Generic Shape Formation Algorithm for a System of Distributed Robots

Chinmay C (2nd year ECE, RV College of Engineering, Bangalore)
R Hariharan (3rd year CSE, PES University, Bangalore)
Mentors:
Ms. Rutuja (ERTS Lab, IIT Bombay, Mumbai)
Ms. Deepa (ERTS Lab, IIT Bombay, Mumbai)

Abstract:

In swarm robotics one of the swarm behaviours is to form patterns and shapes. The algorithm that we are proposing has the capability to form some regular shapes, alphabets and even numbers. The algorithm defines shapes as defined in a 14 segment display. Based on this input each individual robot finds its corresponding position in shape the user desires.

INPUT:

Each shape is defined by a 14 segment display

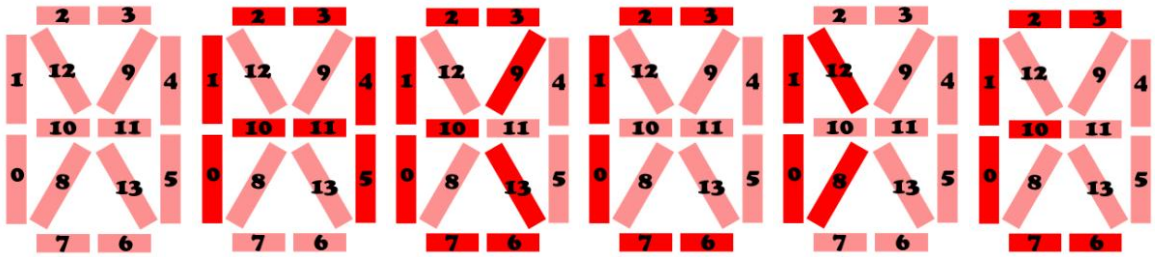

Fig 1. Figure represent shapes A, B, C, D & E formed from the 14 segment display

The 14 segment display can understood like a Boolean array of size 14 (indexes represented in the figure). True representing the segment is ON, False representing segment in OFF. The segments are represented as a pair of vertices. Each vertex is a co-ordinate in a global co-ordinate system. The intersection of segments '0' and '7' is the origin and segment '0' lies on the positive y-axis. With respect to this co-ordinate system and the scale factor the other vertex co-ordinates are established. The scale factor is used to define the number of robots preset to make the shape. To make the shape scalable we can change this variable. So, the input to the algorithm is the shape information as Boolean array of size 14 and the co-ordinates of all the vertices.

Algorithm:

We now know that shapes are represented as segments and segments have vertices. So, in our algorithm the robots fill these vertices. The points the robots fill can be broadly classified as Main vertices, Fill/Secondary vertices and Fill Edge points (Represented as green boxes, blue boxes and white circles in figure 2). The main vertices are the minimum number of vertices to be filled such that the shape can be represented. The fill/secondary vertices are the next order vertices which are filled to represent the shape better after all the main vertices are filled. The fill edge points are filled after by the extra robots which do not have place in either main vertices or fill/secondary vertices. So, the order of filling the shape is Main vertices – Fill/Secondary vertices – Fill edge points.

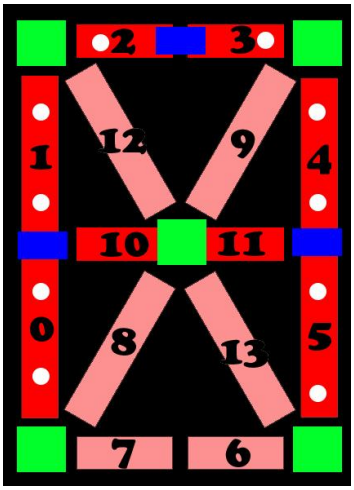

Fig 2. Shape 'A' which shows Main Vertices (Green Boxes), Fill/Secondary vertices (Blue Boxes), Fill Edge points (White circles)

With this hierarchy established, the algorithm first generates the main vertices, fill/secondary vertices and the fill edge points. The robots get their position w.r.t the global co-ordinate system and moves to the nearest main vertex. If the main vertex is filled by another robot then it moves to the next nearest main vertex, this loop runs until it finds a free main vertex or all main vertices are filled. If the main vertices are filled then the robot tries to fill the fill/secondary vertices. If both the main and secondary vertices are filled then the robot fills the fill edge points. If there is no spot available for the robot, it moves out of the shape. The method used to check if the nearest vertex is available or not is based on the robot model. In our implementation of the algorithm, the robot went to each vertex and checked if the spot is available or not, using IR proximity sensor. Other ways of finding out the same may be to use some kind of a communication system (ZigBee).

The flow chart is available in the Appendix.

Complications in algorithm:

The algorithm depends on two factors, 1) while moving to a vertex, how avoid other robots? And 2) when multiple robots are approaching the same vertex, how to break the tie? These can itself be researched to find the best algorithm that we can use. So choosing the best algorithm for the both improves the efficiency of our algorithm.

Future Improvements:

The algorithm as of now is not scalable, as in the number of robots that can fill the shape is fixed and the extra robots must leave the shape. Hence one of the improvements that can be made is to make the shape dynamically scalable. One of the ways to do it is by increasing the number of fill edge points, by increasing the scale factor dynamically. Scale factor determines the vertices of all the segments, so changing it dynamically can scale the size. But the challenge here is to make all the distributed robots agree upon the new scale factor. Another improvement is to make the algorithm work in a relative co-ordinate system environment. Each robot now has its own perspective of the co-ordinate system.

Conclusion:

The proposed algorithm will form some regular shapes (triangle, rectangle & square), all alphabets and all digits. We can even increase the number of segments to show more shapes. When the number of robots forming the shape is less the shape is formed fast, but the shape cannot be

purely identified. As the number of robots increase the two factors we talked about earlier starts to dominate and the time taken to form the shape reduces. So, finding the best dynamic robot avoidance algorithm and tie breaking algorithm for distributed robots plays a crucial role.

Appendix:

Link to the implantation code in V-REP simulation - https://github.com/eYSIP-2017/eYSIP-2017_DistributedRobotics/tree/master/Codes/2017-7-05/V-REP

Flow Chart