A Demonstration of a Robot Formation Control Algorithm and Platform

Ross Mead*, Jerry B. Weinberg*, and Jeffrey R. Croxell†

Southern Illinois University Edwardsville
*Department of Computer Science
†Department of Electrical and Computer Engineering
Edwardsville, IL 62026-1656
qbitai@gmail.com, jweinbe@siue.edu, jcroxel@siue.edu

Abstract

Coordinating a group of robots to work in formation has been suggested for a number of tasks, such as urban search-and-rescue, traffic control, and harvesting solar energy. Algorithms for controlling robot formations have been inspired by biological and organizational systems. In our approach to robot formation control, each robot is treated like a cell in a cellular automaton, where local interactions between robots result in a global organization. The algorithm has been demonstrated in both simulated (Mead & Weinberg 2006) and physical environments (Mead *et al*. 2007). In this paper, we present a detailed insight into the algorithm and its implementation.

Introduction

Robots organizing and working in formation has been suggested for a number of tasks, such as systematic search-and-rescue (Tejada *et al*. 2003), automated traffic cones for road construction (Farritor & Goddard 2004), and construction of a large orbiting solar reflector for harvesting solar energy (Bekey *et al*. 2000). Work on formations has been inspired by biological and organizational systems, such as the flying patterns of geese or marching bands (Fredslund & Mataric 2002, Balch & Arkin 1998).

Formation Definition

Our approach is to treat the formation as a type of cellular automaton, where each robotic unit is a cell (Mead & Weinberg 2006). The robot's behavior is governed by a set of rules for changing its state with respect to its neighbors. By selecting one of the robots as an "initiator", human intervention would change its state, which would propagate to its neighbors, instigating a chain reaction.

Each robot is represented as a cell c_i in a 1-dimensional cellular automaton, where i refers to the index within the automaton; note that an index is not necessarily the robot's identification number or address that is used for communication—it is simply a reference.

Each cell is in a neighborhood, denoted $\{c_{i-1}, c_i, c_{i+1}\}$ or $\{c_{i-1}, c_i, c_{i+1}\}$, where c_{i-1} and c_{i+1} refer to the left and right neighbors of c_i, respectively. Likewise, a particular neighbor is denoted c_j, where j is the index of the cell representing a neighboring robot. It follows that, by combining neighborhoods, the entire automaton can be written as $\{\ldots, c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}, \ldots\}$.

A desired formation F is defined as a geometric description (in the current implementation, a single mathematical function) $f(x)$. This definition is sent to some robot, designating it as the *seed* cell c_{seed} of the automaton. For purposes of determining relationships, a cell considers itself to be at some function-relative position p_i:

$$p_i \leftarrow \langle x_i, f(x_i)\rangle \tag{1}$$

In the case of c_{seed}, the position p_{seed} is given and serves as a starting point from which the formation and relationships will propagate.

The desired relationships between cells (robot) are determined by calculating a vector v from p_i to the intersection of $f(v_x)$ and a circle centered at p_i with radius R, where R is the desired distance to maintain between neighbors in the formation:

$$R^2 \leftarrow (v_x - p_{i,x})^2 + (f(v_x) - p_{i,y})^2 \tag{2}$$

$$r_{i\rightarrow j,des} \leftarrow \langle v_x, f(v_x)\rangle \tag{3}$$

Solving for the *desired relationship vector* $r_{i\rightarrow j,des}$ to some neighbor c_j results in two intersections: one in the positive direction and one in the negative direction. These solutions define right and left neighbor relationships $r_{i\rightarrow i+1,des}$ and $r_{i\rightarrow i-1,des}$, respectively (Figure 1).

The formation definition and relationship information are communicated locally within the neighborhood. Neighboring robots repeat the process, but consider themselves to be at different function-relative positions as determined by the desired relationship from their neighbor. For a neighbor c_j:

$$p_j \leftarrow p_i + r_{i\rightarrow j,des} \tag{4}$$

$$r_{j\rightarrow i,des} \leftarrow -r_{i\rightarrow j,des} \tag{5}$$

Note that relationships $r_{j\to i,des}$ and $r_{i\to j,des}$ are equal in magnitude, but opposite in direction. This property of the algorithm is what guarantees convergence and stability between two robots attempting to establish and maintain relationships with one another.

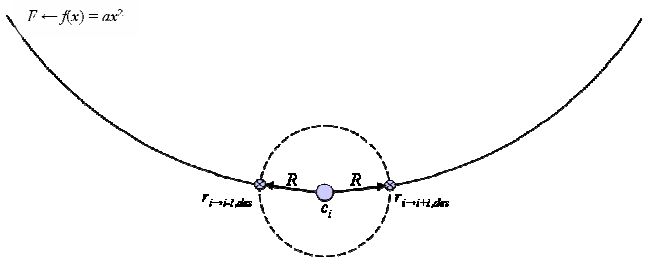

Figure 1: c_i calculates the desired relationship to its neighbors.

Using only sensor readings, robots calculate an actual relationship $r_{i\to j,act}$ with a neighbor c_j. The robot communicates locally (i.e., within the neighborhood) discrepancies in its desired and actual relationships to neighboring cells. Correcting for these discrepancies produces robot movements that result the overall organization of the desired global structure (Figure 2).

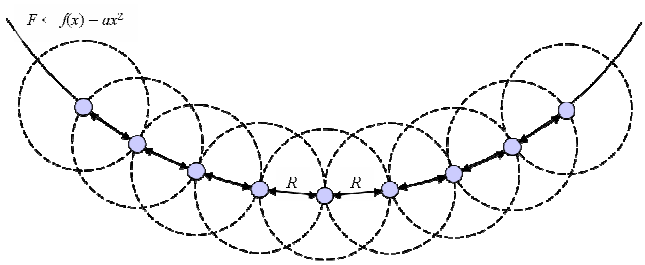

Figure 2: Calculated relationships between robots generate a parabolic formation.

An inherent aspect of the algorithm is that a movement command sent to a single robot will cause a chain reaction in neighboring robots, which then change states accordingly, resulting in a global transformation. Likewise, to change a formation, a seed cell is chosen and given the new geometric description and the process is repeated (Figure 3).

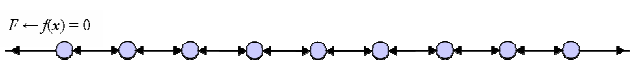

Figure 3: A new formation is given that describes a line.

Motion Control

By evaluating the state of the neighborhood, c_i is able to determine the translational error Γ_i and rotational error Θ_i that will define its movement, which is assumed to be along a safe trajectory. To do this, the robot must first seek a point of reference. Recall that the seed cell c_{seed} was given an initial function-relative position p_{seed}, from which the formation and relationships propagate. As the distance from c_{seed} increases, the propagated error accumulates. It follows that the neighbor of c_i whose function-relative position is closest to p_{seed} will have the least amount of propagated error within the system and, thus, will likely be the most reliable robot to reference. This reference neighbor c_k is then the neighbor that yields the minimum distance $\|p_k'\|$ from p_{seed} to p_k:

$$p_k' \leftarrow \min\{\|p_{i-1} - p_{seed}\|, \|p_{i+1} - p_{seed}\|\} \tag{6}$$

The rotational error Θ_i is a difference in robot orientation, which is propagated to each successive cell in the automaton. To determine Θ_i, c_i considers $r_{k\to i,act}$ with respect to itself. Let $\theta_{i\to k}$ represent the relative angle between the headings of c_i and c_k, and let θ_i and θ_k be the angles of the relationship vectors $r_{i\to k,act}$ and $r_{k\to i,act}$, respectively. Then:

$$\theta_{i\to k} \leftarrow \theta_k - \theta_i + 180°; [-180°, 180°] \tag{7}$$
$$\Theta_i \leftarrow \Theta_k + \theta_{i\to k}; [-180°, 180°] \tag{8}$$

Note that if $\theta_{i\to k} = 0°$, both i and k have the same global heading. The same holds true for every robot in the formation if $\Theta_i = 0°$ for all i. This property of the algorithm to yield an emergent global heading is essential for any subsequent formation commands from an operator. If a translational movement command is given, the common heading of the robots allows for a smooth transition in the same direction.

The appropriate translational movement for each robot in the automaton is determined by an accumulation of error with both x- and y-components. This error is determined by the difference in desired and actual relationships of reference cell c_k. One major consideration is that many of the robots are, themselves, correcting for translational and rotational errors while they are being referenced by other robots. Changes in the orientation of c_k can cause rather entropic behavior in c_i, which depends on it for motion control. To alleviate this problem, $r_{k\to i,des}$ must be rotated, accounting for the propagated rotational error Θ_k within the automaton. Let $r_{k\to i,des}'$ denote $r_{k\to i,des}$ rotated by an angle $-\Theta_k$. We express γ_i as the translational error of c_i with respect to c_k:

$$\gamma_i \leftarrow \Gamma_k + r_{k\to i,des}' - r_{k\to i,act} \tag{9}$$

Recall that $\theta_{i\to k}$ relates the headings of both of these robots, providing a conversion between the relative coordinate systems of c_i and c_k. Thus, rotating γ_i by $-\theta_{i\to k}$ yields the translational error Γ_i.

Simulator

The control algorithm was initially implemented in a simulated environment (Mead & Weinberg 2006). The simulator was written in C++ using OpenGL, and provides an easy means to visualize, manipulate, and test the algorithm. Recall that each robot is represented as a cell c_i, where i is the corresponding index within the automaton; in the simulator; the automaton is stored as a 1-dimensional array of n cells, where n is given at runtime. Tens-to-thousands of cells have been tested against various formation definitions to demonstrate the scalability and generality of the algorithm (Figure 4).

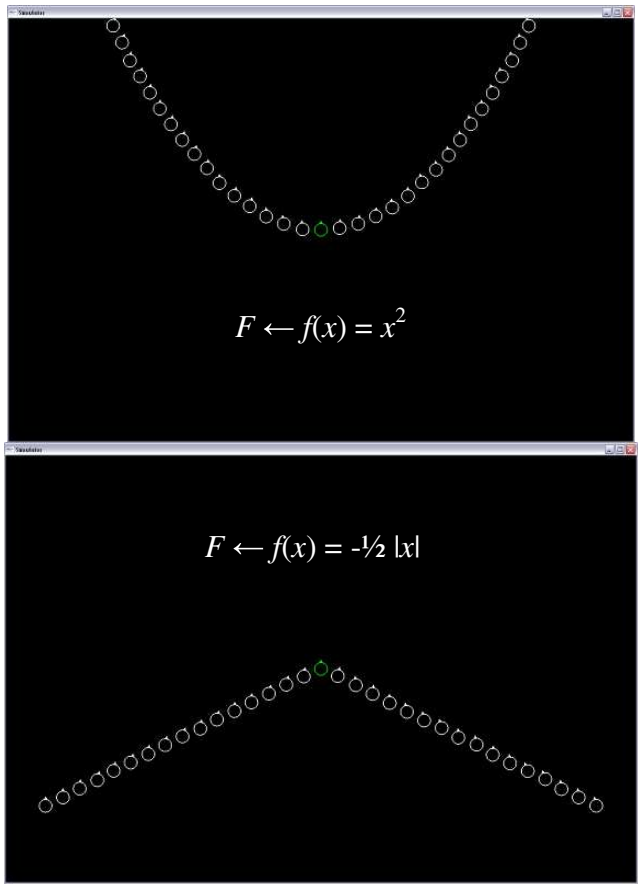

Figure 4: Two robot formations in simulation.

Figure 5: Cells change formation from a parabola to a sine curve (pictured in four steps in time).

To show dynamic switching capabilities of the algorithm, a human operator can send a variety of commands to a seed cell, thus, propagating changes in automaton. A *change formation command* redefines the desired geometric figure (Figure 5). *Translation commands* move the cells along the current formation heading. Rotation commands instigate changes in orientation and come in two forms: formation rotation and cell rotation. A *formation rotation command* causes a change in the orientation of the formation as a whole, thus, requiring each cell to move in such a way as to maintain the shape as it is rotated. In contrast, a *cell rotation command* modifies the orientation of each cell relative to the formation; the position and orientation of the formation itself remain unchanged. In addition, an operator may directly manipulate any cell, relocating or reorienting it to evaluate the stability of the system in the presence of error.

The simulator assumes perfect, continuous, and unlimited sensing. The next section discusses the physical implementation of this algorithm and how we overcame this assumption to prove that the approach is viable in the real world.

Robot Platform & Implementation

A platform was developed to test the algorithm in the physical world (Mead *et al*. 2007; Figure 6). Each robot is built upon a Scooterbot II base (www.budgetrobotics.com). The Scooterbot is 7 inches in diameter and is crafted from expanded PVC, making it durable and light. Two modified servo motors are employed for differential steering.

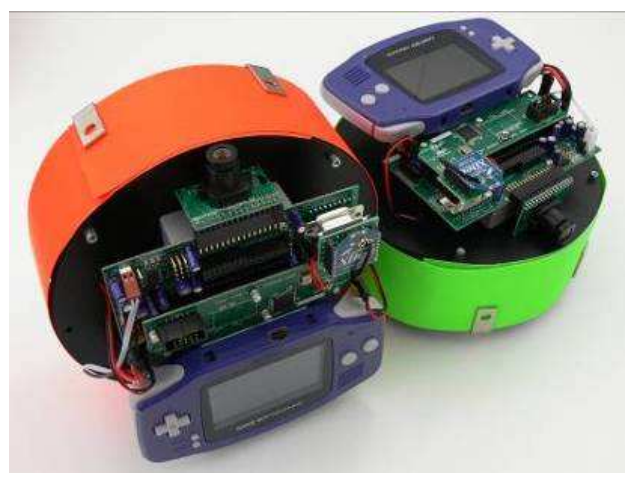

Figure 6: The robot platform with XBC controller.

The formation control algorithm is implemented in Interactive C (www.kipr.org/ic) and runs on an XBCv2 microcontroller (www.botball.org). The XBC utilizes back-EMF PID for accurate motor control. It also features a camera, capable of multi-color, multi-blob simultaneous tracking. Rotating the robot provides a 360° view of the environment and neighboring robots.

Neighbor Localization

A neighboring robot represented by c_j is identified by either an orange or green color band; the color of each robot is assigned based on its ID: green for even; orange for odd. The alternating of color bands reduces the chances of overlapping color blobs and improves the accuracy of detecting a neighbor.

To locate c_j, a robot represented by c_i rotates until the band of the appropriate color is within its view; it then centers on that band. The heading of c_i is always considered to be directed at the x-axis (0°); relative to the robot, left yields positive angles and right yields negative angles. The distance $d_{i\to j}$ between c_i and c_j is determined by recognizing that the perceived vertical displacement Δy between the top and bottom of a color band is proportional to the perceived vertical displacement ΔY at a known physical distance D (Figure 7):

$$d_{i\to j} \leftarrow D \times \Delta Y / \Delta y \qquad (10)$$

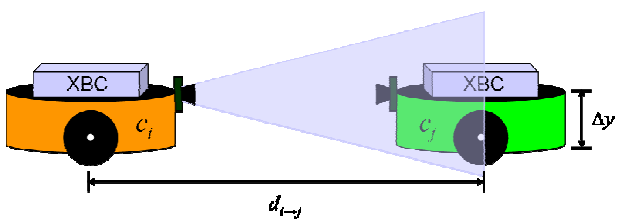

Figure 7: c_i identifies and determines its distance to c_j.

The relative orientation $\alpha_{i\to j}$ from c_i to c_j is simply the angular displacement from the initial location (i.e., prior to the search) of c_i. Thus, the *actual relationship vector* $r_{i\to j,act}$ is written in polar coordinates as:

$$r_{i\to j,act} \leftarrow \langle d_{i\to j}, \alpha_{i\to j} \rangle \qquad (11)$$

Communication

A radio communication module is used to share state information within a robot's neighborhood. The XBee (ZigBee/IEEE 802.15.4 compliant) was chosen for its rich feature set, transparent operation, and high level API [www.maxstream.net]. The ZigBee protocol does not require a host/slave configuration like many similar technologies, allowing for more flexibility in networking topologies such as mesh networking, broadcast mode, and packet rerouting. The XBee also scales well for large applications, using 16-bit addressing to provide for over 65,000 nodes. The low-power model, which we utilized, offers a range of 100 meters, which is equivalent to class-2 Bluetooth, while the XBee Pro variation would allow us to transmit over a range of one mile without any changes to hardware or software.

To use the XBee chips with our XBC microcontrollers several design considerations had to be taken into account. The XBee communicates uses a TTL-level UART, while the XBC uses RS-232 levels. Also, the foot print of the XBee requires interfacing to the serial port on the XBC. To accommodate for the RS-232 levels, a level translation circuit was designed (Figure 8) using a MAX 3221 Transceiver (www.maxim-ic.com). Ten pin single row headers with 2mm spacing matched the XBee so that it could plug into this interface board. The schematic was used to generate a printed circuit board layout as seen in Figure 8. All parts were surface mounted, with the exception of a 9-pin D-Sub male plug that sat on the XBC. The result is a board smaller than the XBee itself that directly plugs into the XBC's 9-pin serial port (Figure 9).

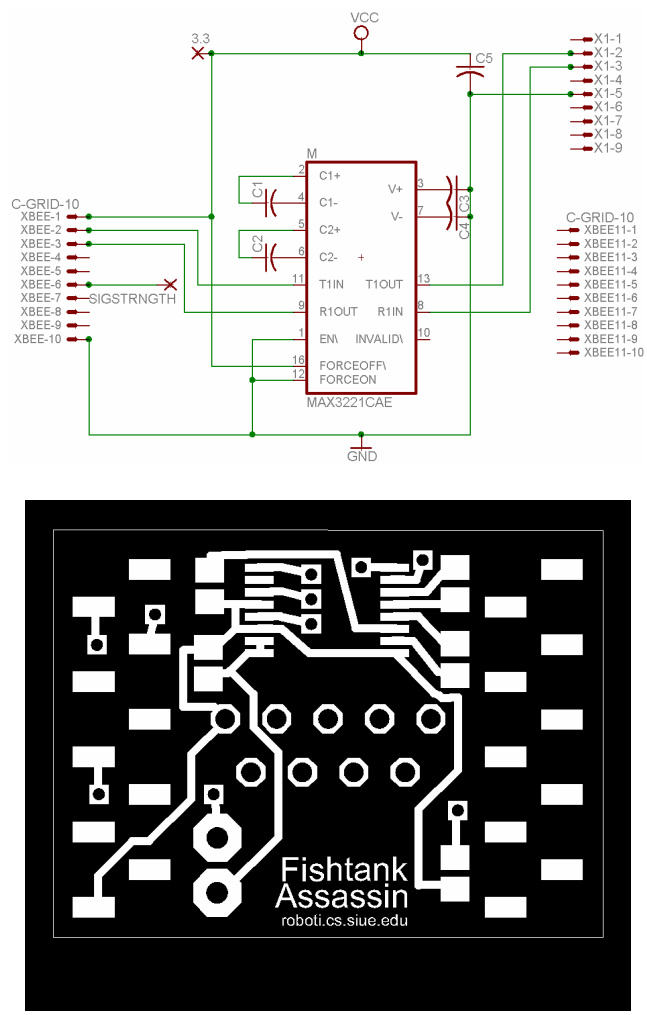

Figure 8: Level translation circuit (top) and the top layer of the PCB interface board (bottom).

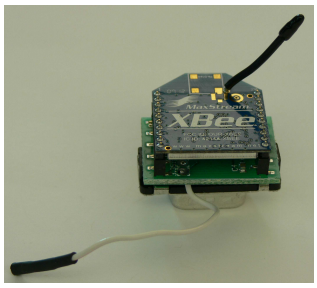

Figure 9: XBee wireless communication module.

A reliable packet communication protocol was implemented that supports automatic retries and acknowledgements. Packets can be addressed to a particular robot or broadcast. This allows each robot to communicate locally within its neighborhood.

Evaluation

The formation control algorithm is implemented on a modest number of robots (Figure 10). We are currently working on a series of experiments that will be conducted and evaluated based on the criteria discussed in Fredslund & Mataric (2002).

Figure 10: Eleven robots in a parabolic formation defined by the geometric description $F \leftarrow f(x) = x^2$.

Future Work

If the robots are not initially put in a formation, then a neighborhood must be established dynamically. This will be accomplished by implementing a market-based auctioning method, where a robot is chosen to be a neighbor based on its distance to the desired relative location in the formation description. For this to be implemented, a robot must be able to identify and track its neighbors. This is easier said than done, as each unit looks identical. We alleviated these problems by utilizing a colored bar-coding system (Figure 11). Each robot features a three-color column; the unique vertical location of the ID color bar (in relation to the start and stop color bars) is proportional to the identification number of the robot. Similarly, the perceived distance between the start and stop color bars of a robot is proportional to the actual distance to that robot.

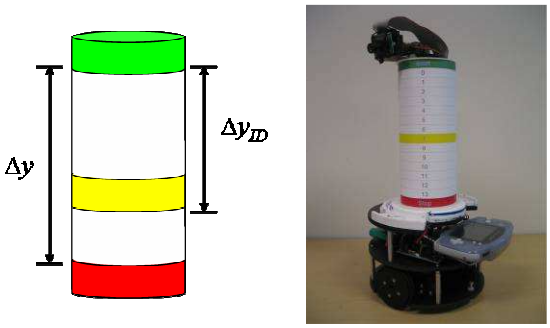

Figure 10: Color bar-coding system (left) implemented on a robot prototype (right).

References

Balch, T. & Arkin R. 1998. "Behavior-based Formation Control for Multi-robot Teams" IEEE Transactions on Robotics and Automation, 14(6), pp. 926-939.

Bekey G., Bekey I., Criswell D., Friedman G., Greenwood D., Miller D., & Will P. 2000. "Final Report of the NSF-NASA Workshop on Autonomous Construction and Manufacturing for Space Electrical Power Systems", 4-7 April, Arlington, Virginia.

Farritor, S.M. & Goddard, S. 2004. "Intelligent Highway Safety Markers", IEEE Intelligent Systems, 19(6), pp. 8-11.

Fredslund, J. & Mataric, M.J. 2002. "Robots in Formation Using Local Information", The 7th International Conference on Intelligent Autonomous Systems, Marina del Rey, California.

Mead, R. & Weinberg, J.B. 2006. "Algorithms for Control and Interaction of Large Formations of Robots", in the Proceedings of The 21st National Conference on Artificial Intelligence (AAAI-06), Boston, Massachusetts, pp. 1891-1892.

Mead, R., Weinberg, J.B., & Croxell, J.R. 2007. "An Implementation of Robot Formations using Local Interactions", in the Proceedings of The 22nd National Conference on Artificial Intelligence (AAAI-07), Vancouver, BC, July 2007, pp. 1989-90.

Tejada, S., Cristina, A., Goodwyne, P., Normand, E., O'Hara, R., & Tarapore, S. 2003. "Virtual Synergy: A Human-Robot Interface for Urban Search and Rescue", in the Proceedings of the AAAI 2003 Robot Competition, Acapulco, Mexico.