Formation of General Position by Asynchronous Mobile Robots

S. Bhagat  
ACM Unit  
Indian Statistical Institute  
Kolkata-700108  
subhash.bhagat.math@gmail.com

S. Gan Chaudhuri  
Department of Information Technology  
Jadavpur University  
Kolkata-700032  
srutiganc@it.jusl.ac.in

K. Mukhopadhyaya  
ACM Unit  
Indian Statistical Institute  
Kolkata-700108  
krishnendu@isical.ac.in

ABSTRACT
The traditional distributed model of autonomous, homogeneous, mobile point robots usually assumes that the robots do not create any visual obstruction for the other robots, i.e., the robots are see through. In this paper, we consider a slightly more realistic model, by incorporating the notion of obstructed visibility (i.e., robots are not see through) for other robots. Under the new model of visibility, a robot may not have the full view of its surroundings. Many of the existing algorithms demand that each robot should have the complete knowledge of the positions of other robots. Since, vision is the only mean of their communication, it is required that the robots are in general position (i.e., no three robots are collinear). We consider asynchronous robots. They also do not have common chirality (or any agreement on a global coordinate system). In this paper, we present a distributed algorithm for obtaining a general position for the robots in finite time from any arbitrary configuration. The algorithm also assures collision free motion for each robot. This algorithm may also be used as a preprocessing module for many other subsequent tasks performed by the robots.

Keywords
Asynchronous, oblivious, obstructed visibility, general position.

1. INTRODUCTION
The study of a set of autonomous mobile robots, popularly known as swarm robots or multi robot system, is an emerging research topic in last few decades. Swarm of robots is a set of autonomous robots that have to organize themselves in order to execute a specific task in collaborative manner. Various problems in several directions, have been studied in the framework of swarm robots, among the others distributed computing is an important area with this swarm robots. This paper explores that direction.

1.1 Framework
The traditional distributed model \cite{12} for multi robot system, represents the mobile entities by distinct points located in the Euclidean plane. The robots are anonymous, indistinguishable, having no direct means of communication. They have no common agreement in directions, orientation and unit distance. Each robot has sensing capability, by vision, which enables it to determine the position (within its own coordinate system) of the other robots. The robots operate in rounds by executing Look-Compute-Move cycles. All robots may or may not be active at all rounds. In a round, when becoming active, a robot gets a snapshot of its surroundings (Look) by its sensing capability. This snapshot is used to compute a destination point (Compute) for this robot. Finally, it moves towards this destination (Move). The robot either directly reaches destination or moves at-least a small distance towards the destination. The choice of active robot in each round is decided by an adversary. However, it is guaranteed that each robot will become active in finite time. All robots execute the same algorithm. The robots are oblivious, i.e., at the beginning of each cycle, they forget their past observations and computations \cite{10}. Depending on the activation schedule and the duration of the cycles, three models are defined. In the fully-synchronous model, all robots are activated simultaneously. As a result, all robots act on same data. The semi-synchronous model is like the fully synchronous, except that the set of robots to be activated is chosen at random. As a result, the active robots act on same data. No assumption, is made on timing of activation and duration of the cycles for asynchronous model. However, the time and durations are considered to be finite.

Vision and mobility enable the robots to communicate and coordinate their actions by sensing their relative positions. Otherwise, the robots are silent and have no explicit message passing. These restrictions enable the robots to be deployed in extremely harsh environments where communication is not possible, i.e an underwater deployment or a military scenario where wired or wireless communications are impossible or can be obstructed or erroneous.

1.2 Earlier works
Majority of the investigations \cite{9} \cite{12} on mobile robots assume that their visibility is unobstructed or full, i.e., if two robots $A$ and $B$ are located at $a$ and $b$, they can see each other though other robots lie in the line segment $ab$ at that time. Very few observations on obstructed visibility (where
A and B are not mutually visible if there exist other robots on the line segment $ab$ have been made in different models; such as, (i) the robots in the one dimensional space [5]; (ii) the robots with visible lights [7, 8] and (iii) the unit disc robot called fat robots [1, 6].

The first model studied the uniform spreading of robots on a line [3]. In the second model, each agent is provided with a local externally visible light, which is used as colors [7, 8, 9, 11, 12, 13, 2]. The robots implicitly communicate with each other using these colors as indicators of their states. In the third model, the robots are not points but unit discs [4, 6, 1] and collisions among robots are allowed.

Obstructed visibility have been addressed recently in [2] and [3]. In [2] the authors have proposed algorithm for robots in light model. Here, the robots starting from any arbitrary configuration form a circle which is itself an unobstructed configuration. The presence of a constant number of visible light(color) bits in each robot, implicitly help the robots in communication and storing the past configuration. In [2], the robots obtain a obstruction free configuration by getting as close as possible. Here, the robots do not have light bits. However, the algorithm is for semi-synchronous robots.

1.3 Our Contribution
In this paper, we propose algorithm to remove obstructed visibility by making of general configuration by the robots. The robots start from arbitrary distinct positions in the plane and reach a configuration when they all see each other. The robots are asynchronous, oblivious, having no agreement in coordinate systems. The obstructed visibility model no doubt improves the traditional model of multi robot system by incorporating real-life like characteristic. The problem is also a preliminary step for any subsequent tasks which require complete visibility.

The organization of the paper is as follows: Section 2 defines the assumptions of the robot model used in this paper and presents the definitions and notations used in the algorithm. Section 3 presents an algorithm for obtaining general position by asynchronous robots. We also furnish the correctness of our algorithm in this section. Finally in section 4 we conclude by providing the future directions of this work.

2. MODEL AND DEFINITIONS
Let $R = \{r_1, \ldots, r_n\}$ be a set of $n$ homogeneous robots represented by points. Each robot can sense (see) 360° around itself up to an unlimited radius. However, they obstruct the visibility of other robots. The robots execute look-compute-move cycle in asynchronous manner. They are oblivious and have no direct communication power. The movement of the robots are non-rigid, i.e., a robot may stop before reaching its destination. However, a robot moves at-least a minimum distance $\delta > 0$ towards its destination. This assumption assures that a robot will reach its destination in finite time. Initially the robots are positioned in distinct locations and are stationary. Now we present some notations and conventions which will be used throughout the paper.

- **Position of a robot**: $r_i \in R$ represents a location of a robot in $R$ at some time, i.e., $r_i$ is a position occupied by a robot in $R$ at certain time. To denote a robot in $R$ we refer by its position $r_i$.

- **Measurement of angles**: By an angle between two line segments, if otherwise not stated, we mean the angle made by them which is less than equal to $\pi$.

- **$V(r_i)$**: For any robot $r_i$, we define the vision of $r_i$, $V(r_i)$, as the set of robots visible to $r_i$ (excluding $r_i$ itself). The robots in $V(r_i)$ can also be in motion due to asynchronous scheduling.

If we sort the robots in $V(r_i)$ by angle at $r_i$, w.r.t. $r_i$ and connect them in that order, we get a star-shaped polygon, denoted by $STR(r_i)$. Note that if and only if $r_i \in V(r_i)$ (Figure 1).

**Figure 1: An example of STR($r_i$)**

- **CR($r_i$)**: This is the set of line segments joining $r_i$ to all its neighbors or all robots in $V(r_i)$.

- **L($r_i r_j$)**: Straight line through $r_i$ and $r_j \in V(r_i)$ (Figure 3).

- **COL($r_i$)**: COL($r_i$) denotes the set of robots for which $r_i$ creates visual obstructions.

- **DISP($r_i r_j$)**: When a robot $r_i$ moves to new position $\hat{r}_i$, we call $\angle r_i r_j \hat{r}_i$ as the angle of displacement of $r_i$ w.r.t. $r_j$ and denote it by $DISP(r_i r_j)$ (Figure 3).

**Figure 3: Examples of $\mathcal{L}(r_i r_j)$, $DISP(r_i r_j) = \angle r_i r_j \hat{r}_i$, $COL(r_i) = \{r_i, r_m\}$**
Figure 4: Examples of $\Gamma(r_i)$, $\alpha(r_i)$, Bisec($r_i$), intersect($r_i$)

- $\Gamma(r_i)$: Set of angles $\angle r_j r_i r_k$ where $r_k$ and $r_j$ are two consecutive vertices of $STR(r_i)$ (Figure 4).
- $\alpha(r_i)$: Maximum of $\Gamma(r_i)$ if maximum value of $\Gamma(r_i)$ is less than $\pi$ otherwise the $2^{nd}$ maximum of $\Gamma(r_i)$. The tie, if any, is broken arbitrarily (Figure 4).
- Bisec($r_i$): Bisector of $\alpha(r_i)$. Note that Bisec($r_i$) is a ray from $r_i$ towards the angle of consideration (Figure 4).
- DIR($r_i$): The direction of Bisec($r_i$). We say that DIR($r_i$) lies on that side of any straight line where infinite end of DIR($r_i$) lies (Figure 4).
- intersect($r_i$): We look at the intersection points of Bisec($r_i$) and $L_{jk}$, $\forall r_j, r_k \in V(r_i)$. The intersection point closest to $r_i$ is denoted by intersect($r_i$) (Figure 4).
- $\Gamma'(r_i)$: Set of angles $\angle r_{i-1} r_j r_i$ and $\angle r_j r_{i+1}$, $\forall r_j \in V(r_i)$, where $r_{i-1}$ and $r_{i+1}$ are the two neighbors of $r_i$ on $STR(V(r_j))$ (Figure 5).

Figure 5: Examples of $\Gamma'(r_i)$, $\beta(r_i) = \angle r_{i-1} r_i r_{i+1}$

- $\beta(r_i)$: Minimum of $\Gamma(r_i) \cup \Gamma'(r_i)$ (Figure 5).
- $\theta(r_i)$: $\frac{\beta(r_i)}{\pi}$.
- $d(r_i)$: Distance between $r_i$ and intersect($r_i$).
- D($r_i$): Distance between $r_i$ and the robot nearest to it.
- $\Delta(r_i)$: $\min\left\{ \frac{d(r_i)}{\pi}, D(r_i) \sin(\theta(r_i)) \right\}$.

- $r_i$ : The point on Bisec($r_i$), $\Delta(r_i)$ distance apart from $r_i$ (Figure 6).
- C($r_i$) : The circle of radius $\Delta(r_i)$ centered at $r_i$. Note that $r_i$ always lies on C($r_i$) (Figure 6).
- T(C($r_i$), $r_j$) : Any one of the tangential points of the tangents drawn to C($r_i$) from $r_j$ (Figure 6).

Figure 6: Examples of C($r_i$), $\hat{r}_i$, T(C($r_i$), $r_j$)

We classify the robots in $R$ depending upon their positions with respect to $CH(R)$ (the convex hull of $R$), as below:

- External vertex robots ($R_{EV}$): A set of robots lying on the vertices of $CH(R)$ . These robots do not obstruct the visibility of any robot in $R$ and hence they do not move during whole execution of the algorithm. Note that, if $r_i$ lies outside of $STR(r_i)$ , then $r_i$ is an external vertex robot.
- External edge robots ($R_{EE}$): A set of robots lying on the edges of $CH(R)$. These robots either block the visibility of external vertex robots or other robot edge robots. Note that, if $r_i$ lies on an edge of $STR(r_i)$, then $r_i$ is an external edge robot.
- Internal robots ($R_I$): A set of robots lying inside the $CH(R)$. Note that, if $r_i$ lies within $STR(r_i)$, $r_i$ is an internal robot.

3. ALGORITHM FOR MAKING OF GENERAL POSITION

Consider initially robots in $R$ are not in general position. Our objective is to move the robots in $R$ in such a way that after a finite number of movements of the robots in $R$, it will be in general position. In order to do so, our approach is to move the robots which create visual obstructions to the other robots. If a robot $r_i$ lies between two other robots, say $r_p$ and $r_q$ such that $r_i$, $r_p$ and $r_q$ are in straight line, then $r_i$ is selected for movement. The destination of $r_i$, say $T(r_i)$, is computed in such a way that, there always exists a $r_j \in R$ (where $r_j$ does not have full visibility), such that when $r_i$ moves, the cardinality of the set of visible robots of $r_j$ increases. Since, the number of robots are finite, the number of robots having partial visibility, is also finite. Our algorithm assures that at each round at-least one robot with partial visibility will have full visibility. This implies that in finite number of rounds all robots will achieve full visibility, hence, the robots will be in general position in finite time.
3.1 Computing the destinations of the robots

A collinear middle robot is selected to move from its position. A robot finds its destination for movement using algorithm ComputeDestination\(r_i\). A robot \(r_i\), selected for moving, moves along the bisector of the minimum angle created at \(r_i\) by the robots in \(V(r_i)\). The destination is chosen in such a way that \(r_i\) will not block the vision of any \(r_j \in V(r_i)\), where the vision of \(r_j\) was not initially blocked by \(r_i\) throughout the paths towards its destination. Each movement of \(r_i\) breaks at least one initial collinearity.

Algorithm 1: ComputeDestination()

Input: \(r_i \in R\) with \(COL(r_i) \neq \phi\).
Output: a point on Bisec\(r_i\).

1. Compute \(\alpha(r_i),\) Bisec\(r_i\), \(\beta(r_i),\) \(\theta(r_i),\) \(D(r_i),\)
2. Case 1: \(\beta(r_i) \neq 0,\)
   \(\Delta(r_i) \leftarrow \min\left(\frac{d(\alpha)}{\alpha}, \ D(r_i)\sin(\theta(r_i))\right)\)
3. Case 2: \(\beta(r_i) = 0,\)
   \(\Delta(r_i) \leftarrow D(r_i)\)
4. Compute the point \(\hat{r}_i\) on Bisec\(r_i\), \(\Delta(r_i)\) distance apart from \(r_i\);
5. return \(\hat{r}_i;\)

Proof of Correctness of algorithm ComputeDestination().

Correctness of the algorithm is established by following observations, lemmas.

![Diagram](image)

**Figure 7: An example for lemma 1**

**Lemma 1.** \(\beta(r_i) \leq \frac{\pi}{2}\).

**Proof.** If all the robots lie on a straight line, then \(\beta(r_i) = 0\). Suppose there are at least three non-collinear robots. For three robots forming a triangle, \(\beta(r_i)\) is maximum when the triangle is equilateral. For all other cases, consider the triangle formed by \(r_i, r_j,\) and \(r_{i-1}\) where \(r_j\) is any robot in \(V(r_i)\) and \(r_{i-1}\) is a neighbor of \(r_i\) on \(STR(V(r_i))\). If \(r_j\) is also a neighbor of \(r_i\) on \(V(r_{i-1})\) (Figure 7(a)), then \(\angle r_i r_{i-1} r_j\) and \(\angle r_i r_{i+1} r_j\) are in \(\Gamma(r_i)\) and either \(\angle r_i r_{i+1} r_j\) or an angle less than it is in \(\Gamma(r_i)\). On the other hand, if \(r_j\) is not a neighbor of \(r_i\) on \(V(r_{i-1})\) (Figure 7(b)), then instead of \(\angle r_i r_{i+1} r_j\), an angle less than it, is in \(\Gamma(r_i)\). In all cases, \(\beta(r_i)\) is less than the minimum of the angles of the triangle formed by \(r_i, r_j\) and \(r_{i-1}\). Hence, \(\beta(r_i) \leq \frac{\pi}{2}\).

\[\square\]

**Observation 1.** Maximum value of \(DISP(r_i, r_j)\), denoted by \(\max\{DISP(r_i, r_j)\}\), is attained when \(\hat{r}_i\) coincides with one of the tangential points \(T(C(r_i), r_j)\).

**Lemma 2.** For any \(r_i,\) \(DISP(r_i, r_j) \leq \theta(r_i) \land r_j\).

**Proof.** Let \(r_j\) be a robot in \(V(r_i)\) and \(r_j\) a robot closest to \(r_i\). By observation 1, maximum values of \(DISP(r_i, r_j)\) and \(DISP(r_i, r_{k})\) are attained at tangential points \(T(C(r_i), r_j)\) and \(T(C(r_i), r_k)\) respectively. Hence, \(DISP(r_i, r_j)\) is less than \(\frac{\pi}{2}\) for all \(j\). By definition,

\[
\frac{\Delta(r_i)}{\max\{DISP(r_i, r_k)\}} = \sin(\max\{DISP(r_i, r_k)\}) \\
\leq \sin(\theta(r_i))
\]

(1)

Again,

\[
\frac{\Delta(r_i)}{\max\{DISP(r_i, r_j)\}} = \sin(\max\{DISP(r_i, r_j)\})
\]

(2)

Since \(\frac{\Delta(r_i)}{\theta(r_i)}\), from (1) and (2) we have,

\[
\sin(\max\{DISP(r_i, r_j)\}) \leq \sin(\theta(r_i)).
\]

(3)

**DISP(r_i, r_j) and \(\theta(r_i)\) are in \([0, \frac{\pi}{2}]\) (by lemma 1 and sine is an increasing function in \([0, \frac{\pi}{2}]\). From (3) we conclude,

\[
DISP(r_i, r_j) \leq \theta(r_i)
\]

\[\square\]

Suppose a robot \(r_i \in R\) moves according to our algorithm. We claim that it will never become collinear with any two robots \(r_j\) and \(r_k\) in \(R\) where \(r_i, r_j\) and \(r_k\) are not collinear initially. Now we state arguments to prove our claim.

**Observation 2.** Let \(ABC\) be a right-angled triangle with \(\angle ABC = \frac{\pi}{2}\). Let \(D\) be a point on the side \(AC\) such that \(\angle DC = \frac{1}{2}\angle AC\). Then,

\[
\angle BDA \leq \angle ACB.
\]

**Lemma 3.** Suppose \(r_i\) and \(r_j\) move to new positions \(\hat{r}_i\) and \(\hat{r}_j\) in at most one computation cycle. Let \(\phi\) be the angle between \(L_{\hat{r}_i r_j}\) and \(L_{\hat{r}_i \hat{r}_j}\) i.e., \(\phi = \angle r_i r_j c\) where \(c\) is the intersection point between \(L_{\hat{r}_i r_j}\) and \(L_{\hat{r}_i \hat{r}_j}\). Then,

\[
\phi < 2 \max\{\theta(r_i), \theta(r_j)\}
\]

**Proof.** If any one \(r_i\) and \(r_j\) moves, then lemma is trivially true. Suppose both of them move once.

**Case 1:**

Suppose \(r_i\) and \(r_j\) move synchronously. Without loss of generality, let \(\Delta(r_i) \geq \Delta(r_j)\).

**Case 1.1:**

Suppose \(\theta(r_i)\) and \(\theta(r_j)\) lie in the opposite sides of \(L_{\hat{r}_i r_j}\) (Figure 8). In view of observation 1, \(\max\{\phi\}\), the maximum value of \(\phi\), is attained when \(L_{\hat{r}_i r_j}\) is a common tangent to \(C(r_i)\) and \(C(r_j)\). Let \(M\) be the middle point of \(\hat{r}_i \hat{r}_j\). If
\[ C(r_i) \] is strictly larger than \( C(r_j) \), \( c \) is closer to \( r_j \) than \( r_i \). If they are equal, \( c \) coincides with \( M \). Consider the right-angled triangle \( \Delta r_i \hat{r}_i r_j \). By observation 2,

\[
\phi \leq \max\{\phi\} \\
\leq 2\text{DISP}(r_i r_j) \\
< 2\max\{\text{DISP}(r_i r_j)\} \\
\leq 2\theta(r_i)
\]

- **Case 1.2:**
  \( \text{DIR}(r_i) \) and \( \text{DIR}(r_j) \) lie in the same side of \( L_{r_i r_j} \) (Figure 9). \( \max\{\phi\} \) is attained when \( L_{r_i,\hat{r}_i} \) is a tangent to \( C(r_i) \) from the point \( c \) and \( c \) coincides with the closest point of \( C(r_j) \) from \( r_i \). Then following same argument as in case-1, we have the proof.

- **Case 2:**
  \( r_i \) and \( r_j \) move asynchronously. Suppose \( r_i \) is moving and is at \( r_i' \) when \( r_j \) takes the snapshot of its surroundings to compute the value of \( \Delta(r_j) \). Since \( r_i \) has already computed the value of \( \Delta(r_i) \) and computation of \( \Delta \) values of \( r_i \) and \( r_j \) are independent, the proof follows from the same arguments as in case 1. In this case the value of \( \Delta(r_j) \) may be different from the value in case 1.

\[ \Box \]

**Lemma 4.** Suppose two robots \( r_i \) and \( r_j \) move to \( \hat{r}_i \) and \( \hat{r}_j \) respectively in at most one movement. Then

\[
\max\{\text{DISP}(r_i \hat{r}_i), \text{DISP}(r_j \hat{r}_j)\} < 2\max\{\theta(r_i), \theta(r_j)\}.
\]

**Proof.** Follows from observation 2 and lemma 3 (Figure 10).

\[ \Box \]

**Lemma 5.** If \( r_i, r_j \) and \( r_k \) are not collinear and mutually visible to each other, then during the whole execution of the above algorithm, they never become collinear.

**Proof.** We have the following cases,

- **Case 1 (Only one robot moves):**
  Without loss of generality, suppose \( r_j, r_k \) stand still and \( r_i \) moves. If \( \text{DIR}(r_i) \) does not intersect \( L_{r_j r_k} \) (Figure 11(a)), then the claim is trivially true.
  Suppose \( \text{DIR}(r_i) \) intersects \( L_{r_j r_k} \) (Figure 11(b)). Since distance traversed by \( r_i \) is bounded above by \( \frac{d(r_i)}{2} \), \( r_i \) can not reach \( L_{r_j r_k} \) and \( r_i, r_j \) and \( r_k \) will not become collinear.

- **Case 2 (Two of the robots move):**
  Without loss of generality, suppose \( r_i \) and \( r_j \) move while \( r_k \) remains stationary. This case would be feasible only if \( n \geq 4 \).
  - **Case 2.1:**
    Suppose \( r_i \) and \( r_j \) move synchronously. Then by lemma 2
    \[
    \text{DISP}(r_i r_k) \leq \frac{\angle r_i r_k r_j}{n^2} \tag{4}
    \]
    And
    \[
    \text{DISP}(r_j r_k) \leq \frac{\angle r_j r_k r_i}{n^2} \tag{5}
    \]
    From equation (4) and (5)
    \[
    \text{DISP}(r_i r_k) + \text{DISP}(r_j r_k) \leq \angle r_i r_k r_j \tag{6}
    \]
    The minimum value of \( \text{DISP}(r_i r_k) + \text{DISP}(r_j r_k) \) for which \( r_i, r_j \) and \( r_k \) could become collinear is \( \angle r_i r_k r_j \). In view of equation (6), we conclude that \( r_i, r_j \) and \( r_k \) would never become collinear.
  - **Case 2.2:**
    Suppose \( r_i \) is in motion and is at \( r_i' \) when \( r_j \) computes the value of \( \Delta(r_j) \). If \( r_j' \) and \( r_j \) lie in opposite sides of \( L_{r_j r_k} \) (Figure 12(a)), then
Figure 12: An example of case 2.2 for lemma 5

\[ \Delta(r_j) \leq \frac{1}{n^2} \text{dist}(r_j, L_{r_i r_k}) \]

which implies that \( r_j \) cannot reach \( L_{r_i r_k} \) when \( r_i \) reaches its destination and hence the lemma. Suppose \( \hat{r}_i \) and \( \hat{r}_j \) lie in same side of \( L_{r_i r_k} \) (Figure 12(b)). Then we have,

\[ \text{DISP}(r_i r_k) \leq \frac{\mathcal{L}_{r_i r_k} r_j}{n^2} \]

Lemma follows from the same arguments as used in Case 2.1.
Consider the case: suppose \( r_j \) takes the snapshot at time \( t \) and moves to its destination at time \( t' \).
In between times \( t \) and \( t' \), suppose \( r_i \) has made at most \( \frac{n-2}{2} \) moves (we shall prove in case 3.2 that number of movements of any robot is bounded above by \( \frac{n-1}{2} \)). If \( r_i \) moves towards \( r_j \), after \( \frac{n-1}{2} \) moves, we would have

\[ \text{DISP}(r_i r_j) < (1 - \frac{1}{n^2}) \frac{n-1}{2} \angle r_i r_k r_j \]

which is less than \( (1 - \frac{1}{n^2}) \angle r_i r_k r_j \). Hence equation (6) is satisfied in this case and we have the proof of the lemma. If \( r_i \) moves away from \( r_j \), then there is nothing to prove.

• Case 3 (All three robots move):

  – Case 3.1:
    Suppose \( r_i, r_j \) and \( r_k \) move synchronously.

  – Case 3.1.1:
    Suppose \( L_{r_i r_j} \) intersects \( L_{r_i r_k} \) at an angle \( \phi > 0 \) (Figure 13).

Figure 13: An example of case 3.1.1 for lemma 5

By lemma 5

\[ \phi < 2 \max \{\theta(r_i), \theta(r_j)\} \]

which implies that \( r_i \) and \( r_j \) are parallel i.e., \( \phi = 0 \) which implies that \( \psi = \angle r_i r_j r_k \) (Figure 14). Let \( \text{Bisec}(r_k) \) intersect \( L_{r_i r_k} \) at \( P \) and \( |r_k P| = t \).

Since \( \Delta(r_k) \leq |r_i r_j| \sin \left( \frac{\mathcal{L}_{r_i r_k} r_j}{n^2} \right) \) and \( n \geq 5 \),

\[ l - \Delta(r_k) \geq |r_i r_j| \sin \left( \angle r_i r_j r_k \right) - \Delta(r_k) \geq |r_i r_j| \sin \left( \frac{\mathcal{L}_{r_i r_k} r_j}{n^2} \right) \geq |r_i r_j| \left( \sin \left( \frac{\mathcal{L}_{r_i r_k} r_j}{n^2} \right) \right) - \Delta(r_k) \geq |r_i r_j| \sin \left( \frac{\mathcal{L}_{r_i r_k} r_j}{n^2} \right) \]

\[ \Delta(r_k) \text{ and } \Delta(r_j) \text{ are bounded above by } |r_i r_j| \sin \left( \frac{\mathcal{L}_{r_i r_k} r_j}{n^2} \right) \]. Hence by equation (11), \( r_i \) and \( r_j \) and \( r_k \) do not become collinear.

• Case 3.2:
Suppose \( r_i, r_j \) and \( r_k \) move asynchronously. The main problem in this case is the following scenario: suppose
r_j or r_k takes the snapshot at time t_j or t_k respectively and starts moving to its computed destination at time t'_j or t'_k respectively. Suppose the configuration has been changed in between the times due to the movements of the other robots. Then the corresponding Δ value of r_j or r_k is not consistent w.r.t. the current configuration. We have to show that this would not create any problem for our algorithm. The main idea of proof in this case is that we have to estimate the maximum amount of inclination of L_{r,r_j} towards r_k between the times r_j or r_k takes the snapshot of surroundings and it reaches the destination. So, in the following proofs we only consider the scenarios (as in the case 3.1.1. and case 3.1.2.) in which there are possibilities of maximum reduction in the ∠r, r_j r_k, which depicts the inclination of L_{r,r_j} towards r_k. Note that the inclination of L_{r,r_j} towards r_k is maximum when both r_i and r_j move synchronously. So, we only prove the case when r_k holds the old value of Δ.

• Case 3.2.1

Suppose r_k holds the old value of Δ w.r.t. to the current configuration. Suppose r_i and r_j are at t_i and t'_j respectively when r_k takes the snapshot at time t_k. Suppose till t'_j, r_i and r_j move x and x’ times respectively. Note that initially r_i and r_j can be collinear with n − 1 robots and to remove these collinearities they have to move at most \( \frac{n-1}{2} \) times if they do not create any new collinearity (this bound is obtained by considering the degenerate case i.e., when all the robots are collinear initially).

First we prove that x and x’ are bounded above by \( \frac{n-1}{2} \). To prove this we show that r_i and r_j do not create any new collinearity while moving. We prove this for arbitrary robots. Suppose some robot r_s, while moving, creates a new collinearity with r_i and r_m for the first time during the execution of our algorithm (Figure 15). Then either one of r_i and r_m or both amount it would reach this straight line. We try to estimate the inclination of L_{r,r_i} towards r_m (which is depicted by the angle ψ as in the case 3.1.1. and by the displacement of L_{r,r_j} towards r_m as in the case 3.1.2.) after \( \frac{n-1}{2} \) number of movements of r_s and r_i (note that we have consider the over estimated value of the number of movements of r_s and r_i). As computed in the case 3.1.1, after first movement,

\[
ψ > (1 - \frac{1}{n})∠r_s r_i r_m
\]

and \( ∠r_s r_i r_m \) will become at most \( (1 + \frac{1}{n}) ∠r_s r_i r_m \). By the same repeated arguments, we can say that after d movements

\[
ψ > (1 - \frac{1}{n^d}) ∠r_s r_i r_m
\]

which is strictly greater than \( \frac{1}{n^d} ∠r_s r_i r_m \) for \( d ≤ \frac{n-1}{2} \).

This contradicts the fact that r_s creates collinearity with r_i and r_m. For the scenario same as the case 3.1.2., we have,

\[
|\overrightarrow{r_m}|sin(∠r_s r_i r_m) = \frac{n-1}{2} |\overrightarrow{r_m}| sin(\frac{∠r_s r_i r_m}{n^2}) > |\overrightarrow{r_m}| sin(\frac{∠r_s r_i r_m}{n^2})
\]

This also contradicts the fact that r_s creates collinearity with r_i and r_m. Hence, we conclude that r_s would not become collinear with r_i and r_m.

In the above proof, we replace r_s, r_j and r_m by r_i, r_j and r_k respectively to conclude that r_s would not become collinear with r_j and r_k during the whole execution of our algorithm.

\[\Box\]

**Lemma 6.** Consider any two robots r_i and r_j, r_i does not cross Bisec(r_j).

**Proof.** If Bisec(r_j) and Bisec(r_i) do not intersect, then there is nothing to prove. Suppose Bisec(r_j) and Bisec(r_i) intersect at a point p (Figure 16). If at least one of intersect(r_i) and intersect(r_j) is closer to r_i and r_j respectively than p, then we are done. Else \( α(r_i) \) and \( α(r_j) \) are angle of same triangle \( Δr_i r_j r_k \) for some \( r_k ∈ R \) i.e., \( α(r_i) = ∠r_k r_i r_j \) and \( α(r_j) = ∠r_k r_j r_i \). In \( Δr_i r_j r_k \), let Bisec(r_i) and Bisec(r_j) intersect \( \overrightarrow{rr_k} \) and \( \overrightarrow{rr_j} \) at a and b respectively. Here \( n > 5 \).
In \( \triangle arjp \),
\[
|ar| = \sin \left( \frac{\angle arjp}{2} \right) \cdot \frac{|pr|}{\sin(\angle aprj)} \tag{13}
\]

In \( \triangle prjr \),
\[
|pr| = \sin \left( \frac{\angle prj}{2} \right) \cdot \frac{|rj|}{\sin(\angle aprj)}
= \sin \left( \frac{\angle prj}{2} \right) \cdot \frac{|rj|}{\sin(\angle aprj)} \tag{14}
\]

From equation 13 and 14,
\[
|ap| = \frac{|pr|}{\sin(\angle aprj)}
\]

Since \(|ar| < |rj|\), \(|ap| < |pr|\) which implies,
\[
\Delta(r_i) < \frac{|rj|}{|pr|} < \frac{|rj|}{|pr|}.
\]

Hence \( r_i \) can not cross \( \text{Bisec}(r_j) \). Similarly, \( r_j \) can not cross \( \text{Bisec}(r_i) \). □

**Lemma 7.** Suppose, for any robot \( r_i \in \mathcal{R} \), \( r_k \notin \mathcal{V}(r_i) \). Then during the whole execution of the algorithm \( r_i \) will not block the vision between \( r_j \) and \( r_k \) where \( r_j \in \mathcal{V}(r_k) \).

**Proof.** Let \( r_j \in \mathcal{V}(r_i) \cap \mathcal{V}(r_k) \). Suppose \( r_l \) be the nearest robot of \( r_i \) such that \( r_k \) lie on \( \mathcal{L}_{r_i r_l} \) (Figure 17). If \( \text{Bisec}(r_i) \) does not intersect \( \mathcal{L}_{r_i r_l} \), there is no possibility that \( r_l \) will block the vision between \( r_j \) and \( r_k \). Let \( \text{Bisec}(r_i) \) intersect \( \mathcal{L}_{r_i r_l} \). Then \( r_l \) is one of the immediate neighbor of \( r_i \) on \( \text{STR}(\mathcal{V}(r_i)) \). Let \( r_j' \) and \( r_k' \) be the other immediate neighbors of \( r_j \) and \( r_k \) respectively on \( \text{STR}(\mathcal{V}(r_i)) \). First we prove that \( r_i \) will always lie on the same side of \( \mathcal{L}_{r_j r_l} \) as it is initially even if \( r_j, r_j', r_k, r_k' \) and \( r_l \) move. By lemma 6 and the observation that the movements of \( r_j, r_j', r_k \) are bounded by the edges and chords of the polygon formed by \( \{r_j, r_j', r_k, r_k', r_j'\} \), we conclude \( r_l \) never crosses the line \( \mathcal{L}_{r_j r_l} \). To block the vision between \( r_k \) and \( r_j \), \( r_i \) has to move on the line segment \( \mathcal{L}_{r_l r_m} \). Since \( r_l \) and line segment \( \mathcal{L}_{r_l r_m} \) lies on different sides of \( \mathcal{L}_{r_j r_l} \), \( r_i \) will never block the vision between \( r_k \) and \( r_j \). Let \( r_j \notin \mathcal{V}(r_i) \). Then there is a robot \( r_m \) which creates visual obstruction between \( r_l \) and \( r_i \). Now the movement of \( r_l \) is bounded by the line \( \mathcal{L}_{r_i r_m} \) and hence the lemma. □

**Lemma 8.** If at any time \( t, r_j \in \mathcal{V}(r_i) \), then at \( t' (> t) \), \( r_j \in \mathcal{V}(r_i) \) even if \( r_i \) changes its position.

**Proof.** The proof is immediate from \( \mathcal{L} \) and \( \mathcal{L}^\prime \) □

**Lemma 9.** Cardinality of \( \mathcal{V}(r_i) \) is strictly increasing.

**Proof.** Lemma \( \mathcal{L} \) and \( \mathcal{L}^\prime \) imply the proof. □

**Lemma 10.** There exist at least two robots \( r_j, r_k \in \mathcal{R} \) for which \( \mathcal{V}(r_j) \) and \( \mathcal{V}(r_k) \) increase whenever \( r_i \) changes its position.

**Proof.** \( r_i \) moves whenever \( r_i \) is collinear with at least one pair of robots, \( (r_j, r_k) \), and \( r_i \) lies in between those robots. If \( r_j \) and \( r_k \) do not move then \( \mathcal{V}(r_j) \) and \( \mathcal{V}(r_k) \) increase whenever \( r_i \) moves because no robot can reach \( \mathcal{L}_{r_j r_k} \) due to the facts stated in lemma \( \mathcal{L} \) and \( \mathcal{L}^\prime \). When either \( r_j \) or \( r_k \) or both \( r_j \) and \( r_k \) moves, one member of \( \mathcal{COL}(r_j) \) and one member of \( \mathcal{COL}(r_i) \) can not see each other. Hence the lemma. □

### 3.2 Moving the robots to obtain general position

Next we will discuss the algorithm \text{MakeGenaralPosition}(), by which the robots in \( \mathcal{R} \) move to obtain full visibility. The robots in \( \mathcal{R}_l \) which create obstacle to other robots and the robots in \( \mathcal{R}_{EE} \) are eligible for movement by this algorithm. The robots compute destinations using \text{ComputeDestination}() and move towards it. The robots keep on executing the algorithm till there exist no three collinear robots in \( \mathcal{R} \).

**Algorithm 2: MakeGenaralPosition()**

**Input:** \( \mathcal{R} \), a set of robots with their positions.

**Output:** \( \hat{\mathcal{R}} \), which is in general position.

**while** \( r_i \in \mathcal{R}_{EE} \lor (r_i \in \mathcal{R}_l \land \mathcal{COL}(r_i) \neq \phi) \)** **do**

1. \( T(r_i) \leftarrow \text{ComputeDestination}(r_i) \);
2. Move to \( T(r_i) \);
3. Compute \( \mathcal{COL}(r_i) \);

**Proof of Correctness of algorithm MakeGenaralPosition().**

The algorithm assures that the robot will form general position in finite number of movements. The termination of the algorithm is established by following observation and lemmas.

**Observation 3.** \text{ComputeDestination} is not executed by a robot \( r_i \in \mathcal{R} \) if \( r_i \in \mathcal{R}_{EV} \lor (r_i \in \mathcal{R}_l \land \mathcal{COL}(r_i) = \phi) \).

**Lemma 11.** \( \mathcal{COL}(r_i) \) will be \( \phi \) in finite time.

**Proof.** In the initial configuration the number of robots in \( \mathcal{COL}(r_i) \) is upper bounded by \( n - 1 \). During the whole execution of our algorithm no new collinearity is created and for each iteration cardinality of \( \mathcal{COL}(r_i) \) is reduced by at least two. Hence after at most \( \frac{n-2}{2} \) number of iterations of the while loop in the above algorithm, \( \mathcal{COL}(r_i) \) will become null. □

**Lemma 12.** \( \forall r_i, \mathcal{V}(r_i) \) will be \( (n - 1) \) in finite number of execution of the cycle.
Proof. Let $\eta = |\bigcup_{i=1}^{n} V(r_i)|$. The algorithm for a robot $r_i$ terminates whenever $|V(r_i)|$ reaches the value $n - 1$. Hence the algorithm for all robots terminates when $\eta = \frac{n(n-1)}{2}$ which is a finite integer. By lemma 8 and 10 the value of $\eta$ increases whenever any robot moves. Hence after finite number of execution cycles $\eta$ reaches its maximum value $\frac{n(n-1)}{2}$. □

From the above results, we can conclude the following theorem:

**Theorem 1.** A set of asynchronous, oblivious robots (initially not in general position) without agreement in common chirality, can form general position in finite time.

4. CONCLUSION

In this paper we have presented an algorithm for obtaining general position by a set of autonomous, homogeneous, oblivious, asynchronous robots having no common chirality. The algorithm assures the robots to have collision free movements. Another important feature of our algorithm is that the convex hull made by the robots in initial position, remains intact both in location and size. In other words, the robots do not go out side the convex hull formed by them. This feature can help in many subsequent pattern formations which require to maintain the location and size and of the pattern.

Once the robots obtain general position, the next job could be to form any pattern maintaining the general position. Most of the existing pattern formation algorithms have assumed that the robots are see through. Thus, designing algorithms for forming patterns by maintaining general position of the robots, may be a direct extension of this work.

5. REFERENCES

[1] C. Agathangelou, C. Georgiou, and M. Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC), 250–259, 2013.

[2] G. Antonio Di Luna, P. Flocchini, S. Gan Chaudhuri, N. Santoro, and G. Viglietta. Robots with Lights: Overcoming Obstructed Visibility Without Colliding In Proc. 16th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS’14), to appear.

[3] G. Antonio Di Luna, P. Flocchini, F. Poloni, N. Santoro, and G. Viglietta. The Mutual Visibility Problem for Oblivious Robots. In Proc. 26th Canadian Conference on Computational Geometry (CCCG’14), to appear.

[4] K. Bolla, T. Kovacs, and G. Fazekas. Gathering of fat robots with limited visibility and without global navigation. In Int. Symp. on Fazekas Comp. 30–38, 2012.

[5] R. Cohen and D. Peleg. Local spreading algorithms for autonomous robot systems. Theoretical Computer Science, 399:71–82, 2008.

[6] J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. Theoretical Computer Science, 410(6â–7):481 – 499, 2009.

[7] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronous robots using visible bits. In Proceedings of the 32nd International Conference on Distributed Computing Systems (ICDCS), 506–515, 2012.

[8] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Synchronized dancing of oblivious chameleons. In Proc. 7th Int. Conf. on FUN with Algorithms (FUN), 2014.

[9] A. Efrima and D. Peleg. Distributed models and algorithms for mobile robot systems. In Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), 70–87, 2007.

[10] P. Flocchini, G. Prencipe, and N. Santoro. Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool, 2012.

[11] P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous of two robots with constant memory. In Proceedings of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO), 189–200, 2013.

[12] D. Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In Proc. 7th Int. Workshop on Distr. Comp. (IWDC), 1–12, 2005.

[13] G. Viglietta. Rendezvous of two robots with visible bits. In Proc. 9th Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS), 291–306, 2013.