

ADAPTABLE SAFE PATH PLANNING UNDER UNCERTAINTY
CONSTRAINTS

by

Shubham Jain

Copyright © Shubham Jain 2019

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2019

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Master's Committee, we certify that we have read the thesis prepared by **Shubham Jain**, titled **Adaptable Safe Path Planning under Uncertainty Constraints** and recommend that it be accepted as fulfilling the dissertation requirement for the Master's Degree.




Jerzy W. Rozenblit

Date: April 22, 2019



Gregory Ditzler

Date: April 22, 2019

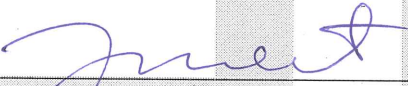


Salim Hariri

Date: April 22, 2019

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to the Graduate College.

I hereby certify that I have read this thesis prepared under my direction and recommend that it be accepted as fulfilling the Master's requirement.



Jerzy W. Rozenblit, Master's Thesis Committee Chair
Electrical and Computer Engineering

Date: April 22, 2019



ARIZONA

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jerzy Rozenblit, for taking me on as his student and for his guidance and support throughout the course of my thesis work, as well as his understanding when it came to having a demanding job in industry while pursuing my graduate studies at the same time. I am grateful to have been brought in and made a member of the Model Based Design Lab. I would also like to extend my thanks to the other members of the MDBL for their camaraderie and help in carrying out my project. In particular, I thank Minsik Hong for his willingness and effort to help and provide invaluable suggestions, as well as Aakarsh Rao, Kuo Peng and Naseer Albalawi for their constant motivation. I thank Dr. Salim Hariri and Dr. Gregory Ditzler for serving on my committee. Finally I would like to thank my mom, my dad, my two brothers, my extended family, and all of my friends who have supported me and spurred me on in my graduate studies. Their love and encouragement has made my success possible and I consider myself very blessed to have them all.

TABLE OF CONTENTS

LIST OF FIGURES	5
LIST OF ALGORITHMS	6
ABSTRACT	7
CHAPTER 1 INTRODUCTION	8
1.1 Path Planning	8
1.1.1 Robotic Mapping Techniques	12
1.1.2 Shortest Path Algorithms	15
1.1.3 Safe Path Planning	18
1.2 Motivation and Objective	20
1.3 Contribution of the Thesis	21
1.4 Related Publication	22
1.5 Thesis Structure	22
CHAPTER 2 BACKGROUND	23
2.1 Available Planners	23
2.1.1 Probabilistic Roadmap Planners	24
2.1.2 Rapidly-exploring Random Trees	28
2.1.3 Safe Path Planners	30
2.2 Case Study - Computer Assisted Surgical Trainer	32
2.3 Problem Definition	35
CHAPTER 3 ADAPTABLE PRM	37
3.1 Problem Formulation	37
3.2 Modified Algorithm	43
CHAPTER 4 SIMULATION RESULTS	48
4.1 Experimental setup	48
4.2 Selecting k and s	50
4.3 Results	51
CHAPTER 5 CONCLUSION AND DISCUSSIONS	56
REFERENCES	62

LIST OF FIGURES

1.1	Environment model	10
1.2	Configurations spaces	11
1.3	Random Sampling	14
1.4	Grid Based Sampling	15
1.5	Artificial Potential field	16
1.6	Modified Configuration Space	19
2.1	Connected Components	24
2.2	Connection Strategies	27
2.3	RRT* Rewiring Procedure	30
2.4	MIS Operating Room	33
2.5	Computer Assisted Surgical Trainer	34
3.1	Path planning formulation	39
3.2	Algorithm Steps Illustration	46
4.1	Setup for simulation	49
4.2	Edge Passing through Virtual Boundary	50
4.3	Simulation results	52
4.4	Path Length Analysis	54
4.5	Multiple-paths Comparison	54
4.6	Graph Construction Overhead Time	55

List of Algorithms

1	Roadmap	41
2	Addmilestone	42
3	Query	42
4	AdaptivePRM	43
5	PathRepair	45

ABSTRACT

Path Planning in robotics is an open fundamental problem in fully autonomous or manipulator systems. Many standardized algorithms have been proposed taking a different aspect of the problem. Following a path generated by a path planner is not accurate because of sensor noise or malfunctioning, inaccuracy of control actions and commands and so on. This causes inaccuracy in the localization of the robot in the environment, which may lead to a collision if path planning is done assuming accurate motion. This uncertainty constraint induced during the mid-flight of the robot should be added with other design constraints so a collision-free path can be generated. As other constraints, this uncertainty constraint modifies the configuration space during mid-flight and since localization error can lead a configuration space to "bloat", this might lead to isolating the initial and goal configurations separated by critical regions. Fortunately, these constraints are not inherited by design and can be made flexible for planning a path by avoiding constraints wherever necessary. Therefore, this work focuses on maintaining two configuration spaces and provides an intelligent method to make the uncertainty constraints flexible by toggling between two spaces wherever necessary. The simulations considering design constraints of Computer Assisted Surgical Trainer (CAST) are presented as proof-of-concept which provides support to the objective of the thesis and demonstrates that the algorithm extends the functionality of a common path planning algorithm called Probabilistic Roadmap Planners, where by algorithm design it is proved that algorithm is adaptable to PRM in the worst case.

CHAPTER 1

INTRODUCTION

With the advancement in technology, it is common to see robotic systems deployed for multiple purposes around humans. They started out being controlled by humans, but over time they are becoming completely independent. Most of these autonomous systems with moving parts, either mobile robots or manipulators, essentially needs a mechanism to generate a path (for navigation) to perform an appropriate task that is to be achieved by the movement. The set of techniques of autonomously generating a path is addressed by one of the most researched areas in the field of robotics intelligence called *Path Planning*, which is the topic of this thesis.

This chapter describes the general path planning problem with important terminology and definitions used in the field. Additionally, it provides basic approaches to robotic mapping and graph search method for finding the shortest path. Then Safe Path planning is described with the motivation behind the thesis and contribution to the path planning community.

1.1 Path Planning

Initially introduced as a technique to find a *collision-free* path between two positions in a static environment, path planning has taken diverse forms and attracted academicians and researchers from diverse fields. Applications of path planning ranges from robotics and communications to drug design. For example, Traveling Salesman

Problem which provides a solution to task scheduling in processors, assembly line optimization, packaging optimization, packet routing and many more, is efficiently solved by path planning algorithms.

In general robotics, a typical path planning problem is described as follows, "*Given an Environment containing Obstacles, find a collision-free path between two positions in the free-space*", where an entity, called *Robot*, is expected to navigate the generated path. This problem of finding a collision-free path has been seen as one of the fundamental problems in robotics and there are many established techniques that address various aspects of the problem. Often, the terms *Path Planning Algorithms* and *Path Planners* are used interchangeably. Segmenting the definition above introduces three important terms that are commonly spread across the discussion in the thesis, each presented pictorially in Figure 1.1. These are:

- **Environment** - This is the representation of the space where the robot is present and Path Planning is done. An environment can either be static, meaning the information about the environment remains constant, or dynamic, meaning the information about the environment is a function in time. Moreover, the environment, or often called a workspace, consists of non-overlapping free-space and the obstacle-space.
- **Obstacles** - These are those regions of the environment which cannot be accessed by the robot, and passing through these regions is considered as a collision. Rest of the environment except the obstacle-space is the free-space and can be navigated by the robot.
- **Initial and Final positions** - These define two locations of the robot in

free-space between which a path should be found. The robot is expected to navigate from initial position to the final position following the path generated by the path planner.

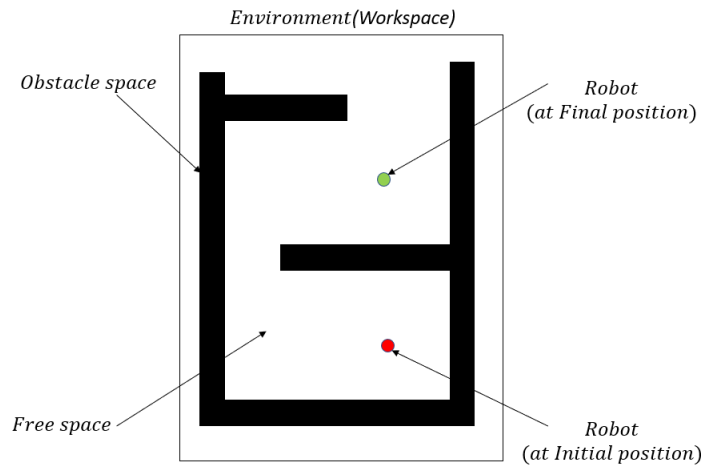


Figure 1.1: Environment model

Moreover, a path planning problem in its most general form is a geometry problem, therefore a geometric description of the robot and environment is essential. Additionally, a description of the *Degrees of Freedom*, design constraints, like non-holonomicity, constraints on the maximum/minimum angle of rotation, etc. and initial and final *configuration* (geometric location and orientation) of the robot are also required. Using these descriptions and constraints, a *Configuration Space* is constructed for the robot. Figure 1.2 shows configuration spaces for various robot types. As depicted, for a circular free-flowing robot, the configuration space is the same as a workspace but the obstacles are "bloated" by the footprint of the robot,

however, it entirely depends on the geometry and constraints of the robot. Moreover, configuration space maps the robot with its constraints to a free-flowing point robot and hence methods of path planning of a free-flowing point robot works perfectly in the configuration space. Therefore, for a point object, configuration space and the workspace are the same.

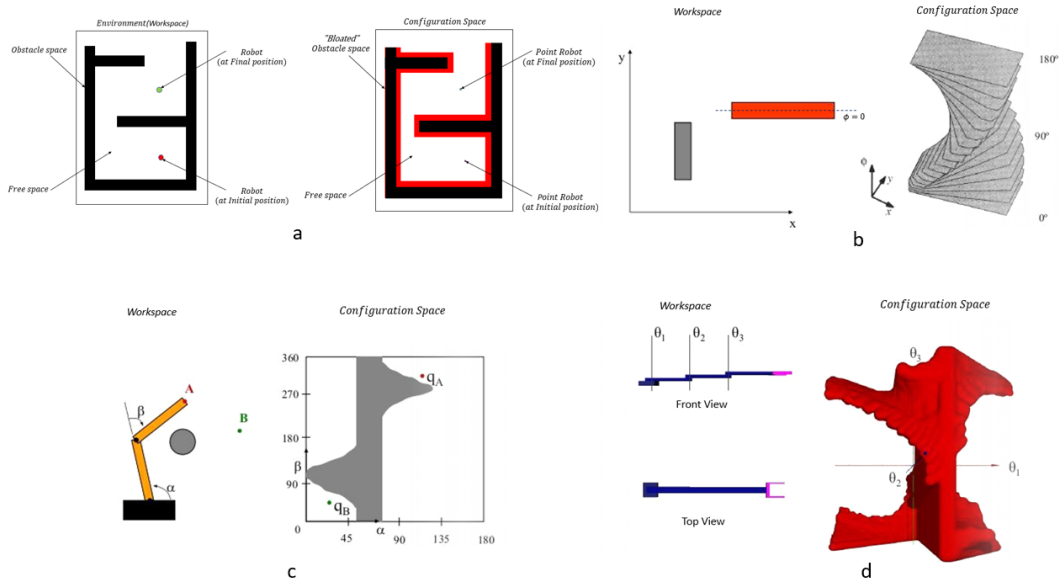


Figure 1.2: Configurations spaces [4] for (a) Free Flowing circular robot, (b) Free flowing rectangular robot, (c) Fixed Robot with two joints and (d) Fixed Robot with three joints

For discrete representation of the real environment, *Topological* framework is commonly used. This framework considers robot's configuration at each geometric location and stores the relevant information, like distance from the nearest obstacle, distance from the initial and/or goal configurations, relation with geometric center, etc. Independent configurations are represented by *Nodes* and the connections between them by the *Edges* in a *Graph*. A typical graph is represented as a set of

nodes and edges, $G = (N, E)$, where, in path planning scenarios, nodes represent the spacial location in the configuration space with other information as described and edges contain the information regarding the distance between the nodes. The path, in this case, is the series of connection of nodes with edges that connect initial and final configuration. Modern planning algorithms commonly use this framework, while method of graph construction differs according to application.

Although a real environment can be static or dynamic, Computer Science and Artificial Intelligence approach ([21]) considers the environment to be static for path planning. This is because the method of finding desired path using Artificial Intelligence using graphs follows a general iterative approach of graph search that evaluates and compare information at every node against a given optimization criteria, which is generally path length in path planning problems, and expand the best possible node using *Shortest Path Algorithms* (described in sub-section 1.1.2). Hence, the information at the expanded node should remain unchanged. However, with the changing environment nodes might modify information, hence modifying the Graph and invalidating search results. Therefore, graph search problems, a graph should remain unmodified at all times during a search. The next sub-section describes the mapping techniques which construct these graphs using the raw environment information using a sensor or some other representation.

1.1.1 Robotic Mapping Techniques

Robotics Mapping is the discipline where a robot tries to locate itself using on-board sensors on a map which is pre-processed or constructed during the exploration of the robot in the environment. Path planning is performed by searching a graph as

described previously, that is using the information from the map, but the methods of graph construction from the map and its manipulation depends on path planner. There are many sophisticated approaches as described by [37], which are fundamentally motivated by one of the following three main types:

- **Random Sampling** - As the name suggests, the method creates a graph by simply sampling the space in random fashion, check for the collision with the obstacle while satisfying the robot constraints and store the appropriate information as a node in the graph, if successful. The node then tries to make the connection with other node/nodes by checking the motion between them, the connection is unsuccessful if the motion leads to the collision with the obstacles. Sampling is essentially an iterative process which is suitable for low as well as high dimensional configuration spaces. Algorithms based on this method are probabilistic complete and asymptotically optimal, that is they provide an optimal solution to a path planning problem, if one exists, in infinite time. This method is widely used in most popular path planners like Probabilistic Roadmap Planners as described in [16] and Random Exploring Random Trees as described in [20]. These are discussed in details in Chapter 2.
- **Grid-based Method** - In this method, complete environment is divided into small regular grid cells which, in case of two-dimensional spaces are either 4 connected or 8 connected to the neighboring grid cells. Similar to the Random sampling approach, a graph is constructed, but in a regular fashion. The granularity of the grid decides the accuracy of the environment represen-

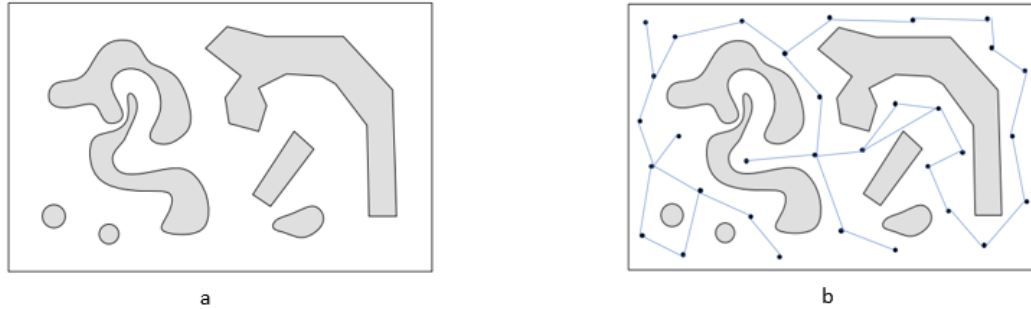


Figure 1.3: Random Sampling: (a) Environment [42] and (b) Random Sampling for point robot

tation, higher granularity means better information about the environment. Typically, the cells represent either the obstacle or the free space, but there are certain modifications for better performance, like the Quadtree method which divided the environment in regular grid first, finding a path in sparsely represented grid and increasing granularity of the grid cells which has both obstacles and free space which are part of the path. Although path planners based on this method are complete, computational complexity increases with increasing dimensions. However, grid-based planners are popular in planning maneuvers for low dimensional planning problems like in Autonomous vehicles. Moreover, the hybrid approaches which combines cell decomposition method with probabilistic sampling ([23]) and the other which combines artificial potential field and probabilistic cell decomposition ([31]) are proposed for high dimensional problems.

- **Artificial Potential Field Method** - Artificial Potential Field method described in [41] considers the robot to be moving in an environment with po-

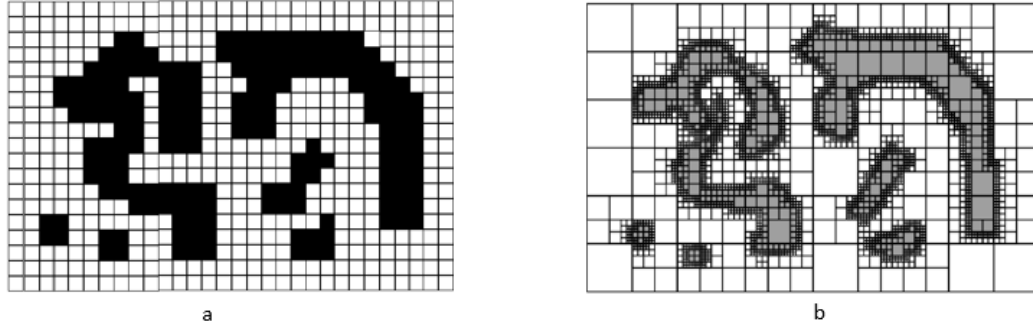


Figure 1.4: Grid Based Sampling (a) Regular Grid and (b) Quadtree method [42]

tential field. The environment consists of the attractive force from the goal and the repulsive forces from the obstacles. The potential field, in this case, is a vector field with the magnitude inversely proportional to the distance from the obstacles and directed away from it (Figure 1.5). Alternatively, the potential field can be presented as the scalar function over the free space and robot moves in the direction that negates the potential gradient at any point. The approach is simple but the robot can get stuck in the local minima of the potential fields. Though [39] proposed a method to resolve the problem, this method has almost become obsolete in the path planning domain.

1.1.2 Shortest Path Algorithms

The shortest path algorithms are used to find optimal solution according to the optimization criteria, which as already suggested is often path length in path planning problems. These algorithms are used for decision making, in case of graphs they decide which node should be expanded among multiple candidates. These algorithms compares the value of nodes against an optimization criteria and decide the order

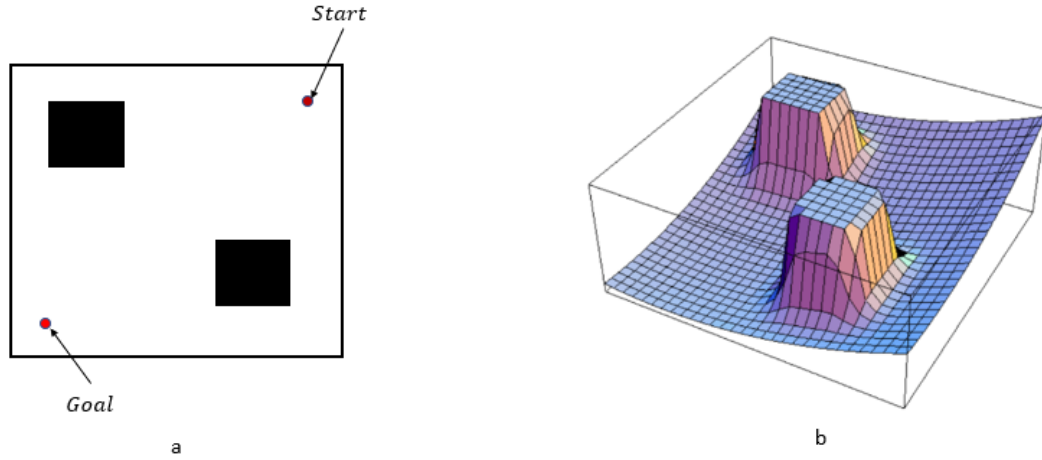


Figure 1.5: Artificial Potential field: (a) Environment model and (b) Potential fields [4] for environment in (a)

in which they are expanded. The most commonly used shortest path algorithms are as follows:

Dijkstra's Algorithm - Dijkstra's algorithm ([5]) is a shortest path algorithm that is used to find the optimal path from one node to any other node in a graph. This works by maintaining for each node, n , the shortest possible distance from the start node to the present node, which is represented by $g(n)$. The distance is a non-negative cost of moving from a node n_1 to node n_2 and is associated with the edge (n_1, n_2) connecting both these nodes. This distance between two nodes is computed by a cost function $c(n_1, n_2)$. Moreover, a back-pointer at every node is maintained, that direct towards the node closest to it and help in tracing the path from the current node to the start node for the shortest path. The procedure is simple and works as follows. Initially, all the nodes are initiated by infinite shortest path length, that is $g(n) \rightarrow \infty$, except the start node where $g(n) = 0$. Then

starting from the start node, n_s , each node n connected to it is updated, that is $g(n) = \min(g(n), g(n_s) + c(n_s, n))$ and arranged in a priority queue for expansion. After expanding a node, it is marked as visited and the next node in the queue is expanded. This continues until the goal node is expanded or complete graph is expanded, that is, queue is empty. This approach is successful in finding the shortest path to a goal from different nodes in a graph. Moreover, [7] demonstrated the use of Fibonacci heap for optimal running time, instead of using a priority queue.

A* Algorithm - A* algorithm ([11]) is another popular shortest path algorithm, which works exactly like Dijkstra's algorithm, but with one major difference. Instead of just maintaining the information about the shortest path at every node, A* maintains a heuristic value at every node, which is computed by the heuristic function $h(n)$, that is the best possible guess of the distance to reach the goal. A* guarantees optimality of solution if the heuristic function is a lower bound estimate of the shortest path from the present node to the goal node, that is, the value at every node at any point of algorithm execution should be less than or equal to the actual overall shortest path. The node value is thus represented by the sum of two values $g(n)$ and $h(n)$. Moreover, when planning problems are time-bound, Anytime Planning A* ([22]) which uses the concept of Weighted A* can be helpful. The solution, in this case, is sub-optimal with a lower bound on optimality and this bound is tightened if planning time allows improvement of solution. A* performs faster than the Dijkstra's algorithm in most cases and is useful when path is to be computed between specified initial and goal. This thesis uses A* approach for its speed and already defined initial and goal configurations.

1.1.3 Safe Path Planning

Safety in path planning is an important research direction. As robots are getting in direct contact with the humans, it is important that the collision never happens. Although a *Safe Path* is a subjective term, the problem defined at the beginning of the section defines the baseline for safety, that is a "collision-free" path. There are established techniques in artificial intelligence for path planning, but what is generally overlooked is the assumption that robot can actually locate itself with the same accuracy at all time and can follow the path accurately. This leads to change in configuration space.

Consider an *Unmanned Aerial Vehicle* (UAV) navigating an environment using a pre-computed map, and localize and navigate using the data from its onboard sensors. Depending on how reliable is the data provided by the sensor, each sensor has a *Confidence Value* in the information extracted by its data, higher confidence value indicates more accuracy in the information. However, due to increased noise for loose connections, sensor malfunctioning, or inaccurate GPS location, or other external conditions, confidence value might change in the mid-flight. This may lead to inaccuracy in localization on the map provided to the vehicle and the best path cannot be followed due to deviation from the path caused by uncertainty in the location. This leads to inappropriateness in the map information which was fed to the vehicle, leading to increased possibility of collision. Moreover, it is logical to assume that due to localization issue, the point robot can be "bloated" during run-time.

As a safety measure in such situations, the UAV should maintain at least a spe-

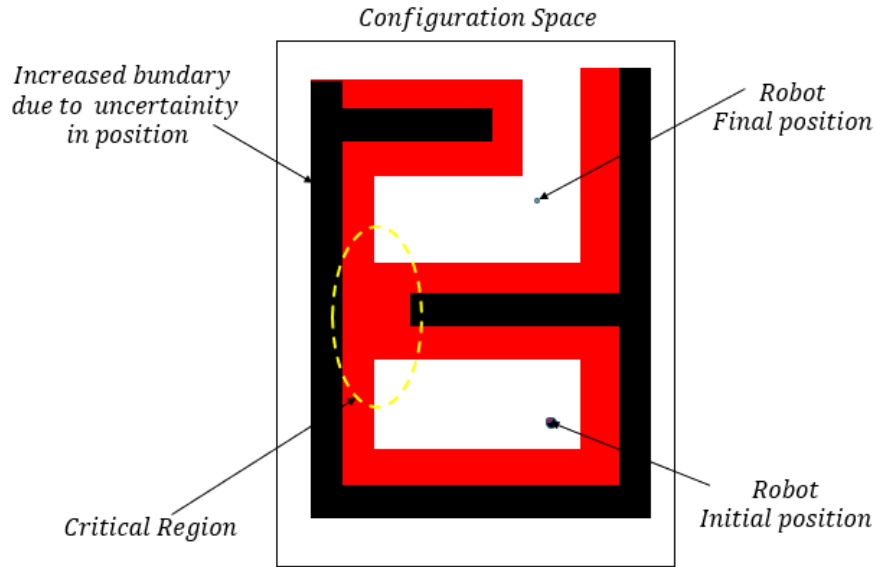


Figure 1.6: Modified Configuration Space due to induced uncertainty in localization

cific distance or *Clearance* from the obstacles in the environment where it navigates. This least clearance can be given as a constraint. However, increase in localization uncertainty may lead to transformation of the configuration space, which may block the robot in a closed space with no path to the goal as shown in Figure 1.6. In other words, the only way to find a path to goal was blocked due to inaccuracy in localization. These regions which are blocked due to uncertainty and are important for the robot to find a path are referred to as *Critical Regions* in this thesis, as shown in Figure 1.6 with dashed oval region. This thesis addresses the problem by making the localization constraint/ uncertainty constraint flexible in critical regions, so a path can be found. It should be noticed that this constraint can be made flexible as it is induced by the robot's problem in localization in mid flight and is not the inherited constraint by design. Hence, this thesis proposes an algorithm that tries following

the uncertainty constraint to reduce the length of path covered in the region of map that violates this constraint.

1.2 Motivation and Objective

Safe path planning in the presence of critical regions motivates this thesis. The uncertainty in the localization of the UAV due to degradation in sensor's confidence value is introduced as an example, but the concept can be generalized to multiple applications. Since this is a path planning problem, it is important to model uncertainty to distance and hence, any problem where distance can be evaluated from uncertainty, this approach can be applied. For instance, in Minimal Invasive Surgery (described in Chapter 2) the surgeon's or trainee's expertise in Navigation task can be modeled as the deviation from the recommended path while navigating the instrument tip, or in case of wheeled robots, error in control signals can lead to deviation from the navigation path, which is considered as an uncertainty.

To provide a path in scenarios where critical regions block the path generation, this thesis proposes a path planner that tries to follow the uncertainty constraint and avoid the constraints while when passing through critical regions, as following the constraints can lead to failure for a path planner to compute a path. Moreover, it would be established further in the thesis that identification of the critical regions is not important if planner can make the constraints flexible in those regions.

Therefore, the objectives of this thesis are:

- To reduce the path length a robot would navigate in the regions which violate the uncertainty constraints.

- Automatically making the uncertainty constraint flexible in the critical regions without explicit identification of critical regions.
- To develop an adaptive iterative process to compute a path that satisfies the above objectives or in case of non-existent of such a path (worst-case scenario), provides the shortest path eliminating uncertainty constraints.
- Provide the proof of concept by simulating Computer Assisted Surgical Trainer with its design constraints as a case study.

1.3 Contribution of the Thesis

This thesis extends the functionality of *Probabilistic Roadmap Planners* (PRM) by repairing the shortest path to find an alternative path that satisfies uncertainty constraint in the free space except for the critical regions by maintaining strategically constructed map.

Moreover, a novel method to repair a base path/shortest path that satisfies the objectives described above is presented. The approach differs from the traditional path planners by incorporating flexibility in constraint applicability and hence, no comparison can be made with the traditional planners which either return no path or return path with altered uncertainty constraint. Moreover, the proposed planner always returns a solution, if one exists. In worst case, the planner works similar to traditional PRM.

1.4 Related Publication

To support the work conference paper titled "Proficiency based Planner for Safe Path Planning and applications in Surgical Training" have already been submitted and accepted in *Proceedings of the Symposium on Modeling and Simulation in Medicine. Society for Computer Simulation International, 2019.*

1.5 Thesis Structure

Chapter 2 provides a background on the popular sampling based-path planners and some safe path planning approaches defined by various researchers. The chapter defines the best choice for the safe path planning considered in the thesis and defines the case study used in the thesis. Then problem definition is provided in the context of case study.

Chapter 3 presents the formal definition of the problem, describing it mathematically. This chapter provides algorithm behind the base planner and continues to describe the modified version of the algorithm in details, which help satisfy all the objectives.

Chapter 4 and 5 present the results performed on a simulated environment, and conclusion and discussion, respectively. These also highlight a few limitations of the algorithm and the future direction on the research.

CHAPTER 2

BACKGROUND

This chapter first explains in details, the two popular sampling-based approaches used often for path planning, traditional Probabilistic Roadmap Planners and Rapidly Exploring Random trees. Then, the attempts to safe path planning in multiple contexts are explained which certainly differs based on the definition of the problem by the researchers. Then, minimally invasive surgery and CAST are introduced as case-study for the thesis. This continues to define the problem definition in the context and definition of safety for the purpose of the thesis, which takes a formal mathematical form in Chapter 3.

2.1 Available Planners

As described in previous chapter the issue with most path planning problems is to deal with how to create a graph out of free configuration space to represent the connectivity in various regions. This section describes the two most popular sampling-based path planning algorithms which use different graph structures, these are *Probabilistic Roapmap Planners* and *Rapidly-exploring Random Trees*

2.1.1 Probabilistic Roadmap Planners

The basic PRM

Probabilistic Roadmap Planner, abbreviated as PRM, uses the graphs, or *Roadmaps* for the representation of the connected-ness of the environment in which planning is to be performed. The roadmaps are based on *Graph Theory*, where a graph is represented as a set of nodes, or *Milestones* and edges, $G = (N, E)$. There can be one or more sets of nodes in a graph represented by separate sub-graphs, or *Connected Components*. Figure 2.1 shows a graph with 4 connected components, however, it is important to note that a graph with all the nodes connected into one is itself a connected component and even an independent node with no edges is also a connected component in itself. Moreover, for two nodes in a connected component, it is guaranteed that there exists a path between them with at least one edge connection.

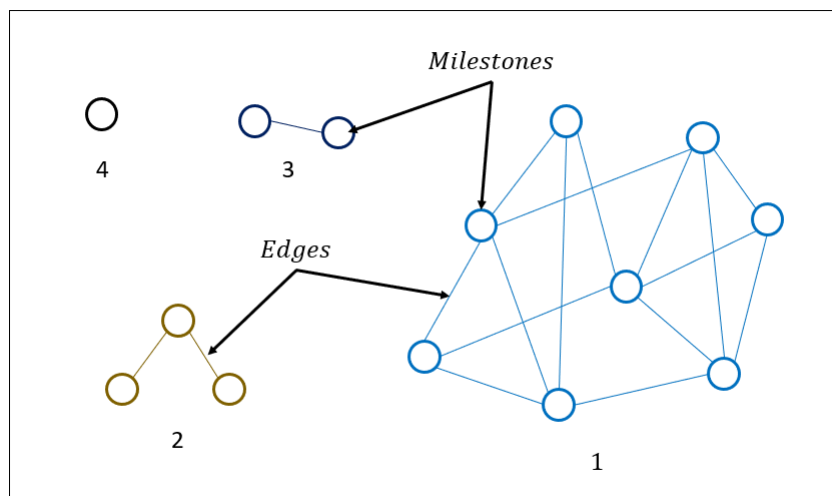


Figure 2.1: One graph with 4 Connected Components

The basic PRM has is the simplified version of the planner described in [16]. They described path planning in two phases, *Learning Phase* and the *Query Phase*. The learning phase is the process of roadmap construction, first, the environment is sampled using a *Sampling Scheme* and check if it lies in the free configuration space using a collision checker. If yes, then this new sample, is added to the roadmap of milestones as a new connected component and tries to connect to the nearest milestone(s), from other connected components, if one exist, using a *Local Planner*. Since the graph can become large and it may take a long time to connect to all possible milestones, *Connection Strategy* selects a set of milestones into *Nearest Neighbor Set*. If the connection is successful, an edge connects new and previous milestone, hence combining the two connected components into one. This process repeats until the learning phase is terminated by a termination condition, usually a constraint on planning time. The second phase, which is the query phase, is just a shortest path search on the roadmap provided by the learning phase using A*. The idea of PRM is to cache the roadmap for either single or multi-query path planning problems. For the three components of the basic PRM described: Local Planner, Nearest Neighbor Set, and Sampling Scheme, important choices are to be made. These are discussed independently next.

Local Planner - A local planner is used to find direct and the quickest path between two milestones so these can be added to the graph with an edge. This planner performs minimal computation to check the collision of intermediate states with the obstacle using the collision checker and if successful make a connection between them. This planner encodes the kinetic constraints of the robot to check the motion from one configuration to the next. Moreover, the milestones to which a local

planner connects should be in the *visibility region* of the local planner ([10]). Larger this visibility region better is the possibility of connection and better connectivity. For planning problems of holonomic robots, usually the local planner tries to connect to the node with a straight line. For robots with non-holonomicity, straight line connections with a direction other than the heading direction is not possible, so the local planner, takes the motion while making a successful connection.

Nearest Neighbor Set - For an ideally connected roadmap, every new milestone in the configuration space should try to connect to every other existing milestone. However, since this process of using local planner to connect to every node is time consuming when the size of the roadmap is large. Connection strategy selects a set of neighbors which should be attempted by the local planner for the connection. Commonly, a set containing maximum k neighbors is selected by the strategy, but depending on the configuration space other metrics are also helpful. Analysis done by [15] and [43], divides PRMs connection strategy into three major forms, namely k , s , and $k-s$. Figure 2.2 shows the nearest neighbor set in a part of a graph when a new milestone is sampled in the space. with three strategies and. It should be noted that every strategy provides a set of milestones sorted based on increasing *Euclidean Distance* from the newly added milestone. k -strategy selects the maximum nearest k number of milestones for this set. Although this strategy has an advantage of adopting enough milestones to ensure smoothness by adjusting the parameter k , it has a bias towards the denser region of sampling space. Moreover, since there is no bound on the length of the edge, distant connections are possible, which provides limited information about the intermediate configurations of the path. The s -strategy uses a parameter s , which defines a distance bound inside which milestones are selected

for Neighbor Set. This overcomes some shortcomings of K -strategy, by choosing all milestones within a specified radius and limiting the maximum distance between two different milestones which ensures more information about the path, but this comes with reduced smoothness and increased computation time if the sampled space is dense (since there can be many milestones when space becomes dense). Finally, $k - s$ strategy is an adaptable s strategy, which has the flexibility of selecting k and s as described above, helping to tune smoothness, guaranteeing all direction connections, reducing the computational burden and providing more information about the path. Selections of the value of parameters k and s are application dependent. As it would be explained later, more information about the intermediate points would be needed to accomplish the objectives of this thesis, therefore $k - s$ strategy would be used to select the Nearest Neighbor set.

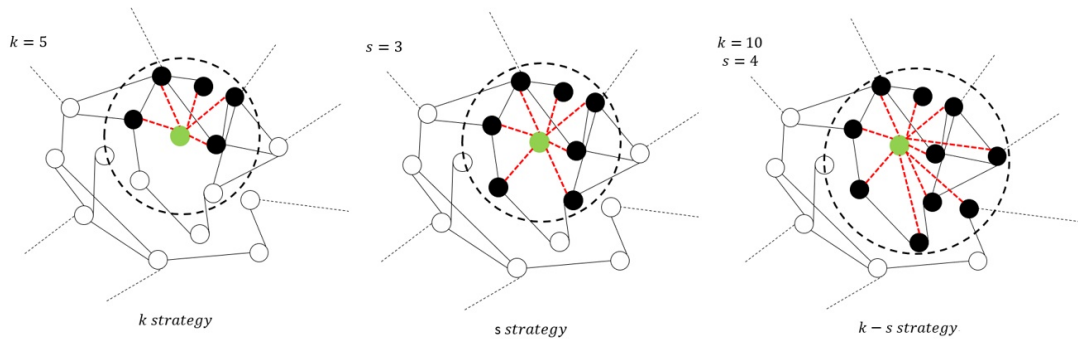


Figure 2.2: Connection Strategies

Sampling Scheme - Selecting a good sampling scheme is one of the biggest challenges of the PRM analysis ([17]) for the problem of existence of *Narrow Passages* in the configuration space. This happens when the passages through which the planning is to be done are too narrow that the local planner is not able to make

a connection between two separated connected components and hence fails to plan a path. In other words, the visibility of the local planner is limited by the physical narrow passage. Although there are various sophisticated approaches for sampling the narrow passages ([13], [12]), the most straight forward is the uniform sampling of the space.

PRM has many variants which rely on the same underlying concepts and these are defined to enhance the capability of the basic approach, some of them includes, Visibility PRM ([34]), Fuzzy PRM ([25]), Lazy PRM ([3]), Probabilistic cell decomposition ([23]), etc. Moreover, it is important to note that PRM is probabilistically complete and asymptotically optimal in theory, that is, if given sufficient time, it can find an optimal path. However due to the use of a connection strategy to select the milestones for Nearest Neighbor set and for adjusting computation time, these property faces certain implications which are demonstrated by the analysis performed by [15]. Moreover, by these properties, it is easy to deduce that the path provided by the query phase might not be optimal and smooth, but it is optimal for the roadmap constructed within the termination condition, which when goes to infinity, ideally provides an optimal path. Additionally, the paths generated by PRM are not visually smooth, hence the path generation phase is often followed by post-processing for smoothness ([9]).

2.1.2 Rapidly-exploring Random Trees

Rapidly-exploring Radon Trees, abbreviated as RRTs, is another family of sampling-based path planning algorithms introduced in [20]. A major breakthrough in optimal planning using RRT is the RRT* algorithm which was first introduced by [15]. The

algorithm is proved to be asymptotically optimal, that is, given sufficient time, then it can produce optimal solutions. Being random in nature, most of the properties of PRM holds true, except this family of planners is single query and follows a *Tree* structure, which is a special form of minimally connected graph, with only one path between two configurations, as opposed to a traditional graph which can have more than one paths between two configurations.

RRT* follows a *Tree Expansion* which is combination of short path organized to get a complete path from start configuration to the goal configuration. The procedure works as follows. First a sample, z_{new} is taken from the environment and check against collision with the obstacle using a collision-checker, if the sample is valid it tries to connect to the nearest node, z_{near} , in the tree using the local planner. Similar to PRM, the connection can be limited by the distance between the nodes. However, if the new node is not connected to previous nodes in the tree, the sample is rejected. In the second step, the node after connection to the nearest node, tends to find a *Best Parent* for itself through a procedure called *Rewiring*. The default procedure of selecting the candidate nodes for best parent, z_{parent} , is same as k strategy in PRM. The node selects the parent which makes the shortest path to the initial node and rewire to the best parent, removing the connection to the nearest node. This procedure is shown in the Figure 2.3. This procedure of rewiring contributes to the asymptotic optimal property of RRT*.

Due to the fast path computation and tree structure, the RRTs are mostly researched for path planning in dynamic environments. [26] provides a comprehensive survey for all the variants of the RRT planners. Moreover, it is important to note that, like PRM choosing local planner, nearest neighbor and sampling scheme is an

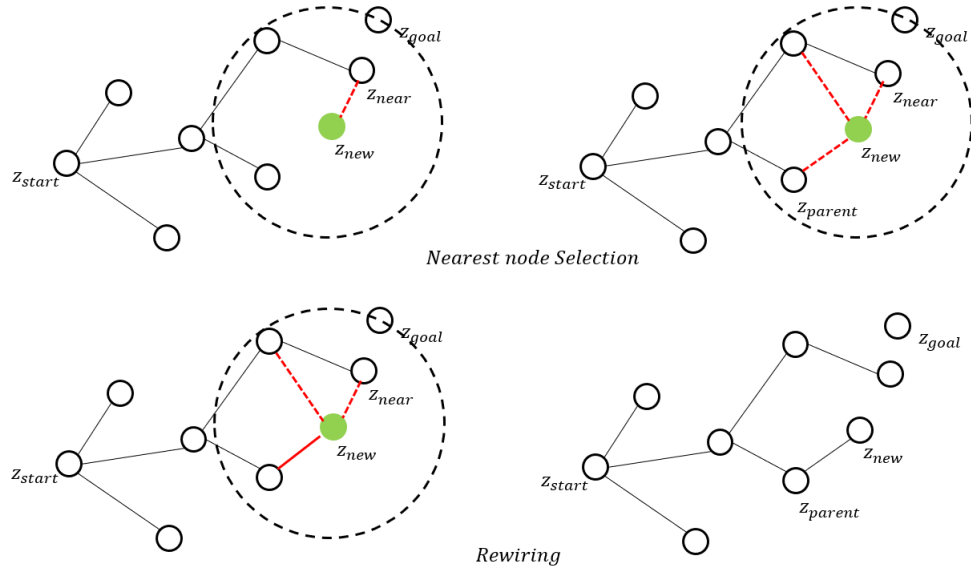


Figure 2.3: RRT* Rewiring Procedure

important choice for the RRT planners. However, RRTs does not have a concept of connected components as there is no fully connected graph. Therefore, it cannot divide the regions with the additional uncertainty constrains which should be made flexible for critical regions which is the main focus of this thesis. Hence, PRM makes the best choice for the implementation of the objectives described.

2.1.3 Safe Path Planners

As already defined, the Safe Path planning is a subjective term and depends on the problem definition but the baseline for safety at any time in path planning is a collision-free path in the configuration space, from initial configuration to the goal configuration.

Depending on the problem at hand, researches consider safety in various forms. Most commonly defined safety is in the planning in dynamic environments where the

planner has either no information or incomplete information about the environment for the time path planning and motion of the robot is done. [2] considers the incomplete information of the environment taking kinodynamic constraints of the robot for re-planning in environments with incomplete knowledge. The re-planning is possible in real time as the planner store the information of the previous searches and uses it for future path planning. Another approach by [29] introduced the concept of wait state for mobile wheeled robots where the robot waits at a state in the path by interpreting the motion of the obstacle and move when the path is clear. Therefore, instead of re-planning, the same path is followed avoiding the collision with the obstacle by waiting at a particular state.

The other broad category which considers the safe path planning are the environments with dynamic obstacles but with known trajectories of the obstacles or the other agents in the environment. The approaches like [33], [24], and [38] assumed pre-computation of the regions with high probability of collision with moving obstacles. They considered time as an additional dimensionality so path planning can be done according to the dynamics of the obstacle. Since planner has prior knowledge, the environment can be considered as static in few cases.

[18] performed the first ever systematic study for the evaluation of safety and introduced the concept of *Danger Fields* which are a function of velocity and the distance of the obstacle from the robot. The danger fields are inversely proportional to the distance, in other words, these are strong in the region near to the obstacle. [19] used this concept of danger fields to compute a path in the environments where robots and humans work together. Moreover, [6] used the concept of *Support Vector Machines* to maximize the distance of the vehicle robot from the obstacle, maintain-

ing equidistant path from the obstacle. In other works they managed to propose an innovative solution to make vehicle path in the center of the passage through which it passes. However, they considered only the symmetric environments, and hence the approach is likely to create unnecessary deviations and sub-optimal paths.

This thesis addresses the safety as to stay away from an obstacle, as described by danger fields, to avoid collisions because of the localization uncertainty. Moreover, making this parameter flexible in critical regions is also an objective which, to the best of knowledge, is not addressed in literature. Hence, the importance and innovation of approach lie in the fact that it follows the uncertainty constrains to find the shortest path in all the regions except the critical regions.

2.2 Case Study - Computer Assisted Surgical Trainer

Laparoscopic surgery, also called, *Minimally Invasive Surgery* (MIS) provides immense benefits to the patient under medical operation. With the benefits like reduction in blood loss, less post operative pain, less recovery time and smaller incisions, this remains a common operative approach for surgical operations as opposed to open surgery. These benefits to the patients come with more constrained operating environments to the surgeons. Surgeons perform surgery using a few inches long and thin instruments which are inserted using incisions into the patient's body along with a fiber optic cable to give surgeon a view of the operating site on a two dimensional screen (Figure 2.4). Various challenges in the procedure, as described in [8], include minimal haptic feedback, single camera view with limited depth perception making hand eye coordination difficult, limited motion flexibility at the surgical site

resulting in loss of dexterity, and adding to these is counter intuitive movement of the instrument.



Figure 2.4: MIS Operating Room [1]

The surgeon has to perform numerous tasks in the operating room in order to complete the surgery: suturing, object transfer, grasping, instrument navigation, etc. However the challenges described above makes it difficult for the surgeon to perform the tasks and he/she might require significant amount of practice to be successful. Fortunately, many simulation trainers are available for helping train a trainee surgeon to prepare for the actual surgery in the operating room in relatively less time as compared to traditional mentor based training, which is sometimes subjective. There are various inexpensive off the shelf "box" trainers and sophisticated commercially available trainers ([14], [27], [28], [35]) but none can be considered as perfect for training. However, *Computer Assisted Surgical Trainer* (CAST) ([32]) developed at the Model based Design Laboratory, University of Arizona have com-

ponents that try to overcome the difficulty of MIS. CAST, shown in Figure 2.5 is an attempt to bridge the gap between inexpensive off the shelf "box" trainers and expensive and sophisticated, commercially available high-end VR trainers.

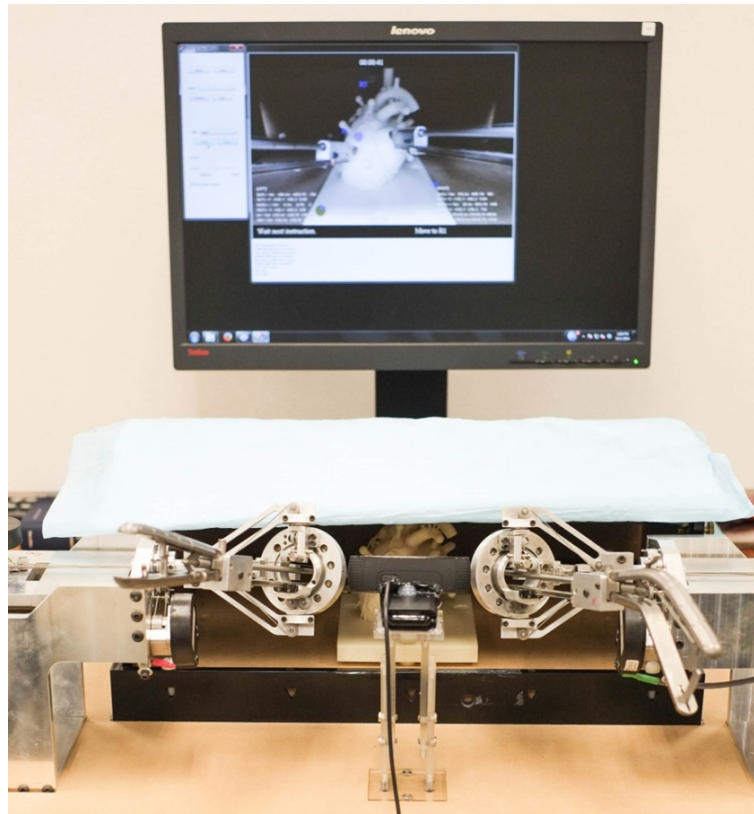


Figure 2.5: Computer Assisted Surgical Trainer

This thesis focuses on the safe path planning of the instrument tip for the navigation task so as to avoid the collision which might have adverse effects like internal bleeding while operating in the operating room. For training the trainees CAST provides visual guidance ([40]) and haptic feedback for the task of path navigation by using a system of electromechanical custom build gimbal system copying the laparoscopic instrument used in actual surgery. The CAST system is a sophisticated

attempt to transform the way laparoscopic trainers are trained but for the purpose of this thesis a simplified version of the CAST configuration is used to demonstrate the proposed planner in action.

2.3 Problem Definition

In the ideal case, a path planner provides a suggested path for the purpose of instrument tip navigation task and it is considered that the suggested instrument movements would be safe enough to avoid collisions with the environment if navigation is done perfectly. However, surgeons get better in the task of navigation with the experience and the novice trainees have to practice a lot in simulated environments before they perform the actual surgery. Moreover, it is logical to assume that due to inexperience in the initial phases of training, a trainee is expected to deviate from the suggested path which in case of actual operating room might lead to injuries such as internal bleeding due to instrument tip collision with the sensitive internal organs. The deviation, in this case, can be modeled as uncertainty in location of the instrument tip and is related to a certain *Proficiency Level* of the trainee under practice.

Consider a training scenario, where the recommended path for an instrument navigation task passes close to the obstacle but due to inexperience, there is a high possibility that a novice trainee would hit the obstacle while performing the task. Intuitively, it is better to have a path planner that considers trainees Proficiency Level and generates tasks where paths are sufficiently distant from the obstacle, so that the possibility that trainee hits the obstacle decreases. Therefore, according to

the safety criteria defined above, each portion of the path should be at reasonable distance from the obstacle to minimize collision possibility. The concept of Cumulative Danger Fields describes danger field at a point as a function of the position of a point and a velocity vector. For the purpose of this paper, we are only interested in the position of the point. By analysis in [18], it can be said that the farther a point is from the obstacle, the lower is the danger field. Moreover, Proficiency Level depends on how perfectly a trainee performs the instrument navigation task ([30]), for instance, an experienced surgeon tends to deviate less from the suggested path as compared to a novice trainee and hence it can be said that the former has higher proficiency level than the latter.

However, there may be certain regions (critical regions) in the environment, where if the path is not allowed to pass close to the obstacle, most path planners may fail to return a path. Although, there is a risk while navigating through these regions, compensating safety just for critical regions guarantees a path, which is safe except when navigating close to obstacles those cannot be avoided. Next chapter describes the stated algorithm in all algorithmic details. The next chapter provides the simulation results using the instrument constraints of CAST and demonstrates the algorithm in action.

CHAPTER 3

ADAPTABLE PRM

This chapter explains the algorithm in detail, which helps to achieve the objectives described earlier in the document. First, the problem is formulated using a mathematical description, which forms the basis of rest of the thesis formulation. Second, since this is a PRM based approach, PRM re-defined using the mathematical formulation and algorithmic steps. Finally, the new proposed Adaptable PRM is described in detail.

3.1 Problem Formulation

In this section, mathematical formulation of already described terms is presented and few important terms are introduced with their definitions which will be used in rest of thesis. Let C be the configuration space, where, $C \in R^d$, $d \in N$, $d \geq 2$. The obstacle space is defined by C_{obs} and the obstacle-free region is defined by C_{free} , where $C_{obs} \subset C$, $C_{free} = C \setminus C_{obs}$ and $C = C_{obs} \cup C_{free}$. For sampling-based path planning algorithms, a path is represented by connected intermediate configurations between the initial and goal configurations which are obtained by sampling C_{free} . The discrete set of samples. or configurations is represented by X , where each sample $x_i \in X$, $i \in N$ is sampled in the free region, C_{free} . A typical path planner is based on generating x_i in an iterative manner and connecting them by Edges, with an aim to find a path between two configurations formed with two or more

nodes connected by edges. The sampling and connection procedure in general forms a tree or a graph as already described and differs by application's preference. Let $P[p_0, p_1, \dots, p_j, \dots, p_{n-1}] \rightarrow R^d$ represent a path, where $p_j \in P$, $j \in [0, n-1]$, $j, n \in N$, where p_j and p_{j+1} , $j \in 0, n-2$, are intermediate configurations connected by edges. Let x_{init} and x_{goal} be the initial and goal, $x_{init}, x_{goal} \in X$, respectively. For a path, P , it's convenient to say that $p_0 = x_{init}$ and $p_{n-1} = x_{goal}$.

To satisfy the safety criteria for this thesis, the path generated by the planner should maintain a minimum distance, d_{thresh} , $d_{thresh} \geq 0$, from the obstacles. In other words, d_{thresh} is the proficiency level of the trainee (or the maximum uncertainty in localization in general). Additionally, d_{thresh} depends on the expertise of the trainee; higher proficiency means less d_{thresh} . The value of d_{thresh} is application dependent parameter that is tuned based on domain and geometric scale of the environment. The Euclidean Distance, d of x_i from the nearest point of the nearest obstacle is termed *Clearance*. d_{thresh} can be added to the environment simply by considering a Virtual Boundary of thickness d_{thresh} around an obstacle as an uncertainty constraint, as shown in Figure 3.1(a). It is important to note that existence of virtual boundary modifies the configuration space. The purpose of virtual boundary is to avoid the connection by local planner when the local path passes through virtual boundary and if possible, an alternate path should be found (Figure 3.1(a)). However, considering the virtual boundaries as hard impassable obstacles might block the path construction by separating the regions containing start and goal as shown in Figure 3.1(b), we call these as Critical Regions. In these environments, most planners will fail to return the path if it is to be guaranteed outside the virtual boundaries. Since a path planner should provide a path, whenever one exists, it is

better to allow planning virtual boundaries (Figure 3.1(c)) just for Critical Regions and consider the virtual boundary constraints for the rest of environment.

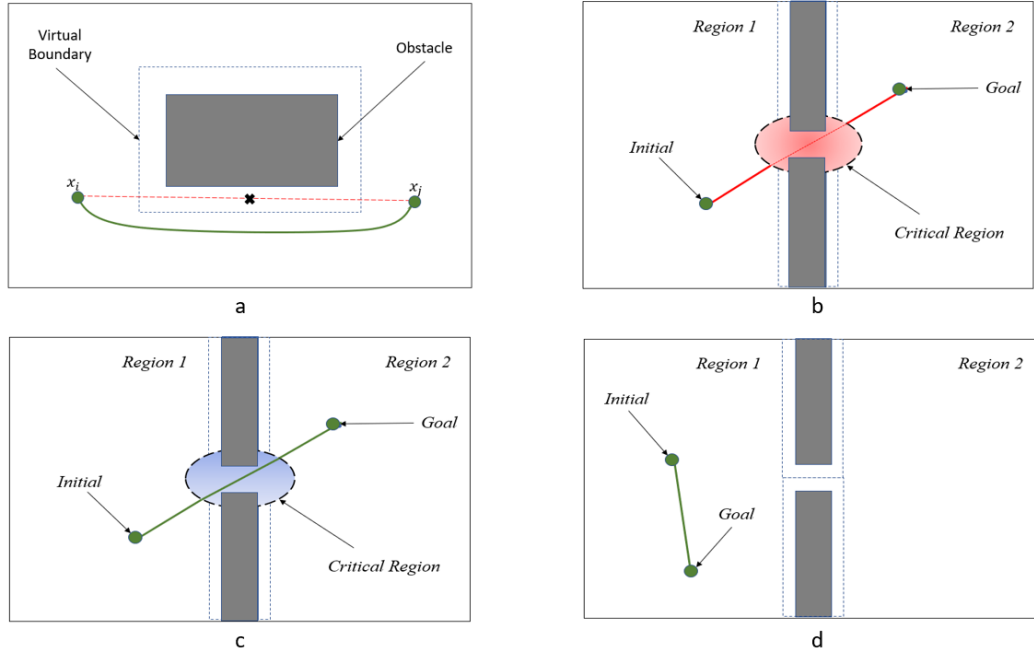


Figure 3.1: Path Planning Examples showing virtual Boundaries and Critical Regions

Additionally, note that critical regions should be defined relative to the path as they might or might not be important depending on the locations of the initial and goal configurations, for example, if both are in the same region (Figure 1(d)), either Region 1 or Region 2, then there is no need to care for the Critical Regions. Moreover, scanning the whole environment to identify critical regions can be another approach to make constraints flexible in these regions but it is an expensive process and would not be beneficial in cases as shown in Figure 3.1(d).

To solve the problem described above in Figure 3.1(b), two separate configuration spaces are constructed according to virtual boundaries. Therefore, based on the

virtual boundary, C_{free} can be divided into C_{safe} , space outside the virtual boundary and C_{risk} , space inside the virtual boundary, $C_{safe} \cup C_{risk} = C_{free}$ and $C_{safe} \cap C_{risk} = \emptyset$. Therefore, x_i that lies in C_{safe} is added to X_{safe} , where $X_{safe} \subset X$ and those in C_{risk} are added to X_{risk} , where $X_{risk} \subset X$, $X_{safe} \cup X_{risk} = X$ and $X_{safe} \cap X_{risk} = \emptyset$. For explanation, these sets are termed to be sampled from the *Safe Region* and the *Risk Region*, respectively. The main contribution of this thesis is to provide a method of path planning to generate a path in C_{safe} except for the regions where C_{risk} cannot be avoided. Intuitively, planning is done by toggling the planner to plan in two configuration space when needed, hence, when there is no requirement, the planner stays in one configuration space and hence it is adaptable to the original planner.

There are various sampling-based algorithms for path planning, but PRM is helpful for planning in two configuration spaces. Its graph structure, particularly connected components helps decide on when to toggle from one configuration space to others to make the uncertainty constraint flexible. For the problem at hand, in Figure 3.1(b), the initial and goal configurations are in different connected components (region 1 and region 2, respectively), however, in Figure 3.1(d) they are in same one, that is Region 1.

Basic PRM as described, plans a path in two phases, Learning Phase and Query Phase. Learning Phase (Algorithm 1 and Algorithm 2) constructs a Roadmap, R , using milestones, $\forall x_i \in X$, connected with edges sampled from C_{free} . A set of nodes, N , which contains all milestones is maintained. Connection Strategy provides a sorted list of Nearest Neighbors, N_c , from N based on distance d from newly added milestone. A new milestone x_i attempts to connect to every n milestone, $n \in N_c$,

and successful valid direct connection adds an edge to the existing set of edges, E and update the R s connected component. Update of R s connected component is an important step since the new milestone can combine two or more connected components into one, which increases the probability of having the initial and goal points in the same connected component, an essential condition for finding a path. Learning Phase procedure provides R , which helps execute the second phase, that is, the Query Phase (Algorithm 3). This procedure adds x_{init} and x_{goal} to the roadmap and tries to find a connecting path, P , using A* search. Moreover, a path is a set of intermediate connected milestones, hence, x_{init} and x_{goal} should lie in the same connected component to guarantee a path.

Algorithm 1: Roadmap

Data: *PlannerTerminationCondition*

Result: R

begin

$N \leftarrow \emptyset$

$E \leftarrow \emptyset$

$R \leftarrow (N, E)$

while *PlannerTerminationCondition* **do**

$x_i \leftarrow \mathbf{sample}(C_{free})$

$R \leftarrow \mathbf{addmilestone}(x_i, R)$

Moreover, as it would become clear, selecting which configuration space to plan in requires sufficient information about the path, thus $K - S$ -strategy is a good choice for selecting nearest neighbor set. Next section describes the modified PRM algorithm which provides a potential solution to the problem of finding a path by making the uncertainty constraints flexible.

Algorithm 2: Addmilestone

Data: c, R
Result: R
begin
 $N_c \leftarrow \mathbf{ConnectionStrategy}(c, R)$
 $N \leftarrow N \cup c$
 for $n \in N_c$ *sorted by increasing* $d(c, n)$ **do**
 if $\mathbf{validConnection}(c, n)$ **then**
 $E \leftarrow E \in (c, n)$
 update R 's Connected Component

Algorithm 3: Query

Data: x_{init}, x_{goal}, R
Result: P
begin
 $P \leftarrow \emptyset$
 $R \leftarrow \mathbf{addmilestone}(x_{init}, R)$
 $R \leftarrow \mathbf{addmilestone}(x_{goal}, R)$
 if $\mathbf{sameComponent}(x_{goal}, x_{init})$ **then**
 $P \leftarrow \mathbf{getPath}(x_{goal}, x_{init})$

3.2 Modified Algorithm

The main idea of the adaptable PRM is based on "repairing" the path to get a "safer" path that satisfies the uncertainty constraints wherever possible. Given an environment, a planner should provide a path between the initial and goal configurations, but if it does not meet the specified criteria for clearance it might fail to produce a path. Hence, in this section, a modified PRM (Algorithm 4) is described that repairs a path generated by PRM to meet the stated requirements.

Algorithm 4: AdaptivePRM

Data: *PlannerTerminationCondition*, x_{init} , x_{goal} , C_{free} , C_{safe}

Result: P

begin

$R, R_{safe} \leftarrow \emptyset$

if *not*(**Valid**(x_{init})) *or* *not*(**Valid**(x_{goal})) **then**

\lfloor *return* *invalidConfiguration*

while *PlannerTerminationCondition* **do**

$x_i \leftarrow \mathbf{sample}(C_{free})$

if x_i *in* C_{safe} **then**

$R_{safe} \leftarrow \mathbf{addmilestone}(x_i, R_{safe})$

$R \leftarrow \mathbf{addmilestone}(x_i, R)$

if P *exists* **then**

if *not*($\forall p_i \in P$ *in* R_{safe}) **then**

$P \leftarrow \mathbf{pathRepair}(P, R_{safe})$

\lfloor *return* P

\lfloor *return* *noPath*

For this approach, in addition to constructing only one roadmap, R , as in basic PRM where milestones are sampled from C_{free} , we maintain an additional roadmap R_{safe} , where milestones are sampled only from C_{safe} . Before that, x_{init} and x_{goal} are checked for validity, that is if these lie in the valid space, C_{free} . If yes, then

the two mentioned Roadmaps are constructed. After adding x_{init} and x_{goal} to R , and a Query is made in R , which returns a path, if one exists (Figure 3.2(a)). The path may pass through Safe Region and/or Risk Region, so we need to inspect the path if its portion is inside the risk region. For this, we should have sufficient information about the path, ideally, the milestones constructing the path and the C_{free} should be infinite, $n \rightarrow \infty$. and the graph should be fully connected. This is computationally impractical, hence to simplify, we use the $K - S$ -strategy, which can tune parameters like distance between two milestones, s and maximum nearest neighbors k for adjusting the computation time for Roadmap construction and to maintain sufficient information about the path. Moreover, because of PRM being conditionally Probabilistic Complete and Asymptotically Optimal, the path generated at given query and configuration is considered the best path, although it might not be intuitively shortest. The path P is cached and if all milestones in the path are in R_{safe} , then the path is already safe and the algorithm returns this path, otherwise procedure PathRepair, which is the heart of the algorithm, repairs P according to desired clearance. However, x_{init} and x_{goal} might be inside C_{safe} . these can be projected to C_{safe} and corresponding path can be saved. These are accumulated in the end for the final path.

Procedure PathRepair (Algorithm 5) operates as follows: first, the milestones constituting the path are marked invalid if they are not in R_{safe} , as shown in red in Figure 3.2(b). The second step involves initiating the first valid milestone as start and next as goal (Figure 3.2(b)), then it traverses all the valid milestones to make the last milestone on the path in the same connected component as the goal (Figure 3.2(c)). After that Query between the new initial (p_l) and new goal (p_m)

Algorithm 5: PathRepair

Data: P, R_{safe}
Result: P
begin
 $I \leftarrow \forall p_i \in P \text{ not in } R_{safe}$
 $l \leftarrow 0$
while $l < n-1$ **do**
while $p_l \in I$ **do**
 $l++$
 $q \leftarrow l$
while $q < n-1$ **do**
 $q++$
if $p_q \notin I$ **then**
if $\text{sameComponent}(p_l, p_q)$ **then**
 $m \leftarrow q$
 $P'_{l \rightarrow m} \leftarrow \text{query}(p_l, p_m, P_{safe})$
if $P'_{l \rightarrow m} \neq \emptyset$ **then**
 $P_{l \rightarrow m} \leftarrow P'_{l \rightarrow m}$

 update l, n
 $\text{return } P$

configuration provides a path in Safe Region. Since both the initial and goal points are in the same component, a path is guaranteed. After getting a sub-path, the

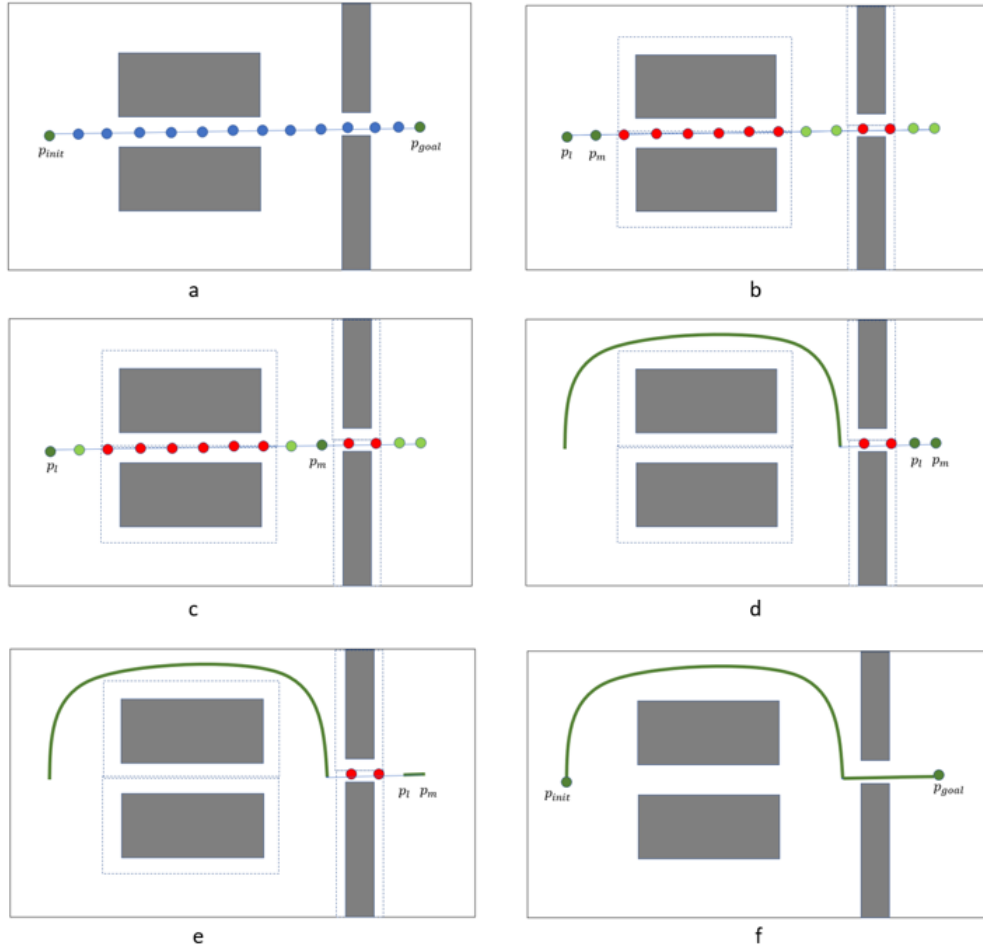


Figure 3.2: Algorithm Steps Illustration

section of the original path from the new initial to new goal is replaced with the newly generated path (Figure 3.2(d)). In the next iteration, the milestone next to the goal is taken as the next initial milestone and the procedure is repeated until it reaches the end milestone in the path (Figure 3.2(e)). In the end, a repaired path

is returned (Figure 3.2(f)).

It is important to note that during path repair, a query is generated by only using the milestones in the safe region, the reconstructed sub-section of the path is optimal for the given graph and is guaranteed to be in the safe region. Moreover, since the path cannot be repaired when checked against Critical Regions, it intuitively ignores the uncertainty constraint in these regions, making those flexible in critical regions.

CHAPTER 4

SIMULATION RESULTS

This chapter provides the setup development environment of the algorithm and the simulation results for problem of instrument tip path planning in an environment to demonstrate the algorithm in action. The simulations satisfy the objectives described in the beginning.

4.1 Experimental setup

The implementation of the algorithm described in Chapter 3 is done on a popular motion planning framework called Open Motion Planning Library (OMPL) ([36]). OMPL is an open source software build for Linux based operating robotic systems and consists of many state-of-the-art sampling-based motion planning algorithms. The CAST is windows based so the first step in the setup involves building this software on Windows. Moreover, the simulation results are visualized using MATLAB.

The simulation setup uses CAST configuration (Figure 4.1), that has two laparoscopic instruments, left and right, mounted on two fixtures, each having a gimbal. To mimic a trocar in laparoscopic surgery, the gimbal center provides four degrees of freedom, namely, insertion, pitch, yaw, and roll, all centered around a single-entry point which corresponds to the incision. For the sake of simplicity of demonstrating the algorithm in action, we consider three degrees of freedom: insertion, pitch, and yaw.

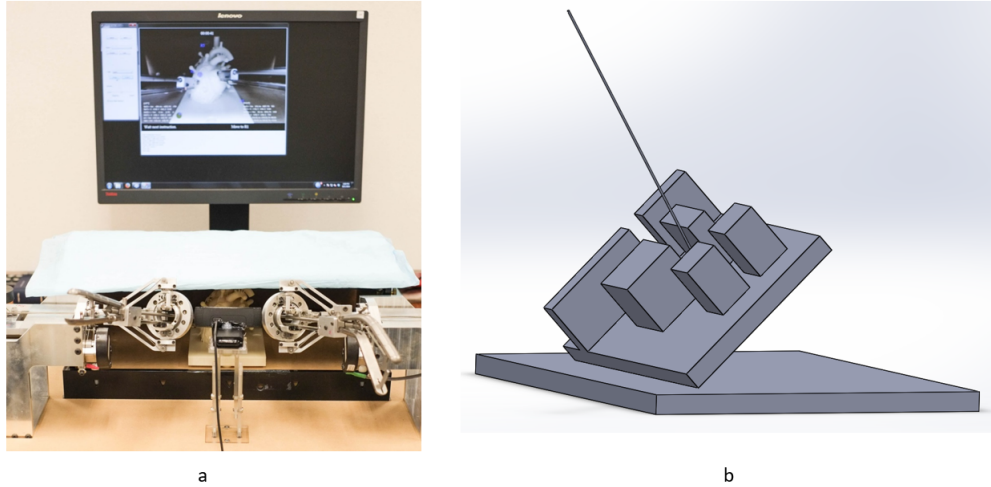


Figure 4.1: (a) CAST, and (b) model used for simulation

The instrument navigation task in CAST requires a trainee to follow a recommended path for task efficiency and collision minimization. The recommended path is generated based on the proficiency level of the trainee. Three proficiency levels of the trainees are considered, namely: Beginner, Intermediate, and Expert, where as the name suggests beginner tends to have maximum deviation from the path navigation task due to not much experience with the surgery, therefore maximum d_{thresh} , while an expert is expected to perform the task almost perfectly, hence minimum d_{thresh} . In other words, a beginner is provided most virtual boundary and for expert, there is no virtual boundary, that is, the path in entire C_{free} as an expert is expected to navigate almost perfectly.

The simulated environment, shown in Figure 4.1(b) is developed as a .dae file and is carefully designed to show the intuitive distinctions in the paths for different proficiency levels. The environment is a 3D model with the actual constraints on pitch, yaw, and insertion as in CAST. For demonstrating results, only the right

instrument is used.

4.2 Selecting k and s

As suggested in the beginning, the importance of $k - s$ strategy lies in the fact that it maintains the information about the path. Therefore, one of the most important steps is selecting the values k and s . The value of k largely determines the time required to construct the graph when the graph becomes dense. Moreover, k and s together represent the connectivity of the graph. Although [15] demonstrates that value of $k = 50$ performs well, it entirely depends on value of s . For better connectivity, large s and k are required, however, this comes with the trade-off by increasing the time for roadmap construction.

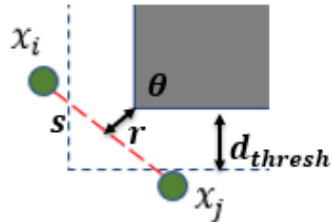


Figure 4.2: Edge Passing through Virtual Boundary

The value of s is an important parameter to select and should be selected very carefully. This is because of the visibility property ([10]) of PRM, that is, every milestone connects to the one it "sees" if all conditions are satisfied. Since the virtual boundaries selected for modeling uncertainty are not real objects, two nodes can be connected if they are visible to each other even if edge passes through the virtual boundaries as shown in Figure 4.2. Checking for passing virtual boundary is

expensive and can take exponential time. Therefore, around path can pass through virtual boundaries around the corners of the obstacle with the path as near as r units from obstacle. Equation provides a relation between the value of s , d_{thresh} and r .

$$r = d_{thresh} \times 2 \cos(\theta/2) + s \times \sin(\theta/2)$$

where θ is the angle, the nearest point on the obstacle makes with the obstacle surface. The equation provides the lower limit on the nearest distance a path can get to the obstacle while passing through the virtual boundary of the obstacles around the corner.

4.3 Results

Figure 4.3 shows the simulation results performed for three levels of proficiency in the environment ($150 \times 100 \times 40mm$) shown in Figure 4.1(b). As described, the value of d_{thresh} depends on application and geometric scale of the environment, hence for demonstrating the proof-of-concept, value of d_{thresh} selected according to environment dimensions, though other values can be selected. Path length is selected as optimization criteria. The simulation use $k - s$ strategy with value of $s = 15mm$ and $k = 50mm$. However other values can be selected.

Figure 4.3(a) illustrates a recommended path for an expert; it considers no virtual boundary ($d_{thresh} = 0mm$), because of no expected deviation. Therefore, the new path planner generates the path a PRM planner with same strategy configuration would provide. Therefore, this works similar to traditional planner in case there is no virtual boundary.

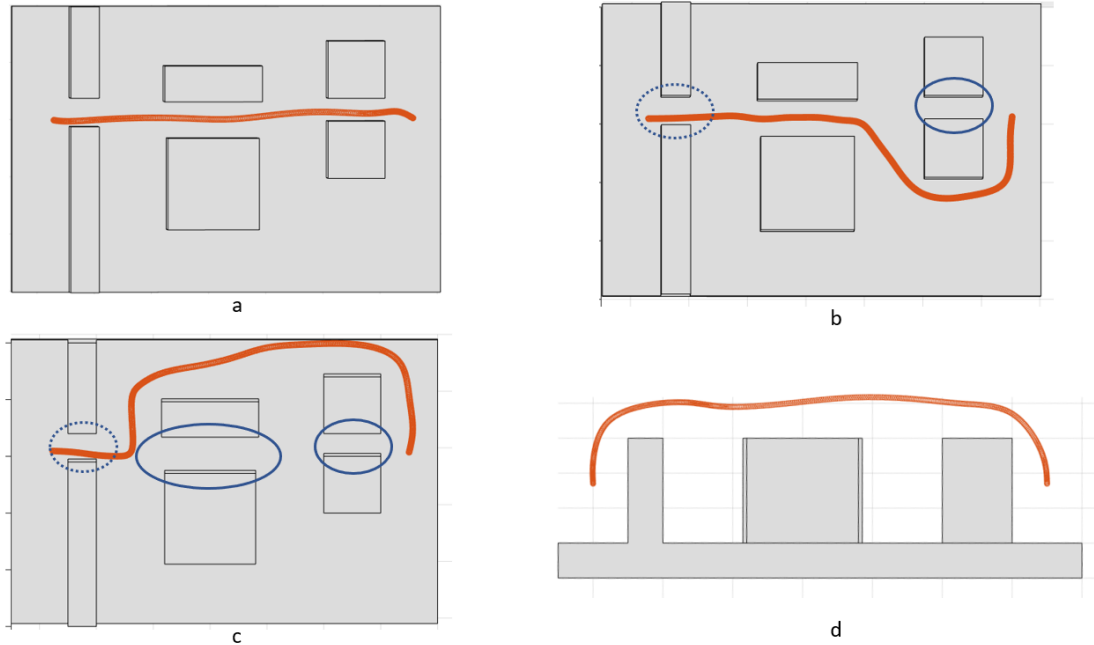


Figure 4.3: Simulation results: solid oval - region where detour is possible, dashed oval - critical region

The path shown in Figure 4.3(b) is suggested for an intermediate trainee, hence it considers a virtual boundary ($d_{thresh} = 4mm$) around the obstacle. Though not depicted in the figure, d_{thresh} creates a virtual boundary that blocks the passage between the right-most (solid oval marking) and the left-most pair (dashed oval markings) of obstacles. The traditional planners, in this case, would fail because of no possible detour due to blockage of the left-most region, which is a critical region. However, due to possibility of detour around the right-most pair of obstacles, the proposed planner finds an alternative path and allow planning through the left-most region as there was no alternate path around the left-most blocked region. Thus, the path repairs for the regions where detour is possible but allow passing through the critical regions. This typically involves toggling from on configuration space to

the other when there is a possibility to re-plan, replacing the partial path in one configuration with the path in other configuration (right-most region) and retaining the path in base configuration where path is not possible otherwise.

Figure 4.3(c) demonstrates the same concept but with increased thickness of the virtual boundaries ($d_{thresh} = 8mm$) blocking the passage between all three pair of obstacles. Therefore, like the previous case, it detours where an alternate path is possible. It should be noted that these simulations are performed by considering C_{free} in a bounding box ($150 \times 100 \times 40mm$) fixed to the environments dimensions, but for the beginner(or any other level), if dimensions are extended (i.e., bounding box change to $(150 \times 100 \times 80mm)$), a path that guarantees d_{thresh} is obtained, Figure 4.3(d). This is because there is a possibility to completely plan a path in safe region. This path is like any planner with guaranteed d_{thresh} and same strategy would have provided. Hence, this demonstrates that the proposed method adapts well to the traditional planner if the space allows entire planning in safe region, otherwise, it provides repaired path as shown in simulations.

Figure 4.4 and Figure 4.6 provides interesting results and validation of the concept for a total of 90 simulations for 9 different values of k . The values marked for k are averaged for 10 simulations. As anticipated, the path length is optimises with the increasing value of k (which means increase in information and hence shorter path length). The average path length for all these simulation was found out to be 168.86 mm and standard deviation was 2.26 mm which can be considered good, but for simulations greater than or equal to $k = 100$, standard deviation was found to be 0.53 mm which is acceptable for path length optimization.

Moreover, the algorithm provides very consistent paths. Figure 4.5 shows 10

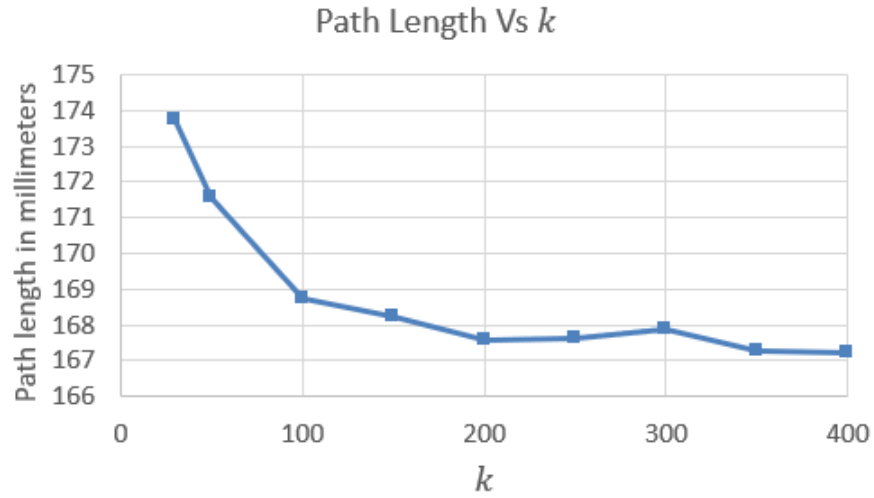


Figure 4.4: Path Length Analysis

paths simulated for intermediate level trainee, visualized on the same environment. It can be seen clearly that the paths have pretty consistent shapes.

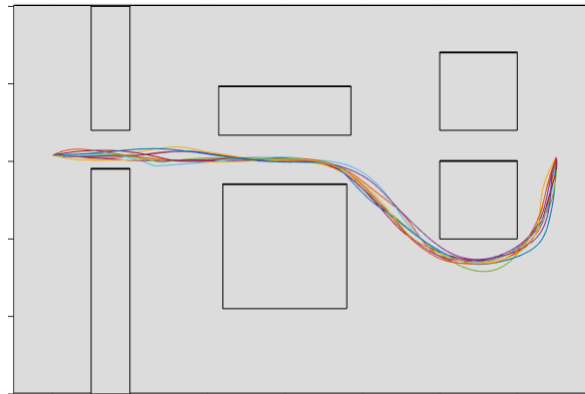


Figure 4.5: Multiple-paths Comparison

Moreover, it should be noted that there are two important steps in AdaptablePRM, one is constructing an additional graph and second is the PathRepair subroutine. Figure 4.6 represents the time overhead for construction of the additional graph using the same metrics described above and PathRepair subroutine

takes an additional approx. 378 milli-seconds average time overhead, where query phase is multi-threaded. Therefore, based on these simulations, the average overhead approximates to 650 milli-seconds on the environment, which is quite acceptable for most applications in general.

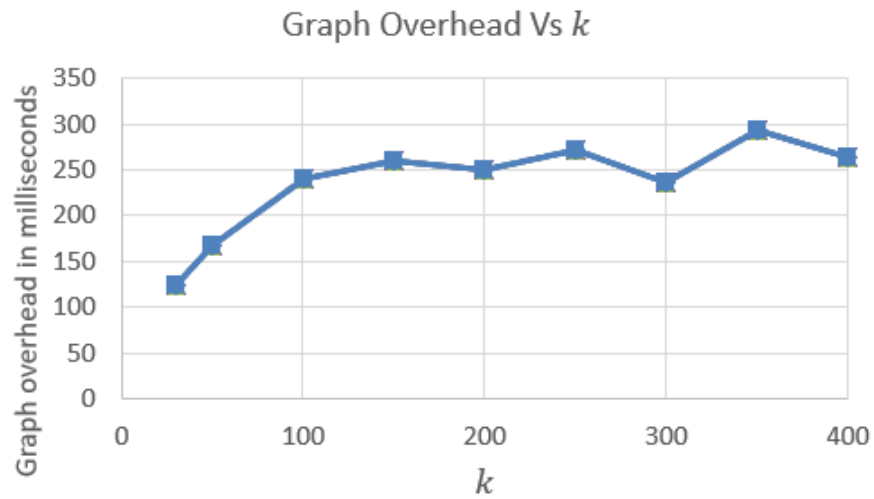


Figure 4.6: Graph Construction Overhead Time

CHAPTER 5

CONCLUSION AND DISCUSSIONS

This thesis introduced a new adaptive algorithm based on PRM which takes the uncertainty in localization as a constraint to avoid collision from the environment while planning a path from initial configuration to the goal configuration. The importance of the planner lies in the fact that a robot cannot navigate a path perfectly, either because of noise in controls or accuracy in localization, and the amount of noise can change during mid-flight. This can change the configuration space of the environment where the path planning is performed and sometimes might separate the two regions containing Initial and goal configuration, hence blocking the path. However, the fact that these uncertainty constraints are induced due to external effects and not by the robot design, involving a bit of risk while passing these regions which block the path planning might be helpful for planing a path which is safe in the rest of regions except the risk regions. This is done by constructing the base path first without considering uncertainties and then repair the path to generate a safer path. This is done by constructing two configuration spaces, planning base path in the space without uncertainty constraint and toggling to the safe space whenever possible. This automatically avoids the critical regions even without need of identification.

Alternatively, the proposed algorithm provides a path which avoids entering risk regions in the free space except for the critical regions, as these are necessary to

navigate to compute a path for the navigation task. In other words, the proposed method allows planning through critical regions by ignoring the distance bounds in these regions and not in rest of the space. The simulation results successfully demonstrate the algorithm in action and provide a lower bound on the distance a path can get close to an obstacle around the corner.

However, there are a few important things to notice. First, it should be noted that since the modification is based on repairing a path when it lies inside the Risk Regions, therefore, there should be at least two milestones in the safe region for the repair to take place. In other words, if the path completely lies in the risk region, a repaired path cannot be computed, and the algorithm provides a path which is completely inside the risk region. The algorithm in this case, that is the worst-case, works like a traditional planner without taking virtual boundaries into consideration. Alternatively, if the path completely lies in the safe region, for example, the configuration shown in Figure 3.1(d), then no repair is required. Moreover, the path always may not lie completely in the safe region, so it is difficult to annotate the path as safe. Therefore, the path is called to be safer when repair is performed on the path. Moreover, in Chapter 3, projecting the initial and/or goal state if they are inside the risk region to the safe region is performed using an incremental sphere around them and selecting the nearest sample which lies inside the safe region. The projected and original configurations and their paths are connected at appropriate locations for the final path.

The simulations are performed to demonstrate the algorithm in action using medical applications in Minimally Invasive Surgery, but it is equally applicable to other systems involving path planning problem for navigation. For instance, appli-

cation to the UAVs are explained simultaneously throughout the thesis. Moreover, uncertainty constraint can be modeled to distance according to the application, for example Proficiency level in the for the instrument navigation in CAST or error in localization for UAVs. Finally, the present research suggests further improvement. Due to the use of the K-S strategy, the quality of the path is not sufficiently smooth, therefore for future work, we will focus on the quality of the path, which involves finding a generic smoothing function for almost all applications. Moreover, we consider virtual boundaries to construct the safe regions, but some paths, due to Visibility property of PRM, may cross virtual boundaries around the corners (as explained in Chapter 4), making a small section of the path occasionally closer than expected. However, for perfectly avoiding the virtual boundaries, infinite milestones would be required which is not practically possible. Therefore, exploring how to avoid such scenarios is another research scope. Moreover, a human study would be important to verify if the proposed method would actually be helpful to assist laparoscopic trainees in overcoming difficulties of MIS surgery.

REFERENCES

- [1] Gi & laparoscopic surgery. <http://danashivamhospital.com/gi-laparoscopic-surgery.html>. Accessed: 2019-03-23.
- [2] K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 704–710. IEEE, 2007.
- [3] R. Bohlin and L. E. Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000.
- [4] H. Choset. *16-735, Howie Choset with slides from G.D. Hager, Z. Dodds, and Dinesh Mocha.*
- [5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
- [6] Q. H. Do, H. T. N. Nejad, K. Yoneda, S. Ryohei, and S. Mita. Vehicle path planning with maximizing safe margin for driving using lagrange multipliers. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 171–176. IEEE, 2013.
- [7] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.
- [8] A. G. Gallagher and G. C. O’Sullivan. *Fundamentals of surgical simulation: principles and practice.* Springer Science & Business Media, 2011.
- [9] R. Geraerts and M. H. Overmars. Clearance based path optimization for motion planning. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*, volume 3, pages 2386–2392. IEEE, 2004.
- [10] R. Geraerts and M. H. Overmars. Reachability analysis of sampling based planners. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 404–410. IEEE, 2005.

- [11] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [12] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, volume 3, pages 4420–4426. IEEE, 2003.
- [13] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin, et al. 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.
- [14] N. Jaber. The basket trainer: a homemade laparoscopic trainer attainable to every resident. *Journal of minimal access surgery*, 6(1):3, 2010.
- [15] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
- [16] L. Kavraki, P. Svestka, and M. H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, volume 1994. 1994.
- [17] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3020–3025. IEEE, 1996.
- [18] B. Lacevic and P. Rocco. Kinetostatic danger field—a novel safety assessment for human-robot interaction. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2169–2174. IEEE, 2010.
- [19] B. Lacevic and P. Rocco. Towards a complete safe path planning for robotic manipulators. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5366–5371. IEEE, 2010.
- [20] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.
- [21] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [22] M. Likhachev, G. J. Gordon, and S. Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *Advances in neural information processing systems*, pages 767–774, 2004.

- [23] F. Lingelbach. Path planning for mobile manipulation using probabilistic cell decomposition. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2807–2812. IEEE, 2004.
- [24] V. Narayanan, M. Phillips, and M. Likhachev. Anytime safe interval path planning for dynamic environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4708–4715. IEEE, 2012.
- [25] C. L. Nielsen and L. E. Kavraki. A two level fuzzy prm for manipulation planning. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, volume 3, pages 1716–1721. IEEE, 2000.
- [26] I. Noreen, A. Khan, and Z. Habib. Optimal path planning using rrt* based approaches: a survey and future directions. *Int. J. Adv. Comput. Sci. Appl*, 7(11):97–107, 2016.
- [27] P. Ordonez, P. Kodeswaran, V. Korolev, W. Li, O. Walavalkar, B. Elgamil, A. Joshi, T. Finin, Y. Yesha, and I. George. A ubiquitous context-aware environment for surgical training. In *2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*, pages 1–6. IEEE, 2007.
- [28] T. Pham, L. Roland, K. A. Benson, R. W. Webster, A. G. Gallagher, and R. S. Haluck. Smart tutor: a pilot study of a novel adaptive simulation environment. *Studies in health technology and informatics*, 111:385–389, 2005.
- [29] M. Phillips and M. Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5628–5635. IEEE, 2011.
- [30] M. Riojas, C. Feng, A. Hamilton, and J. Rozenblit. Knowledge elicitation for performance assessment in a computerized surgical training system. *Applied Soft Computing*, 11(4):3697–3708, 2011.
- [31] J. Rosell and P. Iniguez. Path planning using harmonic functions and probabilistic cell decomposition. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1803–1808. IEEE, 2005.
- [32] J. W. Rozenblit, C. Feng, M. Riojas, L. Napalkova, A. J. Hamilton, M. Hong, P. Berthet-Rayne, P. Czapiewski, G. Hwang, J. Nikodem, et al. The computer

- assisted surgical trainer: design, models, and implementation. In *Proceedings of the 2014 Summer Simulation Multiconference*, page 30. Society for Computer Simulation International, 2014.
- [33] D. Silver. Cooperative pathfinding. *AIIDE*, 1:117–122, 2005.
- [34] T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.
- [35] N. Stylopoulos, S. Cotin, S. Maithel, M. Ottensmeyer, P. Jackson, R. Bardsley, P. Neumann, D. Rattner, and S. Dawson. Computer-enhanced laparoscopic training system (celts): bridging the gap. *Surgical Endoscopy and Other Interventional Techniques*, 18(5):782–789, 2004.
- [36] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. <http://ompl.kavrakilab.org>.
- [37] S. Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002.
- [38] J. Van Den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2366–2371. IEEE, 2006.
- [39] J. Vanualailai, B. Sharma, and S.-i. Nakagiri. An asymptotically stable collision-avoidance system. *International Journal of Non-Linear Mechanics*, 43(9):925–932, 2008.
- [40] A. Wagner and J. W. Rozenblit. Augmented reality visual guidance for spatial perception in the computer assisted surgical trainer. In *Proceedings of the Symposium on Modeling and Simulation in Medicine*, page 5. Society for Computer Simulation International, 2017.
- [41] C. W. Warren. Global path planning using artificial potential fields. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 316–321. Ieee, 1989.
- [42] D. T. Wooden. *Graph-based path planning for mobile robots*. PhD thesis, Georgia Institute of Technology, 2006.
- [43] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia. Survey of robot 3d path planning algorithms. *Journal of Control Science and Engineering*, 2016:5, 2016.