Simulating Behavior Based Formation Control in Multi-robot Teams

Robert Rolin
McGillID: 260458133
robert.rolin@mail.mcgill.ca

1. INTRODUCTION

This project deals with formation control in multi-robot systems. This is based on the paper *Behavior-based Formation Control in Multi-robot Teams*, Balch and Arkin, 1999[2], which deals with the same topic. The task of formation control involves the assignment of robots to specific positions that define a desired formation. This desired position is usually the basis for some other task. The problem addressed in the paper is pertinent to military applications; the proposed target of the work is a robotic platoon or scout unit. Formation control can be of paramount importance in this area to maximize sensor coverage or minimize risk of adversarial attack. Other applications include agricultural work or search and rescue missions.

This work aims to achieve formation control, navigation to a goal, and obstacle avoidance simultaneously. The approach has the virtues of being simple, intuitive and effective. It can be shown to perform well under varying conditions however the simplicity of the method prevents it from being exhaustively successful. A simulation was developed to explore the benefits and pitfalls of an implementation presented in the paper as well as to incorporate ideas discussed but not implemented in the original work.

2. BACKGROUND AND RELATED WORK

Egocentric behavior based models were first studied in computer graphics. Reynolds [9] was able to simulate flocks of birds and schools of fish by giving each individual simple rules to follow. These rules depended on local information only and required no communication between individuals. Group models were later expanded to robots by [11] where a small number of robots were shown to be able to maintain specific positions relative to a leader. Chen and Luh [3] were able to achieve geometric formations in a small group of robots without a centralized control. Parker [8] developed a simulation where robots moved in a line formation toward a goal. Balch and Arkin [2] presented novel techniques to allow for Line, Column, Diamond, and Wedge formations for groups of two and four robots. In addition to the commonly used *Leader Reference* they also implemented a *Unit-centre Reference* and discussed *Neighbour Reference.* Formation control, in their paper, is integrated simultaneously with obstacle avoidance and navigation to a goal. [4], [5], [6], and [7] describe more recent methods to provide multi-robot formation control to much larger teams of robots. Trends show that multi-robots teams are increasingly relying on local information only to decrease the need for communication amongst larger groups. Recently the problem of learning in multi-robot systems has also been addressed, like the work done by Tangamchit [10] to allow robots to adapt to dynamic environments and changing tasks. Large swarms of small robots are also the topic of much research being done Michael Rubenstein, Radhika Nagpal, and James McLurkin, among others.

3. METHODS

3.1 Overview

The method is a behaviour-based method. Four desired behaviours are defined:

1. Maintain the selected formation
2. Move toward the goal
3. Avoid collisions with static obstacles
4. Avoid collisions with other robots

Each behaviour can be represented by a vector indicated how the robot should move given a certain local environment. Four motor schemas are then defined based on these behaviours. A motor schema will calculate a vector representing the effect of that behavior at any given time. The basis for using motor schemas to create the behaviour of the robot was developed in [1]. An additional motor schema is added to provide noise for the robot and help the robot from getting stuck in local minima. To get the movement of a robot at any given time a weighted sum of the vector defined by each motor schema is taken. Each schema is accompanied by a parameter called its *gain* that acts as the weight of the vectors. Adjusting these gains can affect the behavior of the robots in interesting ways. Increasing the gain of a motor schema will cause it to adhere to that behavior more strictly while lowering it will loosen its adherence. If the schema to maintain the formation has low gain the robots may split into groups to get around an obstacle but if the gain is high they will stay together and possibly not make it to the goal. Depending on our situation, these gains may be tuned according to what specifications need to be met. In this paper the parameters used were taken exactly from the first implementation of Balch and Arkin's paper (Table 1).

Table 1. Motor Schema parameters

Parameter	Value	Units
avoid-static-obstacle		
gain	1.5	
sphere of influence	50	meters
minimum range	5	meters
avoid-robot		
gain	2.0	
sphere of influence	20	meters
minimum range	5	meters
move-to-goal		
gain	0.8	
noise		
gain	0.1	
persistence	6	time steps
maintain-formation		
gain	1.0	
desired spacing	50	meters
controlled zone radius	25	meters
dead zone radius	0	meters

Groups of four robots are investigated here. Four formations are analyzed: Line, Column, Wedge, and Diamond (Figure 1). In each formation each position is assigned a number so that a robot's id number may be matched with a position. The motor schema implementations used are very similar to those presented in [2] with a couple adjustments introduced.

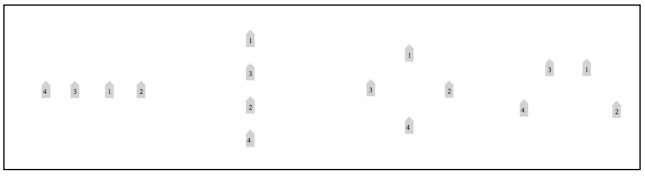

Figure 1. From left to right: *Line, Column, Diamond,* and *Wedge* formations. Each robot id number corresponds to a position in each formation.

3.2 Motor Schema Implementations

The motor schemas are implemented to provide attractive forces toward the goal and to the proper position in formation and repulsive forces from obstacles and other robots.

3.2.1 Maintain_Formation

The primary problem is formation maintenance. This task can be broken down into two parts. The first is for the robot to determine what position is the correct position for the robot to be in in order to be properly in formation, this is called *formation detection*. The second is to determine how to act in order to get to that position, this is called *formation maintenance*.

3.2.1.1 Formation Detection

The problem of finding the correct position for a robot is not an obvious one. Three different methods are given. They may be used interchangeably and often given very different results.

Leader Reference – Each robot knows of the position of a specified leader robot and the position it should be relative to the leader.

Unit-centre Reference – Each robot is aware of the positions of all the other robots. It computes the centre by averaging all the positions and uses a position relative to the centre

Neighbour Reference – Each robot knows of the position of a neighbouring robot. The robot uses a position relative to this.

Each of the formation detection schemes uses a different algorithm for each formation to compute the desired position of a robot given its id and a desired spacing parameter. *Neighbour Reference* was not implemented in Balch and Arkin's paper so an implementation and comparison to the two other methods is shown here.

3.2.1.2 Formation Maintenance

In order to move toward the desired position, the vector output must in the direction from the robot's current point toward its desired point. To determine the magnitude of the vector, three zone of influence are defined around the desired position:

Dead Zone - the smallest zone. Here the magnitude of the vector is set to zero.

Controlled Zone – surrounds the dead zone. Here the magnitude of the vector varies linearly with the robot's distance to its desired position.

Ballistic Zone – everything outside the controlled zone. Here the magnitude of the vector is set to a maximum.

The radii of these zones are adjustable.

3.2.2 Move_to_Goal

The vector to move a robot to the goal is computed from the straight line from the robot toward the goal point. The magnitude of the vector is set to be the gain of the motor schema. The effect of this vector will stay constant throughout a run.

3.2.3 Avoid_Robot and Avoid_Static_Obstacle

The motor schemas *Avoid_Robot* and *Avoid_Static_Obstacle* act in the same way but use different parameters. For these schemas a parameter called *sphere of influence* is used. It is a measure of distance, and if a robot is more than this distance away from an obstacle, the obstacle will not affect it. If the robot comes within this distance from an obstacle, a vector will be produced to have the affect of moving the robot away from the obstacle. The magnitude of the vector increases linearly as the robot gets closer to the obstacle, achieving a maximum when the distance between the robot and the obstacle is zero. The direction of the vector points from a point on an obstacle closest to the robot towards the robot. This is an adaptation of the method presented in Balch and Arkin's paper, which assumes only perfectly circular obstacles and uses the distance and direction from the centre of the obstacle. The method presented here allows for a more general class of obstacle.

3.2.4 Noise

The noise vector points in a direction chosen uniformly randomly and has a magnitude equal to the gain of the motor schema. The motor schema includes a parameter called *persistence,* which controls how long a given noise vector will affect the robot. Noise is an essential part of this system. There exist many cases where the group of robots may become stuck around certain obstacles between the start and the goal. Often the ideal path to the goal involves deviating from what may appear to be a shorter path. Noise provides the robots with a chance of finding a path to reach the goal when situations like local maxima, minima and cyclic behavior may prevent it. The gain of this schema is adjustable and in some cases the robots cannot succeed unless its gain has been raised. However, the gain may not be raised higher permanently because any level of noise that would guarantee success for the team would prevent them from being in a strict formation for most of the time.

3.3 Simulation

An application to simulate a team of robots using this implementation to control their formation was developed. In this simulation, the robots are represented as four blue squares, the goal as a red square and obstacles as black rectangles. The obstacles are generated as squares with a Gaussian amount of noise on the height and width and are placed randomly around the screen. The user may choose the formation and formation detection scheme as well as change any of the parameters, with the defaults provided. The user may place the goal but the robots always start at the bottom in the middle. Parameters can be changed in the middle of a run with the *Apply* button, or can be set

for the next run with the *Restart* button; *Reset* sets the system to default parameters and randomly generates new obstacles.

The robots move by repeating the execution cycle:

1. Get weighted sum of motor schema outputs
2. Add it to current position

The graphical user interface watches a model of the system running with the input parameters and updates the screen with the current positions of the robots, obstacles and goal (Figure 2).

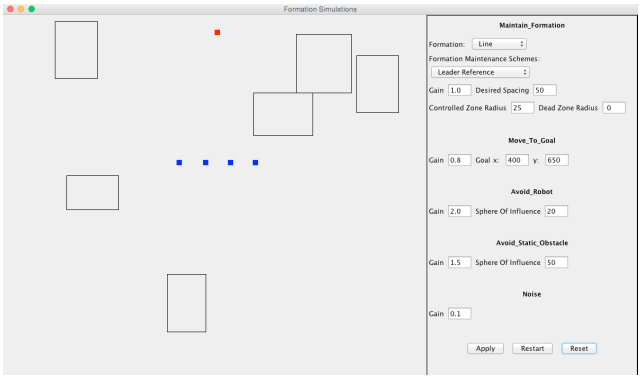

Figure 2. Running simulation.

The behavior of the robots appears to meet the objectives presented in [2]. The robots move toward a goal, stay in a preselected formation and avoid obstacles simultaneously. However, there are failure modes caused by the simplifications of the simulation. In real-life objects move continuously from one place to another but in the simulation they jump a small discrete amount. This causes behaviour around obstacles, specifically between multiple obstacles, to sometimes be erratic. The *avoid_static_obstacle* schema may push a robot away from one obstacle causing it to get closer to a different one and be pushed back. The robot oscillates and may even jump into an obstacle. The most common failure mode is when a large obstacle blocks a direct path from the robots to the goal and the team must split up to get around it. Often the robots succeed but if the distance they must split to get around the obstacle is too large they will not.

4. EXPERIMENTAL RESULTS

Experiments were run on the simulation. Because the *Neighbour Reference* formation detection scheme was a novel implementation, it should be compared to the other implementations in certain respects. The first metric looked at was *Time to goal*, the time between when the robots started to when a robot touched the goal. Ten trials were run on each formation using each formation detection scheme (Figure 3).

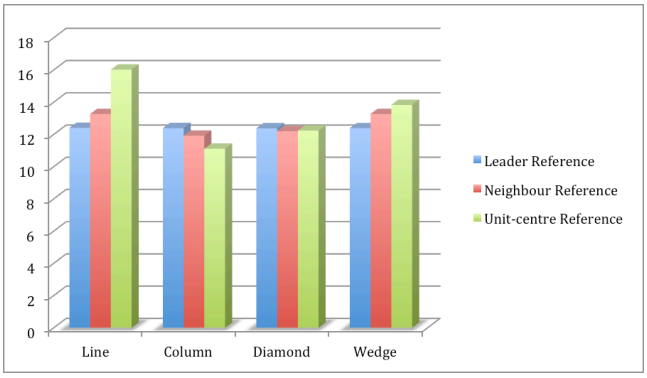

Figure 3. A comparison of the average amount of time taken to reach the goal on randomly generated obstacles.

The results show that *Neighbour Reference* performs as a middle ground between *Leader Reference* and *Unit-centre Reference*. In Line and wedge formations, the *Unit-centre Reference* cause the robots to move slower on average; this is generally because these formations are the wider ones and thus move more slowly when encountering obstacles where the robots have to split up to get around them. The *Unit-centre Reference* is less flexible to the robots splitting and aims to keep them together longer. In the column formation, *Unit-centre Reference* takes less time because the reference point for the robots is further behind than where it is for *Leader Reference* so they are better able to see paths around obstacles. The leader in the column formation will get very close to an obstacle before realizing it must move sideways, which it will do slowly. Conversely, the unit-centre will be further behind the obstacle when the first robot hits it, and the centre can make a faster diagonal path to get around it. *Neighbour Reference* acts to mitigate the difficulties in both these circumstances. When a wide formation meets an obstacle, the robot with the clearest path around it will begin to move and its neighbor will follow, which brings the rest of the robots, they are not tied to the middle of the formation. Similarly in column formation the robot at the back will have the clearest path around an obstacle and will begin the move around it bringing all the others. The reason it performs worse than the optimal scenario is due to the amount of *fighting* or competing movements that happens before a path of least resistance is settled on.

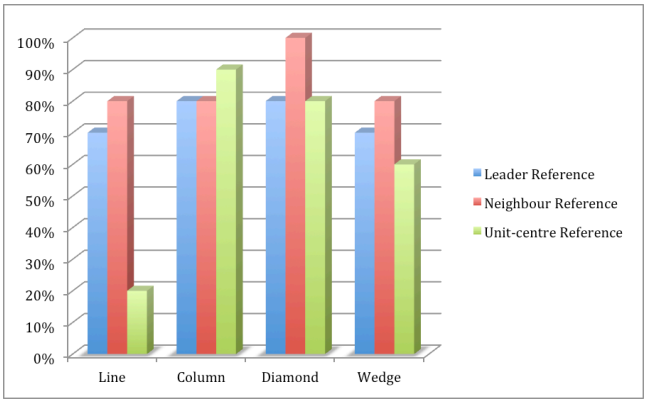

Figure 4. A comparison of the percentage of runs completed on randomly generated obstacles.

The second experiment measured the percentage of runs finished. Ten random arrangements of obstacles were chosen and on each of the ten, each formation and formation detection scheme was run to see if it was able to finish it. This percentage of runs completed can be thought of as a measure of the versatility of the formation and formation detection scheme combination; an indication of how well the combination will perform on unknown terrain (Figure 4).

From this experiment it can be seen that *Neighbour Reference* is almost always more reliable than the other two reference schemes. When using *Unit-centre Reference* the robots usually fail to meet the goal when there exists a large obstacle between the centre of the robots and the goal. When using *Leader Reference*, they fail when a large obstacle blocks the leader from the goal. Using

Neighbour Reference gives the robots more opportunities to find some path for the robots to get to the goal. If only one robot on the side can see a path, it will begin to follow it and then the other robots will be pulled after. With the other formation detection schemes a robot on the side that may see a path will be tied to the centre or leader, which may prevent it from following the path.

Observe that the diamond formation using *Neighbour Reference* achieved a 100% success rate, and has an average completion time very close to the optimal observed average completion time.

5. CONCLUSION

Our simulation shows that this approach works reasonably well on a simulation with four robots. An advantage of this method is that it is very simple to understand and thus easy to reason about the observed behaviour of the robots at a given point. It also can be easily applied with different formations and different formation detection schemes, allowing for lots of variability. Formation detection schemes that have not been proposed yet, could easily be integrated into the system in the future.

A disadvantage of this method is that it does not generalize easily to larger real systems like the proposed applications. Balch & Arkin [2] implemented a system similar to this, using a *Unit-centre Reference* on two large Unmanned Ground Vehicles. They did not use a *Leader Reference* because they feared a follower robot could enter a situation where it was not able to keep up and would get lost. They experienced up to seven second latencies during robot-to-robot communication. This communication latency is a big problem and poses a large hindrance on scalability. One possible solution is to incorporate a passive sensing system (vision) to avoid any communication between robots. However, using *Unit-centre Reference* a passive sensing system may also be quite expensive. The robots would be required to maintain a tracking of every other robot. The robots may span a 180 degree field of view, and could potentially be blocked by obstacles. *Unit-centre Reference* also negates the possibility of using a human leader. Although this was not a requirement of the system, it is a logical desire that may arise when using this system. A human would be unable to calculate the centre of all the robots as quickly as necessary.

The solution to these problems could be addressed by using *Neighbour Reference* in these systems. When scaled to larger systems, each robot would still only be required to know the location of one other robot. By active or passive sensing methods, this is much more feasible. Also a human leader can easily be introduced by having him maintain a position relative to one other robot. It has been shown that *Neighbour Reference* will increase the likelihood of traversing a randomly presented obstacle field and given an unknown formation will perform better on average than the other two reference schemes would. Thus it seems much more natural to want a *Neighbour Reference* when implementing this approach on larger systems. A scheme very similar to *Neighbour Reference* is used by Fredslund and Mataric [6] but it was also only tested as a simulation on four robots.

Another disadvantage of this approach is that the formation detection schemes require specific algorithms for each formation and the robots need to be told beforehand which position in the formation they are meant to occupy. Presumably a more general approach could be taken where a robot, knowing only the desired formation and local sensor information, would interpret what position it must occupy in the formation. This would allow for systems with larger numbers of robots. It would obviate the need for defining for every position in a formation a relative angle and distance to a reference point. This, in turn, could allow for a generalization over formations, perhaps only needing a picture as description.

6. REFERENCES

[1] Arkin, R. C. (1989). Motor schema—based mobile robot navigation. *The International journal of robotics research*, *8*(4), 92-112.

[2] Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, *14*(6), 926-939.

[3] Chen, Q., & Luh, J. Y. S. (1994, May). Coordination and control of a group of small mobile robots. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on* (pp. 2315-2320). IEEE.

[4] Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., & Taylor, C. J. (2002). A vision-based formation control framework. *Robotics and Automation, IEEE Transactions on*, *18*(5), 813-825.

[5] Dudenhoeffer, Donald D., and Michael P. Jones. "A formation behavior for large-scale micro-robot force deployment." *Simulation Conference, 2000. Proceedings. Winter*. Vol. 1. IEEE, 2000.

[6] Fredslund, J., & Mataric, M. J. (2002). A general algorithm for robot formations using local sensing and minimal communication. *Robotics and Automation, IEEE Transactions on*, *18*(5), 837-846.

[7] Kostelnik, P., Samulka, M., & Janosik, M. (2002, November). Scalable multi-robot formations using local sensing and communication. In *Robot Motion and Control, 2002. RoMoCo'02. Proceedings of the Third International Workshop on*(pp. 319-324). IEEE.

[8] Parker, L. E. (1993, May). Designing control laws for cooperative agent teams. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on* (pp. 582-587). IEEE.

[9] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25-34.

[10] Tangamchit, P., Dolan, J. M., & Khosla, P. K. (2002). The necessity of average rewards in cooperative multirobot learning. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on* (Vol. 2, pp. 1296-1301). IEEE.

[11] Wang, P. K. C. (1991). Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, *8*(2), 177-195.