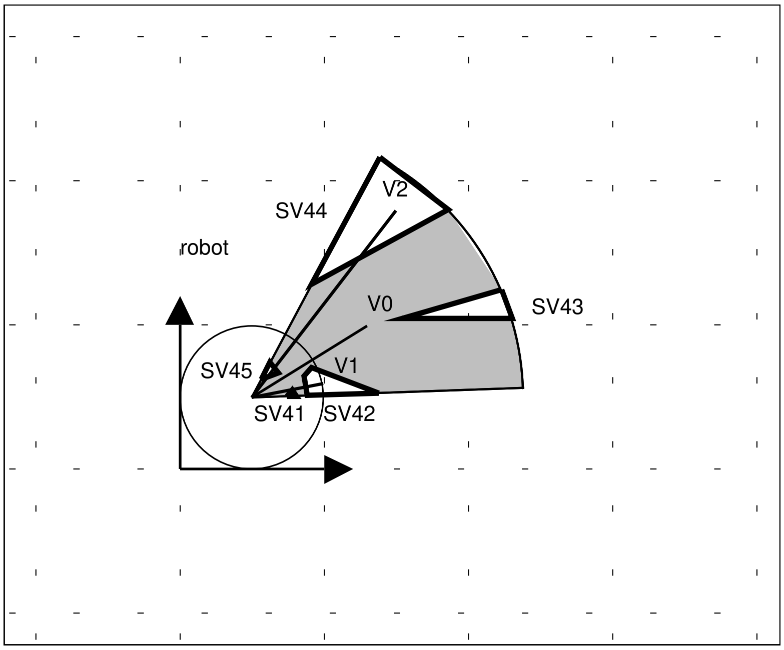

Figure 2.22: Safe velocity set.

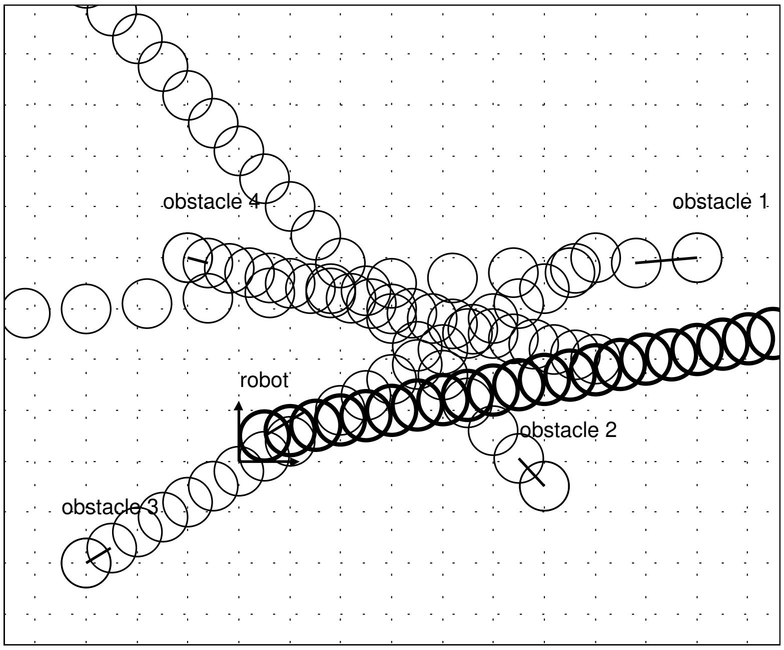

Figure 2.23: Simulation of the avoidance maneuver due to $\mathbf{v}_1$.

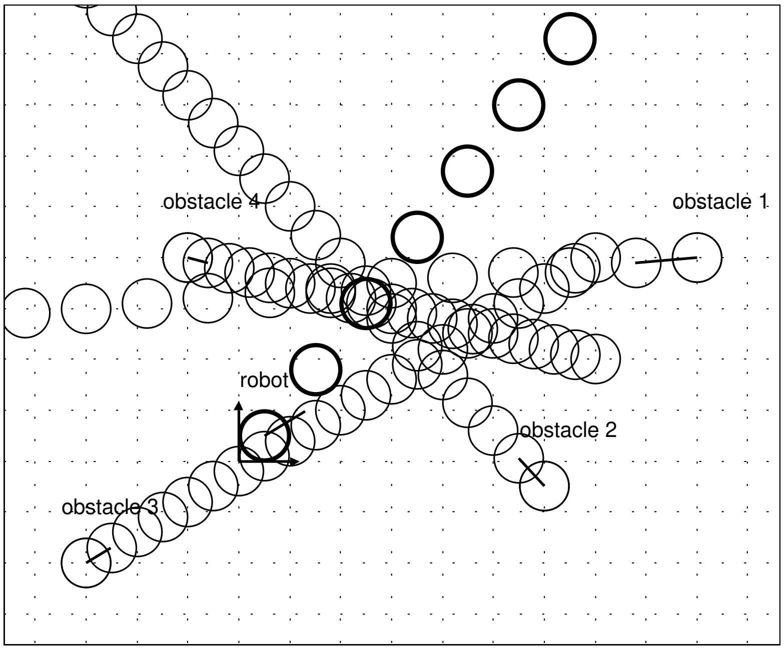

Figure 2.24: Simulation of the avoidance maneuver due to $\mathbf{v}_2$.

The two avoidance maneuvers corresponding to velocities $\mathbf{v}_1$ and $\mathbf{V}_2$ are shown in Figure 2.24 and in Figure 2.23. The first maneuver corresponds to the avoidance velocity $\mathbf{v}_1 \in \mathbf{SV}^{4,2}$ of Figure 2.22. This maneuver is closer to obstacles 2,3 and 4, since it has the possibility of being tangent to them. The simulation clearly shows the front avoidance of obstacle 4, the rear avoidance of obstacle 2, and the right avoidance of obstacle 3. Figure 2.23 shows the avoidance maneuver corresponding to velocity $\mathbf{v}_2 \in \mathbf{SV}^{4,4}$ of Figure 2.22. This maneuver avoids obstacle 1 with the smallest clearance, since $\mathbf{SV}^{4,4}$ represents avoidance velocities that could be tangent to obstacle 1.

2.7 Spatial Collision Avoidance

In this section the concept of planar **Velocity Obstacle** is extended to the case of collision avoidance among spheres moving in $\Re^3$. An approach to computing a three-dimensional VO is briefly discussed using the example of Figure 2.25 showing a robot $\mathbf{A}$ with radius r_A and velocity $\mathbf{v}_A$ and an obstacle $\mathbf{B}$ with radius r_B and velocity $\mathbf{v}_B$.

A spatial avoidance maneuver is approximated by a sequence of planar maneuver segments, each lying on one of the planes belonging to the star of planes intersecting at the center of $\mathbf{A}$. The plane containing the avoidance maneuver segment is called the *maneuver plane*. Once the maneuver plane has been chosen, the theory developed for the planar avoidance becomes directly applicable to this case. By computing the avoidance maneuver segments at an appropriate interval Δt, a continuous spatial maneuver can be approximated by a sequence of planar segments. In the following, to simplify the visualization of the avoidance maneuver, only the first segment of the maneuver is computed and the maneuver plane is the the $\mathbf{XY}$ plane.

The geometric derivation of the velocity obstacle due to $\mathbf{B}$, $\mathbf{VO}_B$ is shown in Figure 2.26. Point $\widehat{\mathbf{A}}$ and sphere $\widehat{\mathbf{B}}$, with radius $r = r_A + r_B$, represent the configura-

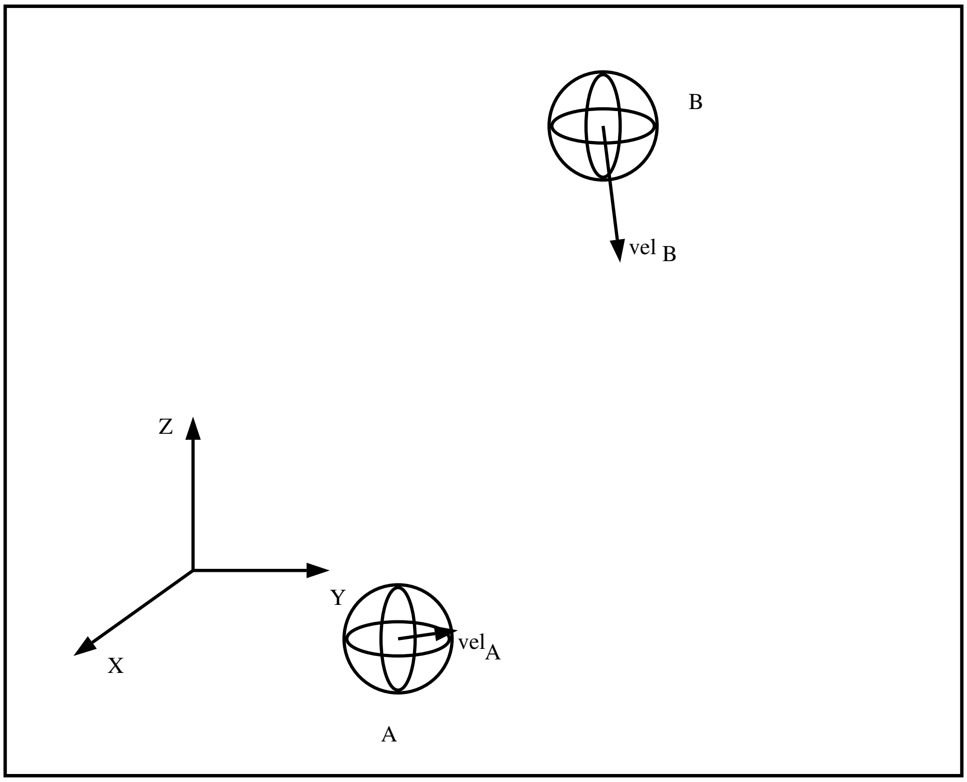

Figure 2.25: Scenario for a spatial avoidance.

tion space obstacle for **A** due to **B**. The collision avoidance condition can be written as for the planar case, requiring that the absolute velocity of **A**,$\mathbf{v}_A$, be outside the absolute cone of **B**, $\mathbf{CC}_B$, that is:

$$\mathbf{v}_A \notin \mathbf{CC}_B \tag{2.16}$$

where now $\mathbf{CC}_B$ is a three-dimensional cone. The surface of the cone is the boundary between safe and collision velocities. The set of feasible velocities on the **XY** plane, $\mathbf{FV}_{A,x}$ is represented by the circular sector $\widehat{\mathbf{A}DC}$. The cone $\mathbf{CC}_B$ intersects the maneuver plane in a conic $\mathcal{E}$. Therefore, the *velocity obstacle* **VO** for **A** due to **B** relative to the chosen maneuver plane is the portion of the conic included in the $\mathbf{FV}_A$ [24]. When the maneuver plane is internal to the absolute collision cone, the intersection is parabolic or hyperbolic, and it is not discussed here, since the

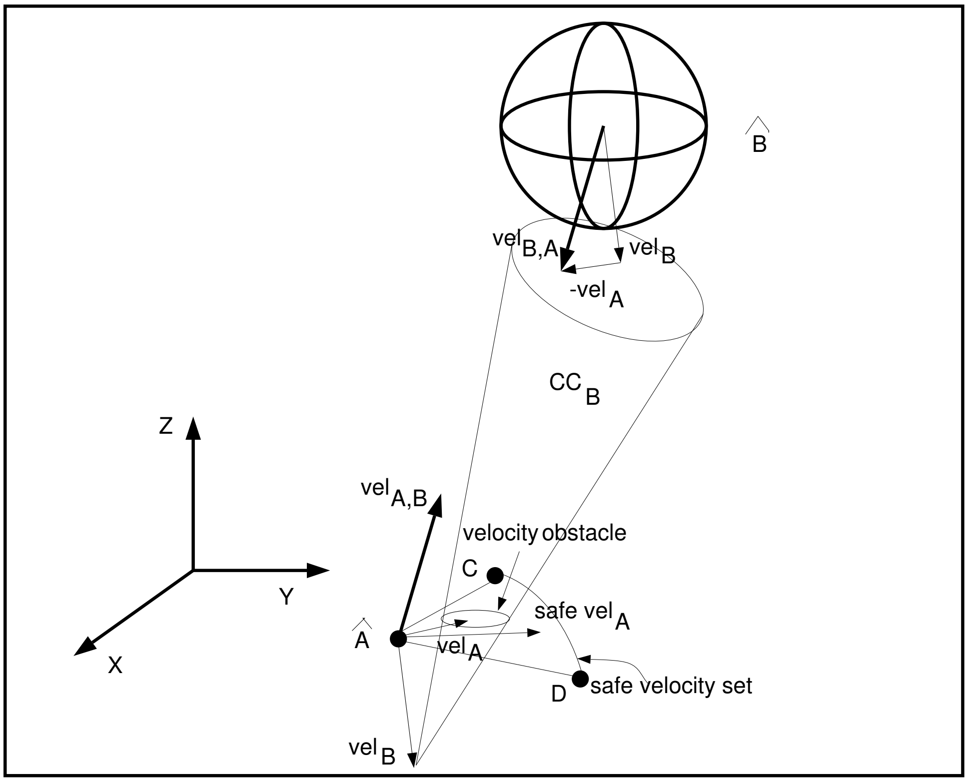

Figure 2.26: Computation of a spatial avoidance maneuver.

computation of the **VO** can be treated as in the planar case. The selection of the avoidance velocity when the intersection is an ellipse is shown in Figure 2.26 where the original velocity is shown with its tip within the elliptic velocity obstacle, and the avoidance velocity outside the ellipse.

The avoidance of m moving obstacles in three-dimensional space can be computed from the union of the single velocity obstacles, by defining the *multiple velocity obstacle* $\mathbf{MVO} = \cup_{i=1}^{m} \mathbf{VO}_i$ and by following the same procedure used in the planar case.

2.7.1 An Example of Avoidance Maneuver in 3D Space

This example shows the use of the Multiple Velocity Obstacle for selecting an avoidance maneuver in the environment of Figure 2.27. The robot is indicated with *sphere0* and the moving obstacles are *sphere1-4*. The velocities of each object are shown as segments positioned at the center of each sphere. All the obstacles are colliding with *sphere0*, as shown in Figure 2.28 by the orthogonal projection of the simulation on the **XY** plane, where the collisions are represented by darkened circles. The avoidance maneuver on the **XY** plane is computed by first finding the Multiple Velocity Obstacle, and then by choosing a velocity outside **MVO**, as shown in Figure 2.29.

An example of avoidance velocity is shown in Figure 2.30 where the avoidance velocity $\mathbf{v}_0^* \notin \mathbf{MVO}$ is chosen so that $|\mathbf{v}_0^*| = |\mathbf{v}_0|$. This specific velocity is of very simple computation, since it can be found as one of the intersection of the boundary of **MVO** with the circle of radius $|\mathbf{v}_0|$. As expected, the resulting trajectory shows no collision with the moving obstacles.

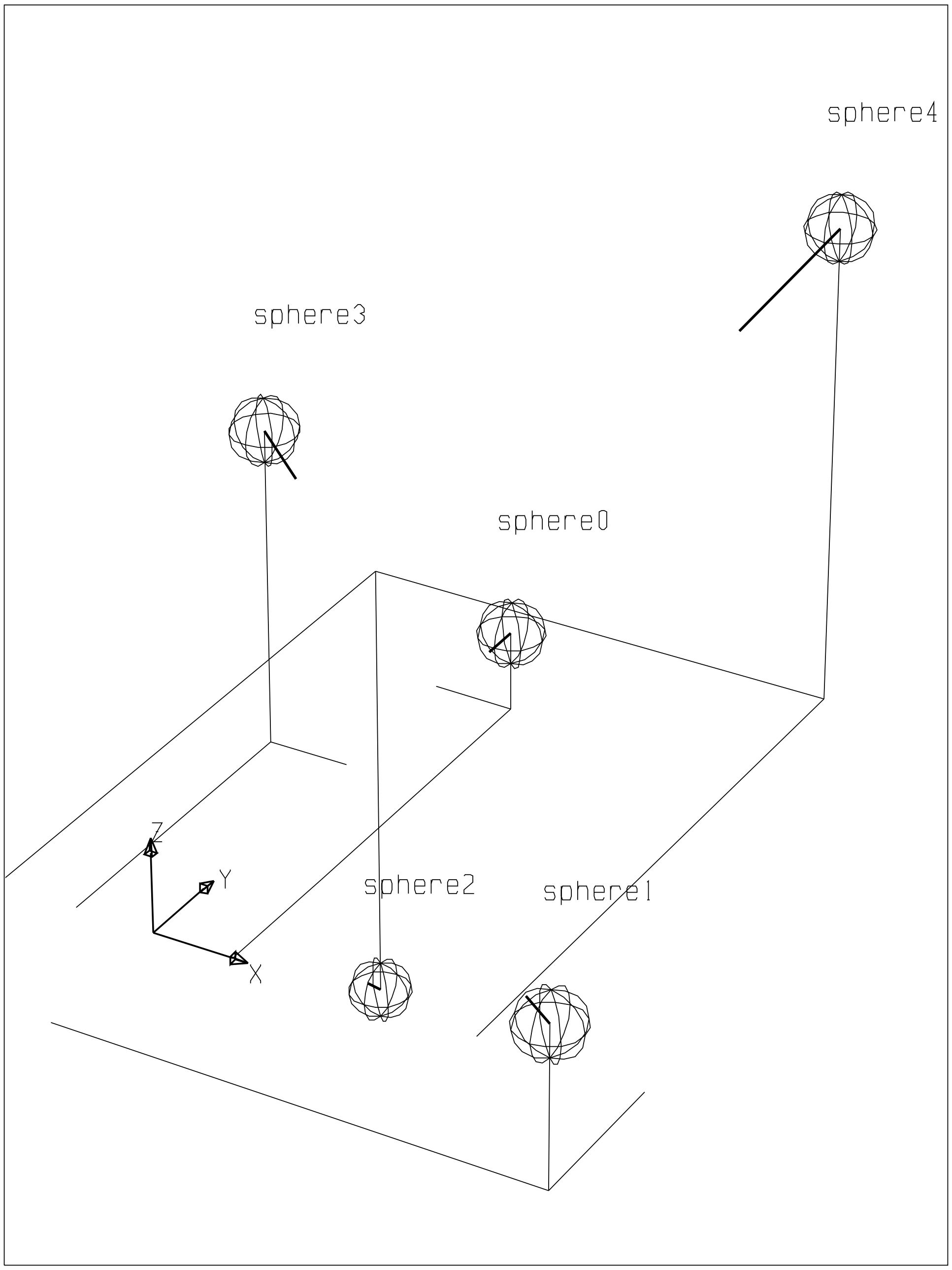

Figure 2.27: Perspective view of the spatial planning example.

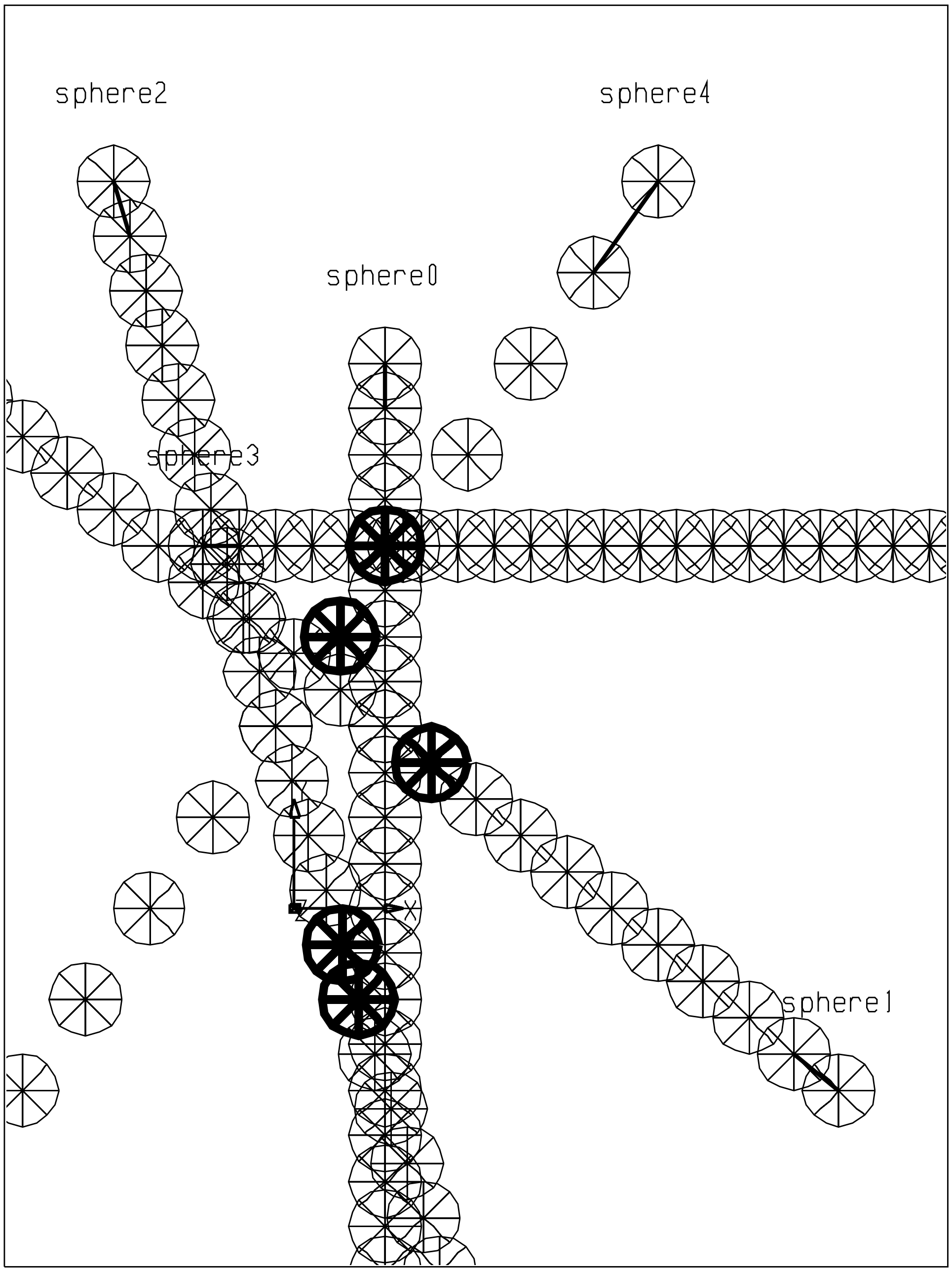

Figure 2.28: Simulation of the initial scenario.

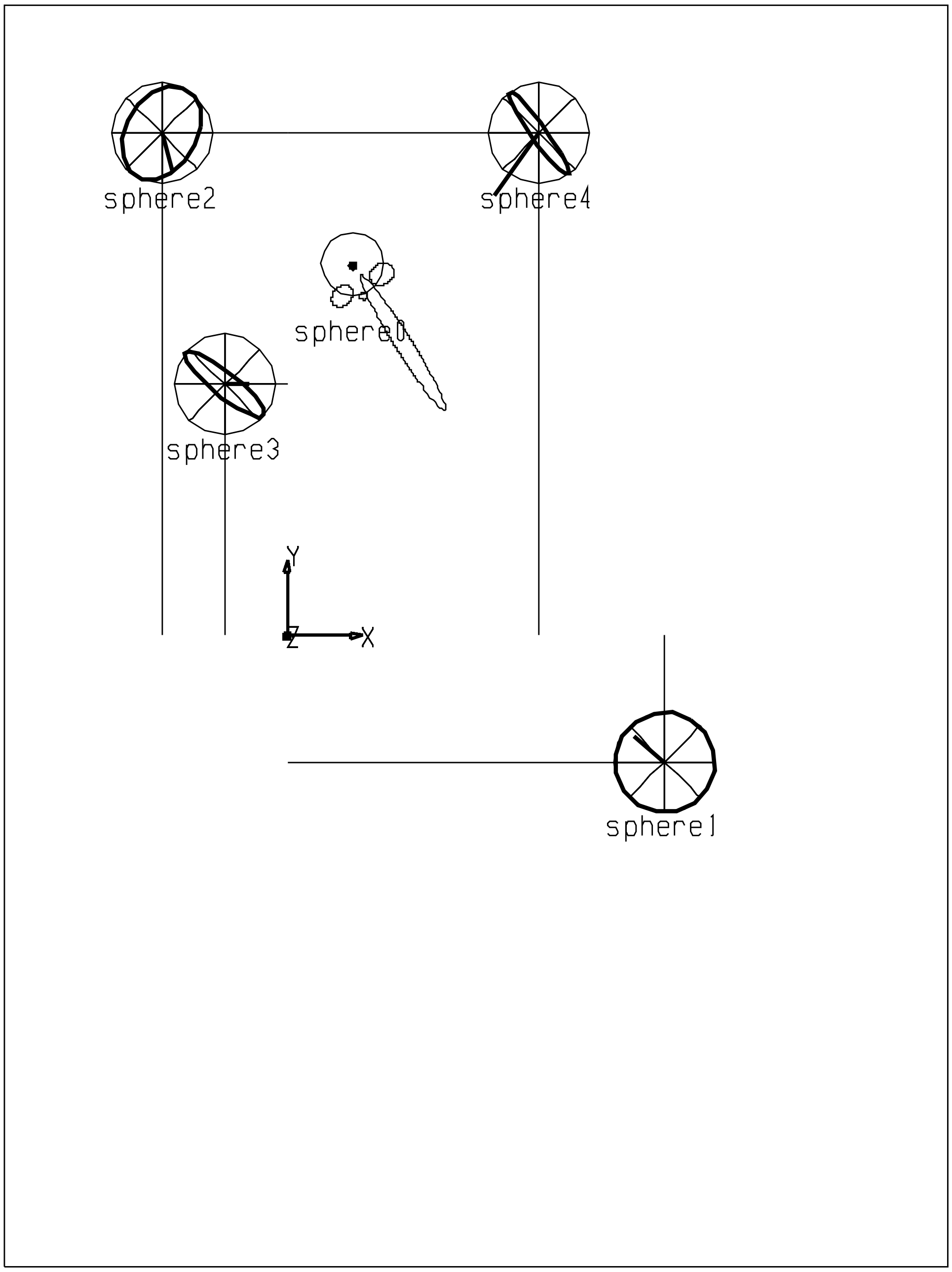

Figure 2.29: Multiple velocity obstacle on the XY plane.

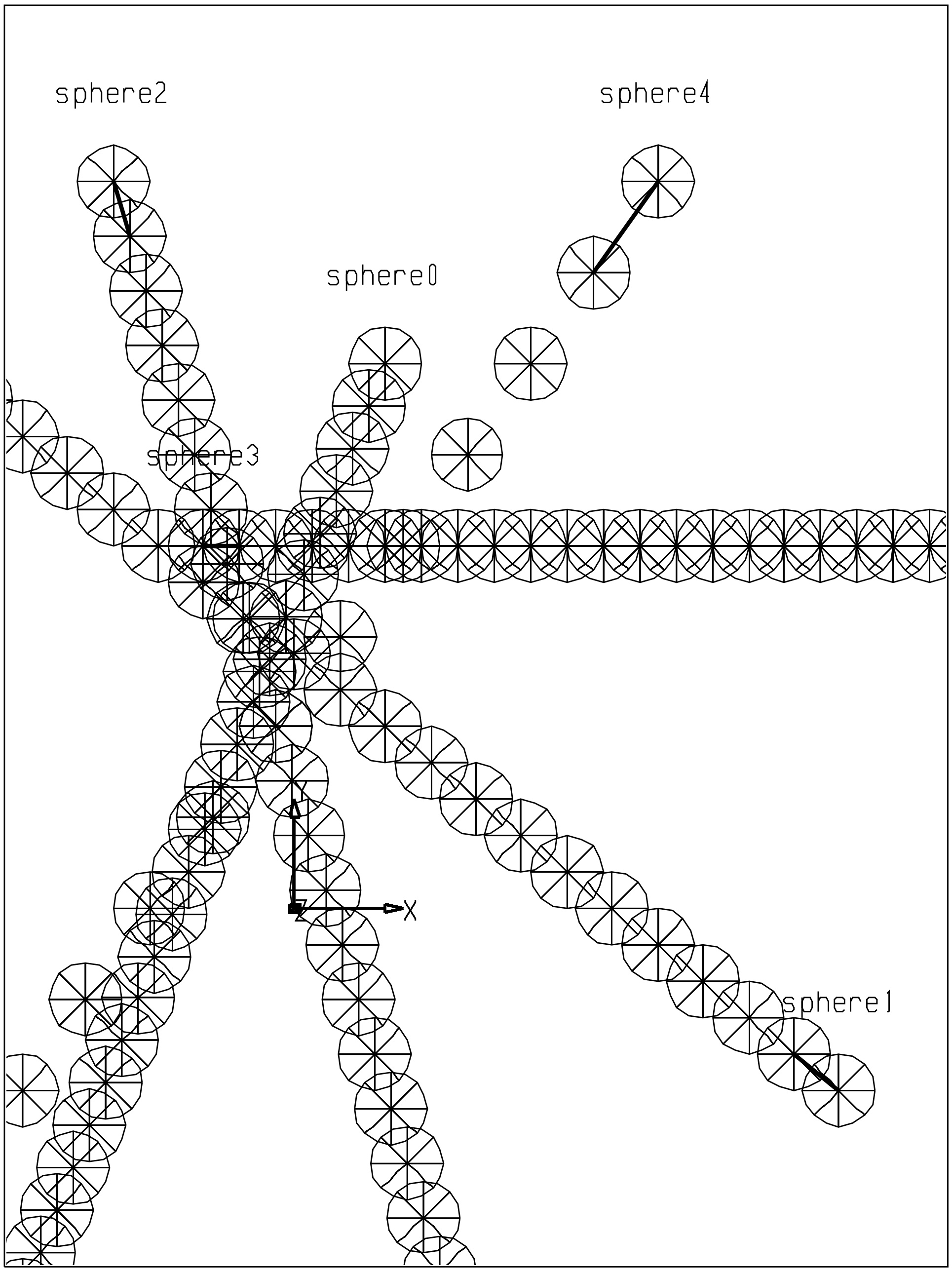

Figure 2.30: The maneuver avoiding all the moving obstacles.

2.8 Solving the Kinematic Problem

This section presents an approach to solving the *kinematic problem* of motion planning in time-varying environments that uses the velocity obstacles. As discussed in Section 2.1, the kinematic problem consists of computing a trajectory that avoids the obstacles in the environment, reaches the goal and satisfies approximate dynamic constraints. The main tool for solving this problem has been developed in the previous sections of this chapter, and consists of the velocity obstacle of a robot due to the surrounding obstacles. The velocity obstacle maps some of the constraints of the problem into simple geometric conditions that can be tested to compute the trajectory. Therefore, the approach used in this section to solve the kinematic problem consists of decomposing the trajectory into a sequence of elementary avoidance maneuvers that are computed using the velocity obstacles. The selection of each avoidance maneuver can be carried out by a search program that finds the shortest trajectory to the goal. Conversely, each maneuver can be chosen using some of the heuristics discussed in Section 2.4, to give the trajectory a specific structure.

In this section the two approaches are briefly examined on an example of Intelligent Highway System. The scenario of the example is shown in Figure 2.31, and it consists of three lanes of a tract of a highway. The robotic vehicle in the leftmost lane needs to reach the exit to the right avoiding the other two vehicles. The initial velocity of the robot and of vehicle 2 is ($v_x = 3.0\ m/s,\ v_y = 0.0\ m/s$), and the initial velocity of vehicle 1 is ($v_x = 2.3\ m/s,\ v_y = 0.0\ m/s$). These low velocities have been chosen for display purposes, and similar results can be obtained with proportionally higher velocities.

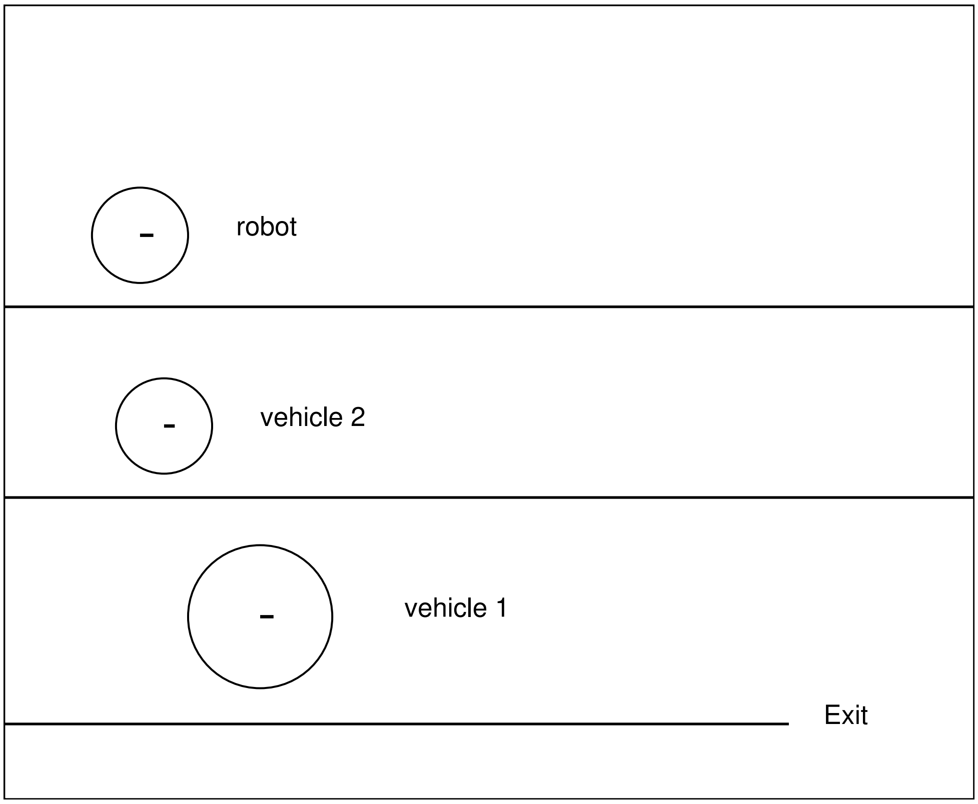

Figure 2.31: Example of kinematic problem.

2.8.1 Exhaustive Search

The fastest trajectory solving the problem of Figure 2.31 can be computed by performing an exhaustive search over all trajectories reaching the goal and avoiding the obstacles.

Standard search techniques [52], [49] require that the search space be a discrete set characterized by nodes and operators. Each node represents the state of the system at a given time, and the operators are functionals mapping a node at time t_i into its successor at time $t_{i+1} = T_i + \Delta t$.

The choice of the interval Δt depends on the available computation speed and on the dynamics of the obstacles. A fine temporal resolution requires a very long elaboration time, but it computes a trajectory that can be precisely tracked. On the other hand, a coarser interval implies that the avoidance velocity used in the search is a less accurate approximation of the real avoidance velocity, thus increasing the risk of errors and the need for accurate real time sensing and execution. For the search used in the example of Figure 2.31, the interval is chosen arbitrarily as $\Delta t = 10s$.

Here, the velocity space of the intelligent vehicle is transformed into a discrete set by the discretization of time, and is mapped into a tree with *nodes* corresponding to the position of the vehicle and *edges* representing the maneuvers executed in the interval Δt. At each node, the vehicle can choose one *operator* to move to the next node. The operators are velocity transformations that move the robotic vehicle from its current position to the new position after the interval Δt. At each node, the set of operators consists of the velocities in the Safe Velocity set, that is discretized by considering only the velocities corresponding to the vertices and a few intermediate points on the boundary. The search tree is then defined as follows:

$$n_{i,j} = \{\mathbf{x}_{i,j} = (x_j(t_i), y_j(t_i)), \mathbf{v}_{i,j} = (v_{j,x}(t_i), v_{j,y}(t_i))\} \tag{2.17}$$

$$o_{i,j,l} = \{\mathbf{v}_l \mid \mathbf{v}_l \in \mathbf{SV}_j(t_i)\} \tag{2.18}$$

$$e_{j,k} = \{(n_{i,j}, n_{i+1,k}) \mid n_{i+1,k} = n_{i,j} + (o_{i,j,l}\,\Delta t)\} \tag{2.19}$$

where $n_{i,j}$ if the jth node at time t_i, $o_{i,j,l}$ is the lth operator on node j at time t_i, and $e_{j,k}$ is the edge between node n_j at time t_i, and node n_k at time t_{i+1}.

A node $n_{i,j}$ is completely expanded when all the operators $o_{i,j,l}$ have been applied, and all the edges emanating from $n_{i,j}$ have been examined. Four edges of the tree starting at t_i are shown in Figure 2.32. In this Figure, the Safe Velocity set, $\mathbf{SV}(t_i)$ is discretized with a grid subdivision. Two different velocities are applied to the node $n_{i,j}$ to compute two new nodes at time t_{i+1}, $n_{i+1,k}$ and $n_{i+1,h}$. Two new discretized $\mathbf{SV}(t_{i+1})$ are shown at time t_{i+1}, with three operators selected for the next expansion. The tree generated by expanding the nodes has a constant time interval between nodes

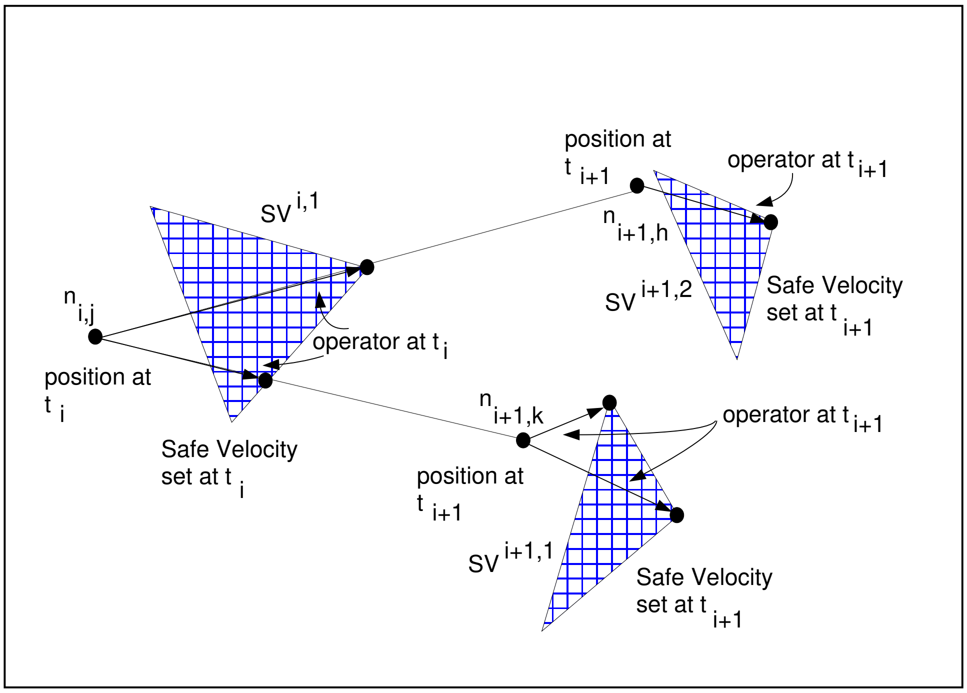

Figure 2.32: Tree representation for the global search.

and variable branch number, depending on the shape of the $\mathbf{SV}^{i,j}$. A search over this tree allows to select the fastest trajectory reaching the goal.

The trajectory shown in Figure 2.33 is computed using a Depth First Iterative Deepening algorithm [40]. This algorithm returns the fastest path to the target, the first time it reaches it. In the example, the **SV** sets have been discretized by considering four points on each boundary segment and the maximum feasible velocity in the direction to the goal. The search has been expanded to 45 s and a depth of 5 levels . The total motion time of this solution is of 35 s.

2.8.2 Heuristic Search

Velocity Obstacles (VO) can be used to develop heuristics for pruning the search tree. Here, only one velocity per node expansion is used, thus there is no a priori guarantee that the goal can be reached. This approach uses the hierarchy of goals described in

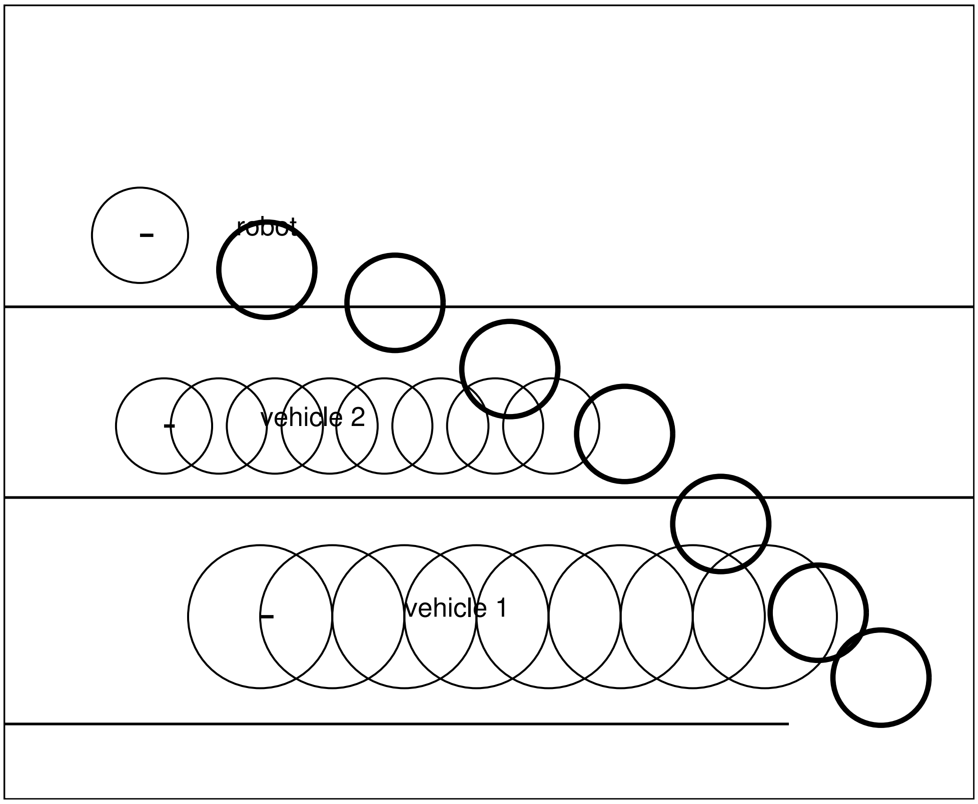

Figure 2.33: Trajectory computed by global search.

Section 1.4, and tries to achieve the various goals in order of decreasing importance. Survival of the vehicle is naturally the primary goal, while secondary goals include reaching the desired goal and minimizing motion time. The hierarchical satisfaction of these goals comes naturally within the VO approach, since the avoidance maneuver at each time interval avoids all obstacles, provided that a safe avoidance velocity exists. Then, depending on the appropriate selection of the avoidance velocity, some of the secondary goals can also be achieved.

Not all velocities within a safe set are good candidates for avoidance maneuvers, since they may move the robot away from its target or they may produce a very slow trajectory. Furthermore, other considerations may affect the choice of the avoidance velocity, such as the type of obstacle that is to be avoided, its speed, size, and so

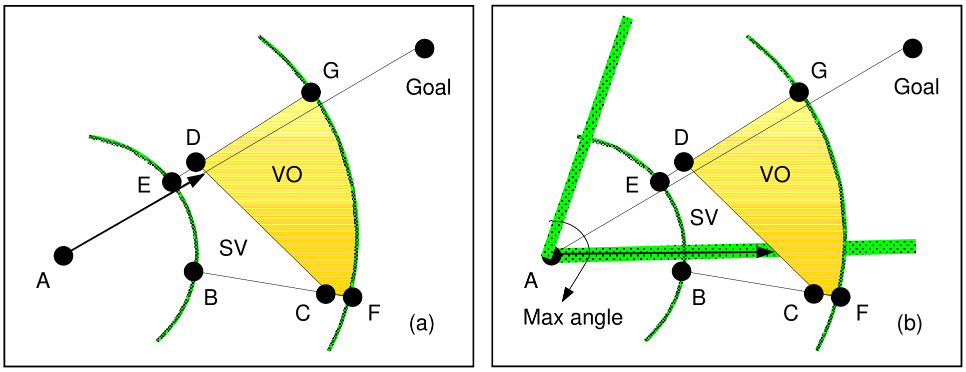

Figure 2.34: **a**: TG strategy. **b**: MV strategy.

on. These considerations can be used to approximately classify the elements in the environment in the two large categories of high risk and low risk obstacles. The natural heuristic strategy used in the presence of high risk obstacles is to let them go away without crossing their path. This heuristics maps naturally into a rear avoidance maneuver. The avoidance of a low risk obstacle can be done with a more adventurous maneuver, such as a front avoidance maneuver. Both heuristics follow from the structure of the avoidance trajectory described in Section 2.4.

Given these considerations, several heuristic strategies can be implemented. A possibility is to choose the highest safe velocity along the line to the goal, as shown in Figure 2.34-a, so that the trajectory takes the robot towards its target. This strategy is denoted in the following by **TG** (to goal). A second heuristics could be to select the maximum safe velocity within some specified angle α from the line to the goal, as shown in Figure 2.34-b, so that the robot moves fast, even if this implies not aiming directly at the goal. This second strategy is called **MV** (maximum velocity). Other heuristics may combine **TG** and **MV**, so that the velocity pointing to the goal is chosen when it is fast enough, and a higher velocity, pointing towards the vicinity of the goal, is chosen when the velocity towards the goal is too low. More sophisticated

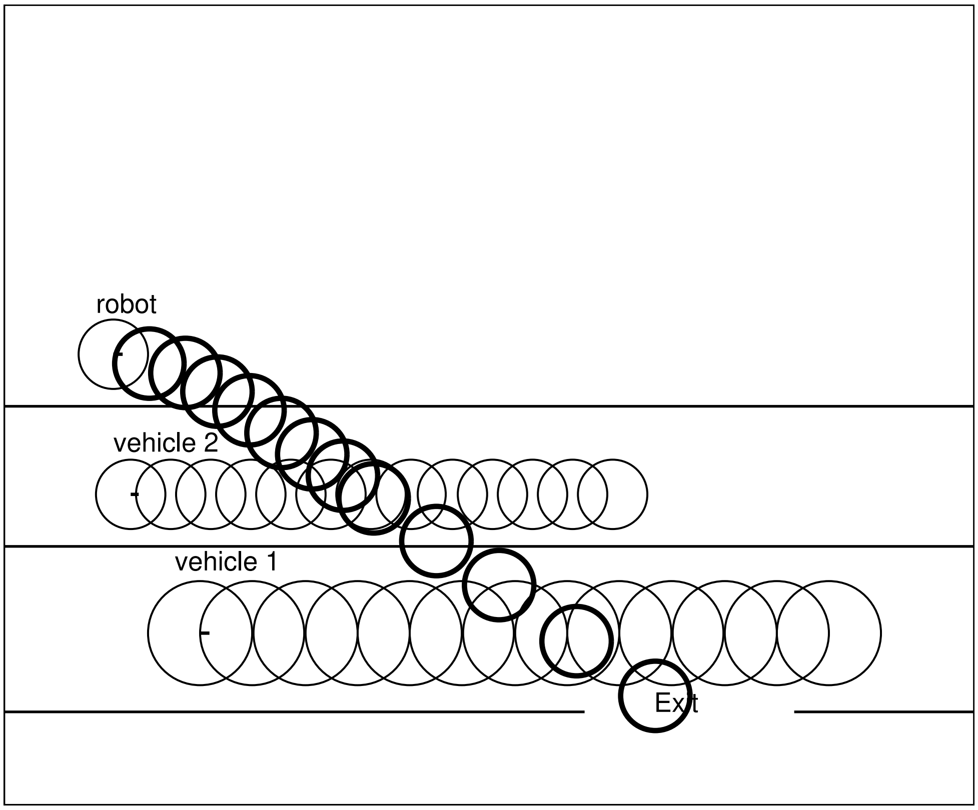

Figure 2.35: Trajectory computed with the TG strategy.

strategies could also include the safety consideration described in Section 2.4, and prescribe a sequence of rear and front avoidance maneuvers.

The selected heuristics is used at each node expansion to choose one velocity in **SV** that becomes the operator in the node expansion. Naturally, in planning the motion in a real environment, the time interval Δt determines the level of confidence with which the trajectory can be executed and the ability to react to changes in the environment.

The trajectory resulting from these heuristics are conservative, since every trajectory segment is itself a safe trajectory over an infinite time horizon. This is also a drawback of this approach, since it excludes feasible trajectories whose velocities are outside of the safe sets for a limited time period.

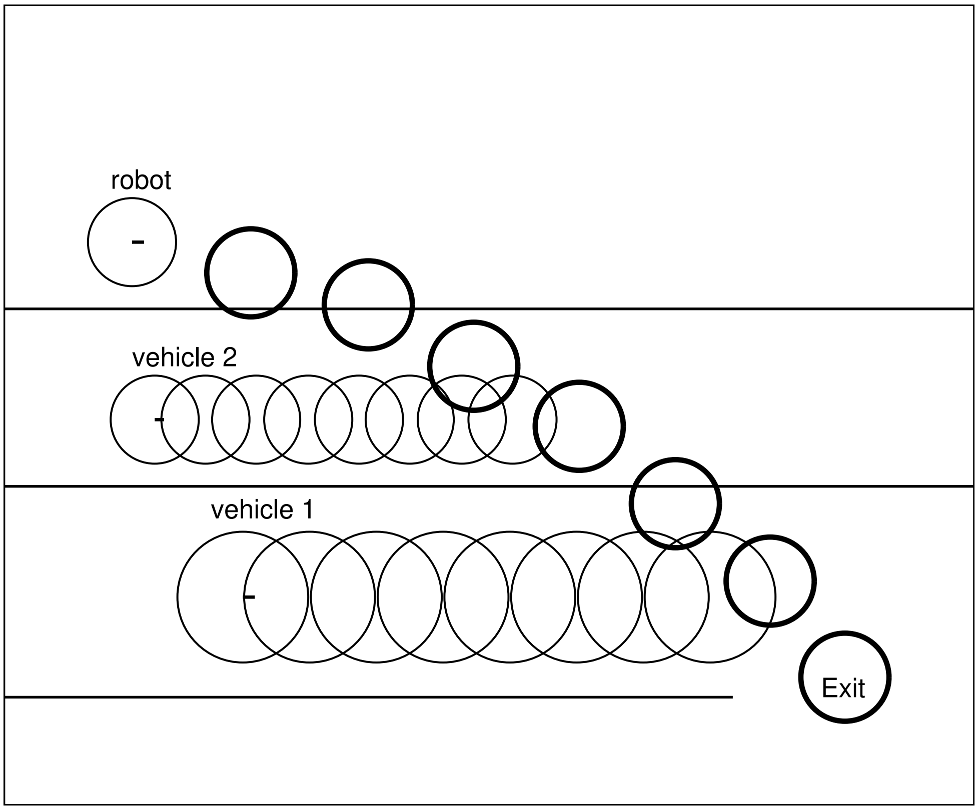

Figure 2.36: Trajectory computed with MV strategy.

The **TG** strategy has been applied to the test example of Figure 2.31 generating the trajectory shown in Figure 2.35. Along this trajectory, the robot slows down and lets vehicle 2 pass, and then speeds up towards the exit, behind vehicle 1. The total motion time for this trajectory is 60.7 s.

The solution computed with the **MV** heuristics is shown in Figure 2.36. Along this trajectory, the robot speeds up and passes in front of both vehicles 1 and 2. The total motion time along this trajectory is 35.6 s.

In the case of Figure 2.36, the **MV** heuristics managed to reach the goal. The trajectory computed using both **MV** and **TG** strategies is shown in Figure 2.37. Along this trajectory, the robot first speeds up to pass vehicle 2, then slows down to let vehicle 1 pass on, and then speeds up again towards the goal. The motion time for

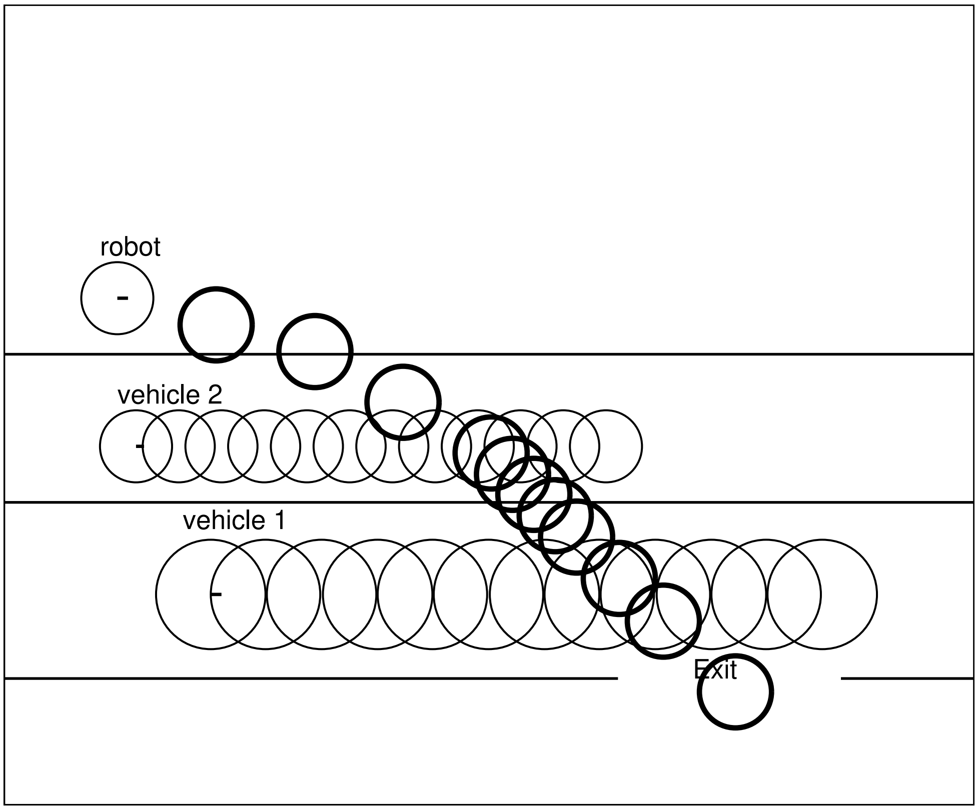

Figure 2.37: Trajectory computed with both TG and MV strategies.

this trajectory is 53.1 s. This trajectory was computed by using the **MV** heuristics for $0 \leq t \leq 20$ s and the **TG** heuristics afterwards.

Figure 2.38 illustrates the use of the safe velocity sets in the heuristic selection of the avoidance velocities. This figure shows the absolute cones of vehicle 1 and 2 at time $t = 20$ s of the trajectory shown in Figure 2.37, and the corresponding velocity obstacles as the two grey areas. Also shown as a solid line is the current velocity vector, and as a dashed line is the direction to the exit. Areas **BCDE** and **GHF** represent the safe velocity sets of the robot at that specific time. In this case, the line to the goal intersects the safe velocity set **BCDE**. This implies that, at this node, both **TG** and **MV** heuristic can be used to select the avoidance velocity.

The **TG** heuristics would choose the velocity along the line to the target, and

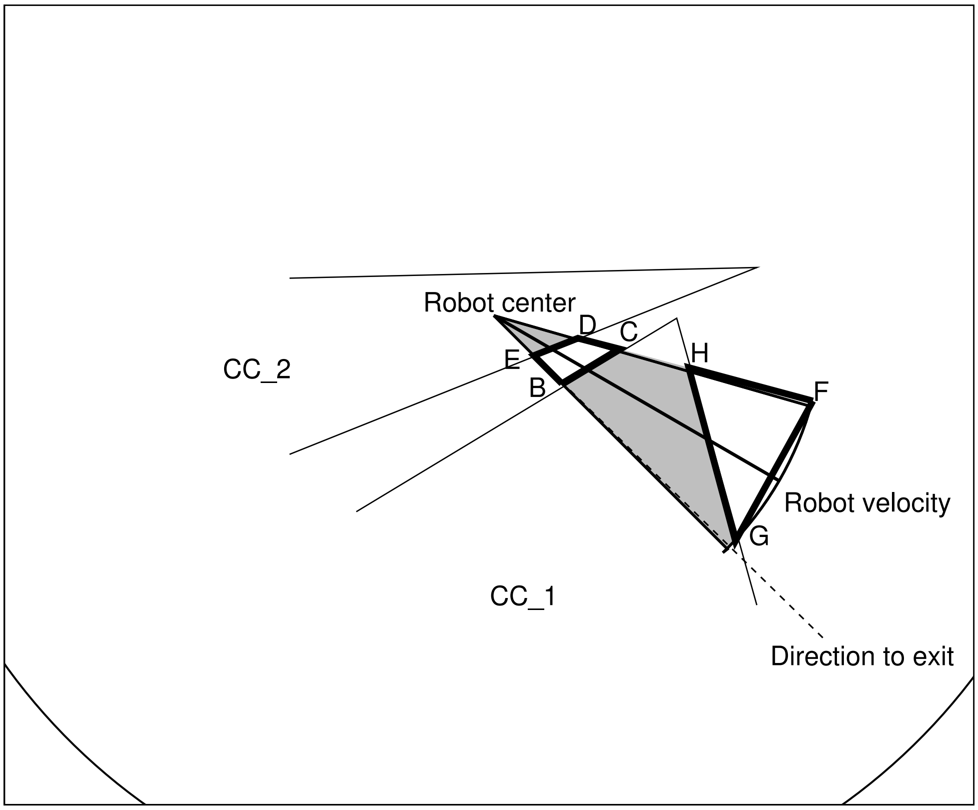

Figure 2.38: Safe velocity set in the (TG,MV) trajectory.

compute a slow trajectory, such as the one shown in Figure 2.35. The **MV** heuristics would select a larger velocity in **FGH**, pointing at some angle from the straight line to the goal, and would compute a trajectory such as the one shown in Figure 2.36. The trajectory shown in Figure 2.37 combines the characteristics of the two previous trajectories, using the **MV** heuristics as long as the velocity towards the goal is small, switching to the **TG** heuristics when the velocity towards the goal is above a preset threshold.

By observing the safe velocity set shown in Figure 2.38, it is possible to determine the type of each avoidance maneuver. The safe set, **BCDE**, is bounded by the front side of the absolute cone of vehicle , $\mathbf{CC}_2$, and by the rear side of the absolute cone of vehicle 1, $\mathbf{CC}_1$. Thus, the velocities in **BCDE** correspond to a front avoidance of

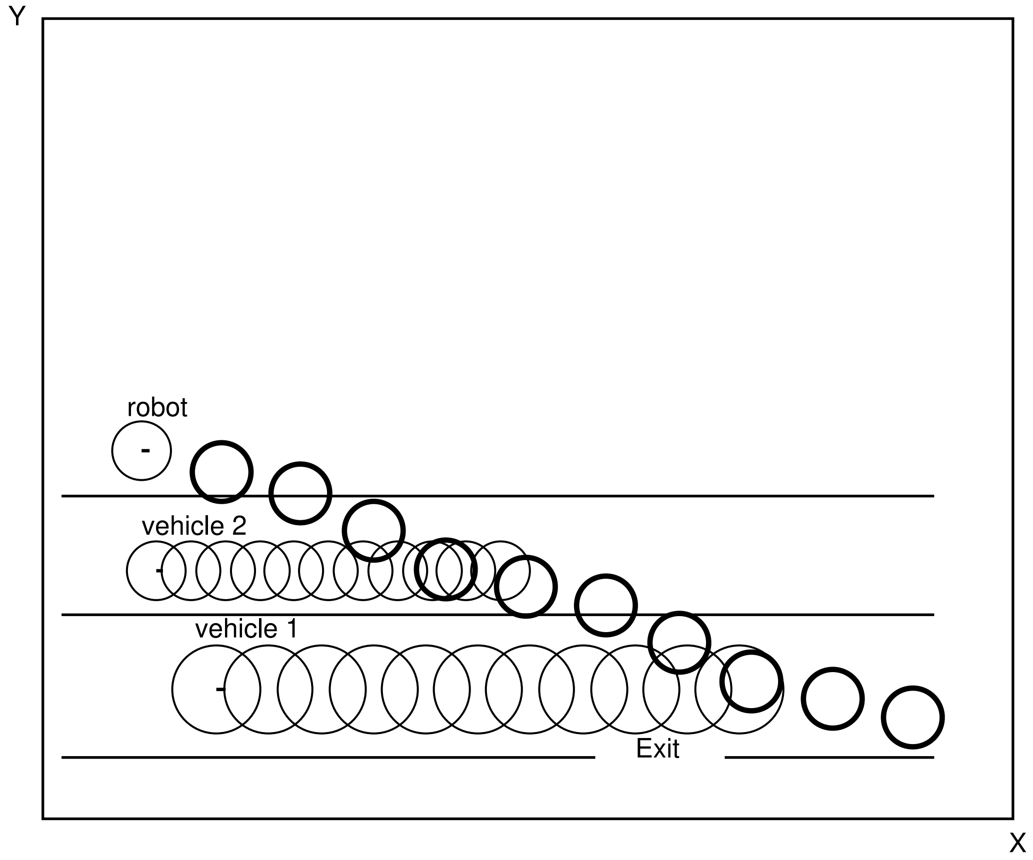

Figure 2.39: Failed trajectory computed with the MV heuristics.

vehicle 2 and to a rear avoidance of vehicle 1. The velocities in **FGH** correspond to the front avoidance of both vehicles 1 and 2. Recognizing front and rear maneuvers allows the planner to choose conservative and risky avoidance maneuvers. In this example, vehicle 1 may represent a big truck. It might be therefore safer to avoid vehicle 1 by passing behind it, as shown in Figure 2.37, rather than pass in its front, as shown in Figure 2.36.

The availability of different heuristic strategies is essential for finding a solution to a given motion planning problem. Since motion planning is an intractable problem, a particular heuristics does not guarantee that a solution can be found, even when such a solution exists. This fact is demonstrated in the example shown in Figure 2.39, where the **MV** strategy is for computing the solution. This example is similar to the one shown in Figure 2.36, with a small increase in the velocity of vehicle 1 to ($v_x = 3.3\ m/s,\ v_y = 0.0\ m/s$). This small change is sufficient to cause a failure of the search, and the trajectory only ensures the survival of the robot, but not the achievement of the lower priority goal of reaching the exit. Thus, motion planning in time-varying environments requires an ample set of fast tools to have a good chance of finding a satisfactory solution.

2.8.3 Conclusion

The complete trajectory can be computed with a global search or with a heuristic search, using the safe velocity sets defined by the velocity obstacles. In the specific examples discussed in this section, the **MV** heuristic strategy computed a solution whose completion time compares very well with the solution computed with a global search. Although this result depends on the specific example and cannot be generalized, it shows the power of an appropriate heuristic strategy. Furthermore, global search is not guaranteed to find the solution of a planning problem, since the **VO** planning algorithm is not complete. An other important feature of heuristic search is the ability of controlling the shape of the solution by using different strategies. This feature is often used in static planning, and the velocity obstacle approach extends its application to time-varying environments, thus enabling rapid adaptation to changes in the environment.

2.9 Summary

This chapter introduced the concept of Velocity Obstacle that is used to compute the solution of the Kinematic Problem in time-varying environments. The kinematic problem consists of planning a trajectory that satisfies the kinematic constraints of the problem, i.e. the avoidance of collisions with the obstacles, an approximation of the dynamic constraints, and reaches the target. The method proposed in this chapter consists of identifying the *Velocity Obstacle*, i.e. the velocity set that will produce a collision between the robot and each obstacle. Then the set difference between the set of velocities available to the robot, called the *Feasible Set*, and the velocity obstacle yields a new set, called the *Safe Velocity Set*, which consists of all feasible velocities that do not collide with any obstacle.

The analysis of the structure of the velocity obstacles permits to further refine the avoidance maneuver by considering the type of an avoidance maneuver, i.e. whether the robot would avoid each obstacle by passing in front or behind it. Thus a planner using the velocity obstacles would be able to specify the shape of the complete avoidance trajectory. The algorithm used to compute the velocity obstacles has been analyzed and results on its computational complexity and completeness have been presented. The basic ideas of velocity obstacles and safe velocity sets have also been extended to the three-dimensional space.

Several examples of planning with the velocity obstacle have been discussed. A planar avoidance maneuver illustrates the various phases of the algorithm, with the graphical display of the different sets. Similarly, an avoidance maneuver in 3D space is computed for several moving spheres. Finally, the complete solutions of the kinematic problem have been presented for an Intelligent Highway scenario. The complete trajectories has been computed using global and heuristic search, and the characteristics of the solutions have been discussed.

Chapter 3

The Dynamic Problem

The kinematic trajectory computed in the previous chapter is an approximation to the true solution of the time-varying motion planning problem, since the dynamic constraints have been modeled with the *Feasible Velocity* set. This simplification means that the actual trajectory executed by the robot may differ from the *kinematic trajectory* and that it may perform inefficiently. A way to overcome these limitations is to refine the kinematic trajectory with a dynamic optimization that would guarantee the avoidance of the obstacles and compute an optimal trajectory in the vicinity of the kinematic trajectory. This refinement is referred to as the Dynamic Problem of motion planning in time-varying environments.

The optimality of the trajectory is achieved here by minimizing its *completion time*, and therefore the solution of the dynamic problem is the minimum time trajectory for a given kinematic trajectory. As mentioned earlier in Section 1.4, motion time minimization is one of the goals in the hierarchy dealing with the intractability of planning in a time-varying environment. Then, the solution of the dynamic problem achieves one of those hierarchical goals. The optimal trajectory is also useful in the design and evaluation of heuristics for the kinematic problem, since kinematic trajectories can be compared to the optimal trajectory and their generating heuristics graded accordingly.

The dynamic optimization presented in this chapter is based on the necessary conditions of optimality derived from Pontryagin's Minimum Principle [43]. The time-varying state constraints due to the moving obstacles are transformed into state-dependent control constraints, and the planning problem is cast in a form compatible with Pontryagin's Principle. The optimal trajectory is computed with a steepest descent method modified to account for the moving obstacles [19].

The chapter is organized as follows. The first section formulates the time-varying motion planning problem as a minimum time problem, subject to state dependent control constraints. Next, Pontryagin's Minimum Principle is stated, and the necessary conditions of optimality are summarized. Then the numerical algorithm for computing the optimal solution is derived and shown to converge to a solution satisfying the necessary conditions. Finally, examples of optimal trajectories avoiding fixed and moving obstacles are discussed.

3.1 Problem Formulation

The motion planning problem can be formulated mathematically as follows. Find the control $u(t) \in \mathbf{U}$ in $t_0 \leq t \leq t_f$ that minimizes the performance index J:

$$J = \mathcal{G}(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))dt \tag{3.1}$$

subject to:

- **kinematic constraints**:
  - initial manifold:
$$\Gamma(\mathbf{x}(t_0), t_0) = 0 \tag{3.2}$$
  - terminal manifold:
$$\Omega(\mathbf{x}(t_f), t_f) = 0 \tag{3.3}$$

– time-varying obstacle constraints:

$$\Psi : \bigcup_{i=1}^{n} [S_i(\mathbf{x}(t), t) = 0] \tag{3.4}$$

- **dynamic constraints**:

– systems dynamics:

$$\dot{\mathbf{x}} = \mathcal{F}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{3.5}$$

– admissible controls:

$$\mathbf{U} = \{\mathbf{u} \mid u_i(min) \leq u_i \leq u_i(max)\} \tag{3.6}$$

For minimum time problems with specified boundary conditions, L and $\mathcal{G}$ of equation (3.1) are:

$$\begin{aligned} L(\mathbf{x}(t), \mathbf{u}(t)) &= 1 \\ \mathcal{G}(\mathbf{x}(t_f), t_f) &= 0 \end{aligned} \tag{3.7}$$

Then, the performance index J can be rewritten, using Mayer formulation, as a function of the terminal states and the terminal time:

$$J = t_f = \phi(\mathbf{x}(t_f), t_f) = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))dt \tag{3.8}$$

The classical approach to solving a constrained optimization problem such as this consists of appending the constraints to the performance index J, via suitable arrays of Lagrange multipliers. In this specific problem, this approach cannot be applied directly, because of the time-varying state constraints (3.4). State constraints can be adjoined directly to the cost functional of equation (3.8) [69], [35], or they can be differentiated with respect to time, until the controls $\mathbf{u}$ become explicit. These equations can then be appended as state-dependent control constraints to the functional

J [10], [19]. The number of differentiations of each constraint (3.4) represents the *order* of that constraint. With the latter approach, the state constraints are replaced by control constraints on each obstacle boundary. To ensure that the original state constraints are not violated, the solution is required to be tangent to the obstacles at the entry point on the boundary of each obstacle [69].

The latter approach may over-constrain the optimization problem since the solution is now forced to satisfy the state constraint as an equality along a finite arc on each obstacle boundary [35]. This may exclude solutions that are tangent to a state constraint only at isolated points. However, it has been shown that second order constraints, such as the circular obstacles treated here, do not suffer from this problem [35]. This method will therefore be used to solve the dynamic problem.

To demonstrate this treatment of the state constraints, consider, for simplicity, the constraint due to a single obstacle:

$$\Psi : S(\mathbf{x}(t), t) = 0 \tag{3.9}$$

Differentiating (3.9) p times with respect to time exposes the control $\mathbf{u}$, and denoting the pth derivative with $S^{(p)}$, the new control constraint becomes:

$$S^{(p)} : S(\mathbf{x}, \mathbf{u}) = 0 \tag{3.10}$$

where p is the order of the constraint.

Equation (3.10) is assumed to be active between time t_1, the entry time on the constrained arc on Ψ, and time t_2, the exit from the boundary of Ψ, $\partial\Psi$. To ensure that the original state constraint (3.9) is satisfied, all derivatives of Ψ, up to S^{p-1}, must be zero at t_1.

This leads to new constraints, consisting of a set of tangency conditions Ψ_1 at t_1, and the control constraint Ψ_2 on $\partial\Psi$:

$$\Psi_1 \quad : \quad \left(\begin{array}{c} S(\mathbf{x}(t), t) = 0 \\ \dot{S}(\mathbf{x}(t), t) = 0 \\ \vdots \\ S^{(p-1)}(\mathbf{x}(t), t) = 0 \end{array} \right) \qquad t = t_1 \tag{3.11}$$

$$\Psi_2 \quad : \quad S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \qquad t_1 \leq t \leq t_2 \tag{3.12}$$

The admissible control set is defined now by the original control constraints (3.6) and by the constraint (3.12), resulting in the new set $\mathcal{U}$:

$$\mathcal{U} : \left\{ \begin{array}{l} \mathbf{U} : \mathbf{u}(\mathbf{x})(min) \leq \mathbf{u} \leq \mathbf{u}(\mathbf{x})(max) \\ S^{(p)}(\mathbf{x}(t), u(t), t) = 0 \quad \text{when} \quad S(\mathbf{x}) = 0 \end{array} \right. \tag{3.13}$$

The set $\mathcal{U}$ is variable, with constraint Ψ_2 replacing one of the constraints in $\mathbf{U}$ for $t_1 \leq t \leq t_2$. The set of active constraints in $\mathcal{U}$ is represented by $\varphi(\mathbf{x}, \mathbf{u})$.

The optimization problem given by equations (3.1) to (3.13) is essentially a Three Point Boundary Value Problem, since its solution must satisfy the terminal manifold Ω at t_f, and be tangent to Ψ_1 at t_1. The optimal solution is reached when the following necessary conditions of optimality, derived from Pontryagin's Minimum Principle, are satisfied.

3.2 The Necessary Optimality Conditions

Pontryagin's Minimum Principle can be stated as follows [43].

Theorem 3.1: Given a dynamical system governed by the state equations:

$$\frac{dx_j}{dt} = \mathcal{F}_j(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m) \qquad j = 1, 2, \ldots, n$$

where $\mathbf{u} \in \mathbf{U}$ and $\mathbf{U} \subset E^m$, independent of $\mathbf{x}$ and t, with the performance index defined as:

$$J = \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t)) dt$$

from a given initial manifold Γ at time t_0 to a given terminal manifold Ω at time t_f, with t_f not prescribed, and defining

$$\mathcal{H}(\mathbf{\Lambda}, \mathbf{x}, \mathbf{u}) = \mathbf{\Lambda} \cdot \mathcal{F}(\mathbf{x}, \mathbf{u})$$

where $\mathbf{\Lambda}$ is a vector of Lagrange multiplier functions, then:

if $\mathbf{u}^*(t),\quad t_0 \leq t \leq t_f$, is an optimal control, then there exists a nonzero continuous vector function $\mathbf{\Lambda}(t) \in \Re^{n+1}$, satisfying the co-state equations:

$$\frac{d\Lambda_j}{dt} = -\sum_{i=0}^{n} \left. \frac{\partial \mathcal{F}_i(\mathbf{x}, \mathbf{u}^*(t))}{\partial x_j} \right|_{\mathbf{x}=\mathbf{x}^*(t)} \Lambda_i \qquad j = 0, 1, \ldots, n$$

with x_0 defined by $\frac{dx_0}{dt} = L(\mathbf{x}, \mathbf{u})$, such that:

$$\left.\begin{array}{lll} (a)\ \min_{\mathbf{u}\in\mathbf{U}} \mathcal{H}(\mathbf{\Lambda}(t), \mathbf{x}^*(t), \mathbf{u}) & = & \mathcal{H}(\mathbf{\Lambda}(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) \\ (b)\ \mathcal{H}(\mathbf{\Lambda}(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) & = & 0 \\ (c)\ \Lambda_0(t) & = & \text{constant} \geq 0 \end{array}\right\} \quad \forall t \in [t_0, t_f] \quad (3.14)$$

and $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_n(t)$ is normal to the end manifolds Γ and Ω at $t = t_0$ and $t = t_f$, respectively. □

The above theorem remains valid if the admissible controls are bounded, i.e. $\mathbf{u} \in \mathbf{U}$, but it does not include state and time dependent constraints, nor it admits a variable control set $\mathcal{U}$. The explicit dependency on time of equations (3.8) and (3.13) is easily removed by increasing the dimension of the state space by setting

$$x_{n+1} \equiv t$$

and by augmenting the system equations to:

$$\begin{aligned} \frac{dx_j}{dt} &= \mathcal{F}_j(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m) \qquad j = 1, 2, \ldots, n \\ \frac{dx_{n+1}}{dt} &= 1 \end{aligned}$$

This casts the motion planning problem in the time independent form required by theorem (3.1).

Then, the set of active control constraints, $\varphi(\mathbf{x}, \mathbf{u}) \subset \mathcal{U}$ at time t, $t_0 \leq t \leq t_f$, can be appended to the Hamiltonian as:

$$\mathcal{H}(\Lambda, \mathbf{x}, \mathbf{u}) = \Lambda^T(\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \tag{3.15}$$

where μ is a new vector of adjoint variables.

When $S^{(p)} = 0$, $t_1 \leq t \leq t_2$, where t_1 and t_2 are, respectively, the entry and the exit time on the constraint, the optimal solution is on a constrained arc, and then equation 3.14-a is replaced with:

$$\mathcal{H}_u^*(\Lambda(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) = 0 \qquad t_1 \leq t \leq t_2 \tag{3.16}$$

Using equation 3.16, μ is:

$$\mu = -\Lambda^T \mathbf{g}(\mathbf{x}) \left(\frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right)^{-1} \tag{3.17}$$

and the co-states λ satisfy the new adjoint equation, for $t_1 \leq t \leq t_2$:

$$\frac{d\mathbf{\Lambda}}{dt} = -\mathbf{\Lambda} \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \left(\frac{\partial \varphi}{\partial \mathbf{u}} \right)^{-1} \frac{\partial \varphi}{\partial \mathbf{x}} \right] \tag{3.18}$$

where φ is the active set of control constraints defined by equations (3.13). These new co-state equations satisfy the control constraints $\mathcal{U}$ ([43], p. 88).

The point constraints of equation (3.11) at time t_1 affect the multipliers by introducing the discontinuity given by [10]:

$$\mathbf{\Lambda}(t_1^+) = \mathbf{\Lambda}(t_1^-) + \eta \ \nabla \Psi_1(\mathbf{x}^*(t_1)) \tag{3.19}$$

where η is a vector of constant Lagrange multipliers.

Assuming normality of the solution, Λ_0 is normalized to unity [48], [70], and is henceforth dropped from the derivation.

These extensions make dynamic motion planning compatible with theorem (3.1) that can now be used to justify the following necessary conditions of optimality in the presence of time-dependent state constraints.

It is given the performance index J:

$$J = \phi(\mathbf{x}(t_f))$$

and a Hamiltonian function $\mathcal{H}$ defined as:

$$\mathcal{H}(\Lambda, \mathbf{x}, \mathbf{u}) = \Lambda^T(\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \tag{3.20}$$

where Λ and μ are vectors of adjoint variables, and $\varphi(\mathbf{x}, \mathbf{u}) \subset \mathcal{U}$ represents the set of active control constraints at time t, $t_0 \leq t \leq t_f$, i.e. the equations in (3.13) satisfied as equalities.

Then, the optimal control, $\mathbf{u}^*(t)$, generating the optimal solution, $\mathbf{x}^*(t)$, in the interval $t_0 \leq t \leq t_f$, minimizing J, and satisfying the fixed end manifolds Γ and Ω, satisfies the following necessary conditions [10], [8], [43].

1. Hamilton equations are verified:

$$\begin{aligned}
\dot{\mathbf{x}} &= \left(\frac{\partial \mathcal{H}}{\partial \Lambda}\right)^T && (3.21)\\
\Gamma(\mathbf{x}(t_0)) &= 0 && (3.22)\\
\Omega(\mathbf{x}(t_f)) &= 0 && (3.23)\\
\dot{\Lambda} &= -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^T && (3.24)\\
\Lambda^T(t_f) &= \left(\frac{\partial \phi}{\partial x} + \nu^T \frac{\partial \Omega}{\partial x}\right)_{t_f} && (3.25)
\end{aligned}$$

2. The transversality conditions are met at the terminal time t_f :

$$\left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H}^*\right)_{t_f} = 0 \tag{3.26}$$

3. The jump conditions are:

   - at the entry time on the state constraint t_1:

$$\Lambda^T_{t_1^-} = \Lambda^T_{t_1^+} + \eta^T \frac{\partial \Psi_1}{\partial x(t_1)} \tag{3.27}$$

- at the exit time from the state constraint t_2 :

$$\Lambda^T_{t_2^-} = \Lambda^T_{t_2^+} \tag{3.28}$$

4. The Hamiltonian is minimized:

$$\mathcal{H}^*(\Lambda, \mathbf{x}^*, \mathbf{u}^*) \leq \mathcal{H}(\Lambda, \mathbf{x}^*, \mathbf{u}) \tag{3.29}$$

5. The Hamiltonian along the optimal trajectory is:

$$\mathcal{H}^*(\Lambda(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) = 0 \qquad t_0 \leq t \leq t_f \tag{3.30}$$

where ν and η are two vectors of constant Lagrange multipliers.

When $S^{(p)} = 0$, equation (3.29) can be replaced with:

$$\mathcal{H}^*_u(\Lambda(t), \mathbf{x}^*(t), \mathbf{u}^*(t)) = 0 \qquad t_0 \leq t \leq t_f \tag{3.31}$$

and the adjoint equations (3.24) become:

$$\dot{\Lambda}^T = -\Lambda^T \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}(\mathbf{x})\mathbf{u}}{\partial \mathbf{x}} - \mathbf{g}(\mathbf{x}) \left(\frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right)^{-1} \frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right] \tag{3.32}$$

When the bounded controls appear linearly in the Hamiltonian, as they do when the optimal trajectory is not on $\partial\Psi$, the optimal controls are saturated according to:

$$u_i = \begin{cases} u_{max} & \text{if } (\mathcal{H})_{u_i} < 0 \\ u_{min} & \text{if } (\mathcal{H})_{u_i} > 0 \end{cases} \tag{3.33}$$

The quantity $\mathcal{H}_{u_i}$ is called the switching function for u_i.

In summary, the trajectory minimizing the performance index $J = t_f$ is characterized by the following: *(i)* its controls $\mathbf{u}^*$ are in the admissible set $\mathcal{U}$, *(ii)* the states satisfy the terminal manifolds Γ at t_0 and Ω at t_f, *(iii)* the co-state equations are described by (3.24) on the free arcs and by (3.32) on the constrained arcs, with a discontinuity at the junction of the two given by (3.27), and *(iv)* the Hamiltonian satisfies (3.30) over the entire interval.

These conditions can be used to check the optimal solution. In particular, they must be satisfied by the trajectory computed by the numerical method derived in the following section. It will be shown, that the steepest descent algorithm computes a solution satisfying the necessary conditions and, therefore, converges to the minimum time trajectory.

3.3 Computation of the Optimal Trajectory

In the previous sections, the time-varying motion planning problem has been formulated as a Three Point Boundary Value Problem, for which a set of necessary conditions of optimality has been presented. This is a difficult problem to solve analytically and, except for a few simple cases, it can only be solved numerically.

The approach presented here uses a gradient descent to compute the optimal controls, $\mathbf{u}^*$, starting from an initial guess $\mathbf{u}$ that is incrementally adjusted until the corrections fall below a preset threshold. The corrections are computed as functions of the errors in the terminal manifold Ω and in the intermediate constraint Ψ_1. Key to this approach is the backward integration of the Lagrange multipliers λ, whose initial conditions depend on the terminal and intermediate errors. Once the Hamiltonian $\mathcal{H}$ is computed with these multipliers, equation (3.30) yields the proper corrections.

A gradient descent is a local minimization procedure, and therefore it can only find a local minimum in the vicinity of the initial guess. This fact makes the selection of a suitable initial trajectory an essential part of the computation, since the optimal solution will exhibit most of its features. For the purpose of demonstrating this method, the guesses in this chapter are computed with Hermite splines and cycloidal interpolations, as described in Appendix B.

To develop the expressions for the control corrections, a single obstacle is assumed. The state constraint represented by the obstacle is converted into the point constraint Ψ_1 and the control constraint Ψ_2 of equations (3.11) and (3.12). The performance index is the completion time $J = t_f$.

The corrections to $\mathbf{u}$ are computed by setting the first variation of J equal to zero [19]. This approach eventually leads to an extremal for J and to vanishing control corrections. The derivation of the expressions for the control corrections is organized into three steps. First, dJ, $d\Omega$ and $d\Psi_1$ are computed without taking into consideration the effect of the constraints Ψ_1 and Ψ_2 on their respective multipliers. Then, the effect of these constraints on the multipliers is computed explicitly by using equation (3.27), and by modifying the co-state equations according to (3.32). Finally, dJ, $d\Omega$ and $d\Psi_1$ are combined into an augmented performance index $\underline{J}$, whose differential is minimized to compute the expression of the steepest descent increments. The differential $d\underline{J}$ is written in discrete form as a function of the switching times for the case of bang-bang controls. The corrections to the switching times are then derived explicitly for this last case.

3.3.1 The Differentials without the Obstacles

3.3.1.1 The Performance Index

Following the classic approach to constrained optimization, system dynamics and control constraints are adjoined to the performance index J via two vectors of Lagrange multiplier functions $\lambda_\phi(t) \in \Re^n$ and $\mu(t) \in \Re^k$, where n is the dimension of $\mathbf{x}$ and k is the number of active constraints in φ. The new performance index $\widetilde{J}$ is [10], [8]:

$$\widetilde{J} = \phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left[\lambda_\phi^T \left(\mathcal{F}(\mathbf{x}, \mathbf{u}) - \dot{\mathbf{x}}\right) + \mu^T \varphi(\mathbf{x}, \mathbf{u})\right] d\tau \tag{3.34}$$

Recalling the definition (3.20) of the Hamiltonian:

$$\mathcal{H}(\lambda, \mathbf{x}, \mathbf{u}) = \lambda_\phi^T \mathcal{F}(\mathbf{x}, \mathbf{u}) + \mu^T \varphi(\mathbf{x}, \mathbf{u})$$

equation (3.34) becomes:

$$\widetilde{J} = \phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left(\mathcal{H}(\lambda_\phi, \mathbf{x}, \mathbf{u}) - \lambda_\phi^T \dot{\mathbf{x}}\right) d\tau \tag{3.35}$$

For fixed initial time t_0, the differential $d\widetilde{J}$ is [9], [19]:

$$\begin{aligned} d\widetilde{J} &= \left(\frac{\partial \phi(\mathbf{x}(t_f))}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \phi(\mathbf{x}(t_f))}{\partial t} dt_f\right)_{t_f} + \varphi(\mathbf{x}(t_f), \mathbf{u}(t_f)) dt_f \\ &+ \int_{t_0}^{t_f} \left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}} \delta \mathbf{x} + \frac{\partial \mathcal{H}}{\partial \mathbf{u}} \delta \mathbf{u} - \lambda_\phi^T \delta \dot{\mathbf{x}}\right) d\tau \end{aligned} \tag{3.36}$$

Integrating by parts the term $-\int_{t_0}^{t_f} \lambda_\phi^T \delta \dot{x} d\tau$ yields:

$$-\int_{t_0}^{t_f} \lambda_\phi^T \delta \dot{\mathbf{x}} d\tau = -\left(\lambda_\phi^T \delta \mathbf{x}\right)_{t_f} + \left(\lambda_\phi^T \delta \mathbf{x}\right)_{t_0} + \int_{t_0}^{t_f} \dot{\lambda}_\phi^{\,T} \delta \mathbf{x} d\tau \tag{3.37}$$

From $d\mathbf{x} = (\delta \mathbf{x} + \dot{\mathbf{x}} dt)$ and to emphasize the expected discontinuity at t_1, the integral in equation (3.37) is split into two parts, t_0 to t_1^- and t_1^+ to t_f:

$$\begin{aligned} d\widetilde{J} &= \left(\lambda_\phi^T - \frac{\partial \phi}{\partial \mathbf{x}}\right)_{t_f} d\mathbf{x}(t_f) + \left(\frac{\partial \phi}{\partial t} + \mu^T \varphi + \lambda_\phi^T \dot{\mathbf{x}}\right)_{t_f} dt_f \\ &+ \int_{t_0}^{t_1^-} \left[\left(\dot{\lambda}_\phi^T + \frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right) \delta \mathbf{x} + \frac{\partial \mathcal{H}}{\partial \mathbf{u}} \delta \mathbf{u}\right] d\tau + \int_{t_1^+}^{t_f} \left[\left(\dot{\lambda}_\phi^T + \frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right) \delta \mathbf{x} + \frac{\partial \mathcal{H}}{\partial \mathbf{u}} \delta \mathbf{u}\right] d\tau \end{aligned} \tag{3.38}$$

The choice of

$$\dot{\lambda}_\phi(t) = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^T \tag{3.39}$$

$$\lambda_\phi(t_f) = \left(\frac{\partial \phi}{\partial \mathbf{x}}\right)_{t_f} \tag{3.40}$$

reduces $d\widetilde{J}$ to:

$$d\widetilde{J} = \left(\frac{\partial \phi}{\partial t} + \mathcal{H}\right)_{t_f} dt_f + \int_{t_0}^{t_f} \frac{\partial \mathcal{H}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau \tag{3.41}$$

3.3.1.2 The Terminal Constraint

The differential of the terminal constraint Ω can be computed using [9]:

$$(d\Omega)_{t_f} = \left(\frac{\partial \Omega}{\partial x} dx + \frac{\partial \Omega}{\partial t} dt\right)_{t_f} \tag{3.42}$$

Using $dx = \delta x + \dot{x}dt$ it follows that

$$\begin{aligned}(d\Omega)_{t_f} &= \left(\frac{\partial\Omega}{\partial x}(\delta x + \dot{x}dt) + \frac{\partial\Omega}{\partial t}dt\right)_{t_f} \\ &= \left(\frac{\partial\Omega}{\partial x}\delta x\right)_{t_f} + \left(\frac{\partial\Omega}{\partial x}\dot{x} + \frac{\partial\Omega}{\partial t}\right)_{t_f} dt_f \\ &= (\delta\Omega)_{t_f} + \dot{\Omega}_{t_f} dt_f \end{aligned} \tag{3.43}$$

The variation δx satisfies the first order perturbation equation:

$$\delta\dot{x} = \frac{\partial\mathcal{F}}{\partial x}\delta x + \frac{\partial\mathcal{F}}{\partial u}\delta u \tag{3.44}$$

Therefore, there exists a state transition matrix $\Phi(t,\tau)$ from which the variation $(\delta x)_{tf}$ can be expressed as [10]:

$$(\delta x)_{tf} = \Phi(t_f, t_0)\delta x(t_0) + \int_{t_0}^{t_f} \Phi(t_f,\tau)\frac{\partial\mathcal{F}}{\partial u}\delta u(\tau)d\tau \tag{3.45}$$

From this expression, $\delta\Omega$ is:

$$(\delta\Omega)_{t_f} = \left.\frac{\partial\Omega}{\partial x}\right|_{t_f} \left(\Phi(t_f, t_0)\delta x(t_0) + \int_{t_0}^{t_f} \Phi(t_f,\tau)\frac{\partial\mathcal{F}}{\partial u}\delta u(\tau)d\tau\right) \tag{3.46}$$

By defining a multiplier $\lambda_\Omega \in \Re^n \times \Re^l$, where n is the dimension of $\mathbf{x}$ and l is the number of the terminal constraints:

$$\lambda_\Omega^T(t) = \left(\frac{\partial\Omega}{\partial x}\right)_{t_f} \Phi(t_f, t) \tag{3.47}$$

and using the properties of the state transition matrix Φ [10]:

$$\dot{\Phi}(\tau, t) = -\frac{\partial f}{\partial \mathbf{x}}^T \Phi^T(\tau, t) \tag{3.48}$$

$$\Phi(t_f, t_f) = I \tag{3.49}$$

the adjoint equations for λ_Ω are:

$$\dot{\lambda}_\Omega(t) = -\left(\frac{\partial\mathcal{F}}{\partial x}\right)^T \lambda_\Omega(t) \tag{3.50}$$

with boundary conditions:

$$\lambda_\Omega(t_f) = \left(\frac{\partial \Omega}{\partial x}\right)_{t_f} \tag{3.51}$$

The variation of the terminal quantity Ω at time t_f, $(\delta\Omega)_{t_f}$ becomes:

$$(\delta\Omega)_{t_f} = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \delta\Omega_0 \tag{3.52}$$

Assuming fixed initial conditions, this expression for $(\delta\Omega)_{t_f}$ can be used in equation (3.43), and the total differential of Ω becomes:

$$d\Omega(t_f) = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Omega}{\partial x}\dot{x} + \frac{\partial \Omega}{\partial t}\right)_{t_f} dt_f \tag{3.53}$$

3.3.1.3 The Point Constraint

Similarly, the differential of the intermediate constraints Ψ_1 at time t_1, is expressed by:

$$d\Psi_1(t_1) = \int_{t_0}^{t_1} \lambda_\Psi^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Psi}{\partial x}\dot{x} + \frac{\partial \Psi}{\partial t}\right)_{t_1} dt_1 \tag{3.54}$$

With adjoint equations for $\lambda_\Psi \in \Re^n \times \Re^k$, where n is the dimension of $\mathbf{x}$ and k is the number of constraints Ψ_1, given by:

$$\dot{\lambda}_\Psi(t) = -\left(\frac{\partial \mathcal{F}}{\partial x}\right)^T \lambda_\Psi(t) \tag{3.55}$$

and boundary conditions:

$$\lambda_\Psi(t_1) = \left(\frac{\partial \Psi}{\partial x}\right)_{t_1} \tag{3.56}$$

The co-state equations defined above do not take into account the effects of the constraints Ψ_1 and Ψ_2 on λ_Ω and on λ_ϕ, which are treated next.

3.3.2 Effect of Point Constraints on the Multipliers

The discontinuity of the multipliers λ given by equation (3.27):

$$\Lambda_{t_1^-}^T = \Lambda_{t_1^+}^T + \eta^T \frac{\partial \Psi_1}{\partial x(t_1)}$$

affects the differentials $d\widetilde{J}$ and $d\Omega$. Multipliers λ_Ω are used here to illustrate the computation of this discontinuity [19].

The unknown η is computed by relating the value of λ_Ω at t_1^-, i.e. just before reaching the constraint Ψ_1, to its value at t_1^+, i.e. just after reaching Ψ_1. To do this, $\lambda_\Omega(t_1^+)$ and $\lambda_\Omega(t_1^-)$ are first computed independently of each other, using the expressions for $d\Omega$ at t_f and t_1. The expression of $\lambda_\Omega(t_1^-)$ involves the differential of t_1, dt_1, which is computed by considering the optimality of the trajectory at t_1, since the discontinuity of λ_Ω is meaningful only when $\Psi_1 = 0$ and $S^{(p)} = 0$. Finally, the multiplier η can be computed by substituting the values of $\lambda_\Omega(t_1^+)$ and $\lambda_\Omega(t_1^-)$ in the expression of $d\Omega(t_1)$.

The value of $\lambda_\Omega(t_1^+)$ is computed first. The differential of Ω at the terminal time t_f can be expressed in the two equivalent forms:

$$\begin{aligned} d\Omega(t_f) &= (\delta\Omega)_{t_f} + \dot{\Omega}_{t_f} dt_f \\ d\Omega(t_f) &= \left(\frac{\partial\Omega}{\partial x}dx + \frac{\partial\Omega}{\partial t}dt\right)_{t_f} \end{aligned}$$

By setting the variations of $\mathbf{u}$, $\delta\mathbf{u}(t)$, equal to zero for $t > t_1$, all the changes in $d\Omega(t_f)$ are function of the variations of $\mathbf{x}(t_1)$, $\delta\mathbf{x}_1$, and of dt_1. Therefore, the first form of $d\Omega(t_f)$ can be written as:

$$d\Omega(t_f) = \lambda_\Omega^T \delta\mathbf{x}(t_1^+) = \lambda_\Omega^T \left(d\mathbf{x}_1 - \dot{\mathbf{x}}dt_1\right)_{t_1^+} \tag{3.57}$$

Similarly, the second form of $d\Omega(t_f)$ can be rewritten as:

$$d\Omega(t_f) = \left(\frac{\partial\Omega}{\partial\mathbf{x}}d\mathbf{x} + \frac{\partial\Omega}{\partial t}dt\right)_{t_1^+} \tag{3.58}$$

These two expressions lead to the following identities:

$$\frac{\partial\Omega}{\partial\mathbf{x}_1} = \lambda_\Omega^T(t_1^+) \tag{3.59}$$

$$\frac{\partial\Omega}{\partial t_1} = -\lambda_\Omega^T(t_1^+)\dot{\mathbf{x}}(t_1^+) \tag{3.60}$$

An expression analogous to (3.57) is used to compute the value of $d\Omega$ at t_1^-:

$$d\Omega(t_1^-) = (\delta\Omega)_{t_1^-} + \dot{\Omega}_{t_1^-} dt_1^- \tag{3.61}$$

where:

$$(\delta\Omega)_{t_1^-} = (\lambda_\Omega \delta\mathbf{x})_{t_0} + \int_{t_0}^{t_1^-} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau$$

Since $\Psi_1 = 0$ and t_1 is minimal on the optimal trajectory, the value of dt_1 in (3.61) is computed from $dS^{(p-1)}(\mathbf{x}) = 0$. By setting $V = S^{(p-1)}$ and by using a new multiplier λ_V, the expression of dV at t_1^- is:

$$dV(t_1) = \left(\lambda_V^T \delta\mathbf{x}\right)_{t_0} + \int_{t_0}^{t_1} \lambda_V^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau + \dot{V} dt_1 \tag{3.62}$$

On the extremal solution, $dV(t_1) = 0$, and

$$dt_1 = \frac{1}{\dot{V}} \left[-\left(\lambda_V^T \delta\mathbf{x}\right)_{t_0} - \int_{t_0}^{t_1} \lambda_V^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} dt \right] \tag{3.63}$$

By replacing dt_1 in $d\Omega$ of equation (3.61) with 3.63, and since $\dot{V}$ and $\dot{\Omega}$ are both independent of the integration variable, $d\Omega(t_1^-)$ becomes:

$$d\Omega(t_1^-) = \left(\lambda_\Omega^T \delta\mathbf{x}\right)_{t_0} + \int_{t_0}^{t_1^-} \left(\lambda_\Omega^T - \frac{\dot{\Omega}}{\dot{V}} \lambda_V^T\right) \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau - \left.\frac{\dot{\Omega}}{\dot{V}}\right|_{t_1^-} \left(\lambda_V^T \delta\mathbf{x}\right)_{t_0} \tag{3.64}$$

The desired expression of λ_Ω at t_1^-, satisfying $dV = 0$ is then:

$$\lambda_{\Omega,V}^T(t_1^-) = \left(\lambda_\Omega^T - \frac{\dot{\Omega}}{\dot{V}} \lambda_V^T\right)_{t_1^-} \tag{3.65}$$

This equation can be further simplified. The expression for $\dot{\Omega}$ in equation (3.65) is

$$\dot{\Omega}(t_1^-) = \frac{\partial \Omega}{\partial \mathbf{x}_1} \dot{\mathbf{x}}(t_1^-) + \frac{\partial \Omega}{\partial t_1} \tag{3.66}$$

which, using equations (3.59) and (3.60), becomes:

$$\dot{\Omega}(t_1^-) = \frac{\partial \Omega}{\partial \mathbf{x}_1} \dot{\mathbf{x}}(t_1^-) - \lambda_\Omega^T(t_1^+) \dot{\mathbf{x}}(t_1^+) \tag{3.67}$$

Since the differentials $d\mathbf{x}_1$ and dt_1 are the same at t_1^- and t_1^+, equation (3.59) gives:

$$\left.\frac{\partial \Omega}{\partial \mathbf{x}_1}\right|_{t_1^-} = \lambda_\Omega^T(t_1^+) \tag{3.68}$$

and similarly

$$\lambda_V^T(t_1^-) = \left.\frac{\partial V}{\partial \mathbf{x}}\right|_{t_1} \tag{3.69}$$

By using equations (3.68), (3.69), and (3.67) in (3.65), the discontinuity of λ_Ω at t_1 becomes:

$$\lambda_{\Omega,V}^T(t_1^-) = \lambda_\Omega^T(t_1^+)\left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)} \left.\frac{\partial S^{(p-1)}}{\partial \mathbf{x}}\right|_{t_1}\right) \tag{3.70}$$

which is equivalent to constraint (3.27) if the multiplier η is equal to:

$$\eta^T = -\lambda_\Omega^T(t_1^+)\left(\frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1^-)}\right) \tag{3.71}$$

3.3.3 The Augmented Performance Index

The augmented performance index $\underline{J}$ is the quantity needed to compute the steepest descent increments for $\mathbf{u}$. The differential $d\underline{J}$ is obtained by appending the expression for $d\Omega$ and $d\Psi$ to $d\tilde{J}$ using constant multipliers η, and ν:

$$\begin{aligned} d\underline{J} &= \left[\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \left(\frac{\partial \phi}{\partial \mathbf{x}} + \nu^T \frac{\partial \Omega}{\partial \mathbf{x}}\right)\dot{\mathbf{x}} + \mu^T \varphi(\mathbf{x}, \mathbf{u})\right]_{t_f} dt_f \\ &+ \int_{t_0}^{t_1^-} \left[\frac{(\lambda_\phi^T + \nu^T \lambda_\Omega^T + \eta^T \lambda_\Psi^T)\mathcal{F} + \mu^T \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right] \delta \mathbf{u} d\tau \\ &+ \int_{t_1^+}^{t_f} \left[\frac{(\lambda_\phi^T + \nu^T \lambda_\Omega^T)\mathcal{F} + \mu^T \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right] \delta \mathbf{u} d\tau \end{aligned} \tag{3.72}$$

Since the constraints Ψ_1 are not affected by the states after t_1, the corresponding multipliers can be set to zero for $t > t_1$:

$$\lambda_\Psi(t) = 0 \quad \text{for} \quad t > t_1 \tag{3.73}$$

and a global Lagrange multiplier $\mathbf{\Lambda}$ is defined as:

$$\mathbf{\Lambda}^T = \lambda_\phi^T + \nu^T \lambda_\Omega^T + \eta^T \lambda_\Psi^T \tag{3.74}$$

The Hamiltonian can be written as in the necessary conditions (3.20):

$$\mathcal{H} = \mathbf{\Lambda}^T \mathcal{F} + \mu^T \varphi \tag{3.75}$$

Equation (3.73) becomes:

$$d\underline{J} = \left(\frac{\partial \phi}{\partial t} + \nu^T \frac{\partial \Omega}{\partial t} + \mathcal{H} \right)_{t_f} dt_f + \int_{t_0}^{t_1^-} \mathcal{H}_u \delta u d\tau + \int_{t_1^+}^{t_f} \mathcal{H}_u \delta u d\tau \tag{3.76}$$

with the Lagrange multipliers satisfying equations (3.32) and jump conditions (3.70).

3.3.4 The Bang-Bang Solution

The derivation of $d\underline{J}$ assumes that controls $\mathbf{u}$ are continuous functions subject to small variations. In the following, instead, the controls are assumed to be *bang-bang*, since, excluding singular arcs, the structure of the optimal control is indeed bang-bang [76]. The singular arcs, i.e. segments of the solution for which $\mathcal{H}_u = 0$ on a finite interval, are thus approximated by a finite number of switches [48].

3.3.4.1 The Steepest Descent Increments

Assuming bang-bang controls, the u_i take one of two values α_M or α_m, and change from one to the other only at the switching times $t_{i,j}$. For this reason, the variations in $\mathbf{u}$ are now given by:

$$\delta u_i = (\alpha_M - \alpha_m)\, sgn(dt_{i,j})$$

where sgn is the signum function, and $dt_{i,j}$ is the change of the jth switching time for control u_i. Obviously, $\delta u_i \neq 0$ only at the switching times. The specific expression for the δu_i depends on when the u_i switch sign. For example, if the switching strategy is such that the input values are switched from the highest value to the lowest value in the odd intervals, and from the lowest value to the highest one in the even intervals, then $\delta u_i = (-1)^{j-1} \Delta\alpha\, dt_{i,j}$, with $\Delta\alpha = \alpha_M - \alpha_m$.

In the discrete expression of $d\underline{J}$, the number of switches in the l segment of the trajectory for the rth control is represented by $s_{l,r}$, with $l = 1, \ldots, s_g$, $r = 1, \ldots, m$, m indicating the dimension of $\mathbf{u}$, and g the number of segments in the trajectory. The augmented performance index $d\underline{J}$ for bang-bang controls is replaced by the following summation over the switching time variations:

$$
\begin{aligned}
d\underline{J} \;=\;& \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}}(\mathcal{H}_{u_i})_{t_{ij}}\delta u_i dt_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{s_{2,i}}(\mathcal{H}_{u_i})_{t_{ij}}\delta u_i dt_{ij} \\
&+\sum_{i=1}^{m}\sum_{j=1}^{s_{3,i}}(\mathcal{H}_{u_i})_{t_{ij}}\delta u_i dt_{ij} + \left(\frac{\partial\phi}{\partial t} + \nu^T\frac{\partial\Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f
\end{aligned}
\tag{3.77}
$$

In this expression, the second segment is the arc on the obstacle boundary, and therefore the corrections $\delta\mathbf{u}$ are computed only for the controls not defined by $S^{(p)} = 0$.

In the bang-bang case, the controls are improved by changing their switching times by $dt_{i,j}$'s computed according to a chosen criterion. In the steepest descent method used here, the criterion for computing the $dt_{i,j}$ is the minimization of the differential $d\underline{J}$. Since the expression for $d\underline{J}$ is linear, a simple method for creating a minimum is to add quadratic terms in the independent variables to equation (3.78) [10]. With b a positive number and $\mathbf{W}$ a symmetric positive definite matrix, the augmented cost becomes:

$$
\begin{aligned}
d\widehat{J} \;=\;& \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}}((\mathcal{H}_{u_i})_{t_{ij}}\delta u_i dt_{ij} + \frac{1}{2}w_{ii}\Delta\alpha_i^2 dt_{ij}^2 \\
&+\sum_{i=1}^{m}\sum_{j=1}^{s_{2,i}}(\mathcal{H}_{v_i})_{t_{ij}}\delta u_i dt_{ij} + \frac{1}{2}w_{ii}\Delta\alpha_i^2 dt_{ij}^2 \\
&+\sum_{i=1}^{m}\sum_{j=1}^{s_{3,i}}(\mathcal{H}_{u_i})_{t_{ij}}\delta u_i dt_{ij} + \frac{1}{2}w_{ii}\Delta\alpha_i^2 dt_{ij}^2 \\
&+\left(\frac{\partial\phi}{\partial t} + \nu^T\frac{\partial\Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f + \frac{1}{2}b(dt_f)^2
\end{aligned}
\tag{3.78}
$$

The minimum of $d\widehat{J}$ is obtained for the values of the variables dt_i that make the first variation of equation (3.78) equal to zero. Thus the improvement of the jth switching time, t_{ij}, of control u_i, and of the terminal time t_f are:

$$
\begin{aligned}
dt_{ij} &= -\frac{(\mathcal{H}_{u_i})_{t_{ij}}}{w_{ii}\delta u_i} && (3.79)\\
dt_f &= -\frac{1}{b}\left(\mathcal{H} + \frac{\partial\phi}{\partial t} + \nu^T\frac{\partial\Omega}{\partial t}\right)_{t_f} && (3.80)
\end{aligned}
$$

The values of $dt_{i,j}$ and dt_f in equations (3.79) and (3.80) depend on the multipliers η and ν. These multipliers can be computed by requiring that dt_{ij} and dt_f satisfy equations (3.53) and (3.54), thus zeroing the differentials $d\Psi_1(t_1)$ and $d\Omega(t_f)$. Such a single step correction would violate the assumption of first-order approximation made earlier, and therefore $dt_{i,j}$ and dt_f can only reduce the non-zero differentials $d\Psi_1(t_1)$ and $d\Omega(t_f)$ by a small fraction. If k_1 and k_f are two small, positive numbers, the desired reduction of the differentials made by $dt_{i,j}$ and dt_f is:

$$
\begin{aligned}
-k_1 * (d\Psi_1)_{t_1} &= \epsilon && (3.81)\\
-k_f * (d\Omega)_{t_f} &= \theta
\end{aligned}
$$

Equations (3.53) and (3.54) are rewritten by replacing $d\Psi_1(t_1)$ and $d\Omega(t_f)$ with ϵ and θ, respectively, and solved in the unknowns η and ν:

$$
\begin{aligned}
\epsilon &= \left(\frac{\partial\Psi_1}{\partial t} + \lambda_\Psi^T\dot{x}\right)_{t_1^-} dt_1 + \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}}\left(\lambda_\Psi^T\frac{\partial\mathcal{F}}{\partial u_i}\right)_{t_{ij}}\delta u_i dt_{ij} && (3.82)\\
\theta &= \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}}\left(\lambda_\Omega^T\frac{\partial\mathcal{F}}{\partial u_i}\right)_{t_{ij}}\delta u_i dt_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{s_{2,i}}\left(\lambda_\Omega^T\frac{\partial\mathcal{F}}{\partial u_i}\right)_{t_{ij}}\delta u_i dt_{ij} && (3.83)\\
&\quad + \sum_{i=1}^{m}\sum_{j=1}^{s_{3,i}}\left(\lambda_\Omega^T\frac{\partial\mathcal{F}}{\partial u_i}\right)_{t_{ij}}\delta u_i dt_{ij}\left(\frac{\partial\Omega}{\partial t} + \lambda_\Omega^T\dot{x}\right)_{t_f} dt_f
\end{aligned}
$$

Algebraic manipulations lead to:

$$
\begin{aligned}
0 &= \epsilon + {}^1I_{\Psi\Psi}\eta + {}^1I_{\Psi\Omega}\nu + {}^1I_{\Psi\phi} && (3.84)\\
0 &= \theta + {}^1I_{\Omega\phi} + {}^1I_{\Omega\Omega}\nu + {}^1I_{\Omega\Psi}\eta + {}^2I_{\Omega\phi} + {}^2I_{\Omega\Omega}\nu + {}^3I_{\Omega\phi} + {}^3I_{\Omega\Omega}\nu\\
&\quad + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\phi}{dt}\right)_{t_f} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\Omega}{dt}^T\right)_{t_f}\nu
\end{aligned}
$$

from which:

$$\begin{aligned}
\eta &= -I_{\Psi\Psi}^{-1}(\epsilon + I_{\Psi\Omega}\nu + I_{\Psi\phi}) && (3.85)\\
\nu &= -\left({}^1I_{\Omega\Omega} + {}^1I_{\Omega\Psi}{}^1I_{\Psi\Psi}^{-1}{}^1I_{\Psi\Omega} + {}^2I_{\Omega\Omega} + {}^3I_{\Omega\Omega} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\Omega}{dt}^T\right)_{t_f}\right)^{-1} \\
&\quad \left(\theta + {}^1I_{\Omega\phi} - {}^1I_{\Omega\Psi}{}^1I_{\Psi\Psi}^{-1}\left(\epsilon + {}^1I_{\Psi\phi}\right) + {}^2I_{\Omega\phi} + {}^3I_{\Omega\phi} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\phi}{dt}\right)_{t_f}\right)
\end{aligned}$$

where the terms $I_{h,k}$ are defined as:

$$^lI_{h,k} = \sum_{i=1}^{m}\sum_{j=1}^{s_{l,i}}(\lambda_h^T \frac{\partial \mathcal{F}}{\partial u_i} w_{ii}^{-1} \frac{\partial \mathcal{F}}{\partial u_i}^T \lambda_k)_{t_{ij}} \tag{3.86}$$

with $h = \Psi_1, \Omega$, $k = \Psi_1, \Omega, \phi$, and $l = 1, 2, 3$. Index l indicates the segments of the trajectory before, on and after the state constraint, and index i on the constraint boundary indicates only the independent controls.

The numerical method just derived converges to a solution satisfying the necessary conditions of equations (3.21) to (3.30). In fact, the definition of the Hamiltonian given in equation (3.75) meets the one in the necessary condition (3.21). The adjoint equations (3.24) are satisfied since $\mathbf{\Lambda}$ is a combination of individual λ's, where λ_ϕ, λ_Ω and λ_Ψ satisfy equations (3.39) and (3.40), (3.50) and (3.51), and (3.56), respectively. When $dt_f = 0$ in equation (3.80), it follows that condition (3.26) is satisfied.

Equation (3.79) shows that the improvements in switching time t_j for control u_i, have opposite sign of $\partial\mathcal{H}/\partial u_i$, at time t_{ij}, meaning that each control improves in the opposite direction of the corresponding $\mathcal{H}_{u_i}$. When the increments dt_{ij} have converged to zero, also $\mathcal{H}_u$ at time t_{ij} is zero and the controls and $\mathcal{H}_u$ switch sign at the same time. This is true, though, only as long as $\mathcal{H}_{u_i}$ is not identically zero for a finite period of time. In this case, the singular arc is approximated with a finite number of switches, and the resulting trajectory is, therefore, near-time optimal. Finally, the derivation of the jump equation (3.70) ensures that condition (3.27) is verified, since all the variations of the components of $\mathbf{\Lambda}$ in equation (3.74) are computed without violating the constraints Ψ. The trajectory computed with this numerical method satisfies the necessary optimality conditions and therefore it is the local minimum time solution to the time-varying motion planning problem.

3.3.4.2 An Additional Velocity Constraint

When the optimal trajectory lies on the surface of an obstacle, its computation can be sped up by using an additional constraint on the magnitude of the velocity at the entry on the obstacle boundary. In general, the computation of the controls u_i satisfying constraints $\mathcal{U}$ of equation (3.13) requires several iterations, since the magnitude of the velocity satisfying the tangency conditions Ψ_1 of equation (3.11) at t_1 is too large. Constraints $\mathcal{U}$ are satisfied only when the tangent velocity at t_1 is below the *Velocity Limit Curve* of the robot on $\partial\Psi$ [4], [63], [65]. If the velocity is above the limit curve, the controls would exceed the limits in $\mathcal{U}$, and the robot will not track $\partial\Psi$.

The constraints Ψ_1 require only that the optimal trajectory at the entry point on $\partial\Psi$ be tangent to Ψ, and they do not set any limit on the magnitude of the tangency velocity. To avoid violating the control limits, the maximum admissible velocity at t_1, v_e, can be computed as a function of the coordinates of the entry point on $\partial\Psi$, and the constraint

$$\Psi_3: \quad v(t_1) \leq v_e \tag{3.87}$$

can be added to Ψ_1 and Ψ_2 of equations (3.11) and (3.12), to speed up the computation of a tangency velocity at t_1 that would track $\partial\Psi$.

The maximum velocity v_e is computed using a parametric representation of $\partial\Psi$ that allows to relate position and velocity of a point moving on $\partial\Psi$ with the dynamics of the robot [65], [54]. The parametric equations of the boundary of the obstacles are:

$$S(\mathbf{x}) : \begin{cases} x_{c,1} = g_1(s) \\ x_{c,2} = g_2(s) \end{cases} \tag{3.88}$$

where $s(t)$ is the arc length. The Cartesian velocity and acceleration $\dot{\mathbf{x}}_c$ and $\ddot{\mathbf{x}}_c$ on $\partial\Psi$, and the dynamic equations of the robot (A.13) are combined to express the velocity $\dot{s}$ and the acceleration $\ddot{s}$ on the boundary as:

$$\mathbf{M}(\mathbf{x}, s)\ddot{s} + \mathbf{B}(\mathbf{x}, s)\dot{s}^2 = \mathbf{u} \tag{3.89}$$

where θ are the joint angles of the manipulator and $\mathbf{u}$ its controls. The controls expressed by (3.89) can be combined with the first constraint of $\mathcal{U}$ of equation (3.13) to form the new constraints:

$$\mathbf{u}(\mathbf{x})(min) \leq \mathbf{M}(\mathbf{x}, s)\ddot{s} + \mathbf{B}(\mathbf{x}, s)\dot{s}^2 \leq \mathbf{u}(\mathbf{x})(max) \tag{3.90}$$

Then, the maximum velocity $\dot{s}$ on $\partial\Psi$ is computed from the above inequalities for every pair $(\mathbf{x}, s)$. The maximum Cartesian velocity v_e at $(\mathbf{x}, s)$ follows directly from $\dot{s}$ and the differential kinematics (A.7).

3.4 Examples

This section presents examples of optimal trajectories computed with the dynamic optimization just described for the two degree-of-freedom planar manipulator of Figure 3.1.

The problem is simplified by computing the collision free trajectory of the end effector only, treating the planar manipulator as a SCARA robot with the links placed above the plane of the obstacles. Only the tip of the manipulator reaches the plane of the obstacles by means of a prismatic joint in the z-axis direction, as shown in Figure 3.1-b. The kinematic and dynamic equations of this manipulator are given in Appendix A.

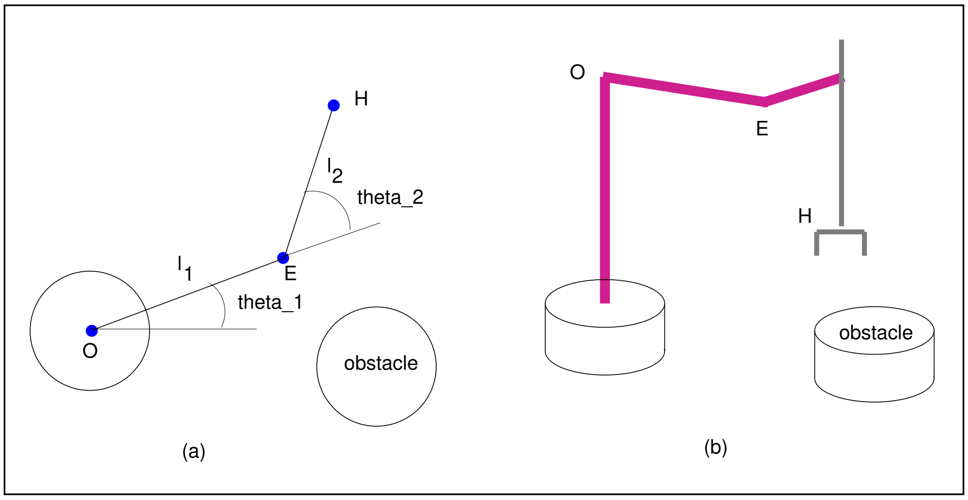

Figure 3.1: Planar 2-dof Manipulator: a) top view, b) side view

The objective in the following examples is to move the robot from a starting position $\mathbf{S}$: $\mathbf{x} = (-.15\ m,\ .55\ m)$, $\mathbf{v} = (0, 0)$ to a goal position $\mathbf{G}$: $\mathbf{x} = (1.5\ m,\ -.5\ m)$, $\mathbf{v} = (0, 0)$. In the first example, no obstacle is present, in the second example there is one static obstacle between start and goal, and, in the last example, the obstacle moves across the plane. In all cases, the robot configuration is chosen to have the *elbow-up* posture.

The initial guess for the dynamic optimization in all three examples is shown in Figure 3.2. This trajectory have been computed with the method described in Appendix B. The path is the Hermite spline between $\mathbf{S}$ and $\mathbf{G}$ with zero terminal velocities, and represented by equations B.1 in the parameter s:

$$x(s) = a_3\, s^3 + a_2\, s^2 + a_1\, s + a_0$$
$$y(t) = b_3\, s^3 + b_2\, s^2 + b_1\, s + b_0$$

The velocity profile is computed by representing the parameter s with a cycloidal interpolation in the parameter t and represented by equations (B.9):

$$\omega = \frac{2.0\pi}{T}$$

$$
\begin{aligned}
s &= \frac{\omega t - \sin(\omega t)}{2.0\pi} \\
\dot{s} &= \frac{\omega(1 - \cos(\omega t))}{2.0\pi} \\
\ddot{s} &= \frac{\sin(\omega t)\omega^2}{2.0\pi}
\end{aligned}
$$

where s, $\dot{s}$ and $\ddot{s}$ are respectively the value of the spline parameter, its first time derivative, its second time derivative, and T is the initial guess of the motion time. The marks on the trajectory of Figure 3.2 are drawn at a constant interval of .4 s and show the velocity profile of the robot motion. The bang-bang controls for the dynamic optimization are computed by discretizing the continuous torques given by the inverse dynamics of the robot and summarized in Appendix A. The bang-bang controls are shown in Figure 3.3 and the corresponding trajectory of the manipulator tip is shown in Figure 3.4. This trajectory is the initial guess for the dynamic optimization. The manipulator is represented by a small circle drawn on the trajectory every .5 s. The two small black circles represent the start and the goal of the trajectory.

The controls for the solution in the free environment are shown in Figure 3.5, and the corresponding trajectory is shown in Figure 3.6. Figure 3.7 shows the Hamiltonian as a function of time, and Figure 3.8 shows the switching functions $\frac{\partial \mathcal{H}}{\partial u_1}$ and $\frac{\partial \mathcal{H}}{\partial u_2}$. The trajectory of Figure 3.6 is thought to be close to the optimal solution since it satisfies with good approximation the necessary conditions of optimality, and it has a similar shape and a shorter motion time than the solution computed by the optimization program described in [63].

The motion time of the trajectory shown in Figure 3.6 is reduced from the initial guess of 4.0 s to 3.59 s, and the number of switching times is significantly smaller than the initial guess. Control u_2, shown in Figure 3.6, has converged to a single switch, as required by the switching function $\frac{\partial \mathcal{H}}{\partial u_2}$, as shown in Figure 3.8. Control u_1 should have converged to the two switches indicated by the switching function $\frac{\partial \mathcal{H}}{\partial u_1}$, as shown in Figure 3.8. Instead, control u_1 shows six switches in the first 1.0 s of the trajectory. This fact may indicate the presence of a singular arc in the trajectory. In the first 1.0 s of the trajectory in fact, the shoulder joint $\mathbf{O}$ moves very little, thus indicating that this part of the trajectory is almost independent of u_1 and therefore creating a singular arc. This conclusion is consistent with the results given in [48], where trajectories similar to the one in Figure 3.6 created multiple switches in one of the controls.

The approximation of a singular arc with a finite number of switches leads to the computation of a *near-optimal* solution. The sub-optimality of the solution also appears in the plot of the Hamiltonian $\mathcal{H}$, shown in Figure 3.8, that is not zero in the first part of the trajectory, and in the difference in the zero-crossings of the switching functions shown in Figure 3.8, and the actual switches of the controls, shown in Figure 3.5.

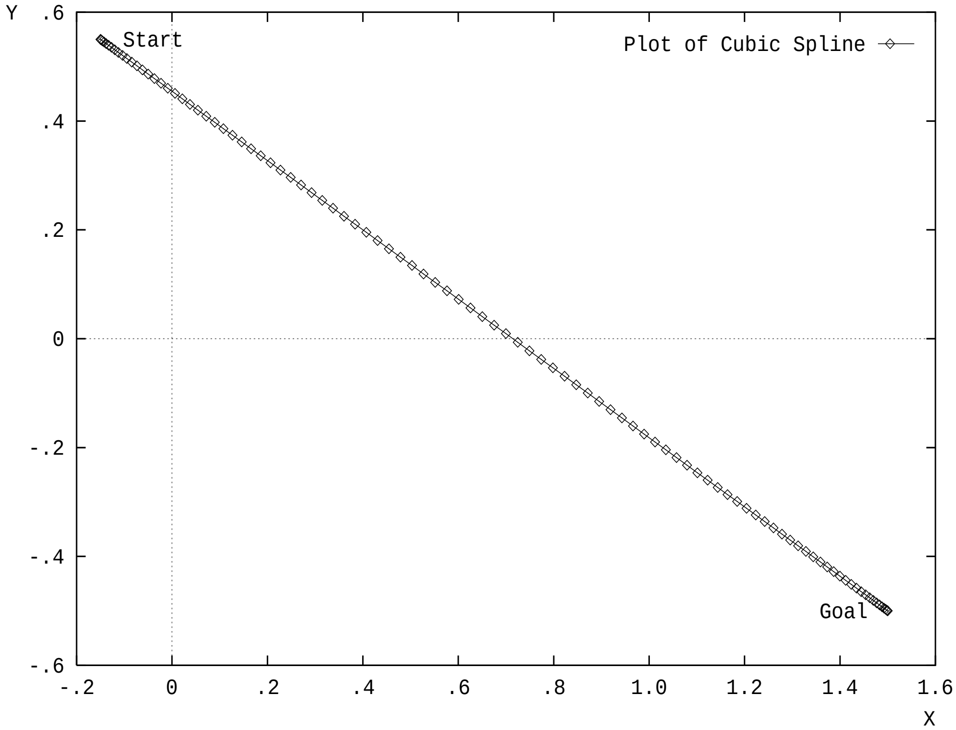

Figure 3.2: Hermite spline between start and goal

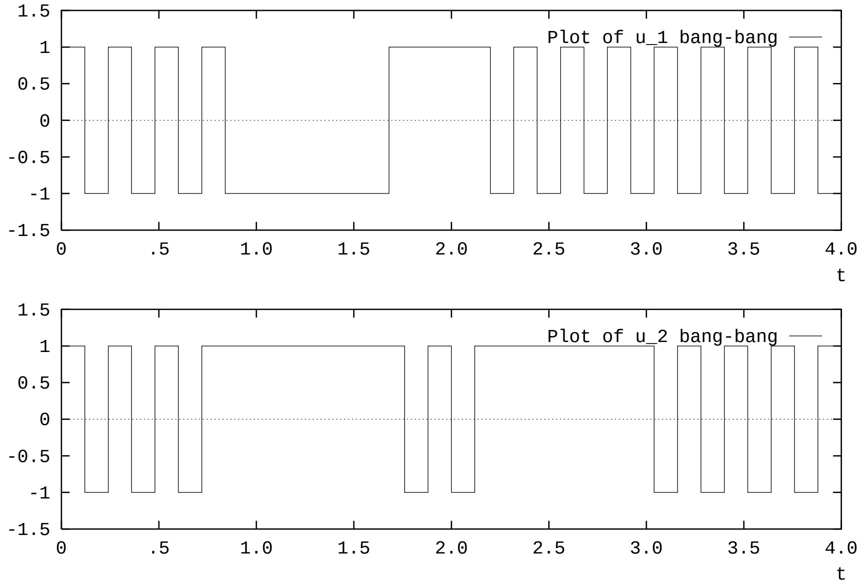

Figure 3.3: Controls for the initial guess

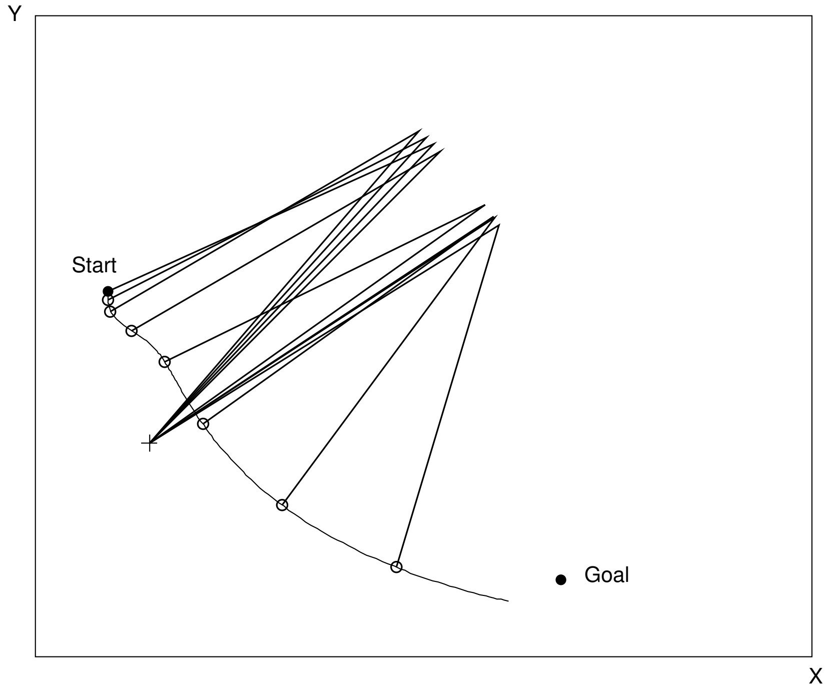

Figure 3.4: Initial guess for the dynamic optimization

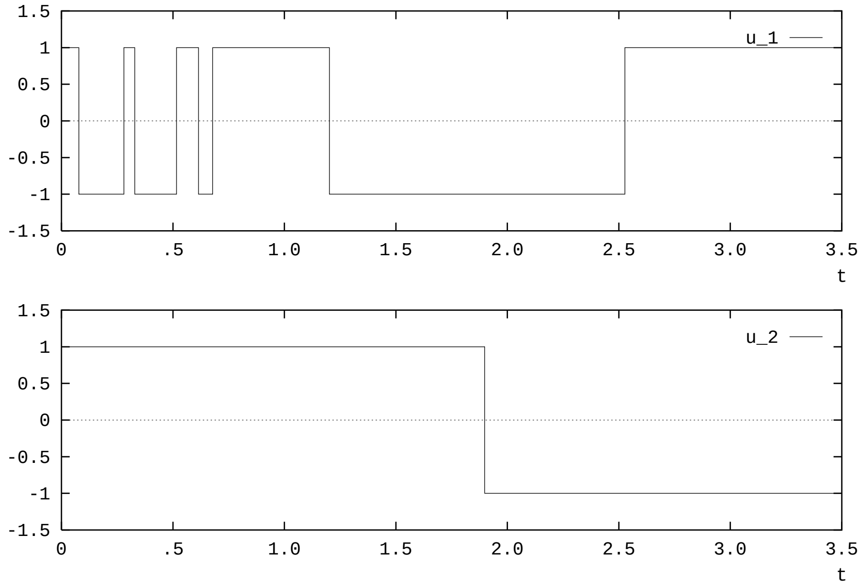

Figure 3.5: Optimal controls in the free environment

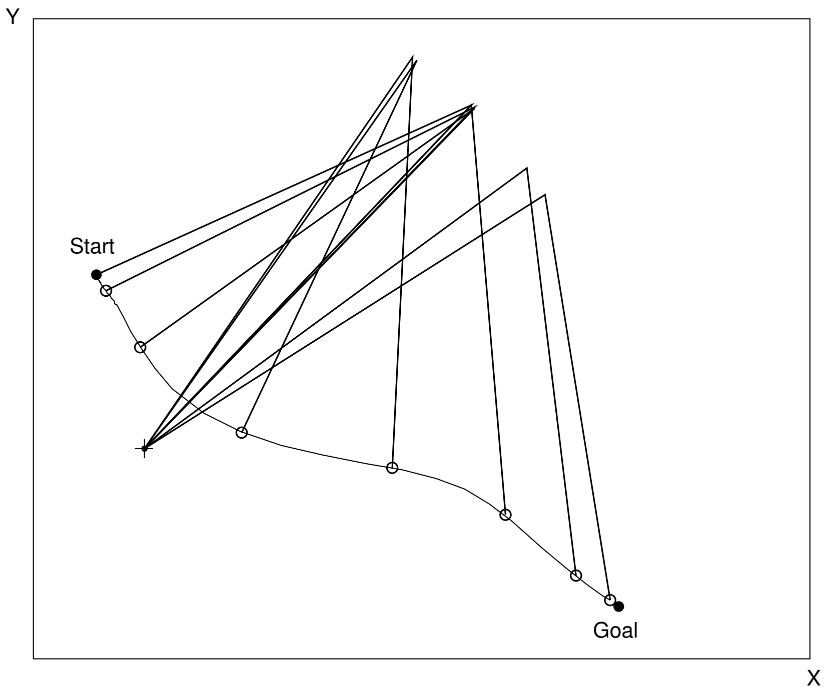

Figure 3.6: Optimal trajectory in the free environment

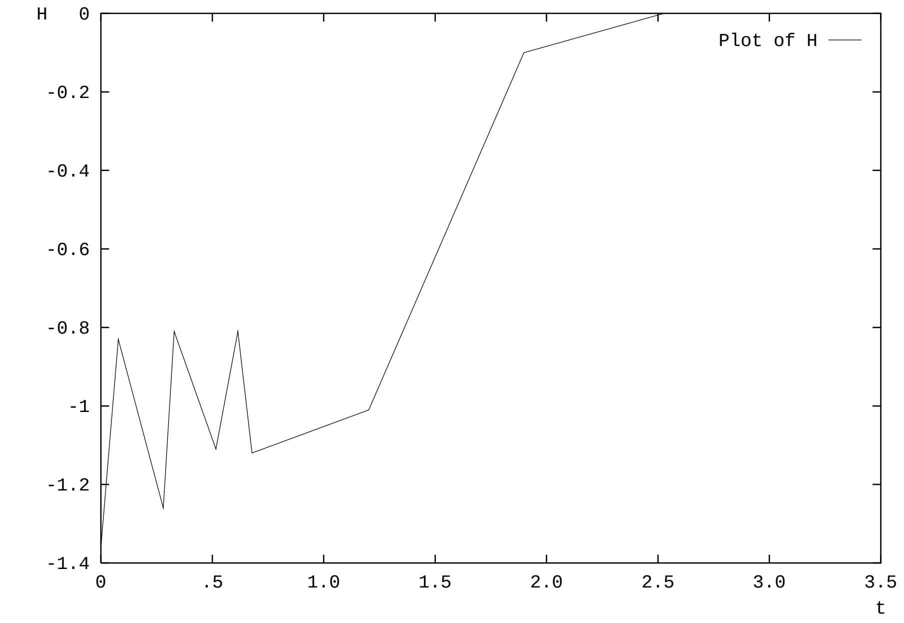

Figure 3.7: Hamiltonian for trajectory in the free environment

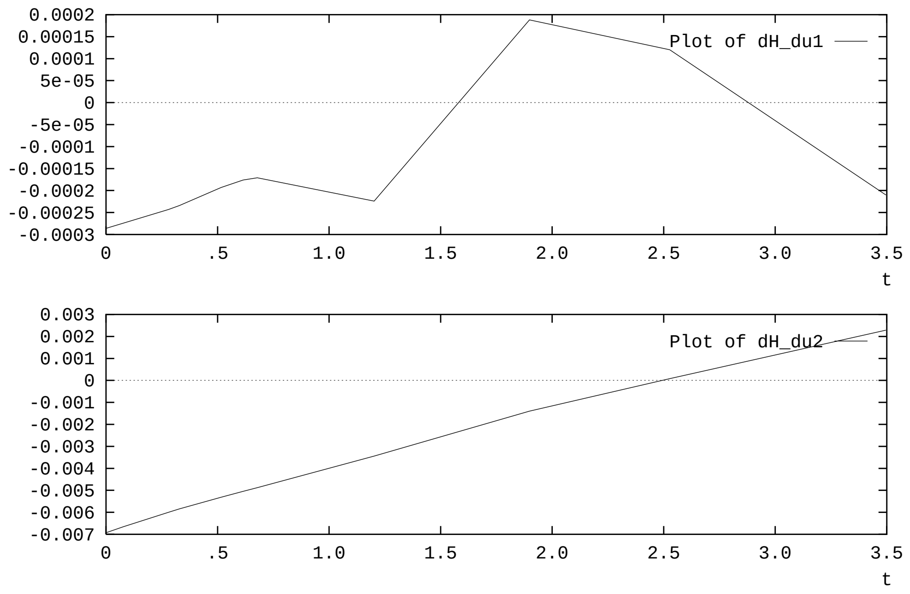

Figure 3.8: Plot of $\frac{\partial \mathcal{H}}{\partial \mathbf{u}_2}$ for the free environment

The second example has a static obstacle between start and goal represented by the circle at $\mathbf{C} = (.6\ m,\ -.2\ m)$ with radius $r = .4\ m$. The constraints Ψ_1 and Ψ_2 due to this obstacle are:

$$\begin{array}{lcll} \Psi_1 & : & \left(\begin{array}{rcl} (x-x_o)^2 + (y-y_o)^2 - r^2 & = & 0 \\ (x-x_o)v_x + (y-y_o)v_y & = & 0 \end{array} \right) & t = t_1 \\ \Psi_2 & : & v_x^2 + (x-x_o)a_x + v_y^2 + (y-y_o)a_y = 0 & t_1 \leq t \leq t_2 \end{array}$$

The controls of the solution are shown in Figure 3.9, and the corresponding trajectory is shown in Figure 3.10. Figure 3.11 shows the profile of the Hamiltonian, and Figure 3.12 shows $\frac{\partial \mathcal{H}}{\partial u_1}$ and $\frac{\partial \mathcal{H}}{\partial u_2}$. The completion time of this trajectory is 5.17 s.

The solution is tangent to the obstacle approximately at time $t = 2.5\ s$, and then it continues towards the goal without following the boundary of the obstacle. The controls shown in Figure 3.9 are characterized by the presence of numerical glitches both in u_1 and u_2. The spikes in u_1, approximately at times $t = .6\ s,\ .8\ s,\ 3.2\ s,\ 3.4\ s$, and in u_2, at time $t = 2.5\ s$, would disappear, had the optimization been continued for more iterations. Neglecting these spikes, the number of switches in the controls is consistent with the switching functions $\frac{\partial \mathcal{H}}{\partial u_1}$ and $\frac{\partial \mathcal{H}}{\partial u_2}$ shown in Figure 3.12.

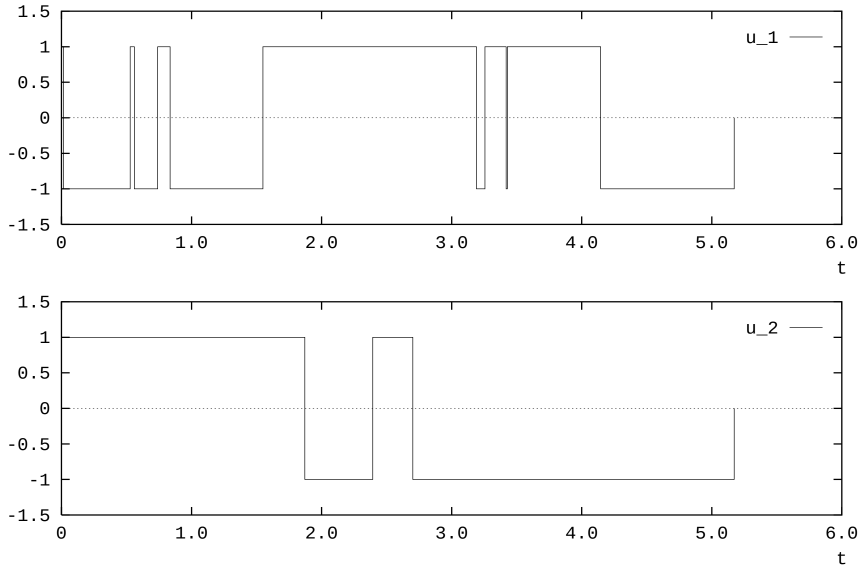

Figure 3.9: Optimal controls with a fixed obstacle

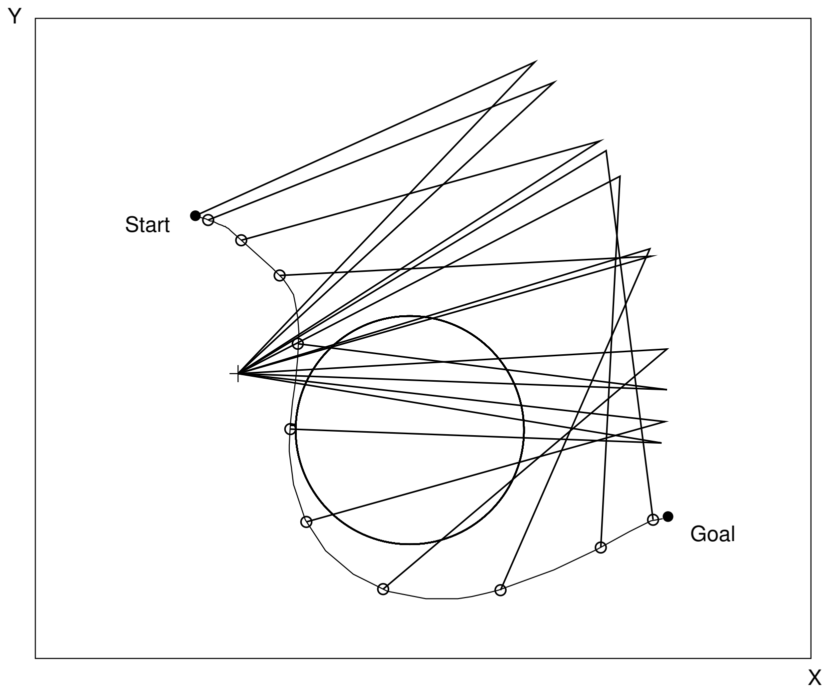

Figure 3.10: Optimal trajectory with a fixed obstacle

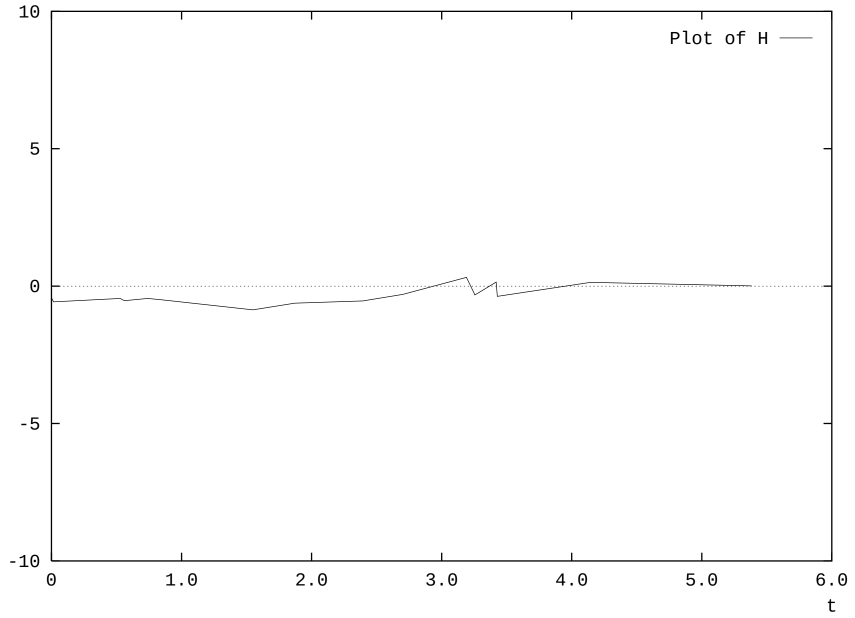

Figure 3.11: Hamiltonian of the trajectory with fixed obstacle

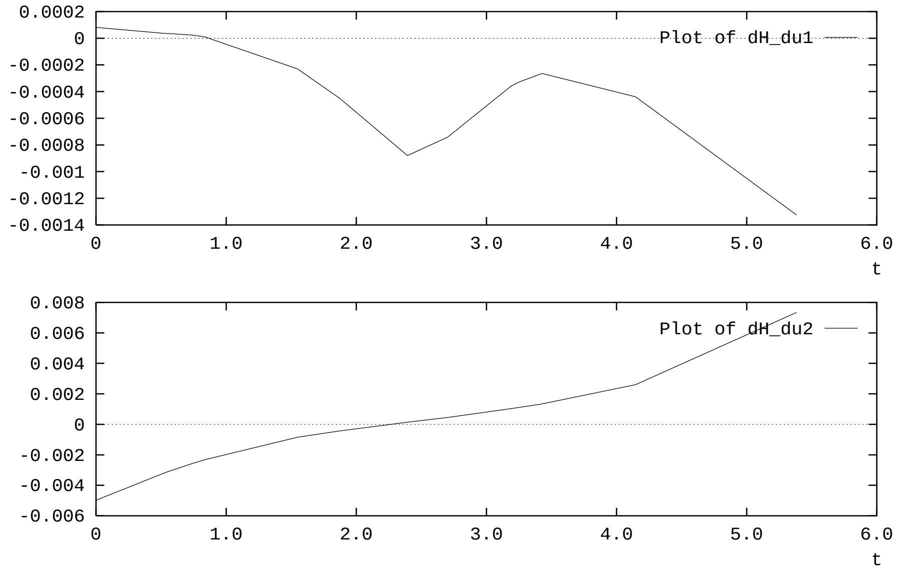

Figure 3.12: Plot of $\frac{\partial \mathcal{H}}{\partial \mathbf{u}}$ with fixed obstacle

The solution of a second case of avoidance of a static obstacle has been computed to examine the effect of the obstacle Ψ when its radius is increased. Figure 3.13 shows the solution computed when the obstacle is located at $\mathbf{C} = (.8\ m,\ -.15\ m)$ and has radius $r = .6\ m$. In this case, the trajectory follows the obstacle boundary, and has a completion time of $5.38\ s$.

The most evident qualitative differences between the trajectory shown in Figure 3.10 and the trajectory shown in Figure 3.13 are the trajectory segment approaching the obstacle, and the constrained arc on the obstacle boundary. The shape of the approach segment of the solution shown in Figure 3.13 reminds of the trajectory in the free environment shown in Figure 3.6, as if the intermediate constraint Ψ_1 were acting as the terminal constraint. The solution shown in Figure 3.10 shows instead that this particular obstacle can be avoided with a smoother maneuver, as if the avoidance were within the dynamic capabilities of the manipulator.

Finally, the third example has a moving obstacle crossing the free space. The constraints Ψ_1 and Ψ_2 are now:

$$\begin{array}{lcll} \Psi_1 & : & \left(\begin{array}{ccc} (x-(v_{ox}t+x_o))^2+(y-(v_{oy}t+y_o))^2-r^2 & = & 0 \\ (x-(v_{ox}t+x_o))(v_x-v_{ox}) & & \\ +(y-(v_{oy}t+y_o))(v_y-v_{oy}) & = & 0 \end{array} \right) & t=t_1 \\ \Psi_2 & : & \begin{array}{c} ((v_x-v_{ox})^2+(x-(v_{ox}t+x_o))a_x+ \\ ((v_y-v_{oy})^2+(y-(v_{oy}t+y_o))a_y \quad = \quad 0 \end{array} & t_1 \le t \le t_2 \end{array}$$

Figure 3.14 shows the optimal trajectory of the manipulator tip and the obstacle, at $.5\ s$ intervals. Its controls are shown in Figure 3.15. Figure 3.16 shows the Hamiltonian, and Figure 3.17 shows the switching functions $\frac{\partial \mathcal{H}}{\partial u_1}$ and $\frac{\partial \mathcal{H}}{\partial u_2}$. The motion time of this solution is $t = 4.36\ s$.

The shape of this trajectory is consistent with the solutions of the previous examples, with the manipulator tip reaching the boundary of the obstacle approximately at time $t = 2.5\ s$, and then staying close to the obstacle until the path to the goal

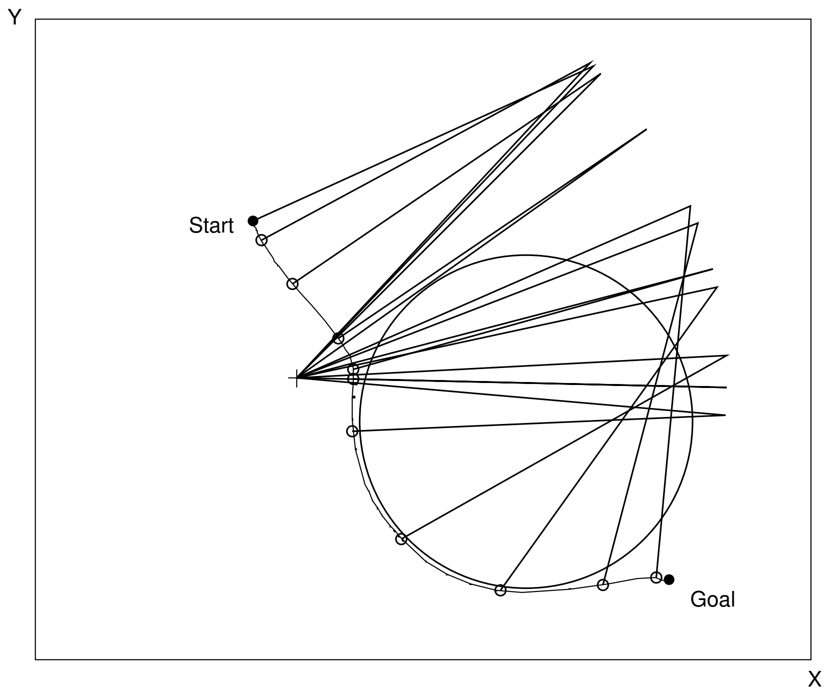

Figure 3.13: Optimal trajectory with a large static obstacle

is clear. The control u_1, shown in Figure 3.15, has more switches than those indicated by the switching function $\frac{\partial \mathcal{H}}{\partial u_1}$ of Figure 3.17. In particular, the switches at time $0 \leq t \leq 1.0\ s$ may indicate the presence of a singular arc, as in the solution of the first example. The switches at time $3.5\ s \leq t \leq 4.4\ s$ would disappear if the optimization had been allowed more iterations. A similar argument applies also to control u_2, shown in Figure 3.15, where the extra switch would disappear with more iterations of the dynamic optimization.

The observation of Figure 3.14 shows that the tip of the manipulator is penetrating slightly the obstacle, as represented by the small black bars appearing at time $2.5\ s \leq t \leq 2.8\ s$ and time $3.0\ s \leq t \leq 3.5\ s$. These small trajectory errors are due to the optimistic assumption of complete knowledge of the environment. A more realistic planner would combine this algorithm with sensor-based reactive control, which would compensate in real-time the errors in the environment model.

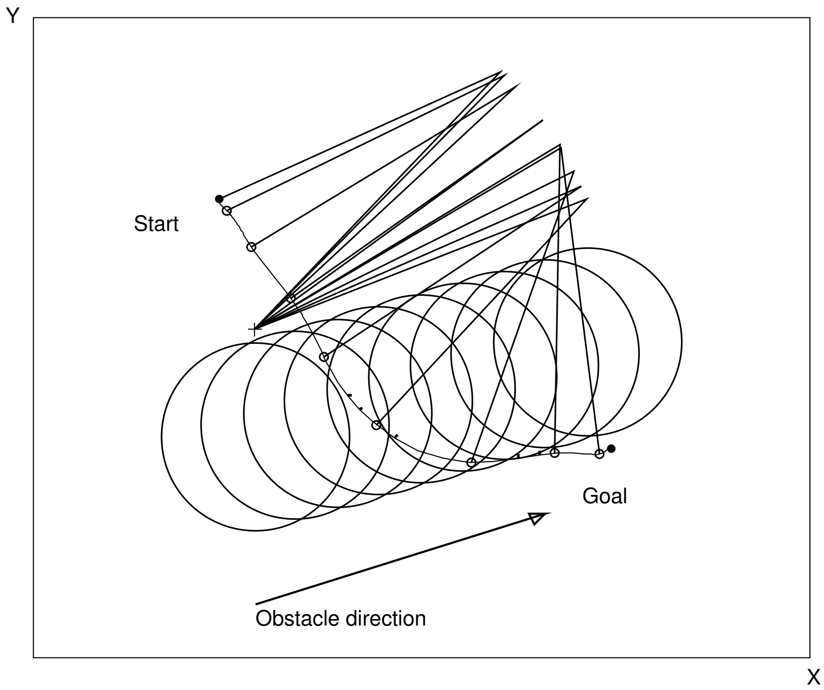

Figure 3.14: Optimal trajectory with a moving obstacle

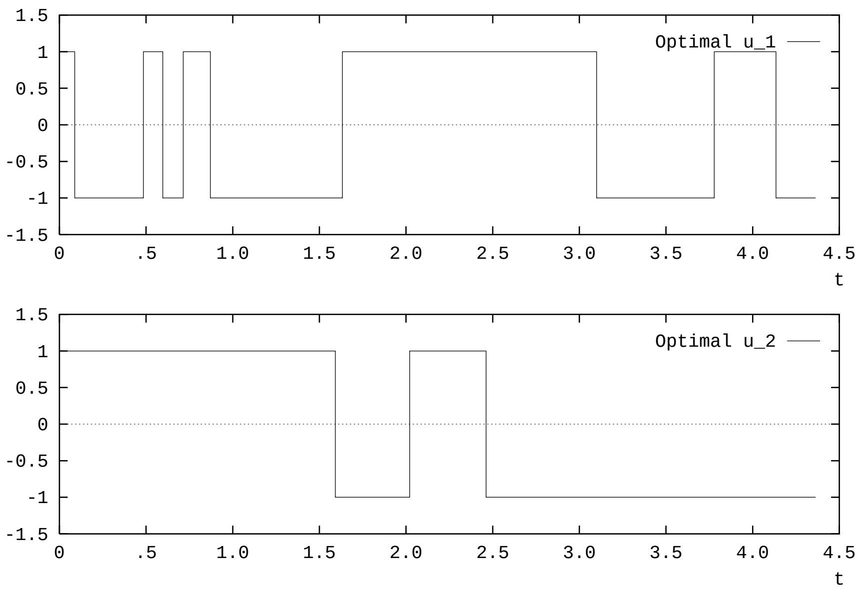

Figure 3.15: Optimal controls with a moving obstacle

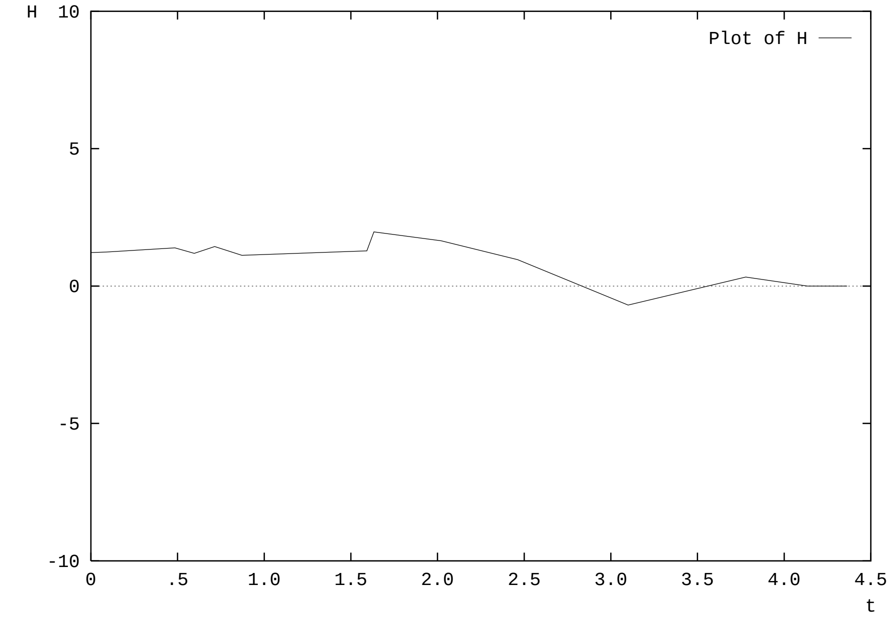

Figure 3.16: Hamiltonian of the trajectory with a moving obstacle

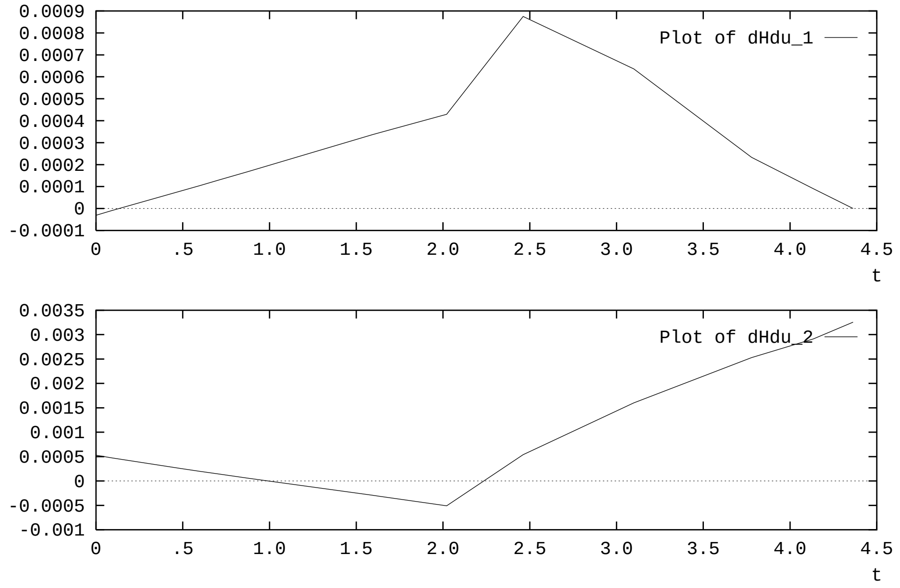

Figure 3.17: Plot of $\frac{\partial \mathcal{H}}{\partial \mathbf{u}}$ with a moving obstacle

3.5 Summary

This chapter presented the computation of the time-optimal trajectory of a manipulator, subject to system dynamics, control and state inequality constraints. The approach followed has been to transform the state inequality constraints into a tangency constraint at the entry point of the obstacle boundary, followed by a control constraint. A gradient algorithm has been developed for incrementally computing the optimal trajectory satisfying the terminal conditions and the intermediate point constraint. The steepest descent assumes bang-bang controls improving the switching times with corrections that are shown to lead to the time-optimal solution.

The tests carried out as well as the examples presented in the previous section show that this approach computes good solutions for the cases of free environment, a single static obstacles, and a single moving obstacle. This approach, as used in the examples, is not appropriate for more complex complex environments, since it does not provide any mean to control the shape of the trajectory and the relation between the robot and the obstacles. In the next chapter, this problem will be addressed by using as initial guess of the optimization the kinematic trajectories computed with the method presented in Chapter 2.

Chapter 4

Examples of Motion Planning

4.1 Introduction

This chapter uses the procedures developed earlier to compute the trajectory of a robotic system moving in a time-varying environment. The motion planning problem is divided into two simpler problems, the *kinematic problem* and the *dynamic problem*, that are solved efficiently. The kinematic problem refers to the computation of a trajectory satisfying the kinematic constraints and an approximation of the dynamic constraints of the system. The dynamic problem refers to the computation of the time-minimal solution of the problem, subject to the true constraints, that uses the solution of the kinematic problem as its initial guess. The kinematic problem is solved using the Velocity Obstacle method described in Chapter 2 and the dynamic optimization is carried out with the numerical algorithm presented in Chapter 3.

The solution of the kinematic problem is computed as a sequence of positions and avoidance velocities given at constant time intervals. The initial guess expected by the dynamic optimization consists of the bang-bang controls for the actuators of the robot. There is then the need of an intermediate step to convert the output of the velocity obstacle algorithm into the nominal trajectory expected by the optimization. The nominal trajectory for the optimization is computed by interpolating the avoidance velocities with a sequence of splines. The resulting continuous trajectory is used to compute the controls using the inverse dynamics of the robot. The controls are then discretized to produce the expected bang-bang controls.

This process is presented in the following sections by discussing a few examples of motion planning. Next section gives the analytical expression of the spline interpolation of the kinematic trajectory. Then, the following section discusses an example of trajectory planning among intelligent vehicles. Next section presents an example of planning for a robotic manipulator avoiding two moving obstacles. Finally, the last section summarizes the chapter.

4.2 Computation of the Nominal Trajectory

A kinematic trajectory computed with the Velocity Obstacle method consists of a sequence of straight line segments and velocities assigned to each segment. Since the abrupt velocity change between segments is not acceptable, the trajectory is first smoothed by computing a spline interpolation. Each trajectory segment i is divided into the two half i_1 and i_2, and the spline interpolation is applied to the trajectory segments i_2 and $(i+1)_1$. In this way the velocity discontinuities are replaced by a polynomial blend.

Because of the availability of position and velocity at the end points of each trajectory segment, the interpolation chosen is a third order Hermite polynomial [25]. The Hermite spline is described, following Appendix B, by the polynomials:

$$x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \qquad (4.1)$$

$$y(t) = b_3 t^3 + b_2 t^2 + b_1 t + b_0 \qquad (4.2)$$

The coefficients a_i and b_i for the generic segment i are computed by requiring that the polynomials satisfy the following conditions:

$$
\begin{aligned}
x(t_i) &= x_m(i) \;\text{ midpoint of segment } i && (4.3)\\
y(t_i) &= y_m(i) && (4.4)\\
v_x(t_i) &= v_x(i) \;\text{ velocity in segment } i && (4.5)\\
v_y(t_i) &= v_y(i) && (4.6)\\
x(t_{i+1}) &= x_m(i+1) \;\text{ midpoint of segment } i+1 && (4.7)\\
y(t_{i+1}) &= y_m(i+1) && (4.8)\\
v_x(t_{i+1}) &= v_x(i+1) \;\text{ velocity in segment } i+1 && (4.9)\\
v_y(t_{i+1}) &= v_y(i+1) && (4.10)
\end{aligned}
$$

where t is the time computed by the VO trajectory. For the initial and final splines of the interpolation, the mid-points $(x_m(i), y_m(i))$ and $(x_m(i+1), y_m(i+1))$ are replaced respectively by the starting point of the first segment and by the ending point of the last segment.

The computation of the bang-bang controls, follows the procedure outlined in Appendix B. First, the actuator torques are computed from the accelerations given by the spline interpolation, using the inverse system dynamics. Then, the torques are discretized by fixing an arbitrary threshold. The maximum bang-bang value is assigned to the motor torques when the interpolated torques are above the threshold. Similarly, the minimum value is assigned to the motor torques when the interpolated torques are below the threshold. This discretization introduces an approximation in the solutions, and the integration of the bang-bang controls produces a trajectory that differs from the initial kinematic solution.

In this thesis, the selection of the threshold has been done manually, by ensuring that the integrated trajectory captures the structure of the kinematic trajectory, i.e. maintains the same type of avoidance of the obstacles. The selected threshold contributes to the solution of the dynamic problem, since the numeric optimization improves the initial sequence of control switches. The solution of the time-varying planning problem is therefore the local minimum that is function of the kinematic trajectory and of the number of control switches determined by the threshold.

4.3 The Intelligent Highway Scenario

The example presented in this section is similar to those described in Section 2.8 and shown in Figure 2.31. In this example, the robot and vehicle 2 have an initial velocity of ($v_x = 3.0\ m/s,\ v_y = 0.0$), and vehicle 1 has an initial velocity of ($v_x = 3.3\ m/s,\ v_y = 0.0\ m/s$). A solution of this example is shown in Figure 4.1, and it has been computed using the combination of the **MV** and **TG** heuristics described earlier. The motion time of the trajectory shown in Figure 4.1 is 45.7 s, and each vehicle is drawn every 5 seconds. For the purpose of this example, the dynamics of the robot is modeled with a double integrator.

The kinematic trajectory is actually computed as the path and the velocity profile shown respectively in Figure 4.2 and in Figure 4.3. The marks on the path shown in Figure 4.2 are drawn at constant time intervals and visualize the different velocity of each segment. The result of the spline interpolation and the original trajectory are shown in Figure 4.4, where the marks on the spline trajectory are again drawn at constant time intervals. This interpolation achieves the velocities computed by the kinematic trajectory at the mid-point of each segment. Figure 4.5 shows the velocity profile of the spline trajectory, which qualitatively follows the velocity profile of the kinematic trajectory.

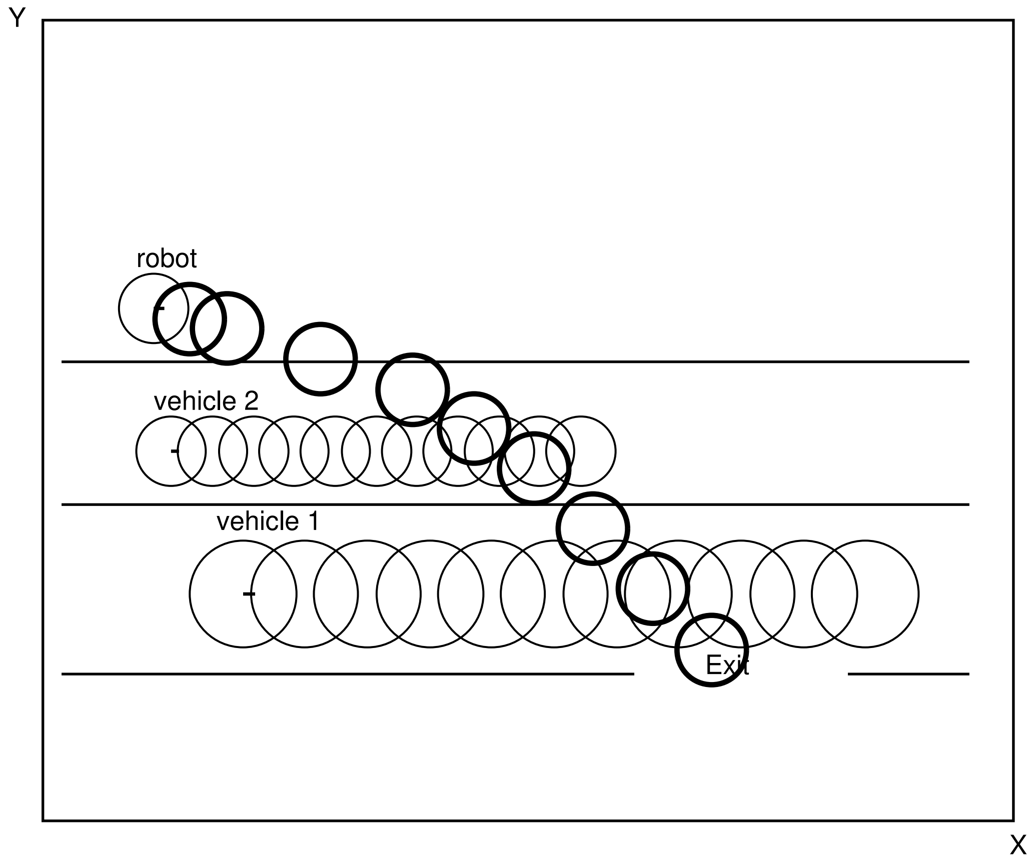

Figure 4.1: Trajectory computed with the (MV,TG) heuristics.

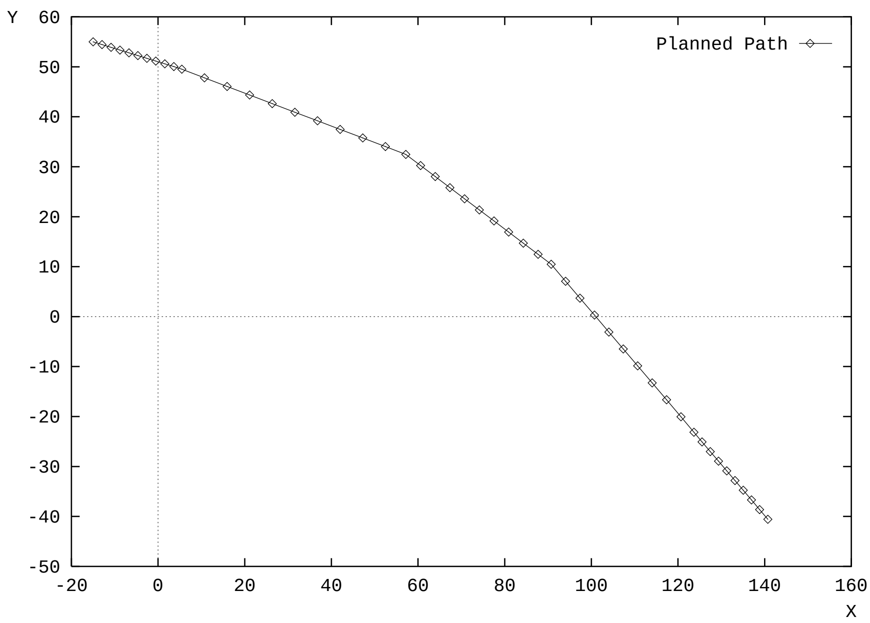

Figure 4.2: Trajectory planned by the heuristic search

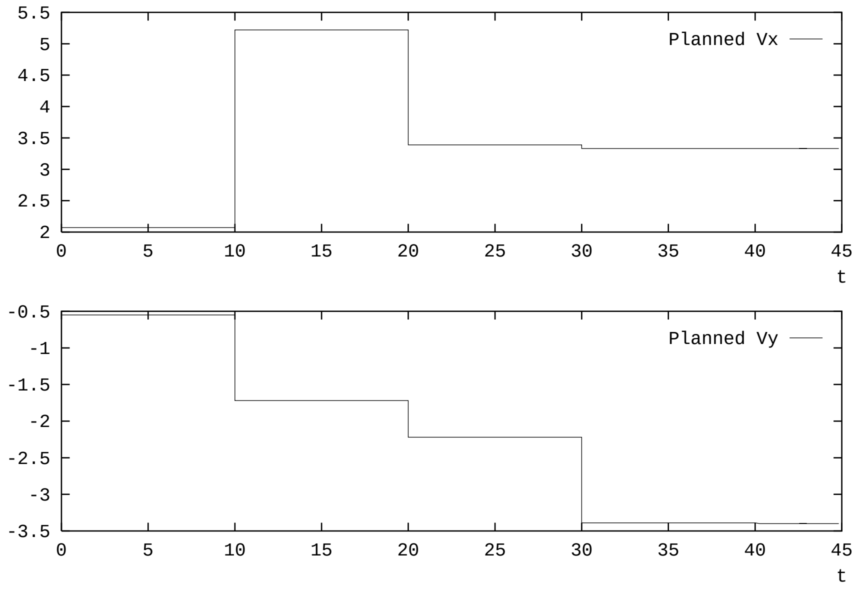

Figure 4.3: Velocities planned by the heuristic search

Figure 4.6 shows the profile of the acceleration of the spline trajectory. These accelerations are also the controls for the simplified model of the robot used in this example. The bang-bang controls are computed by thresholding the accelerations, as shown in Figure 4.7. Figure 4.8 shows the trajectory obtained by integrating these controls. This trajectory is the initial guess used by the dynamic optimization. In Figure 4.8 the robot is represented by the darker circle. Each circle is drawn at a constant time interval of 5 seconds.

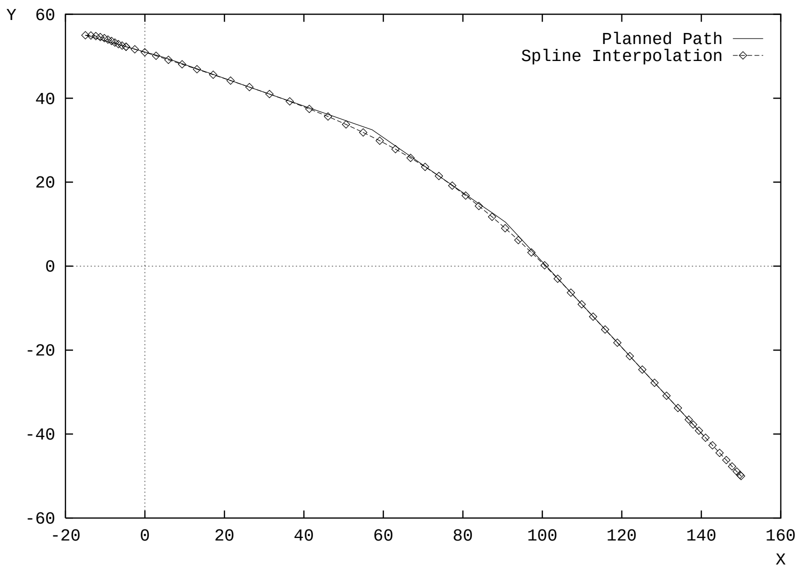

Figure 4.4: Spline interpolation of the trajectory

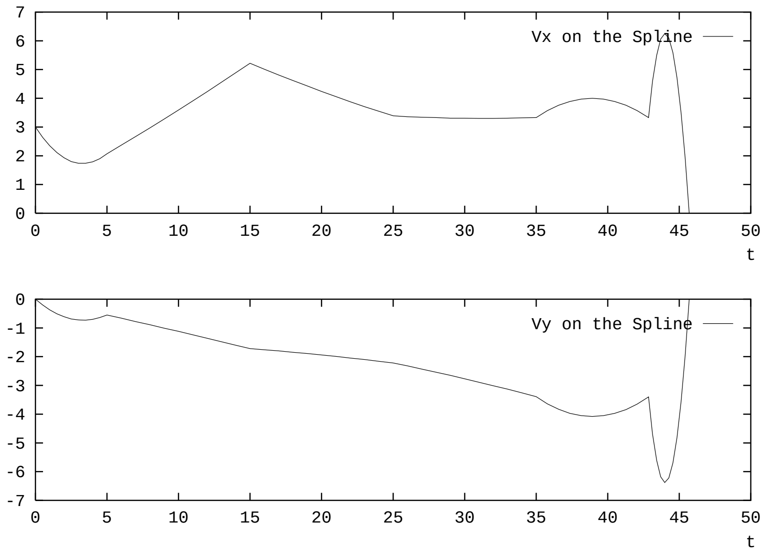

Figure 4.5: Velocities on the spline interpolation

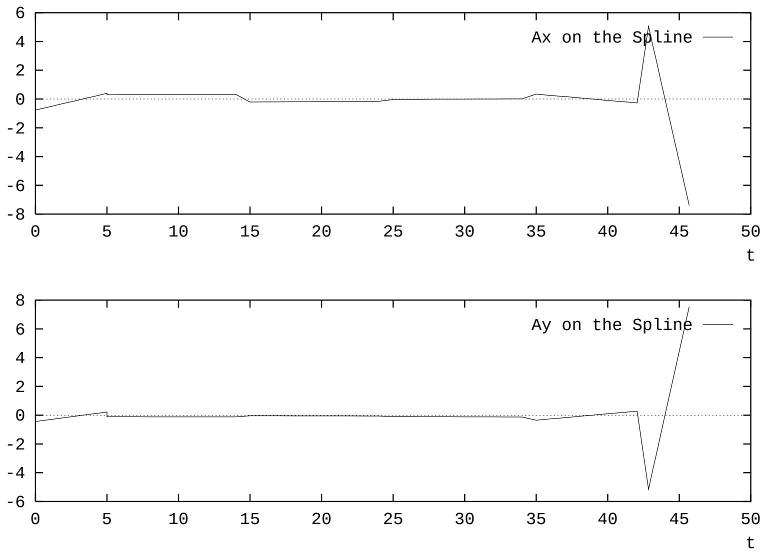

Figure 4.6: Accelerations on the spline interpolation

The nominal trajectory suffers from the effects of the approximations introduced by the spline interpolation and by the discretization of the controls. As shown in Figure 4.8, in fact, it only approximates the planned trajectory of Figure 4.1. The two major differences between the kinematic trajectory and the nominal trajectory are the terminal position, far from the target, and the collision with vehicle 1. This collision is represented in Figure 4.8 by the black marks between time $t = 25s$ and time $t = 30s$. The marks represent the depth of the penetration of the robot into vehicle 1. In spite of these differences, the spline interpolation succeeds in preserving the key features of the trajectory of Figure 4.1, namely the type of maneuver used in avoiding the obstacles, and the time of the avoidance of vehicle two, at approximately $t = 25s$. In fact, similarly to the trajectory shown Figure 4.1, vehicle 1 is avoided with a rear maneuver and vehicle 2 is avoided with a front maneuver. Thus, it can be expected that by using this initial guess, the dynamic optimization will compute a solution that satisfies the kinematic and dynamic constraints of the problem, and that will also preserve the structure of the kinematic trajectory.

The solution shown in Figure 4.9, with controls shown in Figure 4.10, does in fact satisfy the constraints and has the same structure of the kinematic trajectory. The desired structure of the solution has been achieved by using larger gains for the kinematic and dynamic constraints than for the motion time. In fact, the comparison of the controls of Figure 4.7 with those of Figure 4.10 shows that the motion time has been reduced by only a small amount.

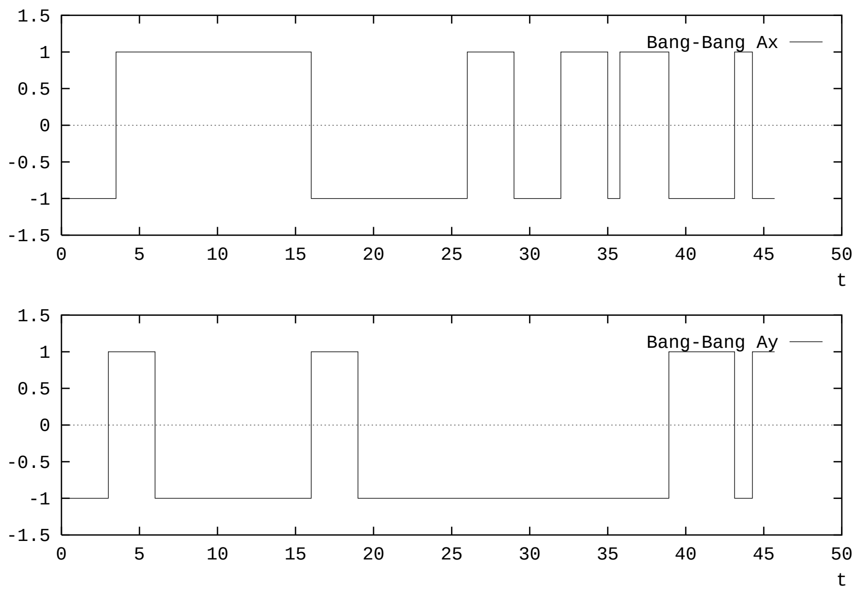

Figure 4.7: Bang-bang approximation of the accelerations

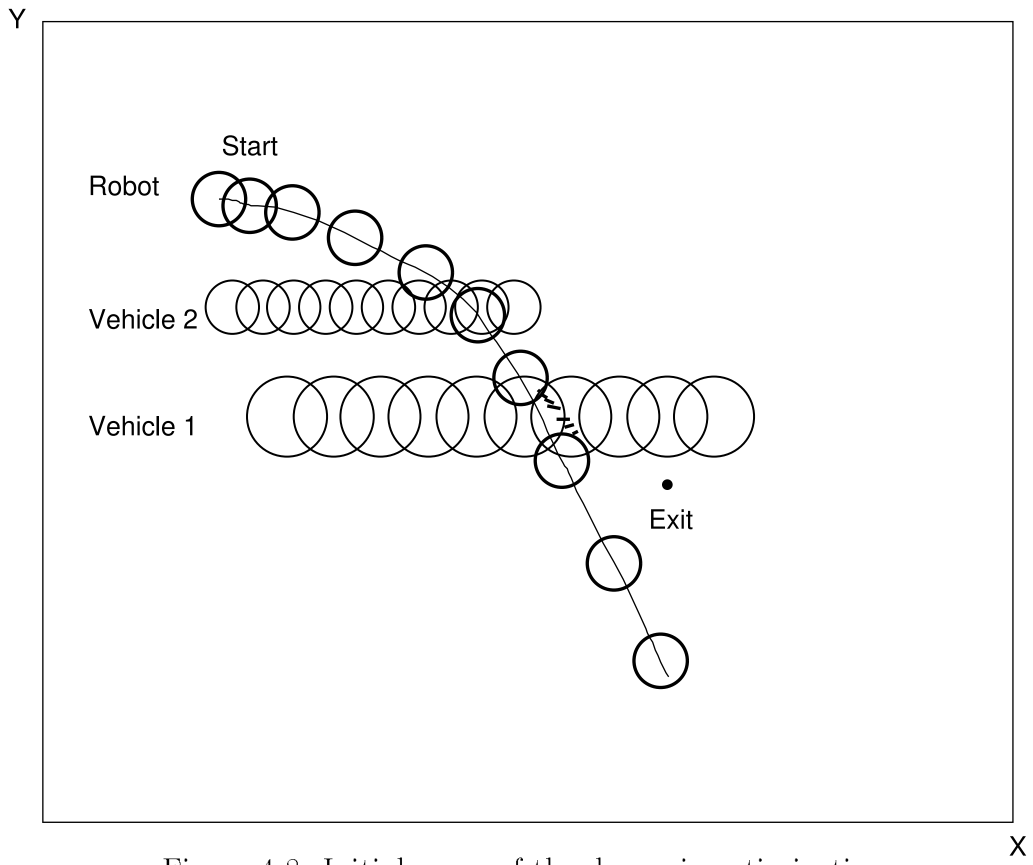

Figure 4.8: Initial guess of the dynamic optimization

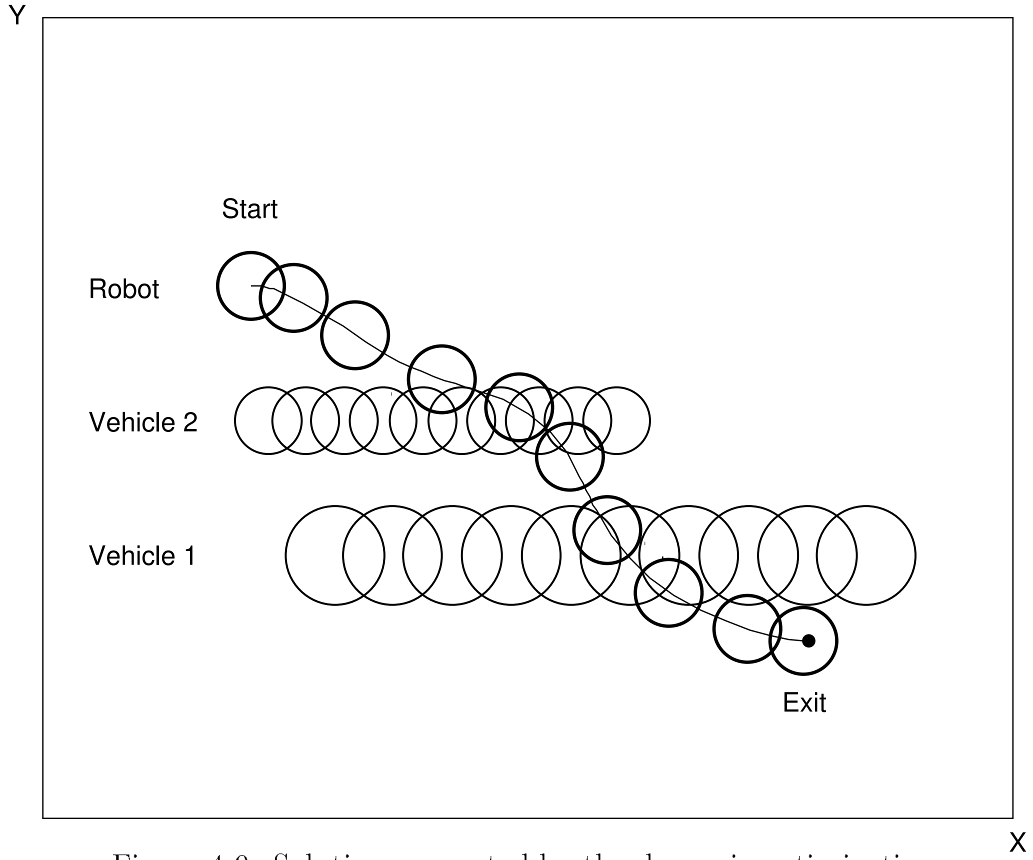

Figure 4.9: Solution computed by the dynamic optimization

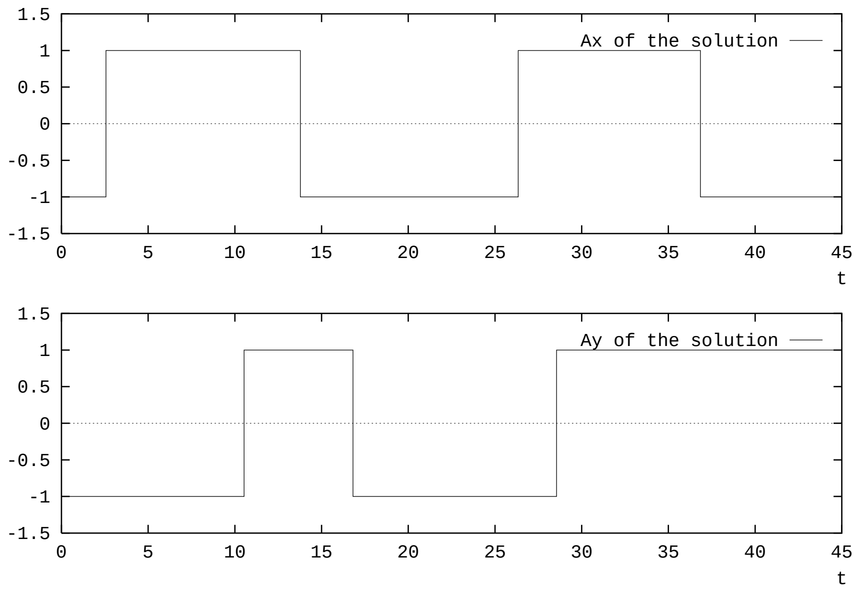

Figure 4.10: Bang-bang controls for the solution

A second example shows the effects of a different initial guess in the computation of the solution. Figure 4.11 shows another kinematic trajectory that achieves the same structure of the trajectory shown in Figure 4.1. The initial guess computed from this trajectory avoids vehicle 2 with a front maneuver and vehicle 1 with a rear maneuver, as shown in Figure 4.12. Also in this case, the initial guess misses the goal and has a contact with an obstacle, i.e. it touches vehicle 2, as shown by the marks at time $10\ s \leq t \leq 15\ s$. The optimization of this initial guess leads to the solution shown in Figure 4.13, whose controls are in Figure 4.14 and with motion time of 34.2 s. This solution is more dangerous than initially suggested by the kinematic trajectory, since the robot is touching both vehicles in several points.

It is also important to point out the differences between the controls of this trajectory, shown in Figure 4.14, and the controls for the trajectory of Figure 4.9, shown in Figure 4.10. The solution shown in Figure 4.13 is achieved with a higher number of switches than the slower solution of Figure 4.9, in spite of the fact that both solutions have the same structure. There are seven switches for A_x in Figure 4.14, and four switches for A_x in Figure 4.10. Similarly, there are five switches for A_y in Figure 4.14, and three switches for A_y in Figure 4.14. Since the numerical optimization minimizes motion time while reducing the number of switches, it follows that it is not possible to achieve the solution of Figure 4.14 starting from the slower trajectory of Figure 4.10. This example shows that different local minima of the dynamic optimization may share the same structure of avoidance maneuvers.

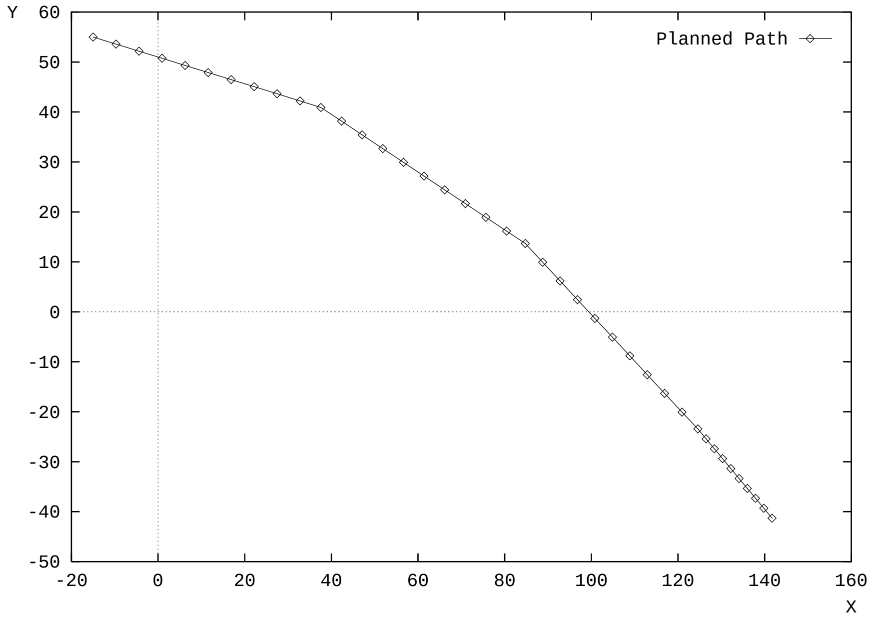

Figure 4.11: Faster heuristic trajectory

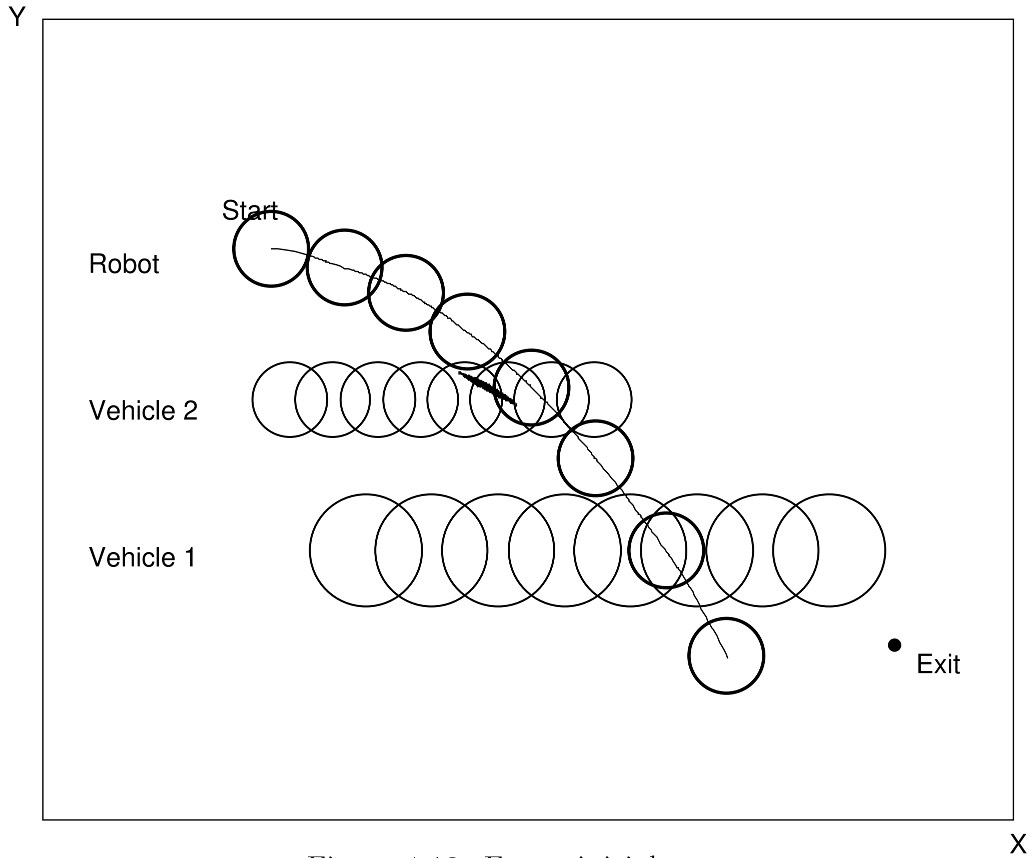

Figure 4.12: Faster initial guess

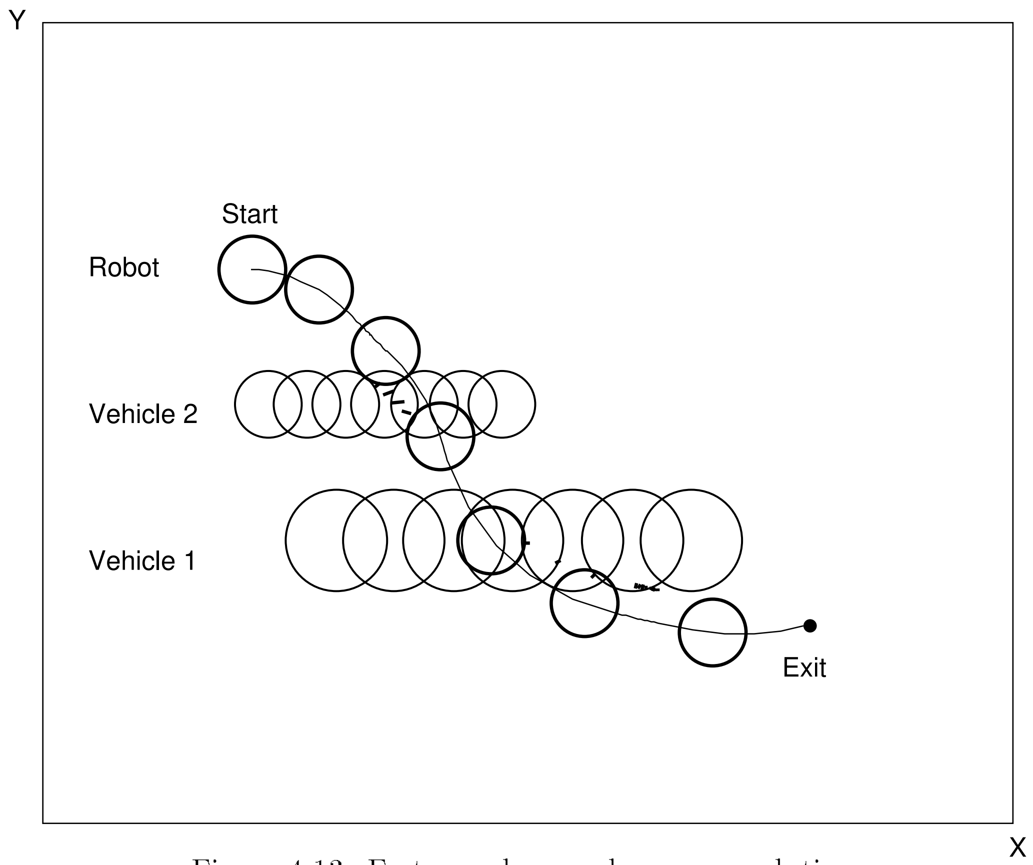

Figure 4.13: Faster and more dangerous solution

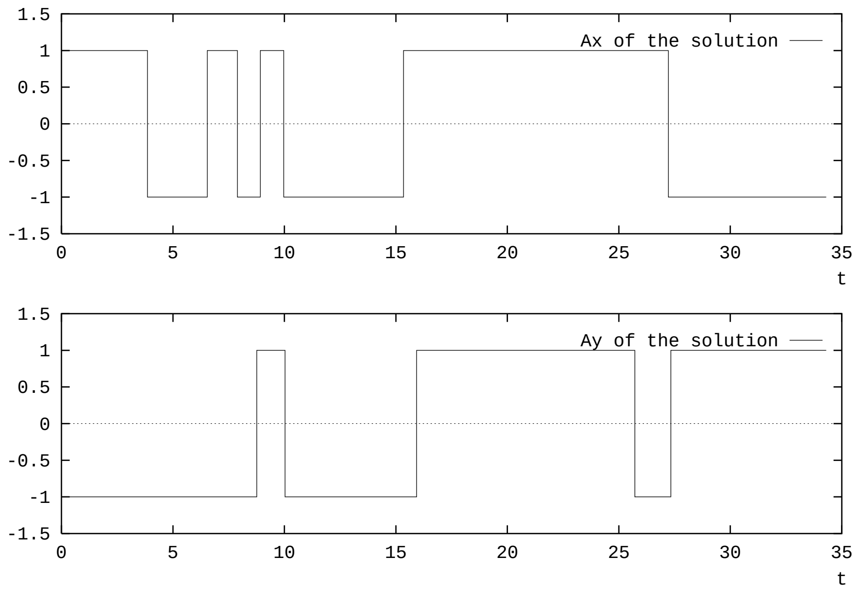

Figure 4.14: Bang-bang controls for the faster solution

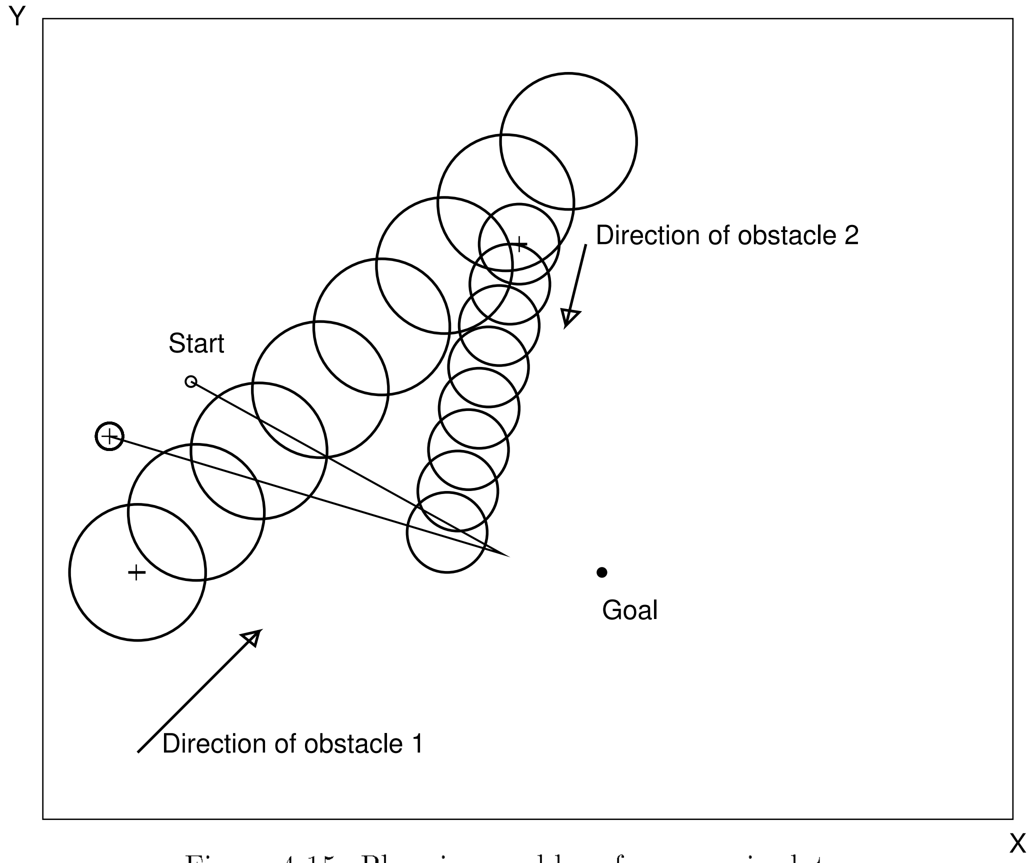

Figure 4.15: Planning problem for a manipulator.

4.4 Planning for a robotic manipulator

The initial scenario of this example is shown in Figure 4.15. The robot is represented by the model described in Appendix A, with the base marked by the sign + at the origin. The two obstacles move with constant velocity, starting from the positions marked by the sign +. Obstacle 1 starts from position ($x = .1\ m,\ -.5\ m$), with velocity ($v_x = .045\ m/s,\ v_y = .045\ m/s$), and obstacle 2 starts from position ($x = 1.15\ m,\ .7\ m$) with velocity ($v_x = -.007\ m/s,\ v_y = -.03\ m/s$). The robot starts with the hand located at ($x = .3\ m,\ .2\ m$) and zero velocity, and is required to reach the goal marked by the black dot.

The spline interpolation of trajectory planned by the velocity obstacle algorithm

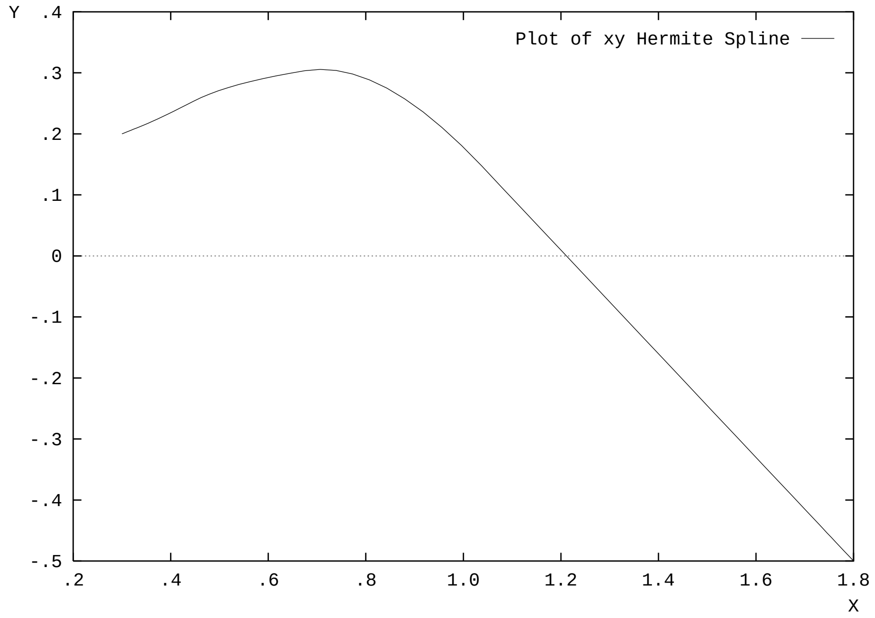

Figure 4.16: Trajectory planned by the velocity obstacle method.

is shown in Figure 4.16. The motion time of the nominal trajectory is 4.81 s. Figure 4.17 shows the bang-bang controls obtained by thresholding the torques computed from the interpolated trajectory. Figure 4.18 shows the trajectory computed by integrating these controls. As in the previous examples, also this nominal trajectory has a small collision with the obstacle 1 approximately at time $t = 1.5$ s, and the terminal position of the hand is far from the goal. This trajectory is characterized by the rear avoidance of both obstacles.

The solution computed by the dynamic optimization is shown Figure 4.19, and its controls are shown in Figure 4.20. The motion time of the solution is 2.6 s. In solving this problem, the larger gains were assigned to the time minimization, and therefore the sequence of avoidance maneuvers of the trajectory in Figure 4.18 is lost in the solution of Figure 4.19.

The spikes in the controls u_1, approximately at times .7 s, and u_2, at time .9 s, are due to the numerical algorithm, and they would disappear with more iterations of the algorithm. Similarly to the example shown in Figure 3.6, the torque shoulder u_1 has more switches than the elbow torque u_2, as shown in Figure 4.20. This fact may indicate the presence of a singular arc, since the shoulder joint must wait for the elbow to complete its longer motion.

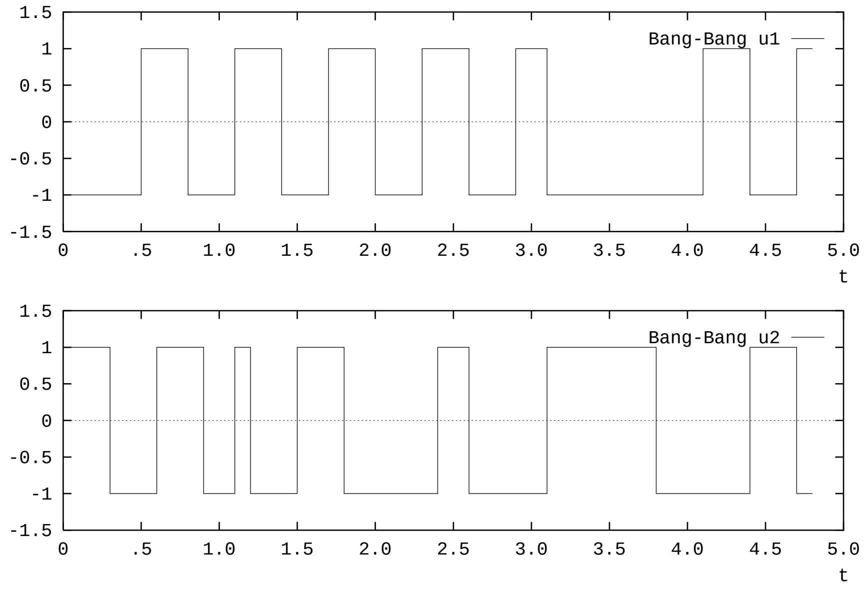

Figure 4.17: Bang-bang controls for the nominal trajectory.

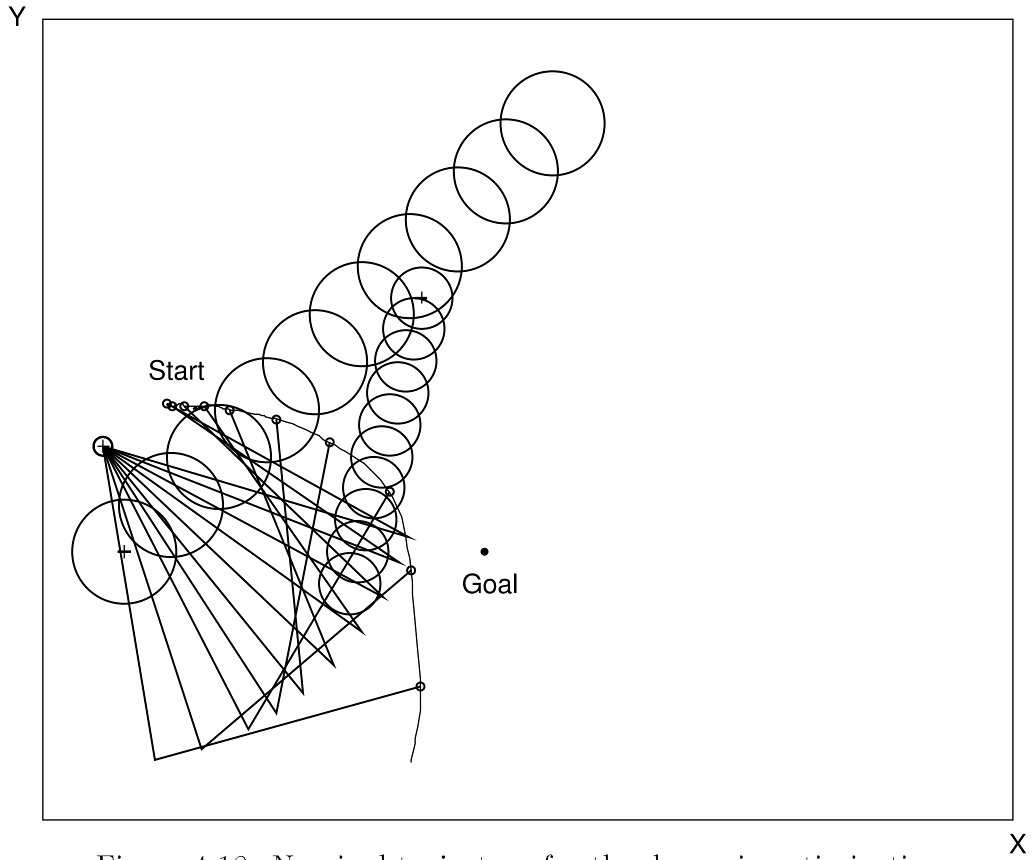

Figure 4.18: Nominal trajectory for the dynamic optimization.

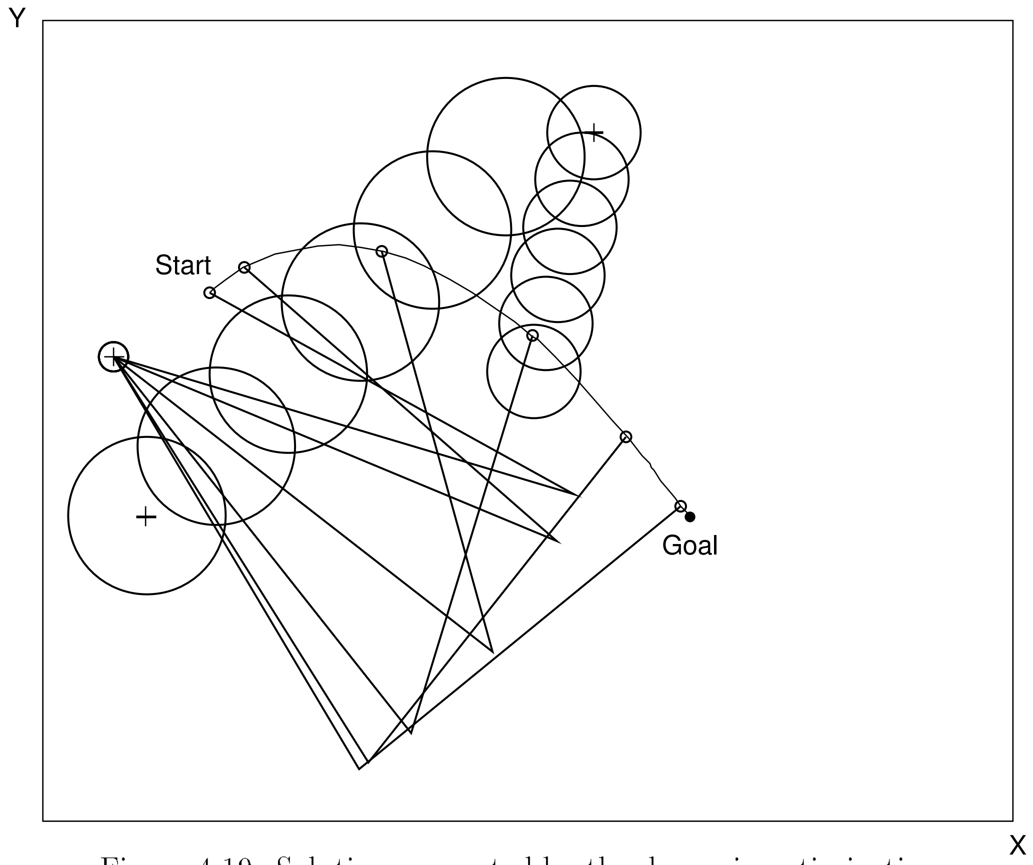

Figure 4.19: Solution computed by the dynamic optimization.

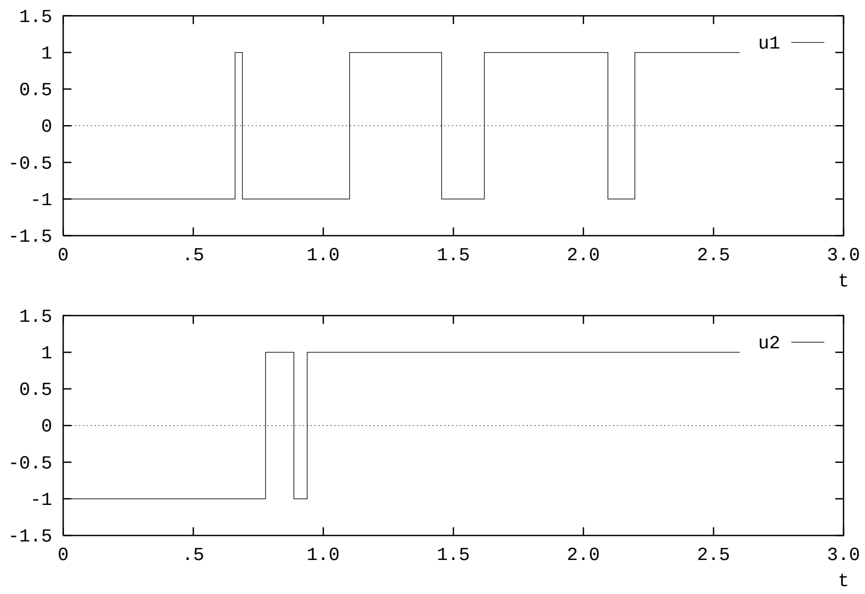

Figure 4.20: Bang-bang controls for the solution.

4.5 Summary

This chapter has presented the solution to a few problems of motion planning in a time-varying environment. The examples have been used to illustrate the complete two-step approach proposed for solving these problems. The approach consists of using the kinematic trajectory computed by the velocity obstacle method as the initial guess of the dynamic optimization. In these examples, this approach has been able to find a solution that satisfied the constraints of the problems and achieves the desired structure of the solution. The gains in the numerical optimization can be used to emphasize either the satisfaction of the constraints, or the minimization of the motion time. In the intelligent highway example, the dynamic optimization has been used mostly to satisfy the problem constraints, thus achieving only a small improvement in the motion time. In the robot arm example, instead, the emphasis of the optimization has been on minimizing the motion time, thus losing the structure of the kinematic trajectory computed by the velocity obstacle method.

Chapter 5

Summary and Recommendations for Future Research

5.1 Summary

A new method for planning the motion of a robotic system in a time-varying environment, the Velocity Obstacle approach, has been presented. It is significantly different from currently available planning algorithms, since it computes, at the same time, path and velocity that avoid all obstacles and satisfy approximate system dynamics.

Dynamic Optimization has been adapted to this specific motion planning problem, and a direct numerical method has been used to compute the time minimal trajectory that satisfies the dynamics of the robot and avoids all the obstacles in the environment.

The combination of the Velocity Obstacle method and of the Dynamic Optimization has lead to a two step algorithm for motion planning in time-varying environments. First the Velocity Obstacle method computes an approximate trajectory for the robotic system, then this trajectory is used as the initial guess of the Dynamic Optimization, compensating the previous approximations and minimizing motion time. This approach combines some of the important features of heuristics planning and of optimal planning.

5.2 Contributions

The major contributions of this thesis can be summarized as follows.

- The motion planning problem in time-varying environment has been solved with applications to intelligent highways and manipulator systems.
- A method for generating feasible trajectories in time-varying environments has been developed.
- The concept of Velocity Obstacle has been introduced, which led to the definition of the Feasible and of the Safe velocity sets.
- A method for classifying the structure of the feasible trajectories has been developed, based on the type of avoidance maneuvers.
- Global search and heuristic search have been used to compute the feasible trajectories.
- The motion planning problem in time-varying environments has been formulated as a dynamic optimization problem, subject to time and state dependent constraints.
- The numerical gradient procedure for computing the time optimal solution has been derived and implemented.
- This approach has combined heuristic planning with dynamic optimization.

5.3 Recommendations for Future Research

The following subjects are suggested for further research:

1. **Velocity Obstacle**:

- The definition of an instantaneous map of the dynamic constraints into the velocity space of the robotic system would eliminate the need for computing the Safe Velocity sets using approximate constraints.

- Other types of kinematic constraints can be included in this formalism. For example, the kinematic singularities of a robotic manipulator are time-varying obstacles that could be gracefully avoided with appropriate avoidance maneuvers.

- This method may be applicable to the design of distributed motion planning algorithms to coordinate the motion of several robotic systems in a time-varying environment.

- Similarly, the Velocity Obstacle method seems very appropriate for real time motion planning, when interfaced to suitable sensors and motion control algorithms.

- When the obstacles have a known non-linear trajectory, the approximation with straight-line segments may lead to inefficient solutions, and non-linear trajectories should be included directly in the velocity obstacle method.

2. **Dynamic Optimization**:

   - The numerical computation of optimal trajectories is an important issue. It should be made fast enough so that it could be integrated into a real time planning system. Furthermore, different numerical algorithms should be investigated to determine the most appropriate method for this specific optimization problem.

   - When the dynamics of the robot is of higher order, the conversion of the state constraints into control constraints leads to an inaccurate solution.

A numerical method should be developed to deal with higher order system dynamics.

- The structure of the solution space should be investigated. The presence of several local minima associated with a single trajectory structure suggests in fact that there is a finer classification of the trajectories than the one proposed.

3. **Complete Algorithm**:

- The computation of a bang-bang control from a given set of specifications should be addressed in detail. In particular, indirect methods, such as quantization of continuous torque profiles, and direct methods, such as direct computation of the switching times, should be investigated and compared to find the best approximation of a given control.

5.4 Conclusion

This thesis has developed three main aspects of motion planning in time-varying environment: the computation of an approximate solution using the velocity obstacle, the computation of an exact solution using the dynamic optimization, and the combination of these two methods to compute the final solution.

The velocity obstacle method is a fast and powerful tool that identifies some of the main features of an avoidance maneuver, i.e. the safe avoidance velocities, the front and rear avoidance maneuvers, and the grazing velocities. When an approximate solution is desired, this method leads to good solutions by using appropriate heuristic strategies. When an exact solution is necessary, this solution is a good initial guess for a dynamic optimization.

Dynamic optimization is well suited to solve a time-varying planning problem, and to explore the effects of the various constraints on the final solution. The selection of the gains in the numerical optimization is the tool that is used for this purpose. The careful adjustment of the gains allows the computation of different solutions that give a better understanding of the underlying solution space.

Finally, when these two methods are used one after the other, the result is to compensate the effects of each method limitations, and to achieve a solution that is better than the solution computed by each single method. The lack of accurate dynamic modeling in the velocity obstacle method is compensated by the dynamic optimization. Similarly, the lack of control on the shape of the optimal solution is removed by using the initial guess provided by the velocity obstacle method. Although still in its infancy, this combination of fast heuristics and exact optimization has demonstrated its power in solving complex motion planning problems.

Appendix A

Kinematic and Dynamic Equations of a Two-Link Manipulator

Figure A.1 shows the model of the two link manipulator used in the examples of Chapters 3 and 4. The symbols used in the model are the following:

O the shoulder of the arm,

E the elbow of the arm,

H the tip of the forearm,

C_1 center of mass of link one,

C_2 center of mass of link two,

l_1 length of link one,

l_2 length of link two,

l_{C_1} distance of $\mathbf{C}_1$ from **O**,

l_{C_2} distance of $\mathbf{C}_2$ from **A**,

m_1 mass of link one,

m_2 mass of link two,

θ_1 angle between link one and the **X** axis,

θ_2 angle between link two and link one,

I principal central moment of inertia of each link, $I = \frac{m\ l_i^2}{12}$

$\tau_{1,2}$ torques applied at joints 1 and 2.

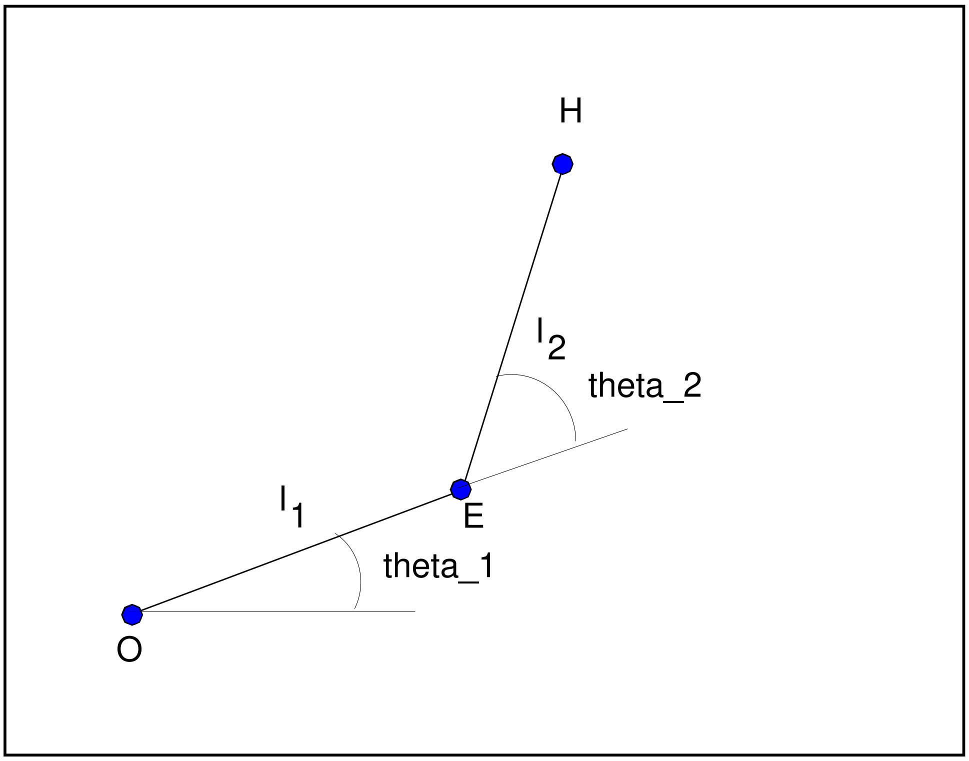

Figure A.1: Planar 2-dof Manipulator

The kinematic equations of the arm are [78]:

- Direct kinematics:

$$x_h = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \tag{A.1}$$

$$y_h = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \tag{A.2}$$

- Inverse kinematics:

$$\theta_1 = \arctan 2(y_h, x_h) \mp \arctan 2(k, x_h^2 + y_h^2 + l_1^2 - l_2^2) \tag{A.3}$$

$$\theta_2 = \pm \arctan 2(k, x_h^2 + y_h^2 - l_1^2 - l_2^2) \tag{A.4}$$

$$k = \sqrt{(x_h^2 + y_h^2 + l_1^2 + l_2^2)^2 - 2\left[(x_h^2 + y_h^2)^2 + l_1^4 + l_2^4\right]} \tag{A.5}$$

$$\tag{A.6}$$

with the condition

$$(l_1^2 - l_2^2) \leq (x_h^2 + y_h^2) \leq (l_1^2 + l_2^2)$$

In the symbols ($\mp$, $\pm$), the top sign refers to the *elbow up* configuration of the arm, and the bottom sign refers to the *elbow down* configuration.

- Differential kinematics:

$$J = \begin{bmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{A.7}$$

$$J^{-1} = \begin{bmatrix} \frac{\cos(\theta_1+\theta_2)}{l_1 \sin\theta_2} & \frac{\sin(\theta_1+\theta_2)}{l_1 \sin\theta_2} \\ \frac{l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2)}{l_1 l_2 \sin\theta_2} & -\frac{l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2)}{l_1 l_2 \sin\theta_2} \end{bmatrix} \tag{A.8}$$

The dynamic equations of the arm are [2]:

- State equations

$$x_1 = \theta_1 \tag{A.9}$$

$$x_2 = \dot{\theta}_1 \tag{A.10}$$

$$x_3 = \theta_2 \tag{A.11}$$

$$x_4 = \dot{\theta}_2 \tag{A.12}$$

- Dynamic equations

$$
\begin{aligned}
\dot{x_1} &= x_2 && \text{(A.13)}\\
\dot{x_2} &= (\tau_1 - H_{12}\dot{x_4} + h\,x_4^2 + 2\,h\;x_2 x_4)/H_{11} && \text{(A.14)}\\
\dot{x_3} &= x_4 && \text{(A.15)}\\
\dot{x_4} &= (\tau_2 - H_{12}\dot{x_2} - h\,x_2^2)/H_{22} && \text{(A.16)}
\end{aligned}
$$

with:

$$
\begin{aligned}
H_{11} &= m_1 l_{c_1}^2 + I_1 + m_2(l_1^2 + l_{c_2}^2 + 2\,l_1 l_{c_2}\cos\theta_2) + I_2 && \text{(A.17)}\\
H_{22} &= m_2 l_{c_2}^2 + I_2 && \text{(A.18)}\\
H_{12} &= m_2 l_1 l_{c_2}\cos\theta_2 + m_2 l_{c_2}^2 + I_2 && \text{(A.19)}\\
h &= m_2 l_1 l_{c_2} sin\theta_2 && \text{(A.20)}
\end{aligned}
$$

From these equations it follows that:

$$
\begin{aligned}
\dot{x_2} &= (\tau_1 - \frac{H_{12}}{H_{22}}\tau_2 + \frac{h\,H_{12}}{H_{22}}x_2^2 + h\,x_4^2 + 2\,h\,x_2 x_4)\frac{H_{22}}{H_{11}H_{22} - H_{12}^2} && \text{(A.21)}\\
\dot{x_4} &= (\tau_2 - H_{12}\dot{x_2} - h\,x_2^2)/H_{22} && \text{(A.22)}
\end{aligned}
$$

The values of the parameters used in the examples are the following:

- $l_1 = 1.5$ m, $l_2 = 1.3$ m.
- $m_1 = 10.0$ Kg, $m_2 = 10.0$ Kg.
- $\tau_1 = 10.0$ Nm, $\tau_2 = 3.0$ Nm.

Appendix B
Computation of the Initial Bang-Bang Controls

The dynamic optimization described in Chapter 3 assumes the existence of an initial guess of the trajectory given in terms of the controls of the robot joints. Thus, the need for developing a method for generating a trajectory for the robot and computing the corresponding joint torques. Furthermore, the torques should be bang-bang, as discussed in Section 3.3.4.

A few methods are available for computing the bang-bang torques of a manipulator, given its dynamic model and the end points of the trajectory [77], [16]. These methods have been developed for computing a final trajectory without a successive optimization step. In this case, instead, the trajectory is only the initial guess for the dynamic optimization, and therefore the computation of the joint torques need not be very accurate.

The method used is quite simple, with a cubic spline representing the path between the two end points, and a cycloid interpolating the velocity on the path. This method is developed for the case of given end points with assigned initial and terminal velocities.

When position and velocity of the end points are given, an analytical expression of the trajectory can be obtained with a third order Hermite spline. This spline represents the path of the robot that takes into account the desired initial and final direction of motion. A third-order polynomial becomes a Hermite spline when its coefficients are computed using the position of the terminal points, and the desired slope of the curve at the terminal points. This spline is described by the following equations in the parameter s:

$$\begin{aligned} x(s) &= a_3\, s^3 + a_2\, s^2 + a_1\, s + a_0 \\ y(t) &= b_3\, s^3 + b_2\, s^2 + b_1\, s + b_0 \end{aligned} \tag{B.1}$$

The coefficients a_i and b_i are computed by satisfying the following conditions:

$$\text{starting point} \quad \begin{cases} x(s_0) = x_0 \\ y(s_0) = y_0 \end{cases} \tag{B.2}$$

$$\text{starting velocity} \quad \begin{cases} v_x(s_0) = v_{x,0} \\ v_y(s_0) = v_{y,0} \end{cases} \tag{B.3}$$

$$\text{ending point} \quad \begin{cases} x(s_f) = x_f \\ y(s_f) = y_f \end{cases} \tag{B.4}$$

$$\text{ending velocity} \quad \begin{cases} v_x(s_f) = v_{x,f} \\ v_y(s_f) = v_{y,f} \end{cases} \tag{B.5}$$

(B.6)

The Hermite spline can be written in matrix form [25]:

$$\mathbf{X} = \mathbf{A} * \mathbf{M} \tag{B.7}$$

$$\mathbf{Y} = \mathbf{B} * \mathbf{M} \tag{B.8}$$

where:

$$\mathbf{X} = \begin{pmatrix} x(s_0) \\ x(s_f) \\ v_x(s_0) \\ v_x(s_f) \end{pmatrix} \qquad \mathbf{Y} = \begin{pmatrix} y(s_0) \\ y(s_f) \\ v_y(s_0) \\ v_y(s_f) \end{pmatrix}$$

$$\mathbf{A} = (a_j) \ \ \text{with } j = 0, \ldots, 3 \qquad \mathbf{B} = (b_j) \ \ \text{with } j = 0, \ldots, 3$$

$$\mathbf{M} = \begin{pmatrix} m_{0,k} = s_0^k \ , \ k = 0, \ldots, 3 \\ m_{f,k} = s_f^k \ , \ k = 0, \ldots, 3 \\ m_{0,k} = k\, s_j^{k-1} \ , \ k = 3, \ldots, 0 \\ m_{f,k} = k\, s_j^{k-1} \ , \ k = 3, \ldots, 0 \end{pmatrix}$$

Figure B.1 shows the spline connecting two points $\mathbf{S}: \ \mathbf{x} = (-.15\ m,\ .55\ m)$ and $\mathbf{G}: \ \mathbf{x} = (1.5\ m, -.5\ m)$, and with slopes $v_x = 3.0\ m/s,\ v_y = 0.0\ m/s$ at both end points.

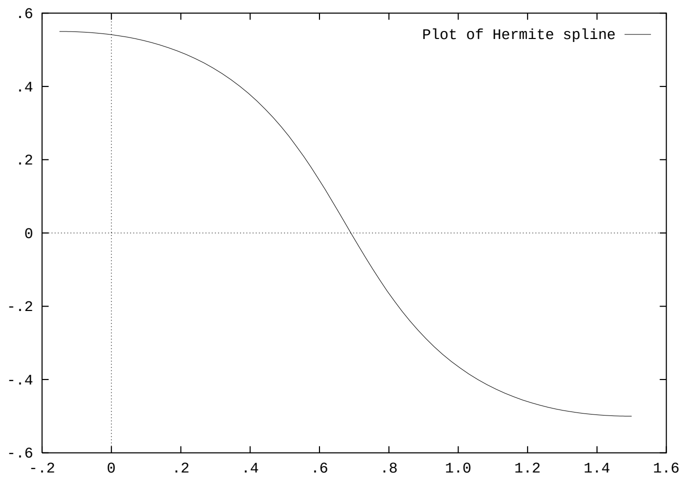

Figure B.1: Hermite spline with non-zero terminal velocities

The Hermite spline represents only the desired path of the robot. The actual velocity profile on such a path can be computed using a cycloidal interpolation for the parameter s of the spline, thus ensuring that the end points of the trajectory have continuous velocity and zero acceleration. The equations of the cycloid, written using t as a parameter, are:

$$\omega = \frac{2.0\pi}{T} \tag{B.9}$$

$$s = \frac{\omega t - \sin(\omega t)}{2.0\pi} \tag{B.10}$$

$$\dot{s} = \frac{\omega(1 - \cos(\omega t))}{2.0\pi} \tag{B.11}$$

$$\ddot{s} = \frac{\sin(\omega t)\omega^2}{2.0\pi} \tag{B.12}$$

where s, $\dot{s}$ and $\ddot{s}$ are respectively the value of the parameter of the Hermite spline, its first time derivative, its second time derivative, and T is the initial guess of the motion time. The plot of the three interpolants is shown if Figures B.2.

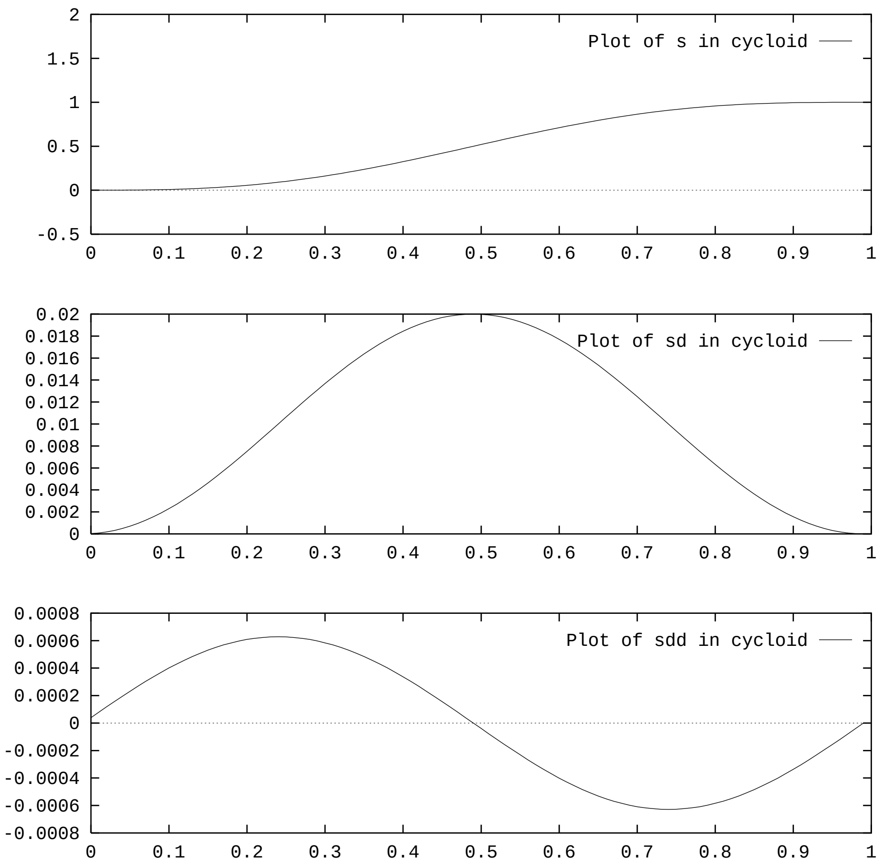

Figure B.2: Plot of the cycloids for position, velocity and acceleration

Figure B.3 shows the Hermite spline connecting two points

$$\mathbf{S}: \ \mathbf{x} = (-.15\ m,\ .55\ m),\ \mathbf{v} = (0,0) \text{ and } \mathbf{G}: \ \mathbf{x} = (1.5\ m, -.5\ m),\ \mathbf{v} = (0,0)$$

. Since the end points have zero terminal vectors, the spline is simply the segment connecting the two points. The marks on the spline are drawn at a fixed interval of 0.4 s and show the effect of the cycloidal interpolation on the velocity.

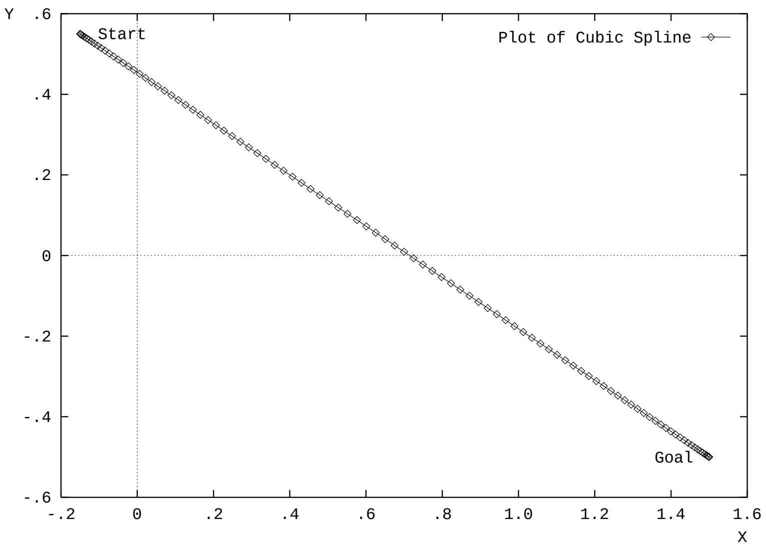

Figure B.3: Hermite spline with zero terminal velocities

The Cartesian position $\mathbf{x}$, velocity $\mathbf{v}$ and acceleration $\mathbf{a}$ follow immediately from equations (B.1). The angular acceleration of each joint are computed using the kinematic equations of the robot in Appendix A, and the corresponding torques τ_i are computed using the inverse dynamics of Appendix A. For example, Figure B.4 shows the joint torques computed from the trajectory of Figure B.1, and Figure B.5 shows the joint torques required for the trajectory of Figure B.3.

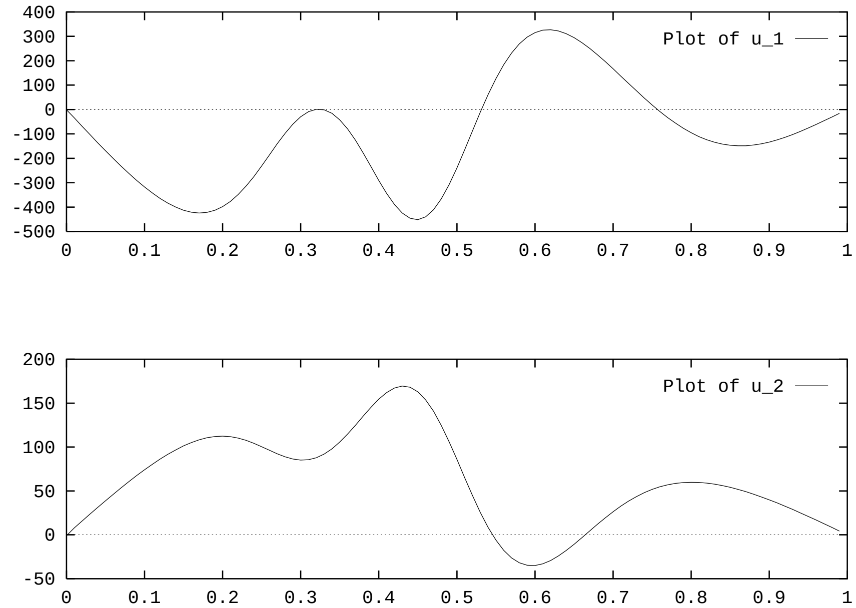

Figure B.4: Joint torques for the spline of Figure B.1.

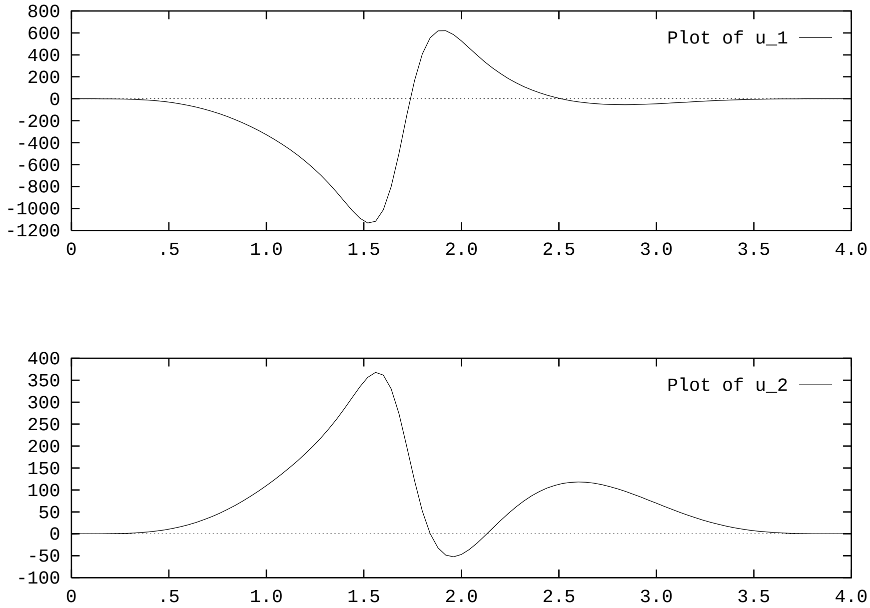

Figure B.5: Joint torques for the spline of Figure B.3.

These controls can be converted to bang-bang form by fixing a threshold for the torques. Here the torque limits are $\tau_1 = \pm 10\ Nm$ and $\tau_2 = \pm 3\ Nm$, and the threshold has been fixed at $.2\,\tau_i,\quad i = 1, 2$. The segments in which the torques are almost zero are approximated with a number of equally spaced switches. In this case, the choice of the distance between two switches is arbitrary, as long as the total number of switches is enough to allow for precise adjustments of the trajectory by the dynamic optimization. The initial number of switches does not affect the final solution of the optimization, since redundant switches are eventually eliminated. The bang-bang torques approximating the joint torques of Figure B.5 are shown in Figure B.6.

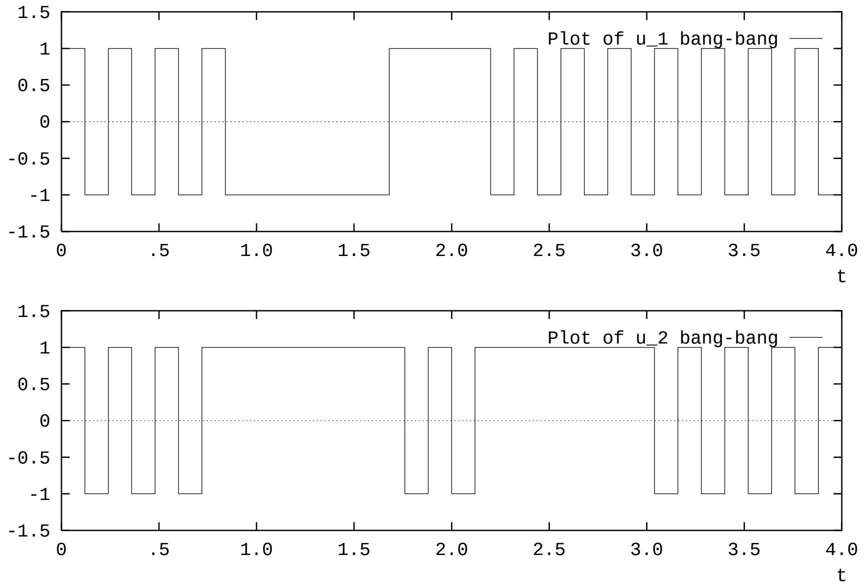

Figure B.6: Bang-bang approximation of the torques in Figure B.5.

BIBLIOGRAPHY

[1] G.M. Anderson. Comparison of optimal control and differential game intercept missile guidance law. *AIAA Journal of Guidance and Control*, 4(2):109–115, March-April 1981.

[2] H Asada and J.-J. Slotine. *Robot Analysis and Control.* John Wiley and Sons, New York, 1986.

[3] A. Blanquière, F. Gérard, and G. Leitmann. *Quantitative and Qualitative Games.* Academic Press, New York, NY, 1969.

[4] J.E. Bobrow. A direct minimization approach for obtaining the distance between convex polyhedra. *The International Journal of Robotics Research*, 8(3):65,76, June 1989.

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

[6] J.V. Breakwell. Some simple two-target games. *Journal of Dynamic Systems, Measurement and Control*, 111:589–591, December 1989.

[7] R. Brooks. Planning collision free motions for pick and place operations. *The International Journal of Robotic Research*, 2(4):19–44, Winter 1983.

[8] A.E. Bryson and S.E. Denham, W.F.and Dreyfus. Optimal programming problems with inequality constraints i: Necessary conditions for extremal solutions. *AIAA Journal*, 1(11):2544–2550, November 1963.

[9] A.E. Bryson and W.F. Denham. A steepest-ascent method for solving optimum programming problems. *ASME Journal of Applied Mechanics*, (29):247–257, June 1962.

[10] A.E. Bryson and Y.C. Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., New York, NY, 1975.

[11] W. Burke. *Applied Differential Geometry*. Cambridge University Press, 1985.

[12] S.A. Cameron. *Modelling Solids in Motion*. PhD thesis, Edinburgh, UK, 1984.

[13] J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Boston, MA, 1988.

[14] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *28th IEEE Symposium on Foundation of Computer Science*, 1987.

[15] J. Canny, J. Reif, B. Donald, and P. Xavier. On the complexity of kinodynamic planning. In *29th IEEEE Symposium on Foundation of Computer Science*, 1988.

[16] Y. Chen, M. Hsieh, A. Inselberg, and H. Lee. Constrained planar conflict resolution for air traffic control. In *ACM Conference on Computational Geometry*, 1990.

[17] J.H. Davenport. A piano mover problem. *Sigsam Bullettin*, 20(1-2):15–17, 1986.

[18] A. Davidovitz and J. Shinar. Eccentric two-target model for qualitative air combat game analysis. *Journal of Guidance*, 8(3):325–331, May-June 1985.

[19] A.E. Denham, W.F.and Bryson. Optimal programming problems with inequality constraints *ii*: Solution by steepest ascent. *AIAA Journal*, 2(2):25–34, January 1964.

[20] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE International Conference on Robotics and Automation*, pages 1419–1424, San Francisco, CA, April 7-10 1986.

[21] M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, (2):477–521, 1987.

[22] R. Featherstone. Swept bubbles: A method of representing swept volume and space occupancy. Technical Report TR-90-024 and MS-90-069, Philips Laboratories, North American Philips Corporation, Briarcliff Manor, New York 10510, August 21 1990.

[23] P. Fiorini and J. Chang. Autonomous organization of grasping tasks. In *SPIE Symposia on Aerospace Sensing, Artificial Intelligence VII*, Orlando, FL, March 27- 31 1989.

[24] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *IEEE International Conference of Automation and Robotics*, volume 1, pages 560–566, May 1993.

[25] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics*. Addison-Wesley Publishing Company, Reading, MA, 1990.

[26] K. Fujimura. On motion planning amidst transient obstacles. In *IEEE International Conference on Robotics and Automation*, pages 1488–1493, Nice, France, May 1992.

[27] K. Fujimura and H. Samet. Time-minimal paths among moving obstacles. In *IEEE International Conference on Robotics and Automation*, pages 1110–1115, Scottsdale, AZ, May 1989.

[28] K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *IEEE International Conference on Robotics and Automation*, pages 324–330, Cincinnati, OH, May 1990.

[29] W.M. Getz and G. Leitmann. Qualitative differential games with two targets. *Journal of Mathematical Analysis and Applications*, 68:421–430, 1979.

[30] D. Ghose. True proportional navigation with maneuvering target. *IEEE Transactions on Aerospace and Electronic Systems*, 30(1):229–237, January 1994.

[31] D. Ghose and U.R. Prasad. Qualitative analysis of secured outcome regions for two-target games. *Journal of Mathematical Analysis and Applications*, 68(2):233–255, February 1991.

[32] K.Y. Hwang and N. Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24(3):219–291, September 1992.

[33] A. Inselberg, B. Dimsdale, and M. Boz. Conflict resolution intervals for: Early conflict detection, resolution and one shot problem. In *First CCCG, University of Montreal*, Montreal, Canada, August 1989.

[34] R. Isaac. *Differential Games.* John Whiley and Sons, New York, NY, 1965.

[35] D.H. Jacobson, M.M. Lele, and J.L. Speyer. New necessary conditions of optimality for control problems with state-variable inequality constraints. *Journal of Mathematical Analysis and Applications*, 35(2):255–284, 1971.

[36] D.W. Johnson and E.G. Gilbert. Minimum time robot planning in the presence of obstacles. In *IEEE Conference on Decision and Control*, pages 1748–1753, Ft. Lauderdale, FL, December 1985.

[37] M.E. Kahn. *The Near-Minimum Time Control of Open Loop Articulated Kinematic Chains*. PhD thesis, Stanford, 1969.

[38] K. Kant and S.W. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotic Research*, 5(3):72–89, Fall 1986.

[39] K. Kant and S.W. Zucker. Planning collision free trajectories in time-varying environments: a two level hierarchy. In *IEEE International Conference on Robotics and Automation*, pages 1644–1649, Raleigh, NC, 1988.

[40] Richard Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, (27):97–109, 1985.

[41] J.C. Latombe, editor. *Robot Motion Planning*. Kluwer Academic Publisher, Boston, MA, 1991.

[42] B.H. Lee and C.S.G. Lee. Collision-free motion planning of two robots. *IEEE Transactions on System Man and Cybernetics*, SMC-17(1):21–32, January-February 1987.

[43] G. Leitman. *An Introduction to Optimal Control*. McGraw-Hill Book Co., New York, NY, 1966.

[44] T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Transaction on Computers*, C-32(2):108–120, February 1983.

[45] T. Lozano-Pérez. A simple motion-planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation*, RA-3(3):224–238, June 1987.

[46] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[47] J.H.S. Luh and M.W. Walker. Minimum-time along the path for a mechanical arm. In *IEEE Conference on Decision and Control*, pages 755–763, New Orleans, Luisiana, December 1977.

[48] E.B. Meier. *An Efficient Algorithm for Bang-Bang Control Systems Applied to a Two-Link Manipulator*. PhD thesis, Stanford, December 1987.

[49] N Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.

[50] Ó'Dúnlaing. Motion planning with inertial constraints. Technical Report TR-230, NYU Robotics Laboratory, 1986.

[51] J. O'Rourke and N. Badler. Decomposition of three-dimensional objects into spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):295–305, July 1979.

[52] J. Pearl. *Heuristics*. Addison Wesley Publishing, Co., Reading, MA, 1985.

[53] F. Preparata and M.I. Shamos. *Computational Geometry*. Springer-Verlag, New York, NY, 1985.

[54] M. Prinz. *Optimal Control of Robot Manipulators*. PhD thesis, Stanford, June 1986.

[55] V.T. Rajan. Minimum time trajectory planning. In *IEEE Conference on Robotics and Automation*, pages 759–764, St. Louis, Missouri, March 1985.

[56] I. Rhee and J. Speyer. Robust momentum management and attitude control system for the space station. *Journal of Guidance, Control and Dynamics*, 15(2):342–351, March-April 1992.

[57] G. Sahar and J.M. Hollerbach. Planning of minimum-time trajectories for robot arms. In *IEEE Conference on Robotics and Automation*, St. Louis, Missouri, March 1985.

[58] J.C. Sanborn and J.A. Hendler. A model of reaction for planning in dynamic environments. *International Journal of Artificial Intelligence in Engineering*, 3(2):95–101, April 1988.

[59] V. Scheinman and B. Roth. On the optimal selection and placement of manipulators. In *Fifth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 531–541, Udine, Italy, June 1984.

[60] J. Schwartz, M. Sharir, and J. Hopcroft, editors. *Planning, Geometry and Complexity of Robot Motion.* Ablex Publishing Corp., Norwood, New Jersey, 1987.

[61] J. Schwartz and C. K. Yap, editors. *Algorithmic Aspects of Robotics.* Lawrence Erlbaum Associted, 1987.

[62] Z. Shiller. Time-energy optimal control of articulated systems with geometric path constraints. *ASME Journal of Dynamic Systems, Measurements and Control*, June 1994.

[63] Z. Shiller and S. Dubowsky. Robot path planner with obstacles, actuators, gripper and payload constraints. *The International Journal of Robotics Research*, 8(6):3–18, December 1989.

[64] Z. Shiller and S. Dubowsky. On computing the global time optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions of Robotics and Automation*, 7(6):785–797, December 1991.

[65] Z. Shiller and H.H. Lu. Robust computation of path constrained time optimal motions. In *IEEE Conference on Robotics and Automation*, pages 144–149, Cincinnati, OH, May 1990.

[66] Z. Shiller and H.H. Lu. Computation of path constrained time optimal motions with dynamic sungularities. *ASME Journal of Dynamic Systems, Measurements and Control*, 114:34–40, March 1992.

[67] K.G. Shin and N.D. McKay. Open loop minimum time control of mechanical manipulators. In *American Control Conference*, pages 1231–1236, June 1984.

[68] J.M. Skowronski and R.J. Stonier. The barrier in a pursuit-evasion game with two targets. *Computational Mathematical Applications*, 13(1-3):37–45, 1987.

[69] J. Speyer. *Optimization and Control of Nonlinar Systems with Inflight Constraints*. PhD thesis, Harvard, February 1968.

[70] J. Speyer. Dynamic systems stochastic estimation and control. Class Notes Ucla MANE 271B, February 1991.

[71] J. Speyer, K. Kim, and M. Tahk. Passive homing missile guidance law based on new taregt maneuver models. *Journal of Guidance*, 13(5):803–812, September-October 1990.

[72] S. Sundar and Shiller Z. Optimal obstacle avoidance based on sufficient conditions of optimality. In *IEEE Conference on Robotics and Automation*, San Diego, CA, May 1994.

[73] H. Tominaga and B. Bavarian. Global robot path planning using exact variational methods. In *IEEE Conference on Systems, Man and Cybernetics*, pages 617–619, Los Angles, CA, September 1990.

[74] H. Tominaga and B. Bavarian. Solving the moving obstacle path planning problem using embedded variational methods. In *IEEE Conference on Robotics and Automation*, pages 450–455, Sacramento, CA, April 1991.

[75] D. Wang and Y. Hamam. Optimal trajectory planning of manipulators with collision detection and avoidance. *The International Journal of Robotic Research*, 11(5):460–468, October 1992.

[76] A. Weinreb and A.E. Bryson. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, AC-30(d116), November 1985.

[77] J. Wen and A.A. Desrochers. An algorithm for obtaining bang-bang control laws. *Journal of Dynamic Systems, Measurement, and Control*, 109:171–175, June 1987.

[78] Tsuneo Yoshikawa. *Foundation of Robotics*. The MIT Press, Cambridge, MA, 1990.

[79] P. Zarchan. *Tactical and Strategic Missile Guidance*. American Institute of Aeronautics and Astronautics, Washington, DC, 1990.