Defence Science Journal, Vol. 75, No. 4, July 2025, pp. 411-417, DOI : 10.14429/dsj.20508

Obstacle Prone Area Coverage by Swarm of Mobile Robots with Limited Visibility

Banashree Mandal[#,*], Deepanwita Das[#,*], and Niladri Mandal[$]

[#]*National Institute of Technology, Durgapur – 713209, India*
[$]*DRDO-Defence Research & Development Laboratory, Hyderabad – 500058, India*
[*]*E-mail: bm.20cs1101@phd.nitdgp.ac.in*

ABSTRACT

This paper studies a distributed coverage algorithm of a bounded rectangular region in the presence of horizontal line obstacles by an autonomous swarm of asynchronous mobile robots. They follow the basic Look-Compute-Move model, formally known as the CORDA model. The robot has no prior knowledge about the internal environment of the target region, especially the number and location of the robots, as well as obstacles. Robots are assumed to be anonymous, small, identical, simple, oblivious, inexpensive, and non-communicating in nature. The robots have a limited range of visibility. The robots unanimously decompose the whole region into several non-overlapping horizontal strips, where each robot is responsible for painting at most two strips based on its initial position. The painting of the entire region is achieved within finite time without any collision and repetition.

Keywords: Area coverage; Limited visibility; Horizontal line obstacle; Robot swarm

NOMENCLATURE

d	: Visibility range
UPB	: The upper boundary of the strip
LWB	: The lower boundary of the strip
LFB	: The left boundary of the strip
RTB	: The right boundary of the strip

1. INTRODUCTION

Scientists have developed small robots that mimic social behaviors like coordination and synchronization to perform tasks, known as a swarm of robots[1-4]. Using a distributed system, each robot in the swarm completes a part of the task, achieving the overall goal when all parts are done. Swarm robots are widely applied in area coverage. In this work, *N* robots divide a rectangular area into non-overlapping horizontal strips, each assigned to a robot. The area, obstructed by horizontal line obstacles, is covered when all strips are painted. Robots drop lights as passive communication since no active communication exists.

The coverage algorithm has two phases: *EXPLORE* and *PAINT*. Robots, scattered initially, assemble on the left boundary using *ASSEMBLE* algorithm[5-6] and compute strip boundaries in the *EXPLORE* phase before covering them in the *PAINT* phase. Based on the CORDA model[7], robots observe their surroundings and calculate their next move, working asynchronously with limited visibility and local coordinate systems. These constraints make the coverage problem challenging, but the proposed solution addresses them effectively.

Received : 14 August 2024, Revised : 18 February 2025
Accepted : 05 March 2025, Online published : 26 June 2025

2. LITERATURE REVIEW

Authors have proposed a distributed painting algorithm[8] for non-repeated coverage of obstacle-free areas using CORDA model. The region is divided into virtual cells, with each robot assigned a cell based on rank. After all robots finish painting, the task is complete. To address realism, they proposed a second algorithm[9] for rectangular regions with horizontal obstacles, dividing the region into blocks with obstacles on upper boundaries. Robots paint these blocks, ensuring agreement on shared boundaries to avoid collisions. To increase complexity, authors have introduced another algorithm[10] for limited visibility of robots. Robots are connected by a visibility graph and paint strips by moving between boundaries. If multiple robots share a strip, it is divided by mutual agreement, ensuring complete, non-overlapping coverage. Another algorithm has been addressed[11], which presents the Simultaneous Allocation and Path Planning (SAPP) algorithm, which optimizes task assignment and path planning for drone swarms. The goal is to efficiently distribute tasks among drones while ensuring collision-free trajectories in dynamic environments. Later, in literature[12], authors proposed one method to demonstrate that a single operator can manage a swarm of 100+ heterogeneous robots using advanced AI-driven command interfaces. This was studied in DARPA's OFFSET (OFFensive Swarm-Enabled Tactics) program, which conducted six field exercises (FXs) at U.S. Army training sites. Additionally, several area coverage algorithms have been proposed for autonomous robots. The method in this paper[13] divides a target region into fixed-size sub-regions, each assigned to a robot for coverage, increasing complexity due to coordination. The approach in this paper[14] uses boustrophedon decomposition with adjacency graphs for non-overlapping coverage. Literature[15] employs vertical stripes

with cyclic paths, updating a Reeb graph at a communication cost. The offline algorithm[16] uses EEC and CPP with obstacle maps, while[17] tackles unknown environments using Repart-Coverage and auction-based allocation, ensuring adaptability but requiring high communication.

2.1 Our Contribution

Previous research[8-17] has focused on unlimited or limited visibility and various obstacle settings in the presence or absence of direct communication in both known and unknown environments. However, no solution exists for robots with limited visibility in unknown environments with horizontal line obstacles. We propose an algorithm for limited-visibility robots following an asynchronous timing model, which is more realistic than full or semi-synchronous models. The challenge is to achieve complete coverage without collisions or repeated painting in finite time. Our approach considers a region with fixed horizontal line obstacles, where robots, initially on the left boundary, perform asynchronous area coverage with *sleep* states, using the CORDA and full-compass models.

3. CHARACTERISTICS, MODELS AND ASSUMPTIONS

3.1 Characteristics

- **Identical and homogeneous:** Robots are identical.
- **Mobile:** Robots are allowed to move freely in any direction.
- **Autonomous:** Robots take their decision and complete their tasks independently.
- **Memory:** Robots have a memory to retain a few past pieces of information with some variables.
- **Limited Visibility:** A robot can only see within a fixed range, called visibility range (V_r), represented by a circle of radius d, where $V_r=d$. To extend visibility, robots are equipped with *Lights*[18-20] which they drop as needed. These *Lights* help detect robots outside the visibility range to prevent collisions. If a robot sees illumination, it knows another robot located nearby has dropped a *Light*.

  The exact *Light* may not be visible, but its illumination can be detected if it's within a distance, $d+h$ where, $h=d-\epsilon$ (ϵ negligible amount). This is shown in Fig. 1, where robot R, at point D, sees illumination from a *Light* at point A. While the source at A is not visible to R, it can estimate the source's location within a range $L+D$, where L is the distance from A to C, where, $0 \leq L \leq D$.

  Robots drop one *Light* at a time, ensuring a vertical gap of $d+\epsilon$ between consecutive *Light*s. If multiple *Lights* are detected, the robot considers only the illumination aligned with the y-axis to determine its next move, ignoring others. The *Light*s have no energy constraints, providing continuous support to overcome visibility limitations.
- ***Communication*:** Robots do not directly exchange information about their states, locations, etc. A robot carries some *Lights*; those, when dropped, signify its mark of presence to other robots. If such *Lights* are viewed, a robot figures out the next collision-free move accordingly. Hence, passive communication exists among robots.

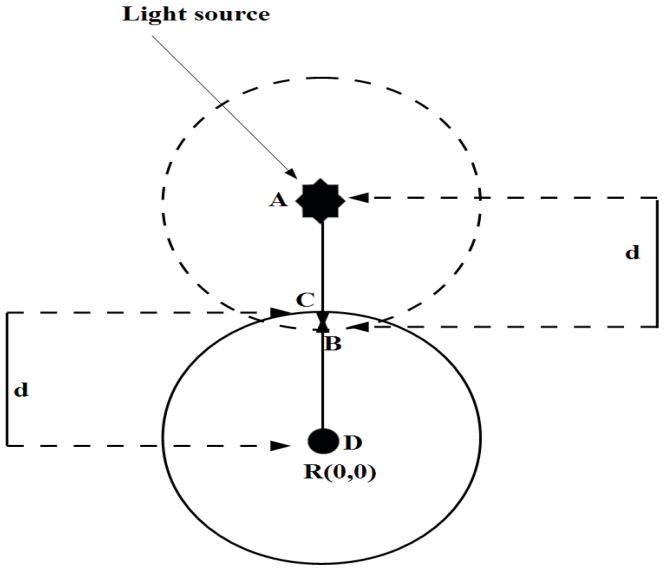

Figure 1. A robot views the illumination of light within its visibility range.

- ***Sensing-zone*:** Robots are point robots, and each of them has a sensing zone of radius λ. In this work, standing in a position, a robot can paint its entire sensing zone with the help of its actuator. Whenever a robot performs its assigned task for a specific region instead of actually reaching there, a robot executes its job from that λ distance.

3.2 Models

In this work, robots follow the CORDA model[7], which consists of sequential *Look-Compute-Move* phases. Each robot observes its surroundings (*Look*), calculates its next move (*Compute*), and moves accordingly (*Move*). This cycle repeats, with each robot running the same algorithm independently. Robots have their own local coordinate systems, assuming they are at the origin, and use a full-compass model, where the local x and y axes are common for all robots. Robots are asynchronous, so robots do not share a clock and may compute based on different snapshots. Robots can be either in *active* state (computing and moving) or in *sleep* state, but cannot remain asleep indefinitely..

3.3 Assumptions

The environment for exploration and painting is a bounded rectangular region with randomly placed horizontal obstacles of negligible width. These obstacles do not touch the region's boundaries and have gaps between them. However, robots can pass horizontally over obstacles, distinguishing them from other robots.

4. WORKING PROCEDURE AND ALGORITHM

The target area is divided into non-overlapping strips, where each strip is assigned to a robot. The algorithm ensures no gaps between strips, leading to full coverage once all strips are covered. The process involves two phases: *Explore*, where a robot determines its strip boundaries, and *Paint*, once the strip is covered. Once all the assigned strips are painted, the algorithm terminates. Robots begin from the left boundary of the region.

Generally, a robot paints two vertically consecutive strips: one above and one below its starting position, sharing a common lower boundary, *LWB*. The exploration and painting processes for the upper and lower strips are denoted as $Explore_1$, $Paint_1$, $Explore_2$, and $Paint_2$. $Explore_1$ includes two steps: $Explore_1$-*RTB*, the robot moves from the left boundary (*LFB*) to the right boundary (*RTB*) along the common *LWB*, and in $Explore_1$-*UPB*, it explores the upper boundary of the first strip. During $Paint_1$, the robot re-calculates the *LWB* and paints the first strip. In $Explore_2$ and $Paint_2$, the robot determines the upper boundary of the second strip and paints the second strip.

4.1 Explore$_1$-RTB

At the very beginning, *R* drops a *Light* at its initial position (say, *A* on *LFB*) shown in Fig. 2 (a) and computes common *LWB* (*AA'*) along its initial location. Next, *R* computes the *UPB* of both strips. To do that, *R* inspects objects above it, i.e., another robot (*Q*), *Light* (*K*), painted strip, and upper boundary of the region to determine the *UPB* of the first strip. If *R* detects any of these objects, it considers a horizontal line (*BB'*) through its current position (*B'*) as *UPB* of the first strip shown in Fig. 2(b). However, if the upper boundary of the region is detected above, then this boundary is considered as *UPB* of the first strip.

If *R* is at upper boundary of the region, this boundary serves as both its *LWB* and *UPB*. Similarly, *R* also determines UPB_{exp2} by checking the presence of any object (robot, *Light*, painted strip, or lower boundary) below it. If *R* finds any object(s) located below, *R* considers a horizontal line passing through the nearest object as its UPB_{exp2}; otherwise, it considers lower boundary of the region as its UPB_{exp2}.

If *R* detects no object above and/or below its initial position, it is unable to determine the *UPB* of either strip initially. However, for both cases (with known or unknown *UPB*s), *R* moves to its right along the *LWB* (*AA'*) towards the *RTB* and drops another *Light at A'*. The $Explore_1$-*RTB* ends once *R* reaches the *RTB*.

If *R* cannot determine the *UPB* of the first strip by end of $Explore_1$-*RTB*, *R* enters into $Explore_1$-*UPB*. If *R* calculates the upper boundary of the region as the *UPB* of the first strip, it also moves to $Explore_1$-*UPB*. Otherwise, it proceeds to $Explore_2$. During movement from *LFB* to *RTB*, if *R* encounters horizontal line obstacles, it bypasses them from above or below them.

4.2 In Explore$_1$-UPB

R moves upward from *RTB* to find *UPB* of the strip above. *R* performs vertical and horizontal movements alternatively, creating a zigzag pattern. It drops *Light*(s) when changing direction from vertical to horizontal. *R* continues this process until it detects another robot (*Q*) / *Light* (*K*) / painted strip above. At this point, *R* assumes a horizontal line through its current position as *UPB* of the first strip, as shown in Fig. 2(b). If *R* can see the upper boundary of the region, this boundary is considered *UPB* of the first strip.

Once the *UPB* of the first strip is determined, the robot drops a *Light* at its current location, ending the $Explore_1$-*UPB* step. If the robot computes the upper boundary of the region

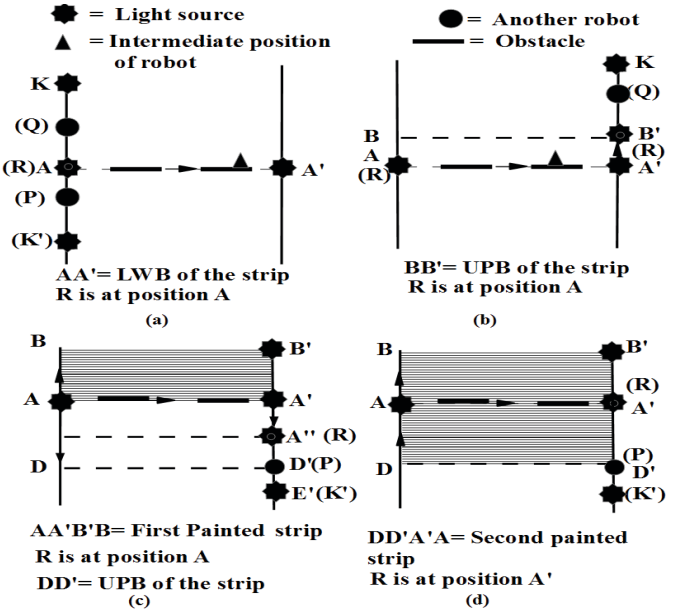

Figure 2. (a) Computation of *UPB* of both the strips at *LFB* due to the presence of another robot (Q) and/or light source (K) and completion of $Explore_1$-*RTB* (b) determine *UPB* of the first strip (c) Completion of $Paint_1$ (d) Determination of UPB_{exp2} and completion of $Paint_2$.

as *UPB*, it moves directly towards it using vertical movements instead of zigzag. Upon reaching the upper boundary, the $Explore_1$-*UPB* terminates, and *R* enters into $Paint_1$.

During $Paint_1$, the robot paints the first strip from *UPB* towards *LWB*. After finishing, *R* returns to *LWB* and ends $Paint_1$, shown in Fig. 2(c).

If *R* encounters obstacles during $Paint_1$, it moves from either above or below the obstacles. On termination of $Paint_1$, *R* enters the $Explore_2$.

After completing painting of its first strip, *R* begins exploring the strip below *LWB* by executing $Explore_2$. The second strip, is located directly below the first, sharing the *LWB* as a common boundary. In $Explore_2$, *R* moves downward to locate the UPB_{exp2}. Depending on where *R* finished $Paint_1$, it may start exploring along *RTB* or *LFB*. In some cases, *R* may skip $Explore_1$-*RTB* and $Paint_1$, transitioning directly into $Explore_2$ after $Explore_1$-*RTB*.

During $Explore_2$, located on its *LWB,* if a robot *R* finds the presence of one or more objects below its current position, it considers a horizontal line, passing through the current position of the nearest object as its UPB_{exp2}. Once the *UPB* is computed, $Explore_2$ terminates, and *R* enters $Paint_2$.

If *R* is on *LWB* and finds no object below, it performs zigzag movements downward to compute UPB_{exp2}. During this movement, *R* drops *Lights* at each change in direction from vertical to horizontal until it detects an object below. Once, *R* finds another robot (*P*) or a light (*K*) below, it considers the horizontal line passing through *P* or *K*'s position as the UPB_{exp2}. *R* can view a painted region below only if *P* has already painted it. *R* considers the visible boundary of this region to be its *UPB*. If *R* detects the lower boundary as an object below, it designates the lower boundary of that region as UPB_{exp2}.

After determining UPB_{exp2}, R reaches it while avoiding obstacles as previously described. Once UPB_{exp2} is computed, $Explore_2$ ends and begins $Paint_2$ to cover new strip. During $Paint_2$, R handles obstacles in the same way as in $Paint_1$.

During $Paint_2$, R starts painting from the UPB_{exp2} and stops at the common *LWB* is shown in Fig. 2 (d). R finishes its task when both the strips are covered, and the entire process ends once the whole region is fully painted.

5. CORRECTNESS PROOFS

The various lemmas have been proposed:

Lemma 1: Two robots never cross each other vertically.

Proof: It is assumed that two consecutive robots start their exploration at different vertical heights from the region's *LFB*. In the first cycle, they determine their *LWB*, *UPB*, $UPBexp_2$ based on rules in Section 4. Since the robots begin at different vertical heights, their horizontal movements start distinctly, preventing horizontal crossovers. Thus, this proof focuses on vertical crossovers. However, scenarios where robots move along common boundaries are managed using the sensing abilities discussed in Section 3.

During the *EXPLORE*, each robot is required to carry *Lights* and place them in appropriate positions. These Lights are colorless. For clarity in the figures, different-colored *Lights* represent the two robots. Throughout execution, each robot is either *EXPLORE* or *PAINT* state, leading to three possible state combinations for two robots.

Case (A): Both robots are in an *EXPLORE* state.

Case (B): One is in the *EXPLORE* state and other one is in the *PAINT* state.

Case (C): Both are in *PAINT* state.

Case (A): During the *EXPLORE* state, R starts at *LFB* and moves to *RTB* after calculating *LWB*, *UPB*, and $UPBexp_2$. Upon reaching *RTB, R* enters either $Explore_1$-*UPB* or $Explore_2$ based on *LWB*, *UPB*, and $UPBexp_2$. For proof, two robots, R_1 and R_2, are considered, initially not visible to each other. Throughout the process, there are three possible cases for the robots' locations.

Case (i): Both R_1 and R_2 are in $Explore_1$-*UPB.*

Case (ii): Both R_1 and R_2 are in $Explore_2$.

Case (iii): R_1 is in $Explore_1$-*UPB,* and R_2 is in $Explore_2$ or vice-versa.

Case (i): In $Explore_1$-*UPB*, a robot explores the upper boundary of the first strip if $UPB = \infty$. Starting from *LFB*, the robot moves to *RTB* along common *LWB*, initiating its first vertical movement along *RTB*. During exploration, the robot moves between *LFB* and *RTB* and can switch to sleep at any point.

When both R_1 and R_2 are in $Explore_1$-*UPB*, they begin at *LFB* and follow separate horizontal paths along their respective *LWB*s to reach *RTB*, from where they move upward to explore their strips. Each robot drops *Lights* at *RTB* to ensure they don't cross vertically, even if one robot falls asleep. The *Lights* prevent the lower robot from crossing the upper robot's boundary when it reactivates.

However, the adverse situation remains, as shown in Fig. 3. Starting the movement from A, R_1 enters *sleep* state at A' instead of reaching the opposite boundary at A_1. R_2 also continues its movement through points B_1, B_2, B_3, and so on to reach B_6 where, $V_{dist}(A_1, B_6) \geq d$, causing R_2 to cross R_1's strip. If R_1 reactivates and resumes exploration, collisions may occur.

To prevent this, the two robots never cross vertically each other. As depicted in Fig. 3, R_1 and R_2 are initially separated by a vertical distance D, where $D >> d$ and $d < D \leq L$ (where L is the region's length), making them invisible to each other.

Considering the *Light*s' illumination range, the minimum visibility distance between robots is $\leq$ d. Starting at A, R_1 enters *sleep* state at A' before reaching A_1. Meanwhile, R_2 moves through points B_1 to B_6, where $V_{dist}(A_1, B_6) \geq d$. If R_1 reaches A_1 without *sleep*, it drops a *Light* there, allowing R_2 to view it and compute the *UPB* of its first strip.

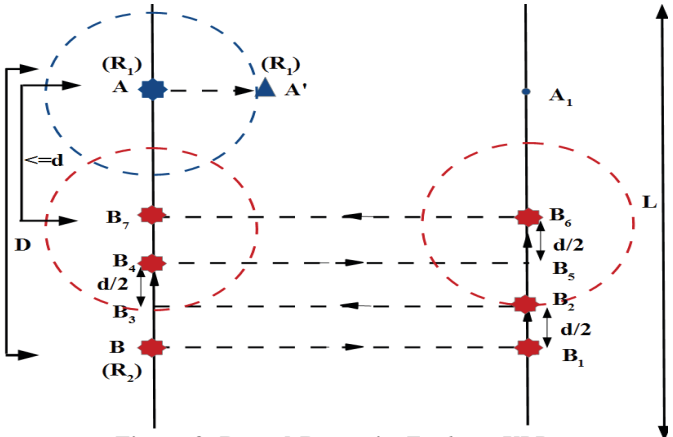

Figure 3. R_1 and R_2 are in $Explore_1$-*UPB*.

However, in this Scenario, R_2 does not detect any *Light* at B_6 and switches to horizontal exploration towards *LFB*. Upon reaching B_7, R_2 sees the *Light* dropped by R_1 at A, confirming that even if R_2 couldn't detect R_1 at B_6, but view it at B_7. As a result, the robots never cross vertically despite starting asynchronously and outside each other's visibility range. R_2 computes its *UPB* upon seeing the *Light* above and stays below it in later phases. Meanwhile, R_1 explores upward towards the *RTB*, proving that in $Explore_1$-*UPB*, the robots never cross vertically.

Case (ii): Since both are in $Explore_2$.They won't cross each other vertically.

Case (iii): In this Scenario, two robots, R_1 (above) and R_2 (below), are in different states: either (a) R_1 is in $Explore_1$-*UPB* while R_2 is in $Explore_2$, or (b) R_2 is in $Explore_1$-*UPB* while R_1 is in $Explore_2$.

Case (a): Since the robots' movements are exactly opposite there is no chance of vertical crossover.

Case (b): As both robots move toward each other, the risk of collision is high. Imagine a reference line $A'B'$ where robots R_1 and R_2 are at endpoints A' and B'. R_1 is in $Explore_2$ has followed a path $A \rightarrow A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow B' \rightarrow A'$. If R_1 continues, it enters R_2's explored region. Similarly, R_2, in $Explore_1$-*UPB*, has followed $B \rightarrow B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow A' \rightarrow B'$, and continued movement would lead R_2 into R_1's explored area, risking repeated exploration and collision as their paths overlap i.e., $A' \rightarrow B'$ and $B' \rightarrow A'$. Figure 4(a) shows the scenario.

However, these Scenarios never happen because (a) vertical crossing requires both robots to be on a same horizontal line, which is impossible, and (b) both robots sense each other's presence before reaching line *A'B'* due to the *Lights* dropped along their paths. These *Lights*, visible from points like A_3 and B_3, allow robots to detect each other well before collision as shown in Fig. 4(b). Even if a robot sleeps, the other can follow a zigzag path to sense dropped *Lights*. Hence, the proof.

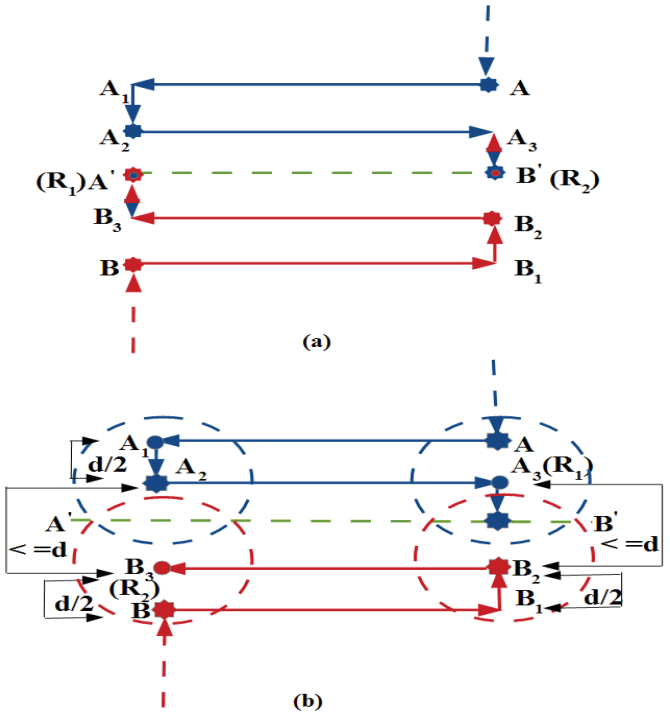

Figure 4. (a) R_1 crosses R_2 vertically and vice versa; and (b) R_1 never crosses R_2 vertically and vice-versa.

Case (B): Each robot computes the boundaries of its strip independently. A robot only enters the *PAINT* state after calculating both *LWB* and *UPB* in *EXPLORE* state. Once boundaries are set, robots stay within their strips and stop exploring. During *PAINT*, *R* covers the accessible area within its boundaries (*UPB* and *LWB*). Both robots work without interfering, preventing vertical crossovers. Even if one enters *sleep* state, *EXPLORE* and *PAINT* are atomic states, so one robot's action does not affect other.

Case (C): Both robots are in *PAINT* state, indicating they have determined the *UPB* and *LWB* of their strips. As *EXPLORE* and *PAINT* are independent, there is no interference between them. Since boundaries are fixed, no overlap occurs, and crossovers are avoided. The robots continue their tasks independently, even if one enters *sleep* state.

Lemma 2: The optimal distance that a robot should move in every computational cycle to ensure collision-free movements is $\frac{d}{2}$.

Proof: Robot movements involve both vertical and horizontal moves. If multiple robots align on the same horizontal or vertical line, or if one is on a horizontal line while another is on a vertical line, there is a high chance of collision. However, Lemma 1 ensures that during exploration, two robots never be on the same horizontal line, nor they cross each other vertically. Each robot moves in every computational cycle to avoid collisions. The minimum distance a robot moves per cycle depends on its visibility range, *d*. To maintain collision-free movement, the distance covered in each move, ∂, should be optimized based on *d*, with $\partial = \frac{d}{2}$. Keeping constant the value of *d*, different values of ∂ is taken to justify the optimality. Three different scenarios are taken into account and ∂ considered as $\frac{3d}{2}$, *d* and $\frac{d}{2}$.

Let's consider two robots say, R_1 and R_2 and it is also considered $V_{dist}(R_1, R_2)=d+h$, where $\frac{d}{2} < h \leq d$. This means R_1 *and* R_2 are located out of each other's visibility range and are vertically moving towards each other with ∂ moves.

Scenario (1)*:* $\partial=\frac{3d}{2}$.

Figure 5 (a) shows that R_1 is at *A* and R_2 is at *B* and both compute their destination points as *A'* and *B'* respectively. As $\partial > d$, so to reach *A'*, R_1 needs to cross R_2 and vice versa. This scenario never guarantees collision-free movement.

Scenario (2)*:* $\partial=d$.

If $h=d$, then two robots compute their destinations on the same point on their visibility circle. However, for $h<d$ they vertically cross each other as per *Scenario (1)*. So, this scenario also causes a collision. Figure 5(b) shows the scenario.

Scenario (3)*:* $\partial = \frac{d}{2}$

In above two scenarios, robots collide during their movements. Now, as $\partial<d$, robots R_1 and R_2 never cross each other to reach destination points *A'* and *B'* respectively. Hence, there is no collision during their movements. This is also applicable to all values where $\partial<\frac{d}{2}$. Figure 5(c) illustrates the scenario. From all the above-mentioned scenarios, it can be concluded that $\partial=\frac{d}{2}$, justifies the optimality.

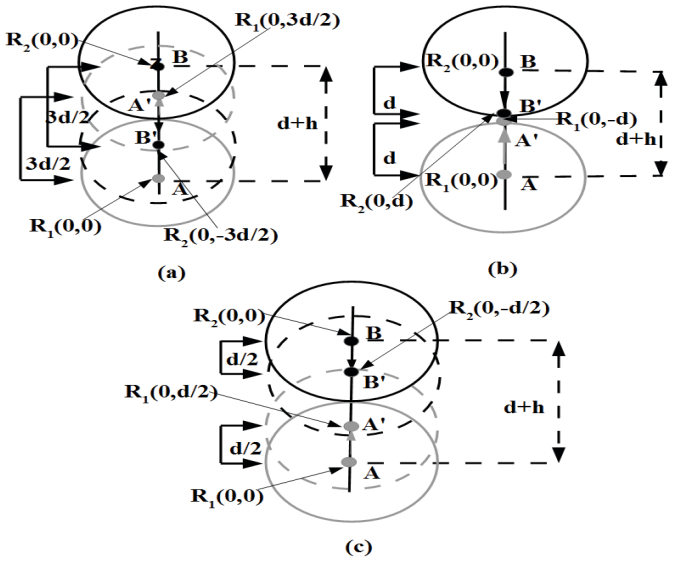

Figure 5. (a) Each computational cycle, a robot moves (a) $\frac{d}{2}$ distance (b) *d* distance (c) $\frac{3d}{2}$ distance.

Lemma 3: Throughout the exploration process a robot, R drops maximum of $2\times\left\lfloor\frac{L}{d}\right\rfloor+2$ Lights.

Proof: This work addresses the issue of limited visibility, with a fixed range $V_r=d$, by using *Light*s with a fixed illumination range that do not reset between cycles. Robots detect *Light*

from beyond their visibility range to identify other robots. Each robot drops *Lights* one at a time, with a maximum number dropped when starting at the upper boundary and *LFB*. When starting on the *LFB* and upper boundary, the robot performs two steps: *Explore*$_1$*-RTB* and *Explore*$_2$. During *Explore*$_1$*-RTB*, it drops up to two *Light*s along the same horizontal line. In *Explore*$_2$, it follows zigzag moves, dropping a *Light* each time it changes direction from vertical to horizontal, resulting in up to two *Light*s per zigzag move. To cover entire region *L*, the robot can drop up to $2\times\left\lfloor\frac{L}{d}\right\rfloor$ number of *Light*s.

So, to explore the whole region while ignoring the fractional values of $\left\lfloor\frac{L}{d}\right\rfloor$, the cumulative number of *Lights* required is represented as

$$2\times\left\lfloor\frac{L}{d}\right\rfloor+2 \quad (1)$$

To illustrate Eqn. (1), considering following scenario where $L=10$ units, and $\frac{d}{2}=1$ unit distance. Therefore, as per Eqn. (1), a robot drops maximum $=2\times\left\lfloor\frac{L}{d}\right\rfloor+2=\left(2\times\left\lfloor\frac{10}{2}\right\rfloor+2\right)=12$ *Lights*.

There remains another scenario in which *R* is on *LFB* but not on upper and lower boundaries of region.

Consider, $L=L_1+L_2$, where L_1 and L_2 are the vertical heights of the two strips respectively. So, during the whole process maximum number of *Lights* dropped by *R* is: *Lights* dropped in (*Explore*$_1$*-RTB*+*Explore*$_1$*-UPB*+ *Explore*$_2$)

$$=2+\left[\left(2\times\left\lfloor\frac{L1}{d}\right\rfloor+2\right)+\left(2\times\left\lfloor\frac{L2}{d}\right\rfloor+2\right)\right].$$

As $L> L_1$ and L_2, so always $\left\lfloor\frac{L}{d}\right\rfloor>\left\lfloor\frac{L1}{d}\right\rfloor+\left\lfloor\frac{L2}{d}\right\rfloor$.To illustrate, consider the same region with $L_1 = L_2 = 5$ units and $\frac{d}{2}=1$ unit. To explore whole region, *R* drops *Lights,*

$$=2+\left[\left(2\times\left\lfloor\frac{L1}{d}\right\rfloor+2\right)+\left(2\times\left\lfloor\frac{L2}{d}\right\rfloor+2\right)\right]$$

$$=2+\left[\left(2\times\left\lfloor\frac{5}{2}\right\rfloor+2\right)+\left(2\times\left\lfloor\frac{5}{2}\right\rfloor+2\right)\right]=2+4+4=10$$

Considering, $L_1=3$ and L_2=7 units and $\left\lfloor\frac{d}{2}\right\rfloor=1$ unit, the total number of *Lights*

$$=2+\left[\left(2\times\left\lfloor\frac{3}{2}\right\rfloor+2\right)+\left(2\times\left\lfloor\frac{7}{2}\right\rfloor+2\right)\right]=2+2+6=10.$$

So, it is observed that a robot positioned anywhere between upper and lower boundaries drops fewer *Lights* compared to when it is positioned on either upper or lower boundary.

So, the maximum number of *Lights* dropped throughout the execution is $2\times\left\lfloor\frac{L}{d}\right\rfloor+2$.

Lemma 4

The painting of the entire target region is completed within a finite amount of time.

Proof: The algorithm specifies that each robot can paint up to two consecutive strips. Occasionally, a robot might only paint one strip or none, with different termination processes for each case:

(i) Robots with no specific painting area

If robot *R* is positioned at *LFB* along lower boundary and detects objects above, it has no region to paint and moves to *RTB*, terminating after completing *Explore*$_1$*-RTB*.

(ii) Robots with two consecutive strips for painting

Knowing $UPB=V_{dist1}$ or $UPB=\infty$ and $UPB_{Exp2}=\infty$ or $UPB_{Exp2}=V_{dist2}$, *R* moves horizontally along *LWB* towards *RTB* by setting *LWB=0*, terminating *Explore*$_1$*-RTB*. It then moves upward to determine *UPB* during *Explore*$_1$*-UPB*, after which it enters *Paint*$_1$. Once first strip is painted, *R* calculates UPB_{Exp2} of second strip (*Explore*$_2$), then paints the second strip during *Paint*$_2$. If UPB_{Exp2} is pre-determined, *R* starts painting from *LWB* to UPB_{Exp2}. After painting both strips, *R* successfully terminates.

(iii) Robots with only one strip for painting

Below mentioned are such scenarios:

Scenario 1: A robot is located on the upper boundary with or without knowledge of UPB_{Exp2}.

Scenario 2: A robot is located anywhere on *LFB* where *LWB=0*, *UPB=0*, with/ without knowledge of UPB_{Exp2}.

Scenario 3: A robot is located on lower boundary of region without knowledge of *UPB*.

In all scenarios, *R* explores and paints one strip above/ below, following same execution and termination order as in (ii). For *Scenario 1* and *Scenario 2*, *R* performs *Paint*$_2$ or *Explore*$_2$*+ Paint*$_2$. For *Scenario 3*, the robot performs *Explore*$_1$*-UPB* and *Paint*$_1$.

EXPLORE and *PAINT* phases are atomic; once strip boundaries are set, painting is uninterrupted. Robots complete tasks in finite time, working autonomously to paint entire region.

5. CONCLUSION

This work introduces a distributed algorithm for a swarm of oblivious, silent robots to paint a bounded rectangular area with opaque horizontal line obstacles. Using a *full-compass* and *asynchronous* timing model, the algorithm guarantees collision-free, non-repetitive coverage and finite-time completion despite limited visibility. Future research could extend this to polygonal obstacles and direction-only models.

REFERENCES

1. Tan Y, Zheng Z. Research advance in swarm robotics. Defence Technology. 2013; 9(1), 18–39. doi: 10.1007/s10472-009-9120-2
2. Shahzad M, Saeed ZM, Akhtar A, Munawar H, Yousaf MH, Baloach NK, Hussain F. A review of swarm robotics in a NutShell. Drones. 2023;*7*(4), 269–297. doi:10.3390/drones7040269
3. Alqudsi Y, Makaraci M. Exploring advancements and emerging trends in robotic swarm coordination and control of swarm flying robots: A review. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science. 2025; 239(1), 180–204. doi:10.1177/09544062241275359
4. Mandal B, Sardar M, Das, D. Swarm Coverage in Continuous and Discrete Domain: A Survey of Robots' Behaviour. In: volume 653: Advances in Data-driven

Computing and Intelligent Systems 2022. Singapore: Springer Nature; 2023 [cited 2025 May 6]. p. 573–587. Available from: doi:10.1007/978-981-99-0981-0_44
5. Tokas P, Das D. A distributed algorithm for assembling of asynchronous swarm of mobile robots with limited visibility in presence of horizontal line obstacle. In: 2nd International Conference for Convergence in Technology (I2CT) 2017. Mumbai, India: IEEE xplore; 2017 [cited 2025 May 6]. p. 426–429. Available from: doi: 10.1109/I2CT.2017.8226165
6. Mandal B, Sardar M, Das D. Assembling of Swarm Robots in Distributed Environment in the Presence of Polygonal Obstacles. In: Proceedings of the 6th International Conference on Advances in Robotics 2023. IIT Ropar, India: ACM Digital Library; 2023 [cited 2025 May 6]. p. 1–6. Available from: doi: 10.1145/3610419.3610504
7. Prencipe G. Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. In: Volume 2202: *ICTCS* 2001. Springer; 2001 [cited 2025 May 6]. p. 154–171. Available from: doi: 10.1007/3-540-45446-2
8. Das D, Mukhopadhyaya, S. Distributed painting by a swarm of robots with unlimited sensing capabilities and its simulation. In: International Journal on Information Processing. 2013; 7(3): 1–15. doi: 10.48550/arXiv.1311.4952
9. Das D, Mukhopadhyaya S, Nandi D. Swarm-based painting of an area cluttered with obstacles. International Journal of Parallel, Emergent and Distributed Systems. 2021; 36(4): 359–379. doi: 10.1080/17445760.2021.1879071
10. Das D, Mukhopadhyaya S. Distributed algorithm for painting by a swarm of randomly deployed robots under limited visibility model. International Journal on Information Processing. 2018; 15(5): 17298814–18804508. doi: 0.1177/1729881418804508
11. Alqudsi, Y. Integrated Optimization of Simultaneous Target Assignment and Path Planning for Aerial Robot Swarm. The Journal of Supercomputing. 2025; 81(1): 1–24. doi:10.1007/s11227-024-06620-w
12. Adams JA, Hamell J, Walker P. Can a single human supervise a swarm of 100 heterogeneous robots?. IEEE Transactions on Field Robotics. 2025; (2), 46–80. doi: 10.1109/TFR.2024.3502316
13. Latimer D, Srinivasa S, Lee-Shue V, Sonne S, Choset H, Hurst A. Towards sensor based coverage with robot teams. In Volume 1: International Conference on Robotics and Automation 2002. USA: IEEE; 2002 [cited 2025 May 6]. p. 154–171. Available from: doi: 10.1109/ROBOT.2002.1014684
14. Kong CS, Peng NA, Rekleitis I. Distributed coverage with multi-robot system. In: International Conference on Robotics and Automation 2006. ICRA IEEE; 2006 [cited 2025 May 6]. p. 2423–2429. doi: 10.1109/ROBOT.2006.1642065
15. Rekleitis I, New AP, Rankin ES, Choset H. Efficient boustrophedon multi-robot coverage: An algorithmic approach. Annals of Mathematics and Artificial Intelligence. 2008; 52: 109–142. doi: 10.1007/s10472-009-9120-2
16. Karapetyan N, Benson K, McKinney C, Taslakian P, Rekleitis I. Efficient multi-robot coverage of a known environment. In International Conference on Intelligent Robots and Systems (IROS) 2017. IEEE/RSJ; 2017 [cited 2025 May 6]. p. 2423–2429. Available from: doi: 10.1109/IROS.2017.8206000
17. Hungerford K, Dasgupta P, Guruprasad K. Distributed, complete, multi-robot coverage of initially unknown environments using repartitioning. In: International conference on Autonomous agents and multi-agent systems 2014. ACM; 2014 [cited 2025 May 6]. p. 1453–1454. doi:10.5555/2615731.2617519
18. Viglietta G. Rendezvous of two robots with visible bits. In: Algorithms for Sensor Systems 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2013. Berlin, Heidelberg: Springer; 2013 [cited 2025 May 6]. p. 291–306. Available from: doi:10.1007/978-3-642-45346-5_21
19. Das S, Flocchini P, Prencipe G, Santoro N, Yamashita M. Autonomous mobile robots with lights. Theoretical Computer Science. 2016; 609(1), 171–184. doi:10.1016/j.tcs.2015.09.018
20. Takashi O, Koichi W, Yoshiaki K. Optimala synchronous rendezvous for mobile robots with lights, arXiv:1707.04449 2017. 2017 [cited 2025 May 6]. p. 1–15. Available from: doi: 10.48550/arXiv.1707.04449

CONTRIBUTORS

Ms Banashree Mandal is a PhD scholar in Department of CSE at NIT Durgapur. Her research interests include: Swarm robotics, distributed algorithms.
In the current study she contributed to the conceptualization, implementation of the algorithm, and correctness proof analysis, as well as writing the final manuscript.

Dr Deepanwita Das is an Assistant Professor at NIT Durgapur. Her research interest include: Swarm robotics, distributed algorithms.
In the current study she verified the algorithm and thoroughly reviewed and revised manuscript as the main supervisor.

Dr Niladri Mandal is a Scientist 'F' at DRDO-DRDL, Hyderabad. His research interest include: Hybrid manufacturing technologies, non-conventional manufacturing, artificial intelligence, and automation.
In the current study he contributed to the overall analysis and provided insights for developing various aspects of the algorithm.