ON THE MUTUAL VISIBILITY OF FAT MOBILE ROBOTS

A thesis submitted
to Kent State University in partial
fulfillment of the requirements for the
degree of Master of Science

by

Rusul Jabbar Abbas Alsaedi

May 2016
Thesis written by
Rusul Jabbar Abbas Alsaedi
B.S., University of Al-Qadisiyah, 2011
M.S., Kent State University, 2016

Approved by
Gokarna P. Sharma, Advisor
Javed I. Khan, Chair, Department of Computer Science
James L. Blank, Dean, College of Arts and Sciences
ABSTRACT

Given a set of $n \geq 1$ autonomous, anonymous, history-oblivious, silent, and possibly disoriented mobile robots operating following Look-Compute-Move cycles in the Euclidean plane, we consider the fundamental problem of providing mutual visibility for them, i.e., the robots must reposition themselves to reach a configuration in finite time without collisions where they all see each other. This problem arises under obstructed visibility where a robot can not see another robot if there lies a third robot on the line segment connecting their positions. This problem is important since it provides a basis to solve many other problems under obstructed visibility, and it has applications in many scenarios including coverage, intruder detection, etc. The literature on this problem assumed that the robots are dimensionless points, i.e., they occupy no space. However, this assumption can be easily refuted. For example, in reality, robots are not dimensionless, but they have a physical extent.

Therefore, in this thesis, we initiate the study of the mutual visibility problem for the robots with extents. We address this problem in the recently proposed robots with lights model, where each robot is equipped with an externally visible persistent light that can assume colors from a fixed set of colors. This model corresponds to the classical oblivious robots model when the number of colors in the set is 1. In particular, we first develop a deterministic algorithm that provides mutual visibility for robots with extents of unit disc size avoiding collisions using only 4 colors in the color set. Note that this algorithm works for fat robots under the fully synchronous and semi-synchronous settings. We then present a faster algorithm that solves this problem in $O(n)$ rounds in the fully synchronous setting.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>ix</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>x</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Background and Motivation ........................................ 1
1.2 The Model and the Mutual Visibility Problem .............. 2
1.3 Performance Metrics ............................................ 5
1.4 Contributions .................................................. 5
1.5 Solution Approach ............................................... 6
1.6 Thesis Outline .................................................. 8

## 2 Literature Review

2.1 Literature on Solvability ....................................... 9
2.2 Literature on Runtime .......................................... 12

## 3 Solving the Mutual Visibility Problem

3.1 Outline ......................................................... 15
3.2 Model and Preliminaries ........................................ 15
3.3 The Mutual Visibility Algorithm ........................... 18
   3.3.1 Interior Depletion ........................................ 21
   3.3.2 Side Depletion ............................................ 31
   3.3.3 Special Cases ............................................. 34
3.4 Analysis of the Algorithm for the FSYNCH setting .... 35
3.5 Analysis for the SSYNCH Setting ............................ 41

## 4 A Faster Solution for the Mutual Visibility Problem

4.1 Outline ......................................................... 42
4.2 The Mutual Visibility Algorithm ........................... 42
   4.2.1 Interior Depletion ........................................ 43
   4.2.2 Side Depletion ............................................ 49
   4.2.3 Special Cases ............................................. 50
4.3 Analysis of the Algorithm .................................... 50
LIST OF FIGURES

1.1 An illustration of how to create the final configuration (convex hull) from an initial configuration. .................................................. 4

3.1 An illustration of safe zone for an edge $e = v_1v_2$ and also for a side robot $r$ in $e$. The safe area $S'(e)$ for any robot $r$ in $e$ is a subset of $S(e)$. .................................................. 18

3.2 An illustration of how the ID phase works .................................................. 19

3.3 An illustration of how the SD phase works .................................................. 20

3.4 An illustration of how internal robots select edges of $H_k$ to include in their $Q$’s. The robot $r_i$ includes $ef$ and $e\overline{v_1}$ in its $Q$. Robots $r_j$ has $ef$, and $r_l$ has $fv_2$ in their $Q$’s. However, the robots between $r_i, r_j$ have empty $Q$’s. .................................................. 25

3.5 An illustration of how an internal robot, $r_i$ chooses a point to move to on the edge $ef \in Q$ that is the closest to when it is the only closest robot to $ef$. Since there are side robots on $ef$, $r_i$ chooses either $m$ or $m'$ as the point on $ef$ to try to move to. If there were no side robots on $ef$, $r_i$ would have chosen the midpoint of $ef$ as the point to try to move to. .................................................. 26

3.6 An illustration of how an internal robot $r_i$ chooses a point to move to on the edge $ef \in Q$ that is the closest to when there are other robots that are equidistant to $ef$. Since there are side robots on $ef$, and $r_i$ sees equidistant robots in only one direction, $r_i$ chooses $m_i$ as the point on $ef$ to try to move to. Robots can do this without agreeing on the coordinate system of other robots. Since $r_j$ also sees equidistant robots in only one direction, it chooses point $m_j$ to try to move to. Note that, the robots in between $r_i$ and $r_j$ do not try to move to. .................................................. 27

3.7 An illustration of the configuration in which the interior robot that is the closest to some edge $ef$ collides with some other internal robot while moving toward a computed point (one of $m, m'$) on $ef$. An intermediate move we describe in Fig. 3.8 handles this case. .................................................. 29
3.8 An illustration of the intermediate move done by an interior robot, \( r_i \), for the situation of Fig. 3.7. Since robot \( r_i \) finds that \( \text{length}(r_im''') \geq 2 \), it moves distance 1 towards \( ef \) on line \( r_im'' \) and changes its color to Yellow, where \( m''' \) is the intersection point of line \( L \) perpendicular to \( ef \) passing through \( r_i \) and \( ef \). It can move to \( m \) since no position of other robots fall between dotted parallel lines w.r.t. \( r_i \) \( m \) going from distance 1/2 from \( r_i \). If \( \text{length}(r_im''') < 2 \), \( r_i \) waits until the length becomes at least 2.

3.9 A corner robot \( e \) moves to a direction away from the convex hull if the distance between any of its left or right neighbor in \( H_k \) is \(< 2 \). It draws a line which bisects the angle \( \angle aeb \) and moves a unit distance on that line away from the convex hull.

3.10 A corner robot \( e \) moves to a direction away from the convex hull if the distance from the closest internal robot to either of \( ea \) or \( eb \) is \(< 2 \). It draws a line which bisects the angle \( \angle aeb \) and moves a unit distance on that line away from the convex hull.

3.11 A corner robot \( e \) moves to a direction away from the convex hull if it realizes that the Yellow robot collides with some other robot on the edge \( ea \) or \( eb \) if it moves to its destination \( m \) in either \( ea \) or \( eb \). It draws a line which bisects the angle \( \angle aeb \) and moves a unit distance on that line away from the convex hull.

3.12 An illustration of how the SD phase works. The Blue robots with one neighbor Red on \( e = v_1v_2 \) move to a point \( x \) inside the safe zone \( S(e) \) computed using the technique described in Section 3.2 without changing their color.

3.13 Robots \( r_i \) and \( r_j \) that moved to their local \( S(e) \) in Fig. 3.12 change their lights to Red only after all robots on \( e \) moved to \( r_i \) \( r_j \). Robots \( r_i, r_j \) do not move in future rounds.

3.14 If there is only one robot in \( S(e) \), then \( r \) waits until \( r_i \) moves to \( S(e) \). Robot \( r \) sees only one robot in \( S(e) \) on one side of the line \( L \) perpendicular to \( e \) going through its position, and it also sees robots on the other side of \( L \). Moreover, it is not adjacent to \( v_2 \).

4.1 An illustration of how internal robots select edges of \( H_k \) to include in their \( Q \)'s. The robot \( r_i \) includes \( ef \) and \( ev_1 \) in its \( Q \). Robots, \( r_j \) has \( ef \), and \( r_i \) has \( fv_2 \) in their \( Q \)'s. However, the robots between \( r_i \) and \( r_j \) have empty \( Q \).

4.2 The corner robots, \( e \) and \( f \) move to a direction away from the convex hull if the distance between the closest internal robot and the edge that is the closest to this robot in \( H_k \) is \(< 3 \). It draws a line which bisects the angle \( \angle aeb \) and moves some unit distance on that line away from the convex hull.
4.3 An illustration of how an internal robot $r_i$ chooses a point to move to in the edge $ef \in Q$ that is the closest to when it is the only closest robot to $ef$. Since there are side robots on $ef$, $r_i$ chooses either $m$ or $m'$ as the point in $ef$ to try to move to. If there were no side robots on $ef$, $r_i$ would have chosen the midpoint of $ef$ as the point to try to move to.

4.4 An illustration of how an internal robot $r_i$ chooses a point to move to in the edge $ef \in Q$ that is the closest to when there are other robots that are equidistant to $ef$. Since there are side robots on $ef$, and $r_i$ sees equidistant robots in only one direction, $r_i$ chooses $m_i$ as the point in $ef$ to try to move to. Robots can do this without agreeing on the coordinate system of other robots. Since $r_j$ also sees equidistant robots in only one direction, it chooses point $m_j$ to try to move to. Note that the robots that are between $r_i$ and $r_j$ do not try to move to.

4.5 An illustration of the corner robots, $e$ and $f$ move if the distance between the corner robot and the side robot on the edge $ef$ is $< 5$, and this is done by choosing a point outside $eb$ as an example on line $L$ perpendicular to $eb$ going through $b$, and line $L'$ that is angle bisector of $\angle aeb$, then move to a point in $L'$ outside the hull such that the distance of the line perpendicular to $L$ going through the point is 5.

4.6 An internal robot $r_i$ with light Yellow moves to a point that is at distance one from $m$ inside the convex hull and changes its color to Purple.

4.7 An internal robot $r_i$ with light Purple moves to the point $m$ on the edge $ef$ to be a side robot and changes its color to Blue.

4.8 An illustration of how the SD phase works. The Blue robots with one neighbor $Red$ in $e = \bar{v}_1 \bar{v}_2$ move to a point $x$ inside the safe zone $S(e)$ computed using the technique described in Section 3.2 without changing their color.

4.9 Robots $r_i$ and $r_j$ that moved to their local $S(e)$ in Fig. 4.8 change their light to Red only after all robots on $e$ moved to $r_i \bar{r}_j$. Robots $r_i, r_j$ do not move in future rounds.
DEDICATION

I would like to dedicate this thesis to
my family (daddy, mammy, my sisters, and my brother),
my country (IRAQ),
and everyone who wishes me the best
ACKNOWLEDGEMENTS

First of all, I would like to thank my god, Allah who has given me the graces in my whole life. Also, I would like to thank all my family members for their continuous support and encouragement specifically daddy and mammy since daddy is a professor, and he has encouraged and helped me to study my master’s degree in the USA, and he applied me for HCED-IRAQ scholarships to make me to have this chance for study, and mammy has cared about my study in my whole life.

My grateful thanks to my supervisor Dr. Gokarna P. Sharma for his help and guidance during my thesis work.

I would like to express my thanks to my thesis committee members professor Dr. Hassan M. Peyravi and professor Dr. Feodor F. Dragan for their acceptance to be the committee members.

Also, my thanks are extended to Professor Dr. Zuhair Humadi and Professor Dr. Abdulhakeem Alrawi for their help and support to all of HCED-IRAQ students. As well as, I would like to thank all of HCED-IRAQ employees and members for their continuous help.

Furthermore, I would like to thank my spirituous friend Marcy Curtiss for her continuous help for every student in Computer Science department.

Finally, I would like to thank all my friends who have helped me in my whole life.
CHAPTER 1

Introduction

1.1 Background and Motivation

In an abstract form, robots are machines with vision and/or locomotion capabilities. The vision capabilities are generally provided by camera. These robots can be classified into two types based on mobility: static and mobile robots. Mobile robots have locomotion capabilities in addition to vision capabilities, but static robots do not have locomotion capabilities. Therefore, it is better to choose mobile robots to work on since these robots can move from place to place, and they have a perfect success in the world of industrial manufacturing [38]. Also, mobile robots could be able to travel throughout the manufacturing plant, and they apply their talents flexibly [38]. Mobile robots can again be categorized into two types: robots with external control and robots without external control. The robots without external control are called autonomous mobile robots. It is better to choose autonomous mobile robots to work on since they can work in an unpredictable, unknown, and non-predefined environment [25]. Autonomous mobile robots can also have some distinct attributes; they may be memory-less, i.e., they may not have any memory about the past, and they may be silent. They may not communicate with other robots. They do not have any additional markers or identifiers, and they do not know any information about each other. They work independently observing only the environment using their vision capability and do their actions.

Industrial and technical applications of autonomous mobile robots have become very important
Autonomous mobile robots have already been widely used for surveillance, cleaning, inspection, and transportation tasks [35]. Furthermore, the autonomy has helped this kind of robots to be desirable in some fields such as household maintenance, space exploration, delivering goods and services, and waste water treatment [1].

Moreover, autonomous mobile robots play a vital role in automated inspection [35]. Manual inspection is a very costly process. Also, in [25], autonomous mobile robots are used for looking landmines.

Because of the importance of autonomous mobile robots in many applications and the flexibility of the system when they are used, there is a growing interest in the distributed computing, sensor network, computational geometry, and robotics community in solving many different problems using autonomous mobile robots. In this thesis, we try to solve a fundamental problem of mutual visibility. This problem is important since any solution for this problem helps to solve other fundamental problems such as gathering for these robots.

1.2 The Model and the Mutual Visibility Problem

Consider a set of \( n \geq 1 \) mobile robots in the Euclidean plane \( \mathbb{R}^2 \) which are autonomous (without external control), anonymous (do not have unique identifiers), indistinguishable (do not have external markers), history-oblivious (no memory of activities done in the past), and silent (without any direct means of communication). Due to being history-oblivious and silent, the only thing that these robots can do is to observe the environment and perform actions based on the information perceived from the environment. To observe the environment, robots use their local coordinate system and sensor capabilities (i.e., vision). The robots execute the same algorithm.

The robots operate following Look-Compute-Move (LCM) cycles and work towards achieving a common goal, i.e., when a robot becomes active, it uses its vision to get a snapshot of its surroundings (Look), computes a destination point based on the snapshot (Compute), and finally moves towards the computed destination (Move). Being both history-oblivious and silent
is the very limiting characteristic for these robots which restricts them to rely only on the environment to decide their next step. However, this is desirable since it provides fault-tolerance and self-stabilization, and enables their deployment in extremely harsh environments, where any sort of communication is impossible or can be impeded by interference [20]. Therefore, this robot model has many applications in mobile sensor networks, robotics, computational geometry, and distributed computing. The applications include coverage, exploration, intruder detection, data delivery, symmetry breaking, etc. This robot model is called \textit{classical oblivious robots model} in the literature \cite{7, 9, 10, 17, 18, 23, 24, 26, 29, 31, 42, 44} and has a long history of research.

There has been an extensive research under the assumption that visibility of the robots is \textit{unobstructed} - three collinear robots are mutually visible to each other \cite{3, 10, 14, 17, 18, 21, 24, 26, 29, 42, 44}. However, this assumption can be easily refuted. For example, unobstructed visibility does not address reality because robots may not be dimensionless points, and they may block the view of the other collinear robots. However, very little is known about computing with obstructed visibility, except for one-dimensional space \cite{10}. Therefore, the computability of these robots under obstructed visibility has been the subject of recent intensive research investigations \cite{2, 5, 6, 10–12, 19, 20, 32–34, 39–41, 43}. Under obstructed visibility, a robot $r_i$ can see another robot $r_j$ if and only if there is no other robot on the line segment connecting their positions. The recent intensive research under obstructed visibility focuses on the \textit{Mutual Visibility} problem: Starting from arbitrary distinct positions in the plane, determine a schedule to reposition the robots without collisions such that they reach a configuration where they all see each other and terminate their computation \cite{32–34, 39, 40, 43}. This focus is important because the \textit{Mutual Visibility} problem and the algorithms designed to solve it are fundamental in nature and provide a basis to solve several other problems under obstructed visibility. For example, in the fully synchronous setting, solving the \textit{Gathering} problem requires only one round beyond \textit{Mutual Visibility} \cite{43}. Similarly, solving the \textit{Circle Formation} problem requires only a few extra rounds \cite{32, 33, 39}. 

3
Figure 1.1: An illustration of how to create the final configuration (convex hull) from an initial configuration.

**Mutual Visibility** can be immediately achieved without any computation with unobstructed visibility, however, the difficulty of solving it increases significantly with obstructed visibility. This is because, under obstructed visibility, unless a robot has a priori knowledge of $n$, the robot might be unable to decide if it sees all $n$ robots, i.e., termination might not be possible [20, 32, 33]. Recently, a variation of the classical model, called the *robots with lights model*, has become the focus of significant interest [32–34, 36, 39, 40, 43]. In this model, robots are assumed to be equipped with an externally visible persistent light which can assume different colors from a fixed set of colors. This model corresponds to the classical oblivious robots model when the number of colors in the set is 1 [20, 21, 32, 40]. Moreover, the robots are history-oblivious except for the direct communication capability provided by lights. The robots with lights model provides advantages in many situations, such as hostile environments, where (limited) direct communication is assumed, and $n$ does not need to be known to solve *Mutual Visibility* even under obstructed visibility [32, 39]. Therefore, the objective in this model is to solve problems minimizing the size of the color set.

A series of work in the literature [32, 34, 39, 40, 43] solved the *Mutual Visibility* problem in the robots with lights model assuming that robots are dimensionless points, i.e., these works
provided a technique to overcome obstructed visibility but fell short in taking into account the extent of robots. Therefore, the techniques developed for dimensionless robots can not be applied directly to solve Mutual Visibility for robots with extents because, due to their extents, only a few robots can be accommodated in a given space without overlapping and colliding. Moreover, in a relatively different setting, the robots with extents were considered in the so-called fat robots model \cite{2,12,21} under the setting of the classical oblivious robots model. However, the focus there was on only solving the Gathering problem.

1.3 Performance Metrics

We consider the following two performance metrics while solving the mutual visibility problem that we study in this thesis:

- Solvability: this performance metric focuses on whether the mutual visibility problem can be solved using the system of robots described above.

- Runtime: this performance metric focuses on the number of the rounds needed to successfully terminate the algorithm solving the problem for all the robots in the system. The aim is to minimize the runtime of the algorithm.

1.4 Contributions

We consider a system of robots with extents (that occupy certain space in the plane) and initiate the study of providing mutual visibility for them. We address this problem in the recently proposed robots with lights model. In particular, we first present and analyze a 4-color algorithm that provides mutual visibility for robots with extents of unit disc size in finite time avoiding collisions. The algorithm works for both fully synchronous and semi-synchronous settings. To the best of our knowledge, this is the first algorithm for providing mutual visibility for the robots with extents.
We then provide a modified algorithm that solves the mutual visibility problem in $O(n)$ rounds in the fully synchronous setting.

The most closely related work to ours is due to Agathangelou et al. [2] in which they solved Mutually Visibility for fat robots before solving the Gathering problem. However, there are four main fundamental differences: (a) their model is the classical oblivious model (and $n$, the total number of robots, is assumed to be known by each robot) whereas ours is the robots with lights model (and robots do not know $n$); (b) collisions are not avoided in their solution whereas our solution is collision-free; (c) they need an assumption of chirality – robots agree on the orientation of the axes of their local coordinate system – whereas we provide Mutually Visibility without that assumption. Agathangelou et al. [2] asked whether the assumption of chirality can be removed from their solution. Our algorithm answers this in the affirmative but in the lights model; and (d) they only showed that their algorithms terminates in finite time, whereas we give an algorithm that terminates in $O(n)$ rounds in the fully synchronous setting.

Our first algorithm works for fully synchronous and semi-synchronous settings of robots with extents under rigid movements. In the fully synchronous (FSYNCH) setting, all robots are activated in each round, and they perform their Look-Compute-Move activities simultaneously in synchronized rounds. In the semi-synchronous (SSYNCH) setting, the robots operate in rounds, but all robots may not be activated in a round, and the robots that are activated perform their cycles in a perfectly synchronous setting. The moves of the robots are RIGID if an adversary does not have the power to stop a moving robot before reaching its computed destination [21]. Otherwise, the moves are NON-RIGID. The only assumption in NON-RIGID move is that robots move at least a minimum distance $S > 0$. Our second algorithm works for the fully synchronous setting.

### 1.5 Solution Approach

The literature has established that positioning the robots in the vertices of a convex hull provides a solution for Mutually Visibility [20, 32, 33, 39, 40, 43]. Therefore, our algorithm asks
the robots to position themselves in the vertices of the convex hull. To accomplish this, our algorithms first differentiate the robots on the boundary of the convex hull from the robots that are in the interior of the hull, and ask the internal robots to move to the hull boundary (the *interior depletion* phase). The robots on the boundary and the interior robots are differentiated through the colors of their lights (despite being silent other than the implicit communication capability provided by lights, the boundary robots can assign a different color to their lights than the lights of the internal robots). Due to the extents of robots, the completion of the interior depletion phase without collisions is challenging, and our main contribution lies on a technique to successfully complete this phase. Since, space in the convex hull boundary might be a problem to accommodate interior robots while they move to the boundary, we employ a technique that allows corner robots to detect that space problem and move to a direction away from the hull to create required space. Therefore, the interior depletion phase combines the movement of corner and interior robots until all interior robots be accommodated on the convex hull boundary. The corner robots also move to a direction away from the hull if they detect that the interior robots that try to move to the boundary have the possibility of collision with some other robots that are positioned in their paths.

After all internal robots reached the boundary, our technique differentiates the robots that are in the vertices (corners) of that hull with those on its edges (this can be done by robots since edge robots see one robot each in its clockwise and counterclockwise direction making an angle of exactly $180^\circ$ on it). The robots in the vertices do not move, and the algorithm asks the robots on the edges to move in the area outside of the hull without making the corner robots internal (the *side depletion* phase). Non-trivial technique is used which performs the moves of side robots until they all eventually become the vertices of the hull solving *Mutual Visibility*. The technique used in our first algorithm is relatively simple compared to the technique used in the second algorithm; however, it only guarantees that the algorithm terminates in finite time. The technique used in the second algorithm is quite involved, but it guarantees that the algorithm terminates in $O(n)$ rounds.
1.6 Thesis Outline

In this thesis, we proceed as follows. We present the literature review in Chapter 2 and present our first algorithm that solves the MUTUAL VISIBILITY problem for fat robots in finite time in Chapter 3. We then present a faster solution to solve the MUTUAL VISIBILITY problem for fat robots in Chapter 4. Finally, we conclude in Chapter 5 with a brief discussion of possible future work.
CHAPTER 2

Literature Review

2.1 Literature on Solvability

Peleg, in [36] was the first to propose robots with lights model. After that Das et al. [13] studied the power of these robots compared to classical oblivious robots with no lights.

Recently, the interest is on solving some specific problems in this model. Di Luna et al. [33] were the first to study the MUTUAL VISIBILITY problem in this model. They presented an algorithm that solves it with 6 colors in the SSYNCH setting and with 10 colors in the ASYNCH setting under both RIGID and NON-RIGID moves. They solved this problem when the visibility is obstructed (i.e., three robots that are on the same line segment are not assumed to be visible to each other) without collisions, without the assumption of chirality, and without any knowledge of the number of robots, n, in the system.

Vaidyanathan et al. [43] presented an algorithm in the FSYNCH setting that optimizes time complexity $O(\log n)$ rounds compared to $O(n)$ rounds needed for the algorithm of Di Luna et al. [33]). Although the runtime complexity is significantly improved, their solution uses chirality assumption and allows robots’ paths to be cross. In addition to not avoiding collisions, they used 12 colors in their algorithm.

Later, Di Luna et al. [32] presented two algorithms, one for the SSYNCH setting under RIGID moves with 2 colors (shrink algorithm), and the other for the SSYNCH setting under NON-RIGID
and the ASYNCH setting under both RIGID and NON-RIGID moves with 3 colors (contain algorithm); for the ASYNCH setting under NON-RIGID moves, this 3-color solution was obtained under the assumption of one-axis agreement of robots: all robots have the same view of the North and South. They also studied two other problems for oblivious robots: collision-less convergence to a point (also called near-gathering) and circle formation in the robots with lights model under obstructed visibility.

Sharma et al. [39] considered the mutual visibility problem for a set of \( n \) autonomous point robots and studied the runtime bounds of contain and shrink algorithms due to [32] in the FSYNCH setting. They showed that contain has the lower bound of \( \Omega(n) \) rounds, and shrink has the lower bound of \( \Omega(n^2) \) rounds. They also showed that contain has the runtime upper bound of \( O(n) \) rounds. They then presented a new algorithm called Modified-Shrink for the fully synchronous setting that has runtime \( O(n \log n) \) rounds and uses 3 colors. Also, they proved that the significance of this result of that runtime is significantly reduced using one additional color; note that shrink uses 2 colors. Modified-Shrink has the lower bound of \( \Omega(n) \) rounds.

Recently, Sharma et al. [40] presented an algorithm that improves the number of colors required in the solution of Di Luna et al. [32] from 3 to 2. Their solution is optimal since at least 2 colors are needed for MUTUAL VISIBILITY even in the robots with lights model. Also, they developed a near-optimal color solution to the Circle Formation problem.

However, all these above described work considered dimensionless (points) robots. J. Czyzowicz et al. [12] were the first to study the gathering problem in the plane for fat robots. These robots do not communicate, and they do not have any memory about the past. In addition, these robots are asynchronous. However, they solved the gathering problem for at most four robots.

K. Bolla et al. [5] studied the gathering problem for fat robots in the FSYNCH setting. They presented two algorithms for the gathering problem for fat robots. They showed that these two algorithms are able to solve the gathering problem for fat robots without requiring any communication. In addition to that, they showed that the performance of these two algorithms is better than the previous gathering algorithms for fat robots.
MUTUAL VISIBILITY is considered also in the work of Agathangelou et al. \cite{2} in the so-called fat robots model. They presented a distributed algorithm to solve the gathering problem for any number of fat robots in the ASYNCH setting (where robots are not equipped with lights) in the model first proposed by Czyzowicz et al. \cite{12}. Agathangelou et al. \cite{2} considered the problem with a state-machine representation. The major goal of this algorithm is to form a convex hull that holds all the robots on its boundary, and every robot sees all other robots. Also, they aimed in their paper to create a connected configuration. However, their solution does not avoid collisions and needs an assumption of chirality and knowledge of $n$ to each robot.

Dutta et al. \cite{19} presented a distributed algorithm that solves the circle formation with multiple autonomous, oblivious mobile fat robots in the ASYNCH setting. They solved this problem with limited visibility, that is, they assumed that every robot can see up to a specific distance around it. They also assumed that all robots agree on a common origin and common axis.

Chaudhuri and Mukhopadhyaya \cite{6} presented a distributed algorithm to solve the gathering problem for multiple autonomous, oblivious, fat mobile robots by applying asynchronous setting (the robots are not equipped with lights). They assumed the robots have full visibility, and they are transparent. Also, they assumed that the robots are initially stationary, and they are visible in their motions. The robots have no memory about the past. In addition to above, the robots work without chirality assumption or common coordination system assumption. In their paper, they presented a leader election algorithm that works by ordering the robots, and the robot that ordered first is elected to be the leader.

Flocchini et al. \cite{21} focused on the distributed computing algorithms by autonomous oblivious mobile robots and studied the effect of different assumptions on the strength of the robots computability.

Cord-Landwehr et al. \cite{11} presented an algorithm that solves the gathering problem for fat robots (robots with extents). In contrast to traditional gathering algorithms, they try to gather the robots as close as they can (i.e. the robots create a sphere with small radius around a known point). Therefore, the algorithm solves this problem by moving the robots towards the center of the sphere.
without blocking each other in the FSYNCH setting. Also, they proved the runtime of $O(nR)$ for the discrete setting, when $R$ is the distance of the farthest robot from the gathering point.

Luna et al. [34] proposed an algorithm that solves the mutual visibility problem for oblivious robots with obstructed visibility (i.e., three collinear robots are not assumed to be mutually visible to each other) in the semi-synchronous setting. They assumed that the robots must reach the configuration in finite time and see each other solving the mutual visibility problem without collisions. However, their algorithm is completely different from our algorithm; they assumed that the number of robots ($n$) is known for each robot, but ours has no such assumption.

2.2 Literature on Runtime

Vaidyanathan et al. [43] showed that the algorithm of Di Luna et al. [33] for mutual visibility terminates in at most $O(n)$ rounds under the fully synchronous setting, and $\Omega(n)$ rounds are sometimes needed. Vaidyanathan et al. [43] also presented an algorithm for the MUTUAL VISIBILITY problem and showed that their algorithm has a running time of $O(\log n)$ rounds in the fully synchronous setting. The robots are points, and the algorithm has the assumption of chirality, and collisions are not avoided.

Recently, Sharma et al. [39] proved that the mutual visibility algorithm of Di Luna et al. [34] has the runtime of $\Omega(n^2)$ rounds for some initial configuration of $n \geq 3$ robots in the FSYNCH setting. Then, they presented an algorithm that has the running time of $O(n \log n)$ rounds for any initial configuration of $n \geq 3$ robots in the FSYNCH setting in the robots with lights model with 3 colors. Also, they proved that their algorithm has the lower bound of $\Omega(n)$ rounds.

Similarly, much work on the classical model [4, 8–10, 28, 37] showed that GATHERING is achieved in finite time without a full runtime analysis, except in a few cases [8, 9]. In [4], they worked on the formation and agreement problems for autonomous robots with limited visibility to within distance $e$ for $e > 0$ in a synchronous setting. They presented an algorithm to solve the formation problem for a single point, and this algorithm is oblivious. In [8], they proved the
Correctness of the center-of-gravity algorithm for two robots in the fully asynchronous model. In [9], the authors worked on the convergence problem for autonomous mobile robots. The original algorithm for this problem needs the robots to move towards the center of gravity. In this paper, they completed the proof of the gravitational algorithm for any number of robots in the fully asynchronous model and analyzed its convergence in spite of crash faults. In [10], they studied local algorithms for autonomous robots focusing on the spreading problem. They presented an algorithm for the one-dimensional spreading problem, and they proved its convergence in the synchronous and semi-synchronous settings. Also, they proposed an algorithm that provides an equal spaced configuration in the synchronous setting in finite time.

Degener et al. [16] were the first to systematically analyze the running time of a local synchronous GATHERING algorithm. They proved that their algorithm gathers point robots in a (not predefined) point in \( O(n^3 \log n) \) expected time. They analyzed the running time of the gathering problem for any number of autonomous point robots with limited visibility.

Subsequently, Degener et al. [15] studied the runtime of the seminal algorithm due to Ando et al. [4] for GATHERING with limited visibility in synchronous discrete time and proved that robots gather in \( O(n^2) \) time, which is asymptotically optimal for some initial configurations of robots.

Recently, Kempkes et al. [30] considered the point robot GATHERING problem in continuous space and time and proved that limited visibility robots can be gathered at a point in \( O(n) \) rounds; this bound is also asymptotically optimal for some initial configurations in this model. Moreover, they gave other runtime bound of their algorithm of \( O(OPT \log OPT) \) rounds meaning that their local algorithm is only \( O(\log OPT) \) factor away from the cost \( OPT \) of the optimal global algorithm. They also proved that their algorithm runs to within \( O(\log f(n)) \) rounds of the optimal, where \( f(n) \) is the optimal number of rounds needed.

Furthermore, Izumi et al. [29] considered the robot scattering problem (just the opposite of the gathering problem) in the semi-synchronous setting and proved that the robots scatter to pairwise different positions in the Euclidean place in \( O(\min\{n, D^2 + \log n\}) \) rounds in expectation, where \( D \) is the diameter of the initial configuration.
Our algorithms provide MUTUAL VISIBILITY solution for robots with extents (fat robots) avoiding collisions and without the assumption of known n and chirality. The first algorithm uses 4 colors, but it does not guarantee runtime except it is finite. The second algorithm uses 5 colors, but it guarantees the termination in $O(n)$ rounds in the FSYNCH setting.
CHAPTER 3

Solving the Mutual Visibility Problem

3.1 Outline

In this chapter, we present the algorithm that solves the Mutual Visibility Problem for fat robots without collisions and without the assumption of chirality for two settings FSYNCH and SSYNCH. We solve this problem by using the light model, so the number of robots is not assumed to be known for each robot, but this model needs a color set with 4 colors $C = \{\text{Off, Yellow, Blue, Red}\}$, and we solve this algorithm in finite time. Therefore, in this chapter, we proceed as follows. We present the model and preliminaries, then we present our algorithm that solves the Mutual Visibility Problem with all of its cases and phases: interior depletion phase, side depletion phase, and special cases. After that, we present the analysis of the algorithm for two settings: FSYNCH and SSYNCH.

3.2 Model and Preliminaries

Consider a set of $n \geq 1$ anonymous robots $\mathcal{R} := \{r_1, r_2, \ldots, r_n\}$ operating in the Euclidean plane $\mathbb{R}^2$; $n$ is not assumed to be known to the robots. We assume that each robot $r_i \in \mathcal{R}$ is a non-transparent disc with diameter 1, i.e., $\text{diam}(r_i) = 1$. In other words, the disc is a bounding circle of radius 1/2 and $c_i$ is its center, which is also called the position of $r_i$. We denote by $\text{dist}(r_i, r_j)$ the distance between two robots, i.e., the distance from $c_i$ to $c_j$. To avoid collisions
among robots, we assume that \( \text{dist}(r_i, r_j) \geq 1 \) between any two robots \( r_i \) and \( r_j \) at all times. Each robot \( r_i \) has its own coordinate system centered in itself and it knows its position with respect to its coordinate system. Robots may not agree on the orientation of their coordinate systems, i.e., there is no common notion of clockwise direction. Since all the robots are of unit size, they agree implicitly on the unit of measure of other robots. The robots have a camera to take the snapshot of the plane and the visibility of the camera is unlimited provided that there is no obstacle (i.e., another robot) \([2]\).

Following the fat robot model \([2, 12]\), we assume that a robot \( r_i \) can see another robot \( r_j \) if there is at least one point on the bounding circle of \( r_j \) that is visible by \( r_i \). we say that a point \( p \) in the plane \( \mathbb{R}^2 \) is visible by a robot \( r_i \) if there is a point \( p_i \) in the bounding circle of \( r_i \) such that the straight line segment \( p_i p \) does not contain any point of any other robot. Therefore, robot \( r_i \) has mutual visibility if \( r_i \) can see all other robots in \( \mathcal{R} \). Two robots \( r_i \) and \( r_j \) are said to collide at time \( t \) if the bounding circles of \( r_i \) and \( r_j \) share at least a common point at \( t \). For simplicity, we use \( r_i \) to denote both the robot \( r_i \) and its position \( c_i \).

Each robot \( r_i \) is equipped with an externally visible persistent light that can assume any different color from a fixed constant set of colors \( C \). The colors in \( C \) are the same for all robots in \( \mathcal{R} \). We use variable \( r_i\.light \) to denote the light of robot \( r_i \). We assume that the color of the light of robot \( r \) at time \( t \) can be seen by all robots that are visible to \( r \) at time \( t \). Each robot executes the same algorithm locally every time it is activated.

A configuration \( C \) is a set of \( n \) tuples in \( C \times \mathbb{R}^2 \) which defines the position and color of a robot. Let \( \mathcal{C}_t \) denote the configuration at time \( t \). Let \( \mathcal{C}_t(r_i) \) denote the configuration \( \mathcal{C}_t \) for robot \( r_i \). A configuration \( C \) is obstruction-free if \( \forall r_i \in \mathcal{R} \), we have that \( |\mathcal{C}(r_i)| = n \) (all robots can see each other). Let \( \mathcal{H}_t \) denote the convex hull formed by \( \mathcal{C}_t \) which can be easily computed using Graham’s convex hull algorithm \([27]\). Let \( \partial \mathcal{H}_t = \mathcal{V}_t \cup \mathcal{S}_t \) denote the robots in the boundary of \( \mathcal{H}_t \), where \( \mathcal{V}_t \subseteq \mathcal{R} \) are the set of robots lying at the corners of \( \mathcal{H}_t \) and \( \mathcal{S}_t \subseteq \mathcal{R} \) are the set of robots lying at the sides (or edges) of \( \mathcal{H}_t \). The robots in the set \( \mathcal{V}_t \) are called corner robots and in the set \( \mathcal{S}_t \) are called side robots. The robots in the set \( \mathcal{I}_t = \mathcal{H}_t \setminus \partial \mathcal{H}_t \) are called internal robots. Given a robot
At any time $t$, a robot $r_i \in \mathcal{R}$ is either active or inactive. When active, $r_i$ performs a sequence of Look-Compute-Move (LCM) operations:

- **Look** – a robot takes the snapshot of the positions of the robots visible to it in its own coordinate system;
- **Compute** – executes its algorithm using the snapshot which returns a destination point $x \in \mathbb{R}^2$ and a color $c \in \mathcal{C}$; and
- **Move** – moves towards the computed destination $x \in \mathbb{R}^2$ (if $x$ is different than its current position) and sets its own light to color $c$.

We consider two schedulers for the activation of the robots in $\mathcal{R}$: FSYNCH and SSYNCH. In FSYNCH, the time is discrete and at each time instant $t$ all robots in $\mathcal{R}$ are activated and perform their LCM operations instantaneously, ending at time $t + 1$. SSYNCH is similar to FSYNCH except that a subset of robots (from empty set to all of $\mathcal{R}$) are activated at each time. Therefore, we use round $t$ in FSYNCH and SSYNCH instead of time $t$.

We assume that the execution starts at time 0. Therefore, at time $t = 0$, the robots start in an arbitrary configuration $\mathcal{C}_0$ with $\text{dist}(r_i, r_j) \geq 1$ for any two robots $r_i, r_j \in \mathbb{R}^2$ and the color of the light of each robot is set to $\text{Off}$. The **Mutual Visibility** problem is defined as follows: Given any $\mathcal{C}_0$, reach in finite time an obstruction-free configuration without collisions. An algorithm is said to solve **Mutual Visibility** if it always achieves an obstruction-free configuration from any arbitrary initial configuration. We measure the quality of the algorithm by counting the number of distinct colors in the set $\mathcal{C}$ needed to solve **Mutual Visibility**.

We need the following definition in the **Mutual Visibility** algorithm we present. Let $e = v_1v_2$ be a line segment connecting two corner robots $v_1$ and $v_2$ of $\mathbb{H}_t$. A safe zone is the portion of plane outside $\mathbb{H}_t$ such that the corner robots $v_1$ and $v_2$ of $\mathbb{H}_t$ remain as the corner robots despite
Figure 3.1: An illustration of safe zone for an edge $e = \overline{v_1v_2}$ and also for a side robot $r$ in $e$. The safe area $S'(e)$ for any robot $r$ in $e$ is a subset of $S(e)$.

the movements of the side robots in that area. Following Di Luna et al. [33], the safe zone of $e$, denoted as $S(e)$, consists of the portion of plane outside $H_t$, such that for all points $x \in S(e)$, we have that $\angle xv_1v_2 \leq \frac{180 - \angle v_0v_1v_2}{4}$ and $\angle v_1v_2x \leq \frac{180 - \angle v_1v_2v_3}{4}$ (the left of Fig. 3.1). Side robots may not always be able to compute $S(e)$ exactly due to the mutual obstructions of visibility leading to different local views. However, if there is only one side robot in $e$, then it can compute $S(e)$ exactly. When there are more than one robot in $e$, $S'(e)$ computed by a robot based on its local view is such that $S'(e) \subseteq S(e)$ (the right of Fig. 3.1 for robot $r$).

3.3 The Mutual Visibility Algorithm

In this section, we present an algorithm (Algorithm 1) that solves MUTUAL VISIBILITY for $n \geq 1$ robots with unit disc size under RIGID moves in the robots with lights model. Our algorithm works for both FSYNCH and SSYNCH settings of robots. We analyze this algorithm for the FSYNCH setting in Section 3.4 and discuss in Section 3.5 how the analysis extends for the SSYNCH setting. Algorithm 1 uses Algorithms 2–5 as subroutines and needs 4 colors in the color set $C = \{\text{Off, Yellow, Blue, Red}\}$.

Algorithm 1 works in two phases similar to the previous MUTUAL VISIBILITY algorithms for dimensionless robots [33, 40]: (i) interior depletion (ID) and (ii) side depletion (SD). However, in [33, 40], the robots that are already in the corner of the convex hull never move and only the robots that are either in the interior or in the side move during the execution of the algorithm. However, due to the extents of robots, there are situations in this problem where we may need to
move corner robots as well to make space for interior robots to move to the hull boundary. Since robots are silent (except lights), movement of one corner may trigger the moves of other corner robots, and the goal is to stop the corner robots from moving forever as soon as all the internal robots reach the hull boundary. Achieving this is non-trivial, and our focus will be on handling these moves. Asynchronicity of robot settings exacerbates this situation.

The interior depletion phase moves the robots that are in the interior of the hull to the boundary of the hull and assign the color Red or Blue to their lights. Specifically, the robots in the corner of the convex hull $H_t$ will be assigned color Red to their lights and the robots on the edges of $H_t$ will be assigned color Blue to their lights. Figure 3.2 shows how the ID phase works.

The side depletion phase moves the robots that are on the edges of $H_t$ to outside of $H_t$, make them corners, and assign light Red to their lights. Figure 3.3 shows how the SD phase works.
More formally, let $C_\gamma$ be a configuration at some time $\gamma > 0$ such that there is no robot in the interior of $H_\gamma$, i.e., all the robots in $R$ are in the set $\partial H_\gamma$. The goal of the ID phase is to reach a configuration $C_\gamma$ at some time $\gamma > 0$. The goal of the SD phase is to move all the robots in the set $S_\gamma$ (side robots) to the outside of $H_\gamma$ to make them corner robots and assign light Red to their lights. After all the robots in $S_\gamma$ become corner robots, our algorithm terminates solving Mutual Visibility. We proceed with describing in detail how the ID and SD phases work and then discuss some special cases.
Algorithm 1: MUTUAL VISIBILITY algorithm for robot $r_i$ of unit disc size for any time $k > 0$

1 // Look-Compute-Move cycle for robot $r_i$ of unit disc size
2 $C_k(r_i) \leftarrow$ configuration $C_k$ for robot $r_i$ (including $r_i$);
3 $H_k(r_i) \leftarrow$ convex hull of the positions of the robots in $C_k(r_i)$;
4 if $|C_k(r_i)| = 1$ then Terminate;
5 else if $H_k(r_i)$ is a line segment then
6     if $|C_k(r_i)| = 2$ then
7         Let $r_j \in C_k(r_i)$;
8         if $r_i$.light $=$ Off then
9             Move orthogonal to line $\overrightarrow{r_ir_j}$ by any non-zero distance;
10            $r_i$.light $\leftarrow$ Red;
11         else if $r_j$.light $=$ Red then Terminate;
12     else if $|C_k(r_i)| = 3$ then
13         Let $r_j, r_l \in C_k(r_i)$;
14         if $r_i$.light $=$ Off $\wedge r_j$.light $=$ Red $\wedge r_l$.light $=$ Red then
15             Move orthogonal to line $\overrightarrow{r_jr_l}$ by any non-zero distance;
16            $r_i$.light $\leftarrow$ Red;
17     else if $r_i$ is a vertex robot of $H_k(r_i)$ then Corner($r_i, C_k(r_i), H_k(r_i)$);
18     else if $r_i$ is an interior robot of $H_k(r_i)$ then Internal($r_i, C_k(r_i), H_k(r_i)$);
19     else if $r_i$ is a side robot of $H_k(r_i)$ then Side($r_i, C_k(r_i), H_k(r_i)$);

3.3.1 Interior Depletion

Recall that the objective of this phase is to move the robots that are in the interior of the hull to the boundary of the hull and assign the color Red to the lights of the corner robots in $H_k$ and Blue to the lights of the edge robots.

The phase is crucial for MUTUAL VISIBILITY and asks both the interior and corner robots to move until all the interior robots are successfully accommodated on the convex hull boundary. We will discuss in the following how interior and corner robots move to have all the robots on the boundary of $H_k$ for some time $k > 0$. The interior depletion is done using Algorithms 2 and 3.

In any initial configuration $C_0$, the lights of all robots in $R$ are set to Off. If a robot $r_i$ with light Off is activated at some time $k \geq 0$, and it sees that $C_k(r_i)$ contains a region of plane that is free of robots and wider than $180^\circ$ (wide exactly $180^\circ$), then $r_i$ knows that it is a corner (side) robot of $H_k$ and sets its light to Red (Blue). After all the side and corner robots of $H_k$ activated once, all the robots in the side and corner of $H_k$ are colored Red or Blue. If a robot $r_i$ with light Off is activated at some time $k \geq 0$, and it sees that it is in the interior of $H_k(r_i)$, it tries to move to position itself...
Algorithm 2: \textit{Internal}(r_i, C_k(r_i), H_k(r_i))

1. if $r_i\.light = \text{Red}$ then
2. $e \leftarrow$ edge in $H_k(r_i)$ that is closest to $r_i$;
3. $m \leftarrow$ intersection point of $e$ and a line $L$ that is perpendicular to $e$ and passes through $r_i$;
4. Move to the point in $L$ distance 1 away from point $m$;
5. else if $r_i\.light = \text{Blue}$ then
6. if $r_i$ sees at least a robot with light $\in \{\text{Off, Yellow}\}$ then
7. $e \leftarrow$ edge in $H_k(r_i)$ that is closest to $r_i$;
8. Move to the intersection point of $e$ and a line $L$ that is perpendicular to $e$ and passes through $r_i$;
9. else
10. Let $H_{\text{Blue}}$ be the half-plane such that there are only robots with lights \text{Blue} in it;
11. $L \leftarrow$ line perpendicular to the edge robot $r_i$ belongs to in the convex hull in the other side of $H_{\text{Blue}}$ that passes through $r_i$;
12. if there are at least 2 robots in $H_{\text{Blue}}$ then
13. $L' \leftarrow$ the line formed by the robots in $H_{\text{Blue}}$;
14. Move to the intersection point of $L$ and $L'$;
15. else
16. if at least one neighbor of $r_i$ in the edge it belongs to in the other side of $H_{\text{Blue}}$ has light \text{Red} \& no robot in the side of $r_i$ in $H_{\text{Blue}}$ divided by line $L$ then
17. Execute Lines 4–8 of Algorithm 3;
18. else
19. Order the robots in $H_k(r_i)$ starting from any arbitrary robot $v_1$ in the clockwise order so that $T = \{v_1, \ldots, v_{\text{last}}, v_1\}$, where $v_1$ is the first robot and $v_{\text{last}}$ is the last robot;
20. Let $c, d$ be any pair of two consecutive robots in $T$ with $c\.light = \text{Red}$ and $d\.light = \text{Red}$;
21. Let $H_{\text{cd}}$ be the half-plane divided by line parallel to $\overrightarrow{cd}$ that passes through $r_i$ such that $c, d$ are in $H_{\text{cd}}$;
22. $Q \leftarrow$ set of line segments $\overrightarrow{ef}$ such that:
23. (a) the triangle $r_i, c, d$ does not contain neither inside nor on its edges any other robot of $C_k(r_i)$ except the robots in edge $\overrightarrow{cd}$, and
24. (b) either there is no robot in $\overrightarrow{cd}$ or all the robots in it have light \text{Blue}, and
25. (c) there is no robot in $C_k(r_i) \setminus H_k(r_i)$ closer to edge $\overrightarrow{cd}$ than $r_i$, and
26. (d) there are no two robots with equal distance to $\overrightarrow{cd}$ appearing to both the counterclockwise and clockwise direction of $r_i$ with respect to the local coordinate system of $r_i$ (however, there might be robots in either the counterclockwise or the clockwise direction of $r_i$);
27. if $Q$ is not empty then
28. $\overrightarrow{ef} \leftarrow$ the line segment in $Q$ between two robots $e, f$ that is closest to $r_i$;
29. if $r_i\.light = \text{Yellow}$ then
30. if there is no robot with light \text{Blue} in edge $\overrightarrow{ef}$ then
31. if there is no other robot with light \text{Yellow} that is at equal distance to $\overrightarrow{ef}$ then
32. $m \leftarrow$ midpoint of $\overrightarrow{ef}$;
33. Move($r_i, C_k(r_i), H_k(r_i), e, f, m)$;
34. else if there exists a robot in the clockwise direction of $r_i$ (with respect to the local coordinate system of $r_i$) with light \text{Yellow} that is at equal distance to $\overrightarrow{ef}$ then
35. $m \leftarrow$ point in $\overrightarrow{ef}$ at $\text{length}(\overrightarrow{ef})$ from endpoint $e$;
36. Move($r_i, C_k(r_i), H_k(r_i), e, f, m)$;
37. else if there exists a robot in the counterclockwise direction of $r_i$ (with respect to the local coordinate system of $r_i$) with light \text{Yellow} that is at equal distance to $\overrightarrow{ef}$ then
38. $m \leftarrow$ point in $\overrightarrow{ef}$ at $\text{length}(\overrightarrow{ef})$ from endpoint $f$;
39. Move($r_i, C_k(r_i), H_k(r_i), e, f, m)$;
40. else
41. $g, h \leftarrow$ robots closer to endpoints $e$ and $f$, respectively, in edge $\overrightarrow{ef}$;
42. if there is no other robot with light \text{Yellow} that is at equal distance to $\overrightarrow{ef}$ then
43. $\overrightarrow{gh} \leftarrow$ choose one between $\overrightarrow{ef}$ or $\overrightarrow{gh}$ arbitrarily;
44. $m \leftarrow$ midpoint of $\overrightarrow{gh}$;
45. Move($r_i, C_k(r_i), H_k(r_i), e, f, m)$;
46. else if there exists a robot in the clockwise direction of $r_i$ (with respect to the local coordinate system of $r_i$) with light \text{Yellow} that is at equal distance to $\overrightarrow{ef}$ then
47. $m \leftarrow$ midpoint of $\overrightarrow{gh}$;
48. Move($r_i, C_k(r_i), H_k(r_i), e, f, m)$;
49. else if there exists a robot in the counterclockwise direction of $r_i$ (with respect to the local coordinate system of $r_i$) with light \text{Yellow} that is at equal distance to $\overrightarrow{ef}$ then
50. $m \leftarrow$ midpoint of $\overrightarrow{gh}$;
51. Move($r_i, C_k(r_i), H_k(r_i), e, f, m)$;
52. else if $r_i\.light = \text{Off}$ then
53. $m \leftarrow$ intersection point of $\overrightarrow{ef}$ and a line $L$ perpendicular to $\overrightarrow{ef}$ that passes through $r_i$;
54. if $\text{length}(\overrightarrow{ef}) > 2$ then Move distance 1 to the point in $L$ towards $\overrightarrow{ef}$ and set $r_i\.light \leftarrow \text{Yellow}$.
Algorithm 3: Corner($r_i, C_k(r_i), \mathbb{H}_k(r_i)$)

1. if $r_{i,\text{light}} = \text{Off}$ then $r_{i,\text{light}} \leftarrow \text{Red}$;
2. else if $\forall r \in C_k(r_i), r_{\text{light}} = \text{Red}$ then Terminate;
3. else if $r_{i,\text{light}} = \text{Blue}$ then
4. $L \leftarrow$ edge of $\mathbb{H}_k(r_i)$ closest to $r_i$ with $x_{\text{light}} = y_{\text{light}} = \text{Red}$;
5. if there is no robot in edge $L$ then $r_{i,\text{light}} \leftarrow \text{Red}$;
6. else if $\forall r \in I_k(r_i), r_{\text{light}} \in \{\text{Off, Yellow}\}$ then
7. $a \leftarrow$ clockwise neighbor in the boundary of $\mathbb{H}_k(r_i)$;
8. $b \leftarrow$ clockwise neighbor in the boundary of $\mathbb{H}_k(r_i)$;
9. if $\text{length}(\overline{a}b) < 2 \vee \text{length}(\overline{b}c) < 2$ then
10. $L \leftarrow$ line that is angle bisector of $\angle ar_i b$;
11. Move to a point in $L$ that is distance 1 away from $r_i$ in the outside of $\mathbb{H}_k(r_i)$;
else
12. $r_j, r_k \leftarrow$ robots in the interior of $\mathbb{H}_k(r_i)$ that are closest to edges $\overline{a}b$ and $\overline{c}d$, respectively (if there are more than one such robots, take as $r_j, r_k$ the robots that are closer to $r_i$);
13. if $r_{j,\text{light}} = \text{Off}$ then
14. $m \leftarrow$ intersection point of $\overline{a}b$ and line $L$ perpendicular to $\overline{a}b$ that passes through $r_j$;
15. if $\text{length}(\overline{a}m) < 2$ then
16. $L' \leftarrow$ line that is angle bisector of $\angle ar_i b$;
17. Move to a point in $L'$ that is distance 1 away from $r_i$ in the outside of $\mathbb{H}_k(r_i)$;
18. else if $r_{k,\text{light}} = \text{Off}$ then
19. $m \leftarrow$ intersection point of $\overline{c}d$ and line $L$ perpendicular to $\overline{c}d$ that passes through $r_k$;
20. if $\text{length}(\overline{c}m) < 2$ then
21. $L' \leftarrow$ line that is angle bisector of $\angle ar_i b$;
22. Move to a point in $L'$ that is distance 1 away from $r_i$ in the outside of $\mathbb{H}_k(r_i)$;
else if $r_{j,\text{light}} = \text{Yellow}$ then
23. if $a_{\text{light}} = \text{Red}$ then
24. $m \leftarrow$ point in $\overline{a}b$ at length($\overline{a}b$)/4 from $r_i$;
else
25. $m \leftarrow$ point in $\overline{a}b$ at length($\overline{a}b$)/3 from $r_i$;
26. else
27. $m \leftarrow$ midpoint of $\overline{a}b$;
28. $L_{r_j m} \leftarrow$ line segment connecting $r_j$ and $m$;
29. $L'_{r_j m}, L''_{r_j m} \leftarrow$ lines parallel to $L_{r_j m}$ at distance 1/2 in both sides of it;
30. if $L'_{r_j m}$ or $L''_{r_j m}$ shares at least a point occupied by any other robot towards edge $\overline{a}b$ then
31. $L \leftarrow$ line that is angle bisector of $\angle ar_i b$;
32. Move to a point in $L$ that is distance 1 away from $r_i$ in the outside of $\mathbb{H}_k(r_i)$;
else if $r_{k,\text{light}} = \text{Yellow}$ then
33. if $b_{\text{light}} = \text{Red}$ then
34. $m \leftarrow$ midpoint of $\overline{c}d$;
else
35. $m \leftarrow$ point in $\overline{c}d$ at length($\overline{c}d$)/3 from $r_i$;
36. else
37. $m \leftarrow$ midpoint of $\overline{c}d$;
38. $L_{r_k m} \leftarrow$ line segment connecting $r_k$ and $m$;
39. $L'_{r_k m}, L''_{r_k m} \leftarrow$ lines parallel to $L_{r_k m}$ at distance 1/2 in both sides of it;
40. if $L'_{r_k m}$ or $L''_{r_k m}$ shares at least a point occupied by any other robot towards edge $\overline{c}d$ then
41. $L \leftarrow$ line that is angle bisector of $\angle ar_i b$;
42. Move to a point in $L$ that is distance 1 away from $r_i$ in the outside of $\mathbb{H}_k(r_i)$;
Algorithm 4: Move(r_i, C_k(r_i), H_k(r_i), e, f, m)

1. \(L_{r,m} \leftarrow \) line segment connecting \(r_i\) and \(m\);
2. \(L'_{r,m}, L''_{r,m} \leftarrow \) line parallel to \(L_{r,m}\) at distance 1/2 in both sides of \(L_{r,m}\) towards \(e\);
3. if \(L'_{r,m}\) and \(L''_{r,m}\) share no point occupied by any other robot then
   4. Move to \(m\);
   5. \(r_i\).light \(\leftarrow\) Blue;

Algorithm 5: Side(r_i, C_k(r_i), H_k(r_i))

1. if \(r_i\).light = Off then \(r_i\).light \(\leftarrow\) Blue;
2. else if \(\forall r \in C_k(r_i), r\).light = \{Blue, Red\} \& no robot \(r \in C_k(r_i)\) is in the interior of \(H_k(r_i)\) then
   3. if at least one neighbor of \(r_i\) in the edge it belongs to has light Red then
      4. Order the robots in the counterclockwise order of \(r_i\) (with respect to the local coordinate system of \(r_i\)) such that the order is \(T_i = \{v_3, v_2, r_i, v_0\}\), where \(v_3\) is the first robot non-collinear to \(r_i\) in the clockwise direction of \(r_i\) with \(v_3\).light = Red, \(v_2\) is the robot that is collinear with \(r_i\) in the clockwise direction of \(r_i\) with \(v_2\).light \(\in\) \{Blue, Red\}, and \(r\) is the collinear robot in the counterclockwise direction of \(r_i\) with \(r\).light \(\in\) \{Blue, Red\}, and \(v_0\) is the first non-collinear robot to \(r_i\) in the counterclockwise direction of \(r_i\) with \(v_0\).light = Red;
   5. Compute angles \(\alpha = 180 - \angle v_3 r v_1\) and \(\beta = 180 - \angle r_v_2 v_3\), and set \(\delta = \min\{\alpha/4, \beta/4\}\);
   6. Compute a point \(x'\) such that \(\angle x' v_2 r_i = \delta\) and a point \(x''\) such that \(\angle x'' r v_i = \delta\);
   7. \(x \leftarrow x'\) or \(x''\) whichever is nearest to \(e\);
   8. Move perpendicular to \(e\) with destination \(x\);

on an edge of \(H_k(r_i)\).

The move of \(r_i\) is done as follows. Robot \(r_i\) first checks whether any edge of \(H_k(r_i)\) is in fact an edge of \(H_k\) (the global convex hull). If no edge in \(H_k(r_i)\) is of \(H_k\), \(r_i\) does not move. It determines whether some edge of \(H_k(r_i)\) is of \(H_k\) by checking whether there is an edge in \(H_k(r_i)\) such that two endpoint robots of that edge have light colored Red. In other words, if \(r_i\) sees an edge in \(H_k(r_i)\) with endpoint robots colored Red, it is an edge of \(H_k\); otherwise \(H_k(r_i)\) does not contain any edge of \(H_k\).

However, if \(r_i\) moves as soon as it sees at least an edge of \(H_k\) with endpoint robots colored Red in its \(H_k(r_i)\), collisions might occur. This is because other interior robots may move at the same time towards \(H_k\) and their paths may cross. Therefore, \(r_i\) uses the technique described in the following to determine whether it is eligible to move to an edge of \(H_k(r_i)\) or not after it sees that at least one edge of \(H_k(r_i)\) has endpoint robots with lights Red. It then moves when it finds itself eligible to move to that edge. The eligibility for \(r_i\) is determined as follows. It orders the edges of \(H_k(r_i)\) in the clockwise order with respect to its own coordinate system starting from and
Figure 3.4: An illustration of how internal robots select edges of $H_k$ to include in their $Q$'s. The robot $r_i$ includes $ef$ and $e\overline{v_1}$ in its $Q$. Robots $r_j$ has $\overline{ef}$, and $r_l$ has $fv_2$ in their $Q$'s. However, the robots between $r_i, r_j$ have empty $Q$'s.

ending at any arbitrary corner robot $v_1 \in H_k(r_i)$. An edge connecting two corner robots might be fragmented to many sub-edges when there are one or more side robots with lights $Blue$ on that edge. In this case, the order $T$ calculated by $r_i$ takes into account only the robots at the endpoints of that edge; that is, only the corner robots of $H_k(r_i)$ will be included in the order $T$ computed by $r_i$.

Let $c, d$ be any pair of two consecutive robots in $T$ such that they both have lights $Red$. Let $L$ be a line parallel to the edge $\overline{cd}$ that passes through (the position of) $r_i$. Let $HP_{cd}$ be the half-plane divided by line $L$ such that robots $c, d$ are in that half-plane. The robot $r_i$ includes in $Q$ the edge $\overline{cd}$ if the following three conditions are satisfied: (a) there is no other robot with light $Off$ in the triangular area $r_i\overline{cd}$ (inside or on its edges), (b) there is no other robot closer to $\overline{cd}$ than $r_i$, and (c) either there is no other robot on line $L$ or if there are robots on line $L$ with lights $Off$, the robots are either only in the counterclockwise direction of $r_i$ or only in the clockwise direction of $r_i$. Robot $r_i$ repeats this process for all the edges in $T$ such that endpoint robots of those edges have light colored $Red$. See Fig. 3.4 for an example where $r_i$ satisfies all three criteria for edges $\overline{ef}$ and $\overline{e\overline{v_1}}$ so both the edges will be included in $Q$ maintained by $r_i$. For $r_j$, there will be only $\overline{ef}$ in its $Q$ since other robots are closer to edges $\overline{e\overline{v_1}}$ and $\overline{fv_2}$ than $r_j$. The edge $\overline{fv_2}$ is included in $r_i$'s $Q$ since $r_i$ is the closest to $\overline{fv_2}$ compared to other robots. Note also that robots on the line between $r_i$ and $r_j$ are not eligible and their set $Q$ will be empty since these robots see other internal robots equidistant to $\overline{ef}$ in both their counterclockwise and clockwise directions.

Although the selection process for the edges to include in $Q$ helps to avoid collisions, it is
Figure 3.5: An illustration of how an internal robot, $r_i$, chooses a point to move to on the edge $ef \in Q$ that is the closest to when it is the only closest robot to $ef$. Since there are side robots on $ef$, $r_i$ chooses either $m$ or $m'$ as the point on $ef$ to try to move to. If there were no side robots on $ef$, $r_i$ would have chosen the midpoint of $ef$ as the point to try to move to.

not the complete solution however. Therefore, among the edges $Q$, $r_i$ is asked to choose an edge $ef \in Q$ such that $ef$ is the closest edge to $r_i$ compared to the other edges of $Q$. Note that the closest edge is measure with respect to the perpendicular distance from $r_i$ to that edge. Moreover, while moving to $ef$, the point $m \in ef$ where $r_i$ targets to move to is also carefully computed. Robot $r_i$ computes the point $m$ in edge $ef$ (the closest edge among the edges in $Q$) as follows. If there is no side robot on $ef$ and $r_i$ is the only robot that is closest to $ef$, then $m$ for $r_i$ is the midpoint of $ef$. (Robot $r_i$ knows it is the only robot that is closest to $ef$ if there is no other robot on the line $L$ parallel to $ef$ passing through $r_i$.) If $r_i$ sees only one other equidistant robot in the counterclockwise direction, $m$ is the point in $ef$ at length$(ef)/3$ from endpoint $f$ in $ef$. However, if $r_i$ sees only one other equidistant robot in the clockwise direction, then $m$ is the point in $ef$ at length$(ef)/3$ from endpoint $e$. Recall that internal robots that see equidistant robots in their both directions are not eligible to move. If there are side robots on $ef$, then if $r_i$ is the only closest robot to $ef$, then the robot $r_i$ computes in $ef$ is either $m$ or $m'$, the midpoints of $eg$ and $hf$, as shown in Fig. 3.5 where $g, h$ are the side robots on $ef$ that are neighbors of $e, f$, respectively. If there are equidistant robots to $r_i$, then the point $m_i$ for $r_i$ is either the midpoint of $eg$ or the midpoint of $hf$ depending on in which direction it sees other equidistant robot to $ef$ (see Fig. 3.6).

Note that the line connecting $r_i$ with the point $m_i$ computed in $ef$ may not be perpendicular to $ef$. Therefore, even if $r_i$ is the only closest robot to $ef$, it may collide with other robots due to their extents while moving toward $m$. One such example is given in Fig. 3.7 where $r_i$ collides with
Figure 3.6: An illustration of how an internal robot $r_i$ chooses a point to move to on the edge $ef \in Q$ that is the closest to when there are other robots that are equidistant to $ef$. Since there are side robots on $ef$, and $r_i$ sees equidistant robots in only one direction, $r_i$ chooses $m_i$ as the point on $ef$ to try to move to. Robots can do this without agreeing on the coordinate system of other robots. Since $r_j$ also sees equidistant robots in only one direction, it chooses point $m_j$ to try to move to. Note that, the robots in between $r_i$ and $r_j$ do not try to move to.

The robots near to it while going toward either of $m$ or $m'$. Therefore, for $r_i$ being able to move to the computed point in $ef$ without colliding with other robots, we first make an intermediate move for robot $r_i$ such that it does not collide with any other internal robot when it makes a move to the computed point in $ef$ from that intermediate position. Robot $r_i$ needs to make this intermediate move only once and after that it can move to the computed point in $ef$ without colliding with other internal robots. The color is changed to Yellow from Off to indicate that $r_i$ moved once.

The intermediate move for $r_i$ is done as follows. Let $L'$ be a line perpendicular to $ef$ that passes through $r_i$ and $m''$ be the intersection point of $L'$ and $ef$. If $\text{length}(r_im'')$ is at least 2, then $r_i$ moves distance 1 towards $m''$ and changes its color to Yellow (see Fig. 3.8). If $\text{length}(r_im'')$ is less than 2, $r_i$ waits until it becomes at least 2. This happens since as we describe below the robots $e, f$ also detect this situation and move outside of the hull to make sure that $\text{length}(r_im'')$ becomes at least 2.

Although making an intermediate move for $r_i$ avoids collisions with other internal robots, however $r_i$ (after colored Yellow) still needs to check the possibility of collision with the robots on $ef$ if it moves to the computed point $m_i$ in $ef$. This is handled as follows. Robot $r_i$ draws two lines parallel to line $r_im_i$ one in each side with distance $1/2$ from it. Robot $r_i$ then makes a move to $m_i$ if and only if these lines share no point of other robots towards edge $ef$ from $r_i$; for example, Yellow colored robot in Fig. 3.8 can move to point $m$ in $ef$ as the parallel (dotted) lines to $r_im$. 

27
share no point of other robots.

If \( r_i \) can not make a move to the computed point in \( \overline{ef} \) from its intermediate position due to possible collisions with other robots on \( \overline{ef} \), the robots on \( \overline{ef} \) must detect that situation and move appropriately to a direction outside of the hull so that \( r_i \) can move to \( \overline{ef} \) without colliding with other robots. The robots on \( \overline{ef} \) indeed detect that situation and move outside of the hull.

We describe here how \( e \) figures out the necessity of move (it follows similarly for \( f \)). Let \( a, b \) be the clockwise and counterclockwise neighbors of \( e \) in \( \mathbb{H}_k \). Particularly, the robot \( e \) moves outside when it detects the following three situations: (a) either \( \text{length}(\overline{ea}) < 2 \) or \( \text{length}(\overline{eb}) < 2 \); (b) the distance from the closest internal robot \( r_i \) with light \( \text{Off} \) to either \( \overline{ea} \) or \( \overline{eb} \) is < 2; and (c) the internal robot with light \( \text{Yellow} \) collides with other robots if it moves to the destination point \( m \) computed in either \( \overline{ea} \) or \( \overline{eb} \). In all situations, \( e \) draws a line that bisects the angle \( \angle aeb \) and moves distance 1 away from the convex hull following that line. These situations are illustrated in Figs. 3.9–3.11. Figs. 3.9 illustrates the first situation in which either \( \text{length}(\overline{ea}) < 2 \) or \( \text{length}(\overline{eb}) < 2 \). Fig. 3.10 illustrates the second situation in which the distance from the interior robot \( r_i \) to its closest edge \( \overline{ef} \) is < 2. Finally, Fig. 3.11 illustrates the third situation in which after becoming \( \text{Yellow} \), the internal robot still collides with the robots on \( \overline{ef} \). Note also that \( e \) does not move forever since the situations described above are triggered only when \( e \) sees at least a robot with light either \( \text{Yellow} \) or \( \text{Off} \). That means, \( e \) does not move if it does not see any robot with those lights.

The side robots, if any, that were originally on \( \overline{ef} \) (before \( e, f \) move outside) also move to \( \overline{ef} \) before any internal robot with light \( \text{Yellow} \) moves to \( \overline{ef} \). This is done by drawing a line perpendicular to \( \overline{ef} \) and moving to the intersection point of \( \overline{ef} \) and the line perpendicular to \( \overline{ef} \) going through the position of that side robot (which now became internal due to the moves of \( e \) and/or \( f \) outside of the hull).

Note also that the moves of corners \( e, f \) outside the hull may trigger the moves of other corner robots of \( \mathbb{H}_k \), which otherwise would not move. This is because some corner robots may become internal due to the moves of its neighboring corner robots outside of \( \mathbb{H} \). The corner robot \( r_j \) that becomes internal handles this case as follows. It finds the closest edge \( \hat{e} \) in \( \mathbb{H}_k(r_j) \) (with two corner
Figure 3.7: An illustration of the configuration in which the interior robot that is the closest to some edge $ef$ collides with some other internal robot while moving toward a computed point (one of $m, m'$) on $ef$. An intermediate move we describe in Fig. 3.8 handles this case.

Figure 3.8: An illustration of the intermediate move done by an interior robot, $r_i$, for the situation of Fig. 3.7. Since robot $r_i$ finds that $\text{length}(r_i m'') \geq 2$, it moves distance 1 towards $ef$ on line $r_i m''$ and changes its color to Yellow, where $m''$ is the intersection point of line $L$ perpendicular to $ef$ passing through $r_i$ and $ef$. It can move to $m$ since no position of other robots fall between dotted parallel lines w.r.t. $r_i m$ going from distance $1/2$ from $r_i$. If $\text{length}(r_i m'') < 2$, $r_i$ waits until the length becomes at least 2.

robots have light Red) and draws a line $L$ perpendicular to that edge going through it. Let $m$ be the intersection point of $L$ and $\hat{e}$. Robot $r_j$ then moves distance 1 away from $m$ in line $L$ outside the hull without changing its color.

Note that only 4 colors \{Off, Yellow, Blue, Red\} are used to successfully execute the ID phase.
Figure 3.9: A corner robot $e$ moves to a direction away from the convex hull if the distance between any of its left or right neighbor in $H_k$ is $< 2$. It draws a line which bisects the angle $\angle aeb$ and moves a unit distance on that line away from the convex hull.

Figure 3.10: A corner robot $e$ moves to a direction away from the convex hull if the distance from the closest internal robot to either of $ea$ or $eb$ is $< 2$. It draws a line which bisects the angle $\angle aeb$ and moves a unit distance on that line away from the convex hull.
Figure 3.11: A corner robot $e$ moves to a direction away from the convex hull if it realizes that the Yellow robot collides with some other robot on the edge $ea$ or $eb$ if it moves to its destination $m$ in either $ea$ or $eb$. It draws a line which bisects the angle $\angle aeb$ and moves a unit distance on that line away from the convex hull.

### 3.3.2 Side Depletion

This phase starts only after the ID phase finishes. In particular, the ID phase finishes when there is no robot with light $\in \{\text{Off, Yellow}\}$ in $H_k$, $k \in \mathbb{N}^+$. Actually, we will prove later that the SD phase does not start when there are robots with light $\in \{\text{Off, Yellow}\}$ in the system. This phase is handled using Algorithm 5. In this phase, corner robots do not move, and only the side robots (with light Blue) move to become corner robots of $H_k$ and change their light color to Red. After all side robots change their color to Red, the robots terminate solving the Mutual Visibility configuration.

In this phase, a side robot, as soon as it is activated, moves from its edge $e = v_1v_2$ to a point in the safe zone $S(e)$ (as defined in Section 3.2). Specifically, if a side robot $r_i$ with light Blue is activated, it checks whether one of its neighbors on the edge it belongs to has light Red. The Blue robots that satisfy this criteria move to a point in $S(e)$ without changing their color. There are at most 2 such robots on $e$ which make such moves at the same time. Fig. 3.12 illustrates one such move of two side robots $r_i, r_j$ that are neighbors of two corner robots $v_2, v_1$ of edge $e$, respectively. If there are no more robots on $e$ then the robots that moved to $S(e)$ become corners when they wake up next time. Otherwise, they wait until all the side robots on $e$ move to the line connecting them. Fig. 3.13 illustrates such waiting since $r_i, r_j$ of Fig. 3.12 wait until all robots on the edge $v_1v_2$ move to the line connecting $r_i, r_j$ before becoming corner robots of $H$. Note that $r_i, r_j$ do not
move in future rounds after becoming corners.

After the moves of $r_i, r_j$ to their safe areas, the remaining robots on $e$ move to the line formed by the robots that moved to $S(e)$ as follows. Let $L'$ be a line that connects the robots in $S(e)$. A robot on $e$ draws a line $L$ perpendicular to $e$ passing through its position and moves to the intersection point of $L$ and $L'$. The Blue robots that became internal can do this move because their color signifies them that although they are internal they do not need to perform interior depletion as described in the previous subsection.

Due to the asynchronicity of robot settings, no two robots may reach outside $v_1v_2$ at the same time. As shown in Fig. 3.14, suppose $r_j$ reached $S(e)$ first. Then, $r_j$ sees all robots on $e$ and waits until $r_i$ moves to $S(e)$. The robot $r$ does not move to $S(e)$ before $r_i$ moves since $r$ sees $v_1$ and $r_j$ in two opposite sides of line $L$ perpendicular to $e$ passing through $r$. Therefore, when all robots on $e$ wake up one time 2 robots become corners and the remaining robots become side robots again and continue the side depletion process until they eventually become corners of $\mathbb{H}$. The number of colors are still 4 since we did not introduce new color in the SD phase.
Figure 3.13: Robots $r_i$ and $r_j$ that moved to their local $S(e)$ in Fig. 3.12 change their lights to *Red* only after all robots on $e$ moved to $r_i r_j$. Robots $r_i, r_j$ do not move in future rounds.
Figure 3.14: If there is only one robot in $S(e)$, then $r$ waits until $r_i$ moves to $S(e)$. Robot $r$ sees only one robot in $S(e)$ on one side of the line $L$ perpendicular to $e$ going through its position, and it also sees robots on the other side of $L$. Moreover, it is not adjacent to $v_2$.

3.3.3 Special Cases

There are two special cases in our algorithm. The first special case is when $n = 1$. This case can be easily recognized by the only robot in the system since it does not see any other robot. The second special case is when $H_0$ formed by $C_0$ is a line segment. In this case, we differentiate the robots that see only 1 other robot (i.e., $|C(r_i)| = 2$) with the robots that see 2 other robots (i.e., $|C(r_i)| = 3$). Let $|C(r_i)| = 2$ and $r_j$ be the only other robot that $r_i$ sees. In this case, when $r_i$ is activated for the very first time it moves orthogonal to the line $r_i r_j$ some positive distance and sets its light to Red. When $r_i$ is activated in future rounds and $H$ is still a line segment, it does nothing until it sees $r_j$ with light Red. As soon as $r_i$ sees $r_j$ with light Red, $r_i$ recognizes this situation and terminates. Let $|C(r_i)| = 3$ and $r_j$ and $r_l$ be the two other robots that $r_i$ sees. Robot $r_i$ is able to tell if $H_k(r_i)$ is a line segment, i.e., if all the robots in $R$ are collinear. In this case, $r_i$ will move orthogonal to line $r_j r_l$ and set its light to Red if and only if it sees that the lights of $r_j$ and $r_l$ are set to Red. This is only useful when $n = 3$. For $n \geq 4$, we can also prove similarly as in the case of $n = 3$ such that the line segment $H_0$ correctly evolves into a polygonal configuration from which the normal algorithmic process of the ID and the SD phases starts. Actually, the orthogonal moves of (at least) endpoint robots of the line segment $H_0$ provide that polygonal configuration and that polygonal configuration never fall backs to a line segment configuration.
3.4 Analysis of the Algorithm for the FSYNCH setting

We prove the following lemmas and theorems which guarantee that given any initial configuration $C_0$ of a system of $n \geq 1$ fat robots, the robots in $I_0$ successfully deplete to the convex hull boundary in finite time and then become corners solving the Mutual Visibility problem. The lemmas and theorems we prove in this section apply for the FSYNCH setting; we discuss how they extend to the SSYNCH setting later in Section 3.5. We also prove that no collisions of robots occur during the execution of the algorithm. We use round $k$ instead of time $k$ due to the assumption of the FSYNCH setting in this section.

We first prove lemmas and theorems for the ID phase. The goal here is to prove that $I_k = \emptyset$ in some round $k \in \mathbb{N}^+$. We will show later that the SD phase starts only after the ID phase finishes. This allows us to analyze the ID and SD phases separately. We start with this lemma.

**Lemma 3.4.1** In each round $k \in \mathbb{N}^+$ until $I_k = \emptyset$, there is at least one robot in $I_k$ for which the set of line segments $Q$ is not empty.

**Proof.** Let $H_k'$ be the convex hull of the robots in $I_k$. Note that $H_k'$ is either a line segment or a polygon with at least 3 corners. For the line segment case, at least two endpoint robots of $H_k'$ are closer to some (different) edges of $H_k$. Note that $H_k$ must have at least three corners, otherwise the robots in $I_k$ would not be interior robots. For the polygonal shape $H_k'$, at least 3 corners of $H_k'$ are closer to some edges of $H_k$. Therefore until $I_k \geq 2$, at least 2 robots in $I_k$ are the closest to some edges of $H_k$, and those edges are in their $Q$’s. For the case of $I_k = 1$, there is only one robot which has non-empty $Q$. The lemma follows. \[\square\]

**Lemma 3.4.2** There $\exists k \in \mathbb{N}^+$ such that the robots in $I_k$ with non-empty $Q$ are colored light Yellow.

**Proof.** Let $r_i$ be a robot in $I_k$ with non-empty $Q$. We have from Lemma 3.4.1 that one such robot exists until $I_k = \emptyset$. Let $ef$ be an edge that is the closest to $r_i$ among all the edges in $Q$. 35
Let $L$ be a line perpendicular to $ef$ passing through $r_i$ and $m''$ be the intersection of $L$ and $ef$. If $\text{length}(ri, m') \geq 2$, $r_i$ simply able to move unit distance towards $ef$ and changes its color to Yellow.

If $\text{length}(ri, m') < 2$, $e, f$ detect that situation and move outside of the hull until $\text{length}(ri, m') \geq 2$.

Therefore, $\exists k \in \mathbb{N}^+$ such that each robot in $I_k$ with non-empty $Q$ are colored light Yellow.

**Lemma 3.4.3** There $\exists k \in \mathbb{N}^+$ such that the robots in $I_k$ with light Yellow can be able to move to the boundary of the convex hull $H_k$ and become side robots with light Blue.

**Proof.** The robot $r_i$ with light Yellow does not collide with other internal robots while it tries to move toward the edge $ef$ of $H_k$. The only problem for $r_i$ is that it may collide with some robots that are already on $ef$. Since corner robots $e, f$ detect that situation and move outside, eventually there is a configuration such that no robot of $ef$ collide with $r_i$ if $r_i$ moves to its computed point in $ef$. Since the moves are RIGID, $r_i$ reaches $ef$ once it moves and changes its color to Blue. $\square$

**Lemma 3.4.4** Given any initial configuration $C_0$, $\exists k \in \mathbb{N}^+$ such that $I_k = \emptyset$ in $C_k$ and corner robots do not move in any round $k' > k$.

**Proof.** We have that when $I_k \neq \emptyset$ each robot on the boundary of $H_k$ (both side and corners) see at least a robot with light $\{\text{Off}, \text{Yellow}\}$. Therefore, until $I_k \neq \emptyset$, the robots on the boundary of $H_k$ move outside to make space on the boundary of $H_k$ for the interior robots to move to. Therefore, combining the results of Lemmas 3.4.1–3.4.3 with this observation, we have that, given any $C_0$, there is $I_k = \emptyset$ in some round $k \in \mathbb{N}^+$. Corner robots do not move after $I_k = \emptyset$ since they do not see robots with light $\{\text{Off}, \text{Yellow}\}$ and the SD phase can be done avoiding collisions without moving the corner robots. This is because robots on each edge move perpendicular to that edge and hence space will not be a problem. $\square$

**Lemma 3.4.5** Given any initial configuration $C_0$, no collisions of robots occur until $I_k = \emptyset$ is reached at some round $k \in \mathbb{N}^+$ in $C_k$.

**Proof.** This lemma is proved by looking at the working principle of Algorithm 2. The interior robot $r_i$ with light Off does not collide with any other internal robot since the move of $r_i$ that
is perpendicular to the closest edge $\overline{ef}$ avoids collisions with other robots. The move of $r_i$ after colored $Yellow$ also does not have collision with other robots since it makes sure that it does not collide with any other robot before moving to $\overline{ef}$. The robots moving to different edges of $H_k$ do not collide since those robots are the closest robots to those edges.

**Theorem 3.4.6** Given any initial configuration $C_0$, there is some round $k \in \mathbb{N}^+$ in which the robots in $\mathcal{R}$ occupy different positions of $H_k$, and the corner robots have light Red and the side robots have light Blue.

**Proof.** The robots in the interior of $H_k$ do not move to the edges of $H_k$ until all the robots in $H_k$ have light $\in \{Blue, Red\}$. Moreover, the internal robots that moved to the edges of $H_k$ change their lights to $Blue$ as soon as they reach there. Furthermore, the internal robots never move to the endpoints of the edges of $H_k$, and the points that they move to in any edge of $H_k$ are always the points that are between two endpoint robots. Moreover, the points they move to in any edge is not occupied by any other robot. Therefore, the robots occupy different positions of $H_k$.

The ID phase finishes after the configuration of Theorem 3.4.6. In other words, the ID phase finishes when there is no robot with light $\in \{Off, Yellow\}$ in $H_k$, $k \in \mathbb{N}^+$. Actually, we can prove the following two lemmas which collectively show that the SD phase does not start when there are still robots with light $\in \{Off, Yellow\}$ in the system.

**Lemma 3.4.7** Given a configuration $C_k$ and an edge $e = v_1v_2$ of $H_k$, no robot in any edge $e$ of $H_k$ moves to $S(e)$ if there is an internal robot with light $\in \{Off, Yellow\}$.

**Proof.** If there are internal robots in $H_k$, it is easy to see that every side and corner robot of $H_k$ sees at least one internal robot. Otherwise, there must not be any robot in the interior of $H_k$. Therefore, when there are internal robots with light $\in \{Off, Yellow\}$, side robots can easily infer that the ID phase is still not finished and hence they do not move to their respective $S(e)$.

**Lemma 3.4.8** Given a robot $r_i \in \mathcal{R}$ with light $\in \{Blue, Red\}$ and a round $k \in \mathbb{N}^+$, if all the robots in $C_k(r_i)$ have light $\in \{Blue, Red\}$ and no robot is in the interior of $H_k(r_i)$, then $C_k$ does not contain internal robots with respect to $H_0$.
**Proof.** When all the robots in $C_k(r_i)$ have light $\in \{Blue, Red\}$, this means that there is no robot has light $\in \{Off, Yellow\}$. This immediately proves that there is no internal robot with respect to $H_0$ since the internal robot should have colored Off or Yellow and that would have been seen by $r_i$. 

Denote by $C_{ID}$ the configuration $C_k$ of robots at round $k \in \mathbb{N}^+$ after the ID phase of our algorithm is finished such that $I_k = \emptyset$ and by $H_{ID}$ the convex hull created by $C_{ID}$. We have the following lemma.

**Lemma 3.4.9** Given a configuration $C_{ID}$ and an edge $e = v_1v_2$ of $H_{ID}$, if the robot $r_i \in e$ moves from $e$, it moves inside the safe zone $S(e)$.

**Proof.** We prove this lemma using the proof technique similar as of [33, Lemma 3]. This lemma is immediate when there is a single robot $r \in e$ because $r$ can compute exact $S(e)$. Consider now the situation when there are at least two side robots on $e$. Let $r_1$ and $r_2$ be the two robots on $e$ that are neighbors of $v_1$ and $v_2$ (the corners), respectively. Due to the assumption of FSYNCH robots, both $r_1$ and $r_2$ move from $e$ to $S(e)$ in a round. Consider only the move of $r_2$ to $S(e)$ (the move of $r_1$ follows similarly). Robot $r_2$ has two neighbors on $e$: A side robot $r$ (a neighbor of $r_2$ in the side of $r_1$) and a corner $v_2$. Robot $r_2$ orders the robots in its view according to its local notion of clockwise direction. Following the rules of our algorithm, $r_2$ computes $\alpha = 180 - \angle v'v_1r_2$, $\beta = 180 - \angle r_2rv_3$, and $\delta = \min\{\alpha/4, \beta/4\}$. Since we calculate $\alpha$ by subtracting $\angle v'v_1r_2$ from 180, $\alpha$ is in fact a lower bound on the angle that any robot on $e$ will compute. Therefore, since any point $x$ in the safe area computed by $r_2$, even under the obstruction of $v_2$ from other robots on $e$, is inside the safe area of $e$. Therefore, $r_2$ will move inside $S(e)$. The same will hold for $r_1$ and all other robots on $e$. 

**Lemma 3.4.10** Let $r_i$ and $r_j$ be the robots that are neighbors of endpoints $v_1$ and $v_2$ on edge $e$, respectively. When there are $1 \leq \rho \leq 2$ side robots on $e$, $r_i$ and $r_j$ become corners and change their light to Red in the next round. When there are $\rho > 2$ side robots on $e$, $r_i$ and $r_j$ become
corners and change their light to Red after all the robots on e moved outside once on the line connecting $r_i$ and $r_j$.

**Proof.** Since we assume the FSYNCH setting, both $r_i$ and $r_j$ move to $S(e)$ in a single round. If $\rho \leq 2$, $r_i$ and $r_j$ do not see any other robot on $e$. They can simply change their color to Red since when they move to $S(e)$ at some round $k$ using the computation $\delta = \min\{\alpha/4, \beta/4\}$ (Lemma 3.4.9), we have from [33] that both $r_i$ and $r_j$ become corners of $H_k$ after they move once. When $\rho > 2$, since $r_i$ and $r_j$ are corners, they see all the robots on $e$ and can wait until all robots on $e$ move to a line connecting them. After all remaining robots on $e$ moved to that line, $r_i$ and $r_j$ do not see any robot on $e$ and can simply change their light to Red. \(\square\)

**Lemma 3.4.11** The robots that become corners in the SD phase at round $k$ do not move in any future round $k' > k$.

**Proof.** This is because the newly become corner robots do not see any robot with light Off or Yellow, and the side robots in $H_k$ can still be moved in the area between two corners without collisions and without any space problem since they move perpendicularly to the edge they belong to when they go to their $S(e)$. \(\square\)

**Lemma 3.4.12** Given a configuration $C_{1D}$, let $e = \overline{v_1v_2}$ be an edge with $q \geq 1$ side robots. Eventually all these robots will become corner robots and set their lights to Red.

**Proof.** Assume for simplicity that all the side robots of $H_{1D}$ are on a single edge $e$; we later discuss how the proof can be extended when there are side robots on many edges of $H_{1D}$. When there is only one robot $r$ on $e$ (i.e. $q = 1$, and there is no other side robot in $H_{1D}$), it can compute exact $S(e)$ and move to a point $x \in S(e)$ as soon as it is activated. The robot $r$ then simply change its color to Red and we have the lemma. When there are two robots, say $r_1$ and $r_2$ on $e$, both of them move to $S(e)$, become corners, and do not move in any future rounds. When there are more than two robots on $e$, i.e. $q > 2$, two robots become corners every time all the robots on $e$ move to $S(e)$. Therefore, all robots on $e$ eventually become corner robots and set their lights to Red. Now
when there are side robots on many edges of $H_{1D}$, the robots synchronize in such a way that they
move to $S(e)$ only after all the robots are in corners and sides of $\mathbb{H}_k$. Therefore, all side robots
eventually become corner robots and terminate setting their lights to Red.

Finally, we have the following theorem from the analysis of our algorithm for the FSYNCH setting.

**Theorem 3.4.13** MUTUAL VISIBILITY is always solvable in finite time without collisions for fat
robots with 4 colors in the FSYNCH setting in the robots with lights model.

**Proof.** We have from Theorem 3.4.6 that from any initial non-collinear configuration $C_0$, we
reach a configuration $C_{ID}$ where $I_{ID} = \emptyset$ in finite time. We have from Lemma 3.4.5 that there is
no collisions of robots while executing the ID phase. We have from Lemma 3.4.8 that robots can
locally detect whether the configuration $C_{ID}$ is reached and start executing the SD phase. We have
from Lemma 3.4.12 that side robots on each edge $e$ of $H_{ID}$ eventually become corners. Moreover,
it is immediate that no collision of robots occur in the SD phase as well. Therefore, starting from
any non-collinear configuration $C_0$, all robots eventually become corners of a convex hull and can
not obstruct each other, solving MUTUAL VISIBILITY without collisions.

It only remains to show that, starting from any initial collinear configuration $C_0$, the robots
correctly evolve into some non-collinear configuration from which applying the analysis above,
the robots become corners of a convex hull and terminate. If $n \leq 3$, we can immediately prove
that robots become corners and terminate through a case analysis. For $n = 1$, when the only
robot becomes active it sees no other robot, that is, it changes its color to Red and immediately
terminates. For $n = 2$, the robot changes its color to Red when it becomes active for the very first
time and moves orthogonal to line $r_ir_j$ that connects it to the only other robot $r_j$ it sees in $C(r_i)$.
When $r_i$ realizes later that $|C(r_i)|$ is still 2 and $r_j$.light = Red, it simply terminates. For $n = 3$,
if $r_i$ realizes that both of its neighbors in $C(r_i)$ have light set to Red and are collinear with it, it
moves orthogonal to that line and sets its light to Red. When it becomes active next time, it finds
itself at one of the corners and simply terminates as it sees all the other robots in the corners of
the hull with light set to \textit{Red}. For \( n \geq 4 \), let \( a \) and \( b \) be two robots that occupy the corners of the line segment \( H_0 \) (i.e., the endpoint robots of \( H_0 \)). Nothing happens until \( a \) or \( b \) is activated, setting its light to \textit{Red}, and moving orthogonal to \( H_0 \). After \( a \) or \( b \) moved, the other robots in \( H_0 \), when become active, realize that they are not in a line anymore and enter the normal execution of our algorithm. It is also easy to see that after the line segment \( H_0 \) evolves in a polygonal shape, it does not become a line segment anymore. Note that in the whole process only 4 colors are used in the color set \( C \).

\[ \square \]

### 3.5 Analysis for the SSYNCH Setting

Our lemmas and theorems for the ID phase hold also in the SSYNCH setting of fat robots without any change. This is because the robots that are activated in a round perform their Look-Compute-Move cycles in a perfect synchronization. For the SD phase, however, the \textit{Blue} colored robots on \( v_1v_2 \), say \( r_i \) and \( r_j \), that are neighbors of endpoints \( v_1 \) and \( v_2 \) may not be able to move to \( S(e) \) at the same round. But, even in this case, the remaining \textit{Blue} robots on \( v_1v_2 \) detect that situation and wait until both \( r_i \) and \( r_j \) reach to their \( S(e) \). Fig. 3.14 serves as an example for this situation such the robot \( r \) waits until \( r_i \) moves to \( S(e) \) since \( r \) sees robot \( r_j \) in \( S(e) \) outside the line \( L \) perpendicular to \( v_1v_2 \) towards the endpoint \( v_1 \). Therefore, we have the following theorem.

\textbf{Theorem 3.5.1} **Mutual Visibility** is always solvable in finite time without collisions for fat robots with 4 colors in the SSYNCH setting in the robots with lights model.
CHAPTER 4

A Faster Solution for the Mutual Visibility Problem

4.1 Outline

In this chapter, we present another algorithm that solves the Mutual Visibility Problem for fat robots without collisions and without the assumption of chirality for FSYNCH setting. We solve this problem by using the light model, so the number of robots is not assumed to be known for each robot, and this model needs a color set with 5 colors \[ C = \{\text{Off, Yellow, Purple, Blue, Red}\}, \] and we solve this algorithm in finite time which is linear time \( O(n) \). Therefore, in this chapter, we present a faster algorithm to solve the Mutual Visibility Problem, and we proceed as follows. We present our algorithm that solves the Mutual Visibility Problem in \( O(n) \) time for all of its cases and phases: interior depletion phase, side depletion phase, and special cases. After that, we present the analysis of the algorithm for FSYNCH setting. We do not present the model and preliminaries since it is the same as the model and preliminaries for the previous algorithm, and it is presented in Chapter 3. We only outline the differences in this chapter.

4.2 The Mutual Visibility Algorithm

In this section, we present a linear time runtime algorithm (Algorithm 6) for the Mutual Visibility problem. The algorithm works in the fully synchronous setting. The robot model is the same as in Chapter 3. The analysis of the runtime is given in Section 4.3. Algorithm 6 uses
Algorithms 7–9 as subroutines and needs 5 colors in the color set $C = \{\text{Off, Yellow, Purple, Blue, Red}\}$. Note that our algorithm in Chapter 3 needs only 4 colors. The use of an extra color helps us to prove the runtime of $\mathcal{O}(n)$ rounds. Algorithm 6 works similarly as of Algorithm 1 and has two phases (i) interior depletion (ID) and (ii) side depletion (SD).

**Algorithm 6:** MUTUAL VISIBILITY algorithm for robot $r_i$ of unit disc size for any round $k > 0$

1. // Look-Compute-Move cycle for robot $r_i$ of unit disc size
2. $C_k(r_i) \leftarrow$ configuration $C_k$ for robot $r_i$ (including $r_i$);
3. $H_k(r_i) \leftarrow$ convex hull of the positions of the robots in $C_k(r_i)$;
4. if $|C_k(r_i)| = 1$ then Terminate;
5. else if $H_k(r_i)$ is a line segment then
6. if $|C_k(r_i)| = 2$ then
7. Let $r_j \in C_k(r_i)$;
8. if $r_i$.light = Off then
9. $r_i$.light $\leftarrow$ Red;
10. Move orthogonal to line $\overrightarrow{r_j r_i}$ by any non-zero distance;
11. else if $r_j$.light = Red then Terminate;
12. else if $|C_k(r_i)| = 3$ then
13. Let $r_j, r_l \in C_k(r_i)$;
14. if $r_i$.light = Off $\land$ $r_j$.light = Red $\land$ $r_l$.light = Red then
15. $r_i$.light $\leftarrow$ Red;
16. Move orthogonal to line $\overrightarrow{r_j r_l}$ by any non-zero distance;
17. else if $r_i$ is a vertex robot of $H_k(r_i)$ then Corner($r_i, C_k(r_i), H_k(r_i)$);
18. else if $r_i$ is an interior robot of $H_k(r_i)$ then Internal($r_i, C_k(r_i), H_k(r_i)$);
19. else if $r_i$ is a side robot of $H_k(r_i)$ then Side($r_i, C_k(r_i), H_k(r_i)$);

4.2.1 Interior Depletion

The interior depletion is again done using Algorithms 7 and 8. Starting from $C_0$, the corner robots change their color to Red, and the side robots change their color to Blue similarly as in Algorithm 1. Since the lights of all robots in $R$ is set of Off in $C_0$, except the corner and side robots, other robots have light Off when the corner and side robots change their colors to Red and Blue, respectively.

The move of the internal robot $r_i$ with light Off to become robot with light Yellow is also similar to Algorithm 1. Let $\overline{ef}$ be an edge of $H$ in $Q$ that is the closest to $r_i$ compared to the other edges of $Q$. Let $L$ be a line perpendicular to $\overline{ef}$ passing through $r_i$. In Algorithm 1, $r_i$ moves to become Yellow colored robot when the length of $L$ from $r_i$ to the intersection point of $L$ and $\overline{ef}$ is at least
Algorithm 7: \textit{Internal}(r_i, \mathcal{C}_k(r_i), \mathcal{H}_k(r_i))

1. if \( r_i \).light = Red then
2. \( e \leftarrow \) edge in \( \mathcal{H}_k(r_i) \) that is closest to \( r_i \);
3. \( m \leftarrow \) intersection point of \( e \) and a line \( L \) that is perpendicular to \( e \) and passes through \( r_i \);
4. Move to the point in \( L \) distance 1 away from point \( m \);
5. else if \( r_i \).light = Blue then
6. \( e \leftarrow \) edge in \( \mathcal{H}_k(r_i) \) that is closest to \( r_i \);
7. Move to the intersection point of \( e \) and a line \( L \) that is perpendicular to \( e \) and passes through \( r_i \);
8. else
9. Let \( H_{P_{\text{blue}}} \) be the half-plane such that there are only robots with lights Blue in it;
10. \( L \leftarrow \) line perpendicular to the edge robot \( r_i \) belongs to in the convex hull in the other side of \( H_{P_{\text{blue}}} \) that passes through \( r_i \);
11. \( L' \leftarrow \) the line formed by the robots in \( H_{P_{\text{blue}}} \);
12. Move to the intersection point of \( L \) and \( L' \);
13. else
14. Order the robots in \( \mathcal{H}_k(r_i) \) starting from any arbitrary robot \( v_1 \) in the clockwise order so that \( T = \{ v_1, \ldots, v_{\text{last}}, v_1 \} \), where \( v_1 \) is the first robot and \( v_{\text{last}} \) is the last robot;
15. Let \( c, d \) be any pair of two consecutive robots in \( T \) with \( c \).light = Red and \( d \).light = Red;
16. Let \( H_{P_{cd}} \) be the half-plane divided by line parallel to \( \overline{cd} \) that passes through \( r_i \) such that \( c, d \) are in \( H_{P_{cd}} \);
17. \( Q \leftarrow \) set of line segments \( \overline{cd} \) such that:
18. (a) the triangle \( r_i, c, d \) does not contain neither inside and neither on its edges any other robot of \( \mathcal{C}_k(r_i) \) except the robots in edge \( \overline{cd} \), and
19. (b) either there is no robot in edge \( \overline{cd} \) or all the robots in it have light Blue, and
20. (c) there is no robot in \( \mathcal{C}_k(r_i) \)\((\overline{cd} \) closer to edge \( \overline{cd} \) than \( r_i \), and
21. (d) there are no two robots with equal distance to \( \overline{cd} \) appearing to both the counterclockwise and clockwise direction of \( r_i \) with respect to the local coordinate system of \( r_i \) (however, there might be robots in either the counterclockwise or the clockwise direction of \( r_i \));
22. if \( Q \) is not empty then
23. \( f \leftarrow \) the line segment in \( Q \) between two robots \( c, f \) that is closest to \( r_i \);
24. if \( r_i \).light = Yellow then
25. if there is no robot with light Blue in edge \( \overline{cd} \) then
26. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through its midpoint \( m \);
27. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through \( m \);
28. \( g, h \leftarrow \) robots closer to endpoints \( c \) and \( f \), respectively, in edge \( \overline{cd} \);
29. if there is no other robot with light Yellow that is at equal distance to \( \overline{cd} \) then
30. if \( \text{length}(\overline{cd}) \geq 5 \) or \( \text{length}(\overline{cd}) \geq 5 \) then
31. \( m \leftarrow \) point in \( \overline{cd} \) at \( \frac{\text{length}(\overline{cd}) - 1}{2} \) from endpoint \( e \);
32. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through \( m \);
33. \( g, h \leftarrow \) robots closer to endpoints \( e \) and \( f \), respectively, in edge \( \overline{cd} \);
34. if there exists a robot in the clockwise direction of \( r_i \) (with respect to the local coordinate system of \( r_i \)) with light Yellow that is at equal distance to \( \overline{cd} \) then
35. \( m \leftarrow \) point in \( \overline{cd} \) at \( \frac{\text{length}(\overline{cd}) - 1}{2} \) from endpoint \( f \);
36. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through \( m \);
37. \( g, h \leftarrow \) robots closer to endpoints \( e \) and \( f \), respectively, in edge \( \overline{cd} \);
38. if there exists a robot in the clockwise direction of \( r_i \) (with respect to the local coordinate system of \( r_i \)) with light Yellow that is at equal distance to \( \overline{cd} \) then
39. if \( \text{length}(\overline{cd}) \geq 5 \) then
40. \( m \leftarrow \) point in \( \overline{cd} \) at distance 1 from \( g \);
41. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through \( m \);
42. if there exists a robot in the clockwise direction of \( r_i \) (with respect to the local coordinate system of \( r_i \)) with light Yellow that is at equal distance to \( \overline{cd} \) then
43. if \( \text{length}(\overline{cd}) \geq 5 \) then
44. \( m \leftarrow \) point in \( \overline{cd} \) at distance 1 from \( h \);
45. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through \( m \);
46. else if \( r_i \).light = Yellow then
47. \( L'' \leftarrow \) line perpendicular to \( \overline{cd} \) passing through \( r_i \);
48. Move to the intersection point of \( \overline{cd} \) and \( L'' \) and set \( r_i \).light = Blue;
49. else if \( r_i \).light = Off then
50. \( m \leftarrow \) intersection point of \( \overline{cd} \) and a line \( L \) perpendicular to \( \overline{cd} \) that passes through \( r_i \);
51. if \( \text{length}(\overline{cd}) \geq 3 \) then Move distance 3 to the point in \( L \) towards \( \overline{cd} \) and set \( r_i \).light = Yellow;
Algorithm 8: \textit{Corner}(r_i, \mathbb{C}_k(r_i), \mathbb{H}_k(r_i))

1. \textbf{if} \(r_i\.light = \text{Off} \) then \(r_i\.light \leftarrow \text{Red};\)
2. \textbf{else if} \(\forall r \in \mathbb{C}_k(r_i), r\.light = \text{Red} \) then \textbf{Terminate};
3. \textbf{else if} \(r_i\.light = \text{Blue} \) then
4. \hspace{1em} \(L \leftarrow \text{edge of } \mathbb{H}_k(r_i) \text{ closest to } r_i \text{ with } x\.light = y\.light = \text{Red};\)
5. \hspace{1em} \textbf{if} \text{there is no robot in edge } L \text{ then } r_i\.light \leftarrow \text{Red};
6. \textbf{else if} \(\forall r \in \mathbb{I}_k(r_i), r\.light \in \{\text{Off}, \text{Yellow}, \text{Purple}\} \) then
7. \hspace{1em} \(a \leftarrow \text{counterclockwise neighbor in the boundary of } \mathbb{H}_k(r_i);\)
8. \hspace{1em} \(b \leftarrow \text{clockwise neighbor in the boundary of } \mathbb{H}_k(r_i);\)
9. \hspace{1em} \(r_j, r_k \leftarrow \text{robots in the interior of } \mathbb{H}_k(r_i) \text{ that are closest to edges } r_i\overline{a} \text{ and } r_i\overline{b}, \text{respectively (if there are more than one such robots, take as } r_j, r_k \text{ the robots that are closer to } r_i);\)
10. \hspace{1em} \textbf{if} \(r_j\.light = \text{Off} \text{ or } r_k\.light = \text{Off} \) then
11. \hspace{2em} \(m \leftarrow \text{intersection point of } r_i\overline{a} \text{ and line } L \text{ perpendicular to } r_i\overline{a} \text{ that passes through } r_j;\)
12. \hspace{2em} \(m' \leftarrow \text{intersection point of } r_i\overline{b} \text{ and line } L \text{ perpendicular to } r_i\overline{b} \text{ that passes through } r_k;\)
13. \hspace{2em} \textbf{if} \(\text{length}(r_j\overline{m}) < 3 \text{ or length}(r_k\overline{m'}) < 3 \) then
14. \hspace{3em} \text{Choose one between } r_j, r_k \text{ (say } r_j) \text{ such that the length of } r_jm \text{ or } r_km' \text{ is smaller;}
15. \hspace{3em} \(m'' \leftarrow \text{point in } L \text{ outside } r_i\overline{a} \text{ such that length}(r_j\overline{m}) = 3;\)
16. \hspace{3em} \(L \leftarrow \text{line parallel to } r_i\overline{a} \text{ passing through } m'';\)
17. \hspace{3em} \(L' \leftarrow \text{line that is angle bisector of } \angle a\overline{r_i}b;\)
18. \hspace{3em} \text{Move to the intersection point of } L \text{ and } L';
19. \textbf{else if} \(r_j\.light = \text{Yellow} \text{ or } r_k\.light = \text{Yellow} \) then
20. \hspace{3em} \textbf{if} \(\text{length}(r_i\overline{a}) < 5 \text{ or length}(r_i\overline{b}) < 5 \) then
21. \hspace{4em} \text{Choose between } r_i\overline{a} \text{ or } r_i\overline{b} \text{ that is smaller (say } r_i\overline{a});
22. \hspace{4em} \(m'' \leftarrow \text{point outside } r_i\overline{a} \text{ in line } L \text{ perpendicular to } r_i\overline{a} \text{ going through } a;\)
23. \hspace{4em} \(L' \leftarrow \text{line that is angle bisector of } \angle a\overline{r_i}b;\)
24. \hspace{4em} \text{Move to the point } m \text{ in } L' \text{ such that the distance of the line perpendicular to } L \text{ going through } m'' \text{ is } 5;\)

Algorithm 9: \textit{Side}(r_i, \mathbb{C}_k(r_i), \mathbb{H}_k(r_i))

1. \textbf{if} \(r_i\.light = \text{Off} \) then \(r_i\.light \leftarrow \text{Blue};\)
2. \textbf{else if} \(\forall r \in \mathbb{C}_k(r_i), r\.light \in \{\text{Blue, Red}\} \land \text{no robot } r \in \mathbb{C}_k(r_i) \text{ is in the interior of } \mathbb{H}_k(r_i) \) then
3. \hspace{1em} \textbf{if} \text{at least one neighbor of } r_i \text{ in the edge it belongs to has light Red} \text{ then}
4. \hspace{2em} \text{Order the robots in the counterclockwise order of } r_i \text{ (with respect to the local coordinate system of } r_i) \text{ such that the order is } T_i = \{v_3, v_2, r_i, v_0\}, \text{ where } v_3 \text{ is the first robot non-collinear to } r_i \text{ in the clockwise direction of } r_i \text{ with } v_3\.light = \text{Red}, v_2 \text{ is the robot that is collinear with } r_i \text{ in the clockwise direction of } r_i \text{ with } v_2\.light \in \{\text{Blue, Red}\}, \text{ and } r \text{ is the collinear robot in the counterclockwise direction of } r_i \text{ with } r\.light \in \{\text{Blue, Red}\} \text{ and } v_0 \text{ is the first non-collinear robot to } r_i \text{ in the counterclockwise direction of } r_i \text{ with } v_0\.light = \text{Red};\)
5. \hspace{2em} \text{Compute angles } \alpha = 180 - \angle v_0r_i r \text{ and } \beta = 180 - \angle r_1v_2v_3, \text{ and set } \delta = \min\{\alpha/4, \beta/4\};
6. \hspace{2em} \text{Compute a point } x' \text{ such that } \angle x'v_2r_i = \delta \text{ and a point } x'' \text{ such that } \angle x''r_i r = \delta;
7. \hspace{2em} \(x \leftarrow x' \text{ or } x'' \text{ whichever is nearest to } e;\)
8. \hspace{2em} \text{Move perpendicular to } e \text{ with destination } x;\)
2. However, in Algorithm 6, we ask that distance to be at least 3. This is to help avoid collisions in the later stages of the algorithm.

In Algorithm 1 after $r_i$ becomes a Yellow colored robot, it waits until it can move to position itself on $\overline{ef}$ without colliding with other robots. Note that $r_i$ may need to wait for several rounds to become eligible to move to $\overline{ef}$. This is because the corner robots $e, f$ even after detecting the situation that $r_i$ can not move to $\overline{ef}$, they simply move distance 1 outside at all times. The one main ingredient in the new algorithm is that $e, f$ move out one time to make $r_i$ eligible to move to $\overline{ef}$ in the next round. To accomplish this, first $r_i$ does not directly tries to move to $\overline{ef}$ after colored Yellow. Instead, it makes another move from which it can immediately move to $\overline{ef}$ in the next round. To identify this move, $r_i$ changes its color to Purple from Yellow.

Specifically, the move is done as follows. The robot $r_i$ after becoming Yellow, it checks whether there is at least an edge robot on $\overline{ef}$ with color Blue. If there is no Blue robot on $\overline{ef}$, $r_i$ checks whether there is other Yellow colored robot that is at equidistant to $\overline{ef}$. If this is not the case, $r_i$ draws a line $L'$ perpendicular to $\overline{ef}$ passing through its midpoint. It then computes a point $m'$ on $L'$ at distance 1 from $\overline{ef}$ towards inside of the hull and moves to $m'$ changing color to Purple. If there is another equidistant Yellow robot, it considers point $(\text{length}(\overline{ef}) - 1)/2$ from $e$ or $(\text{length}(\overline{ef}) - 1)/2$ from $f$ depending on in which side the other Yellow colored robot is. It then draws a line $L'$ perpendicular to $\overline{ef}$ going through point $(\text{length}(\overline{ef}) - 1)/2$ from $e$ or $(\text{length}(\overline{ef}) - 1)/2$ from $f$. Robot $r_i$ then computes point $m'$ on the line $L'$ as described in the first case and moves to that point. In the case where there are Blue robots, let $g, h$ be the Blue colored robots on $\overline{ef}$ such that $g$ is the closest to $e$, and $h$ is the closest to $f$, respectively. Robot $r_i$ then makes sure that the length of $\overline{eg}$ or $\overline{hf}$ that it can move to is at least 5. As soon as the criteria satisfies, it computes a point in $\overline{eg}$ or $\overline{hf}$ at distance 1 from $g$ or $h$. It then draws a line $L'$ perpendicular to $\overline{ef}$ passing through that point and moves to point $m'$ on line $L'$ as described above.

Note that in the case where length of $\overline{eg}$ or $\overline{hf}$ is less than 5, $r_i$ does not make a move to $m'$ to become Purple colored robot to avoid possible collisions. In this case, the corner robots $e, f$
Figure 4.1: An illustration of how internal robots select edges of $\mathbb{H}_k$ to include in their $Q$’s. The robot $r_i$ includes $ef$ and $ev_1$ in its $Q$. Robots, $r_j$ has $ef$, and $r_l$ has $fv_2$ in their $Q$’s. However, the robots between $r_i$ and $r_j$ have empty $Q$.

Figure 4.2: The corner robots, $e$ and $f$ move to a direction away from the convex hull if the distance between the closest internal robot and the edge that is the closest to this robot in $\mathbb{H}_k$ is $< 3$. It draws a line which bisects the angle $\angle aeb$ and moves some unit distance on that line away from the convex hull.

Figure 4.3: An illustration of how an internal robot $r_i$ chooses a point to move to in the edge $ef \in Q$ that is the closest to when it is the only closest robot to $ef$. Since there are side robots on $ef$, $r_i$ chooses either $m$ or $m'$ as the point in $ef$ to try to move to. If there were no side robots on $ef$, $r_i$ would have chosen the midpoint of $ef$ as the point to try to move to.

We need to detect that situation of the length of $eg$ or $hf$ is less than 5 and move outside of the hull appropriately so that the length of $eg$ and $hf$ becomes at least 5. We argue here that the nodes $e$, $f$ in the fact can detect this situation and move outside appropriately. This is done as follows. Let consider $e$ (the case for $f$ is analogous). Let $a, b$ be $e$’s neighbors in the hull. Since $ea$ or $eb$ is
Figure 4.4: An illustration of how an internal robot $r_i$ chooses a point to move to in the edge $ef \in Q$ that is the closest to when there are other robots that are equidistant to $ef$. Since there are side robots on $ef$, and $r_i$ sees equidistant robots in only one direction, $r_i$ chooses $m_i$ as the point in $ef$ to try to move to. Robots can do this without agreeing on the coordinate system of other robots. Since $r_j$ also sees equidistant robots in only one direction, it chooses point $m_j$ to try to move to. Note that the robots that are between $r_i$ and $r_j$ do not try to move to.

less than 5, $e$ chooses either $a$ or $b$ based on the shorter length between $ea$ or $eb$ (say $ea$). Robot $e$ then draws a line perpendicular to $ea$ going through $a$. It then computes another line $L'$ that is angle bisector of $\angle aeb$. Now $e$ computes a line perpendicular to $L$ such that the distance from the intersection point in $L$ to the intersection point in $L'$ is 5. Robot $e$ then moves to that intersection point in $L'$.

The side robots, if any, that were originally on $ef$ (before $e, f$ move outside) also move to $ef$ before any internal robot with light Purple moves to $ef$. This is done by drawing a line perpendicular to $ef$ and moving to the intersection point of $ef$ and the line perpendicular to $ef$ going through the position of that side robot (which now became internal due to the moves of $e$ and/or $f$ outside of the hull).

Note also that the moves of corners $e, f$ outside the hull may trigger the moves of other corner robots of $H_k$, which otherwise would not move. This is because some corner robots may become internal due to the moves of its neighboring corner robots outside of $H$. The corner robot $r_j$ that becomes internal handles this case as follows. It finds the closest edge $\hat{e}$ in $H_k(r_j)$ (with two corner robots have light Red) and draws a line $L$ perpendicular to that edge going through it. Let $m$ be the intersection point of $L$ and $\hat{e}$. Robot $r_j$ then moves distance 1 away from $m$ on the line $L$ outside the hull without changing its color. Note that we used 5 colors here compared to 4 colors in Chapter 3.
Figure 4.5: An illustration of the corner robots, $e$ and $f$ move if the distance between the corner robot and the side robot on the edge $ef$ is $< 5$, and this is done by choosing a point outside $eb$ as an example on line $L$ perpendicular to $eb$ going through $b$, and line $L'$ that is angle bisector of $\angle aeb$, then move to a point in $L'$ outside the hull such that the distance of the line perpendicular to $L$ going through the point is 5.

Figure 4.6: An internal robot $r_i$ with light Yellow moves to a point that is at distance one from $m$ inside the convex hull and changes its color to Purple.

Figure 4.7: An internal robot $r_i$ with light Purple moves to the point $m$ on the edge $ef$ to be a side robot and changes its color to Blue.

### 4.2.2 Side Depletion

This phase starts only after the ID phase finishes, and it is similar to the side depletion phase of Algorithm[1] in Chapter[3] This phase does not introduce new colors.
Figure 4.8: An illustration of how the SD phase works. The Blue robots with one neighbor Red in \( e = v_1v_2 \) move to a point \( x \) inside the safe zone \( S(e) \) computed using the technique described in Section 3.2 without changing their color.

Figure 4.9: Robots \( r_i \) and \( r_j \) that moved to their local \( S(e) \) in Fig. 4.8 change their light to Red only after all robots on \( e \) moved to \( r_i r_j \). Robots \( r_i, r_j \) do not move in future rounds.

### 4.2.3 Special Cases

There are two special cases in our algorithm which are again tackled similar to Chapter 3.

### 4.3 Analysis of the Algorithm

We prove the following lemmas and theorems which guarantee that given any initial configuration \( C_0 \) of a system of \( n \geq 1 \) fat robots; the robots in \( I_0 \) successfully deplete to the convex hull boundary in \( O(n) \) rounds and then become corners in another \( O(n) \) rounds, solving the Mutual Visibility problem. Note that we consider the FSYNCH setting.

The correctness of the algorithm follows similar to the correctness analysis in Chapter 3. If there is no Yellow, Blue, Purple and Red colored robots in the interior of \( \mathbb{H} \), then at least one robot with light Off becomes Yellow colored robots. We prove later that this happens in constant number
of rounds. The Yellow colored robot again becomes Purple colored robot in constant number of rounds. After a robot becomes Purple colored, it then can become Blue robot in the next round.

There are no collisions as proved in Chapter 3.

Therefore, we prove the following theorem.

**Theorem 4.3.1** Within a constant number of rounds, an internal robot in $C_0$ becomes a side robot and changes its color to Blue.

**Proof.** Since all robots perform their LCM cycles in each round in the FSYNCH setting, even if the initial configuration is a line segment, it changes to a polygonal configuration after one round. In the next round, all the corner and side robots of H color themselves Blue and Red. After that either one internal robot with light Off becomes a Yellow colored robot, or some corner robots move outside to make the distance from an Off colored robot to its nearest edge at least 3. After that in the next round, the blue colored robots if any can move to become side robots. Therefore, in at most 3 rounds, at least an internal robot with color Off becomes Yellow colored robot. After the robot becomes Yellow colored, we now show that it becomes Purple colored robot in next at most 3 rounds. First of all, if the length of the line segments $ef$ and $hf$ is 5, then the Yellow colored robots becomes Purple colored robot in next round by computing a point on the edge and drawing a line perpendicular to the edge passing through the point, then the robot moves to a point that is distance 1 away from the computed point inside the hull and changes the color to Purple. If length of either of the line segment is less than 5, their lengths become at least 5 in the next round since the corners $e, f$ move outside detecting that situation. It is immediate that when $e$ and/or $f$ move outside once, their lengths become at least 5. If there are side robots on $ef$ before moving outside, they now become side robots in the next round without violating the lengths of $ef$ and $hf$ being 5. After that the Yellow colored robot can become Purple colored robot in the next round. After a robot becomes Purple, it can immediately move to its computed point on the side and become Blue colored robot in the next round. Therefore, in $O(1)$ rounds, at least a robot with light Off becomes Blue colored robot. □
We now immediately have this corollary.

**Corollary 4.3.2** The interior depletion phase of Algorithm 6 finishes in $O(n)$ rounds. Moreover, at the end of the phase, the side robots are colored Blue, and the corner robots are colored Red.

We have from Chapter 3 that the side depletion phase starts only after the internal depletion phase finishes. This applies also in this algorithm since if there are robots with color Off, Yellow, or Purple, the side and corner robots must see at least one robot with those colors. If the side and corner robots see robots with those colors, it means that the internal depletion phase has not finished yet. Otherwise, if they do not see any robot with those colors, it means that the internal depletion phase is finished, and the side depletion phase can be started. We now show that at least one side becomes a corner robot in constant number of rounds in the side depletion phase.

**Theorem 4.3.3** Within a constant number of rounds, a side robot becomes a corner robot and changes its color to Red in the side depletion phase.

**Proof.** Similar to Chapter 3, the moves of the side robots on two different edges of $H$ do not interfere with each other. Therefore, we prove this lemma for an edge, and the same argument applies for the side robots on other edges of $H$. If there is a single side robot on an edge, it immediately becomes a corner robot as soon as it moves to safe area by computing a point in the safe area to move to it to become a corner robot. If there are two or more robots on an edge, two robots move outside in the safe area in each round which are one from the clockwise most and one from the counterclockwise most side robots until there left a single robot on an edge. After these two robots move outside, all the robots on the side can move to the line joining these two robots in the next round. After that, the two endpoint robots on that edge now become new corners by changing their color to Red, and this process repeats. Therefore, in at most every three rounds, at least two robots become corners until there is only one robot left on that edge. 

We now immediately have this corollary, since in the worst-case all the side robots may be on an edge, we need $O(n)$ rounds for this process to be finished.
Corollary 4.3.4  The side depletion phase of Algorithm 6 finishes in $O(n)$ rounds. Moreover, at the end of the phase, all the robots in the system are in the corners and colored Red.

We now have the following theorem on the runtime of Algorithm 6 combining the results of Corollaries 4.3.2 and 4.3.4.

Theorem 4.3.5  Algorithm 6 solves Mutual Visibility for robots with unit disc size in $O(n)$ rounds with 5 colors in the FSYNCH setting in the robots with lights model.
CHAPTER 5

Concluding Remarks and Future Work

5.1 Concluding Remarks

We studied the fundamental MUTUAL VISIBILITY problem – starting from any arbitrary initial configuration, reposition the robots to reach to a configuration within finite time without collisions in which all robots see each other – for a system of autonomous, anonymous, and oblivious robots with extents (so-called fat robots) under obstructed visibility. Ours are the first such solutions for robots with extents; all previous work considered dimensionless point robots which do not reflect reality since physical robots have size (i.e. they are not points). Our objective was to minimize the number of colors and runtime to solve this problem in recently proposed robots with lights model \[36\].

Our first solution uses 4 colors and works for both FSYNCH and SSYNCH settings of robots with unit disc size extents under rigid movements. However, it does not guarantee upper bound on runtime except that the algorithm terminates in finite time. We then present our second algorithm that uses 5 colors, but it provides a runtime upper bound of $O(n)$ rounds in solving the mutual visibility problem in the FSYNCH setting. As a byproduct, our algorithms also solve the CIRCLE FORMATION problem \[14, 17, 19, 22\] – position robots on the perimeter of a circle – even for the robots with extents. We just need a simple modification to the algorithm. The modified algorithm uses 4 colors for the FSYNCH robots and 5 colors for the SSYNCH robots. There is no previous
solution for this problem for robots with extents (fat robots).

5.2 Future Work

We have the following seven directions for possible future work:

• It will be interesting to extend our algorithm for non-rigid movements of robots. In non-rigid movements of robots, there is an adversary that obstructs the robots from their movements to the destination points.

• It will be interesting to reduce the number of colors used in our solutions. For the dimensionless point robots, Sharma et al. [40] recently presented a 2-color solution for all settings of robots under both rigid and non-rigid moves, which is optimal.

• It will be interesting to solve the MUTUAL VISIBILITY problem avoiding collisions and without the assumption of chirality for fat robots that are not equipped with lights. In the closely related work, Agathangelou et al. [2] solved the problem, but their solution does not avoid collisions, and it needs the assumption of chirality.

• It will be interesting to explore the solvability of other problems such as FLOCKING, SCATTERING, etc. for the robots with extents under obstructed visibility.

• It will be interesting to extend our algorithm for the ASYNCH setting.

• It will be interesting to extend our $O(n)$ round algorithm for the FSYNCH setting to $O(n)$ round solution in the SSYNCH and ASYNCH settings in the robots with lights model.

• Finally, it will be interesting to design an algorithm that solves the mutual visibility problem for fat robots in $O(\log n)$ time in all FSYNCH, SSYNCH, and ASYNCH settings in the robots with lights model.
Bibliography


[38] Illah R. Nourbakhsh Roland Siegwart and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2011.


