A Practical Framework for Uniform Circle Formation by Multiple Mobile Robots

Avinash Gautam
Department of Computer Science and Information Systems
Birla Institute of Technology and Science
Pilani, India
avinash@bits-pilani.ac.in

Sudeept Mohan
Department of Computer Science and Information Systems
Birla Institute of Technology and Science
Pilani, India
sudeeptm@bits-pilani.ac.in

Janardan Prasad Misra
Department of Computer Science and Information Systems
Birla Institute of Technology and Science
Pilani, India
jpm@bits-pilani.ac.in

Abstract—**This paper gives a software framework for positioning multiple mobile robots in a circular formation. The robots are initially arbitrarily placed on a 2D plane. The definition of the circle formed by these robots is governed by two strict rules (a) all robots should be positioned on the circle circumference (b) all robots should be uniformly placed on circle circumference. The software framework proposed in this paper utilizes two fundamental design patterns the decorator and the observer. Each robot is assigned a unique identity, which is used for conflict resolution in various situations. The model treats the environment as a first class entity providing partial support to the robots for circle formation. All robots are situated in this environment and when a robot is first introduced its position is known in terms of its x and y coordinates. The approach suggested in this paper is a leader-follower approach wherein the environment that determines the leader and then the leader determines the positions on the circle circumference for the follower robots, such that, the total distance travelled by each follower is minimum and therefore the distance travelled by all the robots to form a circle is minimized.**

Keywords-multi-robot coordination, pattern formation, software framework, design patterns, leader-follower

I. INTRODUCTION

In this work the problem of the circle formation by arbitrarily placed robots in 2D plane is addressed. The robots are further required to maintain an equal distance with their neighbors on the circle circumference. The proposed software framework utilizes two well known design patterns [1] (a) decorator and (b) observer, to accomplish this task. From a practical standpoint this approach is highly credible and doesn't take any hypothetical assumptions about the system or individual robots themselves. This is a leader-follower approach wherein the leader guides the followers to their respective positions on the circle circumference. A unique identity is assigned to each robot which is used for the purpose of tie breaking and conflict resolution in various situations. The rest of this paper is divided into five sections. Section-II provides a brief introduction to the pattern formation problem in multi-robot systems and related work. Section-III presents the proposed software framework for the circular formation of multi-robot systems. Section-IV discusses the details of the system model and specific algorithms run by the Environment, the Leader and the Followers. Section-V presents results of computer simulation. We conclude the paper in Section-VI with a discussion on the scope of future work.

II. RELATED WORK

Pattern formation is identified as a classical problem in the area of multi-robot systems. It addresses an interesting question of how simple robots can interact with each other and with their environment to form an arbitrary geometric pattern. Research in the area of pattern formation using multi-robot systems can be categorized under three heads: pattern creation, pattern stabilization, and pattern transformation. Approaches that deal with pattern creation [2, 3, 4, 5, 6] suggest ways by which multiple robots interact with each other and create predefined geometric shape(s). Research work in multi-robot systems is inspired from the natural behavior of living organisms such as birds, fishes, ants etc. The natural phenomenon of flocking in many living organisms is highly related to pattern formation and is always discussed in conjunction. A pioneering work [6] gives a clear cut distinction between pattern formation and flocking. Patterns that emerge out as a result of flocking are loose geometric shapes while patterns which are created with an intention to bring multiple robots into predefined geometric shapes are strictly geometric. The proposed framework in this paper facilitates creation of strict geometric pattern, a circle, such that, robots place themselves on circle circumference and pattern converges towards uniformly distributing them on circle circumference.

Approaches to pattern stabilization consider that multiple robots are already in a predefined geometric shape. We need to stabilize the pattern in the presence of external disturbances, such as other moving objects or obstacles. The environment in which they are situated might itself be in turbulence or the formation which is moving as a team might encounter

This work is being funded by CAIR-DRDO, Ministry of Defense (Government of India) via its Grant# CAIR/ROB/CARS-2010/2. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agency.

978-1-4673-2605-6/12/$31.00 © 2012 IEEE

obstacles which could be static or dynamic, on its path. They also need to stabilize when the group scales in the event of new robots joining the formation and/or robots leaving the formation. All such works fall under the category of self organizing/ self healing systems. The approach for accomplishing self-organization and self-healing developed in [7] describes a method that can grow and heal any shape. However this work requires the robots to reproduce, which is not likely to happen with the existing state of the art in robotics. A method for self-healing is developed in [8] which allow pattern formation and healing of the impressive number of 3D shapes. This method makes use of explicit communication from one agent to another over a private communication channel between connected robots. This increases the complexity of the robots. They also require a starting seed to start formation which is a form of centralized control.

The problem of pattern transformation cannot be entirely segregated from the problem of self stabilization/ self organization/ self healing. During pattern transformation the system organizes itself to adapt to the state of the environment. For example a team of mobile robots which is already in a particular shape other than the line, flocks in its environment and finds out a narrow passage from which only one robot can pass through. In such a case the current formation of the team requires transformation into a straight line. One of the most recent works [9] gives a critical review and implementation of swarm pattern formation and transformation model. This work clearly highlights some of the key trends and issues in pattern formation and transformation. It provides a simulation for a pattern transformation and starts with an assumption that team of mobile robots has been already in some well defined formation and requires transformation.

Multi-robot formations have several applications. For instance, formations of multiple robots can act as sensor arrays and can be trained to collect spatial information about their environment which can be used for simultaneous and parallel exploration. Individual robots can exchange their sensory impressions with their neighbors and therefore minimize the time of exploration. Formations of multiple robots are very useful in search and exploration tasks, especially those in which the spatial pattern of the environment is complex as in the case of sound [11] or odor [12]. Another important where multi-robot formations are highly desirable is the task of mapping [13] the environment. Redundancy in the measurements of both the environment and the relative positions among the robots might allow them to build more accurate maps than those generated using single robots.

The present work proposes a robust and scalable framework to enable multiple mobile robots to form a circle as a geometric pattern and to uniformly distribute themselves on the circle circumference. The framework is effective and can easily be ported to an experimental test bed. It can also handle enhancements such as adding different algorithms for patterns other than circle.

III. SOFTWARE FRAMEWORK

The proposed software framework for circle formation of multiple robots arbitrarily scattered on a 2D plane is robust and utilizes decorator and observer design patterns [1]. Although the approach we have used is a leader-follower approach, we have tried to keep the leader robot lightly loaded. The system being modeled here consists of two primary entities (a) environment in which the robots are situated (2D plane) and (b) arbitrarily placed robots on the 2D plane. The environment is treated as a first class entity providing support to the robots that are situated in it. When the system parameters are successfully initialized and robots appear on the plane they are all anonymous. None of these robots can be discriminated as leader or follower. It is the environment which determines the leader robot. The choice of decorator design pattern is corroborated to be valid at this time as we decorate the elected robot leader with leadership responsibilities. All other robots are decorated with follower responsibilities. The primary entity, robot, can function either as a leader or as a follower. To achieve this, two virtual entities (a) robot leader and (b) robot follower are created. The virtual entities consume the robot to impart leader or follower role to the undecorated robot. The robot leader registers all robot followers, computes and notifies appropriate positions for all of them on the circle circumference. The observer pattern provides the mechanism for information exchange between leader and follower. The software framework is presented in terms of UML class diagrams. We now illustrate scenarios that are critical to the problem of circle formation.

A. *Class Diagram of the Framework*

Primarily two classes are important in our framework (a) Environment and (b) Robot. The other two classes RobotLeader and RobotFollower are decorators which are used to decorate Robot. There also exist some classes like, Circle and Point. Circle describes the Smallest Enclosing Circle (SEC) which encloses all the robots on the 2D plane (discussed in Section-IV) and class Point describes a point on a 2D plane in terms of its x and y coordinates. A robot must agree to implement one of the two contracts discussed below, when it takes up a leader role or the follower role. A Robot comes into the contract when it is decorated as a RobotLeader, which implements an interface IRobotLeader or RobotFollower, which implements an interface IRobotFollower, as shown in Fig. 1. This way we make sure that Robots taking the role of RobotLeader or RobotFollower accomplish their respective responsibilities.

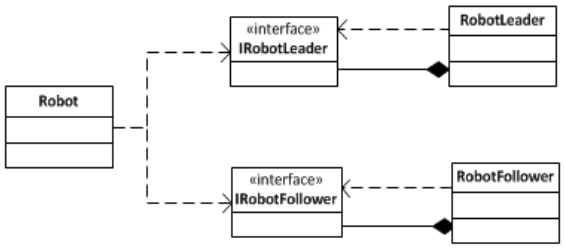

Fig. 1: Decorate Robot as Leader or Follower

Fig. 2 represents RobotLeader as a Subject and RobotFollower as an Observer. RobotFollowers use RobotSubject interface to register as Observers and also to remove themselves from being an Observer. This interface has just one method that gets called when the Subject's (i.e. RobotLeader's) states change. RobotLeader is a concrete subject which implements RobotSubject interface and has implementations to register, remove and a method to notify observers. RobotFollower is a concrete observer as it implements the RobotObserver interface. Each RobotFollower registers itself with RobotLeader to receive updates.

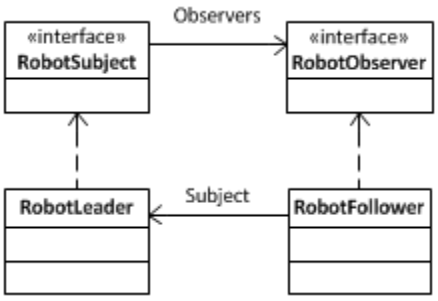

Fig. 2: RobotLeader as Subject and RobotFollower as Observer

B. *Critical scenarios*

1) Construct Robots: Robots are constructed by the user on the 2D plane. The Robots are assigned ranks from 1 to N on first constructed first numbered basis.

2) Find Smallest Enclosing Circle: After the Robots are initialized and randomly constructed on a 2D plane, it is the environment which initiates the scenario to find out the smallest enclosing circle (SEC) [10]. This circle encloses all the Robots. At least two Robots are on the circumference, and all other Robots are in this circle.

3) Find Leader Robot: Again it is the Environment that finds out the leader robot. Leader robot is the one which is nearest to the center of the SEC. If two robots are equidistant from the center of SEC we choose the leader as the one which has lower rank. This Robot is ordered to move to the center of SEC.

4) Decorate Leader Robot as RobotLeader: The Robot which is elected as leader is decorated as RobotLeader. It then gains all the capabilities of a leader.

5) Decorate Non-Leader Robots a RobotFollowers: All other Robots are decorated as RobotFollowers. They gain all the capabilities of a follower.

6) RobotFollowers register with RobotLeader: All RobotFollowers register themselves with RobotLeader. They will be notified by the leader of any state change.

7) RobotLeader calculates uniform positions for RobotFollowers on the circumference of SEC: The algorithm for finding out the positions is discussed in Section-IV.

8) RobotLeader finds out right positions for RobotFollowers and notifies them: It is done such that a position is picked up and the RobotFollower which is closest to this position is found. If two RobotFollowers are equidistant to a selected position the one which is lower in rank is assigned this position.

9) RobotFollowers move to respective positions on the circumference of SEC: Execution of this and above scenarios results in the formation of a circle by multiple mobile robots which are uniformly distributed on the circumference of this circle. We call this as convergence towards uniformity.

IV. SYSTEM MODEL AND ALGORITHMS

A. *System Model*

The Environment is modeled in software as a 2D plane. Robots are modeled as movable point objects. All robots have distinct identities (ranks) which are used for conflict resolution and/or tie breaking at various occasions. When a Robot is created and initialized, its initial position in terms of its x and y coordinates in the Environment are known. Robots can explicitly communicate. All Robots share common coordinate system. Robots are oblivious, such that, they do not maintain history.

B. *Algorithms run by Environment*

1) Calculate Smallest Enclosing Circle: Assume a set P of N points, $P = \{p_1, p_2, p_3 \ldots, p_N\}$ in the Euclidian plane R^2. The smallest enclosing circle [10] of P, SEC(P), is the circle with minimal radius enclosing all points in P, see Fig. 3. It is also well know that SEC(P) = SEC(H), where H is a proper subset of P, consisting of extreme points on the convex hull of P.

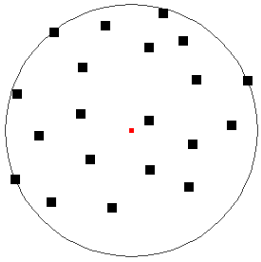

Fig. 3: Diagrammatic Representation of Smallest Enclosing Circle (SEC)

Algorithm-1 presented in [10] calculates the SEC(P) in O(nlogn) time. The algorithm is implemented in the Environment itself.

2) Leader Robot Selection: After SEC(P) is found, each point p in set {P – robots on boundary} we calculate p's distance from the center of SEC. Leader is the one which is closest to the center of the SEC. The worst case running time of this algorithm is O(n-2) as atleast two robots must fall on SEC circumference. The leader is decorated as RobotLeader and moves to the center of the SEC.

C. *Algorithms run by RobotLeader*

1) RobotLeader calculates uniform positions on SEC circumfrence: Once the RobotLeader is positioned at the center of SEC it calculates uniform positions for the RobotFollowers on SEC circumference.

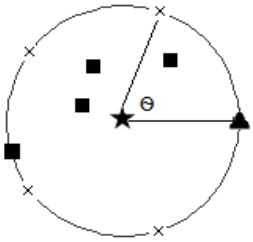

Fig. 4: SEC with two Robots on circumference and three Robots inside it

In Fig. 4 the star and triangle robots are active, such that, the one that is on the center (star) is RobotLeader and the other one (triangle) is one of those RobotFollowers which was already on the circumference and is randomly picked up as a reference point. It is named FirstFollower and is positioned (will not move). Now the RobotLeader finds the coordinates of N points for N robots beginning from FirstFollower, such that, all points are separated from each other by an angular distance of θ degrees:

a) θ = (2*PI)/N, N is the number of RobotFollowers

b) Now we know x and y coordinates of RobotLeader, FirstFollower, and an angle θ. We want to determine the x and y coordinates of cross points on the SEC circumference, as shown in Fig. 4.

c) We calculate two points x and y as follows:

x = FirstFollower.x – RobotLeader.x

y = FirstFollower.y – RobotLeader.y

d) To determine x and y coordinates of cross points we run a loop from i = 1 to N-1 and perform the following calculation:

$a_i.x$ = RobotLeader.x + x*cos(i*θ) – y*sin(i*θ)

$a_i.y$ = RobotLeader.y + y*cos(i*θ) – x*sin(i*θ)

This way we determine all uniform positions where RobotFollowers are finally positioned. The running time of this algorithm is O(n-1).

2) Find right position for RobotFollowers: RobotLeader maintains the list of uniform positions. Leaving FirstFollower and its position, RobotLeader picks up the first position from the list and finds out its distance from each RobotFollower. RobotFollower which is closest is notified this position. This process repeats for all the uniform positions and hence all RobotFollowers are notified their positions on SEC circumference. RobotFollowers move to their respective postions. As a result circle formation by arbitrarily scattered multiple mobile robots in 2D plane is achieved.

V. SIMULATION RESULTS

In the first step of simulation system allows user to choose N robots arbitrarily placed in 2D plane as shown in Fig 5(a). The next step is the formation of SEC by the environment as shown in Fig 5(b). Fig. 5(c) shows the process of leader selection where distance of each robot from center of SEC is calculated and the robot closest to the center is designated as leader by the environment and moves to the center of SEC as shown in Fig 5(d). Once the leader is selected it computes equidistant points on SEC circumference and communicates it to the follower robots which are closest to those points. The follower robots then move to their respective positions on the circumference of SEC as shown in Fig 5(e).

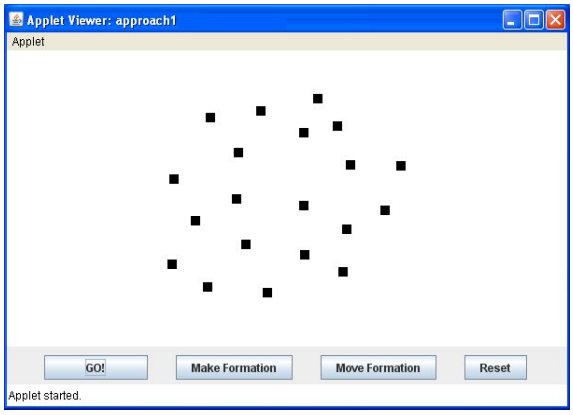

Fig. 5(a): Robots arbitrarily scattered on a 2D plane

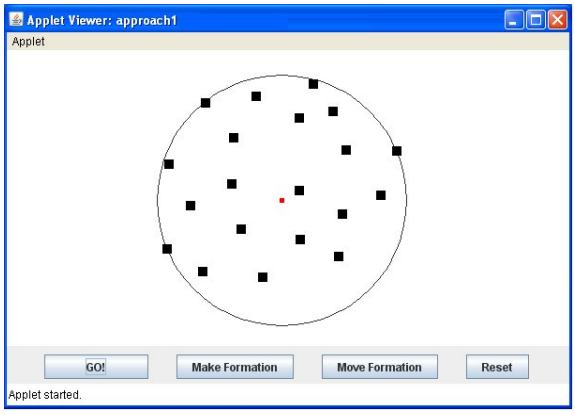

Fig. 5(b): Formation of Smallest Enclosing Circle

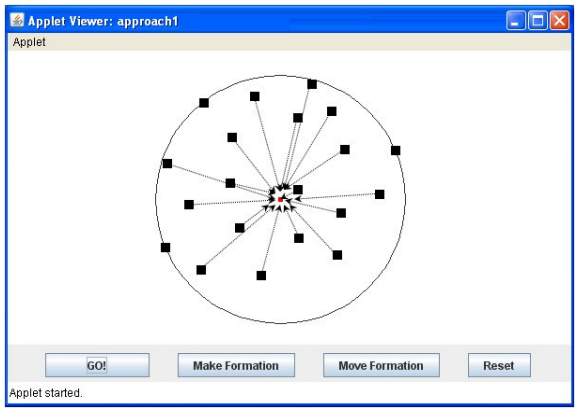

Fig. 5(c): All robots calculate their distance to the center of the SEC

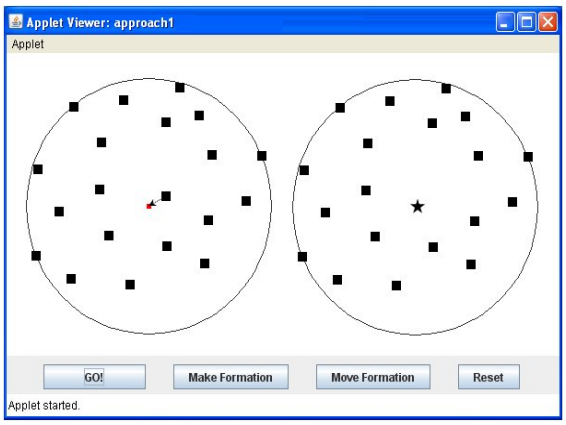

Fig. 5(d): Robot that is nearest is chosen as a leader and positions itself at the center of the SEC (shown by star)

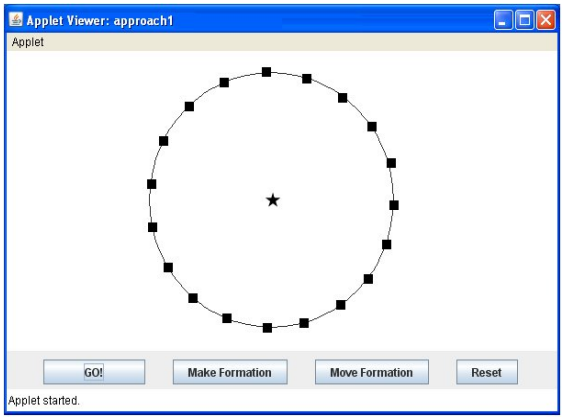

Fig. 5(e): Follower robots move to their respective positions.

VI. CONCLUSION AND FUTURE WORK

In this paper we have designed and implemented a practical framework for circle formation by multiple mobile robots with convergence towards uniformity. We have utilized two of the most frequently used design patterns, decorator and observer in our framework. The proposed framework is robust and scalable. A leader-follower approach has been adopted in this paper. The algorithm gives good performance in spite of being centralized in nature. The worst case time complexity of any of the algorithms is O(n).

Future work will involve leader re-election in particular for three cases wherein leader willfully wishes to retire, leader dies, and byzantine failure. We will also implement fault-tolerance for follower robots. More extensions are possible such as pattern transformation. The problem can be extended to form any general pattern.

REFERENCES

[1] Gamma, E.; Helm, R.; Johnson, R.; and Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995.

[2] Ando, H.; Oasa, Y.; Suzuki, I.; Yamashita, M.; "Distributed memoryless point convergence algorithm for mobile robots with limited visibility", IEEE Transactions on Robotics and Automation, Vol. 15, no.5, pp. 818-828, Oct 1999.

[3] Flocchini, P.; Prencipe, G.; Santoro, N.; Widmayer, P.; "Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots," Algorithms and Computation, LNCS, vol. 1741/1999, pp. 93-102.

[4] Flocchini, P.; Prencipe, G.; Santoro, N.; Widmayer, P.; "Gathering of asynchronous robots with limited visibility", Theoretical Computer Science, Volume 337, Issues 1–3, 9 June 2005, Pages 147-168, ISSN 0304-3975, 10.1016/j.tcs.2005.01.001.

[5] Défago, X.; Konagaya, A.; "Circle formation for oblivious anonymous mobile robots with no common sense of orientation", in Proc. of the second ACM international workshop on Principles of mobile computing, New York, USA, 2002.

[6] Balch, T.; Arkin, R.C.; "Behavior-based formation control for multi-robot teams", IEEE Transactions on Robotics and Automation, Vol. 14 No. 6, pp. 926-39, 1999.

[7] Kondacs, A.; "Biologically-inspired Self-Assembly of 2D Shapes, Using Global-to-local Compilation", in Proc. of International Joint Conference on Artificial Intelligence (IJCAI), San Francisco, CA, USA, 2003.

[8] Stoy, K.; Nagpal, R.; "Self-repair through scale independent self-reconfiguration," in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)., pp. 2062- 2067, 28 Sep-2 Oct. 2004.

[9] Varghese, B.; McKee, G.; "A review and implementation of swarm pattern formation and transformation models", International Journal of Intelligent Computing and Cybernetics, Vol. 2, no. 4, pp.786 – 817, 2009.

[10] Skyum, S.; "A simple algorithm for computing the smallest enclosing circle", Information Processing Letters, Vol. 37, no. 3, Feb-18 1991, Pages 121-125, ISSN 0020-0190, 10.1016/0020-0190(91)90030-L.

[11] Pugh, J.; Martinoli, A.; , "Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization," *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE* , vol., no., pp.332-339, 1-5 April 2007.

[12] Hayes, A.T.; Martinoli, A.; Goodman, R.M.; , "Distributed odor source localization," *Sensors Journal, IEEE* , vol.2, no.3, pp. 260- 271, Jun 2002.

[13] Howard, A.; , "Multi-robot mapping using manifold representations," *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on* , vol.4, no., pp. 4198- 4203 Vol.4, April 26-May 1, 2004.