1997

Line and Circle Formation of Distributed Physical Mobile Robots

Yun, Xiaoping; Alptekin, Gokhan; Albayrak, Okay

Journal of Robotic Systems 14(2), 6376 (1997)
http://hdl.handle.net/10945/40166

Line and Circle Formation of Distributed Physical Mobile Robots

Xiaoping Yun,* Gokhan Alptekin, and Okay Albayrak
Department of Electrical and Computer Engineering
Naval Postgraduate School
Code EC/YX
Monterey, CA 93943-5121
e-mail: yun@ece.nps.navy.mil

Accepted September 26, 1996

The formation problem of distributed mobile robots was studied in the literature for idealized robots. Idealized robots are able to instantaneously move in any directions, and are equipped with perfect range sensors. In this study, we address the formation problem of distributed mobile robots that are subject to physical constraints. Mobile robots considered in this study have physical dimensions and their motions are governed by physical laws. They are equipped with sonar and infrared range sensors. The formation of lines and circles is investigated in detail. It is demonstrated that line and circle algorithms developed for idealized robots do not work well for physical robots. New line and circle algorithms, with consideration of physical robots and sensors, are presented and validated through extensive simulations.

この研究は、分散型モービル・ロボットの隊形問題に関して、理想化されたロボットに対して行ったものである。理想化されたロボットは、任意の方向に瞬間的に移動でき、完璧な距離センサーを装備している。この発表では、物理的な制約のある分散型モービル・ロボットの隊形問題について説明する。ここでは、モービル・ロボットは物理的な次元を持ち、その動作は物理法則に支配されていることを前提にしている。これらのロボットには、超音波と赤外線の距離センサーが装備されている。そして、直線または円形の隊形についての詳細な調査結果を示す。また、理想化されたロボット用に開発した直線と円形のアルゴリズムは、実際のロボットでは使えないことを証明する。さらに、実際のロボットとセンサー用に新たに開発した直線と円形のアルゴリズムについて説明し、集中的なシミュレーションによってその妥当性を証明する。

*To whom all correspondence should be addressed.

Journal of Robotic Systems 14(2), 63–76 (1997)

CCC 0741-2223/97/020063-14

1. INTRODUCTION

Given a group of mobile robots (say, 20 robots) randomly placed on a laboratory floor, how would one control them to form a geometric pattern such as a circle without using a centralized coordinator? This is the formation problem of distributed mobile robots studied previously. Distributed robots make motion plans based on a given task goal of the group and the perceived information about their environment from onboard sensors without the aid of a centralized coordinator.

The formation problem of distributed mobile robots has been studied for idealized mobile robots[1,2]—robots that are represented by a point, able to move in any direction, and equipped with range sensors that can determine the position of all other robots. Since a robot is a point, two or more robots may occupy the same location. Each robot has its own coordinate system and there is no common, global coordinate system. Furthermore, these robots do not communicate with each other. Under these assumptions, Suzuki and his colleagues have developed a number of distributed formation algorithms. In particular, they developed algorithms for multiple distributed mobile robots to form circles, simple polygons, and line segments; to uniformly distribute robots within a circle or a convex polygon; and divide them into groups.[1–4]

In the previous study,[1–2] even though the number of robots participating in a given task is assumed to be unknown, the perfect sensor assumption makes it possible for each robot to ''see'' the location of all other robots, and hence to determine the number of robots. Perfect sensors are not occluded by the presence of other robots. One of the biggest challenges in implementing existing formation algorithms is the inability to sense the location (or even just the presence) of all other robots by using sonar or infrared sensors. Each robot may see a different number of robots at each instant of time.

Based on earlier work, we study the formation problem of distributed *physical* mobile robots. We consider robots that have physical dimensions (hence two robots cannot occupy the same spot), and whose motions obey physical laws (hence wheeled mobile robots must satisfy nonholonomic constraints). Furthermore, robots are assumed to be equipped with range sensors having realistic physical properties. We use Robot Simulator from Nomadic Technologies, Inc. Robots in the Simulator realistically simulate the motion behavior and sensor systems of Nomad 200 mobile robots. The Nomad robot has a synchronous drive mechanism that enables it to translate, steer, and rotate (its turret) independently. The robot is nonholonomically constrained, thus not able to instantaneously move in the lateral direction. The robot's sensor systems include tactile (bumper) sensors, infrared sensors, ultrasonic sensors, and laser sensors. All but the laser sensors are used in the simulation of this study.

Motion control and collision avoidance in this study are achieved by implementing a potential field algorithm.[5,6] To each robot of concern, the presence of other robots generates a repulsive force that keeps them apart, and the goal position produces an attractive force. Because the workspace is assumed to be obstacle-free, the shape of robots is circular, and the goal position changes as other robots move, the local minimum problem of the potential field method is rarely encountered in the simulations.

Line and circle formation, or formation of any geometric patterns in general, is only one of many issues of distributed mobile robots.[7] Representative work addressing other issues of distributed mobile robots includes cellular robotics systems[8–10] and dynamically reconfigurable robotic systems.[11] These systems can change their overall shape depending on the task and the environment by autonomously detaching and combining cells.

Different schemes for collision avoidance were examined in references.[4,12–17] The method proposed in reference 4 is discussed in the following section. The strategy proposed in reference 14 is that if a robot detects another robot on its way, it stops and waits some fixed period of time. If a robot is still present, the robot turns left and proceeds forward. The method proposed in reference 15 adds an initial step to the algorithms from reference 1 to avoid collisions. Motor schemas[18] is another method for navigation and collision avoidance. More relevant to the present study is the previous work on formation and agreement problems of distributed mobile robots.[1–4,15]

2. LINE FORMATION ALGORITHMS

In this section we show the results and associated problems of the existing algorithm when it is implemented with physical robots. We then present a modified version of the existing algorithm that works well for physical robots, and a completely new algorithm using the least-square line fitting.

2.1. Existing Algorithm

The following is the original line algorithm proposed in refs. 1 and 19. It is assumed that each robot repeat-

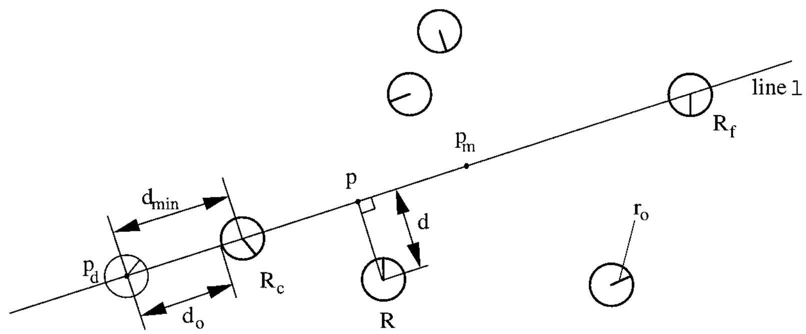

Figure 1. Illustration of notations used in line formation algorithms.

edly becomes active and inactive (sleep mode) at unpredictable time instants. Each time a robot becomes active, it does the following:

Step 1. Determines the furthest robot R_f and the closest robot R_c.
Step 2. Calculates the distance d from its current position to the point p that is the foot of the perpendicular drop from itself to the line ℓ passing through R_c and R_f (see Fig. 1).
Step 3. Moves $\min\{d, v\}$ toward point p, where v is the maximum distance the robot can move at a time.

Assuming that each robot is a dimensionless point, the algorithm enables all robots to form a line segment.[1,19] In a revised version of the algorithm, the physical dimension of the robots was considered, and a simple collision avoidance strategy was implemented.[4] The strategy works as follows. If a robot detects another robot within a certain distance in the direction of its move, it then swerves to the left minimally, provided that it successfully finds a direction that is clear of any robots. If no such left swerve is found possible, the robot decides not to move until either its path becomes clear or a suitable left swerve becomes possible.

We first simulated this algorithm without any collision avoidance scheme, and experienced an unacceptable number of collisions, which prevented robots from forming a line. We then implemented the algorithm with the simple left-swerve collision strategy and ran 20 simulations, each time with a random initial distribution of the robots. We observed a number of problems with the implementation of the algorithm. One of the problems occurs when four or more robots are very close to each other and try to avoid collisions. In this case the robots jam each other. Another frequently encountered problem is illustrated in Figure 2. For the convenience of discussion, let's name the robots R_1 to R_6 from the upper-right corner to the lower-left corner in Figure 2(a), respectively. Figure 2(a) is a typical distribution where robots get closer to form a line segment. At that moment, R_3 swerves to its left to avoid R_4. R_4 moves downward to its goal location, which is the perpendicular drop to the line passing through the closest robot R_3 and furthest robot R_5. At the same time, the other robots go through a similar process. Figure 2(b) illustrates the distribution of the robots a few iterations later.

As the simulation continues, robots reconvene into a distribution similar to the one shown in Figure 2(a). Robots repeat the motion sequences and fail to form a line segment.

It is observed that robots form a line only when the initial distribution is very close to a line segment and little or no collision avoidance is required. Finally, it is noted that a uniform distribution of robots along the line segment cannot be accomplished using this method.

Aiming to overcome the collision avoidance problem encountered in the above implementation, we replaced the left-swerve strategy with the potential field method. It turns out that the potential field method does not help at all. The robots show various group behaviors other than forming a line segment. The result of a simulation is discussed below.

Figure 3(a) shows the initial, random distribution of six robots, while Figures 3(b) to (e) illustrate their progressive movements. As soon as the simulation begins, robots start getting closer to each other as a natural result of the algorithm. A problem occurs when point p is within the physical dimensions or repulsive force range of a robot, or between two robots that do not have enough room in-between for another robot. Unfortunately this happens in most

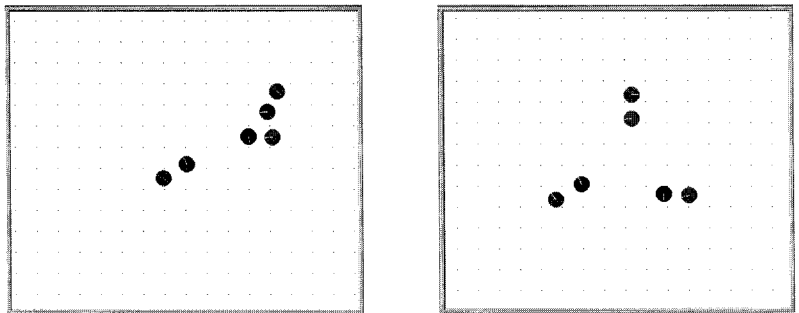

Figure 2. Implementation of the existing algorithm with the left-swerve collision avoidance strategy.

simulations, unless the initial distribution is very close to a line segment.

Eventually robots approach a deadlock configuration as shown in Figure 3(f). In this configuration, the attractive force generated by goal point p is negated by the repulsive force generated by surrounding robots. Consider the top robot in Figure 3(f), for instance. Its furthest robot R_f is the upper one in the two-robot group (it cannot see the lower one in the group), and its closest robot R_c is the closest one in the usual sense. Goal point p in this case corresponds to a point within the closest robot's repulsive force range. This is also true for all other robots in the four-robot group.

2.2. Modified Line Algorithm

Although the existing algorithm does not work well for physical robots to form a line segment, its main idea is still valid. As discussed above, a problem occurs if the goal point p on the line passing through the closest (R_c) and furthest (R_f) robots is occupied by another robot, or if there is not enough room for another robot at the goal point. To circumvent the problem, we modify the algorithm so that the goal point p is still chosen to be on the same line, but at a location where there is room for another robot. Since each robot executes the same program, the discussion below is concerned with a robot called R for convenience, which can be any one of the robots. At each iteration, robot R does the following:

Step 1. Determines the closest robot R_c and furthest robot R_f based on its sensor readings.

Step 2. Determines if point p, the foot of the perpendicular drop from its current position to the line ℓ passing through R_c and R_f, is between R_c and R_f.

Step 3. If yes, and if there is enough room for robot R to fit in between R_c and R_f, it proceeds toward the mid-point, which is denoted by p_m.

Step 4. If p is not between R_c and R_f, or if there is not enough room between R_c and R_f, it proceeds towards point p_d on the line ℓ that is d_{min} distance away from R_c in the opposite direction of R_f, where d_{min} is the minimum distance that would prevent any repulsive force being applied to either robot.

Figure 4 shows the results of a simulation of this algorithm. Figure 4(a) is the same starting distribution as in Figure 3(a). Figures 4(b) through (e) show some selected intermediate distributions while Figure 4(f) illustrates the final distribution of robots.

Some comments on the algorithm are in order. As mentioned earlier, the potential field algorithm is utilized to avoid collision. If we use d_o to denote the cut-off distance of repulsive forces in the potential field algorithm, any obstacles (in this case other robots) that are less than d_o distance away from robot R will generate repulsive forces to robot R. In Step 3, when determining if there is enough room to fit another robot between R_c and R_f, the distance from R_c and R_f must be at least

$$|R_cR_f| = 2d_o \tag{1}$$

which is empirically determined based on simulations. That is, if there is at least $2d_o$ distance between R_c and R_f, robot R will be able to "squeeze" in. This is true even if there are other robots between R_c and R_f.

In Step 4, the distance d_{min} is given by

$$d_{min} = r_o + d_o \tag{2}$$

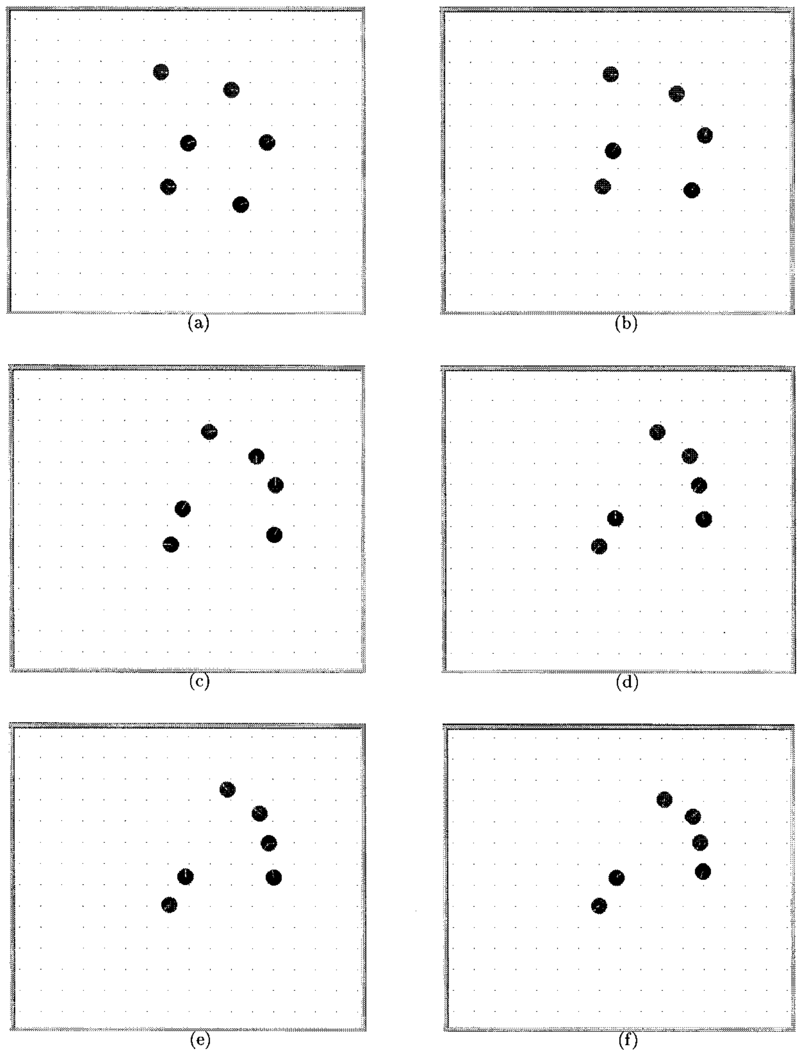

Figure 3. Selected images of the existing line algorithm simulation using the potential field method from initial distribution to final stage: (a) the initial distribution, (b)–(e) intermediate steps, and (f) the final distribution of the robots trying to form a line segment. The two robots on the lower-left side are approximately where they should be while the remaining ones are in deadlock.

where r_o is the radius of the robots. (A Nomad 200 robot is cylindric in shape. In the Simulator, it is represented by a circle.) There is, however, a problem in implementing Step 4 if point p is between R_c and R_f. Sending robot R directly to p_d mostly results in a local minimum while the robot tries to move to the other side of R_c. This happens if the closest robot to R_c is R. R_c and R move head to head and lock in a local minimum. We avoid this problem by sending the robot to an intermediate point that is d_{min} distance away from R_c, on the line that is perpendicular to ℓ at R_c. There are two points on this perpendicular line that are d_{min} distance away from R_c, but there is an obvious choice (the one which is closer to robot R). As soon as robot R reaches this intermediate point within a close proximity, its goal point is changed to p_d.

If robot R detects only one robot nearly (which is the case if it is at the endpoint of the line segment), it positions itself d_o distance away from the detected robot. If robot R does not detect any robots nearby, it can execute an algorithm to search for possible existence of other robots, which is not implemented here.

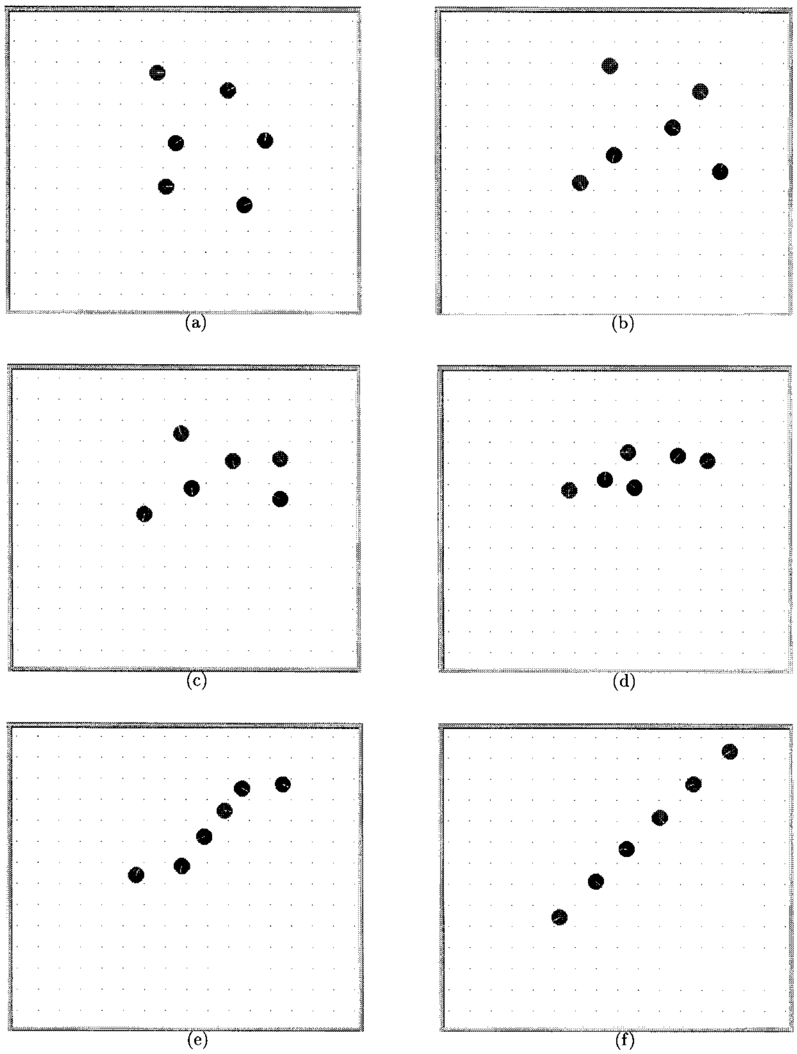

Figure 4. Selected images of the modified line algorithm simulation: (a) the initial distribution, (b)–(e) intermediate steps, and (f) the final distribution of the robots forming a line segment.

To confirm the effectiveness of the modified algorithm, it was run from the deadlock configurations that resulted from the original algorithm. The modified algorithm is able to break these deadlocks and accomplish the line formation task. Finally it is noted that all robots will uniformly distribute in the line segment because each robot tries to go to the midpoint between its neighbors until it gets into the repulsive force range of its neighbors.

2.3. Least-Square Algorithm

The existing line algorithm and the modified one described in preceding subsections only utilize position information of the furthest and closest robots. In this subsection, we describe a least-square line algorithm that utilizes position information of all robots seen by each robot.

The basic idea is that, at each iteration, each robot finds the least square line fitting of all visible robots and moves toward this line. It is emphasized that there is not a common coordinate system for all robots. Each one uses its own coordinate system to compute the line fitting. It is also noted that at each instant of time each robot may see a different number of robots.

Assume that robot R sees n robots in its surrounding at the current instant of time. The positions

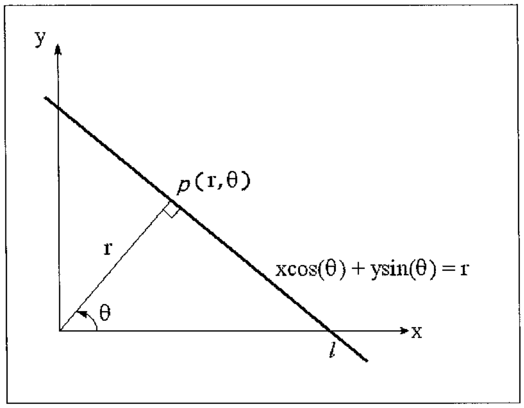

Figure 5. Parametric representation of lines using r and θ.

of the visible robots are represented in the coordinate system of robot R as n pair of data, (x_i, y_i), $i = 1, 2, \ldots, n$. We will also include the coordinates (x_o, y_o) of robot R itself in the line fitting computation. Following the standard numerical procedure,[20] we may find the least-square line fitting of the $(n + 1)$ pair of data:

$$y = ax + b \tag{3}$$

Nevertheless, this representation of lines has a singularity when the resulting line is parallel to the y-axis. Instead, we use a parametric representation of lines[21]

$$x \cos(\theta) + y \sin(\theta) = r \tag{4}$$

where r and θ are two parameters depicted in Figure 5.

After finding θ and r, the robot is directed to move to point p on the line as shown in Figure 5. It is noted that the robot does not check if there is enough room at point p in this case. Because the robot utilizes position information of all visible robots, it is able to squeeze in, even if there are other robots at or near point p. Figure 6 depicts a simulation of the least-square line algorithm. Figure 6(a) shows the initial distribution, which is the same as in Figure 3(a) and Figure 4(a). Once again we used the potential field algorithm for collision avoidance in our simulation. It is interesting to note that, starting from the same initial distribution as in Figure 4(a) and Figure 6(a), the two algorithms form line segments of different slopes.

3. CIRCLE FORMATION ALGORITHMS

In this section we first introduce the existing circle formation algorithm, and simulation results from implementing the existing algorithm. Observing problems encountered with the existing algorithm, we propose a sequence of modification and improvement. The improvement is directed towards forming a better approximation of a circle, uniformly distributing robots on a circle, forming circles with large radii, and forming circles under limited sonar range.

3.1. The Existing Circle Formation Algorithm

Let robot R by any one of the distributed robots participating in the task of circle formation. The existing circle algorithm works as follows.[1,4] As before, robot R becomes active and inactive at random instants of time. Each time robot R becomes active, it:

Step 1. Determines the furthest robot R_f and closest robot R_c.
Step 2. Calculates the distance d from its current position to the middle point p_m between R_c and R_f.
Step 3. Moves a distance of $\min\{d - r, v\}$ toward p_m if $(d - r) \geq 0$, or a distance of $\min\{r - d, v\}$ away from p_m if $(d - r) < 0$, where v is the maximum distance that a robot can move at a time, and r is the desired radius of a circle to be formed.

Figure 7 shows the results of a simulation of this algorithm. The desired radius of the circle is 28.0 inches. (The radius of the Nomad robot is 9.0 inches.) Figure 7(a) is the initial, random distribution of robots. Figures 7(b) to (e) show the intermediate positions of robots, and Figure 7(f) illustrates the final stage of the simulation. The final distribution of robots is a good approximation of a circle, and robots are fairly uniformly distributed. However, the degree of uniformity depends on the number of robots. Figure 8(a) shows the final distribution of five robots. The distribution is apparently less uniform.

A minor problem is that the radius of the final circle is always smaller than the desired radius (20 inches versus 28 inches in this case). This is because p_m does not correspond to the origin of the circle. Consequently, the final formation appears as two half-circles put together. In Figure 7(f), the three

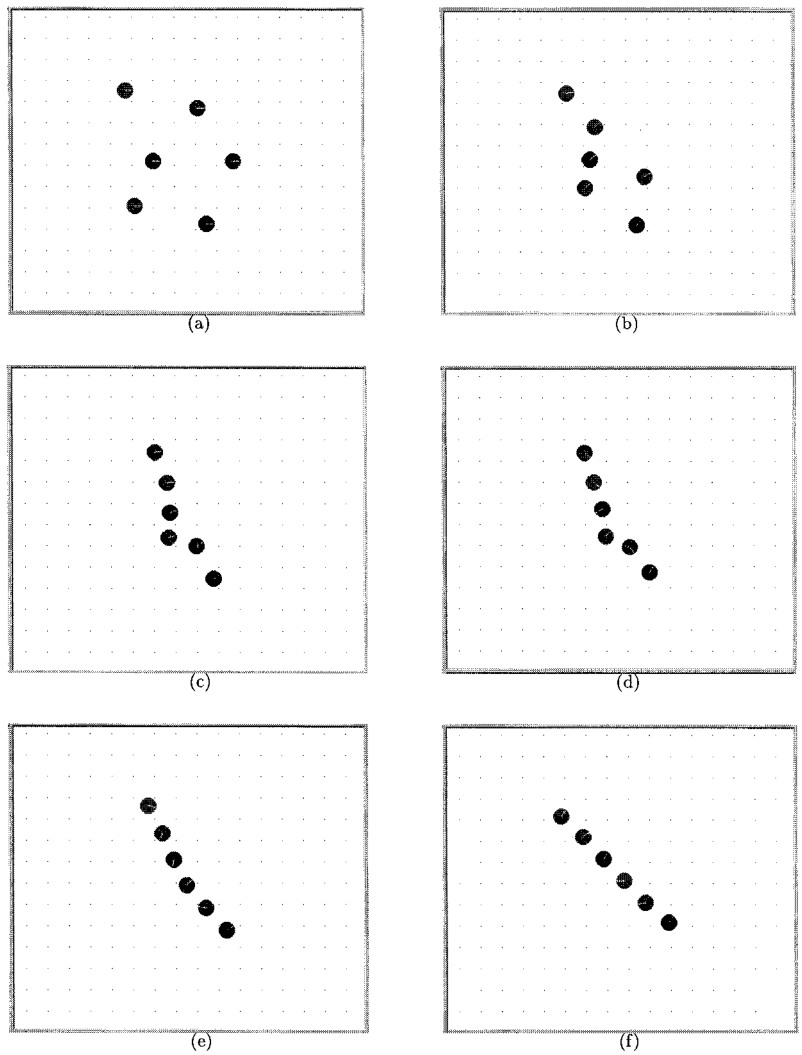

Figure 6. Selected images from a simulation of the least-square algorithm: (a) the initial distribution, (b)–(e) intermediate steps, and (f) the final distribution of the robots.

lower-left robots form a half-circle, as do the three upper-right robots. This is the same source that causes robots to form a Reuleaux's triangle.[1,4]

Another problem occurs when the desired radius becomes relatively large. With limited sonar range, a robot is not able to see some robots as it moves outwards to form a large circle. With the same initial distribution as in Figure 7(a), a simulation is carried out to form a circle with radius of 120 inches. The resulting distribution is shown in Figure 8(b). The pair of robots at the upper-right corner cannot see the two at the lower-left corner due to limited sensor range. In this case, the two robots at the upper-right corner and the two in the middle form a circle. Similarly, the two robots at the lower-left corner and the two in the middle form another circle.

3.2. Modified Circle Algorithm

In this subsection, we present a modified circle algorithm. The objective of the modified algorithm is to yield a better approximation of circles, and to uniformly distribute robots. The existing algorithm utilizes position information of the furthest and closest robots. We attempt to improve it by utilizing position information of one more robot. More specifically, the first two steps of the modified algorithm are as follows. At each iteration, robot R:

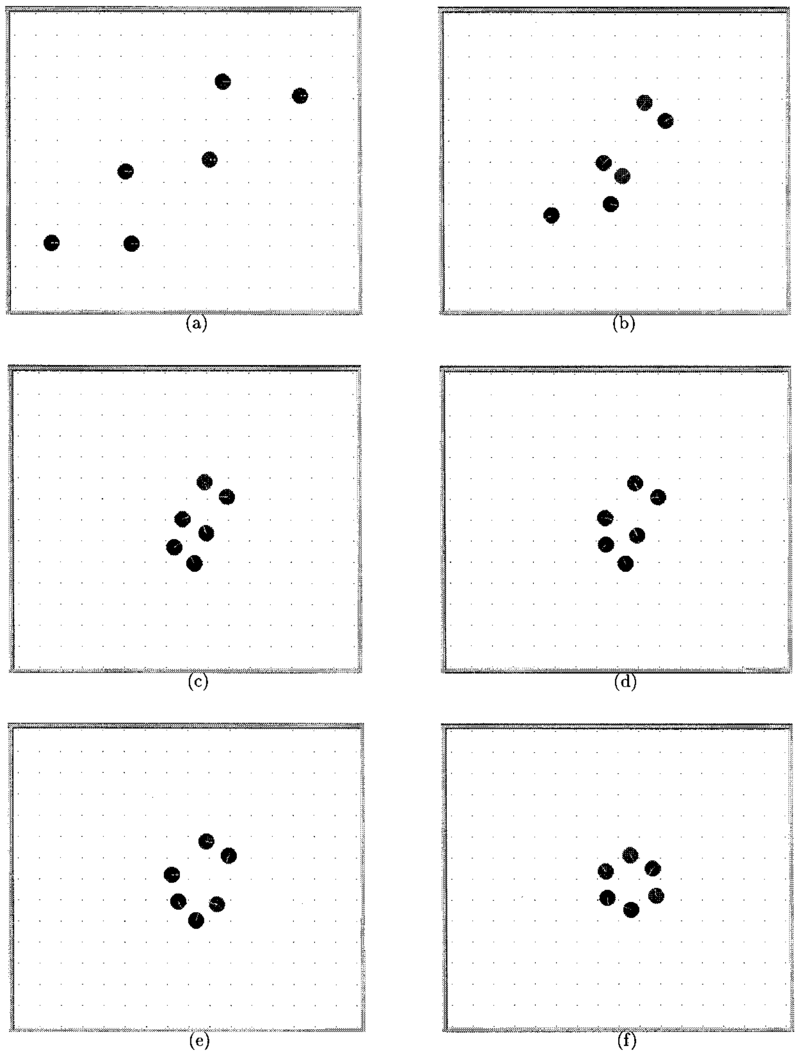

Figure 7. Selected images from a simulation of the existing circle algorithm implemented by using the potential field method: (a) the initial distribution, (b)–(e) intermediate steps, and (f) the final distribution of the robots forming a circle.

Step 1. Determines the furthest robot R_f, the closest robot R_{c1}, and the second closest robot R_{c2}.

Step 2. Calculates the distance d from its current position to the centroid p_m of R_f, R_{c1}, and R_{c2}.

The third step is the same as the existing algorithm. Figure 9 shows the results of a simulation of the modified algorithm with a desired circle of 28-in radius. An advantage of this approach is that p_m is closer to the origin of the desired circle, which makes the final formation a much better approximation of a circle. The radius of the resulting circle is still smaller than the given radius (21 inches versus 28 inches).

Another advantage is that robots will be uniformly distributed along a circle, independent of the number of robots. Nevertheless, it is observed that smaller number of robots tend to form a smaller circle. Figure 10 shows the final results with four and five robots.

3.3. Merge-Then-Circle Algorithm

In this subsection, we present an algorithm that allows robots to form a relatively large circle. Since sonar ranges are limited, a robot will not be able to see robots on the other side of a circle if the radius is relatively large. We describe an algorithm in which

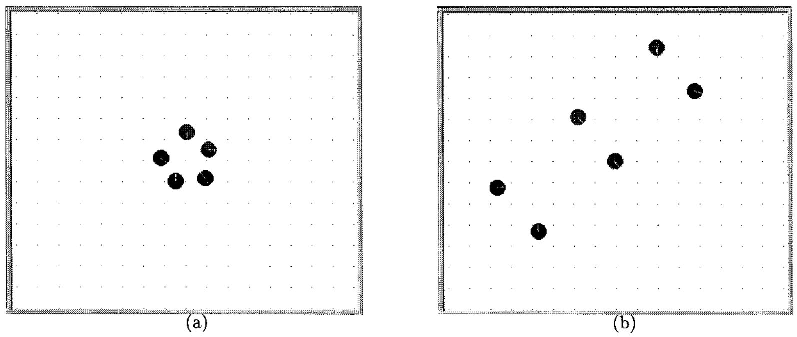

Figure 8. The final distributions of two more simulations of the existing circle algorithm started from the same initial distribution as the earlier simulation: (a) If a robot is missing, the remaining robots still form a circle, but are not uniformly distributed (r = 28 in). (b) The robots railed to form a circle for a relatively large desired radius (r = 120 in).

each robot relies on position information of the two closest robots, and does not use position information of the furthest robot. In this way, robots are able to form a circle with a diameter greater than the sonar range limit.

The algorithm is divided into two stages: first converge all robots into a single cluster and then diverge them from the cluster to form a circle. The algorithm works as follows:

Step 1. Robot R moves to the middle point between the furthest robot R_f and closest robot R_c.

Step 2. If the speed of robot R is less than some small value (1 inch/sec in our simulation) for N successive iterations, robot R goes to sleep. It wakes up after T seconds to get the sensor data to determine the empty spaces around and sleeps again for another T seconds. T is empirically determined in simulations.

Step 3. After waking up, if robot R sees an empty area based on the sensor data it got between the two sleep periods, it moves r distance toward the middle of the empty area and goes back to sleep for another period of T seconds. If there is no empty space around, i.e., it is surrounded by other robots, it disregards its previous data collected between two sleep periods. It searches the surrounding area to look for an empty space. As soon as an empty space is detected, the robot travels (r + d_o + r_o) toward the center of the empty space and then sleeps for T seconds.

Step 4. Let R_{c1} and R_{c2} be the closest robots to robot R, one on each side of a line passing through from its position in the merged cluster to its present position. After waking up, robot R moves toward R_{c1} or R_{c2} until the distance to them are equal.

In Step 2, robot R goes to sleep after N successive iterations, which happens when all robots are nearly merged. By waiting T seconds, robot R ensures all the robots are merged. Taking a snapshot between the two sleep periods makes certain that all robots collect data before anyone wakes up. The sleep time at the end of Step 3 ensures that all robots reach their goal positions and form a rough circle. In Step 4, robots may use the modified circle algorithm if the given radius is small.

Figure 11 shows a simulation result of the merge-then-circle algorithm with a desired radius r = 120 inches. Figure 11(a) is the initial starting distribution. Figure 11(d) is the merged cluster after Step 2. Figure 11(e) is the rough circle after Step 3. Figure 11(f) is the final distribution of robots on a circle after Step 4.

3.4. Limited Range Algorithm

In this subsection, we consider a scenario where robots are initially randomly placed in a large rectangular field. The field is so large that a robot may not see other robots due to limited sensor range. The objective is again to form a circle with a given radius. Even though the field is assumed to be rectangular in shape, its size is unknown. For all robots in the large field to form a circle, one possible method is to have each robot search for all other robots and then execute a circle formation algorithm. We propose an

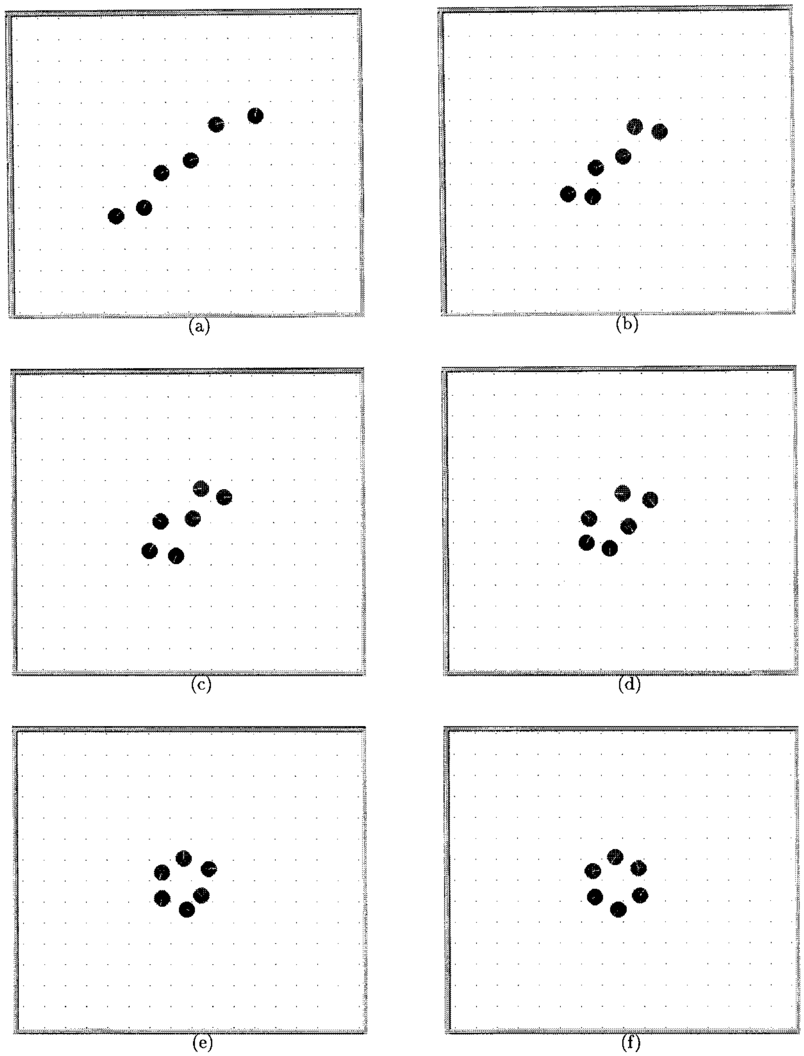

Figure 9. Selected images of a simulation of the modified circle algorithm: (a) the initial distribution, (b)–(e) intermediate steps, and (f) the final distribution of the robots forming a circle.

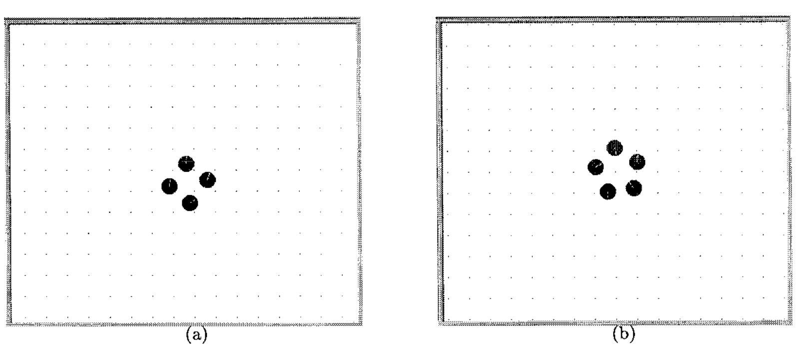

Figure 10. Final distributions of the modified circle algorithm simulations with four and five robots.

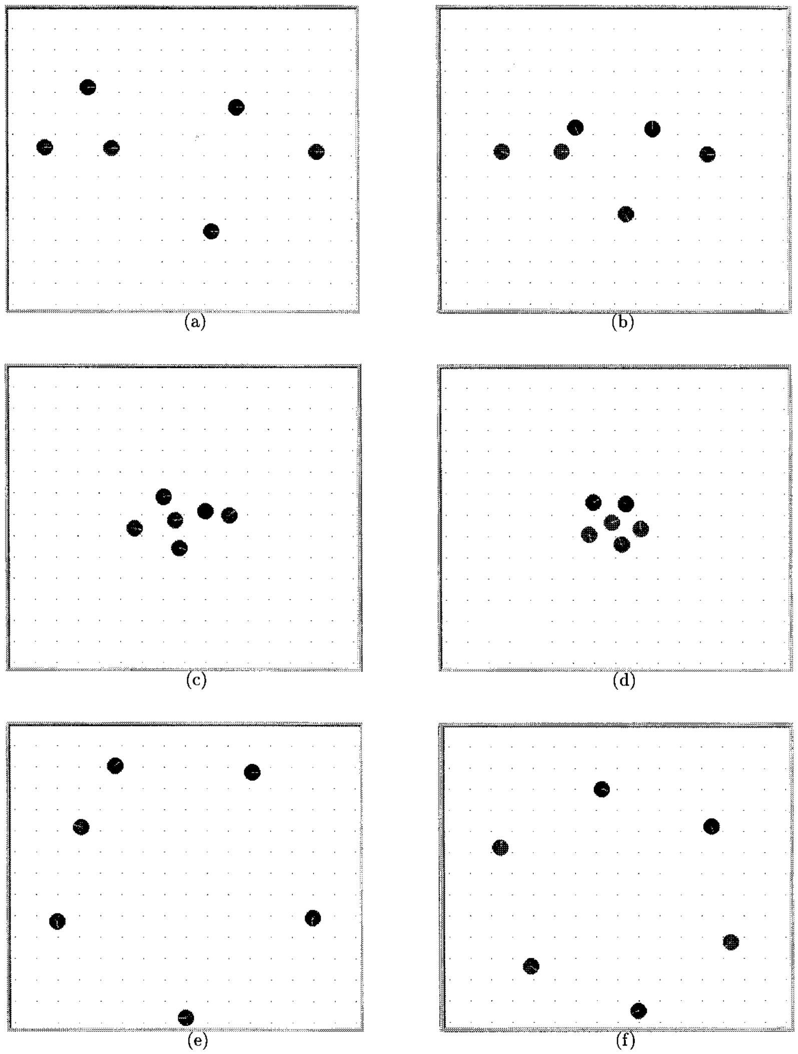

Figure 11. Selected images of a simulation of the merge-then-circle algorithm: (a) the initial distribution, (b)–(e) intermediate steps, and (f) the final distribution of the robots.

alternative method that is based heavily on the fact that the field is rectangular. All robots converge to the center of the field before executing a circle formation algorithm. This method can be described as follows:

Step 1. Starting from its initial position, robot R moves straight until it reaches a wall (an edge of the field). It may need to avoid other robots before reaching a wall.

Step 2. Robot R follows the edges of the field in counterclockwise direction until it has encountered three corners. It records the coordinates of the first and third corners.

Step 3. It computes the center point p_m of the field, which is the middle point between the first and third corners.

Step 4. It converges to p_m and goes to sleep for T seconds. The sleep mode is waiting until all robots converge. Time T is determined by a worst case analysis.

Step 5. After waking up, it executes the latter half of the merge-then-circle algorithm.

Figure 12 depicts a simulation result of this algorithm. Figure 12(a) shows an initial distribution of robots. In Figure 12(b) robots are following edges of the field. Figure 12(c) is a merged cluster at the center of the field. Figure 12(d) is a rough circle occurring in the intermediate steps of merge-then-circle algorithm. Figure 12(e) is the final distribution of the robots on a circle after completion of Step 5.

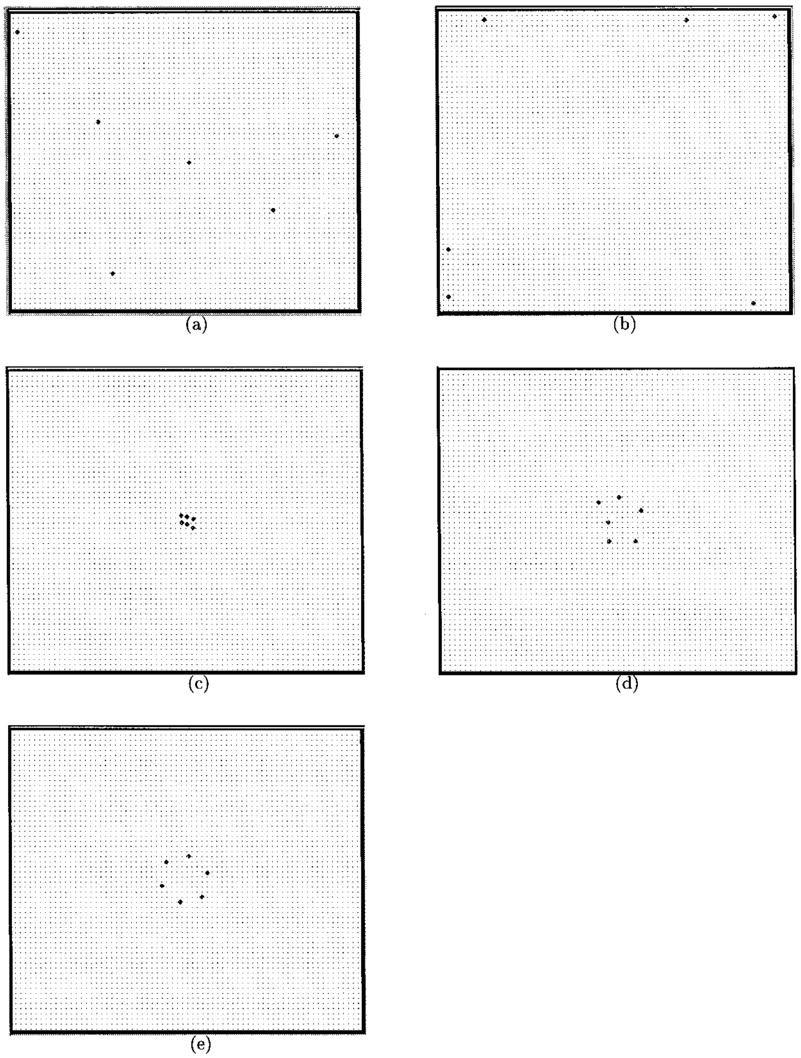

Figure 12. Selected images of a simulation of the limited range algorithm: (a) the initial distribution, (b)–(d) intermediate steps, and (e) the final distribution of the robots.

4. CONCLUSION

Line and circle formation of distributed mobile robots was studied. It was observed that existing line and circle formation algorithms do not work well when implemented with realistic robots. Modified version of the existing algorithms as well as new algorithms were described and verified through simulation. As demonstrated, the proposed algorithms not only achieved the goal of forming a line or a circle; they also uniformly distribute robots on a line segment or circle. Continuing work focuses on improving the convergence rate of formation and incorporating obstacles (other than robots themselves) in the workspace.

This work is in part supported by NSF grants IRI-95-96026 and CDA-95-96021, and the NPS RIP grant. The authors would like to thank Professor Ichiro Suzuki for valuable discussions on the formation problem of distributed mobile robots.

REFERENCES

1. K. Sugihara and I. Suzuki, ''Distributed motion coordination of multiple mobile robots,'' *Proc. IEEE Int. Symp. Intell. Control,* Philadelphia, PA, 1990, pp. 138–143.
2. I. Suzuki and M. Yamashita, ''Formation and agreement problems for anonymous mobile robots,'' *Proc. Annu. Allerton Conf. Commun. Control Comput.,* University of Illinois, Urbana, IL, 1993, pp. 93–102.

3. H. Ando, I. Suzuki, and M. Yamashita, ''Formation and agreement problems for synchronous mobile robots with limited visibility,'' *Proc. IEEE Int. Symp. Intell. Control,* Monterey, CA, 1995, pp. 453–460.
4. K. Sugihara and I. Suzuki, ''Distributed algorithms for formation of geometric patterns with many mobile robots,'' *J. Rob. Syst.,* **13**(3), 127–139, 1996.
5. J. C. Latombe, *Robot Motion Planning,* Kluwer Academic Publishers, Norwell, MA, 1991.
6. Y. Koren and J. Borenstein, ''Potential field methods and their inherent limitations for mobile robot navigation,'' *Proc. IEEE Int. Conf. Rob. Autom.,* 1991, pp. 1398–1404.
7. T. Arai, H. Asama, T. Fukuda, and I. Endo, *Distributed Autonomous Robotic Systems,* Springer Verlag, New York, 1994.
8. T. Ueyama, T. Fukuda, G. Iritani, and F. Arai, ''Optimization of group behavior on cellular robotics system in dynamic environment,'' *Proc. IEEE Int. Conf. Rob. Autom.,* San Diego, CA, 1994, pp. 1027–1032.
9. M. Inaba, Y. Kawauchi, and T. Fukuda, ''A principle of decision making of cellular robotics system (CEBOT),'' *Proc. IEEE Int. Conf. Rob. Autom.,* vol. 3, Atlanta, GA, 1993, pp. 833–838.
10. J. Wang and G. Beni, ''Cellular robotics systems: Self organizing robots and kinetic pattern generation,'' *Proc. IEEE Int. Workshop Intell. Rob. Syst.*, Tokyo, Japan, 1988, pp. 139–144.
11. T. Fukuda and S. Nakagawa, ''Approach to the dynamically reconfigurable robot systems,'' *J. Intell. Rob. Syst.*, 1, 55–72, 1988.
12. M. J. Mataric, ''Minimizing complexity in controlling a mobile robot population,'' *Proc. IEEE Int. Conf. Rob. Autom.,* Nice, France, 1992, pp. 830–835.
13. P. Tournassoud, ''A strategy for obstacle avoidance and its application to multi-robot systems,'' *Proc. IEEE Int. Conf. Rob. Autom.,* San Francisco, CA, 1986, pp. 1224–1229.
14. M. J. Mataric, ''Designing emergent behaviors: From local interactions to collective intelligence,'' in *From Animals to Animats 2: International Conference on Simulation of Adaptive Behavior,* MIT Press, Cambridge, MA, 1993, pp. 432–441.
15. Q. Chen and J. Y. S. Luh, ''Coordination and control of a group of small mobile robots,'' *Proc. IEEE Int. Conf. Rob. Autom.,* San Diego, CA, 1994, pp. 2315–2320.
16. J. Borenstein and Y. Koren, ''Real-time obstacle avoidance for fast mobile robots,'' *IEEE Trans. Syst. Man Cybern.,* **19**(5), 1179–1187, 1989.
17. L. Parker, ''Adaptive action selection for cooperative agent teams,'' In *From Animals to Animats 2: International Conference on Simulation of Adaptive Behavior,* MIT Press, Cambridge, MA, 1993, pp. 442–450.
18. R. C. Arkin, ''Motor schema based mobile robot navigation,'' *Int. J. Rob. Res.,* **8**(4), 92–112, 1989.
19. I. Suzuki and M. Yamashita, ''A theory of distributed anonymous mobile robots—Formation and agreement problems,'' Technical Report TR-94-07-01, Department of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, 1994.
20. S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers,* McGraw-Hill, New York, 2nd ed., 1988.
21. R. O. Duda and P. E. Hart, ''Use of the Hough transformation to detect lines and curves in pictures,'' Communications of the ACM, 15(1), 11–15, 1972.