Coordination of Realistic Decentralized Control Systems

John A. Turner and Ashia C. Wilson, *Student Members, IEEE*

Abstract— Existing algorithms for the distributed control and coordination of multi-vehicle systems assume that each agent possesses sophisticated sensing and mobility capabilities. In contrast to prior work, here we focus on the design of distributed coordination algorithms for ant-sized robots that will not possess the sensing and movement capabilities of the existing full-scale ones. Our research also involves the creation of a modular simulation environment, which will be useful as a design aid since it can be used to test the functionality of new algorithms. Algorithms have been developed for the following tasks: rendezvous, following a leader, and an equidistant circular formation. We are coordinating with other groups at the University of Maryland, College Park, who are working on the fabrication of ant-sized robots, and will assist them in determining what kinds of sensing, communication, and computation will be necessary to achieve coordinated movement.

Index Terms— Control Systems, Decentralized Control, Formation Control, Multi-robot Systems, Coordination, Low Bandwidth

I. Introduction

In the field of robotic control, various approaches for achieving decentralized control of communicating agents have proven successful. Most systems have demonstrated that the emergence of complex behavior can emerge from simple rules. Czirók et al. demonstrates this principal by using an algorithm in which agents align their orientation to the local average velocity [1]. Bullo et al. and Vicsek et al. have also included this principal, using the average direction of motion of neighboring particles and by using a circumcenter calculation of objects within a certain radius to determine movement [2], [3]. Bailleul and Ren et al. use this principal assuming even more advanced capabilities such as the presence of a virtual coordinate frame and highly complex dynamics [4], [5]. Given that the rules governing the behavior of decentralized control tend to be simple, it is both surprising and problematic that the bulk of existing literature presumes robots that either have unrealistic or unnecessarily expensive sensing and movement capabilities. Limitations in communication range and bandwidth for the majority of cheap small and lower powered robots indicates the need for distributed algorithms that require less information. This paper will continue to apply these simple rules for use of smaller, cheaper, more realistic robots, with fewer capabilities.

Manuscript received August 3, 2009. This material is based upon work supported by the National Science Foundation under Grant No. 0755224.

John A. Turner is a senior at University of Maryland (e-mail: jturner2@umd.edu)

Ashia C. Wilson is a junior at Harvard University (e-mail: acwilson@fas.harvard.edu)

II. Procedure And Assumptions

The goal of this project is to approach coordination control from a different angle than those in much of the existent literature. Typical procedures define a goal for the agents to achieve, and afterwards define the agents' sensing and movement capabilities in a way that makes the goal obtainable. This procedure has led to many intricate and fascinating algorithms, but rarely are these algorithms realistic to run on existing, reasonably priced robots.

Before goals were set for behavior of the robots, a realistic, small, and cost effective model of a robot was defined, to ensure that the control algorithms written could be realistically implemented on minimally capable robots.

A. Robot Limitations

The biggest self-imposed limitation on these robots is their sensing capabilities. Robots will only be given information about the distance between themselves, and will not be given any information about the orientation of the swarm of robots (this information must be obtained experimentally). This set of system constraints was chosen because it can be easily and cost effectively implemented using RSSI or sonar in a small form factor.

The next self-imposed limitation on our robots is their movement capabilities. Omni-directional robots are very popular in literature, but expensive and difficult to implement in a small form factor. The movement model focused on bristle bots[2], given that the first implementation of these algorithms is planned on bristle bots. While bristle bots are a very specific, and perhaps esoteric type of robot, this choice will still allow realistic implementation on many robots, because bristle bots move in similar fashion to a car with no reverse, which is a reasonably general model of a small, low cost robot.

The final limitation was on power consumption. This was neither defined as strictly, nor analyzed as closely as the other constraints, but algorithms were designed to limit unnecessary computation, communication and movement.

[2] Bristle bots are small robots made from the head of a toothbrush, and motors from a pager.

B. Simulation

All simulations were done in MATLAB. The program is structured as a loop in the following order:

1) All robots receive distance inputs from their sensors (with noise)
2) All robots communicate information (assumed to be digital without errors)
3) All robots would make necessary calculations and determine movement directions,
4) Finally, all robots move and the system status updates.

Each iteration of this "loop" will be referred to as a round.

The noise on the distance calculations was assumed to be Gaussian, and the noise on each distance measurement was assumed to be independent of all other noise measurements. Robots are represented by arrows, and assumed to be circular objects in a 2-dimensional plane. When a robot collides with another robot or a wall, that robot is no longer able to proceed in the direction of the obstruction, but may move in an otherwise normal fashion (e.g., if a robot collides with a wall along the axis of x = 0, the robot's movement in the x-direction will be halted, but the robot may still move in the y direction, and will still be able to rotate.) To give an idea of scale, the robots are assumed to have a diameter of 0.25, in a ten by ten room, move a maximum distance of 0.10 in a round, and turn up to 10 degrees in a round.

C. Notation

Each robot is assigned a Unique Identification number (or UID), starting from 1 and working upwards. Robots will be identified by their UID number, and the leader may be identified by either its UID number, or, the letter "L."

Furthermore, "n" will be used to represent the total number of robots, and "d" will be used to represent the distance between two different robots. For example, $d_{1L}(t-1)$ will represent the distance between the leader and the robot with a UID of one, at time t minus one.

III. Successful Algorithms

A. Rendezvous with Leader

Broad Overview

The goal of this algorithm is for all of the robots to rendezvous around a stationary robot that will be designated the "leader." The only distance measurement that the robots used in this algorithm is the distance between themselves and the leader.

The basic idea of this algorithm is for each robot, *i,* to find the direction of travel that will minimize the temporal derivative of distance to leader (1).

$$\frac{dd_{iL}(t)}{dt} \quad (1)$$

Once the direction minimizing the derivative has been found, then robot *i* will proceed straight until the distance $d_{iL}(t)$ is acceptably low for the rendezvous to be complete.

Implementation

Discrete spatial locations and time steps are used, so taking derivatives does not make sense. Furthermore, even with a smooth curve, the calculation of the second derivative of distance to find a local minimum in the first would be too much computation for the target size and intelligence of our robots.

To remedy this, the robots will approximate the derivative of Eqn. (1) using the difference formula shown in Eqn. (2). The robot will then turn until the conditions specified by Eqns. (3) and (4) are both true.

$$\frac{dd_{iL}(t)}{dt} = d_{iL}(t) - d_{iL}(t-1) \quad (2)$$

$$\frac{dd_{iL}(t)}{dt} > \frac{dd_{iL}(t-1)}{dt} \quad (3)$$

$$\frac{dd_{iL}(t)}{dt} < 0 \quad (4)$$

Eqns. (3) and (4) will become true exactly one round after encountering the minimum of (1) is passed. To correct this, the robot will make a short movement in the opposite direction of that in which it was turning, and then proceed straight towards the leader.

In practice, the robot will not always find the perfect angle to head towards the leader (the discrete points measurements are typically taken 10 degrees apart), from a great distance might pass the leader completely. The solution to this problem is to introduce two possible conditions that will cause the robot to stop heading straight. The first, is that when $d_{iL}(t)$ is sufficiently low, suggesting the rendezvous is complete, the robot stops. The second condition is that when Eqn. (2) becomes positive, the rendezvous point has been passed, and the algorithm restarts (the robot is now closer to the rendezvous and will not miss a second time).

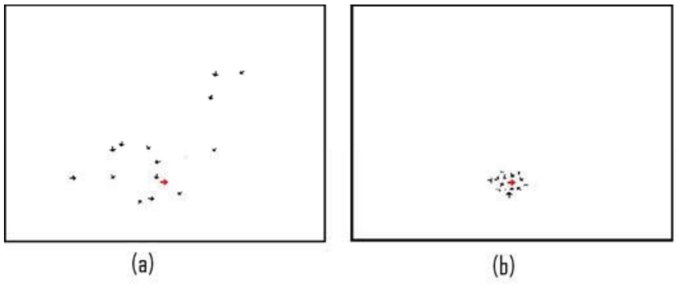


Fig. 1 shows the execution of rendezvous with leader. The black arrows are searching for the leader, and the red arrow is the leader. The simulation is run with n=15 and no noise in the measurements. (a) shows random scattering at the start of algorithm, while (b) shows the end behavior of all robots gathered around the leader.

Results

With no noise, this algorithm executes smoothly every time. A number of the robots must restart the algorithm once as described above, but will always make it on the second time through. The algorithm begins to encounter problems with (noise with a standard deviation about a half length of the robot, or about as far as the robot moves in one round.) In this case, robots will travel in circles multiple times before heading towards the rendezvous, and will avert progress towards the rendezvous prematurely, travelling again in circles. When noise is increased past that magnitude, coordinated behavior fades. Fig. 1 shows two frames in the execution of the rendezvous with leader algorithm, and a flowchart can be seen in the appendix.

B. Follow the Leader

Broad Overview

The goal of this algorithm is for all robots to follow a leader, travelling at less than full speed, and maintain as close a distance as possible while travelling with the leader. The algorithm used in simulation assumes that the leader is given a pre-ordained course, although the algorithm would still work if the leader were to determine the course as new information was gained. The rest of the robots will execute the rendezvous with leader algorithm while far away, and travel in the manner of the leader once they are near the leader

Implementation

The rendezvous with leader algorithm needed very little tweaking to work with this algorithm. That algorithm works by selecting the direction that would maximize decrease in distance which is still the goal. The movement of the leader makes it more likely that (2) will become positive, but this is desirable as the robots should re-align more often to successfully find a moving target.

The leader will always broadcast its speed, direction of turning, and radius of turn to the other robots. When the robots get sufficiently close, they will abandon the rendezvous algorithm and move in the exact manner of the leader, until they exit the leader's zone of control.

When a large number of robots all try to rendezvous with and follow the leader, collisions become a large problem. To remedy this, a robot will consider itself close to the leader if it is close to the leader, or close to a robot that is close to the leader.

If a robot loses the leader (or the robot it was following), it will broadcast to the group that it is no longer close to the leader. This way, any robot that was using that robot to consider itself close to the leader will no longer be able to do so, and will restart the algorithm.

Results

Robots do not tend to reach equilibrium where the entire group stays close to the leader for long periods. However, it is clear to an observer that the robots are all following (reasonably close) to the leader.

The proposed method of redefining "close to the leader," very effectively solves the collision problems that tend to arise from this algorithm.

With no noise this algorithm will consistently produce a group of robots following the leader. This algorithm is the least resistant to noise, and when any reasonable amount of noise is added to the distance measurements, the robots will look rather disorganized. The failing is actually in the rendezvous with leader algorithm. While the slower rendezvous observed with noise is acceptable with a stationary leader, the moving leader makes the delay unacceptable. The execution of this algorithm illustrated in Fig. 2, and a flowchart can be seen in the appendix.

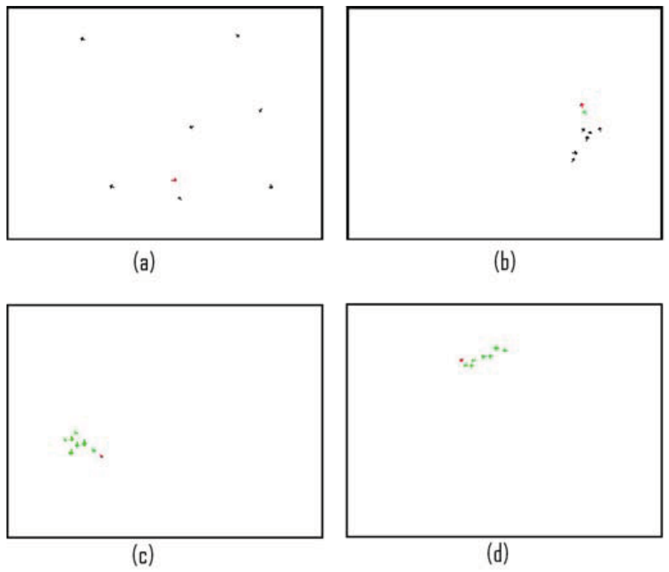


Fig. 2 shows the execution of the follow the leader algorithm. The black arrows are executing rendezvous with leader, the green arrows are "close" to the leader, and the red arrow is the leader. This simulation was run with n=8 and no noise. (a) shows the initial random placement, (b) shows the robots shortly before becoming close for the first time, (c) shows the first time they are all close, and (d) shows the second time they are all close.

C. Equidistant Circular Travel

Broad Overview

The goal of this algorithm is for all robots to travel counterclockwise around a circle while equally spaced. This algorithm uses a potential field, which is simply an invisible immobile robot in the center of the room (this algorithm could just as easily be executed with a robot in the center of the circle instead of a potential field.)

This algorithm is broken into two basic parts: the algorithm to keep all robots traveling on the circle and the algorithm to keep them equidistantly spaced. When the algorithm begins, the robots only execute the first part, and all proceed to position themselves on the circle. When the robots determine that they are all traveling along the circle, then both algorithms will run at the same time, the first telling them how to continue along the circle, while the second stops different robots from time to time to keep them equidistantly spaced.

Implementation

The algorithm that keeps the robots along the circle uses only the distance measurement from the potential field, (represented by a P) to $d_{iP}(t)$ line up around the circle. The robots will act differently depending on which of the following conditions is true, where r is the radius, and Δ is the change in radius over the "acceptable" range.

$$d_{iP}(t) < r - \Delta/2 \quad (5)$$

$$r - \Delta/2 < d_{iP}(t) < r + \Delta/2 \quad (6)$$

$$d_{iP}(t) > r + \Delta/2 \quad (7)$$

When the condition specified in Eqn. (5) is true, then the robot will turn slightly right. When the condition specified in Eqn. (6) is true, the robot will turn with a turning radius equal to that of the circle. When condition of Eqn. (7) is true the robot will turn left with a smaller turning radius than r. With these rules, the robots will slowly stabilize and travel counterclockwise along the circumference of the circle. Two changes were made to increase the rate at which robots reach equilibrium and stay on the circle (i.e., Eqn. (6) is true) by introducing memory into the robots' behavior. First, every round that Eqn. (6) is true, but Eqn. (5) or Eqn. (7) was true the round before, the robot will turn in the opposite direction it turned to make Eqn. (6) become true (hard left if Eqn. (5) was true last round, hard right if Eqn. (7) was true last round). Second, was to change the turning radius the robot uses in Eqn. (7) based on the number of times the robot has crossed from the condition of Eqn. (6) to Eqn. (7). This is done on the (very likely) assumption that the first few times the robot crosses from (6) to (7), the larger a correction will be needed than at the later times.

Before the robots begin spreading out, each robot must determine its two nearest robots, and which one is in front of the robot, and which one is behind. For the following description, a robots "neighbors" are the two adjacent robots on the circle, and not the two closest. Determining the robot's neighbors in the circle is rather simple, the nearest robot must be a neighbor, and the second neighbor can be determined by comparing the robot's distance measurements with the nearest robot's distance measurements.

Determining which neighbor is in front, and which is behind, is more difficult, and must be experimentally determined. To do this, one robot will stop, while all others continue moving. This robot will easily be able to determine which neighbor is in front and which is behind. The robot that stopped will then broadcast to the robot behind that the stopped robot was in front. The robot receiving that information will be able to determine which robot is behind through process of elimination, and in this way the information will ripple through the circle.

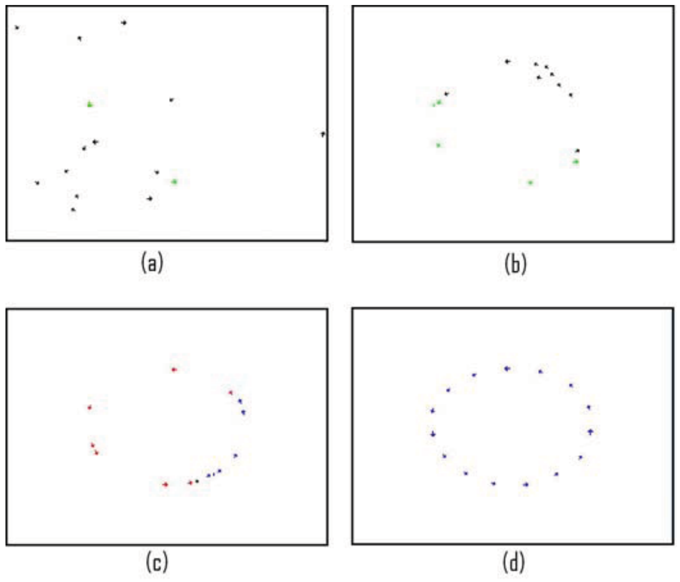


Fig. 3 shows the execution of the equidistant circular travel algorithm. The algorithm was run with n=15 and no noise. The black arrows are searching for the circle, while the green arrows are the robots satisfying Eqn. (6). In (c) and (d), blue arrows represent robots aware of their neighbors, while red arrows are unaware. (a) shows the initial random placement of the robots. (b) shows the robots as the circle begins to take shape. (c) shows the robots a couple turns after the robots began determining neighbors, and (d) shows the end behavior of the algorithm.

The final part of spreading out is rather simple. If $d_{i\to FRONT}(t) < d_{i\to BACK}(t)$ robot i will stop. Otherwise the robot will obey the first algorithm.

Results

This algorithm works exactly as planned without noise. When noise was added, a couple of changes were made to improve operation. Firstly, the Δ was increased because robots were turning off the circle while they were quite close to it. Secondly, logic was added to determine if the calculations of "neighbors" was done incorrectly. If it was, the linking of neighbors is disregarded, and the calculations would be redone. With these changes, the algorithm was able to operate well in the low noise area (noise with a standard deviation of about .05-.01, or the distance travelled in one turn.) With higher noise, the circle would begin looking loose and misshapen, and if both noise and number of robots per size of circle became high, it became near impossible to determine the neighbors. Fig. 3 shows execution of this algorithm, and a flowchart of the first part can be seen in the appendix

D. *Center of Mass Rendezvous*

Broad Overview

The goal of this algorithm is to complete a rendezvous without any stationary robots or potential fields to use as a reference. The robots will turn to minimize the derivative of the sum of all distance measurements, D_i, as defined in Eqn. (8). After the minimum is found, robots will proceed straight until the derivative of D_i becomes positive, then restart. Every round, the robot with the lowest D_i is closest to the center, and is the leader for that round. When the average distance to the leader gets below a certain value, the leader and any robots directly next to the leader will cease movement. Robots who are not close to the leader (as defined in the follow the leader algorithm) continue moving until they are close to the leader.

$$D_i = \sum_{j=1}^{n} d_{ij}(t) \qquad (8)$$

Implementation

Robots begin by turning and calculating D_i each turn. Robots are attempting to find the angle that will minimize the temporal derivative of the aggregate distance as define by Eqn. (9). As in the first algorithm, to simplify calculations, the robot will be searching for the point in time when the conditions specified in Eqns. (10) and (11) are both true, and correcting a small amount in the direction opposite the turn. The robots will then proceed forwards until the conditions specified in Eqn. (11) becomes false, or until the robots' conditions for completing the rendezvous are met. If Eqn. (11) becomes false, the algorithm will restart, and if the robot's conditions for rendezvous are met, then the robot will stop.

$$\frac{dD_i}{dt} = \sum_{j=1}^{n} d_{ij}(t) - \sum_{j=1}^{n} d_{ij}(t-1) \quad (9)$$

$$\frac{dD_i(t)}{dt} > \frac{dD_i(t-1)}{dt} \quad (10)$$

$$\frac{dD_i(t)}{dt} < 0 \quad (11)$$

The stopping conditions for this algorithm are different from the previous rendezvous. All robots will continue moving until the current leader's D_i value becomes sufficiently small to stop. When this happens, the leader will stop, as will any robot that is close to the leader. Similar to the follow the leader algorithm, any robot near the leader or near a stopped robot will stop and define itself as close to the leader. If for some reason the leader's D_i value increases again (an extremely rare case), the algorithm will restart (and likely reach rendezvous in a small number of rounds).

Results

The algorithm will always succeed under ideal conditions, and works excellently with low and moderate noise. This algorithm was the most noise resistant algorithm studied. Because the summation used in (8) has $n-1$ additions of independent Gaussian noise, the higher n is, the more resistant the algorithm becomes to noise. This means that many robots can execute a rendezvous with very high noise. As the standard deviation of the noise becomes close to the size of one robot, the conditions that help the robots decide when to stop begin to fail. This is not catastrophic, as the robots will still rendezvous, but will then move after the rendezvous is complete. This would likely be remedied by adding a threshold. This algorithm is seen working in figure 4, and a flowchart can be seen in the appendix.

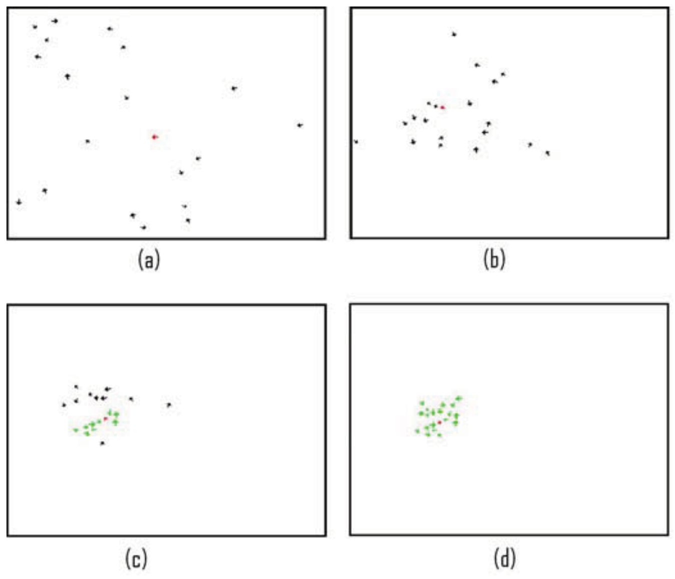


Fig. 4 shows the execution of the center of mass rendezvous. The red robot is the current leader, the black robots are searching, and the green robots are close to the leader. The simulation was run with n=20, and no noise. (a) shows the initial random placement of the robots. (b) shows most robots heading towards the center of mass. (c) is shortly after the leader stopped for the first time, and (d) is the end behavior.

IV. CONCLUSION

The aim of this work is to present examples of decentralized control processes for low-cost robots that possess minimal sensing and movement capabilities. The first step for future work will likely consist of the addition of a larger memory of past measurements and a filter to discard outliers resulting from noise. It is theorized that this would make the algorithms more noise resistant, as many of the problems are caused by a small number of noisy measurements, and not a failure in the concept. The success of each algorithm in the absence of minimal noise demonstrates proof of concept and indicates a plausible techniques of decentralized control for multi-robot systems.

.

ACKNOWLEDGMENTS

We would like to thank Dr. Pamela Abshire, Dr. Nuno Martins Timir Datta, the MERIT program and the National Science Foundation for their support and guidance during the course of our research.

REFERENCES

[1] Czirók, András, Albert-László Barabási, and Tamás Vicsek. "Collective Motion of Self-Propelled Particles: Kinetic Phase Transition in One Dimension." *Physical Review Letters* 82.1 (1999): 209-12.

[2] Bullo, Francesco, Jorge Cortes, and Sonia Martinez. *Distributed Control of Robotic Networks*. Princeton UP, 2009. Applied Mathematics Ser.

[3] Vicsek, Tamas, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. "Novel Type of Phase Transition in a System of Self-Driven Particles." *Physical Review Letters* 75.6 (1995): 1226-229.

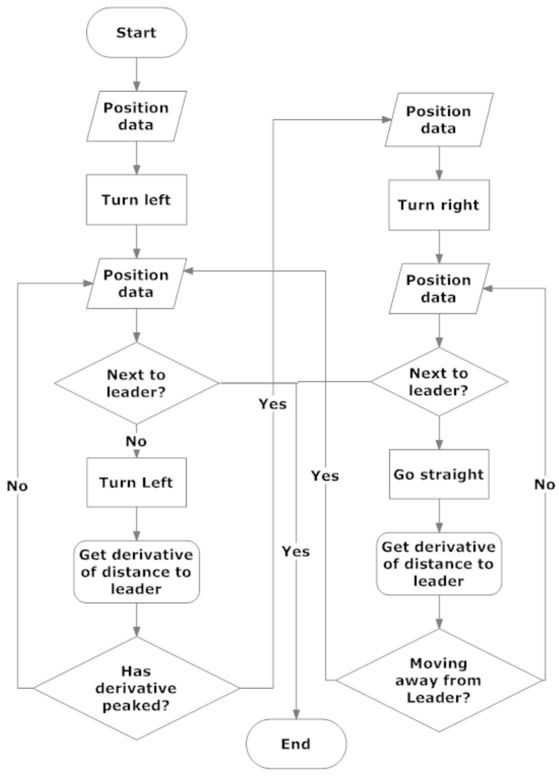
[4] J. Bailleul, 2006 "Remarks on a Simple Control Law for Point Robot Format with Exponential Complexity," *Proceedings of the 45th IEEE Conference on Decision and Control,* San Diego, California, December 13-15, ThB10.6, pp. 3357-3362.

[5] Ren, Wei, and Nathan Sorensen. "Distributed Coordination Architecture for Multi-robot Formation Control." *Robotic and Autonomous Systems* 56 (2008): 324-33.

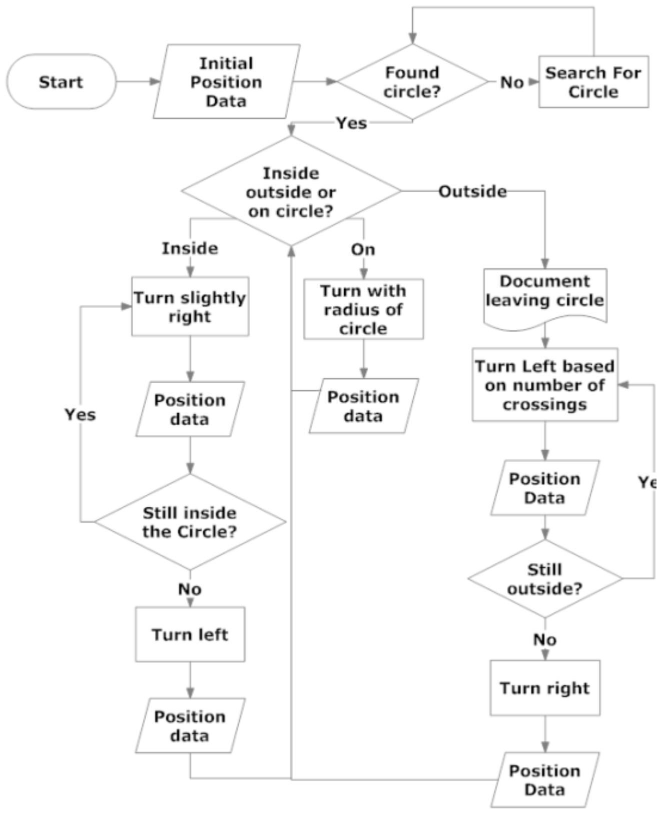
John A. Turner is a senior at University of Maryland and a member of the MERIT program. He will receive his bachelors in Electrical Engineering in June 2010.

Ashia C. Wilson is a junior at Harvard University and a member of the MERIT program. She will receive her bachelors in Applied Mathematics in June 2011.

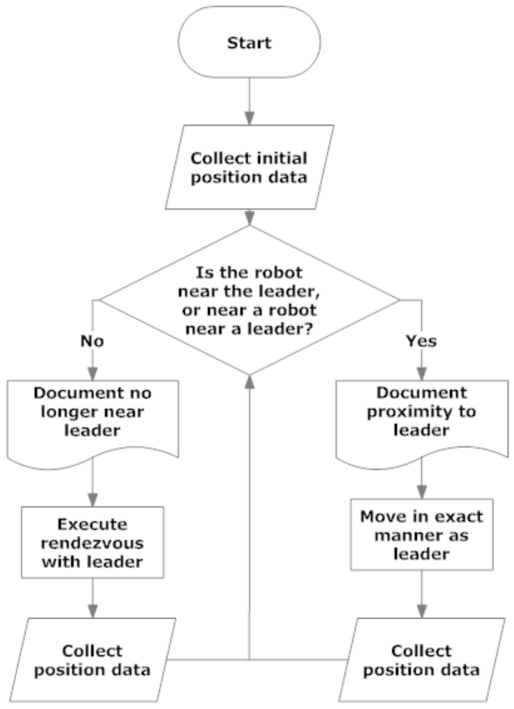
Appendix



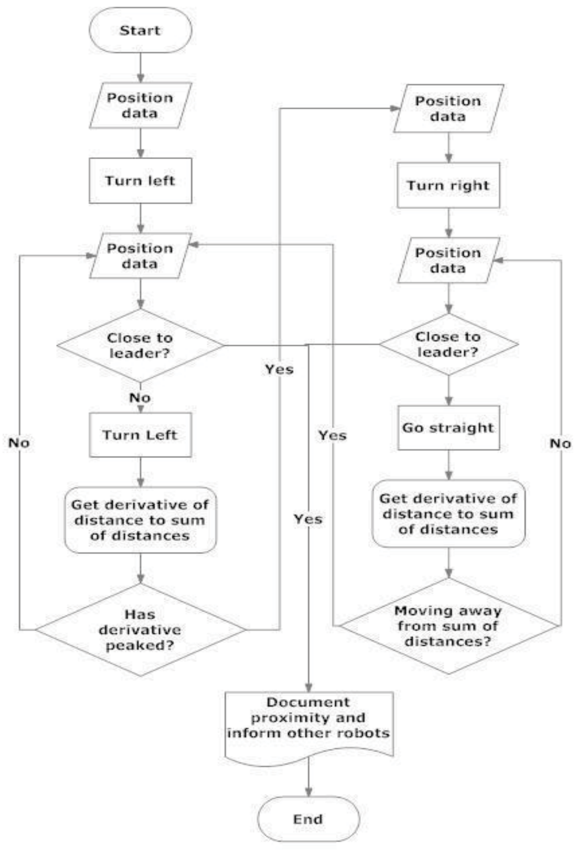
This flowchart is the logic for rendezvous with leader



This flowchart is the logic for the first part of the circular travel algorithm


Start
Collect initial position data
Is the robot near the leader, or near a robot near a leader?
No
Yes
Document no longer near leader
Execute rendezvous with leader
Collect position data
Document proximity to leader
Move in exact manner as leader
Collect position data

This flowchart is the logic for follow the leader


Start
Position data
Turn left
Position data
Close to leader?
No
Turn Left
Get derivative of distance to sum of distances
Has derivative peaked?
No
Yes
Yes
Yes
Position data
Turn right
Position data
Close to leader?
Go straight
Get derivative of distance to sum of distances
Moving away from sum of distances?
No
Document proximity and inform other robots
End

This flowchart is the logic for the center of mass rendezvous