UNIVERSITY OF ALBERTA

Vision-Based Navigation: Path Planning and Motion Detection in Dynamic Environments

BY ©

Ashraf Elnagar

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy

DEPARTMENT OF COMPUTING SCIENCE

Edmonton, Alberta
Fall 1993

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

ISBN 0-315-88333-2

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Ashraf Elnagar**

TITLE OF THESIS: **Vision-Based Navigation: Path Planning and Motion Detection in Dynamic Environments**

DEGREE: **Doctor of Philosophy**

YEAR THIS DEGREE GRANTED: **1993**

(Signed) . . A. Elnagar . . . . .

Ashraf Elnagar
211 R.H. Michener Park,
Edmonton, Alberta,
Canada T6H 4M5

Date: Oct. 4th, 1993

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Vision-Based Navigation: Path Planning and Motion Detection in Dynamic Environments** submitted by **Ashraf Elnagar** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

. . . .
Dr. A. Basu (Supervisor)

T.A. Marsland
. . . .
Dr. T.A. Marsland (Examiner)

. . . .
Dr. H. Zhang (Examiner)

. . . .
Dr. B. Cockburn (Examiner)

. . . .
Dr. W.A. Gruver (External)

Dr. W. Dobosiewicz (Chair)

Date: Oct. 1st, 1993.

To my parents

Abstract

Visual navigation is a major goal in both the robotics and computer vision communities. One specific example of this goal is the design of a vision-based navigation system. To achieve this task, there are two basic components that should be considered. The first one is sensors which provide information about the environment (obstacle and motion detection) whereas the other component is a control strategy that decides about the next planning step (motion planning and collision avoidance) based on the available information.

In motion planning, the problem of planning a safe (collision-free) path for a mobile platform, in the presence of static and moving obstacles, based on local information is to be solved. First, we study the problem in static domains. We place constraints on the velocity and the acceleration of the robot. The concept of *safety optimization* is proposed and used in conjunction with two heuristics to solve the problem. Moreover, we propose a new technique for generating not only safe but also smooth piecewise sub-trajectories that minimize acceleration amidst circular obstacles in local environments. Based on the above, we extend our investigation of the problem to dynamic domains. We take into account uncertainties in estimating the velocity of each

obstacle as well as the mobile robot.

As for motion detection, we address the problem of detecting moving objects from a moving camera using *background constraints*. Background compensation is used to correlate pixels in different images. The use of morphological filtering of motion images is explored to de-sensitize the detection algorithm to inaccuracies in background compensation.

Theoretical analysis of all of the above techniques are described and implemented. Experimental results, which demonstrate the validity of these techniques, are also presented.

Acknowledgements

I would like to express my sincere thanks to my supervisor, Anup Basu, for his guidance and assistance throughout the entire course of this work. I would also like to thank my examining committee members, Dr. H. Zhang, Dr. T.A. Marsland, Dr. B. Cockburn, and Dr. W.A Gruver. Their suggestions and comments were valuable in improving the quality of this thesis.

I would like also to express my indebtedness to my family who provided not only the inspiration, but also the moral, emotional support and relentless motivation without which this dream would never have come to fruition.

The joyful moments which I spent away from school with my son, Mohammed, are gratefully acknowledged.

And most of all, my warm thanks to my wife, Hala, whose love, patience, and support kept me going through all of it.

Finally, I would like to thank the Department of Computing Science, Faculty of Graduate Studies and Research, Province of Alberta, Natural Engineering Research Council (NSERC), and Northern Underwater Systems Ltd. for their financial assistance.

Chapter 1

Introduction

1.1 Motivations

An important issue in robotics research is the development of autonomous vehicles that can operate in the real world without human intervention. An autonomous land-vehicle is an example of a system towards this goal, combining research from the areas of motion planning (in the robotics community) and motion detection (in the vision community).

A vision-based navigation system needs to recognize the presence of static objects as well as moving ones in order to avoid collisions. In other words, it needs to interact adaptively with its environment. There are two major components in such a system: sensors which provide information about the environment, and a control strategy that decides on the next move.

The first component requires that the robot senses the environment through some sort of sensory media to guide its motion. In our model, we use a camera to acquire such information. One important function of a vision system is to recognize the presence of moving objects in a scene. This problem, however, is considerably more difficult if the camera is moving. This is because of the need to detect objects moving with respect to the environment, rather than the much simpler problem of finding objects moving with respect to a stationary camera. In our research, we will examine moving object detection based primarily on *background constraints.*

The second component can be loosely defined as the capacity of the robot to decide what motions to execute in order to achieve a task specified by initial and goal spatial arrangements of physical objects. The motion planning problem, however, does not consist of a single, pre-determined problem. Instead, it can be viewed as a collection of several problems, which are more or less variants of each other.

Current research in path planning revolves around two models that are based on different assumptions about information available for planning. In the first model, called *path planning with complete information (the Piano Movers Problem),* perfect knowledge about the environment is assumed. The second model, named *path planning with incomplete information (or path planning with uncertainty),* assumes partial knowledge about the environment. This thesis uses the second model.

In our research, we try to solve the problem of motion planning in a static domain by transferring the problem of motion planning from its physical form into a geometric one. This requires two assumptions. The first is that the robot is the only moving object in the workspace (i.e. static environment), ignoring its dynamic properties. The second is to overlook the issues related to mechanical interaction between two physical objects in contact. Furthermore, we simplify the geometric issues by assuming that the robot is a single rigid object. Unlike the static motion problem, the dynamic motion problem cannot be solved by merely constructing a geometric path. Instead, a continuous function of time specifying the robot's configuration at each instant has to be generated. The dynamic motion planning problem can be made more realistic by imposing constraints on the robot's acceleration and velocity bounds. It is worth mentioning that, from a computational point of view, the problem with bounded velocity and acceleration is substantially harder than the problem without such bounds.

Next we present a brief summary of the existing techniques for motion planning.

1.2 Planning Approaches: An Overview

There exist many methods for solving the basic motion planning problem. As yet only a few solve the problem in its generality. For instance, some methods require a two-dimensional workspace; others require the obstacles to be of a specific shape. Despite the external differences, the existing methods can be classified into three main approaches: roadmap, cell decomposition, and potential fields. In addition, a minor fourth approach which uses tactile sensing will be considered. These methods are introduced briefly in the following subsections.

1.2.1 Roadmap Approach

The general idea behind roadmaps is to capture the connectivity of the robot's free space in the form of a network of one-dimensional curves. Once it is constructed, path planning is reduced to connecting the initial and goal configuration by searching the roadmap. The resultant path is a concatenation of three subpaths: a subpath from the initial configuration to the roadmap, a subpath contained in the roadmap, and a subpath connecting the roadmap to the goal configuration.

The fundamental idea in this approach is obviously the construction of the roadmap. Different methods have been proposed to produce various types of roadmaps. They are: visibility graphs, Voronoi diagrams (retraction), freeways, and silhouettes. With the exception of the last method, the others are limited to two-dimensional (or three-dimensional) spaces. However, they are efficient and easier to implement.

The visibility graph method is not only one of the earliest for path planning, it is also one of the simplest [26]. It is constructed by connecting every pair of the obstacle's vertices in the free space by a straight segment which does not pass through the interior of any other obstacle. An example of a visibility graph is shown in Figure 1.1. This method has been used widely in the implementation of path planners for mobile robots, and several algorithms have been proposed. Such examples are given elsewhere [22, 2, 13]. Moreover, the method has been considered a seed for a stream of research work on the *shortest path problem* [1]. Extensions of the basic concept of this method are studied by Canny [7].

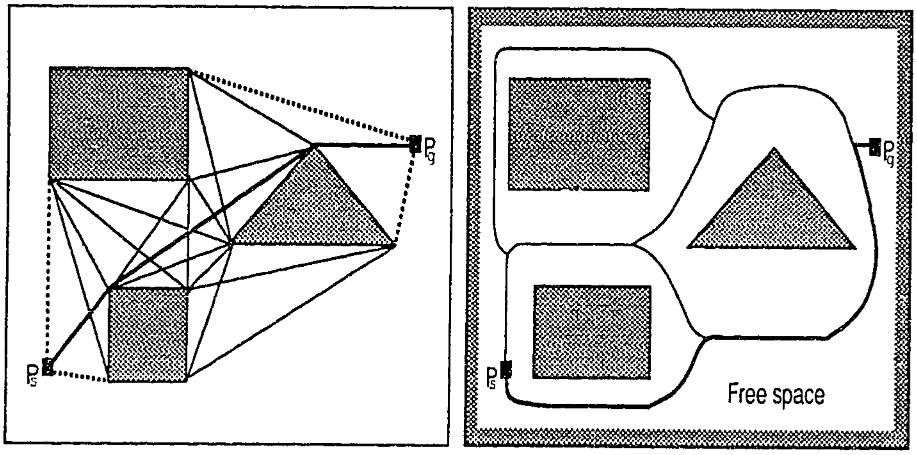

Figure 1.1: *(Left) The visibility graph for a set of obstacles. (Right) The Voronoi diagram for the same set of obstacles (note that the free space is assumed to be externally bounded by a polygon). Free paths are shown in bold straight (and parabolic) segments.*

The retraction method [27] consists of constructing a roadmap by defining a continuous mapping, called retraction, of the free workspace onto the roadmap. In a two-dimensional space, the free workspace is retracted on its Voronoi diagram (one-dimensional subset of the free workspace that maximizes the clearance between the robot and an obstacle). This diagram is the set of all possible free positions whose minimal distance from the obstacles is achieved with at least two points on the boundary of two obstacles. The free paths that generated by this method are safer than the paths generated by the visibility graph methods. This is because this method maximizes the clearance between the robot and the obstacles. When the obstacles are of a polygonal type, the Voronoi diagram consists of straight and parabolic segments as shown in Figure 1.1. Different algorithms have been proposed

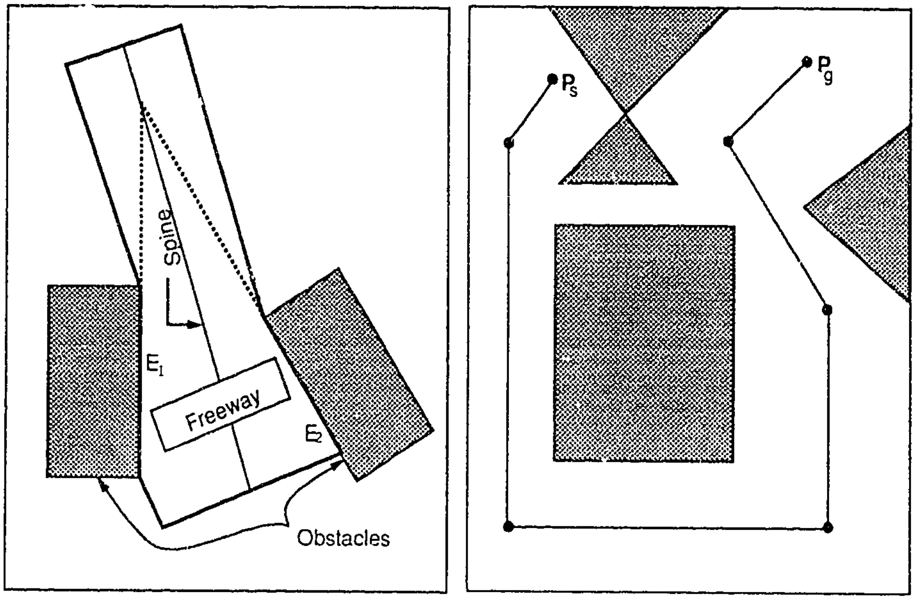

Figure 1.2: *(Left) The basic geometry of a freeway. (Right) A path generated using the freeway method is shown. The robot translates along the spines (straight line segments).*

based on this method [21, 12].

The freeway method, developed by Brooks [5], is specifically applied to robots translating and rotating among polygonal obstacles. It is simply a graph (freeway net) in which geometric figures (freeway) from the workspace are connected. A freeway is a straight linear generalized cylinder whose straight axis (spine) is annotated with the description of the free orientation of a robot when it moves along it. The freeway net is a representation of the possible motions of a robot along the spines and between them. The pair of edges of different obstacles that construct the freeway should be facing each other. Figure 1.2 illustrates the geometry of a freeway, and shows a free path where the robot translates along the spines and may rotate at the intersecting points. It has been shown that this method works quickly for a relatively uncluttered workspace. The method, however, is incomplete, and hence may not always find a free path even if one exists [14].

Canny [7] developed the silhouette method which is guaranteed to find a path if one exists, or reports failure otherwise. It is the only known complete path planning algorithm. Nevertheless, it is complex to understand and to implement. It involves tools from Differential Geometry, Elimination Theory, and Topology.

1.2.2 Cell Decomposition Approach

Cell decomposition methods are the most extensive methods studied so far. The basic idea is to decompose the free workspace into simple regions called cells. A non-directed graph (connectivity graph) representing the adjacency relation between the cells is constructed and then searched for a free path. Each cell represented by a node and every two adjacent cells are linked together. This approach can be subdivided into two classes of methods: exact and approximate.

Exact methods decompose the free workspace into cells whose union is exactly the free space. As in the Voronoi diagram, the free space is externally bounded by a polygon. The free space is exactly decomposed into rectangular cells. These cells are constructed by drawing vertical rays from the obstacle's vertices. Two cells are adjacent if they share a portion of an edge of nonzero length. Once the connectivity graph has been built, a free path is computed by connecting the initial and goal positions through the midpoints of the intersections of every two successive cells. Several references [29, 8, 3] provide examples of planning algorithms that use this method.

The difference between approximate methods and exact methods is that the union of the decomposed cells of the workspace does not always give the original free space. Furthermore, the cells in this class are required to have a simple pre-specified shape (e.g. rectangular shape). Such cells do not in general allow one to represent the free space exactly. Instead, an approximation of this free space is constructed. The free space is recursively decomposed into smaller rectangles. Each decomposition generates four identical new rectangles. If the interior of a rectangle lies completely in either the free space or an obstacle, it will not be decomposed any further. Otherwise, the process of decomposition will continue until a predefined resolution is attained. The free path is obtained as a sequence of free cells that are adjacent to each other starting with the cell containing the initial position and ending at the cell containing the goal. This technique was first proposed by Lozano-Peréze and Brooks [6]; then, it was used by other researchers [10, 15].

1.2.3 Potential Field Approach

The potential field approach presents a different idea other than constructing the global connectivity of the robot's free space in the form of a graph that is subsequently searched for a path. Rather, it treats the robot represented as a point in the configuration space as a particle under the influence of an *artificial potential field*. The potential function is typically defined as the sum of an *attractive potential* pulling the robot towards the goal and a *repulsive potential* pushing the robot away from the obstacles.

The potential field method was originally developed as an on-line collision-avoidance approach applicable when the robot does not have a prior model of obstacles, but rather senses them while in motion [17]. In this approach emphasis is put on real time efficiency, rather than on guaranteeing the attainment of the goal. Because of the on-line criterion, this approach may get stuck at a local minimum of the potential function other than the goal configuration. Therefore, most planning methods based on the potential field approach are not complete. However, some of them are quite fast in a wide range of situations [18, 19, 11, 4]. A generalized potential field function, proposed by Krogh [20], is an extension in which the potential function is defined by both the position and the velocity of the robot. It is constructed in such a way that the robot is repelled by an obstacle only if it is close to that obstacle and its velocity points towards the obstacle. An application of this idea is reported by Tilove [31].

1.2.4 Algorithms Based on Tactile Sensing

Lumelsky and Stepanov [24] proposed two path planning algorithms where no prior knowledge of the workspace is available. They assumed that the robot is equipped with a position sensor and a touch sensor. It is then shown that tactile sensory information is sufficient to guarantee reachability of the goal. They also have presented upper and lower bounds on the length of the paths generated by the algorithms. Later, Sankaranarayanan and Vidyasagar [28] proposed new set of algorithms for the same problem but with lower time complexity. In [23], a study is made after incorporating vision into the navigation module as another sensor. Lumelsky and others [25, 30, 9] described other algorithms in robot motion planning (for manipulators and mobile robots).

1.3 Thesis Organization

This thesis is presented in a paper format. Each subsequent chapter, except the last one, represents a separate research study[1].

In Chapter 2 we describe a heuristic technique for solving the problem of path planning based on local information for a mobile robot with acceleration constraints moving amidst a set of stationary obstacles. The concept of safety is introduced to design a planning strategy. A path which maximizes the product of safety (based on local information) and attraction towards the goal is chosen. The safety function depends on the acceleration bounds. The attraction towards the goal depends on the distance from the goal. Two additional heuristics are proposed to improve the efficiency of the search process, and to enhance the ability of the robot to avoid obstacles. Furthermore, we incorporate a sensing capability such as vision in the planning model.

A new approach to generating smooth piecewise local trajectories for mobile robots is proposed in Chapter 3. Given two configurations (position and direction), we search for the trajectory that minimizes the integral of acceleration (tangential and normal). The resulting trajectory should not only be

[1] Chapters 2 and 3 are already accepted as journal publications whereas Chapters 3 and 4 are still under revision.

smooth but also safe in order to be applicable in real-life situations. Therefore, we investigate two different obstacle-avoidance constraints that satisfy the minimization problem. Unfortunately, in this case the problem becomes more complex and not suitable for real-time implementations. Therefore, we introduce two simple solutions, based on the idea of polynomial fitting, to generate safe trajectories once a collision is detected with the original smooth trajectory. Simulation results of the different algorithms are presented.

In Chapter 4 we address the problem of planning a safe path for a mobile robot, in the presence of moving obstacles, based on local information. We consider constraints on the speed and acceleration of the robot. Moreover, we take into account uncertainty in estimating the velocity of each obstacle. The concept of safety is used to design a planning strategy. A path which maximizes the product of static safety, goal attraction, and dynamic safety, is chosen. The static safety depends on the acceleration bounds. The goal attraction depends on the distance from the goal. Dynamic safety depends on how far the robot is from a moving obstacle. We show that the velocity-decomposition technique [16] is a sub-case of our approach when the direction of motion is fixed locally. Simulation results of this approach are presented.

We introduce a technique for detecting moving objects from a moving camera using the *background constraint* in Chapter 5. The camera is mounted on a platform that can both rotate and translate. Motion is detected by computing a mapping that correlates pixels in successive images whenever it is possible. Assuming the background is planar provides knowledge that is used to recover projected local velocity (u, v). Background compensation can be used to eliminate the effects of rotation and translation on the mapping function. To increase the robustness of this technique, false motion caused by inaccuracies in sensor readings are eliminated. This is achieved by using a morphological filter, which consists of two successive operations *erosion* and *dilation*, performed on the motion detection image. Experimental results with real images are presented, which show the robustness of the proposed algorithm in detecting independently moving objects from a mobile platform.

The interrelationships among Chapters 2, 3, 4, and 5 and the conclusions arrived at due to this whole work are presented in Chapter 6. As well, possible future research avenues are discussed.

Bibliography

[1] V. Akman. Unobstructed shortest paths in polyhedral environments. *Lecture Notes in Computer Science*, 251, 1987.

[2] T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1):49–63, 1986.

[3] J. Bañon. Implementation and extension of the ladder algorithm. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1548–1553, May 1990.

[4] J. Barraquand, B. Langlois, and J. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. *Proceedings of the Fifth International Symposium of Robotics Research*, 1:74–83, 1989.

[5] R. Brooks. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, Cybernetics*, 13(3):190–197, March/April 1983.

[6] R. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. *Proceedings of the 8th International Conference on Artificial Intelligence*, pages 799–806, 1983.

[7] J. Canny. The complexity of robot motion planning. *MIT press, Cambridge, MA*, 1988.

[8] B. Chazelle. Approximation and decomposition of shapes. *in [Schwartz and Yap, 1987]*, pages 145–185, 1987.

[9] E. Cheung and V. Lumelsky. Motion planning for a whole sensitive robot arm manipulator. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1:344–349, 1987.

[10] B. Faverjon. Object level programming of industrial robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1406–1412, 1986.

[11] B. Faverjon and P. Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1152–1159, 1987.

[12] S. Fortune. A sweeping algorithm for voronoi diagrams. *Proceedings of the second ACM Symposium on Computational Geometry*, 1:313–322, 1989.

[13] S. Ghosh and D. Mount. An output sensitive algorithm for computing visibility graphs. *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, 1:11–19, 1987.

[14] L. Gouzénes. Strategies for solving collision-free trajectories problems for mobile and manipulator robots. *International Journal on Robotics Research*, 3(4):51–65, 1984.

[15] S. Kambhampati and L. Davis. Multiresolution path planning for mobile robots. *IEEE Transactions on Robotics and Automation*, 2(3):135–145, 1986.

[16] K. Kant and S. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotics Research*, 5:72–89, 1986.

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

[18] P. Khosla and R. Volpe. Superquadric artificial potentials for obstacle avoidance and approach. *Proceedings of the IEEE International Conference on Robotics and Automation*, 3:1178–1784, 1988.

[19] D. Koditschek. Exact robot navigation by means of potential functions: some topological considerations. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1:1–6, 1987.

[20] B. Krogh. A generalized potential field approach to obstacle avoidance control. *International Robotics Research Conference*, August 1984.

[21] D. Leven and M. Sharir. Intersection and proximity problems and voronoi diagrams. *in [Schwartz and Yap, 1987]*, pages 187–228, 1987.

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

[23] V. Lumelsky and T. Skewis. A paradigm for incorporating vision in the robot navigation function. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2:734–739, 1988.

[24] V. Lumelsky and A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, AC-31(11):1058–1063, 1986.

[25] V. Lumelsky and A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[26] N. Nilson. A mobile automaton: an application of artificial intelligence techniques. *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, 1:509–520, 1969.

[27] C. Ó'Dúnlaing, M. Sharir, and C. Yap. Retraction: A new approach to motion planning. *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 207–220, 1983.

[28] A. Sankaranarayanan and M. Vidyasagar. A new path planning algorithm for moving a point object amidst unknown obstacles in a plane.

Proceedings of IEEE International Conference on Robotics and Automation, 3:1930–1936, 1990.

[29] J. Schwartz and M. Sharir. On the piano movers problem: II. general techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

[30] K. Sun and V. Lumelsky. Motion planning with unc ctainty for a 3D cartesian robot arm. *Proceedings of the Fifth International Symposium of Robotics Research*, 1:57–64, 1989.

[31] R. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. *Proceedings of IEEE International Conference on Robotics and Automation*, 1:566–571, 1990.

Chapter 2

Heuristics for Local Path Planning[†]

2.1 Introduction

The problem of path planning for autonomous vehicles can be classified according to two criteria: global versus local, and exact versus heuristic.

Global path planning requires a complete specification of the environment. This may not be possible in many real life situations. For example, when driving on the road we only have local information about surrounding obstacles and vehicles. Thus we need planning strategies based on local information.

[†] A. Elnagar and A. Basu. Heuristics for Local Path Planning. *IEEE Transactions on System, Man, and Cybernctics*, 23(2):624-634, 1993. Preliminary results of this paper appeared in the Proceedings of the 1992 IEEE International Conference on Robotics and Automation.

Exact path planning requires that a path be found if one exists. This can often be very difficult to accomplish even when a complete and precise description of the surroundings is available. Various researchers have proved that most exact path planning problems are intractable, unless the environment or planning task is extremely simple. Heuristic strategies, on the other hand, may not find a path even if one exists. However, these methods have significantly lower time complexity. In situations where complete information about the environment is not available, heuristic schemes are useful since global algorithms are not applicable.

The algorithm discussed in this work is a heuristic strategy based on local information. We introduce a new concept of "safety-optimization" for path planning. Consider the problem of driving on the highway. An experienced driver will not drive a sports car and a family sedan the same way. It is safer to drive a sports car at higher speeds than a family sedan because of: higher bounds on its normal acceleration (due to lower center of gravity, better tires, etc.) and better ability to speed up or stop (i.e., higher tangential component of acceleration). There are two factors that control our strategy when we drive: how safe it is to drive with a certain speed given the present environment and how far we are from where we want to go. These factors are henceforth referred to as safety and goal attraction, respectively. Safety is a function of the speed. It is important to note that it may not necessarily be safer to drive slower since speeding up would be more difficult. Similarly, driving too fast may make it more difficult to slow down when necessary.

Goal attraction is inversely proportional to the distance from the goal (i.e. it attracts the robot towards the goal). This ensures that a generated path gets closer to the goal, if possible. Also, the importance of the attraction towards the goal affects safety. Goal attraction in effect measures the urgency of reaching the goal. For example, the way we drive can be quite different depending on whether we are late for work or going on a weekend vacation.

In order to solve the problem of finding collision-free paths based on local information efficiently, it is often necessary to develop a heuristic approach. Thus, we introduce two special-purpose heuristics: the "dead-end" and the "avoid-region". Both heuristics will prune the search space and guide the mobile robot towards its destination avoiding collision.

The next section briefly describes work in the area of path planning. Section 2.3 introduces the notation used and defines our problem. The concept of safety optimizing path planning is described in Section 2.4. Section 2.5 describes the algorithm for planning a local path. Obstacle avoidance heuristics are described in Section 2.6. Some experimental results are given in Section 2.7.

2.2 Previous Work

In the past, several authors [16, 8, 14, 11, 27, 1] have worked on the path planning problem in a static, completely known environment. In [10], Hary and Torras proposed an approach based on the idea of a configuration space [16]. All the above references used global methods, which can generally be viewed as a search process for a path in a graph. Several fundamental questions related to the complexity of various formulations of the path planning problem in a static and known environment are answered in [3, 23, 6, 4]. But in reality, information about the environment is generally not completely known, with the exception of some specific industrial environments.

Khatib [12] has indicated that global methods will limit the real time capabilities of robots in a cluttered environment because of the time needed to perform the planning task. Consequently, he and other researchers studied the problem of navigating between initial and goal positions in a static and unknown environment.

Khatib [12] and Krogh [13] developed the idea of using artificial potential fields from two different perspectives. The main idea was to model a repulsive potential field around each obstacle and an attractive potential field around the goal. The net force of both fields will push the robot away from the obstacles and towards the goal.

Chattery [5] introduced some heuristics to move a mobile robot using sonar sensors, in an unknown environment. However, these schemes cannot always guarantee a path simply because the mobile robot can either be trapped or oscillate between obstacles under certain conditions. As a result, several researchers attacked this problem proposing different solutions, for example see [7].

Tilove [26] presented an overview of the artificial potential field method, described the common variations in a unified framework, compared the performance of different algorithms, and corrected some misunderstandings on this topic.

In [20], Mitchell presented a path planning algorithm for a special domain which contains roads, trees, and cars. Rowe [21] extended Michell's work by including some extra objects in the domain and by considering some new heuristics.

Lumelsky and Stepanov [18, 19] studied the problem of reaching a given goal position in an unknown static environment. They proposed two non-heuristic algorithms to solve this problem. Recently, Sankaranarayanan and Vidyasagar [22] discussed the time complexity of the previous non-heuristic algorithms, and proposed a new set of algorithms with a lower complexity. Unfortunately, these algorithms are not always applicable in real time, especially in cluttered environments where intensive computation is needed to generate a path. Lumelsky and Skewis proposed other algorithms for the same problem after incorporating vision [17, 24].

Steer and Larcombe [25] presented an algorithm designed to provide a robot vehicle with sufficient intelligence to optimize its behavior while navigating between different configurations in the workspace. They studied issues concerned with producing planned paths that cannot be achieved in practise by a real vehicle.

The method presented in [9] contributed some of the basic ideas used in developing the algorithm described here. Our algorithm computes a path under certain constraints on the mobile robot such as the acceleration bounds and the local visibility. Now, we will introduce some notation and define the problem.

2.3 Notation and Basic Definitions

First we describe a theoretical approach to addressing our problem to motivate the concepts, such as safety optimizing strategies, introduced in this paper. Then we will describe a practical approach to this problem and show some implementation results.

2.3.1 Notation and Problem Statement

Specifically, the problem can be stated as follows:

A mobile robot (MR) has to be continuously moving from p_s, the starting point, with an initial orientation (θ) to p_g, the goal, in a 2-dimensional plane with obstacles. At any point on its path, the robot knows only its current co-ordinates and those of the goal. Nothing is known about the obstacles outside a given visible region. Also there are bounds on the speed and acceleration of the MR. Under the above conditions we want to find a local velocity such that a given function (to be defined) is optimized. Failure is reported if a path does not exist.

To address the above problem we need the following notations:

Notations	
Symbol	Definition
$\vec{v}$	velocity : (v_x, v_y)
s	speed : $\sqrt{(v_x^2 + v_y^2)}$
s_f	future speed, an unknown variable which is assumed to follow a certain distribution.
A_T	Maximum tangential component of acceleration.
A_N	Maximum normal component of acceleration.
ΔT	Time within which a velocity change needs to be achieved.
S	Maximum bound on speed.
$a_T(t)$	Tangential acceleration at time t.
$a_N(t)$	Normal acceleration at time t.
$\kappa(t)$	Curvature at time t.
θ	Orientation of the robot.
ϕ	Angle of visibility field.
r	Radius of visibility field.
n	Number of obstacles.
m	Number of rays in any local search window.
k	Number of local search windows.
I	Indicator function, defined by : $I(A) = \begin{cases} 1 & \text{if condition A is true} \\ 0 & \text{if A is false} \end{cases}$

2.3.2 Definitions

To introduce the problem formally, some basic definitions are needed. Consider W to be the workspace of all possible positions in which a mobile robot (MR) may be placed. W is represented as a subset of the Euclidean space $\Re^2$. F_W is a fixed frame of reference embedded in W. Similarly F_{MR} is the cartesian frame of MR. At any time, we shall consider the MR (circular object) as a point in W which can be represented by its coordinates with respect to F_W, and an orientation (θ) of F_{MR} with respect to F_W. The initial and goal positions in W are denoted by p_s and p_g respectively. $\hat{p}$ defines the MR position in W. The function $d : W \times W \rightarrow \Re$ denotes the Euclidean distance.

Definition 2.3.1 *A configuration q of MR is a specification of the position and the orientation of F_{MR} with respect to F_W. The position refers to the cartesian coordinates (x, y) in W, and the orientation refers to the angle (θ) between the y-axes of F_{MR} and F_W.*

Definition 2.3.2 *Let $O_0, O_1, \ldots, O_{n-1}$ be fixed static polygonal objects distributed in W. Each O_i represents a subset of W. We define a function*

$$F_O \; : \; I \rightarrow \{O_i\}_{i \in I}$$

where $I = \{0, 1, \ldots, n-1\}$ is the domain, and the family of sets $\{O_i\}_{i \in I}$ is the range .

Definition 2.3.3 *The total workspace region occupied by n obstacles is given by:*

$$W_O = \bigcup_{i=0}^{n-1} F_O(i)$$

Definition 2.3.4 *The Local Search Window (LSW) specifies the visibility range of the MR, which is defined as a circular sector with an area $\frac{1}{2}\phi r^2$, where ϕ ($\phi \in [0, 2\pi)$) is the visibility angle, and r is the radius of the visibility range. We define r^+ (r^-) as the right (left) sides of the circular sector. The LSW is represented by a set of equally spaced bounded rays, $\{R_j\}_{j \in J}$. Each ray is defined by a set of points. The number of points $= \lfloor r/\epsilon \rfloor$, where ϵ is the discretization size. $\Delta\phi$ is the angle between any two successive rays. Thus the actual set of grid points in the i^{th} LSW is:*

$$LSW^i_{gp} = \{R_j\}_{j \in J}.$$

where $J = \{0, 1, \ldots, m-1\}$, and $m = \lfloor \frac{\phi}{\Delta\phi} \rfloor$.

Definition 2.3.5 *The set of grid points that are free in the i^{th} LSW is:*

$$LSW^i_{fgp} = \{LSW^i_{gp} \setminus (\bigcup_{j=0}^{n-1} (O_j \bigcap (\bigcup_{l=0}^{m-1} R_l)))\}$$

Definition 2.3.6 *The set of grid points that are visible to the MR in the i^{th} LSW is:*

$$LSW^i_{vgp} = \{\bigcup_{l=0}^{m-1} R_l^{min}\}, \quad where$$

$$R_l^{min} = \begin{cases} R_l & if\ (R_l \bigcap_{j=0}^{n-1} (O_j)) = \{\ \} \\ R_j & otherwise \end{cases}$$

The number of grid points in $R_l = \lfloor r/\epsilon \rfloor$, whereas in R_j

$$= \lfloor \frac{min\ d(p, \hat{p})}{\epsilon} \rfloor \ \forall p \in (R_l \bigcap_{j=0}^{n-1} (O_j))$$

Definition 2.3.7 *The* **Reachability Condition** *(REACH) is a test performed at each grid point (p) to verify if it is reachable from the current position of the MR given constraints on the acceleration and the speed.*

$$REACH^i(p) \quad = \quad \begin{cases} \{p\} & \text{if } [(a_T(p) \leq A_T) \wedge (a_N(p) \leq A_N) \wedge (s_f(p) \leq S)] \\ \{\} & \text{otherwise} \end{cases}$$

Definition 2.3.8 *The set of reachable grid points for the MR in the i^{th} LSW is:*

$$LSW^i_{rgp} \quad = \quad \{\{p | p \in LSW^i_{vgp}\} \bigcap REACH^i(p)\}$$

where p is a grid point.

Definition 2.3.9 *The local free workspace along the generated path is a subset of W given by:*

$$W^{local}_{free} \; = \; \{LSW_{rgp}\}_{i \in I}, \quad I = \{0, 1, \ldots, k-1\}$$

Definition 2.3.10 *A free path of the MR from p_s to p_g is a continuous mapping*

$$T \;:\; [0, t_f] \;\rightarrow\; W^{local}_{free},$$

where $T(0) = p_s$ and $T(t_f) = p_g$.

The next section describes the two factors that are used in our planning algorithm.

2.4 Safety and Goal Attraction

There are two factors that are taken into account while computing the optimum local velocity. They are :

- The likelihood of remaining within the acceleration bounds at a future instant of time.
- How much closer are we to the goal.

For the current analysis we do not consider clearance from obstacles [26] as a criterion for safety. However, this factor can also be taken into account. The first condition above generates a value which we refer to as the *safety*. The second factor measures the *attraction to the goal*.

2.4.1 The Safety Function

To compute safety we consider all velocities in a given range to be equally likely at a future instant of time. Then, given a local velocity and the acceleration bounds, we compute the probability that the vehicle can maintain the kineodynamic constraints. Thus the safety corresponding to a speed s is:

$$f(s) \;=\; Pr[a_T(t) \le A_T \text{ and } a_N(t) \le A_N|s], \;\; 0 \le t \le t_0$$

Since obtaining an optimal acceleration strategy is not easy, we assume for the analysis:

$$\frac{d\theta}{dt} = \frac{\Delta\theta}{\Delta T}$$

i.e., the rate of change of direction is constant with respect to time in the interval ΔT in which the velocity change takes place. It follows immediately that:

$$\kappa(t) \;=\; \frac{d\theta}{dt}\frac{dt}{dl} = \frac{1}{s(t)}\frac{\Delta\theta}{\Delta T}$$

So, if s is the speed before time t_0 [1] and s_f is the speed after t_0, and ΔT is a small time interval around t_0 in which the velocity change has to be achieved, then

$$\text{maximum normal acceleration} = \begin{cases} \kappa(t)s^2 = s\frac{\Delta\theta}{\Delta T} & \text{if } s \geq s_f \\ s_f\frac{\Delta\theta}{\Delta T} & \text{if } s < s_f \end{cases}$$

Now assume that all speeds between (0, S) and all directions between (0, Π) are equally likely in future. Then,

$$\begin{aligned} f(s) \;&=\; \frac{1}{S\Pi}\int_0^s\int_0^\Pi I(s\frac{\Delta\theta}{\Delta T} \leq A_N)\;\; I(\frac{(s-s_f)}{\Delta T} \leq A_T)\;\; d\theta ds_f \\ &+\; \frac{1}{S\Pi}\int_s^S\int_0^\Pi I(s_f\frac{\Delta\theta}{\Delta T} \leq A_N)\;\; I(\frac{(s_f-s)}{\Delta T} \leq A_T)\;\; d\theta ds_f \end{aligned}$$

We can show that,

$$\begin{aligned} \text{if} \qquad & \frac{A_N\Delta T}{s} \leq \Pi \text{ and } A_T\Delta T \leq s, \quad \text{then} \\ f(s) \;=\; & \frac{A_N\Delta T}{\Pi S}(\frac{A_T\Delta T}{s} + (log(s + A_T\Delta T) - log(s))) \end{aligned}$$

Similarly, $f(s)$ can be obtained for several other cases (see Appendix A for details). It can be shown that the safety function increases monotonically to a peak, the safest speed, then decreases monotonically. Figure 2.1 shows the shape of f for a given set of parameter values.

[1] t_0 is the time between successive planning steps. We assume velocity changes continuously in the interval $(0, \Delta T)$ (i.e. acceleration is constant in this interval). Note that ΔT is less than t_0.

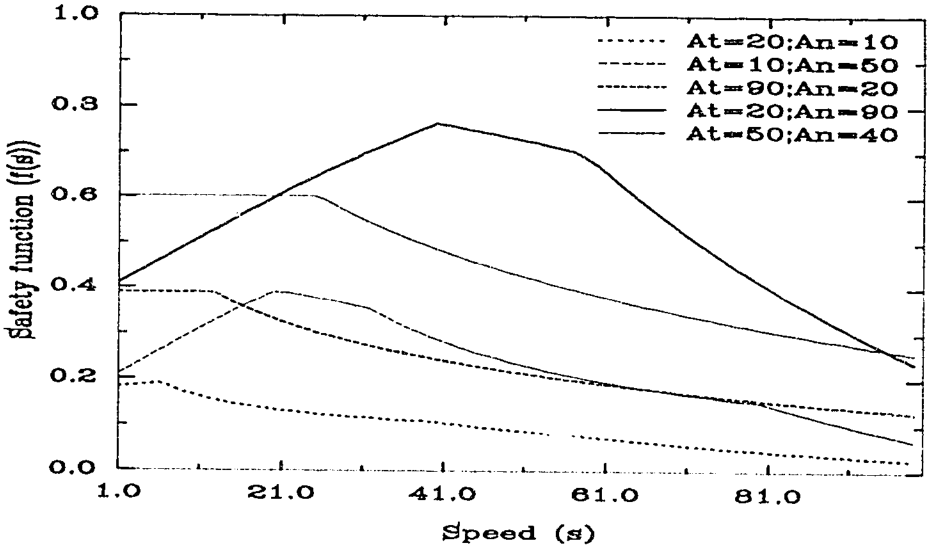

Figure 2.1: *The effect of acceleration bounds on the safety function*

Proposition 2.4.1 *The safety function f increases monotonically to a peak and then decreases monotonically. In the degenerate cases where the peak is at the left (or right) boundary, f is monotonically decreasing (or increasing).*

Proof:

Refer to Appendix B.

2.4.2 The Goal Attraction Function

The second criterion that we consider for planning a path is: "How important is it to reach the goal quickly?" If there is no urgency in reaching the goal, then all we need to do is be safe at present. As the importance of reaching the destination is increased the relative importance of being safe at present is decreased. This can be modeled by selecting a velocity which maximizes the product of the safety and the goal attraction.

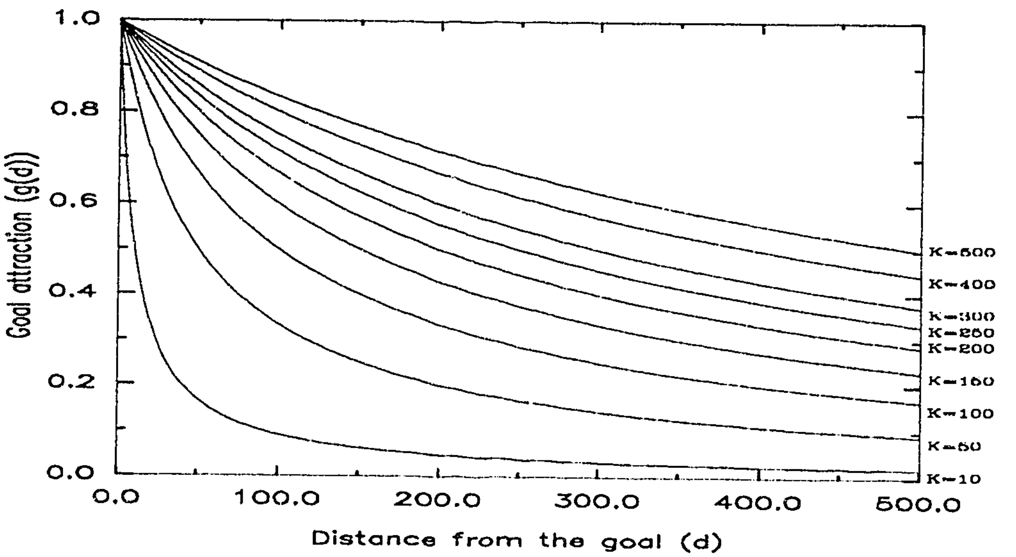

Figure 2.2: *The effect of K on g(d).*

Let us now consider the second function, the goal attraction. We define :

$$g(d) \;=\; (\frac{K}{K+d})^C, \quad \textit{where } K \textit{ and } C \textit{ are constants} > 0.$$

This implies that $g(d)$ increases as d (the distance to the goal) decreases. As in other potential field methods, this property makes $g(d)$ a parabolic-well function. The parabolic-well functions have good stabilizing characteristics: they converge to a certain maximum value (1 for $g(d)$) when the robot gets closer to the goal position [12]. Thus, if we maximize the product of f and g, the strategy would be to try to maximize safety and at the same time reduce the distance to the goal[2]. The constant K in the function g indicates the importance of getting closer to the goal. If K is large, g increases slowly as d decreases. This implies that the urgency of reaching the goal is inversely related to K. Figure 2.2 shows the shape of the function g for various values of K.

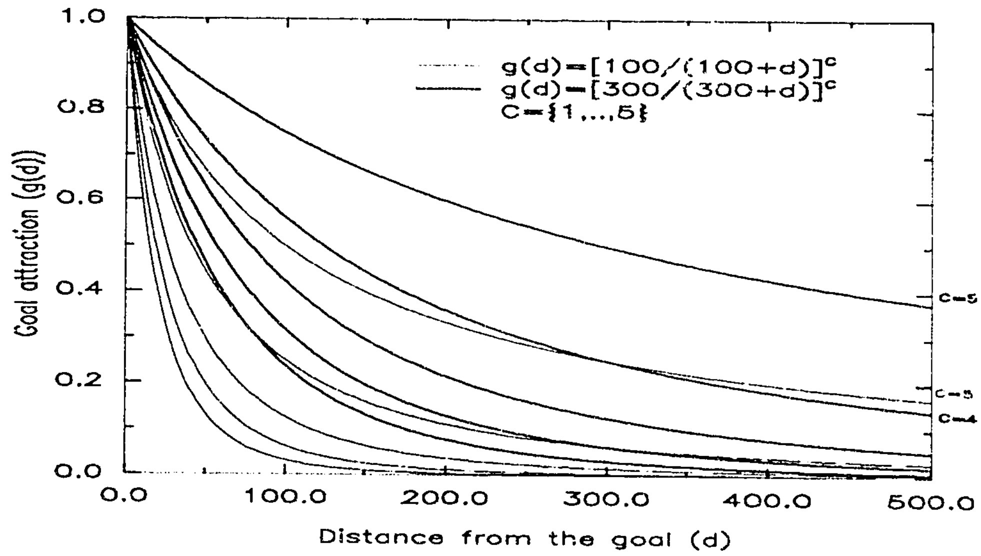

Figure 2.3: *The effect of C on g(d).*

The constant C can be used to determine if g dominates f. If C is increased, g increases more sharply as d decreases. The shape of g for various values of C is shown in Figure 2.3. We now proceed to describe our algorithm in greater detail.

[2] The attraction function ($g(d)$) may be redefined as a constant-magnitude function (conic-well type) that is independent of the distance to the goal [2]. However the conic-well functions do not have the stabilizing characteristics of the parabolic-well functions [15].

2.5 Algorithm

We model the local nature of information about the environment by considering a visibility range. Inside this range the regions that are accessible with a single velocity from the current point (referred to as reachable regions) are determined. The reachable regions are then searched, using a grid, to obtain the optimum local velocity that maximizes the product of safety and goal attraction. This process is continued until the goal is reached. Thus the algorithm used can be described by the following steps:

I. *Discretize the local visibility window.* Searching a continuous space for an optimum local solution is hard. So we discretize the local visible region into a set of equally spaced grid points, and search for the optimum in the discretized space. The size of the window searched represents the region that is visible to the robot.

II. *Find optimum velocity.* First we determine if a local grid point is reachable by a single velocity from the current point, without going through any obstacles and maintaining the acceleration constraints. The points selected are then pruned so that a direction which leads the robot into a local obstacle is avoided. The "dead-end" and "avoid-region" heuristics are used for obstacle avoidance, and are described in section 2.6. For a reachable grid point we compute the product of goal attraction and safety. Then, we choose the grid point for which this product (henceforth referred to as overall safety) is maximum. This determines the optimum local velocity.

III. *Continue process.* Repeat steps I and II until the goal is reached.

Note that we do not construct the reachable regions, but, for each grid point we check whether it is reachable or not. This makes the algorithm inherently parallel and easily implementable in real time. We now outline a simple proposition.

Proposition 2.5.1 *The local velocity selected always takes the robot closer to the goal, if any such reachable points exist, provided:*

(a) The safety function f has a positive value at 0.

(b) C is large enough.

Proof:

Suppose a robot is at a distance α from the goal. There are many reachable points that the robot can move to. However, they can be classified into two sets. The first one contains all reachable points that takes the robot closer to the goal. On the other hand, the second set contains all reachable points that does not take the robot closer to the goal. If we want the robot to get closer to the goal, if possible, the value of the overall safety for any point in the first set should be higher than the corresponding value for any point in the second set. In other words, the minimum overall safety for the first set should be greater than the maximum overall safety for the second.

- The minimum in the first set is achieved for a point which is at a distance $(\alpha - \epsilon)$, ϵ is the discretization size, from the goal, with safety s_0.

- The maximum in the second set is achieved for a point which is at a distance α from the goal with safety s_{max}.

It is important to note that $\epsilon < \alpha$ and $(\alpha - \epsilon) < \alpha$. Also, $s_0 < s_{max}$. Overall safety for the first point is:

$$s_0 g(\alpha - \epsilon) = s_0 (\frac{K}{K + (\alpha - \epsilon)})^C$$

and, overall safety for the second point is:

$$s_{max} g(\alpha) = s_{max} (\frac{K}{K + \alpha})^C$$

In order for the robot to choose the first position instead of the second one, C has to be large enough so that the following inequality is satisfied:

$$\begin{aligned}
s_0 (\frac{K}{K + (\alpha - \epsilon)})^C &> s_{max} (\frac{K}{K + \alpha})^C \\
\Rightarrow (\frac{K + \alpha}{K + (\alpha - \epsilon)})^C &> \frac{s_{max}}{s_0} \\
\Rightarrow C log(\frac{K + \alpha}{K + (\alpha - \epsilon)}) &> log(\frac{s_{max}}{s_0}) \\
\Rightarrow C &> \frac{log(\frac{s_{max}}{s_0})}{log(\frac{K+\alpha}{K+(\alpha-\epsilon)})} \\
C &= \left\lfloor \frac{log(\frac{s_{max}}{s_0})}{log(\frac{K+\alpha}{K+(\alpha-\epsilon)})} \right\rfloor
\end{aligned}$$

In the case $s_0 = s_{max}$, the first point will be always chosen, since the value of the attraction to goal function of the first point is higher than the corresponding value of the second point.

2.6 Heuristics for Obstacle Avoidance

In this section, we introduce two special-purpose heuristics that exploit domain-specific knowledge to guide the search for a solution avoiding obstacles. The "dead-end" heuristic is used to guide the mobile robot (MR) around an obstacle in a dead-end situation, and the "avoid-region" heuristic prunes the search space.

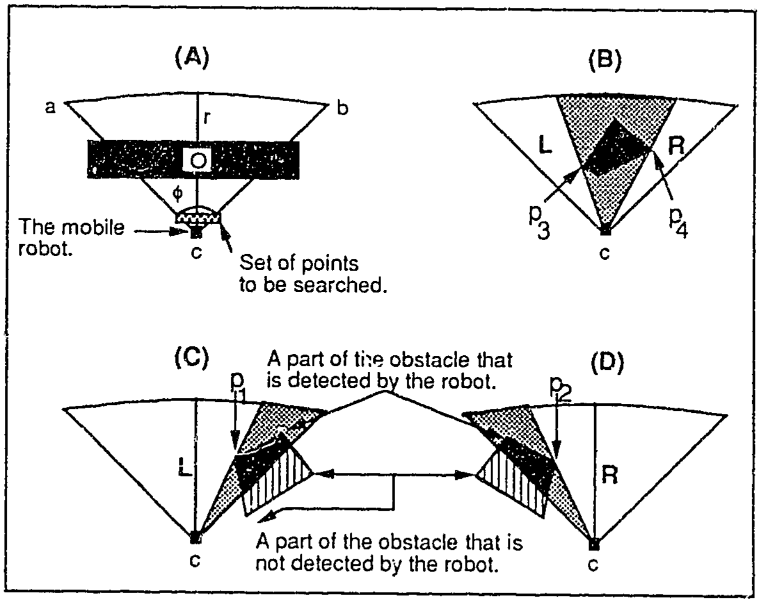
(A)
a
b
r
O
φ
The mobile robot.
c
Set of points to be searched.
(B)
L
R
p_3
p_4
c
(C)
A part of the obstacle that is detected by the robot.
(D)
p_1
p_2
L
R
A part of the obstacle that is not detected by the robot.
c
c

Figure 2.4: *(A) shows complete blockage with the MR stuck at c. (B) shows partial obstruction, where the obstacle is totally inside the MR's local search window. (C) and (D) illustrate the case where a part of the obstacle has been detected.*

The local path, computed as described in the algorithm, defines a solution unless the MR gets stuck in front of an obstacle (Figure 2.4(A)). We define a dead-end point (DEP) as follows:

Definition 2.6.1 **Dead-End Point** *(DEP) is a point where the MR faces an obstacle which blocks its LSW while navigating towards the goal.*

For example, the point c in Figure 2.4(A) is a DEP where the area (acb) represents the local search window of the MR at point c.
A local cycle is defined by:

Definition 2.6.2 *A path contains a local cycle if the MR encounters a previously visited DEP in its path towards the goal.*

2.6.1 Dead-End Heuristic

The "dead-end" heuristic is designed to avoid an obstacle whenever a DEP is encountered. Formally, it is:

Definition 2.6.3 **Dead-End Heuristic** *(DEH) is a special-purpose heuristic applied whenever a DEP is encountered. We define $(FP(R))$ as a function which takes a set (R) as input and returns the first element of that set or the empty set as output. If the MR encounters a DEP in the i^{th} LSW, then*

$$LSW_{rgp}^{DEH_i} = \bigcup_{j=0}^{m-1} \{(p|FP(R_j) = p) \bigwedge (REACH^i(p))\}$$

For instance, if there is an obstacle which blocks the local search window of the MR at a certain instant of time, the heuristic will guide the MR to go around the obstruction. This can be achieved by selecting a new position and a new orientation for the MR. The new position should be at a minimum distance from the current position of the MR, and have a maximum possible angle (ϕ) from its current orientation so that the MR avoids the obstruction as much as possible. It is clear that we have two sets of positions satisfying the above requirement. One set is to the left of the MR and the other is to its right. The choice between these sets is determined randomly at a DEP, and the chosen direction is followed until the current obstacle is avoided. If the MR encounters another DEP with a different obstacle, it will apply the heuristic again. We adopt this approach because randomization is much better than choosing a specific direction (left or right) along the path. It guarantees that the MR does not keep on moving in a loop (Proposition 2.6.2). After selecting a set of points, the MR starts testing for the point that has the maximum possible orientation angle from its current orientation (θ). If the point is reachable then the MR will move to it and start planning again from that point, otherwise it will look for the point with the next maximum orientation, and repeat the process until it finds a suitable position.

One advantage of the above heuristic is that it increases the capability of the MR to avoid dead-end situations. Another benefit is that it prunes the search space. However, it does not take into account partial obstructions, the removal of which constitutes a key factor in improving the efficiency of the algorithm (Figure 2.4(B,C and D)). Therefore we introduce the "avoid-region" heuristic.

2.6.2 Avoid-Region Heuristic

The "avoid-region" heuristic is applied whenever the MR defines its local search window. It can be defined formally as follows:

Definition 2.6.4 **Avoid-Region Heuristic** *(ARH) is a special-purpose* *heuristic applied whenever the MR encounters an obstacle or part of an obstacle in its LSW, and $d(p_s, p_g) > r$. We define $(RI(p))$ as the function which takes a point (p) as input and returns the nearest ray index to p in the i^{th} LSW as output. Then we define three cases depending on the obstacle (O_l) intersecting r^+, r^-, or neither.*

$$LSW^{ARH_i}_{rgp} = \begin{cases} \{(p|p \in (\bigcup_{j=RI(p_1)}^{m-1} R_j)) \wedge (REACH^j(p))\} & \mathit{if}\, O_l \cap r^+ \neq \{\,\} \\ \{(p|p \in (\bigcup_{j=0}^{RI(p_2)} R_j)) \wedge (REACH^i(p))\} & \mathit{if}\, O_l \cap r^- \neq \{\,\} \\ \{(p|p \notin (\bigcup_{j=RI(p_3)}^{RI(p_4)} R_j)) \wedge (REACH^i(p))\} & \mathit{if}\, O_l \subset LSW^i_{gp} \end{cases}$$

where p_1, p_2, p_3, and p_4 represent the obstacle vertices (O_l) that are in the i^{th} LSW (see Figure 2.4(B,C, and D) for explanation), and p is a grid point.

The purpose of this heuristic is to detect the nearest visible obstacle from the MR, if one exists. When the detection is successful there are two possible cases. First, when the MR detects an obstacle that is entirely inside its local search window (Figure 2.4(B)), and second when the MR detects a part of an obstacle in its local search window (Figure 2.4(C and D)). In both cases the MR will define the area which will be avoided and no longer searched. The shaded areas in Figure 2.4 represent the avoided regions. For the first case (Figure 2.4(B)) the MR will search two areas (L and R) to the left and to the right of the obstacle. Note that the MR does not need to know exactly if an obstacle is entirely inside the LSW but rather its visible parts from the MR position. In the other situation (Figure 2.4(C and D)) the MR will search only one area (L or R) either to the left or to the right of the obstacle.

In some cases the collision-free paths generated after applying the proposed heuristics may lack smoothness. This happens because the proposed search process may lead to sudden direction changes, in particular when the "dead-end" heuristic is applied. It should be noted that the MR will never generate these paths without the heuristics. Thus, we sacrifice the smoothness criterion for increasing the MR's capability of generating collision-free paths in specific environments. Experimentally it has been observed that both heuristics help in reducing the search time.

Using the obstacle-avoidance heuristics the local free workspace may be reduced. Hence the local free space needs to be redefined as:

Definition 2.6.5 *The local free workspace along the generated path is a subset of W given by:*

$$W_{free}^{local^*} = \{LSW_{rgp}\}_{i \in I}, \quad I = \{0, 1, \ldots, k-1\}$$

In the following subsection we will describe some formal results of the heuristics.

2.6.3 Properties of Heuristics

Lemma 2.6.1 *If $p_i \in LSW_{rgp}^j$ then $p_{\frac{\phi}{\Delta\phi}-i} \in LSW_{rgp}^j$ provided that $p_{\frac{\phi}{\Delta\phi}-i} \in LSW_{vgp}^j$.*

Proof:

Since p_i and $p_{\frac{\phi}{\Delta\phi}-i}$ are symmetric around the radius (r). Then

$$\begin{aligned} d(\hat{p},p_i) &= d(\hat{p},p_{\frac{\phi}{\Delta\phi}-i}), \text{ and} \\ |\theta_{p_i}-\theta| &= |\theta_{p_{\frac{\phi}{\Delta\phi}-i}}-\theta|. \end{aligned}$$

$\Longrightarrow$ Both points have the same speed and acceleration.

So, $p_i \in LSW^j_{rgp} \wedge p_{\frac{\phi}{\Delta\phi}-i} \in LSW^j_{vgp} \Longrightarrow p_{\frac{\phi}{\Delta\phi}-i} \in LSW^j_{rgp}$.□

Lemma 2.6.2 *At any DEP, the choice of a new configuration $q = (p_i, \theta_{p_i})$ should satisfy the condition:* $(max_{\theta_{p_i}\in[\theta-\frac{\phi}{2},\theta+\frac{\phi}{2}]}|\theta_{p_i}-\theta|)$, $\forall p_i \in LSW^j_{rgp}$.

Proof:

Suppose that the MR is at a DEP (p_{i-1}) with an orientation $\theta_{p_{i-1}}$. According to the algorithm, the new position of the MR (p_i) should have a maximum possible orientation angle (θ_{p_i}) from $\theta_{p_{i-1}}$. That is

$$(\theta_{p_i} \geq (\theta_{p_{i-1}} + \frac{\phi}{2})) \vee (\theta_{p_i} \leq (\theta_{p_{i-1}} - \frac{\phi}{2}))$$

provided that $p_i \in LSW^j_{rgp}$. Hence the proof. □

Proposition 2.6.1 *Consider an obstacle (O_j) that is not overlapping with any other obstacle (O_l, $l \in \{0,1,\ldots,n-1\} \wedge (l \neq j)$) in W. If the MR encounters a DEP (p) with O_j, it will define a finite minimal number of DEP's in its chosen direction of motion before clearing from O_j provided that a reachable point exists.*

Proof:

Follows from Lemma 2.6.2. □

Lemma 2.6.3 *If* $\bigcap_{i=0}^{n-1} F_O(i) = \{\}$ *then a free path always exists from* p_s *to* p_g.

Proof:

Follows directly from Lemma 2.6.2 and Proposition 2.6.1. □

Lemma 2.6.4 $W_{free}^{local^*} \subseteq W_{free}^{local}$

Proof:

Follows directly from Definition 2.3.10, and Definition 2.6.5.

Proposition 2.6.2 *The* MR *always finds a path from* p_s *to* p_g *if the following conditions hold:*

(1) *A path exists.*

(2) $W_{free}^{local} \neq \{\}$, *and* $p_g \in W_{free}^{local}$.

Proof:

To prove this proposition, we have to consider two cases :

(a) $\forall\, i,j\ ((i \neq j) \wedge (0 \leq i,j \leq n-1)) \rightarrow (O_i \bigcap O_j) = \{\}$

(b) $\exists\, i,j\ ((i \neq j) \wedge (0 \leq i,j \leq n-1)) \rightarrow (O_i \bigcap O_j) \neq \{\}$

The proof of (a) follows from Lemmas (2.6.1 and 2.6.5), and Proposition 2.6.1. Whereas to prove (b) we need to consider the case where the MR is located among a set of obstacles. Consider the case when l obstacles are overlapping (say $\{O_0, O_1, \ldots, O_{l-1}\}$ and $l \leq n$). Since a path exists (condition (1)) then there should be a gap $\geq \epsilon$ between two obstacles (say O_0 and O_{l-1}). The MR will either escape through this gap and then generate its path towards the goal (Proposition 2.6.1), or get trapped in a local cycle by looping and returning to the same DEP. But this is impossible because the probability that the MR will repeat the same cycle (after applying the DEH) forever is: $\lim_{n\rightarrow\infty} \frac{1}{2^n} = 0$. □

2.7 Experimental Results

In this section, some simulation examples of the algorithm described before are discussed. These simulations correspond to different navigation environments. As mentioned earlier, several parameters have to be input to the system. We will study the effect of varying some oi them, namely: acceleration bounds, initial orientation, and visual range. Moreover, we will present some experiments where the dead-end heuristic is applied. The moving circular sector in all figures corresponds to the local search window or the visual range. Keeping the distance between the grid points small helps in generating smooth paths. The shaded areas in the local search windows correspond to the forbidden areas generated by the avoid-region heuristic. For illustrative purposes, we used three grey levels and a unique dark shade, next to black, for indicating blockage zones (Figure 2.5).

We start with the following parameter values: distance between grid points = 1 cm, i.e. the navigation space is represented by a square grid of 10 × 10 m^2, attraction to goal factor K = 300, C = 1, tangential acceleration bound = 5 cm/sec^2, normal acceleration bound = 4 cm/sec^2, time

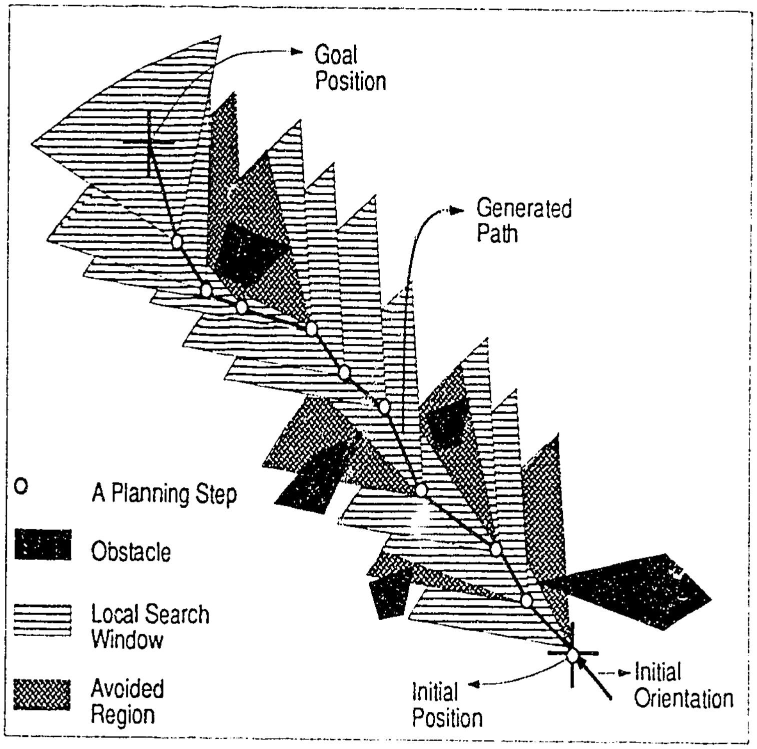

Figure 2.5: *A simulated environment to explain the shades and components of the planning process.*

to accelerate from current velocity to the next one = 3 sec, maximum speed bound = 15 cm/sec, initial speed for the car while approaching the start point = 1 cm/sec, initial orientation = 130°, time to plan a path in the local search window = 10 sec, radius of local search window = 10 cm, angle of local search window = 60°. Figure 2.6 illustrates the path generated by the MR from the starting point to the goal according to the data given above. Subsequent experiments will depend on the same set of data except for the values of one or two parameters that will be studied. The arrow, at the starting point in all figures, represents the initial orientation of the MR.

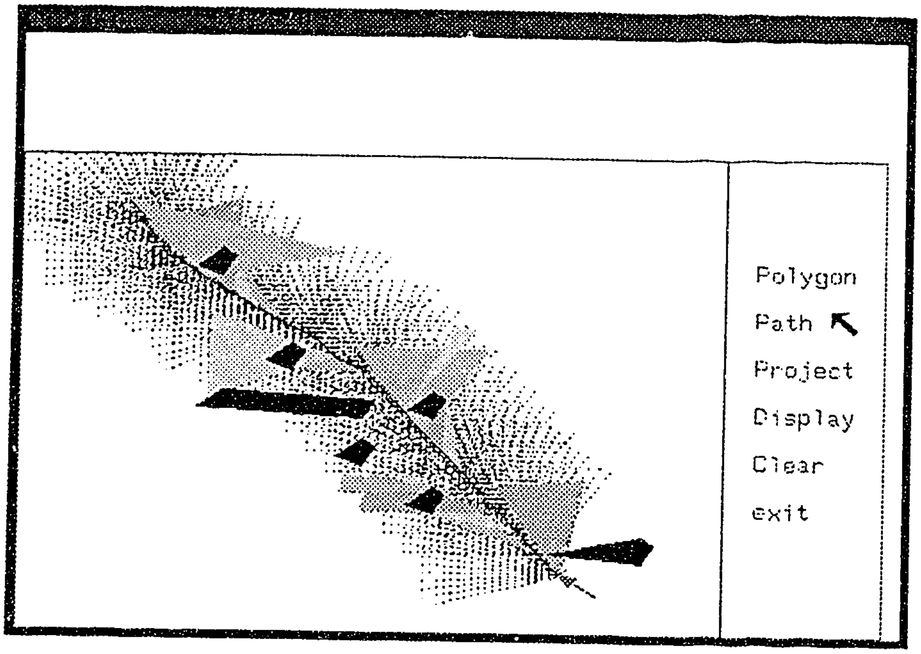

Figure 2.6: *Initial path*

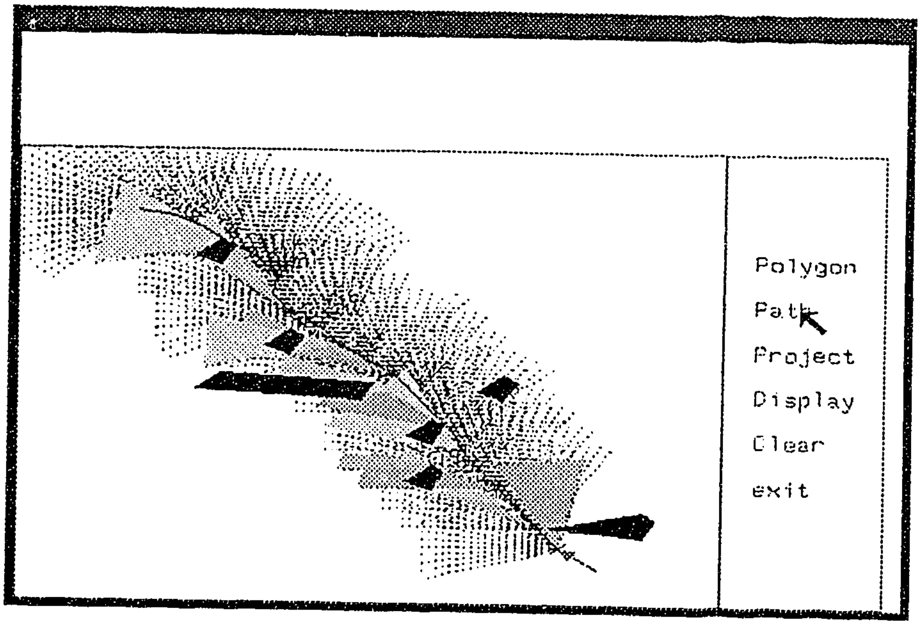

Figure 2.7: *A new path generated after a slightly different allocation of obstacles.*

Figure 2.7 shows the effect of modifying obstacle locations in the work space. It demonstrates how a slightly different allocation of obstacles may change the path. Figure 2.8 shows the path after increasing the tangential and normal acceleration bounds to 9 cm/sec^2 and 8 cm/sec^2 respectively. From the newly generated path, we conclude that the time needed for the MR to plan and follow its new path is much less than the time needed before (Figure 2.6). This is a direct consequence of the reduced number of local search windows used. The effect of changing only the normal acceleration bound is studied in Figures 2.9 and 2.10. We start with $A_n = 6$ cm/sec^2 and then decrease it to 3.5 cm/sec^2 in Figure 2.10. The resulting path shows clearly the ability of the MR to make sharper turns. For example, the turn after the third obstacle in Figure 2.10 is sharper than the corresponding one in Figure 2.9.

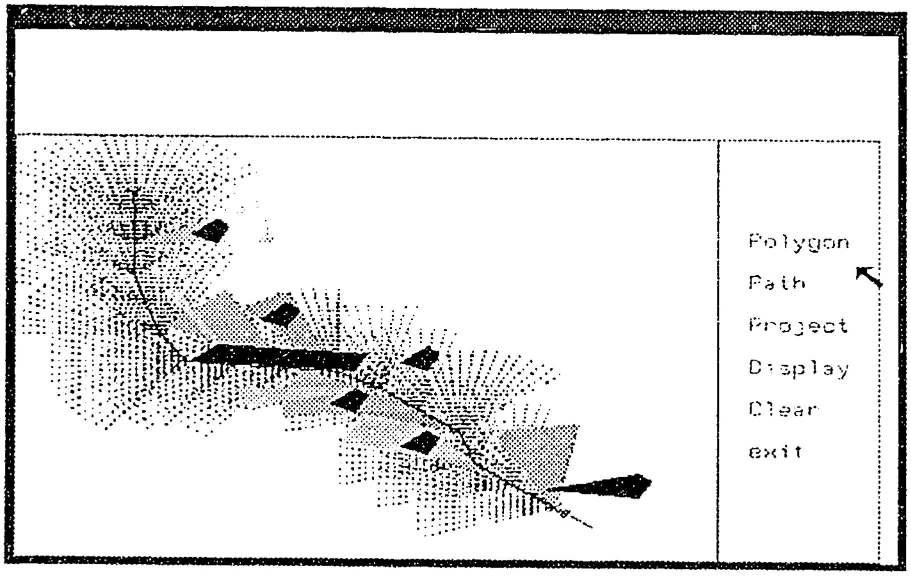

Figure 2.8: *A new Path where* $A_t = 9$ cm/sec^2 *and* $A_n = 8$ cm/sec^2.

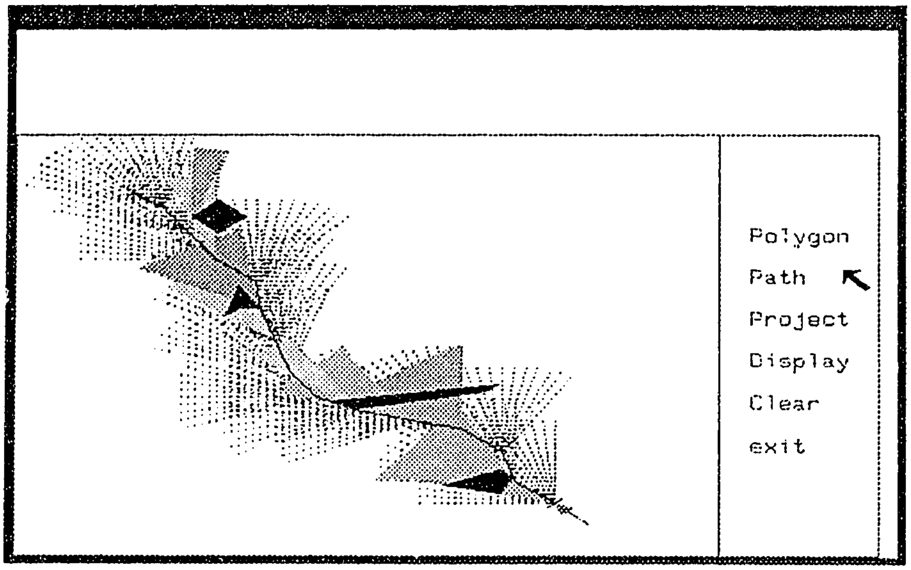

Figure 2.9: *A new path where $A_t = 9\ cm/sec^2$ and $A_n = 6\ cm/sec^2$. Note the turn after the third obstacle.*

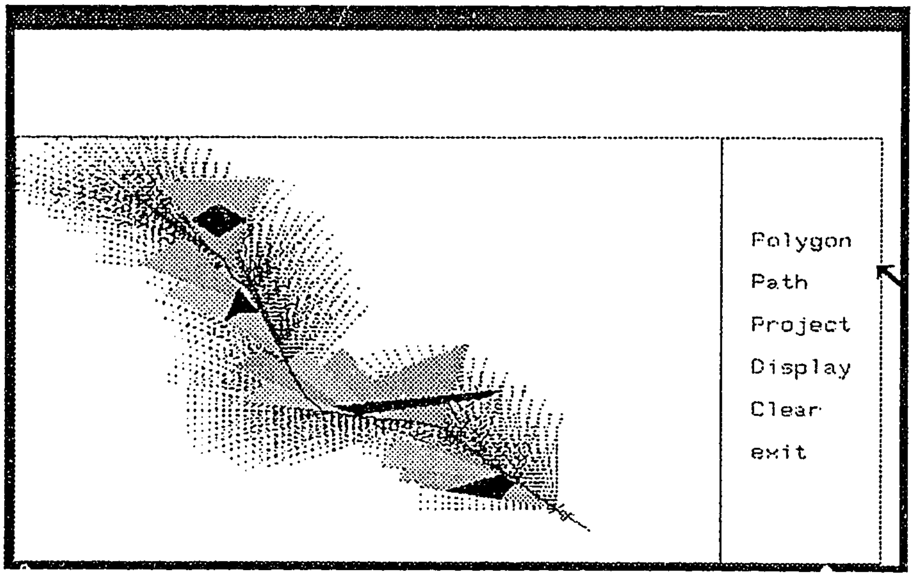

Figure 2.10: *A new Path where $A_t = 9\ cm/sec^2$ and $A_n = 3.5\ cm/sec^2$. Note how the turn after the third obstacle has been smoothed.*

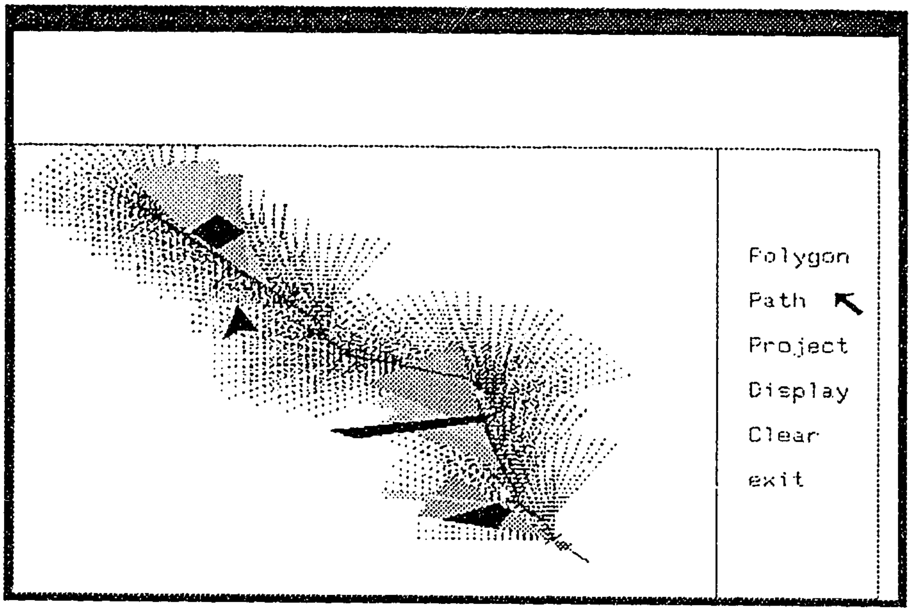

Figure 2.11: *A new path where $A_t = 5\ cm/sec^2$ and $A_n = 6\ cm/sec^2$.*

Figure 2.11 shows the path generated after reducing the tangential acceleration bound from 9 cm/sec^2, in Figure 2.9, to 5 cm/sec^2 keeping the normal acceleration bound fixed. In order to investigate the effect of tangential acceleration on smoothness of the path, we show the paths in 3D, where the vertical axis represents time. Figures 2.12 and 2.13 correspond to projections of the path in Figure 2.9 from different viewpoints. Similarly, we generate the Figures 2.14 and 2.15 of the path in Figure 2.11 from the same points of view. The comparison of both sets leads to the conclusion that the path of Figure 2.11 is smoother. For clarity, we show all obstacles only in the first set in 3D, and do not display temporally grown obstacles in the remaining figures.

Examples using the dead-end heuristic are shown in Figures 2.16 and 2.17.

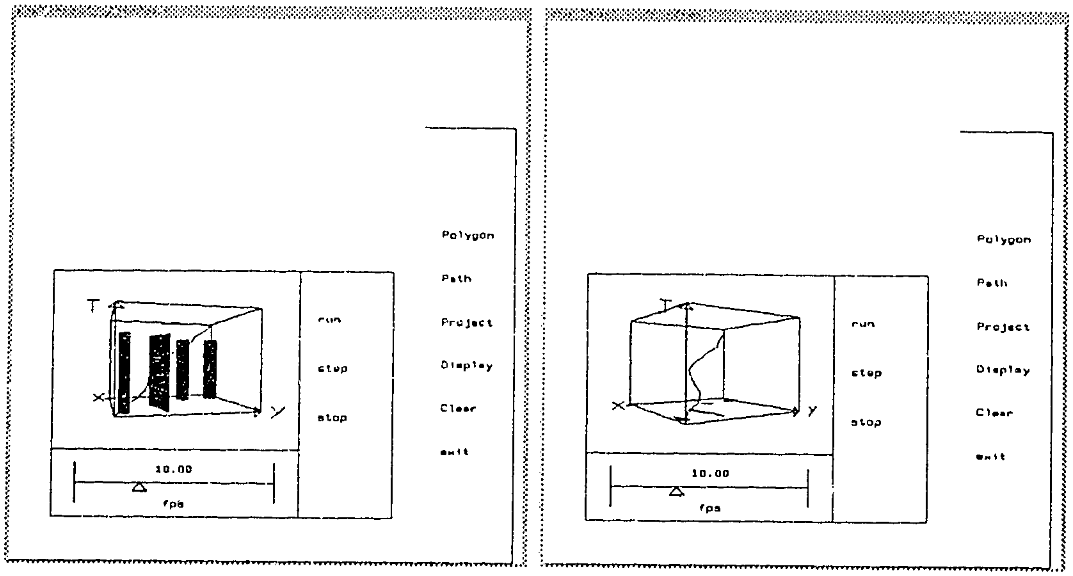

Figure 2.12: *3D projections for the path of Figure 2.9. (Left) shows the obstacles in 3D. (Right) the same path from another viewpoint.*

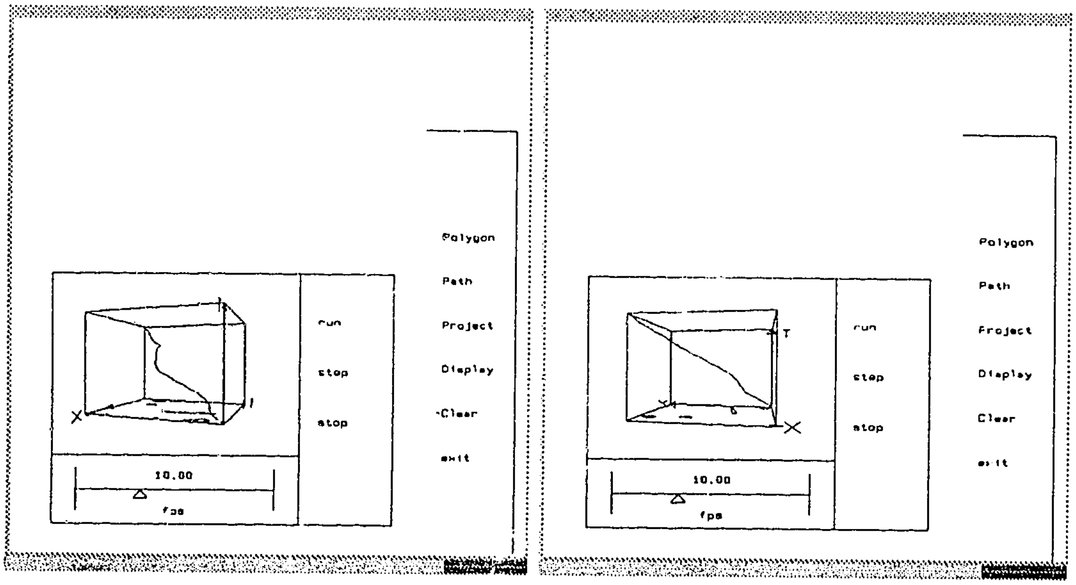

Figure 2.13: *(Left) A projection where we can see the last part of the trajectory is not very smooth. (Right) Another projection of the same path.*

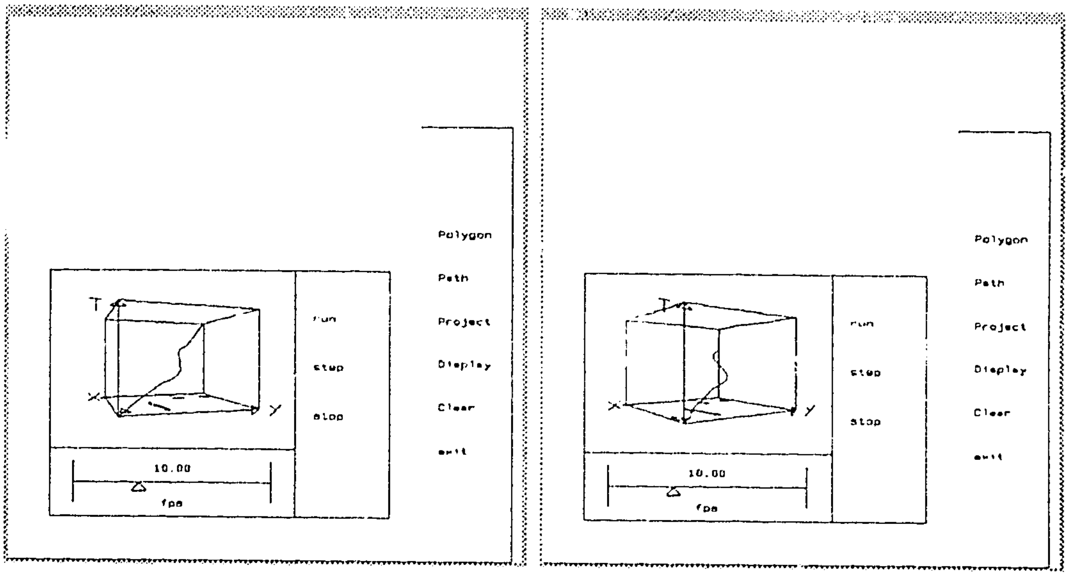

Figure 2.14: *3D projections for the path in Figure 2.11. (Left) and (right) show different views.*

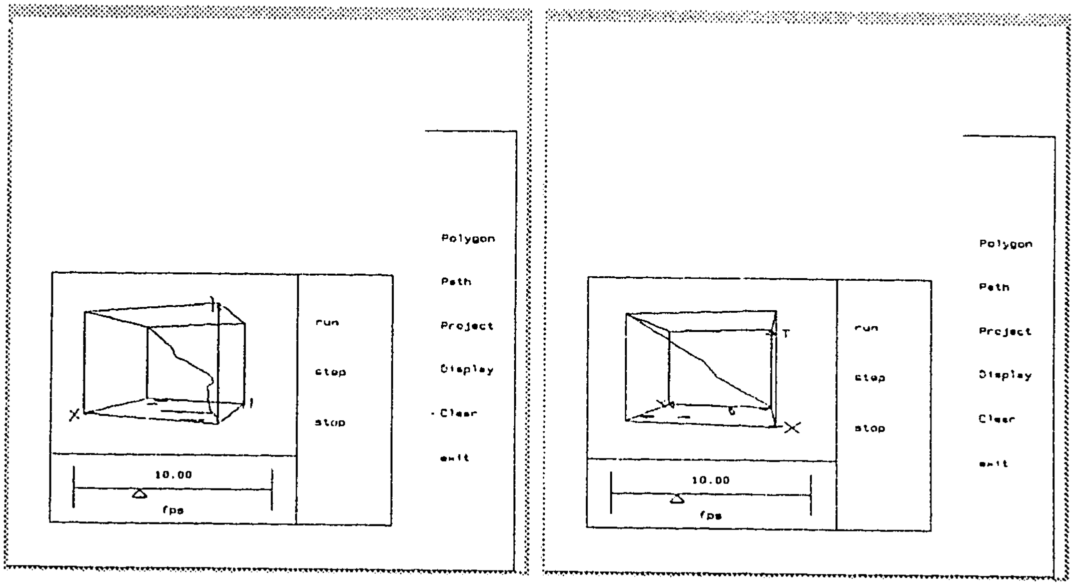

Figure 2.15: *(Left) A projection where we can see the last part of the trajectory is smooth. (Right) Another projection (compare with Figure 2.13(right)).*

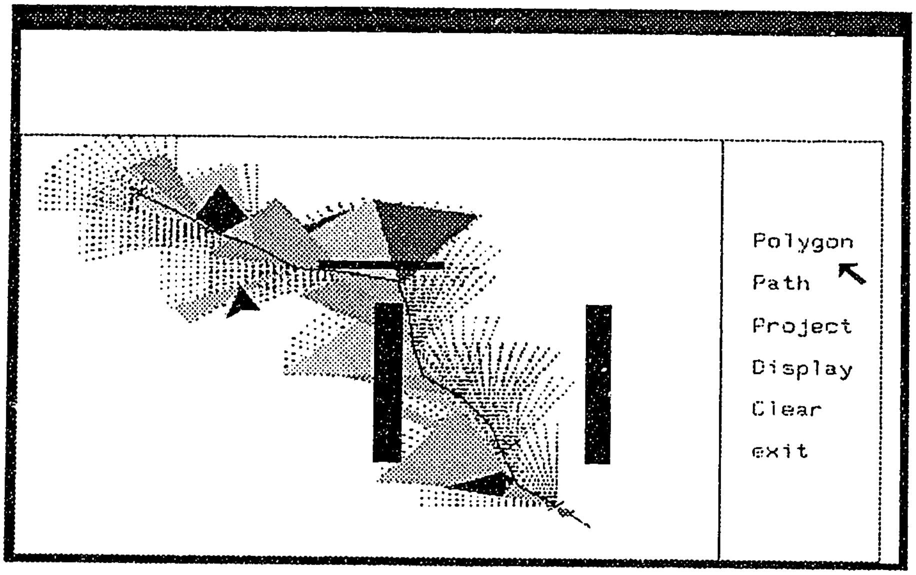

Figure 2.16: *A new path where the MR chooses the left direction. Note the DEP in front of the fifth obstacle.*

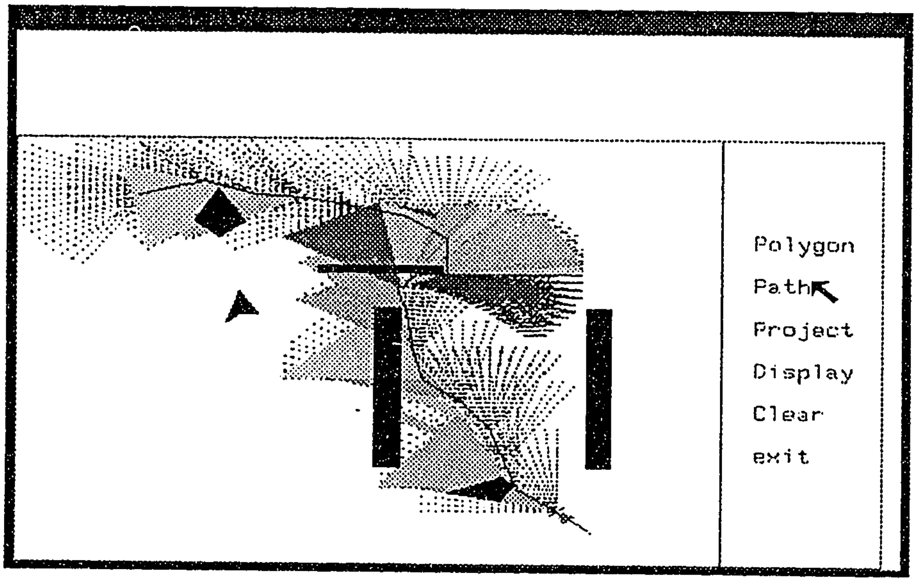

Figure 2.17: *A new path where the MR selects to go right randomly. Also, it follows the fifth obstacle boundary.*

As mentioned earlier, the direction of navigation is chosen randomly when the MR gets stuck in front of any obstacle. We show two experiments in which the MR chooses the left direction in one (Figure 2.16), and goes right in the other (Figure 2.17). The dark gray color, next to black, corresponds to the blocked region. For example, consider the DEP in front of the fifth obstacle in both experiments. Due to the goal attraction, in Figure 2.17, we notice that the MR follows the boundary of the obstacle. This will be clearly explained in the last set of experiments[3].

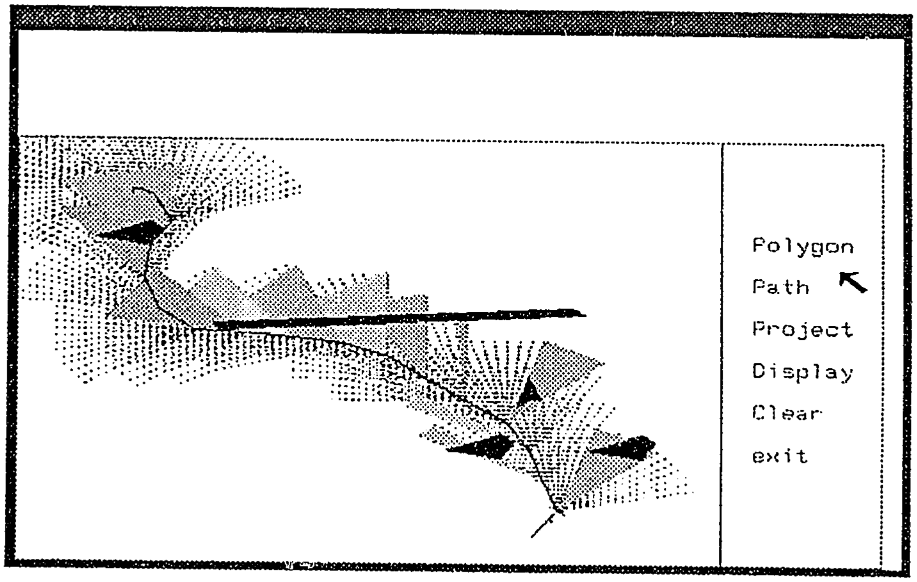

Figure 2.18: *A new path after increasing θ to* 210°.

Finally, we study the effect of changing the initial direction. In Figure 2.18 we let $\theta = 210°$, while in Figure 2.19, $\theta = 30°$. We also show two cases of obstacle boundary following. The first one appears in Figure 2.18, when

[3] Obstacles can be expanded with a certain tolerance to overcome the problem of touching obstacle boundaries.

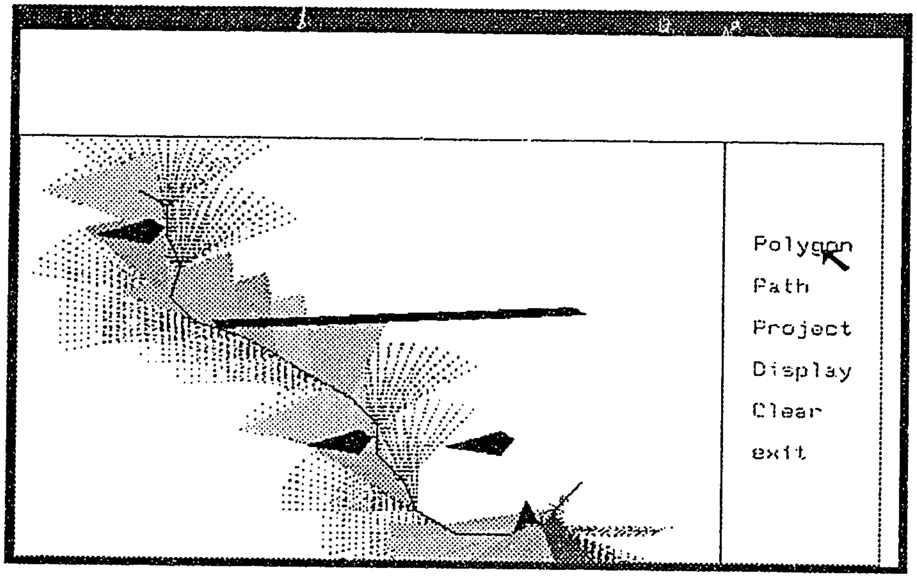

Figure 2.19: *A new path after decreasing θ to* 30°. *Note how the MR follows the boundary of the first obstacle.*

the MR detects the last obstacle on its way to the goal. The MR selects the direction shown in the Figure because it is the shortest way to the goal. Similarly, the MR finds its way around the first obstacle in Figure 2.19.

A major drawback of most potential field methods is the possibility of getting stuck at a local minima. In Figure 2.20 we illustrate this problem with an example taken from [15]. Here the robot gets trapped at position X, where the net repulsive force (F_{rep}) from the obstacle exactly balances the attractive force of the goal ($\vec{F}_{att}$). Thus X is a local minimum of the total potential function[4]. However, using our method we generate a free path for the same environment (Figure 2.21).

[4] For more details on this subject, please refer to [15, 26].

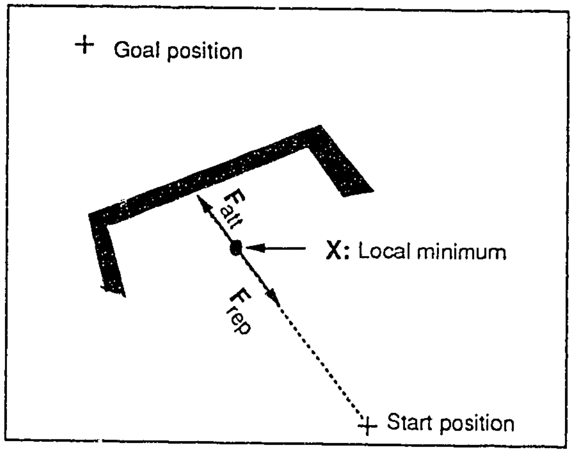

Figure 2.20: *An example where a potential field method fails to find a path.*

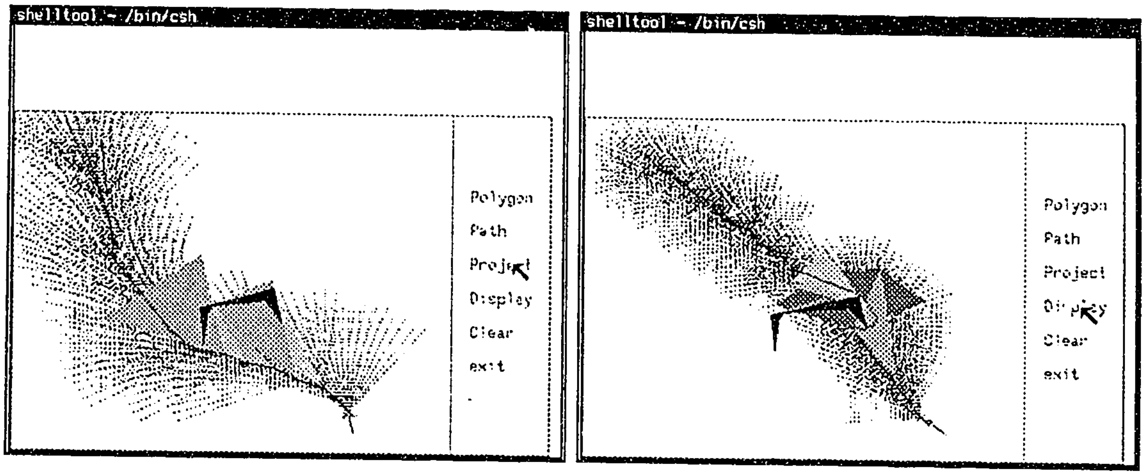

Figure 2.21: *(Left) The MR plans its path avoiding the obstacle totally using the avoid-region heuristic. (Right) The resulting path after reducing the LSW (note that the dead-end heuristic is applied first).*

2.8 Conclusion

We presented a new approach with heuristics to path planning using only local information and a knowledge of the start and the goal positions. The experimental results showed how heuristics help enhance the search process, and the robot's ability to avoid obstacles. The concept of safety was introduced. Safety is a function of the speed of the vehicle with the acceleration bounds being the parameters. In order to draw the robot closer to the goal, a goal attraction function was used. To avoid obstacles, we proposed two special-purpose heuristics: the "dead-end" and the "avoid-region". Both heuristics were used to prune the search space and to enhance the ability of the mobile robot to avoid obstacles.

In future we intend to address the planning problem in the presence of moving obstacles. For instance, the safety model described in this paper is intended for generalization to dynamic domains. We assume that the robot may need to speed up in order to avoid other moving obstacles. We have not completed the work yet.

Bibliography

[1] J. Barraquand, B. Langlois, and J. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. *Proceedings of the Fifth International Symposium of Robotics Research*, 1:74–83, 1989.

[2] J. Boerenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

[3] R. Brooks. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, Cybernetics*, 13(3):190–197, March/April 1983.

[4] J. Canny. The complexity of robot motion planning. *MIT press, Cambridge, MA*, 1988.

[5] R. Chattery. Some heuristics for the navigation of a robot. *The International Journal of Robotics Research*, 4, February 1985.

[6] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *Technical Report No. 266, New York University*, December 1986.

[7] C. Connolly and J. Burns. Path planning using Laplace's equation. *Proceedings of the IEEE International Conference on Robotics and Automation*, 3:2102–2106, 1991.

[8] J. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal on Robotics and Automation*, RA-1(1):31–41, March 1985.

[9] A. Elnagar and A. Basu. Heuristics for local path planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2481–2486, May 1992.

[10] J. Ilary and C. Torras. 2D path planning: A configuration space heuristic approach. *The International Journal of Robotics Research*, 9:75–91, February 1990.

[11] K. Kant and S. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotics Research*, 5:72–89, 1986.

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

[13] B. Krogh. A generalized potential field approach to obstacle avoidance control. *International Robotics Research Conference*, August 1984.

[14] B. Krogh and C. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1664–1669, April 1986.

[15] J. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers Group, Massachusetts, USA, 1991.

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

[17] V. Lumelsky and T. Skewis. A paradigm for incorporating vision in the robot navigation function. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2:734–739, 1988.

[18] V. Lumelsky and A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, AC-31(11):1058–1063, 1986.

[19] V. Lumelsky and A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[20] J. Mitchell. Shortest path among obstacles, zero road regions, and roads. *Paper delivered at the Joint National Meeting of TIMS/ORSA*, 1987.

[21] N. Rowe. Roads, rivers, and obstacles: Optimal two-dimensional path planning around linear features for a mobile agent. *The International Journal of Robotics Research*, 9:67–74, December 1990.

[22] A. Sankaranarayanan and M. Vidyasagar. A new path planning algorithm for moving a point object amidst unknown obstacles in a plane. *Proceedings of IEEE International Conference on Robotics and Automation*, 3:1930–1936, 1990.

[23] J. Schwartz and M. Sharir. On the piano movers' problem: I. the case of a two-dimensional rigid polygonal body motion amidst polygonal barriers. *Communications on Pure Applied Mathematics*, 36:345–398, 1983.

[24] T Skewis, J. Evans, V. Lumelsky, B. Krishnamurthy, and B. Barrows. Motion planning for a hospital transport robot. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1:58–63, 1991.

[25] B. Steer and M. Larcombe. A goal seeking and obstacle avoiding algorithm for autonomous mobile robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2:1518–1528, 1991.

[26] R. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. *Proceedings of IEEE International Conference on Robotics and Automation*, 1:566–571, 1990.

[27] C. Yap. *Algorithmic Motion Planning*. Erlbaum, Hillsdale, New Jersey, 1986.

Chapter 3

Piecewise Smooth and Safe Trajectory Planning[†]

3.1 Introduction

The problem of finding a time-minimal path within a certain class of paths defined on a time interval was first addressed by Dubins [5]. He found a solution to the shortest curvature bounded path in the absence of obstacles. Later, Laumond [13] studied this problem in the case where obstacles in the workspace must be avoided. Unfortunately, his solution is not guaranteed to always find a path. Different treatments of the above problem may be found in [9, 4].

[†] A. Elnagar and A. Basu. Piecewise Smooth and Safe Trajectory Planning. *Accepted to appear in the International Journal of Information, Education and Research in Robotics and Artificial Intelligence (Robotica).* Part of this work will appear in the Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots.

The problem of finding the plane curve of minimal elastic energy with prescribed endpoints and end-directions was solved by Horn [8]. He minimized the integral of the squared curvature in order to obtain a smooth planar trajectory that passes through two different points with given directions. Kallay [10] extended Horn's work by adding another constraint on the length of the trajectory. Recently, Brucktein and Netravali [1] presented an interpolation method, based on the work of Horn and Kallay, that is simple to implement and yields optimal trajectories with expected behavior (i.e., circular arcs in symmetric situations). Moreover, they described a simple numerical procedure for computing piecewise linear approximations of optimal trajectories as a solution to the discrete two-point boundary value problem.

Kanayama and Hartman [11] proposed "cubic spiral" curves, which provide optimal smooth paths for turns. They used two different measures for smoothness of curves: one expressed in terms of curvature and the other in terms of the derivative of curvature.

Smooth trajectories are useful in many different fields of research, such as computer graphics, geometric design, and robotics (motion planning). We want to address the problem of finding a smooth trajectory between two given positions and two given orientations of a mobile robot (MR) in 2D. The trajectory of the MR can be represented as a sequence of configurations, where each configuration specifies the position and orientation of the MR at a specific instant of time. In robotics, this problem is addressed in most of the nonholonomic motion planners that have been proposed so far by the use of straight segments of lines and/or arcs to connect configurations [9, 6].

Smooth trajectories are desirable, and may be essential in some cases, for mobile robots. We optimize a smoothness criterion which is a function of acceleration. There are two components of acceleration: tangential (forces on gas pedal or brakes) and normal (forces that tend to drive a car off the road while making a turn). We want to search for the trajectory along which a mobile robot will be able to accelerate (or decelerate) to a safe speed in an optimal way. We assume partial knowledge about the environment, and the presence of static obstacles.

In the next section we briefly describe the different components of acceleration. In Section 3.3 we introduce the minimization problem, and then divide it into two subproblems which are solved to find smooth trajectories. In Section 3.4 we answer the question: "How can we find a trajectory that is not only smooth but also safe ?". Finally, we present some experimental results.

3.2 Tangential and Normal Acceleration

Suppose that at time t a mobile robot is located at point $p = (x, y)$ with orientation $\phi(t)$ on a curve τ. τ is defined parametrically in 2D by

$$x = f(t) \text{ and } y = g(t)$$

where f'' and g'' exist. If we denote the speed ($\frac{dl}{dt}$) by s; l is the arc length along τ; and the curvature $\kappa = \frac{1}{\rho}$, where ρ is the radius of the curvature of τ, then the acceleration at time t can be expressed in terms of a tangential component ($\frac{ds}{dt}$) and a normal component ($\frac{s^2}{\rho}$) as follows:

$$a(t) = \frac{ds}{dt}T(l) + \frac{s^2}{\rho}N(l),$$

where $T(l)$ and $N(l)$ are the tangential and normal unit vectors, respectively. Note that

$$a_N = \frac{s^2}{\rho} = s^2 \frac{d\phi}{dt}\frac{dt}{dl} = s\frac{d\phi}{dt}$$

where $\frac{d\phi}{dt}$ is the rate of change in orientation with respect to time along τ. Figure 3.1 illustrates the geometric interpretation of a. If the tangential and normal components of a are denoted by a_T and a_N, respectively, then we may write:

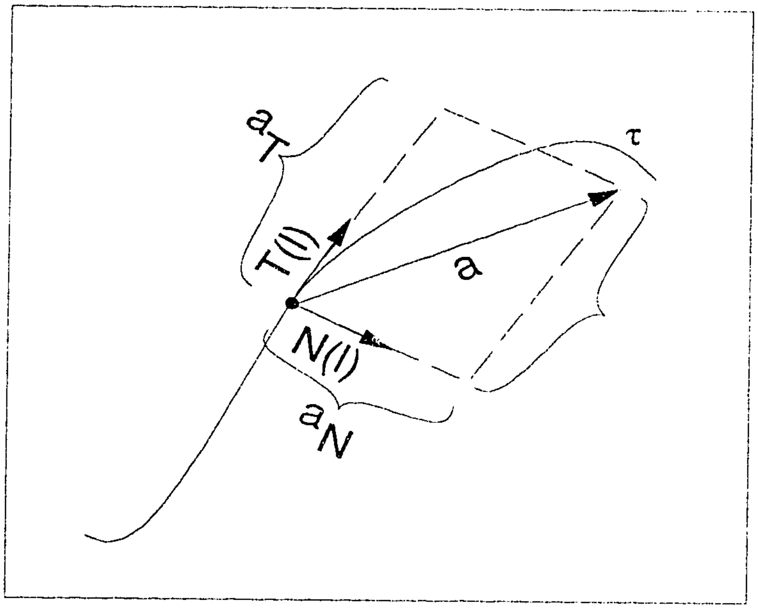

Figure 3.1: *A geometric interpretation of* $a = a_T T + a_N N$.

$$a = a_T T + a_N N$$

Since T and N are mutually orthogonal unit vectors

$$a^2 = a_T^2 + a_N^2$$

3.3 The Minimization Principle

A trajectory τ represented parametrically by $(x(t), y(t), \phi(t))$ on a time interval $I = [t_i, t_f]$ is called smooth if $x'(t)$ and $y'(t)$ are continuous on I and not simultaneously zero, except possibly at the endpoints. τ is called piecewise smooth if it is smooth on each subinterval (τ_j) of some partition of $I = [t_i^j, t_f^j]$. A trajectory τ can be represented as a sequence of sub-trajectories $\tau_1 \ldots \tau_n$ (n is the number of planning steps). Each sub-trajectory (say τ_j) is represented by a sequence of configurations $(x(t), y(t), \phi(t)), \forall t \in [t_i^j, t_f^j]$ joining the boundary-configurations $(x(t_i^j), y(t_i^j), \phi(t_i^j))$ and $(x(t_f^j), y(t_f^j), \varphi(t_f^j))$.

The problem described here is as follows: given two boundary-configurations (x_i, y_i, ϕ_i) and (x_f, y_f, ϕ_f) in a 2D plane, we want to determine and to compute the smoothest sub-trajectory τ_j joining these boundary-configurations.

The minimization function that finds the smoothest sub-trajectory (τ_j) is defined as:

$$\begin{aligned} \mathcal{J} &= \min_a \int_{t_i^j}^{t_f^j} a^2 dt \qquad (3.1) \\ &= \min_{s,\phi} \int_{t_i^j}^{t_f^j} [s'(t)]^2 + [s(t)\phi'(t)]^2 dt \end{aligned}$$

We now proceed to find the set of actual extremizing functions $s(t)$ and $\phi(t)$ of $\mathcal{J}$. Let

$$\mathcal{J} = \min_{s,\phi} \int \mathcal{F}(t, s, s', \phi, \phi') dt$$

subject to the boundary conditions: $\begin{cases} s(t_i^j) = s_0 \\ s(t_f^j) = s_1 \\ \phi(t_i^j) = \phi_0 \\ \phi(t_f^j) = \phi_1 \end{cases}$

From calculus of variation [7], the system of simultaneous Euler-Lagrange equations which must be satisfied by the functions $s(t)$ and $\phi(t)$ that render the above integral an extremum are:

$$\mathcal{F}_s - \frac{d}{dt}\mathcal{F}_{s'} = 0, \quad and \quad \mathcal{F}_\phi - \frac{d}{dt}\mathcal{F}_{\phi'} = 0$$

where $\mathcal{F}_s$, $\mathcal{F}_{s'}$, $\mathcal{F}_\phi$, and $\mathcal{F}_{\phi'}$ are the partial derivatives of $\mathcal{F}$ with respect to s, s', ϕ, and ϕ', respectively. Solving these Euler-Lagrange equations we obtain:

$$s\phi^2 - s'' = 0, \quad and \quad s\phi'' + 2\phi's' = 0$$

This set of nonlinear equations should determine the set of extremals: $s(t)$ and $\phi(t)$. However, it is not only hard to solve them, but also impossible to come up with a closed form solution. Therefore this minimization problem is not suitable for on-line (real-time) trajectory planning in this form. Instead, we divide the problem into two minimization subproblems, and look for a solution of the second problem:

$$\int_{t_i^j}^{t_f^j} [s(t)\phi'(t)]^2 dt$$

in the class of $s(t)$ (the solution of the first problem) where

$$\int_{t_i^j}^{t_f^j} [s'(t)]^2 dt$$

is minimized. In other words, the smoothest speed used for tangential acceleration will be considered in choosing the smoothest speed for performing the normal acceleration too. As a result, the minimization problem can be rewritten in the form

$$\begin{aligned} \mathcal{J} &\leq \mathcal{J}_1(s) + \mathcal{J}_2(\phi) \\ &= \min_s \int_{t_i^j}^{t_f^j} [s'(t)]^2 dt + \min_{s,\phi} \int_{t_i^j}^{t_f^j} [s(t)\phi'(t)]^2 dt \end{aligned} \tag{3.2}$$

Now the problem is to find the curve for which $\mathcal{J}$ is minimum. We first find the optimal curve for $\mathcal{J}_1$ and then use it to solve for the set of extremals of $\mathcal{J}_2$ as in the following subsections.

3.3.1 Tangential Acceleration

To find the extremal $(s(t))$ of $\mathcal{J}_1$, let

$$\mathcal{J}_1(s) = \min_s \int \mathcal{F}(t, s, s')dt = \min_s \int_{t_i^j}^{t_f^j} [s'(t)]^2 dt \tag{3.3}$$

subject to the boundary conditions: $s(t_i^j) = s_0$ and $s(t_f^j) = s_1$. The Euler-Lagrange equation is:

$$\mathcal{F}_s - \frac{d}{dt}\mathcal{F}_{s'} = 0$$

The set of extremals (solutions of the E-L equation) are:

$$s(t) = kt + c$$

where k and c are constants. Using a boundary condition, we can deduce the admissible arcs (extremals satisfying boundary conditions) to be:

$$\hat{s}(t) = kt + (s_i - kt_i^j) \tag{3.4}$$

Proposition 3.3.1 *All admissible solutions $(\hat{s}(t))$ are weakly local minimum.*

Proof:

$$\mathcal{F} \in C^3,$$

$$\mathcal{F}_{s's'} = 2 > 0.$$

Let $G(t,k)$, and $G_k(t,k) \in C^2$, such that:

- $G(t,k)$ is extremal for $\mathcal{J}(s)$, and $G(t_i^j, k) = s_0$.
- $G(t,\hat{k}) = \hat{s}(t)$, and $G(t_f^j, k) = s_1$.

Now to check whether or not t_i^j has conjugate points in $(t_i^j, t_f^j]$, we let

$$\begin{aligned} G_k(t,\hat{k}) &= t - t_i^j = 0 \\ \Rightarrow t &= t_i^j, \textit{ which } \notin (t_i^j, t_f^j] \end{aligned}$$

So by the Jacobi Theorem, $\hat{s}(t)$ is a weakly local minimum. □

Lemma 3.3.1 *$\hat{s}(t)$ is embeddable in a field of extremals $G(t,k)$.*

Proof:

Follows directly from Proposition 3.3.1, since for every admissible solution $\hat{s}(t)$ there exists $\hat{k}$ such that $\hat{s}(t) = G(t,\hat{k})$. □

Proposition 3.3.2 *All admissible solutions $(\hat{s}(t))$ are strongly local minimum.*

Proof:

Let $\tilde{s}$ be an admissible arc, then by the Weierstrass function [7]:

$$\begin{aligned} \mathcal{J}_1(\tilde{s}) - \mathcal{J}_1(\hat{s}) &= \int_{t_i^j}^{t_f^j} E(t, \tilde{s}, p, \tilde{s}')\, dt \\ &= \int_{t_i^j}^{t_f^j} (\mathcal{F}(t, \tilde{s}, \tilde{s}') - \mathcal{F}(t, \tilde{s}, p) - (\tilde{s}' - p(t, \tilde{s}'))\mathcal{F}_{s'}(t, \tilde{s}, p))\, dt \end{aligned}$$

where $p(t, \tilde{s})$ is the slope of the tangent of the extremal passing through $(t, \tilde{s}')$. Using Taylor theorem [7], we obtain

$$= \int_{t_i^j}^{t_f^j} \frac{1}{2}(\dot{s} - p(t, \tilde{s}'))^2 \mathcal{F}_{\tilde{s}'\tilde{s}'}(t, \tilde{s}, v(t))\, dt; \quad v(t) \in [\tilde{s}'(t), p(t, \tilde{s}')] \quad (3.5)$$

Since $\mathcal{F}_{s's'} = 2 > 0$, it follows from (3.5) and Lemma 3.3.1 that $\hat{s}$ is a global strong minimum. □

3.3.2 Normal Acceleration

Similarly, we minimize $\mathcal{J}_2$, which can be written in the form:

$$\mathcal{J}_2(\phi) = \min_{\phi} \int \mathcal{F}(t, \phi, \phi')dt = \min_{\phi} \int_{t_i^j}^{t_f^j} [s(t)\phi'(t)]^2 dt \qquad (3.6)$$

subject to the boundary conditions: $\phi(t_i^j) = \phi_0$, and $\phi(t_f^j) = \phi_1$. $s(t)$ is determined from (3.3). To compute k, we use the other boundary condition $(s(t_f^j) = s_1)$ to obtain a unique solution:

$$s(t) = \frac{s_1(t - t_i^j) + s_0(t_f^j - t)}{(t_f^j - t_i^j)} \qquad (3.7)$$

Substituting for $s(t)$ in (3.6), and then solving the Euler-Lagrange equation $(\mathcal{F}_\phi - \frac{d}{dt}\mathcal{F}_{\phi'} = 0)$ yields:

$$2[(s_1(t - t_i^j) + s_0(t_f^j - t))(s_1 - s_0)]\phi'(t) + [s_1(t - t_i^j) + s_0(t_f^j - t)]^2\phi''(t) = 0 \quad (3.8)$$

Applying reduction of order, separation of variables, and integration on (3.8), we obtain:

$$d\phi = \frac{c^{C_1}}{[s_1(t - t_i^j) + s_0(t_f^j - t)]^2}$$

Integrating again yields:

$$\phi(t) = \frac{-e^{C_1}}{(s_1 - s_0)(s_1(t - t_i^j) + s_0(t_f^j - t))} + C_2$$

where C_1 and C_2 are integration constants. Using one of the boundary conditions (say $\phi(t_i^j) = \phi_0$), the set of admissible arcs is:

$$\hat{\phi}(t) = \phi_0 + \frac{e^{C_1}(s_1(t - t_i^j) + s_0(t_f^j - t) - s_0(t_f^j - t_i^j))}{s_0(s_1 - s_0)(t_f^j - t_i^j)[(s_1(t - t_i^j) + s_0(t_f^j - t)]} \qquad (3.9)$$

where $(s_0 \neq s_1)$.

Proposition 3.3.3 *In the case $(s_0 = s_1)$,*

$$\hat{\phi}(t) = \frac{\phi_1(t - t_i^j) + \phi_0(t_f^j - t)}{t_f^j - t_i^j}$$

Proof:

Follows directly as for $\hat{s}(t)$. □

Proposition 3.3.4 *All admissible solutions $(\hat{\phi}(t))$ are weakly local minimum.*

Proof:

$$\mathcal{F} \in C^3,$$

$$\mathcal{F}_{\phi'\phi'} = 2s^2 > 0.$$

Let $G(t, C_1)$, $G_{C_1}(t, C_1) \in C^2$ such that:

- $G(t, C_1)$ is extremal for $\mathcal{J}(s)$, and $G(t_i^j, C_1) = \phi_0$.
- $G(t, \hat{C}_1) = \hat{\phi}(t)$, and $G(t_f^j, C_1) = \phi_1$.

Now to check whether or not t_i^j has conjugate points $(t_i^j, t_f^j]$, we let

$$\begin{aligned} G_{C_1}(t, \hat{C}_1) &= \frac{e^{C_1}(s_1(t - t_i^j) + s_0(t_f^j - t) - s_0(t_f^j - t_i^j))}{s_0(s_1 - s_0)(t_f^j - t_i^j)[(s_1(t - t_i^j) + s_0(t_f^j - t))]} = 0 \\ &= (s_1(t - t_i^j) + s_0(t_f^j - t) - s_0(t_f^j - t_i^j) = 0 \\ &\Rightarrow \; t = t_i^j, \; \textit{which} \; \notin (t_i^j, t_f^j] \end{aligned}$$

So by the Jacobi Theorem, $(\phi\hat{(t)})$ is a weakly local minimum. □

Lemma 3.3.2 *$\hat{\phi}(t)$ is embeddable in a field of extremals $G(t, C_1)$.*

Proof:

Follows directly from Proposition 3.3.4, since for every admissible arc $\hat{\phi}(t)$ there exists $\hat{C}_1$ such that $\hat{\phi}(t) = G(t, \hat{C}_1)$. □

Proposition 3.3.5 *All admissible solutions $(\hat{\phi}(t))$ are strongly local minimum.*

Proof:

Let $\tilde{\phi}$ be an admissible arc, then by the Weierstrass condition

$$\begin{aligned} \mathcal{J}_2(\tilde{\phi}) - \mathcal{J}_2(\hat{\phi}) &= \int_{t_i^j}^{t_f^j} E(t, \hat{\phi}, p, \tilde{\phi}')\, dt \\ &= \int_{t_i^j}^{t_f^j} (\mathcal{F}(t, \tilde{\phi}, \tilde{\phi}') - \mathcal{F}(t, \hat{\phi}, p) - (\tilde{\phi}' - p(t, \tilde{\phi}'))\mathcal{F}_{\phi'}(t, \hat{\phi}, p))\, dt \end{aligned}$$

Using Taylor theorem,

$$= \int_{t_i^j}^{t_f^j} \frac{1}{2}(\tilde{\phi}' - p(t,\tilde{\phi}'))^2 \mathcal{F}_{\phi'\phi'}(t,\tilde{\phi},v(t))\,dt; \quad v(t) \in [\tilde{\phi}'(t;\, p(t,\tilde{\phi}')] 3.1$$

Since $\mathcal{F}_{\phi'\phi'} = 2s^2 > 0$, it follows from (3.10) and Lemma 3.3.2 that $\hat{\phi}$ is a global strong minimum. □

Proposition 3.3.6 *Among the class of sub-trajectories defined on* $t \in [t_i^j, t_f^j]$, *the smoothest and safe sub-trajectory is the one that is computed by (3.4) and (3.9) provided that there are no obstacles.*

Proof:

Follows directly from Propositions 3.3.1 and 3.3.5 as both (3.4) and (3.9) are proven to be strongly local minimum which implies smoothness of the sub-trajectory. The trajectory is also safe since there are no obstacles in the environment. □

3.4 Obstacle Avoidance

The theory introduced in the previous section shows how to generate a smooth trajectory that minimizes the acceleration, but is not necessarily safe. In a local environment where partial knowledge of the environment is known, it is impossible to generate smooth trajectories unless the initial and goal positions are in the same local domain. However, it is possible to generate piecewise smooth trajectories (sub-trajectories are smooth locally). These sub-trajectories may not be safe because they do not take into account obstacles that may be present in the environment. To avoid this situation, we have either to consider obstacles as constraints in the minimization problem, or use other strategies in order to generate smooth and safe paths.

We assume that obstacles are static and circular in shape. We further assume that obstacles are detectable if they exist within a specific region (visibility field). If more than one obstacle is detected in a planning step, then we consider the one closest to the line segment that connects the boundary-configurations of this planning step. In the following subsection, we will explore obstacle avoidance techniques in more detail.

3.4.1 Minimizing Strategies

The smoothness problem can be theoretically described as a classical problem of calculus of variation to find a new trajectory that minimizes (3.1), and subject to a specific constraint — "obstacle avoidance constraint" — under the same boundary conditions. The choice of this constraint plays an important role in defining the exact nature of the minimization problem. We introduce two different constraints. The first one is to constrain the arc length of the smooth trajectory (to be computed) to be of length L (equality constraint). Whereas the second constraint is to always keep the distance between the smooth trajectory and the detected obstacle greater than a certain value (inequality constraint).

The first constraint can be expressed mathematically as follows:

$$\int_{t^j_i}^{t^j_f} \sqrt{(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2}\, dt \;=\; L \tag{3.11}$$

The parametric representation of (x, y) along a smooth trajectory (τ_j) is:

$$
\begin{aligned}
x(t) &= x_0 + s(t)t\cos(\phi(t)) \qquad (3.12)\\
y(t) &= y_0 + s(t)t\sin(\phi(t)) \qquad (3.13)
\end{aligned}
$$

where (x_0, y_0) is the start position on τ_j. Minimizing (3.1) under the above constraint is a typical isoperimetric problem of calculus of variation. For example, consider minimizing (3.3) subject to (3.11). The E-L equation is:

$$
\begin{aligned}
\mathcal{F}_\phi - \frac{d}{dt}\mathcal{F}_{\phi'} = 0, \quad &where\\
\mathcal{F}(t, \phi, \phi') &= [s(t)\phi'(t)]^2 + \lambda\sqrt{(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2}
\end{aligned}
$$

where λ is a Lagrange multiplier. The E-L equation yields the following nonlinear equation for determining the extremals $\phi(t)$:

$$
\begin{aligned}
\phi'' &= A(\phi')\phi' + B(\phi')\phi', \quad where\\
A(\phi') &= -\frac{2ss'(2C^{\frac{3}{2}} + \lambda C) - \lambda(2ss' + 2s'^2 + ss'\phi'^2)}{2s^2C'^{\frac{3}{2}} - \lambda s^2\phi'^2}\\
B(\phi') &= -\frac{\lambda s^2 C}{2s^2C^{\frac{3}{2}} - \lambda s^2\phi'^2}\\
C &= [(s + s't)^2 + (s\phi')^2]
\end{aligned}
$$

Solving this nonlinear equation is not only complicated but is a time consuming process. It also does not have an enclosed general solution form. Therefore, this method is not suitable for real-time applications (on-line planning). However, numerical methods — shooting methods — can be used to find a solution [12, 14].

Another way of avoiding obstacles is to constrain the distance between an obstacle and the smooth trajectory to be always greater than a given distance value (say δ). Mathematically this constraint can be expressed as:

$$\psi : (\tilde{r}^2 - [(x - x_c)^2 + (y - y_c)^2]) \leq 0 \; ; \quad \tilde{r} = r + \delta \tag{3.14}$$

where (x_c, y_c) is the center of the obstacle, and r is its radius. Minimizing (3.3) under (3.14) becomes a constrained optimization problem. From the local theory of constrained optimization, we want to evaluate the following function:

$$\int_{t_i^j}^{t} \mathcal{F}_\phi dt + \int_{t_i^j}^{t} \psi_\phi d\lambda - \mathcal{F}_{\phi'} = c \tag{3.15}$$

where $t \in [t_i^j, t_f^j]$, c is a constant, an[illegible]$^j, t_f^j]$ and is nondecreasing (normalized bounded vector). Substitu[illegible], (3.13), and (3.14) into (3.15), we obtain:

$$\int_{t_i^j}^{t} 2s(t)t[x_c \sin(\phi(t)) - y_c \cos(\phi(t))]d\lambda - 2s^2(t)\phi'(t) = c \tag{3.16}$$

The solution of (3.16) yields a nonlinear equation that has to be searched over λ. It is clear that this method suffers from the same problems as the previous technique. To avoid these problems, we introduce simpler techniques (in the next section) that can generate smooth and safe arcs to avoid obstacles.

3.4.2 Polynomial Fitting

There are several types of mathematical models that can be used for curve-fitting techniques. Most typical ones are polynomials because of their ease of handling, storing, and computation. In the past, for example, Kanayama and Hartman [11] used splines and cubic-spirals for curve fitting. In this paper, we explore two other techniques that can be used also as smoothness models.

If a smooth trajectory is not safe (i.e. the path generated collides with an obstacle), a new smooth path should be generated. Knowledge about times of collision between the obstacle and the smooth trajectory (see Appendix C), in addition to knowing a point (to be defined) on the obstacle is needed to generate smooth trajectories. We propose two techniques to resolve the problem of finding safe and smooth trajectories. The first one uses a cubic fitting technique and the other uses Bezier curves. Note that both techniques will be applied only when the generated smooth trajectory is not safe.

Cubic Fitting

This technique is easy to compute and to implement compared with the methods of the previous subsection. Because of the availability of information about the desired orientation at both initial and goal positions, and the first derivatives at these two points, we use a cubic curve. The parametric representation of such a curve is:

$$\phi(t) \;=\; At^3 + Bt^2 + Ct + D$$

where A, B, C, and D are real constants. To determine these constants, we compute the closest point ($\tilde{P} : (\tilde{t}, \tilde{\phi})$) (see Appendix D) on the detected obstacle from the line segment that connects the initial and goal positions (say P_0 and P_1, respectively). Given (t_0, ϕ_0) and (t_1, ϕ_1) at these points, the slopes of two line-constraints ($\overline{P_0\tilde{P}}$ and $\overline{P_1\tilde{P}}$) are easily determined (Figure 3.2). Thus we obtain four equations in four unknowns to solve for $\phi(t)$. The resulting unique solution should pass through P_0, P_1 and should satisfy the slopes of the line-constraints at P_0 and P_1. Since the curve is dependent

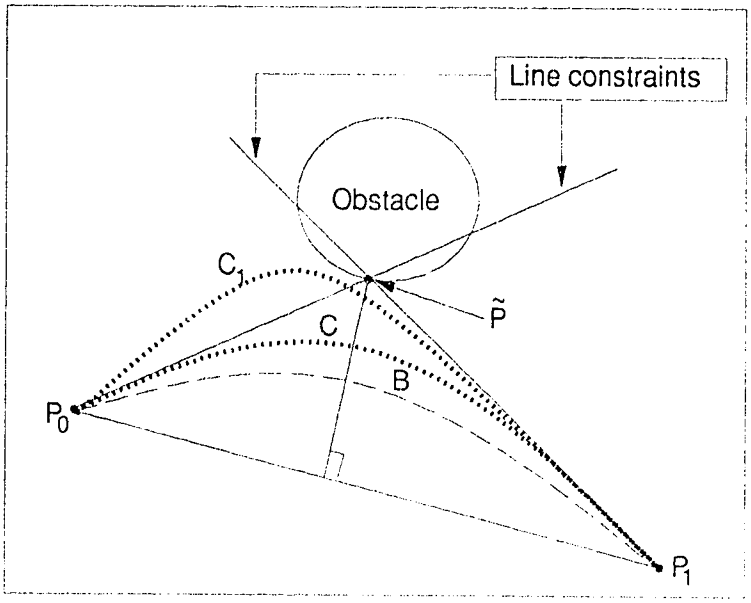

Figure 3.2: *Cubic and Bezier curves.*

on time, we can obtain more than one cubic polynomial that satisfies the above conditions by changing the arrival time ($\tilde{t}$). For example, C and C_1 in Figure 3.2 are different, but both are safe and smooth. Note that the cubic fitting curve may cross any of these line-constraints — for example, C_1 in Figure 3.2. In our particular application, this feature is of no importance as long as there is only one obstacle. Otherwise, the problem may be divided into two parts, where each one is treated as a separate problem. Another solution is to use other types of curve fitting techniques. In the following subsection, we present Bezier curves as an example.

Bezier Curve

In this method, the knowledge of three points is sufficient to determine a unique quadratic Bezier curve that is defined using the points: (t_0, ϕ_0), (t_1, ϕ_1), and $(\tilde{t}, \tilde{\phi})$. The parametric representation of the curve is:

$$\phi(t) = (1 - t^2)\phi_0 + 2t(1 - t)\tilde{\phi} + t^2\phi_1$$

where $(0 \leq t \leq 1)$. For example, B in Figure 3.2 represents a Bezier curve. For more details on this subject see [3, 2].

3.5 Experimental Results

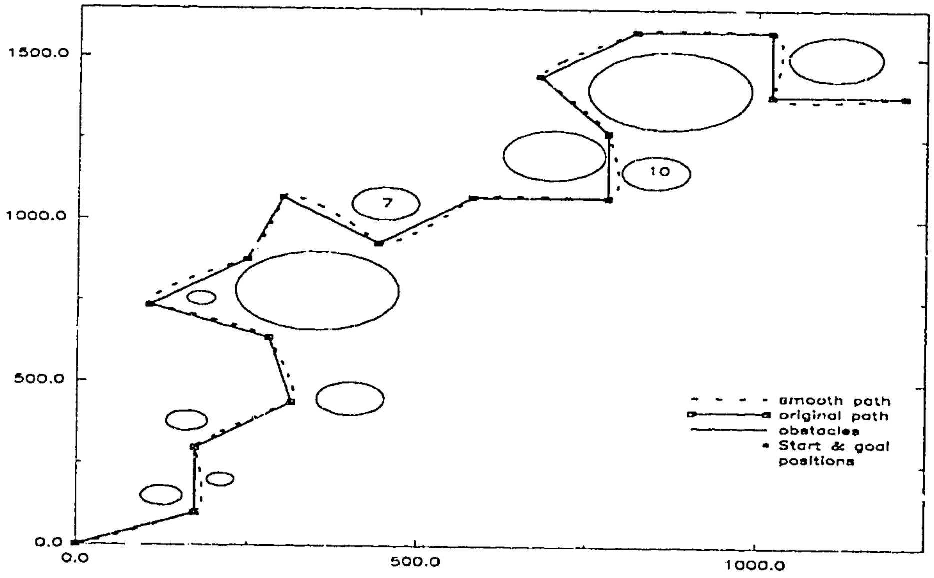

Figure 3.3: *Generating a piecewise smooth trajectory.*

The following simulation results show how a mobile robot may plan smooth and safe trajectories in a static local environment. Obstacles in the environment are represented by circles. We assume further that a mobile robot is mapped to a point and that obstacles are grown correspondingly.

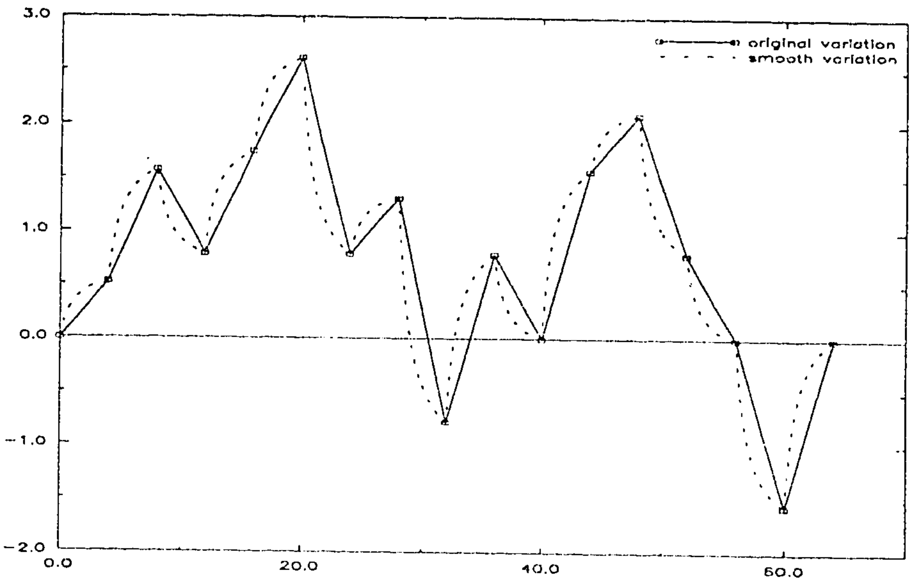

Figure 3.4: *The variation of ϕ along the time axis.*

Figure 3.3 describes a trajectory of 16 sub-trajectories that have been generated by a planning system in a cluttered environment with obstacles. The size of the navigation space is: 1250×1750 (units). The start and goal positions are: (0,0) and (1220,1380), respectively. Data obtained from the planning system are loaded into the GRTOOLS package in order to study parts of interest from the original trajectory, and to produce clearer figures that are easy to understand. The original path is described by a sequence of line segments used to connect boundary-configurations of each planning step. We allow the mobile robot to perform sharp turns (for example 90°). The piecewise smooth trajectory computed by (3.7), (3.9), (3.12), and (3.13) is also shown (the dotted curve). The corresponding change in the value of ϕ over time is demonstrated in Figure 3.4 for both trajectories.

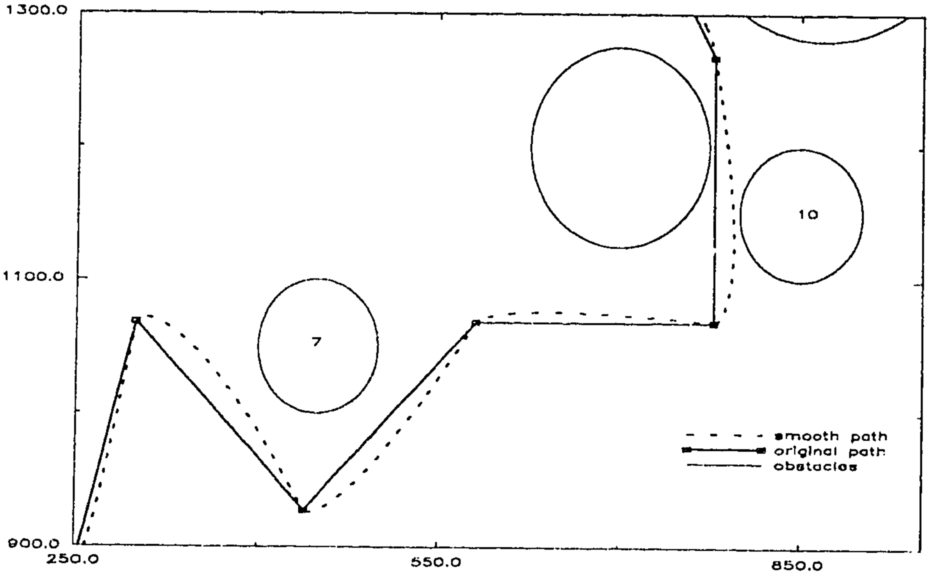

Figure 3.5: *Portion of the original trajectory of* Figure 3.3.

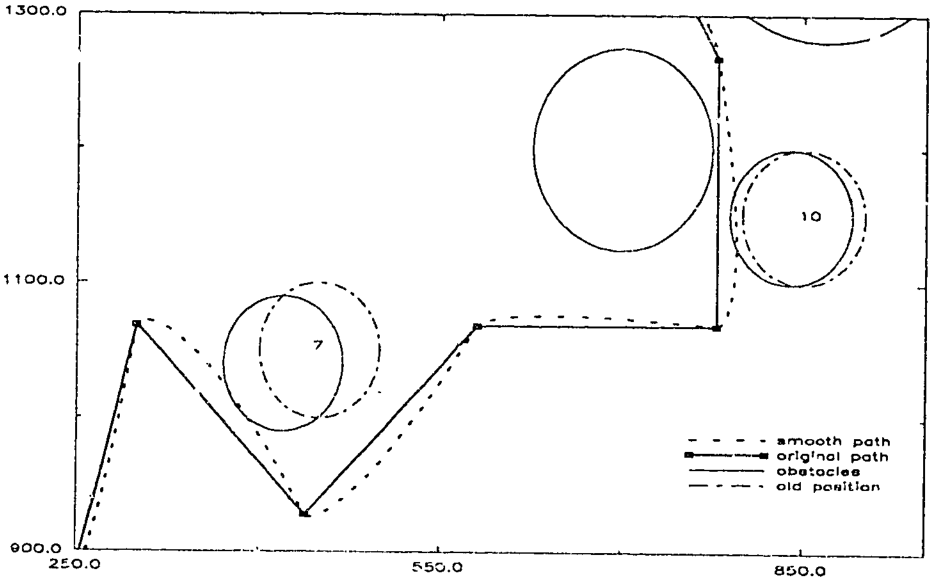

Figure 3.6: *Modifying obstacle positions in the environment.*

Figure 3.5 shows a portion of the trajectory of Figure 3.3. Note how close the smooth trajectory is to Obstacle 10. Let us change the positions of Obstacles 7 and 10 so that the smooth trajectory collides with them (Figure 3.6). Since the piecewise smooth trajectory is not applicable any more (leads to collisions), a new smooth sub-trajectory is generated instead. Using the cubic fitting and Bezier techniques (described before), two (or more) smooth sub-trajectories can be generated to replace the old one. Figure 3.7 shows an example of new smooth sub-trajectories that are generated to replace the sub-trajectory that collides with Obstacle 7. Note that the Bezier curve overlaps with one of the cubic fitting curves. Similarly, Figure 3.8 describes smooth sub-trajectories that can be used to resolve the collision problem with Obstacle 10. In Figures 3.7 and 3.8, two cubic fitting curves

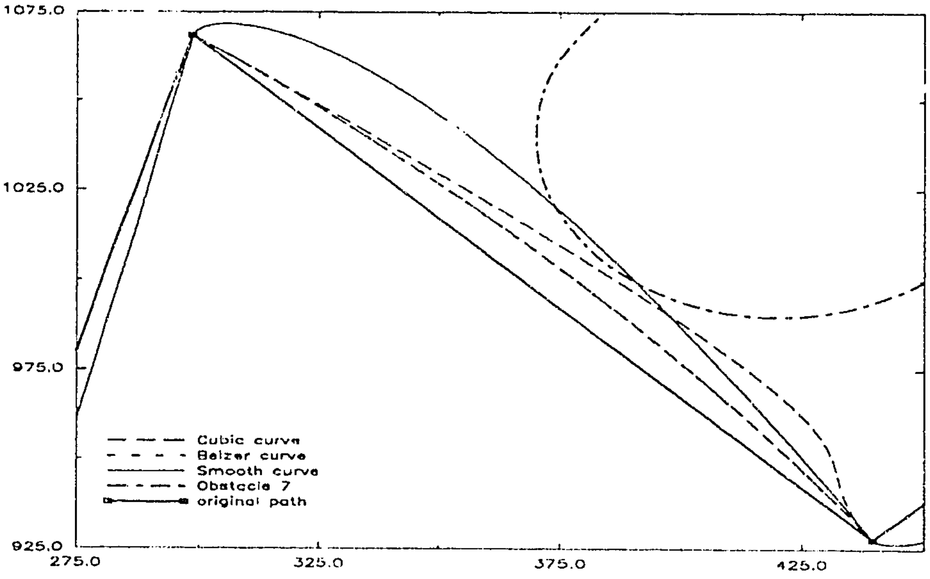

Figure 3.7: *Generating new sub-trajectories.*

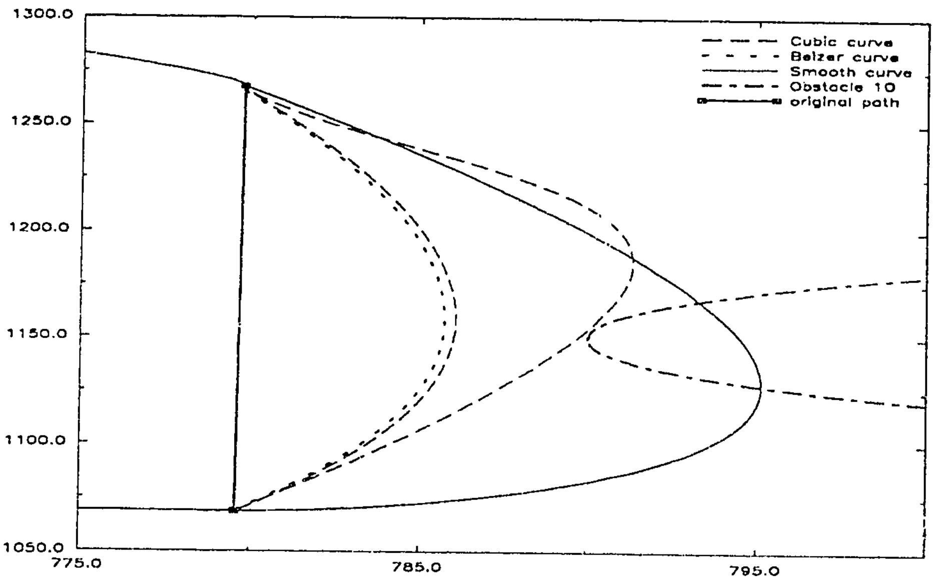

Figure 3.8: *Generating new sub-trajectories.*

are generated in each case. We may obtain even more sub-trajectories by changing the arrival time ($\tilde{t}$) at $\tilde{P}$. However, cubic curves that are close to the Bezier curves in shape are the ones generated by the planning system. They are smoother than the others because the speed achieved at $\tilde{t}$ is the desirable speed. But, in dynamic path planning, controlling the speed of a mobile robot at $\tilde{t}$ is very useful since time plays an important role in avoiding moving obstacles.

Figures 3.9 and 3.10 show two different experiments (obtained from the planning system) of generating smooth and safe piecewise trajectories. The initial orientation is indicated by the arrow placed at the start position. We use cubic fitting techniques only to generate safe sub-trajectories when it is necessary to do so.

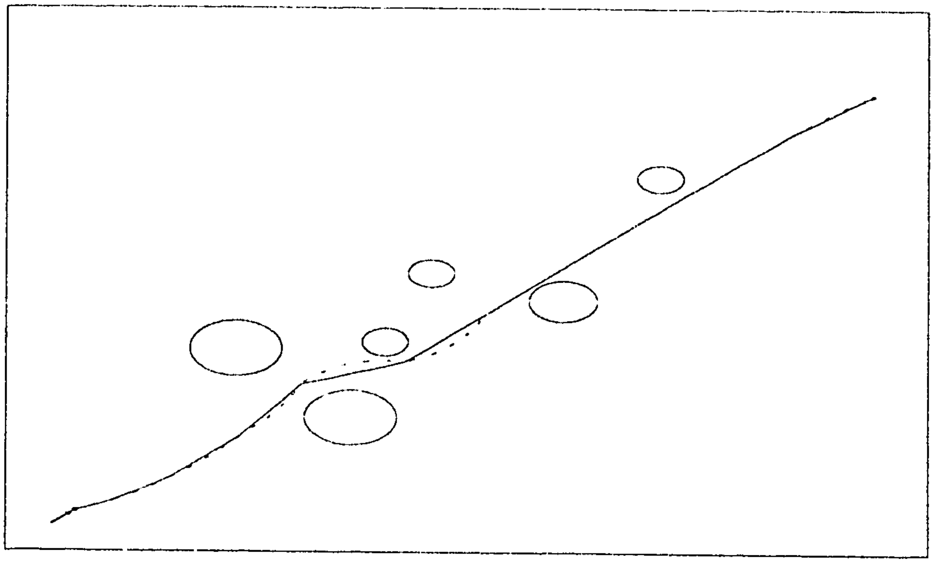

Figure 3.9: *A smooth and safe piecewise trajectory generated among obstacles.*

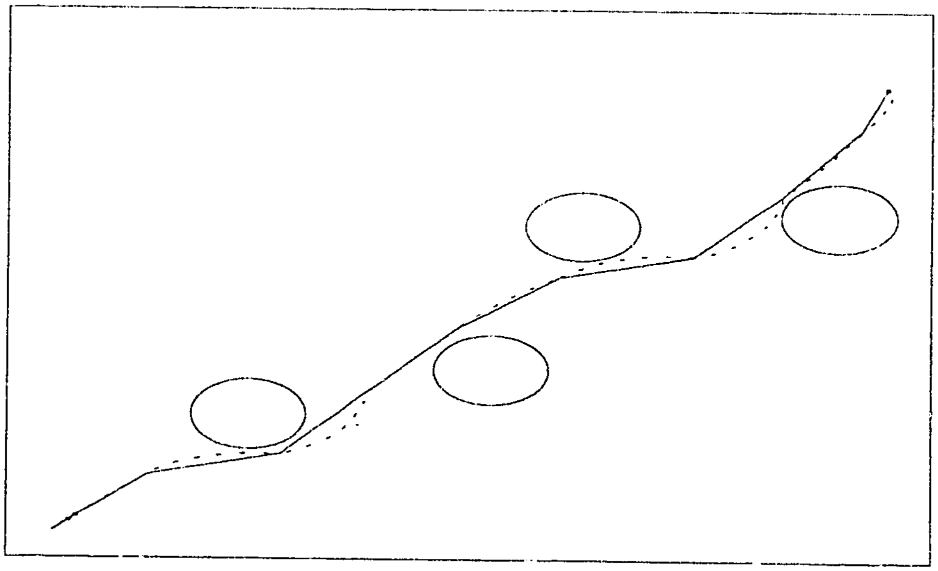

Figure 3.10: *Another smooth and safe piecewise trajectory generated in a different environment.*

3.6 Conclusion

We presented a new approach to generating piecewise smooth trajectories for mobile robots in a local environment. The algorithm minimizes the integral of the acceleration. It was shown how equality and inequality constraints can be taken into account in the minimization problem to avoid nearby obstacles. Since the solution was complex and therefore not suitable for real-time implementation, two other techniques were used, namely, cubic and Bezier curves. Each of these curves generated a smooth and safe trajectory once a collision was detected with the original smooth trajectory. Currently, we are interested in investigating the minimization problem in a dynamic environment where obstacles are moving. We also intend to study the possibility of obtaining a numerical procedure that solves the constrained minimization problems discussed in this paper.

Bibliography

[1] A. Bruckstein and A. Netravali. On minimal energy trajectories. *Computer Vision, Graphics and Image Processing*, 49:283–296, 1990.

[2] S. Chasen. *Geometric Principles and Procedures for Computing Graphic Applications.* Prentice-Hall, New Jersey, USA, 1978.

[3] C. DeBoor. *A Practical Guide to Splines.* Springer Verlag, New York, USA, 1978.

[4] D. Donald and P. Xavier. A provably good approximation algorithm for optimal-time trajectory planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 958–963. 1989.

[5] L. Dubins. On curves of minimal length with constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516. 1957.

[6] A. Elnagar and A. Basu. Heuristics for local path planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2481–2486, May 1992.

[7] L. Elsgolc. *Calculus of Variation.* Addison-Wesley, Massachusetts, USA, 1961.

[8] B. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9:441–460, 1983.

[9] P. Jacobs and C. Canny. Robust motion planning for mobile robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2–7, 1990.

[10] M. Kallay. Plane curves of minimal energy. *ACM Transactions on Mathematical Software*, 12:219–222, 1986.

[11] Y. Kanayama and B. Hartman. Smooth local path planning for autonomous vehicles. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1265–1270, 1989.

[12] H. Keller. *Numerical Methods for Two-point Boundary-Value Problems.* Blarsdell Publishing Co., Massachusetts, USA, 1968.

[13] J. Laumond. Finding collision-free smooth trajectories for a nonholonomic mobile robot. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1120–1123, 1987.

[14] S. Roberts and J. Shipman. *Two-point Boundary Value Problems: Shooting Methods.* American Elsevier Publishing Co., New York, USA, 1972.

Chapter 4

Safety Optimizing Strategies for Local Path Planning in Dynamic Environments[†]

4.1 Introduction

The problem of path planning for autonomous vehicles can be classified according to two criteria: global versus local, and exact versus heuristic. Global path planning requires a complete specification of the environment which may not be possible in many real life situations. For example, when driving on the road we have only local information about surrounding obstacles and vehicles and so we need planning strategies based on local informa-

[†] A. Basu and A. Elnagar. Safety Optimizing Strategies for Local Path Planning in Dynamic Environments. *Submitted for publication.*

tion.

The algorithm discussed in this work uses the concept of "safety-optimization" (introduced in [8]) to solve the dynamic motion planning problem in a local domain containing one (or more) moving obstacle. Moreover, we take into account the uncertainty in estimating the velocity of each moving obstacle. To solve this problem, it is necessary to define a continuous function of time which specifies the positions of the robot and the obstacles at each instant of time. We therefore add a time dimension to the configuration space in order to generate a configuration time-space. Consequently, we grow all obstacles with time and then try to find a free path among the grown obstacles. It is worth mentioning that the dynamic motion planning problem in 2-D is mapped to a static motion planning problem in 3-D. For a detailed discussion on the configuration time-space, please see [16].

There are three factors that control our strategy when we drive: how safe it is to drive at a certain speed given the present environment; how far we are from where we want to go; and how far we are from the closest moving obstacle. These factors are henceforth referred to as static safety, goal attraction, and dynamic safety, respectively. Static safety is a function of the speed. It may not necessarily be safer to drive more slowly since speeding up would be more difficult. Similarly, driving fast may make it more difficult to slow down when necessary. Goal attraction is a decreasing function of the distance to the goal which ensures that a generated path gets closer to the goal, if possible. The importance of the attraction towards the goal also affects safety since goal attraction, in effect, measures the urgency of reaching the goal. Dynamic safety is a function of time-to-collision and helps the robot choose safer positions in which it will be relatively far from moving obstacles.

We assume that the robot is equipped with a camera through which it can detect the positions of moving and static obstacles in a local visibility window. In general, sensory data are not robust; they do not provide an exact knowledge of the environment because of the presence of noise. It is therefore necessary for the planner to deal with uncertainty.

The next section briefly describes work in the area of dynamic motion planning. Section 4.3 describes the planning module in which a mathematical model for solving the problem with uncertainty is introduced. Section 4.4 cribes the navigation strategy. The planning algorithm is presented in tion 4.5. Experimental results are shown in Section 4.6.

Previous Work

Several authors have worked on the problem of path planning in a static, completely known environment (for example [3, 1, 17, 7, 15]). Some fundamental questions related to the complexity of various formulations of this problem are answered in [3, 21, 6, 5]. In reality, however, information about the environment is generally not completely known, with the exception of some specific industrial environments. Lumelsky [18] studied the problem of reaching a given goal position in a static but unknown environment.

Khatib [14] has indicated that global methods will limit the real time capabilities of robots in a cluttered environment, due to the time needed to perform the planning task. He and other researchers have studied the problem of navigating between initial and goal positions in a static and unknown environment.

Recently, some attempts have been made to solve the motion path planning problem in dynamic environments. Early works by Reif, Sharir, and Canny show some complexity results from specific cases of the general problem. For example, Reif and Sharir [19] show that motion planning in a 3D time-changing environment is PSPACE-hard when the robot's velocity is bounded, and NP-hard without such a bound. A more powerful result, is reported later by Canny and Reif [5]. They show that motion planning for a point robot in the plane with bounded velocity is NP-hard when the moving obstacles are complex polygons moving at a constant linear velocity without rotation.

Kant and Zucker propose the "path-velocity" decomposition technique [11]. A trajectory is assumed to be known from "off-planning". Next the velocity is changed along this trajectory so that collisions are avoided. Later, the work is extended to include uncertainties in the positions of obstacles and the robot [12]. Fujimura introduces the concept of accessibility graphs as a generalization of the visibility graph method, and shows how it can be used to solve for minimal-time paths in a dynamic environment [9]. An application of an unsteady diffusion equation model to path planning in a time-varying known world is described in [20]. Another navigation scheme used an "iterated forecast and planning" approach to solve this problem [23]. In this model, the planning and motion are iterated frequently according to a specific plan. All the preceding references, which have addressed the dynamic motion problem, assume complete knowledge of the environment.

The problem appears to be much more difficult when partial knowledge of the environment is available. Lamadrid [10] proposes a method that has to follow a predefined path with a given tolerance and reach the goal before a given time. Probabilistic models also have been used to solve the problem. One approach proposed by Kehtarnavaz [13] is to establish a collision zone around each moving obstacle and then treat such zones as stationary obstacles. Collision zones represent forbidden regions which are defined based on a high likelihood of collision. Other probabilistic models may be found in [4, 22].

The method presented in [2] contributed some of the ideas used in the approach described here. Now we will introduce some notations and define the problem.

4.3 Dynamic Planning under Uncertainty

4.3.1 Statement of the Problem

A mobile robot (MR) has to be continuously moving from p_s, the starting point, with an initial orientation ($\bar{\theta}$) to p_g, the goal, in the presence of other moving obstacles. Motion of each obstacle is estimated locally with uncertainty. At any point on its path, the robot knows only the current coordinates and those of the goal. Nothing is known about the obstacles outside a given visible region. There are also bounds on the speed and acceleration of the MR. Under these conditions, we want to find a local velocity such that a given function (to be defined) is optimized. Failure is reported if a path does not exist.

To address this problem we make the following assumption

- The obstacles to be considered are of elliptic shape. (In general for a polygonal obstacle, we consider the smallest covering ellipse.)
- Each obstacle moves with a constant velocity locally.
- The robot is equipped with sensors through which it estimates the position of moving obstacles with a given accuracy.
- The robot is mapped to a point, and obstacles are grown accordingly.

4.3.2 Growing Obstacles with Uncertainty

Given the velocity vector ($\vec{v} = (v_x, v_y)$) of an object (obstacle or robot) we estimate its speed and direction as follows:

$$\begin{aligned} s_e &= \sqrt{(v_x^2 + v_y^2)} \\ \theta_e &= \tan^{-1}(\frac{v_y}{v_x}) \end{aligned}$$

The motion of any object is estimated locally with uncertainty. That is, s_e and θ_e have errors ϵ_s and ϵ_θ respectively, and the actual speed and direction are in the range:

$$\begin{aligned} s_a &\in \begin{cases} (s_e - \epsilon_s, s_e + \epsilon_s) & \text{if } s_e > \epsilon_s \\ (0, s_e + \epsilon_s) & \text{otherwise} \end{cases} \\ \theta_a &\in (\theta_e - \epsilon_\theta, \theta_e + \epsilon_\theta) \end{aligned}$$

Consider a point in the configuration space-time located initially at (x, y). Then at time t the estimated speed of the point lies in the interval $[s_e - \epsilon_s, s_e + \epsilon_s]$ if $s_e > \epsilon_s$ or in the interval $[0, s_e + \epsilon_s]$ if $s_e \leq \epsilon_s$. Taking into account θ_e and ϵ_θ, the new position of the point at time t is:

$$(x + st\cos(\theta), y + st\sin(\theta)) \tag{4.1}$$

where $(\max\{0, s_e - \epsilon_s\} \leq s \leq s_e + \epsilon_s)$ and $(\theta_e - \epsilon_\theta \leq \theta \leq \theta_e + \epsilon_\theta)$.

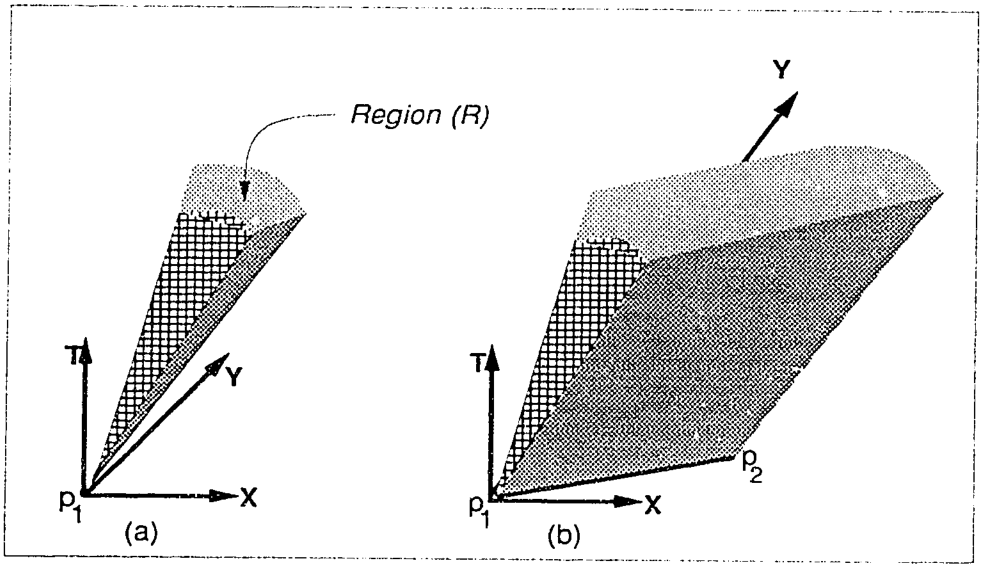

Figure 4.1: *(a) and (b) show examples of growing a point and a line respectively in space-time with uncertainty.*

Figure 4.1(a) shows an example of growing a point (p_1) located initially at (0,0). The shaded region (R) represents all possible positions of point p_1 at time t. In the case of a line segment, we determine the shape of the grown obstacle region by growing its end points (an example appears in Figure 4.1(b)). This procedure can be applied to determine the grown obstacle region for a polygon. Modeling the resulting obstacle region (of a polygon) mathematically is not easy. Therefore each polygon (i.e. obstacle) is enclosed in a minimum covering ellipse.

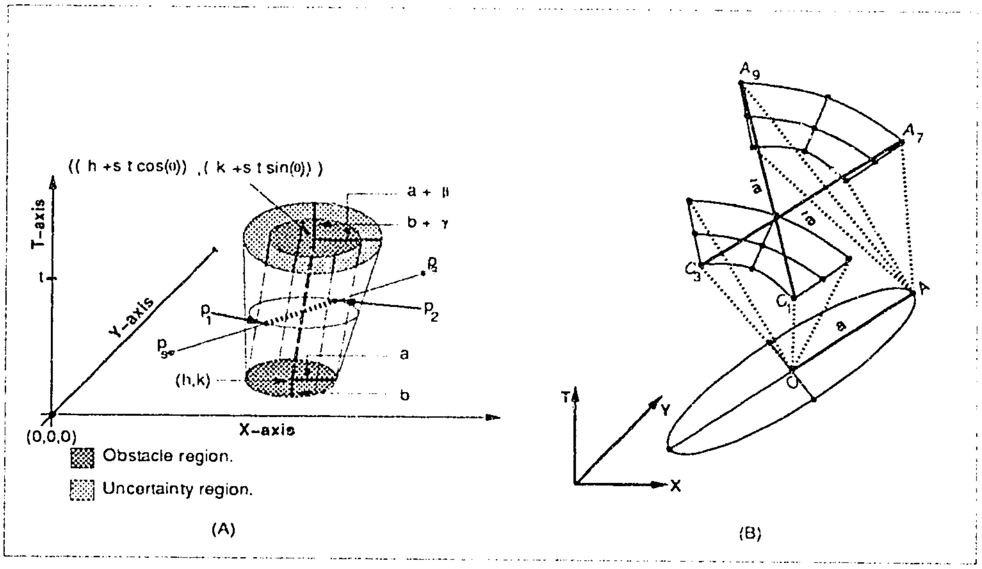

Figure 4.2: *(A) growing an ellipse in space-time with uncertainty. (B)* $\beta = \bar{a} - a$.

Let us consider the general form of the equation of an ellipse centered at (h, k) and with axes rotated by an angle ϕ:

$$\frac{(\tilde{x} - (h\cos(\phi) - k\sin(\phi)))^2}{a^2} + \frac{(\tilde{y} - (h\sin(\phi) + k\cos(\phi)))^2}{b^2} = 1$$

where,

$$\begin{aligned}\tilde{x} &= x\cos(\phi) - y\sin(\phi)\\ \tilde{y} &= x\sin(\phi) + y\cos(\phi)\end{aligned}$$

The major axis, is along the x-axis $(a > b)$. The foci lie on the major axis c units from the center (h,k) with $c^2 = a^2 - b^2$. Note that the major axis is along the y-axis if $a < b$. Equation (4.1) describes an ellipse (obstacle region) in the xy-plane at t=0. We want to compute the grown obstacle region at time t, provided that the obstacle is moving with speed s_ϵ and in direction θ_e. To define the new equation, we make the following assumptions:

- The center of the ellipse is moving with speed (s_e) and direction (θ_ϵ) along the time axis.
- At each instant of time, the resulting obstacle region is approximated by an ellipse that includes the uncertainty in both speed and direction, by growing the major and minor axes only (Figure 4.2(A)).

The resulting equation, which describes the grown obstacle region, is:

$$\frac{(\tilde{x} - (\tilde{h}\cos(\varphi) - \tilde{k}\sin(\phi)))^2}{\tilde{a}^2} + \frac{(\tilde{y} - (\tilde{h}\sin(\phi) + \tilde{k}\cos(\phi)))^2}{\tilde{b}^2} = 1 \quad (4.2)$$

where,

$$\begin{aligned}
\tilde{h} &= h + s_\epsilon t\cos(\theta_\epsilon) && (4.3)\\
\tilde{k} &= k + s_\epsilon t\sin(\theta_e) && (4.4)\\
\tilde{a} &= a + \beta\\
\tilde{b} &= b + \gamma\\
\beta &= 2\sqrt{t^2 s^2 + t^2\cos^2(\epsilon_\theta)(\epsilon_s^2 - s^2) + C_1}\\
\gamma &= 2\sqrt{t^2 s^2 + t^2\cos^2(\epsilon_\theta)(\epsilon_s^2 - s^2) + C_2}\\
C_1 &= t\,a((\epsilon_s\cos(\theta)cos(\epsilon_\theta) + s\sin(\theta)\sin(\epsilon_\theta))\\
C_2 &= t\,b((\epsilon_s\cos(\theta)cos(\epsilon_\theta) + s\sin(\theta)\sin(\epsilon_\theta))
\end{aligned}$$

Equation (4.2) is an elliptic cone which defines all ellipses along the line segment connecting the center of the ellipse at time t with the center of the ellipse at time t=0 (Figure 4.2(A)). For each ellipse, the major and minor axes are redefined in a way to include the uncertainties in speed and direction ($\epsilon_s, \epsilon_\theta$, respectively). To illustrate this, consider Figure 4.2(B). The maximum distance that can be obtained between the center C and a vertex A is when C moves to C_1 or C_3 and A moves to A_9 or A_7 respectively. The new major and minor axes lengths computed at that instant of time will be $(a + \beta)$ for the major axis and $(b + \gamma)$ for the minor one. β and γ define the value of the maximum error that can be obtained due to the uncertainties ϵ_s and ϵ_θ. Formally, $\beta = (\tilde{a} - a)$ and $\gamma = (\tilde{b} - b)$.

4.3.3 Obtaining Safe Strategies under Uncertainty

Once the grown obstacles are obtained, the next step is determining which area is reachable without changing velocity in a given interval. Consider Figure 4.2(A) where p_s and p_g are the start and goal positions. We want to compute at what time(s) the line segment $\overline{p_s p_g}$ intersects the obstacle volume, if there is an intersection.

Let $p_s = (x_1, y_1, t_1)$ and $p_g = (x_2, y_2, t_2)$. The direction vector of $\overline{p_s p_g}$ is $(a_1 = x_2 - x_1, a_2 = y_2 - y_1, a_3 = t_2 - t_1)$. The parametric equations for $\overline{p_s p_g}$ are:

$$x = x_1 + a_1 t \qquad (4.5)$$

$$y = y_1 + a_2 t \qquad (4.6)$$

To compute the times of intersection, we substitute (4.3) and (4.4) in (4.2). Expanding and simplifying the resulting equation yields a polynomial of degree 4. The solution set (T) of this polynomial may include some imaginary roots. If all roots are imaginary then there is no point of intersection. In the case where T includes two imaginary roots, the line segment will intersect the obstacle at either one or two points, depending on whether or not the real roots are repeated. Similarly, if all the roots in T are real, either the line falls on the boundary of the conic volume or there are repeated roots. The next proposition derives the points of intersection of a line with a conic volume.

Proposition 4.3.1 *In the case where the grown obstacle is a circular cone, a line segment (not lying on the cone) intersects it in, at most, two points.*

Proof:

Equating the major and minor axes in Equation (4.2) $(\tilde{a} = \tilde{b})$, and not considering the rotation of axes yields a circular cone:

$$\frac{(x-\tilde{h})^2}{\tilde{a}^2} + \frac{(y-\tilde{k})^2}{\tilde{a}^2} = 1 \tag{4.7}$$

To compute times of intersection with a given line segment, we substitute (4.3,4.4,4.5, and 4.6) into (4.7) obtaining:

$$\frac{((x_1 + a_1 t) - (h + s_e t\cos(\theta_e)))^2}{(a+t\beta)^2} + \frac{((y_1 + a_2 t) - (k + s_e t\cos(\theta_\epsilon)))^2}{(a+t\beta)^2} = 1$$

which simplifies to:

$$At^2 + Bt + C = 0 \tag{4.8}$$

where,

$$A = a_1^2 + a_2^2 + s_e^2 - \beta^2 - 2\,s_e(a_1\cos(\theta_e) + a_2\sin(\theta_e))$$

$$\begin{aligned} B &= 2s_e(\cos(\theta_e)(x_1 + h) - \sin(\theta_e)(y_1 + k)) + D \\ C &= x_1^2 + y_1^2 + h^2 + k^2 - a^2 - 2(x_1 h + y_1 k) \\ D &= 2a_1(x_1 - h) + 2a_2(y_1 - k) - 2\,a\beta \end{aligned}$$

Solving the quadratic Equation (4.8), for the times of intersection, produces:

$$T \;:=\; \{(t_1, t_2) = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}\} \tag{4.9}$$

Now if t_1 and t_2 are real, we substitute (4.9) in (4.3) and (4.4) to compute the points of intersection:

$$\{(x_1 + a_1 t_1, y_1 + a_2 t_1)\,,\,(x_1 + a_1 t_2, y_1 + a_2 t_2)\} \quad \square$$

4.4 Navigation Strategy

To obtain a safe strategy for path planning with uncertainty, there are three factors which must be taken into account while computing the optimum local velocity. They are:

- The static safety at the destination point where the robot can maintain its acceleration bounds. This is given by the static safety function ($f(s)$).

- How much closer the robot is to the goal, which is modeled by the goal attraction function ($g(d)$).

- Clearance from obstacles (or dynamic safety) as modeled by $h(t_c)$. This is a function of the time to collision.

4.4.1 Preliminaries[1]

To introduce the motion planning problem formally, some basic definitions are needed. Consider W to be the workspace of all possible positions in which a mobile robot (MR) may be placed. W is represented as a subset of the Euclidean space $\Re^2$. F_W is a fixed frame of reference embedded in W. Similarly F_{MR} is the Cartesian frame of MR. At any time, we shall consider the MR (circular object) as a point in W which can be represented by its coordinates with respect to F_W, and an orientation (θ) of F_{MR} with respect to F_W. The initial and goal positions in W are denoted by p_s and p_g respectively. The MR position in W is defined by $\hat{p}$. The function $d : W \times W \rightarrow \mathcal{R}$ denotes the Euclidean distance. The time needed for the MR to plan its trajectory is denoted by $T = [0, t_f]$. The symbol t_f denotes the final (arrival) time. A configuration P is a specification of the position (x, y) and the orientation of F_{MR} with respect to F_W at time t_i ($t_i \in [0, t_f]$). The set of obstacles (elliptic rigid objects) distributed in W are represented by $O = O_0, O_1, \ldots, O_{n-1}$. Each O_i represents a subset of W at a specific instant of time. The space occupied by these (n) obstacles at time t is given by:

$$O(t) = \bigcup_{i=0}^{n-1} O_i(t), \ \forall t \in T$$

It is assumed that each moving obstacle has a known trajectory τ_i. At any instant of time t, τ_i defines the center of obstacle i.

Definition 4.4.1 *Local Search Window (LSW) specifies the visibility range*

[1] This section is similar to Section 2.3 but definitions are revised to suite the dynamic case.

of the MR, which is defined as a circular sector with an area $\frac{1}{2}\phi r^2$, where ϕ ($\phi \in [0, 2\pi)$) is the visibility angle, and r is the radius of the visibility range. LSW is represented by a set of equally spaced bounded rays, $\{R_j\}_{j \in J}$. Each ray is defined by a set of points. The number of points $= \lfloor r/\epsilon \rfloor$, where ϵ is the discretization size. $\Delta\phi$ is the angle between any two successive rays. Thus the actual set of grid points in the i^{th} LSW is:

$$LSW^i_{gp} \;=\; \{R_j\}_{j \in J},$$

where $J = \{0, 1, \ldots, m-1\}$, and $m = \lfloor \frac{\phi}{\Delta\phi} \rfloor$.

Definition 4.4.2 *The set of grid points that are free in the i^{th} LSW at time t is:*

$$LSW^i_{fgp}(t) \;=\; \{LSW^i_{gp} \setminus (\bigcup_{j=0}^{n-1}(O_j(t) \bigcap (\bigcup_{l=0}^{m-1} R_l)))\}$$

The set of grid points that are visible to the MR in the i^{th} LSW is:

$$\begin{aligned} LSW^i_{vgp}(t) &= \{\bigcup_{l=0}^{m-1} R_l^{min}\}, \;\; where \\ R_l^{min} &= \begin{cases} R_l & if\ (R_l \bigcap_{j=0}^{n-1}(O_j(t))) = \{\ \} \\ R_j & otherwise \end{cases} \end{aligned}$$

The number of grid points in $R_l = \lfloor r/\epsilon \rfloor$, whereas in R_j

$$= \lfloor \frac{min\ d(p, \hat{p})}{\epsilon} \rfloor \;\; \forall p \in (R_l \bigcap_{j=0}^{n-1}(O_j(t)))$$

Definition 4.4.3 *The Reachability Condition (REACH) is a test performed at each grid point (p) to verify if it is reachable from the current position of* *the MR, given constraints on the acceleration and the speed.*

$$REACH^i(p) \;=\; \begin{cases} \{p\} & \text{if } [(a_T(p) \leq A_T) \wedge (a_N(p) \leq A_N) \wedge (s_f(p) \leq S)] \\ \{\,\} & \text{otherwise} \end{cases}$$

Consequently, the set of reachable grid points for the MR in the i^{th} LSW at time t is:

$$LSW^i_{rgp}(t) \;=\; \{\{p|p \in LSW^i_{vgp}(t)\} \bigcap REACH^i(p)\}$$

where p is a grid point.

Definition 4.4.4 *A free trajectory of the MR from p_s to p_g is a continuous mapping*

$$\tau \;:\; [0, t_f] \rightarrow W^{local}_{free}.$$

where $\tau(0) = p_s$ and $\tau(t_f) = p_g$, and

$$W^{local}_{free} \;=\; \{LSW_{rgp}(t)\}_{i \in \{0,1,\ldots,k-1\}}, \quad \forall t \in T$$

k is the number of planning steps along the trajectory.

Proposition 4.4.1 *If $p_i \in LSW^j_{rgp}$ then $p_{\frac{\phi}{\Delta\phi}-i} \in LSW^j_{rgp}$ provided that $p_{\frac{\phi}{\Delta\phi}-i} \in LSW^j_{vgp}$.*

Proof:

Since p_i and $p_{\frac{\phi}{\Delta\phi}-i}$ are two points symmetric around the radius (r). Then

$d(\hat{p}, p_i) \quad = d(\hat{p}, p_{\frac{\phi}{\Delta\phi}-i})$, and

$$|\theta_{p_i} - \theta| \quad = |\theta_{p_{\frac{\phi}{\Delta\phi}-i}} - \theta|.$$

$\Longrightarrow$ Both points have the same speed and acceleration.

So. $(p_i \in LSW^j_{rgp}) \wedge (p_{\frac{\phi}{\Delta\phi}-i} \in LSW^j_{rgp}) \Longrightarrow (p_{\frac{\phi}{\Delta\phi}-i} \in LSW^j_{rgp})$ □

Proposition 4.4.2 *If* $\bigcap_{i=0}^{n-1} O_i(t) = \{\ \}$ *then there is a free trajectory from* p_s *to* p_g *given that reachable points always exist.*

Proof:

Follows directly from definitions above. □

4.4.2 Static Safety

To compute safety we consider all velocities in a given range to be equally likely at a future instant of time. Then, given a local velocity, and the tangential and normal acceleration bounds (A_T and A_N respectively), we compute the probability that the vehicle can maintain the kineodynamic constraints. Thus the safety corresponding to a speed s is:

$$f(s) = Pr[a_T(t) \leq A_T \text{ and } a_N(t) \leq A_N|s],\ 0 \leq t \leq t_0$$

where $a_T(t)$ and $a_N(t)$ are the computed tangential and normal acceleration at time t, and t_0 is the time interval.

4.4.3 Goal Attraction

The second criterion that we consider for planning a path is: "How important is it to reach the goal quickly?". If there is no urgency in reaching the goal, then all we need to do is to be safe at present. As the importance of reaching the destination is increased the relative importance of being safe at present is decreased. Let d define the Euclidean distance between the new selected position and the goal. Then the goal attraction function is defined as:

$$g(d) = (\frac{K}{K+d})^C, \quad K \text{ and } C \text{ are} > 0.$$

4.4.4 Dynamic Safety

Let T_R be the solution set of all real roots of the polynomial in Equation(4.2). The function $(\min(T_R))$ outputs the minimum root in T_R. We define t_c (time-to-collision) as the difference between $\min(T_R)$ and t_0 (time to plan a step of the path), i.e. $t_c = \min(T_R) - t_0$. Let t_s represent a safety time margin. Given a line segment that intersects the obstacle region at two points (say p_1 and p_2) at two different times (t_1 and t_2 respectively) and the destination point p_d, we define $\tilde{d} = |p_s p_d|$, $d_1 = |p_s p_1|$, and $d_2 = |p_s p_2|$. Henceforth, the function $h(t_c)$ is defined as follows:

$$h(t_c) = \begin{cases} 0 & \text{if } (t_c < 0) \\ 1 & \text{if } T_R = \{\ \} \vee (t_c \geq t_s) \\ e^{(\frac{-\lambda}{t_c})} & \text{if } (0 < t_c < t_s) \wedge ((\tilde{d} < d_1) \vee (\tilde{d} > d_2)) \end{cases}$$

where λ is a positive constant. Figure 4.3 shows how $h(t_c)$ — dynamic safety — varies with λ, considering $t_s = 4.0$.

Suppose we have a MR at location p_s (Figure 4.2(A)) and want it to move to p_g while an obstacle is passing through. The function $h(t_c)$ allows the robot to head towards its destination only if it is safe to do so. For example, if the

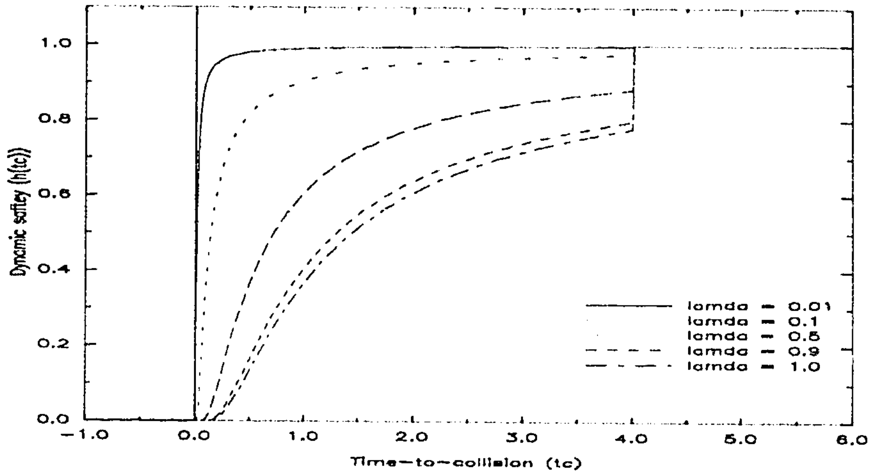

Figure 4.3: *The effect of λ on $h(t_c)$ (considering $t_s = 4.0$).*

robot does not collide with the grown obstacle (i.e. $T_R = \{\ \}$) then the value of $h(t_c)$ will be maximum (i.e., 1). On the other hand, if the time of collision is less than the time used for the robot to plan its path step, then the value of $h(t_c)$ will be minimum (i.e., 0). Otherwise the value of $h(t_c)$ will depend on how far from the obstacle the robot is. The closer the robot gets to the obstacle, the less the value of $h(t_c)$ will be. As a result, the overall safety function for navigating between two points (optimality criterion) is defined as:

$$N(s, d, t_c) = f(s)\, g(d)\, h(t_c) \qquad (4.10)$$

Thus if we maximize $N(s, d, t_c)$, the strategy would be to try to maximize static as wel' as dynamic safety and, at the same time, reduce the distance to the goal.

Proposition 4.4.3 *The local velocity selected always takes the robot closer to the goal, if any reachable points exist, provided:*

(a) *The safety function f has a positive value at 0.*

(b) $t_c > t_s$.

(c) *C is large enough.*

Proof:

Suppose a robot is at a distance d from the goal. There are many reachable points that the robot can move to. They can be classified into two sets. The first one contains all reachable points that take the robot closer to the goal. The second set contains all reachable points that take the robot further away from the goal. In order to move the robot closer to the goal, the value of the overall safety for any point in the first set should be higher than the corresponding value for any point in the second set. In other words, the minimum overall safety for the first set should be greater than the maximum overall safety for the second.

- The minimum in the first set is achieved for a point which is at a distance $(d - \epsilon)$ from the goal, ϵ is the discretization size with static safety s_0 and dynamic safety $h(t_{c_1})$.

- The maximum in the second set is achieved for a point which is at a distance d from the goal with static safety s_{max} and dynamic safety $h(t_{c_2})$.

It is important to note that $\epsilon < d$ and $(d - \epsilon) < d$. Also, $s_0 < s_{max}$. Overall safety for the first point is:

$$s_0 g(d - \epsilon) h(t_{c_1}) \quad = \quad s_0 (\frac{K}{K + (d - \epsilon)})^C$$

and, overall safety for the second point is:

$$s_{max} g(d) h(t_{c_2}) \quad = \quad s_{max} (\frac{K}{K + d})^C$$

In order that the robot chooses the first position instead of the second one, C has to be large enough so that the following inequality is satisfied:

$$\begin{aligned} s_0 (\frac{K}{K + (d - \epsilon)})^C &> s_{max} (\frac{K}{K + d})^C \\ \Rightarrow (\frac{K + d}{K + (d - \epsilon)})^C &> \frac{s_{max}}{s_0} \\ \Rightarrow C log(\frac{K + d}{K + (d - \epsilon)}) &> log(\frac{s_{max}}{s_0}) \\ \Rightarrow C &> \frac{log(\frac{s_{max}}{s_0})}{log(\frac{K+d}{K+(d-\epsilon)})} \quad \Box \end{aligned}$$

In the case $s_0 = s_{max}$, the first point will be always chosen, since the value of the goal attraction function of the first point is higher than the corresponding value of the second point.

Consider that the MR is moving with a fixed direction locally while a moving obstacle is approaching it. In order to avoid collision, the MR should either speed up and pass in front of this moving obstacle, or slow down and let the obstacle pass first (more details are given in the experimental section). This idea of controlling the speed along a planned path to avoid collisions is the main concept behind the velocity-decomposition technique

[11]. Using our approach the MR (in addition to the above solution) can also plan paths to other intermediate positions. Moreover, it takes into account uncertainty in obstacles' positions. The following proposition shows how the MR can control its speed when its direction of motion is locally fixed to avoid obstacles.

Proposition 4.4.4 *Consider a mobile robot (MR) moving in a fixed direction locally while a moving obstacle is approaching it. The MR avoids collision with this obstacle by adjusting its speed (s) so that s does not lie in $[s_{min}^{collision}, s_{max}^{collision}]$.*

Proof:

Let $\hat{p} = (\hat{x}, \hat{y}, \hat{t})$ represents the current location of the MR at time $\hat{t}$. Since the direction of motion of the MR is fixed, all possible destination points are constrained to lie on a one dimensional locus of points that is perpendicular to the xy-plane. Given the motion equation of the grown obstacle (detected in LSW), the locus intersects this obstacle in, at most, two points. Let $p_{min}^{collision} = (x_{min}, y_{min}, t_{min})$ and $p_{max}^{collision} = (x_{max}, y_{max}, t_{max})$ define points of intersection ($t_{min} \leq t_{max}$). All points that lie on the locus between $p_{min}^{collision} = (x_{min}, y_{min}, t_{min})$ and $p_{max}^{collision} = (x_{max}, y_{max}, t_{max})$ are collision points. The MR avoids moving to a collision point by controlling its speed (s).

The direction vectors of the two lines tangent to the grown obstacle ($\overline{\hat{p}p_{min}^{collision}}$and $\overline{\hat{p}p_{max}^{collision}}$) are respectively:

$$(a_{11} = (x_{min} - \hat{x}), a_{12} = (y_{min} - \hat{y}), a_{13} = (t_{min} - \hat{t})),$$

$$(a_{21} = (x_{max} - \hat{x}), a_{22} = (y_{max} - \hat{y}), a_{23} = (t_{max} - \hat{t}))$$

Consequently, the points of intersection (in parametric form) are:

$$(\hat{x} + a_{11}t_{min}, \hat{y} + a_{12}t_{min}) \quad and \quad (\hat{x} + a_{21}t_{max}, \hat{y} + a_{22}t_{max})$$

Thus the speeds at points of intersection are:

$$\begin{aligned} s_{min}^{collision} &= \frac{|\hat{p}p_{min}^{collision}|}{t_0} = t_{min}\sqrt{a_{11}^2 + a_{12}^2} \\ s_{max}^{collision} &= \frac{|\hat{p}p_{max}^{collision}|}{t_0} = t_{max}\sqrt{a_{21}^2 + a_{22}^2} \end{aligned}$$

To determine whether or not the speed (s) of the MR leads to a collision, we check if it lies in the interval $[s_{min}^{collision}, s_{max}^{collision}]$. To avoid collision, s has to be modified by either slowing down ($s < s_{min}^{collision}$) or speeding up ($s > s_{max}^{collision}$). For more illustration, consult *Figure 4.2*(A). □

In the case where there is no speed (s) satisfying the tangential acceleration bound such that $s \in [s_{min}^{collision}, s_{max}^{collision}]$, a collision is inevitable since there is no safe speed. This situation occurs when the visibility field of the MR is narrow (r and ϕ are very small in LSW), and an obstacle is moving with high speed towards the MR. Note that in our approach, the MR may avoid this collision by planning for an intermediate position (direction of motion is not fixed). In general most local dynamic planning approaches fail to deal with this situation.

Corollary 4.4.1 *Given a fixed local direction of the MR's motion, the velocity-decomposition technique[11] is a special case of our approach.*

Proof:

Follows directly from Proposition 4.4.4. □

4.5 Algorithm

As stated in the problem statement, there are several parameters that control the planning process. These parameters are necessary to define, for example, the dimensions of the local search window (r, ϕ), the robot's velocity and acceleration bounds, and the time interval within which the velocity change needs to be achieved. We model the local nature of information about the environment by a circle (visibility field). Inside this field, the LSW is represented by a circular sector where the robot can plan its planing step. The algorithm used can be described by the following steps:

I. *Initialization.* The user should specify the values for the parameters as well as choose an initial configuration (p_s, θ) and a final goal position p_g for the robot to navigate.

II. *Discretize the local visibility window.* Searching a continuous space for an optimum local solution is hard. So we discretize the local visible region into a set of equally spaced grid points. At this stage we grow obstacles with time taken into account uncertainties. We also determine LSW^i_{fgp}, LSW^i_{vgp}, and LSW^i_{rgp} at time t. This process enable the robot to avoid choosing any grid point that will lead it to a collision (illustrated in the previous propositions).

III. *Find optimum velocity.* First we determine if a local grid point is reachable[2] by a single velocity[3] from the current point, without going through any obstacles and while maintaining the acceleration constraints. For a reachable grid point we compute the product of goal attraction, static safety, and dynamic safety. Then, we choose the grid point for which this product (henceforth referred to as overall safety) is maximum. This determines the optimum local velocity.

IV. *Continue process.* Repeat steps I and II until the goal is reached.

We now outline a simple proposition which emerges from the last two propositions.

Proposition 4.5.1 *Given p_s and p_g (start and goal configurations), there exists a free path from p_s to p_g provided that:*

(a) *p_s and p_g* $\in W^{local}_{free}$.

(b) $LSW^{i}_{rgp}(t) \neq \{\}, \forall t \in T$, $i = \{0, 1, \ldots, k-1\}$.

4.6 Experimental Results

In this section, some simulation examples of the algorithm described before are discussed. These simulations correspond to different navigation environments.

[2] A point is reachable if it is within the LSW, accessible with a velocity satisfying the acceleration bounds, and visible from the current point.

[3] We consider velocities while maintaining the acceleration bounds, which may be different from the current velocity. However, we do not change the computed velocity before the next step.

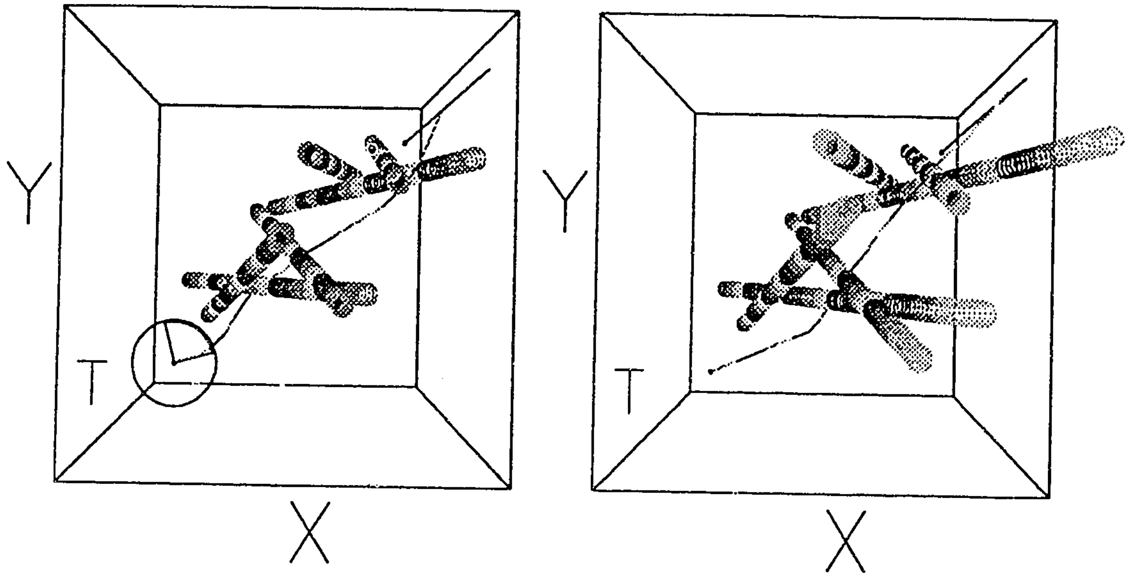

Figure 4.4: *(Left) A free path is generated in 3-D. (Right) A new free path is generated after increasing* t_0 *for the same environment.*

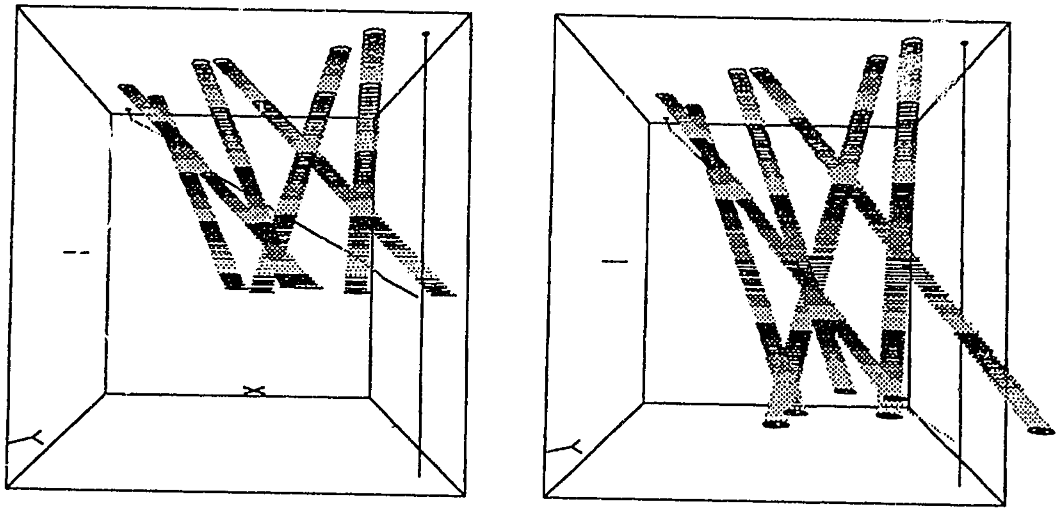

Figure 4.5: *(Left) A different viewpoint of the path in* Figure 4.7(Left). *(Right) A similar different viewpoint for the path in* Figure 4.7(Right).

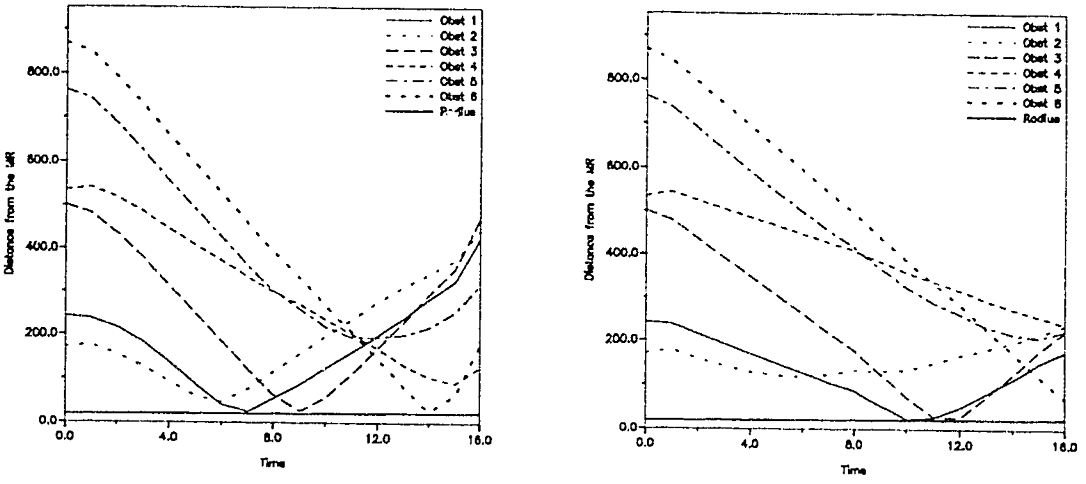

Figure 4.6: *How far is the MR from obstacles?*

In all experiments, the space is represented by a square grid. The start position is located at the lower left corner in the xy-plane at t = 0, and the goal is located at the upper right corner (the goal is grown over time to be a line).

The circle and the circular sector in Figure 4.4(Left) correspond to the visibility field and the local search window, respectively. The local search window is the part of the visibility field that the robot can plan through. It is described by a grid map distributed in a specific area determined by a radius (r) and an angle (ϕ). For illustrative purposes, we used four grey levels for indicating grown obstacles in different time intervals.

Figure 4.4(Left) illustrates the path generated by the *MR* from the starting point to the goal, according to certain parameter values. Figure 4.4(Right)

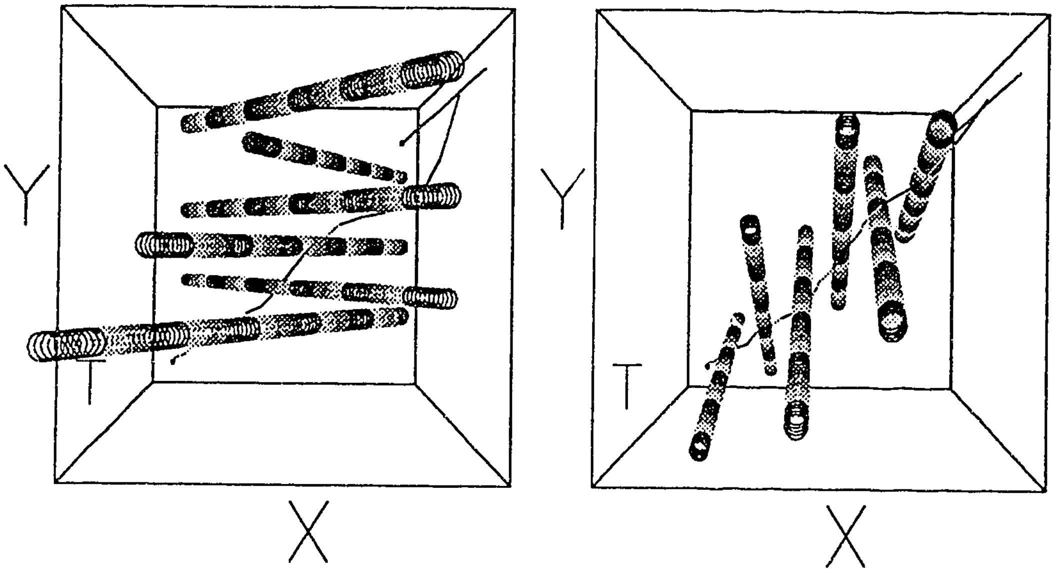

Figure 4.7: *Examples of different dynamic environments.*

shows the effect of increasing the time interval t_0. Different viewpoints for the paths of Figure 4.4 are shown in Figure 4.5. Note the difference in the total time needed to plan the paths and how it affected the size of the grown obstacles in both figures.

Figure 4.6 shows the distance-graphs between the MR and each obstacle in the environments of Figure 4.4. Each distance-graph provides a geometric interpretation of how far each obstacle is from the MR in the time interval [0, t_f]. In order to match each obstacle with its distance-graph, we assume that all obstacles in a certain environment are numbered from left to right (along the x-axis) and bottom-up (if the same x-position is occupied by more than one obstacle) in the xy-plane at t=0. Note that there is another graph labelled Radius which represents the radius of the largest detected obstacle at a specific instant of time. When all obstacles have the same size, the

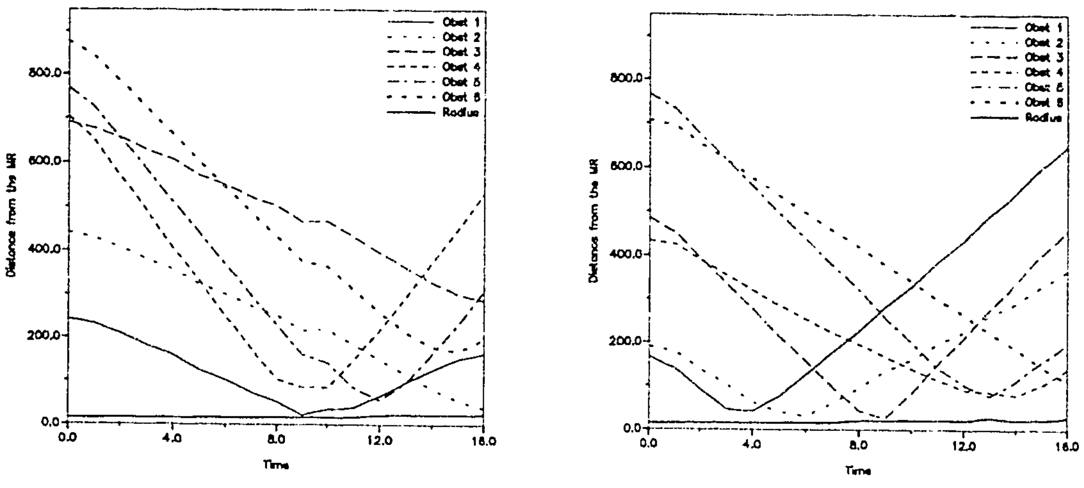

Figure 4.8: *Distance graphs between the MR and obstacles of* Figure 4.10.

radius-graph is represented by a piecewise horizontal line. If any obstacle's distance-graph intersects the radius-graph at a certain time, then a collision occurs between the *MR* and the obstacle at that time. In Figure 4.6(Left), it can be observed that the *MR* comes very close to Obstacles (1, 2, and 3) at different times. Similarly the *MR* approaches both Obstacles (1 and 3) in the time interval [10,12] in Figure 4.6(Right).

We show two different examples in Figure 4.7. In the first one (Figure 4.7(Left)), the robot avoïds all obstacles that are moving horizontally towards its path by either speeding up or slowing down while navigating towards the goal. Similarly, in the second example (Figure 4.7(Right)), the robot avoids obstacles heading towards it. The robot passes in front of a moving obstacle (the path is hidden) if the time-to-collision is large enough to do

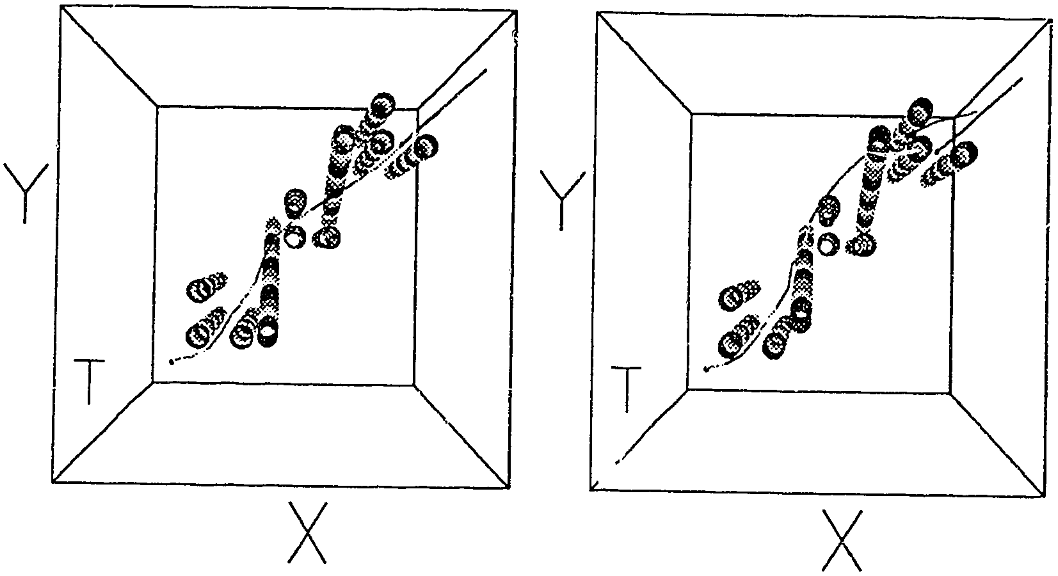

Figure 4.9: *(Left) A free path is generated amidst stationary and moving obstacles. (Right) A new free path is generated after decreasing the speed of the first moving obstacle (moving downwards).*

so. Otherwise, the robot waits for the moving obstacle to pass (the path is drawn over the obstacle). This is illustrated in Figure 4.8. The MR gets very close to Obstacle 1 at t = 9 (units) — the time when the MR passed before Obstacle 1 in Figure 4.7(Left)). Figure 4.8(Right) depicts distance-graphs of obstacles in Figure 4.7(Right). Note that the radius-graphs in Figure 4.8 are not horizontal due to the difference in obstacle sizes used in this experiment.

Finally, Figure 4.9 presents an example in which the MR plans its path amidst stationary and moving obstacles (there are two moving obstacles: the first one is moving downwards and the second one is moving upwards). Two different paths have been generated due to changing the speed of the first moving obstacle. In both figures, the MR avoids the first dynamic obstacle by moving in the direction opposite to the obstacle's motion. The MR behaves

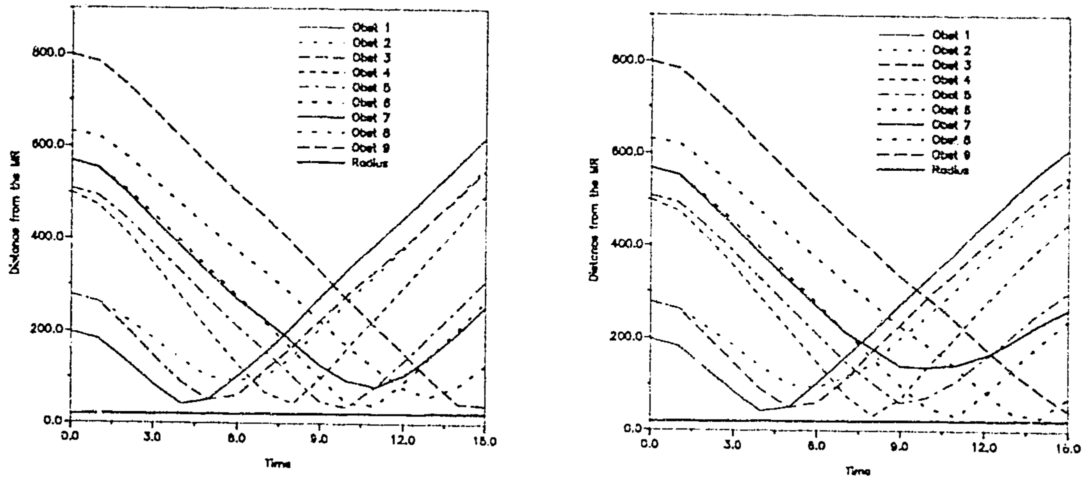

Figure 4.10: *Distance graphs between the MR and obstacles of* Figure 4.12.

similarly when facing the second moving obstacle in Figure 4.9(Left), whereas in Figure 4.9(Right) the MR passes in front of the second moving obstacle (path is hidden) because it is safer to do so. Corresponding distance-graphs of Figure 4.9 are described in Figure 4.10. We omit the distance-graphs of Obstacles 10 and 11 because the package used to generate these graphs (GRTOOLS) has a capacity of 10 graphs per figure.

4.7 Conclusion

We have presented a new method for dynamic path planning using only local information and a knowledge of the start and goal positions. Uncertainty in obstacle positions was also taken into account. The concept of safety was introduced, static safety being a function of the speed of the robot with the acceleration bounds being the parameters. Dynamic safety was described as a function of the time-to-collision with an obstacle. In order to draw the robot closer to the goal, a goal attraction function was used. We showed that the velocity-decomposition technique is a special case of the overall safety function when the direction of motion of the robot is fixed in a planning step. Currently we are working on obstacle detection using visual information. In the future we intend to integrate the vision and planning modules, and develop a real-time implementation on a TRC lab-mate platform.

Bibliography

[1] J. Barraquand, B. Langlois, and J. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. *Proceedings of the Fifth International Symposium of Robotics Research*, 1:74–83, 1989.

[2] A. Basu. A framework for motion planning in the presence of moving obstacles. Technical Report CS-TR-2378, University of Maryland, 1989.

[3] R. Brooks. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, Cybernetics*, 13(3):190–197, March/April 1983.

[4] P. Burlina, D. DeMenthon, and L. Davis. Navigation with uncertainty: Reaching a goal in a high collision risk region. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2440–2445, 1992.

[5] J. Canny and J. Reif. New lower bound techniques for robot motion planning. *Proceedings of the IEEE conference on Foundations of Computer Science*, 1:49–60, 1988.

[6] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *Technical Report No. 266, New York University*, December 1986.

[7] J. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal on Robotics and Automation*, RA-1(1):31–41, March 1985.

[8] A. Elnagar and A. Basu. Heuristics for local path planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2481–2486, May 1992.

[9] K. Fujimura. Route planning for a mobile robot amidst moving obstacles. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots*, pages 433–438, 1992.

[10] J. Gil De Lamadrid and M. Gini. Path tracking through uncharted moving obstacles. *IEEE Transactions on Systems, Man and Cybernetics*, 20(6):1408–1422, 1990.

[11] K. Kant and S. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. *The International Journal of Robotics Research*, 5:72–89, 1986.

[12] K. Kant and S. Zucker. Planning collision-free trajectories in time varying environments: A two-level hierarchy. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1644–1649, 1988.

[13] N. Kehtarnavaz and N. Griswold. Establishing collision zones for obstacles moving with uncertainty. *Computer Vision, Graphics and Image Processing*, 49:95–103, 1990.

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

[15] B. Krogh and C. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1664–1669, April 1986.

[16] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers Group, Massachusetts, USA, 1991.

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

[18] V. Lumelsky and A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[19] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, pages 144–154, 1983.

[20] G. Schmidt and K. Azarm. Mobile robot navigation in a dynamic world using an unsteady diffusion equation strategy. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots*, pages 642–649, 1992.

[21] J. Schwartz and M. Sharir. On the piano movers' problem: I. the case of a two-dimensional rigid polygonal body motion amidst polygonal barriers. *Communications on Pure Applied Mathematics*, 36:345–398, 1983.

[22] R. Sharma. A probabilistic framework for dynamic motion planning in partially known environments. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2459–2464, 1992.

[23] T. Tsubouchi, K. Hiraoka, T. Naniwa, and S. Arimoto. A mobile robot navigation scheme for an environment with multiple moving obstacles. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots*, pages 1791–1798, 1992.

Chapter 5

Motion Detection Using Background Constraints[†]

5.1 Introduction

One of the ultimate goals in robotics is to create autonomous vision-based robots that are capable of executing different tasks, in a real world, without the need for human intervention. An autonomous land-vehicle is an example of a system towards this goal, combining research from the areas of motion planning (in the robotics community) and motion detection (in the vision community).

A vision-based navigation system needs to recognize the presence of static objects as well as moving ones in order to avoid collisions. In other words,

[†] A. Elnagar and A. Basu. Motion Detection using Background Constraints. *Submitted for publication.*

it needs to interact adaptively with its environment. There are two major components in such a system: sensors which provide information about the environment, and a control strategy using which a decision about the next move is made. We have already addressed the second component and proposed methods for the motion planning problem ([3, 9, 10]). In this paper we describe a technique for independent motion detection (related to the first component) from a mobile platform.

For a static camera, a simple technique that subtracts successive images and marks non-zero regions in the resulting image can be used for detecting motion. For a moving camera, the problem is more complex because of the possibility that every object in the environment may be moving with respect to both the camera and the environment[1]. Several researchers have proposed methods for detecting motion in the case of a static camera, but few of them addressed the problem for a moving camera. Motion detection techniques, for a moving camera, can be divided generally into two broad classes which are termed quantitative and qualitative. in the first class, the aim is to produce a 3-D representation of the environment (structure-from-motion). Whereas in the second class, the objective is to recognize a particular pattern (situation) of interest. Both classes of techniques use optic flow fields for detecting motion.

This paper argues that the problem of motion detection in vision-based navigation systems can be solved using *background constraints* and without the need for optic flow estimation. The basic idea of this constraint stems

[1] For an example that shows how hard the motion detection problem can be in a dynamic environment, please see [36].

from the fact that in vehicular motion moving objects are usually in contact with a planar surface. Therefore, if objects are not moving within the camera's field of view, equations show that the motion field varies inversely with distance from a fixed point. Since depth discontinuities do not occur on planar surfaces, any abrupt change indicates a point on the boundary of an object. Using knowledge about the background and the camera motion, the displacement field at each point in the image is computed. A point that lies on a moving object is unlikely to satisfy the estimated displacement field at that specific point (i.e., it is inconsistent with the global pattern of motion).

In order to make this technique more robust, we take into account inaccuracies in determining the displacement field. These errors may result from inexact rotation and/or position readings which generate false motion detection (narrow regions on the boundaries of objects in the environment). To overcome this problem, we use a morphological filter that consists of two operations — erosion followed by dilation. This filter proves to be sufficient for removing false motion due to inaccurate background compensation.

The organization of this paper is as follows: In the next section we survey past research in the area of motion detection. Section 5.3 describes the camera model used throughout this paper. In Section 5.4 we introduce the cartesian displacement field. The related motion equations based on the background constraint are developed in Section 5.5, and then present the algorithm for independent motion detection from a translating and rotating camera. Experimental results demonstrating the validity of the background constraint are presented in Section 5.6. Finally, error analysis dealing with magnitude and type of inaccuracies caused by rotation and/or translation is discussed in Section 5.7.

5.2 Previous Work

In the past few years, several researchers have worked on the motion detection and estimation problems based on optic flow or motion field vectors. Since there are different approaches for estimating optic flow, three different classes of techniques to detect motion have been introduced. The first one used feature based methods to estimate motion field [21], while the second group used correspondence of points or lines [32, 35, 38, 40]. The third class viewed the problem of optic flow estimation as a minimization problem under certain constraints. For example, Horn and Shunk [16] used the smoothness constraint, whereas Nagel [25] and later he and Enkelmann [26] used the oriented smoothness constraint. Shunk [31] developed the line clustering constraint. For a comparative discussion of these constraints see [41].

Motion detection in the case of a stationary camera can be solved simply by subtracting successive images, and then looking for significant differences which identify the boundaries of moving objects [18]. This procedure is also used when a camera is moving but the environment is static [5, 22, 39]. Most of the above references are restricted to one rigid moving object. However, they are not suitable for detecting motion when both the camera and the objects are undergoing motion.

The early work of Ullman [40] is considered the first trial to detect motion while the camera and multiple rigid objects are in motion. He proposed a decomposition procedure of the resulting optic flow into sets that correspond to each moving object in the scene. Later, Jain [17] studied the problem in the case of a translating camera. He used the motion epipolar constraint after applying a complex logarithmic mapping on the image. A similar approach for detecting only the vertical motion is reported by Frazier and Nevatia [13]. Heeger and Hager [15] described a technique that first estimates the motion parameters of the camera and then finds any inconsistent values (that correspond to a moving object) with respect to the estimated parameters. Zhang *et el.* [42] used a similar approach but based it on rigidity constraints.

In a recent detailed study by Thompson and Pong [36], two general classes of techniques are presented. One class uses the motion epipolar constraint while the other one exploits information about camera motion and depth. However, it is not clear how some of these techniques can be implemented in practice.

All techniques presented so far are considered to be quantitative except for some methods in [36]. On the other hand, there are some qualitative techniques in which researchers argue that the motion detection problem can be solved without having to solve the entire structure-from-motion problem. Thompson and Kearney [34] described a strategy that makes use of inaccurate optic flow to detect motion. Later, Bhanu *et al.* [4] proposed a technique for motion detection that is based on identifying a fuzzy focus of expansion and a qualitative measure of the motion of scene points. Nelson [28] described a technique that uses the normal flow component and two qualitative measures to tag motion that is inconsistent with some stored motion patterns. Aloimonos *et al.* [1, 2] presented a more comprehensive study to solve some of the vision problems in the context of the purpose of a specific problem. Recently, Tistarelli and Sandini [37] have explored more applications of active vision — obstacle detection being one of them.

Motion detection in some particular applications such as obstacle avoidance has also been researched. Olin *et al.* [30] tried to segment single images in order to detect obstacles, while Daily *et al.* [8] used range data. Moreover, the idea of field divergence has been exploited, for example [29, 7]. Grosso *et al.* [14] used stereo vision and motion analysis to infer scene structure and to control the movement of a mobile robot. Among these works, Mallot *et al.* [23] and Enkelmann [11] studied approaches similar to ours. Both of them assumed a camera moving parallel to the planar surface. Mallot used spatial disparities to detect obstacles while Enkelmann used temporal disparities. Our approach is close to Enkelmann's approach. However, our methodology is more general since we consider both translation and rotation. Moreover, we do not use optic flow in motion detection.

5.3 Camera Model

Motion detection requires a mathematical model which describes the relationships between a 3-D point (P), its projection on the image plane (p), and the motion parameters of the camera. We will define the coordinate systems that will be used and derive relationships between them in the following section.

Throughout this work, the pinhole camera model is used. As shown in Figure 5.1, the origin of a cartesian coordinate system $OXYZ$ is the viewpoint of the camera and its optical axis is aligned with the Z axis. The

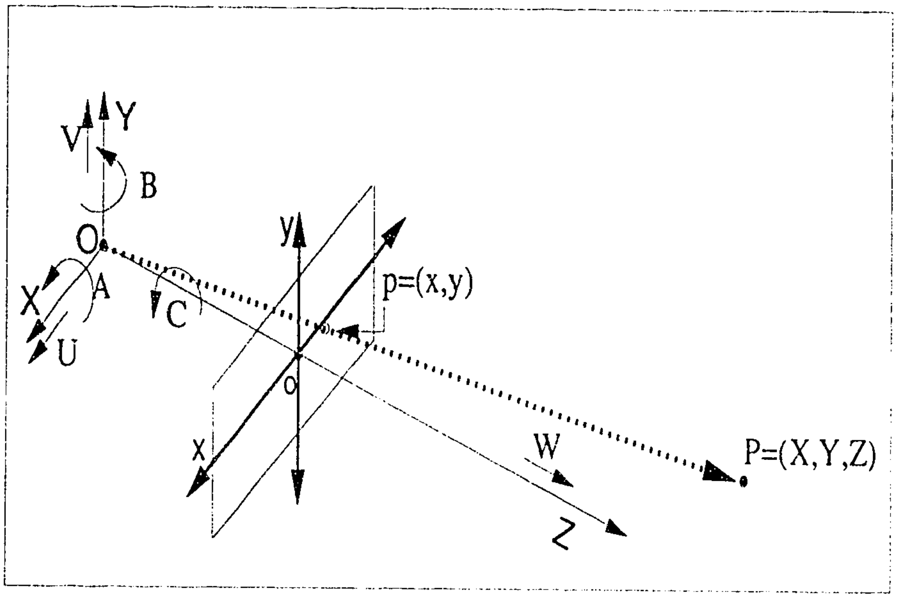

Figure 5.1: *Camera coordinate system and image coordinate system.*

image is the projection of a 3-D scene onto a plane located at distance f (focal length) from O. The image coordinate system is represented by oxy in Figure 5.1. Using the above model and assuming perspective projection, the relationships between points in the image plane and points in 3-D with respect to the camera coordinate system are:

$$x = f_x \frac{X}{Z} \qquad y = f_y \frac{Y}{Z} \tag{5.1}$$

where $P : (X, Y, Z)$ is a point in the camera coordinate system, and $p : (x, y)$ is its projection on the image plane; f_x: number of pixels correspond to f along the x-axis; f_y: number of pixels correspond to f along the y-axis.

The camera system considered in this work is mounted on a pan/tilt device that allows rotation and a platform that supports translation. Since

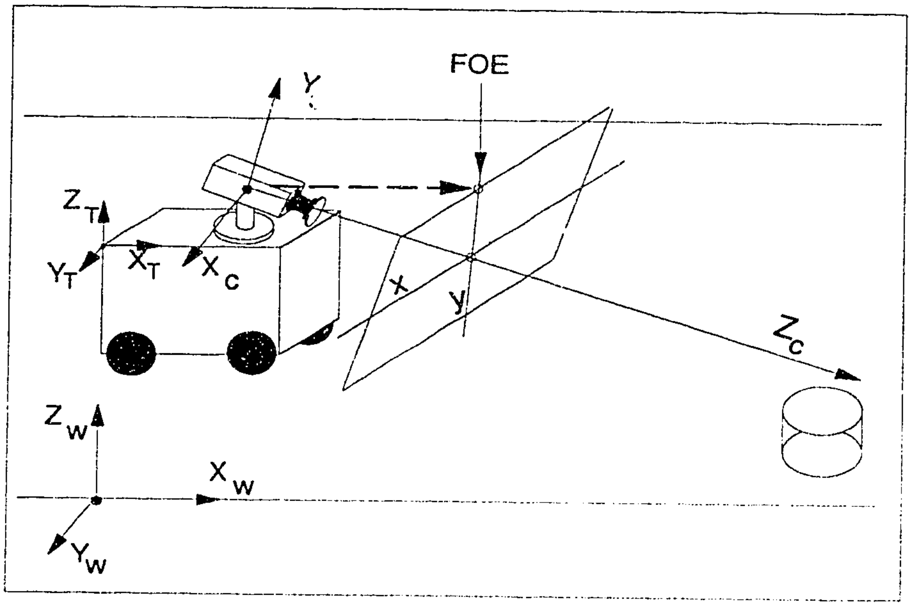

Figure 5.2: *The camera model*

we are interested in mobile robots (i.e., vehicular motion), we only consider rotating the camera around the pan axis (Y_C-axis) keeping the tilt angle fixed. We allow translation to take place only in the XZ plane (always in contact with the planar surface). The relative motion of the camera with respect to a rigid surface is described by a translational velocity $T = (U, V, W)$; and a rotational velocity $\Omega = (A, B, C)$ around O (see Figure 5.1). A simple drawing of the camera system is depicted in Figure 5.2. We will now show the derivation of the displacement field.

5.4 Cartesian Displacement Field

We assume the camera is moving parallel to a planar surface. We further assume that the ego-motion parameters with respect to the planar surface are known[2]. To estimate the displacement field, let $P = (X, Y, Z)$ be a point in the camera coordinate system at time t, and $\tilde{P} = (\tilde{X}, \tilde{Y}, \tilde{Z})$ be the corresponding coordinates at time $\tilde{t} = t + \delta t$. Let (x,y) and $(\tilde{x}, \tilde{y})$ be the image projections of P and $\tilde{P}$, respectively. The components of the displacement field (u,v) are computed from the displacement vectors (assuming time interval to be unity) as follows:

$$u = \tilde{x} - x \tag{5.2}$$

$$v = \tilde{y} - y \tag{5.3}$$

where $\tilde{x} = (f_x \frac{\tilde{X}}{\tilde{Z}})$ and $\tilde{y} = (f_y \frac{\tilde{Y}}{\tilde{Z}})$. $\tilde{X}$ and $\tilde{Y}$ are computed using the relation

$$\tilde{P} = R \times P + T \tag{5.4}$$

where R is the rotation matrix defined as:

$$R = \begin{pmatrix} 1 & -C & B \\ C & 1 & -A \\ -B & A & 1 \end{pmatrix}$$

Solving (5.4) and substituting in (5.2) and (5.3), we obtain:

$$u = \frac{-\frac{xy}{f_y}A + (f_x + \frac{x^2}{f_x})B - \frac{f_x}{f_y y}C + \frac{U - xW}{Z}}{1 + \frac{yA}{f_y} - \frac{xB}{f_x} + \frac{W}{Z}} \tag{5.5}$$

$$v = \frac{(\frac{-y^2}{f_y} - f_y)A + \frac{yx}{f_x}B + \frac{f_y}{f_x x}C + \frac{V - yW}{Z}}{1 + \frac{yA}{f_y} - \frac{xB}{f_x} + \frac{W}{Z}} \tag{5.6}$$

[2] If Ego-motion parameters are not known, several different techniques can be used to estimate them [6, 12, 19, 27, 33].

If $\frac{W}{Z} << 1$, the camera field of view is relatively narrow, and image sampling is assumed to be fast (i.e., the rotation angles are small, less than 5°) then the denumerator in (5.5) and (5.6) is approximately equal to 1. Thus the displacement field becomes the same as the velocity fields [5]. Note that each displacement vector above can be represented as the sum of a translational component and a rotational one:

$$\begin{aligned} u &= u_T + u_R \\ &= [\frac{f_x U - xW}{Z}] + [-\frac{xy}{f_y}A + (f_x + \frac{x^2}{f_x})B - \frac{f_x}{f_y y}C] \qquad (5.7) \\ v &= v_T + v_R \\ &= [\frac{f_y V - yW}{Z}] + [(\frac{-y^2}{f_y} - f_y)A + \frac{yx}{f_x}B + \frac{f_y}{f_x x}C] \qquad (5.8) \end{aligned}$$

5.5 Background Constraint

If moving objects are in contact with the planar surface (vehicular motion) a technique that depends only on knowing the image plane locations corresponding to discontinuities in range is needed. If no object is moving within the camera field of view, motion equations show that flow varies inversely with distance for a given fixed point on the planar surface. Discontinuities in motion field correspond to discontinuities in depth of that point. In the case of a planar surface depth discontinuities are not possible, therefore any significant variation in motion field should correspond to the boundary of a moving object.

Consider a planar surface which can be described in both spatial and image coordinate systems as follows:

$$\begin{aligned} D &= k_x X + k_y Y + k_z Z, \quad or \\ \frac{D}{Z} &= k_x \frac{x}{f_x} + k_y \frac{y}{f_y} + k_z \end{aligned} \tag{5.9}$$

where (k_x, k_y, k_z), denoted N^C, is the planar surface's normal with respect to the camera coordinate system, and D is a constant.

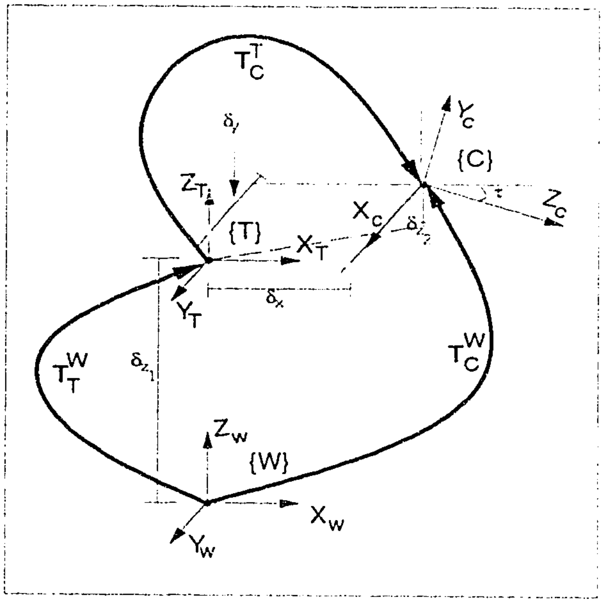

Figure 5.3: *The transformation graph that relates camera, robot, and world coordinate systems.*

Since we assumed that a mobile robot moves parallel to the planar surface, the plane's normal (slope of the plane) with respect to the world frame of reference, denoted N^W, is chosen to be $(0, 0, 1)$. To compute N^W with respect to the camera coordinate system, we substitute in the following relation:

$$\begin{bmatrix} (N^C)^{\mathrm{T}} \\ 1 \end{bmatrix} = T_W^C \begin{bmatrix} (N^W)^{\mathrm{T}} \\ 1 \end{bmatrix} \tag{5.10}$$

where T_W^C is a 4×4 homogeneous transformation matrix defined as:

$$T_W^C = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \delta_x \\ r_{21} & r_{22} & r_{23} & \delta_y \\ r_{31} & r_{32} & r_{33} & \delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The 3×3 sub-matrix ($R_W^C = [r_{11}...r_{33}]$) represents the rotation matrix relating the world coordinate system ($\{W\}$) to the camera coordinate system ($\{C\}$), and $[\delta_x, \delta_y, \delta_z]$ denotes the displacement vector from the origin of $\{W\}$ with respect to $\{C\}$.

Figure 5.3 describes the transformation graph between three coordinate systems: the world, the robot $\{T\}$, and the camera. To obtain the orientation of $\{C\}$ from $\{T\}$ (or $\{W\}$), we rotate W about Z_W by an angle -90 ($Rot(Z_W, -90)$), and then rotate about X_W by an angle $-(90+\tau)$ ($Rot(X_W, -(90+\tau))$), where τ is the tilt angle. Formally,

$$R_C^T = R_C^W = Rot(Z_W, -90) Rot(X_W, -(90+\tau))$$

From the transformation graph of Figure 5.3, we know:

$$T_C^W = T_T^W T_C^T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta_{z_1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \sin(\tau) & \cos(\tau) & \delta_x \\ 1 & 0 & 0 & -\delta_y \\ 0 & \cos(\tau) & -\sin(\tau) & \delta_{z_2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \sin(\tau) & \cos(\tau) & \delta_x \\ 1 & 0 & 0 & -\delta_y \\ 0 & \cos(\tau) & -\sin(\tau) & \delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\delta_z = \delta_{z_1} + \delta_{z_2}$. Since T_C^W is an orthogonal matrix, it follows directly:

$$\begin{aligned} T_W^C &= (T_C^W)^{-1} \\ &= \begin{bmatrix} 0 & 1 & 0 & \delta_y \\ \sin(\tau) & 0 & \cos(\tau) & -\delta_x \sin(\tau) - \delta_z \cos(\tau) \\ \cos(\tau) & 0 & -\sin(\tau) & -\delta_x \cos(\tau) + \delta_z \sin(\tau) \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Substituting in (5.10) yields:

$$\begin{bmatrix} (N^C)^{\mathrm{T}} \\ 1 \end{bmatrix} = T_W^C \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \delta_y \\ (1 - \delta_z)\cos(\tau) - \delta_x \sin(\tau) \\ (\delta_z - 1)\sin(\tau) - \delta_x \cos(\tau) \\ 1 \end{bmatrix}$$

Consequently,

$$\begin{aligned} k_x &= \delta_y \\ k_y &= (1 - \delta_z)\cos(\tau) - \delta_x \sin(\tau) \\ k_z &= (\delta_z - 1)\sin(\tau) - \delta_x \cos(\tau) \end{aligned}$$

An equation of the plane (background) through $P_0^C = (X_0^C, Y_0^C, Z_0^C)$ with normal vector N^C is:

$$N^C \cdot (P^C - P_0^C) = 0 \tag{5.11}$$

where $P^C = (X, Y, Z)$ and P_0^C is the origin of $\{T\}$ in the world's frame of reference with respect to the camera frame of reference computed as follows:

$$\begin{bmatrix} (P_0^C)^{\mathrm{T}} \\ 1 \end{bmatrix} = T_W^C \begin{bmatrix} (P_0)^{\mathrm{T}} \\ 1 \end{bmatrix} = \begin{bmatrix} \delta_y \\ -\delta_z \cos(\tau) - \delta_x \sin(\tau) \\ \delta_z \sin(\tau) - \delta_x \cos(\tau) \\ 1 \end{bmatrix}$$

Substituting in (5.11) and using the perspective projection equations, we obtain:

$$Z = \frac{\delta_z(\delta_z - 1) + \delta_x{}^2 + \delta_y{}^2}{k_x \frac{x}{f_x} + k_y \frac{y}{f_y} + k_z} \tag{5.12}$$

where δ_x, δ_y are updated as the camera translates. This is simply because the camera coordinate system changes in position. As an illustration consider Figure 5.4 and assume the origin of the camera coordinate system to be at (X_1, Y_1, Z_1) before motion, and at (X_2, Y_2, Z_2) after the camera moves with a speed S for t seconds (time elapsed between the two processed images). It follows:

$$\begin{aligned} \delta_x &= \delta_x + (X_2 - X_1)t = \delta_x + S\cos(B) \cdot t \\ \delta_y &= \delta_y + (Y_2 - Y_1)t = \delta_y + S\sin(B) \cdot t \\ \delta_z &= \delta_z \end{aligned}$$

These relationships are described graphically in Figure 5.4. The new position of the camera coordinate system after motion is denoted by $\{C'\}$.

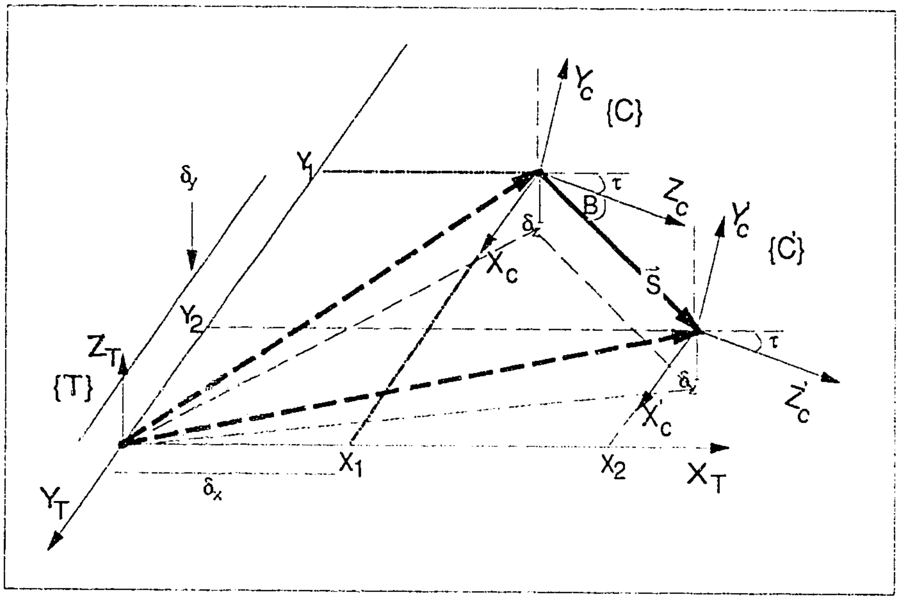

Figure 5.4: *Updating δ_x and δ_y.*

5.5.1 Motion Detection

The detection of moving objects is certainly a prerequisite for many tasks. For example, it is an important issue in visual navigation. In this paper, a general solution for motion detection in mobile robot navigation on planar surfaces is described. We consider the general case, where an object in view and the camera are moving rigidly (rotating and/or translating) with respect to each other as well as the background.

For a stationary camera, the pixel-by-pixel subtraction technique can be used to detect motion, since with a static scene a given 3-D point will continuously project to the same position in the image plane. For a moving camera this is not the case. To apply pixel-by-pixel comparison with an active camera image sequence, we must map pixels which correspond to the same 3-D point to the corresponding image plane positions.

In the following subsections, we will derive the mapping function between images resulting from camera motion. For each pair of images processed in the image sequence, the image at time t is mapped so as to correspond pixel-by-pixel with the image at time $t+\delta t$. Regions with no match between the two images are ignored since they represent distant regions of the environment that can be dealt with later on.

Translating Camera

Using (5.12) and the assumptions about vehicular motion (mentioned earlier), we can easily compute the displacement field (in the translational case) denoted by (u_T, v_T):

$$u_T = \frac{(f_x U - xW)}{Z} \tag{5.13}$$

$$v_T = \frac{(f_y V - yW)}{Z} \tag{5.14}$$

Given the speed of the camera (S), pan angle (B), and tilt angle (τ), translation motion components of the camera (U, V, W) can be determined easily:

$$\begin{aligned} U &= S\cos(\tau)\sin(B) \\ V &= S\sin(\tau) \\ W &= S\cos(\tau)\cos(B) \end{aligned}$$

The location of the *focus of expansion* (FOE) on the image plane (see Figure 5.2) is

$$(FOE_x, FOE_y) = [\frac{U}{W}, \frac{V}{W}]$$

The magnitude of motion field (v=(u,v)) associated with any surface point on the background (say p=(x,y)) is proportional to the distance between the point and the camera. It is defined as:

$$\|\mathrm{v}\| = \frac{W}{Z}\sqrt{(FOE_x - x)^2 + (FOE_y - y)^2}$$

The orientation of the motion field at a certain point on the image plane can be determined if the translational motion parameters are known.

$$FOE_\theta = \tan^{-1}(\frac{FOE_y - y}{FOE_x - x})$$

Note that (5.13) and (5.14) can be rewritten in terms of FOE:

$$\begin{aligned} u_T &= \frac{(FOE_x - x)W}{Z} \\ v_T &= \frac{(FOE_y - y)W}{Z} \end{aligned}$$

Based on the magnitude and the orientation of the motion field at a certain point, we can determine easily if this point lies on an obstacle and whether this obstacle is static or moving[3]. The magnitude of the motion field at a certain point on the background is estimated from the motion parameters. We can also determine the motion field at each point in the image plane using Horn's algorithm, for example. The estimated and computed values of the motion field at a certain point should be approximately the same if this point lies on the background. Otherwise, this point belongs to an obstacle. This is a simple way of detecting obstacles.

[3] Detecting static obstacles is not discussed here.

Given two consecutive images grabbed a short time apart, we would like to establish a relationship between any pixel in the first image with its corresponding one in the next one, provided that the camera is only translating. A simple subtraction test between corresponding pixels is used to produce a binary image in which non-zero regions represent motion. Assume that an image ($I(t)$) is grabbed at time t and another image ($I(t+1)$) is grabbed at time $t+1$. Let P be a 3-D point that is projected on both images:

$$p(t+1) = (x(t+1), y(t+1)) = (x(t) + u_T, y(t) + v_T) = p(t) + \mathrm{v} \quad (5.15)$$

The basic idea of detecting independently moving objects is based on the observation that given the camera motion, any point in the image should satisfy (5.15). Whereas a point that lies on an independently moving object is unlikely to satisfy (5.15). Therefore, detecting moving objects becomes easy depending on this constraint, which can be used for both translation and rotation. The analysis so far dealt with the problem of motion detection from a translating camera. Next we consider the case of rotational camera motion.

Rotating Observer

The rotational component around the pan-axis is the only parameter of motion considered here. This is not a very restrictive assumption because of the nature of our application (mobile robot navigation). A rotational component around the Z-axis is not realistic since we, humans, do not roll our heads while driving. The other component (rotation around the X-axis (tilt)) is not crucial either. There is no need to abruptly move ones head up and down while driving on a road. Therefore we consider varying the pan angle while keeping the tilt angle fixed.

Similar to the translational case, we would like to establish a relationship between pixels that represent the projection of a 3-D point in two images grabbed at different instants of time. Note that the camera coordinate system and the change in the image plane as the camera rotates (Figure 5.5). This is ideal for background compensation, since visual information is invariant to camera rotation [20].

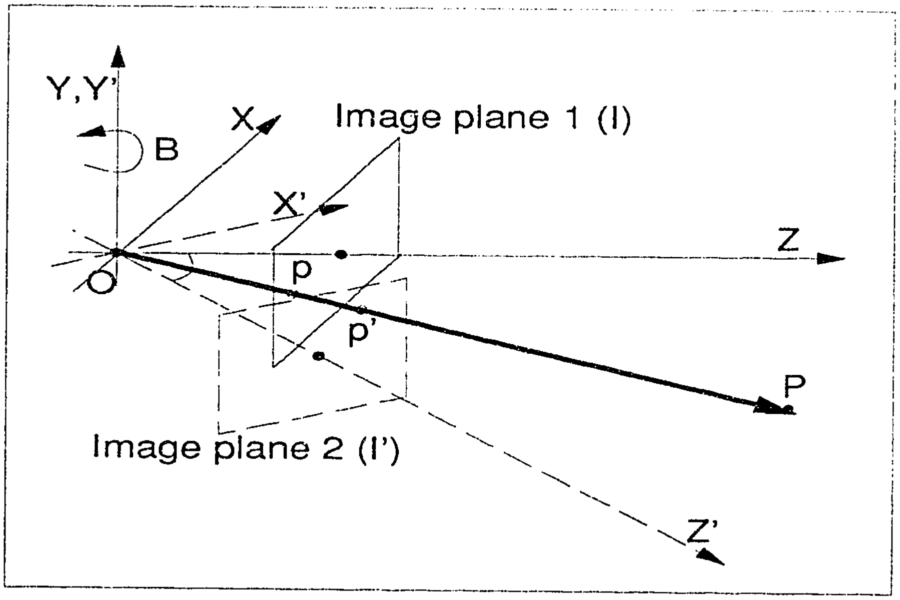

Figure 5.5: *The effect of rotation on the camera coordinate system.*

To develop the theory, let $p_{ij}(t) \in I(t)$ correspond to $p'_{ij}(t) \in I(t+1)$ after a small step of rotation (B) around the pan-axis. Denote the new camera coordinate system by $\{C^N\}$. Given B, any pixel in $I(t)$ can be related to its new location in $I(t+1)$ depending on the transformation that relates $\{C\}$

to $\{C^N\}$ except for some pixels in the periphery. The relationship is:

$$\begin{bmatrix} (P^{C^N})^{\mathrm{T}} \\ 1 \end{bmatrix} = T^{C}_{C^N} \begin{bmatrix} (P^{C})^{\mathrm{T}} \\ 1 \end{bmatrix}$$

where

$$T^{C}_{C^N} = \begin{bmatrix} \cos(B) & 0 & \sin(B) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(B) & 0 & \cos(B) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Assume $P^C = (X(t), Y(t), Z(t))$ and $P^{C^N} = (X(t+1), Y(t+1), Z(t+1))$, then

$$\begin{bmatrix} X(t+1) \\ Y(t+1) \\ Z(t+1) \\ 1 \end{bmatrix} = \begin{bmatrix} X(t)\cos(B) + Z(t)\sin(B) \\ Y(t) \\ -X(t)\sin(B) + Z(t)\cos(B) \\ 1 \end{bmatrix} \tag{5.16}$$

Using (5.1) and (5.16) we can establish the relationship in the image coordinate system:

$$x(t+1) = f_x \frac{x(t)\cos B + f_x \sin(B)}{-x(t)\sin B + f_x \cos(B)} \tag{5.17}$$

$$y(t+1) = \frac{f_x y(t)}{-x(t)\sin B + f_x \cos(B)} \tag{5.18}$$

Assuming a small rotational angle between consecutive frames, we can approximate (5.17) and (5.18) by

$$x(t+1) = f_x \frac{x(t) + f_x B}{f_x - x(t)B}, \qquad y(t+1) = \frac{f_x y(t)}{f_x - x(t)B}$$

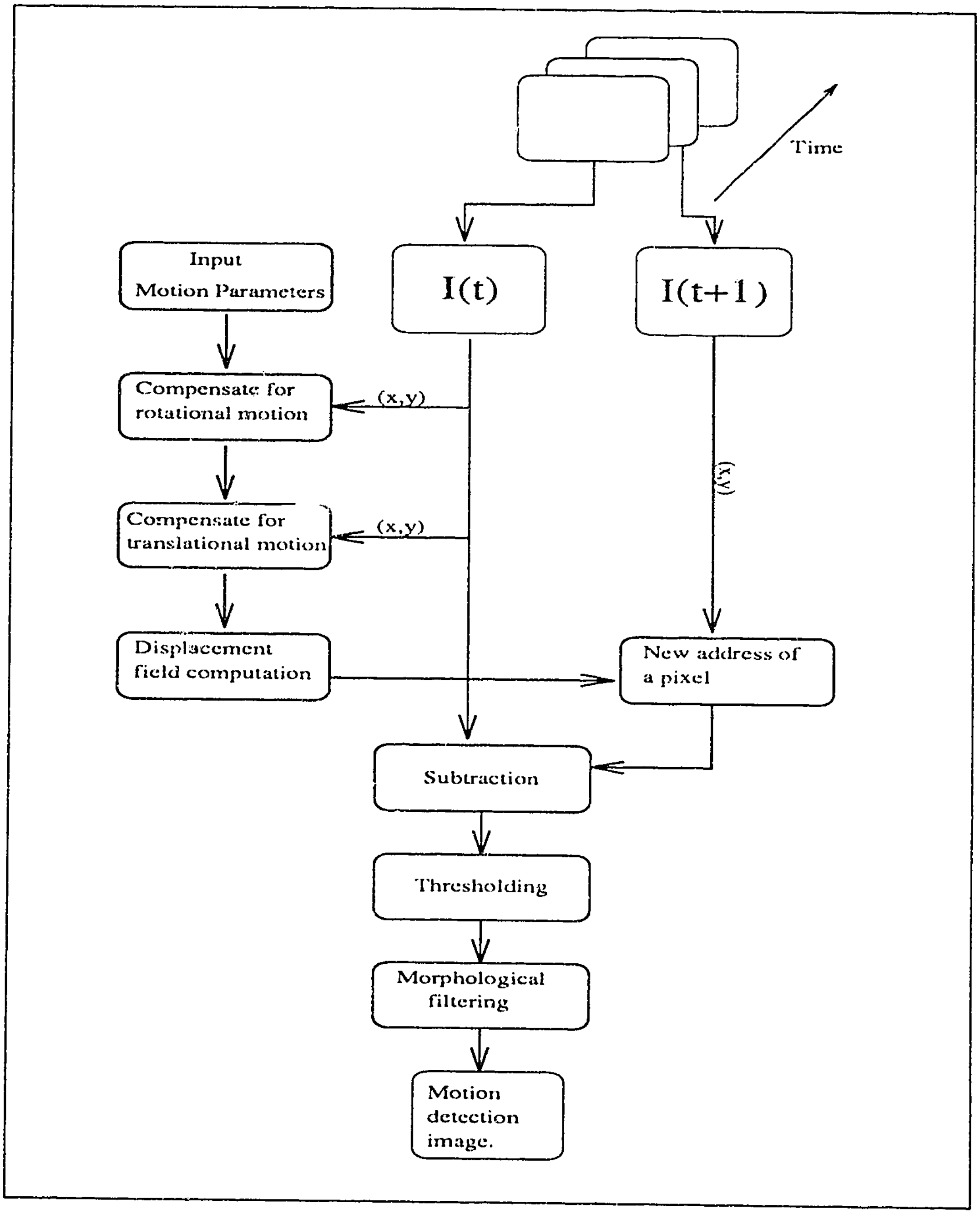

Figure 5.6: *Block diagram of the algorithm.*

This defines the relationship between the projections of a 3-D point on two image planes (e.g., the relationship between p and p' in Figure 5.5). The rotational component of the displacement field (u_R, v_R) becomes:

$$u_R = x(t+1) - x(t) = \frac{(x^2(t) + f_x^2)B}{f_x - x(t)B} \tag{5.19}$$

$$v_R = y(t+1) - y(t) = \frac{(x(t)y(t))B}{f_x - x(t)B} \tag{5.20}$$

This completes the computation of the displacement field **v** resulting from a moving (translating and rotating) camera. A simple flowchart of the algorithm used to detect motion on planar surfaces is described in Figure 5.6.

If we could achieve exact background compensation, the method described so far would be sufficient. In the presence of position inaccuracies, however, the results of these methods deteriorate. Since errors in angle and position readings are inevitably present, it is desirable to develop methods of motion detection that can robustly reject the false motion they cause.

An example of the result of motion detection after inaccurate background compensation is shown in Figure 5.8(C). In this example, no motion should be detected since all objects are static except the camera (i.e., the resulting image should be a black image). However, false motion is detected, which is characterized by narrow (white) bands bordering some of the strong edges of the scene background. Our approach to removing the false motion utilizes the expectation of a wide region of true motion being present. By using morphological erosion and dilation we eliminate narrow regions of detected motion, while preserving the original size and shape of the wide regions.

Morphological Filtering

One application of morphological filters is in noise reduction or suppression[4]. The basic idea is to apply a mask **M** over an image I. The value of an element (m_{ij}) in **M** is either 0 or 1.

Two operations in morphological filtering are of interest to us. They are *erosion* and *dilation*. The erosion operation can be considered as a single pass of a thinning operator. The dilation operation, on the other hand, is the reverse of erosion. In other words, it is an expansion of a set (e.g., an object) into all the background pixel cells which border the set (the object). Thus, erosion shrinks an object whereas dilation enlarges it. Formally, given a mask M ($n \times n$) and a part of a binary image A of the same size as the mask, we can define the erosion operator as follows:

$$A \ominus M = \begin{cases} 1 & \text{if } ((\forall p_{ij} \in A) \wedge (p_{ij} = 1)) \\ 0 & \text{otherwise} \end{cases}$$

Dilation is the dual of erosion:

$$A \oplus M = \begin{cases} 1 & \text{if } ((\exists p_{ij} \in A) \wedge (p_{ij} = 1)) \\ 0 & \text{otherwise} \end{cases}$$

By applying erosion to the image, narrow regions can be eliminated while wider ones are thinned. In order to restore these wide regions back, dilation is applied using a mask of the same size. For example Figure 5.8 (C) shows false motion that has been detected between the frames (A) and (B) of the same figure due to errors in position readings. To eliminate this noise, image (C) is

[4] For a similar application of morphological filtering in motion tracking, see [24].

eroded by two different masks of sizes 3×3 and 5×5. The results of applying these masks are shown in Figures 5.8(D) and 5.8(E), respectively. For this particular image sequence we can see that to completely eliminate the noise due to position inaccuracies, we must use a mask size of 5×5. Figure 5.12 (to be introduced in Section 5.7) provides a geometric explanation of these masks.

5.6 Experimental Results

The camera system used for experimentation is mounted on a pan/tilt device and a platform that supports translation. The experimental results presented in this section are from four sequences of images. All image sequences use a camera that has a fixed tilt angle and restricts the camera rotation around the pan-axis. The image sequences are processed off-line.

Figure 5.7 (image sequence 1) is taken with camera motion constrained to translation only, whereas in other experiments (Figures 5.8, 5.10, and 5.11) we use both rotational and translational motions. In image sequence 1, the two objects (Figures 5.7(A) and 5.7(B)) are stationary while the camera is the only moving object in that environment. The camera is translating towards the scene with speed S parallel to the background. Images (A and B) of Figure 5.7 are grabbed before and after motion, respectively. If the computation of the displacement field is exact then no motion should be detected, which agrees with the experimental result of Figure 5.7(C). In a variation of this experiment, we consider the camera speed to be $\frac{S}{2}$ and then

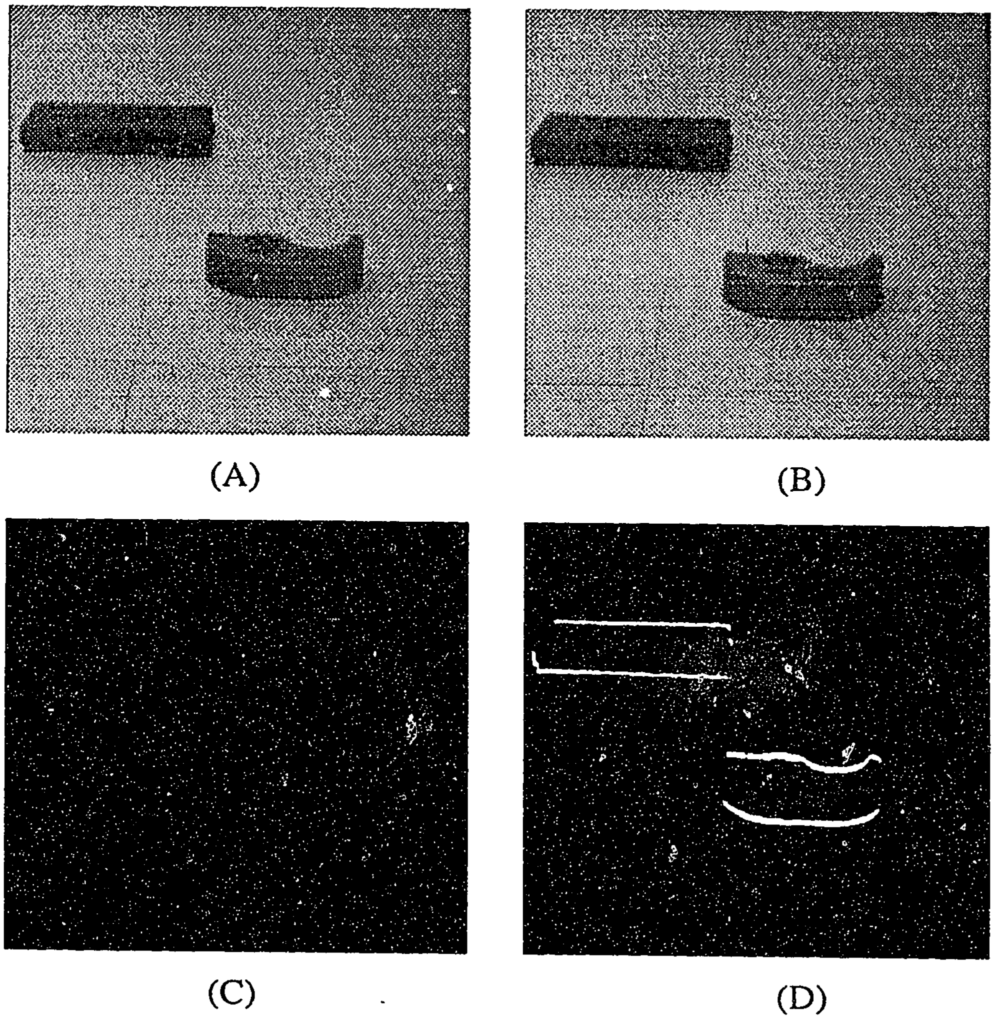

Figure 5.7: *Image sequence 1: The camera is translating in a static environment. Motion detection is produced as a result of considering the speed to be* $(\frac{S}{2})$.

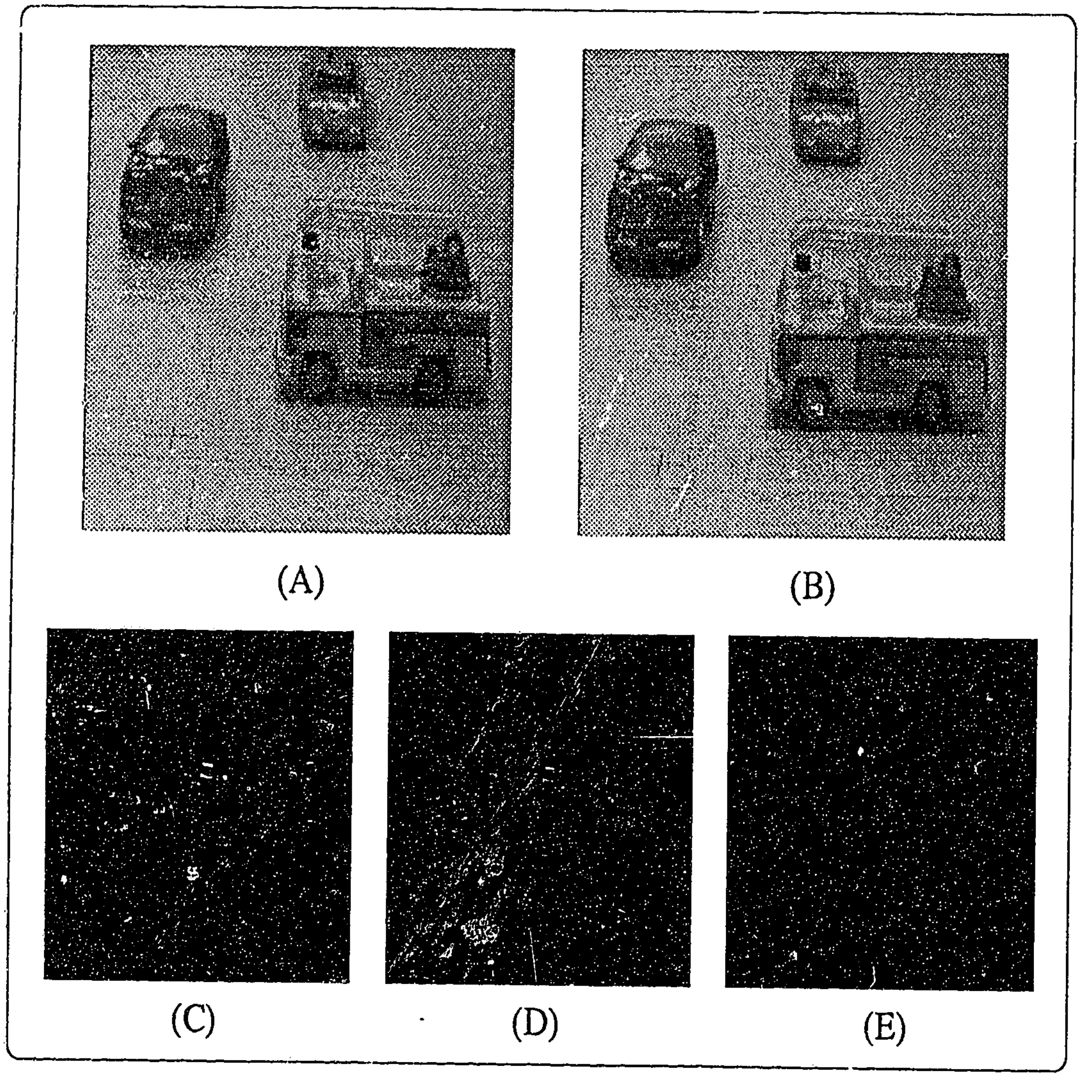

Figure 5.8: *Image sequ*^*nce 2: Noise in motion detection and the effect of morphological filtering (erosion and dilation).*

perform the same computation of the displacement field. In other words, we assume that both objects are moving towards the camera while the camera is translating with speed $\frac{S}{2}$ towards them. The result is shown in Figure 5.7(D) which indicates motion around both objects.

Figure 5.8 (image sequence 2) is taken while the camera is translating and rotating in small steps around the Y-axis. Images (A and B) of Figure 5.8 represent a static scene of three cars. Figure 5.8(C) shows the motion detection result before filtering. After applying morphological filters 3 × 3 and 5 × 5 masks, respectively, better performance is obtained (Figures 5.8(D) and 5.8(E)). The false motion detected in part (C) is attributed to errors in the readings of either the pan-rotation angle or the speed (position) of the camera. This topic will be further discussed in the next section.

Images (A and B) of Figure 5.9 are taken while the fire-truck (closest one to the camera) is moving to the left while the camera is translating towards it. The resulting motion detection images are shown in Figures 5.9(C) and 5.9(D). Image (C) describes the motion that took place between image (A) of Figure 5.8 and image (A) of Figure 5.9 whereas image (D) shows the accumulated motion between image (A) of Figure 5.8 and image (B) of Figure 5.9.

Images (A, B, and C) of Figure 5.10 are taken by a camera moving toward the center of the image. There are three cars on the background. Two are moving and one is static. Image (B) is obtained when the car (closest one to the camera) is passing by from the right. Image (C) describes the situation in which the second car (to the left of the camera) is moving towards the camera while the other one is moving backwards. The results depicted in

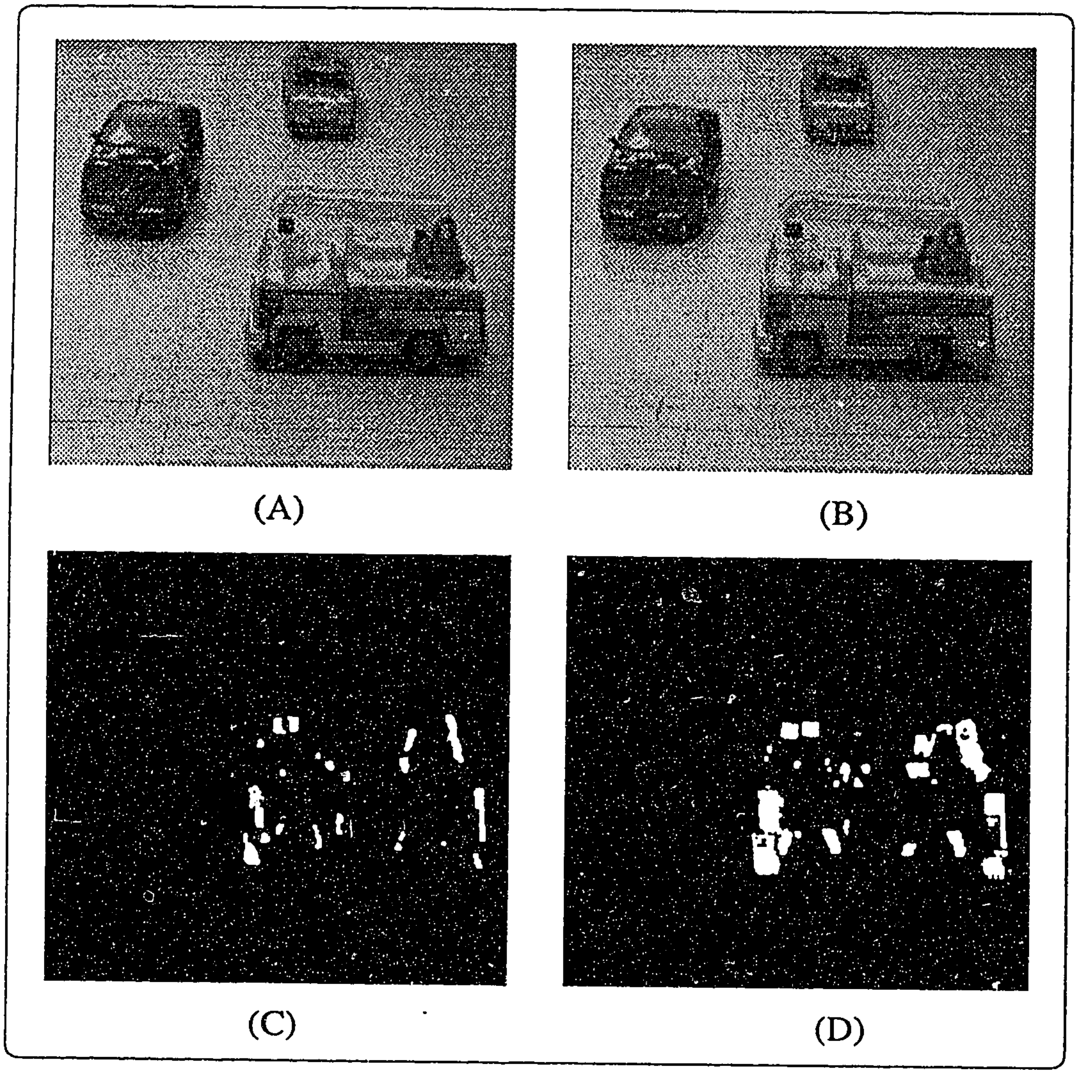

Figure 5.9: *Image sequence 2: motion detection results from moving the firetruck while the camera is moving.*

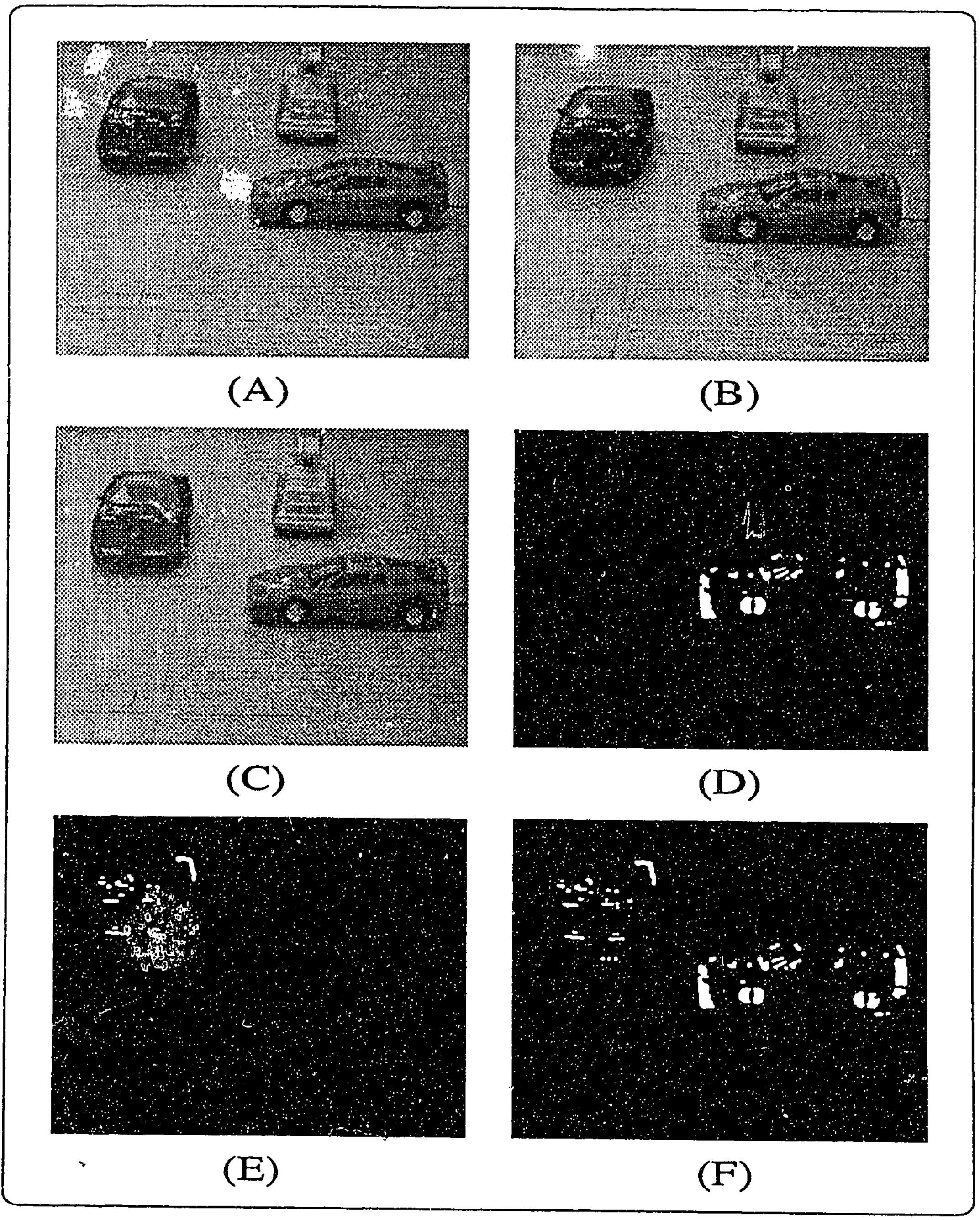

Figure 5.10: *Image sequence 3: An example of motion detection where two objects move between image frames.*

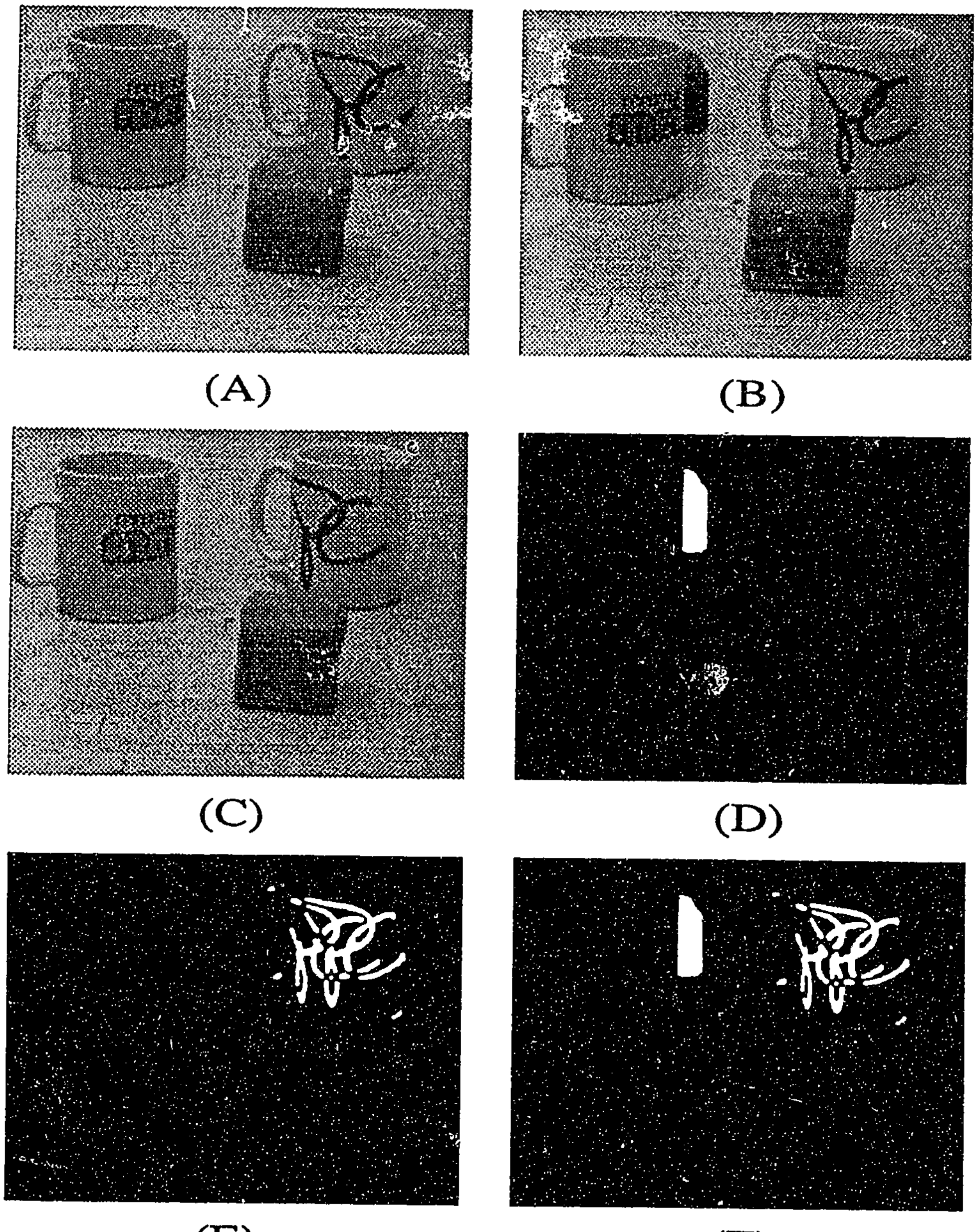

Figure 5.11: *Image sequence 4: An example of motion detection where two objects moved (one rotated and the other entered the scene).*

images (D, E, and F) correspond to images (A and B), (A and C), and (B and C), respectively. Note that in image (E), the backward motion of the sports-car puts it back almost in its original place (i.e., no motion is detected between frames (A and C) of Figure 5.10).

Finally, Figure 5.11 describes a similar example but with rotation and translation object motions. This image sequence (Figures 5.11(A), 5.11(B), and 5.11(C)) is obtained from a translating and rotating camera. We start with two cups and a block that are stationary (Image (A)). Image (B) is taken after the camera moves towards the scene and a new object enters (from behind the left cup) is moving to the right. The detection of this moving block is shown in Image (D). Image (C) is obtained after hiding the new block and rotating the cup (located at the middle-right part of the image) clockwise — note the characters (*pc*) written on it. The result of this motion detection is shown in image (E). The rotated cup is not detected since it does not translate. However, the characters (*pc*) is detected twice. This is because the rotation applied on the cup is slightly large. Image (F) reflects the motion that took place between images (B and C).

As evidenced from the results shown in Figures 5.7, 5.8, 5.10 and 5.11, some false motion is detected due to inaccurate background compensation. In image sequence 2, for example, we can see that false motion is present. Specifically, motion is detected along the border of the flashing-light on the fire-truck. However, our proposed motion detector appears to perform remarkably well in all the above experiments. Minor false motion is eliminated by using morphological filtering.

It should be noted that if the heights of objects in the scene are very large (i.e., almost blocking the camera's field of view) then false motions will be generated at the upper portions of these objects. This is because of the large distance between the points on the upper portions of these objects and the background. Therefore, some constraints on the height of the objects should be taken into account. Formal derivation of such constraints is presented in Appendix E. However, this problem is significant when the field of view of the camera is wide, which is not the case in our system as well as all other successful experimental systems for motion detection as pointed out by Thompson and Pong [36].

In the next section, we analyze (theoretically) the magnitude and types of errors caused by different sources — namely, pan angle variation and position reading.

5.7 Error Analysis

It should be noted that noisy position information corrupts background compensation algorithms and necessitates additional noise removal techniques. Morphological filtering has been presented as one alternative to remove narrow regions of false motion from subtracted images. For effective noise removal to occur, the morphological erosion mask must be at least as wide as the regions of false motion. If the mask is not wide enough, some noise will remain after erosion and will be expanded to its original size after dilation. This means that no noise will be removed. Because of this behavior, it is important that we use filters large enough to completely remove the expected noise. However, in order to reducing computational cost and preserving true motion, it is desirable to limit filtering to the minimum required. This motivates us to investigate the relationship of noise characteristics to filtering requirements.

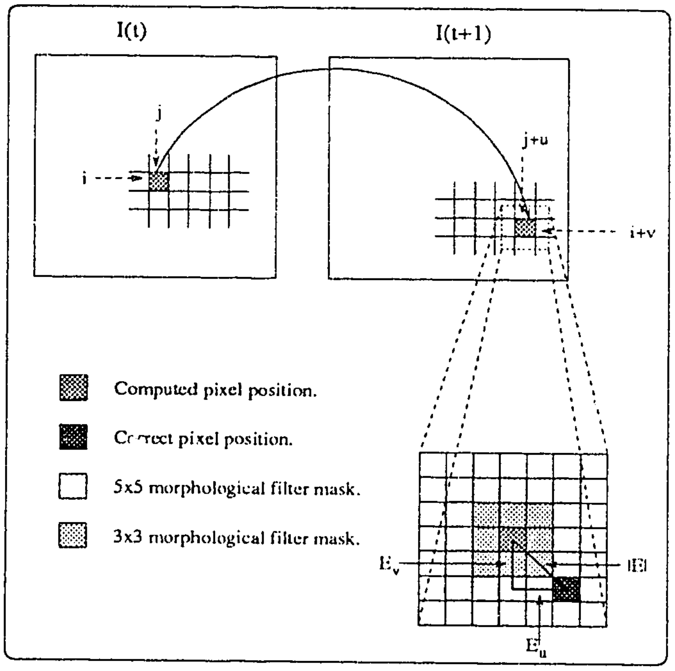

Figure 5.12: *The magnitude of error ($|E|$) in pixel-to-pixel mapping.*

5.7.1 Error in Position Readings

Recall that the displacement mapping functions in the translational case are

$$\begin{aligned} u_T &= x_{t+1} - x_t = \frac{S\cos(B)\sin(\tau) - x_t S\cos(\tau)\cos(B)}{Z} \\ v_T &= y_{t+1} - y_t = \frac{S\sin(\tau) - x_t S\cos(\tau)\cos(B)}{Z} \end{aligned}$$

The error between the correct pixel positions and the pixel position found with inaccurate position information in the mapping function can be expressed as

$$E_{u_T} = u_T(S + \Delta_S, B + \Delta_B) - u_T(S, B) \tag{5.21}$$

$$E_{v_T} = v_T(S + \Delta_S, B + \Delta_B) - v_T(S, B) \tag{5.22}$$

Where E_{u_T} and E_{v_T} are the errors in mapped pixel position in the x and y directions. Δ_B and Δ_S are inaccuracies in measurements of rotation and translation, respectively. For evaluating the error in pixel mapping, we consider several cases depending on the location of (x_t, y_t). In general, the error in the mapped pixel position is greater as we move further from the center of the image. We use pixel positions in the image center to simplify the error equations when determining general error characteristics, and consider border pixels to determine the worst case behavior.

To evaluate $u_T(S+\Delta_S, B+\Delta_B)$ and $v_T(S+\Delta_S, B+\Delta_B)$, we approximate them using a first order Taylor series expansion as follows:

$$u_T(S + \Delta_S, B + \Delta_B) = u_T(S, B) + \frac{\partial u_T}{\partial S}\Delta_S + \frac{\partial u_T}{\partial B}\Delta_B \tag{5.23}$$

$$v_T(S + \Delta_S, B + \Delta_B) = v_T(S, B) + \frac{\partial v_T}{\partial S}\Delta_S + \frac{\partial v_T}{\partial B}\Delta_B \tag{5.24}$$

Substituting Equations (5.23) and (5.24) into the equations for error in the compensated pixel position [Equations (5.21) and (5.22)] we obtain

$$E_{u_T} = \frac{\partial u_T}{\partial S}\Delta_S + \frac{\partial u_T}{\partial B}\Delta_B = \frac{\cos(\tau)[\Delta_S(B - x) + \Delta_B S]}{Z} \tag{5.25}$$

$$E_{v_T} = \frac{\partial v_T}{\partial S}\Delta_S + \frac{\partial v_T}{\partial B}\Delta_B = \frac{\Delta_S[\sin(\tau) - y\cos(\tau)]}{Z} \tag{5.26}$$

The magnitude of the error in pixel position due to translation (E_T) is:

$$|E_T| \;=\; \sqrt{E_{u_T}^2 + E_{v_T}^2} \tag{5.27}$$

$\lvert E_T \rvert$ in cm	Δ_S in cm	E_{v_T} in pixels	E_{u_T} in pixels
0.001788	0.127588	0.852204	0.523366
0.003577	0.255177	1.704407	1.046731
0.005366	0.382765	2.556611	1.570097
0.007154	0.510353	3.408814	2.093463
0.008942	0.637941	4.261017	2.616828
0.010731	0.765530	5.113221	3.140194
0.012520	0.893118	5.965424	3.663560
0.014308	1.020706	6.817628	4.186925
0.016097	1.148295	7.669832	4.710291
0.017885	1.275883	8.522035	5.233656
0.019674	1.403471	9.374239	5.757022
0.021462	1.531060	10.226443	6.280388

Table 5.1: *Worst-case compensation error for translation*

To determine the translation error Δ_S provided that there is no rotation (i.e., $B = \Delta_B = 0$), we substitute Equations (5.25) and (5.26) into Equation (5.27):

$$\Delta_S = \frac{E_T \; Z}{\sqrt{x^2 \cos(\tau)^2 + (\sin(\tau) - y\cos(\tau))^2}} \tag{5.28}$$

The error magnitude increases as a pixel (x, y) moves further from the center of the image and correspond to a 3-D point that is close to the camera. To estimate the worst-case error, we choose $(x, y) = (255, -240)$ — right bottom corner of the image. The tilt angle and the speed of the camera are set to $\tau = 17.4°$ and $S = 2$ (cm/time-unit). Rotation angle (B) is set to zero. Using Equation (5.28), we generated Table 5.1 for different error values in $|E_T|$. The values of E_{u_T} and E_{v_T} are also computed and shown in Table 5.1. Notice how the errors are distributed along the x and y axes, and observe the linear relationship between them and Δ_S.

5.7.2 Error in Pan Angle Variation

The mapping functions in the rotational case are:

$$\begin{aligned} u_R &= \frac{(f_x^2 + x^2)B}{f - xB} \\ v_R &= \frac{f_x xy}{f_x - xB} \end{aligned}$$

Similar to the analysis in the translational case, the errors in mapped pixel position in the x and y directions due to inaccuracies in measurement of the pan-rotation B are:

$$E_{u_R} = u_R(S + \Delta_S, B + \Delta_B) - u_R(S. B) \tag{5.29}$$

$$E_{v_R} = v_R(S + \Delta_S, B + \Delta_B) - v_R(S, B) \tag{5.30}$$

To determine the error, we again approximate the function with a first order Taylor series expansion and then substitute in Equations (5.29) and (5.30):

$$E_{u_R} = \frac{\partial u_R}{\partial S}\Delta_S + \frac{\partial u_R}{\partial B}\Delta_B = f_x \frac{x^2 + f_x^2}{(f_x - Bx)^2}\Delta_B \tag{5.31}$$

$$E_{v_R} = \frac{\partial v_R}{\partial S}\Delta_S + \frac{\partial v_R}{\partial B}\Delta_B = f_x \frac{yx}{(f_x - Bx)^2}\Delta_B \tag{5.32}$$

The magnitude of the error in pixel position due to rotation (E_R) is:

$$\begin{aligned} |E_R| &= \sqrt{E_{u_R}^2 + E_{v_R}^2} \\ &= \frac{\Delta_B f_x}{(f_x - Bx)^2}\sqrt{(x^2 + f_x^2)^2 + (xy)^2} \end{aligned} \tag{5.33}$$

For pan-only rotation, the error is predominantly in the x direction, since the change in the y component for pixels at different viewpoints is effected only by changes in perspective. Therefore, to determine the pan-angle error Δ_B, for a given pixel mapping error, from Equation (5.31) we obtain

E_{u_R} in cm	Δ_B in degrees	E_{v_R} in cm	$\|E_R\|$ in pixels
0.000894	0.020930	0.000032	0.500311
0.001788	0.041860	0.000063	1.000621
0.002683	0.062790	0.000095	1.500932
0.003577	0.083720	0.000126	2.001243
0.004471	0.104650	0.000158	2.501553
0.005366	0.125580	0.000189	3.001864
0.006260	0.146510	0.000221	3.502175
0.007154	0.167440	0.000252	4.002485
0.008048	0.188370	0.000284	4.502797
0.008942	0.209300	0.000315	5.003107
0.009837	0.230230	0.000347	5.503418
0.010731	0.251161	0.000378	6.003728

Table 5.2: *Pan-only compensation error*

$$\Delta_B = \frac{E_{u_R}(f_x - Bx_t)^2}{f_x(x_t^2 + f_x^2)} \tag{5.34}$$

Once Δ_B is determined, we can solve for E_{v_R} using Equation (5.34) to verify our initial assumption that E_{v_R} is negligible. For our system, where $f_x =$

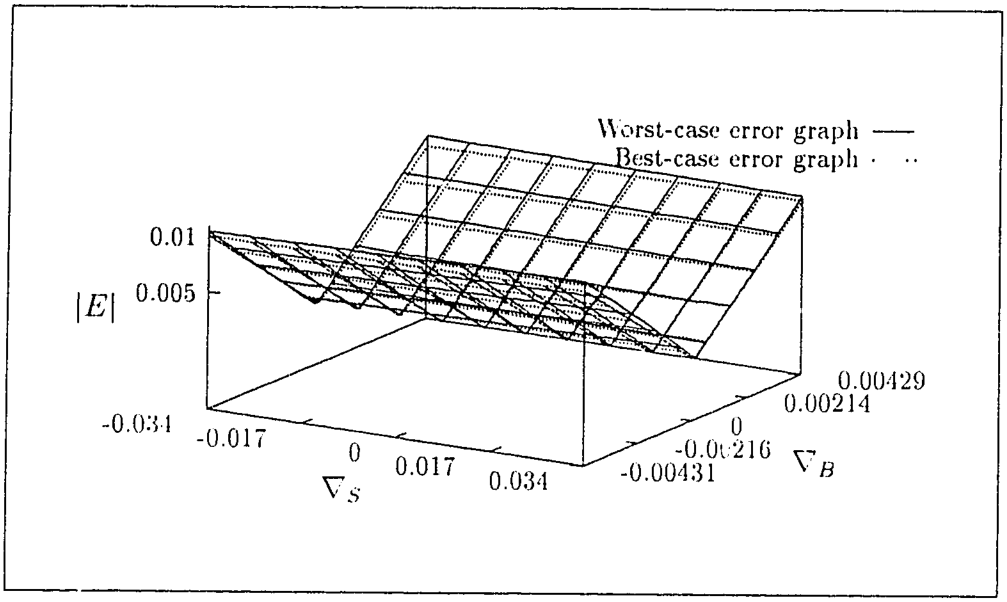

Figure 5.13: *Best-case and worst-case compensation error for translation*

2.31595 cm and the maximum $x_t = 255$ we can make a table of values of Δ_B for given errors in E_{u_R}. The corresponding E_{v_R} error for this position and $|E_R|$ are computed. The value for B used was $\equiv 2.5°$ (this is the value of the angle used in our experiments). The results are shown in Table 5.2; notice the linear relationship between Δ_B and E_{u_R}.

The magnitude of net error due to translation and rotation is

$$|E| = \sqrt{(E_{u_T} + E_{u_R})^2 + (E_{v_T} + E_{v_R})^2}$$

Figure 5.13 shows the effect of varying Δ_S and Δ_B on the computations of E_u and E_v. The two error-graphs depicted in the figure correspond to the worst-case and best-case error analysis. The worst-case is when $x = 255$ and $y = -244$ whereas the best-case is at the center of the image $((x, y) =$

$(0,0)$). As was noted earlier, the relationship between Δ_B, Δ_S, and $|E|$ is linear. In other words, the pixel mapping error is linearly dependent upon the magnitude of the error in angle and speed information. This analysis of error sources enable us to predict reasonable filter-mask sizes based on the information presented in Tables (5.1) and (5.2). Mask sizes of 5×5 (in the worst case) is proven to be sufficient to remove false motions in our particular system since error percentage (in Δ_B and Δ_S) is less than 10% of the actual readings (Figure 5.12).

5.8 Conclusion

We presented a direct solution for the problem of independent motion detection on planar surfaces from a moving camera (mounted on a mobile platform) using the background constraint. This technique exploits information about the camera motion in order to derive a mapping function relating pixels in two successive images, excluding certain peripheral regions. The basic idea is that any point in the image should satisfy the *background constraint* if it is static, whereas a point that lies on an independently moving object is unlikely to satisfy this constraint. Therefore, detecting moving objects becomes easy and can be used for both translation and rotation.

Since compensation, in general, is susceptible to errors caused by poor camera position information, morphological filters are employed to remove erroneously detected motion. While this method successfully removes false motion, it imposes an additional computational burden on the system that is proportional to the filter mask size. In our experiments, masks of small sizes (3×3 or 5×5) proved to be very effective in removing false motion. Note that the proposed algorithm for motion detection is inherently parallel, therefore a significant speed up of the computations involved in pixel-to-pixel mapping can be achieved.

Experimental results with real images were presented. Inaccuracy in position readings was taken into account while conducting all experiments. The results demonstrated the validity and the robustness of the method/algorithm based on background constraints.

Bibliography

[1] J. Aloimonos. Purposive and qualitative active vision. *Proceedings DARPA Image Understanding Workshop*, pages 816–828, 1990.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1:333–356, 1988.

[3] A. Basu and A. Elnagar. Safety optimizing strategies for local path planning in dynamic environments. Technical report, University of Alberta, 1993.

[4] R. Bhanu, P. Symosek, J. Ming, W. Burger, H. Nasr, and J. Kim. Qualitative motion detection and tracking. *Proceedings DARPA Image Understanding Workshop*, pages 370–398, 1989.

[5] A. Bruss and B. Horn. Passive navigation. *Computer Vision, Graphics and Image Processing*, 21(1):3–20, 1983.

[6] S. Carlsson and S. Eklundh. Object detection using model based prediction and motion parallax. *Proceedings of the first European Conference on Computer Vision*, pages 297–306, 1990.

[7] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. *Proceedings of the second European Conference on Computer Vision*, pages 178-202, 1992.

[8] F. Daily, M. Harris, and K. Reiser. detecting obstacles in range imagery. *Image Understanding Workshop*, pages 87-97, 1987.

[9] A. Elnagar and A. Basu. Heuristics for local path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 23(2):624-634, 1993.

[10] A. Elnagar and A. Basu. Smooth and acceleration minimizing trajectory planning for mobile robots. *The Proceedings of the IEEE/RSJ International Conference on Intelligent Robots*, pages 2215-2221, July 1993.

[11] W. Enkelmann. Obstacle detection by evaluation of optical flow fields from image sequences. *International Journal of Image and Vision Computing*, 9(3):160-168, 1991.

[12] C. Fermueller and J. Aloimonos. Tracking facilitates 3d motion estimation. *Biological Cybernetics*, (67):259-268, 1992.

[13] J. Frazier and R. Nevatia. Detecting moving objects from a moving platform. *Proceedings DARPA Image Understanding Workshop*, pages 348-355, 1990.

[14] E. Grosso, M. Tistarelli, and G. Sandini. Active-dynamic stereo for navigation. *Proceedings of the second European Conference on Computer Vision*, pages 178-202, 1992.

[15] D. Heeger and G. Hager. Egomotion and the stabilized world. *Proceedings of the 2nd International Conference on Computer Vision*, pages 435–440, 1988.

[16] B. Horn and B. Shunck. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[17] R. Jain. Segmentation of frame sequences obtained by a moving observer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):624–629, 1984.

[18] R. Jain, D. Militzer, and H. Nagel. Seperating non-stationary from stationary scene components in a sequence of real world TV images. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 425–428, 1977.

[19] A. Jepson and D. Heeger. A fast subspace algorithm for recovering rigid motion. *Proceedings of the IEEE Workshop on Visual Motion*, pages 124–131, 1991.

[20] K. Kanatani. Camera rotation invariance of image characteristics. *Computer Vision, Graphics, and Image Processing*, 39(3):328–354, September 1987.

[21] R. Kories and G. Zimmermann. A versatile method for estimation of displacement vector fields from image sequences. *Workshop on Motion: Representation and Analysis Charleston*, pages 101–106, 1986.

[22] D. Lawton. Motion analysis via local translational processing. *Proceedings of the IEEE Workshop on Computer Vision*, pages 59–72, 1982.

[23] H. Mallot, E. Schulze, and K. Storjohann. Neural network strategies for robot navigation. In *L. Personnaz and G. Dreyus, Neural Networks for models to Applications IDSET*, pages 560–569, 1989.

[24] D. Murray and A. Basu. Motion tracking with an active camera. *To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

[25] H. Nagel. Constraints for the estimation of displacement vector fields from image sequences. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 945–951, 1983.

[26] H. Nagel and W. Enkelmann. An invistigation of smoothness constraints for estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, 1986.

[27] S. Negahdaripour and S. Lee. Motion recovery from image sequences using first-order optical flow information. *Proceedings of the IEEE Workshop on Visual Motion*, pages 132–139, 1991.

[28] R. Nelson. Qualitative detection of motion by a moving observer. In *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 173–178, Maui, Hawaii, USA, June 1991.

[29] R. Nelson and J. Aloimonos. Using flow field divergence for obstacle avoidance in visual navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1102–1106, 1989.

[30] K. Olin, F. Vilnrotter, F. Daily, and K. Reiser. Development in knowledge-based vision for obstacle detection and avoidance. *Image Understanding Workshop*, pages 78–86, 1987.

[31] B. Shunck. Motion segmentation by constraint line clustering. *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, pages 58–62, 1984.

[32] M. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990.

[33] V. Sundareswaren. Egomotion from global flow field data. *Proceedings of the IEEE Workshop on Visual Motion*, pages 140–145, 1991.

[34] W. Thompson and J. Kearney. Inexact vision. *Proceedings Workshop on Motion: Representation and Analysis*, pages 15–21, 1986.

[35] W. Thompson, P. Lechleider, and E. Stuck. Detecting moving objects using the rigidity constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):162–166, 1993.

[36] W. Thompson and T. Pong. Detecting moving objects. *International Journal of Computer Vision*, 4(1):39–58, 1990.

[37] M. Tistarelli and G. Sandini. Dynamic aspects in active vision. *Computer Vision, Graphics and Image Processing*, 56(1):108–129, 1992.

[38] R. Tsai and T. Huang. Estimating 3D motion parameters of a rigid planar patch, i. *IEEE Transactions on Acoust., Speech, Single Processing*, ASSP-29:1147–1152, 1981.

[39] R. Tsai and T. Huang. Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):13–27, 1984.

[40] S. Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, MA, 1979.

[41] D. Willick and Y. Yang. Experimental evaluation of motion constraint equations. Technical Report TR 89-4, University of Saskatchewan, 1989.

[42] Z. Zhang, O. Faugeras, and N. Ayache. Analysis of a sequence of stereo scenes contatining multiple moving objects using rigidity constraints. *Proceedings of the 2nd International Conference on Computer Vision*, pages 177–186, 1988.

Chapter 6

General Discussion and Conclusions

In this chapter, we summarize the contributions of the dissertation and show the interrelationships between chapters. As well, we discuss possible future research directions.

6.1 Contributions of the Research

In this dissertation we have presented algorithms, heuristics, a planning methodology, and a motion detection technique for a vision-based navigation system. Our development of these component modules has helped us understand the issues involved in designing such a system. There are two major topics that have been addressed throughout the dissertation: motion planning and motion detection.

In the first stage of our research (Chapter 2) we presented a new approach with heuristics to solve the problem of path planning using only local information and a knowledge of the start and goal positions. The experimental results showed how heuristics help in enhancing the search process, and the robot's ability to avoid obstacles. The concept of safety was introduced. Safety is a function of the speed of the vehicle with the acceleration bounds being the parameters. In order to draw the robot closer to the goal, a goal attraction function was used. To avoid obstacles, we proposed two special-purpose heuristics: the "dead-end" and the "avoid-region". Both heuristics were used to prune the search space and to enhance the ability of the mobile robot to avoid obstacles. This technique was shown to be less susceptible to the local minima problem than most potential field methods by using a heuristic that allowed the robot to follow obstacle boundaries.

The path obtained from the planning algorithm (in Chapter 2) is described by a sequence of line segments. This is not a desirable characteristic of any planning approach in real-life navigation because the path might be impossible to follow (e.g., zigzag path). Therefore, the smoothness of paths should be taken into account[1]. We presented a new approach to generate piecewise smooth trajectories for mobile robots in a local environment, which minimizes the integral of the acceleration (tangential and normal). It was shown how equality and inequality constraints can be taken into account in the minimization problem to avoid nearby obstacles. Unfortunately, the solution of nonlinear systems is complex and not suitable for real-time imple-

[1] In some cases, smooth paths are essential to prevent wheel slippage.

mentation, two other techniques were used, namely, cubic and Bezier curves. Each of these curves generated a smooth and safe trajectory once a collision was detected with the original smooth trajectory.

In the basic problem of motion planning we assumed that obstacles were fixed. One obvious extension is to remove this assumption. The problem then becomes much harder and it can no longer be solved by merely constructing a geometric path. Instead, a continuous function of time specifying the robot's configuration space at each instant of time must be generated. Therefore, we presented a new method for dynamic path planning using only local information and a knowledge of the start and goal positions. Uncertainty in obstacle positions was also taken into account. The concept of safety was used, static safety being a function of the speed of the robot. Dynamic safety was described as a function of the time-to-collision with an obstacle. In order to draw the robot closer to the goal, a goal attraction function was used. We showed that the velocity-decomposition technique is a special case of the overall safety function when the direction of motion of the robot is fixed in a planning step.

In the treatment of the motion planning problem, we assumed that the robot senses its environment through some sort of a sensory device. A camera is used to accomplish this task, especially to detect moving obstacles in its field of view. We presented a direct solution for the problem of independent motion detection on planar surfaces from a moving camera (mounted on a mobile platform) using the background constraint. This technique exploits information about the camera motion in order to derive a mapping function relating pixels in two successive images, excluding certain peripheral regions.

The basic idea is that any point in the image should satisfy the *background constraint* if it is static, whereas a point that lies on an independently moving object is unlikely to satisfy this constraint. Therefore, detecting moving objects becomes easy and can be used for both translation and rotation.

Since compensation, in general, is susceptible to errors caused by poor camera position information, morphological filters are employed to remove erroneously detected motion. While this method successfully removes false motion, it imposes an additional computational burden on the system that is proportional to the filter mask size. In our experiments, masks of small sizes (3×3 or 5×5) pixels proved to be very effective in removing false motion. It should be noted that the proposed algorithm for motion detection is inherently parallel, therefore a significant speed up of the computations involved in pixel-to-pixel mapping can be achieved.

Experimental results with real images were presented. Inaccuracy in position readings was taken into account while conducting all experiments. The results demonstrated the validity and the robustness of the method/algorithm based on background constraints.

This completes the study of the basic components of an autonomous navigation system: motion planning and motion detection. A motion planning methodology was discussed in Chapters 2, 3, and 4 whereas a novel technique for detecting motion was described in Chapter 5.

6.2 Future Research Directions

Our work gives rise to a number of possible research problems for future investigation:

- Refinement of the planning algorithm. We have considered two types of obstacles: polygons and circles. An extension is to include arbitrarily-shaped obstacles. This can be achieved by either developing a geometric planning technique that takes into account these obstacles or, from computational geometry, exploring algorithms that can contain any arbitrary-shaped object in a minimum circle or ellipse.

- Proposing heuristics to search the robot's configuration space for a free path in a time-varying environment.

- Planning shortest-distance paths or time-minimal paths in dynamic environments.

- Smoothness of trajectories generated in global path planning where a complete knowledge of the environment is available. This can be achieved by applying constraint optimization techniques.

- Proposing numerical procedures for computing piecewise linear approximation of optimal trajectories as a solution of discrete two-point boundary value problems (e.g., nonlinear equations described in Chapter 3).

- Generalizing the planning algorithm to deal with 3D motion. For example, planning collision-free paths for a manipulator in a dynamic environment. The manipulator might be equipped with a dextrous multi-joint multi-finger hand allowing complex types of motions within the hand. Optimization theory can be used to optimize some performance criterion (e.g., minimize the total amount of time needed to perform a task).

- The ability to recover robust spatial description from sensory information and to efficiently utilize these descriptions in planning algorithms is a crucial requirement in an autonomous vehicle system. Therefore, a technique for sensor data interpretation (e.g., building depth maps or occupancy grid maps) of the environment is needed.

- Throughout our research we were interested in controlling one robot in the workspace. What if the workspace is populated with more than one robot? How to coordinate motion between them? How can they be used to achieve a specific task? This is a topic of great importance and needs to be investigated in detail in order to answer the above questions.

- Further developments are necessary to extend our proposed approach for detecting motion into more complex environments. For example, motion detection on terrain surfaces or in 3D.

- One important issue in motion planning and detection is the computational complexity (time and space) of the algorithms. Analysis of the complexity of navigation algorithms is essential before the design of operational systems. Another topic of interest is parallelism which could be used to enhance the processing-speed of these algorithms.

- A full implementation of the theory introduced in this dissertation on a platform robot needs to be done. Of course, there are some other problems that should be considered. For instance, localization, locomotion, world modeling (maps), and computational architectures.

Appendix A

Derivation of the Safety Function

$$\begin{aligned}
f(s) &= A + B \\
&= \frac{1}{S\Pi}\int_0^s \int_0^\Pi I(s\frac{\theta}{\Delta T} \leq A_N)\ \ I(\frac{(s-s_f)}{\Delta T} \leq A_T)\ \ d\theta ds_f \\
&+ \frac{1}{S\Pi}\int_s^S \int_0^\Pi I(s_f\frac{\theta}{\Delta T} \leq A_N)\ \ I(\frac{(s_f-s)}{\Delta T} \leq A_T)\ \ d\theta ds_f
\end{aligned} \tag{A.1}$$

Let us first solve A:

$$\begin{aligned}
A &= \frac{1}{S\Pi}\int_0^s \int_0^\Pi I(s\frac{\theta}{\Delta T} \leq A_N)\ \ I(\frac{(s-s_f)}{\Delta T} \leq A_T)\ \ d\theta ds_f \\
&= \frac{1}{S\Pi}\int_0^s I(\frac{(s-s_f)}{\Delta T} \leq A_T)\ \ (\int_0^\Pi I(s\frac{\theta}{\Delta T} \leq A_N)\ \ d\theta)ds_f \\
&= \frac{1}{S\Pi}\ A_1\ A_2
\end{aligned} \tag{A.2}$$

$$A_2 \ = \ \int_0^\Pi I(s\frac{\theta}{\Delta T} \leq A_N)\ \ d\theta \ = \ \int_0^\Pi I(\frac{A_N \Delta T}{s} \geq \theta)\ \ d\theta$$

To solve A_2, we have to consider two cases :

- If $\frac{A_N \Delta T}{s} \geq \Pi$

$$\int_0^{\Pi} I(\frac{A_N \Delta T}{s} \geq \theta) \ d\theta \ = \ \Pi \tag{A.3}$$

- If $\frac{A_N \Delta T}{s} < \Pi$

$$\int_0^{\frac{A_N \Delta T}{s}} I(\frac{A_N \Delta T}{s} \geq \theta) \ d\theta \ = \ \frac{A_N \Delta T}{s} \tag{A.4}$$

$$A_1 \ = \ \int_0^s I(\frac{(s - s_f)}{\Delta T} \leq A_T) \ ds \ = \ \int_0^s I(s_f \geq s - A_T \Delta T) \ ds_f$$

Like A_2, we have to consider two cases to solve A_1:

- If $s \leq A_T \Delta T$

$$\int_0^s I(s_f \geq s - A_T \Delta T) \ ds_f \ = \ s \tag{A.5}$$

- If $s > A_T \Delta T$

$$\int_{s - A_T \Delta T}^s I(s_f \geq s - A_T \Delta T) \ ds_f \ = \ A_T \Delta T \tag{A.6}$$

To compute A, we should study all possible combinations of (A.3) and (A.4) with (A.5) and (A.6), as follows:

$$A = \begin{cases} \frac{s}{S} & \text{if (A.3)\&(A.5)} \\ \frac{A_T \Delta T}{S} & \text{if (A.3)\&(A.6)} \\ \frac{A_N \Delta T}{S\Pi} & \text{if (A.4)\&(A.5)} \\ \frac{A_N \Delta T A_T \Delta T}{sS\Delta T} & \text{if (A.4)\&(A.6)} \end{cases}$$

Similarly, B in (A.1) can be computed as follows :

$$\begin{aligned}
B &= \frac{1}{S\Pi}\int_s^S \int_0^\Pi I(s_f\frac{\theta}{\Delta T} \leq A_N)\ I(\frac{(s_f - s)}{\Delta T} \leq A_T)\ d\theta ds_f \\
&= \frac{1}{S\Pi}\int_s^S I(\frac{(s_f - s)}{\Delta T} \leq A_T)\ (\int_0^\Pi I(s_f\frac{\theta}{\Delta T} \leq A_N)\ d\theta)ds_f \\
&= \frac{1}{S\Pi}\ B_1\ B_2 \qquad (A.7)
\end{aligned}$$

Since the computation of B_2 affects the computation of B_1, the final result of B_1 will take the form, $B_1 = B_{11} + B_{12}$, depending on the intervals where the speed is integrable.

Computing B_1 and B_2 follows as in A, resulting in the following :

$$B_1 = \begin{cases} \frac{A_N\Delta T}{S\Pi} - \frac{s}{S} & \text{if } s \leq \frac{A_N\Delta T}{\Pi} \leq s + A_T\Delta T \\ 1 - \frac{s}{S} & \text{if } s \leq S \leq s + A_T\Delta T \leq \frac{A_N\Delta T}{\Pi} \\ \frac{A_T\Delta T}{S} & \text{if } s \leq s + A_T\Delta T \leq \frac{A_N\Delta T}{\Pi} \end{cases}$$

$$B_2 = \begin{cases} \frac{A_N\Delta T}{S\Pi}(log(S) - log(s)) & \text{if } \frac{A_N\Delta T}{\Pi} \leq s \leq S \leq s + A_T\Delta T \\ \frac{A_N\Delta T}{S\Pi}(log(S) - log(\frac{A_N\Delta T}{\Pi})) & \text{if } s \leq \frac{A_N\Delta T}{\Pi} \leq S \leq s + A_T\Delta T \\ \frac{A_N\Delta T}{S\Pi}(log(s + A_T\Delta T) - log(s)) & \text{if } \frac{A_N\Delta T}{\Pi} \leq s \leq s + A_T\Delta T \leq S \\ \frac{A_N\Delta T}{S\Pi}(log(s + AT\Delta T) - log(\frac{A_N\Delta T}{\Pi})) & \text{if } s \leq \frac{A_N\Delta T}{\Pi} \leq s + A_T\Delta T \leq S \end{cases}$$

To compute $f(s)$ we should consider all combinations of A, B_1, and B_2. There are 16 different combinations. As an example, 4 of them are listed as follows :

1. if $s \leq \frac{A_N\Delta T}{\Pi} \leq s + A_T\Delta T \leq S$ and $s \leq A_T\Delta T$

$$f(s) = \frac{A_N\Delta T}{S\Pi}(1 + log(s + A_T\Delta T) - log(\frac{A_N\Delta T}{\Pi})) \quad (A.8)$$

2. if $\frac{A_N \Delta T}{\Pi} \leq s \leq s + A_T \Delta T \leq S$ and $s \leq A_T \Delta T$

$$f(s) \;=\; \frac{A_N \Delta T}{S\Pi}(1 + log(s + A_T \Delta T) - log(s)) \qquad (A.9)$$

3. if $\frac{A_N \Delta T}{\Pi} \leq s \leq S \leq s + A_T \Delta T$ and $s \leq A_T \Delta T$

$$f(s) \;=\; \frac{A_N \Delta T}{S\Pi}(1 + log(s + A_T \Delta T) - log(s)) \qquad (A.10)$$

4. if $\frac{A_N \Delta T}{\Pi} \leq s \leq S \leq s + A_T \Delta T$ and $s > A_T \Delta T$

$$f(s) \;=\; \frac{A_N \Delta T}{S\Pi}(\frac{A_T \Delta T}{s} + log(S) - log(s)) \qquad (A.11)$$

Equations (A.8-A.11) show various forms of $f(s)$ for different values of s under the above conditions.

Appendix B

Proof of Proposition 2.4.1

Proposition 2.4.1

The safety function f increases monotonically to a peak and then decreases monotonically. In the degenerate cases where the peak is at the left (or right) boundary, f is monotonically decreasing (or increasing).

Proof: We will prove the monotonicity for only one of the sixteen different forms of $f(s)$. The proof is similar for the other cases. Consider the functions of $f(s)$ that we introduced in functions (A.8) through (A.11) in Appendix A. To prove the monotonicity, it is necessary to study the existence of the critical points of the functions (A.8-A.11). From basic calculus it follows:

1. if $s \leq \frac{A_N \Delta T}{\Pi} \leq s + A_T \Delta T \leq S$ and $s \leq A_T \Delta T$

$$f'(s) = \frac{A_N \Delta T}{S\Pi}(\frac{1}{s + A_T \Delta T}) \qquad \text{(B.1)}$$

2. if $\frac{A_N\Delta T}{\Pi} \leq s \leq s + A_T\Delta T \leq S$ and $s \leq A_T\Delta T$

$$f'(s) = \frac{A_N\Delta T}{S\Pi}(\frac{1}{s + A_T\Delta T} - \frac{1}{s}) \qquad \text{(B.2)}$$

3. if $\frac{A_N\Delta T}{\Pi} \leq s \leq S \leq s + A_T\Delta T$ and $s \leq A_T\Delta T$

$$f'(s) = \frac{A_N\Delta T}{S\Pi}(\frac{1}{s + A_T\Delta T} - \frac{1}{s}) \qquad \text{(B.3)}$$

4. if $\frac{A_N\Delta T}{\Pi} \leq s \leq S \leq s + A_T\Delta T$ and $s > A_T\Delta T$

$$f'(s) = \frac{A_N\Delta T}{S\Pi}(\frac{-A_T\Delta T}{s^2} - \frac{1}{s}) \qquad \text{(B.4)}$$

where $s > 0$. Assume that the speed intervals I_1, I_2, I_3, and I_4 define the ones appear above, respectively. Studying the above derivatives shows the following results:

- $f(s)$ is increasing on I_1 since $f'(s) > 0$, $\forall s \in I_1$.
- $f(s)$ is decreasing on I_2, I_3, and I_4 since $f'(s) < 0$, $\forall s \in I_2$, I_3, and I_4, respectively.

The above results indicate the possibility of having a critical point between (A.8) and (A.9),or (A.8) and (A.10), or (A.8) and (A.11). It is a local maximum ($s = \frac{A_N\Delta T}{\Pi}$), and it is unique (Figure 2.1). Similarly, we can study the monotonicity for other cases.

Intuitive meaning : Proposition 2.4.1 describes the relation between the safety function and the speed. For example, if we assume that there is no normal acceleration bound, then the graph of the safety function will be symmetric. Moreover, the safety value will correspond to either a peak or a plateau. In the case of a plateau, the safest speed would be in the interval $[(S - A_T\Delta T), A_T\Delta T]$. Otherwise, when $A_T\Delta T = S - A_T\Delta T$, the safest speed is given by $A_T\Delta T$. When the normal acceleration bound is introduced, the graph of the safety function will not be symmetric (Figure 2.1). Instead, the peak will be skewed to the left. This is because the normal acceleration bound prefers lower speeds as is clear from its definition (curvature multiplied by the square of the speed).

Appendix C

Times of Intersection Between a Curve and an Obstacle

We outline the computation of times of intersections between a smooth curve and an obstacle. Let τ be a curve intersecting an obstacle centered at (x_c, y_c) with radius r. The curve and the obstacle equations are:

$$y = x \tan \phi(t) \tag{C.1}$$

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \tag{C.2}$$

Assume that $x_0 = 0$ and $y_0 = 0$, then equations (3.12) and (3.13) become:

$$x = s(t)t \cos(\phi(t)) \tag{C.3}$$

$$y = s(t)t \sin(\phi(t)) \tag{C.4}$$

To find points of intersection, we substitute (C.1) in (C.2) and then solve for

x, which yields:

$$x = \frac{(x_c + y_c \tan(\phi)) \pm \sqrt{(x_c + y_c \tan\phi)^2 - \sec^2\phi(x_c^2 + y_c^2 - r^2)}}{sec^2\phi} \tag{C.5}$$

Equating (C.3) and (C.4) produces:

$$s(t)t \;=\; (x_c \cos(\phi) + y_c \sin(\phi) \pm \sqrt{-(\pm x_c \sin(\phi) \mp y_c \cos(\phi))^2 + r^2}$$

The solution of x depends on the value under the square root. We consider the following cases:

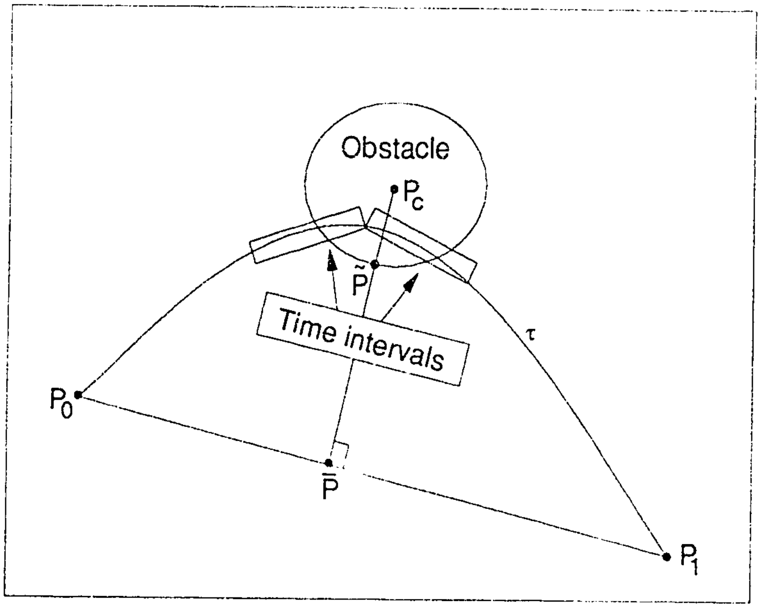

Figure C.1: *Intersection-time intervals.*

- If $x_c \sin(\phi) = y_c \cos(\phi)$ then

$$s(t)t \;=\; [x_c + \frac{y_c^2}{x_c}]\cos(\phi) \pm r \tag{C.6}$$

- If $(\pm x_c \sin(\phi) \mp y_c \cos(\phi)) = r$ then

$$s(t)t \quad = \quad [x_c + \frac{y_c^2}{x_c}]\cos(\phi) + \frac{y_c}{x_c}r \tag{C.7}$$

From principles of infinite series, we may represent $\cos(\phi(t))$ as a polynomial of $\phi(t)$ depending on certain error value. Since φ is a function of t, we can simply solve for times of intersection. Because of the nature of this particular problem we will obtain intervals of intersections, instead of points of intersection. However, we do not need to find points of intersection, but rather we need to know if an intersection occurs. This is achieved by checking the intersection time intervals. Figure C.1 illustrates this situation.

Appendix D

Computing $\tilde{P}$

Since the closest point ($\dot{P}$) on the circle from $\overline{P_0P_1}$ is the one that forms the minimum distance from $\overline{P_0P_1}$ (perpendicular distance). From Figure C.1, it follows that the equation of the line $\overline{P_c\dot{P}}$ is:

$$y = \begin{cases} y_c - \frac{1}{m}(x - x_c) & \text{if } x_1 \neq x_0 \\ y_c & \text{otherwise} \end{cases}$$

where the slope of $\overline{P_0P_1}$ is:

$$m = \frac{y_1 - y_0}{x_1 - x_0}; \quad x_1 \neq x_0$$

Substituting y in (C.2), and solve for x, two points are obtained. By a simple distance test we can find $\tilde{P}$. $\tilde{\phi}$ can be determined from (C.1), and $\tilde{t}$ from Equation 3.9. The speed at this point can be computed by Equation 3.7.

Appendix E

Constraints on the Height

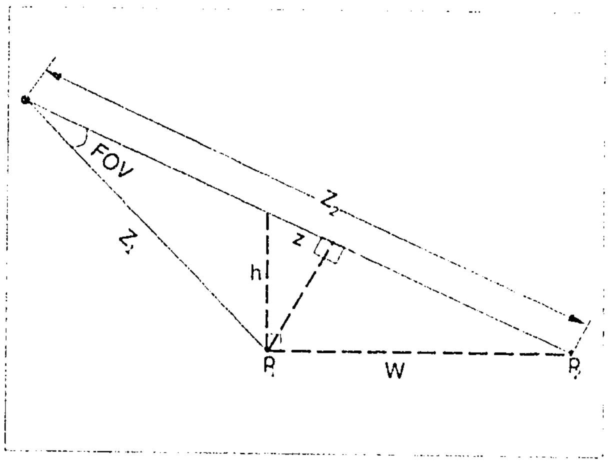

Figure E.1: *A simple geometry to derive the constraints.*

Our aim is to derive some constraints that can be placed on the heights of objects inside a certain scene. Figure E.1 describes a simple geometry of the closest (p_1) and farthest (p_2) points on the background that can be viewed by the camera at a certain instant of time. The distances Z_1 and Z_2 represent the distances between these points (p_1 and p_2) and the camera, respectively. The height of an object that blocks the camera's field of view (θ) at p_1 is denoted by h. W is the width of the field of view. From Figure E.1, h can be estimated in two different ways:

$$h = \sqrt{Z_1^2 \sin^2(\theta) + z^2} \tag{E.1}$$

$$h = \sqrt{(z + \sqrt{W^2 - Z_1^2 \sin^2(\theta)})^2 - W^2} \tag{E.2}$$

Equating both Equations above (E.1 and E.2), we obtain:

$$z = \frac{Z_1^2 \sin^2(\theta)}{Z_2 - Z_1 \cos(\theta)}, \quad Z_2 \neq Z_1 \cos(\theta) \tag{E.3}$$

Consequently from Equation E.1,

$$h = Z_1 \sin(\theta) \sqrt{1 + \frac{Z_1^2 \sin^2(\theta)}{(Z_2 - Z_1 \cos(\theta))^2}}, \quad Z_2 \neq Z_1 \cos(\theta) \tag{E.4}$$

The height of an object at p_1 that blocks the camera's field of view can be determined by the above relation provided that Z_1, Z_2 , and θ are given.

Formally if we let the right hand side of Equation E.4 to be defined as a function of θ ($F(\theta)$), then we can place the constraints as follows:

$$0 < h \leq F(\varphi), \qquad \text{where } \phi \in (0, \alpha * \theta]$$

α is a constant chosen empirically in the interval $(0, 1)$.