OPERATIONS RESEARCH IN MOTION PLANNING

BY

HASEEB TAHIR

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INDUSTRIAL & SYSTEMS ENGINEERING

22nd May, 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **HASEEB TAHIR** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN INDUSTRIAL & SYSTEMS ENGINEERING**.

Thesis Committee

Mujahid

Dr. Syed. N. Mujahid (Adviser)

(Co-adviser)

Dr. Shokri Selim (Member)

U. Baroudi

Dr. Uthman Baroudi (Member)

(Member)

17-8-2017

Dr. Hesham K. Al-Fares
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

22/8/17

Date

Dedicated to the Almighty Allah for blessing me with the company of my wife who always stood besides me and gave me the emotional strength during the course of my thesis.

ACKNOWLEDGMENTS

Glory be to Allah, the Almighty, Who created & proportioned and Who destined then guided, and taught the man what he knew not. Peace be upon the Prophet Mohammad, his family, his companions, and all those who followed him until the Day of Judgment. First of all I would like to thank Allah, the Almighty who has provided me all what I needed in my life in spite of my disobedience towards Him. I then would like to extend my deepest gratitude and respect to my parents, my siblings, my beloved wife and my friends who had always been a source of encouragement, motivation and love. Specially, I would like to acknowledge my father in law, who helped and motivated me in making the decision to pursue with my graduate studies.

Also, I would like to thank King Fahd University of Petroleum and Minerals which gave me the opportunity to pursue a graduate degree and has been an epitome of innovation. I would like to thank my research advisor Dr. Syed. N. Mujahid who extended his full support and guidance from the very beginning of this research. My appreciations are also extended to my committee members: Dr. Shokri Selim and Dr. Uthman Baroudi for their useful discussions, feedback and

the time that they spent reviewing this thesis. All my teachers deserve thankfulness especially Dr. Shokri Selim and Dr. Saleh Duffuaa who introduced to me the subject of Operations Research and which ended up becoming my topic of MS thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME:	Haseeb Tahir
TITLE OF STUDY:	Operations Research in Motion Planning
MAJOR FIELD:	Industrial & Systems Engineering
DATE OF DEGREE:	$22^{nd} May, 2017$

Motion planning is one of the fundamental research problems for autonomous systems where some agents have to carryout certain tasks in a complex environment. Some recent research areas involving motion planning are found in self driving cars, computer games, autonomous robots for exploration, surveillance and emergency situations etc. Typically, motion planning needs to be done locally at agent's onboard computing systems. An important aspect of motion planning is collision avoidance of the agents with their environment. Such problems have been well studied in the literature. Typically, the nonlinear avoidance constraints are non-convex. Thus, the optimal motion planning of multiple agents in the presence of dynamic obstacles is an NP-Hard problem [1]. Additionally, the solution of such problems may be computationally very expensive and eventually may not be practical to be performed on the on-board computing system of the agent. In this *thesis, we present a novel algorithm for motion planning in mutli-agent dynamic environment. It is assumed that the agents can obtain the surface information and velocities of other agents and obstacles present in some vicinity around them. Additionally, it is assumed that the agents cannot communicate with each other. A novel mathematical formulation is developed using the above information. The algorithm runs locally by each agent for its trajectory planning. The algorithm has been successfully tested in various single and multi agent dynamic scenarios. The algorithm is also designed to avoids oscillations in various complex scenarios. Some simplifications for 2-D and 3-D cases have been presented at the end which further reduce the computation times of the proposed algorithm.*

مخلص الرسالة

الاسم الكامل: حسيب طاهر

عنوان الرسالة: بحوث العمليات في تخطيط الحركة

التخصص: هندسة الصناعة والنظم

تاريخ الدرجة: ٢٢ مايو ٢٠١٧

تخطيط حركة الاجسام في النظم المستقلة يعد مسالة بحثية اساسية حيث ان بعض هذه الاجسام تنفذ بعض المهام في بيئة معقدة. الابحاث الحديثة الخاصة بتخطيط الحركة تشمل سيارات القيادة الذاتية، ألعاب الكمبيوتر، الروبوتات المستقلة للاستكشاف والمراقبة وحالات الطوارئ الخ. عادة، يتم تخطيط الحركة محليا في الحاسب المركب على متن الوكيل(الجسم). من الجوانب الهامة في تخطيط الحركة تجنب الاصطدام بالاجسام الاخرى المتواجدة في بيئتها. وقد درست هذه المشاكل بشكل جيد في الأدب. لتجنب الاصطدام نستعمل-عادة- قيود غير خطية غير محدبة. وهكذا، فإن التخطيط الأمثل لحركة أجسام متعددة في حالة وجود عقبات او اجسام أخرى متحركة (ديناميكية) هي مشكلة ذات صعوبة اكثر من المشاكل متعددة الحدود [١]. بالإضافة إلى ذلك، قد يكون حل مثل هذه المشاكل مكلف جدا حسابيا، وفي نهاية المطاف قد لا يكون من العملي أن يتم تنفيذها على الحاسب المركب على متن الوكيل. في هذه الأطروحة، نقدم خوارزمية جديدة لتخطيط الحركة في بيئة ديناميكية متعددة الاجسام. في هذه الخوارزمية نفترض أن الجسم يمكنه الحصول على معلومات عن سطح وسرعات الاجسام الأخرى والعقبات الموجودة في الحيز المحيط به. بالإضافة إلى ذلك، نفترض أن الاجسام لا يمكنها التواصل مع بعضها البعض. تم تطوير صياغة رياضية جديدة باستخدام المعلومات المذكورة أعلاه. الخوارزمية تدار محليا من قبل كل جسم لتخطيط مساره. وقد تم اختبار الخوارزمية بنجاح في سيناريوهات ديناميكية ذات جسم واحد واخرى متعددة الاجسام. تم تصميم الخوارزمية أيضا لتجنب التردد الحركي في سيناريوهات معقدة مختلفة. وقد قدمت في النهاية بعض التبسيط لحالات ذات بعد ثنائي و ثلاثية الابعاد مما يقلل أكثر زمن حساب الخوارزمية المقترحة.

CHAPTER 1

INTRODUCTION

In pursuit of the technological advancement, man has always been striving to create more and more efficient processes and systems. Automation is one of the most prominent factors in the overall technological revolution in 20^{th} and 21^{st} centuries. Not only does it save time, energy and labor but also improves quality and accuracy.

The ever growing demands for precision with tighter profit margins make the study of optimality in the automation of processes, an important area of research. Motion planning is one the most significant aspects of autonomous systems which is why it still continues to be the area of interest for researchers, especially for the past three to four decades.

1.1 Background

The motion planning problem has its origins from the trajectory optimization problem. Its roots can be traced back more than three centuries ago when, a

Swiss mathematician, Johann Bernoulli invited all the renowned mathematicians of the time to solve a problem famously known as “Brachystochrone” [1] (which is a trajectory optimization problem by minimizing the time of traverse). For the next couple of centuries, scientists and mathematicians continued to study this problem by the method of calculus of variations [1], [2]. Later on, with the advent of computers, other techniques related to the subject of optimal control were developed [1], [3]-[5].

Motion planning is one the most significant attributes of autonomous systems and still continues to be an active area of research for the past few decades. Some early applications of motion planning were found in trajectory optimization problems related to rocket propulsion and space exploration[2, 3]. Furthermore, automation in the manufacturing industry brought forward new ways that could be explored using the motion planning techniques [4, 5, 6, 7, 8, 9]. Recent applications of motion planning can be seen in diverse research areas like machine learning, advanced manufacturing processes, puzzle solutions, computer graphics, games, navigation in self-driving cars and Autonomous Unmanned Vehicles (AUV) [8, 9]. These applications are typical examples of trajectory planning where multiple decision making agents are planning their motion in a common environment. Most of these applications occur in large, dynamic and complicated environments where global information is difficult to obtain and each agent may have to rely on the information obtained in some nearby vicinity. The agents may or may not be able to communicate with each other and the environment may also contain static or dynamic obstacles of any shape. Therefore, the problem becomes difficult to solve due to the above mentioned issues and, typically, onboard motion planning algorithms must be developed that can be deployed locally on the agents (limited resources).

1.2 Motivation

In spite of the wide variety of problems being addressed by motion planning, new possibilities still continue to emerge, particularly due to the revolutionary improvement in drone technology. For instance, one key area is logistics which is usually the most critical and expensive component of any supply chain. With the expansion of internet and its usage, a lot of retail business has shifted online. Big companies like DHL and Amazon are putting in huge investments in the research of AUVs capable of planning the optimal route to numerous customer locations and thus managing the delivery of goods. Likewise, remote controlled drones are common these days for surveillance purposes. Enabling these machines to make requisite decisions for trajectory planning and perform required surveillance tasks autonomously, doesn't seem to be far from now. Flying machines can also be used in emergency situations like firefighting, rescue operations, bomb disposal, first-aid delivery, fog and smog prevention etc. Similarly, the progression in nanotechnology, nanofabrication and microelectronics has significantly reduced the size of onboard embedded computing systems. This opens further avenues for the research and development of miniature robots and mechanisms, capable of making decisions and performing complicated tasks in the areas that cannot be accessed otherwise. One such application is in the field of medicine where such miniature agents can plan their motion while moving with the blood stream or in the intestinal canal of the human body to reach the assigned target location and perform a certain task like operating a tumor. In all of the above discussed areas, the onboard motion planning specially performed in dynamic environment is a complicated task and the algorithms developed thus far to deal with such problems are either computationally very expensive or compromise too much on the optimality. Of course, most of the above applications are available with only local nearby knowledge of the environment and, therefore, global optimal solutions in these situations may not be expected. However, techniques may be explored which can provide better results in terms of optimality and also reduce the computational complexity as compared to the available onboard motion planning methods.

1.3 Computational Complexities

Motion planning for single agent in the presence of static obstacles can be modeled as a mathematical programming problem. The difficulty arises due to the presence of collision avoidance constraint as shown in Equation (1.1) which makes the problem non-convex:

$$||p_a - q_i||_m \geq D \tag{1.1}$$

where, p_a and q_i are the current positions of the agent a and obstacle i respectively, $|| \bullet ||_m$ is the m^{th} norm and D is the minimum distance between their centers to avoid the collision. In addition to that, motion planning in the presence of dynamic obstacles becomes a combination of path planning and velocity planning problems [10]. The dynamic motion planning problem where each agent is modeled as a point among multiple moving bodies with bounded velocity is proven to be a NP-hard problem [1].

The onboard motion planning specially performed in dynamic environment is a complicated task. Current algorithms developed thus far to deal with such problems are either computationally very expensive or compromise too much on the optimality (shortest path).

The trajectory planning algorithms give some set of finite transformations that can be applied to an agent from its initial to the goal location. The set of all possible transformations that may move an agent from initial location to any other location is referred as state space or Configuration Space (C-space) [8]. In simple words, C-space is actually how the agent sees and understands its environment and primarily depends on how the problem is modeled. Obviously, there is a lot of unnecessary detail in the environment and therefore, the problem should be formulated in a way that it uses the available information efficiently and thus, may be optimized with the limited onboard resources. The model of the problem also depends on the available sensors on the agent that give information about the environment. The algorithm should find the best sequence of transformations that yields the optimal trajectory to the target.

1.4 Structure of Thesis

In this thesis report, a novel algorithm is presented in multi agent multi obstacle scenario. The report is structured as follows; Chapter 2 presents the comparison of motion planning algorithms in literature along with some related problems and limitations. Detailed methodology is covered in Chapter 3, which includes problem statement, mathematical formulation and the proposed algorithm to find collision free trajectories for each agent in multi agent and multi obstacle dynamic environment. The algorithm is tested by simulating various scenarios and the results are presented in Chapter 5. The report is concluded in Chapter 6 with details regarding its applicability in real life motion planning problems.

1.5 Problem Statement

We are given with N number of agents present in an environment with k number of obstacles. The agents are assumed to be spherical in shape with radius r_i of the i^{th} agent and are capable of holonomic motion. The agents have the information of their respective target locations. The obstacles present in the environment may be static or dynamic. The agents have the capability to recognize the obstacles that are present in some vicinity of radius s_i. Specifically, each agent can obtain the surface edge information of the obstacles and other agents present in its vicinity. Each agent is also capable of acquiring the velocities v_i and w_j of every i^{th} agent and j^{th} obstacle respectively (that are present in its sensor's vicinity). The agents may not be able to communicate with each other and the position and velocity information as mentioned above is assumed to be available only due to the agent's own sensing capability.

Given the above scenario, the agents need to plan their motion while moving through this dynamic environment. As evident from the description above, the agents do not have global information of the environment and therefore may not be able to find the global optimal paths. Also as stated earlier, even if the global information is available, finding global optimal paths in such situations is a NP-hard problem [1]. The objective is to have an algorithm in place which can find collision free paths of traverse for each agent. Furthermore, the algorithm should be fast and simple enough to suit the limited onboard computational capability of the agents.

CHAPTER 2

LITERATURE REVIEW

The algorithms developed thus far for motion planning can be classified into 4 types: mathematical programming algorithms, Artificial Potential Field methods (APF), sampling based methods and reactive or sensor based methods. The mathematical programming techniques use the usual Operations Research (OR) models such as Mixed Integer Linear Programming (MILP) or Mixed Integer Quadratic Programming (MIQP) to find optimal trajectories. On the other hand, APFs construct a virtual force function which produces a force field creating attractive force towards the target and repulsive force away from the obstacles. Hence, the resultant force guides the agent with collision free motion towards its target. The concept was first introduced in [11] and was later on applied for robotic arm mechanisms. In contrast to the above, sampling based algorithms can address complicated motion planning problems and the challenges faced by potential field methods may be solved by them, especially where a complicated multi degree of freedom motion is required. The reactive methods utilize the local information available to the agent in its nearby vicinity. These methods are generally suitable for onboard path planning where computational resources are limited.

2.1 Mathematical Programming Models

A classical way to model multi agent path planning problems with no obstacle is discussed in [12]. The problem is solved by a quadratic programming model that minimizes the thrust required for each agent to traverse a collision free path and reach the known target location. The difficulty arises due to the presence of collision avoidance constraints which are typically non-convex. These are linearly approximated with sequential convex programming approach to find collision free paths. The method is further extended with an Incremental Sequential Convex Programming (ISCP) approach in [13] with both coupled and decoupled variations. ISCP slightly improves the results presented in [12] in terms of computational time and gives feasible solutions even in non-convex shaped environments. However, these techniques are computationally expensive methods and perform poorly in the presence of dynamic agents. The reverse convex constraints as presented in Equation (1.1) have been well studied in the literature and algorithms have been suggested for finding optimal solutions specifically for convex problems with only one additional reverse convex constraint [14, 15]. However, these techniques are inappropriate for multi agent dynamic environments where numerous such constraints may be present and the feasible space may become complicated enough to be analyzed for global optimality. MILP models can also be used for non-convex optimization such as in [16], where the usual quadratic cost function is expressed as weighted norm-1 instead of the positive definite weighting matrices and each obstacle as a polytope. Each of the linear constraints forming these polytopes is associated with an integer variable so that the non-convex feasible space can be searched for the optimal path. A similar formulation, for a set of spacecraft travelling in space, is presented in [17] that avoids collision (plume impingement) with other spacecraft and obstacles. A slightly different variation of [17] is presented in [18] where multi agent motion planning problem is formulated as Mixed Integer Non Linear Programming (MINLP) model and later converted to MILP by obtaining schedules that form upper and lower bounds on optimal solution via solving two MILP problems. Two point boundary value problems are solved to find minimum and maximum times taken by each agent to traverse a segment by putting constraints on accelerations and finding achievable final velocities. Motion planning in quadrotors is also an application of Mixed Integer Programming (MIP) models where the trajectories are planned in 3-D space, for instance in [19]. MIQP model is presented as an extension of MILP. The control inputs for position, roll, pitch and yaw angles are optimized for each of the discrete time instants and piece-wise smooth polynomial functions are used at the end to synthesize smooth trajectories so that they can be followed accurately. Also, Legendre polynomials are used as basis functions to ensure numerical stability of the solver. The MIP techniques may render optimal or near optimal solutions, but are computationally very expensive due to the presence of integer variables. The problem dimensionality increases exponentially with increasing number of agents and huge computational resources and time may be required for optimization. Additionally, mathematical programming techniques require central computing and sensing capabilities which may not be feasible for many real world applications.

2.2 Potential Field Methods

The numerical complexities faced by the mathematical programming models may be avoided by another set of methods known as Artificial Potential Field (APF) (first introduced in [11]). The collision free trajectory is obtained by introducing a virtual force field function which depends on the location and shape of target and obstacles. APF methods provide a simple procedure of collision avoidance in motion planning problems but their mathematical analysis shows some limitations making them unsuitable for some real world applications. One difficulty is the trapping of algorithm in local minima which occurs in situations when the forces due to attractive fields and repulsive fields balance out. This may be caused due to the relative location of target and obstacles with respect to the agent or the obstacle shape or size. Second issue is the limitation of motion between the closely spaced obstacles. This is due to the reason that repulsive forces produced by the obstacles are high enough to overcome the attractive forces even when the space between the obstacles allows robot motion. Third issue is the oscillating motion, while moving near the obstacles and narrow passages [20].

The issue of local minima has been investigated with detailed mathematical analysis (for example see [21]). The study of Hessian of the potential field function shows its non-convexity particularly near the obstacles. So, the navigation functions are constructed by incorporating certain parameters in the potential function in a way that local maxima only occurs at the boundary of the active space. First, a Boolean combination of sets are used to transform the Euclidean space and then scalar conditioning functions are introduced to level out the non-convexity to some extent. But tweaking these parameters for different scenarios may become impractical for dynamic environment and get very complicated even for static environments. Typically, the onboard computing system of the agent may not be capable to handle the complex scenarios.

Attempts have also been made to address the issue of oscillations while moving through narrow passages, for example in [22] where the grid histogram model of the environment is reduced to one dimensional polar histogram. Similarly, a Newtonian Potential Field (NPF) based model has been proposed in [23], where a uniform charge distribution on the boundaries of polygonal regions is assumed that can be derived in closed form. The representation of charge distribution in closed form avoids computationally expensive numerical evaluation of repulsion which requires discretization of regions' boundaries. The repulsive forces are minimized by a gradient search method and then a local planning algorithm is used to find collision free paths in nearby vicinity that are latter connected to obtain the global path. This method may produce oscillation free motion through narrow passages but cannot address the issues related to the existence of local minima. Another issue faced by APF methods is Goal Non-Reachable with Obstacles Nearby (GNRON), which is because of the non-convexity of potential function in the presence of obstacles near goal position. To ensure that global minimum of potential function lies only at the goal position, an additional function is introduced in the potential field expression that depends on the distances of goal and obstacles from the agent's position [24].

APF approaches may also be extended to address dynamic motion planning problems requiring soft landing of agents on moving targets. Such problems may be addressed by incorporating velocities, in addition to the positions, of targets and obstacles in the potential function. A similar model is presented in [25] which is later formulated with non-holonomic differential constraints and the algorithm is applied in different scenarios of multi agent dynamic environment problems. This approach may provide feasible solutions but assumes the environment to be dynamic enough such that the issue of local minima does not persist for long. Different local planning methods and heuristics are proposed for dealing with the local minima issues, however, there seems to be no inherent solution to it. An even bigger complication is to find out whether the agent is trapped in local minima or is it just trying to oscillate because of the overall motion of target and obstacles.

The potential field methods have been applied in several real scenarios like [26] where multiple soccer playing robots are present. However, these methods may require sophisticated sensing systems to accurately capture the environmental details so that the requisite force fields are generated. Also, the issues of optimality of path and being stuck into local minima still persist. Although, these challenges have been addressed through some heuristic methods and regression based techniques, their pratical use is a complicated task. The situation complicates even further when the obstacles and target may even be moving in a multi agent environment and this simplistic model of the motion planning problem may not achieve the desired results [27, 28].

2.3 Sampling Based Methods

In contrast to the above, sampling based algorithms can address complicated motion planning problems and the challenges faced by the exact methods may be solved by them, especially where a complicated multi degree of freedom motion is higher dimensional motion is required. Sampling based algorithms generally consist of two steps; first is to acquire a probabilistic sample of configuration space and second is to search for the desired trajectories by an appropriate metaheuristic. Two of the most well-known classical sampling based algorithms are Probabilistic Road Maps (PRMs) and Rapidly exploring Random Trees (RRTs). Roadmaps define the configuration space topology by developing a network of collision free trajectories and PRMs are simply the Monte-Carlo evolution of the roadmaps. On the other hand, incremental search methods such as dynamic programming, A*, bi-directional search etc. evolved into randomized methods like RRT. PRMs evolved from the concept of expansive spaces where, the sampling is expanded in only relevant portion of the configuration space. Such methods have been applied in multi-degree complex robotic movements especially in the maintainability of automotive industry [29]. Alternatively, RRTs randomly explore the environment by biasing the search through random sampled points in unexplored portion of the state space and are specifically designed to handle problems with non-holonomic constraints and high degree of freedom [30, 31]. Several versions of RRTs such as single RRT planners, bi-directional planners and few other approaches have been studied and presented for various practical applications with non-linear models of three, seven and nine dimensions (due to the inclusion of kinematic variables and constraints) [31, 32]. The RRT and PRM based methods are only probabilistically complete [33], i.e., the probability that they return a solution if one exists increases with the number of samples. For instance, when RRTs are run multiple times, they show a considerable improvement in the solution quality with lower cost paths although they may not converge to the optimal solution [34]. Also, they are proven not to be asymptotically optimal and for simplified PRMs, where optimality may not be the concern, they are computationally expensive [35]. To deal with bigger configuration spaces with higher memory requirements, RRT* is extended as RRT* Fixed Nodes [36] by limiting the memory requirements using node removal procedure. The optimality is comparable to RRT* with limited memory resources. RRTs may also be used in multi agent problems where each agent may have its own RRT instance which makes the search process time consuming. RRG, an extension of RRT [37], is another way to deal with multi agent problems where each agent develops a random sub-graph biased towards its goal node and then different sub-graphs are connected together in a network. Although RRG is computationally more expensive than RRTs, it is better in terms of optimality and overall processing time as it is run individually on each agent. PRMs have also been used with Dynamic Robot Networks (DRN) [38] in dynamic motion planning problems but heavily relies on robot network communication for environment sampling.

Another group of sampling based methods is of Artificial Intelligence (AI) algorithms such as A*, D*, focused D*, LPA*, D* Lite etc. [39]. The environment may be modeled using regular grids, irregular grids, navigation mesh, visibility graph and veronoi diagram. These graphing methods are more suitable to 2-D environments. The computationally expensive environmental sampling in AI algorithms may be improved to some extent by using the concept of super nodes where each super node represents a group of connected sub-graphs [40]. AI methods have also been extended for 3-D spaces as well [41]. Another set of algorithms are based on meta-heuristic methods like: Ant-Colony (ACO), Simulated Annealing (SA) and Genetic Algorithms (GA) [39]. Some recent work in the literature on evolutionary algorithms is presented in [42, 43, 44, 45]. The performance of GA is analyzed in [43] in large sized grid environments with various crossover and mutation probabilities. It performs similar to A* and it is concluded that GA may be used to improve the existing solutions obtained by A*. Similarly, SA is compared with A* and GA, and it is shown to give near optimal solution for large grids [44]. A hybrid method is suggested in [45] to combine ACO and GA. The algorithm looks for near optimal solution through an improved version of ACO and then tries to improve solution quality using GA. The main advantage of metaheuristics is their solution efficiency but same results may not be reproduced because of their stochastic nature [39]. Although, the convergence in sampling based algorithms requires many samples, they may still be used as heuristic methods for escaping the local minima issues.

In contrast to all the above discussed methods, reactive methods plan motion by only using the local information available to the agent in its nearby vicinity. Such methods are suitable for onboard path planning where computational resources are limited. The issue of local minima faced by previous methods is solved by certain sensors based methods [46] that work similar to the boundary following approach found in the bug algorithms. The algorithm uses instant goal approach that combines the lower level boundary following approach with the higher level path planning algorithm to find trajectory of the agent. This helps in planning path when obstacles boundaries are far away and the algorithm does not get stuck in local minimas. Another similar sensor based method is named as Nav [47]. Using the same principle, the agent moves straight towards the target and starts to track obstacles boundaries when the obstacle is detected. To deal with the issue of loops trap, the potential function which guides the agent towards its target includes an indicator whose value increases whenever the agent is trapped in a loop. The algorithm is able to solve navigation problem with minimal information and does not require self-localization which is computationally expensive. However, there is no systematic way to detect the of loop traps without any self-localization mechanism. Some biological immune systems inspired fuzzy models have also been proposed to deal with the local minima trap [48]. The method is referred to as reactive immune networks in which the affinity functions are defined by the same idea used in APF methods, i.e., creating a repulsive and attractive force using obstacles and target respectively. Detection of the trap of local minima is done by analyzing the change in movement angle of motion of the agent. An algorithm is developed to obtain virtual targets and this information in combination with the reactive immune network leads the agent out of the trap.

2.4 Reactive Methods

Some reactive methods also use the concept of collision cones. The concept was first suggested in [10] where it is assumed that the agent can acquire velocity information of objects in the environment and based on that, a set of collision cones are constructed. If the current velocity vector of the agent falls into any of these cones, the collision will occur. Collision cones are used to model the real map and are then transformed into virtual maps by transforming each obstacle as a virtual robot. Collision is detected by analyzing these cones and the requisite change in agent velocity or its angle is done [49]. These changes are incorporated in the differential motion parameters to obtain the required motion. The difficulty arises in the presence of cluttered environment and narrow passages when there are too many collision cones and there should be a mechanism in place to decide the priority of obstacle to be avoided. As already discussed, the problem becomes non-convex as some optimal velocity needs to be searched in a set of non-convex collision cones. Bearing angles, which can easily be obtained through camera sensors data, may also be used like collision cones for navigation by air or sea [50]. These methods may be simple to implement and effective for very small number of obstacles but are infeasible for cluttered environments. The reactive methods have also been suggested for non-holonomic motion planning [51]. If global information is not available, then the reactive methods relies on boundary tracking with similar results as in some previously discussed methods. Reactive path planning techniques are generally greedy type algorithms, but the idea of collision cones provides a sound mathematical foundation of the problem for further optimization. Therefore, such ideas have been extended with some mathematical programming techniques to avoid collision in multi agent environment, for example in [52]. The problem is first modeled with non-convex collision cones. Later, hyperspaces are chosen such that only feasible velocities are left and require minimum deviation from the agent's preferred velocity. Thus, the non-convex feasible space is approximated into convex polytope. Each agent solves a quadratic objective function to avoid collision in multi agent dynamic environment. The algorithm is further improved with Hybrid Reciprocal Velocity Obstalces (HRVO) to deal with some issues related to oscillations in agents [53].

2.5 Limitations

The mathematical programming methods mostly assume the availability of global information, which is not possible for many the real world applications. Also, these methods require considerable computational resources to run the optimization solvers and, therefore, may not be suitable for limited onboard computational capabilities. Potential field methods also typically require global information of the environment. Moreover, the performance of the sampling based method depends upon the number of samples. These methods also have additional limitations of getting stuck into local minima and oscillations which are difficult to address. Sampling based methods may address the above issues but generally require a pre-processing step to acquire sampled topology of the environment. For multi agent dynamic environments, the sampling of environment in each time instant becomes a challenging task.

The reactive planning methods require limited computational resources and, therefore, are suitable for onboard motion planning applications. However, these methods also have some limitations. Simple reactive methods cannot find efficient trajectories and may only be used in addition to some other planning algorithm. They are greedy algorithms and may get stuck in local minima especially in cluttered environments. Reactive methods in combination with optimization techniques may render good solution quality but may require high computational capabilities for onboard optimization. Also, non-convexity of the collision cones may be avoided by convex approximation of the feasible space but doing this is itself a cumbersome task.

CHAPTER 3

METHODOLOGY

This chapter presents the proposed algorithm in detail. Initially, the problem scenario has been established in Section 1.5 with all the assumptions and initial conditions. Based on the problem statement, the mathematical formulation is presented in Section 3.1 for the case of single agent multi obstacles in 2-D dynamic environment. Section 3.2 elaborates the extension of the proposed model for 3-D environment. A search heuristic is then developed for the agent's velocity that generates a collision free trajectory. The model is later extended for multi agent environments, where all the agents simultaneously find collision free trajectories.

3.1 Formulation for 2-Dimensional Scenario

Consider a scenario where an agent is present in a multi agent and multi obstacle dynamic environment. The agent is assumed to be able to gather boundary / edge information of other agents and the obstacles that are present in its vicinity. A 2-D pictorial representation of single agent and multiple obstacles scenario is shown in Figure (3.1), where the agent has obtained the velocities and surface edges of the obstacles present in its vicinity. The edge information of each j^{th} obstacle present in the vicinity of the agent can be obtained by forming two vectors, R_j and L_j, showing relative position of the right and left edges of its surface from the center of the agent respectively (the agent's center is considered as the origin of the vector space). In order to avoid collision with an obstacle, the center of the

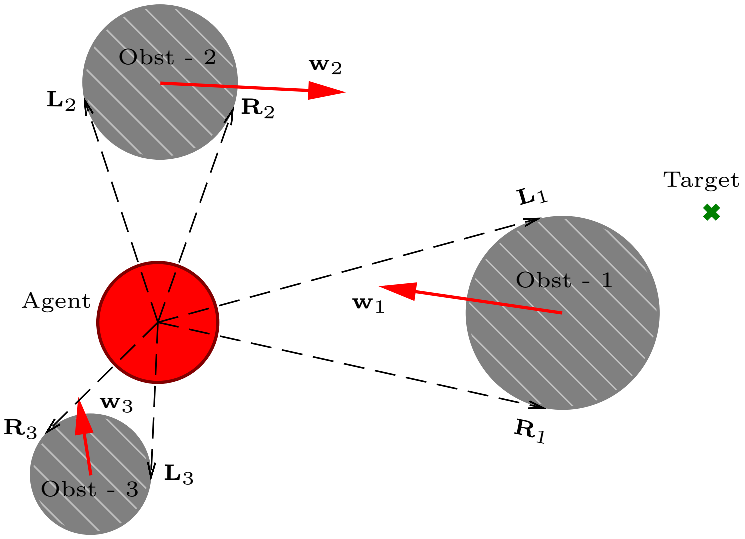


Figure 3.1: Single agent multiple obstacles scenario, where $\mathbf{w}_j$ is the current velocity of j^{th} obstacle

agent should always remain at some distance away from the obstacles. The set of all such minimum distances, thus, form Collision Spaces (CS) which depend on the dimensions of the agent and each obstacle. The collision spaces with respect to each obstacle can be conveniently formed by taking the following Minkowski sum (See Figure (3.2)):

$$A \oplus P_j = \{\mathbf{a} + \mathbf{p_j} | \mathbf{a} \in A, \mathbf{p_j} \in P_j\} \tag{3.1}$$

where A and P_j are the vector spaces representing the shape of the agent and the j^{th} obstacle respectively. It is important to point out here that the agent may not be able to find the minkowski sum due to the sensory limitations, however, the model of the problem presented here does not require the agent to identify all the elements of CS for the collision avoidance. Each pair of vectors, R_j and L_j for each obstacle as illustrated in figure 3.2, can be used in combination with the sum as obtained in Equation (3.1) to form the collision cone (CC). The agent will not collide with any of the obstacles in a future time, if and only if, its current relative velocity vectors lie outside of the CC formed by the edge vectors of all the obstacles in its vicinity [10]. A smaller rectangular region from Figure (3.2) has been chosen to show the formation of the CC w.r.t Obstacle-1 (see Figure (3.3)). Vectors R_1 and L_1 can be used together with the agent's radius, r, to find vectors

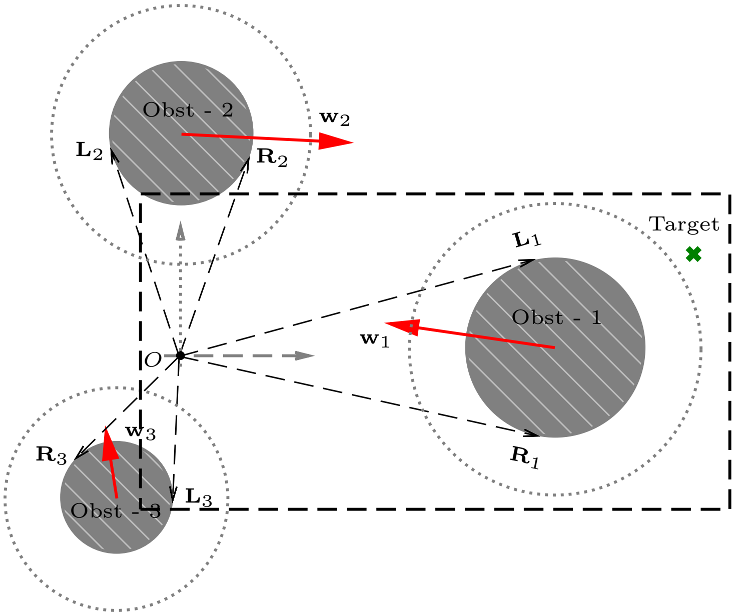


Figure 3.2: Collision spaces represented by the region enclosed by dotted circles around each obstacle

$\mathbf{a}_{R_1}$ and $\mathbf{a}_{L_1}$ respectively in the right angled triangles shown in Figure (3.3).

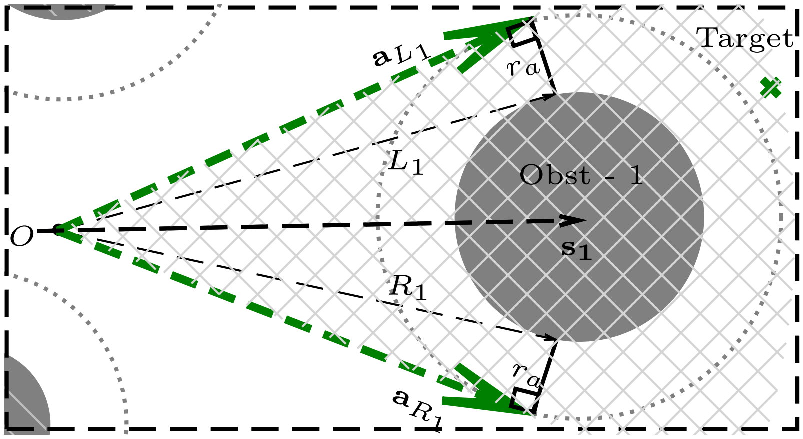


Figure 3.3: Construction of the collision cones using simple geometry

The CC thus formed by these two vectors is convex. Similar CCs can be constructed with respect to all the obstacles detected by the agent. The agent's relative velocity, $\mathbf{c}_j$, w.r.t to each of the j^{th} obstacle should exist in a non-convex space outside of all the CCs in order to avoid collision with any of the obstacles, which is shown mathematically as:

$$\mathbf{c}_j = \mathbf{c} - \mathbf{w}_j \notin CC \tag{3.2}$$

where,

$$CC = \{x : x = \lambda_{R_j}\mathbf{a}_{R_j} + \lambda_{L_j}\mathbf{a}_{L_j} \mid \lambda_{R_j}, \lambda_{L_j} \geq 0\} \quad \forall j \tag{3.3}$$

3.1.1 Method-1

Let $\mathbf{s}_j$ represent the vector for each agent from the center of the agent to the center of each of the j^{th} obstacle. Let any one of the two edge vectors of the CC represented in Figure (3.3) be represented by $\mathbf{E}_j$ for the j^{th} obstacle. It can be seen that the new relative velocity vector of the agent, $\mathbf{c_j}$ w.r.t each of the j^{th} obstacle will generate a collision free trajectory, if and only if, the relative velocity $\mathbf{c_j}$ lies outside of the CCs. In order for this to happen, the following condition must satisfy for the agent w.r.t each of the j^{th} obstacle:

$$cos(\alpha_j) \geq cos(\gamma_j) \tag{3.4}$$

where, α_j is the angle between $\mathbf{s}_j$ and $\mathbf{E}_j$ and γ_j is the angle between $\mathbf{s}_j$ and $\mathbf{c}_j$ $\forall j$. The above constraint can also be written in the following form:

$$\frac{\mathbf{s}_j^T \mathbf{E}_j}{||\mathbf{E}_j||} \geq \frac{\mathbf{s}_j^T \mathbf{c}_j}{||\mathbf{c}_j||} \tag{3.5}$$

In order to find the agent's velocity which covers maximum distance in the direction of it's target such that the agent's relative velocities w.r.t each obstacle lie outside of all the CCs, the following formulation must be solved:

$$\begin{aligned}
&max && \mathbf{c}^T\mathbf{T} \\
&s.t: \\
&\frac{\mathbf{s}_1^T\mathbf{E}_1}{||\mathbf{E}_1||} - \frac{\mathbf{s}_1^T\mathbf{c}_1}{||\mathbf{c}_1||} && \geq 0 \\
&\qquad\ddots && \vdots \\
&\qquad\qquad\frac{\mathbf{s}_j^T\mathbf{E}_j}{||\mathbf{E}_j||} - \frac{\mathbf{s}_j^T\mathbf{c}_j}{||\mathbf{c}_j||} && \geq 0 \\
&\qquad\qquad\qquad\ddots && \vdots \\
&\qquad\qquad\qquad\qquad\frac{\mathbf{s}_k^T\mathbf{E}_k}{||\mathbf{E}_k||} - \frac{\mathbf{s}_k^T\mathbf{c}_k}{||\mathbf{c}_k||} && \geq 0
\end{aligned} \tag{3.6}$$

where, $\mathbf{c}_j = \mathbf{c} - \mathbf{w}_j$ and $\mathbf{E}_j$ represents any of the edge vectors of the CCs, $\forall j$.

The agent will have a collision free trajectory, if and only if, the constraints given in Formulation (3.6) are feasible. The constraints given in Formulation (3.6) forms a non-linear optimization problem. Its solution techniques will be discussed in the subsequent chapters.

3.1.2 Method-2

Using Equations (3.2) and (3.3), following set of constraints can be written for the agent's relative velocity w.r.t Obstacle-1:

$$\mathbf{A_1}\boldsymbol{\lambda_1} \neq \mathbf{c_1} \tag{3.7}$$

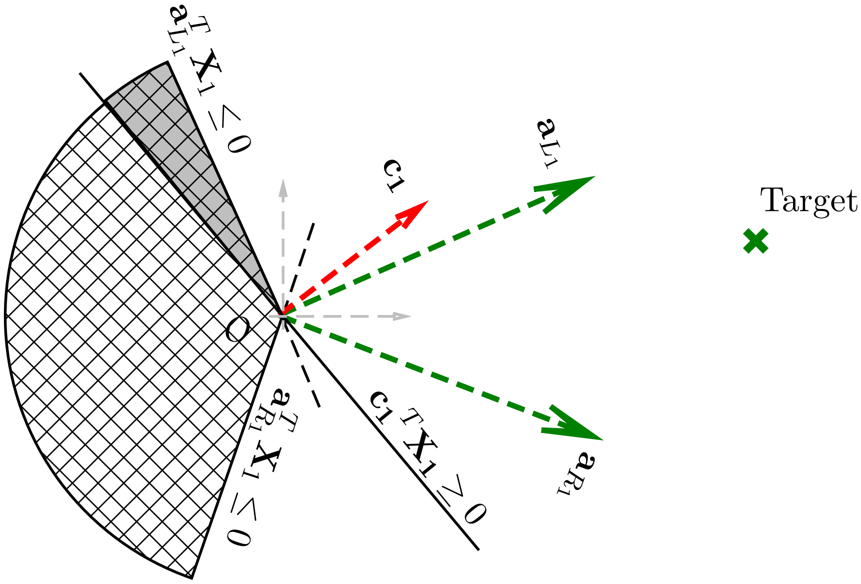
where $\mathbf{A_1} = \begin{bmatrix} \mathbf{a}_{R_1} & \mathbf{a}_{L_1} \end{bmatrix}$, $\boldsymbol{\lambda}_1 = \begin{bmatrix} \lambda_{R_1} \\ \lambda_{L_1} \end{bmatrix}$ such that $\boldsymbol{\lambda}_1 \in R^m$ and the column vectors $\mathbf{a}_{R_1}, \mathbf{a}_{L_1} \in R^n$. The above system holds true, if and only if, the following system does not have a solution:

$$\mathbf{A_1}\boldsymbol{\lambda_1} = \mathbf{c_1}, \quad \boldsymbol{\lambda_1} \geq \mathbf{0} \tag{3.8}$$

According to the Farkas's lemma, if the system in Equation (3.8) does not have a solution, then the following system must have a solution (and vice versa):

$$\begin{gathered} \mathbf{A_1}^T\mathbf{X}_1 \leq 0 \\ \mathbf{c_1}^T\mathbf{X_1} > 0 \end{gathered} \tag{3.9}$$

The feasible space created by the above system of inequalities for Obstacle-1



$\mathbf{a}_{L_1}^T\mathbf{X}_1 \leq 0$
$\mathbf{c_1}$
$\mathbf{a}_{L_1}$
Target
O
$\mathbf{a}_{R_1}^T\mathbf{X}_1 \leq 0$
$\mathbf{c_1}^T\mathbf{X}_1 \geq 0$
$\mathbf{a}_{R_1}$

Figure 3.4: Shaded area shows the set of hyperspaces creating feasible space formed by the two collision cone vectors of Obstacle-1 and vector **c**

is shown in Figure (3.4). Again, it can be seen that the new relative velocity vector of the agent, $\mathbf{c_1}$, will generate a collision free trajectory with respect to Obstacle-1, if and only if, the above set of constraints in Equation (3.9) has a

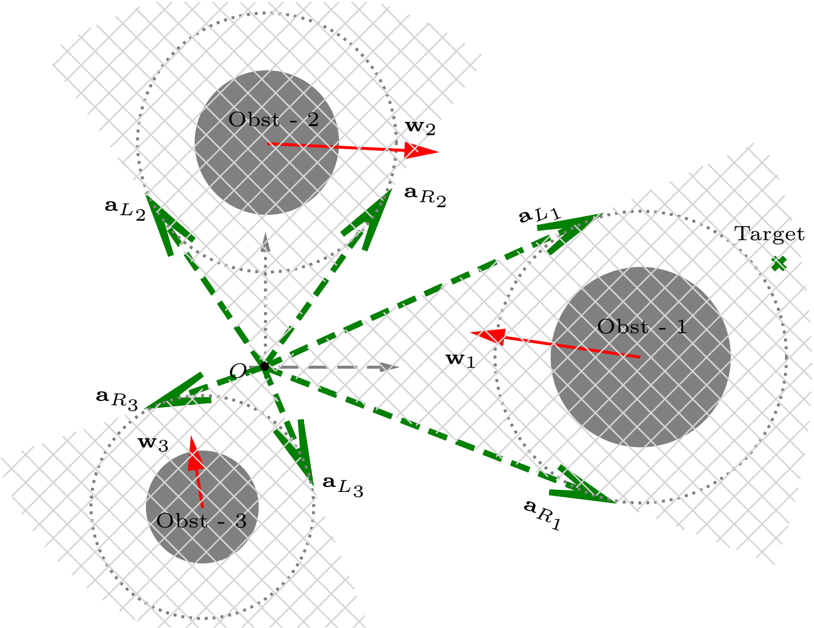


Figure 3.5: Obstacles moving with velocity, $\mathbf{w}_j$, and CCs formed by their respective vectors

solution. The vector $\mathbf{c_1}$ in Figure (3.4) is outside the collision cone and therefore, the hyperspaces formed by the vectors have a solution, i.e,. they form a feasible space.

Equation (3.9) forms a block of set of constraints required to check the feasibility of agent's new velocity with respect to one obstacle. Similar CCs can also be formed for the other obstacles as shown in Figure (3.5). The problem can be formulated as a maximization problem, where the objective is to find the nearest possible agent's velocity vector $\mathbf{c}$ in the direction of its target.

Each CC can be used to form the set of half spaces for each j^{th} obstacle and then formulated in a blocked structured form as presented in Formulation (3.10). The constraints in Formulation (3.10) will be feasible, if and only if, all $\mathbf{c}_j$ are outside their respective collision cones.

$$
\begin{array}{llcr}
max & \mathbf{c}^T\mathbf{T} & & \\
s.t: & & & \\
& \mathbf{a}_{R_1}^T\mathbf{X}_1 & \leq & 0 \\
& \mathbf{a}_{L_1}^T\mathbf{X}_1 & \leq & 0 \\
& -\mathbf{c}_1^T\mathbf{X}_1 & \leq & -\varepsilon \\
& \ddots & & \vdots \\
& \mathbf{a}_{R_j}^T\mathbf{X}_j & \leq & 0 \\
& \mathbf{a}_{L_j}^T\mathbf{X}_j & \leq & 0 \\
& -\mathbf{c}_j^T\mathbf{X}_j & \leq & -\varepsilon \\
& \ddots & & \vdots \\
& \mathbf{a}_{R_k}^T\mathbf{X}_k & \leq & 0 \\
& \mathbf{a}_{L_k}^T\mathbf{X}_k & \leq & 0 \\
& -\mathbf{c}_k^T\mathbf{X}_k & \leq & -\varepsilon
\end{array} \tag{3.10}
$$

where, T is the target vector w.r.t the agent's current position, $\mathbf{c}_j = \mathbf{c} - \mathbf{w}_j, \quad \forall j$, and ε is a small positive scalar. Every 3rd constraint in each block of the above formulation has bi-linear terms which makes the above formulation non-convex.

3.2 Formulation for 3-Dimensional Scenario

Formulation for 3-D scenario can be done in the same way as 2-D. However unlike in 2-D, the CC created by obstacles in 3-D scenarios form non-linear convex conic sets. Such a cone for a single agent and single obstacle is shown in Figure (3.6).

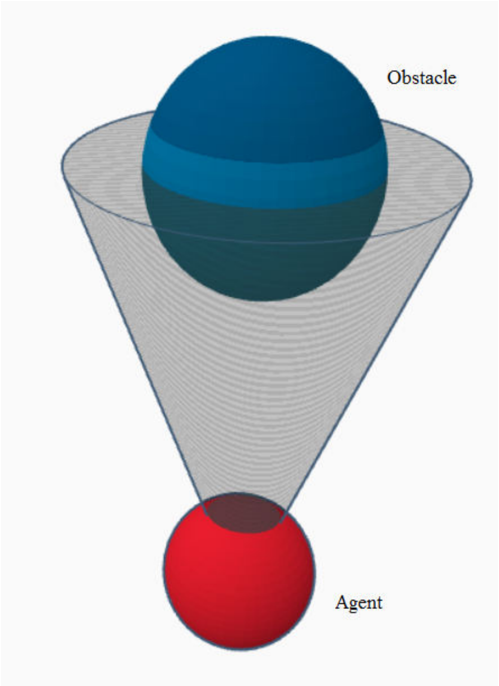


Figure 3.6: 3-D convex collision cone constituting the set of all collision velocities

Method-1 Formulation (3.6) can be used without any change for 3-D scenario as well. However, each of the 3-D non-linear CC form a set of uncountable number of edge vectors out of which any vector may be chosen as $\mathbf{E_j}$ in Formulation (3.6).

For Method-2, the CC of an obstacle may also be approximated by the space enclosed by m number of hyperplanes. The CC for the scenario in Figure (3.6) is approximated with 5 hyperplanes as shown in Figure (3.7). Such CC can also be represented by the convex combinations of the edge vectors $\mathbf{a_q}$ formed by each pair of the intersecting hyperplanes from all the set of hyperplanes for each obstacle in 3-D scenario.

It is assumed for such cases that the edge vectors $\mathbf{a_q}$ $\forall q$ such that $max\{q\} = m$, are already available. Thus, Formulation (3.10) that was developed for the 2-D

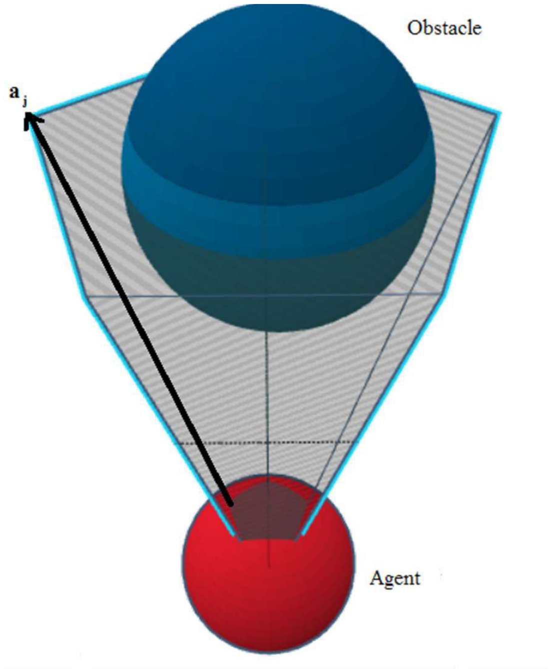


Figure 3.7: 5 hyperplanes approximating CC for one obstacle

case can now be extended for 3-D as follows:

$$
\begin{array}{llllll}
max & \mathbf{c}^T\mathbf{T} & & & & \\
s.t: & & & & & \\
 & \mathbf{a}_{1_1}^T\mathbf{X}_1 & & & \leq & 0 \\
 & \vdots & & & & \vdots \\
 & \mathbf{a}_{m_1}^T\mathbf{X}_1 & & & \leq & 0 \\
 & -\mathbf{c}_1^T\mathbf{X}_1 & & & \leq & -\varepsilon \\
 & \quad\ddots & & & & \vdots \\
 & & \mathbf{a}_{1_j}^T\mathbf{X}_j & & \leq & 0 \\
 & & \vdots & & & \vdots \\
 & & \mathbf{a}_{m_j}^T\mathbf{X}_j & & \leq & 0 \\
 & & -\mathbf{c}_j^T\mathbf{X}_j & & \leq & -\varepsilon \\
 & & \quad\ddots & & & \vdots \\
 & & & \mathbf{a}_{1_k}^T\mathbf{X}_k & \leq & 0 \\
 & & & \vdots & & \vdots \\
 & & & \mathbf{a}_{m_k}^T\mathbf{X}_k & \leq & 0 \\
 & & & -\mathbf{c}_k^T\mathbf{X}_k & \leq & -\varepsilon
\end{array}
\tag{3.11}
$$

Solving the above model gives us the agent's velocity that maximizes the motion of the agent in the direction of its target.

3.3 Kinodynamic Constraints

The agent's new velocity vector $\mathbf{c}$ should be chosen such that it satisfies the agent's kinodynamic constraints. The agent's current acceleration, $\mathbf{g}_a$, and its current velocity, $\mathbf{v}_a$, determines its future velocity, $\mathbf{c}$, in the next iteration after τ time step. Both $\mathbf{g}_a$ and $\mathbf{v}_a$ are bounded and the kinodynamic constraints on acceleration and velocity can be written as follows:

$$v_{min} \leq ||\mathbf{c}||_2 \leq v_{max} \tag{3.12}$$

$$a_{min} \leq ||\mathbf{g}_a||_2 \leq a_{max} \tag{3.13}$$

$$\mathbf{c} = \tau \mathbf{g}_a + \mathbf{v}_a \tag{3.14}$$

where a_{min} and v_{min} are the lower bounds, while a_{max} and v_{max} are the upper bounds on agent's current acceleration and future velocity after τ time step, respectively.

3.4 Solution Techniques

3.4.1 Solution by McCormick Envelopes

McCormick Envelopes are used to obtain convex relaxation of the above formulations. The lower and upper bounds of bi-linear variables are used to find linear hyperplanes that approximate each bi-linear term [54]. The newly transformed linear and convex form of the original problem, thus, guarantees global optimal solution. However, the new approximate linear constraints are typically tight if one of the bi-linear variables is binary.

In order to use McCormick Envelopes for the solution of Method-2 in Formulation (3.10) and Formulation (3.11), $\mathbf{X}_j \quad \forall j$ cannot be considered as binary. Considering $\mathbf{c}$ as binary variable will force the motion of the agent in one of the two directions of the x and y axes only and that too in static environment, $\mathbf{w}_j = 0 \quad \forall j$.

3.4.2 Solution by Heuristic Search

Both Method-1 and Method-2 can also be solved by a heuristic search method. A finite set of the possible values of the agent's new velocity vector $\mathbf{c}$, can be obtained by discretizing the set of the agent's kinodynamic constraints. The problem, thus, converts into an LP which can be used to check the feasibility all its constraints. The velocity vector $\mathbf{c}$ that satisfies all the constraints and maximizes the objective function in Formulation 3.10 can be chosen to obtain an efficient collision free trajectory for the agent. The algorithm for the search of the agent's new velocity vector $\mathbf{c}$ is presented in Chapter 4 in detail.

Although the size of the constraint matrix formed in the above LP increases with the increase in the number of obstacles, the sparse structure of the constraint matrix significantly reduces the solution time. The solution speed also improves as the optimization of the above LP is not the prime intention here rather it is just to check the feasibility, which is usually done in the pre-processing step.

CHAPTER 4

ALGORITHMS

This chapter presents the details of the proposed algorithm. First, to show the basic working of the LP model formulated in the previous section, an algorithm is presented in Algorithm (1), which utilizes a greedy approach to find the feasible velocity vector **c** and may create stalling in some scenarios. Next, the proposed randomized greedy algorithm is presented wherein, a smart approach is used to avoid the phenomenon of stalling while keeping intact the speed of the solution. Some simplifications of the proposed model are also discussed in Section 4.1.6 that may significantly improve the computation times for 2-D and a special case of 3-D scenarios.

4.1 Developing the Algorithm

In the following subsections, the proposed algorithms are presented. The basic algorithm, called Greedy Algorithm is presented first. Then the limitations of the basic algorithm is enhanced and improved algorithm is proposed in the latter subsections.

4.1.1 Greedy Algorithm

The solution procedure of Formulation (3.10) can be seen in Algorithm (1). All the edge vectors, $\mathbf{a}_{R_j}$ and $\mathbf{a}_{L_j}$ are obtained for each j^{th} obstacle in the agent's vicinity. According to Equations (3.12), (3.13) and (3.14), a uniform random sample of the Possible Velocity Space (PVS) is generated. These possible velocity vectors are then stored according to the descending order of their dot products with the relative target vector of the agent, $\mathbf{Z}_a$. Algorithm (1) selects the velocity that produces maximum motion in the direction of the target while avoiding collision with the obstacles in its vicinity in any future time.

As shown in Algorithm (1), the sorted velocity vectors from the sampled PVS are tested for feasibility and the first feasible velocity is chosen as $\mathbf{c}$.

4.1.2 Stalling Phenomenon

The phenomenon of stalling can be explained with the example as shown in Figure (4.1). In order to move as close as possible to the target from position O, the only two best possibilities are to move in either of the direction of the edge vectors, $\mathbf{a}_R$ and $\mathbf{a}_L$. Let us suppose that using Algorithm (1), the agent finds $\mathbf{a}_R$ as its most feasible direction of motion. Hence, O and O' show the agent's initial location and the location it moved in one iteration of τ time-step respectively. The target in this case is closer to the agent w.r.t the center of an imaginary circle

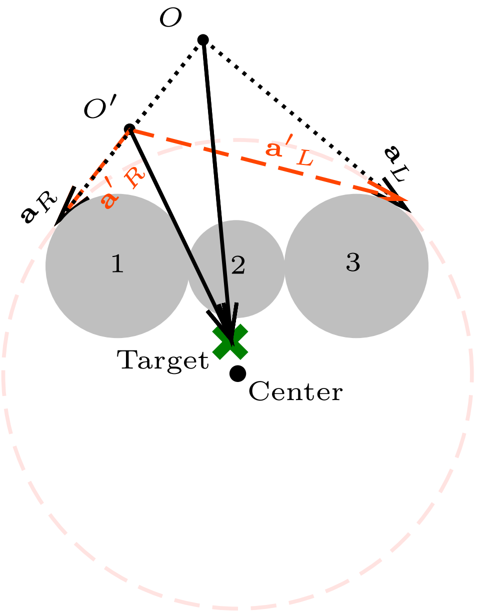


Figure 4.1: Stalling in a single agent - multi static obstacle case

formed by the two tangents, which are represented by the two extreme edge vectors, $\mathbf{a}_R$ and $\mathbf{a}_L$, of the obstacles. In such a scenario, following conditions always hold:

$$\mathbf{a'_R}^T \mathbf{Z'}_a < \mathbf{a_R}^T \mathbf{Z}_a \tag{4.1}$$

$$\mathbf{a'_L}^T \mathbf{Z'}_a > \mathbf{a_L}^T \mathbf{Z}_a \tag{4.2}$$

where $\mathbf{Z}_a$ and $\mathbf{Z'}_a$ are vectors from the agents locations O and O' to the target respectively. As per the above results, the algorithm has found a better direction to move and the greedy approach as explained earlier will results in the agent to change its direction abruptly in the very next iteration. The criterion for the selection of $\mathbf{c}$ in Algorithm (1) will stall the agent's motion. Stalling may be avoided by enforcing the agent not to change the direction of its velocity very rapidly as compared to its previous velocity. Constraining the agent's velocity in this manner may also help to reduce mechanical jerks on the system and produce smooth trajectories.

Thus, the following additional constraint is put on agent's velocity to avoid stalling and rapid changes in the direction of its velocity:

$$\mathbf{c}^{\mathbf{T}}\mathbf{v}_a \geq 0 \tag{4.3}$$

where $\mathbf{v_a}$ is the agent's current velocity.

4.1.3 Proposed Algorithm

For static environments, it can be easily seen that if there exists a collision free path, the $\mathbf{c}$ vector in the direction of at least one of the edge vectors of the obstacles must be feasible. Therefore, all the possibilities of the edge vector directions are first tested for feasibility in the PVS. For this, the agent's acceleration constraints as given by Equation (3.13) are assumed to be such that a nominal velocity with the magnitude of at least, μ_a, is possible to achieve in any direction from the current velocity, $\mathbf{v}_a$.

The edge vectors are sorted, similar to other possible velocity vectors from PVS, in descending order of their dot product with the target vector. If there comes a situation when the new velocity $\mathbf{c}$ in the direction of some edge vector is feasible but does not satisfy the constraint in Equation (4.3), it is neglected and the search continues. The next vector from the sorted edge vectors stack is chosen for feasibility test as given in Algorithm (2).

The algorithm not only finds collision free trajectories in an efficient manner but also avoids stalling situations. However, it is not desirable to use the edge vectors created by the obstacles that are moving away from the agent, in a direction opposite to the direction of the target. Similarly, for cases where no solution is found due to the presence of obstacles all around the agent, the edge vectors of the farthest obstacle may be neglected and the algorithm be repeated until a solution is obtained. If no solution is obtained after considerable reduction of scan radius, then agent will be assigned a zero velocity.

Same strategy can be applied for the multi-agent scenario where each agent considers all the agents around it as obstacles (since the agents are assumed to have no communication protocol between them). Most of the recent trajectory planning algorithms in dynamic environments are computationally very expensive and may require global information to effectively plan the trajectories. Our algorithm is designed to effectively deal with dynamic situations where global information may not be available. Such methods are practical in scenarios where acquiring global information is difficult and expensive, and may require huge onboard computational resources. For cases where global information is available, it may be incorporated in the proposed algorithm by updating the target value for each agent at each time period.

4.1.4 Oscillations in Mutli Agent Scenarios

Velocity Obstacles (VO) in general may render collision free motion, but face the problem of oscillation in multi agent scenarios [53]. Therefore, HRVO had been introduced to avoid the phenomenon of oscillations or 'reciprocal dance'. Similar oscillations are observed in multi agent scenarios for the algorithm proposed in Algorithm (2), as its formulation is developed on the basis of VO as well. A similar technique as proposed in [53] has been incorporated in Algorithm (2) to avoid the oscillations. However, instead of shifting the original VO cones for all other agents as done in [53], we just multiply our Edge Vector Stack (EVS) generated by the left edges of each obstacle with a factor of 0.5. The multiplication reduces the magnitudes of all left EVS by half as compared to their right counterparts. Now, when the overall EVS (right & left) is sorted, the agent's motion to the right edge side of any other agent / obstacle is given priority over the left side while the collision free velocity is being searched for in Algorithm (2). This selection procedure of the collision free velocity direction not only generates smooth trajectories but also saves computational effort. Additionally, in contrast to HRVO, the above approach does not require the agents to have the sensing capabilities to distinguish between an agent and an obstacle.

4.1.5 Special Case

A special case of Algorithm (2) with the above modifications is the case of multi agents moving to their antipodal positions, while initially present on the periphery of a circle, when the agent sensing vicinity includes all the neighbouring agents around it. In such cases, the agents initially come closer to each other while moving to the center of the circle in a spiral shaped trajectory and then keep moving in the circular direction. The circular motion continues until all the agents are in front of their targets with no other agents in between them and each agent can then move straight to its target. It is important to note that such trajectories will always guarantee collision free motion for the case of multi agents present on the periphery of some circle. Figure (4.2) shows the resulting motion in the above discussed special case for 10 agents and compares to the case when the agents have a smaller sensing vicinity (cannot recognize all other agents). The agents move to their antipodal positions with different trajectories.

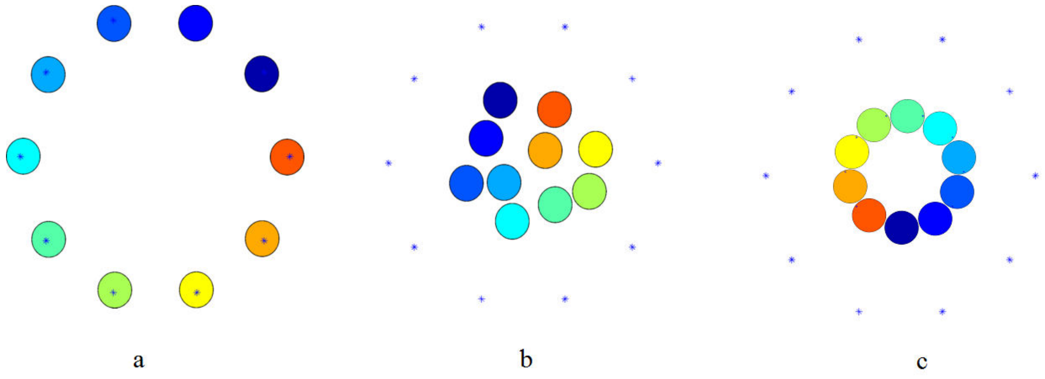


Figure 4.2: (a) Initial positions of the agents. (b) Trajectories with limited sensing. (c) Trajectories with full sensing

The phenomenon explained above may not always happen perfectly as the agents sometimes get too close to each other while moving inwards towards the center of the circle and some random collision free velocity from PVS in Algorithm (2) may be chosen to avoid future collision which disturbs the smooth circular motion in Figure (4.2-c). This phenomenon is shown in Figure (4.3) for 100 agents in circle moving to their antipodal positions.

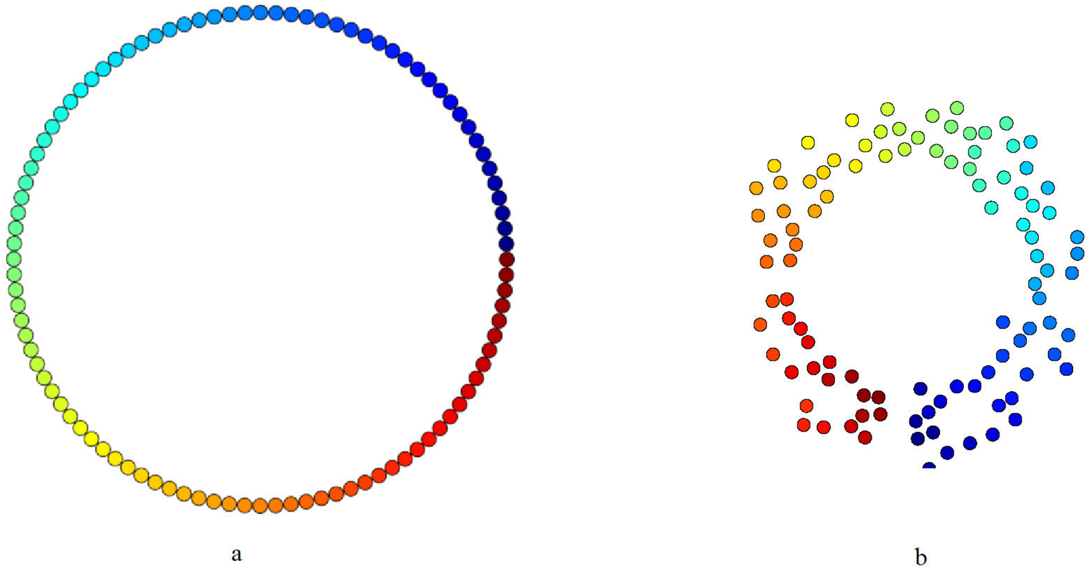


Figure 4.3: (a) Initial positions of the agents. (b) Distorted Circle

Most of the renowned industrial optimization solvers such as CPLEX, employ various pre-processing techniques to reduce the complexity of the problem before starting the optimization iterations. The pre-processing step performs basic checks for problem's feasibility, constraints' redundancy, fixed variables etc. As stated earlier, our proposed algorithm is designed in a way that it requires no simplex iteration and its feasibility check (which is the sole purpose of solving the LP) is always achieved in the presolve step of CPLEX. This definitely gives us the motivation to investigate the presolve methods available in the literature so that the optimization solver may be avoided altogether to further improve computation times for the proposed algorithm.

A comprehensive survey of the presolve techniques is given in [55] where simple presolving methods are first presented to address tightening of constraints, redundancy and in-feasibility by checking the empty columns, fixed variables, singleton rows etc. Forcing and dominated constraints and columns are then identified to check the in-feasibility of the LP. However, our proposed formulation has unrestricted variables and the above discussed presolve methods may not be directly applicable for our case. Similarly, an algorithm to reduce problem dimensions has been presented in [56] where similar issues as in [55] are addressed. Additionally, [57] presents ways to make the constraint matrix sparser and proposes a Primal-Dual method to analyze the LP before applying the Interior Point method.

4.1.6 Primal-Dual Relationship

The dual of the formulation presented for Method-2 has been analyzed in this section. Initially, Formulation (3.10) is converted to its dual to investigate what simplifications can be made to improve computation times of our algorithm. As described earlier that setting the agent's new velocity vector **c** as constant in Formulation (3.10) converts the fomulation into an LP problem. It's dual can be written as follows:

$$min \quad \sum_{j=1}^{k} [0 \quad 0 \quad -\varepsilon]^T \mathbf{U_j} \quad \forall j$$

$$\begin{array}{llll} s.t: & & & \\ \mathbf{D_1}\mathbf{U}_1 & & \geq & 0 \\ & \ddots & & \vdots \\ & \mathbf{D_j}\mathbf{U}_j & \geq & 0 \\ & & \ddots & \vdots \\ & & \mathbf{D_k}\mathbf{U}_k \geq & 0 \end{array} \tag{4.4}$$

where $\mathbf{D_j} = \begin{bmatrix} \mathbf{a}_{R_j} & \mathbf{a}_{L_j} - \mathbf{c}_j \end{bmatrix}$, $\mathbf{U}_j = \begin{bmatrix} u_{R_j} \\ u_{L_j} \\ u_{c_j} \end{bmatrix}$ and all elements of $\mathbf{U}_j \geq 0 \quad \forall j$

Similar to the primal formulation, the above dual is obtained in a blocked structured homogeneous form.

4.1.7 Simplification for 2-D

We know that when dual problem is in homogeneous form, the dual will always be unbounded when the primal is in-feasible. It can be observed that the objective value has a negative sign. In order to make Formulation (4.4) as unbounded, u_{R_j} and u_{L_j} should satisfy their non-negativity constraints when u_{c_j} is a very large number. To analyze this, we take u_{c_j} to the right hand side and rewrite the j^{th} block of constraints from the above dual as follows:

$$\begin{aligned} a_{Rx_j}u_{R_j} + a_{Lx_j}u_{L_j} &= c_{x_j}u_{c_j} \\ a_{Ry_j}u_{R_j} + a_{Ly_j}u_{L_j} &= c_{y_j}u_{c_j} \end{aligned} \tag{4.5}$$

We convert the inequality constraints in Formulation (4.4) to equality constraints because both the constraints in Equations (4.5) can be considered as binding when the j^{th} block of constraints in the dual is unbounded. If we replace u_{c_j} with a very large number say L, the above set of equations will transform as follows:

$$\begin{aligned} a_{Rx_j}u_{R_j} + a_{Lx_j}u_{L_j} &= c_{x_j}L \\ a_{Ry_j}u_{R_j} + a_{Ly_j}u_{L_j} &= c_{y_j}L \end{aligned} \tag{4.6}$$

Using Cramer's rule, we can find the closed form solution for u_{R_j} and u_{L_j} as follows:

$$u_{R_j} = L\frac{\begin{vmatrix} c_{x_j} & a_{Lx_j} \\ c_{y_j} & a_{Ly_j} \end{vmatrix}}{\begin{vmatrix} a_{Rx_j} & a_{Lx_j} \\ a_{Ry_j} & a_{Ly_j} \end{vmatrix}} \tag{4.7}$$

$$u_{L_j} = L\frac{\begin{vmatrix} a_{Rx_j} & c_{x_j} \\ a_{Ry_j} & c_{y_j} \end{vmatrix}}{\begin{vmatrix} a_{Rx_j} & a_{Lx_j} \\ a_{Ry_j} & a_{Ly_j} \end{vmatrix}} \tag{4.8}$$

As we know that L is a very large positive number, its value can be ignored from the above equations as its value is not detrimental in finding signs of the dual variables. Also, it can be proven that the determinants in the denominator in Equations (4.7) & (4.8) will always be non-negative.

Proof. The determinants in the denominator of Equations 4.7 and 4.8 are non-negative.

Let

$$M = \begin{vmatrix} a_{Rx_j} & a_{Lx_j} \\ a_{Ry_j} & a_{Ly_j} \end{vmatrix} = a_{Rx_j}a_{Ly_j} - a_{Ry_j}a_{Lx_j} \tag{4.9}$$

where $\mathbf{a}_{R_j} = \begin{bmatrix} a_{Rx_j} \\ a_{Ry_j} \end{bmatrix}$ and $\mathbf{a}_{L_j} = \begin{bmatrix} a_{Lx_j} \\ a_{Ly_j} \end{bmatrix}$ represent the left and the right edge vectors respectively, formed by the obstacle in a 2-D environment. Lets consider a simple case where $\mathbf{a}_{R_j} = \begin{bmatrix} a \\ 0 \end{bmatrix}$ is in the positive x direction, $a \geq 0$, and it's y component is zero. This is shown in Figure (4.4). Now the vector, $\mathbf{a}_{L_j} = \begin{bmatrix} x \\ y \end{bmatrix}$ will lie on the left of $\mathbf{a}_{R_j}$ if and only if, x is unrestricted and $y \geq 0$. In such case, Equation (4.9) will simplify to:

$$M = \begin{vmatrix} a & x \\ 0 & y \end{vmatrix} = ay - 0 = ay \geq 0 \tag{4.10}$$

Now, if the whole system is rotated in a counter-clockwise direction by some angle θ, $\mathbf{a}_{R_j}$ and $\mathbf{a}_{L_j}$ will transform as follows:

$$\mathbf{a}'_{\mathbf{R_j}} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} a \\ 0 \end{bmatrix} = \begin{bmatrix} acos\theta \\ asin\theta \end{bmatrix} \tag{4.11}$$

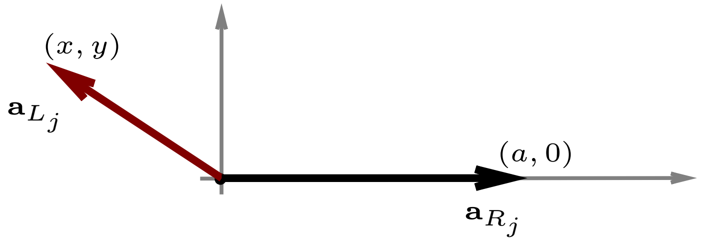


Figure 4.4: Orientation of the right and left edge vectors

$$\mathbf{a}'_{\mathbf{L_j}} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ycos\theta \end{bmatrix} \tag{4.12}$$

From Equations (4.11) & (4.12), the new value of M can be represented as:

$$\begin{aligned} M = \begin{vmatrix} acos\theta & xcos\theta - ysin\theta \\ asin\theta & xsin\theta + ycos\theta \end{vmatrix} &= axcos\theta sin\theta + aycos^2\theta - axcos\theta sin\theta + aysin^2\theta \\ &= ay(cos^2\theta + sin^2\theta) = ay \geq 0 \end{aligned} \tag{4.13}$$

This proves that as long as $\mathbf{a}_{L_j}$ fulfills the above conditions and is on the left of $\mathbf{a}_{R_j}$, M will always be a positive number which means that M is not detrimental in finding signs of the dual variables in Equations (4.7) & (4.8) and can be ignored from the equations. ∎

The resulting equations simplify to just finding out the determinants of two 2×2 matrices for each obstacle. However, it should be noted that the above simplifications will only work for the 2-D case.

The above simplifications can be programmed to run in a loop for all j blocks of dual constraints and the loop is executed by changing values of the agent's velocity until the time any of the dual variable has a non-negative sign for any j^{th} block of dual constraints (meaning those set of dual variables are unbounded which implies that the constraints in j^{th} block of the original problem are infeasible).

4.1.8 Simplification for Special Case

Similar to above, we can find the dual variables in closed form for 3-D scenario. Lets suppose that the surface of the j^{th} spherical obstacle in the vicinity of the agent is approximated by three edge vectors, namely; $\mathbf{a}_{1_j}$, $\mathbf{a}_{2_j}$ and $\mathbf{a}_{3_j}$ as shown in the Figure (4.5). As compared to the 2-D case, where we had the two edge vectors making tangents to the right and left edges of the obstacles, we now have three hyperplanes making tangent to the obstacle from three different points in the 3-D case. The intersection of the three tangent hyperplanes give us three edge vectors as shown in Figure (4.5).

By using the Cramer's rule again, the solution of dual variables can be found as follows:

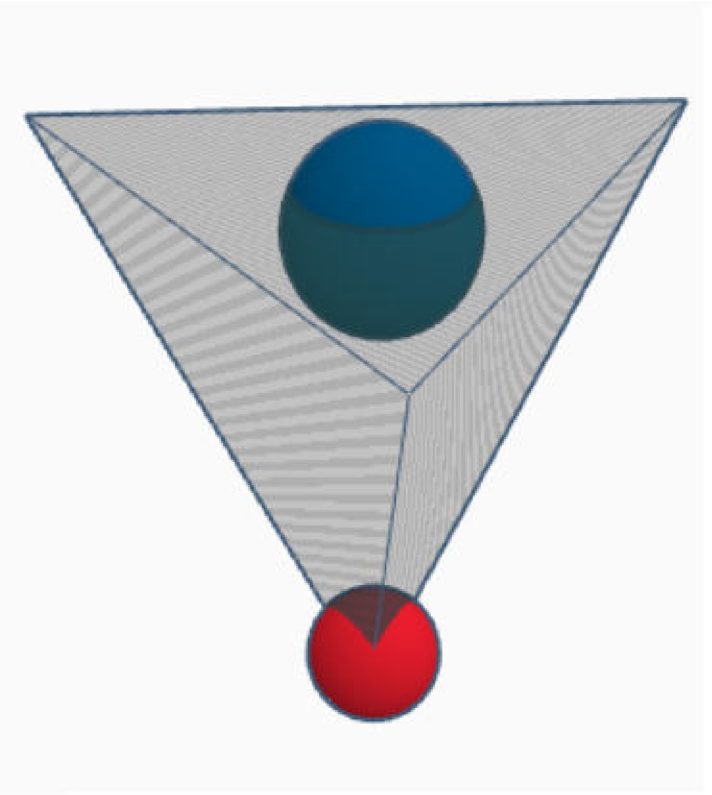


Figure 4.5: Orientation of the right and left edge vectors. The outer light grey sphere represents the CS for this obstacle

$$u_{1_j} = L\frac{\begin{vmatrix} c_{x_j} & a_{2x_j} & a_{3x_j} \\ c_{y_j} & a_{2y_j} & a_{3y_j} \\ c_{z_j} & a_{2z_j} & a_{3z_j} \end{vmatrix}}{\begin{vmatrix} a_{1x_j} & a_{2x_j} & a_{3x_j} \\ a_{1y_j} & a_{2y_j} & a_{3y_j} \\ a_{1z_j} & a_{2z_j} & a_{3z_j} \end{vmatrix}} \tag{4.14}$$

$$u_{2_j} = L\frac{\begin{vmatrix} a_{1x_j} & c_{x_j} & a_{3x_j} \\ a_{1y_j} & c_{y_j} & a_{3y_j} \\ a_{1z_j} & c_{z_j} & a_{3z_j} \end{vmatrix}}{\begin{vmatrix} a_{1x_j} & a_{2x_j} & a_{3x_j} \\ a_{1y_j} & a_{2y_j} & a_{3y_j} \\ a_{1z_j} & a_{2z_j} & a_{3z_j} \end{vmatrix}} \tag{4.15}$$

$$u_{3_j} = L\frac{\begin{vmatrix} a_{1x_j} & a_{2x_j} & c_{x_j} \\ a_{1y_j} & a_{2y_j} & c_{y_j} \\ a_{1z_j} & a_{2z_j} & c_{z_j} \end{vmatrix}}{\begin{vmatrix} a_{1x_j} & a_{2x_j} & a_{3x_j} \\ a_{1y_j} & a_{2y_j} & a_{3y_j} \\ a_{1z_j} & a_{2z_j} & a_{3z_j} \end{vmatrix}} \tag{4.16}$$

Again, the dual variable signs can be calculated to check the feasibility of the primal problem.

4.1.9 Limitations of the Dual Approach

The dual approach may give good results in terms of the computation times in 2-D and in a special case of 3-D environment where the surface of each of the obstacles is approximated using only three vectors. However, approximating each of the obstacle's surface with only three hyperplanes gives a very loose approximation, specially in dense scenarios. Therefore, when more than n number of edge vectors are required to approximate each obstacle in n-dimensional environments, the solution of the dual variables cannot be found in closed form. In such cases, the original algorithm as presented in Algorithm (2) may be used as it is.

4.2 Comparison of Method-1 & Method-2

For 2-D scenario, there are only two edge vectors required for each obstacle. If Method-1 is used in the proposed algorithm, its computational complexity for a particular choice of the agent's velocity will be $O(9n)$ for n number of obstacles as there is a dot product and norm-2 to be solved in each of the constraints in Formulation (3.6).

On the other hand, if Method-2 is used in 2-D scenario, the computational complexity slightly reduces. It has been proven in Section (4.1.7) for Method-2 that if the right and left edge vectors for each obstacle are arranged in a special way in Equations (4.7) & (4.8), the determinants in their denominators will always be positive hence not having any influence on the signs of the dual variables. Therefore, the dual approach reduces the problem of 2-D to just solving a 2×2 determinant for each of the dual variables. Computational complexity to find out the signs for each pair of the dual variables for any particular choice of the agent's velocity will be $O(6n)$ for n number of obstacles.

For 3-D scenarios, Method-1 may be more efficient in terms of computational complexity as compared to Method-2 but assumes that the center vectors $\mathbf{s_j}$ are easily available at hand. The calculation of vectors $\mathbf{s_j}$ is dependant on the set of edge vectors that form the continuous CC for each of the j^{th} obstacle. Accurate calculation of $\mathbf{s_j}$ is crucial for the success of Method-1, and may end up increasing the computational burden of the algorithm.

Algorithm 1: Greedy Algorithm

```
Input:
$r_a \leftarrow$ Agent's radius;
$\mathbf{p}_a \leftarrow$ Agent's current location;
$\mathbf{T}_a \leftarrow$ Agent's target location;
$\mathbf{S}_a \leftarrow$ Agent's velocity sample size;
while $||\mathbf{T}_a - \mathbf{p}_a||_2 \geq \xi$ do
    /* $\xi$ is a small positive scalar                                  */
    $\mathbf{Z}_a \leftarrow \mathbf{T}_a - \mathbf{p}_a$;
    Data: Obtain the edge vectors $\mathbf{a}_{R_j}$, $\mathbf{a}_{L_j}$ and the velocity vector $\mathbf{w}_j$ for
          each $j^{th}$ obstacle present in the agent's vicinity such that,
          $\|\mathbf{q}_j - \mathbf{p}_a\|_2 < \|\mathbf{Z}_a\|_2 \quad \forall j$
    /* Generate uniform random sample of possible velocities
       according to Eq.  (3.12), (3.13) and (3.14)                       */
    PVS $\leftarrow$rand($\mathbf{S}_a$,2);
    PVS $\leftarrow$ sort(PVS,dot(PVS[$i$],$\mathbf{Z}_a$));
    for $i$=1 to rows of PVS do
        $\mathbf{c} \leftarrow$ PVS[$i$];
        Formulate LP as in Eq. (3.10);
        Solve LP;
        if LP is feasible then
            Solution for $\mathbf{c}$ is found;
            break out of for loop;
        else
            $\mathbf{c} \leftarrow 0$
    $\mathbf{v}_a \leftarrow \mathbf{c}$;
    $\varphi \leftarrow$ runtime of current while loop;
    Wait($\tau - \varphi$); /* where $\tau > \varphi$                   */
    /* Update agent's position                                           */
    $\mathbf{p}_a \leftarrow \tau\mathbf{v}_a + \mathbf{p}_a$;
```

Algorithm 2: Proposed Algorithm

```
Input: Same procedure for the parameters $r_a$, $\mathbf{p}_a$, $\mathbf{T}_a$, $\mathbf{S}_a$ and $\mathbf{v_a}$ as in
       Algorithm (1)
while $||\mathbf{T}_a - \mathbf{p}_a||_2 \geq \xi$ do
    Data: Same as in Algorithm (1)
    Repeat steps 3 to 5 of Algorithm (1);
    /* Edge Vector Stack (EVS)                                           */
    EVS $\leftarrow \mathbf{a}_{R_j}, \mathbf{a}_{L_j} \quad \forall j$;
    /* Arrange all EVS stack in descending order of the dot
       product with $\mathbf{Z}_a$                                       */
    EVS $\leftarrow$ sort(EVS,dot(EVS[2j],$\mathbf{Z}_a$));
    for i=0 to rows of (EVS + PVS) do
        if $i = 0$ then $\mathbf{c} \leftarrow \mu_a \times (\mathbf{Z}_a/||\mathbf{Z}_a||_2)$;
        else if $i \leq 2j$ then
            $\mathbf{c} \leftarrow \mu_a \times$ (EVS[$i$]/||EVS[$i$]||$_2$);
        else
            $\mathbf{c} \leftarrow$ PVS[$i$-2$j$];
        Repeat steps 7 and 8 of Algorithm (1);
        if LP is feasible $\wedge\ \mathbf{c}^T\mathbf{v}_a \geq 0$ then
            Repeat steps 10 and 11 of Algorithm (1);
        else
            $\mathbf{c} \leftarrow 0$
    Repeat steps 14 and 17 of Algorithm (1);
```

CHAPTER 5

RESULTS & COMPARISONS

5.1 Experimental Results

The proposed algorithm has been tested for numerous single and multi agent situations with obstacles. The performance has also been compared with other known reactive path planning methods such as APF, Reciprocal Velocities, Hybrid Reciprocal Velocities and ClearPath. In this section, results of the following simulated scenarios are presented to show the efficiency of the algorithm:

1. Static Complicated Scenarios

    (a) Exploration Scenarios

    (b) U-shaped scenarios with Non-Linear Velocity Obstacles (NLVO).

2. Multi Agent Dense and Dynamic Scenarios

    (a) Multi Agents with Randomly Moving Obstacles

(b) Multi Agent scenarios with evenly placed agents on the periphery of a circle and the agents have to navigate to their antipodal positions on the circle.

(c) Multi Agent scenario with evenly placed agents on the periphery of a 3-D helix

5.1.1 Static Complicated Scenarios

Exploration Scenarios

As discussed in Section 4.1.3, situations in which the agent is surrounded by the obstacles generate infeasible space and the algorithm will not find any solution. However, there may exist a solution if the obstacles are not evenly located around the agent. In such cases, the CC of the farthest obstacle is neglected to enable the agent to explore the environment. A spiral maze scenario is shown in Figure (5.1). The algorithm may not find a solution initially in a particular time instant but as the constraints on CC of the farthest obstacles are relaxed, the agent finds its way out of the spiral maze to its target.

A more complicated maze scenario is shown in Figure (5.2) where the agent's sensing vicinity is set to a small number similar to the width of the passage ways of the maze. Reducing the agents sensing radius results in tracking of the wall which enables the agent to navigate to its target. However, such scenarios typically require global path planning methods and since the presented algorithm is a local planning method, it may not be able to find a feasible path in all similar scenarios.

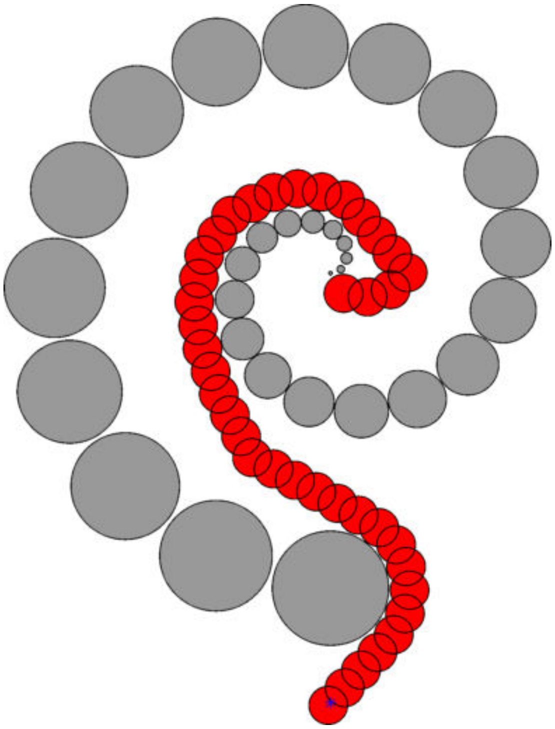


Figure 5.1: Agent (shown in red) found the trajectory to navigate outside the spiral maze

U-Shaped Scenarios with NLVO

Local planning methods are typically greedy in nature and may face phenomenon of stalling as explained in Section 4.1.2. However, the inclusion of an additional constraint given by Equation (4.3) in the proposed algorithm enables it to handle situations, similar to the one shown in Figure (5.4) where some static obstacles are positioned in a U-Shaped structure. Additionally, there are 2 moving obstacles having non-linear velocity profiles. Initially, Obstacle-1 is outside while Obstacle-2 and the agent are inside the U-shaped structure. The y-component of the velocity of Obstacle-1 is constant and is in negative y direction while its x-component accelerates constantly in the positive x direction. Obstacle-2 has a sinusoidal velocity profile which moves it out of the U-shaped structure. The instances of

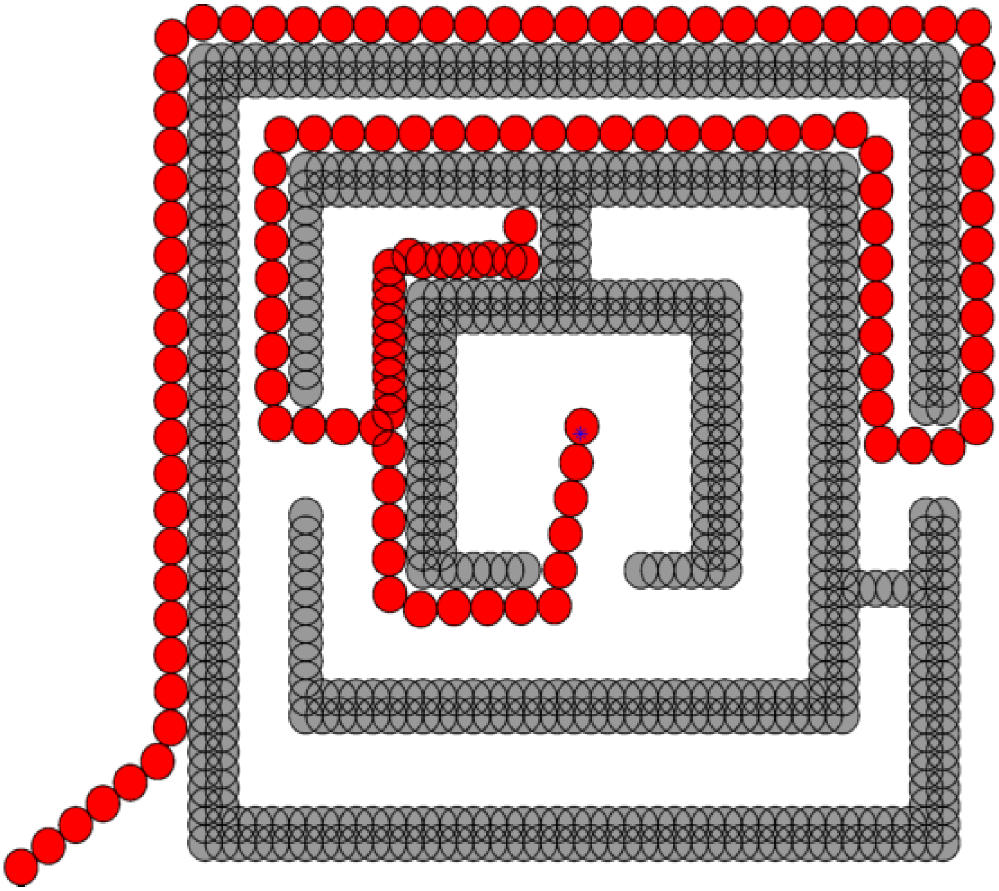


Figure 5.2: Agent navigates to the target located inside the complicated maze structure

the moving obstacles and the agent are printed with the index number of each iteration to show their motion w.r.t time.

Comparison with APF

As discussed in Section 2.2, the APF methods are known to inherently get stuck in local minima situations. Also, the attractive and repulsive forces generated by the target and the obstacles respectively, usually generate trajectories that may not ensure shortest path even in simple scenarios. As can be seen in Figures(5.1) & (5.2), our proposed algorithm generates efficient (shortest path) trajectories even in very complicated scenarios and therefore, outperforms many versions of the APF methods.

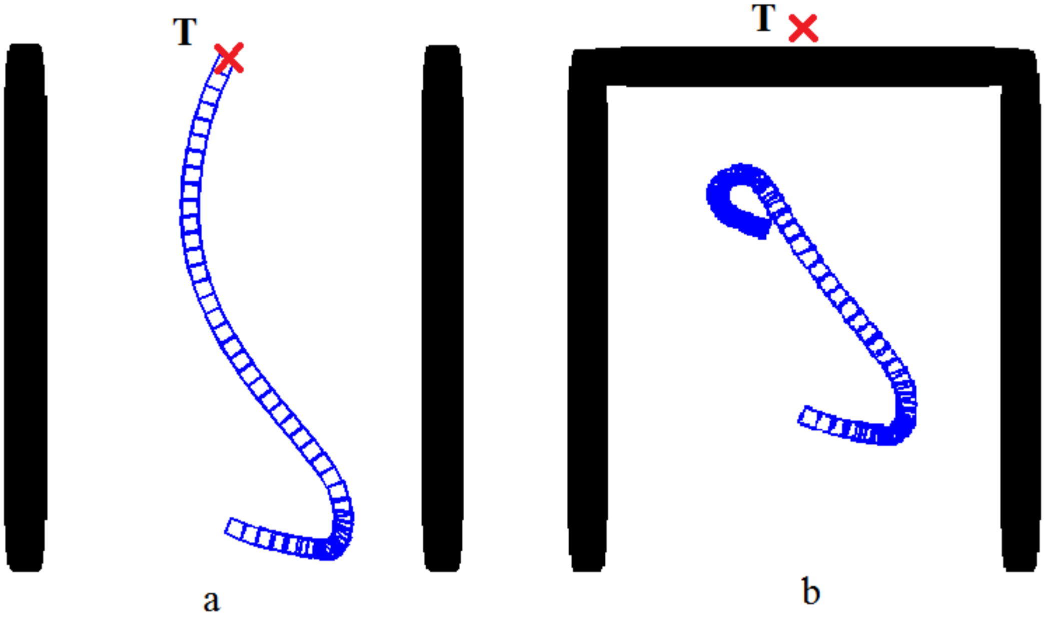


Figure 5.3: (a) Disturbed trajectory of the agent while moving to the target in a close corridor with APF method. (b) APF method causing the agent to get stuck in a local minima situation inside a U-Shaped structure

The agent first tries to avoid collision with Obstacle-1 by changing its direction of motion in the positive x direction. However, as the agent proceeds with its motion after iteration number 5, it slightly changes its direction of motion and reduces the speed to avoid a potential future collision with both Obstacle-1 and Obstacle-2 (see iteration 9-13 in Figure (5.4)) and reaches its target.

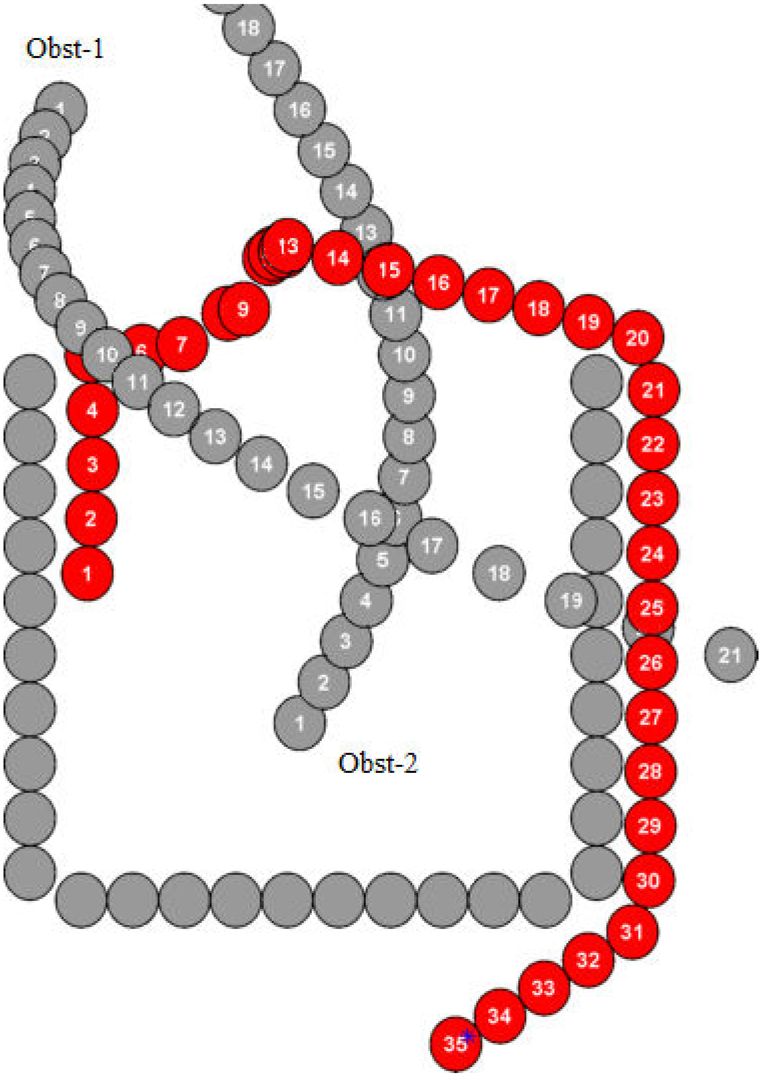


Figure 5.4: Agent navigates to the target located inside the complicated maze structure

5.1.2 Multi Agent Dense & Dynamic Scenarios

Multi Agents with Randomly Moving Obstacles

The algorithm also performs well in dense and dynamic scenarios. One such scenario is presented in Figure (5.5), where 4 agents are present among 80 randomly positioned, randomly sized (radius = 3.5 - 4.5 units) and randomly moving obstacles inside the area formed by a square of sides 100 units each. The agents are positioned at the corner of the square while each of their targets are located at diagonally opposite corners. The agents locally find collision free trajectories and reach to their targets as shown in Figure (5.5).

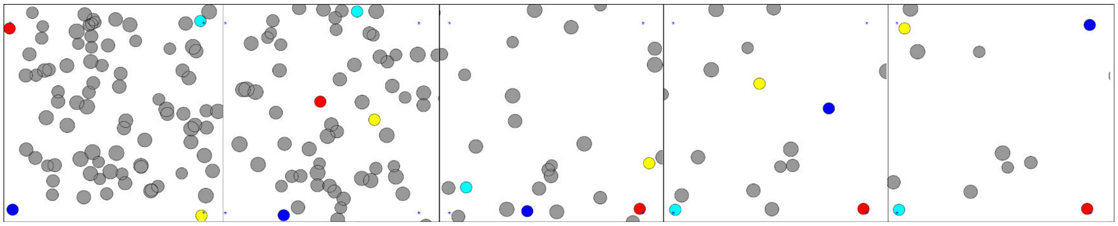


Figure 5.5: 4 agents initially located at corners of a square of sides 100 units each in a dense environment with 80 randomly placed and randomly moving obstacles. The agents locally find collision free trajectories to reach to their targets located diagonally opposite to the agents' initial position

Multi Agents in Circle

The proposed algorithm has also been tested for scenarios typically used as benchmark for multi agent dense situations. The agents, initially located on the periphery of a circle, have to move to their antipodal positions while avoiding collisions from other agents. Figure (5.6) shows one such scenario where 50 agents are symmetrically placed at the periphery of a circle and are shown to navigate to their antipodal positions on the circle. The agents are first seen to converge at around the center of the circle and then effectively avoid collision to navigate to their respective targets.

The number of collisions per time step and the computation times are shown in Figures (5.7) & (5.8) respectively, for multi agent simulations ranging from 10 agents to 1000 agents. First, we take full sensing such that all other agents are considered in the agent's vicinity. The graph shows that as the number of agents

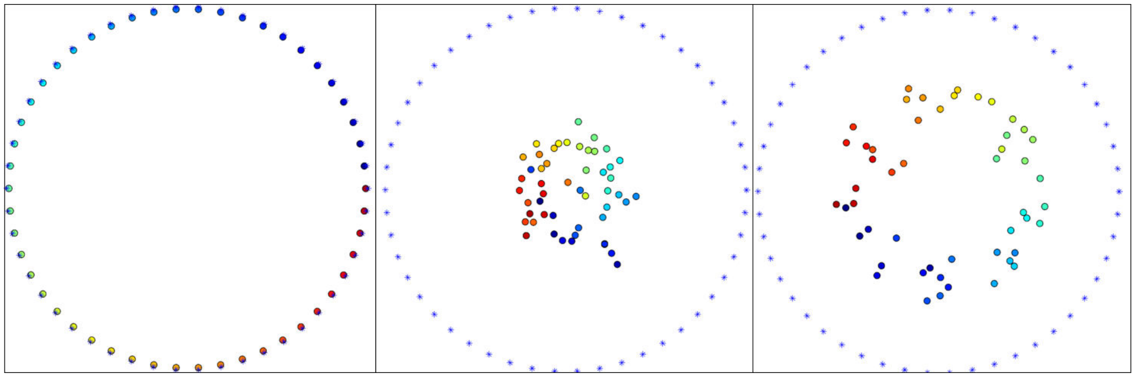


Figure 5.6: 50 agents initially located symmetrically on the periphery of a circle with their targets located at the antipodal position of their initial location.

increase, the computation time per agent also increases.

However, if the agents within some vicinity are considered, then the computational time converges to a specific value with increasing number of agents. Figure (5.8) shows computational times for the circular vicinity of radius 50 units for each agent in all the mentioned multi agent scenarios.

The proposed algorithm experiences very few collisions (almost negligible). The number of collisions are also presented in Figure (5.9):

The algorithm performs very well to avoid the collisions. It can be observed that the average number of collisions in each time step are very small and almost negligible.

Multi Agents in 3-D Helix

A similar experiment to the above has been performed in 3-D case. All the agents are assumed to be spherical lying on the circumference of a helix. The agents have to move to their antipodal positions (exact opposite side on the helix) with collision free trajectories.

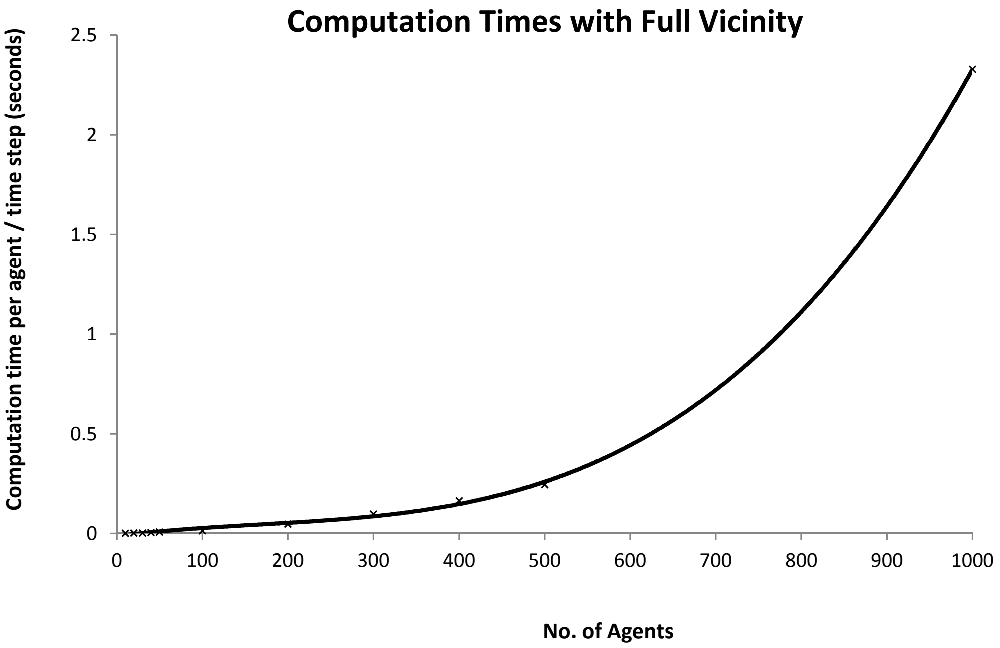


Figure 5.7: Average computation time for each agent per time step with full sensing (all agents are considered in the vicinity)

All the agents' bodies are modeled with 3 edge vectors each. The algorithm is tested with 10, 16 and 30 agents and no collisions have been reported. Illustration in Figure (5.10) shows the setup of the experiment.

Comparison of Efficiency with HRVO & ClearPath

We compare our results of the antipodal scenarios with HRVO and ClearPath. ClearPath is a highly parallel algorithm that exploits certain parallel processing techniques to reduce computation time of the agents [58]. HRVO uses ClearPath together with some modification in the formation of the velocity obstacles cones. The resulting algorithm (HRVO) tries to minimize the oscillations and collisions in the agents' motion with minimum possible computation times [53].

Table (5.1) compares the collision results of the antipodal scenarios for the pro-

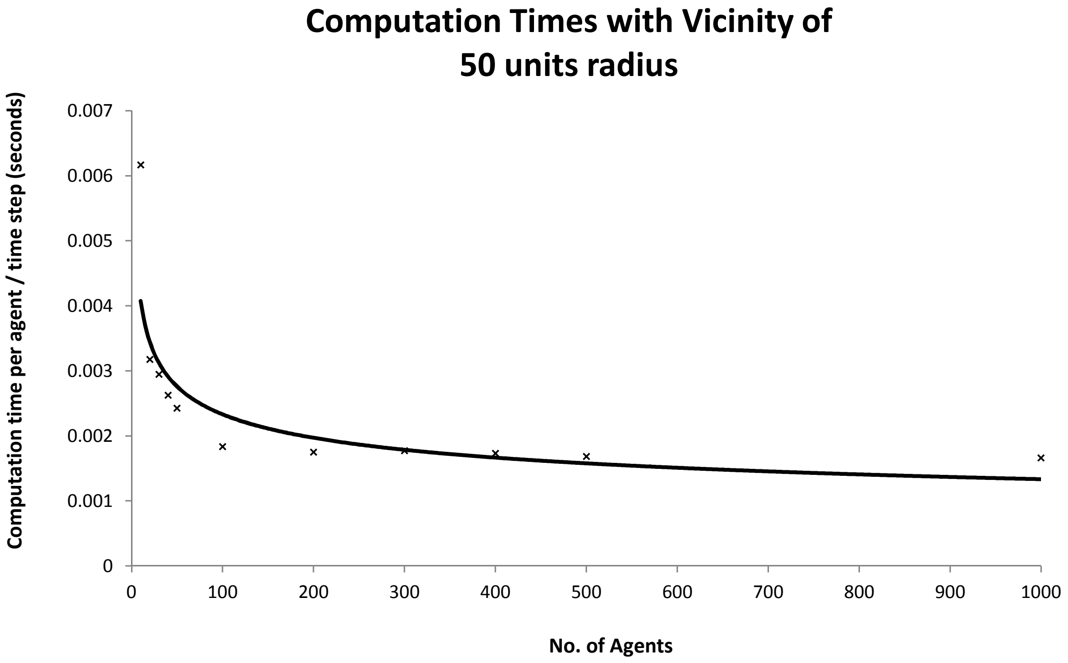


Figure 5.8: Average computation time for each agent per time step with limited sensing (agents within a circular vicinity of radius 50 units) are considered

posed algorithm with HRVO and ClearPath. Our algorithm generates trajectories with significantly reduced collisions (almost negligible).

No. of Collisions per Time Step		
No. of Agents	Proposed Algorithm	HRVO
10	0.0000	0
100	0.0016	0.18
200	0.0010	0.93
300	0.0014	1.93
400	0.0014	3.05
500	0.0019	4.36
1000	0.0019	15.14

Table 5.1: Comparison of Proposed Algorithm with HRVO

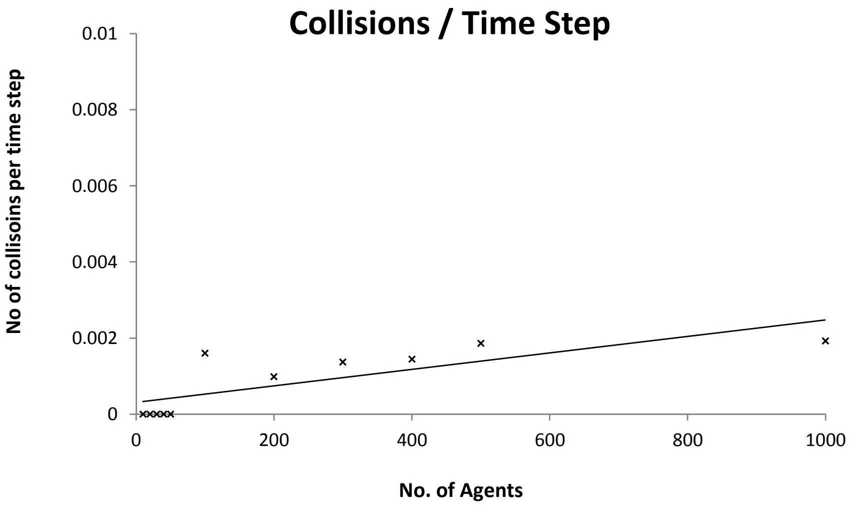


Figure 5.9: Average no. of collisions in each time step

Comparison of Computational Complexity with HRVO & ClearPath

As far as computation times are concerned, Figure (5.7) shows that the computation times of the proposed algorithm are considerably high as compared to those reported in [53] by HRVO. However, it may be noted that it may not be feasible to compare the computation times of our algorithm with HRVO or ClearPath due to the difference in programming languages and code implementations [59]. As for 2-D case, the LP check in Algorithm (2) has been proved to reduce to two simple matrix multiplications. The proposed algorithm is better in terms of the computational complexity as compared to HRVO and ClearPath. In HRVO / ClearPath, first the intersection points for all the lines of all cones are obtained by mathematically solving the simultaneous equations in a loop which is a doubly-nested loop whose complexity will be $O(n^2)$. Then for each of these intersection points are tested to see if they are inside any of the other cones which is another $O(n)$ test. On the other hand, for 2-D case the computational complexity of the proposed algorithm is much better as given in Section (4.2).

Furthermore, with similar parallel processing techniques as in [58], we hope to further improve the results of computation times as reported in [53] with our proposed algorithm.

5.2 Comments on Experimental Results

5.2.1 Collision Avoidance

The proposed algorithm finds the relative velocity of the agent that is outside of all the CCs. Ideally, there should be no collisions when such relative velocities are found. However as per Figure (5.9), the collisions do occur in dense scenarios. The reason is that in dense scenarios when the agent is surrounded by other agents or obstacles from all sides, the algorithm starts to ignore the obstacles farthest away from it in its calculations. This is done uptil a certain minimum sensing radius. If still no collision free velocity is found, the algorithm returns zero velocity which makes the agent to stop in its position and ultimately may collide with the other approaching obstacle. Secondly, two agents very close to each other in a dense scenario may end up with zero velocity in a particular iteration which will be used in the very next iteration input velocities. Both considering each others velocity as zero, may plan a trajectory where collision may happen in the next iteration.

5.2.2 Effect of Surrounding Dimensions

The performance of the algorithm is generally independent of the surrounding environmental dimensions. The only limiting factor related to the dimensions of the environment is the density of the agents and obstacles present in the environment. The collision may happen for dense scenarios as discussed above but other than that, the algorithm is consistent with its performance. The relative velocity of the agent w.r.t its surroundings lying outside of all the CCs will ensure collision avoidance.

5.2.3 Effect of Surrounding Speeds

Collisions may also happen if the time for an approaching obstacle is lesser than the computation time required to find the collision free velocity due to the high speed of the obstacle. This scenario may also occur in dense situations when the agent keeps ignoring the furthest obstacles until some collision free velocity is found and a collision may occur while the agent is busy doing its computations.

Such aspects of the experiment are related to the performance capabilities of the on-board system which is out of the scope of this thesis.

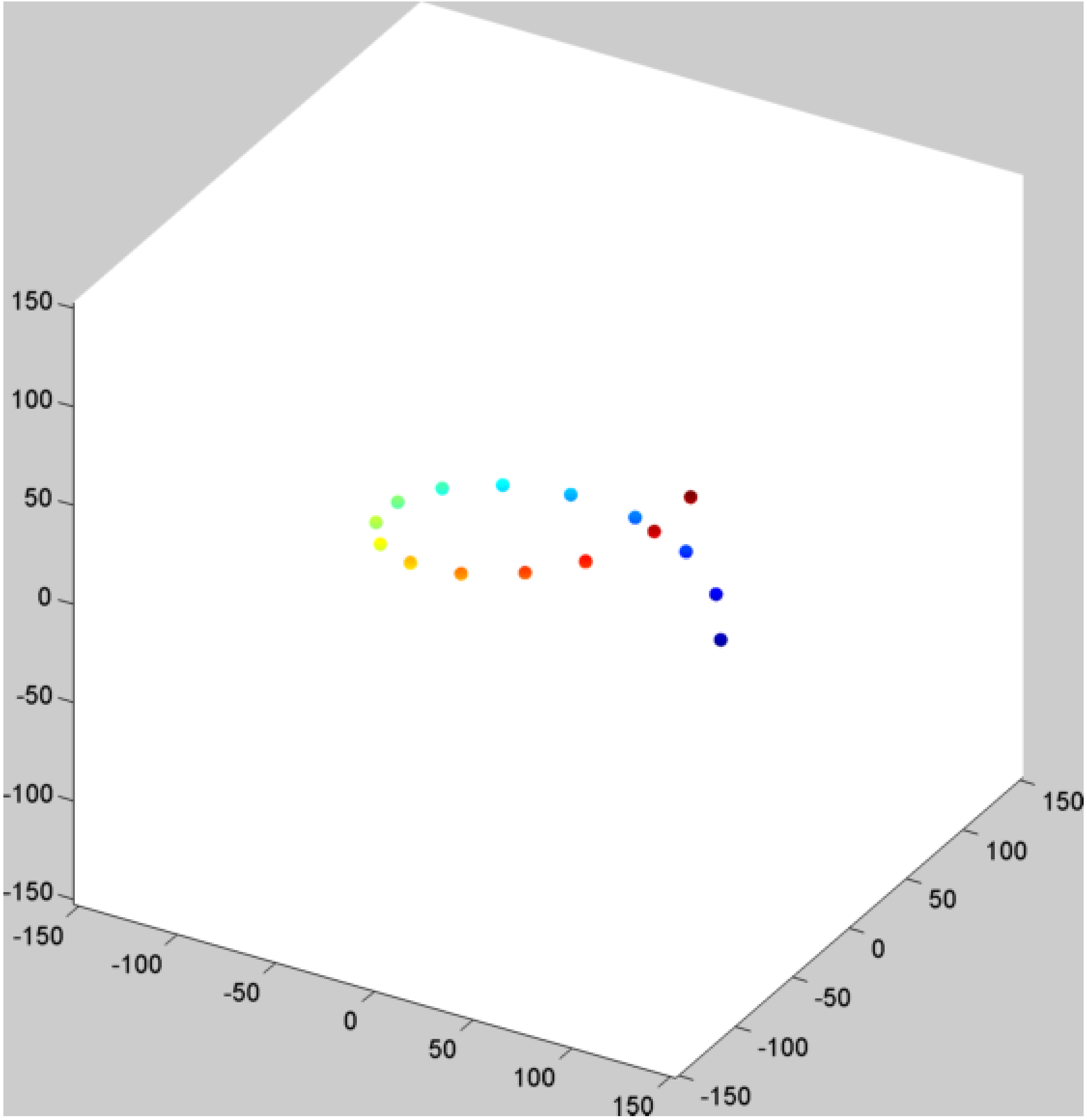


Figure 5.10: 16 agents initially located symmetrically on the periphery of a helix with their targets located at the antipodal position of their initial location.

CHAPTER 6

CONCLUSION

Motion planning is an important area of research for autonomous systems. In spite of the considerable research literature available on motion planning, it still continues to be an active area of research. Primarily, the motion planning for multiple agents in dynamic environment is a challenging task, both computationally and sensing capability wise.

The mathematical programming approaches available in the literature are computationally very expensive and may not be feasible to be run locally on the limited onboard resources of the agent. Secondly, these methods typically require global information which may not be feasible to obtain in most of the real life applications. Some reactive and sampling based methods try to reduce the above computational complexities but may face issues such as finding efficient trajectories (shortest path), getting stuck in local minima situations.

A novel algorithm has been presented in this thesis which not only finds efficient collision free trajectories in multi agent dynamic scenarios but also saves computational effort specially in 2-D and a special case of 3-D scenarios. The algorithm has been tested with different single and multi agent simulated scenarios and compared with the several reactive algorithms already found in the literature. The algorithm performs better in terms of efficiency of trajectories in single complex agent scenarios and significantly reduces the number of collisions in mutli agent dense scenarios.

Some simplifications have also been proposed for 2-D and 3-D cases improving the computational complexity of the algorithm and making it possible to avoid using LP solver in Algorithm (2). Secondly, the algorithm presented in the report assumes all agents and obstacles to be of spherical shape but may easily be implemented for regular shaped objects as well with some possible extensions and approximations.

REFERENCES

[1] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pp. 49–60, IEEE, 1987.

[2] J. V. Breakwell, "The optimization of trajectories," *Journal of the Society for Industrial and Applied Mathematics*, vol. 7, no. 2, pp. 215–247, 1959.

[3] H. Robbins, "Optimality of intermediate-thrust arcs of rocket trajectories," *AIAA Journal*, vol. 3, no. 6, pp. 1094–1098, 1965.

[4] K. Shin and N. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 491–500, 1986.

[5] M. K. Jouaneh, Z. Wang, and D. A. Dornfeld, "Trajectory planning for coordinated motion of a robot and a positioning table. i. path specification," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 735–745, 1990.

[6] V. Rajan, "Minimum time trajectory planning," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, pp. 759–764, IEEE, 1985.

[7] J. Angeles, A. Rojas, and C. S. Lopez-Cajun, "Trajectory planning in robotic continuous-path applications," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 380–385, 1988.

[8] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[9] D. Schoenwald, "Auvs: In space, air, water, and on the ground," *IEEE Control Systems Magazine*, vol. 20, no. 6, pp. 15–18, 2000.

[10] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[12] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1917–1922, IEEE, 2012.

[13] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Robotics and Automa-*

tion (ICRA), 2015 IEEE International Conference on, pp. 5954–5961, IEEE, 2015.

[14] H. Tuy, "Convex programs with an additional reverse convex constraint," *Journal of Optimization Theory and Applications*, vol. 52, no. 3, pp. 463–486, 1987.

[15] P. T. Thach, R. E. Burkard, and W. Oettli, "Mathematical programs with a two-dimensional reverse convex constraint," *Journal of Global Optimization*, vol. 1, no. 2, pp. 145–154, 1991.

[16] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Control Conference (ECC), 2001 European*, pp. 2603–2608, IEEE, 2001.

[17] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[18] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[19] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Robotics*

and Automation (ICRA), 2012 IEEE International Conference on, pp. 477–483, IEEE, 2012.

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

[21] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on robotics and automation*, vol. 8, no. 5, pp. 501–518, 1992.

[22] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, man, and cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[23] J.-H. Chuang and N. Ahuja, "An analytically tractable potential field model of free space and its application in obstacle avoidance," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 5, pp. 729–736, 1998.

[24] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[25] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[26] P. Vadakkepat, T. H. Lee, and L. Xin, “Application of evolutionary artificial potential field in robot soccer system,” in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, pp. 2781–2785, IEEE, 2001.

[27] J. Sheng, G. He, W. Guo, and J. Li, “An improved artificial potential field algorithm for virtual human path planning,” in *International Conference on Technologies for E-Learning and Digital Entertainment*, pp. 592–601, Springer, 2010.

[28] G. Li, A. Yamashita, H. Asama, and Y. Tamura, “An efficient improved artificial potential field based regression search method for robot path planning,” in *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 1227–1232, IEEE, 2012.

[29] D. Hsu, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, pp. 2719–2726, IEEE, 1997.

[30] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[31] S. M. LaValle and J. J. Kuffner Jr, “Rapidly-exploring random trees: Progress and prospects,” 2000.

[32] P. Cheng, Z. Shen, and S. La Valle, "Rrt-based trajectory design for autonomous automobiles and spacecraft," *Archives of control sciences*, vol. 11, no. 3/4, pp. 167–194, 2001.

[33] J. Barraquand, L. Kavraki, R. Motwani, J.-C. Latombe, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," in *Robotics Research*, pp. 249–264, Springer, 1996.

[34] P. H. Borgstrom, M. Stealey, M. A. Batalin, and W. J. Kaiser, "2006 ieee/rsj international conference on intelligent robots and systems, iros 2006," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006*, 2006.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[36] O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, pp. 354–359, IEEE, 2013.

[37] R. Kala, "Rapidly exploring random graphs: motion planning of multiple mobile robots," *Advanced Robotics*, vol. 27, no. 14, pp. 1113–1122, 2013.

[38] C. M. Clark, "Probabilistic road map sampling strategies for multi-robot motion planning," *Robotics and Autonomous Systems*, vol. 53, no. 3, pp. 244–264, 2005.

[39] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pp. 1–8, IEEE, 2013.

[40] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[41] J. Carsten, D. Ferguson, and A. Stentz, "3d field d: Improved path planning and replanning in three dimensions," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3381–3386, IEEE, 2006.

[42] R. Kala, "Multi-robot path planning using co-evolutionary genetic programming," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3817–3831, 2012.

[43] M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, and A. Ammar, "Global path planning for mobile robots in large-scale grid environments using genetic algorithms," in *Individual and Collective Behaviors in Robotics (ICBR), 2013 International Conference on*, pp. 1–8, IEEE, 2013.

[44] I. Châari, A. Koubâa, H. Bennaceur, A. Ammar, S. Trigui, M. Tounsi, E. Shakshuki, and H. Youssef, "On the adequacy of tabu search for global robot path planning problem in grid environments," *Procedia Computer Science*, vol. 32, pp. 604–613, 2014.

[45] I. Châari, A. Koubâa, S. Trigui, H. Bennaceur, A. Ammar, and K. Al-Shalfan, "Smartpath: An efficient hybrid aco-ga algorithm for solving the

global path planning problem of mobile robots," *International Journal of Advanced Robotic Systems*, vol. 11, no. 7, p. 94, 2014.

[46] S. S. Ge, X. Lai, and A. Mamun, "Boundary following and globally convergent path planning using instant goals," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 2, pp. 240–254, 2005.

[47] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "Robust navigation in an unknown environment with minimal sensing and representation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 212–229, 2009.

[48] G.-C. Luh and W.-W. Liu, "An immunological approach to mobile robot reactive navigation," *Applied Soft Computing*, vol. 8, no. 1, pp. 30–45, 2008.

[49] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 902–911, 2009.

[50] R. Sharma, J. B. Saunders, and R. W. Beard, "Reactive path planning for micro air vehicles using bearing-only measurements," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 409–416, 2012.

[51] A. V. Savkin and M. Hoy, "Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments," *Robotica*, vol. 31, no. 02, pp. 323–330, 2013.

[52] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*, pp. 3–19, Springer, 2011.

[53] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[54] "Mccormick envelopes," (accessed Mar 1, 2017).

[55] E. D. Andersen and K. D. Andersen, "Presolving in linear programming," *Mathematical Programming*, vol. 71, no. 2, pp. 221–245, 1995.

[56] A. Brearley, G. Mitra, and H. P. Williams, "Analysis of mathematical programming problems prior to applying the simplex algorithm," *Mathematical programming*, vol. 8, no. 1, pp. 54–83, 1975.

[57] J. Gondzio, "Presolve analysis of linear programs prior to applying an interior point method," *INFORMS Journal on Computing*, vol. 9, no. 1, pp. 73–91, 1997.

[58] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 177–187, ACM, 2009.

[59] T. Andrews, "Computation time comparison between matlab and c++ using launch windows," 2012.

Vitae

- Name: Haseeb Tahir
- Nationality: Pakistani
- Date of Birth: 1st Oct, 1986
- Email: *haseeb.tahir1@gmail.com*
- Permenant Address: House-269, Street-19, GECHS Township, Lahore, Pakistan