A Swarm Based Area Partitioning Technique by Robots With Limited Capabilities and its Simulation

James A S[a,1], Deepanwita D[a,1], Sruti K[a,1]

[a]*Department of Information Technology, National Institute of Technology Durgapur.*

Abstract

This paper presents a novel approach for area division of an unknown polygon by a team of robots. We assume initially every robot is connected by a visibility graph. Here, we use concept of convex hull to divide the area. Our algorithm guarantees non-overlapping division of the area into N parts, where N is the total number of robots in the system. In the proposed algorithm robots follow observe-compute-move model.

Keywords: Area Partition, Robot Swarm, Multi-Robot System, Distributed Algorithm.

1. Introduction

Area partition problem is an important problem in the field of multi-robot systems. Whenever we want to use a team of robots to perform any task say sweeping, painting, exploration etc, we always need to partition the area and assign it to the robots. It's basically a task division or a task allocation problem. Our main goal is to generate a map of an unknown area with obstacles. This problem is divided into two subproblems as *area partition* and *area exploration problem*. The work we present in this paper is a part of our main problem. In this paper we address the area partition problem and to solve this we use concepts from computational geometry.

The basic idea of our solving approach is, the team of robot form a convex hull [1] and calculates it's center, then with reference to the center the area is partitioned. Each of the robot in the team unanimously agrees on that partition. This algorithm results a completely non overlapped area partitions. When each of the robot is assigned with an area they perform area exploration but here we addressed area partition only.

Area partition problem being an important problem in the field of multi-robot systems, significant amount of contribution have been done to solve the problem. Here we survey few of them. Markus et. al. [2], used grid to perform

[1]Department of Information Technology, National Institute of Technology Durgapur, West Bengal-713209.

the partition of the area, knowledge of the area is known a priori and its a centralized way as robots does not execute the area division algorithm. They have overlaid the area with a grid and each of the grid cells divide the area into small polygons. Each robot uses an allocation mechanism with which they allocate themselves a polygon. If more than one robot try to allocate the same polygon, then they communicate among themselves and agree upon one solution. Though the allocation mechanism is a decentralized one but the partition of the polygon is centralized. Mazda et. al. [3] also used a centralized concept of grid to divide the area into grid cells for sweeping. A robot take certain action to reach to next state(grid cell), for every action a cost function is defined. So a robot try to find a policy(set of actions), so that the overall cost function is minimized. Upma et. al., [4] presents circle partitioning method to divide the area, without mentioning anything about the calculation of the center and the radius.
Most of the works studied uses centralized partitioning techniques. In this paper we propose a completely distributed algorithm, in which every robot independently execute the algorithm and finally divides the unknown polygon into non overlapping partitions.

Rest of the section is organized as follows, section 2 describes problem definition and models of the robot. Section 3 presents the proposed algorithm. In section 4 we report our simulation results and finally section 5 presents conclusion and future works.

2. Problem Definition, Assumptions and Models

Our problem is to divide an unknown area with obstacles into N number of non-overlapping parts using a distributed algorithm, where N is the total number of robots in the system.
Robots are assumed to be distributed across the area with one robot visible to at least one other robot i.e, they form a visibility graph. Every robot is color coded and the information of color codes is available to every robot. They use CSMA/CA as media access protocol for Wireless Communication. It is assumed that no message is lost in the network i.e, every sent message is received by the receiver.

2.1. Characteristics and Models of robot [5]:

Identical and Homogeneous: All robots are identical and have the same computational ability.
Autonomous: Every robot work asynchronously and independently in a completely distributed manner. No central authority.
Computational Model: Here we follow the basic *Observe-Compute-Move* [6] model. A *computational cycle* is defined to be a sequence of "observe", "compute" and "move" steps. Each of the robots executes same instructions in all the computational cycles. The actions taken by a robot in *compute* and *move* steps, entirely depend on the observations made in *observe* step.

Limited Visibility and Communication: Robots have limited visibility. Each robot can view only those robots which are located within a distance d from it. So, the area of visibility for each robot can be measured by a circle with radius d and the robot is located at the center of the circle. This circle is known as the *circle of visibility* and d is known as the *visibility radius*. Communication is also limited, every robot also has a communication radius, communication is based on line of sight. If a robot is visible to another robot then they can communicate. In our case *communication radius* is same as *visibility radius*.
Non-Oblivious: Every robot have some amount of memory and can retain the information gathered in the previous computational cycle.
Visibility Graph: Based on the relative positions of the robots in the region, we define a visibility graph $G = (V, E)$. The vertices in the graph correspond to the robots in the swarm. Two nodes in the graph are connected by an edge iff the corresponding robots are mutually visible as shown in Fig. 2(b).
Robots are assumed to follow the following models [5]
Full-compass: Direction and orientation of both axes are common to all the robots. Each robot has its local co-ordinate system. All the robots would assume that they occupy the position $(0, 0)$ with respect to their local co-ordinate system.
Asynchronous Model: Every robot in the system operates independent computational cycle of variable length. They do not share a common clock.

3. Proposed Algorithm

Before discussing solution to the *area partition problem*, lets first discuss the communication module of robots i.e, how robots communicate with each other. In our scenario, no robot has the coordinates of each and every robot in the system or team. And coordinates of every robots is required for finding the convex hull. Thus each robot gets this information through communication.

3.1. Communication Module

Before starting communication module lets first discuss a lemma on coordinate translation.

3.1.1. Lemma 1 : Coordinate Translation

Since every robot in the team has it's own local coordinate system, so whenever a robot receives coordinate from other robots, those data would be w.r.t the coordinate system of the sending robot thus before using it, receiving robot should translate it into it's own coordinate systems.
In Fig. 1(a), Robot 2 is not visible to robot 4 thus cannot communicate. So robot 2 will be receiving robot 4's coordinate through robot 3. As you can see in Fig. 1(b) Coordinate of R_4 w.r.t R_3 i.e, $R_{43} = (2, 4)$ and coordinate of R_3 w.r.t R_2 i.e, $R_{32} = (3, -3)$. Now the coordinate of R_4 w.r.t R_2 i.e, R_{42} is calculated as: $R_{42} = R_{43} + R_{32}$, therefore $R_{42} = (2+3, 4-3)$, So, $R_{42} = (5, 1)$. Here, R_{ij} represent the coordinate of robot i calculated w.r.t robot j.

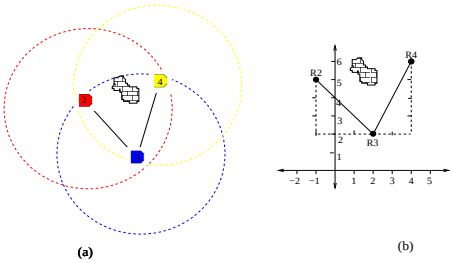


Figure 1: Coordinate Translation

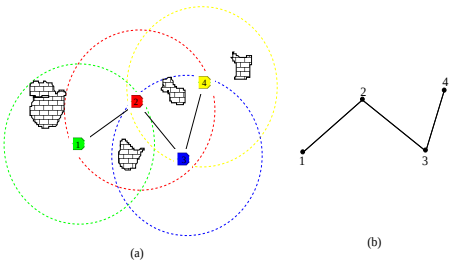


Figure 2: Robot Communication

3.1.2. Definitions

- R : Set of Robot in the system. R = $\{R_i : i \in [1, n]\}$.
- N_{R_i} : Set of Robots that have neighbors of R_i. $N_{R_i} = \{R_j : R_j \in R\}$.
- M_{R_i} : Set having coordinates of other robots. $M_{R_i} = \{R_j < x, y >: R_j \in R\}$.
- C_{R_i} : Vector having index of all the robots in the system along with a flag V_j with value either 0 or 1
  $C_{R_i} = \{< R_j, V_j >: R_j \in R\}$. e.g $C_{R_i} = \{< R_j, 1 >, < R_k, 0 >\}$
  Here $< R_j, 1 >$ means coordinates of R_j is available to R_i
  and $< R_k, 0 >$ means coordinates of R_k is not available to R_i.
- MQ_{R_i} : Message Queue of Robot R_i
  Every robot has a Message Queue which stores the raw data received from other robots, which is then translated into R_i's coordinate system and then stored into M_{R_i}.
- **Sense**(): *Observe*
  R_i sense the environment with the sensors attached and gets the coordinates of neighboring robots. Then R_i updates M_{R_i}, C_{R_i}
  Algorithm :
  Line 1. $N_{R_i} \leftarrow \{R_j | \exists j, R_j \in R\}$
  Line 2. $M_{R_i} \leftarrow \{R_j < x, y > | \forall j, R_j \in N_{R_i}\}$
  Line 3. For all j, $R_j \in M_{R_i}$
  $C_{R_i} \leftarrow < R_j, 1 >$
- **Send**(M_{R_i}) :
  R_i broadcasts the coordinates of other robots present in M_{R_i}.
- **Receive**():
  R_i Extracts data from Message Queue MQ_{R_i} and translate it to it's own coordinate system using Lemma 1 then store it in M_{R_i}.
  Line 1. $M_{R_i} \leftarrow$ **Translate**(MQ_{R_i})
  Line 2. **Return** M_{R_i}

Table 1: Algorithm 1 : Communication Module executed by every robot

1.**Sense()**
2.**While**(V_j == **1,** $\forall$ **j,** $V_j \in C_{R_i}$)
3. **Send**(M_{R_i})
4. $M_{R_i} \leftarrow$ **Receive()**
5. **Compute()**
6.**Send**(M_{R_i})
7.Execute Area Partitioning Algorithm

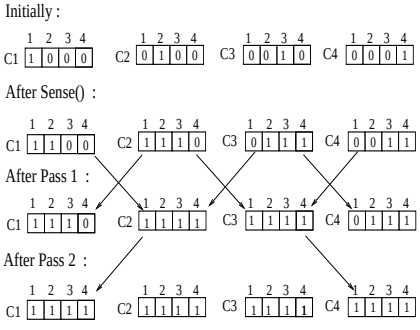


Figure 3: Execution of Communication Module

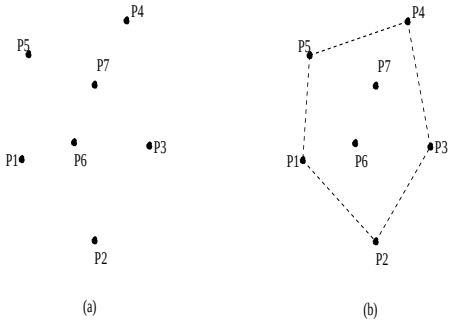


Figure 4: Convex Hull Formation

- **Compute() :** Updates the C_{R_i}
  for all j, $R_j \in M_{R_i}$
  $V_j \leftarrow 1$
  $C_{R_i} \leftarrow < R_j, V_j >$

3.1.3. Algorithm 1: Case Study

As we can see in Fig. 2(a), we have four robots R_1, R_2, R_3 and R_4. Corresponding visibility graph of the scenario is shown in Fig. 2(b). As we can see from the figure visibility is restricted. In order to get the coordinates of other robots which are not visible, every robots executes communication algorithm (**Algorithm 1**) in Table 1.

Here we execute the algorithm w.r.t robot 1, for R_1, $C_{R_1} \approx C_1$. Initially before Sense() operation content of vector C_1 is shown in figure 3. C1 = [1,0,0,0].

Line 1. After executing sense() operation, R_1 discovers one new robot, red robot(R_2). The color code information is available to all the robots. R_1 updates C1 = [1,1,0,0].

Line 2. Pass 1, R_1 executes the while loop.

Line 3. R_1 broadcasts the data i.e, M_{R_1} to the network (Send()).

Line 4. R_1 updates M_{R_1} with the data received from R_2 i.e, C2 = [1,1,1,0] coordinate information of R_3.

Line 5. R_1 updates C1 = [1,1,1,0].

Line 2. Pass 2, R_1 checks whether C1 has all the vectors = 1, which is false as Coordinate information of R_4 is not available yet, so R_1 enters the loop again.

Line 3. R_1 broadcasts the data i.e, M_{R_1} to the network (Send()).
Line 4. R_1 updates M_{R_1} with the data received from R_2 i.e, C2 = [1,1,1,1] coordinate information of R_4.
Line 5. R_1 updates C1 = [1,1,1,1].
Line 2. Now as R_1 has all the vectors = 1, so R_1 comes out of the loop.
Line 7. R_1 broadcasts the data i.e, M_{R_1} to the network (Send()).
Every other robots in the system executes the same algorithm and finally receives the coordinates of every robot in the system.

3.2. Area Partition Module

We have an unknown area for which we need to make non overlapping partitions. Say N number of robots in the system. We divide the unknown area into N parts and assign each robot a part. Before going to the algorithm of area partition, lets first discuss a lemma for finding the center of a polygon(convex).

3.2.1. Lemma 2 : Center Calculation

Let we have a polygon with n points $p1(p1_x,p1_y)$, $p2(p2_x,p2_y)$....., $pn(pn_x,pn_y)$ and C(C_x, C_y) is the center of the polygon.
Some Definitions :
d_{ij} = Distance between p_i and p_j.
$m_{(i,j)x}$ = x coordinate of p_i and p_j's mid point
$m_{(i,j)y}$ = y coordinate of p_i and p_j's mid point

$$C_x = \frac{(\sum_{i=1}^{n} m_{(i,i+1)x} * d_{i,i+1}) + m_{(n1)x} * d_{n1}}{(\sum_{i=1}^{n} d_{i,i+1}) + d_{n1}} \qquad C_y = \frac{(\sum_{i=1}^{n} m_{(i,i+1)y * d_{i,i+1}}) + m_{(n1)y} * d_{n1}}{(\sum_{i=1}^{n} d_{i,i+1}) + d_{n1}}$$

3.2.2. Lemma 3 : Finding Convex Hull

- Convex Hull [1] : Convex Hull of a set Q of points in the Euclidean plane or Euclidean space is the smallest polygon formed by smallest set of points that contains all the points in Q. As we can see in Fig. 4(a) and 4(b).

- Algorithm for finding Convex Hull is given by Ronald Graham, Graham's Scan [7].

3.2.3. Algorithm 2: Case Study

As we see in figure 5(a) we have an unknown area with obstacles. we have a team of seven robots distributed across the area. Fig. 5(b) shows visibility graph.
Robot R_i is executing the area partition algorithm 2 in Table 3.
Line 1: Coordinates of robots obtained after communication is present in Q.
Line 2-3: R_i executes Convex Hull Algorithm (Graham's Scan) on Q and stores the coordinate of robots which are vertex of convex hull in V_{convex} and coordinates of those robots which are not in convex hull set are stored in V_{inside}. Fig. 6(a) shows convex hull formed with five vertices or five robots out of those seven robots.

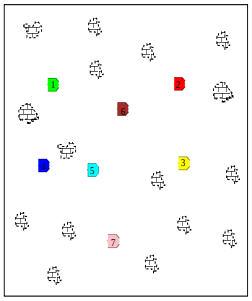
Table 2: **Parameters**

Parameters	Descriptions
R_i	Robot with id 'i'
Q	Set of Coordinates of all the robots which are obtained after Communication
V_{convex}	Set of Coordinates of robots which are the vertex of Convex Hull
V_{inside}	Set of Coordinates of robots which are inside the Convex Hull
V'_{inside}	Set of Final Coordinates of robots which were inside the Convex Hull
$insideFlag$	Flag which determines whether a robot is inside a convex hull or not
P_{left}	Nearest Coordinate of robot left to R_i
$findAdjacent_{left}$	Finds Nearest Coordinate of robot left to R_i
Eqn_{left},Eqn_{right}	Equation of lines which are used as a virtual wall
R_{ix},R_{iy}	Coordinate of Robot R_i
C_x,C_y	x and y coordinates of center of the convex hull

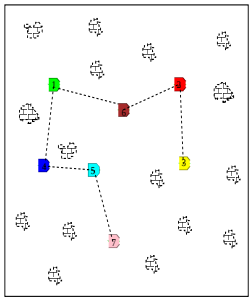
Table 3: Algorithm 2 : Area Partition

1.Q ← getCoordinates()
2.V_{convex} ← Graham's Scan(Q)
3.V_{inside} ← Q - V_{convex}
4.**if**(Coordinate (0,0) ∈ V_{convex})
5. insideFlag ← False
6.**else**
7. insideFlag ← True
8.**if**(insideFlag is True)
9. V'_{inside} ← Mid pt. of nearest side of the hull
10. Move to Mid pt. of nearest side of the hull
11.V'_{inside} ← V'_{inside} ∪ Calc.final pts of rbts inside
12.C ← Calculate Center() //*Using Lemma* 3
13.P_{left} ← $findAdjacent_{left}(V_{convex} \cup V'_{inside})$
14.P_{right} ← $findAdjacent_{right}(V_{convex} \cup V'_{inside})$
15.Eqn_{left} ← $getEquation(C,\text{mid}(P_{left},R_{ix}))$
16.Eqn_{right} ← $getEquation(C,\text{mid}(P_{right},R_{iy}))$

Line 4-7: As each robot has it's own local coordinate system, so if R_i is at the vertex of the convex hull then (0,0) must be in V_{convex}, if it belongs then insideFlag is False, which means Robot is not inside the hull, other wise insideFlag is True.
Line 8-10: If R_i is inside the convex hull, then R_i moves to the mid point of nearest side of the convex hull. Then R_i inserts it's new coordinate to V'_{inside}. Fig. 6(b) shows robots inside the hull moved to the mid point of nearest side of the hull.
Line 11: R_i calculates final point of all the robots inside the convex hull and stores it in V'_{inside}.
Line 12: R_i calculates center of the hull using Lemma 3.
Line 13-14: R_i finds the coordinates of nearest robots to the left and right as P_{left} and P_{right} respectively.
Line 15-16: R_i calculates equation of line which behaves as a virtual wall with points (C and P_{left}) for left side and with (C and P_{right}) for right side. Fig. 7(a) shows the virtual wall represented by the equation of the line which partition

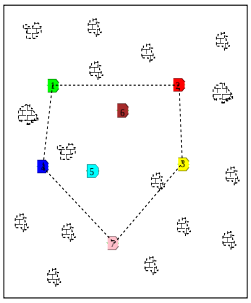


(a) Unknown Area

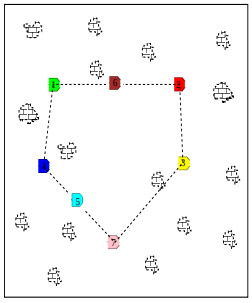


(b) Visibility Graph

Figure 5: Unknown Area

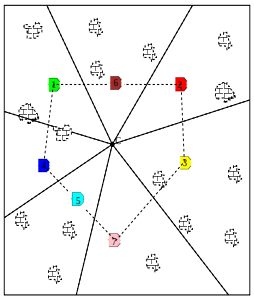


(a) Without inside robots

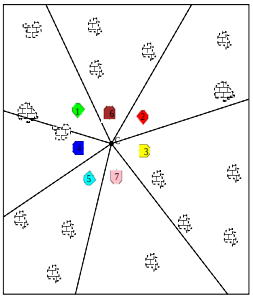


(b) With inside robots

Figure 6: Convex Hull



(a) Area Partition and Assignment



(b) Area Exploration

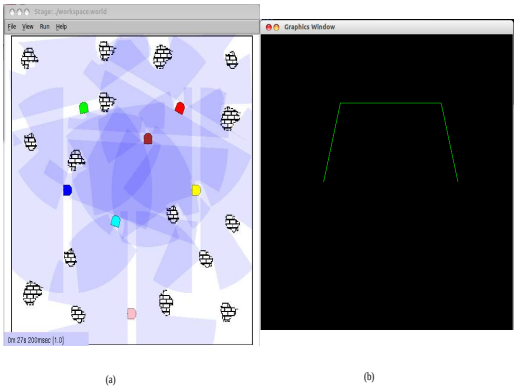
Figure 7: Partitioned Area

the area. Every robot has it's left wall and a right wall. In fig 7(b) every robot converge towards center and starts executing exploration of the assigned area.

4. Simulation Results

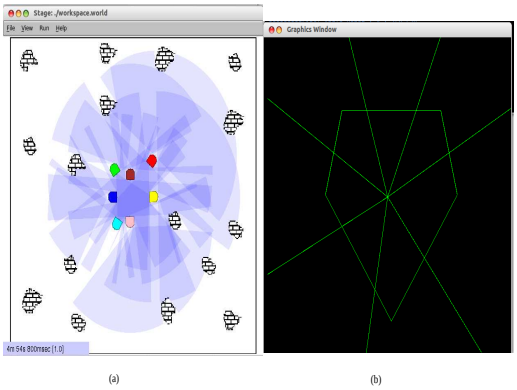
The simulation is conducted using the Player/Stage [8] multi-robot simulation software. The Player (Version 3.2.2) and Stage (version 3.0.2) softwares is configured on Ubuntu 12.04 LTS with support of Intel Corei3 Processor with 3.00 GHz speed and 2.00 GB RAM. The robots used in this simulation are of eight different colors and having (1unit x 1unit x 1unit) dimension. The robots are equipped with the following devices:

- Infrared Laser sensors: Attached at the left and right side of the robot with sensing range upto 10 units, 180 scan lines and 180 field of view and 180 samples.
- The Blobfinders: Attached at the left and right side of the robot and capable to recognize all the eight colors image of size 160 x 120 square unit. It has a sensing range of 10 unit and 180 field of view.



(a) (b)

Figure 8: Initial Scenario



(a) (b)

Figure 9: Final Scenario

The proposed algorithm is implemented in C++ programming language, using the library libplayerc++ to communicate with the Stage.
The display Graphic Window is implemented in python programming language using the library graphic.py [9] which simply reads data file of every robot and generate the graphics.

Fig. 8(a) shows initial position of the robots, in Fig. 8(b) convex hull is starting form and in Fig. 9(a) robots have gathered towards center of the hull and ready for *area exploration*. In fig 9(b) convex hull is formed and non-overlapping partitions are created.

5. Conclusions and Future Works

In this paper, we proposed an algorithm which completely divides an unknown area into total number of robots in the system. The algorithm also ensures the non overlapping partition of the area. For future work, we would like to develop an algorithm for area exploration so that every individual robot could explore the assigned area and generate the map. Thus combining each part of the map we finally get the complete map of the area.

References

[1] http://en.wikipedia.org/wiki/Convexhull

[2] Jager, M. and Nebel, B., "Dynamic decentralized area partitioning for cooperating cleaning robots," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on (Volume:4)*, 2002, 3577 - 3582 vol.4.

[3] Mazda A. and Peter S., "A Multi-Robot System for Continuous Area Sweeping Tasks," in *Learning and Adaption in Multi-Agent Systems, Springer Berlin Heidelberg*, 2006.

[4] Jain, U. and Tiwari, R. and Majumdar, S. and Sharma, S., "Multi Robot Area Exploration Using Circle Partitioning Method," in *International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012)*, Procedia Engineering 41 (2012) 383 387.

[5] Das, D. and Mukhopadhyaya, S, "Painting an Area by Swarm of Mobile Robots with Limited Visibility," in *Wireless Networks and Computational Intelligence Communications in Computer and Information Science Volume 292* , 2012, pp 446-455.

[6] Das, D. and Mukhopadhyaya, S, "An Algorithm for Painting an Area by Swarm of Mobile Robots," in *Int. Conf. on Control, Automation and Robotics, Singapore*, 2011, pp. C1C6.

[7] Graham, R.L, "An Efficient Algorithm for determining the convex hull of a finite planar set," in *North-Holland Publishing Company*, 1972.

[8] $http://en.wikipedia.org/wiki/Player_Project$

[9] $http://mcsp.wartburg.edu/zelle/python/graphics.py$