A Hybrid Path Planning Technique for Indoor Autonomous Robots Developed by Integrating Global and Local Path Planner

Author

MUHAMMAD IMRAN

NUST201261255MCEME35512F

Supervisor

DR. KUNWAR FARAZ AHMAD KHAN

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

JANUARY, 2016

A Hybrid Path Planning Technique for Indoor Autonomous Robots Developed by Integrating Global and Local Path Planner

Author

MUHAMMAD IMRAN

NUST201261255MCEME35512F

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Mechatronics Engineering

Thesis Supervisor:

DR. KUNWAR FARAZ AHMAD KHAN

Thesis Supervisor's Signature:_____

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,

ISLAMABAD

JANUARY, 2016

Declaration

I certify that this research work titled "*A Hybrid Path Planning Technique for Indoor Autonomous Robots, Developed by Integrating Global and Local Path Planner"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Muhammad Imran

NUST201261255MCEME35512F

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Muhammad Imran

NUST201261255MCEME35512F

Signature of Supervisor

Dr. Kunwar Faraz Ahmad Khan

Copyright Statement

Acknowledgements

I am thankful to Allah Subhana-Watala to have guided me throughout the life and through this work. I am thankful for creating the circumstance and providing me with the opportunities to work on this thesis. I am thankful for every new thought which You setup in my mind to complete and improve my work. Indeed I could have done nothing without Your help and blessings. Whosoever helped me throughout the course of my thesis was Your will, so indeed none be worthy of praise but You.

I am particularly thankful to my parents who brought me up to what I am today and to my wife for her unconditional and continuous support.

I would also like to express special thanks to my supervisor Dr. Kunwar Faraz Ahmad Khan for his extended and ever forthcoming support throughout my thesis. Without his guidance and advice, I would have not been able to accomplish this work.

I would like to express my special gratitude to Dr. Aamer Ahmed Baqai for his round the clock availability to guide and support me for completion of my thesis.

I would also like to acknowledge the support and guidance of Dr. Umar Shahbaz Khan and Dr. Khurram Kamal whose worthy ideas helped me to improve my work.

I can not forget to acknowledge the support of Brigadier Hassan Akhtar Kayani and Colonel Zulfiqar Ali Shirazi who kept me motivated to reach this milestone.

Dedicated to all those kids who can not make to school,

my parents who could have not done more for me,

my family and siblings whose prayers always carried me along,

my beloved wife and kids

who sacrificed the most while I was busy working on this thesis.

Abstract

Path planning is one of the most critical aspects of developing and employing mobile robots. Apart from finding a suitable path, there is a great concern regarding quality of path as well. Among others, distance traversed from start point to goal is an important measure of quality and so is the safety of robot while adopting this path. Depending upon applications, the requirements from path planners are varying and sometimes conflicting, it is therefore a great deal of research is focused in this field including this one.

This research is an effort to develop a hybrid path planner by integrating visibility graph as global and artificial potential field as local path planner. The development of hybrid path planner by integration of global and local path planners focuses at exploiting their advantages and mitigating their shortcomings. This hybrid path planner, not only generates a near optimal path but also keeps robot safe as it ensures that throughout the course, it does not touches any obstacle or comes closer to it than a user defined distance. The work is also about making hybrid planner robust, flexible and reactive.

Key Words: *Autonomous robots; Autonomous path planning; Hybrid path planning; Intelligent systems; Motion planning; Integration of global and local path planners; Visibility graph and artificial potential field*

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

The research work presented in this dissertation is about developing a hybrid path planning technique for indoor autonomous robots by integrating global and local path planner. The thesis has been organized into chapters. Chapter 1 develops a background about different categories of path planners by briefly narrating their methods, advantages and disadvantages. Chapter 2 presents literature review of relevant researches as regards to this thesis. Chapter 3 is about selection of suitable global and local path planner for the purpose of integration. Chapter 4 presents architecture and methodology of proposed hybrid path planner. Chapter 5 presents the ways and scenarios in which proposed hybrid path planner has been implemented and also the results and chapter 6 which is also the last chapter, concludes this thesis.

1.1 Path Planning – Background

Path planning is, probably, the most focused area of research in the field of mobile robotics. It is known as the art of finding an optimized collision-free path from the starting location toward the predefined destination [1]. Path planning is defined in another way by Siegwart and Nourbakhsh [2] as a trajectory through a map with which a robot can reach a well-known location (goal) from its starting location. It is important to mention that it is not only about finding a path to goal location, the concern about quality of path that is found by a path planner is equally important. The definition of quality regarding path is relative and keeps varying depending upon robots and their applications. Sometimes it is about finding shortest path and at others, concern is about safety of robot. Sometimes measure of quality is time taken by robot to reach its goal and at others, it is power consumption. We also want to keep computational and memory requirements of path planners to a certain limit. In attempts to meet varying and sometimes conflicting requirements, many path planners have been proposed by researchers. These path planning methods have also been categorized and this categorization is mainly based on two aspects. First aspect of classification is completeness i.e. exact or heuristic and the second aspect is their scope i.e. global or local [3].

1.2 Classification of Path Planning Methods Based on Completeness

Completeness of path planner is an important measure and can never be over emphasized. A path planning method basing on its capability of finding solution can be classified as complete, probabilistic complete, resolution complete or heuristic. Considering their completeness, path planning algorithms can be broadly classified in two categories which are given in succeeding paragraphs [3].

1.2.1 Classical Methods

The methods which are well established and had proven their exactness are known as classical methods. If there exists a solution to a given path planning problem, classical methods would find it otherwise they would indicate non availability of path in the given problem.

These methods are characterized by their computational intensiveness and inability to handle variations and uncertainties inherent with real world environments. It is because of these characteristics that their use in real world applications is greatly restricted. Following methods are generally categorized as classical methods [4]: -

- Road Maps
- Cell Decomposition (CD)
- Artificial Potential Field (APF)
- Sub goal Network (SN)

1.2.2 Heurist – Based Methods

Heuristic-based methods are also called in literature as population-based and behavior-based methods [5, 6]. These methods suggest the use of behavioral-based approach to address the constraints of real world environments [7]. In behavioral-based approaches a set of simple, predefined behaviors would be designed in a way to solve complex scenarios. Being dependent on the current state and a set of behavioral rule for that state and not on localization and mapping, the heuristic-based methods are capable of handling dynamic environments. Although these methods have performed reasonably well in dynamic environments, they have limitations with regards to uncertainties which sometimes result in dead lock and robot fails to choose next state [7].

Heuristic-based methods can not always find a solution but it is for sure that if they found one, it would be found in much shorter time as compared to classical methods [8]. It is also pertinent to mention that solution found by these methods may not be optimal. One of the greatest advantages of these methods is their capability of coping with combinatorial explosion and local minima [8].

Heuristic-based methods include following [4]: -

- Artificial Neural Network (ANN)
- Genetic Algorithm (GA)
- Particle Swarm Optimization (PSO)
- Ant Colony Optimization (ACO)
- Fuzzy Logic (FL)

1.3 Classification of Path Planning Methods Based on Scope

Basing on scope of path planning, these methods can be classified as global path planners and local path planners [3, 4]. The detail of these planners is given in succeeding paragraphs.

1.3.1 Global Path Planners

Global path planners are the planners which require complete information of environment before planning a path. Since global planners have knowledge of environment before planning a path therefore, path generated by them is often optimal. Global path planners are also efficient in finding path. These planners, however, have limitations as well. Since their planning is based on accurate information of strictly static environment therefore these planners can not handle any variation in environment, no matter how small it may be. For a real world environment, assumption of environment being static does not seem logical therefore these planners are really challenged in real world applications. Global path planners, despite all their strengths, are seriously devoid of robustness and flexibility. These path planners can be grouped and listed as follows: -

- Road Maps
    - Visibility Graph
    - Voronoi Diagram

- Cell Decomposition Methods
    - Exact Cell Decomposition Method
    - Approximate Cell Decomposition Method
    - Modified Exact Cell Decomposition Method
    - Probabilistic Cell Decomposition Method
- Probabilistic or Sampling-Based Methods
    - Probabilistic Road Maps (PRM)
    - Rapidly Exploring Random Trees (RRTs)

1.3.1.1 Road Maps

The road map methods are also called skeleton, highway or retraction methods. A road map is actually a collection of paths such that each path is a set of collision-free configurations between two points which can be used for path planning [9]. The roadmap once created can be used for multiple path planning assignments. The road map generated in a given environment can not only be used for multiple different tasks in the same environment but can also be used by multiple agents employed in the similar environment. Generating a road map is the main part of these methods. After generating a road map, start point and goal are connected to corresponding nearest nodes on the map and then a graph search algorithm is used to find path on road map.

The principles of generating a road map may differ from each other and so would be generated road map. Among different methods available for generating road maps, visibility graph and voronoi diagram are most known and used [4].

1.3.1.1.1 Visibility Graph

Visibility Graph is a collection of lines in free space such that each line connects a feature of an obstacle to another visible feature of the same or different obstacle or in other words these features are vertices of obstacles that can see each other. It is also important to note that start point and goal are also part of this graph. After development of road map i.e. visibility graph, a graph search technique, commonly A* algorithm is applied to find shortest path from start point to goal on this graph. This method was used for robot motion planning in [10].

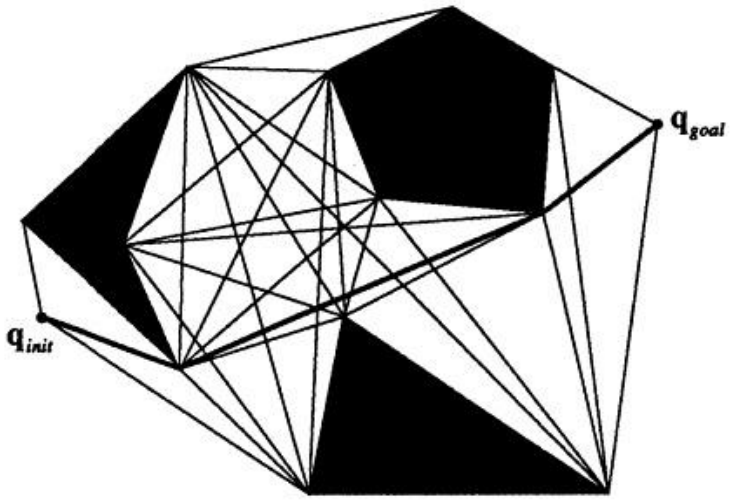

Figure 1.1: Visibility graph and path from start point to goal [11].

Since visibility graph is generated on the principle of visibility therefore path generated using this method is optimal in terms of distance so resulting in minimum length solution [11]. It is worth nothing that some of the lines in visibility graph are actually the edges of obstacles therefore moving on this optimal path generated by visibility graph is not a safe proposition for robot. While optimal path length is the main advantage of visibility graph, its complexity in densely cluttered environment and safety of robot are major concerns.

1.3.1.1.2 Voronoi Diagram

Voronoi diagram is another road map method used for robot motion planning [12, 13]. This road map is generated by including points which are at the similar distance from two or more obstacles or workspace boundary. In voronoi diagram, start point and goal are not part of graph as in case of visibility graph. The next step, after generating voronoi diagram, is to ascertain accessibility and deportability with regards to start point and goal. Accessibility means start point can access the road map generated by voronoi diagram and deportability is that a node on voronoi diagram can connect to goal. A suitable connection strategy is to be employed for connecting start point and goal to respective nodes on voronoi diagram. As a last step, graph search technique is to be employed for finding path from start point to goal.

It is important to note that as this method tends to maximize the distance from all the involved obstacles therefore the road map is safe for robot but may not be optimal in terms of distance involved [12].

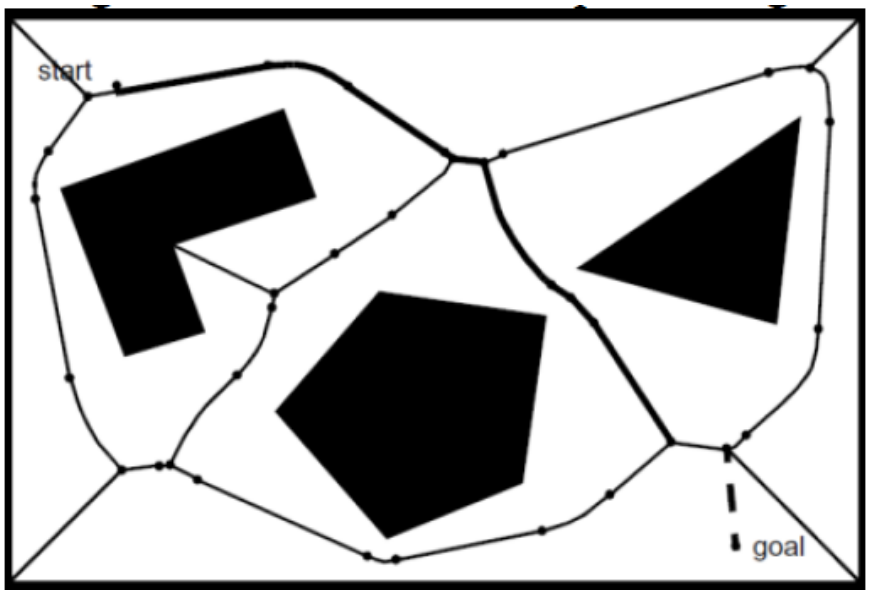

Figure 1.2: Voronoi Diagram [12]

1.3.1.2 Cell Decomposition Methods

Cell Decomposition (CD) methods are another well known and used class of path planning algorithms. With some variations, in all these methods space is decomposed in smaller cells so as they can be segregated as occupied i.e. obstacle cells and free cells. After division, a graph of free cells is generated based on their connections. A graph search method is then used to find series of mutually connected cells from start cell to goal cell. Start cells and goal cell are the free cells containing start point and goal and sequence of mutually connected cells in between are the cells robot can use from start point to reach goal [14].

Assuming that robot motion planning problem has been reduced to a point moving in free space, Seda [15] suggests following step to implement cell decomposition methods:-

- Divide free space into connected cells
- Generate a graph of adjacent cells such that nodes of graph are free cells and connecting edges indicate mutual boundary between cells.
- Identify start cell and goal cell containing start point and goal and search sequence of cells from start cell to goal in graph generated in previous step.
- Find path by connecting center points (centroid) of boundary of cells.

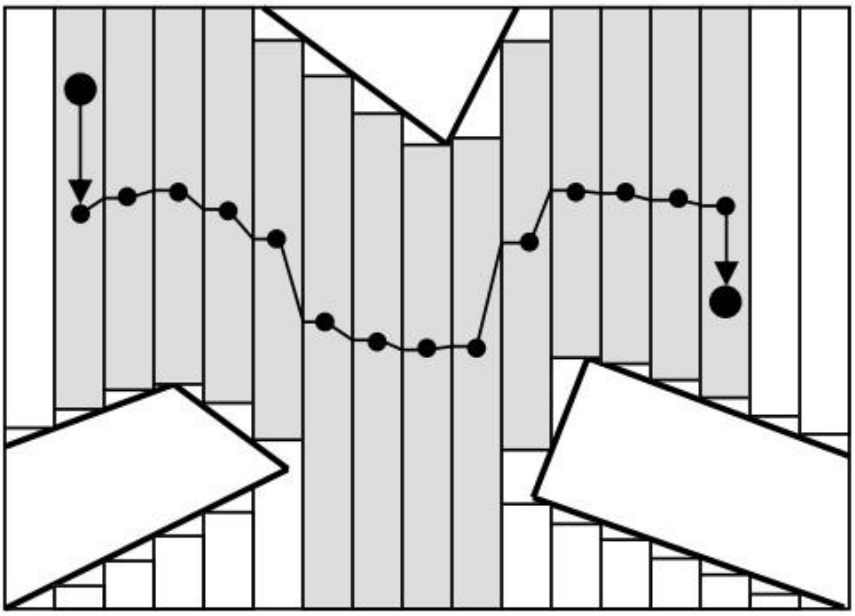

Figure1.3: Cell Decomposition [15]

Cell decomposition methods have been implemented in variety of ways. Three main sub categories of these methods have been identified as follows [4]: -

- Exact cell decomposition [16, 17, 18].
- Approximate Cell decomposition [19, 20, 21].
- Probabilistic cell decomposition [22, 23].

Exact cell decomposition decomposes the work space in such a way that union of all the decomposed cells is exactly equal to workspace. Moreover exact cell decomposition is based on the features of obstacles and workspace. Since exact cell decomposition is an effort to find an exact solution therefore it is time consuming and less efficient with respect to time. Modified exact cell decomposition is little different from exact CD in that it differs in as what point of boundary between adjacent cells would be used as crossing places [24].

In approximate cell decomposition workspace is so decomposed that union of decomposed cells does not necessarily makes entire workspace. In this method boundary of decomposed cells does not indicate any physical meaning related to workspace or obstacles. Since it is an approximation method therefore it is comparatively faster than exact cell decomposition. Moreover because of being an approximation its completeness depends upon the

resolution of approximation. The method, despite being fast and easy to implement is not very efficient for path planning [22].

Probabilistic CD (PCD) is actually a combination of CD and probabilistic methods. It is another method of cell decomposition that approximates workspace in a way not much different from approximate CD. In probabilistic CD methods, cells have a predefined shape and cell boundaries are not based on physical obstacles. The method is fast and easy to implement but in an environment where free space is much smaller as compared to occupied, the reliability of method is seriously challenged [22].

1.3.1.3 Probabilistic or Sampling Based Methods

The path planning methods which try to represent the workspace precisely and attempt to find an exact solution are exhaustive in computation and time consuming. These methods have manifested sufficiently satisfactory performance in problems involving limited dimensions but with the increase in dimensions of problem their complexity increase exponentially and they tend to suffer from combinatorial explosion. This is why sampling-based or probabilistic path planning approaches have been proposed which work well in high dimensional problems.

Sampling based methods do not try to represent workspace instead they generate random samples in entire space. Later these samples are checked using a collision detector whether they lie in free space or in / on obstacle. The samples which are not in free space are discarded and only sample from free space are used to generate a graph. The path planning problem is thus reduced to a graph search problem from start point to goal in generated graph. Though many variations of probabilistic methods exist most common are Probabilistic Road Map (PRM) and Rapidly Exploring Random Tree (RRT).

1.3.1.3.1 Probabilistic Road Map (PRM)

The probabilistic road map planner carries out sampling of the configuration space using an appropriate distribution, mostly uniform. From these samples, only those samples which are lying in free space as detected by collision detector are kept for further use and remaining are discarded. The samples lying in free space forms the nodes of graph. A local planner is used to connect each node of graph to a certain number of nearest neighboring nodes. For all those nodes which are connected to each other by local planner without any collision, an edge is added in the graph. Nodes of this graph are the land marks through which robot shall move and edges are the

roads which it would take to maneuver through these lands marks. It is very important to use such local planner which is very fast and simple without worrying about its failure. At most of the instances, the most suitable choice is straight line connector. The steps mentioned so far are part of construction phase of implementing PRM.

Next step in implementation of PRM is to use graph generated in construction phase to find path from start point to goal. This step is called query phase. Start point and goal are added to graph and connected to it using local planner. Graph search method is then used to search path in terms of sequence of nodes from start point to goal.

Variants of PRM have also been proposed by researchers. Lazy-PRM is a variant that delays detecting collision until very last moment of query phase [25]. Another variant is Obstacle-Based PRM that evaluates and suggest strategies for generating nodes in general and in difficult regions of configuration space in particular [26].

1.3.1.3.2 Rapidly Exploring Random Tree (RRT)

Rapidly Exploring Random Tree was first introduced in [27] as a data structure which was efficient and based on a sampling scheme. The algorithm is capable searching high dimensional space quickly and efficiently. The search space may have constraints as well. With reference to path planning these constraints could be algebraic which are because of obstacles and differential which are due to dynamics of the robot. The core idea behind this algorithm is to direct or bias the exploration in unsearched areas by sampling in those regions and incrementally pulling search tree towards goal.

1.3.2 Characteristics of Global Path Planner

After brief introduction of the global path planners it can be concluded that since method of planning of these global planners is mostly exhaustive and is based on complete information of the environment, therefore they manifest following strengths: -

- Global planners generate optimized path.
- These planners are efficient in finding path.

Besides the advantages mentioned above, global path planner have some serious disadvantages and these are: -

- Inability to plan or re-plan dynamically.
- These methods fail in uncertain environment.

- They are not capable of avoiding collisions with unknown obstacles.
- Performance is limited in presence of dynamic obstacles.
- These methods fail in case of minor variations in layout of obstacles.

1.3.3 Local Path Planners

Reactive or local path planners, on the other hand require no prior knowledge of environment while planning a path. They plan their path dynamically through obstacles. Since their path to goal is guided on-line by information provided by their sensors, this is why these are good at collision avoidance and finding their path through dynamic or unknown obstacles. The path generated by these local planners however is more often than not is not optimal nor are these planners graded as efficient. Their main quality and strength is their robustness and flexibility in that they can avoid unknown or dynamic obstacles and can handle variations in environment. Most used and known local path planners are: -

- Artificial potential field (APF)
- Vector Field Histogram
- Dynamic window approach

1.3.3.1 Artificial Potential Field

The method of virtually applying artificial potential fields to mobile robots for path planning was first introduced by Khatib [28]. This method suggests development of virtual potential fields by goal and obstacles present in environment. The field generated by goal is attractive whereas that of obstacles is repulsive. This is because of these forces that robot is guided to goal while avoiding obstacles in between its path as shown in figure 1.4.

The method has great reactive and online planning capability and this is why it has been extensively used for obstacle and collision-avoidance. It is however pertinent to mention that it also has few shortcomings. Falling trapped in local minima is its most critical weakness. It is because of this weakness that robot in certain situations gets trapped and can not reach its goal. The other issue related with this approach is problem while moving through narrow corridors where because of repulsive forces robot starts oscillating and at times lose equilibrium.

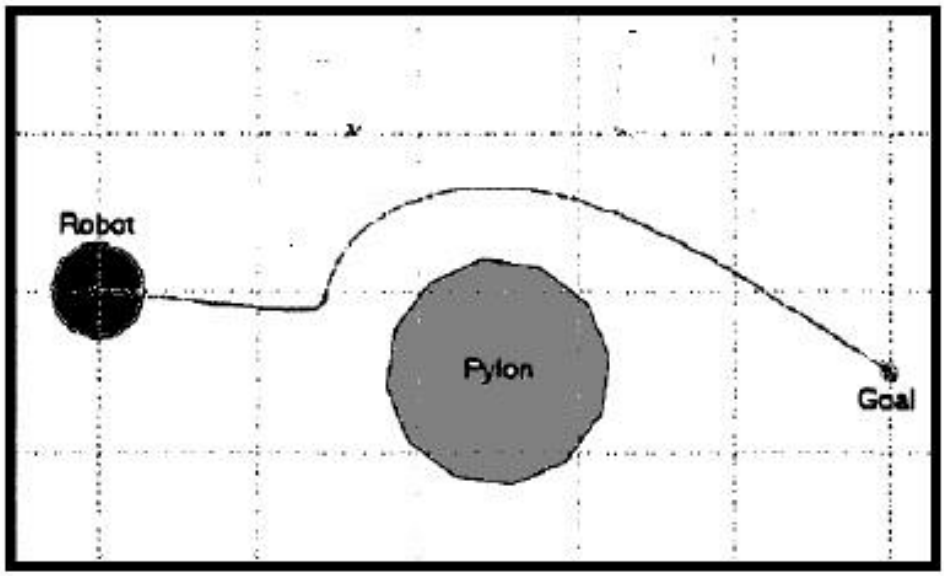

Figure 1.4: Artificial potential field

1.3.3.2 Vector Field Histogram (VFH)

Vector field histogram (VFH) is a method that has the capability of dynamically detecting unknown obstacles and avoiding collision while maintaining its movement towards goal [29]. VHF uses two-dimensional Cartesian histogram which is updated online by the data received from the sensors of robot. Data reduction technique is then applied to this two-dimensional histogram in two steps to eventually generate steering command signal to robot.

In first step of data reduction, 2D Cartesian histogram is reduced to 1D polar histogram. The value corresponding to each sector in polar histogram represents density of obstacle in that direction. Higher the value, more is the obstacle density and vice versa.

In second step of data reduction the each sector is evaluated and one the most suitable from the sectors having low polar density value is chosen and robot is directed to steer to chosen sector.

Three steps involved in implementation of VFH are listed as follows [29, 30]: -

- Develop a 2D Cartesian histogram to represent obstacles as in figure 1.5.
- Considering an active window around robot current location and reduce Cartesian histogram to polar histogram of suitable resolution as in figure 1.6
- By optimizing find velocity and steering angle.

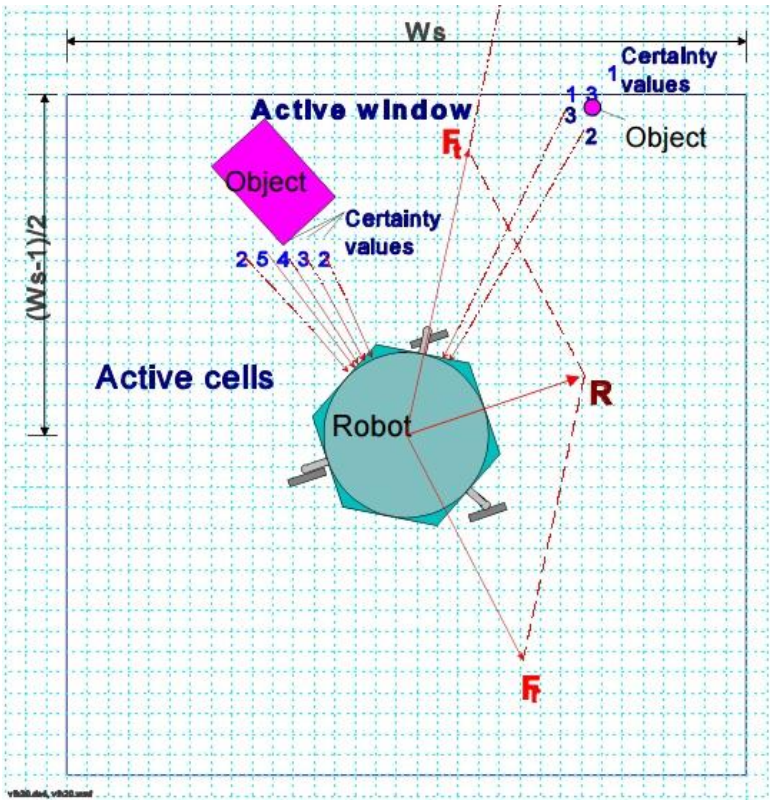

Figure 1.5: 2D Cartesian histogram [30]

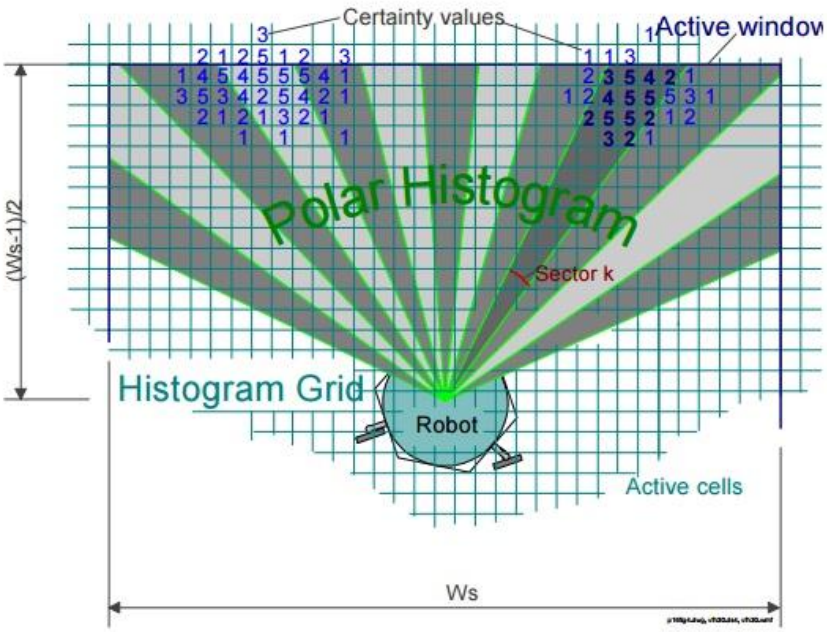

Figure 1.6: 1D Polar histogram [30]

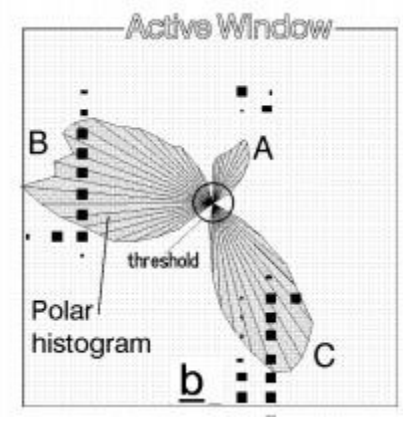

Figure 1.7: 1D Polar histogram as overlaying on 2D Cartesian histogram [30]

VFH is a great method to minimize inaccuracies and errors from sensors data as this data is averaged while forming 2D Cartesian histogram which is further reduced in polar histogram. It is one of the reasons that instability as experienced by APF while moving across a narrow corridor is eliminated in this approach. Moreover VFH does not suffer from issue of local minima however the greatest disadvantage of method is that it can lead the robot away from target location [30].

1.3.3.3 Dynamic Window Approach

Dynamic window approach presented in [31] differs from other local or reactive path planners mentioned in previous paragraphs. It, as against APF and VFH, takes into account the kinematic and dynamic constraints of the robot i.e. velocity motion model. This approach assumes movement of robot in the form of circular arcs and each of these arcs is given by a pair of linear and angular velocities i.e. (v, ω). Search space of velocity pairs is reduced by considering only those velocities which can be safely adopted. A velocity pair is safe if robot can stop before reaching the closest obstacle lying on arc corresponding to it. These velocities are called admissible velocities. An additional restriction is placed on admissible velocities by dynamic window and that is only those velocities which can be achieved in short interval of time considering acceleration of robot is reachable velocities.

An objective function is used to evaluate the velocities from dynamic window. The objective function for the purpose is takes into account following three terms with adaptive weights [31]: -

- Target heading which is an indication of movement towards goal.

- Clearance which is distance to closest obstacle on the arc corresponding to velocity pair being evaluated.
- Velocity of robot.

Dynamic window approach has manifested success by controlling a robot along a safe path at reasonably fast speed but it can face trap situations and does not guarantee to find a solution.

1.3.4 Characteristics of Local Path Planners

After having a brief introduction of different type of local path planners it is logical to mention that these planners are characterized by following strengths and advantages: -

- Capable of planning and re-planning dynamically on the move.
- Ability to detect obstacles and change direction in real-time to avoid collisions.
- Effective in dynamic environment.
- These methods are robust and can handle variations in environment.

While reactive or local path planners have many strengths and advantages particularly with regards to online planning and performance in dynamic environment, they have some weaknesses as well:-

- The path generated by these methods is not optimal.
- These methods are not efficient.
- These methods suffer from local minima so they are trapped in deal locks.
- Inaccurate sensors and their inherent errors affect the performance of these methods.

1.3.5 Motivation

With sufficient background developed in this chapter about global and local path planners, their strengths and weaknesses it sounds logical to consider developing a hybrid planer by integrating a global and local path planner. These planners can be integrated to exploit advantages of each one of integrated planner and mitigate their disadvantages.

1.3.6 Research Objectives

The research to develop a hybrid path planner by integrating global and local planner has following objectives:-

- To optimize advantages and strengths of global and local path planners.
- To mitigate shortcomings of integrated planners.
- To make integration and connectivity of planners seamless.
- To develop a hybrid planner capable of developing optimal path.
- To ensure safety of robot by avoiding collision with obstacles.
- To develop a planner capable of performing effectively and efficiently in static as well as dynamic environment.
- To develop simple and easy to implement planner.
- To develop a robust and flexible planner.

CHAPTER 2: LITERATURE REVIEW

Like a light house that guides ships to home, literature review guides researchers through the length and breadth of field to discover and develop new ideas and methods. It is for the same reason that a great deal of research papers have been reviewed in detail to accomplish this research which lead to idea of developing hybrid path planner by integrating a global and local planner. Initially the literature was reviewed to identify an area of research and finalize research title and later attention was focused on understanding and analyzing research related to hybrid path planners. Although the breadth of literature reviewed is quite large but only the most relevant literature is reviewed in succeeding paragraphs.

2.1 Hybrid of Distance Transform and Artificial Potential Field

In 2002, L.C. Wang et al. presented the idea of developing a hybrid path planner by integrating distance transform method as global and artificial potential field as local planner [32]. They justified the need of developing hybrid planner on the argument that global path planners are good and capable of generating optimal path in static environment but are not capable of performing in environment with unknown or dynamic obstacles. On the other hand local planners are capable of handling dynamic environments with unknown or dynamic obstacles but are greatly inefficient specially in difficult and cluttered environments therefore a hybrid path planner capable of handling both kind of environments and with strengths of both planners may be developed.

The research considered different planners both from global and local and finalized on using distance transform as global and artificial potential field as local planner. The distance transform method [33] was justified because it was easy to implement and unlike many other global planners it starts to find path from goal location therefore it has the flexibility of having any location in free space as start point. Besides above, another reason as mentioned in [32] was its potential to extend it for dynamic environments. Artificial potential field has been used as local planner because of its ease of implementation and integration without compromising on robustness of hybrid planner.

The hybrid planner proposed in research uses path generated by distance transform method as back bone and artificial potential field guides robot on this path through sub goals. Figure 2.1 indicates start point, goal and the path generated by distance transform.

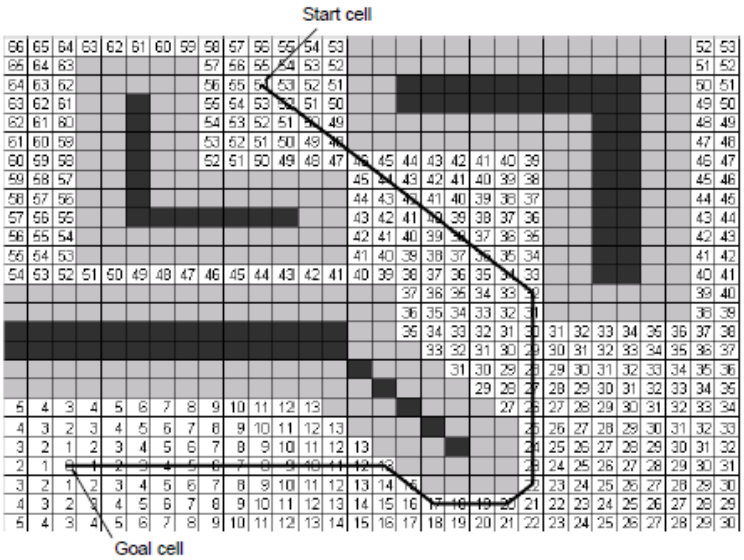

Figure 2.1: Path generated by distance transform method (global planner) [32]

Hybrid path planner selects sub goals for artificial potential field by drawing a circle at current position of robot and intersection point of circle and global path are taken as sub goals. Figure 2.2 indicates sub goal at start point and figure 2.3 shows sub goal at an intermediate point. It is important to note that in cases where circle intersects global path at two locations, the intersection with lower distance value is selected.

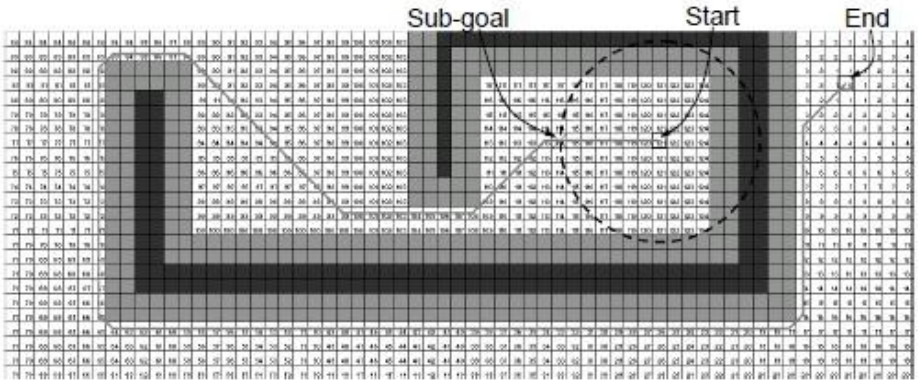

Figure 2.2: Selection of sub goal at start point [32]

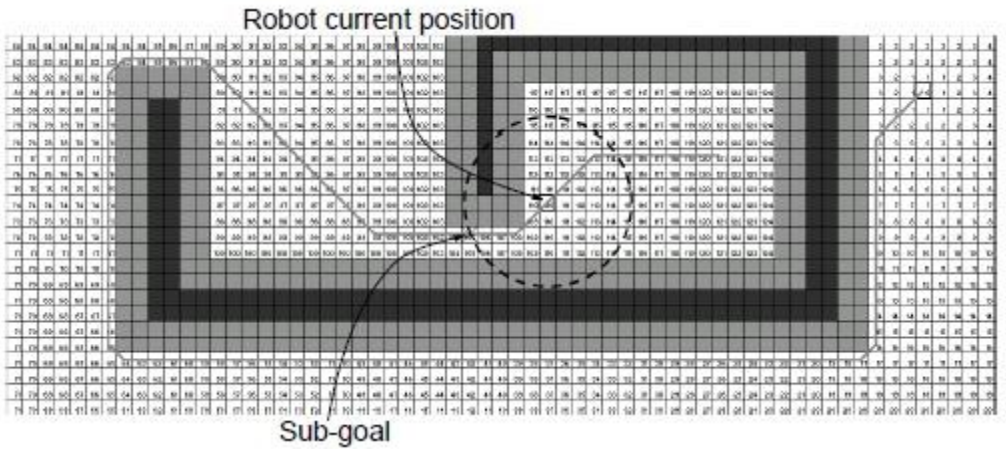

Figure 2.3: Selection of sub goal at an intermediate point [32]

The results of implementation as mentioned in [32] manifested that hybrid planner was able to avoid local minima which would have been otherwise encountered by artificial potential field in test environment. The robot was also able to navigate through narrow corridor using hybrid method however it is pertinent to mention that neither path generated by method is optimal nor any capability of hybrid planner with respect to dynamic or unknown obstacles have been manifested.

2.2 Hybrid of Ant Colony Optimization and Artificial Potential Field

In 2006, H. Mei et al. presented a hybrid ant colony optimization algorithm for path planning in dynamic environment [34]. The hybrid planner employs ant colony optimization as global and artificial potential field as local planner. The author discusses the reasons for the need of developing hybrid planner by mentioning the weaknesses and shortcomings of existing planners. Most of these reasons relate to inability of global planners to perform in dynamic environments and dead lock because of local minima for local planners as in case of artificial potential field.

The research recommends some improvements in ant colony optimization method while integrating it with local planners. The first of suggested improvements is about initialization of pheromone value at the start of problem. It recommends that instead of assigning same value to all cells, pheromone value should be assigned on the basis of distance from obstacle i.e. if the distance of a cell from obstacle is more it should have higher pheromone value and vice versa.

$$Pheromone \propto \sqrt{distance} \quad (2.1)$$

The second improvement is about modifying objective function by including obstacle-avoidance information and smoothness of path. The objective function is defined as follows and the path with smaller value is chosen as solution: -

$$Fk = \left(\frac{Lk}{L}\right) + W1 \sum_{i=1}^{m} \frac{1}{di} + W2 \times T \quad (2.2)$$

- L is shortest path length from start point to goal
- L_k is length of path of k_{th} ant
- T indicates number of the turns in respective path.
- *m* indicates number of obstacles present in environment.
- d_i is the distance between the i_{th} obstacle and path of k_{th} ant
- W terms are weights.

Moreover, the method also suggests change in the way pheromone value of cells is updated. It suggests that instead of updating all or the best solution, best one third of all solutions should be updated to ensure flexibility and faster convergence. The method proposed in [34] uses ant colony optimization method with suggested improvements to generate path from start point to goal.

Local path planner in this hybrid approach is called only when needed i.e. to avoid collision with a dynamic obstacle which is located on the path planned by global planner or likely to interfere with the robot on its pre-planned path.

The local planner which is artificial potential field in this case has also been modified. The first modification is about adding another attractive force other than that of goal. This force is because of pheromone concentration in the environment. After modification, forces acting on robot are given as follows:-

$$F\ net = F\ attractive + \ F\ repulsive\ + Fpheromone \quad (2.3)$$

Where $F_{pheromone}$ is proportional to concentrated pheromone and distance from closest obstacle and is given by

$$F\ pheromone \propto Pheromone\ \times Distance \quad (2.4)$$

The method also introduces use of taboo table to restrict robot from visiting already visited cells and thus it emphasizes that because of global information added into artificial potential function in the form of pheromone concentration and use of taboo table, the proposed method is not likely to be trapped into local minima.

The success of the method has been manifested by implementing it in a simple and a complex environment. In figure 2.4, the "×" represents the path generated by global planner and small " o " represent the path actually taken by robot after interference by local planner.

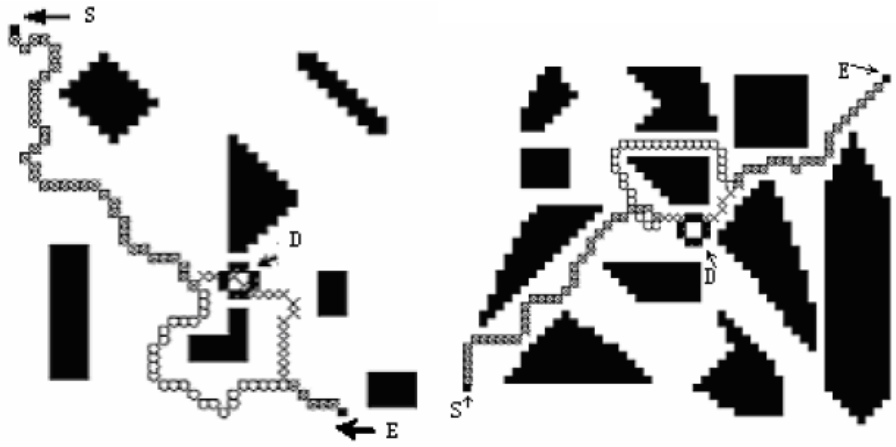

Figure 2.4: Hybrid planner implemented in simple environment (left) complex environment (right) [34]

On the basis of simulation results, it is claimed that proposed hybrid method can find a near optimal solution while avoiding collision with obstacles present in environment however as seen in case of simple environment, a more optimal path is still possible.

2.3 Hybrid of Voronoi diagram and D* Algorithm

In 2012, Ming-Chih Lu et al. proposed hybrid path planning incorporating global and local search for mobile robot [35]. The method proposed to use voronoi diagram as global planner to generate a global path which was to later serve as backbone path for the map. D* algorithm is used to find shortest path from start point to goal while using path generated by voronoi diagram.

In this hybrid scheme, *n* numbers of nodes are selected adjacent to both start point and goal. From these adjacent nodes, two optimal nodes corresponding to start point and goal are chosen. Optimality of nodes is ascertained by an evaluation function that minimizes the distance from start point to goal through nodes being evaluated. Later, D* algorithm is used to find shortest path from optimal node corresponding to start point and to that of goal and subsequently A* algorithm is used to find shortest path to connect these corresponding nodes to start point and goal respectively.

The proposed algorithm has been implemented in an environment along with D* algorithm and generalized voronoi diagram for comparison. The proposed algorithm is time efficient and maintains maximum safety distance from obstacles as shown in figure 2.5.

Figure 2.5: D* algorithm (left) voronoi diagram (center) and Hybrid (right) [35]

2.4 Hybrid of Artificial Potential Field and Simulated Annealing

In 2006, Qidan Zhu et al. proposed artificial potential field approach with simulated annealing [36]. The research is about getting rid of the greatest shortcoming of artificial potential field i.e. trapping in local minima, by integrating it with simulated annealing.

The research identifies local minima and other problem associated with artificial potential field as follows: -

- Problem of passing through narrow corridors in between closely located obstacles.
- Problems in maintaining equilibrium in narrow spaces and resultant oscillations and closed loop movements.
- Problems in reaching goal located near obstacles.

The research proposes to compliment artificial potential field with simulated annealing method to escape from local minima traps. Simulated annealing is applied to rescue robot only when it has been trapped. The research lists following steps of simulated annealing algorithm for local path planning: -

- Set x = S (where x is trap location and is set as start point for simulated annealing method).
- Choose parameters for annealing; Set T_0 sufficiently high and set $T = T_0$.
- While $T \geq T_f$ (minimum temperature allowed for cooling) and still trapped, do:
    - Choose a random neighbor $x' = x + \Delta x$ (Δx is uniformly distributed random distribution).
    - Calculate U(x'), the potential at x'.
    - Set $\Delta U = U(x') - U(x)$.
    - If $\Delta U \leq 0$, set x = x' else set x = x' with probability $P = e^{-\Delta / kT}$ where k is Boltzmann constant.
    - If $U(x') \leq U(x)$, local minima has been escaped
- If not escaped, then return failure, else escape.

The method has been implemented in 2D static environment with convex polygonal obstacles and the results show that in circumstance where artificial potential field got trapped in local minima simulated annealing successfully helped robot to escape from it. It is however, important to note that method has not been manifested in the environment having unknown or dynamic obstacles. Figure 2.6 shows an environment in which robot got trapped in local minima while using artificial potential field whereas for the same environment and same start point and

goal, robot successfully reached goal while employing simulated annealing. The same kind of situation is shown in figure 2.7 but in a different environment.

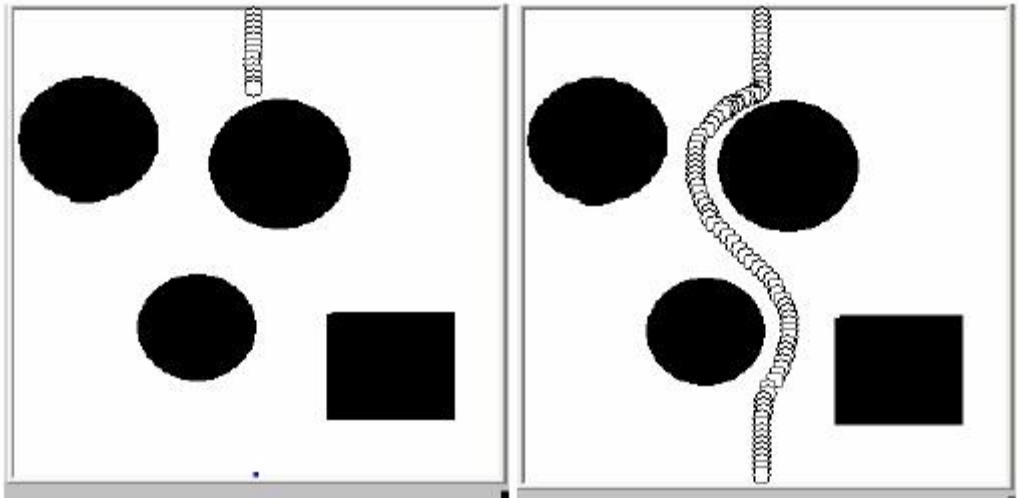

Figure 2.6: APF trapped in local minima (left) Simulated annealing rescued (right) [36]

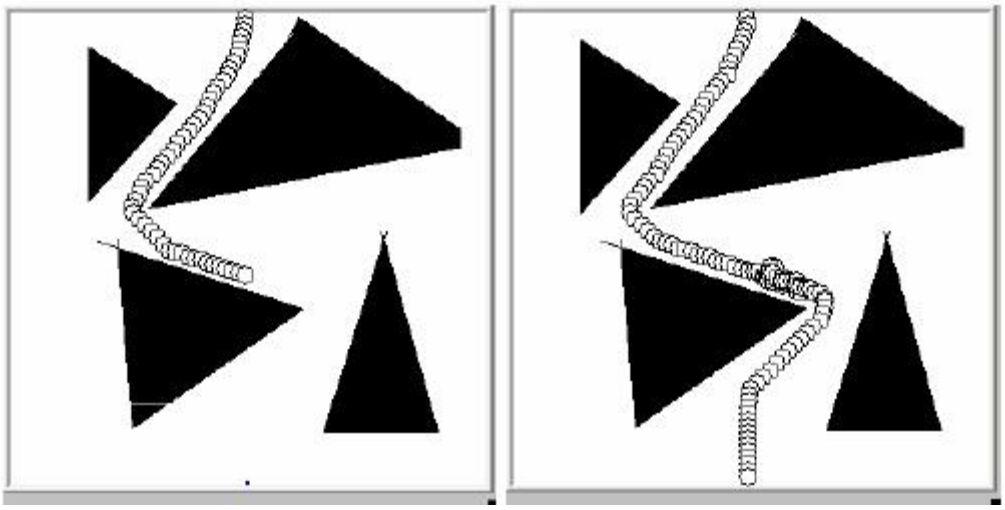

Figure 2.7: APF trapped in local minima (left) Simulated annealing rescued (right) [36]

2.5 Hybrid of Steering Control and Improved Distance Propagating

In 2012, Yan, Zh et al. proposed a hybrid path planner based on steering control and improved distance propagating [37]. The method is an attempt to integrate obstacle avoidance capability with optimization capability of local and global planners respectively. The method employs a variant of distance propagating method [38]. It uses grid points instead of grid for the

reasons [39]. Among others, the greatest advantage of using grid points in this method is the ease of implementation and integration of two planners.

The proposed improved distance propagating method integrates obstacle information as against [39] which does not account for obstacles while planning path. The difference of results is manifested in implementation as shown in figure 2.8 where it can be noticed that by accounting for obstacle information, the path is kept at a distance from obstacles instead of going very close to them.

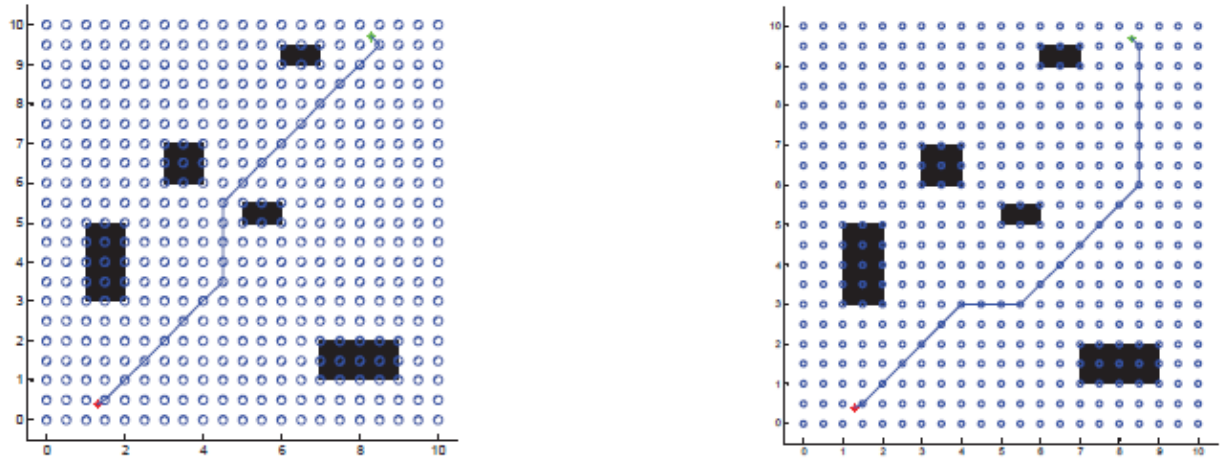

Figure 2.8: Distance propagating (left) Improved distance propagating (right) [37]

A steering control approach is proposed in the research for local planning in unknown cluttered environment. The method processes the range scanning data in three steps as follows: -

- In first step, range histogram is created from sensor data.
- In second step, histogram is converted into binary histogram based on suitable safety distance required to be maintained from obstacles.
- In third step, obstacle free areas are found as candidate steering directions.

The set of candidate directions is then evaluated as in [29] however, a modified evaluation function is proposed in this research for evaluation. After selection of steering angle, a suitable angular velocity is selected on the basis of rules proposed in this research. Besides above mentioned controls, an additional control strategy has also been proposed in the research which takes into account the width of robot as well and it ensures that robot maneuvers through obstacles safely.

The proposed method has been implemented in three different scenarios as shown in figure 2.9, 2.10 and 2.11. In figure 2.9, environment is simple and known priori and the proposed method finds a near optimal path. In figures 2.10 and 2.11 the environment is little complex but there is a difference in both these environments as in former case there is no unknown obstacle whereas in later case few unknown obstacles are placed on path of robot. The results show that proposed method safely guides robot to goal in both cases. It is, however, pertinent to mention that no dynamic obstacles are present in any scenario.

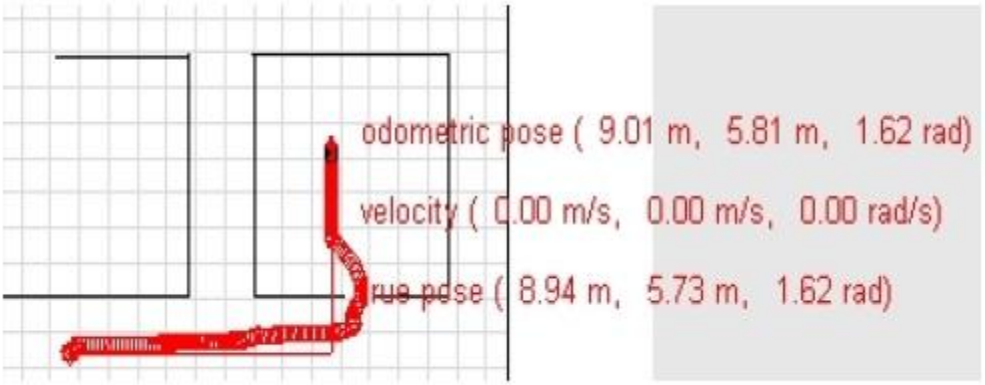

Figure 2.9: Simple environment [37]

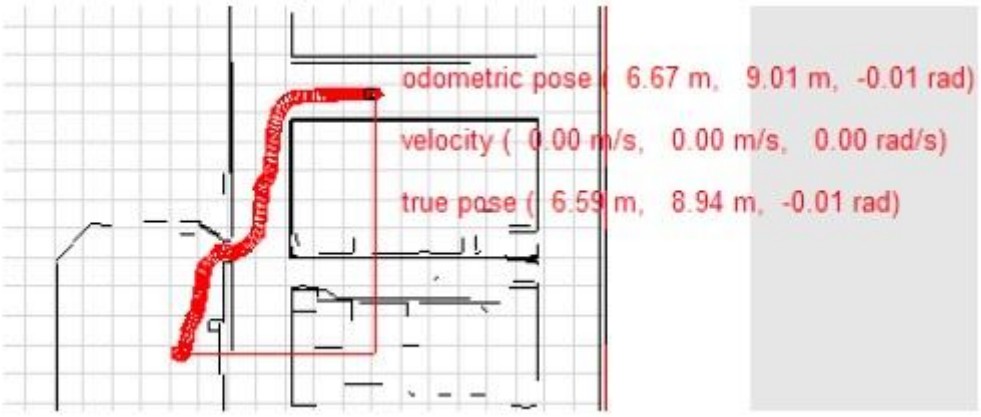

Figure 2.10: Complex environment without any unknown obstacle [37]

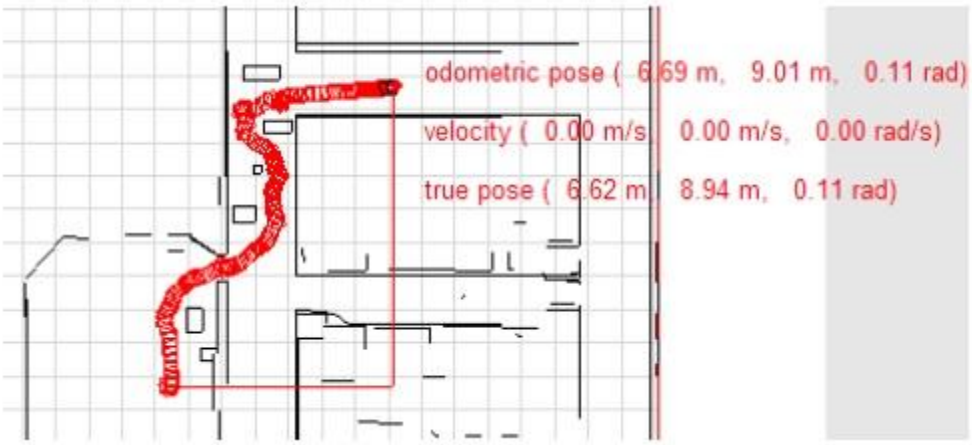

Figure 2.11: Complex environment with unknown obstacles [37]

2.6 Summary

From the literature review presented in this chapter, it is evident that many attempts have been successfully made to develop hybrid path planners by integrating different existing planners. While each of the proposed hybrid planner has been successful in improving some aspects of path planning or resolving some issues, there is still room for further research in this area particularly with the focus to improve upon optimality of path and robustness vis-à-vis unknown and dynamic obstacles.

CHAPTER 3: SELECTION OF GLOBAL AND LOCAL PLANNER

While developing a hybrid path planner, it is of prime importance to select suitable path planners from available global and local path planners. The selection is mainly dictated by the research objectives already narrated in Chapter 1. Among various considerations while choosing a planner, the most important is completeness. A planner must be able to find a solution if there exists any or it should return failure in reasonably limited time and it is particularly important for global planner. The application and the environment in which a planner has to be employed is also a major deciding factor. A path planner may be exceptional in a static environment by all measures but could disastrously fail in a dynamic environment. A static and deterministic environment could merely be a dream in real life. In real world an environment, at best, could be predictable that too with limited degree of certainty, therefore a planner must be robust enough to perform effectively in real world. Besides considerations mentioned above, there are many other measures to gauge the performance and suitability of path planners.

3.1 Performance Measures for Path Planners

There are many different measures and bench marks for gauging the performance of a path planner. Some of the important measures are discussed in succeeding paragraphs [40].

3.1.1 Path Length

Path length is one of the most important parameter to gauge the performance of a planner as so many other parameters are directly proportional to path length. Ideally length of path from start point to goal, found by a planner should be shortest possible however due to different limitations, it is not always possible for a planner to find a shortest path so shorter is the better.

3.1.2 Computation Time

Total time taken by an algorithm in finding a path is its computation time and a serious constraint. The constraint really gets stringent for planners to be used for on-line path planning or for those being employed in environment with dynamic obstacles. Computation time of an algorithm depends upon: -

- Complexity of algorithm.
- Number of calls to math library.
- Capability of hardware.
- Complexity of environment.
- Type of environment i.e. static or dynamic.
- No of obstacles.
- Type of obstacles.
- Density of obstacles.
- Velocity of dynamic obstacles.

3.1.3 Computation Time per Unit Travelled

Computation time per meter travelled is a measure that weighs and balances path length with computation time taken to find that path. The advantage of finding shorter path is set void by this measure if time taken in finding this path is more.

3.1.4 Orientation Changes

The number of times robot has to change its orientation on its way from start point to goal while following a path should be as low as possible. This measure relates to quality of path. Time taken in traversing a given path by a robot is also affected by number of changes in orientation as each change results in deceleration. A longer path requiring fewer turns may be traversed in shorter time as compared to a shorter path with more turns. The changes in orientation dictated by hardware are not considered in this measure [40].

3.1.5 Robustness

Robustness of an algorithm is its capability to perform effectively and efficiently in the presence of undesirable errors and unforeseen and unpredictable changes in environment. No matter how good an algorithm may be by any measure, it would not be able to perform effectively in ever changing and unpredictable real world environment if it lacks robustness.

3.1.6 Memory Requirement

The memory required by an algorithm to run should be as low as possible. Like other hardware requirements, memory is also eventually translated into cost and load, which is always a concern and must be kept as low as possible.

3.1.7 Simplicity

Simplicity is a measure of ease of implementation. Ease of implementation includes ease of architecting the software part and writing instruction to implement it.

3.2 Additional Considerations for Selection of Planners for Integration

Basing on parameters listed in previous paragraphs, any algorithm can graded and best one can be selected for implementation in a given environment for a particular application. The selection of planners from available global and local planners for integration however, demands subtle and additional considerations. The integration of these planners should exploit and maximize advantages and merits of each one of them and mitigate their shortcomings. These additional considerations are given in succeeding paragraphs.

3.2.1 Need for Integration

The most important factor to consider before selecting planners is answer to question that why an integration of planners is needed? This is basically about exactly knowing and understanding the need. It is also about precisely comprehending the environment for which a hybrid planner is being designed by integrating existing planners. Development of a hybrid planner is like finding a customize solution by balancing various parameters and therefore due attention is required to be given to following: -

- Understanding of environment including static and dynamic obstacles and uncertainties.
- Desired characteristics of required path i.e. optimal, safe, involving least number of orientation changes etc.
- Desired characteristics of hybrid planner i.e. complete, time efficient, hardware requirement etc.
- Acceptable compromises.

3.2.2 Ease of Integration

While considering two different planners for integration the most important factor after it has been established that they serve the requirement is the ease of their integration. Ease of integration is highly desirable and other factors listed in succeeding paragraphs are also dictated by it therefore an effort should be made two select planners which can be readily integrated without any additional computation or hardware burden.

3.2.3 Seamless Connectivity

The integrated planners should be so seamlessly connected that at no point during implementation and running of resultant hybrid planner, switching from one to other or vice versa is noticed. The path generated by hybrid planner should be continuous and should be continuously updated without any interruptions in case of on-line path planning.

3.2.4 Additional Parameters and Computational Requirement

The ease of integration is eventually translated in this requirement. If two planners can be readily integrated then definition of additional parameters and computation shall not be required.

3.2.5 Additional Hardware Requirements

Requirement of additional hardware should be considered and must be weighed vis-à-vis effectiveness and performance hybrid planner.

3.2.6 Synergic Effect

Synergic effect is an effect arising or resulting from combination of two or more planners and is greater than the sum their individual effects. Hybrid planner developed by integrating two different existing planners, therefore, should be superior to each of integrated planners by performance measure listed in the beginning of this chapter. Apart from satisfying performance measures, it must be able to perform to desired standard for the application it has been developed.

3.3 Selection of Global and Local Planner

Basing on background developed regarding global and local path planners and their characteristics in introductory chapter of this thesis, research objectives and factors considered

earlier in this chapter, it is logical to conclude that a global path planner which is complete and generates optimal path efficiently can be integrated with a local path planner which has the capability to perform in dynamic environment. Optimality of path and capability to perform in dynamic environments are core research objectives. Basing on these requirements and factors discussed earlier in this chapter it is logical to consider visibility graph and artificial potential field from global and local path planners respectively for integration. Visibility graph is known for its optimal path and artificial potential field has established dynamic performance and on-line planning capability.

3.3.1 Visibility Graph

After reasons discussed in previous paragraphs, it is logical to use visibility graph among global path planners when distance involved from start point to goal is major concern. Visibility graph seems to be best choice for two reasons. Firstly; it is complete and ends up finding a path if there exists any secondly; it provides an optimal solution in terms of distance traversed by the robot to reach its goal [41]. Visibility graph is implemented in configuration space that is work space obstacles are swollen by the size of robot and boundary of work space is squeezed in. After these changes in work space, robot is assumed to reduce to a point in configuration space [41]. After transformation of work space into configuration space, visibility graph is generated. Visibility graph lists all vertices visible to each other. While generating visibility graph, each node or vertex of obstacle is considered one by one so as to evaluate what all other vertices of configuration space are visible to it.

A* algorithm is, then used to find shortest path efficiently from start point to goal through the nodes of visibility graph. A* algorithm is the best choice because of following two reasons [42]: -

- A* is admissible, that is it finds the shortest path from start point to goal.
- A* is minimal, that is it searches minimum number of nodes to find solution.

Inputs to A* algorithm are start node, goal node and visibility graph, as already found in previous step. A* algorithm being based on heuristic, requires a heuristic to be defined for selecting which node to move to from a given home node. The heuristic should be so defined that

in our problem of finding shortest path to goal, distance traveled from start point to goal is minimized.

In this research it is assumed that basic step of transforming work space into configuration space and reducing robot to a point has already been performed [32]. Figure 3.1 shows a configuration space of 300 x 260 with its lower left edge on the origin. It has three disjointed concave polygonal obstacles tagged as A, B and C. The vertices of these obstacles are numbered from lower left corner to its adjacent vertex in counter clockwise direction. The obstacles A, B and C have 1-24, 25-30 and 31-36 vertices respectively. Some of these vertices are also indicated for ease of reference. Figure 3.1 also indicates start point (105, 235) as red dot and goal (25, 35) as green. This configuration space features concave obstacles to highlight the problem of local minima which shall be emphasized later part of this thesis.

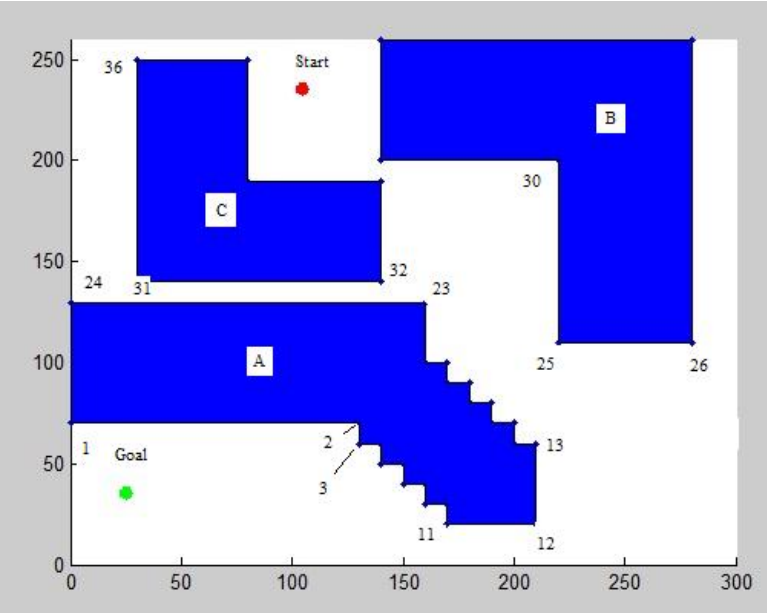

Figure 3.1: Configuration space

After having configuration space available, visibility graph can be generated using plane sweep algorithm [10]. A* search algorithm is used to find the shortest path from start point to goal through nodes of visibility Graph. Figure 3.2 shows the path in green from start point to goal searched from visibility graph using A * algorithm. The path starts from start point, moves through node 33, 13, 12, 11 and finally reaches goal. It is worth noting that path is not only the shortest possible in given configuration space but also unique, as well. As far as quality of path is concerned it is well established that visibility graph always provides shortest path. It is however,

worth noting that while traversing from node 13 to node 11, robot moves along / on the edges of obstacles. The question is how safe or unsafe this proposition is from the perspective of robot safety? In many applications, the main strength of visibility graph, i.e. optimality of path is abandoned because of safety concerns. From this point onward, begins the main essence of this research and that is to propose a simple yet effective solution which not only ensures safety of robot while moving along edges of obstacles but also does not compromise on optimality of path beyond a certain degree.

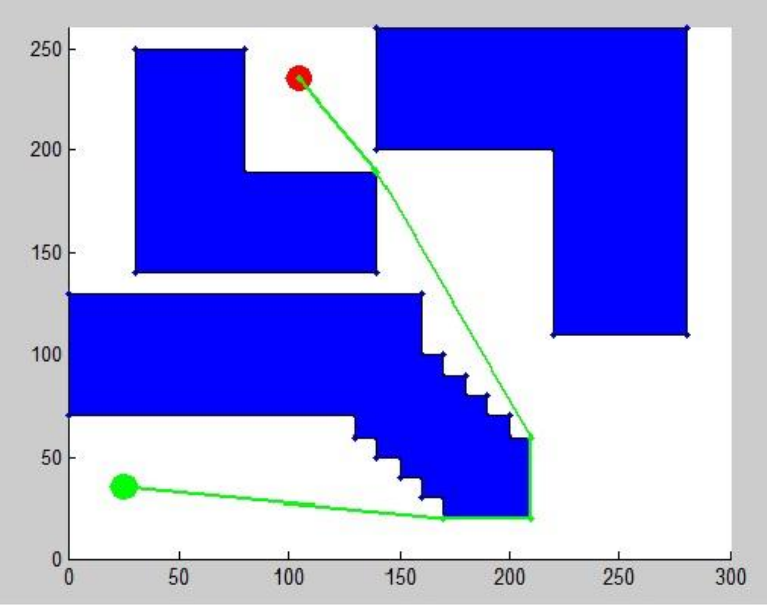

Figure 3.2: Path generated by visibility graph

The solution being proposed in this thesis is based on the idea of complimenting visibility graph with a local path planner that guides the robot from node to node on path generated by visibility graph. The hybrid technique, developed by integrating visibility graph with a local planner is aimed at finding solution to safety concerns. The fundamental idea that could be used to ensure safety of robot along obstacle edges is to repel it from obstacles just as much as required to ensure safety without compromising much on its optimality. The solution lies in using a simplified potential function.

3.3.2 Artificial Potential Field

Artificial potential field is well established solution for path planning problems requiring reactive capability. It is however, another established fact that artificial potential field suffers

from problem of local minima. The method, because of using gradient decent, ends up at critical points which may not be goal. Robots using these methods, while navigating through obstacles, get stuck at local minima and can not reach goal. To overcome local minima problem, methods have also been proposed which may help potential field planner to escape from local minima however hybrid method proposed in this thesis not only overcomes local minima problem associated with artificial potential field but also make generates a near optimal path.

The potential function being used here is rather over simplified. It is, like many other potential functions, a combination of attractive and repulsive potentials as give in equation 3.1.

$$F_{net} = F_{attractive} + F_{repulsive} \quad (3.1)$$

3.3.2.1 Attractive Potential

The attractive potential is quite simple. If robot knows its current position and also knows its target location, i.e. location of goal, it can be attracted towards goal with gradient potential

$$\nabla U_{attractive} = \zeta (q - q_{goal}) / d(q, q_{goal}) \quad (3.2)$$

The terms in equation 3.2 are explained as follows:-

- $\nabla U_{attractive}$ is attractive gradient.
- ζ is scaling factor.
- q is current configuration of robot.
- q_{goal} is goal configuration.
- $d(q, q_{goal})$ is Euclidian distance between robot and goal configuration.
- The term, $\zeta / d(q, q_{goal})$ in equation 3.2 serves as a scaling factor. It scales $\nabla U_{attractive}$ down when robot is far away from goal, and when it is closer, gradient is scaled up thus ensuring constant velocity of robot throughout the course.

The attractive gradient in equation 3.2 is directed in opposite direction of goal, so in order to move towards the goal the robot has to move in direction opposite to this gradient.

3.3.2.2 Repulsive Potential

The repulsive function has been simplified for the purpose of integration. Instead of considering cumulative effect of repulsive forces of all obstacles in configuration space, it just takes into account the effect of nearest obstacle. The repulsive gradient is given by

$$\nabla U_{repulsive} = \eta\,(q - c) / d\,(q,c), \quad for\ d(q,c) < d^{*} \tag{3.3}$$

The terms in equation 3.3 are explained as follows:-

- $\nabla U_{repulsive}$ is repulsive gradient.
- η is repulsive gain.
- q is current configuration of robot.
- c is the closest point on the edge of nearest obstacle.
- $d\,(q,c)$ is Euclidian distance from robot to point c
- d^{*} is minimum distance of robot from obstacle for repulsive forces to take effect.

The repulsive gradient in equation (3.3) points away from the nearest point c so by moving along this direction, the robot moves away from obstacle.

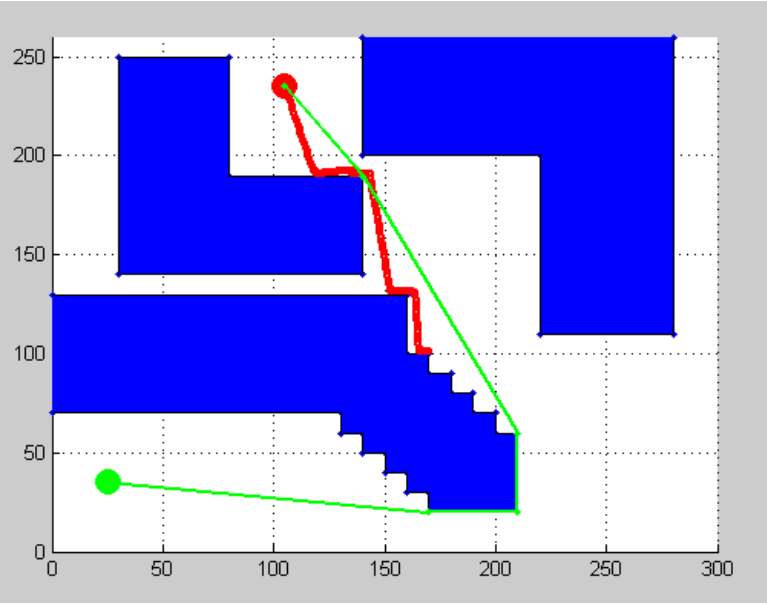

Figure 3.3: Path generated by proposed artificial potential field alone

In figure 3.3, a point robot is tasked to move from start point (105, 235) shown in red to node 11 (170, 20) using proposed potential function. The robot is however, stuck at local minima at point (170, 102.98) and fails to reach goal. Figure 3.3 emphasizes upon two issues related with potential function. Firstly it fails to find solution because of local minima secondly; if, at all, it succeeds in reaching goal, the path is not optimal in terms of distance. It is however worth noticing that artificial potential field always keeps robot away from obstacles and at no point throughout the course of maneuvering through obstacles, it touches any obstacle or comes closer than a defined distance. The best part is about artificial potential field is that safety distance is a user defined parameters and can be set to suit the safety requirements and applications.

Apart from providing flexibility to user for selecting and defining safety distance, potential function has many other flexible parameters which can be selected and tuned to suit varying requirements of applications and environment. There are options available for attractive potential function. One can choose from quadratic or conical potential function to be used for generating attractive potential filed. Similarly there is a choice in deciding whether to consider repulsive forces of all the obstacles or of only those in a given range. Another choice could be to consider repulsive force of only nearest obstacle or just nearest obstacle in one scenario or multiple obstacles in another scenario. Reactive capabilities of algorithm are already well established. Artificial potential function is easy to implement and less demanding for computational hardware. Path generated by artificial potential field is also smoother than most of other algorithms. The only disadvantage of the algorithm in our present context of developing a hybrid path planner is its inability to deliver in certain situations because of local minima for which a solution shall be proposed in succeeding chapter.

3.4 Summary

From the discussion in this chapter, it is evident that visibility graph and artificial potential field are the most suitable choices from global and local planners respectively for the purpose of integration. It is therefore, a hybrid path planner integrating these two is proposed in succeeding chapter.

CHAPTER 4: HYBRID PATH PLANNER – VISIBILTY GRAPH AND ARTIFICIAL POTENTIAL FIELD

The path planning methods global as well as local have their strengths as well as weaknesses. The hybrid of these techniques however, optimizes their strengths and mitigates their weaknesses. This hybrid technique is integration of visibility graph and potential function described in previous chapter. Shortest path generated by visibility graph serves as a backbone to this technique and potential function compliments it by keeping robot sufficiently away from edges of obstacles to ensure its safety. Since robot is moving from node to node on path generated by visibility graph, the path despite use of potential function is still near to optimal. The problem of local minima is also mitigated to single identified situation. The solution to this situation is also proposed in succeeding paragraphs.

4.1 Architecture

The hybrid path planner being proposed here is basically composed of a global and local path planner. In this thesis visibility graph is chosen to be used as global path planner and artificial potential field as reactive / local planner.

Hybrid path planner requires two inputs from environment. First input is a global map composed of static and immoveable obstacles. It is important to note that known dynamic obstacles are not to be included in this map as these would seriously affect the optimality of path and efficiency of planner. The second input from environment is sensing of local environment of robot through it laser sensors. Global map of environment is provided to global component of hybrid path planner i.e. visibility graph and local sensor data is provided to local path planning component of hybrid planner i.e. artificial potential field.

Basing on global map, visibility graph by employing A* search, efficiently generates an optimal path for robot in current environment from start point to goal. This optimal path is, then, fed to local path planner which uses this path as reference for guiding robot in steps from node to node to eventually reach its goal. The architecture of proposed hybrid planner is shown in figure 4.1.

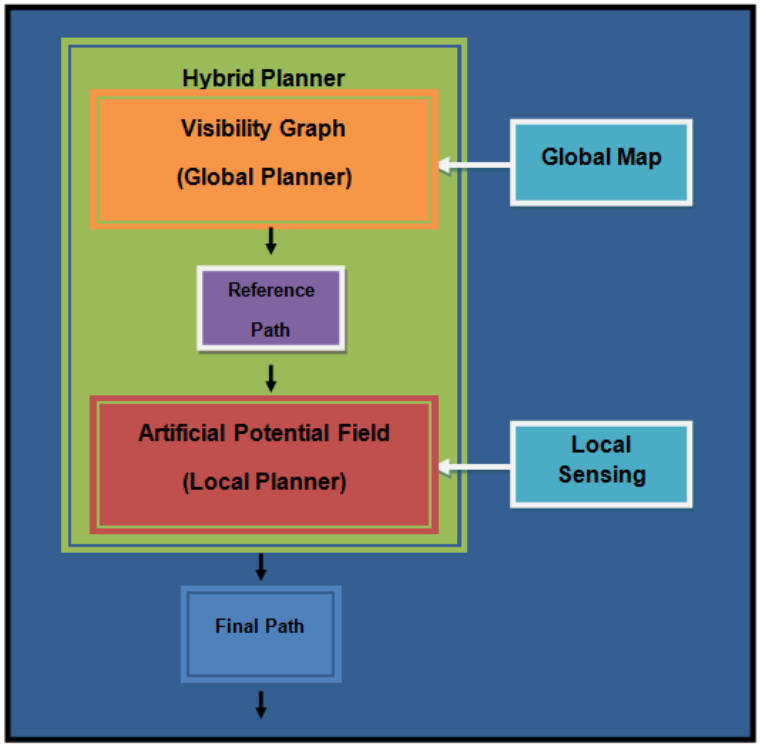

Figure 4.1: Architecture of proposed hybrid planner

Figure 4.1 also shows that artificial potential field being local component of hybrid path planner is provided with two inputs. First input is one time static input which is reference path generated by visibility graph whereas second input is continuously changing local map around robot which is updated simultaneously with the change in robot configuration and changes in environment. Both of these inputs are very important for the ultimate performance of hybrid path planner. Utilization of reference map by artificial potential field ensures that while navigating through dynamic obstacles and dynamically re-planning to avoid obstacles optimality of path is not compromised beyond minimum essential. Local sensing is the core input for artificial potential field as it provides reactive capabilities to this local planner. Therefore basing on these inputs local planner generates a near optimal path while avoiding collision with unknown and dynamic obstacles. The local sensing and reactive capability of local planner also make hybrid planner robust enough to negotiate and handle any disturbances or variation in layout of immoveable obstacles.

4.2 Methodology and Description

The path generated by visibility graph and A * algorithm is a sequence of nodes / vertices starting from start point, traversing through *n* nodes / vertices of obstacles and terminating at global goal. These nodes of path are passed to potential function as start points and sub-goals one by one. On reaching first sub goal, potential function takes it as start point for next step and next node in path is fed into potential function as new sub-goal.

Let $Path_{\text{visibility graph}} = [SP, N_i, N_j, N_k, \ldots N_n, Goal]$ where

- *SP* is global start point.
- N_i is first node after *SP*.
- N_j, N_k, ... is sequence of nodes in between.
- N_n is last node before global goal.
- *Goal* is global goal.

In current case, *SP* and N_i are start point and sub-goal respectively for first step of potential function. For second step, N_i becomes start point and N_j becomes sub-goal. This process continues until robot reaches global goal. For a path involving *n* number of nodes in between *SP* and *Goal*, *(n+1)* steps, local start point and sub-goal for each step is listed in table 4-1.

Table 4-1: Local start point (LSP) and sub-goals for steps of local planner (APF)

Step	*LSP*	*Sub-goal*
1	*SP*	N_i
2	N_i	N_j
3	N_j	N_k
..	..	..
...	...	...
n + 1	N_n	*Goal*

4.2.1 Use of Tolerance - ε

It is important to note that since nodes of path are actually vertices of obstacles we do not want our robot to touch; therefore we define a region of tolerance ε as shown in figure 4.2. The local planner assumes that robot has reached its goal when it reaches in region of tolerance. Introduction of tolerance in reaching goal is also necessary for the purpose of convergence.

Figure 4.2: Use of tolerance - ε

4.2.2 Local Start Point for Subsequent Steps

In proposed method, potential function terminates when robot reaches within defined region of tolerance to sub-goal assuming that it has reached its goal therefore the last configuration of robot at which potential function terminates, is used as start point for next step as indicated in figure 4.3. It is important not to use nodes as start points to ensure that robot does not touch any node at any time to ensure its safety. The use of last configuration of robot as start point is also necessitated by the need of continuity in path.

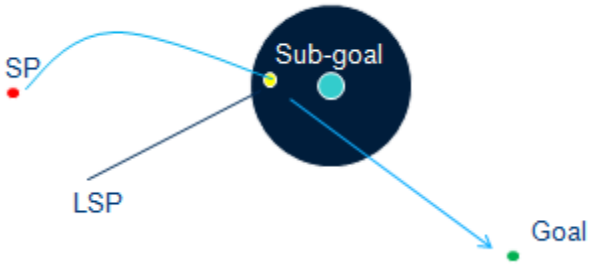

Figure 4.3: Last configuration of robot as local start point for subsequent steps of APF

4.2.3 Addressing Local Minima

Local minima problem, deeply associated with artificial potential field employed in environment with concave obstacles, is not likely to arise in proposed hybrid technique except for special situations given below: -

$$x_s - x_g = 0 \quad (4.1)$$

$$y_s - y_g = 0 \quad (4.2)$$

Let local start point be *LSP* (x_s, y_s), sub-goal be *SG* (x_g, y_g) and last configuration of robot at which potential function terminated in previous step be *q* (x_q, y_q) then if any of the conditions given in equation 4.1 or 4.2 is satisfied, robot is stuck at local minima and can not reach its *SG*. Local minima conditions are also indicated in figure 4.4.

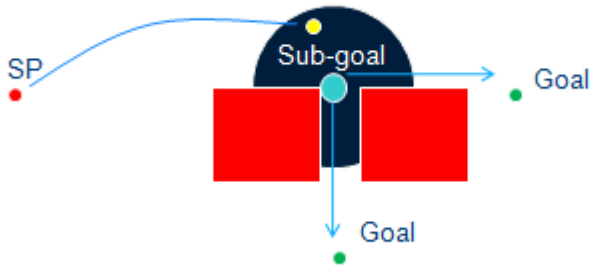

Figure 4.4: Local minima conditions

The issue can be simply resolved by introducing a check before feeding local start points and sub-goals to potential function and if any of the conditions give in equation 4.1 or 4.2 is true, potential function can be directed to a *pre sub-goal* and it is only after reaching this *pre-sub-goal*, that robot is directed towards sub-goal. *Pre-sub-goals* in such cases are found as follows:-

$$\nabla U_{local} = (LSP - q) \quad (4.3)$$

$$pre\text{-}sub\text{-}goal = LSP + \nabla U_{local} \quad (4.4)$$

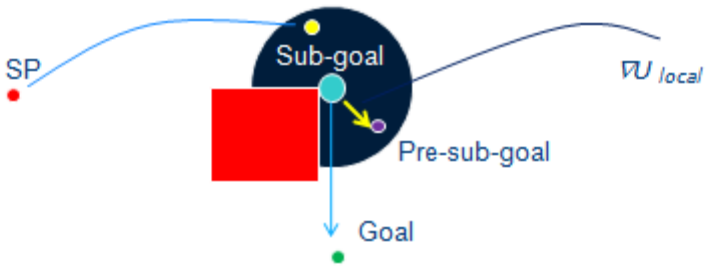

Figure 4.5: Introduction of pre-sub-goal to avoid local minima

Now robot moves from *q* to *pre-sub-goal* first as shown in figure 4.5 and then it moves from *pre-sub-goal* to goal. The above mentioned step conveniently resolves the issue of local minima for intermediate cases however, for cases involving global start point and goal it is assumed that these are neither a vertex of obstacle nor lie on any edge. This should be specially ensured for global start point.

4.2.4 Summary

This chapter introduces the proposed hybrid path planning. In the beginning the architecture of proposed planner is presented and then the methodology. This chapter also explains the technique in detail along with reasons so as to why a particular step was needed.

CHAPTER 5: IMPLEMENTATION AND RESULTS

This chapter deals with the way proposed hybrid path planner has been implemented to validate and manifest its capabilities. It also provides the results of implementing proposed hybrid path planner in five different scenarios in a reasonably complex environment. Moreover, it has also been implemented in environments as used in [37] to benchmark its performance.

5.1 Implementation

The tools and parameters which have been used in implementation of proposed hybrid path planner are listed in following paragraphs.

5.1.1 Implementation Tools

Implementation tools and their specifications are as follows: -

- MATLAB R2013a.
- 32-bit, Intel® Core™ 2 Duo system.
- 2.20 GHz processor.
- 4.00 GB Ram.

5.1.2 Implementation Parameters

Parameters used for implementing artificial potential function as component of hybrid path planners for test scenarios are listed below: -

- Tolerance $\varepsilon = 4$.
- Range of repulsive function $d^* = 3$.
- Attractive gain $\zeta = 2$ (inversely scaled with $d(q, q_{goal})$).
- Repulsive gain $\eta = 2$.

5.1.3 Implementation Scenarios

Hybrid path planner has been implemented to prove its capabilities in five different test scenarios and three benchmarking scenarios. Each of the test scenarios have been deliberately conceived for testing hybrid planner so as if the research objectives and expectations from the planner have been met or otherwise. It is important to highlight that configuration space for all

the test scenarios is the same as in [32] and so are the start point and goal however for benchmark scenarios it is different and has been taken as in [37]. The motive behind keeping the same environment, start point and goal for test scenarios is to set a standard for comparison and gauging performance in different scenarios. The detail of each scenario along with the research objective it tests is given in succeeding paragraphs.

5.1.3.1 Test Scenario No. 1

In this scenario, the same configuration space is used as already described. It is important to mention that it is a static environment where all obstacles are known priori and no dynamic obstacle is present. This scenario tests whether hybrid path planner can ensure safety of robot by keeping it away from vertices and edges of static obstacles without compromising much on optimality or not.

5.1.3.2 Test Scenario No. 2

In this scenario configuration space is the same as in previous one. All obstacles are known priori and no dynamic obstacle is present however obstacle ‘C’ has been disturbed 3 units along horizontal and 1 unit along vertical axis. The scenario is conceived to test robustness and reactive capability of path planner while keeping the optimality of path.

5.1.3.3 Test Scenario No. 3

Configuration space is the same as in previous cases. No dynamic obstacle is present in environment however, while moving from node 33 to 13 robot encounters an unknown obstacle right on the reference path generated by global component of hybrid planner. The scenario tests robustness and reactive capability of planner and also gauges increase in distance traveled while negotiating this unknown obstacle.

5.1.3.4 Test Scenario No. 4

Configuration space is the same as in previous cases however a dynamic obstacle is introduced right on the path generated by global planner. The dynamic obstacle follows the slope of reference path but in opposite direction of robot movement i.e. from node 13 to node 33. The step size of obstacle is 1.05 units per unit time (0.92 along horizontal and 0.5 along vertical axis).

The scenario tests capability of planner to avoid dynamic obstacles and ability to perform in dynamic environment.

5.1.3.5 Test Scenario No. 5

This scenario is same as scenario No. 4 apart from the movement of dynamic obstacle. In this scenario step size of obstacle is same i.e. 1.05 units per unit time however it follows a different slope i.e. translation along horizontal axis is 0.5 and along vertical axis it is 0.92. Purpose of the scenario is the same as in previous one.

5.1.3.6 Benchmarking Scenarios

To validate and benchmark the performance of proposed hybrid path planner it, has also been tested in two different environments with three different situations. Implementation and environment of [37] has been set as benchmark to gauge the performance of method proposed in this thesis in comparison with one proposed in [37].

5.2 Results

The results obtained during development of planner and its implementation can be grouped in two categories. First category of results is in which some issues regarding hybrid planner were highlighted and resolved during course of development. This category is named as accidental results. The other category of results is the one in which fully developed path planner was tested in test scenarios. Both categories of results are presented in succeeding paragraph under different headings. It is also relevant to mention that in all figures of simulation green line indicates the reference path generated by global component and red circles indicate path generated by reactive component of hybrid planner.

5.2.1 Accidental Results

There are two different unexpected results encountered during course of developing hybrid path planner which are worth mentioning. Both of these results featured local minima issue which has been highlighted along with its solution in previous chapter. The first one was encountered while robot got stuck at point (210, 61.72) while moving from node 13 to 12 as shown in figure 5.1. It is because that from node13-12, there is no gradient along horizontal axis.

The same problem as shown in figure 5.2 is faced while moving from node 12-11 in which gradient along vertical axis is zero.

If the technique of introducing pre-sub-goal given in previous chapter is employed, the robot does not get stuck in local minima and reaches successfully to node 12. It is however, again stuck at point (212.62, 20) as gradient between node 12 and 11 along vertical axis is zero. The problem however, is resolved for both steps by introducing pre-sub-goal technique given in previous chapter. It is based on this result that pre-sub-goal technique was developed for hybrid path planner.

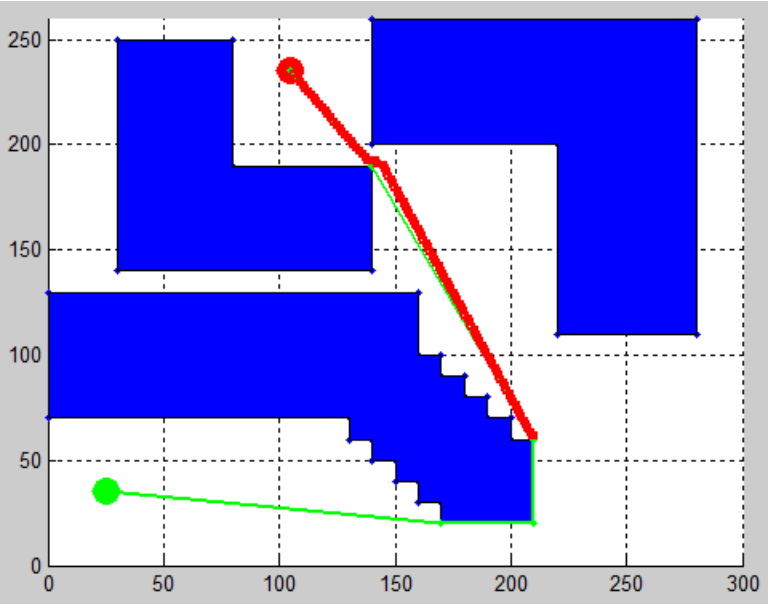

Figure 5.1: Hybrid planner (without using pre-sub-goal technique) stuck at local minima around node 13

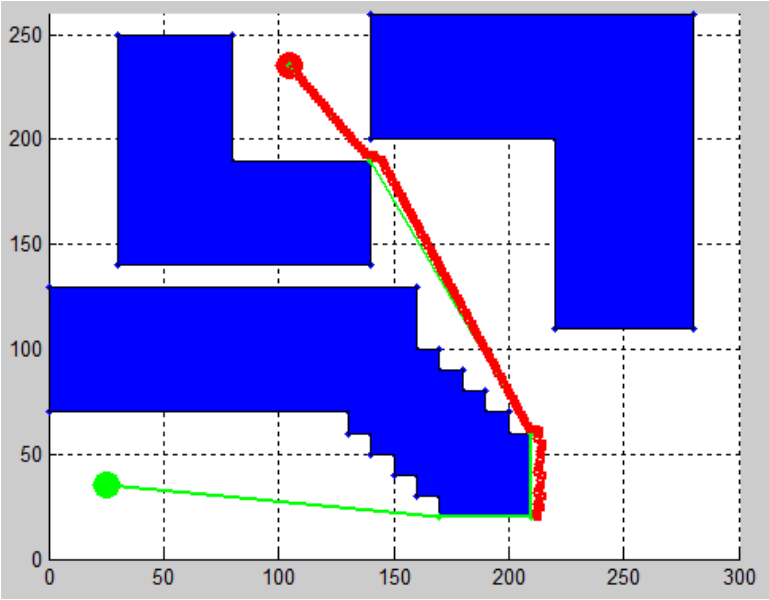

Figure 5.2: Hybrid planner (without using pre-sub-goal technique) stuck at local minima around node 12

5.2.2 Experimental Results

The results of implementing proposed hybrid path planner in all test scenarios are really encouraging. In all cases hybrid path planner successfully reached its goal while avoiding collision with all the obstacles. Besides avoiding collision with static, disturbed, unknown or dynamic obstacles, the robot never came in 1 unit range of any obstacle at any time throughout its course in reaching goal.

5.2.2.1 Results – Test Scenarios

Table 5-1 lists shortest possible path, path length in each case, percentage increase with respect to shortest path and time taken for the test scenarios N0. 1 to 5.

Table 5-1: Test results in 5 test scenarios

Test Scenario	Path Length	%age Increase in Path Length	Time
Shortest possible Path	430.43	-	-
Scenario No. 1	447.19	3.89	2.1
Scenario No. 2	449.39	4.40	2.98

Scenario No. 3	455.93	5.92	3.42
Scenario No. 4	453.99	5.47	7.91
Scenario No. 5	448.33	4.15	8.63

Figure 5.3 displays path taken by robot by adopting hybrid planner in test scenario No. 1. It is obvious from figure the robot slightly deviates from reference or optimal path and that too only when necessary to ensure its safety. In areas away from obstacles it almost precisely follows the reference path.

Figure 5.4 displays path taken by robot in test scenario No. 2 where it successfully respond to avoid displaced obstacle. It is also worth highlighting that while avoiding this displaced obstacles increase in distance from shortest path is only 4.40 percent.

Figure 5.5 displays path taken by robot in test scenario No. 3 where it successfully avoids an unknown obstacle encountered right on the reference path. The increase in distance in this case is 5.92 percent.

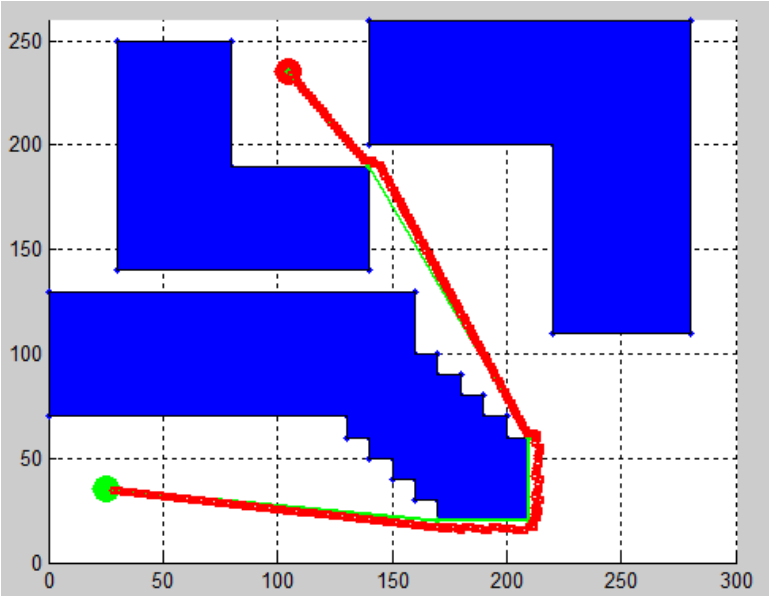

Figure 5.3: Hybrid planner in test scenario No. 1

Figure 5.6 displays path taken by robot in test scenario No. 4 where it successfully negotiated a dynamic obstacle moving right against its direction on reference path. Increase in distance while avoiding this dynamic obstacle is only 5.47 percent.

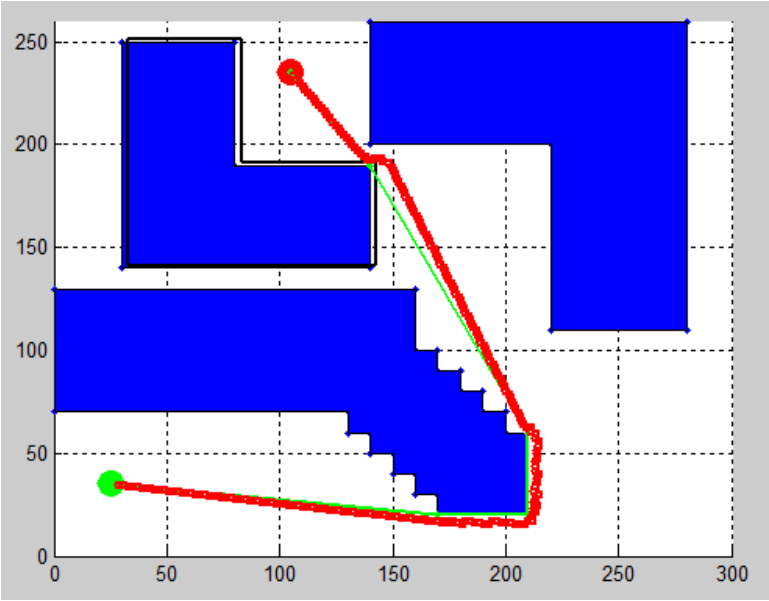

Figure 5.4: Hybrid planner in test scenario No. 2

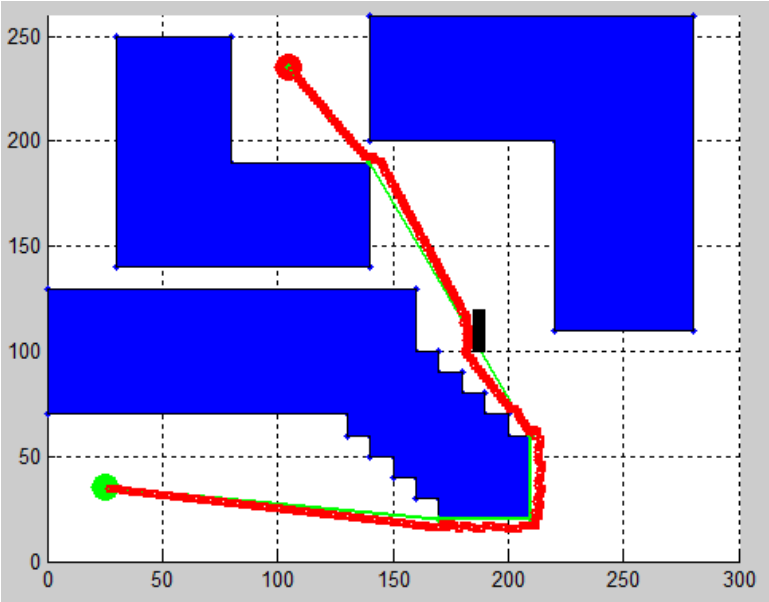

Figure 5.5: Hybrid planner in test scenario No. 3

Figure 5.7 displays path taken by robot in test scenario No. 5 where it successfully negotiated another dynamic obstacle. Increase in distance in this case is 4.15 percent.

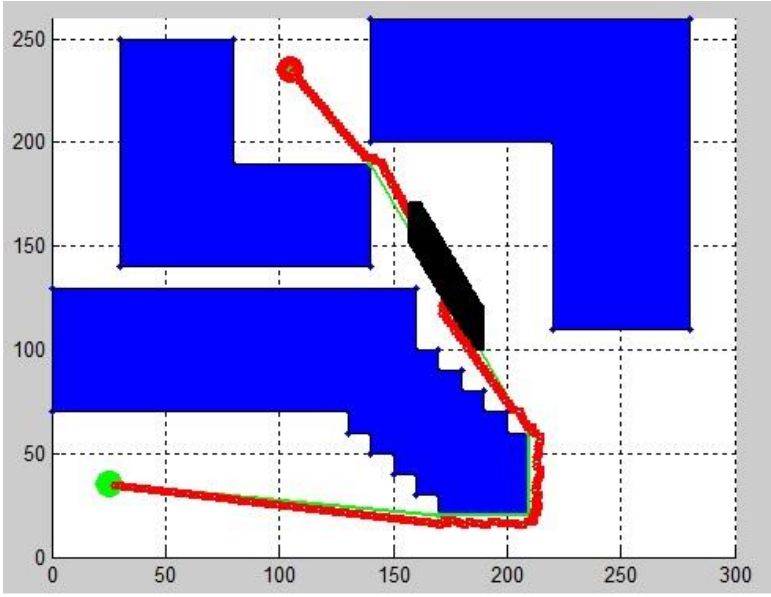

Figure 5.6: Hybrid planner in test scenario No. 4

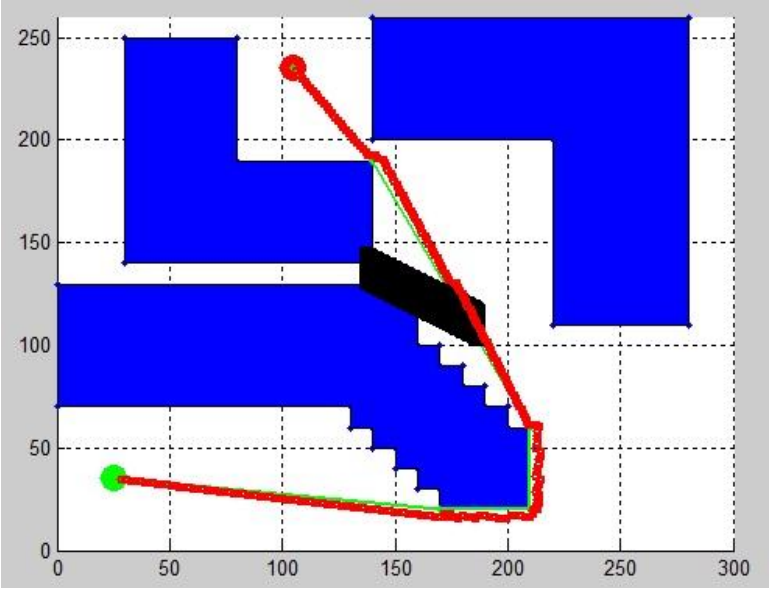

Figure 5.7: Hybrid planner in test scenario No. 5

5.2.2.2 Results – Test Scenarios

Results of three different benchmarking scenarios are presented in figure 5.8 to figure 5.11. It is however important to mention that location of obstacles, their size and location of start point and goal have been approximated by using images as this detail was not available in [37]. In above mentioned figures path by red circles is path proposed by [37] and path in blue is that of hybrid planner proposed in this thesis.

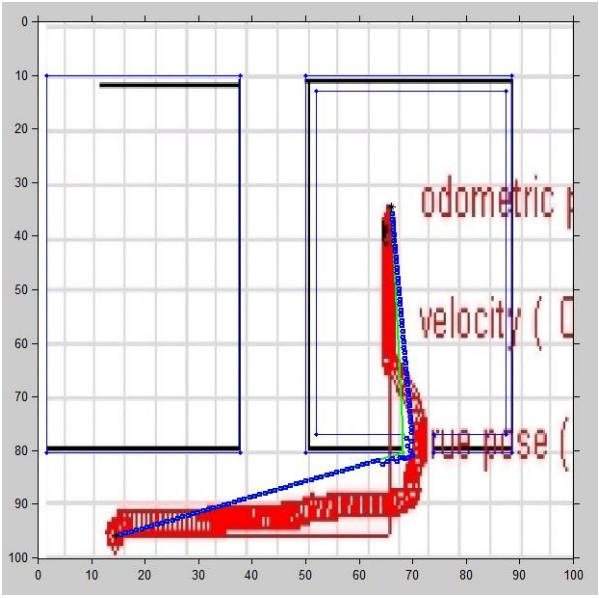

Figure 5.8: Comparison with hybrid planner of [37]

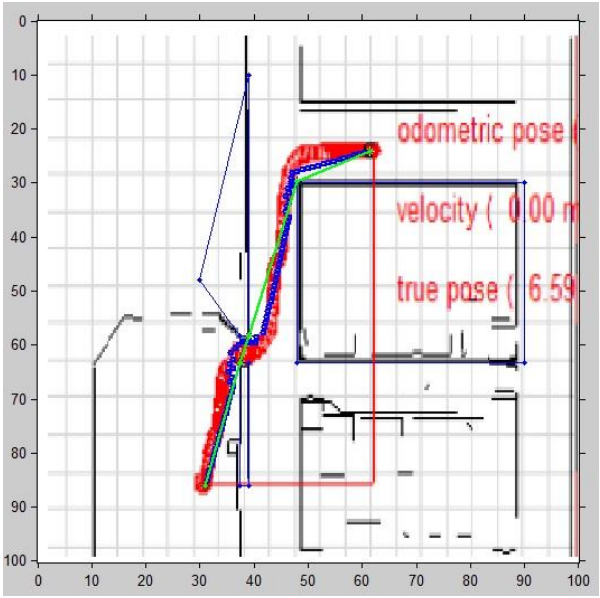

Figure 5.9: Comparison with hybrid planner of [37]

It is evident from figure 5.8 that planner proposed in this thesis not only successfully reached goal but the path generated by it is also shorter in length as compared to path generated by planner proposed in [37].

Similarly for environment shown in figure 5.9 the robot successfully finds its way to goal taking turn from door and passing through corridor. It is however important to note that there are no unknown obstacles present in this environment.

In figure 5.10 however, where the environment has been cluttered with unknown static obstacles, the robot successfully negotiates first four obstacles but is stuck around fifth obstacle. There are two reasons for failure in this case. Firstly, there is not enough space between known and unknown obstacle to allow robot to pass through secondly, since repulsive potential function employs repulsive force of only one i.e. nearest obstacle therefore it can not maneuver around unknown obstacle in this case.

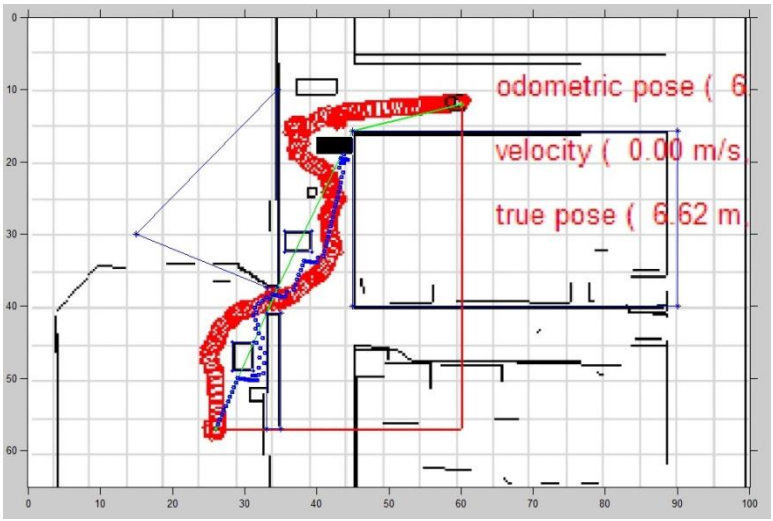

Figure 5.10: Comparison with hybrid planner of [37]

5.3 Discussion

It is very encouraging to notice that both objectives i.e. keeping robot away from edges and nodes of obstacles and not compromising much on optimality of path are achieved. Robot during its movement from global start point to goal never touches any obstacle and a safety distance is maintained by d * thus resolving the safety issue associated with visibility graph.

It is also worth noting that length of path generated by visibility graph is 430.43 units where as length of path actually taken by robot using hybrid technique is 447.19 units. Increase in distance by using hybrid technique is only 3.93 % of shortest distance i.e. path of visibility graph.

It is also worth highlighting that hybrid technique overcomes usual local minima issue associated with concave obstacles and potential field method. Had robot been using potential field alone in our configuration space, it would have faced local minima around node 34 while moving from start point to goal. The hybrid technique also helped robot to navigate a tight space between node 29 and 33.

The path generated by hybrid technique around node 33, between nodes 13 – 12 and between nodes 12 – 11 is staggered. These are the path segments where potential function is keeping robot away from edges and nodes of obstacles while guiding it to goal. It is however, a matter of concern as such behavior results in undesirable and futile directional changes consuming power and time.

The proposed planner also manifests its robustness and reactive capabilities in test scenario No. 2 where it safely maneuvered around a disturbed obstacle. It has also manifested its reactive capabilities in test scenarios No. 3 to 5, where it safely avoided collision with unknown static and dynamic obstacles. Hence it is logical to mention that proposed hybrid planner is almost optimal, safe, robust and capable of planning on-line to perform in dynamic and uncertain environment.

CHAPTER 6: CONCLUSION AND FUTURE WORK

A hybrid path planner is proposed in this research for indoor autonomous robots developed by integrating global and local path planners. To validate the proposed hybrid planner, it has been extendedly tested in really challenging and testing environments and conditions. The results show that proposed hybrid technique not only finds a near optimal path in terms of distance but safety of robot is also ensured by keeping it away from obstacles using potential function. The results of implementation also manifest robustness of proposed hybrid planner as it successfully negotiates already existing obstacle which has been slightly displaced. The planner also handles unknown static as well as dynamic obstacles by avoiding collision with them to find its path to goal thus claiming on-line planning capability.

On the basis of results, it is logical to conclude that proposed hybrid technique is capable of finding almost an optimal path and keeping robot safe. Since the path found by proposed technique is almost shortest it is, therefore, a logical assumption that time taken would also be close to optimal and so would be power consumption. It is pertinent to mention that in proposed method, safety distance of robot from obstacles is proportional to increase in distance travelled. Farther a robot is kept from obstacle, longer would be the distance travelled from start point to goal.

Results also indicate that proposed method finds a path in reasonably complex environment and robot does not get locked in local minima typically associated with concave obstacles despite presence of multiple concave obstacles in known environment. In cluttered environment, however, it has been challenged and that too for an unknown obstacle that does not have space on the side where goal is placed. It is also notified that robot takes a staggering path while moving along or parallel to edges of obstacle and as already mentioned this behavior is not desirable and should be improved upon.

On the basis of results and discussion in preceding paragraphs, it is proposed that future work may be directed to resolve the issue regarding unknown obstacles cluttering the environment. Since planner is principally conceived for indoor environment therefore a simple option could be defining rules which allow necessary space for movement of robot on both sides of unknown obstacles. Considering repulsive forces of all obstacles in range rather than that of only nearest obstacle may be an option or as the researchers may think appropriate. Similarly to

avoid staggering behavior, an attempt could be made to estimate contour of obstacle by utilizing range sensors and basing on this estimate robot may be kept from following a staggering path.

While concluding, it is pertinent to mention that proposed hybrid planner has manifested enough strength however like everything else, it also has room for improvement and future work may be directed to improve it as suggested in previous paragraph.

APPENDIX A

Matlab Code for Test Scenario No.1

```
clc;
clear all;
axis([0 300 0 260])
hold on;
grid on
tic
ver_list=[1 0 70; 1 130 70; 1 130 60; 1 140 60; 1 140 50; 1 150 50; 1 150 40;
1 160 40; 1 160 30; 1 170 30;1 170 20; 1 210 20; 1 210 60; 1 200 60; 1 200
70; 1 190 70; 1 190 80; 1 180 80; 1 180 90; 1 170 90; 1 170 100; 1 160 100; 1
160 130; 1 00 130;2 220 110; 2 280 110; 2 280 260; 2 140 260; 2 140 200; 2
220 200;3 30 140; 3 140 140; 3 140 190; 3 80 190; 3 80 250; 3 30 250];
for i=1:length(ver_list(:,1))
      if i == 1
            last_ver = 24;% first obs has 24 ver
            first_ver = 1;
      elseif i > 24 && i < 31
      last_ver = 30;
      first_ver = 25;
      elseif i > 30 && i < 37
      last_ver = 36;
      first_ver = 31;
      end
      if i < last_ver
line([ver_list(i,2)
ver_list(i+1,2)],[ver_list(i,3)ver_list(i+1,3)],'Marker','.','LineStyle','');
      elseif i == last_ver
line([ver_list(i,2) ver_list(first_ver,2)],[ver_list(i,3)
      ver_list(first_ver,3)],'Marker','.','LineStyle','-');
      end
end
fill(ver_list(1:24,2), ver_list(1:24,3),'b');
fill(ver_list(25:30,2), ver_list(25:30,3),'b');
fill(ver_list(31:36,2), ver_list(31:36,3),'b');
for k=1:24
      if k < 24
            obs1(k,[1 2]) = ver_list(k,[2 3]);
            obs1(k,[3 4]) = ver_list(k+1,[2 3]);
      else
            obs1(k,[1 2]) = ver_list(k,[2 3]);
            obs1(k,[3 4]) = ver_list(1,[2 3]);
      end
 end
 for k=25:30
      if k < 30
           obs1(k,[1 2]) = ver_list(k,[2 3]);
          obs1(k,[3 4]) = ver_list(k+1,[2 3]);
       else
           obs1(k,[1 2]) = ver_list(k,[2 3]);
           obs1(k,[3 4]) = ver_list(25,[2 3]);
       end
 end
```

```
for k=31:36
        if k < 36
             obs1(k,[1 2]) = ver_list(k,[2 3]);
             obs1(k,[3 4]) = ver_list(k+1,[2 3]);
          else
             obs1(k,[1 2]) = ver_list(k,[2 3]);
             obs1(k,[3 4]) = ver_list(31,[2 3]);
         end
 end
line([105 105], [235 235],'marker','o','Color','r','LineWidth',8);%start pt
line([25  25],[35 35],'marker','o','Color','g','LineWidth',8);%Goal pt
line([105 ver_list(33,2)],[235
ver_list(33,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(33,2) ver_list(13,2)],[ver_list(33,3)
ver_list(13,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(13,2) ver_list(12,2)],[ver_list(13,3)
ver_list(12,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(12,2) ver_list(11,2)],[ver_list(12,3)
ver_list(11,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(11,2) 25],[ver_list(11,3)
35],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
  % dist travelled in bench mark paper
  path_dist_bm = eu_dist([105 235],[215 125]) + eu_dist([215 125],[215 15]) +
eu_dist([215 15],[175 15]) + eu_dist([175 15],[155 35])+ eu_dist([ 25 35
],[155 35])
     std_vis_dist = eu_dist([105 235], ver_list(33,[2 3])) +
eu_dist(ver_list(13,[2 3]), ver_list(33,[2 3])) + eu_dist(ver_list(13,[2 3]),
ver_list(12,[2 3])) + eu_dist(ver_list(12,[2 3]), ver_list(11,[2 3]))+
eu_dist([25 35], ver_list(11,[2 3]))
g_pt = [25 35]; % 25 35
start_pt = [105 235];
obs = obs1;
[cp dist1] = fn_potential_func_multiple_obs( start_pt, ver_list(33,[2 3]),
obs1, 1,1,1) ;
[cp dist2] = fn_potential_func_multiple_obs(cp, ver_list(13,[2 3]),obs1,
1,1,1) ;
z = ver_list(13,[2 3]) - cp;
cp1 = ver_list(13,[2 3])+z;
[cp dist3] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1);
[cp dist4] = fn_potential_func_multiple_obs(cp, ver_list(12,[2 3]),obs1,
1,1,1);
 z = ver_list(12,[2 3]) - cp;
cp1 = ver_list(12,[2 3])+z;
[cp dist5] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1) ;
[cp dist6] = fn_potential_func_multiple_obs(cp, ver_list(11,[2 3]),obs1,
1,1,1) ;
[cp dist7] = fn_potential_func_multiple_obs(cp, g_pt,obs1, 1,1,1) ;
dist = dist1 + dist2 + dist3 + dist4 + dist5 + dist6 + dist7
toc

function [q_new, dist] = fn_potential_func_multiple_obs(sp, goal, obs,
tolerence, att_gain, rep_gain, j, result)
q = sp;
tolerence= 4;
i =1;
dist = 0;
distq_goal = 100;
```

```
      while distq_goal > tolerence %i < 100
      if i ~=1
       q = q_new;
       end
      if i == 1000
          i = i+1;
            break
       end
for k=1:length(obs(:,1))
     [s_dist(k,1) n_pt(k,:)] = pt_line_dist(q, obs(k,[1 2]), obs(k,[3 4]));
end
      [d_q_c ind_c] = min(s_dist(:,1));
      c = n_pt(ind_c,[1 2]);
             if d_q_c < 3
                      if d_q_c < 1
                             pause;
                       end
                 change2 = (2 *(q -c))/d_q_c;
              else
                 change2 = 0;
              end
             distq_goal = eu_dist(q,goal);
             delta_q = [q - goal];
             change1 = (2*delta_q)/distq_goal;
             q_new = q + change2 - change1;
             error = eu_dist(q_new,goal);
             distq_goal =  eu_dist(q_new,goal);
             dist = dist + eu_dist(q,q_new);
             i = i+1;
plot(q_new(1,1),
q_new(1,2),'o','LineWidth',2,'MarkerEdgeColor','r','MarkerSize',3);
        drawnow;
        hold on;
        clear s_dist
        clear n_pt
        clear ind_c
        clear d_q_c
        clear c
         end
 cp = q_new;
 end

function [eucledian_dist]= eu_dist (pt_1,pt_2)% [pt_1]=[x y] & [pt_2]=[x y]
eucledian_dist = sqrt((pt_1(1)-pt_2(1))^2 + (pt_1(2)-pt_2(2))^2);
end

function [ s_dist, n_pt ] = pt_line_dist( pt, s_line, e_line )
d_se = norm(s_line - e_line);
d_s_pt = norm(s_line - pt);
d_e_pt = norm(e_line - pt);
       if dot(s_line - e_line, pt - e_line) * dot(e_line - s_line, pt -
s_line) >= 0 %if min dist pt is pt between sp end end pt or not?
         a = [s_line,1; e_line,1; pt,1];
         s_dist = abs(det(a))/d_se;
       else
s_dist = min(d_s_pt, d_e_pt);% shortest dist between line segment and ref pt
    end
```

```
sp_dist = eu_dist(s_line, pt);
        ep_dist = eu_dist(e_line, pt);
        if s_dist == sp_dist
 n_pt = s_line;          % nearest pt n_pt
        elseif s_dist == ep_dist
            n_pt = e_line;          % nearest pt n_pt
        else
        slope_line = (e_line(1,2) - s_line(1,2))/ (e_line(1,1) -
s_line(1,1));  % slope of line segment
          [dist angle] = range_bearing(s_line, e_line);
             theta_line = angle;
             if theta_line < 0
                theta_line = 2*pi + theta_line;
             end
 slope_pt_line = (pt(1,2) - s_line(1,2))/ (pt(1,1) - s_line(1,1));        %
             [dist angle]  = range_bearing(s_line, pt);
             theta_pt_line = angle;
            if theta_pt_line < 0
              theta_pt_line = 2*pi + theta_pt_line;
           end
             dist_pt_line = eu_dist(pt,s_line);
            if theta_pt_line > theta_line
              theta_bw = theta_pt_line-theta_line;
dx = dist_pt_line * cos(theta_bw);           %dist on line from sp
x = s_line(1,1) + (dx * cos (theta_line));   % x coord of n_pt on line
               y = s_line(1,2) + (dx * sin (theta_line));
n_pt= [x y];          % nearest pt n_pt
             else
                theta_bw = theta_line - theta_pt_line;
dx = dist_pt_line * cos(theta_bw);
x = s_line(1,1) + (dx * cos (theta_line));  % x coord of n_pt on line
y = s_line(1,2) + (dx * sin (theta_line));  % y coord of n_pt on line
n_pt= [x y];                                % nearest pt n_pt
            end
        end
end
```

APPENDIX B

Matlab Code for Test Scenario No.2

```
clc;
clear all;
axis([0 300 0 260]);
hold on;
grid on;
tic
ver_list=[1 0 70; 1 130 70; 1 130 60; 1 140 60; 1 140 50; 1 150 50; 1 150 40;
1 160 40; 1 160 30; 1 170 30;1 170 20; 1 210 20; 1 210 60; 1 200 60; 1 200
70; 1 190 70; 1 190 80; 1 180 80; 1 180 90; 1 170 90; 1 170 100; 1 160 100; 1
160 130; 1 00 130;2 220 110; 2 280 110; 2 280 260; 2 140 260; 2 140 200; 2
220 200;3 30 140; 3 140 140; 3 140 190; 3 80 190; 3 80 250; 3 30 250];
for i=1:length(ver_list(:,1))
       if i == 1
              last_ver = 24;     % first obs has 24 ver
              first_ver = 1;
       elseif i > 24 && i < 31
      last_ver = 30;% first obs has 24 ver, 25 to 30 ver correspond to obs #2
              first_ver = 25;
       elseif i > 30 && i < 37
      last_ver = 36; % first 2 obs have 30 ver, 31 to 36 correspond to obs #3
              first_ver = 31;
         end
       if i < last_ver
            line([ver_list(i,2) ver_list(i+1,2)],[ver_list(i,3)
ver_list(i+1,3)],'Marker','.','LineStyle','-');
       elseif i == last_ver
              line([ver_list(i,2) ver_list(first_ver,2)],[ver_list(i,3)
ver_list(first_ver,3)],'Marker','.','LineStyle','-');
       end
 end
  fill(ver_list(1:24,2), ver_list(1:24,3),'b');
  fill(ver_list(25:30,2), ver_list(25:30,3),'b');
  fill(ver_list(31:36,2), ver_list(31:36,3),'b');
  j=1
for i=31:36
      nver(j,:) =transpose( [1 0 3; 0 1 1; 0 0 1]*
[ver_list(i,2);ver_list(i,3);1]);
      j=j+1;
end
for i=1:6
       if i < 6
line([nver(i,1) nver(i+1,1)],[nver(i,2) nver(i+1,2)],'color',
'k','linewidth', 2);
       elseif i == 6
line([nver(i,1) nver(1,1)],[nver(i,2) nver(1,2)],'color','k', 'linewidth',2);
         end
 end
for k=1:24
      if k < 24
               obs1(k,[1 2]) = ver_list(k,[2 3]);
               obs1(k,[3 4]) = ver_list(k+1,[2 3])
```

```
      else
              obs1(k,[1 2]) = ver_list(k,[2 3]);
              obs1(k,[3 4]) = ver_list(1,[2 3]);
         end
 end

 for k=25:30
         if k < 30
              obs1(k,[1 2]) = ver_list(k,[2 3]);
              obs1(k,[3 4]) = ver_list(k+1,[2 3]);
          else
              obs1(k,[1 2]) = ver_list(k,[2 3]);
              obs1(k,[3 4]) = ver_list(25,[2 3]);
          end
 end
for k=31:36
          if k < 36
               obs1(k,[1 2]) = ver_list(k,[2 3]);
               obs1(k,[3 4]) = ver_list(k+1,[2 3]);
           else
              obs1(k,[1 2]) = ver_list(k,[2 3]);
              obs1(k,[3 4]) = ver_list(31,[2 3]);
              end
end
line([105 105], [235 235],'marker','o','Color','r','LineWidth',8); %start pt
  line([25  25],[35 35],'marker','o','Color','g','LineWidth',8);%Goal pt%
  line([105 ver_list(33,2)],[235
ver_list(33,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(33,2) ver_list(13,2)],[ver_list(33,3)
ver_list(13,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(13,2) ver_list(12,2)],[ver_list(13,3)
ver_list(12,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(12,2) ver_list(11,2)],[ver_list(12,3)
ver_list(11,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(11,2) 25],[ver_list(11,3)
35],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
  path_dist_bm = eu_dist([105 235],[215 125]) + eu_dist([215 125],[215 15]) +
eu_dist([215 15],[175 15]) + eu_dist([175 15],[155 35])+ eu_dist([ 25 35
],[155 35])
std_vis_dist = eu_dist([105 235], ver_list(33,[2 3])) +
eu_dist(ver_list(13,[2 3]), ver_list(33,[2 3])) + eu_dist(ver_list(13,[2 3]),
ver_list(12,[2 3])) + eu_dist(ver_list(12,[2 3]), ver_list(11,[2 3]))+
eu_dist([25 35], ver_list(11,[2 3]))
g_pt = [25 35]; % 25 35
start_pt = [105 235];
obs = obs1;
   obs1(31:36,:) = [];
   i = 1
for k=31:36
           if k < 36
                obs1(k,[1 2]) = nver(i,[1 2]);
                obs1(k,[3 4]) = nver(i+1,[1 2]);
                i=i +1
           else
               obs1(k,[1 2]) = nver(i,[1 2]);
               obs1(k,[3 4]) = nver(1,[1 2]);
           end
```

```
 end
[cp dist1] = fn_potential_func_multiple_obs( start_pt, ver_list(33,[2 3]),
obs1, 1,1,1) ;
[cp dist2] = fn_potential_func_multiple_obs(cp, ver_list(13,[2 3]),obs1,
1,1,1) ;
z = ver_list(13,[2 3]) - cp;
cp1 = ver_list(13,[2 3])+z;
[cp dist3] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1);
[cp dist4] = fn_potential_func_multiple_obs(cp, ver_list(12,[2 3]),obs1,
1,1,1) ;
z = ver_list(12,[2 3]) - cp;
cp1 = ver_list(12,[2 3])+z;
[cp dist5] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1) ;
[cp dist6] = fn_potential_func_multiple_obs(cp, ver_list(11,[2 3]),obs1,
1,1,1);
[cp dist7] = fn_potential_func_multiple_obs(cp, g_pt,obs1, 1,1,1) ;
dist = dist1 + dist2 + dist3 + dist4 + dist5 + dist6 + dist7
toc
```

APPENDIX C

Matlab Code for Test Scenario No.3

```
clc;
clear all;
axis([0 300 0 260]);
hold on;
grid on;
tic
ver_list=[1 0 70; 1 130 70; 1 130 60; 1 140 60; 1 140 50; 1 150 50; 1 150 40;
1 160 40; 1 160 30; 1 170 30;1 170 20; 1 210 20; 1 210 60; 1 200 60; 1 200
70; 1 190 70; 1 190 80; 1 180 80; 1 180 90; 1 170 90; 1 170 100; 1 160 100; 1
160 130; 1 00 130;2 220 110; 2 280 110; 2 280 260; 2 140 260; 2 140 200; 2
220 200;3 30 140; 3 140 140; 3 140 190; 3 80 190; 3 80 250; 3 30 250];
for i=1:length(ver_list(:,1))
      if i == 1
               last_ver = 24;     % first obs has 24 ver
               first_ver = 1;
      elseif i > 24 && i < 31
    last_ver = 30;% first obs has 24 ver, 25 to 30 ver    correspond to obs #2
               first_ver = 25;
      elseif i > 30 && i < 37
    last_ver = 36; % first 2 obs have 30 ver, 31 to 36 correspond to obs #3
               first_ver = 31;
       end
      if i < last_verline([ver_list(i,2) ver_list(i+1,2)],[ver_list(i,3)
ver_list(i+1,3)],'Marker','.','LineStyle','-');
      elseif i == last_verline([ver_list(i,2)
ver_list(first_ver,2)],[ver_list(i,3)
ver_list(first_ver,3)],'Marker','.','LineStyle','-');
      end
end
  fill(ver_list(1:24,2), ver_list(1:24,3),'b');
  fill(ver_list(25:30,2), ver_list(25:30,3),'b');
  fill(ver_list(31:36,2), ver_list(31:36,3),'b');
  for k=1:24
         if k < 24
                 obs1(k,[1 2]) = ver_list(k,[2 3]);
                 obs1(k,[3 4]) = ver_list(k+1,[2 3]);
          else
                 obs1(k,[1 2]) = ver_list(k,[2 3]);
                 obs1(k,[3 4]) = ver_list(1,[2 3]);
          end
  end
  for k=25:30
         if k < 30
                 obs1(k,[1 2]) = ver_list(k,[2 3]);
                 obs1(k,[3 4]) = ver_list(k+1,[2 3]);

          else
                 obs1(k,[1 2]) = ver_list(k,[2 3]);
                 obs1(k,[3 4]) = ver_list(25,[2 3])
          end
    end
```

```
for k=31:36
            if k < 36
                  obs1(k,[1 2]) = ver_list(k,[2 3]);
                  obs1(k,[3 4]) = ver_list(k+1,[2 3]);
            else
                  obs1(k,[1 2]) = ver_list(k,[2 3]);
                  obs1(k,[3 4]) = ver_list(31,[2 3]);
            end
     end
   line([105 105], [235 235],'marker','o','Color','r','LineWidth',8);
%start pt
    line([25  25],[35 35],'marker','o','Color','g','LineWidth',8);   %Goal pt
    line([105 ver_list(33,2)],[235
ver_list(33,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(33,2) ver_list(13,2)],[ver_list(33,3)
ver_list(13,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(13,2) ver_list(12,2)],[ver_list(13,3)
ver_list(12,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(12,2) ver_list(11,2)],[ver_list(12,3)
ver_list(11,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(11,2)
25],[ver_list(11,3)35],'Marker','.','Color','g','LineStyle','-
','LineWidth',2);
path_dist_bm = eu_dist([105 235],[215 125]) + eu_dist([215 125],[215 15]) +
eu_dist([215 15],[175 15]) + eu_dist([175 15],[155 35])+ eu_dist([ 2535],[155
35])
std_vis_dist = eu_dist([105 235], ver_list(33,[2 3])) +
eu_dist(ver_list(13,[2 3]), ver_list(33,[2 3])) + eu_dist(ver_list(13,[2 3]),
ver_list(12,[2 3])) + eu_dist(ver_list(12,[2 3]), ver_list(11,[2 3]))+
eu_dist([25 35], ver_list(11,[2 3]))g_pt = [25 35]; % 25 35
start_pt = [105 235];
obs1 = [obs1; [185 100 190 100]; [190 100 190 120]; [190 120 185 120]; [185
120 185 100]];ui_obs = [185 100; 190 100; 190 120; 185 120; 185 100];
     line([185 190],[100 100], 'color','k');
     line([190 190],[100 120], 'color','k');
     line([190 185],[120 120], 'color','k');
     line([185 185],[120 100], 'color','k');
     fill(ui_obs(:,1), ui_obs(:,2),'k');
[cp dist1] = fn_potential_func_multiple_obs( start_pt, ver_list(33,[2 3]),
obs1, 1,1,1) ;
[cp dist2] = fn_potential_func_multiple_obs(cp, ver_list(13,[2 3]),obs1,
1,1,1);
z = ver_list(13,[2 3]) - cp;
cp1 = ver_list(13,[2 3])+z; [cp dist3] = fn_potential_func_multiple_obs(cp,
cp1,obs1, 1,1,1) ;
[cp dist4] = fn_potential_func_multiple_obs(cp, ver_list(12,[2 3]),obs1,
1,1,1) ;
z = ver_list(12,[2 3]) - cp;
cp1 = ver_list(12,[2 3])+z;
 [cp dist5] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1) ;
[cp dist6] = fn_potential_func_multiple_obs(cp, ver_list(11,[2 3]),obs1,
1,1,1) ;
[cp dist7] = fn_potential_func_multiple_obs(cp, g_pt,obs1, 1,1,1) ;
dist = dist1 + dist2 + dist3 + dist4 + dist5 + dist6 + dist7 toc;
```

APPENDIX D

Matlab Code for Test Scenario No. 4 and 5

```
clc;
clear all;
axis([0 300 0 260])
hold on;
grid on;
ver_list=[1 0 70; 1 130 70; 1 130 60; 1 140 60; 1 140 50; 1 150 50; 1 150 40;
1 160 40; 1 160 30; 1 170 30;1 170 20; 1 210 20; 1 210 60; 1 200 60; 1 200
70; 1 190 70; 1 190 80; 1 180 80; 1 180 90; 1 170 90; 1 170 100; 1 160 100; 1
160 130; 1 00 130;2 220 110; 2 280 110; 2 280 260; 2 140 260; 2 140 200; 2
220 200;3 30 140; 3 140 140; 3 140 190; 3 80 190; 3 80 250; 3 30 250];
for i=1:length(ver_list(:,1))
     if i == 1
            last_ver = 24;     % first obs has 24 ver
            first_ver = 1;
    elseif i > 24 && i < 31
 last_ver = 30;     % first obs has 24 ver, 25 to 30 ver correspond to obs #2
             first_ver = 25;
   elseif i > 30 && i < 37
 last_ver = 36;     % first 2 obs have 30 ver, 31 to 36 correspond to obs #3
             first_ver = 31;
    end
   if i < last_ver line([ver_list(i,2) ver_list(i+1,2)],[ver_list(i,3)
ver_list(i+1,3)],'Marker','.','LineStyle','-');
    elseif i == last_ver line([ver_list(i,2)
ver_list(first_ver,2)],[ver_list(i,3)
ver_list(first_ver,3)],'Marker','.','LineStyle','-');
     end
 end
  fill(ver_list(1:24,2), ver_list(1:24,3),'b');
  fill(ver_list(25:30,2), ver_list(25:30,3),'b');
  fill(ver_list(31:36,2), ver_list(31:36,3),'b');
  tic;
for k=1:24
      if k < 24
        obs1(k,[1 2]) = ver_list(k,[2 3]);
        obs1(k,[3 4]) = ver_list(k+1,[2 3]);
      else
        obs1(k,[1 2]) = ver_list(k,[2 3]);
        obs1(k,[3 4]) = ver_list(1,[2 3]);
      end
end
for k=25:30
       if k < 30
           obs1(k,[1 2]) = ver_list(k,[2 3]);
           obs1(k,[3 4]) = ver_list(k+1,[2 3]);
       else
      obs1(k,[1 2]) = ver_list(k,[2 3]);
     obs1(k,[3 4]) = ver_list(25,[2 3]);
      end
end
for k=31:36
```

```
      if k < 36
           obs1(k,[1 2]) = ver_list(k,[2 3]);
          obs1(k,[3 4]) = ver_list(k+1,[2 3]);
      else
           obs1(k,[1 2]) = ver_list(k,[2 3]);
            obs1(k,[3 4]) = ver_list(31,[2 3]);
       end
end
line([105 105], [235 235],'marker','o','Color','r','LineWidth',8);
line([25  25],[35 35],'marker','o','Color','g','LineWidth',8);
line([105 ver_list(33,2)],[235
ver_list(33,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(33,2) ver_list(13,2)],[ver_list(33,3)
ver_list(13,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(13,2) ver_list(12,2)],[ver_list(13,3)
ver_list(12,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(12,2) ver_list(11,2)],[ver_list(12,3)
ver_list(11,3)],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
    line([ver_list(11,2) 25],[ver_list(11,3)
35],'Marker','.','Color','g','LineStyle','-','LineWidth',2);
  path_dist_bm = eu_dist([105 235],[215 125]) + eu_dist([215 125],[215 15]) +
eu_dist([215 15],[175 15]) + eu_dist([175 15],[155 35])+ eu_dist([ 25 35
],[155 35])
std_vis_dist = eu_dist([105 235], ver_list(33,[2 3])) +
eu_dist(ver_list(13,[2 3]), ver_list(33,[2 3])) + eu_dist(ver_list(13,[2 3]),
ver_list(12,[2 3])) + eu_dist(ver_list(12,[2 3]), ver_list(11,[2 3]))+
eu_dist([25 35], ver_list(11,[2 3]))
g_pt = [25 35]; % 25 35
start_pt = [105 235];
obs1 = [obs1; [185 100 190 100]; [190 100 190 120]; [190 120 185 120]; [185
120 185 100]];
[cp dist1] = fn_potential_func_multiple_obs( start_pt, ver_list(33,[2 3]),
obs1, 1,1,1) ;
[cp dist2] = fn_potential_func_multiple_obs_dynamic(cp, ver_list(13,[2
3]),obs1, 1,1,1);
z = ver_list(13,[2 3]) - cp;
cp1 = ver_list(13,[2 3])+z;
[cp dist3] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1) ;
[cp dist4] = fn_potential_func_multiple_obs(cp, ver_list(12,[2 3]),obs1,
1,1,1) ;
z = ver_list(12,[2 3]) - cp;
cp1 = ver_list(12,[2 3])+z;
[cp dist5] = fn_potential_func_multiple_obs(cp, cp1,obs1, 1,1,1);
[cp dist6] = fn_potential_func_multiple_obs(cp, ver_list(11,[2 3]),obs1,
1,1,1);
[cp dist7] = fn_potential_func_multiple_obs(cp, g_pt,obs1, 1,1,1);
dist = dist1 + dist2 + dist3 + dist4 + dist5 + dist6 + dist7
toc;

function [q_new, dist] = fn_potential_func_multiple_obs(sp, goal, obs,
tolerence, att_gain, rep_gain, j, result)
xdata = [obs(37,1) obs(38,1) obs(39,1) obs(40,1) obs(37,1)];
ydata = [obs(37,2) obs(38,2) obs(39,2) obs(40,2) obs(37,2)];
p1 = plot(xdata,ydata,'','LineWidth',2,'MarkerEdgeColor','r','MarkerSize',3);
updated_obs = [];
q = sp;
tolerence= 4;
```

```
i =1;
dist = 0;
distq_goal = 100;
 while distq_goal > tolerence %i < 100
       if i ~=1
            q = q_new;
            obs = updated_obs;
       end
       if i == 1000
          i = i+1;
       break
       end
for k=1:length(obs(:,1))
[s_dist(k,1) n_pt(k,:)] = pt_line_dist(q, obs(k,[1 2]), obs(k,[3 4]));
end
[d_q_c ind_c] = min(s_dist(:,1));
c = n_pt(ind_c,[1 2]);
     if d_q_c < 3
          if d_q_c  < 1
                accident = 1
                pause;
          end
      change2 = (2 *(q -c))/d_q_c;
     else
           change2 = 0;
     end
            distq_goal = eu_dist(q,goal);
            delta_q = [q - goal];
            change1 = (2*delta_q)/distq_goal;
            q_new = q + change2 - change1;
            error = eu_dist(q_new,goal);
            distq_goal =  eu_dist(q_new,goal);
            dist = dist + eu_dist(q,q_new);
            i = i+1;
plot(q_new(1,1),q_new(1,2),'o','LineWidth',2,'MarkerEdgeColor','r','MarkerSiz
e',3);
       drawnow;
        clear s_dist
        clear n_pt
        clear ind_c
        clear d_q_c
        clear c
        if i > 25
            updated_obs = func_update_dynamic_obs (obs);
            obs = updated_obs;
            xdata = [obs(37,1) obs(38,1) obs(39,1) obs(40,1) obs(37,1)];
            ydata = [obs(37,2) obs(38,2) obs(39,2) obs(40,2) obs(37,2)];
           set(p1, 'xdata', xdata, 'ydata',ydata)drawnow
        else
            updated_obs = obs;
         end
     end
cp = q_new;
end
```

```
function [ updated_obs ] = update_dynamic_obs( obs )
j=1;
    for i=37:40
nver(j,:) =transpose( [1 0 -0.92; 0 1 0.5; 0 0 1]* [obs(i,1);obs(i,2);1]);
 j=j+1;
    end
        obs(37:40,:)= [];
         i = 1;
    for k=37:40
     if k < 40
          obs(k,[1 2]) = nver(i,[1 2]);
          obs(k,[3 4]) = nver(i+1,[1 2]);
           i=i +1;
    else
          obs(k,[1 2]) = nver(i,[1 2]);
           obs(k,[3 4]) = nver(1,[1 2]);

     end
 end
updated_obs = obs;
end
```

REFERENCES

[1] Yuan-Qing Qin, De-Bao Sun, Ning Li, and Yi-Gang Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in Proceeding of the Third International Conference on Machine learning and Cybernetics, Shanghai, pp. 2473-2478, 2004.

[2] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza, "Introduction to autonomous mobile robots". In MIT-Press, pp. 1-12, 2004.

[3] Hwang, Y. K. and Ahuja, N. "Gross motion planning-A survey", ACM Comp. Surv., Vol. 24, No.3, (1992), pp.219-291.

[4] Atyabi, Adham; Powers, David M. W."Review of classical and heuristic-based navigation and path planning approaches," International Journal of Advancements in Computing Technology; Oct 2013, Vol. 5 Issue 14, p1

[5] Adham Atyabi, Sepideh Samadzadegan, "Particle Swarm Optimization: A Survey". In Louis P. Walters, ed. Applications of Swarm Intelligence. Hauppauge, USA: Nova Publishers, pp. 167-178, 2011.

[6] A Atyabi, S PhonAmnuaisuk, K Ho Chin, S Samadzadegan, "Particle Swarm Optimizations:A Critical Review", Proceeding of Conf IKT07, Third conference of Information and Knowledge Technology, Ferdowsi University, Iran, PP. 1-7, 2007.

[7] Gordon Cheng, Alexander Zelinsky, "A physically grounded search in a behavior based robot". In Proc. Eighth Australian Joint Conference on Artificial Intelligence, pp. 547-554, 1995.

[8] Ellips Masehian, Davoud Sedighizadeh, "Classic and heuristic approaches in robot motion planning - a chronological review". In Proceedings of world academy of science, engineering and technology, vol. 23, pp. 101-106, 2007.

[9] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza, "Introduction to autonomous mobile robots". In MIT-Press, pp. 1-12, 2004.

[10] Asano, T., Asano, T, Guibas, L., Hershberger, J., and Imai, H."Visibility-polygon search and Euclidean shortest path", 26th Symp. Found. Comp. Science (1985) pp. 155-164.

[11] Jean-Claude Latombe, "Robot Motion Planning". The Springer International Series in Engineering and Computer Science, vol. 124, pp. 153, 1991.

[12] C. O Dunlaing and C. Yap, A “retraction method for planning the motion of a disk, J. Algorithms 6 (1985), 104 – 111.

[13] Canny, J. F., "A Voronoi method for the piano-movers problem". Proc. IEEE ICRA (1985).

[14] Goerzen C, Kong Z, Mettler B, “A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance”, Journal of Intelligent and Robotic Systems, vol. 57, no. 1-4, pp. 65- 100, 2010.

[15] Milos Seda, “Roadmap method vs. cell decomposition in robot motion planning”. In Proceedings of 6th WSEAS international conf on signal processing, Robotics and automation, Grees, pp. 127- 132, 2007.

[16] Michael Barbehenn, Seth Hutchinson, “Toward an exact incremental geometric robot motion planner”, In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 39-44, 1995.

[17] Jacob T. Schwartz, Micha Sharir, “On the piano movers problem: the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers”. In Communications on Pure and Applied Mathematics, vol. 36, pp. 345-398, 1983.

[18] Nora H. Sleumer, Nadine Tschichold-Gürman, “Exact cell decomposition of arrangements used for path planning in robotics”. In Swiss Federal Institute of Technology, 1991.

[19] Michael Barbehenn, Seth Hutchinson, “Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees”. In IEEE Trans. on Robotics and Automation, vol. 11, no. 2, pp. 198-214, 1995.

[20] David Zhu, Jean-Claude Latombe, “New heuristic algorithms for efficient hierarchical path planning”. In IEEE Trans. on Robotics and Automation, vol. 7, no. 1, pp. 9-20, 1991.

[21] Conte G, Zullil R, “Hierarchical path planning in a multi-robot environment with a simple navigation function”. In IEEE Trans. On Systems, Man, and Cybernetics, vol. 25, no. 4, pp. 651-654, 1995.

[22] Frank Lingelbach, “Path planning using probabilistic cell decomposition”. In Int. Conf. on Robotics and Automation, pp. 467-472, 2004.

[23] Jan Rosell, Pedro Iniguez, “Path planning using harmonic functions and probabilistic cell decomposition”. In Int. Conf. on Robotics and Automation, pp. 1803-1808, 2005.

[24] F. Kunwar, F. Wong, R. Ben Mrad, and B. Benhabib, “Rendezvous Guidance for the Autonomous Interception of Moving Objects in Cluttered Environments.” IEEE International Conference of Robotics and Automation, pp. 3776 – 3781, Barcelona, Spain, April 2005.

[25] R.Bohlin and L. Kavraki, “Path planning using lazy PRM,” In Proceedings of International Conference on Robotics and Automation, 2000.

[26] N. Amato, O. Bayazit, L.Dale, C.Jones, and D. Vallejo, “OBPRM: An obstacle-based PRM for 3D workspaces,” In Proc. International Workshop on Algorithmic Foundations of Robotics (WAFR), 1998.

[27] S. M. LaValle. “Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, October 1998.

[28] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in Robotics and Automation. In Proceedings IEEE, International Conference on Robotics and Automation, volume 2, pp 500 – 505, March 1985.

[29] J. Borenstein and Y. Koren, “The Vector Field Histogram – fast obstacle avoidance for mobile robots,” IEEE Transaction on Robotics and Automation, vol. 7, pp. 278 – 288, 1991.

[30] Oroko, Joannes, and G. N. Nyakoe. "Obstacle Avoidance and Path Planning Schemes for Autonomous Navigation of a Mobile Robot: A Review."Proceedings of Sustainable Research and Innovation Conference. 2014.

[31] D. Fox, W. Burgard, and S. Thrun, ”The dynamic window approach to collision avoidance”, in IEEE Robot. Autom. Mag., vol. 4, pp. 23 33,1997.

[32] L.C. Wang, L.S. Yong, and M.H. Ang., “Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment,” in Proceedings of the International symposium on Intelligent Control, Vancouver, BC, Canada, October 2002.

[33] Ray Jarvis, “Distance transform based path planning for robot navigation," in Recent trends in mobile robots, Yuan F. Zheng, Ed., vol. 11 of Robotics and Intelligent Systems, chapter 1.World Scientific, 1993.

[34] H. Mei, Tian Y. and Zu. L., “A Hybrid Ant Colony Optimization Algorithm for path Planning of Robots in Dynamic Environment,” International Journal of Information Technology, Vol. 12, No 3, pp. 78-88, 2006.

[35] Ming-Chih Lu, Chen-Chien Hsu., Yuan-Jun Chen, and Shih-An Li, “Advances in Autonomous Robotics,” Volume 7429 of the series, pp. 441-443, 2012.

[36] Qidan Z.; Yongjie Y. and Zhuoyi X. "Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing" Proc. ISDA'06, (2006) pp. 622-627.

[37] Yan, Zh., Sun, Y. and Wang, W., "Mobile robot hybrid path planning in an obstacle-cluttered environment based on steering control and improved distance propagating," International journal of innovative computing, information & control, ISSN 1349-4198, June 2012.

[38] A. R. Willms and S. X. Yang, "An efficient dynamic system for real-time robot path planning," IEEE Transactions on System, Man and Cybernetics, Part B: Cybernetics, vol. 36, no. 4, pp. 755-766, 2006.

[39] S. M. Lavalle, M. S. Branicky and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps, International Journal of Robotics Research," vol. 23, no.7, pp.673-692, 2004.

[40] Mahajan, PM Bhushan, and P. Marbate. "Literature review on path planning in dynamic environment." International Journal of Computer Science and Network 2.1 (2013): 115-118.

[41] Lozano-Pérez, Tomás Wesley, and Michael A., "An algorithm for planning collision free paths among polyhedral obstacles," Communications of the ACM, 22 (10), pp 560–570, October 1979.

[42] O. Takahashi and R.J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," IEEE Transactions on Robotics and Automation, 5(2), pp 143 –150, April 1989.

Completion Certificate

It is to certify that the thesis titled "A Hybrid Path Planning Technique for Indoor Autonomous Robots, Developed by Integrating Global and Local Path Planner" submitted by registration no. NUST201261255MCEME35512F, Muhammad Imran of MS-74 Mechatronics Engineering is complete in all respects as per the requirements of Main Office, NUST (Exam branch).

Supervisor: _____

Dr. Kunwar Faraz Ahmad Khan

Date: ____ Jan, 2016