Rendezvous of multiple agents amidst obstacles and constraints

A THESIS

submitted by

VUNDURTHY PARVATHISWARA BHASKAR

for the award of the degree

of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2019

THESIS CERTIFICATE

This is to certify that the thesis titled **Rendezvous of multiple agents amidst obstacles and constraints**, submitted by **Vundurthy Parvathiswara Bhaskar**, to the Indian Institute of Technology, Madras, for the award of the degree of **Doctor of Philosophy**, is a bonafide record of the research work done by them under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. K. Sridharan
Research Guide
Professor
Dept. of Electrical Engineering
IIT Madras, Chennai-600036.

Place: Chennai.
Date: 28^{th} of May, 2019.

ACKNOWLEDGEMENTS

I take this opportunity to express my gratitude to my thesis advisor Prof. K. Sridharan, for his exemplary guidance, monitoring and constructive suggestions throughout the course of this project. He always made sure that, everything necessary for the project, be it equipment or knowledge, was readily available. His sense of timing and his vast area of expertise have inspired me in so many ways. It has been a wonderful and enlightening experience to work under him.

I would like to thank Prof. Arun D Mahindrakar, for his valuable feedback, innovative suggestions and his two cents on the concept of PhD itself, very early on during my research. His constant support until the synopsis stage has been of immense help to me.

I would like to thank Prof. Ramkrishna Pasumarthy for giving me a reality check a number of times during my PhD and his humorous suggestions that stopped me from taking a few wrong steps during my time here.

I would like to extend my gratitude to Prof. Bharath Bhikkaji and Prof. Sandipan Bandyopadhyay for offering a few very crucial courses that shaped my thought process in approaching various open ended problems.

I would like to extend a special thanks to Prof. Viswanath Reddy, whose expertise in Game Theory and the caliber to deliver it in a student friendly manner has been of great help to me. His passion towards this field has helped me inculcate the same and utilize the knowledge in one of the chapters of this thesis.

I would like to thank my seniors Dr. Vikramkumar Pudi, Dr. Srinivasu Bodapati and Dr. Vijay Muralidharan for paving the path in this long journey of PhD. Their timely advises, practical examples and humorous ways of handling pressure have helped me quite a few times.

I would like to thank my colleague Dr. Krishna Chaitanya Kosaraju whose light personality and cheerful attitude, in addition to his expertise in research has

made the life at IIT Madras far more enlivening. I would also like to thank Lokesh Chintala for being an instrumental part in the fabrication of the humanoid utilized in chapter 7. Finally, Jeshma Fahad deserves a special thanks for supporting me during my various seminars with her unique style of constructive criticism.

I would like to extend my thanks to all my friends Aravind Devarakonda, Arun Anand, Bhargava S Boga, Dadavali Dudekula, Rajesh KSV, Raju Dasari and Roopesh Kumar Raya who made me the person that I am today. They continue to positively influence my life and will always be my anchors.

I would also like to extend my thanks to my colleagues Sharad Kumar Singh, Yashrajsinh Parmar, Akshit Saradagi, Nagachandrika Reddy and all the students from the associated labs here at IIT Madras. Being part of their wonderful and enlivening groups has helped me in the toughest of days. A special thanks extends to Rama Srinivas, Seshasaye Behera, Vivek Vysyaraju, Marv and Priya for making my life at IIT Madras as fun as it could be.

My family has worked really hard their entire lives to mold me into the person I am today. I would have broken down a number of times, if not for their timely help and support. I extend my heartfelt thanks to my parents Srinivasa Rao and Bhu Lakshmi, my brother Pavan Chandra and my uncle Ramachandra Rao for everything they have done and more. A special thanks extends to my brother for fulfilling various roles in many tough situations and offering his untiring support throughout. In addition, I would like to especially thank my fiancee Udaya Sree Datla, for believing in me. Her practical approach towards life and her compassionate nature has always been very inspiring.

Finally, I would like to thank the almighty God for giving me the strength, knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily.

Chennai

May 2019 Vundurthy Parvathiswara Bhaskar

ABSTRACT

KEYWORDS: Rendezvous, Distance Constraints, Obstacles, Shortest Paths, Hardware efficient Algorithms, Implementation

Teams of robots are typically employed for various purposes to complete a task faster than a single robot. Robots that need to interact to accomplish some task may not, however, be physically at the same location. They may be distributed and need to come together. This is commonly referred to as the *rendezvous* problem in the literature. The rendezvous problem has been studied by researchers in computer science, economics, control and other domains with different assumptions and objectives.

Our interest is in rendezvous of multi-agent systems amidst various constraints and obstacles. In this thesis, we impose constraints on the individual and total distance travelled by all the agents and compute an optimal rendezvous location in the plane amidst various obstacles. We present hardware-efficient algorithms to compute these optimal locations and discuss implementation via locally fabricated mobile robotic setup. While rendezvous has been a well studied area, prior work on computing optimal locations for rendezvous, especially amidst obstacles and distance constraints is limited.

Further, we impose constraints on the time taken for rendezvous and once again compute an optimal location that facilitates rendezvous of multiple agents amidst arbitrary number of obstacles in minimum time. We also explore the notion of competitive rendezvous where the agents compete to achieve rendezvous at two different locations. In other words, we explore the effect of an adversary that attempts capture of an agent (the first rendezvous) while the agent itself is interested in being rescued by a fellow agent at an alternate rendezvous location. Finally, we discuss a sensor-based rendezvous of two heterogeneous robots: a mobile robot and a bipedal robot. Each of these works is supported by experimental verification of the proposed algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATIONS AND ABBREVIATIONS

NOTATIONS

k	Number of agents or robots
n	Number of obstacles
c	Number of vertices per obstacle
$P_1, P_2, \cdots, P_k$	Agent or robot locations
$d_1, d_2, \cdots, d_k$	Weights on agent locations
$t_1, t_2, \cdots, t_k$	Times lost by agents in reaching P_i
$Q_1, Q_2, \cdots, Q_k$	Last turn locations
$O_1, O_2, \cdots, O_n$	Polygonal obstacles
$O_i^1, O_i^2, \cdots, O_i^c$	c vertices of the obstacle O_i
$Z_1, Z_2, \cdots, Z_n$	Reflective zones
D	Delivery Agent
v_d	Maximum speed of delivery agent
R	Rescue Agent
v_r	Maximum speed of rescue agent
d_r	Limiting distance of rescue agent
A	Adversarial agent
v_a	Maximum speed of adversary
d_a	Limiting distance of adversary
C	Dominance circle
S_{da}	Dominance region of D with respect to A
$\neg S_{da}$	Dominance region of A with respect to D
S_{dr}	Dominance region of D with respect to R
$\neg S_{dr}$	Dominance region of R with respect to D
S_h	Safe region for rendezvous
T	Meeting location
T_t	Time for rendezvous

R_t	Time Optimal Rendezvous Point (TORP)
A, B	Heterogeneous agents
T_A, T_B	Transmitters on A and B
R_n	Reflective Index

In the absence of obstacles

P_M	Minimax point
P_M^W	Weighted minimax point
P_{ij}	Weighted minimax point of agents P_i and P_j
P_{ijk}	Weighted minimax point of agents P_i, P_j and P_k
P_S	Minsum point

In the presence of obstacles

P_G	Rendezvous point satisfying minimax distance constraint
P_U	Rendezvous point satisfying minsum distance constraint
CH	Convex Hull of agent locations and obstacle vertices

ABBREVIATIONS

TORP	Time Optimal Rendezvous Point
AGV	Automated Guided Vehicle
DS	Distance Sensor
VS	Vision Sensor
WC	Wireless Communication
CHT	Circular Hough Transform
RF	Radio Frequency

CHAPTER 1

INTRODUCTION

Autonomous mobile robots have been extensively used since approximately the 1970s in applications involving transfer of goods from one point to another. As a robot moves towards its destination, its primary objective is to avoid collisions with objects (machines, furniture etc.) along the way. Considerable research in mobile robotics has focussed on planning motions avoiding obstacles. Different assumptions about the environment and the capabilities of the robot have been made. Early works (Lozano-Perez and Wesley (1979) and Schwartz and Sharir (1983)) have assumed the existence of complete geometric models. Research in the late-1980s (Lumelsky and Stepanov (1987), Rimon and Koditschek (1992)) has been directed to sensor-based planning with limited or no prior knowledge of the geometry of the objects in the environment. Restrictions on capabilities of the sensors have also been investigated. A detailed description of various approaches for planning is available in Latombe (1991).

Since the late 1990s, research in autonomous mobile robotics has moved along another direction. Several research groups have explored the power of teams of mobile robots operating in indoor and outdoor environments to perform different tasks. A team of robots, in general, can accomplish a task faster or more effectively than a single robot. However, several challenges arise when a team of robots is involved. The challenges are no longer limited to collision avoidance among the robots or obstacles. For example, the members that are part of a team may not be in the same physical location to begin with. Further, a team may have to get into a specific "structure" (or formation) to accomplish a certain task. Also, a team may be obstructed in some way by an autonomous intruder. Keeping these in view, three categories of problems have emerged. The first category is *formation*. Early work (Yamaguchi (1999)) involved design of control strategies to coordinate the motion of multiple holonomic mobile robots to capture/enclose a target via troop formations. A number of variations of the basic formation task have been studied (Fax and Murray (2004), Ren and Beard (2005)). The second category of

problems studied is *rendezvous* of a set of robots that are geographically separated. It is worth noting that sensors on a robot may not be powerful enough to allow uniting with another robot readily to form a team. Early work on rendezvous of a set of mobile robots with limited visibility is described in Ando *et al.* (1999). Extensions to this have been explored in Ganguli *et al.* (2009) and Yu *et al.* (2012). In both formation as well as rendezvous, the robots are engaged in cooperative activity. In other words, the robots move together or attempt to unite. A detailed account of recent work on formation and rendezvous is provided in Bullo *et al.* (2009). A third category of problems (arising especially in defence and security applications) involving multiple mobile robots is where there is an *adversary* or a *tracker* to thwart/monitor the movement of other robots. This category of problems has been generally studied in the domain of game theory (Murrieta-Cid *et al.* (2007)).

The focus of this thesis is on *rendezvous of mobile robots*. In order for robots to successfully unite, communication among them is essential. The term *agents* is generally used to refer to autonomous vehicles that communicate with each other. We will use the term agents henceforth and describe the work in this thesis with reference to agents.

1.1 Motivation

While considerable work has been done on the rendezvous problem in general and variants, some aspects that have a direct impact on the energy consumed by the system during rendezvous have not been explored. In particular, the distance travelled by the agents before they meet can directly impact the total energy consumed.

The work described in this thesis is motivated by the following questions: *Does there exist an optimal location in the plane where the agents can meet to perform a collaborative task? If such a location exists, how is it influenced by adding obstacles into the environment? Finally, can the algorithms designed to compute optimal locations be implemented with limited hardware on-board?*

The optimality of the location is in terms of the distance travelled by the agents.

Several notions of distance are possible (and these will be discussed further in section 1.2). Computing an optimal location (and moving to the same) has direct applications in minimizing fuel consumption and the time taken to commute.

Algorithms for rendezvous with constraints on distance are also applicable in defence scenarios where two or more friendly groups intend to meet while ensuring that they optimize their individual or collective distances travelled. The groups can treat hostile/hazardous areas as obstacles. A similar application exists in a busy city environment where computation of an optimal location (amidst various constraints) allows rendezvous of a group in minimum time. Sometimes, rendezvous has to take place in the presence of an adversarial entity which tries to prevent the meeting of two or more agents. In the next section, we explain in detail the various problems addressed in this thesis and the relevant terminology.

1.2 Rendezvous Amidst Obstacles and Constraints

One approach to obtain an optimal location for rendezvous is by enforcing a constraint on the Euclidean distance travelled by each agent. The first problem we address in the thesis concerns minimizing the maximum distance (henceforth referred to as the *minimax distance*) travelled by any agent prior to rendezvous. The distance travelled by an agent is directly related to the energy consumed and by minimizing the maximum distance, the agents effectively optimize the maximum loss in energy during commute. Consequently, agents with a limited power resource, such as autonomous mobile robots, benefit tremendously.

In a practical scenario, however, the agents operate in an environment *where a number of objects obstruct the paths of the robots*. It is thus of interest to compute an optimal location in the plane while imposing a minimax distance constraint as the agents travel *amidst these obstacles*. Fig. 1.1 illustrates the locations of three agents P_1, P_2 and P_3 amidst three quadrilateral obstacles O_1, O_2 and O_3. Two rendezvous points R_1 and R_2 are arbitrarily chosen for illustrative purposes and the respective distances from the agent locations are shown. Assuming each robot is small and can therefore be represented by a point, the distance between any

two points is evaluated as their respective Euclidean 2-norm.

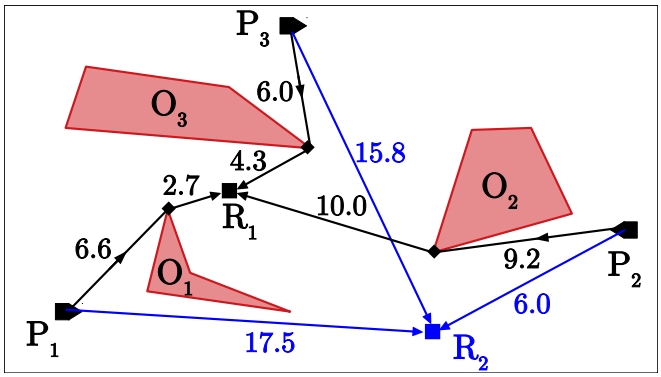


Figure 1.1: Two rendezvous points R_1 and R_2 illustrating minimax distance constraint. R_2 has a lower maximum of distances value compared to R_1.

In the presence of obstacles, the distance from an agent location to a rendezvous point is the length of its shortest path. For instance, when attempting to meet at R_1, the agent at P_1 travels a distance of 9.3 units ($= 6.6 + 2.7$) while incorporating the turn at the vertex of obstacle O_1. Similarly, the distances from P_2 and P_3 are 19.2 and 10.3 respectively. Thus the agent at P_2 travels the maximum distance prior to its arrival at R_1. However, none of the obstacles obstruct the paths of agents when attempting to rendezvous at R_2 where the agent at P_1 travels the maximum distance of 17.5 units. The minimax distance criterion prefers rendezvous at a location with a lower maximum of distances value, which in this case, turns out to be at R_2.

The challenge that arises is efficient search of the plane to obtain a unique location that minimizes the maximum of distances travelled from each of the agent locations. Further, an algorithm that is implementable on autonomous mobile robots needs to be developed that could handle an arbitrary number of agents and obstacles.

While a minimax distance constraint optimizes the distances with respect to each individual agent, an alternate criterion involves minimizing the sum of distances of travel (henceforth referred to as the *minsum distance constraint*). This is the second problem addressed in this thesis. The minsum constraint optimizes the total energy utilized in commute prior to rendezvous. This constraint is especially useful when the agents belong to the same group and the cumulative effect of energy consumed is of interest.

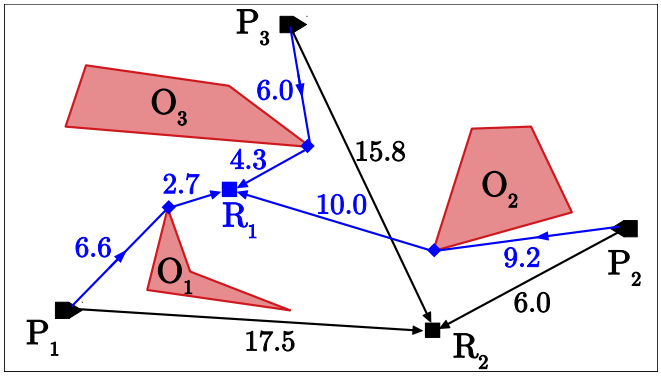


Figure 1.2: Two rendezvous points R_1 and R_2 illustrating minsum distance constraint. R_1 has a lower sum of distances value compared to R_2.

Fig. 1.2 illustrates minsum distance constraint for the same locations of agents and obstacles as in Fig. 1.1. However, the sum of distances to R_1 is 38.8 units while the same to R_2 is 39.3 units. As a result, the minsum distance constraint prefers rendezvous at R_1 over R_2. It is worth noting that the minimax and minsum distance constraints need not lead to the same location. Further, the challenges offered by the two problems are unique in their own way and it is thus of interest to study the development of a hardware-efficient algorithm for computing the optimal location with respect to the minsum distance constraint with an arbitrary number of obstacles.

It is worth mentioning that these problems are different from the path finding problem since the destination is known in the latter. In the computation of an optimal location (for solving the minimax and minsum problems), it is necessary to scan the entire plane where the shortest path computation is merely used as a substep (as illustrated in figures 1.1 and 1.2) in evaluating distances between agent and rendezvous locations.

Thus far, the rendezvous of agents was restricted to the areas in the plane that are not occupied by obstacles. Scenarios where an adversary deliberately tries to prevent rendezvous of agents offer additional challenges since the admissible area for rendezvous is no longer determined by the boundary of obstacles. We refer to this problem as *adversarial rendezvous* and identify regions (and an optimal location) where rendezvous is feasible even in the presence of an adversary. In this problem, cooperation and conflict exist simultaneously and the computation of optimal location is performed in real time utilizing the positions and speeds of

agents and the adversary.

Handling an adversary calls for determining regions in the plane where a specific agent dominates or arrives earlier than others. Fig. 1.3 illustrates the notion of these dominance regions for two agents R and E travelling with unequal speeds v_r and v_e respectively, where $v_r > v_e$. The collection of all the locations where E reaches prior to R turns out to be the region internal to the circle C and is termed as the dominance region of E. On the other hand, the region external to C is the dominance region of R. The curve that separates these dominance regions is referred to as the dominance curve. In Fig. 1.3, dominance curve happens to be a circle C with radius r_{re}. This circle is often referred to as the Apollonius' circle in literature and is defined as the set of points that have a given ratio of distances ($\frac{v_r}{v_e}$) to two given points (agent locations R and E respectively). The agent locations thus act as the foci of this circle.

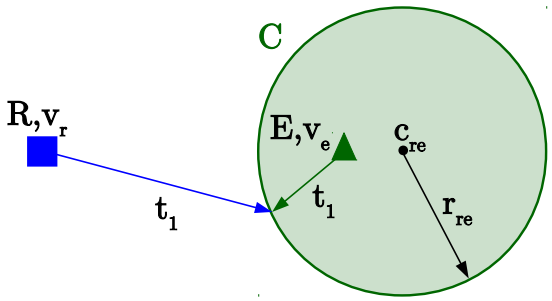


Figure 1.3: Dominance regions of two agents R and E travelling with speeds v_r and v_e. The region shaded in green is dominated by E and vice-versa.

When R acts as an adversary, it is desirable to identify a rendezvous location within the dominance region of E, since any point external to it is dominated by R and ensures capture. One approach to determining C is identifying all the locations in the plane where R and E arrive simultaneously, generating the Apollonius' circle. Such a construction inherently assumes communication among the agents and adversary. This is, however, not practical since an adversary would typically acquire information about the agents via some sensing mechanism. We thus explore the interaction between a vision-guided adversary and mobile agents while allowing communication only among the latter. The contributions in this regard are discussed in Section 1.3.3.

Having studied the effect of distance and speed on rendezvous of agents, we also

explore the problem of minimum time rendezvous of an arbitrary number of mobile agents. We study this in the presence of polygonal obstacles while motivating the need for efficient algorithms that allow a quick recomputation of the optimal location. Finally, we briefly study a sensor based rendezvous of heterogeneous robots (namely, a mobile robot and a bipedal robot) without distance constraints.

1.3 Contributions of the thesis

The contributions of this thesis are as follows.

- Computing an optimal location with minimax distance constraint for k agents amidst n polygonal obstacles in $O(k^2 + kn \log n)$ time.
- Computing an optimal location with minsum distance constraint for 3 and 4 agents amidst n polygonal obstacles in $O(n^2 \log n)$ time.
- Computing an optimal location where two non-identical agents achieve rendezvous when faced with an intelligent obstacle (an adversary) that travels at a superior speed.
- Computing an optimal location that minimizes the total time prior to rendezvous of k agents amidst n polygonal obstacles.
- Extension to sensor based rendezvous of a pair of heterogeneous agents amidst obstacles with minimal sensing in the absence of any distance constraints.
- Implementation of all the algorithms on custom fabricated mobile robots equipped with only a microcontroller (no external memory) and no communication with a central controller.

We now explain each of these contributions in detail in the subsections that follow.

1.3.1 Rendezvous of agents amidst obstacles while minimizing the maximum of distances travelled

The computation of an optimal rendezvous location (denoted by P_g) with the minimax distance constraint amidst obstacles is the first major contribution of the thesis. Prior work on this constraint is largely limited to environments that do not contain obstacles. In particular, to our knowledge, only Wynters and

Mitchell (1993) consider meeting of a pair of agents with line of sight communication between them, amidst obstacles. The authors present an $O(n^3 \log n)$ algorithm under the minimax metric where n is the total number of vertices. In this thesis, we show that when only two agents are involved, the computation of such an optimal rendezvous location takes no more than the time involved in computing the shortest path from one agent to another (i.e., $O(n \log n)$ due to Hershberger and Suri (1999)).

Further, we generalize this result to k agents. We first show that the rendezvous point is a function of either two or three agent locations and provide an algorithm to identify these among the k locations. We then introduce obstacles and show that the rendezvous point in the presence of obstacles lies on the same side of the rendezvous point computed in their absence. With the help of this result, we compute the optimal rendezvous location (P_g) amidst n polygonal obstacles with a constant number of vertices in $O(k^2 + kn \log n)$ time.

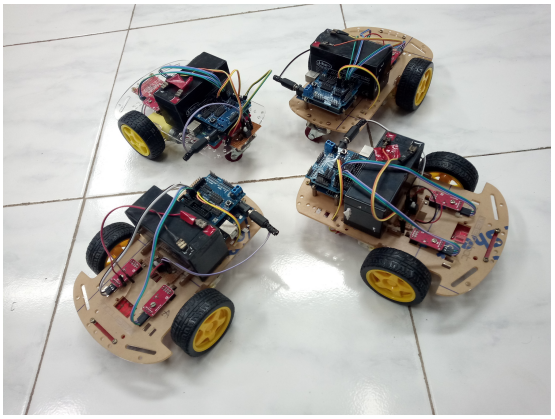


Figure 1.4: Custom fabricated differential drive mobile robots with rotary encoders and distance sensors supported by an Arduino UNO.

We then present an implementation of the algorithm on custom fabricated mobile robots (shown in Fig. 1.4) equipped with only a microcontroller. The computation of the optimal location involves identifying the points of intersection of two or more branches of hyperbolas. Small mobile robots equipped with just microcontrollers cannot directly handle such computations. We thus present a procedure that samples each curve and computes the point of intersection with an accuracy that can be handled by the physical robot.

1.3.2 Rendezvous of agents amidst obstacles while minimizing the sum of Euclidean distances of travel

As indicated earlier, instead of minimizing the maximum distance, one could minimize the total Euclidean distance. To our knowledge, in the presence of obstacles, work on this is once again limited to Wynters and Mitchell (1993) where only a pair of agents are considered. We show that, every point on the shortest path joining the two robots results in identical sum of distances value and thus is the optimal rendezvous location. The point can thus be computed in $O(n \log n)$ time.

Considerable work exists on calculating the point that achieves the least total distance in the absence of obstacles. For three (see Krarup and S.Vajda (1997)) as well as four (see Melzak (1974)) agents, this point can be constructed geometrically. However, when number of agents exceeds four, Cockayne and Melzak (1969) show that this point cannot be obtained as an algebraic expression involving radicals, thus necessitating an iterative algorithm due to Weiszfeld (1936).

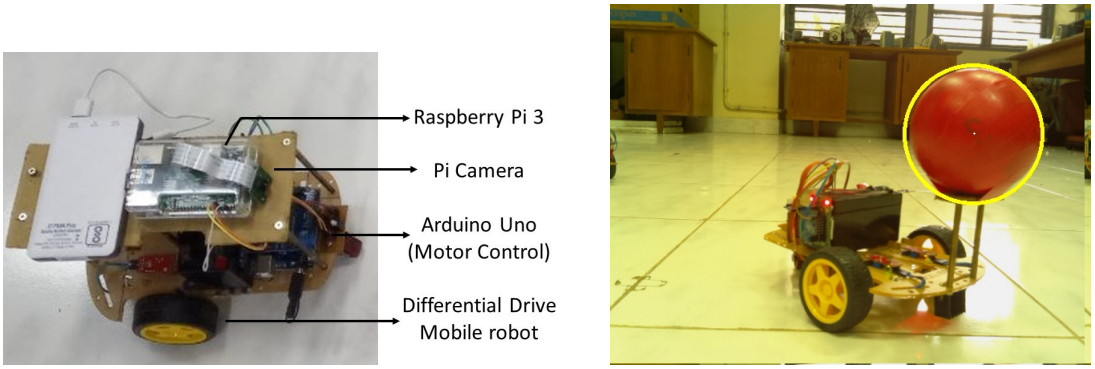
We thus consider the cases of three and four agents *in the presence of n polygonal obstacles*. We show that by handling the environment suitably, one need not scan the entire plane for the rendezvous point. Further, by careful introduction of obstacles in a specific sequence, one quickly arrives at the rendezvous point. We then present a hardware efficient implementation of the algorithm on the aforementioned mobile robots (Fig. 1.4).

1.3.3 Rendezvous of a pair of agents in the presence of an adversary

Instead of having obstacles and considering rendezvous of friendly robots (agents), one may encounter a scenario where one agent serves as an intelligent obstacle and opposes the rendezvous of others. In particular, we consider three agents where one serves as an adversary (or as an intelligent obstacle) for the other two. We equip the adversary with a vision sensor and allow communication only between the pair of friendly agents. The advantage of communicating the rendezvous location between the pair of friendly agents is matched by allowing the adversary traverse

at superior speeds.

When the adversary is equipped with only a vision sensor and does not have complete location information of the agents, the dominance curve is no longer the Apollonius circle. We present algorithms to compute these dominance curves and discuss the notion of safe region where a successful rendezvous can occur, given the maximum speeds of the agents and adversary. The outcome of the game is thus determined by whichever occurs earlier: capture by adversary or rescue by fellow agent.



(a) An adversary equipped with Raspberry Pi (b) Circular Hough Transform (in yellow)

Figure 1.5: Experimental setup for vision-guided adversary

An implementation of the algorithm for capture or rescue has been performed. A Raspberry Pi board provides processing support for the adversary while a Pi camera (as shown in Fig. 1.5(a)) acts as the vision sensor. It is worth noting that the processing of images by vision sensors is handled on-board and no additional support (from a PC) is necessary. Fig. 1.5(b) is an image from the point of view (POV) of the adversary where the Pi computes a Circular Hough Transform on the image captured and identifies a mobile agent. The radius of the circle indicates the distance of the agent from the adversary.

1.3.4 Time optimal rendezvous of multiple agents amidst obstacles

Having explored rendezvous amidst various distance constraints and intelligent obstacles, it is of interest to study rendezvous when time, instead of distance, serves as the constraint. The time optimal rendezvous point or the point corre-

sponding to minimum time rendezvous, in the absence of obstacles turns out to be the center of the smallest enclosing circle of all agent locations. When obstacles are introduced into the environment, the agents need to take turns at various obstacle vertices. Taking this change into account, we introduce the notion of weights at the agent locations where the weight on an agent is the time lost in commute prior to its arrival at the given location. We show that the time optimal rendezvous point for k agents amidst n obstacles is the center of the smallest enclosing circle of k circles whose radii are their respective weights. We present an $O(k^2 + kn \log n)$ algorithm to compute the same and validate it with the help of experiments involving differential drive mobile robots.

1.3.5 Extension to heterogeneous robots without distance constraints

We have so far assumed complete information of the agent locations and the environment. We also briefly address sensor-based rendezvous of agents with limited information. An algorithm that is independent of the kind of robots involved has been developed. However, *no distance constraints are imposed on the rendezvous.* We study a specific setup that involves a Pololu IR beacon to identify the direction of motion for individual robots. Preliminary experiments with a wheeled robot and a bipedal robot are also presented.

1.4 Organization of the thesis

The next chapter **Chapter 2** presents the details of relevant prior work on rendezvous of mobile agents. Emphasis is placed on various topics involving distance constraints while handling obstacles, adversarial rendezvous and sensor-based rendezvous.

Chapter 3 presents hardware-efficient algorithms to compute optimal location with respect to the minimax distance constraint. Experiments on a mobile robotic framework have also been described.

Chapter 4 presents algorithms to compute optimal locations with respect

to the minsum distance constraint. The chapter also presents an incremental algorithm that allows for a quick computation of the optimal location. Implementations of the algorithms on mobile robots are also provided.

Chapter 5 discusses rendezvous amidst an adversarial agent and presents algorithms to compute optimal locations for rendezvous via the notion of safe regions. Experimental verification with autonomous mobile robots is also described.

Time optimal rendezvous of mobile agents amidst obstacles is presented in **Chapter 6**. Algorithms and experimental verification are also given.

Theory and experiments pertaining to a specific sensor-based rendezvous of heterogeneous agents in the absence of distance constraints are presented in **Chapter 7**.

Chapter 8 summarizes this dissertation and discusses directions for future research.

1.5 Summary

This chapter has presented the motivation for the work described in this thesis. Terminology and definitions appropriate for discussing the contributions are given. The contributions have been enumerated and briefly described. In the next chapter, we present details of prior work on rendezvous of mobile agents amidst various constraints.

CHAPTER 2

LITERATURE SURVEY

This chapter is devoted to a review of the literature on rendezvous of multi-agent systems. Throughout this thesis, the emphasis is on rendezvous amidst obstacles with constraints on the distances or time travelled by the agents. Once a rendezvous location incorporating the constraints is computed, the agents need to traverse along the shortest paths from their respective initial locations. We thus begin with the discussion on prior work that computes the shortest path between two locations with polygonal obstacles in the environment.

2.1 Shortest path computation amidst obstacles

The Euclidean shortest path problem is one of the well-studied problems in computational geometry. Given a planar set of polygonal non-intersecting obstacles, the problem involves computing the shortest path between two points avoiding intersections with the interior of all the obstacles. Solution techniques for this problem can be classified into two primary approaches: the visibility graph method and the shortest path map method. While there are algorithms to calculate approximate solution to the shortest path, we focus on the literature dealing with computation of exact solution.

The visibility graph method is based on constructing a graph whose nodes are the vertices of the obstacles and whose edges are pairs of mutually visible locations (or vertices). The shortest path between two locations can then be found by running Dijkstra's algorithm on this graph (Dijkstra (1959), Cormen *et al.* (2009)). Overmars and Welzl (1988) study efficient ways of computing the visibility graph while Rohnert (1986) and Kapoor and Maheshwari (1988) utilize the visibility graph to compute the shortest path. An output sensitive algorithm for computing the visibility graph with a complexity of $O(n \log n + E)$ is presented

by Ghosh and Mount (1987), where E is the number of edges in the graph and n is the total number of obstacle vertices.

The second approach builds a shortest path map for a given source location to all the vertices of obstacles and the destination such that these points have the same vertex sequence as the shortest path to the source location. This map is essentially an encoding of shortest paths from the source location to all the points of interest in the plane. Mitchell (1996) provides an algorithm for computing the shortest path map that runs in $O(n^{3/2+\epsilon})$ time and space, for any $\epsilon > 0$ via an advanced range searching data structure. The research in this direction culminates with the work of Hershberger and Suri (1999) who take the shortest path map approach and build a subdivision of the plane. Optimal search techniques presented in Kirkpatrick (1983) and Edelsbrunner *et al.* (1986) are then utilized to compute the shortest path in an optimal $O(n \log n)$ time.

In the context of rendezvous of multiple agents, computation of shortest path amidst obstacles is merely a substep in determining the rendezvous location. We now move to prior work on rendezvous of multi-agent systems.

2.2 Rendezvous in multi-agent systems

One of the earliest works that concerns a group of autonomous agents is due to Vicsek *et al.* (1995). Early work in multi-agent consensus includes Jadbabaie *et al.* (2003), Olfati-Saber and Murray (2004) and Ren and Beard (2005). Chen *et al.* (2006) discuss coordination among agents from a dynamic systems perspective. The authors in Olfati-Saber *et al.* (2007) present a framework to study consensus and coordination in multi-agent systems. An *et al.* (2007) present algorithms for transitive dependence-based coalition formation.

In this thesis, the interest is on achieving rendezvous in multi-agent systems amidst constraints. Early work on this problem is due to Ando *et al.* (1999) who discuss rendezvous in the context of limited visibility among the agents. Following this, various additional constraints have been imposed either on the environment or on the dynamics of the robots. For instance, Ganguli *et al.* (2009) discuss rendezvous in a simply-connected and non-convex environment with constraints

on visibility sensors. In the context of multi-robot path planning, Bhattacharya *et al.* (2010) impose pairwise constraints on distributed optimization. Yu *et al.* (2012) discuss rendezvous in the absence of coordinates or communication among agents. An article by Cao *et al.* (2013) summarizes the contributions in the last decade to distributed multi-agent coordination.

Han *et al.* (2013) provide necessary and sufficient conditions for robust first and second-order consensus for a class of multi-agent dynamical systems. Consensus in multi-agent systems with sampled position and velocity data has been studied by Yu *et al.* (2013). The flocking problem for multi-agent systems is studied via model predictive control in Zhan and Li (2013). The authors in Leitão *et al.* (2013) present the state of the art in applications of industrial agents. They point to the potential of multi-agent coordination in military, defense and humanitarian relief applications.

2.3 Geometric aspects of rendezvous with minimax and minsum distance criterion

While a large body of work exists on rendezvous of mobile robots in general, the determination of a globally optimal location, especially amidst obstacles, has not received much attention. To our knowledge, this is limited to Wynters and Mitchell (1993), who consider rendezvous of a pair of agents amidst obstacles with respect to the minimax and minsum distance constraints. We now begin a discussion of prior work pertaining to calculation of optimal locations with respect to these two constraints *in the absence of obstacles*.

Work on minimax distance criterion has been reported as early as 1960s in the context of location theory (Francis (1967)). The usefulness of the minimax criterion in defence applications has been addressed by Isaacs (1964) where the advantage of minimizing the maximum distance (as opposed to the total or average distance) to trouble spots to save time in the context of deployment of airborne soldiers is discussed. Elzinga and Hearn (1972) present a finite solution procedure for the minimax problem for Euclidean and rectilinear distance measures based

on geometric arguments. Drezner and Wesolowsky (1980) generalize this result by considering the l_p metric (where $p \geq 1$) and establish empirically the efficiency of their procedure. An $O(n(\log n)^3(\log\log n)^2)$ algorithm has been reported in Megiddo (1983) to find a point in two dimensions that minimizes the maximum weighted distance to a point from a set of n given points.

The minsum distance constraint, however, can be traced back to the seventeenth century, when Torricelli devised a construction method to compute the point that minimizes the sum of distances to three other points (Courant and Robbins (1941)). This point has since then been referred to as the *Fermat point*, crediting the French mathematician who first posed the problem. When four locations are involved, the point is referred to as the Fagnano point (Melzak (1974)). When the number of agent locations exceeds four, the sum of distances is not expressible as an algebraic expression involving radicals since its computation involves finding zeros of higher order (> 5) polynomials. Consequently, an iterative procedure is utilized to compute the optimal location (Cockayne and Melzak (1969)). The problem involving more than three locations was first studied by Weber in the beginning of the twentieth century and the problem has since then been referred to as the "Fermat-Weber problem". Weiszfeld (1936) showed the convergence of an iterative scheme to the optimal location for the Fermat-Weber problem.

In each of these works, the paths from agent locations to the rendezvous point are treated as straight line segments joining these locations. In the presence of obstacles, however, the agents take turns at various vertices of obstacles before arriving at the rendezvous point. As a result, it is a challenge to define the minimax and minsum distance criteria via algebraic expressions when obstacles are involved. We address each of these concerns in Chapters 3 and 4 and present efficient geometric algorithms to compute optimal rendezvous locations in the presence of an arbitrary number of obstacles. Further, we discuss implementation of rendezvous via mobile robots. We now move on to the literature that deals with adversaries (or intelligent obstacles).

2.4 Rendezvous of a pair of agents in the presence of an intelligent obstacle

An intelligent obstacle (also referred to as an adversary) introduces the notion of conflict. Adversaries and their impact on security have been studied in industrial settings (see Long and Wu (2006), Yuan *et al.* (2016)). A well-studied problem in mobile robotics pertaining to adversaries is search and pursuit-evasion (see Chung *et al.* (2011)). Pursuit-evasion typically involves an evader attacking a static target while the pursuer acts to defend the target. An extension to pursuit-evasion is where two players attempt to herd under the influence of an attacker. The rendezvous of two 'friend' agents, that is of importance in this thesis, is somewhat similar to the notion of herding, discussed by Scott and Leonard (2013).

Defending a moving vehicle from an attack has been the subject of active research since the early 1960s (Isaacs (1965), Boyell (1976)). Adversaries act like obstacles to the movement of agents and it is desirable to identify regions of 'safety' where the rendezvous is not interrupted by the adversary. A geometric approach to an Active Target Defense Differential Game (ATDDG) is presented in Garcia *et al.* (2018) while the same is utilized in computing (i) dominance regions for capture and (ii) safe regions for rendezvous in Oyler *et al.* (2016). Contrary to the assumptions in these works, we do not assume any communication between the adversary and the friendly agents. Instead, we equip the adversary with a vision sensor that is used to identify and locate the agent.

Vision-based target tracking has also been well-explored in the past (see Luo and Chang (2012)). A target tracking strategy based on game theory has been proposed in Zhou *et al.* (2014). A cell decomposition-based approach to visibility-based pursuit evasion is presented in Bhattacharya (2011). An enhancement to this work considers visibility-based target tracking with a mobile observer and is reported in Zou and Bhattacharya (2016). A computational geometry-based approach to the lion and man game with visibility constraints has been reported in Noori and Isler (2014). Although visibility-based pursuit-evasion studies have been reported, a detailed geometric framework for determining rescue/capture, when the adversary is equipped only with a vision-based sensor, does not appear

to be available.

2.5 Time Optimal Rendezvous in multi-agent systems

Imposing constraints on time for rendezvous takes into account the speeds of various agents in addition to the distance travelled prior to rendezvous. Notarstefano and Bullo (2006) present a solution to achieve rendezvous in minimum time for a network of first order agents with bounded inputs. A decentralized algorithm to calculate arrival angles at a precomputed time optimal rendezvous point, for Dubin's vehicles, is reported in Bhatia and Frazzoli (2008). Brown *et al.* (2011) discuss algorithms to compute minimum time rendezvous points for multiple agents via level set methods. Recently, new techniques to compute optimal locations with velocity constraints (see Chunhe and Zongji (2014)) and power constraints (see Setter and Egerstedt (2014)) have been reported.

However, obstacles have not been considered in any of these formulations. Kunwar *et al.* (2005) discuss rendezvous with moving objects in dynamic cluttered environments. However, the discussion is limited to a single autonomous vehicle attempting a rendezvous with moving targets.

2.6 Summary

This chapter provides a survey of the literature on various problems introduced in Chapter 1. In the next chapter, we handle rendezvous with minimax distance constraint and present hardware-efficient algorithm for computing the optimal location.

CHAPTER 3

GATHERING OF WHEELED MOBILE ROBOTS WITH COLLISION AVOIDANCE AND A MINIMAX DISTANCE CRITERION

In this chapter, we consider rendezvous of hardware agents and in particular, autonomous mobile robots. Mobile robots operate with power constraints and it is desirable for each robotic agent to operate for several hours without recharging (of the batteries on-board). This goal can be related to the travel distance for each robot before they meet (to exhange supplies). We thus define a measure for distance that takes into account the constraints on resources (power, area) for each of the agents in a multi-agent scenario.

An intuitive approach to define such a distance measure is based on ensuring that the difference between the distances travelled by any two agents (before meeting) is small. In other words, a point that minimizes the maximum of distances travelled by all the agents prior to rendezvous can be treated as an optimal location (for rendezvous). Such a measure is referred to as the minimax distance constraint and is the primary constraint for rendezvous throughout this chapter.

We begin with a formal definition of the problem and then present efficient algorithms to compute such optimal locations while the environment is populated with obstacles. Another contribution of the work presented is efficient hardware realization of the proposed algorithms. We depict an implementation of the rendezvous point computation on small robots, each equipped with only a microcontroller. The memory on the microcontroller is shown to be adequate for implementation of the proposed algorithms. No external storage is required. Further, position information and distance (to obstacle) information are obtained using simple and low-cost units. This work has been reported in Vundurthy and Sridharan (2019).

3.1 Assumptions and Terminology

This chapter deals with computation of an optimal location in the plane that addresses the minimax distance criterion. We thus have the following definition.

Definition 1 *The location that minimizes the maximum Euclidean distance from each of the agent locations, in the absence of any obstacles, is defined as the Minimax point and is denoted by P_M.* ■

$$P_M(x, y) = min\{\underset{1 \leq i \leq k}{max} \sqrt{(x - x_i)^2 + (y - y_i)^2}\} \tag{3.1}$$

For k agents located at $\{P_1(x_1, y_1), P_2(x_2, y_2), \cdots, P_k(x_k, y_k)\}$, P_M is given by Eq. (3.1) and corresponds to an environment without obstacles. When obstacles are introduced, however, the calculation of the point that minimizes the maximum distance is more complex since the distances are now computed as the lengths of the shortest paths between two locations. We refer to this point as the *rendezvous point* and denote it by P_G.

Definition 2 *Rendezvous point P_G for k agents amidst n static obstacles is defined as the point external to all obstacles attaining the minimum of maximum of distances computed from k agent locations (to various points in the plane).* ■

The development of an efficient algorithm for computing P_G requires an appropriate model of the obstacles for collision avoidance. We assume that the area used by machines, furniture etc. can be represented by polygons of arbitrary shape (they could be non-convex). When the obstacles are non-polygonal, however, various approximation techniques exist that can be utilized to obtain a closely bound polygon. One of the earliest works that achieves this is due to Rosin (1997). We assume further that our agents are small and each can be represented by a point mass (similar to assumptions in Vicsek *et al.* (1995)).

The development of efficient algorithms also requires the notions of *visibility* and *last turn* since agents need to avoid collisions with the interior of the obstacles. Two arbitrary points, T_1 and T_2, are visible to each other, if the line segment

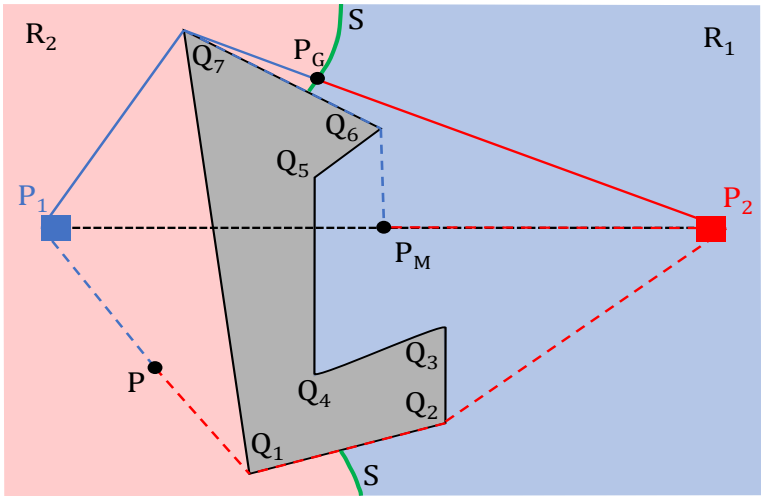


Figure 3.1: Rendezvous of a pair of agents (P_1 and P_2) in the presence of a non-convex polygonal obstacle

joining T_1 and T_2 does not intersect the interior of any obstacle. For instance in Fig. 3.1, P_1 is visible to Q_1 but not to P_2. The *last turn* location is defined as the point (vertex) that an agent reaches before becoming visible to the destination point. The *last turn* location for agent P_2 (to reach P) in Fig. 3.1 is Q_1 while the last turn location for agent P_1 to reach P_G is Q_7.

In the presence of static polygonal obstacles, agents move from one obstacle vertex to another before arriving at the rendezvous point P_G. The distance travelled by an agent from its initial location to an intermediate location (for instance, a vertex of an obstacle) may be thought of as the *weight* on the agent at that intermediate location. For instance, in Fig. 3.1, the weight on agent P_1 at vertex Q_6 is the distance travelled by P_1 to reach Q_6 (which is $(P_1Q_7 + Q_7Q_6)$). Using the notion of weights, we can define a *weighted minimax point* (denoted by P_M^W) as per (3.2).

$$P_M^W(x,y) = min\{\max_{1\leq i\leq k}(\sqrt{(x-x_i)^2+(y-y_i)^2}+d_i)\} \tag{3.2}$$

The definition in (3.2) assumes k agents with weights given by d_i, $i \in \{1, 2, \cdots, k\}$. P_M^W is later used to compute P_G in section 3.4.2. We begin with the computation of rendezvous point for two agents amidst n polygonal obstacles. Fig. 3.1 illustrates the location of minimax (P_M) and rendezvous points (P_G) for two agents in the presence of a non-convex polygonal obstacle. We note that, while there

are two paths between the agent locations, P_G corresponds to the mid-point of the shortest of the two paths. We present and formally prove this result in the following section and present a hardware-efficient algorithm to compute the same.

3.2 Rendezvous Point for a Pair of Agents amidst n Polygonal Obstacles

In this section, we consider two agents (located at P_1 and P_2) and n obstacles (each with a total of c vertices where c is a constant) and present an $O(n \log n)$ time algorithm to compute P_G. The key ideas are as follows. The rendezvous point P_G for a pair of agents is shown in Fig. 3.1. Let the curve S be the locus of all points that are equidistant to P_1 and P_2. Thus, on one side of S lies region R_1 (shown in blue), that is the collection of all points in the plane farthest from P_1. Therefore, the maximum of distances from P_1 and P_2 to any point in R_1 would be the distance from P_1. Minimizing this over R_1 brings one back to the curve S. The same applies to region R_2 (shown in red). The location of rendezvous point P_G can thus be narrowed down to curve S. Lemma 1 extends this idea to n obstacles.

Lemma 1 *The rendezvous point P_G for two agents moving amidst n polygonal obstacles is the point on the shortest path (from one agent to another) that is equidistant from both agents.* ■

Proof: This can be established as an extension of the no obstacles case where P_M corresponds to the mid-point. When obstacles are present, the path joining the two agents need not be just as one piece: it is, in general, a collection of segments. The total length of the collection is of interest and the location corresponding to half this length determines a potential rendezvous point. Since several such collections can exist, we choose the one that has the smallest total length which corresponds to the shortest path. The midpoint of the shortest path corresponds to P_G. **Q.E.D.**

We now present **Algorithm *Rendezvous_point_pair*** that uses Lemma 1 to compute the rendezvous point for two agents amidst n polygonal obstacles.

Algorithm *Rendezvous_point_pair*

INPUT: Two distinct, initial locations of the agents, denoted by P_1, P_2. n non-intersecting polygonal obstacles with c vertices each, labeled $(O_1, O_2, \cdots, O_n)$.

OUTPUT: Rendezvous point P_G

Step 1: Compute the shortest path (S_P) from P_1 to P_2 amidst n polygonal obstacles.

Step 2: Compute the mid point of S_P while moving from one vertex to the other starting from P_1. Output this midpoint as the rendezvous point P_G. ■

Theorem 1 *The time complexity of* ***Algorithm Rendezvous_point_pair*** *is* $O(n \log n)$. ■

Proof: Step 1 of the algorithm computes the shortest path from P_1 to P_2 amidst n obstacles with c vertices each (where c is a constant) in $O(n \log n)$ time based on the approach in Hershberger and Suri (1999). Given the shortest path, Step 2 takes $O(n)$ time to compute the point equidistant from P_1 and P_2 (since the $O(n)$ vertices along the shortest path are available from Step 1 and the length of each segment can be obtained in constant time). Thus, the overall time complexity of **Algorithm *Rendezvous_point_pair*** is $O(n \log n)$. **Q.E.D.**

Fig. 3.2 illustrates the rendezvous point P_G for two agents located at P_1 and P_2 amidst three obstacles with four vertices each. Curve S is the collection of points equidistant from the two agents P_1 and P_2. Rendezvous point P_G is the point of intersection of this curve S and the shortest path from P_1 to P_2. It is worth noting that **Algorithm *Rendezvous_point_pair*** computes this rendezvous point P_G without the need to compute the curve S, thus saving on computational time. We have so far assumed only a pair of agents. We next consider generalization to k agents moving amidst n polygonal obstacles. To this end, we begin with computation of the weighted minimax point for k agents in the absence of obstacles.

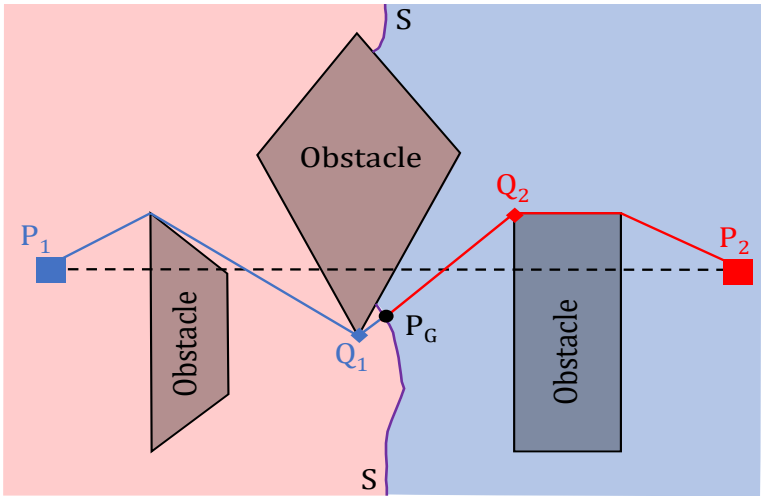


Figure 3.2: Rendezvous point (P_G) for two agents in the presence of three polygonal obstacles

3.3 Computation of Weighted Minimax Point P_M^W

For two agents moving amidst obstacles, the calculation of P_G has been studied by dividing the plane into two regions, each being farthest from one of the two agents. As the number of agents increases to k, the division of the plane into k regions becomes complex. We therefore develop an alternate procedure based on the notion of weighted minimax point (P_M^W), defined in section 3.1, to compute P_G. We begin with the computation of P_M^W for two and three agents and extend the result to k agents. The minimax point P_M can be obtained, if necessary, by computing the weighted minimax point P_M^W with zero weights at all locations.

3.3.1 Weighted Minimax Point for Two and Three Agents

Lemma 2 *For two agent locations P_1 and P_2 with weights d_1 and d_2 respectively, the weighted minimax point P_M^W is given by Eq. (3.3).*

$$P_M^W = \begin{cases} P_i, & \text{if } d_i \geq (l_{12} + d_j) \\ P_{12} & \text{otherwise} \end{cases} \tag{3.3}$$

for $i, j \in \{1, 2\}$ with $i \neq j$. l_{12} is the Euclidean length of the line segment P_1P_2 and

$$P_{12} = \frac{l_{12}(P_1 + P_2) + (d_1 - d_2)(P_1 - P_2)}{2 \times l_{12}} \tag{3.4}$$

■

Proof: Let P be an arbitrary point on line segment P_1P_2 that divides it in the ratio $r : (l_{12} - r)$, $0 \leq r \leq l_{12}$. Thus, the distances from P_1 and P_2 to P are given by

$$\begin{aligned} d_{PP_1} &= d_1 + r \\ d_{PP_2} &= d_2 + l_{12} - r \end{aligned} \tag{3.5}$$

Without loss of generality, let the first condition in Eq. (3.3) be expressed as $d_1 \geq (l_{12} + d_2)$. For any non-negative constant z, we thus have

$$\begin{aligned} d_1 - z &= l_{12} + d_2 \\ \Rightarrow d_{PP_1} &= d_1 + r \\ \Rightarrow d_{PP_2} &= d_1 - z - r \end{aligned} \tag{3.6}$$

Thus, $\max(\{d_{PP_1}, d_{PP_2}\})$ is d_{PP_1}. The minimum for d_{PP_1} occurs when $r = 0$ which indicates that the minimax point is P_1. Similarly, for the condition $d_2 \geq (l_{12} + d_1)$, P_2 is the minimax point. This proves the first condition.

The minimax point in the absence of the constraint described in Eq. (3.6), is the point on line segment P_1P_2 that is equidistant from P_1 and P_2. We have from Eq. (3.5),

$$\begin{aligned} d_{PP_1} &= d_{PP_2} \\ \Rightarrow d_1 + r &= d_2 + l_{12} - r \\ \Rightarrow r &= \frac{l_{12} + (d_2 - d_1)}{2} \end{aligned} \tag{3.7}$$

The expression for P_{12} (Eq. (3.4)) then follows from the section formula which divides line segment P_1P_2 in the ratio $(r : l_{12} - r)$ where r is given by Eq. (3.7).

Q.E.D.

We now present Lemma 3 to compute P_M^W for three agents with arbitrary weights at their locations. We denote by l_{ij} the length of line segment P_iP_j. Further, P_{ij} is the weighted minimax point of agent locations P_i and P_j (with weights d_i, d_j) computed using Lemma 2. d_{ij} is the weight computed at P_{ij}. The

length of line segment from P_{ij} to P_k is denoted by ${}_{ij}l_k$. It is worth noting that the locus of points equidistant from two agent locations with associated weights is a branch of a hyperbola (explained further in proof of Lemma 3).

Lemma 3 *For three agent locations P_1, P_2 and P_3 with weights d_1, d_2 and d_3 respectively, the weighted minimax point P_M^W is given by*

$$P_M^W = \begin{cases} P_i, & \text{if } d_i \geq (l_{ij} + d_j) \ \& \ d_i \geq (l_{ki} + d_k) \\ P_{ij}, & \text{if } d_{ij} \geq ({}_{ij}l_k + d_k) \\ P_{123} & \text{otherwise} \end{cases} \tag{3.8}$$

for $i, j, k \in \{1, 2, 3\}$ with $i \neq j \neq k$. P_{123} is the point of intersection of the three branches of hyperbolas constructed on the three sides with their respective weights.

■

Proof: The first two conditions follow from the proof of Lemma 2. When these two conditions do not apply, the minimax point is the point equidistant to the three agents with their corresponding weights. The locus of points (P) equidistant to two agents P_1, P_2 (with weights d_1, d_2) is a hyperbola as shown in Eq. (3.9).

$$\begin{aligned} PP_1 + d_1 &= PP_2 + d_2 \\ \Rightarrow PP_1 - PP_2 &= d_2 - d_1 \end{aligned} \tag{3.9}$$

The point equidistant to the three agents (which corresponds to P_M^W) is identical to the point of intersection of the three branches of the hyperbolas constructed on the three sides formed by initial locations of agents. **Q.E.D.**

Figures 3.3 (a) and 3.3 (b) illustrate the two cases in Lemma 2 for computing the weighted minimax point for two agents with non-zero weights. Fig. 3.3 (c) illustrates the last case in Lemma 3 and various terms used in computing the same.

We now present numerical illustrations to compute the weighted minimax point P_M^W as given by Lemmas 2 and 3. In Fig. 3.3 (a), $P_1(0, 0)$ and $P_2(10, 0)$ describe

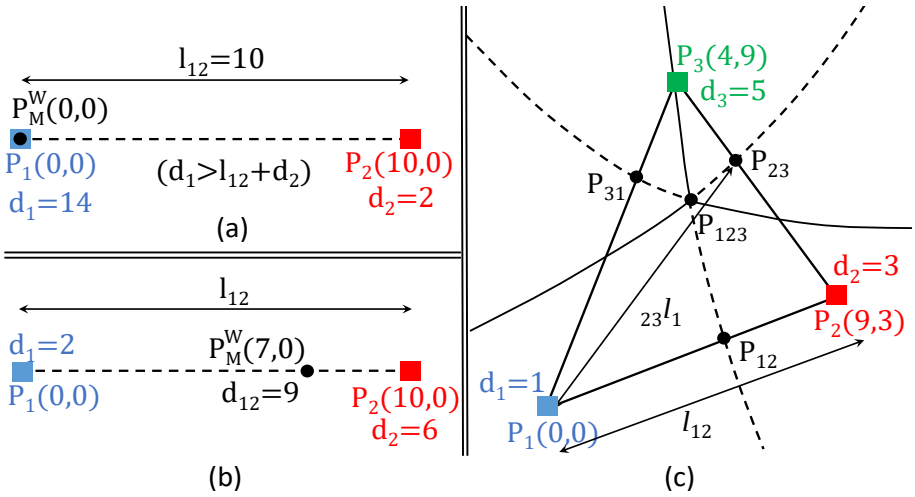


Figure 3.3: Weighted Minimax point P_M^W for two and three agent locations with non-zero weights

locations of agents with weights d_1 and d_2 given by 14 and 2 respectively. The length of line segment P_1P_2, namely l_{12}, is 10 units. Since $d_1 > l_{12} + d_2$, it follows from Lemma 2 that the weighted minimax point P_M^W is the agent location P_1.

In Fig. 3.3 (b), agents are located at $P_1(0,0)$ and $P_2(10,0)$ with weights d_1 and d_2 given by 2 and 6 respectively. With these values, we have $d_1 < l_{12} + d_2$ and $d_2 < l_{12} + d_1$. It thus follows from Lemma 2 that the weighted minimax point P_M^W is P_{12}. For computation of P_{12}, we treat agent locations P_1 and P_2 as vectors and all other entities (d_1, d_2, l_{12}) as scalars, shown in Eq. (3.10).

$$
\begin{aligned}
P_{12} &= \frac{l_{12}(P_1 + P_2) + (d_1 - d_2)(P_1 - P_2)}{2 \times l_{12}} \\
\Rightarrow P_{12} &= \frac{10 \times (\begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}10\\0\end{bmatrix}) + (2-6) \times (\begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}10\\0\end{bmatrix})}{2 \times 10} \\
\Rightarrow P_M^W &= P_{12} = \begin{bmatrix}7\\0\end{bmatrix}
\end{aligned}
\tag{3.10}
$$

In Fig. 3.3(c), we illustrate the computation of weighted minimax point for three agent locations $P_1(0,0), P_2(9,3), P_3(4,9)$ with weights d_1, d_2 and d_3 given by 1, 3 and 5 respectively. Since $d_i < (l_{ij} + d_j)$ & $d_i < (l_{ki} + d_k)$ $\forall$ $i, j, k \in \{1, 2, 3\}$ with $i \neq j \neq k$, the first condition in Eq. (3.8) is invalid. Similar computations show that the second condition in Eq. (3.8) is also invalid. Thus, the weighted minimax point P_M^W is P_{123}.

Locus of points equidistant from two agent locations with their associated weights is a branch of the hyperbola as given by Eq. (3.9). P_{123} is the point of intersection of three such hyperbolas constructed on the three pairs of agent locations (P_1, P_2), (P_2, P_3) and (P_3, P_1) as shown in Fig. 3.3(c).

For instance, consider the agent pair $P_1(0, 0)$ and $P_2(9, 3)$ with their corresponding weights d_1 and d_2 given by 1 and 3 respectively. Let $P(x, y)$ be an arbitrary point equidistant to P_1 and P_2 with weights d_1 and d_2 respectively. The locus of P (branch of a hyperbola) passes through the point $P_{12}(5.45, 1.81)$ and is given by Eq. (3.11) as shown in Fig. 3.3(c).

$$\begin{aligned} &\sqrt{(x-0)^2+(y-0)^2}+1=\sqrt{(x-9)^2+(y-3)^2}+3 \\ &\Rightarrow y=\frac{2\sqrt{43(2x^2-18x+43)}-27x+129}{5} \end{aligned} \tag{3.11}$$

Similar computations on the remaining two agent pairs lead to two additional hyperbolas that pass through points $P_{23}(5.86, 6.77)$ and $P_{31}(2.81, 6.33)$ as given by Eq. (3.12).

$$\begin{aligned} y&=\frac{\sqrt{57(4x^2-52x+201)}+30x-3}{32} \\ y&=\frac{36\sqrt{4x^2-16x+81}-72x+729}{130} \end{aligned} \tag{3.12}$$

The point of intersection of these three branches of hyperbolas is $P_{123}(4.48, 5.75)$ which is the weighted minimax point P_M^W for given agent locations and weights (Fig. 3.3(c)). An efficient method to implement these computations on a microcontroller is presented in section 3.5.1.

3.3.2 Weighted Minimax Point for k Agents

It is worth noting that the conditions in Lemma 3 correspond to cases where either one or two of the points are sufficient to compute the weighted minimax point. We now extend the ideas to k agents.

Lemma 4 *The weighted minimax point of k agents is identical to the weighted minimax point of three agents (among k agents) calculated using Lemma 3.* ■

Proof: When the weights on all the agents are equal, it follows from Rademacher

and Toeplitz (1957) that three points suffice to determine the weighted minimax point. If the weights are unequal, the last case in Lemma 3 holds. When there are more than three agents, the hyperbolas do not intersect (in general) at a single point. From the properties of a polygon, it follows that a maximum of three hyperbolas can intersect at a single point, ignoring redundancy. The agents corresponding to these three hyperbolas determine the triplet that constitutes the weighted minimax point. **Q.E.D.**

The remaining $k-3$ points satisfy the following condition:

$$d_{P_M} \geq d_i + l_{iP_M} \tag{3.13}$$

where d_{P_M} is the weight at the minimax location, d_i is the weight at the i^{th} location and l_{iP_M} is the length of the line segment joining P_i and P_M. Thus, these points do not play a role in computing the minimax point. We now present **Algorithm *Minimax_point_weighted*** to compute the weighted minimax point for k agents, using Lemmas 2, 3 and 4.

Algorithm *Minimax_point_weighted*

INPUT: k distinct initial locations of the agents denoted by $P_1, P_2, \cdots, P_k$. Weights at these k locations are denoted by the set $D = \{d_1, d_2, \cdots, d_k\}$.

OUTPUT: Weighted minimax point, P_M^W

Step 1: Choose the largest, second largest and third largest values from D and denote by (d_a, d_b, d_c) respectively where $a, b, c \in \{1, 2, \cdots, k\}$ and $a \neq b \neq c$.

Step 2: Compute Q_M as the weighted minimax point of three agent locations P_a, P_b, P_c with weights (d_a, d_b, d_c) using Lemma 3. Let d_m be the corresponding weight at Q_M.

Step 3: Return Q_M as the weighted minimax point P_M^W if Eq. (3.14) holds (where l_{im} represents the Euclidean distance between P_i and $Q_M$and **Stop**.

$$d_m \geq l_{im} + d_i \qquad \forall \;\; i \in \{1, 2, \cdots, k\} \tag{3.14}$$

Otherwise, choose an agent location and its weight P_d, d_d (where $d \in \{1, 2, \cdots, k\}$) that does not satisfy Eq. (3.14) and proceed to **Step 4**.

Step 4: Compute weighted minimax points for the three combinations (P_a, P_b, P_d), (P_a, P_c, P_d) and (P_b, P_c, P_d) using Lemma 3. Denote these points as Q_x, Q_y, Q_z and their weights as d_x, d_y, d_z respectively. Replace d_m with the maximum of d_x, d_y, d_z and Q_M with its corresponding minimax point. Replace (P_a, P_b, P_c) with the corresponding combination of d_m and proceed to **Step 3**. ■

Fig. 3.4 illustrates the algorithm for five ($k = 5$) agent locations. Step 1 picks three agent locations P_1, P_2, P_4 with highest weights (Fig. 3.4 (a)) and denotes them as (P_a, P_b, P_c). Step 2 computes the weighted minimax point (Q_M) and its weight (d_m) for (P_a, P_b, P_c) using Lemma 3. Q_M is the point of intersection of three branches of hyperbolas constructed on each side of $\triangle P_a P_b P_c$ (Fig. 3.4 (a)).

Step 3 then uses Eq. (3.14) to check if Q_M minimizes the maximum of distances to all agent locations. Since P_3 does not satisfy Eq. (3.14), P_3 is renamed as P_d (Fig. 3.4 (b)). Step 4 computes the weighted minimax point for the three combinations (P_a, P_b, P_d), (P_a, P_c, P_d) and (P_b, P_c, P_d) and identifies that the combination with highest weight is (P_a, P_b, P_d). Fig. 3.4 (b) illustrates Q_x as the point of intersection of the three branches of hyperbolas for $\triangle P_a P_b P_d$. Eq. (3.14) of Step 3 now reveals that Q_x is indeed the weighted minimax point for all agent locations and is thus the output (P_M^W).

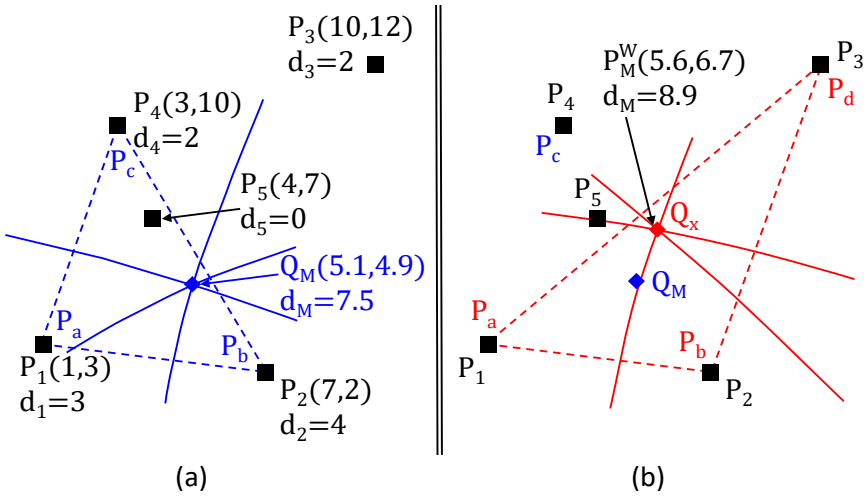


Figure 3.4: Illustration of **Algorithm *Minimax_point_weighted*** (a) Steps 1 and 2 (b) Steps 3 and 4

The convergence of **Algorithm *Minimax_point_weighted*** follows from the fact that the distance value d_m increases with each iteration. Since there are a finite number of points (k), the algorithm terminates when d_m reaches its maximum value. The complexity of **Algorithm *Minimax_point_weighted*** is expressed by Theorem 2.

Theorem 2 ***Algorithm Minimax_point_weighted*** *takes $O(k^2)$ time where k is the number of agents.* ■

Proof: Step 1 computes the first, second and third maximum in $O(k)$ time. Step 2 uses Lemma 3 to compute P_M^W in constant time. Step 3 verifies the condition in Eq. (3.14) for all k agents and thus takes $O(k)$ time. Step 4 computes three minimax points and then repeats Step 3 at most k times, thus taking $O(k^2)$ time. Hence, the overall complexity is $O(k^2)$. **Q.E.D.**

3.4 Computation of the Rendezvous Point P_G

Algorithm *Minimax_point_weighted* presented in section 3.3.2 is used in developing an efficient algorithm to compute the rendezvous point P_G. Polygons formed from the vertices of the obstacles and the agent locations contribute to the location of P_G. In particular, the *rendezvous point turns out to be the weighted minimax point of one of the polygons constructed from the obstacle vertices and agent locations* (an outline of the proof of this is part of Lemma 5).

However, a direct algorithm based on this idea has high computational complexity. This is in view of the fact that for k agents moving amidst n polygonal obstacles with c vertices each, we have up to ${}_{cn+k}P_k$ polygons to consider (and shortest paths from the agents to the vertices of these polygons). We first present a few key ideas that bring down the number of polygons to be considered from ${}_{cn+k}P_k$ to a maximum of ${}_{c+k}P_k$.

3.4.1 Key Results in computing P_G

Fig. 3.5 illustrates the rendezvous point for six agents moving amidst three polygonal obstacles (with four vertices each). One of the ${}_{4*3+6}P_6$ combinations is shown in Fig. 3.5 as the polygon $Q_1Q_2Q_3Q_4Q_5Q_6$. *Weight* d_i *(where* $i \in \{1, 2, \cdots, 6\}$ *) at a location* Q_i*, corresponds to the distance travelled by agent* P_i *from its initial location to* Q_i. This can be observed in Fig. 3.5 where d_3 (at Q_3) is the distance travelled by agent P_3 from its start location to Q_3, which is the length of the line segment P_3Q_3. The weighted minimax point P_M^W of this polygon $Q_1Q_2Q_3Q_4Q_5Q_6$ has the least maximum of distances value and corresponds to the rendezvous point P_G.

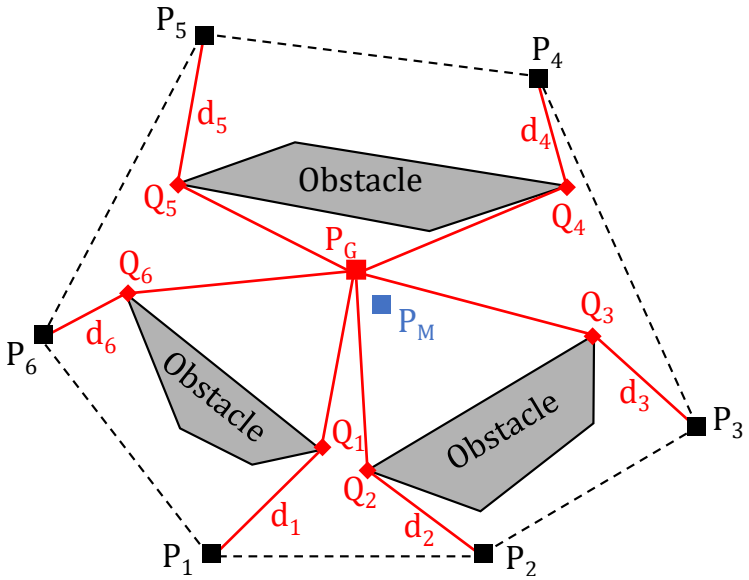


Figure 3.5: P_G and P_M for six agents amidst three polygonal obstacles

The proposed efficient algorithm is based on the following observation. The point P_M, which is the minimax point of k initial locations of agents with zero weights, is 'on the same side' as P_G as illustrated in Fig. 3.5. It is worth noting that P_M and P_G are both visible to the last turn locations to P_M (which are $Q_1, Q_2, Q_3, Q_4, Q_5, Q_6$). Thus, by identifying the last turn locations to P_M, one can immediately identify the polygon $Q_1Q_2Q_3Q_4Q_5Q_6$ whose weighted minimax point is the rendezvous point P_G. Therefore, the search for the rendezvous point P_G can be reduced from multiple polygons to one. This observation is established formally in Lemma 5.

Lemma 5 *Let* P_M *be the minimax point for* k *agents with zero weights and* P_G

be the rendezvous point amidst n polygonal obstacles. Further, let $Q_1, Q_2, \cdots, Q_k$ *be the last turn locations for these k agents attempting to meet (amidst obstacles) at the minimax point* P_M*.*

If P_M *lies external to all obstacles, the weighted minimax point of these last turn locations is the rendezvous point* P_G*.* ■

Proof: Let the maximum of distance values to P_M and an arbitrary point (denoted by P) be r_M and r_P respectively. It follows from Eq. (3.1) that the maximum of distances is lowest for P_M compared to any arbitrary point in the plane (P). We thus have,

$$r_M \leq r_P \tag{3.15}$$

Since the distance between two points generally increases with introduction of obstacles, we have Eq. (3.16).

$$\begin{aligned} r_M &\leq s_M \\ r_P &\leq s_P \end{aligned} \tag{3.16}$$

where the maximum of distances (lengths of the shortest path from agents' locations) to P_M is s_M and to P is s_P. Further, consider $Q_1, Q_2, \cdots, Q_k$ to be the last turn locations of agents attempting to gather at P_M. Let the weighted minimax point of $Q_1, Q_2, \cdots, Q_k$ be P_M^W and the maximum of distances value to P_M^W be s_W. In order to prove that P_M^W is the rendezvous point P_G, it is adequate to show

$$s_W \leq s_P \tag{3.17}$$

It follows from Eq. (3.2) that the weighted minimax point of the polygon constructed using last turn locations has the lower maximum of distances value compared to any other point with same last turn locations. We thus have

$$s_W \leq s_M \tag{3.18}$$

It follows from Eq. (3.15) that the maximum of distances value to P_M (which is r_M) will increase with addition of obstacles (Eq. (3.16)). The only point with lower maximum of distances value compared to P_M would be the weighted minimax point of its last turn locations as given by Eq. (3.2) and Eq. (3.18).

Therefore, the point with lowest maximum of distances value in the presence of obstacles is the weighted minimax point P_M^W which is in turn the rendezvous point P_G as given by Definition 2. We thus have Eq. (3.17). **Q.E.D.**

3.4.2 Efficient Algorithm to compute the Rendezvous Point

Lemma 5 is useful in developing **Algorithm *Rendezvous_point_efficient*** that computes the rendezvous point P_G for k agents moving amidst n obstacles with c vertices each.

Algorithm *Rendezvous_point_efficient*

INPUT: k distinct, initial locations of the agents denoted by $P_1, P_2, \cdots, P_k$. n non-intersecting polygonal obstacles with c vertices each, denoted using the set $O = \{O_1, O_2, \cdots, O_n\}$.

OUTPUT: Rendezvous point P_G.

Step 1: Calculate the minimax point P_M of polygon $P_1P_2\cdots P_k$ (with zero weights). Verify if P_M is contained in any of the n obstacles. Identify the obstacle (if any) that contains P_M, as O_j (where $j \in \{1, 2, \cdots, n\}$) and exclude it from the set O.

Step 2: Calculate the shortest path for the k agents to reach P_M amidst the obstacles. Denote the last turn locations in the shortest paths before reaching P_M as $Q_1, Q_2, \cdots, Q_k$ and the corresponding path lengths as $d_1, d_2, \cdots, d_k$.

Step 3: If O_j is empty, goto **Step 4** else goto **Step 5**.

Step 4: Replace P_M with the weighted minimax point (P_M^W) of polygon $Q_1Q_2\cdots Q_k$ computed with weights $d_1, d_2, \cdots, d_k$ using **Algorithm *Minimax_point_weighted***. Compute the shortest paths for k agents from initial locations to P_M. Identify the last turn locations to reach P_M as $R_1, R_2, \cdots, R_k$ and the distances along shortest paths to reach them as $f_1, f_2, \cdots, f_k$. Go to **Step 6**.

Step 5: Replace P_M with the rendezvous point of polygon $Q_1Q_2\cdots Q_k$ computed with weights $d_1, d_2, \cdots, d_k$ and one polygonal obstacle O_j. Include O_j back into

the set O and compute the shortest path from initial locations to P_M. Denote the last turn locations as $R_1, R_2, \cdots, R_k$ and the corresponding lengths of shortest paths as $f_1, f_2, \cdots, f_k$.

Step 6: If $R_i = Q_i \, \forall \, i \in \{1, 2, \cdots, k\}$, output P_M as the rendezvous point P_G and **Stop**. Else, replace Q_i with R_i and d_i with $f_i \, \forall \, i \in \{1, 2, \cdots, k\}$. Exclude O_j (if any) from the set O and return to **Step 3**. ■

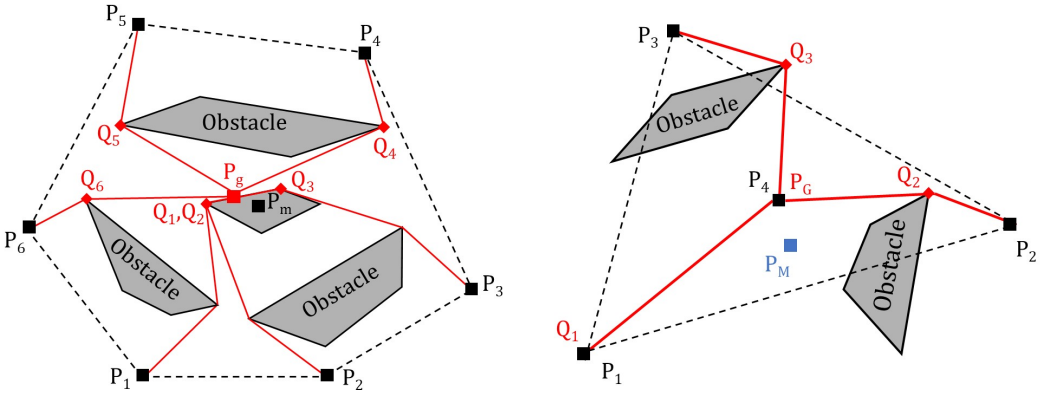
The correctness of this algorithm follows from Lemma 5. **Algorithm *Rendezvous_point_efficient*** is illustrated in Fig. 3.5. Steps 1 and 2 compute the minimax point P_M and paths from agents' initial locations. Since P_M is not contained in any obstacle (Step 3), Step 4 computes the weighted minimax point (P_G in Fig. 3.5) of last turn locations $Q_1, Q_2, \cdots, Q_6$. Recomputing shortest paths to P_G does not affect the last turn locations (Step 6). Thus weighted minimax point computed in Step 4 is output as the rendezvous point P_G. The convergence of this algorithm follows from the fact that there are only a finite number of polygonal obstacles to be considered in computing the rendezvous point.

Theorem 3 *The asymptotic time complexity of* ***Algorithm Rendezvous_point_efficient*** *is* $O(k^2 + kn \log n)$ *where* n *is the number of obstacles with* c *vertices each and* k *is the number of agents.* ■

Proof: Step 1 takes $O(k^2)$ time for computing the minimax point as given by Theorem 2. It takes additional $O(n)$ time for checking if P_M lies in any of the n obstacles. Computation of shortest paths to P_M from k agents in Step 2 takes $O(kn \log n)$ as given by Hershberger and Suri (1999). Computation of minimax point in Steps 4 and 5 once again take $O(k^2)$ time and the corresponding shortest paths take $O(kn \log n)$ time. Step 6 returns the rendezvous point P_G or reassigns variables for further computation. Thus, the overall time complexity is $O(k^2 + kn \log n)$. **Q.E.D.**

In addition to handling k agents amidst n obstacles, **Algorithm *Rendezvous_point_efficient*** is equipped to handle cases where the minimax point is contained in one of the obstacle or when the rendezvous point coincides with an agent location. In the former case, we show that the computation can be performed by

ignoring the obstacle and computing the rendezvous point and its respective last turn locations. The final rendezvous point in the presence of the ignored obstacle can be computed by handling all the ${}_{c+k}P_k$ polygons with respect to the ignored obstacle and k agent locations. This corresponds to Step 5 of the algorithm and an illustration is presented via Fig. 3.6(a). It is worth noting that such a technique does not affect the overall complexity of the algorithm.



(a) P_m lies inside an obstacle (b) P_G coincides with the location of P_4

Figure 3.6: Illustration of special cases in computation of P_G

Another special case is illustrated in Fig. 3.6(b). Here, the rendezvous point P_G coincides with initial location of one of the agents (P_4). As a result, while agent P_4 remains at P_G, the remaining agents follow the shortest path to P_G.

3.4.3 Communication among Agents for rendezvous at P_G

For two agents and n polygonal obstacles, Lemma 1 computes the rendezvous point P_G via the shortest path between agent locations. However, the shortest path need not be unique. So, in principle, there can be multiple choices for the rendezvous point. To ensure that the agents achieve rendezvous, we make use of communication among the agents and present the following Remark 1.

Remark 1 *We ensure that P_1 first communicates its location to P_2. P_2 then computes the rendezvous point via* ***Algorithm Rendezvous_point_pair*** *and transmits the same back to P_1 thus ensuring a unique location for rendezvous.* ■

In the implementation, this is accomplished by an array which stores (and updates) the shortest path. It is worth noting that this approach obviates the

need for a central controller. We now discuss an extension of this protocol to a multi-agent system via Remark 2.

Remark 2 *The agents at $(P_1, P_2, \cdots, P_{k-1})$ transmit their initial locations to the agent at P_k. The latter computes the rendezvous point P_G and transmits it back to the agents at $(P_1, P_2, \cdots, P_{k-1})$.* ■

The communication is implemented as follows. The agents encode their location and index number in the form of a 10-digit constant. The first four digits indicate the X-coordinate of the location while the next four indicate the Y-coordinate (in centimetres). The implementation currently allows 99 agents and hence the last two digits indicate the agent's index number ranging from 01 to 99. Index number 00 is reserved for the rendezvous point transmitted by the agent at P_k. For example, the encoded information transmitted by agent P_1 located at $(3142, 30)$ to P_k is of the form $\langle 3142003001 \rangle$. Similarly, the encoded form of a rendezvous point corresponding to $(2718, 43)$ sent by P_k to other agents is $\langle 2718004300 \rangle$. Since this approach involves transmission of k values (locations of $k-1$ agents and one rendezvous point) each 10 digit in size, the communication complexity is $O(k)$, computed with the help of Abelson (1978) and Yao (1979).

Remark 3 ***Algorithm Rendezvous_point_efficient*** *stops with calculation of the rendezvous point P_G. The task of computing the shortest path to P_G is left to the individual agents which can perform this in parallel. This has the advantage of keeping the time complexity of the algorithm low while at the same time, reducing the information to be transmitted (by P_k) to each of the $k-1$ agents.* ■

3.4.4 Enhancement to handle Collisions among Agents

In an industrial environment involving multiple hardware agents, one (moving) agent could itself be an obstacle to another. The algorithms presented so far handle k agents moving amidst obstacles with distance optimization as the goal. In this section, we answer the following question: *Can the paths generated for each agent avoiding collision with the interiors of the obstacles be used as such when handling collisions among agents on their paths to P_G?*

We present an algorithm for *path following that prevents collision among agents* and enables retaining the location (of P_G) computed with the polygonal obstacles using **Algorithm *Rendezvous_point_efficient***. It is assumed that the agents are equipped with distance sensors and communication modules. As indicated in Remark 3, the agents calculate the shortest path (in parallel) to P_G amidst obstacles. Fellow agents in the path of an agent to P_G are detected via the distance sensor.

Every agent that encounters an obstacle (a fellow agent) broadcasts its current location and receives the locations of similarly obstructed fellow agents. The agents use this information to identify if they are obstructing each other. In such a scenario, the agent with the highest index number proceeds further while the other agents wait for their turn.

Algorithm *Path_Following*

INPUT: Initial locations of all agents and obstacles. Rendezvous point P_G and paths of various agents to P_G. Information from distance sensor on each agent.

OUTPUT: All agents gather at P_G

Step 1: Translate each agent along its corresponding path to P_G until the distance sensor on an agent (say P_i) detects an obstacle. If all agents arrive at P_G, **Stop**.

Step 2: Compare the location of detected obstacle with the already stored information on static obstacles. In case of a match, ignore the obstacle and return to **Step 1**. Else proceed to **Step 3**.

Step 3: Broadcast the current location of the agent (P_i) along with its index number (i) to fellow agents and wait for reception of a similar message from a fellow agent.

Step 4: If P_i does not receive a reply, proceed to **Step 6**. Else, proceed to **Step 5** with the index numbers of all the obstructed fellow agents.

Step 5: If the index i is greatest among all the obstructed agents, conclude that agent P_i should proceed further and return to **Step 1**. Else proceed to **Step 6**.

Step 6: Stop the current agent (P_i) and monitor distance sensor readings. If the sensor continues to detect an obstacle, return to **Step 3**. Else return to **Step 1**.

■

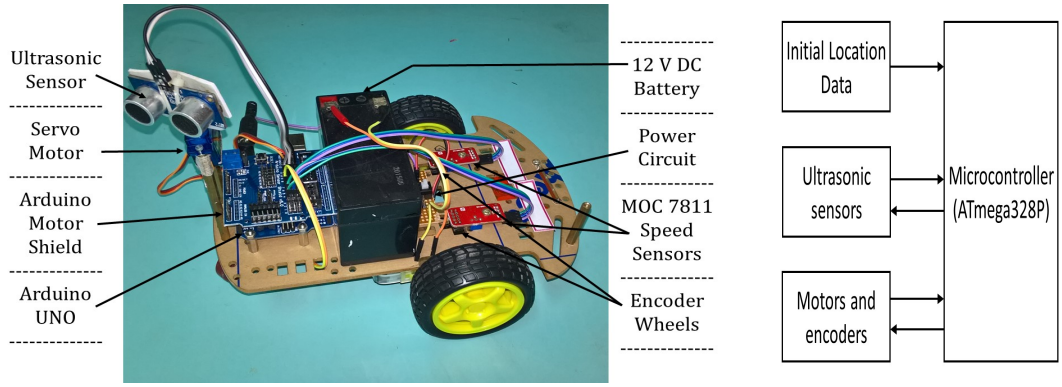
Remark 4 *The choice to stop momentarily is adopted so that the distance criterion and therefore the energy considerations are met. Other approaches to handle collisions among agents, in general, lead to greater travel distance for one or more agents.* ■

3.5 Implementation Aspects for rendezvous of Mobile Robots

In this section, we present the details of an efficient hardware realization of the algorithms on small robots that act as autonomous agents. The hardware on the systems is small so as to keep cost, weight and power consumption low. Fig. 3.7(a) illustrates various components involved in the mobile robot setup used in this thesis. Each robot is equipped with an 8-bit, 20 MHz ATmega328P microcontroller that executes the algorithms (some numerical computation aspects are presented in section 3.5.1). Further, position information is gathered via two MOC7811 sensors on each robot. These are inexpensive and adequate for small agents. The robots are also equipped with an ultrasonic sensor for detecting fellow agents (ultrasonic sensors are simple, low-cost and lightweight). The ultrasonic sensors are mounted on a servo motor that rotate to increase the field of view of the sensors. The entire arrangement is powered by a 12V, 1.3AH sealed maintenance-free battery. The communication among agents is realized with the help of Xbee-PRO RF modules. These units are low cost and low power wireless sensor networks operating at 2.4 GHz with a transmission range of up to 90m. Fig. 3.7(b) captures the interconnections between various components.

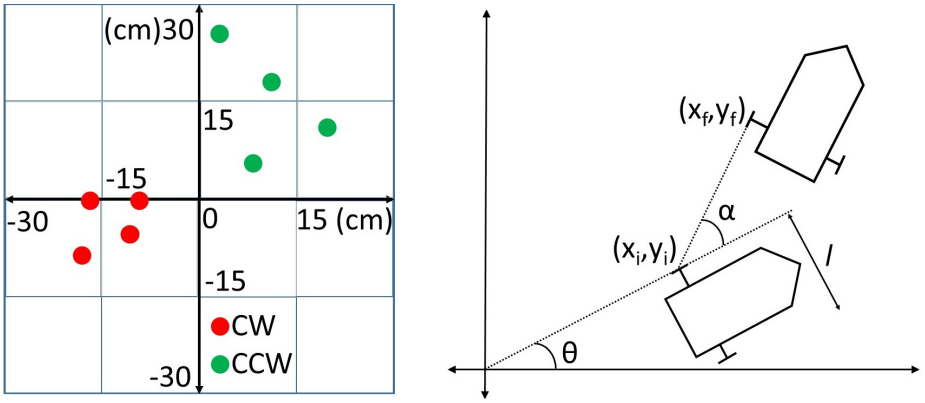
These robots have been tested against UMBmark experiments, discussed in Borenstein and Feng (1996). Fig. 3.8(a) presents the final location of the robot after tracking a clockwise and counter-clockwise square path starting and ending at the origin. The final measure of odometric accuracy for systematic errors, as defined by Borenstein and Feng (1996), is 18.78 cm.

Since a differential drive robot rotates around one of its wheels, we shift the



(a) The mobile robot (all the robots have the same structure) (b) Interconnections

Figure 3.7: Mobile robot and schematic of interconnections between various hardware components



(a) Calibration of the robots (b) Parameters for localization

Figure 3.8: Characteristics of mobile robots used

origin to its left wheel and use the following **Algorithm *Localize_Robot*** to calculate the current location of the robot based on the encoder readings. The mechanical structure of the robot influences the angle of rotation and the distance travelled. We denote the number of encoder slits as p, the diameter of robot's wheel a d and the length between two wheels as l (Fig. 3.8(b)).

Algorithm *Localize_Robot*

INPUT: Initial position (x_i, y_i), initial orientation θ, encoder count during rotation q_r, encoder count during translation q_t

OUTPUT: Current position (x_f, y_f) and orientation β

Step 1: Calculate the angle of rotation 'α' and distance translated 'r' using $\alpha = \frac{\pi d q_r}{pl}$, $\quad r = \frac{\pi d q_t}{p}$.

Step 2: The current orientation β and position (x_f, y_f) can then be computed

using $\beta = \alpha + \theta$; $(x_f, y_f) = (x_i + r * cos(\beta), y_i + r * sin(\beta))$

Step 3: Update initial position (x_i, y_i) and initial orientation θ with (x_f, y_f) and β respectively. Reset the encoder readings q_r and q_t and output the current location (x_f, y_f) and the current orientation (β). ■

We now discuss the adaptation of efficient algorithms for the computation of P_G to the presented mobile robot setup.

3.5.1 Numerical Aspects in Implementation of the Algorithms on Resource-Constrained Platforms

The proposed algorithms involve symbolic computation to arrive at the rendezvous point P_G. This can be observed in Lemma 3 where the minimax point is found by solving two or more second degree equations in two variables. The ATmega328P microcontroller supports 32KB of Flash Memory, 2KB SRAM and 1KB EEPROM. Assuming 4 bytes of storage for each of the x and y coordinates, a maximum of 256 vertices can be handled by the microcontroller simultaneously. Since microcontrollers are not equipped to perform symbolic computation, we develop a numerical procedure to solve the second degree equations in two variables.

The last case of Lemma 3 gives the minimax point as the point of intersection of three hyperbolas as shown earlier in Fig. 3.3 (c). The point of intersection of any two curves (denoted by L_1 and L_2) is the point at which both curves have a function value equal to 0. Further, if one curve (for example, L_1) is sampled at a fixed value as shown in Fig. 3.9, it gives a set of points (denoted by S_i $\forall$ $i \in \{1, 2, \cdots, m\}$, $m \in \mathbb{Z}^+$) on the curve L_1. The values of these points (S_i) when substituted in the curve L_2 determine their proximity to the point of intersection. The point in S_i with the substituted value closest to zero is the best approximation of the point of intersection.

The selection of sampling distance depends on (i) storage space of the microcontroller and (ii) size of the robots. The 32KB flash memory of ATmega328P allows for a maximum of 4096 points. If the points under consideration range a maximum of $10m$ on X and Y axes, the allowable sampling distance is $(\frac{10\times3}{4096})$

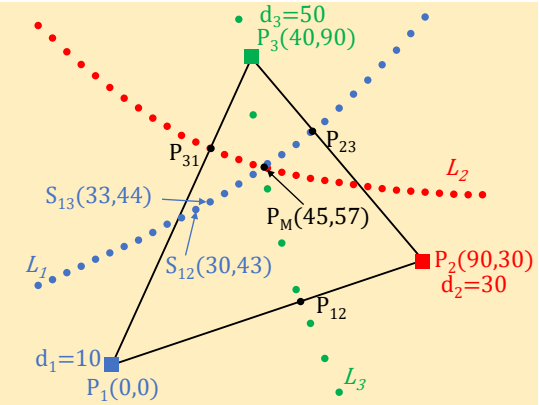


Figure 3.9: Numerical aspects in implementation of the algorithms

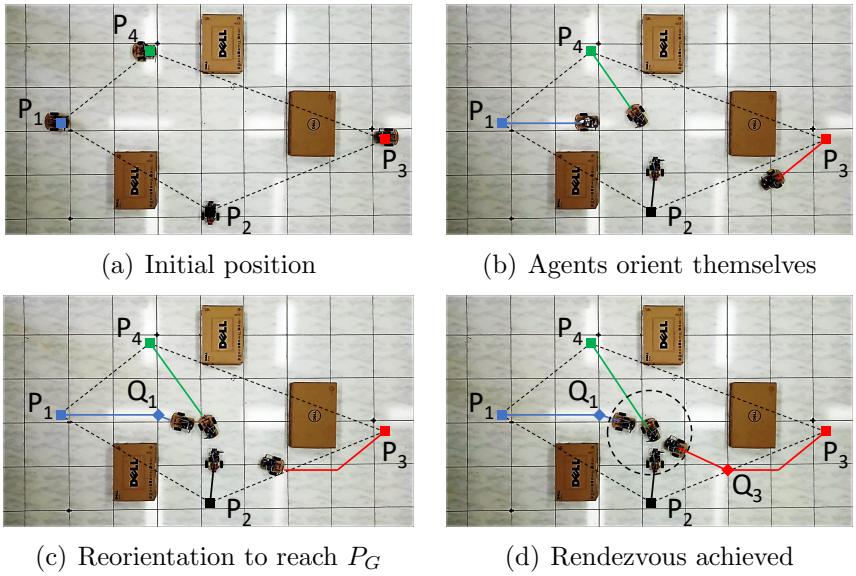
which is $0.7cm$. However, the (robotic) agents used in the experiments have an actual resolution of only $3cm$. We thus have a sampling distance of $3cm$ for all the experiments conducted in the following section.

Fig. 3.9 illustrates this procedure for the locations and weights given. The three hyperbolas L_1, L_2 and L_3 are converted into collection of points with a uniform separation of $3cm$ on the X-axis. Two sampled points S_{12} and S_{13} are shown in Fig. 3.9. The point at which its value is closest to zero is the minimax point $(45, 57)$. This compares well with the minimax point $(44.82, 57.47)$ when computed symbolically using MATLAB.

3.5.2 Summary of Experiments

We begin with an experiment involving four agents moving amidst three static obstacles. Initial positions of the agents at P_1, P_2, and P_3 are transmitted to the agent at P_4 which then uses **Algorithm *Rendezvous_point_efficient*** to compute P_G. Fig. 3.10 captures this experiment. The four initial locations are shown in Fig. 3.10(a). Once the agents have the location of rendezvous point P_G, they orient themselves towards it as shown in Fig. 3.10(b). Figures 3.10(c) and 3.10(d) show the last turn locations Q_1, Q_3 of agents starting at P_1 and P_3. Rendezvous of agents is shown in Fig. 3.10(d).

We now present another experiment where an agent encounters a fellow agent on its path to P_G. Fig. 3.11 summarizes this experiment. Initial locations of three agents and one static obstacle are shown in Fig. 3.11(a). As P_1, P_2 and

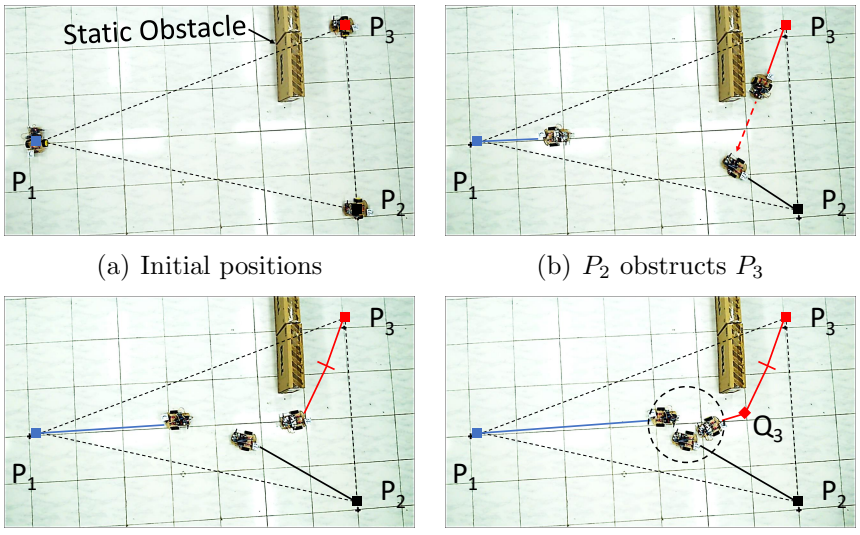


(a) Initial position (b) Agents orient themselves

(c) Reorientation to reach P_G (d) Rendezvous achieved

Figure 3.10: Rendezvous of four agents in the presence of three static obstacles

P_3 start to move towards P_G, the ultrasonic sensor mounted on P_3 detects P_2 as an obstacle (shown in Fig. 3.11(b)). P_3 halts (and waits) as given by **Algorithm *Path_Following*** until P_2 is no longer an obstacle (Fig. 3.11(c)) and then continues to move towards P_G. Agents achieve rendezvous as shown in Fig. 3.11(d).



(a) Initial positions (b) P_2 obstructs P_3

(c) P_3 resumes moving towards P_G (d) Rendezvous achieved

Figure 3.11: Rendezvous of three agents while one agent obstructs another's path

Remark 5 *As observed from this experiment, an agent detecting another agent as an obstacle is also handled by* ***Algorithm Path_Following****. Distance sensors on the agents facilitate detection of fellow agents and the agent that arrives first*

at a location common to the path of two agents proceeds first while the other agent waits.

Additionally, we have performed an experiment on rendezvous of two agents amidst obstacles that makes use of **Algorithm *Rendezvous_point_pair***. The study has also included experiments with humans moving, with larger number of static obstacles and with multiple agents approaching a location simultaneously.

3.6 Summary

A multi-agent rendezvous problem with constraints on distance and environment is studied in this chapter. Efficient geometric algorithms are presented for finding the rendezvous point of two or more agents amidst obstacles minimizing the maximum Euclidean distance of travel of the agents. An efficient hardware realization of the algorithms on multiple small robots is also presented.

A natural extension of this problem would be to a scenario where, instead of imposing a distance constraint that handles the agents individually (minimax distance criterion), emphasis is given on the cumulative distance travelled by all the agents. We discuss this in the following chapter.

CHAPTER 4

RENDEZVOUS OF WHEELED MOBILE ROBOTS AMIDST OBSTACLES MINIMIZING THE SUM OF EUCLIDEAN DISTANCES

In the previous chapter, hardware-efficient algorithms to compute an optimal rendezvous location with a minimax distance constraint have been presented. In this chapter, we seek the smallest (total) Euclidean distance the agents should travel before they meet when travelling amidst obstacles. Enforcing a constraint on the total Euclidean distance travelled can be related to the energy consumed by the system as a whole and it is desirable to reduce this total energy.

The rendezvous of three mobile agents travelling the least (total) distance is related to the notion of *Fermat point*, discussed in Courant and Robbins (1941). It turns out that, even when obstacles are introduced, the task is still identification of the appropriate set of three points to calculate the optimal rendezvous location. We first derive results to limit the search (space) for locating the minsum point when the environment includes multiple polygonal obstacles. We show that the the optimal rendezvous location *cannot lie outside the convex hull* formed by the obstacles and the locations of the three agents. We then show that it is adequate to consider ${}_7C_3$ triangles to calculate the rendezvous location when there is one rectangular obstacle. We extend the results to the case of n rectangular obstacles and develop direct and efficient algorithms with time complexities of $O(n^4 \log n)$ and $O(n^2 \log n)$ respectively.

Experimental results are presented using multiple mobile robots (locally fabricated) in an indoor environment. The robots do not require any communication among themselves or with a remote personal computer. In particular, the entire calculations and storage of data take place merely with the electronics on-board the robots and hence they operate autonomously. We first present the terminology involved in describing the rendezvous of agents in this chapter.

4.1 Terminology and Assumptions

In the absence of obstacles, the location in the plane that minimizes the sum of Euclidean distances to three agents locations is referred to as the Fermat point while the same for four agents is known as the Fagnano point. For greater than four agents, Weiszfeld provided an iterative algorithm that converges at the optimal location. In this chapter, we refer to this optimal location, in the absence of obstacles as the *minsum point* (and denote by P_S), irrespective of the number of agents. When obstacles are introduced into the environment, it is of interest to compute the rendezvous location that optimizes the sum of shortest paths from robot locations. We refer to this location as the *rendezvous point* and denote it by P_U, as opposed to P_G, the rendezvous point for minimax distance criterion.

The computation of P_U requires the notion of visibility between two points and the shortest path between two locations. Two points, p and q, in the plane are (mutually) visible if the line segment pq does not intersect the interior of any object (obstacle). This is used in defining the *complete visibility graph* of an environment consisting of a set of n polygonal objects. When the objects are convex, it is adequate to construct the *reduced visibility graph* (RVG) for calculation of shortest paths, as discussed by Rohnert (1986).

In the presence of n convex polygonal obstacles with a total of r vertices, there can be at most four supporting segments between a pair of convex polygons P_i and P_j (where $1 \leq i, j \leq n$). These segments can be computed in $O(\log r_i + \log r_j)$ time where r_i and r_j are the number of vertices in P_i and P_j respectively. Since there are $\frac{n\times(n-1)}{2}$ pairs of obstacles, we have four $\frac{n\times(n-1)}{2}$ segments altogether constituting the reduced visibility graph. Dijkstra's algorithm (given in Fredman and Tarjan (1984)) can then be used to obtain the shortest path in $O(n^2 + r\log r)$ time. Hershberger and Suri (1999) improve this computation of shortest path to $O(r\log r)$ where r is the total number of vertices in all the obstacles. We now discuss computation of the minsum point for three and four agents followed by an incremental algorithm that computes the rendezvous point P_U amidst n polygonal obstacles.

4.2 Computation of Minsum Point P_S

4.2.1 Minsum Point for Three Agents

The minsum point (also known as the Fermat point) of a set of three points in the plane is the point that minimizes the total (Euclidean) distance from the three points. The notion of minsum point also finds applications in triangulation and vision (Jan *et al.* (2014) and Hartley *et al.* (2011)). Torricelli provided a geometric solution to find this point based on constructing three equilateral triangles on the three sides of the triangle formed by the robot locations. The point of intersection of the circumcircles of these three triangles was shown to be the minsum point, as illustrated in Fig. 4.1.

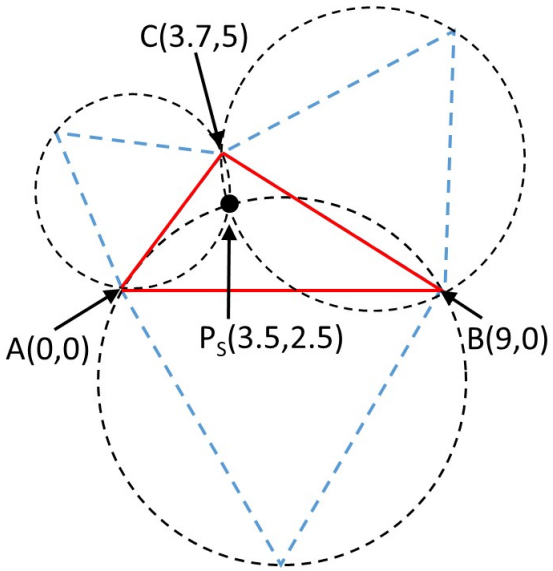


Figure 4.1: Illustration of minsum point P_S for three agents at A, B and C.

The locations of the agents and the minsum point are shown in the figure where the sum of distances to P_S turns out to be 15.37 units. It can be observed that any point adjacent to P_S result in a higher sum of distances value. For instance, the sum of distances to $(3, 2.5)$, $(4, 2.5)$, $(3.5, 3)$ and $(3.5, 2)$ lead to sum of $15.41, 15.41, 15.40$ and 15.41 units respectively.

4.2.2 Minsum point for four agents

When the environment includes a fourth mobile robot, calculation of the point that minimizes the sum of the Euclidean distances travelled, in the absence of obstacles, involves a quadrilateral formed from the locations of the agents. The minsum point (also known as the Fagnano point) for four agents depends on the convexity of the quadrilateral thus formed (Courant and Robbins (1941)). When the robot locations form a convex quadrilateral, the minsum point P_S turns out to be the point of intersection of the diagonals, as shown in Fig. 4.2 (a), while P_S turns out to be the non-convex vertex otherwise, as illustrated in Fig. 4.2 (b).

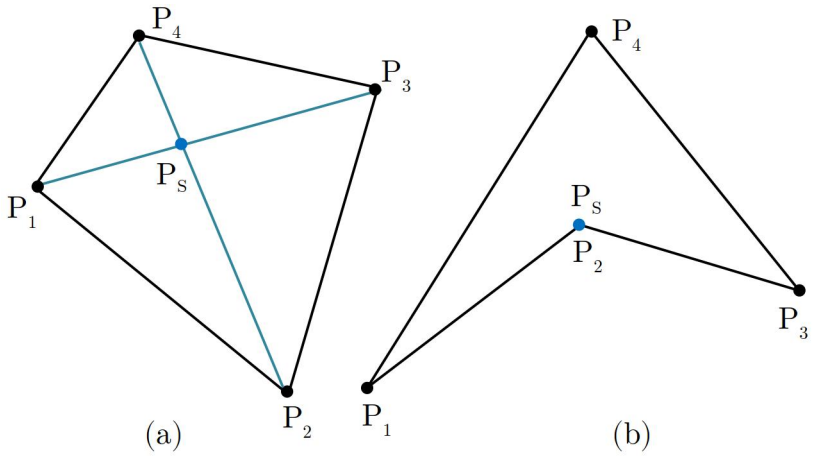


Figure 4.2: Minsum point P_S for four agents where the robot locations form a (a) convex and a (b) non-convex quadrilateral.

When the number of agents exceeds four, the *minsum point* P_S cannot be obtained algebraically. One can, however, take advantage of a procedure called one-point iteration method (developed initially in operations research) to compute P_S in the absence of obstacles. This procedure, referred to as Weiszfeld algorithm (Weiszfeld (1936)) in literature is a gradient descent technique to converge to the minsum location. Variations of Weiszfeld algorithm have also been recently reported Drezner and Hamacher (2004). The algorithm developed in this chapter, for the computation of rendezvous point P_U, relies on the minsum point computation in the absence of obstacles. Since, the computation of P_S, for more than four agents involves an iterative procedure, we restrict our discussion to development of an efficient algorithm for computing P_U for three and four agents amidst obstacles. However, the computation of rendezvous point for two agents requires a different approach and is presented first.

4.3 Rendezvous Point P_U for a Pair of Agents

The computation of P_U for two agents is illustrated in Fig. 4.3. Rendezvous with minsum distance constraint is achieved when the agents move towards each other directly along the straight line connecting their initial locations. In the absence of obstacles, every point on this straight line is *minsum* point P_S since the sum of distances to every point is equal to the length of the line segment joining them. However, when obstacles are introduced, the rendezvous point P_U does not necessarily lie on the line segment joining the location of the two agents as illustrated in Fig. 4.3. Once again, every point on the shortest path from one robot to another computes to the same sum of distances value which is equal to the length of the shortest path. Since the sum of distances to any other point in the plane would be higher than this value, every point on the shortest path can be treated as the rendezvous point. Fig. 4.3 indicates one such point P_U on the shortest path from A to B amidst the obstacles.

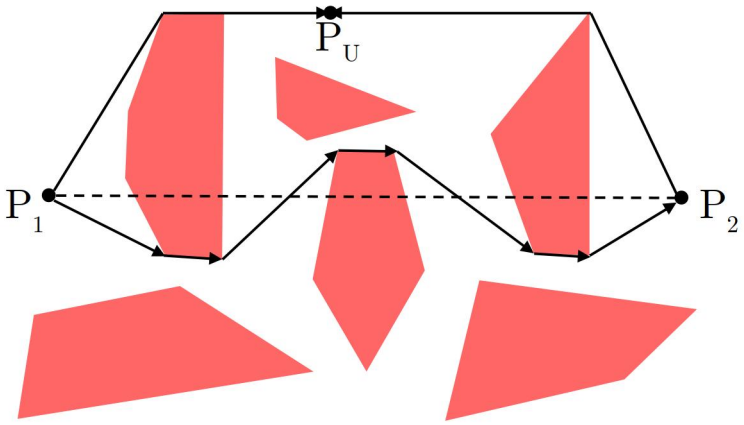


Figure 4.3: Rendezvous point P_U lies on the shortest path from A to B.

4.4 Hardware-Efficient computation of P_U

In this section, our interest is in the computation of the optimal rendezvous location P_U for three robot locations, with the minsum distance constraint, amidst obstacles. We begin with a few key results necessary.

4.4.1 Key Results

The search for P_U calls for examining the space in which the agents traverse. We first present a result on limiting the region of search for the rendezvous point to the convex hull formed from robot and obstacle locations.

Lemma 6 *Consider an environment with three robots located at P_1, P_2 and P_3. Further, let there be n obstacles labelled $O_1, O_2, \cdots, O_n$. Denote the convex hull of the triangle and the obstacles together by CH. The rendezvous point that minimizes sum of Euclidean distances lies within the convex hull CH.* ■

Proof: We establish this result by contradiction. Consider a point P that lies outside the convex hull CH as the rendezvous point. We can show that there exists a counterpart Q (to P) such that the sum of the distances to Q is lesser than the distance to P. To this end, we choose Q to be a point on the convex hull corresponding to the foot of perpendicular from P. If P is visible to a given robot location (say P_3), then P_3P is always greater than P_3Q since P_3P is the hypotenuse of the triangle P_3PQ. On the other hand, if P is not visible to a robot location (say P_2), then the path starting from P_2 would have to pass through another vertex (say F which lies either in the interior or on the boundary of the convex hull CH) to reach P. Since, Q is a foot of the perpendicular from P, $\angle FQP$ would be greater than or equal to 90°. By cosine rule, FP would be greater than FQ. Thus, the sum of the lengths of the paths from P_1, P_2 and P_3 to Q would be shorter compared to the (sum of the lengths of) paths from P_1, P_2 and P_3 to P. This proves that the rendezvous point P cannot lie outside the convex hull CH.

When the perpendicular from P falls on the extension of the nearest edge of the convex hull, the nearest vertex on the hull can be selected. The arguments to show that the rendezvous point cannot be outside the convex hull (for this case) are similar and hence omitted. **Q.E.D.**

Fig. 4.4 illustrates the ideas in Lemma 6. It is worth noting that although P is visible to P_1; Q is not. However, using the cosine rule, $P_1P > P_1S$ where S is the intersection of P_1L (extended) and PQ. Since LS is hypotenuse in $\triangle LQS$,

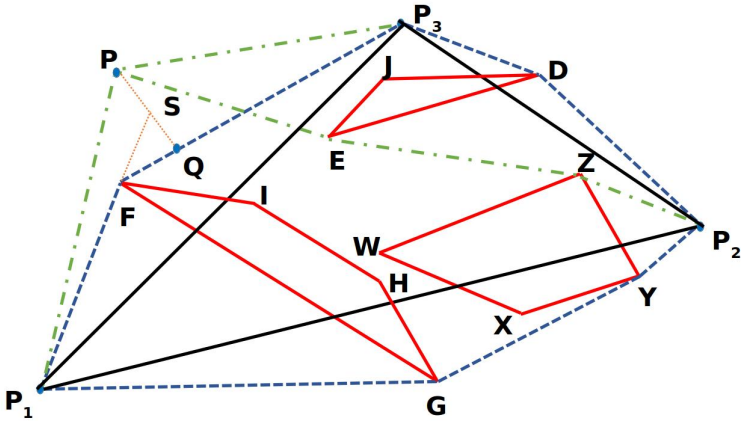


Figure 4.4: The rendezvous at P can be ignored in lieu of the rendezvous at Q due to the latter's lower sum of distances value.

$LS > LQ$. We thus have $P_1P > P_1L + LS > P_1L + LQ$, illustrating the path to Q from P_1 is shorter in length compared to the path to P from P_1. It can additionally be noted that there exists a point in the interior of the convex hull with sum of distances lesser than the distance to Q. This would be elaborated in the following results. While Lemma 6 allows us to search for P_U within the convex hull CH, it does not explicitly say how the convex hull should be handled to locate P_U.

Lemma 7 *Rendezvous point P_U corresponds to the minsum point of a triangle formed by a combination of the initial locations of the agents and the vertices of the obstacles.* ■

Proof: The three agents start from three distinct locations, P_1, P_2 and P_3. In the absence of obstacles, rendezvous point P_U is identical to *minsum point* P_S of $\triangle P_1P_2P_3$. Without loss of generality, let robot P_2 be obstructed by an obstacle O_1 before the rendezvous takes place. In this case, a vertex O_1^3 of the obstacle O_1 replaces P_2 (see Fig. 4.5) so the point that minimizes the sum of distances will be obtained from the new triangle formed by P_1, O_1^3 and P_3. Therefore, while the agents can move from one vertex of the convex hull to another, the final rendezvous point P_U corresponds to the minsum point formed by three locations which correspond either to the vertices of obstacles or to the initial locations of the agents (namely P_1, P_2 and P_3). **Q.E.D.**

Lemma 8 *Consider, as before, three agents located at the vertices of a triangle $\triangle P_1P_2P_3$. Let there be one polygonal obstacle (with c sides where c is some constant) labelled O_1 with its c vertices located at $O_1^1, O_1^2, \cdots, O_1^c$. The rendezvous point P_U for these three agents in presence of the obstacle O_1, is identical with the minsum point of one of the ${}_{c+3}C_3$ triangles formed by the combinations of the $c+3$ points under consideration.* ■

Proof: Let P be any arbitrary point in the convex hull CH of $\triangle P_1P_2P_3$ and polygonal obstacle O_1. Additionally, let us assume that P is not contained in the obstacle O_1. The three agents traverse a path to reach P from their respective positions at P_1, P_2 and P_3. Let Q_1, Q_2 and Q_3 be the last three points on the path traversed by the agents P_1, P_2 and P_3 respectively before reaching P. Now, let Q be the minsum point of the triangle $\triangle Q_1Q_2Q_3$. The total distance travelled by the agents from P_1, P_2 and P_3 to reach Q_1, Q_2 and Q_3 is equal while trying to reach either P or Q. However, by the property of minsum point in a triangle,

$$Q_1P + Q_2P + Q_3P > Q_1Q + Q_2Q + Q_3Q \tag{4.1}$$

Thus, the distance travelled by the three agents to reach Q is shorter than the distance travelled by the three agents to reach P. Thus, for every point P in the convex hull CH, there exists its counterpart Q with a shorter distance to travel. Thus the rendezvous point P_U has to be the minsum point of one among the ${}_{c+3}C_3$ triangles formed by the points under consideration. **Q.E.D**

Fig. 4.5 illustrates Lemmas 7 and 8 with the agents located at P_1, P_2 and P_3. A rectangular obstacle ($c = 4$) is considered and is depicted by $O_1^1O_1^2O_1^3O_1^4$. Let P be any arbitrary point. Since P is visible to both the agents at P_1 and P_3 (line segments from P_1 to P and P_3 to P are not obstructed), Q_1 is identical to P_1 while Q_3 is identical to P_3. However, the robot at P_2 reaches P via O_1^3. We thus identify Q_2 as O_1^3. The minsum point of $\triangle Q_1Q_2Q_3$ is located at Q which is not visible from P_2. However Q is visible from P_1 and P_3 (line segments P_1Q and P_1P_3 do not intersect any other object). Thus the path traversed by the robot P_2 to reach Q is via O_1^3. Eq. (4.1) holds true because Q is the minsum point of $\triangle Q_1Q_2Q_3$.

It is to be noted that the point Q is not necessarily the rendezvous point P_U.

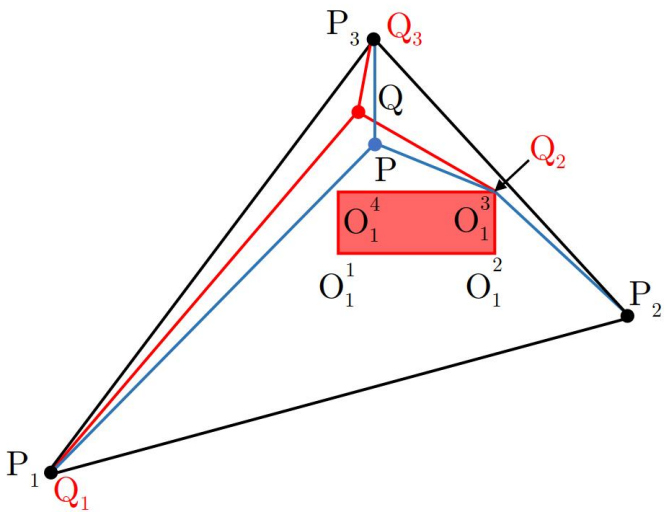


Figure 4.5: $\triangle Q_1Q_2Q_3$ of Lemma 8 turn out to be $\triangle AO_1^3C$

For every P in the convex hull, there exists a corresponding Q which is shorter in distance to travel. The number of triangles is limited by the number of vertices of the obstacles and the number of agents which is $c + 3$. Thus, a total of ${}_{c+3}C_3$ triangles can be formed.

Remark 6 *An important observation from Lemma 8 is that a procedure to calculate the minsum point for an appropriate triangle is adequate to obtain the rendezvous point P_U amidst obstacles. Larger-size polygons need not be considered.*

■

In Lemma 8, it is possible that some of ${}_{c+3}C_3$ triangles have their corresponding minsum point inside an obstacle thus making them invalid. For the remaining valid triangles, the total distance travelled to reach the corresponding minsum point can be calculated using shortest path algorithms. This total distance is then minimized over all of the valid triangles to find the rendezvous point P_U.

The procedure for finding the rendezvous point P_U for one obstacle can be extended to n obstacles by observing that the total points now under consideration would be $(cn + 3)$. We therefore have Corollary 1.

Corollary 1 *Rendezvous point P_U in the presence of 3 agents and n obstacles (with c vertices each) is the minsum point of one of the ${}_{cn+3}C_3$ combinations of triangles formed from the three initial locations of agents and (cn) vertices of n obstacles.* ■

4.4.2 Direct and Efficient Algorithms to compute P_U

A *direct* algorithm to compute the rendezvous point P_U for three agents amidst n polygonal obstacles can be developed using Corollary 1.

Algorithm *Direct_Rendezvous_Point*

INPUT: Three distinct, non-collinear initial locations of the agents: P_1, P_2 and P_3. 'n' non-intersecting rectangular obstacles: $O_1, O_2, \cdots, O_n$.

OUTPUT: Rendezvous point P_U

Step 1: Find the set of minsum points F=$\{f_1, f_2, \cdots, f_g\}$ for the 'g' sets of triangles where g=$_{4n+3}C_3$.

Step 2: Identify and eliminate 'k' elements in F that are contained in any of the 'n' rectangular obstacles.

Step 3: Determine the shortest path for each of the three agents to reach the 'g-k' coordinates in set F and the corresponding set of total path lengths D=$\{d_1, ...d_{g-k}\}$.

Step 4: Identify the minsum point corresponding to the smallest value in set 'D' and assign it as the final rendezvous point P_U (and output its coordinates). ■

Remark 7 *The computational complexity of* ***Algorithm Direct_Rendezvous_Point*** *can be analysed as follows. Step 1 takes $O(n^3)$ time. Step 2 takes $O(n^4)$ time. Step 3 takes $O(n^4 \log n)$ time (using the shortest path algorithm in Hershberger and Suri (1999)) while Step 4 takes $O(n)$ time. Hence, the overall complexity is $O(n^4 \log n)$.* ■

One of the reasons for the high computational complexity of **Algorithm *Direct_Rendezvous_Point*** is that it considers all the $_{4n+3}C_3$ triangles at once. Therefore, it is advantageous to look at alternate approaches. One approach, often studied for construction problems in computational geometry (O'Rourke (1993)), is based on incremental introduction of inputs. Recently Hartline and Sharp (2006) utilized incremental models for optimization problems, especially when complete knowledge of the inputs is available. Motivated by these works, we investigate development of an efficient algorithm for the rendezvous problem by

introducing obstacles one by one and updating the triangle containing the minsum point.

We note that the final rendezvous point P_U is influenced by the cluster of obstacles around the initial location of the minsum point P_S calculated in the absence of obstacles. Our strategy is thus based on introducing the obstacles in increasing order of their distances from the initial minsum point P_S. The initial minsum point P_S is obtained in $O(1)$ time merely based on the location of the three agents. With the introduction of each obstacle, there is a new location for the agents. Lemma 8 helps in finding the new minsum point and the corresponding new minsum triangle with every iteration. To handle cases where the final rendezvous point P_U is in direct line of sight to the initial location of one of the agents, we require another pass in the opposite direction.

Algorithm Incremental_Rendezvous_Point

INPUT: Three distinct, non-collinear initial locations of the agents: P_1, P_2 and P_3, 'n' non-intersecting rectangular obstacles: O_1, O_2, $\cdots$, O_n.

OUTPUT: Rendezvous point P_U

Step 1: Calculate the initial minsum point, P_S, of the triangle formed by the three agents. Arrange the n rectangular obstacles in the increasing and decreasing order of their distances from P_S: In=$\{O_i, O_j, \cdots, O_l\}$, Dn=$\{O_l, \cdots, O_j, O_i\}$ $(i, j, \cdots, l) \in \{1, 2, \cdots, n\}$

Step 2: Introduce all the obstacles, one by one, as per the sequence provided by the sets In and Dn. Use Lemma 8 to identify the current minsum triangle, update the minsum point P_S (including verification that the point is not in the interior of any of the n obstacles) and the path leading to it with each obstacle introduction.

Step 3: Identify the final minsum points unique to the two orders: FF_{PI} and FF_{PD}. Use a shortest path algorithm to find and record the path from the initial location of the three agents to these two points FF_{PI} and FF_{PD}. If the path leads to either FF_{PI} or FF_{PD} after passing through the vertices of the minsum triangle (just found), then go to **Step 5**

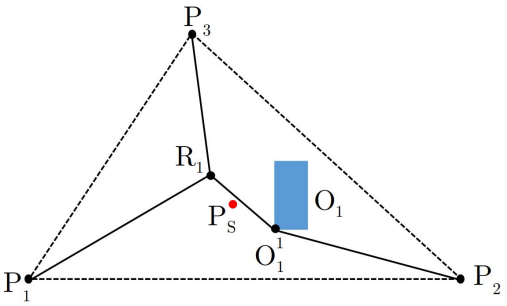
Step 4: Recalculate and update the final minsum point FF_{PI} or FF_{PD} with the

current minsum triangle.

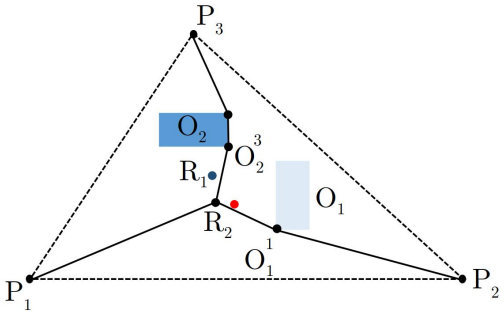
Step 5: Denote the sum of distances from the three agents to FF_{PI} by D_I and to FF_{PD} by D_D. The point with the shorter distance among D_I and D_D would be the rendezvous point P_U and the corresponding path would be the final path taken by three agents to reach their destination P_U. ■

Remark 8 *Updating the minsum point in Step 2 includes verifying that the point is not in the interior of any of the n obstacles. Updating the path includes calculating the shortest path from the robot's initial position to the current minsum triangle. The current minsum triangle would be the new location of agents before introduction of the next obstacle.* ■

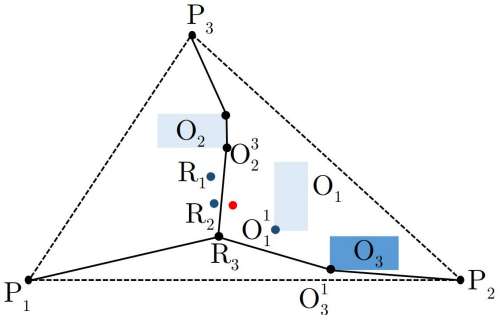
We now illustrate **Algorithm Incremental_Rendezvous_Point** via Fig. 4.6. P_S, initial minsum point, is obtained with the three agents located at P_1, P_2 and P_3. Step 2 calculates the new minsum point R_1 after introduction of the first obstacle O_1 using Lemma 8. The shortest path from the three agents to R_1 is shown in Fig. 4.6(a). R_1 is the minsum point of the triangle formed by the three points P_1, O_1^1 and P_3.



(a) Movement to R_1 from P_S on adding Ob_1.



(b) Movement to R_2 on adding Ob_2



(c) Movement to R_3 on adding Ob_3

Figure 4.6: Computation of P_U via an incremental addition of obstacles

On introduction of the second obstacle (O_2), the rendezvous point shifts to R_2. R_2 is the minsum point of the triangle formed by the three points P_1, O_1^1 and O_2^3. The shortest paths from the agents' initial positions to R_2 are indicated by the blue lines in Fig. 4.6(b). The third obstacle O_3 only obstructs the path from P_2 to O_1^1. Computation of the shortest paths indicates that the second robot need not go through O_1^1 to reach R_2. Changing the path necessarily changes the minsum triangle as well. Thus, Step 3 ensures that the new minsum point is now computed using the P_1, O_2^3 and O_3^1. Step 4 leads to the new minsum point R_3. A shortest path algorithm is run again to ensure that the triangle determined does not change. Thus, the final rendezvous point is R_3. The paths taken by the three agents to reach R_3 are given in black lines in Fig. 4.6(c).

Computational Complexity Analysis

Step 1 in **Incremental_Rendezvous_Point** calculates the (shortest) distances from the initial minsum point to each of the n obstacles in $O(n)$ time. It then arranges them in increasing and decreasing orders. Thus, Step 1 has an overall complexity of $O(n \log n)$. Step 2 includes calculation of the minsum triangle with introduction of each obstacle. During the calculation, we verify that the minsum point is not within any of the obstacles (which takes $O(n)$ time). Additionally, for cases where the path needs to be updated, the shortest path algorithm consumes $O(n \log n)$ time. Hence, the overall complexity of Step 2 is $O(n(n+n \log n))$ which is $O(n^2 \log n)$. Step 3 computes the shortest path from the initial location of agents to the final rendezvous points FF_{PI} and FF_{PD}. This can be accomplished in $O(n \log n)$ time using the shortest path algorithm in Hershberger and Suri (1999). Steps 4 and 5 take constant time. The complexity of **Algorithm Incremental_Rendezvous_Point** is therefore $\mathrm{O}(n^2 \log n)$.

Remark 9 *The proposed algorithms (Direct and Incremental) readily extend to non-rectangular obstacles. In particular, the notion of minsum point is applicable also to general polygonal obstacles. The task is still identification of the appropriate triangle for calculating the minsum point which is the rendezvous point.* ■

We now briefly discuss the computational aspects for the extension of the

incremental algorithm when obstacles are c-sided polygons (where c is some constant) while the total number of vertices is nc. The number of obstacles is still n. The first step involves calculation of the (shortest) distances from the initial minsum point to each of the n obstacles and this takes $O(nc)$ time. Sorting takes $O(n \log n)$. Step 2 involving updation of the minsum triangle takes $O(c^3)$. We then verify that the minsum point is not within any of the obstacles (which takes $O(nc)$ time). Additionally, path updation invokes a shortest path algorithm that takes $O((nc)\log(nc))$ time. Hence, the overall complexity of Step 2 is $O((nc)^2c^2\log(nc))$. Step 3 computes the shortest path from the initial location of agents to the final rendezvous points FF_{PI} and FF_{PD} in $O((nc)\log(nc))$ time. Steps 4 and 5 take constant time. Since c is considered to be a constant, the overall complexity is $O(n^2 \log n)$.

Remark 10 *The results presented thus-far naturally extend to a scenario involving four agents. However, the minsum point will be computed as the discussed in section 4.2.2 and quadrilaterals are considered in determining the rendezvous point, instead of triangles.* ■

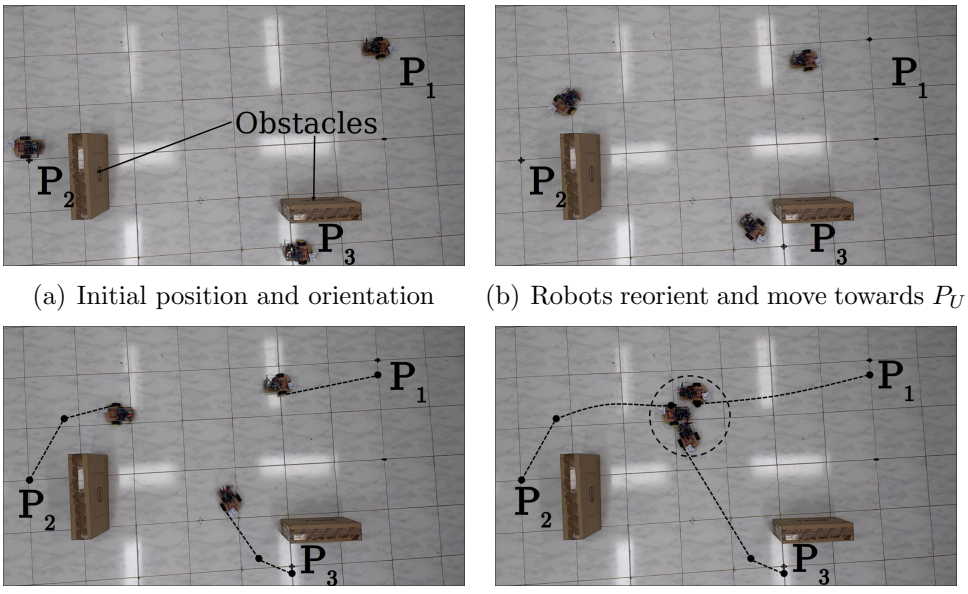
4.5 Implementation of the Algorithm on Mobile Robots

The experimental setup utilized in the implementation of these algorithms has already been discussed in section 3.5. It is worth noting that the rendezvous point computed in the presence of obstacles need not be unique. As a result, the agents could travel to different locations with the same optimal sum of distances value, preventing rendezvous. We address this with the help of communication among the agents. In particular, we allow one of the agents to collect location information of all agents, compute P_U using **Algorithm Incremental_Rendezvous_Point** and transmit the same back to all the agents. The agents can then compute their respective shortest paths to P_U and utilize **Algorithm *Path_Following*** discussed in section 3.4.4 to traverse from their initial locations to P_U. We now present various experiments that illustrate computation of P_U and effecting movement of the robots to P_U.

4.5.1 Experiments with Three Robots Amidst Obstacles

The on-board microcontrollers on each of the robots calculate the rendezvous point P_U (and the path to reach it) using **Algorithm Incremental_Rendezvous_Point**. In order to avoid any collision with the obstacles, a tolerance (of ϵ set to 15 cm in the experiments) around each obstacle is assumed and the coordinates of the 'bounding' rectangle are obtained in advance and stored in the memory of the microcontroller. This information is compared with the location data computed using input from the MOC7811 position sensor at each stage to determine the location of the robots.

Initial locations and orientations of the robots are shown in Fig. 4.7(a) along with two rectangular obstacles. The robots reorient and move towards the rendezvous point P_U as shown in Fig. 4.7(b) and 4.7(c). The path tracked by the robots and their corresponding rendezvous is marked in Fig. 4.7(d).



(a) Initial position and orientation

(b) Robots reorient and move towards P_U

(c) Robots B and C have successfully moved beyond obstacles

(d) Rendezvous of robots achieved

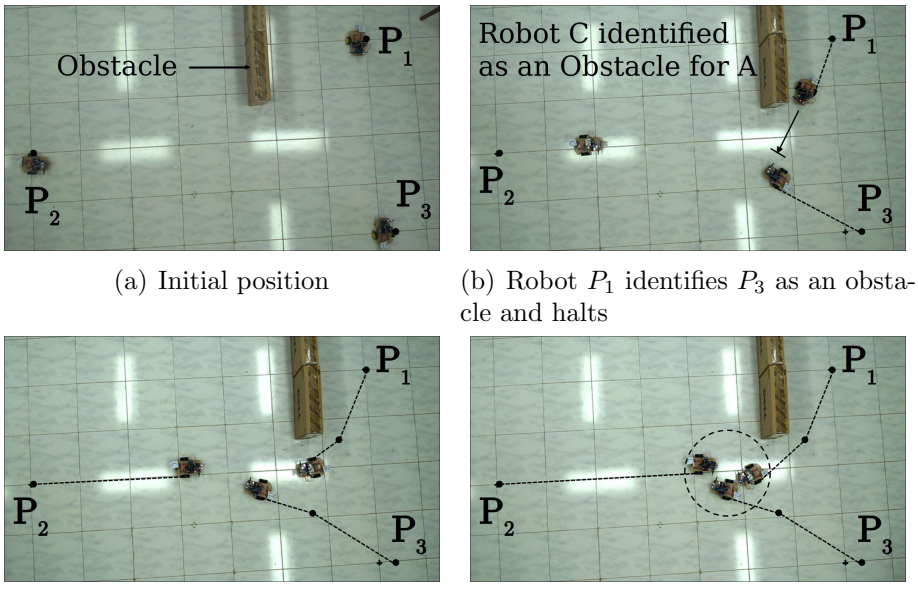
Figure 4.7: Rendezvous of three mobile robots amidst two obstacles

4.5.2 Rendezvous when a Robot Blocks another's Path

One such scenario occurs when the algorithm computes a rendezvous point such that two or more robots might have to traverse a single point on their way to the rendezvous point P_U. Each robot is equipped with ultrasonic sensors to deter-

mine this possibility. It is worth noting that the robots do not need any form of communication to handle such scenarios.

This experiment deals with a scenario where robot P_3 acts as an obstacle to robot P_1 while moving towards the rendezvous point. Initial locations and orientations of the robots are shown in Fig. 4.8(a). Fig. 4.8(b) shows how the computed path of P_3 obstructs the path of P_1. The ultrasonic sensor mounted on P_1 identifies this and halts robot P_1 to let robot P_3 to pass and avoid any collision. Robot P_1 then starts to move (Fig. 4.8(c)) towards rendezvous point P_U. It is worth noting that the robots do not halt as they come closer to the rendezvous point misidentifying other robots as obstacles. This is because the tolerance of ultrasonic sensors is gradually reduced as the robots approach rendezvous point. The final rendezvous of robots is shown in Fig. 4.8(d). It can additionally be noted that the distance constraint has not been compromised to handle this scenario.



(a) Initial position

(b) Robot P_1 identifies P_3 as an obstacle and halts

(c) Robot P_1 begins to move once P_3 is no longer an obstacle

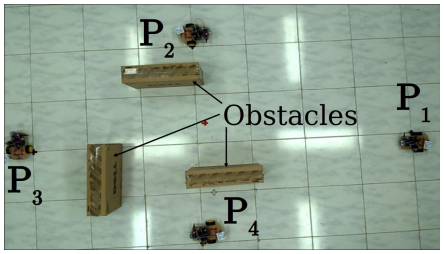
(d) Robots united

Figure 4.8: Rendezvous of three robots while P_3 acts as an obstacle to P_1

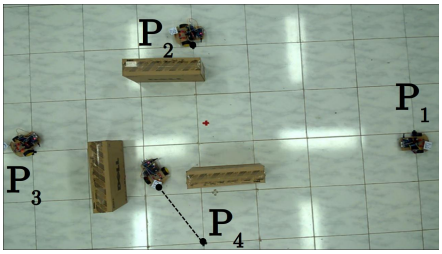
4.5.3 Experiments with Four Robots Amidst Obstacles

Three obstacles are chosen for this experiment where four robots attempt to achieve rendezvous. It is worth noting that the computation of P_U using **Algorithm Incremental_Rendezvous_Point** takes into consideration the results from section 4.2.2.

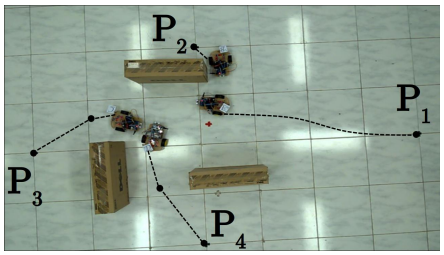
Initial location and orientation of the robots are shown in Fig. 4.9(a) and the obstacles have been marked. It can be noted from Fig. 4.9(b) that robot P_4 begins to move towards P_U. Robot P_4 obstructs the path of robot P_3 as shown in Fig. 4.9(c). Robot P_3 halts for P_4 to clear its path. The robots eventually achieve rendezvous as shown in Fig. 4.9(d).



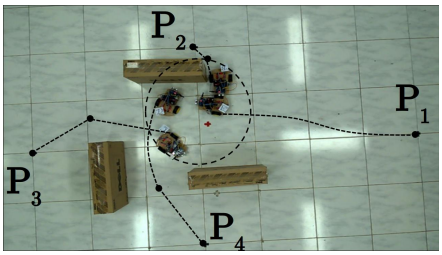
(a) Initial position and orientation



(b) Robots reorienting towards rendezvous point P_U



(c) Robots P_1, P_3, P_4 have succesfully moved beyond an obstacle



(d) Rendezvous of robots achieved

Figure 4.9: Rendezvous of four robots amidst obstacles

4.6 Summary

In this chapter, we have discussed the rendezvous of mobile agents with constraint on the total Euclidean distance travelled by the agents. We show that the computation of rendezvous point P_U amidst polygonal obstacles takes no more than $O(n^2 \log n)$ time. We have presented experiments that execute the algorithms discussed on miniature mobile robots equipped with only a microcontroller.

Thus far, we have considered scenarios where robots consider the locations of obstacles while computing the rendezvous point. In the next chapter, we deal with an intelligent obstacle (an adversary) that deliberately attempts to capture one of the agents and thus prevent rendezvous. We allow the adversary, vision support and derive the conditions under which two agents can achieve rendezvous in the presence of an adversary although travelling with inferior speeds.

CHAPTER 5

GATHERING OF FRIENDLY AGENTS AMIDST AN ADVERSARY EQUIPPED WITH A VISION SENSOR

In the previous chapters, we considered the rendezvous of multiple agents imposing a distance constraint. In this chapter, we consider rendezvous of a different kind. In particular, we assume that an autonomous agent is tasked with delivery of some goods at a prespecified destination. We call this agent as a *delivery agent*. In defence and other applications, the delivery agent may encounter an intelligent obstacle (in the form of an autonomous adversary) prior to completion of the delivery of goods. We study the problem of protection of the delivery agent (against this intelligent obstacle) via another autonomous agent (termed as the rescue agent). The delivery agent and the rescue agent are referred to as 'friendly agents' in this setting. The task of the rescue agent is to rendezvous with the delivery agent before (an undesirable) rendezvous of the adversary with the delivery agent takes place. In a sense, there is competition between the adversary and the rescue agent for rendezvous with the delivery agent.

The various aspects involved here include determining the conditions/scenarios that facilitate rendezvous of the rescue agent with the delivery agent. This chapter presents a geometric framework that identifies regions in the plane where the (rescue agent-delivery agent) rendezvous would successfully take place and where capture of the delivery agent by the adversary would happen. We begin by first detailing the motivation for the study pursued in this chapter.

Autonomous delivery agents, entrusted with the task of carrying goods from one point to another in defence zones and accident sites (Chen and Barnes (2014) and Weiss (2011)), help reduce casualty while simultaneously gathering valuable information on the environment through sensors mounted on them. Goods carried by these agents may have a marker/indicator that guides other members of the

team to recognize the contents on arrival at a prespecified destination. For example, an agent carrying medical equipment may be identified by a certain marker while one carrying food may have a different indicator. However, these markers can be exploited by an adversary. Hence, the delivery agents become vulnerable to predatory attacks prior to reaching the destination. These agents may not be necessarily equipped with hardware for taking (self) defensive action in the event of an attack.

We explore various aspects of the rescue/capture problem we address in this chapter using clues from nature regarding capabilities of the agents and the adversary. In nature, shepherds typically monitor their flock (of sheep) visually and thwart an attack by a predator taking measures based partly on the approximate distance of the predator. Similarly, a predator chooses a sheep based on sight as well as its proximity to the latter. We therefore ask the following questions: *In the autonomous setting, can an adversary successfully capture the delivery agent based on only a vision-sensor ? Also, can a rescue agent protect a delivery agent using merely wireless communication and without vision support ?*

The advantage in not having a vision-sensor on the rescue agent is reduction in hardware (and consequently power consumption). Absence of communication hardware on the adversary has similar benefits. However, limited hardware onboard presents several challenges. The rescue agent needs to gather information on the adversary through the delivery agent. Further, the adversary has to react swiftly to changes in the path of the delivery agent. We present next the definitions and assumptions required for the study.

5.1 Definitions and Assumptions

Definition 3 *For any two given agents (say P_1 and P_2) travelling with speeds (v_1 and v_2 respectively), the* ***dominance region*** *for agent P_1 is defined as the set of points in the plane where agent P_1 reaches prior to P_2. The curve that separates the dominance regions of P_1 and P_2 is defined as the* ***dominance curve****.* ■

The assumptions used for the study are as follows.

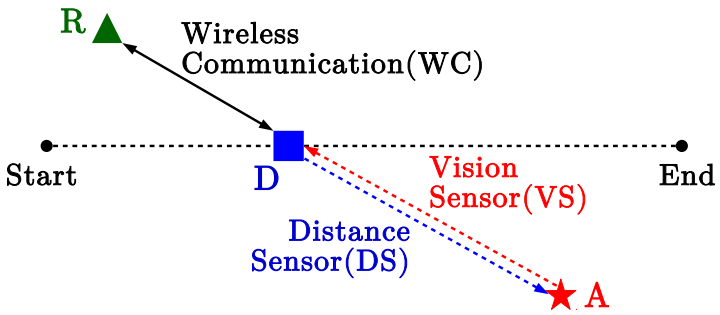


Figure 5.1: Scenario involving one delivery agent (D), one rescue agent (R) and an adversary (A). The rescue agent (R) is indicated by a green triangle, adversary (A) by a red star and delivery agent (D) by a blue square.

- Each of the three entities (delivery agent, rescue agent and adversary) is small and therefore can be approximated by a point.
- Delivery agents are assumed to travel with the least speed (denoted by v_d) in view of the cargo they carry. Rescue agents travel at a somewhat higher speed (denoted by v_r) than delivery agents.
- The delivery agent can sense an attack and communicate this to the rescue agent. The rescue agent provides 'instructions' to the delivery agent to evade the adversary.
- The communication between the delivery and rescue agents offers the latter an advantage to minimize the time for rescue. In order to counter this advantage, we assume the adversary has the maximum speed (denoted by v_a) among the three.
- Markers present on the delivery agents are recognized by the vision sensor (VS) on the adversary. All the entities are also equipped with a distance sensor (DS) for obstacle detection. No communication exists between the adversary and the agents.
- Distance sensor information is used by the adversary to recognize a rescue of a delivery agent and stop. That is, when the distance sensor on the adversary detects two objects at (nearly the) same distance, the process stops. Similarly, communication from the delivery agent (about capture) to the rescue agent will halt the rescue process.

5.2 A Geometric Framework for Solution

We present a geometric approach to address this problem. Using the framework, we provide a characterization of the *safe region* for the delivery agent and identify the outcome (namely, rescue or capture) for given initial locations and speeds of the three agents.

The identification of *safe regions* is based on the notion of Apollonius circles (Isaacs (1965)). The regions partition the plane and help efficiently determine whether a rescue or capture would take place for *different initial locations and speeds of the agents and the adversary.* We begin with a study of the interactions between the three entities, namely the delivery agent, rescue agent and the adversary.

5.2.1 Interactions between the Two Agents and Adversary

The geometric framework is developed by individually handling the two interactions: delivery and rescue agents, delivery agent and adversary.

Dominance region between two friendly agents

Delivery and rescue agent interaction (denoted by D and R respectively) is characterized by communication between the agents. Using this communication, the agents attempt to meet by deciding a location (denoted by T) and proceeding to it along a straight line path. Location T can be computed by identifying the dominance region of the delivery agent, which in turn, depends on the dominance curve (Definition 3) of the two agents.

It is worthwhile to think of the dominance curve as the locus of all points in the plane where two agents arrive simultaneously. This notion is used in Lemma 9 to compute the dominance curve for the two agents. Note that the curve given by Eq. (5.1) is indeed an Apollonius circle constructed with the same parameters (Oyler *et al.* (2016)). Our interest, however, is in computing the dominance region that is valuable to compute the meeting location (T) for the delivery agent in section 5.3.

Lemma 9 *The dominance curve for the delivery agent (D, located at (x_d, y_d)) and rescue agent (R, located at (x_r, y_r)) moving with speeds v_d and v_r respectively, is a circle (C_r) with center at (x_{dr}, y_{dr}) and radius r_{dr} as given by Eq. (5.1).* ■

$$(x_{dr}, y_{dr}) = (\frac{x_d v_r^2 - x_r v_d^2}{v_r^2 - v_d^2}, \frac{y_d v_r^2 - y_r v_d^2}{v_r^2 - v_d^2})$$
$$r_{dr} = \sqrt{x_{dr}^2 + y_{dr}^2 - \frac{v_r^2(x_d^2 + y_d^2) - v_d^2(x_r^2 + y_r^2)}{v_r^2 - v_d^2}} \tag{5.1}$$

Fig. 5.2 illustrates the dominance curve for a delivery agent D and a rescue agent R (depicted by filled blue square and green triangle respectively). The curve corresponds to C_0 with center at N_0. Since the rescue agent is assumed to be moving faster than the delivery agent, the dominance curve encloses the delivery agent. Consequently, the region (in blue) contained by this curve (denoted by S_{dr}) is the dominance region of the delivery agent while the curve itself and the region (in green) external to it (denoted by $\neg S_{dr}$) is dominated by the rescue agent.

Additionally, Fig. 5.2 illustrates the dominance curve C_t constructed at intermediate locations of D and R (denoted by D_t and R_t) on their straight line path to T. It can be observed that C_t intersects (the previously computed) C_0 at T. Such a property of the dominance curves eliminates any need for recomputation of dominance regions as the agents travel to T. This is given by Theorem 4. It is based on Isaacs (1965).

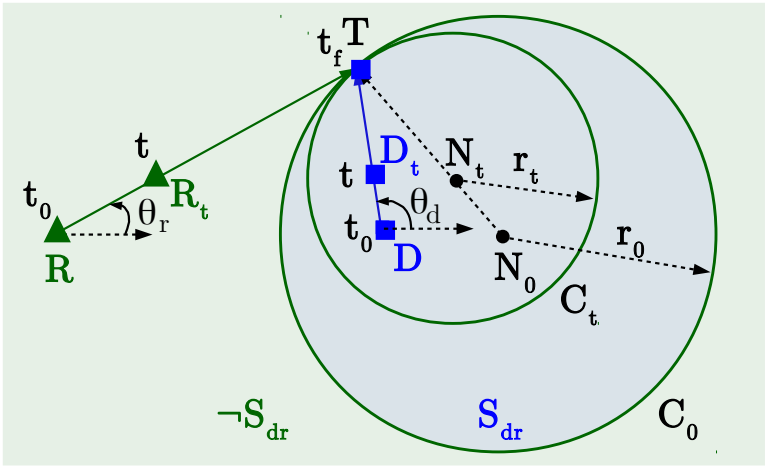


Figure 5.2: Dominance curves (C_0, C_t) for initial and intermediate locations of delivery (D, D_t) and rescue (R, R_t) agents.

Theorem 4 *Let T be the meeting point for two agents located on the dominance curve computed with their initial locations and speeds. Subsequent dominance curves constructed with locations of agents as they move towards T remain tan-*

gential to T. ■

Proof: Let initial locations and speeds of delivery and rescue agents be $\{D(x_d, y_d), v_d\}$ and $\{R(x_r, y_r), v_r\}$ respectively. The center $N_0(x_0, y_0)$ and radius r_0 of the dominance curve C_0 for these initial locations can be obtained using Eq. (5.1). Consider any arbitrary point $T(x, y)$ on the circle C_0. The intermediate locations of D and R after time t (on their way to T) are given by $D_t(x_{dt}, y_{dt})$ and $R_t(x_{rt}, y_{rt})$ in Eq. (5.2).

$$\begin{aligned} D_t(x_{dt}, y_{dt}) &= (x_d + tv_d\cos(\theta_d), y_d + tv_d\sin(\theta_d)) \\ R_t(x_{rt}, y_{rt}) &= (x_r + tv_r\cos(\theta_r), y_r + tv_r\sin(\theta_r)) \end{aligned} \tag{5.2}$$

where θ_d, θ_r are the heading angles from D, R to T.

Dominance curve C_t can now be recomputed using the updated locations D_t and R_t as a circle centered at $N_t(x_t, y_t)$ with a radius of r_t using Eq. (5.1). It follows that the two centers (N_0 and N_t) and T are collinear as given by Eq. (5.3).

$$\overline{N_0N_t} + \overline{N_tT} = \overline{N_0N_t} + r_t = \overline{N_0T} = r_0 \tag{5.3}$$

Thus, the two circles C_t and C_r meet at T. In other words, the meeting location picked up for initial locations continues to hold the property for intermediate locations as well. **Q.E.D.**

We now explore a scenario where the rescue is not characterized by the two agents occupying the same location T. Instead, it is adequate to have the rescue agent come in close proximity to the delivery agent. We quantify this proximity via the notion of a *limiting distance* as per Definition 4.

Definition 4 *The maximum distance between delivery and rescue agent, below which the delivery agent is considered to have been rescued is defined as the* ***limiting distance*** *of the rescue agent. This is denoted by d_r.* ■

The dominance curve is now given as the locus of points reached by the rescue agent where the separation between the two agents is d_r. The dominance curve

thus formed turns out to be an oval, symmetric about the line joining the two agent locations, as given by Eq. (5.4).

$$\frac{PD}{v_d} = \frac{PR - d_r}{v_r} \tag{5.4}$$

Capture region between a vision guided adversary and an agent

The adversary A uses its vision sensor to identify the delivery agent and begins its pursuit. It is worth noting that the direction of delivery agent (and not its position) is sufficient for pursuit. Since the adversary relies solely on the vision sensor (and not on communication), the path taken by the adversary is not necessarily a straight line. In other words, the adversary continuously reorients itself towards the delivery agent while pursuing it with a speed of v_a. These reorientations are separated by a finite time interval to facilitate processing of information from the vision sensor and to enforce the change in direction in a physical system. We define this time interval as the *sampling time* and denote it by t_s.

Fig. 5.3 illustrates the effect of sampling time on the capture of delivery agent. In Fig. 5.3 (a), the path taken by the adversary A (red star) in pursuit of the delivery agent D (blue square) that is heading in a fixed direction (θ_d) is shown. Reorientations at fixed intervals of $t_s = 0.4$ can be observed in the figure for six time instants. At this point, the distance between the agenta and adversary is 0.76 units. Fig. 5.3 (b) illustrates the same scenario for slightly higher sampling time ($t_s = 1.0$). As a result, the adversary repeatedly overshoots the location of the delivery agent after two time instants.

In order to define capture of delivery agents for the two scenarios given above, we revisit the notion of limiting distance. A delivery agent is considered to be captured if its distance from adversary is less than the limiting distance (denoted by d_a) of the adversary. For instance, in Fig. 5.3(a), if the limiting distance is considered to be 1 unit ($d_a = 1$), the delivery agent is considered to be captured after six time instants ($d_a > 0.76$). For the same value of d_a in Fig. 5.3(b), the closest an adversary can get to D is 2.1 units which indicates that a capture is never possible ($d_a < 2.1$).

The dominance curve can now be constructed in terms of the locus of points in the plane where the distance between delivery agent and adversary is less than the limiting distance. We use this in developing **Algorithm *DA_Dominance_Curve*** below. Additionally, it is desirable to minimize the sampling time of the adversary to prevent cases where a capture is impossible (Fig. 5.3 (b)). This can be achieved by employing a method for quick identification of the delivery agent and minimizing the delay in reorientation. Additional details are presented in section 5.4 which involve computing the Circular Hough Transform.

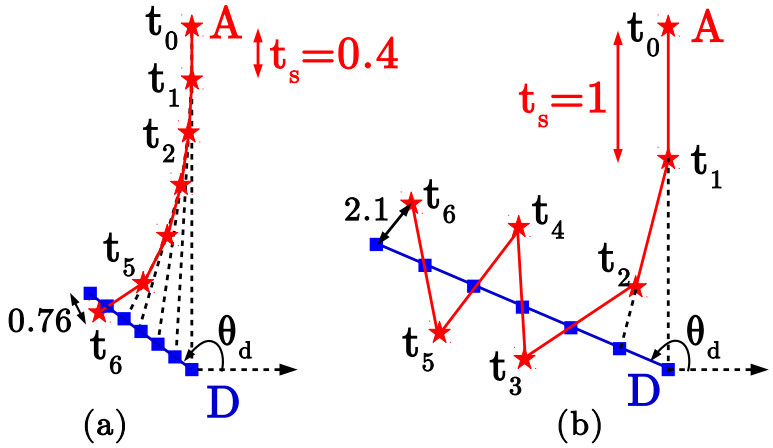


Figure 5.3: Adversary's (A) vision based pursuit of delivery agent (D) heading in a fixed direction θ_d. $v_a = 5, v_d = 2$ and $\overline{DA} = 13$. (a) t_s=0.4 (b) t_s=1.0

Algorithm *DA_Dominance_Curve*

INPUT: Distinct initial locations and speeds of adversary $(A(x_a, y_a), v_a)$ and delivery agent $(D(x_d, y_d), v_d)$. Sampling time (t_s) and limiting distance (d_a) of adversary.

OUTPUT: Dominance curve (C_a)

Step 1: Compute the heading angle ϕ for adversary using the current location of delivery agent, as given by Eq. (5.5).

$$\phi = \arctan(\frac{y_d - y_a}{x_d - x_a}) \tag{5.5}$$

Step 2: For a given heading angle θ of the delivery agent, compute the new

locations of delivery agent and adversary after time t_s using Eq. (5.6).

$$\begin{aligned} A(x_a, y_a) &= (x_a + t_s v_a \cos\phi, y_a + t_s v_a \sin\phi) \\ D(x_d, y_d) &= (x_d + t_s v_d \cos\theta, y_s + t_s v_d \sin\theta) \end{aligned} \tag{5.6}$$

Step 3: If the distance between A and D is less than d_a, conclude that a capture has occurred and proceed to **Step 4** with current location of D. Else return to **Step 1** with the new locations of A and D.

Step 4: Repeat **Steps 1-3** for all angles of θ, where $\theta \in [0, 2\pi)$. Connect all locations of D obtained in **Step 3** (at capture) and output it as the dominance curve C_a. ■

Fig. 5.4 illustrates the **Algorithm DA_Dominance_Curve** for two distinct locations of delivery agent and the adversary. One instance of pursuit with heading angle $\theta = 2\pi/3$ is shown in the figure where the capture occurs after five time instants. Due to its lower speed, dominance curve C_a once again encloses the delivery agent. Dominance region (in blue) of D (denoted by S_{da}) is the region interior to C_a while the curve itself and the region (in red) external to it is dominated by the adversary (denoted by $\neg S_{da}$).

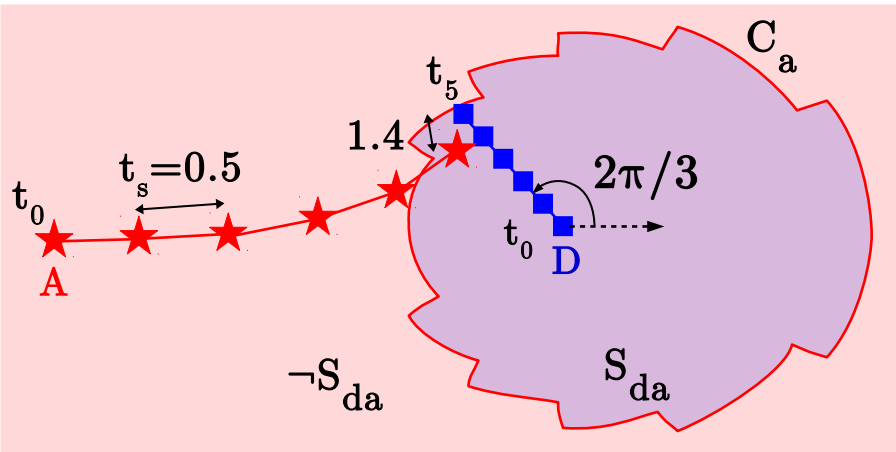


Figure 5.4: Dominance curve C_a and dominance regions $(S_{da}, \neg S_{da})$ of delivery agent and adversary. $v_a = 3, v_d = 1$, $d_a = 2.5, t_s = 0.4$ and $\overline{DA} = 20$.

It is worth noting that the dominance curve is characterized by the locations of the delivery agent (and not adversary) at capture. Further, the point of capture for a given heading angle of delivery agent remains unchanged with subsequent recomputations of the dominance curve. Consequently, the analysis performed on

initial locations of agents and adversary continue to hold until capture or rescue.

5.3 Safe regions and capture-rescue algorithms

We are now equipped with the dominance regions of delivery agent with respect to both rescue agent (S_{dr}) and adversary (S_{da}). These can be used to determine the outcome: capture or rescue. We now define the notion of a safe region.

Definition 5 *The region where a delivery and rescue agent reach prior to the arrival of adversary is termed as the* ***safe region*** *for the delivery agent and is denoted by* S_h. ■

Safe region can be computed by identifying the set of points where the rescue agent reaches prior to or along with delivery agent. However, it is to be ensured that there is no capture by adversary at these points. This observation allows us to derive a relation between the two dominance regions and the safe region. This is established via Theorem 5.

Theorem 5 Given the dominance regions of delivery agent with respect to adversary (S_{da}) and rescue agent (S_{dr}), safe region (S_h) can be computed using Eq. (5.7).

$$S_h = S_{da} \setminus S_{dr} \tag{5.7}$$

■

Proof: It follows from Definition 3 that the interior of dominance region is dominated by the entity contained in it. Since the speed of delivery agent is lowest among the three, its initial location is contained in both dominance regions. Consequently, any point inside S_{da} can be reached by delivery agent before the adversary. Similarly, any point external to S_{dr} can be used for rescue by R. The intersection of these two regions determines the safe region as given by Eq. (5.8).

$$\begin{aligned} S_h &= S_{da} \cap \neg S_{dr} \\ \implies S_h &= S_{da} \setminus S_{dr} \end{aligned} \tag{5.8}$$

Q.E.D.

Fig. 5.5 illustrates computation of safe region using Theorem 5. The dominance curves C_a (shown in red) and C_r (shown in green) are computed using **Algorithm DA_Dominance_Curve** and Eq. (5.1) respectively. The corresponding dominance regions are then used to compute the safe region S_h (shaded in blue). With the knowledge of safe region, an appropriate meeting location for R and D is picked where a rescue is attempted. It is worth noting that a part of the boundary curve C_r is included in the safe region while the entire curve C_a is excluded from it.

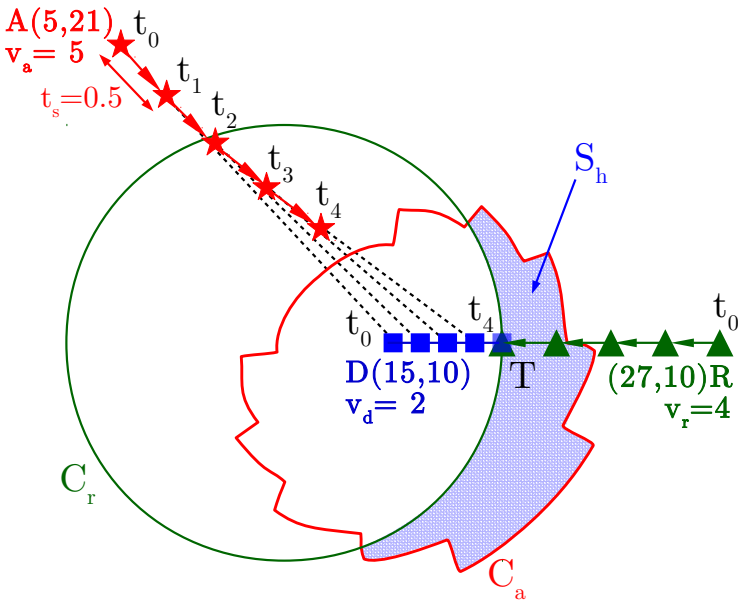


Figure 5.5: Computation of safe region S_h and a successful rescue attempt

In order to minimize the distance travelled by the delivery agent prior to rescue, T is chosen as the closest point in S_h from D. However, in the absence of a safe region, capture is inevitable. In such a scenario, it is favorable to minimize the distance between the delivery and rescue agent at the time of capture. Hence the meeting location is chosen to be the point of intersection of the line segment joining the two agents and the dominance curve C_r. We now present algorithms adopted by the adversary and agents.

Algorithm *Adversary*

INPUT: Data from vision and distance sensor. Speed v_a, limiting distance d_a and sampling time t_s of adversary.

OUTPUT: Capture or Rescue.

Step 1: Use the data from vision sensor to identify the heading direction to the delivery agent.

Step 2: Reorient the adversary towards delivery agent and proceed with speed v_a for time t_s while monitoring distance sensor readings. If the vision or path to D is obstructed, halt and proceed to **Step 4**. Else continue to **Step 3**.

Step 3: Use the distance and vision sensor information to check if the distance to D is less than d_a. If True, output that a capture has occurred and **Stop**.

Else verify if D is rescued by R. If True, output that a rescue has occurred and **Stop**. Else proceed to **Step 1**.

Step 4: Proceed with speed v_a in the direction normal to the line joining the adversary and obstruction, until the path and vision to D is clear. Return to **Step 1**. ■

Algorithm *Rescue_Agent*

INPUT: Locations and speeds of the adversary and agents. Sampling time t_s of adversary and limiting distances d_a, d_r. Distance sensor information.

OUTPUT: Capture or Rescue.

Step 1: Compute the dominance regions S_{dr} and S_{da} using Eq. (5.4) and **Algorithm *DA_Dominance_Curve*** respectively. Verify if a safe region (S_h) exists with the help of Theorem 5.

Step 2: If $S_h = \emptyset$, compute the meeting location T as the point where line joining D and R meets the curve C_r.

Step 3: If $S_h \neq \emptyset$, compute the point in S_h that is closest to D as the meeting location T.

Step 4: Communicate T to the delivery agent. Move the rescue agent to T with speed v_r while monitoring the distance sensor.

Step 5: If a capture is detected or the path is obstructed by the adversary, output that a capture occurred and **Stop**.

Else output that a rescue has occurred at T and **Stop**. ■

Algorithm *Delivery_Agent*

INPUT: Initial location and speed of delivery agent. Meeting location T from rescue agent. Distance sensor information.

OUTPUT: Capture or rescue.

Step 1: Reorient and proceed towards T with speed v_d while monitoring the distance sensor readings.

Step 2: If the distance from adversary A is less than d_a, output that a capture has occurred and **Stop**.

Step 3: If the distance from the rescue agent R is less than d_r, output that a rescue has occurred and **Stop**. ■

The pursuit by adversary and an attempt at rescue by the two agents is illustrated in Fig. 5.5. The pursuit of adversary is governed by **Algorithm *Adversary*** and continues for four time instants. The termination is caused by Step 3 of the algorithm when the adversary discovers that the delivery agent is rescued. The computation of safe region and the meeting location (T) is carried out by the rescue agent with the help of **Algorithm *Rescue_Agent***. **Algorithm *Delivery_Agent*** assists the delivery agent in arriving at T where it is successfully rescued.

In a scenario where the vision of adversary is occluded by the rescue agent, Step 4 of **Algorithm *Adversary*** allows the former to perform an evasive maneuver and regain its vision. The higher speed of adversary becomes especially useful in such a scenario. We illustrate such a maneuver of the adversary via experiments in section 5.4.

5.4 Experimental Verification with mobile robots

Multiple differential drive mobile robots equipped with Arduino UNO boards function as agents and adversary. Localization is handled by MOC7811 position encoders while ultrasonic sensors allow agents to detect an adversary. Each robot is

powered by a 12V, 1.3AH Lead acid battery. Agents communicate with the help of Xbee-PRO RF modules.

The adversary is equipped with a Raspberry Pi 3 board with an integrated Pi camera (with only 2D vision support). The red and green boxes on the delivery agents indicate their cargo. The adversary exploits these indicators by isolating the colors in the captured image and performing a Circular Hough transform (CHT). The circle computed using CHT around the closest delivery agent has been discussed in the introduction and is illustrated in yellow in Fig. 1.5(b)..

We now describe two experiments. The first experiment is depicted in Fig. 5.6. Here, a delivery agent (D) is interrupted on its path by an adversary (A). Delivery agent detects the adversary with the help of its distance sensor and communicates the same to the rescue agent. Adversary begins its pursuit as its vision sensor identifies the delivery agent at P'. In this scenario, the delivery agent is successfully rescued at T before a capture by the adversary.

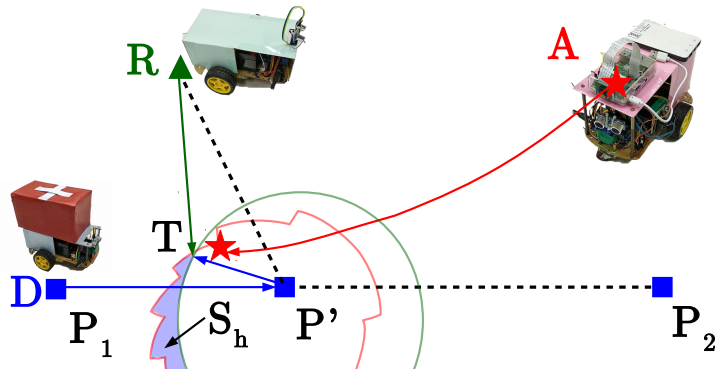


Figure 5.6: Rescue of delivery agent in its safe region

The second experiment involves obstruction of the vision of the adversary. In a scenario where the vision of the adversary is obstructed by one of the agents, the adversary adopts an evasive maneuver to restore its vision. The experiment in Fig. 5.7 illustrates this. Here, the higher speed of adversary allows it to capture the delivery agent although it was initially obstructed by a rescue agent. Note that the adversary accomplishes capture without prior knowledge of its distance to the delivery agent.

In these experiments, the pursuit by the adversary has been improved by exploiting the interrupts on its Arduino UNO board to directly obtain the heading direction from Raspberry Pi 3. Such an approach brings down the sampling time

of adversary to just 0.2 seconds. It has been shown that the proposed algorithms allow a quick computation of meeting location thus assisting in rescue of the delivery agent.

Additional experiments reveal that in some scenarios, the delivery agent is successfully rescued by proceeding towards the adversary instead of evading away. The algorithms presented naturally handle such scenarios as well.

Further, in order to avoid collision between players at capture or rescue, we assume finite limiting distance values that are greater than sum of the maximum lengths of the robots. Since we deal with identical robots that are 0.25 m long, we consider limiting distances to be equal to 0.5 m ($d_r = d_a = 0.5$) leading to an oval shaped dominance curve as given by (5.4).

It is worth noting that the meeting location does not necessarily lie on the line joining delivery and rescue agents' locations as illustrated in Fig. 5.6.

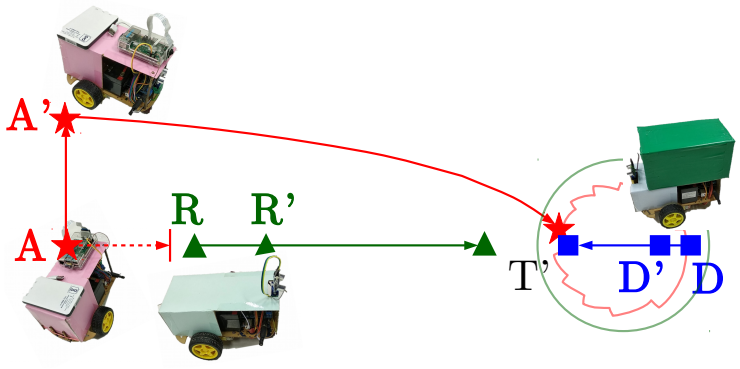


Figure 5.7: Adversary handles occlusion of vision

5.5 Summary

In this chapter, the impact of an intelligent obstacle (adversary) on the movement of an autonomous delivery agent towards its destination has been studied. The study involved understanding the dominant strategies of the adversary and efficient use of communication between the delivery and rescue agents to achieve rendezvous. Additionally, experimental verification involving miniature mobile robots has been presented. The setup requires no communication with a central controller. All the processing, including identification of the delivery agent using images captured by a Pi camera has been performed on-board in real time. The

solution technique involves discussion on dominance and safe regions for various locations and speeds of the three entities.

Having explored the distance-based constraints for rendezvous of mobile agents in the presence of static and intelligent obstacles, we now impose constraints on the time for rendezvous. This is the subject of discussion in the next chapter.

CHAPTER 6

TIME OPTIMAL RENDEZVOUS FOR MULTI-AGENT SYSTEMS

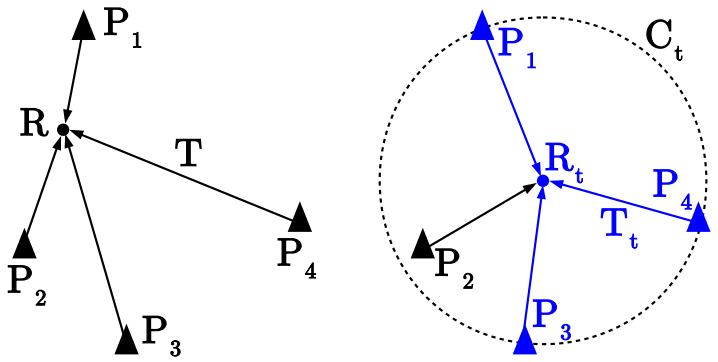
In the previous chapters, we have studied rendezvous with distance constraints and competitive rendezvous (involving a delivery agent, a rescue agent and an adversary). In this chapter, we address the rendezvous problem with a different type of constraint. In particular, we address computation of a point that can be reached by the agents in minimum time from their (given) initial locations in the presence of obstacles. We refer to this point as the Time Optimal Rendezvous Point ($TORP$) and denote it by R_t. We assume that each agent is a point mass (similar to assumption in the earlier chapters) and further the starting locations of the agents are known to all the agents. This work has been reported in Vundurthy and Sridharan (2018).

We approach this problem by observing that TORP is identical to the point in plane that minimizes the maximum time taken by any agent to arrive at the point. This leads to an algorithm to compute TORP for k agents. We then explore the location of TORP when it is computed on intermediate locations of agents, where each agent has traversed a finite time prior to arriving at these locations. We use these two results to compute the TORP for k agents moving amidst n polygonal static obstacles, followed by extending the results to handle moving obstacles.

6.1 Key Results

In this section, we present an algorithm to compute TORP for k agents in the absence of obstacles. We begin by formulating this problem as a minimax problem in travel times of agents. Such a formulation facilitates the computation of the TORP as the center of the smallest enclosing circle for all agent locations.

6.1.1 Computing TORP for k agents



(a) Rendezvous at R (b) Rendezvous at TORP, R_t

Figure 6.1: Illustration of rendezvous points and travel times for four agents

The total time (T) for rendezvous at a location (say R) is equivalent to the time taken by all agents to arrive at the location. Without loss of generality, let the agents arrive at R in the order $P_1, P_2, \cdots, P_r$, indicating that P_1 arrives at R prior to every other agent while every other agent arrives at R before P_k. Thus the time taken for rendezvous is equal to the travel time of agent P_r which is the maximum time taken by any agent to arrive at R. Further, TORP (R_t) can then be computed by comparing the times for rendezvous at every possible location in the plane and identifying the point where the minimum occurs.

Fig. 6.1 illustrates the rendezvous of four agents at two distinct locations R and R_t. In Fig. 6.1(a), the agents arrive at R in the order P_1, P_2, P_3, P_4 and thus the time for rendezvous T is the time taken by P_4 to arrive at R. Fig. 6.1(b) hints at the computation of TORP. It can be observed that the rendezvous point R_t is equidistant to P_1, P_3 and P_4. Thus the time for rendezvous (denoted by T_t) is equal to the travel time of either of these agents. Further, the time for rendezvous at R_t (in Fig. 6.1(b)) is definitively lesser than the time for rendezvous at R (in Fig. 6.1(a)).

Consider the location of agent P_2 in Fig. 6.1(b). Since the time taken by P_2 to arrive at R_t is less than the time taken by remaining agents, agent P_2 does not have any affect on the location of R_t. In fact, as long as P_2 remains within the circle C_t, it would arrive at R_t prior to the remaining agents and thus cannot affect the location of TORP. It can further be observed that any addition of new agents within the circle C_t would not affect the location of TORP either. The

following Theorem 6 utilizes these observations to compute the TORP.

Theorem 6 *The TORP (R_t) for k identical agents is located at the center of the smallest enclosing circle that contains the initial locations of these k agents.* ■

Proof: Without loss of generality, let m agents (where $m \in \mathbb{Z}^+, m < r$) lie on the smallest enclosing circle (denoted by C_t and centered at R_t) and the remaining $r - m$ agents lie within the circle C_t. The $r - m$ agents contained in the circle do not contribute to the TORP since their travel times to R_t are less than those of the n agents that lie on the circle C_t.

For the n agents that lie on the circle, consider a point P that is a finite distance away from R_t. The rendezvous time (T) taken by the n agents to arrive at P would be greater than the rendezvous time to arrive at the center of the circle R_t. Thus every other point in the circle can be discarded in lieu of the center of the circle as a candidate TORP. Consequently, the center of the circle R_t is indeed the TORP. **Q.E.D.**

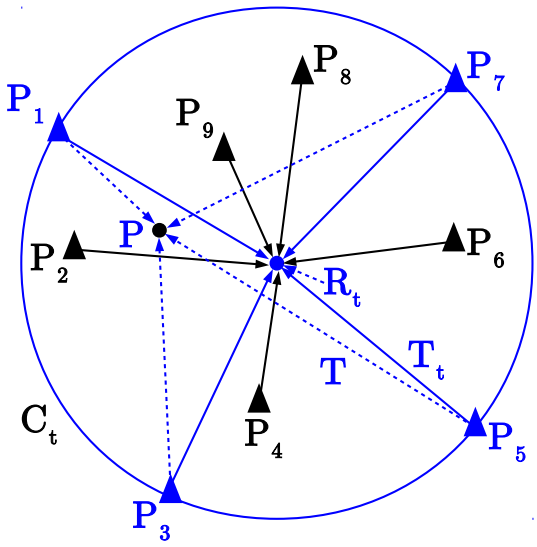


Figure 6.2: TORP for 9 agents is the center of the smallest enclosing circle C_t

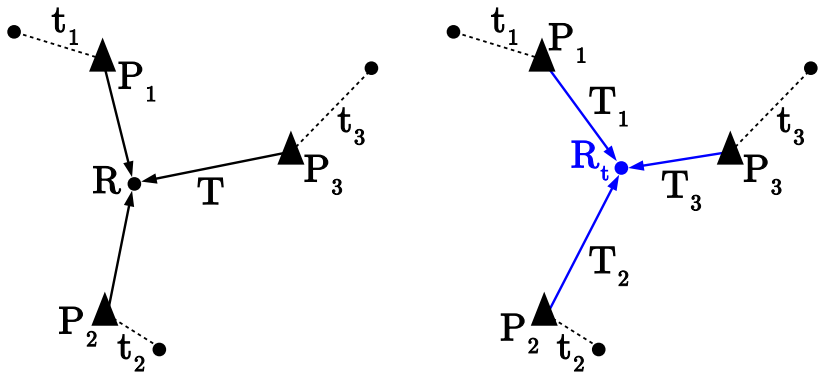
Fig. 6.2 illustrates Theorem 6 for 9 agents. Agents P_1, P_3, P_5, P_7 lie on the smallest enclosing circle ($m = 4$) while the remaining agents lie within the circle C_t. Due to their lower travel times to R_t, the remaining 5 agents (shown in black) do not affect its location. For an arbitrary rendezvous point P, agent P_5 takes the longest time (T) to arrive at P which is the rendezvous time for all agents. Since this time T is greater than the rendezvous time (T_t) to the center of the circle, the

point P is ignored as a candidate TORP. With a similar analysis, every point in the plane has a longer rendezvous time compared to the center of the circle which proves that the TORP is indeed the center of the circle (Theorem 6).

In the following section, we extend this analysis to a scenario where the agents have elapsed a finite time before arriving at the locations that are used in computing the TORP.

6.1.2 Computing TORP using intermediate locations of k agents

In this section, the computation of R_t is performed on the locations of k agents given by $\{P_1, P_2, \cdots, P_r\}$, while taking into account the respective times (denoted by $\{t_1, t_2, \cdots, t_r\}$) the agents spend in arriving at these locations. We refer to the time elapsed as the *weight of an agent* at a given location. The solution presented in the previous section 6.1.1 turns out to be a special case of this problem with zero weights at all agent locations.



(a) Rendezvous at R (b) Rendezvous at TORP, R_t

Figure 6.3: Illustration of rendezvous points for three agents with non zero weights

Fig. 6.3 illustrates the rendezvous of three agents located at P_1, P_2, P_3 with their associated weights t_1, t_2, t_3 respectively. It is worth noting that the information on locations of these agents prior to their arrival at P_1, P_2, P_3 is unknown and the figure illustrates only one among infinite possibilities of such locations. An arbitrary rendezvous location R is picked in Fig. 6.3(a). It can be observed that agents P_1 and P_2 arrive at R prior to the arrival of P_3. Thus the rendezvous time

in this scenario is equivalent to the travel time of agent P_3 which is $T + t_3$.

In order to obtain the TORP, it is desirable to minimize the maximum time of arrival at a given rendezvous point. This is achieved by observing the weights at each agent location and ensuring that the rendezvous point be closer to the agent that has the highest weight. It is further desirable to verify the existence of a location where the agents can arrive simultaneously, as illustrated in Fig. 6.3(b). Such a location (if it exists) would have the following property given by (6.1). In the absence of weights, such a location would be the center of the smallest enclosing circle (Theorem 6). We utilize this to present Theorem 7.

$$t_1 + T_1 = t_2 + T_2 = t_3 + T_3 \tag{6.1}$$

Theorem 7 *Consider k circles with their centers located at agent locations* $P_1, P_2, \cdots, P_k$ *and their radii equal to the weights at each location* $t_1, t_2, \cdots, t_k$.

The TORP (R_t) *for these k agents with their respective weights is the center of the smallest enclosing circle that contains each of these k circles.* ■

Proof: Given the weight and agent location, the locus of points that can be reached from the agent location in fixed time constitutes a circle with center at the agent location and radius equal to its weight. This circle indicates all possible initial locations for an agent to arrive at P_i in time t_i $\forall i \in \{1, 2, \cdots, k\}$.

Given the initial locations of agents, TORP can be computed as the center of the smallest enclosing circle containing all the initial locations, as given by Theorem 6. Consequently, TORP for this problem can be computed as the center of the smallest enclosing circle that contains all the circles constructed at agent locations. **Q.E.D.**

We now present Lemma 10 which restricts the number of agent locations that constitute the smallest enclosing circle. However, the following mathematical facts are necessary in constructing the proof for lemma. For any two circles centered at C_1 and C_2 with radii r_1 and r_2, circle at C_1 is contained in circle at C_2 if (6.2) is satisfied. Additionally, the locus of points P where two agents located at P_1 and P_2 (with weights t_1 and t_2) arrive simultaneously turns out to be a hyperbola as

given by (6.3), where c is a constant.

$$\overline{C_1C_2} \leq r_2 - r_1 \tag{6.2}$$

$$\overline{PP_1} + c \times t_1 = \overline{PP_2} + c \times t_2 \tag{6.3}$$

Lemma 10 *The smallest enclosing circle C_t for k circles with non-zero radii requires a maximum of three circles for its construction. The remaining circles either lie in the interior of C_t or are tangential to its boundary.* ■

Proof: It follows from (6.1) that the center of the smallest enclosing circle is located such that the associated agents arrive at it simultaneously. Such a center is the point of intersection of hyperbolas constructed with the locations of associated pairs of agents as given by (6.3). However, only a maximum of three hyperbolas can intersect at a single point in the plane (excluding degeneracy) which proves the first statement of the lemma.

Among all the points of intersections of hyperbolas, the point (say R_t) which maximizes the rendezvous time (T_t) is the TORP. Further, this rendezvous time is the radius of the smallest enclosing circle, centered at R_t. It thus follows from Theorem 7 that every other circle is contained (internal or tangential) in the circle C_t. **Q.E.D.**

We utilize Theorem 7 and Lemma 10 to present an algorithm that accepts the agent locations and weights as input and computes the TORP. We adopt an incremental approach by beginning with the largest circle, identifying the circles that lie external to it and gradually increasing its radius to enclose all the remaining circles.

Algorithm *Min_Time_Weights*

INPUT: Locations of all k agents $P_1, P_2, \cdots, P_k$ and their corresponding weights $t_1, t_2, \cdots, t_k$.

OUTPUT: Time Optimal Rendezvous Point (TORP), R_t and the time for rendezvous, T_t.

Step 1: Construct k circles with centers at agent locations and radii equal to their weights. Initialize an empty set S.

Step 2: Evaluate R_t and T_t as the location and weight of the agent with the

highest weight and add its location to set S.

Step 3: For a circle C_t centered at R_t with a radius of T_t, use (6.2) to identify the agent whose circle is not contained in C_t. Add the agent location to S.

If all agents' circles are contained in C_t, output the current values of R_t and T_t. **Stop**.

Step 4: Evaluate R_t and T_t using **Step 5** on the set S and return to **Step 3**.

Step 5: If S has only two agent locations, evaluate R_t as the point of intersection of line segment joining them and the hyperbola constructed using (6.3).

If S has three agent locations, overwrite R_t with the point of intersection of three hyperbolas constructed using (6.3) and T_t with its corresponding weight.

If S has four agent locations, identify the triplet whose point of intersection of hyperbolas has the highest weight. Overwrite R_t with this point, T_t with its weight and remove the remaining agent location from S. ■

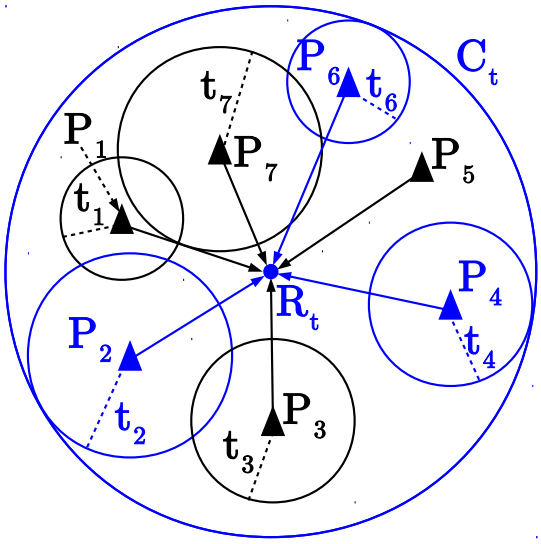


Figure 6.4: TORP (R_t) for 7 agents with non-zero weights at their locations

Fig. 6.4 illustrates **Algorithm *Min_Time_Weights*** for 7 agent locations with their corresponding weights. The algorithm begins by constructing k circles as shown and picks agent P_2 that has the maximum weight. Step 3 of the algorithm identifies that the circle at P_3 lies outside the previous circle at P_2. Set S currently includes P_2 and P_3 which is used in computing and updating TORP in Step 5. Further iterations slowly increases the radius T_t of the smallest enclosing circle eventually computing C_t as shown. The final elements of the set S are P_2, P_4 and P_6 (shown in blue) which are used in constructing R_t in Step 5.

Remark 11 *The enclosing circle grows in size as it absorbs one circle after another. However, it is worth noting that a circle that has been absorbed does not exit the enclosing circle while the latter attempts to absorb the circle due to another agent. In other words, the size of the enclosing circle increases monotonically until all the remaining agents' circles are enclosed.* ***Algorithm Min_Time_Weights*** *utilizes this to improve its complexity and the proof for the same is based on Elzinga and Hearn (1972).*

6.2 Algorithm to Compute TORP for Multiple Agents Amidst Static Obstacles

In this section, we compute the point that minimizes the total time for rendezvous for k agents as they negotiate n polygonal obstacles. In order to minimize the time for travel between two locations amidst obstacles, an agent computes and follows the shortest path from one location to another. This shortest path amidst n polygonal obstacles can be efficiently computed using the algorithm presented in Hershberger and Suri (1999). We use this along with the results presented so far to develop the following Theorem 8 that computes the TORP.

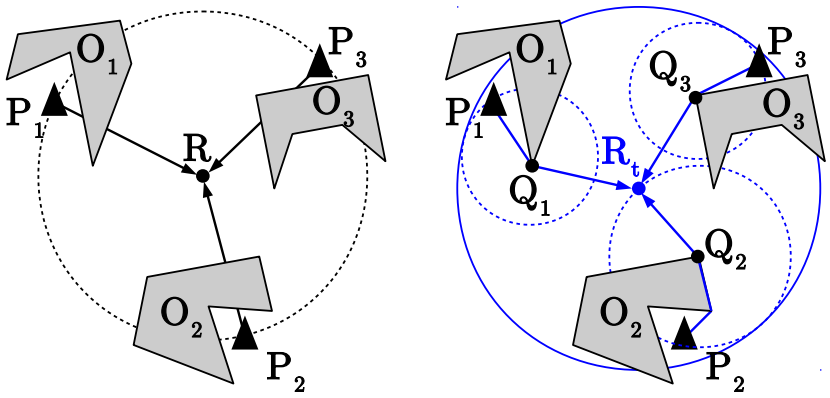
Consider k agents denoted by $P_1, P_2, \cdots, P_k$ moving amidst n polygonal obstacles. Let R be the rendezvous point computed using **Algorithm *Min_Time_Weights*** with initial locations and zero weights. Let $Q_1, Q_2, \cdots, Q_k$ represent the agent locations before arriving at R when the agents take the shortest path from their initial locations (to R). Let $t_1, t_2, \cdots, t_k$ be the corresponding time taken by each agent.

Theorem 8 *TORP (R_t) for these k agents amidst n obstacles is the rendezvous point computed on the locations $Q_1, Q_2, \cdots, Q_k$ with their corresponding weights $t_1, t_2, \cdots, t_k$, using* ***Algorithm Min_Time_Weights***. ■

Proof: The minimum time for rendezvous for k agents in the absence of obstacles occurs at R, as given by Theorem 6. Any deviation from the path to R increases the time for rendezvous. Since the deviation is minimum along the shortest path

in the presence of obstacles, the locations given by $Q_1, Q_2, \cdots, Q_k$ are common to the paths taken by agents to arrive at both R and the TORP, R_t.

Further, the paths from agent locations $Q_1, Q_2, \cdots, Q_k$ to either R or R_t are not obstructed by any obstacle. It follows from Theorem 7 that the center of the smallest enclosing circle with corresponding weights would have a lower time for rendezvous than any other point in the plane, including R; which concludes that the center is indeed TORP (R_t). **Q.E.D.**



(a) Obstacles on the paths to R (b) TORP obtained using Theorem 8

Figure 6.5: Computation of TORP for three agents amidst three obstacles

Fig. 6.5 illustrates rendezvous of three agents amidst three polygonal obstacles. The agents begin their attempt at rendezvous by computing the TORP using **Algorithm *Min_Time_Weights*** with zero weights at initial locations. The obstructions on the path to this rendezvous point R are shown in Fig. 6.5(a). In order to negotiate these obstacles, the agents have to deviate from their straight line path to R. Such a deviation necessitates a recomputation of TORP as indicated in Fig. 6.5(b).

The agents arrive at locations Q_1, Q_2 and Q_3 and recompute TORP using **Algorithm *Min_Time_Weights*** with weights equal to the time elapsed by the agents to arrive at these locations. The weights are represented by the radii of dashed circles. The new rendezvous point (R_t) that minimizes the time for rendezvous and the corresponding smallest enclosing circle are illustrated in Fig. 6.5(b). This is condensed into the following algorithm which is based on Theorem 8. An illustration for higher number of agents is presented via an experiment in section 6.4.

Algorithm *Min_Time_Obstacles*

INPUT: Locations of all k agents and n polygonal obstacles. Weights at initial locations.

OUTPUT: Time Optimal Rendezvous Point (TORP), R_t and the paths of all agents to R_t.

Step 1: Compute the rendezvous point R for k agents using their locations and corresponding weights with the help of **Algorithm *Min_Time_Weights***.

Step 2: Compute the shortest paths for all agents from their initial locations to R using Hershberger and Suri (1999). Identify Q_i as the last vertex of obstacle visited by P_i before arriving at R along the shortest path, where $i \in \{1, 2, \cdots, k\}$. If the shortest path for an agent P_i is not obstructed, identify Q_i as the agent location P_i.

Step 3: Compute TORP R_t using **Algorithm *Min_Time_Weights*** with locations $\{Q_1, Q_2, \cdots, Q_k\}$ and weights equal to the time taken by each agent to travel from P_i to Q_i. **Output** R_t and corresponding shortest paths leading to R_t.

■

Algorithm *Min_Time_ Obstacles* is not only useful in the initial locations of the various agents. It can also be employed for recomputation of TORP after the agents have moved some distance from their initial locations. However, the usefulness of **Algorithm *Min_Time_ Obstacles*** is limited to the setting where we have only static obstacles. This is expressed by Theorem 9.

Theorem 9 *Recomputation of TORP (R_t) using* ***Algorithm Min_Time_ Obstacles*** *at intermediate locations of agents after they have traveled for a finite time, does not affect its location.* ■

Proof: Time optimal rendezvous point is computed by finding the point of intersection of hyperbolas (given by (6.3)) at agent locations while taking their weights into consideration. This corresponds to Step 5 of **Algorithm *Min_Time_ Weights***. Let the recomputation be performed at intermediate locations when every agent has traversed a finite time t_f. A constant factor of t_f thus appears in weights of agent locations as given by (6.4).

$$\overline{PP_1} + c \times (t_1 - t_f) = \overline{PP_2} + c \times (t_2 - t_f) \tag{6.4}$$

Since (6.4) evaluates to (6.3), there is no change in the point of intersection of hyperbolas and thus the TORP remains unaffected. **Q.E.D.**

6.3 Extension to Handle Moving Obstacles

In an industrial setting, the agents would have to negotiate obstacles like such as Automated Guided Vehicles (AGVs) or humans, en route to TORP. It is thus of interest to design an algorithm to ensure rendezvous, even when one or more agents are obstructed by such moving obstacles.

Whenever an agent is obstructed by a moving obstacle, the former waits until its path is cleared. The time spent by an agent in waiting is not uniform across all agents. Thus, **Algorithm *Min_Time_ Obstacles*** is not adequate to handle the case where moving obstacles are also present. It is necessary to recompute the TORP by taking into account the travel times of various agents and the time elapsed in waiting. The modified scheme for computation of TORP amidst moving obstacles is given by **Algorithm *Moving_Obstacle_Handling*** next.

Algorithm *Moving_Obstacle_Handling*

INPUT: Initial locations of all agents and static obstacles. Time Optimal Rendezvous Point (TORP) computed with only the static obstacles, the paths of all agents to TORP (R_t) and distance sensor information.

OUTPUT: Rendezvous of all agents.

Step 1: Allow each agent to proceed on its path to R_t until faced by a moving obstacle or the agent arrives at R_t. If all agents arrive at the same location, **Stop**.

Step 2: If a moving obstacle is detected, halt the agent and communicate the current location of the agent along with the time elapsed in traveling and waiting, at fixed intervals. Request and receive this information from all other agents.

Step 3: Recompute and update the TORP (R_t) with the current locations of all agents and their corresponding weights (time elapsed), using **Algorithm *Min_Time_Obstacles***.

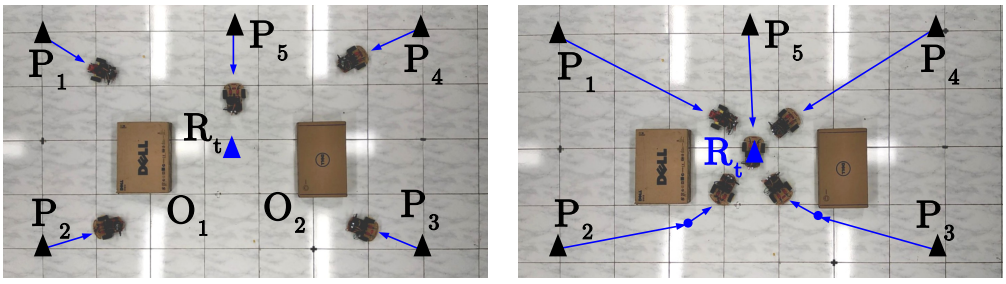
Step 4: Compute and update the shortest paths for all agents from their current locations to R_t (computed in **Step 3**). Proceed to **Step 1**. ■

6.4 Experimental Validation of Algorithms

The hardware realization of algorithms presented thus far is achieved with the help of small differential drive mobile robots discussed in section 3.5. Each robot is equipped with an Arduino UNO board featuring an ATmega328P microcontroller to control the motion of robot and to compute the TORP with the location information on agents and obstacles. The communication between agents for exchanging information on location and elapsed time is achieved with the help of Xbee-PRO RF modules that operate at 2.4 GHz.

Detection of moving obstacles is achieved with the help of ultrasonic range detection sensors mounted on micro-servo motors. The localization of robots is attributed to MOC7811 speed sensor mounted on each wheel of the robot. MOC7811 is an inexpensive opto-coupler that provides adequate accuracy while eliminating any necessity for a motion capture system.

Various experiments have been performed to validate the proposed algorithms, two of which are presented here. Fig. 6.6 illustrates the first experiment where five agents are considered for rendezvous amidst two rectangular obstacles. Agents employ **Algorithm *Min_Time_Obstacles*** to compute the TORP, R_t. **Algorithm *Moving_Obstacle_Handling*** is then used by the agents to travel along their shortest paths to their destination R_t. Intermediate locations of agents are shown in Fig. 6.6(a) while their rendezvous is illustrated in Fig. 6.6(b).



(a) Agents proceed towards R_t (b) Rendezvous of agents at R_t

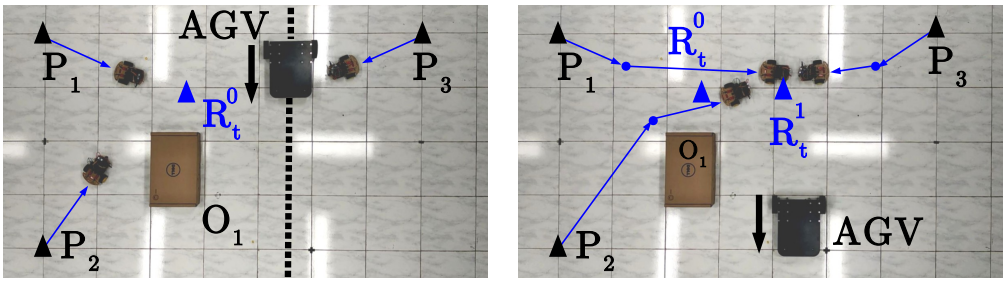
Figure 6.6: Time-optimal rendezvous of five agents amidst two obstacles

In the second experiment (Fig. 6.7), we allow an AGV to obstruct the path of agent P_3 as three agents attempt to rendezvous at R_t^0 in the presence of one polygonal obstacle. The ultrasonic sensor on agent P_3 detects the AGV as a moving obstacle and invokes Step 2 of **Algorithm *Moving_Obstacle_Handling***.

Table 6.1: Comparison of various features of proposed algorithms with prior works involving time-optimal rendezvous

Criteria →	Static and Moving Obstacles	Identical Agents	Hardware Realization	Complete Location Information
Setter and Egerstedt (2014) Chunhe and Zongji (2014) Brown *et al.* (2011)	No	No	No	Yes
Notarstefano and Bullo (2006) Bhatia and Frazzoli (2008)	No	Yes	No	Yes
Kunwar *et al.* (2005)	Yes	One	Yes	No
Proposed	Yes	Yes	Yes	Yes

Once the current location information and the waiting times of all agents are communicated to each other, TORP is recomputed. While the agents P_1 and P_2 keep moving to the current $TORP$, agent P_3 requests for recomputation until the AGV clears its path. It can be observed that the final rendezvous occurs at R_t^1.



(a) P_3 detects moving obstacle (b) Rendezvous at R_t^1

Figure 6.7: Time-optimal rendezvous for three agents amidst one static and one moving obstacle (AGV)

Experimental results reflect the ability of proposed algorithms to quickly compute the TORP on just a microcontroller without any support from a central hub. The computation of shortest path from agents' locations to TORP is also parallelizable by allowing each agent to perform its own computation. Additionally, when handling a moving obstacle, the computation is performed in fixed intervals to further minimize the waiting times of agents.

6.5 Comparisons

The algorithms presented in this chapter have been compared with prior work in Table 6.1. Prior work considering obstacles is, in general, limited. While there have been attempts to achieve time optimal rendezvous in cluttered environments Kunwar *et al.* (2005), the study (and experiments) are limited to a single autonomous vehicle attempting a rendezvous with moving targets. In order

Table 6.2: Comparison of various aspects of experimental setup

Criteria ↓	**Kunwar *et al.* (2005)**	**Proposed**
Number of agents	2	k where $r \in \mathbb{Z}^+, r \geq 2$
Communication with Central Computer	Yes	No
Localization	CCD Camera	On-board Encoders
Processing Support	Central Host Computer	On-board Microcontroller

to enhance our algorithm to handle moving obstacles, we allow the agents to communicate when faced by a moving obstacle and recompute the TORP by taking into account their waiting time, as discussed in section 6.3. Table 6.2 presents a comparison of the key features of experiments in our work and in Kunwar *et al.* (2005).

6.6 Summary

In this chapter, we have explored time optimal rendezvous of multiple agents. In the next chapter, sensor specific rendezvous in the absence of location information is presented. Rendezvous of two heterogeneous agents, namely a bipedal robot and a mobile robot, is shown via experiments.

CHAPTER 7

RENDEZVOUS OF HETEROGENEOUS ROBOTS AMIDST UNKNOWN OBSTACLES WITH LIMITED COMMUNICATION

In the earlier chapters, we have studied rendezvous amidst obstacles and various constraints. In this chapter, we consider sensor-based rendezvous of a pair of robots. In particular, we study *rendezvous* of a pair of mobile robots equipped merely with infrared (IR) beacons amidst obstacles. While rendezvous has been explored in general in the literature, work on rendezvous with specific constraints on the hardware carried is limited. Further, analysis of the capabilities of the hardware (for communication, sensing) to facilitate rendezvous in an environment with obstacles has been scarce. Our algorithm is applicable to heterogeneous robots as well and we report experiments on rendezvous with a pair of heterogeneous robots fabricated locally. Our experiments involve a mobile robot and a bipedal robot.

There are several advantages of rendezvous using merely IR beacons. First, no knowledge of initial positions of the robots is required. Second, precise knowledge of the location of obstacles in the environment is also not required. We assume point-size (or small) robots and determine paths for the robots based on reflection of beams from the IR beacons. For larger-size robots, collision avoidance can be handled using additional sensors (for example, ultrasonic sensors). Further, with the IR beacon approach, one can devise an algorithm that obviates the need for additional hardware for communication (such as bluetooth) even if both the robots move to achieve rendezvous.

A challenge posed by IR is the ambience. The beams tend to reflect from the surrounding walls and create an illusion of an additional robot in the vicinity. One solution to this problem is manipulation of the ambience by absorbing the waves instead of reflecting them. However, robots usually find their place in laboratories amongst other devices (and obstacles) so it is difficult to manipulate the ambience.

The proposed approach addresses this challenge by adapting to the environment via calibration to the surroundings. In particular, the intensity of the lighting determines a threshold value to be set and this corresponds to determining if the signals from the IR beacons need amplification. Experiments on rendezvous of a biped and a mobile robot (fabricated in our laboratory) are presented to validate the proposed approach. This work has been reported in Vundurthy *et al.* (2016).

7.1 Assumptions and Terminology

We begin with a few assumptions necessary in developing algorithms for this problem. We then present the terminology required.

7.1.1 Assumptions

1. The robots operate in an indoor environment.
2. There are six transmitters on each robot, each having a transmission angle of 60°, thus covering the entire 360°.
3. There exists at least one path between the two robots to rendezvous.

7.1.2 Terminology

While the transmitters cover 360°, the entire area within the boundary does not necessarily receive the signal directly from the transmitter. Since we are working with IR beams, the signal may reflect off of many surfaces before it becomes incident on a given region. We represent regions based on the number of reflections a ray undergoes before reaching it.

The space in which the robots are operating in the presence of obstacles and polygonal walls is denoted by 'S'. The subset of 'S' over which the IR beam reaches without any reflection is termed as 'Zone 0'. The area over which the IR beam reaches after one reflection from any of the walls or obstacles is termed as 'Zone 1'. The receiver can detect an IR beam only above a certain intensity. Thus, the maximum number of zones will depend on the maximum number of reflections from various surfaces, walls and obstacles before the incident IR beam loses its

minimum detectable intensity. This value of minimum detectable intensity will depend on the intensities of other light sources in the room along with the kind of reflective surfaces. This is termed as the Threshold value (T_v). The maximum value of zone numbers is defined as the 'Reflective index' and is denoted by R_n.

We denote the robots by A and B. Both the robots A and B have transmitters and receivers which can cover the entire 360° range. The transmitters on A and B are denoted by T_A and T_B respectively. The receivers on A and B are denoted by R_A and R_B respectively. We now present the key results relating to the IR transmission based on which the rendezvous takes place.

7.2 Key Results Pertaining to IR Transmission

Lemma 11 *The change of zone can happen only due to the obstruction of IR beam by an obstacle.* ■

Proof: In the absence of any obstacles, the IR beam with a 360° range of emission covers the entire room with its rays even before the rays hit any walls. When an obstacle is added, the area behind the obstacle as seen from robot A 'becomes dark' with respect to IR beam. However, the reflections from other walls are present and a part of this area lights up. This area is Zone 1. The border between Zone 0 and Zone 1 is thus due to the vertex/edge of the obstacle. **Q.E.D**

Remark 12 *Fig. 7.1 gives an example of various zones in a rectangular room. The robot A (with its transmitters) is placed at (2,2) as indicated in the figure. A line segment obstacle is considered to be extending from Ob_1 to Ob_2. The area in green is Zone 0, area in blue is Zone 1 and area in red is Zone 2. Lemma 11 thus follows from the figure where the boundary of Zone 0, 1 and 2 are marked by lines extending from the vertices of the obstacle.* ■

For describing the effect of the IR beams, we assume that robot A is transmitting while B is receiving. The second robot B is placed at some arbitrary location in the room. Depending on the zone that it is placed in, the action it should

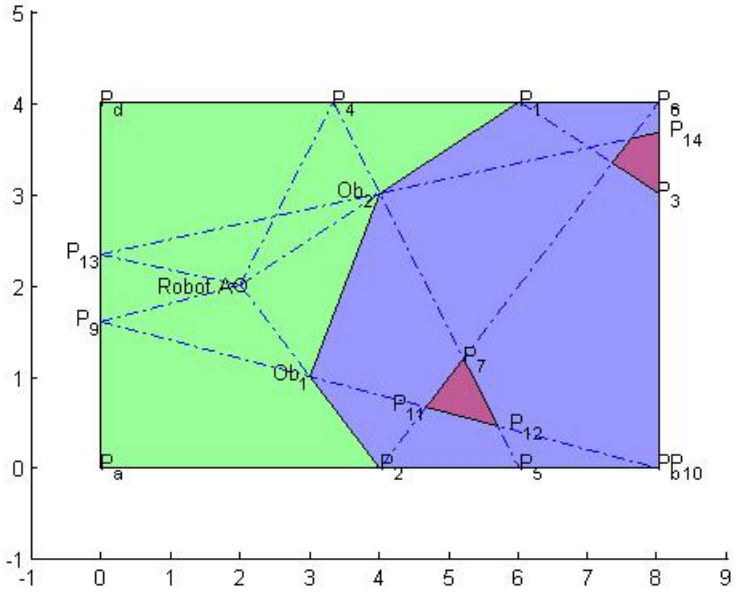


Figure 7.1: Various zones for a transmitter placed at A in the presence of a line segment obstacle Ob_1Ob_2

take will vary. However, we are using only an IR receiver as the sensor and thus the robot itself has no knowledge of the zone it is in. The receiver is capable of recording the direction in which the rays hit it. This direction is then used to orient itself and take further steps. We now present another result that is useful in the development of our algorithm.

Lemma 12 *All the consecutive zone numbers till R_n (the reflective index) need not be present in S.* ■

Proof: The proof is by contradiction. In particular, assume all the consecutive zone numbers till R_n are present in a region S. Now consider the region shown in Fig. 7.2. The boundary of region S and the location of obstacle are such that there cannot exist any subset of S that qualifies as Zone 1. Thus S only contains Zone 0 and higher zones. The reflective index R_n is 3 in this case. Thus, all the consecutive zones do not exist in S. **Q.E.D.**

Theorem 10 *In a given region S, let there be n zones with k deficient zones. The zones in increasing order are adjacent to each other and IR beam enters a zone i through one of its immediate (prior) zones.* ■

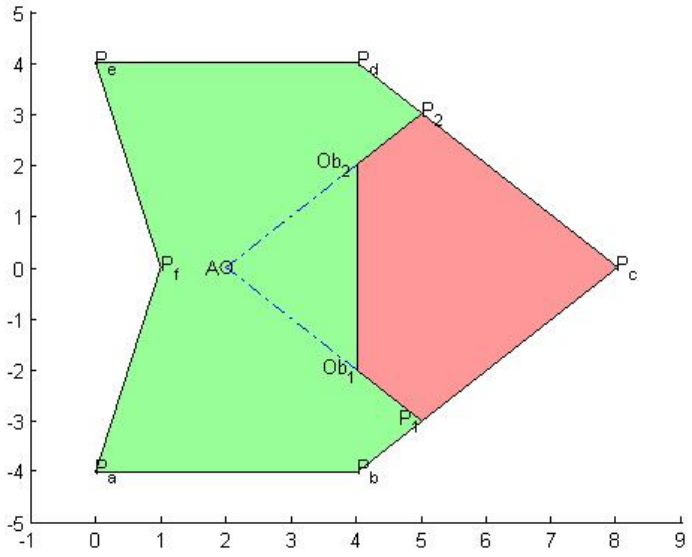


Figure 7.2: Absence of Zone 1 in this setting

Proof: Let the zones be numbered $Z_1, Z_2, ...Z_{i-1}, ...Z_{i+1}, ...Z_n$ with k missing zones. Let a polygonal region A be defined such that the polygonal edges of this region are due to the reflections sliding through the edge or vertex of an obstacle. It follows from Lemma 11 that the IR beam that slides along the vertex of an obstacle divides the region into two halves with different zones. One half consists of rays incident from beyond the obstacle and thus with a zone number say Z_i. The other half cannot have any rays incident in the previous manner due to the obstacle itself. However, IR beam comes to this region due to multiple reflections from other walls or obstacles. This gives the other half region a new zone number Z_j. Thus, by construction, $i < j$. Additionally, if there is more than one way of approaching this second region like $Z_{j1}, Z_{j2}, ...$, we define the zone number with the lowest of these values. However, the lowest of these values will still be greater than Z_i. Thus, if Z_{k-1} and Z_{k+1} are consecutive zones (in other words zone Z_k does not exist), they have to be adjacent to each other. The second statement of Theorem 10 is a consequence of this result. Since zones with increasing zone value are adjacent to each other, any ray entering a zone with higher value will have to come through a zone with lower value. **Q.E.D.**

Remark 13 *Taking Fig. 7.1 as an example, let the region A be defined by the* $\triangle$ $P_7P_{11}P_{12}$ *. It can be noticed from the figure that each edge of A is a consequence of*

IR beam grazing along the vertex of the obstacle. Additionally, these beams divide the space into two halves one from each zone. The zone in blue is Zone 1 and zone in red is Zone 2. In the case of any overlap of zones, the zone is given a number with lower value. Thus the regions $P_2P_5P_{12}P_{11}$*,* $P_5P_{10}P_{12}$*,* $P_7Ob_2Ob_1P_{11}$ *are reverted back to Zone 1 in spite of being capable of having an IR beam with two reflections reaching it.*

It follows from Lemma 11 and Theorem 10 that the IR beam grazing the vertex of an obstacle can create a change of zones in a consecutive manner. Thus a robot can always move from Zone Z_i to Z_{i-1} (if it exists or else the next lower zone) in a straight line path.

7.3 Proposed Algorithm for Rendezvous

The proposed algorithm involves adaptation of the robot to the surroundings especially in view of IR beacons on-board the robot. Further, we need an algorithm that determines the direction to be chosen for motion of a robot. Therefore, sub-algorithms for adaptation and action (direction determination) are presented first. This is followed by the main algorithm for rendezvous.

7.3.1 Adaptation Algorithm

In the rendezvous problem with IR beacons, as the robots plan and move towards each other, the transmitters and receivers of the robots should not interfere with each others' functions. The adaptation algorithm helps to know the bound on time after which a signal would die down (go below the detectable level of the receivers). With the help of this bound, we can turn on the transmitter of one robot along with the receiver of another. After the time given by the adaptation algorithm, the receiver of one robot can be turned on along with the transmitter of another. In this way, by alternating their signals and understanding the directions, the two robots achieve rendezvous. Another common issue in dealing with IR beacons is the ambient lighting conditions. Depending on the illumination of the room and the reflective surfaces, the IR beam can die down sooner. With the help of

adaptation algorithm, the receivers are normalized to a certain value. This helps the beacons to adapt to any given ambient lighting conditions.

As mentioned earlier, the two robots A and B are each assumed to have six transmitters and six receivers to cover the entire range of 360°. The IR beam originating from these transmitters reflects off various obstacles, walls and other surfaces. If the receiver of the same robot is turned on during this time, the reflections create an illusion of another robot being present in the direction of the obstacle or wall.

Algorithm *Adaptation*

Input: minimum distance 'ϵ' between the two robots after rendezvous in cm

Output: $T_v(A)$ and $T_v(B)$

Step 1: For each of the robots A and B, repeat the Steps 2-9

Step 2: Divide 0° to 360° into six directions

Step 3: For each of the six transmitting directions of the current robot, repeat the steps 4-9

Step 4: Turn on the transmitter and all the six receivers

Step 5: Turn off the transmitter with the first ping in any of the six receivers

Step 6: Measure the number of pings at all six receives and store them in $P_1, P_2, \cdots, P_6$

Step 7: Measure the time taken (from the first ping) and the direction for the last ping and store it in t_i where i is the current direction number of the transmitter

Step 8: Calculate ϕ as the average of $P_1, P_2, \cdots, P_6$

Step 9: Compute the threshold value for the current direction with the help of the following formula:

$$T_{vi} = \frac{\phi(1 + \frac{1}{\epsilon})}{2} \tag{7.1}$$

where ϕ is computed as given in Step 8 and

ϵ is the desired distance (in cm) between the two robots after rendezvous

Step 10: Compute $T_v(A)$ and $T_v(B)$ as the average of $T_{v1}(A), \cdots, T_{v6}(A)$ and $T_{v1}(B), \cdots, T_{v6}(B)$ respectively

Step 11: Return $T_v(A)$ and $T_v(B)$

Remark 14 *The adaptation algorithm removes the necessity of centralized control and any form of communication between the two robots. The robots now work in the intervals of* $T_v(A)$ *and* $T_v(B)$ *thus ensuring that there is no signal overlap.*

7.3.2 Algorithm for Determining Direction of Turn

After calibrating the ambience in a given room, we determine the direction of motion. The direction of travel for each robot is determined by the direction in which the IR beam from the other robot hits it first. We now present Algorithm_Action that determines the direction of the next step for both the robots.

Algorithm *Action*

Input: Threshold values $T_v(x)$ and $T_v(y)$; x is the transmitter robot and y is the receiver robot

Output: The direction of movement for x and y; D_x and D_y respectively. The time of travel for opposite IR beams from x and y; R_x and R_y respectively.

Step 1: Initialize all transmitters on x and all receivers on y.

Step 2: Turn off the transmitter on x after time $T_v(x)$.

Step 3: Wait for the first ping of IR pulse on any one of the receivers of robot y. Store the direction in D_y and the time after initialization in R_y. Turn off the receiver on robot y.

Step 4: Initialize all transmitters on y and all receivers on x

Step 5: Turn off the transmitter on y after time $T_v(y)$

Step 6: Wait for the first ping of IR pulse on any one of the receivers of robot x. Store the direction in D_x and the time after initialization in R_x. Turn off the receiver on robot x.

7.3.3 The Rendezvous Algorithm

The algorithm first detects if the minimum distance 'ϵ' of rendezvous given as an input is feasible or not. If it is not, it requests to restart the algorithm with a new 'ϵ'. If it is, it begins with the **Algorithm *Adaptation*** which is non-recursive. Once the threshold values are available, **Algorithm *Action*** is employed to move the robots towards the point of rendezvous.

Algorithm *Rendezvous*

Input: The diagonal length of the robots A and B, l_A and l_B respectively. The minimum distance between the two robots after rendezvous.

Output: Report if rendezvous is not possible due to improper 'ϵ' else achieve rendezvous.

Step 1: If $\epsilon < (l_A + l_B)$ then output that a collision is imminent. Else proceed to Step 2.

Step 2: Compute ($T_v(a)$,$T_v(B)$) using **Algorithm *Adaptation***

Step 3: Initialize two variables R_A and R_B to infinity and perform Steps 4 and 5 while individual threshold values of A and B are smaller than R_A and R_B. Else proceed to Step 6.

Step 4: Compute (D_A,D_B,R_A,R_B) using **Algorithm *Action***

Step 5: Move robot A in the direction of D_A by l_A and robot B in the direction

of D_B by l_B

Step 6: Initialize the Ultrasonic sensors on both robots A and B and detect the distance between each other in variables d_A and d_B. Note that d_A=d_B.

Step 7: While $d_A > \epsilon$, move robot A in direction d_A by l_A and B in direction of d_B by l_B

Step 8: Return robots achieved rendezvous

Some arguments regarding correctness of the algorithm are as follows. In the adaptation phase, a simple averaging technique is used to compute the levels beyond which the IR beam is detectable without any interference from ambient light sources. In the second part of the algorithm, rendezvous is accomplished in an iterative manner. It follows from Theorem 10 that an IR beam enters the region of a robot through its prior zones. Thus, turning the robot towards the direction in which the IR beam is incident and moving it by a fixed distance guarantees that the robot moves in the direction of decreasing zones. The comparison with ϵ ensures that the algorithm terminates in a finite number of steps.

The zones change due to the vertex or edge of an obstacle as given by Lemma 11. Therefore, it is possible that the robot goes and hits the obstacle either at its edge or its vertex. To avoid this, we use an iterative algorithm. An iterative algorithm would change the direction of the robot as soon as it changes zones thus making sure that it never hits an obstacle or a wall. This can be observed from Fig. 7.2. As the robot moves from Zone 2 to Zone 1, the robot gets its IR beam from a different direction altogether. For example, if the robot is now located in the triangle $\triangle\ P_2P_{11}Ob_1$ which is of Zone 1, the IR beam would hit the wall of P_aP_d and reach this zone. So, the algorithm would direct the robot to move towards the wall P_aP_d. However, as soon as the robot crosses the imaginary line P_2Ob_1, it enters into zone 0. The IR beam would now be incident directly from A. Thus the robot orients itself towards A and moves until Rendezvous.

The dynamics of the robots do not form a part of the algorithm. Thus, robots of heterogeneous nature can be used with this algorithm. Since Algorithm_Rendezvous

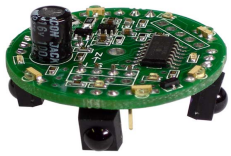


Figure 7.3: Pololu IR Transceiver used for the experiment

ensures that the minimum distance for rendezvous is greater than $l_1 + l_2$(the sum of diagonal lengths of the two robots in the traveling direction), collision of robots is also not an issue.

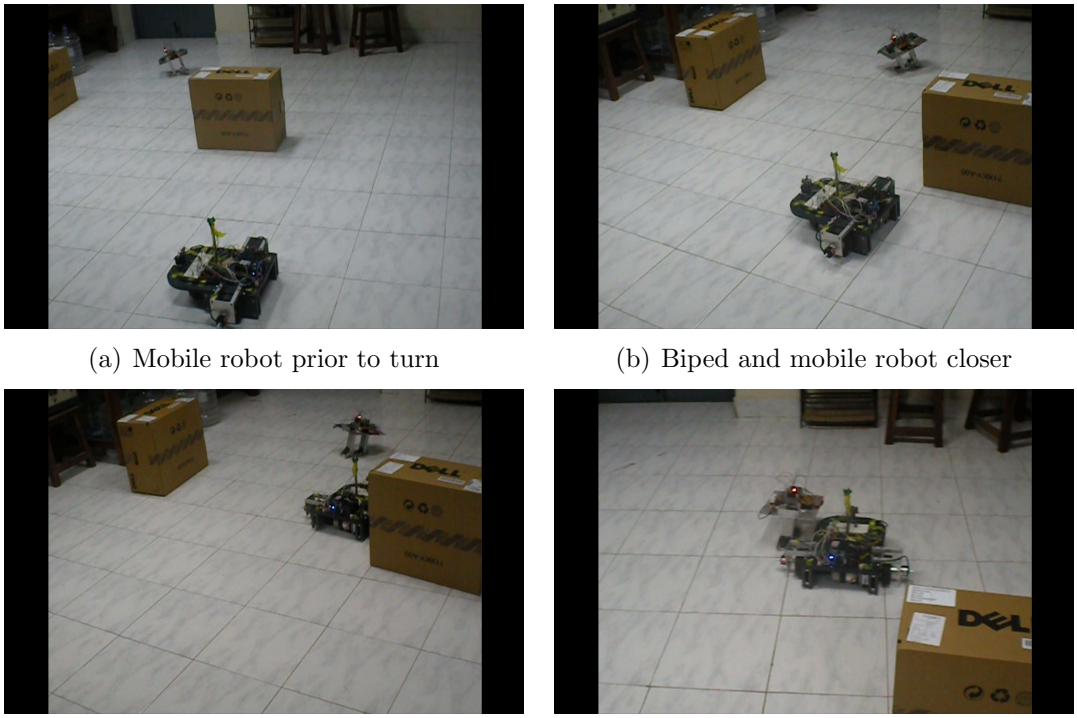
7.4 Experimental verification

In this section, we present the results of our experiments with a bipedal robot and a mobile robot. The mobile robot has two differential drive wheels at the back and two castor wheels at the front. The robot can handle a payload of 5 kg and is powered by 1000 rpm DC motors. A Pololu driver board is used to control the robot using an ATMEGA 32 micro controller. The bipedal robot has two feet and is powered by HSR-5990TG servo motors and has an ATMEGA 32 microcontroller for processing. We use Pololu IR transmitter-receiver pairs (transceivers) for the IR beacons. Each Pololu transceiver uses six transmitters and four receivers. Fig. 7.3 gives a picture of the hardware. Each transmitter covers 60° while each receiver covers 90°.

The details of the experiments are as follows. The receivers and transmitters on both the robots first go through the adaptation mode. Having calculated T_v, for both the robots, they individually start the transmission and reception of signals in pre-allocated time intervals.

The receivers of the Pololu IR transceiver are labelled as North, East, South and West. The robots rearrange the signals coming from various directions to one of these four directions and then go on with the algorithm.

The step length of the biped is just 5 cm while the mobile robot can cover 100 cm in a couple of seconds. Due to this huge difference in speeds, we designed the biped to transmit and receive at a much faster rate and update its direction as compared to the mobile robot. Thus the biped moves continuously until there is



(a) Mobile robot prior to turn

(b) Biped and mobile robot closer

(c) Mobile robot after negotiating an obstacle

(d) Rendezvous achieved

Figure 7.4: Different positions leading to rendezvous

a necessary rotation.

It can be noted that the IR beacons on both the robots are placed at the same level from the ground to ensure proper transmission and reception of signals.

Fig. 7.4 gives various locations of the robots before they could achieve rendezvous. The mobile robot identifies the direction of IR rays hitting it from its front. Thus it moves front by its fixed distance l_1 and reaches the position in Fig. 7.4(a). The rays continue to hit it from the same direction and thus the robot moves further. It then rotates right and moves closer to the biped. When the mobile robot is close to the cardboard box, the rays come from around the box. The robot now rotates again toward the direction of the incident signal which is to its left. This is shown in Fig. 7.4(b). It now moves forward by its prescribed amount to get in complete view of the biped. However, the biped now stands to its right and thus the robot turns right facing the biped. This is shown in Fig. 7.4(c). Fig. 7.4(d) shows the movement of the two robots to their final position.

7.5 Summary

In this chapter, we have considered the problem of rendezvous between a pair of robots with limited communication. In particular, we have assumed that the robots carry only IR beacons.We have discussed issues pertaining to IR signals and derived some important results. We have then presented an algorithm for rendezvous taking care of the ambience. We have also described experiments on rendezvous between a mobile robot and a bipedal robot fabricated indigenously.

CHAPTER 8

CONCLUSIONS

In this thesis, we investigate rendezvous of mobile agents at an optimal location amidst various constraints. The constraints are motivated by the type or nature of agents and the task they intend to carry out. Robots designed to exchange and replenish supplies as they traverse to various sites are concerned with their individual energies and a minimax distance constraint is applicable here. Similarly, robots designated to meet after fixed intervals of time would attempt to minimize the collective distance travelled thus incorporating a constraint on the total Euclidean distance travelled. Finally, robots operating in hazardous environments would want to achieve rendezvous with limited information from fellow agents and in minimum time.

We address each of the constraints and develop efficient geometric algorithms that compute an optimal location for rendezvous. Throughout the thesis, we deal with obstacles that either obstruct the paths of various agents or intelligently attempt to capture one of the agents to prevent rendezvous. The thesis also presents experiments on custom-fabricated robots. We show that the algorithms developed, while involving complex tasks such as computing intersections of hyperbolas or constructing graphs to compute shortest paths, can be implemented on robots equipped merely with microcontrollers and without any external memory.

In this chapter, we provide a summary of the main contributions and discuss possibilities for future work.

8.1 Contributions of the Thesis

The first contribution of the thesis is development of hardware-efficient algorithms for computation of an optimal location for rendezvous with respect to the minimax distance criterion. The chapter (Chapter 3) introduces the notion of last-turn

location and defines a weighted minimax distance constraint when the agents have travelled a finite distance prior to arrival at a given location. This finite distance is referred to as the weight on a location. We present an algorithm that computes the rendezvous location for k agents amidst n polygonal obstacles by using a result that limits the rendezvous location to the vicinity of the minimax location computed in the absence of obstacles. Additionally, we illustrate the hardware efficiency of the algorithms via experiments on mobile robots supported by only a microcontroller (and no external memory). A detailed description of the algorithm and an efficient way to adapt the algorithm to a microcontroller is presented in Vundurthy and Sridharan (2019).

Following the minimax distance constraint, we discuss computation of an optimal location with respect to the minsum distance constraint in Chapter 4. Contrary to the prior observation, the rendezvous location for minsum distance constraint cannot be restricted with respect to its counterpart computed in the absence of obstacles. We thus resolve to an incremental algorithm and show that the two passes, one in the increasing and decreasing order of their distances from minsum location are sufficient to compute the optimal rendezvous location. Once again, we illustrate the algorithm with the help of experiments on mobile robots.

Having discussed two variations of the distances constraints, Chapter 5 explores the effect of an intelligent obstacle (or an adversary) that deliberately tries to prevent rendezvous of a pair of agents. We equip the adversary with a vision sensor and allow communication only among the pair of agents. We discuss the notion of dominance curves and safe regions to compute an optimal meeting location if rendezvous is indeed feasible. Similar to prior results, we illustrate the algorithms via implementation on mobile robots. The processing support for the adversary is provided by a Raspberry Pi module while a Pi camera serves as the vision sensor. No additional support via a PC is necessary to perform the analysis of the image captured. We discuss various aspects of experiments including occlusion of vision sensor by an agent.

In addition to the two variants of distance constraints, we explore rendezvous with a constraint on time lost in commute in Chapter 6. We compute the optimal location, termed as the Time Optimal Rendezvous Point (TOPR), both in the

absence and presence of static obstacles. We show that the development of an efficient algorithm allows for a quick re computation of the rendezvous location when one or more of the agents are blocked by a moving obstacle. We support the efficiency of the algorithms with experiments on mobile robots and illustrate the role of moving obstacles via an AGV (Automated Guided Vehicle). This work has been reported in Vundurthy and Sridharan (2018). Further, we describe a sensor-specific rendezvous that involves two heterogeneous agents (mobile robot and a bipedal robot) equipped with a Polulu IR beacon for identification of their respective heading directions. A detailed description of the solution methodology and experimental setup is provided in Vundurthy *et al.* (2016).

8.2 Extensions and Future Work

The work discussed in this thesis is focussed on developing optimal locations in the plane for rendezvous. The minsum distance constraint deals with three and four agents for rendezvous since the solution for more than four agents case (even in the absence of obstacles) requires an iterative procedure. It is worth exploring the location for rendezvous, with a small tolerance if necessary, that computes the location for an arbitrary number of agents.

When handling an adversary, it is of interest to explore how multiple delivery agents are handled by a single adversary. Exploring the need of the adversary for a dominant strategy might shed some light on the evasive manoeuvres adopted by the rescue and delivery agents.

Further, the time optimal rendezvous point assumes identical agents while the agents start simultaneously from the start location. Additional work on computing this location considering agents travelling with non-identical speeds might be a potential enhancement to the current strategy.

Finally, the sensor specific rendezvous can be extended to a three dimensional setup since the Polulu IR beacon sensor puts no such restrictions.

PUBLICATIONS FROM THIS THESIS

[1] **B. Vundurthy** and K. Sridharan, "Multiagent Gathering With Collision Avoidance and a Minimax Distance Criterion−Efficient Algorithms and Hardware Realization," in IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 699-709, Feb. 2019. Available on IEEEXplore.

[2] **B. Vundurthy** and K. Sridharan, "Time Optimal Rendezvous for Multi-Agent Systems Amidst Obstacles - Theory and Experiments," IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, 2018, pp. 2645-2650. Available on IEEEXplore.

[3] **B. Vundurthy**, A. More, S. V. V. Raju and K. Sridharan, "Rendezvous of heterogeneous robots amidst obstacles with limited communication," 2016 Indian Control Conference (ICC), Hyderabad, 2016, pp. 347-353. Available on IEEEXplore.

REFERENCES

1. **Abelson, H.** (1978). Lower bounds on information transfer in distributed computations. *Proc. of the IEEE 19th Annual Symposium on Foundations of Computer Science*, 151–158.

2. **An, B.**, **Z. Shen**, **C. Miao**, and **D. Cheng** (2007). Algorithms for transitive dependence-based coalition formation. *IEEE Trans. Ind. Informat.*, **3**(3), 234–245.

3. **Ando, H.**, **Y. Oasa**, **I. Suzuki**, and **M. Yamashita** (1999). Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, **15**(5), 818–828.

4. **Bhatia, A.** and **E. Frazzoli**, Decentralized algorithm for minimum-time rendezvous of dubins vehicles. *In 2008 American Control Conference*. 2008.

5. **Bhattacharya, B.** (2011). On the Fermat-Weber point of a polygonal chain and its generalizations. *Fundamenta Informaticae*, **107**(4), 331–343.

6. **Bhattacharya, S.**, **V. Kumar**, and **M. Likhachev**, Distributed optimization with pairwise constraints and its application to multi-robot path planning. *In Proc. of the Robotics: Science and Systems Conference (RSS)*. 2010.

7. **Borenstein, J.** and **L. Feng** (1996). Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, **12**(6), 869–880.

8. **Boyell, R.** (1976). Defending a moving target against missile or torpedo attack. *IEEE Trans. Aerosp. Electron. Syst.*, **AES-12**(4), 522–526.

9. **Brown, T. L.**, **T. D. Aslam**, and **J. P. Schmiedeler**, Determination of minimum time rendezvous points for multiple robots via level set methods. *In ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 6. 2011.

10. **Bullo, F.**, **J. Cortes**, and **S. Martinez**, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, Princeton, NJ, USA, 2009. ISBN 0691141959, 9780691141954.

11. **Cao, Y.**, **W. Yu**, **W. Ren**, and **G. Chen** (2013). An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Trans. Ind. Informat.*, **9**(1), 427–438.

12. **Chen, G.**, **Z. Yang**, and **C. Low** (2006). Coordinating agents in shop floor environments from a dynamic systems perspective. *IEEE Trans. Ind. Informat.*, **2**(4), 269–280.

13. **Chen, J.** and **M. Barnes** (2014). Human-agent teaming for multirobot control: A review of human factors issues. *IEEE Trans. Human-Mach. Syst.*, **44**(1), 13–29.

14. **Chung, T.**, **G. Hollinger**, and **V. Isler** (2011). Search and pursuit-evasion in mobile robotics: A survey. *Autonomous Robots*, **31**(4), 299–316.

15. **Chunhe, H.** and **C. Zongji**, Minimum time rendezvous for multi-vehicle with non-identical velocity constraints. *In Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. 2014.

16. **Cockayne, E.** and **Z. Melzak** (1969). Euclidean constructibility in graph-minimization problems. *Mathematics Magazine (published by Mathematical Association of America)*, **42**(4), 206–208.

17. **Cormen, T. H.**, **C. E. Leiserson**, **R. L. Rivest**, and **C. Stein**, *Introduction to Algorithms, Third Edition*. The MIT Press, 2009, 3rd edition. ISBN 0262033844, 9780262033848.

18. **Courant, R.** and **H. Robbins**, *What is Mathematics ?*. Oxford University Press, 1941.

19. **Dijkstra, E.** (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

20. **Drezner, Z.** and **H. W. Hamacher**, *Facility Location: Applications and Theory*. Springer, 2004.

21. **Drezner, Z.** and **G. Wesolowsky** (1980). Single facility l_p-distance minimax location. *SIAM Journal on Algebraic and Discrete Methods*, **1**(3), 315–321.

22. **Edelsbrunner, H.**, **L. J. Guibas**, and **J. Stolfi** (1986). Optimal point location in a monotone subdivision. *SIAM J. Comput.*, **15**, 317–340.

23. **Elzinga, J.** and **D. Hearn** (1972). Geometric solutions for some minimax location problems. *Transportation Science*, **6**, 379–394.

24. **Fax, J. A.** and **R. M. Murray** (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, **49**(9), 1465–1476.

25. **Francis, R. L.** (1967). Some aspects of a minimax location problem. *Operations Research*, **15**(6).

26. **Fredman, M.** and **R. Tarjan**, Fibonacci heaps and their uses in improved network optimization algorithms. *In Proc. of Twenty Fifth Annual IEEE Symposium on Foundations of Computer Science*. 1984.

27. **Ganguli, A.**, **J. Cortés**, and **F. Bullo** (2009). Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, **25**, 340–352.

28. **Garcia, E.**, **D. W. Casbeer**, **Z. E. Fuchs**, and **M. Pachter** (2018). Cooperative missile guidance for active defense of air vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, **54**(2), 706–721. ISSN 0018-9251.

29. **Ghosh, S. K.** and **D. M. Mount**, An output sensitive algorithm for computing visibility graphs. *In 28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. 1987.

30. **Han, D.**, **G. Chesi**, and **Y. Hung** (2013). Robust consensus for a class of uncertain multi-agent dynamical systems. *IEEE Trans. Ind. Informat.*, **9**(1), 306–312.

31. **Hartley, R.**, **K. Aftab**, and **J. Trumpf**, L1 rotation averaging using the Weiszfeld algorithm. *In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011.

32. **Hartline, J.** and **A. Sharp**, An incremental model for combinatorial maximization problems. *In Proc. of Fifth International Workshop on Experimental Algorithms, Springer-Verlag*. 2006.

33. **Hershberger, J.** and **S. Suri** (1999). An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, **28**, 2215–22566.

34. **Isaacs, R.** (1964). The best deployment of a naval force in the vicinity of potential trouble spots. *Internal Memorandum (CNA), Center for Naval Analysis, Arlington, VA*, 31–64.

35. **Isaacs, R.**, *Differential Games*. Dover Publications, Inc., 1965. ISBN 0486406822.

36. **Jadbabaie, A.**, **J. Lin**, and **A. Morse** (2003). Coordinations of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, **48**(6), 988–1001.

37. **Jan, G. E.**, **C. Sun**, **W. Tsai**, and **T. Lin** (2014). An $O(n \log_2 n)$ shortest path algorithm based on Delaunay triangulation. *IEEE/ASME Transactions on Mechatronics*, **19**(2), 660–666.

38. **Kapoor, S.** and **S. N. Maheshwari**, Efficient algorithms for euclidean shortest path and visibility problems with polygonal obstacles. *In Proceedings of the Fourth Annual Symposium on Computational Geometry*, SCG '88. ACM, New York, NY, USA, 1988. ISBN 0-89791-270-5. URL `http://doi.acm.org/10.1145/73393.73411`.

39. **Kirkpatrick, D. G.** (1983). Optimal search in planar subdivisions. *SIAM J. Comput.*, **12**, 28–35.

40. **Krarup, J.** and **S.Vajda** (1997). On Torricelli's geometrical solution to a problem of Fermat. *IMA Journal of Management Mathematics)*, **8**(3), 215–224.

41. **Kunwar, F.**, **F. Wong**, **R. B. Mrad**, and **B. Benhabib**, Time-optimal rendezvous with moving objects in dynamic cluttered environments using a guidance based technique. *In 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005.

42. **Latombe, J.-C.**, *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991. ISBN 079239206X.

43. **Leitão, P.**, **V. Mařik**, and **P. Vrba** (2013). Past, present, and future of industrial agent applications. *IEEE Transactions on Industrial Informatics*, **9**(4), 2360–2372.

44. **Long, M.** and **C. Wu** (2006). Energy-efficient and intrusion resilient authentication for ubiquitous access to factory floor information. *IEEE Trans. Ind. Informat.*, **2**(1), 40–47.

45. **Lozano-Perez, T.** and **M. A. Wesley** (1979). An algorithm for planning collison-free paths among polyhedral obstacles. *Communications of the ACM*, **22**, 560–570.

46. **Lumelsky, V. J.** and **A. A. Stepanov** (1987). Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape. *Algorithmica*, **2**, 403–430.

47. **Luo, R.** and **C. Chang** (2012). Multisensor fusion and integration: A review on approaches and its applications in mechatronics. *IEEE Trans. Ind. Informat.*, **8**(1), 49–60.

48. **Megiddo, N.** (1983). The weighted Euclidean 1-center problem. *Mathematics of Operations Research*, **8**(4), 498–504.

49. **Melzak, Z.**, *Companion to Concrete Mathematics*. John Wiley & Sons, New York, 1974.

50. **Mitchell, J. S.** (1996). Shortest paths among obstacles in the plane. *International Journal of Computational Geometry & Applications*, **06**(03), 309–332.

51. **Murrieta-Cid, R.**, **T. Muppirala**, **A. Sarmiento**, **S. Bhattacharya**, and **S. Hutchinson** (2007). Surveillance strategies for a pursuer with finite sensor range. *Int. J. Rob. Res.*, **26**(3), 233–253.

52. **Noori, N.** and **V. Isler** (2014). Lion and man with visibility in monotone polygons. *Int. J. Rob. Res.*, **33**(1), 155–181.

53. **Notarstefano, G.** and **F. Bullo**, Distributed consensus on enclosing shapes and minimum time rendezvous. *In Proceedings of the 45th IEEE Conference on Decision and Control*. 2006.

54. **Olfati-Saber, R.**, **J. Fax**, and **R. Murray** (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, **95**(1), 215–233.

55. **Olfati-Saber, R.** and **R. Murray** (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, **49**(9), 1520–1533.

56. **O'Rourke, J.**, *Computational Geometry in C*. Cambridge University Press, 1993.

57. **Overmars, M. H.** and **E. Welzl**, New methods for computing visibility graphs. *In Proceedings of the Fourth Annual Symposium on Computational Geometry*, SCG '88. ACM, New York, NY, USA, 1988. ISBN 0-89791-270-5. URL `http://doi.acm.org/10.1145/73393.73410`.

58. **Oyler, D.**, **P. Kabamba**, and **A. Girard** (2016). Pursuit evasion games in the presence of obstacles. *Automatica*, **65**, 1–11. ISSN 0005-1098.

59. **Rademacher, H.** and **O. Toeplitz**, *The Enjoyment of Mathematics*. Princeton University Press, 1957.

60. **Ren, W.** and **R. Beard** (2005). Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, **50**, 655–661.

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

62. **Rohnert, H.** (1986). Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, **23**, 71–76.

63. **Rosin, P. L.** (1997). Techniques for assessing polygonal approximations of curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(6), 659–666. ISSN 0162-8828.

64. **Schwartz, J. T.** and **M. Sharir** (1983). On the piano movers' problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, **36**(3), 345–398.

65. **Scott, W.** and **N. Leonard**, Pursuit, herding and evasion: A three-agent model of caribou predation. *In 2013 American Control Conference*. 2013.

66. **Setter, T.** and **M. Egerstedt**, Minimum time power-aware rendezvous for multi-agent networks. *In 2014 IEEE Conference on Control Applications (CCA)*. 2014.

67. **Vicsek, T.**, **A. Czirok**, **E. Jacob**, **I. Choen**, and **O. Schochet** (1995). Novel type of phase transitions in a system of self-driven particles. *Phys. Rev. Lett.*, **75**, 1226–1229.

68. **Vundurthy, B.**, **A. More**, **S. V. V. Raju**, and **K. Sridharan**, Rendezvous of heterogeneous robots amidst obstacles with limited communication. *In 2016 Indian Control Conference (ICC)*. 2016.

69. **Vundurthy, B.** and **K. Sridharan**, Time optimal rendezvous for multi-agent systems amidst obstacles - theory and experiments. *In IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. 2018.

70. **Vundurthy, B.** and **K. Sridharan** (2019). Multiagent gathering with collision avoidance and a minimax distance criterionefficient algorithms and hardware realization. *IEEE Transactions on Industrial Informatics*, **15**(2), 699–709. ISSN 1551-3203.

71. **Weiss, L.** (2011). Autonomous robots in the fog of war. *IEEE Spectrum*, **48**(8), 30–57.

72. **Weiszfeld, E.** (1936). Sur le point pour lequel la somme des distances de n points donnes est minimum. *The Tohoku Mathematical Journal*, (43), 355–386.

73. **Wynters, E.** and **J. Mitchell**, Shortest paths for a two robot rendezvous. *In Proc. of Fifth Canadian Conference on Computational Geometry*. 1993.

74. **Yamaguchi, H.** (1999). A cooperative hunting behavior by mobile-robot troops. *The International Journal of Robotics Research*, **18**(9), 931–940.

75. **Yao, A. C.**, Some complexity questions related to distributed computing. *In Proc. of 11th ACM Symposium on Theory of Computing (STOC)*. 1979.

76. **Yu, J.**, **S. LaValle**, and **D. Liberzon** (2012). Rendezvous without coordinates. *IEEE Transactions on Automatic Control*, **57**, 421–434.

77. **Yu, W.**, **L. Zhou**, **X. Yu**, **J. Lu**, and **R. Lu** (2013). Consensus in multi-agent systems with second-order dynamics and sampled data. *IEEE Trans. Ind. Informat.*, **9**, 2137–2146.

78. **Yuan, Y.**, **H. Yuan**, **L. Guo**, **H. Yang**, and **S. Sun** (2016). Resilient control of networked control system under DoS attacks: a unified game approach. *IEEE Trans. Ind. Informat.*, **12**(5), 1786–1794.

79. **Zhan, J.** and **X. Li** (2013). Flocking of multi-agent systems via MPC based on position-only measurements. *IEEE Trans. Ind. Informat.*, **9**, 377–385.

80. **Zhou, X.**, **Y. Li**, **B. He**, and **T. Bai** (2014). GM-PHD-based multi-target visual tracking using entropy distribution and game theory. *IEEE Trans. Ind. Informat.*, **10**(2), 1064–1076.

81. **Zou, R.** and **S. Bhattacharya** (2016). Visibility-based finite-horizon target tracking game. *IEEE Robot. Autom. Lett.*, **1**(1), 399–406.