2011

A Motion Planner For Robot Manipulators Based on Support Vector Machines

Mahdi Anooshahpour

A Motion Planner For Robot Manipulators Based on Support Vector Machines

(Spine title: Motion Planning of Manipulators Using Support Vector Machine)

(Thesis format: Monograph)

by

Mahdi Anooshahpour

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....
Dr. Mehrdad R. Kermani

Supervisory Committee:

Examiners:

.....
Dr. Ilia Polushin

.....
Dr. Michael D. Naish

.....
Dr. Steven S. Beauchemin

The thesis by

Mahdi Anooshahpour

entitled:

A Motion Planner For Robot Manipulators Based on Support Vector Machines

is accepted in partial fulfillment of the
requirements for the degree of
Master of Engineering Science

.....
Date

.....
Chair of the Thesis Examination Board

Abstract

Moving a robot between two configurations without making a collision is of high importance in planning problems. Sampling-based planners have gained popularity due to their acceptable performance in practical situations. This body of work introduces the notion of a risk function that is provided using the *Support Vector Machine (SVM)* algorithm to find safe configurations in a sampled configuration space. A configuration is called safe if it is placed at maximum distance from surrounding obstacle samples. Compared to previous solutions, this function is less sensitive to a selected sampling method and resolution. The proposed function is first used as a repulsive potential field in a local SVM-based planner. Afterwards, a global planner using the notion of the risk function is suggested to address some of the shortcomings of the suggested local planner. The proposed global planner is able to solve a problem with fewer number of milestones and less number of referrals to the collision detection module in comparison to the classical *Probabilistic Roadmap Planner (PRM)*. The two proposed methods are evaluated in both simulated and experimental environments and the results are reported.

Keywords: Motion Planning, Artificial Potential Field, Narrow Passages, Probabilistic Roadmap Planner, Support Vector Machine, Sampling-based Motion Planning, Obstacle-based Motion Planning

Acknowlegements

I would like to gratefully and sincerely thank Dr. Mehrdad Kermani for his support, guidance and encouragement throughout the research. His profound comments and advices led me to the right way.

I am also indebted to my colleague and best friends Peyman Yadmellat , Amir Takhmar, Kamal Mostafavi, Vahid Sotoudehnejad, Nima Najmaei, and especially my brother Farshad Anooshahpour for their help, thoughtful ideas and constructive comments.

Finally, and most importantly, I would like to express my deepest gratitude to my parents. I would not be able to complete this dissertation without their love and everlasting support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

APF	Artificial Potential Field
DoF	Degrees of Freedom
DH	Denavit-Hartenberg
OVR	One Versus Rest
PRM	Probabilistic Roadmap Planner
RBF	Radial-Basis Gaussian Function
RRT	Rapidly-Exploring Random Trees
SVM	Support Vector Machine
SVMP	Support Vector Machine-Based Planner

List of Symbols

$\mathcal{A}$	Set of points occupied by a robot in the workspace
b	Bias term
C	Penalty parameter
$\mathcal{D}$	Input dataset
$\mathcal{E}$	Set of edges of a graph
F	Decision function
$\mathcal{G}$	Graph
K	Kernel function
L	Lagrange function
L_d	Lagrange dual function
M	Margin
O	Set of obstacle points in the workspace
$\mathcal{P}$	Path containing a series of configuration points
P	Labeler hyperplane
q	Configuration point
Q	Configuration space
Q_{free}	Free configuration space
Q_{obst}	Set of obstacle points in the configuration space
R	Risk function
$\mathcal{R}$	The set of real numbers
S_{manip}	Set of control points of a manipulator
S_{obst}	Set of workspace obstacle samples
$\mathcal{V}$	Set of vertices of a graph
$\mathcal{W}$	Workspace
$U_{attract}$	Attractive potential function
$\mathbf{w}$	Weights of a hyperplane (decision function)
$\mathbf{x}$	Vector representing an input point
y	Label value
α	Lagrangian multiplier
ζ	Misclassification error
μ	Attractive potential gain
σ	RBF-kernel parameter
ϕ	Mapping function
$\mathbf{\Phi}$	Vector-valued mapping function

Chapter 1

Introduction and Literature Review

1.1 Introduction

The increasing application of robotic systems, has raised new challenges and introduced new fields of study. Mechanical design, control, sensors, interface, and planning are some of the issues in designing a robotic system. Planning is a process that "convert high-level specifications of tasks from humans into low-level description of how to move" [21]. The very first task in this conversion is composing robot movements as a number of configurations. Assuming two initial and goal configurations are specified, the robot must be able to start from the initial and reach the goal configuration in an environment safely. Path planning algorithms are intended to solve this kind of problem in different circumstances.

Path planning is the task of finding a collision-free path, given geometry of a robot and environment's obstacles. The *Piano mover's problem* is the classic example of path planning. The problem is defined as determining a way to carry a piano between two rooms without hitting any articles. Even though this seems easy, solving this problem is computationally difficult [34]. Complex robot structures such as those of high degree of freedom manipulators, complex obstacles in the environment, capability of working in real-time in dynamic environments, etc., make the problem even more difficult. "Maintenance of cooling pipes in a nuclear plant, point-to-point welding in car assembly, and cleaning of airplane fuselages" are some of the applications in which a reliable planner is able to facilitate robot programming task [14].

This work investigates the problem of finding a path and suggests a solution for high degrees freedom robots in complex static environments. The notion of static environment means that the shape and position of obstacles are known. This chapter starts with defining the problem (Section 1.2) followed by reviewing previous works (Section 1.3). Section 1.4 lays down the objectives of this body of work. Section 1.5 summarizes the main contributions of this work. Finally Section 1.6 plots the thesis outline.

1.2 Problem Definition

Let $\mathcal{W}$ describe the workspace which is a subset of $\mathcal{R}^2$ or $\mathcal{R}^3$. The set O refers to a subset of $\mathcal{W}$ occupied by obstacles and Q composes the robot's configuration space with dimensions equal to the number of controllable degrees of freedom. In the configuration space, a robot separated from its complex structure is represented by a point. If $\mathcal{A}(q_0) \subset \mathcal{W}$ represents the robot at the configuration $q_0 \in Q$ in the workspace then $Q_{obst} = \{q \mid \mathcal{A}(q) \cap O \neq \emptyset\}$ describes the obstacles in the configuration space. The complement of the set Q_{obst} is $Q_{free} = Q - Q_{obst}$ which contains all collision-free configurations that can be used for generating a valid path.

Now, given O, initial (q_{init}) and goal (q_{goal}) configurations, the solution is defined as follows,

$$\mathcal{P} = \{q_{p_0}, \ldots, q_{p_d}\} \tag{1.1}$$

in which $q_{p_0} = q_{init}$ and $q_{p_d} = q_{goal}$. This set is a feasible solution if all configurations in $\mathcal{P}$ are collision-free and moving between two subsequent configurations without making any collision is a trivial task.

The objective of this body of work is to develop a planner for high degree of freedom robotic manipulators. This planner is intended to find a path in the configuration space so as a manipulator can move between two given configurations safely. Speed and generality are two of the features that this planner is expected to achieve. Although the planner is intended to be fast, working in real-time is not required.

1.3 Literature Review

There are many algorithms that tackle this problem from different perspectives. These algorithms can be broadly categorized into three classes based on their completeness. The three class of planners are Exact, Sampling-Based, and Local planners. This section discusses the major contributions in each category and talks about their advantages and shortcomings.

1.3.1 Exact Planners

Most of the planners in this category are of theoretical interest [34] and not applicable to real situations. A common characteristic of these planners is their need for an exact and explicit representation of obstacles in the robot's configuration space. Solving problem in the configuration space is attractive since any complex robot would be described by a point in its configuration space. In order to find an explicit knowledge about the configuration space, some researchers looked at the problem geometrically and analytically. They have proposed descriptive constraints to determine free configuration regions in the configuration space [24, 27]. Assuming

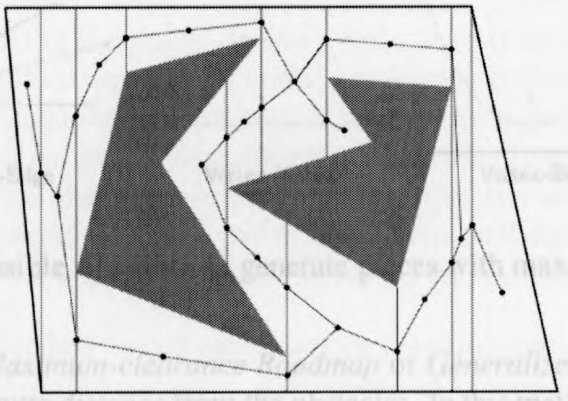


Figure 1.1: Vertical Cell Decomposition [21].

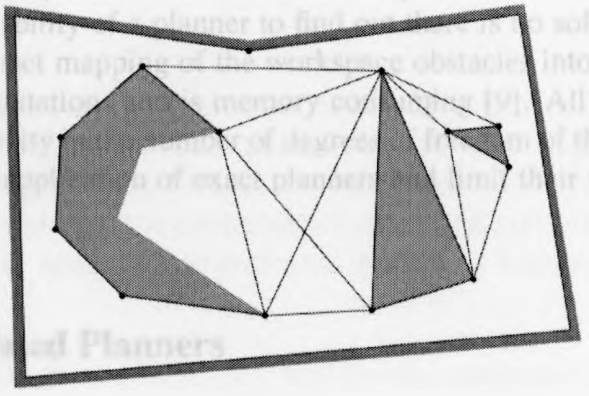


Figure 1.2: Shortest Path Roadmap [21].

complete knowledge of the obstacles is provided in the configuration space, different solutions have been suggested.

The first technique in this category is *Cell Decomposition*. The main idea behind the cell decomposition-based techniques is to divide the robot's configuration space into a number of cells in a way that planning in each cell is trivial [21]. These cells and their boundaries are stored as nodes of a graph. Each boundary node is connected to all adjacent cell nodes. Therefore, the path planning problem turns into a problem of searching for a path in a graph. *Vertical Cell Decomposition* method constructs a number of cells by sweeping the space and determining a boundary at each obstacle vertex. In this method, the graph nodes are placed at the center of each cell and on each boundary [8] (Fig. 1.1).

Shortest Path Roadmap (also called *Reduced Visibility Graph*) puts nodes on each obstacle vertex with the interior angle less than π. Next, each pair of the nodes that are mutually visible to each other will be connected together. To answer a query, the initial and goal configuration points find their way to their nearest nodes and the rest of the path is found by searching the graph [34] (Fig. 1.2).

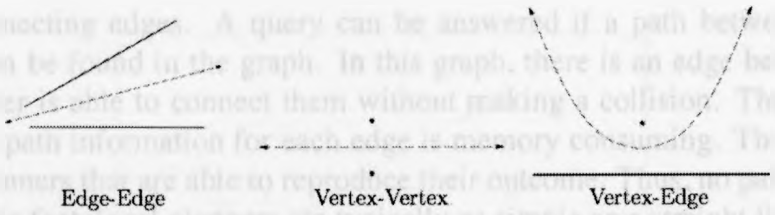


Figure 1.3: Three possible situations to generate pieces with maximum distance [21].

Another idea known as *Maximum-clearance Roadmap* or *Generalized Voronoi Diagram* tries to keep the robot in maximum distance from the obstacles. In this method a roadmap is created based on three types of paths which are shown in Fig. 1.3 [28].

All these methods are complete which means that they are able to report failure or success. Reporting failure is the ability of a planner to find out there is no solution for the problem in finite time. However, exact mapping of the workspace obstacles into the configuration space requires expensive computations and is memory consuming [9]. All complete planners have exponential time complexity in the number of degrees of freedom of the robot [4, 12, 29]. This complexity restricts the application of exact planners and limit their use to robots with lower degrees of freedom.

1.3.2 Sampling-based Planners

The inevitable complexity of exact planners turned the researcher's attention towards methods that embed weaker notions of completeness [14]. Unlike exact planners, these algorithms were mostly designed on practical interests rather than theoretical ones. Avoiding a continuous configuration space, a path planning problem is solvable in a discrete configuration space (sampled configuration space). This approach results in a weaker completeness known as probabilistic completeness. A method is probabilistically complete if approaching the number of samples to infinity, eventually, leads to finding the answer in a finite time, if any exists. Due to this fact, these planners, unlike exact planners, are not able to report failure.

All methods in this category have two fundamental components in common, namely collision detection and local planner. The efficiency of these components directly affect the efficiency of a method. The collision detection module is the bridge between the workspace and the robot's configuration space. More precisely, this module tests a given configuration against collision (refer to [21] to find out about different collision detection techniques). Using this module, an algorithm is able to determine whether a configuration sample point is in the free space or an obstacle region.

The second component, local planner, is responsible for finding a local collision-free path between two configurations typically close together. As it will be explained later, methods in this category build a graph composed of sampled configuration points, as graph nodes, and

a number of connecting edges. A query can be answered if a path between the requested configurations can be found in the graph. In this graph, there is an edge between two nodes if the local planner is able to connect them without making a collision. The problem is that storing this local path information for each edge is memory consuming. The better approach is to use local planners that are able to reproduce their outcome. Thus, no path is needed to be stored. Due to this fact, local planners are typically as simple as a straight line planner. Even though more complicated planners can be used to solve more challenging situations, increasing the time complexity of the local planner module increases the method's time complexity.

Depending on the approach taken toward capturing configuration space connectivity, sampling-based methods are divided into two classes; multi-query and single-query. These two classes are explained in the following.

Multi-Query Planners

A class of sampling-based planners known as multi-query planners, try to capture the connectivity of the entire configuration space at once. As the name suggests, these methods are suitable for answering multiple queries over one environment. They spend a great portion of their execution time on exploring the configuration space and capturing its connectivity. Once this phase is finished, any query can be answered quickly in hundredths of a second using a graph search algorithm.

Probabilistic Roadmap Planner (PRM) is a well-known multi-query planner that is able to solve challenging problems even for a high degree of freedom robot [14]. The planner starts with an empty graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, known as roadmap. Set $\mathcal{V}$ is composed of the graph nodes, called milestones, and $\mathcal{E}$ stores edge information. This graph is expanded at each iteration by adding a randomly picked sample from the free configuration space to $\mathcal{V}$. Here, the collision detection module is called to assess whether a sampled configuration point is collision-free. Next, a local planner tests all possible connections between the newly added node and any neighboring milestones in $\mathcal{V}$. The neighboring distance is an adjustable parameter set by the operator. Any successful connection is represented by an edge in $\mathcal{E}$. The roadmap expansion procedure continues until a predetermined number of nodes is reached. This is the first phase of the method, called the *learning phase*. In the next phase which is called the *query phase*, the method searches for an answer to a requested query. At the beginning, the local planner tries to connect the initial and the goal configurations to the roadmap. Then, a graph search algorithm is employed to find a path based on the verified connections stored in $\mathcal{E}$. The configuration points involved in the resulting path along two subsequent milestones are reproduced using the local planner to compose the set $\mathcal{P}$ as a final answer.

Even though the PRM is a powerful method, there are many situations in which the PRM is not able to find an answer. These situations mostly contain regions with a small volume of free space, known as narrow passages. Since the PRM method selects samples randomly, the probability of picking a sample from a small volume of a narrow passage is low. As a result, simple local planners cannot preserve roadmap connectivity via the few number of milestones

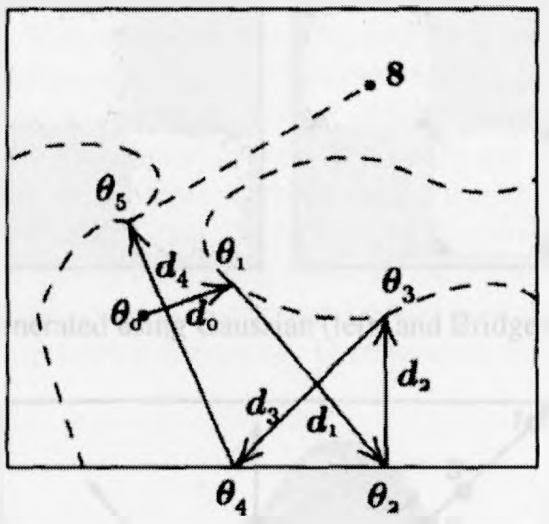


Figure 1.4: Random reflection at configuration space obstacles to connect roadmap components [9].

in a narrow passage. Therefore, instead of having one connected component, the roadmap will be composed of separate connected components, each isolated in one part of the configuration space. The method suggested in [9] is intended to improve the roadmap connectivity by taking random walks in the configuration space. This method starts exploring from one connected component in a random direction and looks for any milestone from the other components. If an obstacle is met, a new node is generated and walking continues towards another random direction (Fig. 1.4). This process continues until two components are unified.

There are other methods intended to improve the PRM by improving the set of milestones. One of the suggested ideas is to oversample some specific regions in order to increase the number of milestones in those regions. The fewer number of milestones in a region is the main characteristic of that region which should be oversampled. This characteristic can be determined in a roadmap in many ways; these include a milestone with a few number of edges and a milestone with a large Voronoi region. A Voronoi region of a milestone is the set of all points closer to that milestone than any other milestones. The objection over this method is the considerable number of referrals to the collision detection module. Moreover, a greater number of milestones requires a greater number of connections. As reported in [32], a large portion of the time of the PRM algorithm is spent on making connections. Therefore, a large number of milestones is a threat to the method's efficiency. Considering this reason several methods were proposed which only retained well-placed milestones. *Artificial Potential Biased PRM (ABPRM)* employs the artificial potential fields idea, explained in Section 1.3.3, to bias the distribution of the nodes in narrow passages [1]. In this method, the workspace potential values calculated for each point of the robot in a specific configuration are combined to define a potential field over the configuration space.

The configuration space obstacles can help in obtaining appropriate milestone to improve a roadmap connectivity. Picking samples close to the obstacles, increases the number of mile-

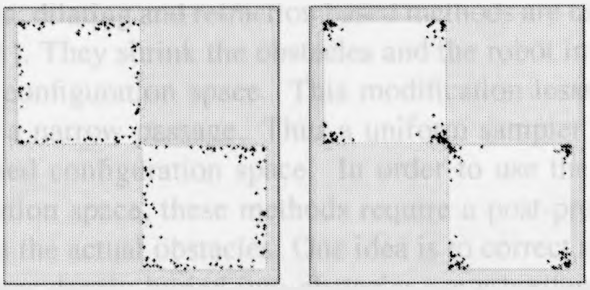


Figure 1.5: Milestones generated using Gaussian (left) and Bridge-test (right) samplers [10].

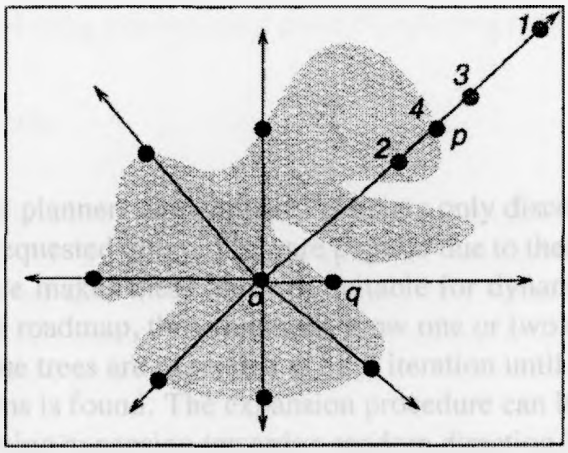


Figure 1.6: Finding obstacle boundary samples in OBPRM [3].

stones in a narrow passage. A *Gaussian Sampler* samples the configuration space in pairs. In order to have a pair, for any picked-up sample, a second sample distanced according to a normal (Gaussian) distribution is selected. The collision-free sample will be stored as a milestone if and only if the other sample in the pair is in collision [5] (Fig. 1.5). Similar to the Gaussian sampler, a *Bridge-Test* sampler obtains paired samples distanced based on a Gaussian distribution [10]. The mid-sample of a pair is considered a milestone, if both samples in a pair are in collision and their mid sample is collision-free (Fig. 1.5). The outcomes of these two methods are strongly dependent on the selected value for the standard deviation of the Gaussian distribution [13]. The proper value for this parameter depends on the configuration space properties which cannot be easily calculated. Unlike previous methods, *Obstacle-Based PRM (OBPRM)* [3, 2] tries to distribute the milestones over the obstacle boundaries. This method begins with a sample selected from an obstacle region and emits multiple rays with determined angles (Fig. 1.6). Next, a binary search is employed to find an obstacle boundary along each ray. These samples, picked up from the obstacle boundaries, compose the set of milestones. As explained earlier, milestones close to the obstacle boundaries enhances the connectivity of the roadmap in narrow passages. However, putting nodes close to obstacles or in touching positions is not suitable for all applications. The reason is that getting closer to an obstacle increases the probability of the collision and consequently reduces the safety. None of the aforementioned methods are able to set a minimum distance for milestones from the obstacles.

To improve the PRM idea, dilating and retraction based methods are used to modify the configuration space [11, 23, 31]. They shrink the obstacles and the robot in the workspace to widen narrow passages in the configuration space. This modification lessens the difficulty caused by the small volume of a narrow passage. Thus a uniform sampler can be used to generate milestones in the widened configuration space. In order to use the same set of milestones in the original configuration space, these methods require a post-processing phase to fix the milestones positioned on the actual obstacles. One idea is to correct all misplaced milestones. However, those milestones deeply buried into obstacles are a bottleneck for the performance of the algorithm. An alternative idea is to compute a penetration depth for each milestone but at the expense of complex geometry operations. Implementing these operations is difficult in higher dimensions.

Single-Query Planners

In contrast to multi-query planners, single-query planners only discover part of the configuration space related to the requested query. They are popular due to their ability to find a solution quickly [35]. This feature makes these methods suitable for dynamic environments. Unlike the multi-query planner's roadmap, these methods grow one or two tree graphs to explore the configuration space. These trees are expanded at each iteration until a path between the initial and the goal configurations is found. The expansion procedure can be as simple as choosing a random node and performing expansion towards a random direction [12]. However, in order to bias the expansion towards unexplored regions different heuristics are suggested; these include expanding the node with fewest number of edges, largest Voronoi region, or the node that is the closet node to the goal configuration.

Rapidly-exploring Random Trees (RRT) is one of the successful proposed methods [20]. This method starts the expansion by generating a random configuration point and finding the closest node of the tree to that configuration. Next, the algorithm searches for an input from the input space that minimizes the distance between the selected node and the random configuration. Finally using this input, the tree will be expanded for one step from the selected node. Since nodes with larger Voronoi regions have higher probability to be selected for expansion, this expansion is biased towards unexplored regions [18] (Fig. 1.7). The *RRT-Connect* method only differs from this method in the expansion step size [18]. It expands the tree from a selected node until it reaches an obstacle.

Besides different heuristics, some methods tried to reduce the total execution time. The *Lazy Collision-Checking* is a technique that delays the process of checking an edge for collisions until that edge is selected as part of the final path [32]. According to the observations reported in [32], a short connection between two collision-free configurations is collision-free with a high probability. There are two other reasons that support this idea. First, most of the connections involved in making a tree do not contribute in the final path. Second, testing collision-free connections are more expensive than the others. Therefore, this intentional delay is able to improve the execution time in some scenarios.

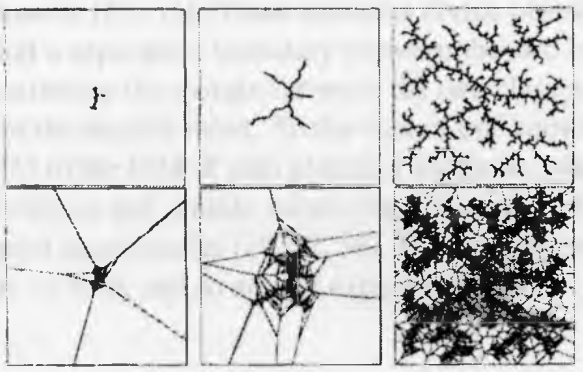


Figure 1.7: Rapidly exploring tree is biased toward exploring large Voronio Regions [18].

1.3.3 Other Planners

There are many other methods that are neither complete nor probabilistically complete. These methods are mostly local, i.e. they decide about the next step based on the current state of a robot. Even though they are able to work in real-time, they suffer from the problem known as *local minima*. The Local minima are situations that lead to a stable positioning of a robot before reaching to the desired goal position.

Artificial Potential Field (APF) method creates two attractive and repulsive potential fields [15]. The former is sourced from the goal configuration and the latter from the obstacles. Superimposing these potentials leads into a potential field that has its minimum at the goal configuration and high potentials on the obstacle regions. Therefore, seeking the negated gradient of this potential field at each configuration leads the robot towards the goal configuration and keeps it away from the obstacles.

As mentioned before, the main drawback of these methods is related to situations in which the robot is trapped into a local minimum. Since there is no systematic way to escape from these situations [4], some methods attempted to create local-minimum free potential fields. These methods are based on harmonic potential fields and theories discussed in fluid mechanics [6, 16, 17, 30]. However, these methods require complex calculations and have not been applied to higher degrees of freedom [14]. Introducing randomness is an alternative solution for escaping local minima. *Randomized Path Planner (RPP)* and *Virtual Spring Method* tackled the problem from this perspective [4, 25].

Support Vector Machine-Based Methods

The *Support Vector Machines (SVM)* method, which is used as a foundation in this body of work, was originally proposed by Vapnik to solve classification and regression analysis problems [37]. This technique has been widely used in a variety of applications such as image recognition and bioinformatics [7]. SVM has been used previously for mobile robot path plan-

ning in cluttered environments [26, 33]. These methods divide obstacles into two classes and utilize classification to find a separation boundary between the two classes of data. Since the SVM method aims to maximizes the margin between the two classes, this boundary is a suitable nominee as a path for the mobile robot. To the best of our knowledge, the only proposed methods based on the SVM in the field of path planning are those listed above. There are other methods in the field of robotics and mobile robots that utilized the SVM for data fusion and coping with the environment uncertainties [19, 22, 36]. However, these methods are not related to the context of this body of work and so are not explained here.

1.4 Objectives

Path planning is one of the basic components in robotic systems and to this date many different methods have been proposed to solve this problem. Each method is intended to solve a certain problem and as such performs poorly in more general problems. This work considers planners that work in the configuration space. Due to the high number of dimensions in the configuration space and the lack of exact knowledge about its topology, practical planners work on discrete (sampled) configuration space.

The objective of this work is to develop a planner for robotic manipulators with high degrees of freedom. The planner is expected to have the advantage of both sample-based and local planners in term of completeness and speed. The planner is intended for the environments in which the safety of operation is of high priority. While the planner is expected to be fast, it is not required to operate in real-time nor in a dynamic environment.

To achieve these objectives, one plausible idea is to estimate obstacle regions based on obstacle samples in the configuration space. This estimation can be the outcome of the SVM method and is used towards defining a function over the configuration space. This function can then be employed to obtain safe configurations. Using the maximal margin concept in the SVM algorithm, the proposed function, called a *Risk Function*, can be defined such that it obtains safe configurations with maximum distance from surrounding obstacles. This feature becomes especially important when dealing with narrow passages. It is envisioned that the proposed idea can be embedded in different planners to improve their performance.

1.5 Contributions

The contributions of this body of work are summarized as follows,

- To propose a function in the configuration space called a risk function using configuration space obstacle samples. The value of a risk function for each configuration can be interpreted as the risk of collision corresponding to that configuration. The proposed

function is expected to have its local minimum at maximum distance from surrounding obstacle samples.

- To propose a local planner based on a risk function that finds a solution in a sampled configuration space for high degree of freedom manipulator.
- To propose a multi-query sampling based planner that employs a risk function to find a set of well-placed milestones. This planner is able to capture the connectivity of the configuration space even in an environment with narrow passages with fewer number of referral to the collision detection module.

1.6 Thesis Outline

The rest of this thesis is organized in four chapters. Chapter 2 presents a local planner that finds a solution in the configuration space using a risk function. The obstacle sampling method, risk function formulation, and robot guidance using the proposed local planner are discussed. Chapter 3 presents another planner based on PRM. The proposed planner takes advantage of a risk function defined using the SVM algorithm. In this chapter, a different obstacle sampling approach is discussed. Also, a procedure for finding milestones and making a roadmap is explained. Chapter 4 discusses different case studies and contains implementation notes. Finally, Chapter 5 concludes the thesis and suggests future directions.

Bibliography

[1] D. Aarno, D. Kragic, and H.I. Christensen. Artificial potential biased probabilistic roadmap method. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 461–466. IEEE, 2004.

[2] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based PRM for 3d workspaces. In *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pages 155–168. Wellesley, MA: AK Peters, 1998.

[3] N.M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 113–120. IEEE, 1996.

[4] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628, 1991.

[5] V. Boor, M.H. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1018–1023. IEEE, 2002.

[6] C.I. Connolly, JB Burns, and R. Weiss. Path planning using laplace's equation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 2102–2106. IEEE, 1990.

[7] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel based learning methods*. Cambridge Univ Pr, 2000.

[8] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag New York Inc, 2008.

[9] T. Horsch, F. Schwarz, and H. Tolle. Motion planning with many degrees of freedom-random reflections at c-space obstacles. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3318–3323. IEEE, 1994.

[10] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4420–4426. IEEE, 2003.

[11] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pages 141–154, 1998.

[12] D. Hsu, J.C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2719–2726. IEEE, 1999.

[13] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid prm sampling with a cost-sensitive adaptive strategy. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3874–3880. IEEE, 2005.

[14] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[15] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90, 1986.

[16] J.O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *Robotics and Automation, IEEE Transactions on*, 8(3):338–349, 1992.

[17] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1–6. IEEE, 1987.

[18] J.J. Kuffner Jr and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[19] G. Kumar, KJ Poornaselvan, and M. Sethumadhavan. Fuzzy support vector machine-based multi-agent optimal path planning approach to robotics environment. *Defence Science Journal*, 60(4):387–391, 2010.

[20] S.M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[21] S.M. LaValle. *Planning algorithms*. Cambridge Univ Pr, 2006.

[22] J. Lei, Q. Song, J. Ma, L. Qiu, and Y. Ge. Application of svm in intelligent robot information acquisition and processing: a survey. In *Information Acquisition, 2005 IEEE International Conference on*, pages 6–pp. IEEE, 2005.

[23] J.M. Lien, S.L. Thomas, and N.M. Amato. A general framework for sampling on the medial axis of the free space. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4439–4444. IEEE, 2003.

[24] A.A. Maciejewski and J.J. Fox. Path planning and the topology of configuration space. *Robotics and Automation, IEEE Transactions on*, 9(4):444–456, 1993.

[25] A. McLean and S. Cameron. The virtual springs method: Path planning and collision avoidance for redundant manipulators. *The International journal of robotics research*, 15(4):300, 1996.

[26] J. Miura. Support vector path planning. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2894–2899, 2006.

[27] W.S. Newman and M.S. Branicky. Real-time configuration space transforms for obstacle avoidance. *The International Journal of Robotics Research*, 10(6):650, 1991.

[28] C. O'DÚNLAING and C.K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6(1):104–111, 1985.

[29] J.H. Reif. Complexity of the mover's problem and generalizations extended abstract. In *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pages 421–427, 1979.

[30] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *Robotics and Automation, IEEE Transactions on*, 8(5):501–518, 1992.

[31] M. Saha, J.C. Latombe, Y.C. Chang, and F. Prinz. Finding narrow passages with probabilistic roadmaps: The small-step retraction method. *Autonomous robots*, 19(3):301–319, 2005.

[32] G. Sánchez and J.C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. *Robotics Research*, pages 403–417, 2003.

[33] Saurabh Sarkar, Ernest L. Hall, and Manish Kumar. Mobile robot path planning using support vector machines. volume 2008, pages 709–715. ASME, 2008.

[34] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer-Verlag New York Inc, 2008.

[35] I.A. Sucan and L.E. Kavraki. On the implementation of single-query sampling-based motion planners. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2005–2011. IEEE, 2010.

[36] J. Tian, M. Gao, and E. Lu. Dynamic collision avoidance path planning for mobile robot based on multi-sensor data fusion by support vector machine. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pages 2779–2783. IEEE, 2007.

[37] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

Chapter 2

Local Support Vector Machine-based Planner

Path planning using local planners has gained popularity due to the ability of these methods to find solutions quickly and effectively. Because of this capability, local planners are widely used in mobile robots' real-time planning. Potential field-based methods are a group of successful local planners for mobile robots [7, 8, 15, 21, 24]. In the field of manipulators, this approach was first exercised by Khatib [11]. In this approach a potential field composed of attractive and repulsive potentials is defined over the work or configuration free-space. Generally, a goal region is the source of an attractive field and obstacles create repulsive potentials. Seeking minimum of the potential field at each point, attracts a robot towards the goal position and keeps it away from obstacles.

The main drawback of potential field-based methods is the problem known as *local minimum*. Local minimum is a situation in which manipulator becomes stable before reaching the goal configuration. In this situation there is an equilibrium between influential forces. Thus having no net force to move the manipulator, it becomes trapped in that situation (Fig. 2.1). To solve this problem, some local minimum-free potential fields have been suggested [5, 13, 14, 18]. However, their application is limited due to the complexity of the required calculations [10] or restrictions over the shape of obstacles [1].

A potential field can be defined over either the workspace or configuration space. Within the workspace, a potential field can affect the end-effector and some control points selected on the manipulator links [3, 11, 12, 16]. These control points either have fixed positions or are selected dynamically as the closest point of each link to the obstacles. The manipulator's forward dynamics are used to simulate the manipulator's reaction to the influence of the artificial potential field. Therefore, the combination of the forces exerted on the manipulator, determines the manipulator's next configuration. In robot manipulator, the problem of local minimum may appear as structural local minimum. This situation happens when the net forces on different joints become zero before reaching the goal position.

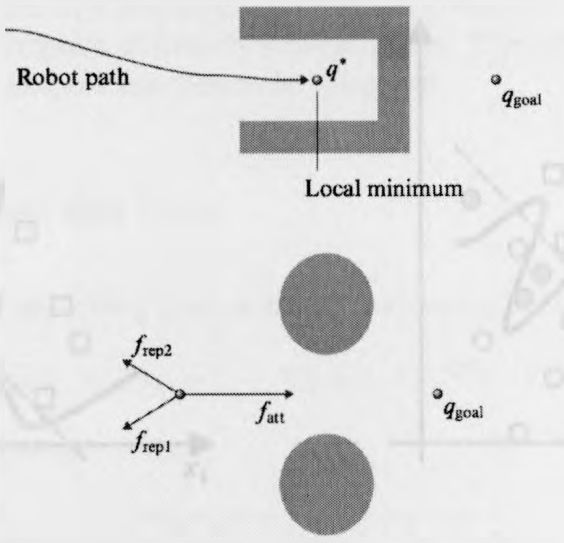


Figure 2.1: Two examples of local minimum situation [20].

A potential field can be defined over the configuration space as well. To achieve this, two possible solutions exist. The first solution needs complete knowledge of the configuration space obstacles which is not always a feasible assumption [1]. If this information is provided, the potential field is defined similar to the workspace for mobile robots. The second approach is to assign a potential value to each configuration based on a potential field defined over the workspace [2]. This is achievable by combining the potential fields influencing the control points on the manipulator links.

This chapter proposes an innovative solution for defining a potential field over the configuration space. The proposed method uses configuration obstacle samples to calculate a repulsive potential field. It implies that no exact information about the configuration space is required. In this space, a manipulator is represented by a point. Since configuration obstacle samples do not provide any reliable information about the obstacle boundaries, approaches merely based on distance are likely to fail. Also, if each sample point acts as a source of repulsive force, they have to have different repulsive force gains, because different obstacles are captured by different number of samples. In this situation, exerting the same amount of repulsive force from each sample point may put the manipulator too close to the obstacles with fewer number of samples. Moreover, attuned repulsive gains may prevent the manipulator from passing in between two obstacles.

The solution suggested in this chapter uses an optimization technique to automatically adjust the obstacle samples' repulsive gain. The resulting repulsive potential field has its local minimum at maximum distance from surrounding obstacle samples in most cases. This is especially important while encountering situation such as Fig. 2.1 in which the manipulator must pass through a narrow passage. We call the repulsive potential field defined in this chapter, a *risk function*. The risk function is calculated with the help of Support Vector Machine (SVM) algorithm. The combination of a risk function and a simple attractive potential field defines the

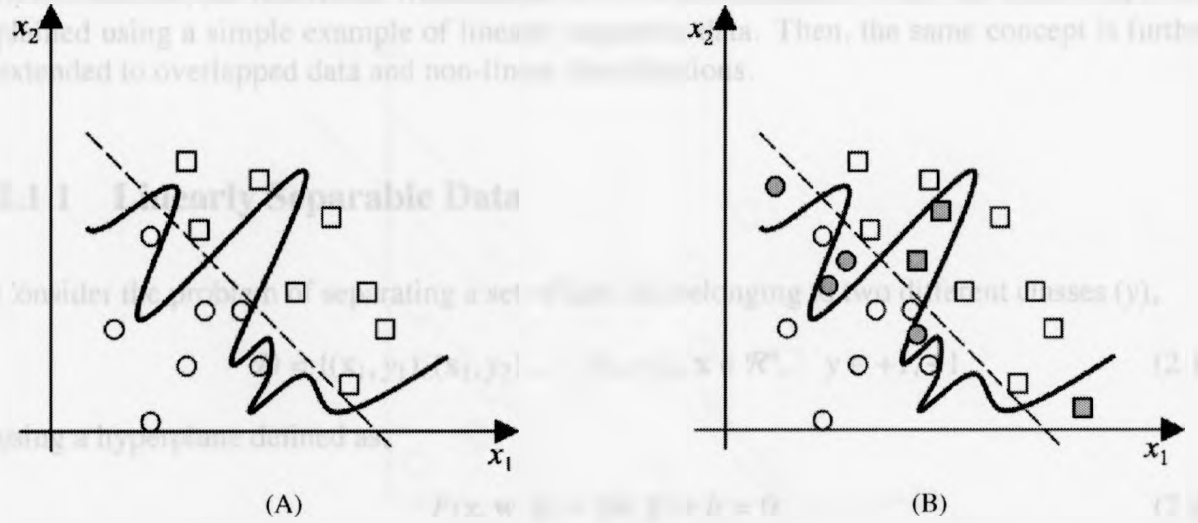


Figure 2.2: (A) Two possible classification with linear (dashed line) and non-linear (solid line) models. Overfitting occurred in the case of non-linear classifier which resulted in less generality. Gray symbols in (B) are the misclassified data based on the non-linear classifier. Squares and circles represent data from different classes [23].

final potential function over the configuration space. This final potential field has its global minimum at the goal configuration.

The rest of the chapter is organized as follows; Section 2.1 explains the support vector machine theoretical foundations. Section 2.2 presents the proposed method and simple simulation results in order to provide a better understanding of the method. Section 2.3 illustrates more simulation results and analyzes the method's parameters. Finally, Section 2.4 concludes the chapter and lists its shortcomings and future directions.

2.1 Support Vector Machine

Support Vector Machine (SVM) is a machine learning method originally proposed for regression analysis and classification purposes [6, 22]. In classification context, the method's input is in the form of n-dimensional data points. Each data belongs to one of two classes. SVM finds a *decision function* (classification model) to minimize the estimation error while maintaining a constant training error. The training error is the error of classification model on input data and the estimation error is related to the model's estimation reliability. Although a model can be built to have zero training error (overfitting), minimizing estimation error increases the generality of a model (Fig. 2.2). An overfitted model is closely fitted around obstacle samples and is not a proper estimation of obstacle boundaries. In contrast, a model with higher generalization capability results in an estimation larger than the real obstacle region while it does not let different obstacles interfere with each other.

In this section, the theoretical foundations of SVM are discussed. First, the basic idea is explained using a simple example of linearly separable data. Then, the same concept is further extended to overlapped data and non-linear classifications.

2.1.1 Linearly Separable Data

Consider the problem of separating a set of data ($\mathbf{x}$) belonging to two different classes (y),

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}, \mathbf{x} \in \mathcal{R}^n, \quad y \in +1, -1 \tag{2.1}$$

using a hyperplane defined as,

$$F(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b = 0. \tag{2.2}$$

The SVM is used to calculate the parameters of this hyperplane, i.e. $\mathbf{w} \in \mathcal{R}^n$ and the scalar b known as weights and bias parameters, respectively. The function $F(\mathbf{x}, \mathbf{w}, b)$ is called a *decision function* and will be used to classify any new data point $\mathbf{x}_{new}$. The classification result is obtained based on the sign of $F(\mathbf{x}_{new}, \mathbf{w}, b)$. In other words,

- if $F(\mathbf{x}_{new}, \mathbf{w}, b) > 0$, $\mathbf{x}_{new}$ belongs to Class 1.
- if $F(\mathbf{x}_{new}, \mathbf{w}, b) < 0$, $\mathbf{x}_{new}$ belongs to Class -1.

Data points for which $F(\mathbf{x}, \mathbf{w}, b) = 0$, form a *decision (separation) boundary* between the two classes.

Note that if $F(\mathbf{x}, \mathbf{w}, b)$ is a valid solution, all $F(\mathbf{x}, k\mathbf{w}, kb)$ functions, where k is a positive scalar, are also valid and can be used instead. Therefore, without losing generality one of these hyperplanes, named *canonical hyperplane*, is selected which fulfills the following condition,

$$\min_i |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1, \quad i = 1 \ldots l \tag{2.3}$$

Separating by a hyperplane, the input set is "optimally separated if it is separated without error and the distance between the closest data to the hyperplane is maximal" (maximal margin concept) [9]. The margin between the two classes can be derived algebraically or geometrically [9] as, (Fig. 2.3),

$$M = \frac{2}{\| \mathbf{w} \|}. \tag{2.4}$$

Therefore based on the definition of a canonical hyperplane, $\| \mathbf{w} \|$ or equivalently $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ must be minimized while,

$$y_i |\langle \mathbf{w}, \mathbf{x} \rangle + b| \geq 1, \quad i = 1, \ldots, l. \tag{2.5}$$

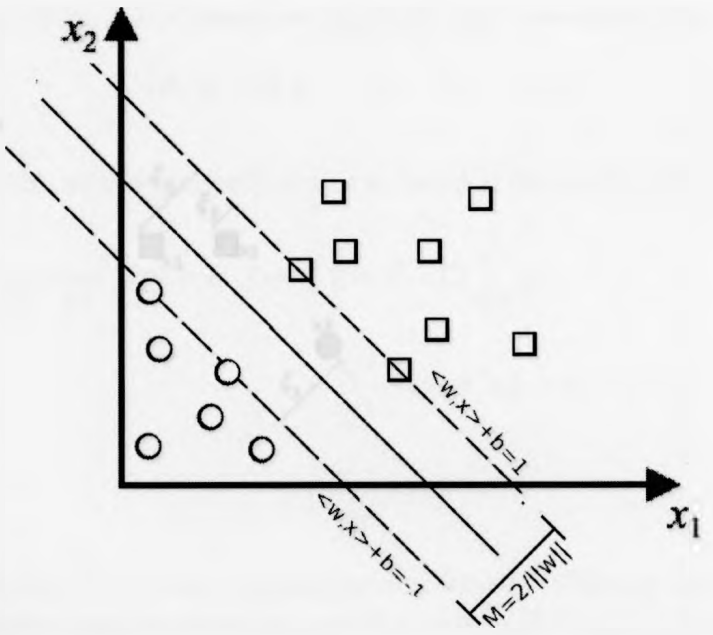


Figure 2.3: Canonical hyperplane (solid line) and the margin width. Circle and square symbols represent data from different classes.

Introducing *Lagrangian multipliers*, the optimization problem can be solved by finding the saddle point of the following equation,

$$\min_{\mathbf{w},b} \max_{\alpha} L(\mathbf{w}, b, \alpha, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i\{y_i|\langle \mathbf{w}, \mathbf{x}_i\rangle + b| - 1\} \tag{2.6}$$

where $\alpha_i \geq 0$ are Lagrangian multipliers. Since the objective function and the constraints are convex, the following conditions are satisfied at the optimal point $(\mathbf{w}_0, b_0, \alpha_0)$,

$$\frac{\partial L}{\partial \mathbf{w}_0} = 0 \rightarrow \mathbf{w}_0 = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \tag{2.7}$$

$$\frac{\partial L}{\partial b_0} = 0 \rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0, \tag{2.8}$$

$$\alpha_i\{y_i|\mathbf{w}^T\mathbf{x}_i + b| - 1\} = 0, \quad i = 1, \ldots, l. \tag{2.9}$$

where, (2.9) is *Karush-Kuhn-Tucker (KKT)* complementary condition. Substituting $\mathbf{w}_0$ and b_0 into (2.6) yields,

$$L_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \tag{2.10}$$

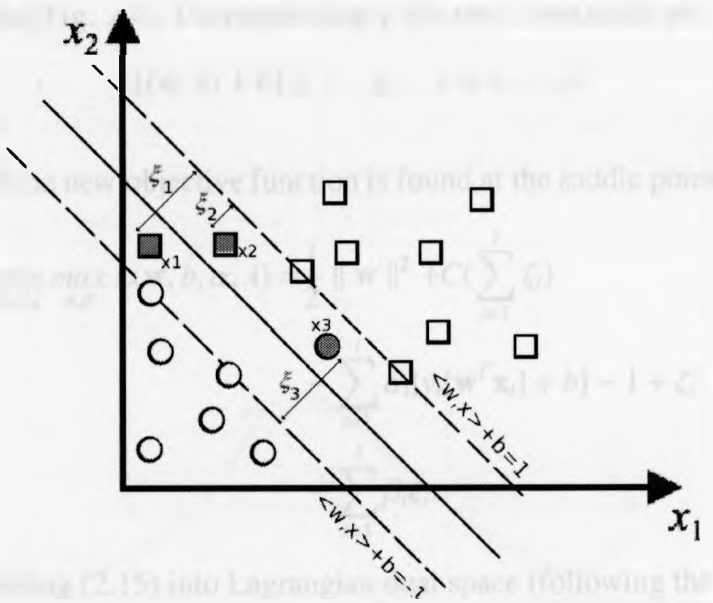


Figure 2.4: Soft margin and error measurement. Circle and square symbols represent data from different classes. Gray symbols are unclassified data inside the soft margin.

with the following constraints,

$$0 \leq \alpha_i, \; i = 1 \ldots l \tag{2.11}$$

$$\sum_{j=1}^{l} \alpha_j y_j = 0 \tag{2.12}$$

This problem is now in the form of a *standard quadratic optimization problem* and can be solved using quadratic programming techniques and programs [23].

2.1.2 Overlapping Data

In this case, the data cannot be separated linearly with zero training error. In other words, constraints given by (2.5) cannot be satisfied and for any misclassified data, α_i tends to infinity. In order to overcome this issue, the SVM must allow some data to be unclassified. The notion of *soft margin* allows the SVM to neglect all input data inside a specific margin (Fig. 2.4). The width of this margin is adjustable using a penalty parameter C. This parameter controls the trade-off between the training error and the estimation error.

The optimization problem now turns into minimization of the following statement,

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C(\sum_{i=1}^{l} \zeta_i) \tag{2.13}$$

where ζ_i are non-negative *slack variables* representing the distance of unclassified data to the corresponding margin (Fig. 2.4). Correspondingly, the new constraints are,

$$y_i[\langle \mathbf{w}, \mathbf{x} \rangle + b] \geq 1 - \zeta_i, \quad i = 1, \ldots, l. \tag{2.14}$$

The optimal point of the new objective function is found at the saddle point of,

$$\begin{aligned} \min_{\mathbf{w},b,\zeta} \max_{\alpha,\beta} L(\mathbf{w}, b, \alpha, \lambda) = & \frac{1}{2} \| \mathbf{w} \|^2 + C(\sum_{i=1}^{l} \zeta_i) \\ & - \sum_{i=1}^{l} \alpha_i \{y_i[\mathbf{w}^T \mathbf{x}_i] + b] - 1 + \zeta_i\} \\ & - \sum_{i=1}^{l} \beta_i \zeta_i. \end{aligned} \tag{2.15}$$

Interestingly, transferring (2.15) into Lagrangian dual space (following the same procedure as for the linearly separable data) leads to the same function as (2.10) [23]. The only difference is related to α_i constraints, which are bounded by C,

$$0 \leq \alpha_i \leq C, \; i = 1 \ldots l \tag{2.16}$$

2.1.3 Non-linear Classifier

In many situations, input data cannot be separated efficiently using a linear classifier. The SVM method can be easily extended to non-linear problems by introducing the notions of *feature space* and *kernel function*. The main idea behind the extension is to map the input data into a feature space with higher dimensions and apply a linear classifier on the feature space.

$$\mathbf{x} \in \mathcal{R}^n \rightarrow \mathbf{\Phi}(\mathbf{x}) = \left[\phi_1(\mathbf{x})\phi_2(\mathbf{x}) \ldots \phi_f(\mathbf{x})\right]^T \in \mathcal{R}^f. \tag{2.17}$$

Taking a closer look at 2.10, it is notable that the input data appear in inner products. Since the result of an inner product is a scalar value, instead of expensive mapping to high dimensional feature space, a kernel function can be used to solve the problem in the input space. A kernel function is defined as,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \tag{2.18}$$

Some commonly used kernels are listed in Table 2.1.

The following example is taken from [23] to better explain the idea of a feature space and non-linear mapping to higher dimensional spaces. Assume a one dimensional classification problem, given following set of data,

$$\mathcal{D} = \{(1, -1), (0, 1), (-1, -1)\}. \tag{2.19}$$

Table 2.1: SVM Kernel Examples

Kernel Function	Description
$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$	Linear dot product
$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + c_0)^d$	Complete polynomial of degree d
$K(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\gamma \lvert \mathbf{x}_i - \mathbf{x}_j \rvert^2\right)$	Gaussian Radial Basis Function
$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\gamma \mathbf{x}_i^T \mathbf{x}_j + c_0\right)$	Sigmoid

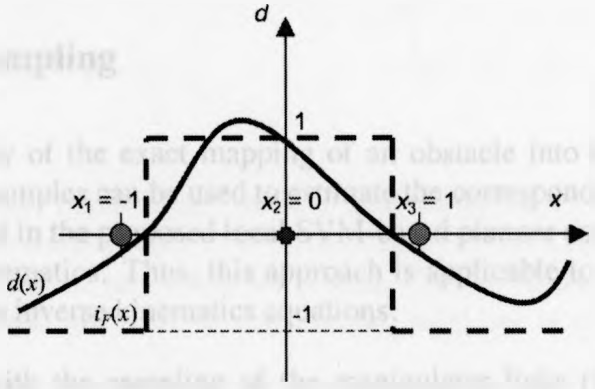


Figure 2.5: Example of one dimensional non-linear classification problem with three input data. The decision function is plotted using a solid line. The dashed line represents the sign of the decision function and the classification result [23].

This problem is illustrated in Fig. 2.5. The input data is given in a form of a 3×1 matrix $\mathbf{x} = [-1\ 0\ 1]^T$ which are labeled as $y = [-1\ 1\ -1]^T$. Consider a mapping function as follows,

$$\mathbf{\Phi}(\mathbf{x}) = [\phi_1(\mathbf{x})\ \phi_2(\mathbf{x})\ \phi_3(\mathbf{x})]^T = [x^2\ \sqrt{2x}\ 1]^T. \tag{2.20}$$

This mapping produces following data points in the three dimensional feature space,

$$F = \begin{bmatrix} 1 & -\sqrt{2} & 1 \\ 0 & 0 & 1 \\ 1 & \sqrt{2} & 1 \end{bmatrix}. \tag{2.21}$$

In (2.21), each row represents one point of data. Now, these three points can be separated linearly by the plane $\phi_3(\mathbf{x}) = 2\phi_1(\mathbf{x})$ in a feature space. The same solution can be found using a quadratic kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^2$. It can be easily shown that (2.18) holds between the above mapping and kernel function.

2.2 Local SVM-based Planner

As explained earlier, the proposed method is intended to define a potential field over the configuration space based on configuration obstacle samples. The main idea is to build a SVM model

over the configuration obstacle samples and derive a risk function using that model. This risk function will be used as the obstacle repulsive potential field.

This method is presented in three steps in the following sections; namely, obstacle sampling, defining a risk function, and path planning. In order to better describe the method, the outcome of each step is illustrated using simple simulations for a two degrees of freedom planar manipulator.

2.2.1 Obstacle Sampling

To avoid the complexity of the exact mapping of an obstacle into the configuration space, configuration obstacle samples can be used to estimate the corresponding obstacle region. The sampling algorithm used in the proposed local SVM-based planner uses the information of the manipulator inverse kinematics. Thus, this approach is applicable to a class of manipulators with known closed-form inverse kinematics equations.

The first step begins with the sampling of the manipulator links ($S_{manip} = \{M_0, \ldots, M_s\}$) with a predetermined resolution. Also, each obstacle in the workspace is sampled ($S_{obst_i} = \{O_{i_0}, \ldots, O_{i_t}\}$ represents all samples belong to ith obstacle). Next for each obstacle, the set of all possible collisions is generated,

$$collisions_i = \left\{\langle m, o\rangle | m \in S_{manip} \wedge o \in S_{obst_i}\right\} \tag{2.22}$$

Each collision $\langle m, o\rangle \in collisions_i$ represents a scenario in which the manipulator's point m is in collision with the sample point o from the ith obstacle. Using the manipulator's kinematics, for each pair, a number of manipulator configurations will be obtained. The set of configurations obtained from all collision scenarios is the set of samples for a specific obstacle region in the configuration space (Fig. 2.6),

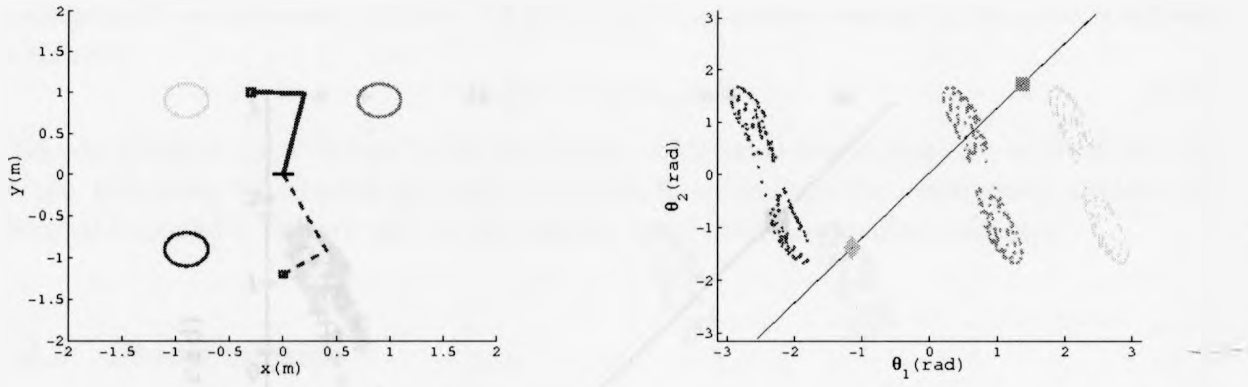
$$\mathbf{Q}_{obst_i} = \{q | A(q) \cap obst_i \neq \emptyset\} \subset Q \tag{2.23}$$

Besides workspace obstacles, the configuration space boundaries and the manipulator joint limitations contribute in forming the configuration space obstacle regions. These regions, also, must be sampled and added to the list of configuration obstacle samples.

All obstacle samples in the configuration space will be used later to calculate a risk function.

2.2.2 Risk Function

Using obstacle sampling, it is not possible to determine accurate boundaries of an obstacle region. However it is reasonable to assume that a safe configuration for a manipulator is a



(A) (B)

Figure 2.6: (A) Workspace: Initial (solid line) and goal configuration (dashed line) for a 2DoF planar manipulator in an environment with three obstacles. (B) Configuration space: Initial (square) and goal configuration (diamond) are plotted. Each workspace obstacle and its corresponding samples are illustrated with the same color. Solid line is used for labeling obstacle samples.

configuration point with maximum distance from surrounding obstacle samples. This feature is exactly what the SVM provides based on the maximal margin concept.

In order to prepare the configuration space for applying the SVM method, obstacle samples must be first labeled as two different classes. To label obstacle samples, a similar approach is used in [17, 19] for two dimensions which is extended to higher dimensions here. In this labeling approach, given the initial and goal configurations, a hyperplane is considered as,

$$P(\mathbf{q}, \mathbf{w}_p, b_p) = \langle \mathbf{w}_p, \mathbf{q} \rangle + b_p = 0. \tag{2.24}$$

where $\mathbf{q} \in Q$. The parameters $\mathbf{w}_p$ and b_p are calculated in such a way that $P(\mathbf{q}_{initial}, \mathbf{w}_p, b_p) = P(\mathbf{q}_{goal}, \mathbf{w}_p, b_p) = 0$. Since there are many solutions for this problem, one of the hyperplanes is selected at random. Next, each obstacle region class is calculated as follows,

$$y = \sum_{\mathbf{q}_s \in \mathbf{Q}_{obst_i}} sign\Big(P(\mathbf{q}_s, \mathbf{w}_p, b_p)\Big) \tag{2.25}$$

where $y \in \{+1, -1\}$ is the class label. In other words, this hyperplane divides the configuration space into two domains. Each obstacle region belongs to the domain in which the majority of its samples belong in (Fig. 2.7).

To this end, obstacle regions and the configuration space boundaries are sampled and labeled into two classes. Now the SVM can be applied directly over the configuration space. The output of the SVM is a decision function $F(\mathbf{q}, \mathbf{w}_0, b_0)$. Due to the maximal margin concept, the decision boundaries formed by $F(\mathbf{q}, \mathbf{w}_0, b_0) = 0$, are placed at maximum distance between obstacle samples of different classes. Thus, these decision boundaries potentially contain the safest possible configuration points (Fig. 2.8).

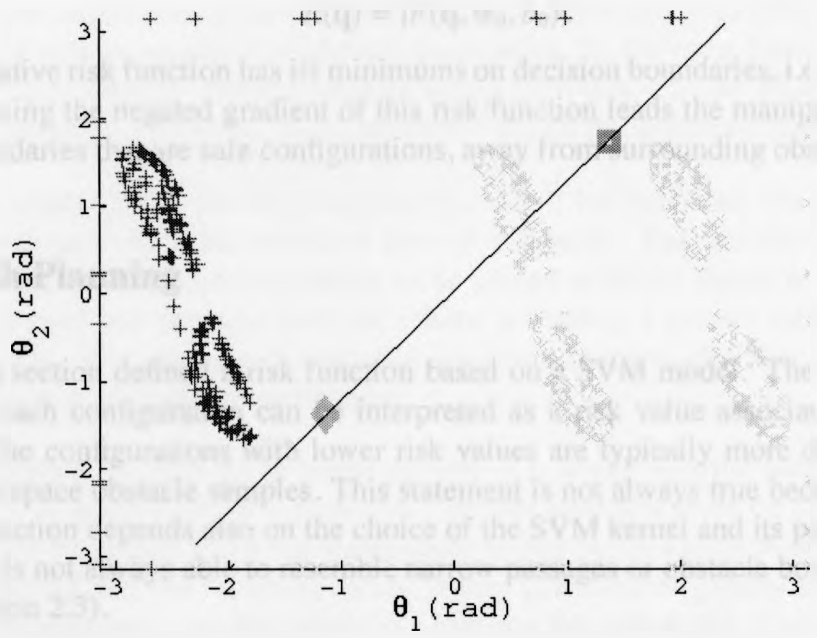
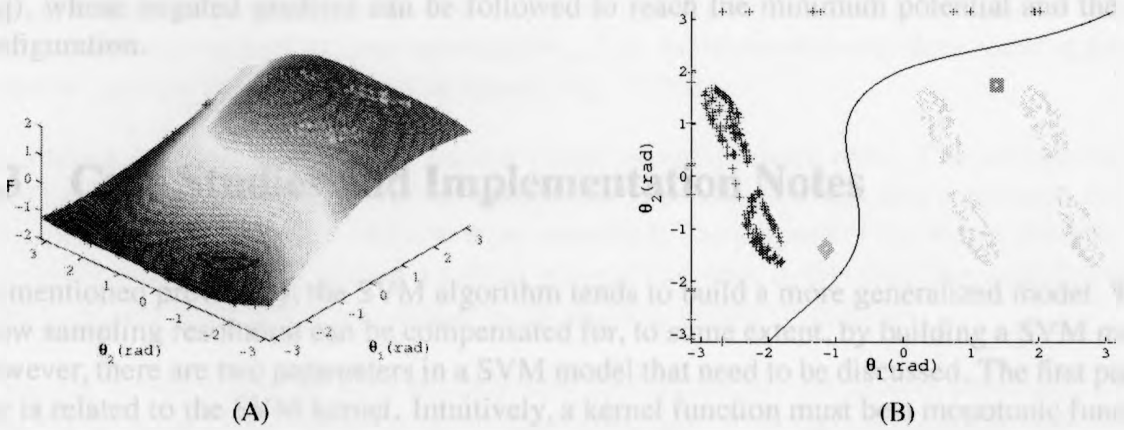


Figure 2.7: 2D hyperplane (a line) which is used for labeling obstacle and configuration space boundary samples. Different classes are shown with different colors and symbols.



(A) (B)

Figure 2.8: (A) Decision function calculated by SVM. (B) Decision boundaries shown over configuration space.

Here there is a need for a procedure which is able to guide the manipulator towards the decision boundaries from each configuration. For this purpose, a function named *risk function* is defined as follows,

$$R(\mathbf{q}) = |F(\mathbf{q}, \mathbf{w}_0, b_0)|. \tag{2.26}$$

This non-negative risk function has its minimums on decision boundaries, i.e., where $R(\mathbf{q}) = 0$. Hence, following the negated gradient of this risk function leads the manipulator towards the decision boundaries that are safe configurations, away from surrounding obstacles.

2.2.3 Path Planning

The previous section defined a risk function based on a SVM model. The value of this risk function for each configuration can be interpreted as a risk value associated with that configuration. The configurations with lower risk values are typically more distanced from the configuration space obstacle samples. This statement is not always true because the curvature of the risk function depends also on the choice of the SVM kernel and its parameters. Thus, a risk function is not always able to resemble narrow passages or obstacle boundaries curvature (refer to Section 2.3).

Intuitively, a risk function can be used as a repulsive potential over configuration space. Following the negated gradient of this potential, leads the manipulator towards safer configurations. However, this function is not sufficient for finding a path for the manipulator to move toward the goal configuration. In order to add this motion, a simple attractive potential field based on the distance is introduced,

$$U_{attract}(\mathbf{q}) = \mu \parallel \mathbf{q} - \mathbf{q}_{goal} \parallel . \tag{2.27}$$

where μ is a constant predetermined gain. This attractive potential has its minimum at the goal configuration. Combining attractive and repulsive potentials leads to the function $U_{attract}(\mathbf{q}) + R(\mathbf{q})$, whose negated gradient can be followed to reach the minimum potential and the goal configuration.

2.3 Case Studies and Implementation Notes

As mentioned previously, the SVM algorithm tends to build a more generalized model. Thus, a low sampling resolution can be compensated for, to some extent, by building a SVM model. However, there are two parameters in a SVM model that need to be discussed. The first parameter is related to the SVM kernel. Intuitively, a kernel function must be a monotonic function. This is due to the fact that the repulsive influence of a sample on a manipulator must decrease monotonically as the manipulator recedes. In our study, a *Gaussian Radial Basis Function* was selected as a suitable kernel (Table 2.1). This kernel can be rewritten in the following form,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2}|\mathbf{x}_i - \mathbf{x}_j|^2) \tag{2.28}$$

where $\gamma = \frac{1}{2\sigma^2}$. The parameter σ determines the neighborhood in which each sample has an influence. Increasing this parameter yields larger estimated regions for obstacles and also reduces the curvature of the resulting decision function and vice versa (refer to Section 4.1.1). On the one hand, an increased curvature allows a decision function to follow more accurately the topology of narrow passages or obstacle boundaries. At the same time, smaller estimated regions are closely fitted to obstacle samples and as such they may not be a proper estimation for the entire obstacle. Therefore, the parameter σ should be adjusted to control this trade-off.

The other parameter is the penalty parameter C. As (2.16) describes, the penalty parameter limits the maximum allowable repulsive gain of a sample. The samples with higher gains are able to force the decision boundaries to be placed at larger distances. Thus, a penalty parameter can bias one obstacle over the others, providing a greater safety margin for the selected obstacle. For example in scenarios involving humans, decision boundaries can be biased in order to stay farther from humans.

An other parameter in local SVM-based planner is μ which sets the gain for the attractive potential field. A larger value of μ results in a greater attractive force on the manipulator and vice versa. This force may assist the manipulator to pull out of a local minimum. However, if the force is greater than the repulsive forces, it may pull the manipulator towards an obstacle region. On the other hand, smaller values of μ increase the probability of getting stuck withing local minima. Since no exact information related to obstacle boundaries exist, setting this gain is challenging.

The proposed method in this chapter was implemented in the C++ language. The *libsvm* library was used to train a SVM model [4]. All experiments were conducted on a 2.53 GHz Intel Core 2 Duo CPU. The proposed method was tested on a 2 DoF planar manipulator and a simplified model of a CRS-F3 manipulator using its first three joints in simulation environment. The former manipulator was mainly used to provide explanations of the method itself (Fig. 2.6(A)). The results of this simulation were discussed throughout this chapter. The CRS-F3 model was used to assess the proposed method in a close-to-real scenario, where a manipulator needs to enter a car frame through its door opening (Fig. 2.9). In this scenario the door opening presents a narrow passage in the configuration space (Fig. 2.10(B)).

To obtain a solution some guides, named virtual obstacles, were used. The virtual obstacles were treated as real obstacles with an exception. The exception was that a collision between the manipulator and a virtual obstacle were permitted. The purpose of the virtual obstacles was to help the manipulator in finding a narrow passage opening. In this case, the virtual obstacles were three line segments perpendicular to the window planes (Fig. 2.10).

Since the virtual obstacles were actually safe configurations, the penalty parameter was set such that it biased real obstacles over virtual obstacles. This selection allowed the SVM to decrease the training error for real obstacle samples and consequently the decision boundaries stayed farther away from real obstacles. The calculated paths for this scenario are depicted in the work and configuration spaces in Fig. 2.11.

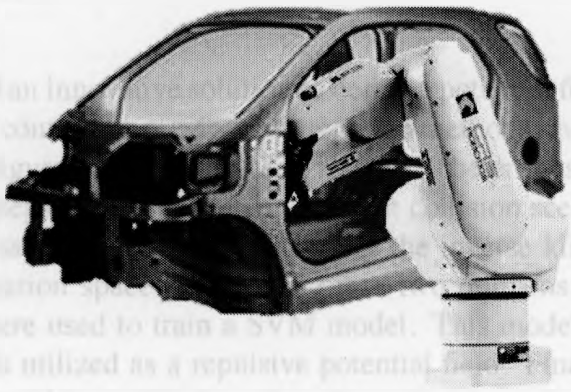
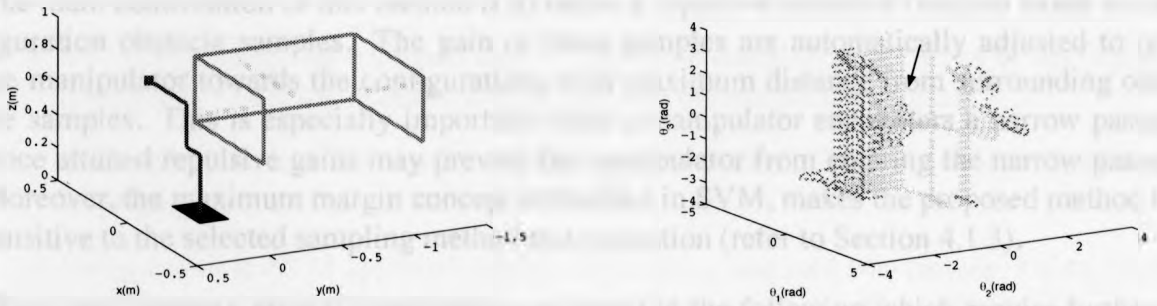
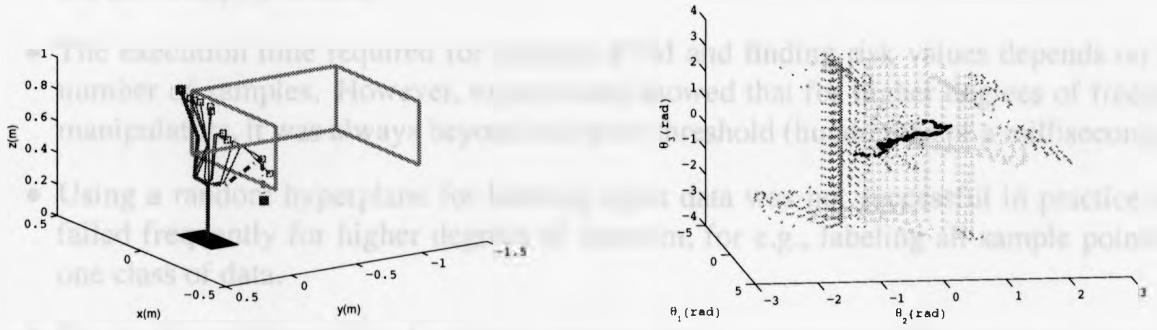


Figure 2.9: A manipulator working on a car body frame.



(A) (B)

Figure 2.10: (A) Counterpart simulation environment of Fig. 2.9. Green dashed lines are virtual obstacles.(B) Corresponding obstacle samples in configuration space. The narrow passage is emphasized by an arrow.



(A) (B)

Figure 2.11: Path obtained using the local SVM-based planner for a CRS-F3 simplified model in (A) the workspace, and (B) the configuration space.

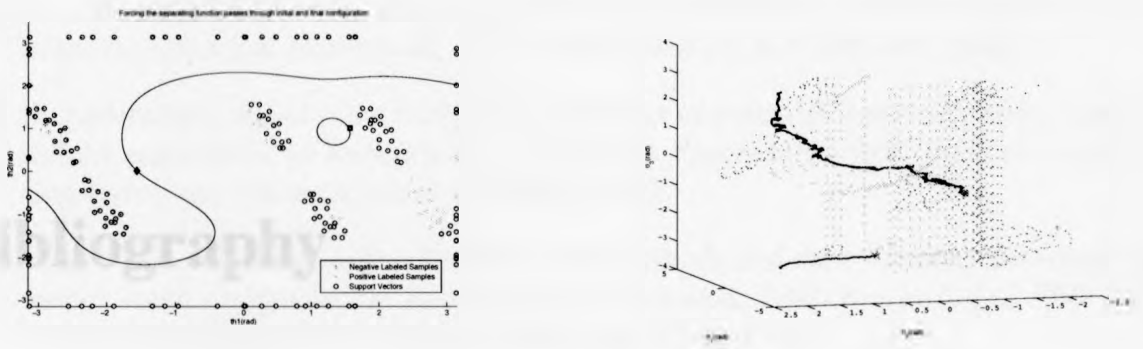
2.4 Conclusion

This chapter introduced an innovative solution to define a potential field directly over a discrete configuration space. In contrast to previously suggested methods, the proposed method needed only samples from configuration space obstacle regions. The proposed local SVM-based planner consisted of three steps. In the first step, different collision scenarios were generated and configuration obstacle samples were obtained using the inverse kinematics of the manipulator. Next, the configuration space was divided into two domains to label obstacle regions. The labeled samples were used to train a SVM model. This model contributed in defining a risk function which was utilized as a repulsive potential field. Finally, this risk function and a simple attractive potential towards the goal configuration were combined to compose a final potential field over the configuration space. The negated gradient of the resulting potential field led the manipulator towards the goal configuration while keeping it away from obstacles. Simulation results for two and three degrees of freedom models were illustrated to demonstrate the method's capabilities.

The main contribution of this method is to define a repulsive potential function based on configuration obstacle samples. The gain of these samples are automatically adjusted to repel the manipulator towards the configurations with maximum distance from surrounding obstacle samples. This is especially important when a manipulator encounters a narrow passage, since attuned repulsive gains may prevent the manipulator from entering the narrow passage. Moreover, the maximum margin concept embedded in SVM, makes the proposed method less sensitive to the selected sampling method and resolution (refer to Section 4.1.3).

There are, however, several shortcomings as listed in the following which require further investigations,

- Decision boundaries do not always form a connected component (Fig. 2.12(A)). Therefore, the problem of local minima stills exist in this method.
- The method used for sampling required the inverse kinematics of the manipulator which are not always provided.
- The execution time required for training SVM and finding risk values depends on the number of samples. However, experiments showed that for higher degrees of freedom manipulators, it was always beyond real-time threshold (hundredths of a millisecond).
- Using a random hyperplane for labeling input data was not successful in practice and failed frequently for higher degrees of freedom, for e.g., labeling all sample points as one class of data.
- Due to the problem of local minima, selecting the μ parameter in the attractive potential function is challenging.
- Due to the inclination of the SVM to find a more general obstacle region estimation, the final path may be placed far away from the optimal one (Fig. 2.12(B)).



(A) (B)

Figure 2.12: Two possible problem of Local SVM-based Planner: (A) isolated decision boundaries, and (B) a non-optimal path. In (B) the final path is shown using a solid black line.

To address the aforementioned shortcomings, some suggestions to improve the work are listed in the following,

- Utilizing a sampling approach used in sampling-based planners so as to decouple the complexity of the manipulator's structure from the sampling method,
- Embedding the risk function in a sampling-based planner in order to bias the selection of graph nodes and milestones,
- Using multiple SVM approach instead of labeling data into two classes. Using this approach, samples of each obstacle can be labeled as a class. Then the *One-Versus-Rest (OVR)* SVM idea can be employed to find the separation boundaries between classes,
- Investigating the possibility of real-time solution if the SVM optimization procedure starts from a proper initial point. In other words, since in a dynamic environment the changes between the two subsequent time steps are small, the risk function parameters found for one step are potentially a good start point for the next step optimization.

Bibliography

[1] KS Al-Sultan and MDS Aliyu. A new potential field-based algorithm for path planning. *Journal of intelligent and robotic systems*, 17(3):265–282, 1996.

[2] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628, 1991.

[3] O. Brock and O. Khatib. Elastic strips: A framework for integrated planning and execution. *Experimental Robotics VI*, pages 329–338, 2000.

[4] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. 2001.

[5] C.I. Connolly, JB Burns, and R. Weiss. Path planning using laplace's equation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 2102–2106. IEEE, 1990.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] S.S. Ge and Y.J. Cui. New potential functions for mobile robot path planning. *Robotics and Automation, IEEE Transactions on*, 16(5):615–620, 2000.

[8] D. Gingras, E. Dupuis, G. Payre, and J. de Lafontaine. Path planning based on fluid mechanics for mobile robots using unstructured terrain models. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1978–1984. IEEE, 2010.

[9] S.R. Gunn. Support vector machines for classification and regression. *ISIS technical report*, 14, 1998.

[10] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[11] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90, 1986.

[12] O. Khatib, K. Yokoi, O. Brock, K. Chang, and A. Casal. Robots in human environments: Basic autonomous capabilities. *The International Journal of Robotics Research*, 18(7):684, 1999.

[13] J.O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *Robotics and Automation, IEEE Transactions on*, 8(3):338–349, 1992.

[14] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1–6. IEEE, 1987.

[15] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404. IEEE, 1991.

[16] A. McLean and S. Cameron. The virtual springs method: Path planning and collision avoidance for redundant manipulators. *The International journal of robotics research*, 15(4):300, 1996.

[17] J. Miura. Support vector path planning. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2894–2899, 2006.

[18] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *Robotics and Automation, IEEE Transactions on*, 8(5):501–518, 1992.

[19] Saurabh Sarkar, Ernest L. Hall, and Manish Kumar. Mobile robot path planning using support vector machines. volume 2008, pages 709–715. ASME, 2008.

[20] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer-Verlag New York Inc, 2008.

[21] I. Ulrich and J. Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1572–1577. IEEE, 1998.

[22] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[23] L. Wang. *Support Vector Machines: theory and applications*, volume 177. Springer Verlag, 2005.

[24] Y. Zhang and K.P. Valavanis. A 3d potential panel method for robot motion planning. *Robotica*, 15(04):421–434, 1997.

Chapter 3

Global Support Vector Machine-based Planner

The theoretical foundations of the Support Vector Machine (SVM) algorithm were discussed in Chapter 2. In a nutshell, the SVM classifiers attempt to find a decision boundary between two classes of data by maximizing the distance of the closest data points to the decision boundary (Fig. 2.3). The same principles are applied in order to obtain a path among obstacle samples within a robot configuration space. Since exact mapping of the configuration space is often not provided, SVM can become useful in estimating obstacle regions. This estimation according to the maximal margin concept can be used to find safe configuration points in a sampled configuration space.

In Chapter 2, a local planner based on the potential fields approach was proposed. In this method, a repulsive potential field was defined using an SVM model in a manipulator's sampled configuration space. The repulsive potential field was combined with an attractive potential field of the goal configuration to create a final potential function. The negated gradient of the resulting potential function led the manipulator towards the goal configuration. However, the application of this method was restricted due to the following reasons; the first reason was that the time needed for the training of a SVM model and acquiring values of a risk function were typically long. Thus, the method did not meet the requirements of real-time planning as expected from local planners. Second, obstacle sampling in this method relied heavily on the inverse kinematics of a manipulator which are not always provided. Finally, similar to other local planners, this method still is prone to the problem of local minima.

In order to tackle the above mentioned issues, this chapter tries to develop a global sampling-based planner while taking advantage of the previously proposed risk function. Unlike the previous method, this current method is not limited to a specific type of robotic manipulator. The method is not, however, still intended for real-time applications.

A *Probabilistic Roadmap Planner* is one of the most accepted global planners which works on a discrete configuration space. The main idea behind a classical PRM is to build a graph

composed of random collision-free samples in the robot's configuration space [9]. This graph is named a roadmap and each of its nodes is called a milestone. There is an edge between two milestones if they can be connected safely using a local planner. If this roadmap captures the connectivity of the configuration space, the path planning problem boils down to a simple graph search task.

The PRM is able to solve difficult problems even in higher dimensional spaces. However, this method is not as successful when a robot's configuration space contains narrow passages. A narrow passage is recognized by the small volume of free space in a part of the configuration space. This difficulty arises due to the low probability of picking a sample from a small volume of a narrow passage [7, 9].

A number of schemes have been suggested to deal with this difficulty. The very first solution is to increase the number of milestones in narrow passages by oversampling regions identified as difficult. [8, 9]. A difficult region is a region which is important in capturing the connectivity of the configuration space and at the same maintaining the connectivity via that region is not an easy task. A region containing a milestone with a large Voronoi region or a few number of edges is typically recognized as a difficult region. As the idea suggests, these methods require a large number of referrals to the collision detection module. Moreover, the greater number of milestones increases the required time for making a roadmap [15].

Another group of methods try to decrease the number of milestones obtained by oversampling to speed up the process of making a roadmap. The *Gaussian Sampler* and *Bridge-Test* methods retain only the milestones that are close to the obstacle boundaries [3, 6]. This approach increases the number of milestones in narrow passages since all narrow passages are surrounded by obstacles. These methods sample the configuration space in pairs. In other words, for any random configuration, another sample in its neighborhood according to a Gaussian (normal) distribution is selected. The former method keeps a collision-free sample as a milestone if the other sample in the pair is in collision. Similarly, the bridge-test selects the mid-sample of the pair as a milestone if it is collision-free and the two samples of the pair are in collision. Even though selecting milestones close to obstacles is a promising heuristic, the results are strongly dependent on the selected standard deviation for the Gaussian distribution. In the Bridge-Test method, an attuned parameter can decrease the number of milestones drastically; even it is possible to select no milestone in a narrow passage. To address this issue an appropriate value for the standard deviation parameter needs to be selected. However, this requires the topology of the configuration space which is not always available. In addition, neither of these methods is able to guarantee a safe margin for the obstacles and may place a milestone in a touching position. Unlike previous methods, *Obstacle Based PRM (OBPRM)* tries to sample obstacle boundaries instead of the free space (Fig. 1.6) [2, 1]. Similar to the Gaussian and bridge-test sample, this method increases the number of milestones in a narrow passage. However, the approach suffers from a number of drawbacks including the touching position of milestones and restrictions applied on the shape of the configuration space obstacles.

The last category of methods intended to improve the classical PRM works on a modified configuration space and then transfers the solution to the original one. These methods are called

Dilation and Retraction-Based methods [12, 14]. These methods shrink the manipulator links and the workspace obstacles to broaden the corresponding narrow passages in the configuration space. The milestones selected in this space need to be fixed before being transferred to the intact configuration space. In other words, milestones positioned on the actual obstacles must be repaired. One idea is to correct all misplaced milestones. However, this approach is not efficient since the milestones deeply buried into the obstacles are a bottleneck for the performance of the algorithm. An alternative is to compute a penetration depth for each milestone which comes at the expense of complex geometry operations. These operations are difficult to implement in higher dimensional spaces.

Considering the shortcomings of the previous methods and the local SVM-based planner, a new global sampling-based planner is proposed in this chapter. There are two main motivations behind this approach. The very first motive is that unlike free-space sampling, obstacle sampling does not encounter the problem of picking samples from the small volume of a narrow passage. The second is that providing no information about the obstacle boundaries, the probability of a collision, intuitively, is decreased by selecting configurations that are farther from obstacles samples. The proposed method builds a risk function in the configuration space based on samples from the obstacle regions. This risk function is calculated using the SVM method. As a result, its local minima are placed in maximum distance from the surrounding obstacle samples. This risk function is used to spur the milestones to appropriate safe positions.

The rest of this chapter is organized as follows; Section 3.1 presents a new method for obstacle sampling. Section 3.2 explains the procedure of calculating a risk function. This risk function plays an important role in the proposed method. Section 3.3 describes the approach for generating milestones followed by making a roadmap. Simulation results related to a 2 DoF point robot are discussed in Section 3.4. Finally, Section 3.5 concludes the chapter and lists some of possible future directions.

3.1 Obstacle Sampling

The idea behind obstacle sampling is to avoid the exact mapping of an obstacle into the configuration space. Since exact mapping of the obstacles is computationally expensive, the planning problem is solved in a discrete (sampled) configuration space. These obstacle samples are used to estimate the obstacle regions in the configuration space.

Therefore, the very first step in the proposed sampling-based planner is to obtain configuration space obstacle samples. Since each obstacle region is estimated only based on its samples, samples from different obstacles must be recognizable. The problem is that even for convex workspace obstacles, it is not guaranteed that the corresponding configuration space obstacles are connected [13]. Thus, it is not feasible to use a uniform sampler.

Instead, an obstacle sampling approach similar to *Rapidly-exploring Random Trees (RRT)* [11] is proposed (Algorithm 1). This approach grows a tree similar to RRT to find obstacle samples

(Fig. 3.1). The only difference is that trees instead of free-space are expanded over obstacle regions. This approach of tree expansion biases the expansion towards unexplored regions. This is due to the fact that a node is selected for expansion by a probability that is proportional to the area of its Voronoi region [10].

The obstacle sampling method grows a number of trees typically equal to the number of obstacle regions. Each tree is rooted at a given configuration space obstacle sample selected from a specific obstacle region. In order to expand a tree, at each iteration a random configuration, q_{rand}, is generated. Then, the nearest node of the tree to this configuration, $q_{expansion}$, is determined. Next, a new configuration, q_{new}, is calculated such that it is distanced one step from $q_{expansion}$ towards q_{rand}. If q_{new} is in collision, it will be added to the tree. Otherwise, $q_{expansion}$ will be labeled as a boundary sample and q_{new} is ignored. The samples marked as boundary will be used later to find milestones. The expansion procedure continues until a predetermined number of nodes are added to the tree.

Algorithm 1 Obstacle Sampling

1: $samples \leftarrow \emptyset$
2: Add q_0 to *samples*
3: N:predefined number of samples requested per CSpace obstacle
4: **while** *samples.size* $< N$ **do**
5: $q_{rand} \leftarrow$ random sample from CSpace
6: $q_{expansion} \leftarrow$closest node to q_{rand}
7: Calculating q_{new} by expanding $q_{expansion}$ toward q_{rand} for one step
8: **if** q_{new} is in free CSpace **then**
9: Label $q_{expansion}$ as *boundary sample*
10: **end if**
11: Add q_{new} to *samples*
12: **end while**
13: **return** *samples*

3.2 Risk Function

The risk function is the core component of the proposed method. This function determines the configurations with maximum distance from surrounding obstacle samples (i.e. the middle of a narrow passage). These configurations have a high probability to be safe and are suitable for placing milestones.

A risk function is calculated based on a trained SVM model over the obtained configuration space obstacle samples. Assume that samples related to two obstacle regions are provided. Using two different labels for samples from different regions and applying SVM, a decision function, $F(\mathbf{q}, \mathbf{w}_0, b_0)$, is obtained (Fig. 3.2(A)). As explained in Section 2.1, this decision function divides the configuration space into three domains, i.e.,

- if $F(\mathbf{q}, \mathbf{w}_0, b_0) > 0$, $\mathbf{q}$ belongs to Class +1.
- if $F(\mathbf{q}, \mathbf{w}_0, b_0) < 0$, $\mathbf{q}$ belongs to Class -1.
- if $F(\mathbf{q}, \mathbf{w}_0, b_0) = 0$, $\mathbf{q}$ belongs to the decision boundaries.

Since the SVM method maximizes the margin between two classes, this function is optimized to place the decision boundaries at maximum distance from surrounding obstacles (Fig. 3.2(C)). Depending on selected values of the SVM parameters, the maximum distance may not be always satisfied . Nevertheless, these configurations are a proper choice for placing milestones.

The linear SVM method is not an appropriate choice for estimating configuration space obstacles. Therefore, a non-linear SVM kernel must be used in training a SVM model. The problem is that by using a non-liner kernel, the decision boundaries cannot be calculated analytically. Therefore, to provide a systematic procedure to reach the configurations on the decision boundaries, a function is defined as,

$$R(\mathbf{q}) = |F(\mathbf{q}, \mathbf{w}_0, b_0)|. \tag{3.1}$$

This non-negative function with local minima on decision boundaries is called a risk function (Fig. 3.2(B)). Hence, starting at a configuration $\mathbf{q}$, the negated gradient of $R(\mathbf{q})$ can be followed to reach a local minimum, i.e., a point on the decision boundaries.

A similar solution can be easily extended to the cases with more than two obstacle regions. This is achieved by associating a risk function to each obstacle region. In other words, first all samples related to one obstacle region are labeled as one class. Next, a multi-class SVM approach known as *One-Versus-Rest (OVR)* is taken to find a set of risk functions. Assuming n classes of data exist, this approach trains n SVM models such that each model separates one class from the rest of the classes. Consequently, n number of risk functions are obtained based on the trained models. The resulting risk function in each scenario is assigned to a region which is classified against the others.

3.3 Obtaining Milestones and Making a Roadmap

To obtain milestones and create a roadmap, multiple obstacle regions were sampled and each region was associated with a risk function. Each of these risk functions has its local minimum in a maximum distance from surrounding obstacles. This condition is not always satisfied depending on the choice of the SVM kernel and its parameters. The kernel and its parameters determine the curvature of the risk function and the region of influence for each sample (refer to Section 4.1.1). Hence, a risk function is not always able to represent a narrow passage or obstacle boundaries.

As mentioned previously, in a sampled configuration space, the configurations that are more distanced from obstacle samples are more likely to be safe and collision-free. This is exactly

the condition that is fulfilled on the local minimum of a risk function. The local minimum coincides with the decision boundaries of the corresponding SVM model. Due to the use of non-linear kernels, these decision boundaries can not be calculated analytically. However, starting from a random configuration and seeking the negated gradient of a risk function typically end in a point on the decision boundaries. Because a risk function is not monotonic all over the configuration space, this procedure does not always reach the decision boundaries. However, selecting a monotonic SVM kernel, the risk function will be monotonic near the decision boundaries. Because these boundaries are placed among obstacles, samples close to the obstacle boundaries have higher probability to fall in these regions. Therefore, instead of adopting random starting points, the previous obstacle samples labeled as boundary are chosen. The boundary samples from each region follow the negated gradient of the risk function associated with that region to reach the local minimum. The final point will be added as a milestone if it is collision-free (Fig. 3.3).

There are several advantages in this approach of selecting milestones. One, due to unknown topology of the configuration space, re-using boundary obstacle samples speeds up the process with fewer referrals to the collision detection module. Second, if a region is sampled sufficiently, this method distributes the milestones more uniformly over the configuration space. Moreover, a proper model and risk function pushes the milestones towards configurations with maximum distance from surrounding obstacle samples, i.e., the middle of a narrow passage. Finally, provided a sufficient number of samples, milestones can be placed in a known distance from obstacle samples (refer to Section 4.1.1). This distance is adjustable by appropriate selection of the kernel parameter.

After obtaining a set of milestones, in order to make a roadmap, each milestone becomes connected to a predetermined number of closest milestones not in the same connected components (Fig. 3.4). If this connection is feasible using a local planner, an edge will be added between the two milestones. Creating a roadmap, each query will be answered by searching the roadmap for a path between closest nodes to the requested initial and goal configurations.

3.4 Simulation Results and Implementation Notes

The proposed method was implemented in the C++ language. The *libsvm* and *Robust and Accurate Polygon Interference Detection (RAPID)* libraries were used to train SVM models and detecting collisions, respectively [4, 5]. All simulations were conducted on a 2.53 GHz Intel Core 2 Duo CPU. The proposed method was evaluated on a 2 DoF point robot in the simulated environments. The results of the simulation for the point robot are demonstrated in this chapter and compared to the classical PRM results. The rest of the results along with parameter analysis are discussed in the next chapter.

The first simulation environment for a 2 DoF point robot is shown in Fig. 3.5. An identical problem is solved using both the classical PRM and global SVM-based planners with the same number of samples picked from the configuration space.

Table 3.1: Comparison of the number of referral to the collision detection module (CD) and the local planner (LP) for the classical PRM and the proposed SVM-based method (SVMP). The number of generated milestones and the execution times are compared.

Environment	Planner	CD	LP	Milestones	Time (ms)
Fig. 3.5	SVMP	2384±108	97±7	47±2	74±2
	PRM	4297±187	284±58	168±5	91±9
Fig. 3.7	SVMP	8027±408	756±117	190±9	1041±48
	PRM	16073±1917	2919±601	962±20	872±32

The comparison between the results of the two methods are depicted in Fig. 3.6. In this experiment both planners were able to find a path but the PRM could not pass through the narrow passage.

In another experiment, the proposed method was evaluated in a more challenging environment (Fig. 3.7). The narrow passage in this environment is longer and preserving the connectivity of the configuration space is more difficult.

Similar to the previous experiment, this problem was solved using both the classical PRM and global SVM-based planners. The comparisons of the results are shown in Fig. 3.8. In this experiment, the classical PRM was not able to capture the connectivity in several different executions.

Table 3.1 compares the number of referral to collision detection module and the number of times that the local planner was called in multiple execution of previous studies. As it was expected, the number of milestones and referral to the collision detection module is significantly less in the global SVM-based planner. However, in these simulations, the total execution time of the classical PRM is typically less than the global SVM-bsed planner. The reason is that training of SVM models and acquiring the risk function values are time consuming.

3.5 Conclusion

This chapter proposed a global SVM-based planner that improved the capability of the classical PRM planner in environments containing narrow passages. The proposed method started by sampling obstacle regions using a method similar to RRT. Next, the resulting samples were classified into a number of classes. The One-Versus-Rest SVM approach was utilized to build a set of risk functions. These risk functions had their local minimum on the corresponding SVM model decision boundaries. Thus, they were used to place milestones at maximum distance from obstacle samples. The proposed method was tested on a 2 DoF point robot. The results of the simulation performed on the point robot were discussed to show the ability of the planner to solve problems containing narrow passages.

The main contribution of the proposed method is related to the definition of a risk function.

This function, which is calculated based on a SVM model, is able to determine the safe configurations with maximum distance from surrounding obstacle samples. Sampling the obstacles instead of the free space alleviates the problem of picking samples from the small volume of a narrow passage. Since no boundary information about the obstacles are provided, staying at a maximum distance decreases the probability of a collision. Due to this approach, the proposed method is able to use fewer milestones that are placed in appropriate positions to capture the connectivity of the configuration space. The fewer number of milestones and the method used for selecting milestones, decreases the number of referral to the collision detection module. The smaller number of referral to the collision detection module improves the speed of the planner in challenging situations. The method also distributes the milestones more uniformly over the configuration space. Utilizing SVM and the notion of maximal margin embedded in SVM results in a planning method which is less sensitive to the sampling method and the resolution.

There are, however, some issues which need further investigations,

- Alternative obstacle sampling methods that focus on boundary samples can replace the current sampling method.
- Due to the fact that a SVM model is trained over the entire configuration space, the act of acquiring a risk function value becomes time consuming. However, based on the selection of the SVM kernel parameters, the influence of the sample points far from a given configuration is zero at that configuration. Hence, some heuristics can be used to obtain the result of a risk function faster.
- Even though the proposed planner tries to place milestones in safe positions, the resulting roadmap may be composed of different connected components. Therefore, improving roadmap connectivity must be the next step. This is achievable by generating random points near a region of disconnectivity and guiding them towards the decision boundaries. This approach increases the number of milestones and consequently improves the roadmap connectivity.
- In situations in which obstacle regions are dominant, more samples are necessary in order to have a good estimation of the obstacle regions. In this situation the classical PRM may perform better. Thus, the idea of combining these methods should be investigated.
- Since the method is less sensitive to sampling resolution, utilizing the idea of incremental sampling can potentially improve the performance.
- Since in dynamic environments the change between two subsequent time steps is not substantial, the result of the SVM in one step can be used as the initial point for the next step optimization. Using this approach, it is conjectured that the global SVM-based planner can be extended to real-time environments.

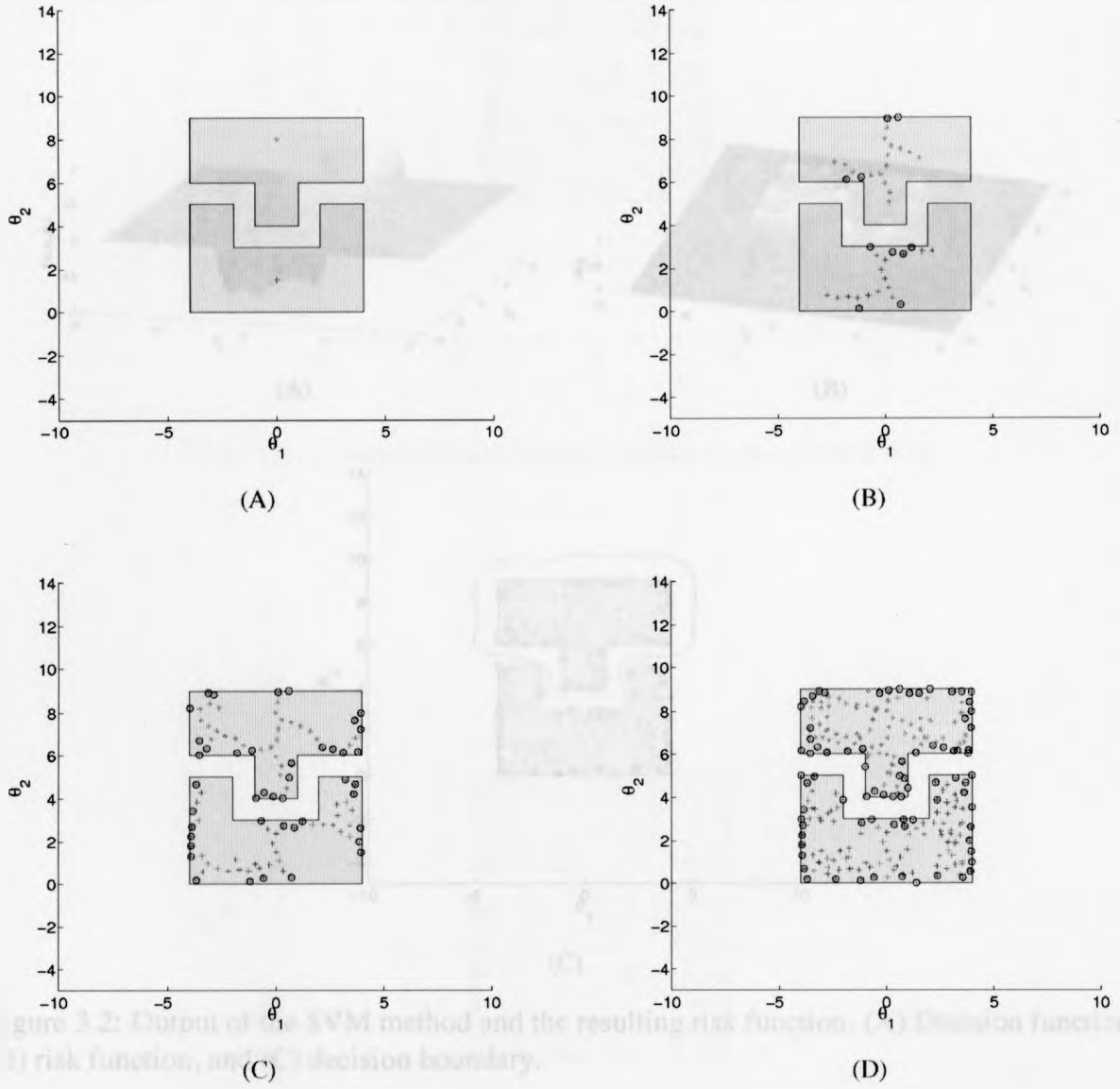
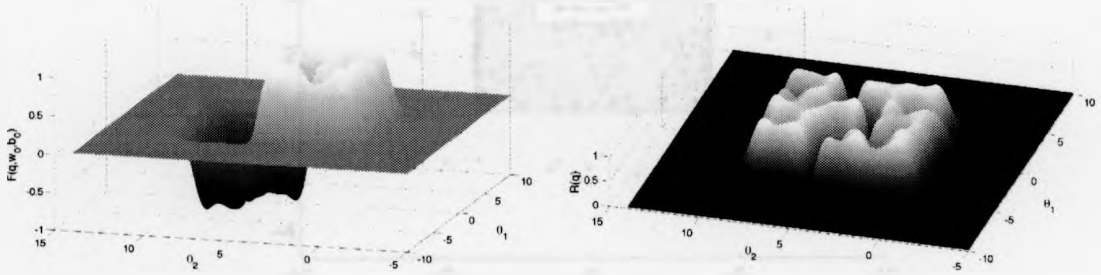
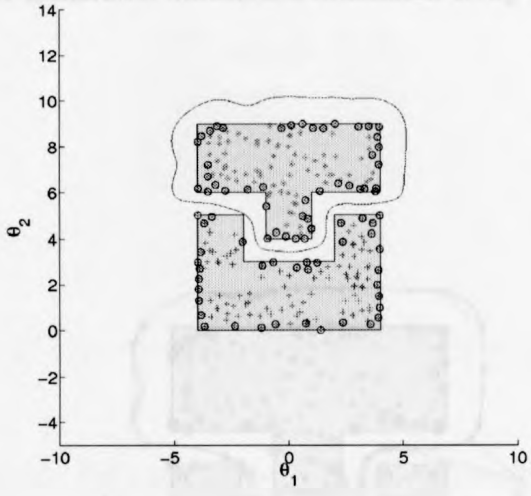


Figure 3.1: Different steps of obstacle sampling using RRT. Boundary samples are enclosed in circles. The samples from two different obstacles are labeled as two classes.



(A) (B)



(C)

Figure 3.2: Output of the SVM method and the resulting risk function. (A) Decision function, (B) risk function, and (C) decision boundary.

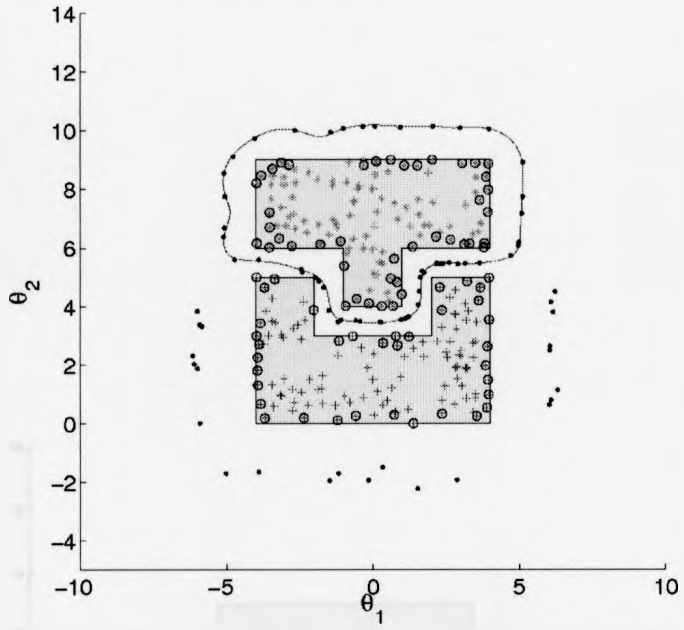


Figure 3.3: Generated milestones illustrated using black dots.

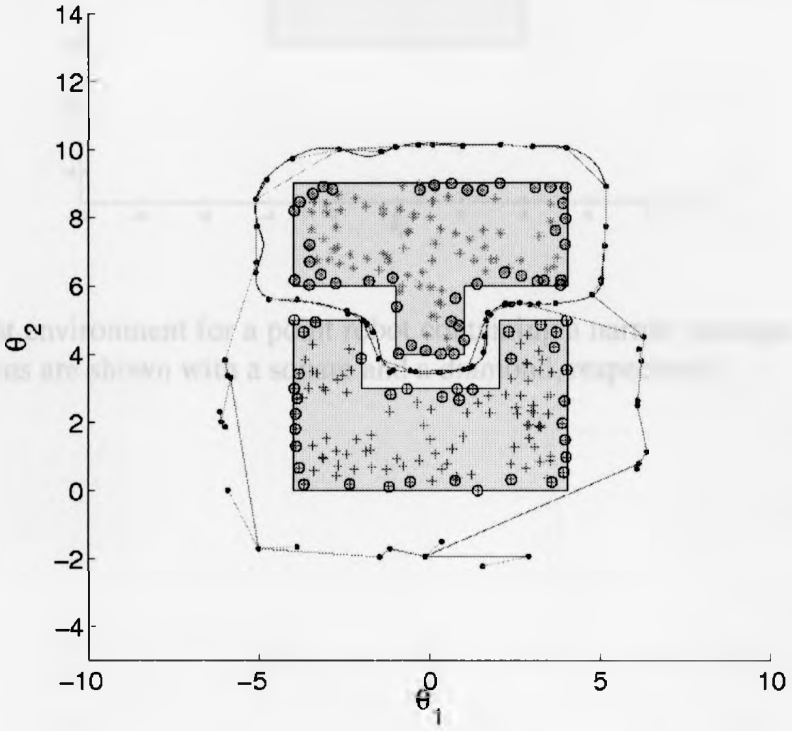


Figure 3.4: Resulting roadmap created based on generated milestones (shown as black dots).

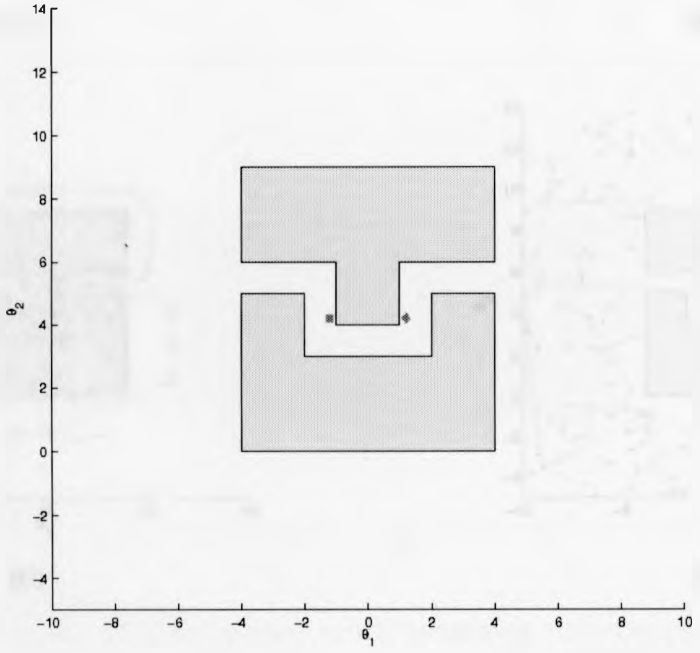


Figure 3.5: A test environment for a point robot containing a narrow passage. The Initial and goal configurations are shown with a square and a diamond, respectively.

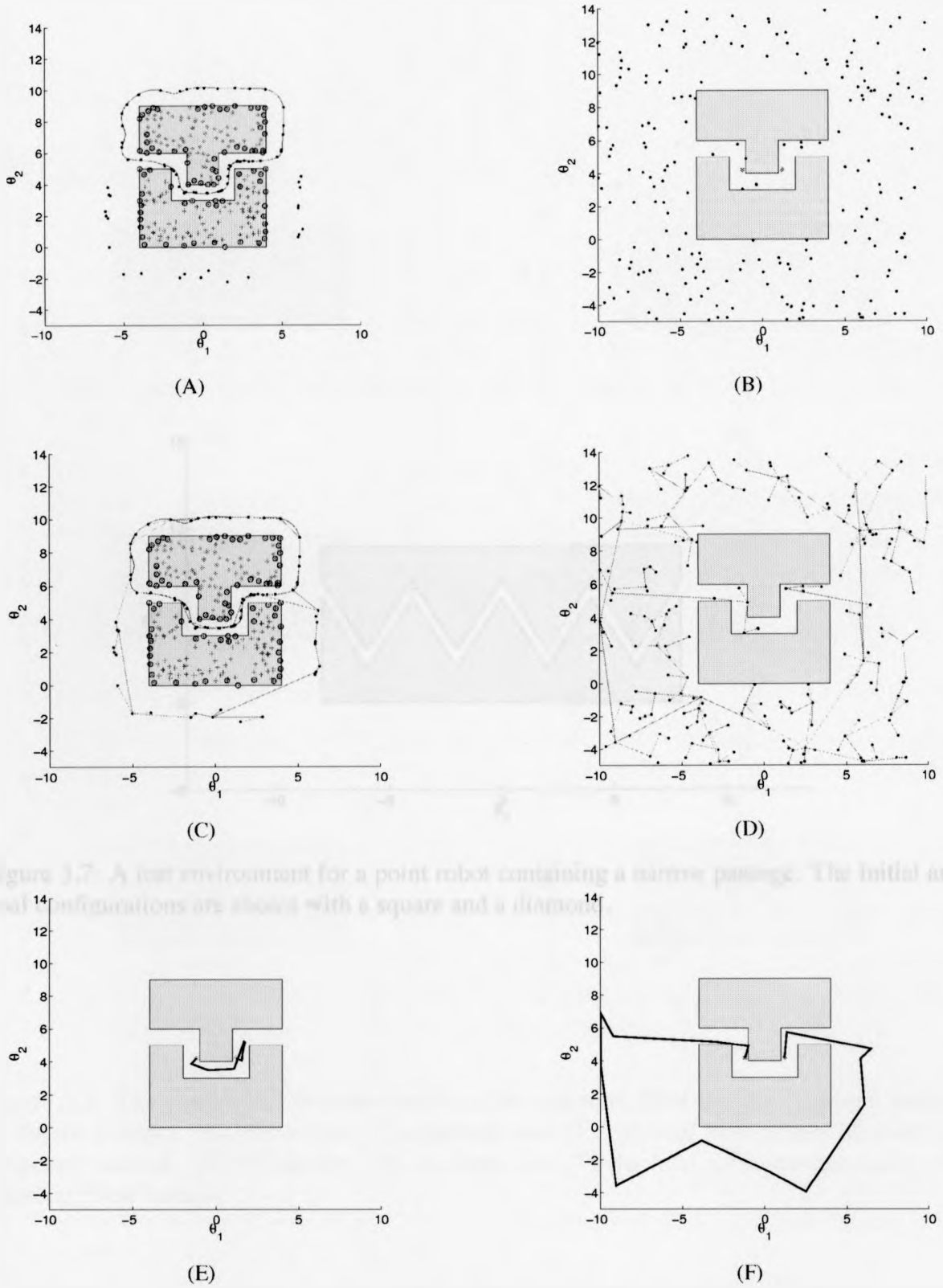


Figure 3.6: Comparison between the results of the classical PRM and the proposed SVM-based planner in different steps. (A) milestones, (C) roadmap, and (E) the final path generated using the proposed method. (B) Milestones, (D) roadmap, and (F) the final path generated using the classical PRM method.

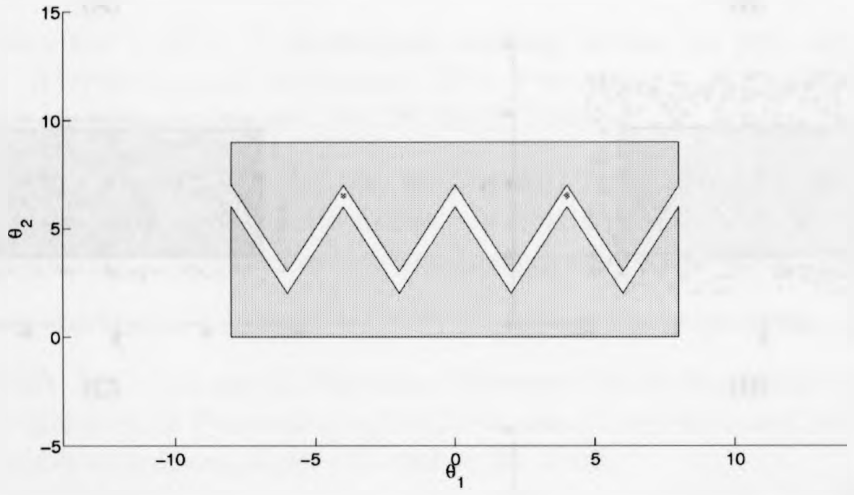


Figure 3.7: A test environment for a point robot containing a narrow passage. The Initial and goal configurations are shown with a square and a diamond.

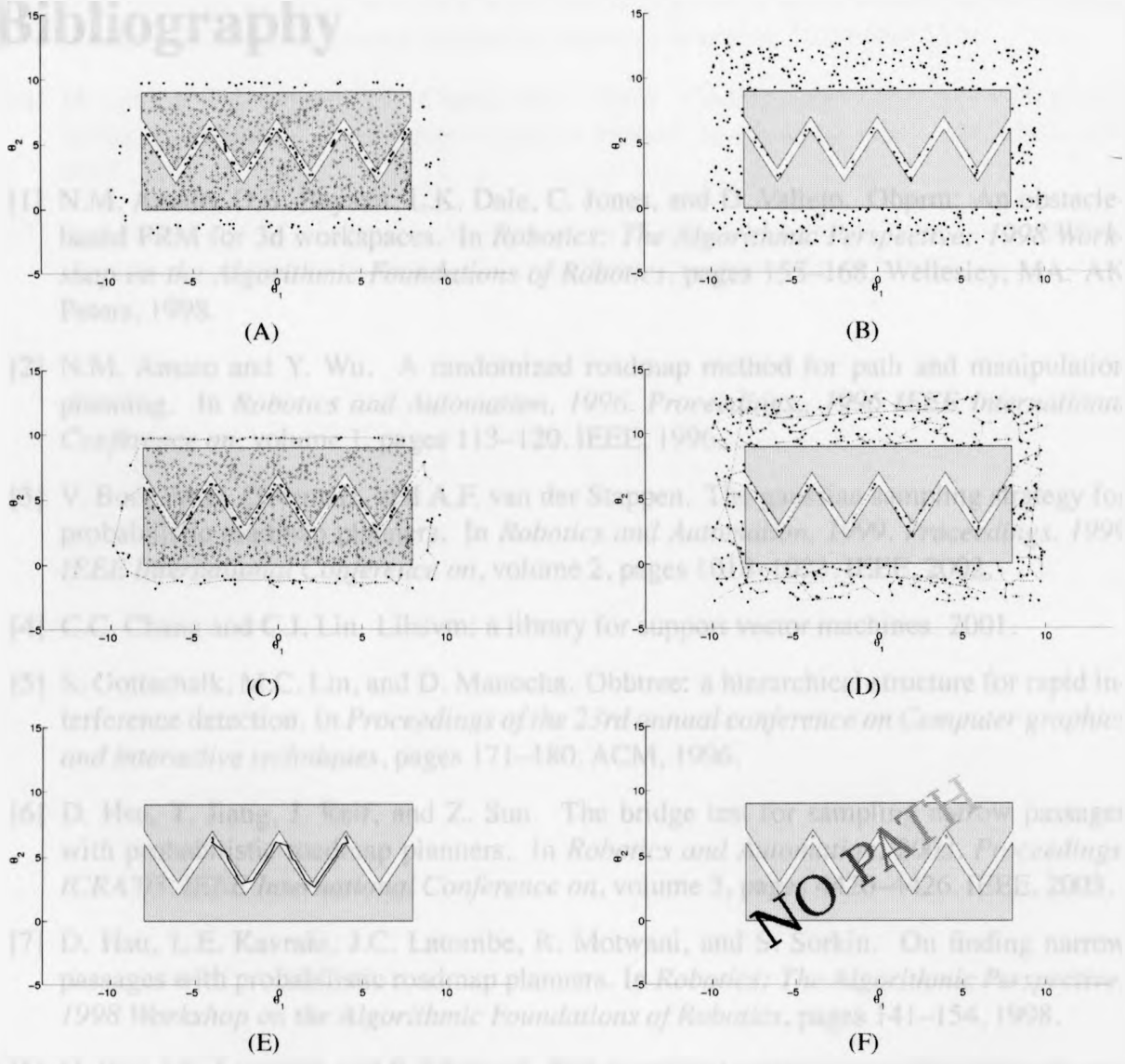


Figure 3.8: The comparison between results of the classical PRM and the proposed method in different steps. (A) Milestones, (C) roadmap, and (E) the final path generated using the proposed method. (B) Milestones, (D) roadmap, and (F) the final path generated using the classical PRM method.

Bibliography

[1] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based PRM for 3d workspaces. In *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pages 155–168. Wellesley, MA: AK Peters, 1998.

[2] N.M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 113–120. IEEE, 1996.

[3] V. Boor, M.H. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1018–1023. IEEE, 2002.

[4] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. 2001.

[5] S. Gottschalk, M.C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

[6] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4420–4426. IEEE, 2003.

[7] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, pages 141–154, 1998.

[8] D. Hsu, J.C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2719–2726. IEEE, 1999.

[9] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[10] J.J. Kuffner Jr and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

[11] S.M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[12] J.M. Lien, S.L. Thomas, and N.M. Amato. A general framework for sampling on the medial axis of the free space. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4439–4444. IEEE, 2003.

[13] W.S. Newman and M.S. Branicky. Real-time configuration space transforms for obstacle avoidance. *The International Journal of Robotics Research*, 10(6):650, 1991.

[14] M. Saha, J.C. Latombe, Y.C. Chang, and F. Prinz. Finding narrow passages with probabilistic roadmaps: The small-step retraction method. *Autonomous robots*, 19(3):301–319, 2005.

[15] G. Sánchez and J.C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. *Robotics Research*, pages 403–417, 2003.

Chapter 4

Case Studies and Experimental Results

The previous chapter introduced a new sampling-based planner that utilized the notion of a risk function. This planner differs from a classical PRM planner in the configuration space sampling method and generating milestones. The proposed method began by sampling obstacle regions instead of the free space to create a set of risk functions. Each risk function was optimized using the SVM method to have its local minimum at maximum distance from surrounding obstacle samples. Finally a set of milestones was generated on local minima of the calculated risk functions to make a roadmap. The method was tested on a two degrees of freedom point robot in two challenging environments and the results were discussed.

This chapter studies the same method more throughly in higher dimensional spaces. Moreover, the chapter investigates the effect of different parameters involved. All designed scenarios were solved using both the proposed method and the classical PRM planner and compared. The comparison of the results are reported. The main purpose of these studies is to evaluate the performance of the proposed planner in higher dimensional spaces. All case studies were, to some extent, challenging in a sense that the configuration space contains narrow passages. The simulation studies were performed using high degree of freedom planar robots and a CRS-F3 articulated manipulator. The final solutions obtained for the CRS-F3 were implemented on a real CRS-F3 manipulator to verify their accuracies.

The rest of the chapter is organized as follows; Section 4.1 discusses the effect of different parameters involved in the method. Section 4.2 illustrates different case studies and compares the results of the global SVM-based planner with classical PRM. Section 4.3 discusses about some implementation notes to improve the method. Moreover, this section describes the structure of the developed application for later references. Finally, Section 4.4 concludes the chapter.

4.1 Parameter Analysis

There are different parameters involved in global SVM-based planners. Two of these parameters are related to the SVM algorithm and are common between the proposed local and global planners. In addition, there are parameters regarding obstacle sampling method which are discussed in this section.

4.1.1 SVM Kernel

There are many different choices for a SVM kernel, some of which are listed in Table 2.1. Due to the nature of the problem at hand, the only sensible assumption is that the influence of an obstacle sample decreases as the distance from the obstacle increases. Thus, the choices of the kernel are limited to monotonic kernels. In this body of work, the Radial Basis Gaussian Function (RBF) was selected as the kernel for SVM models, i.e.,

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\gamma|\mathbf{x}_i - \mathbf{x}_j|^2\right) \tag{4.1}$$

The γ is the only adjustable parameter in this kernel, which is related to the region of influence of each obstacle sample. Increasing γ shrinks the region of influence of each sample and vice versa. To analyze this parameter, an equivalent notation of the RBF kernel was used, i.e.,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2}|\mathbf{x}_i - \mathbf{x}_j|^2) \tag{4.2}$$

where $\gamma = \frac{1}{2\sigma^2}$. The σ parameter in this definition, unlike γ, has a direct relation to the region of influence. In other words, increasing σ broadens the region of influence of each sample and vice versa. Therefore, this parameter can directly control the margin between a decision boundary and a class of samples where no other classes interfere. This parameter, also, affects the curvature of a risk function and decision boundaries. Several decision boundaries are shown in Fig. 4.1 related to different values of σ. As illustrated, a large value of σ decreases the curvature of the decision boundaries whereas small values of σ cause overfitting.

Even though setting a proper value for this parameter requires knowledge about the configuration space, by selecting a small value of σ and a high sampling resolution we can overcome this issue. In this case, the larger number of samples compensates for the problem of overfitting which may happen due to the small value of σ. It is recommended to set σ such that the influence regions of two neighboring samples have an overlap with each other. This approach avoids overfitting and does not let the SVM optimization algorithm divide an obstacle region into smaller regions.

4.1.2 Penalty Parameter (C)

This parameter becomes especially important in building a model where zero training error is not possible (refer to Section 2.1.2). In a planning problem, due to unknown topology of

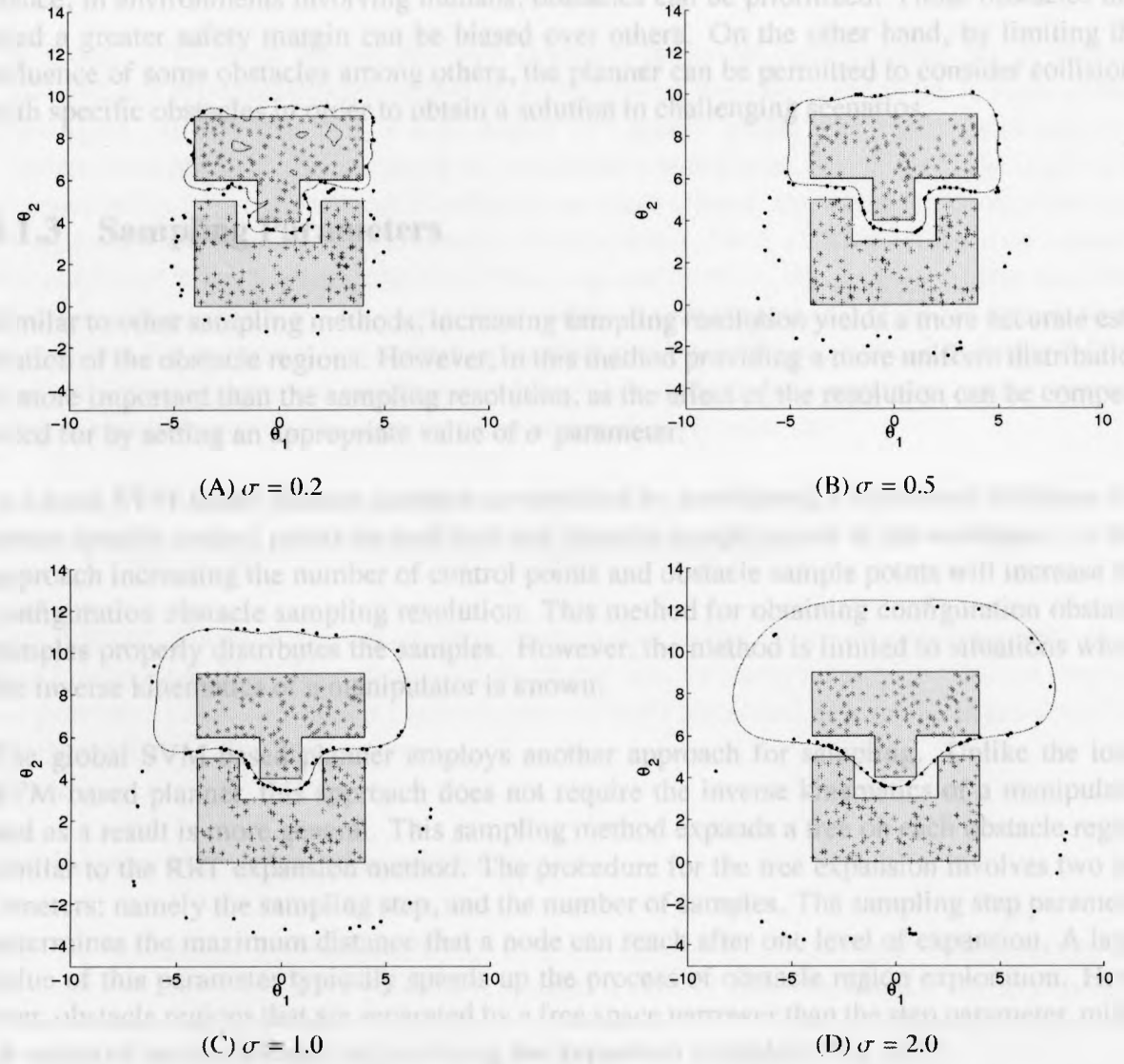


Figure 4.1: Effect of the parameter σ on the curvature of the decision boundaries (blue solid line) and estimated obstacle regions. The width of the narrow passage is 1.0 in the simulated environment. The milestones and two classes of obstacle samples are shown in black dots, blue plus signs, and red stars, respectively.

the configuration space and inaccurate nature of the sampling methods, estimating a proper curvature using σ is difficult. Thus, it is probable to have a model with non-zero training error.

The penalty parameter is intended to bias some samples over others. These biased samples are able to have more influence on the decision boundaries and a risk function as shown in Fig. 4.2. According to (2.16) the amount of the influence of each sample, α_i, is constrained by C. Hence, in environments involving humans, obstacles can be prioritized. Those obstacles that need a greater safety margin can be biased over others. On the other hand, by limiting the influence of some obstacles among others, the planner can be permitted to consider collisions with specific obstacles in order to obtain a solution in challenging scenarios.

4.1.3 Sampling Parameters

Similar to other sampling methods, increasing sampling resolution yields a more accurate estimation of the obstacle regions. However, in this method providing a more uniform distribution is more important than the sampling resolution, as the effect of the resolution can be compensated for by setting an appropriate value of σ parameter.

In a local SVM-based planner, samples are obtained by considering a number of collisions between specific control points on each link and obstacle sample points in the workspace. In this approach increasing the number of control points and obstacle sample points will increase the configuration obstacle sampling resolution. This method for obtaining configuration obstacle samples properly distributes the samples. However, the method is limited to situations where the inverse kinematics of a manipulator is known.

The global SVM-based planner employs another approach for sampling. Unlike the local SVM-based planner, this approach does not require the inverse kinematics of a manipulator and as a result is more general. This sampling method expands a tree on each obstacle region similar to the RRT expansion method. The procedure for the tree expansion involves two parameters; namely the sampling step, and the number of samples. The sampling step parameter determines the maximum distance that a node can reach after one level of expansion. A large value of this parameter typically speeds up the process of obstacle region exploration. However, obstacle regions that are separated by a free space narrower than the step parameter, might be captured as one obstacle region during the expansion procedure (Fig. 4.3).

The other parameter is the maximum number of nodes that each tree contains. Each tree is expanded until the number of its nodes reaches a predetermined value. Even though a greater number of samples improves the estimation, it also increases the required processing time for training a model and calculating risk function values. Typically, picking a larger number of samples results in discovering a larger number of boundary samples. This, consequently, results in a greater number of milestones.

Taking a closer look at Fig. 4.3, a behavioral model can be defined for the σ parameter. The smaller sigma values should be selected as the number of samples or the step size in-

creases. However, this statement is true if different obstacle samples do not cross the other region boundaries as shown in the last column of Fig. 4.3.

4.2 Case Studies and Experimental Results

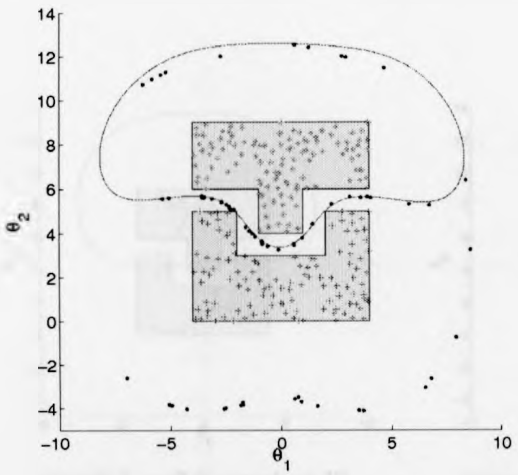
The global SVM-based planner has been evaluated in several environments. We considered simulated and experimental setups and all environments were designed to be, to some extent, challenging. In the first case, a high degree of freedom planar manipulator was used in a simulated environment. Even though the manipulator was planar, the solution was required to be generated in a high dimensional configuration space. Hence, the fact that a manipulator was a planar one did not reduce the complexity of the problem. Next, a CRS-F3 robotic manipulator was employed to study the capabilities of the proposed method. The generated paths were also implemented on a real CRS-F3 manipulator and the results proved the validity of the solutions. In addition, all problems were solved using both a SVM-based planner and a classical PRM multiple times. The average results over successful executions are reported and compared when the success rates were in the same range and more than 60 percents.

4.2.1 High Degree of Freedom Planar Manipulator

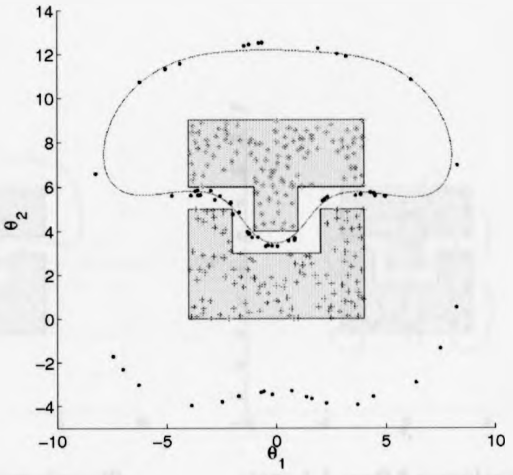
Two different planar manipulators were used in this study. Since all solutions were to be found in the manipulator's configuration space, the use of a planar manipulator did not simplify the problem. These two planar manipulators and their simulated environments are described separately in the following sections.

Five Degrees of Freedom Planar Manipulator

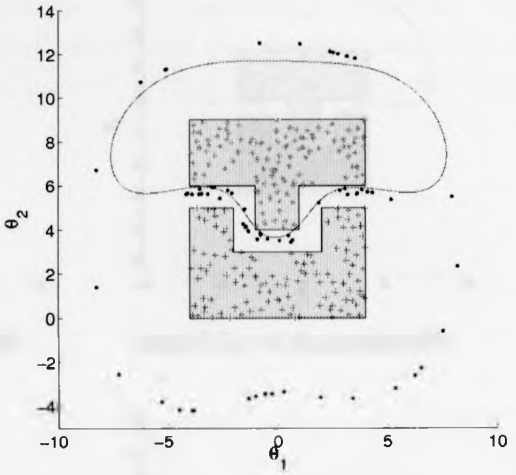
The first simulation environment along with the desired initial and goal configurations are illustrated in Fig. 4.4. Such a problem represents a class of environments for which a solution cannot be easily obtained using local planning methods. The solution obtained using the global SVM-based planner is shown in Fig. 4.5. Table 4.1 shows the comparison of the results with those obtained using a classical PRM planner. This comparison clearly shows that the time spent on the training of SVM models and acquiring risk function values can significantly slow down the process in comparison to a classical PRM method. However, as expected, the global SVM-based can solve the problem with fewer number of milestones and referrals to the collision detection module.



(A) $C_{upper} = 1, C_{lower} = 1$



(B) $C_{upper} = 1, C_{lower} = 2$

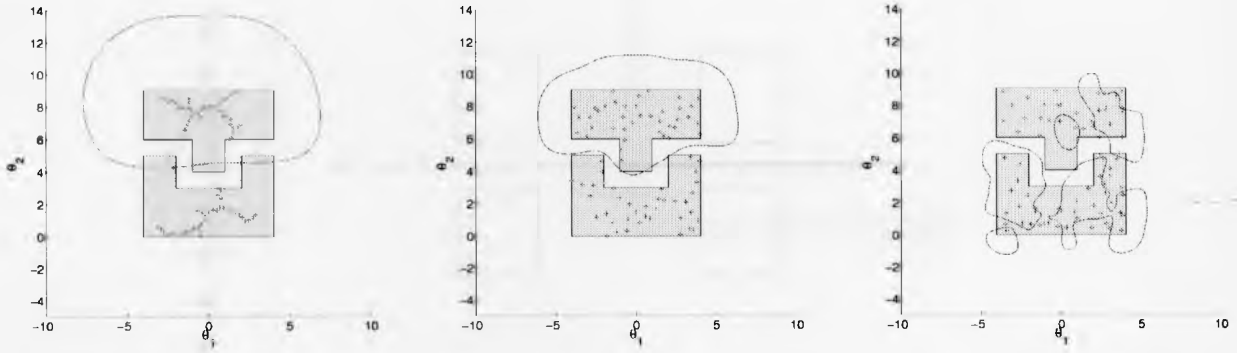


(C) $C_{upper} = 1, C_{lower} = 10$

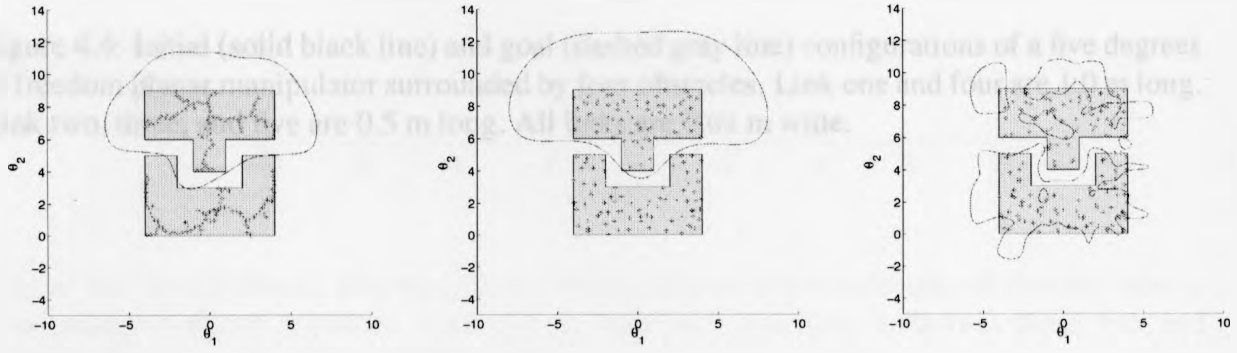
Figure 4.2: Effect of the penalty parameter C on biasing obstacle samples. Samples of the lower shape are biased in (B) and (C).

Table 4.1: Comparison of the global SVM-based (SVMP) and the classical PRM planner in the environment shown in Fig. 4.4. CD and LP refer to the number of referrals to the collision detection and local planner modules, respectively.

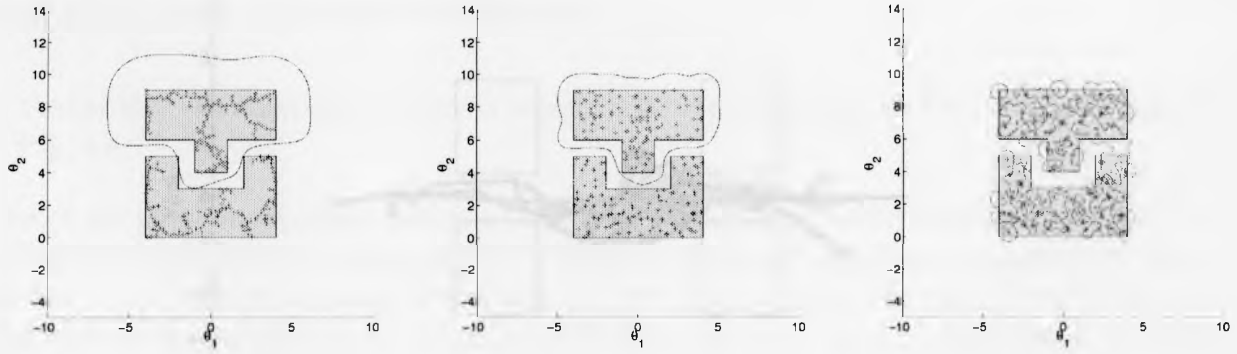
	CD	LP	Milestones	Time(ms)
SVMP	17993±1745	1770±383	171±8	102487±5074
PRM	21833±1874	3165±467	324±20	895±71



step=0.3,σ=2.0,samples=40 step=0.7,σ=1.0,samples=40 step=1.1,σ=0.5,samples=40



step=0.3,σ=1.0,samples=80 step=0.7,σ=1.0,samples=80 step=1.1,σ=0.5,samples=80



step=0.3,σ=1.0,samples=120 step=0.7,σ=0.5,samples=120 step=1.1,σ=0.2,samples=120

Figure 4.3: Effect of different sampling parameters. A decision boundary formed using a proper value of σ is shown with a solid line.

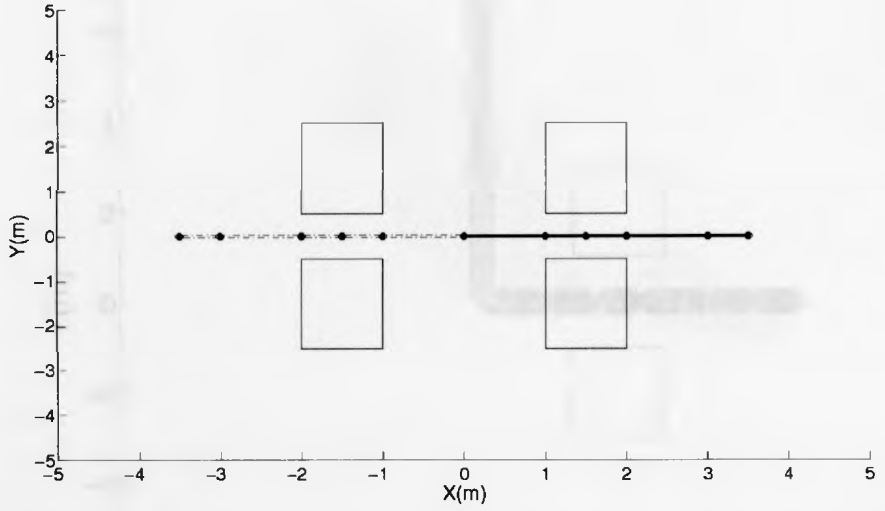


Figure 4.4: Initial (solid black line) and goal (dashed gray line) configurations of a five degrees of freedom planar manipulator surrounded by four obstacles. Link one and four are 1.0 m long. Link two, three, and five are 0.5 m long. All links are 0.01 m wide.

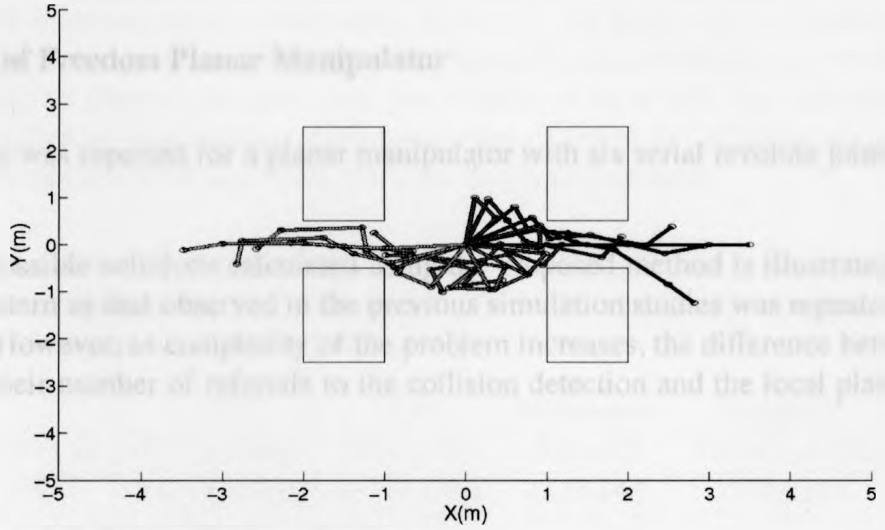


Figure 4.5: Resulting path using the global SVM-based planner.

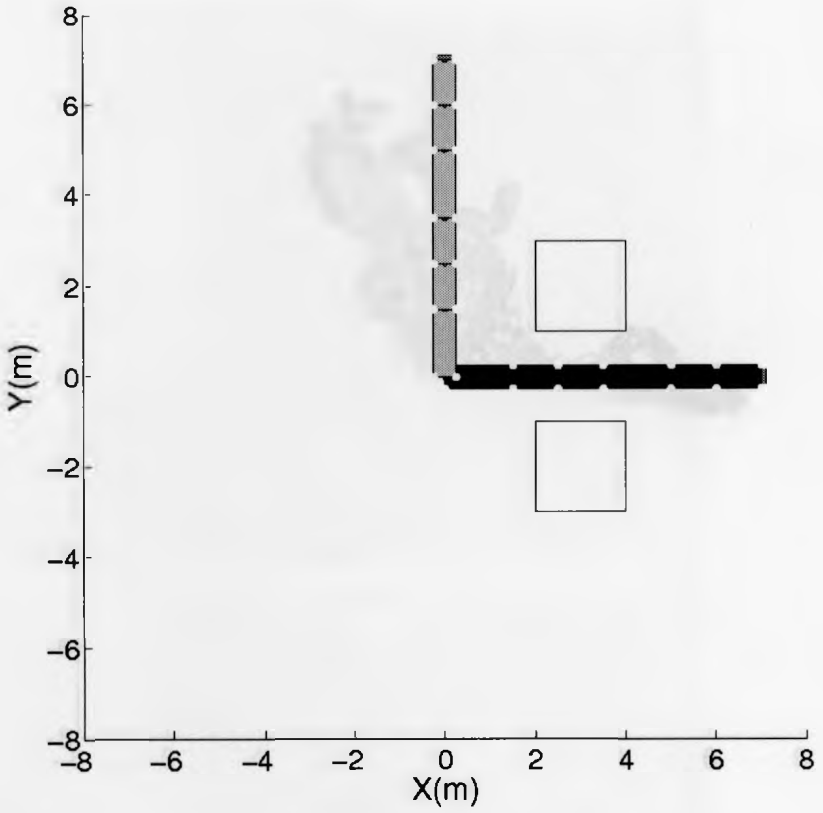


Figure 4.6: Initial (black) and goal (gray) configurations of a six degrees of freedom planar manipulator with two obstacles. Link one and four are 1.5 m long. Link two, three, five, and six are 1.0 m long. All links are 0.5 m wide.

Six Degrees of Freedom Planar Manipulator

A similar test was repeated for a planar manipulator with six serial revolute joints as depicted in Fig. 4.6.

One of the possible solutions calculated using the proposed method is illustrated in Fig. 4.7. The same pattern as that observed in the previous simulation studies was repeated in this case (Table 4.2). However, as complexity of the problem increases, the difference between the two planners in their number of referrals to the collision detection and the local planner modules increases.

4.2.2 CRS-F3 Robotic Manipulator

The *Thermo CRS F3* manipulator is an articulated manipulator with six degrees of freedom (Fig. 4.9(A)). This manipulator is connected to a PC through a C500C controller (Fig. 4.8).

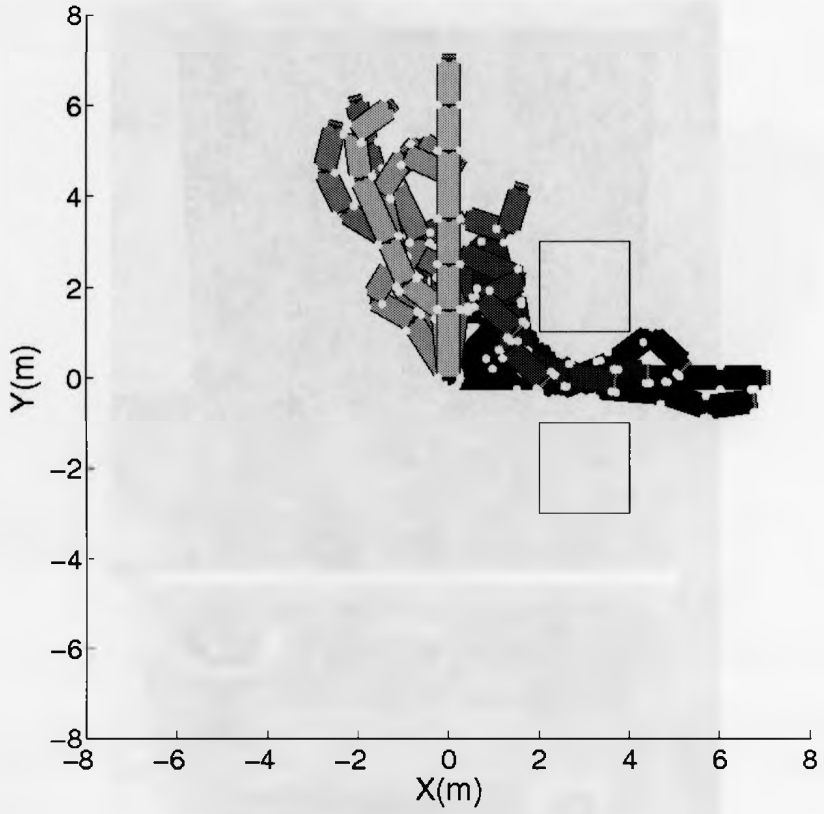


Figure 4.7: Resulting path using the global SVM-based planner.

This controller communicates with a PC via a serial port and transfers the commands given by the user to the manipulator via a fiber optic cable. In this study, the manipulator links were estimated using cubic polygons (Fig. 4.9). This simplification facilitated the use of the RAPID collision detection library. At each step, the location of each link was calculated using the manipulator's *Denavit-Hartenberg (DH)* parameters (Table 4.3). Afterwards, each link was tested against collision with any of the obstacles in the environment.

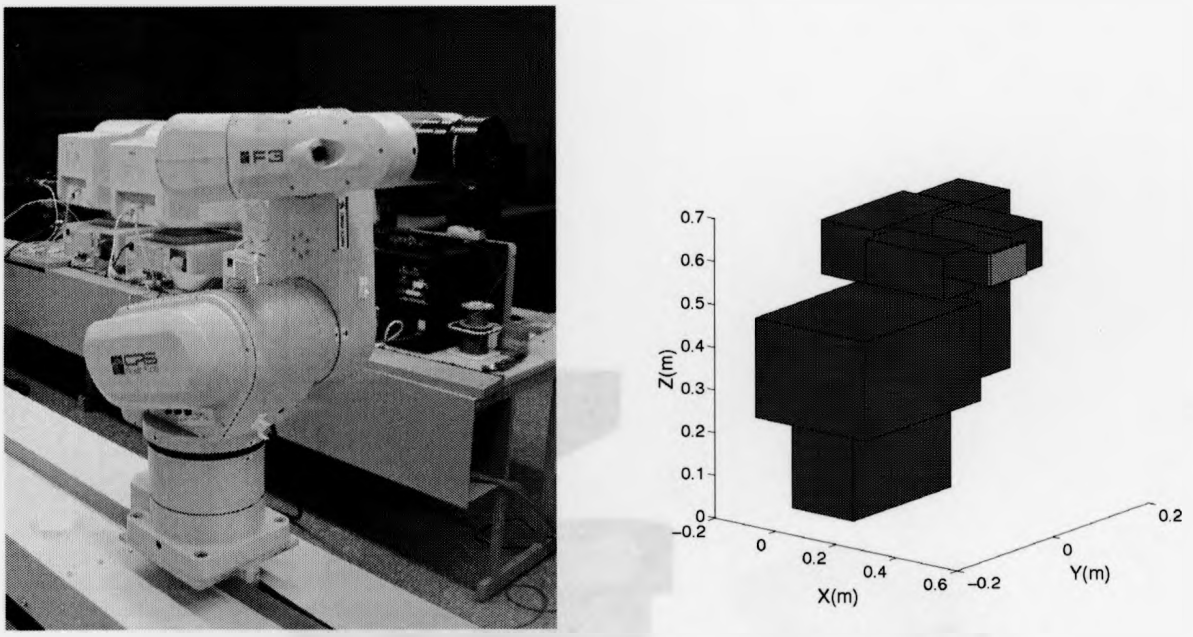
Table 4.2: Comparison of the global SVM-based (SVMP) and the classical PRM planner in the environment shown in Fig. 4.6. CD and LP refer to the number of referrals to the collision detection and local planner modules, respectively.

	CD	LP	Milestones	Time(ms)
SVMP	47722±2144	3754±526	559±32	138743±6182
PRM	53281±5454	5503±1264	1121±44	3278±328

Figure 4.8: A C500C controller which connects the CRS-F3 manipulator to a PC.

Table 4.3: DH parameters of a CRS-F3 articulated manipulator.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0.35	θ_1
2	$\pi/2$	0.1	0	$\theta_2 + \pi/2$
3	0	0.265	0	θ_3
4	$\pi/2$	0	0.27	θ_4
5	$-\pi/2$	0	0	θ_5
6	$\pi/2$	0	0.075	θ_6



(A) (B)

Figure 4.9: (A) Actual and (B) simulated models of a CRS-F3 articulated manipulator.

Table 4.4: Comparison of the global SVM-based (SVMP) and the classical PRM planner in the environment shown in Fig. 4.10. CD and LP refer to the number of referrals to the collision detection and local planner modules, respectively.

	CD	LP	Milestones	Time(ms)
SVMP	50309±2808	1557±246	221±11	137618±6926
PRM	61759±2389	2752±425	360±9	5743±280

Moving between Obstacles

In our first experiment, the CRS-F3 manipulator was moved between two T-shaped obstacles (Fig. 4.10). The manipulator was limited by its joint motions while moving between two obstacles that simulated a narrow passage situation. Fig. 4.11 shows four snapshots of the resulting path. The results of the SVM-based method and the classical PRM planner are compared in Table 4.4.

Similar to the previous case studies, the global SVM-based planner was able to solve the problem with a fewer number of milestones, and fewer referrals to the collision detection and local planner modules. However, the execution time is less in the classical PRM method.

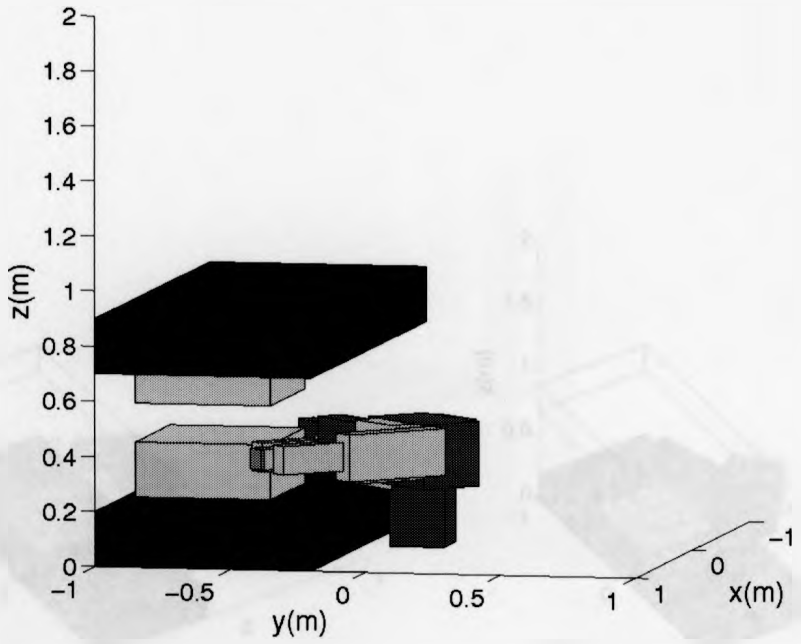


Figure 4.10: Initial (blue) and goal (gray) configurations of a six degrees of freedom CRS-F3 manipulator within two T-shaped obstacles.

Table 4.5: Comparison of the global SVM-based planner (SVMP) and classical PRM in the environment shown in Fig. 4.12. CD and LP refer to the number of referrals to the collision detection and local planner modules, respectively.

	CD	LP	Milestones	Time(ms)
SVMP	502013±37309	76148±8183	3591±458	544392±22219
PRM	1142015±79993	112183±4411	14313±37	234572±5465

Cubic Frame and Bar

In our last experiment, a cubic frame was attached to the end-effector. In this case, the manipulator required to move the frame along a bar without making a collision (Fig. 4.12). Solving this problem was difficult since the manipulator had many restrictions on each joint. Since the SVM classification required two classes of obstacles, another auxiliary bar was arbitrarily placed in the environment. The classical PRM planner frequently failed to solve this problem. However, as Table 4.5 shows the same pattern regarding the number of milestones, the number of referrals to the collision detection and local planner modules, and the execution time were repeated here. Fig. 4.13 illustrates an actual CRS-F3 manipulator while tracking the resulting path of the global SVM-based planner.

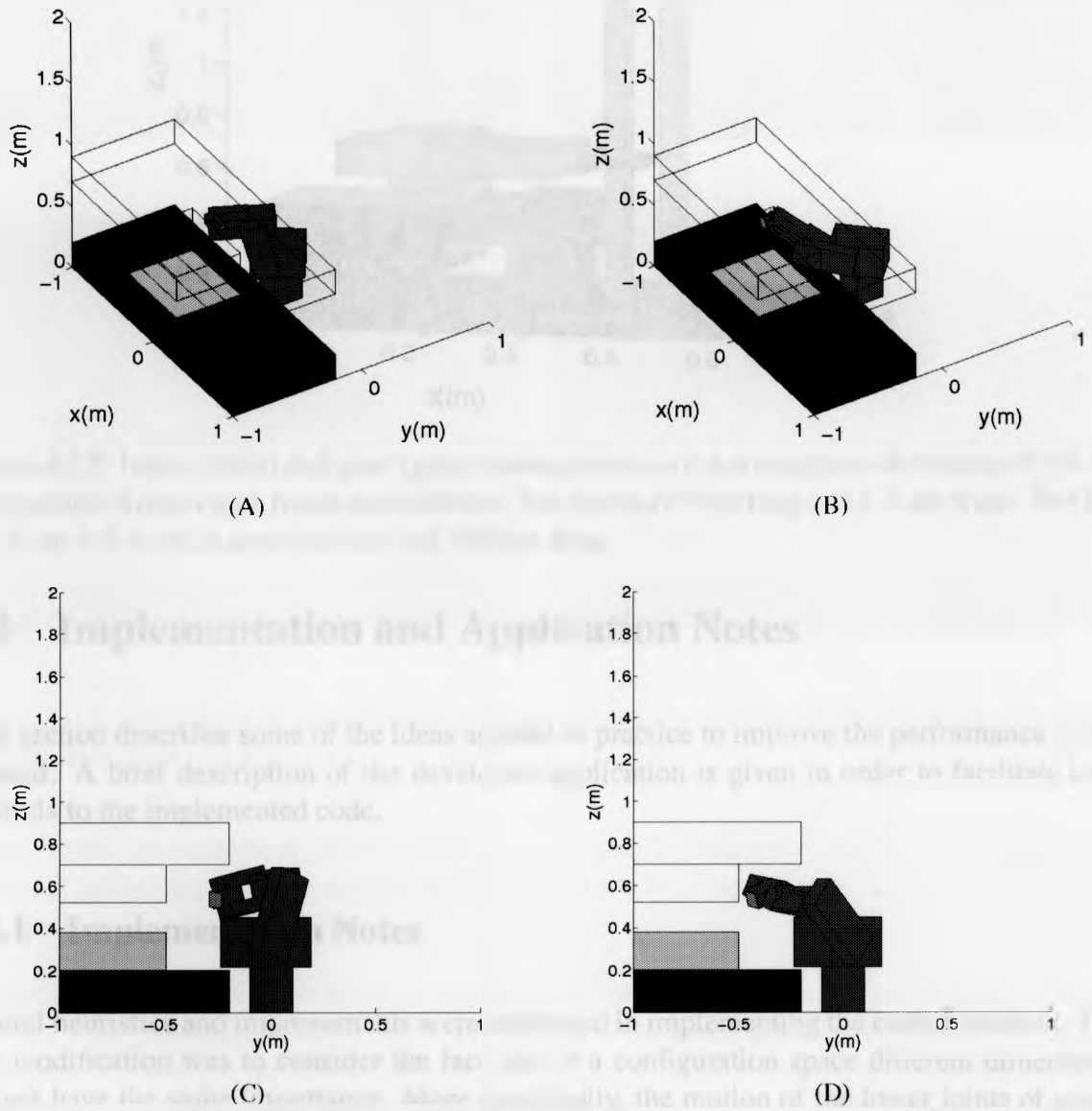


Figure 4.11: Snapshots of the resulting path using the global SVM-based planner.

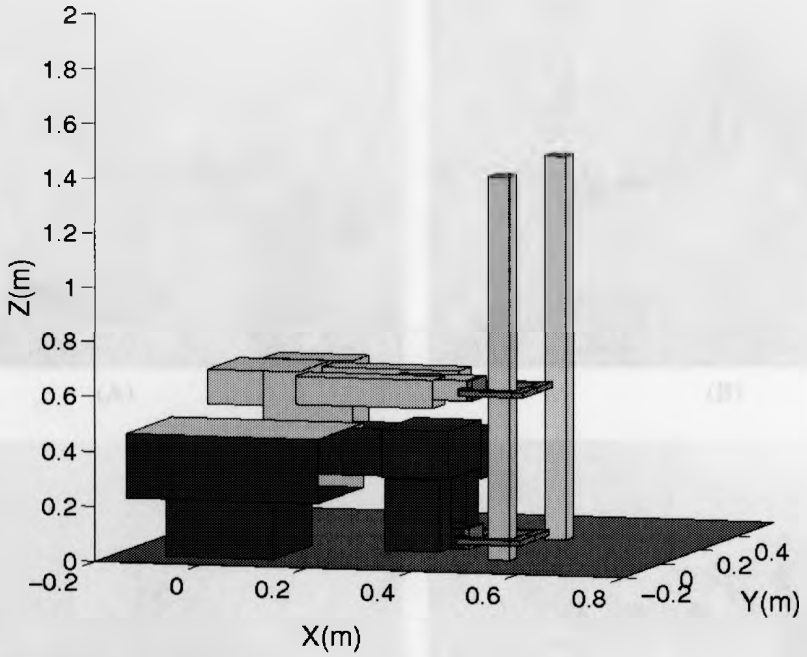


Figure 4.12: Initial (blue) and goal (gray) configurations of a six degrees of freedom CRS-F3 manipulator with a cubic frame end-effector. The frame is 9 cm long and 1.2 cm wide. The bar is 0.4 cm × 0.4 cm in cross section and 104 cm long.

4.3 Implementation and Application Notes

This section describes some of the ideas applied in practice to improve the performance of the method. A brief description of the developed application is given in order to facilitate later referrals to the implemented code.

4.3.1 Implementation Notes

Several heuristics and improvements were employed in implementing the current method. The first modification was to consider the fact that in a configuration space different dimensions did not have the same importance. More specifically, the motion of the lower joints of a manipulator is of higher importance as such motions propagate through the upper joints of the manipulator. As a result, the motion of the lower joints must be sampled with a higher resolution. Due to this reason, the distances in a configuration space were calculated as follows,

$$dist(\mathbf{q}_0, \mathbf{q}_1) = \sqrt{\sum_{d=1}^{n} w_d(\mathbf{q}_{0_d} - \mathbf{q}_{1_d})^2} \tag{4.3}$$

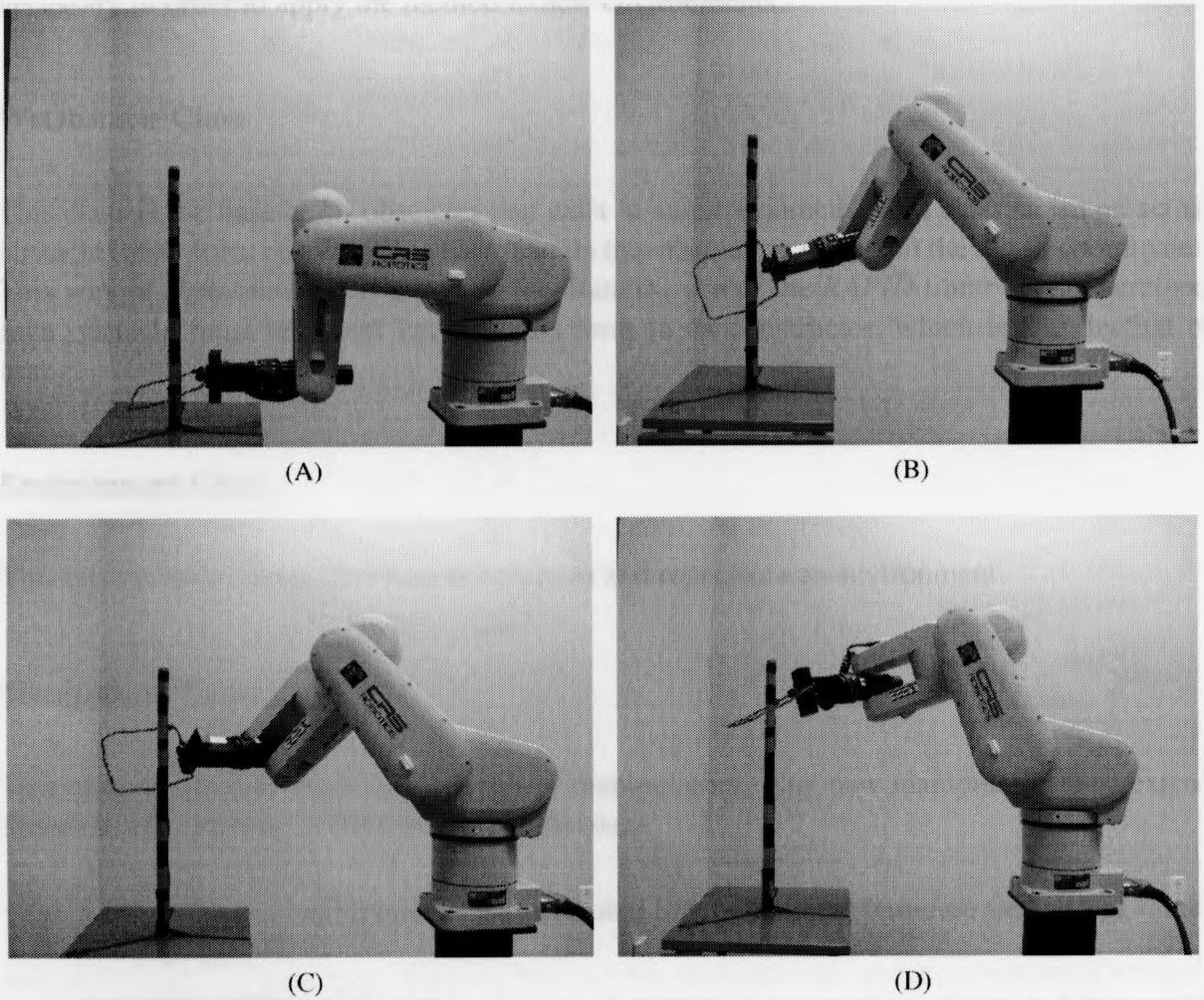


Figure 4.13: Snapshots of the solution generated using the global SVM-based planner in the environment shown in Fig. 4.12.

where n was the number of joints and w_i was the weight assigned to the i^{th} dimension. This approach biases the obstacle sampling procedure to have a higher resolution in the lower dimensions.

The second modification was made in acquiring the value of a risk function where the value of $exp(-5) = e^{-5}$ was considered as zero. The global SVM-based planner spends a great portion of its time in finding milestones. This process is time consuming because of the risk function calculations. This modification could improve the execution time up to 60% in some cases.

4.3.2 Application Notes

The current method was implemented in the C++ language. The implementation consisted of four main classes which are explained in following sections. Understanding these classes are

necessary in order to apply the method to new environments.

WsObstacle Class

This class is the base of all obstacles that exist in an environment. This class stores an actual obstacle in the form of a *RAPID_model* and its transformation related to the world coordinates. This way of representation is aimed to facilitate the use of the *RAPID* library [1]. Therefore, each obstacle must represent itself in this form in its constructor, which is a collection of triangles.

Environment Class

This class contains a set of *Obstacle* instances and represents an environment.

Manipulator Class

This class is the base class for all types of manipulators. Any new manipulator must extend this class and implement the following methods,

- **CalcDhParam** which returns the requested link parameters based on the current values of the joints.
- **IsCollidingWith** which takes an instance of Environment class as input and returns the identification number of the obstacle which is in collision with the manipulator. If the manipulator is not in collision the method must return zero.
- **IsMeetingConstraints** which tests a given configuration against joint limits and self collision. In order to fully capture the configuration space obstacle regions, this method should be called in the *IsCollidingWith* method.

Planner Class

This is the core class of the method in which all different steps are implemented explicitly. This class works on a given instance of *Manipulator* and *Environment* classes. Each step is executed by calling a method and has a state variable that must be set at least once prior to its call. All steps of the planner are implemented in the following methods,

- **SampleObstacle** which takes an initial obstacle sample point as the root of the tree and expands it over the enclosing obstacle region. The corresponding options can be set using the *set_sampling_params* method.

- **TrainMultiSVM** which takes obstacle samples and returns a set of trained SVM models. The corresponding options can be set using the *set_svm_params* method.
- **FindBoundaryMilestones** is responsible for generating milestones based on the previously trained SVM models and obstacle samples. The corresponding options can be set using the *set_find_milestones_param* method.
- **MakeRoadmap** which makes a roadmap by adding possible connections between given milestones using a local planner. The corresponding options can be set using the *set_roadma_params* method.
- **Query** which performs a query on the results of the last step to find a path between given initial and goal configurations. The corresponding options can be set using the *set_query_params* method.

In summary, the method starts by defining a manipulator, obstacles, and an environment. Then each of the above method are called in the given order to find the final solution.

4.4 Conclusion

The proposed global SVM-based method was evaluated in different environments and the results were reported. Considering two simulation environments discussed in Chapter 3, the following pattern was clearly observed in the comparisons. The global SVM-based planner was able to solve the problem with a fewer number of milestones and less number of referrals to the collision detection and local planner modules. However, the classical PRM planner was typically faster. To be compatible with *RAPID* library, all environments were modeled as a collection of triangles. It should be noted that in all cases, the number of triangles used for modeling the entire environment, including the manipulator, never exceeded 50. Therefore, the effect of the smaller number of calls to the collision detection module in the execution time was underestimated. This issue is of paramount importance where the performance of the collision detection module becomes a bottleneck for the performance of the overall method. The performance of a planner becomes highly dependent on the collision detection module in cluttered environments and in situations where an environment is more precisely modeled.

Bibliography

[1] S. Gottschalk, M.C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

Chapter 5

Conclusion and Future Directions

This body of work introduced the notion of a risk function in the robot's configuration space. A risk function was calculated using configuration space obstacle samples. The risk function allowed the determination of safe configurations in a discrete (sampled) configuration space. In a sampled configuration space no obstacle boundary information is provided. The only valid assumption is that the probability of a collision decreases as the distance to the obstacle samples increases. Due to this reason, a safe configuration was defined as a configuration with the maximum distance from the surrounding obstacle samples. The Support Vector Machine method provided a risk function with this property. More precisely, the function had its local minimum on the decision boundaries of a SVM model which were maximally distanced from obstacle samples.

This definition of a risk function was first used as a repulsive potential field in a local SVM-based planner. Taking this approach, a potential field was directly defined over a sampled configuration space. In this method, a manipulator stayed in a safe distance from obstacle samples by following the negated gradient of the calculated risk function. This function was combined with an attractive potential field in order to pull the manipulator towards the goal configuration, simultaneously. The proposed method was evaluated in a simulated environment for two different manipulators with two and three degrees of freedom. The shortcomings of this method which restricted its application in practice was highlighted. The very first issue was related to the embedded obstacle sampling method that heavily relied on a manipulator's inverse kinematics. As a result, the method was limited to a certain class of manipulators with known inverse kinematics. The second issue was related to the execution time of the method that made it not suitable for real-time applications. Local planners are typically expected to work in real-time. Finally, similar to many other local planners, this method was prone to the problem of local minima.

To address the aforementioned shortcomings, another planner was suggested in Chapter 3 that again adopted the notion of the risk function. This method was a global planner and intended to improve the performance of classical PRM planners. A classical PRM planner is not successful in environments containing narrow passages. The proposed method began by sampling

the configuration obstacle regions to make a risk function. In contrast to the previous sampling method proposed for the local SVM-based planner, the sampling method decoupled the sampling process from a manipulator's structural complexities. The suggested sampling method developed a number of trees similar to RRT but over obstacle regions. This set of obstacle samples was used to train multiple SVM models and consequently risk functions. The obtained risk functions were employed to generate a set of milestones with the maximum distance from the surrounding obstacle samples, i.e., the middle of a narrow passage. This method was evaluated and compared to the classical PRM planner in different environments. The simulation results were discussed in Chapters 3 and 4. Two of the case studies evaluated the method on a CRS-F3 manipulator. In these cases, the final solutions were implemented on a actual CRS-F3 manipulator in order to validate the solutions.

The main contribution of this body of work is the introduction of a risk function in a SVM-based planner. This function is calculated using the optimization technique embedded in the SVM method. Since a risk function is able to find safe configurations at maximum distance from surrounding obstacle samples, it can be combined with different methods. Also, employing SVM results in a risk function that is less sensitive to the selected sampling method and resolution. The second contribution of the work is to propose a local method based on a risk function. The risk function that is used as a repulsive potential is created by adding the repulsive force of each obstacle sample. This approach differs from previous works in a sense that each repulsive force is automatically adjusted to keep the manipulator at maximum distance from surrounding obstacle samples. The third contribution of this work is the introduction of a global planner that utilizes a risk function to find a set of well-placed milestones. This method is able to solve a problem with a smaller number of milestones, especially in environments containing narrow passages. Moreover, this method requires fewer referrals to the collision detection and local planner modules in comparison to the classical PRM planner. This feature becomes specially important within cluttered environments.

5.1 Future Directions

To the best of our knowledge, this is the first time that support vector machine algorithms have been used in the context of path planning for higher degrees of freedom robot manipulators. There are several remaining issues that require further investigation,

- Evaluating the method in scenarios where the performance of the method heavily depends on the performance of the collision detection module. A cluttered environment and an environment modeled in details are two possible scenarios.
- The study of alternative obstacle sampling methods that focus on boundary samples.
- Due to the fact that a SVM model is trained over the entire configuration space, the act of acquiring a risk function value becomes time consuming. However, based on the selection of the SVM kernel parameters, the influence of the sample points far from a

given configuration is zero at that configuration. Hence, some heuristics can be used to obtain the result of a risk function faster.

- Even though the proposed planner tries to place milestones in safe positions, the resulting roadmap may be composed of different connected components. Therefore, improving roadmap connectivity must be the next step. This is achievable by generating random points near a region of disconnectivity and guiding them towards the decision boundaries. This approach increases the number of milestones and consequently improves the roadmap connectivity.

- In situations for which obstacle regions are dominant, more samples are necessary in order to have a good estimation of the obstacle regions. In these situations the classical PRM may perform better. Thus, the idea of combining these methods should be investigated.

- The effect of each method's parameters was discussed individually. Further studies need to be conducted to observe the effects of the parameters all together.

- Since the method is less sensitive to the sampling resolution, utilizing the idea of incremental sampling can potentially improve the performance.

- Since in dynamic environments the change between two subsequent time steps is not substantial, the result of the SVM in one step can be used as the initial point for the next step optimization. Using this approach, it is conjectured that the global SVM-based planner can be extended to real-time environments.