Self-Organization of a Robot Swarm into Concentric Shapes

Geoff Nagy and Richard Vaughan
School of Computing Science, Simon Fraser University
Burnaby, Canada
{*gnagy, vaughan*}*@sfu.ca*

***Abstract*—In this paper, we show how a swarm of differential-drive robots can self-organize into multiple nested layers of a given shape. A key component of our work is the reliance on inter-robot collisions to provide information on how the formation should grow. We describe a simple controller and experimentally evaluate how its performance scales as the number of robots in the swarm increases from tens to several hundred robots. The average quality of the formation is shown to be a linearly decreasing function of swarm size, although the steepness of this line depends on the complexity of the formation. We also show that the time for a swarm to form a given shape does not grow quickly even as the number of robots in the swarm increases by a large amount.**

I. INTRODUCTION

We present a simple controller for organizing large numbers of robots into concentric shapes. These shapes can be simple (such as a circle, shown in Figure 1) or more complex, such as a cross (Figure 4). At its heart, our approach is very simple: a large swarm of robots self-organizes along the perimeter of the defined shape. Since all of them might not fit, the remaining robots attempt to organize themselves into a larger version of the original shape surrounding the first formation. When this group of robots experiences too much congestion, they form another outer layer, and so on. The ultimate effect is that the robots will form concentric instances of a predefined shape around a central point. Our controller is simple—robots are anonymous, homogeneous, and do not need to communicate or identify one another. Robots only need simple rangefinders to detect obstacles in front of them and also require the ability to determine their distance and bearing in world frame from the center of the desired formation. Our controller is very scalable: the amount of computation on each robot is small and independent of population size, and formation quality is maintained as the number of robots increases. We demonstrate this empirically using swarm populations ranging from tens to hundreds of robots with several different shape formations[1].

Our work has several applications. Concentric circles are perhaps one of the most obviously useful formations since they can be used to surround, protect, or observe a target [1], [2], [3], [4]. Circular formations also help to maximize mutual visibility. Our controller will also work with stall vehicles (e.g., fixed-wing aircraft [1]) since the robots are always moving. Our algorithm has the property that robots space themselves out evenly, and this can be leveraged to allow actions such as cooperatively lifting an object from underneath such that the load is evenly balanced among the robots. Additionally, nested rectangular or elliptical formations would allow robots to perform foraging or transportation in such a way that interference is minimized.

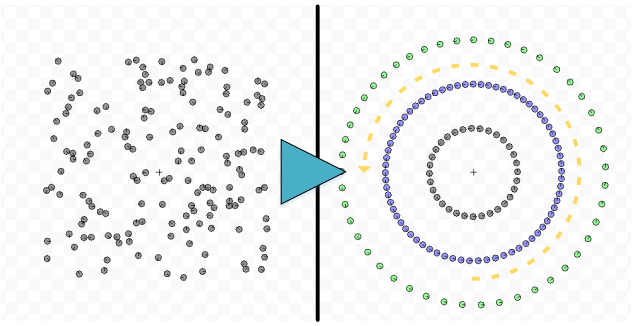


Figure 1: A group of randomly positioned robots self-organize into concentric rotating circles with our controller.

The next section provides a brief overview of related work. We follow this with a detailed description of our robot controller. We present the results of our experimental evaluation for determining the scalability and efficiency of our controller. The paper ends with directions for future work and concluding remarks.

II. PREVIOUS WORK

Groß et al. [5] presented a robot formation controller for surrounding a target based on the well-known Brazilian nut effect, and also evaluated it with a physical swarm of e-pucks [6]. Robots in their work attempted to maintain an appropriate distance to other robots based on their perceived "virtual sizes", while simultaneously attempting to drive towards a common center point. The addition of random motion emulated the effect of shaking a box of nuts—eventually, the "smaller" robots would be gathered near the center of the formation, while the "larger" ones would occupy the outer edges. In contrast, robots in our work are homogeneous and do not need to use pre-assigned properties such as "size" in order to influence their position within the

[1]All source code and data for this project are available at https://github.com/AutonomyLab/concentric-shapes. The SHA1 hash tag is 7933ab3a8e4b79f6236070de40b0068e90b450c8.

formation. In contrast to the fixed membership of a robot to its size-class in Groß's system, a robot's layer number is set dynamically during interactions between robots. Our controller is also able to form a greater variety of shapes.

Litus and Vaughan [7] demonstrated an approach for queuing a large number of robots in a compact Fibonacci spiral, as in the arrangement of sunflower florets. Here we are interested in task-determined shapes rather than compactness.

Several other works explore the construction of circular formations. Défago and Souissi [8] presented an algorithm for anonymous, non-communicating robots that results in a stable circular formation. However, they assume that all robots are able to see and localize one another—in our work, robots are only able to perceive other swarm members indirectly as obstacles. Yu et al. [1] presented an approach that requires inter-robot communication, unlike our work. Their technique is also not generalizable to other shapes. Similar to our work, Zheng et al. [2] proposed a controller that relies on bearing information for surrounding a target, but their approach can only be used to form single (non-concentric) circles. Chen et al. [9] demonstrated a controller enabling multiple robots moving at different speeds to converge to a single formation of concentric circles. In their work, a robot's orbit distance was influenced by its speed, ensuring that robots at different speeds were kept in separate layers of the formation to avoid interference. Similarly, we are also interested in minimizing interference by constructing layered formations. Lee et al. [10] used local interaction rules based on the formation of isosceles triangles to develop a controller that resulted in concentric circles of robots. Similar to our work, they use a minimal robot model (no communication or robot identifiers), but their controller is not generalizable to other shapes.

III. ROBOT CONTROLLER

Our robot controller is composed of three subsystems. The *orbit and steering* module is responsible for directing the robot towards an appropriate location on the perimeter of the desired shape. To help reduce unnecessary interference (as we will explain, the existence of robot-robot interference is actually a driving component of our algorithm) the *collision avoidance* subsystem slows the robot down when it detects a nearby obstacle, but does not steer away to avoid it. Finally, the *layer promotion* subsystem determines when a robot should "promote" itself to the next layer of the formation in order to avoid congestion and free up space for other robots which already occupy the current layer.

A. Orbit and Steering

Our robot controller achieves pseudo-orbital motion around a central point at a specified radius. A constant radius would describe a perfectly circular orbit. Changing this distance dynamically, however, allows a robot to travel

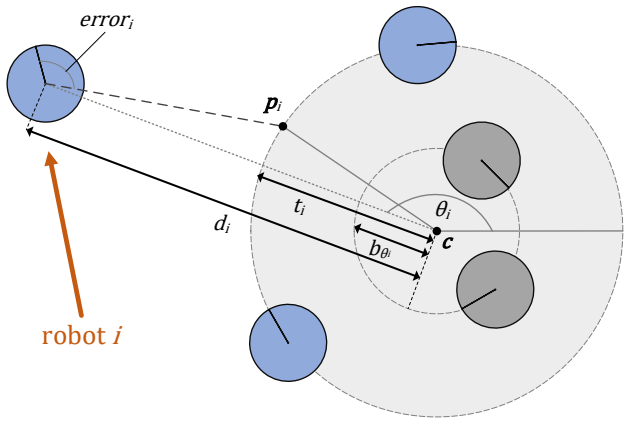


Figure 2: Shows the target orbit position for robot i (labelled). Although a circular formation is shown, the approach is generalizable to more complex shapes.

a path describing the boundary of more complex shapes. Consider the Earth's elliptical orbit, where our distance to the Sun varies constantly. To generate a desired path, we define a robot i's target distance b_{θ_i} from a central point c (the formation center) as a function of the robot's global angle θ_i relative to c. We have implemented this as a lookup table with 1-degree resolution. Using an appropriate sensor, a robot continually monitors its range d_i and relative bearing to the formation center, and uses compass data to determine its global orientation θ_i with respect to this point. θ_i is then used to index into the lookup table, where the base target orbit distance b_{θ_i} at that angle is located.

Based on the available range and bearing information, robot i calculates the estimated formation center point c and computes p_i, the target orbit location. p_i is acquired by adding a small rotational offset u to θ_i and computing a position t_i units away from c at the new angle (1). Using the same value of u for all robots ensures that the swarm will always travel the formation shape in one direction only. The robot then computes the bearing $error_i$ to the point p_i and feeds it into a PID controller that steers the robot to the correct position. The shapes described by each robot in the population are globally aligned due to the global frame of reference. This is depicted in Figure 2.

To allow robots to occupy a different "layer" of a formation, robot i's base orbital distance b_{θ_i} is multiplied by an integer $g_i \geq 0$ that describes which layer the robot should be currently occupying (1). If g_i is zero, the robot will occupy the innermost layer of the formation. As described in Section III-C, the value of g_i will increase as the robot attempts to find a layer that has the least amount of congestion.

$$t_i = b_{\theta_i} + (b_{\theta_i} \times g_i) \quad (1)$$

B. Collision Avoidance

In order to detect congestion and reduce unnecessary collisions, robots are equipped with forward-facing rangefinder sensors. Robots reduce speed based on the distance to the nearest sensed neighbour robot. This results in a time- and energy-efficient (and visually-pleasing) "zipper merge" effect when robots attempt to join a new layer that is already populated with many robots. It also encourages robots to maintain a suitable distance from each other.

We apply (2) to the left and right motor speeds M_{L_i} and M_{R_i} of robot *i* when it detects a frontwards obstruction *j* within some threshold distance *F*, where $D_{i,j}$ is the distance from the detecting robot *i* to object *j*. The slowdown factor *S* is particularly sensitive: too high a value results in robots not slowing down enough and thus not merging efficiently. Values that are too low will slow down the merge process unacceptably since robots will brake too hard. We used hand-tuned values of $S = 1.2$ and $F = 30$cm in our implementation. (A sensitivity analysis of these parameters is beyond the scope of this paper—informally, it is not difficult to find appropriate values for them.)

$$M_{\{L_i,R_i\}} = M_{\{L_i,R_i\}} \times \left(\frac{D_{i,j}}{S}\right) \quad (2)$$

Note that the collision avoidance subsystem does not actually steer the robot away from potential collisions. For this reason, an implementation on real robots would not require advanced rangefinder sensors that provide bearing information—an array of inexpensive short-range infrared or sonar sensors would suffice.

C. Layer Promotion

A key element of our approach is the reliance on the interference between robots to provide useful information to each robot [11], [12], [13], rather than treating it as something purely undesirable.

Recall from Section III-A that every robot *i* in the swarm is assigned an initial layer ID g_i that determines which layer of the formation the robot should occupy. To ensure that the swarm is physically spread out over successive layers of the formation, robots may decide to *promote* themselves to the next layer when they detect too much congestion, by incrementing the value of g_i.

Using its front-facing rangefinder, robot *i* will consider itself *overcrowded* if both of these conditions are met: (*a*) *i* is within some overcrowding distance *V* of another robot, and (*b*) *i*'s distance from the orbit center, d_i, is greater than or equal to its preferred orbital distance t_i. If overcrowded, the robot will increment the value of its *overcrowded timer*, which controls how long the robot is willing to wait for the interference to reduce. If the timer reaches its maximum value (see (3)), the timer is reset and the robot increments g_i. Conditions (*a*) and (*b*) collectively have the effect that a robot will only promote itself if pushed outwards from its current orbital radius by another robot, or if there is too much congestion to settle into a comfortable "lower" orbit. The intuition here is that a robot should not needlessly promote itself to a higher layer if it is still attempting to travel outwards to its desired orbit distance. The *overcrowded timer* is repeatedly reset to zero whenever either condition (*a*) or (*b*) above is not met.

The maximum value of the overcrowding timer is defined as a function of g_i, as shown in (3). The intent is to encourage robots occupying inner layers to promote themselves more quickly. This is desirable since inner layers cannot accommodate as many robots as outer ones—if a robot experiences heavy congestion in an inner layer, then it is statistically likely that the robot does not belong there. Conversely, robots in the outer layers will be hesitant to expand the formation further, and essentially wait out any congestion in the hopes that interference will gradually reduce. This has the advantage that a maximum value for g_i does not need to be specified. While this technically means that robots can promote themselves excessively or even endlessly (we refer to this phenomenon as *runaway promotion*), this does not occur because *g* will typically converge to suitable values across the population as promotion results in reduced congestion. That is, a sufficient number of robots promote themselves in a timely enough manner that the formation does not explode. Conditions (*a*) and (*b*) above also help to ensure that runaway promotion does not occur.

Figure 4 shows a time lapse of robots promoting themselves when attempting to form a *cross* shape.

$$max\_overcrowd\_time = 1000ms + (g_i \times 1500ms) \quad (3)$$

IV. EVALUATION

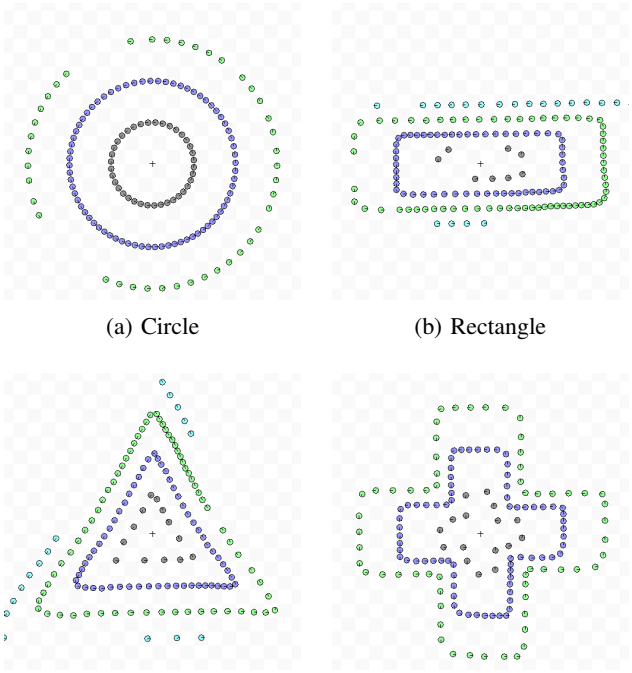
A. Simulator, Robots, and Formations

We experimentally evaluated our controller using a custom robot simulator written in C++. Robots were modelled as differential drive vehicles 10cm in diameter. Simple physics prevented intersections and allowed robots to push against and move each other. The simulator arena was 12m x 12m in size, and was bounded and finite (i.e., not toroidal). The large arena size was chosen because preliminary experiments showed that this was more than sufficient to comfortably hold formations made up of 500 robots or less. In all configurations, robots were placed randomly within the inner 60% of the environment, and were all assigned an initial layer ID g_i of 0. Significant parameters used in our trials are shown in Table I. These were chosen to be representative of a typical lab robot (e.g., a Pioneer).

Our simulator allows us to draw formations with the mouse, and save or load them to and from disk. For our experiments, we created four different formations (Figure 3): *circle*, *rectangle*, *triangle*, and *cross*, to evaluate the performance of our robot controller. For each of these formations,

Parameter	Value
initial layer ID, g	0
robot radius, E	5cm
robot top speed	50cm/s
robot rangefinder range	30cm
robot rangefinder FOV	180°
slowdown factor, S	1.2
overcrowd threshold, V	1cm
arena size	12m x 12m
trial time	120s

Table I: Parameters used in our simulations.



(a) Circle (b) Rectangle

(c) Triangle (d) Cross

Figure 3: The four formations used in our evaluation. The circle formation specifies a constant orbit distance, while the other three are more complex and were drawn by hand.

we ran experiments with swarms sizes of 50 to 500 robots, at 50 robot increments. Every experimental configuration was repeated 10 times. This yielded a total of 400 experimental trials. Every trial was run for 120 simulated seconds.

B. Performance Criteria

We examined how fast robots would converge to the desired shape and how accurately that shape could be formed, particularly as the number of robots increased. We ran each trial for two simulated minutes, but we observed that in all trials the robots required less than 30 seconds to reach a relatively stable configuration. We record the *average orbital error* (4), defined as the mean of the distance (in robot radii) from each robot to its desired orbit distance, at any instant:

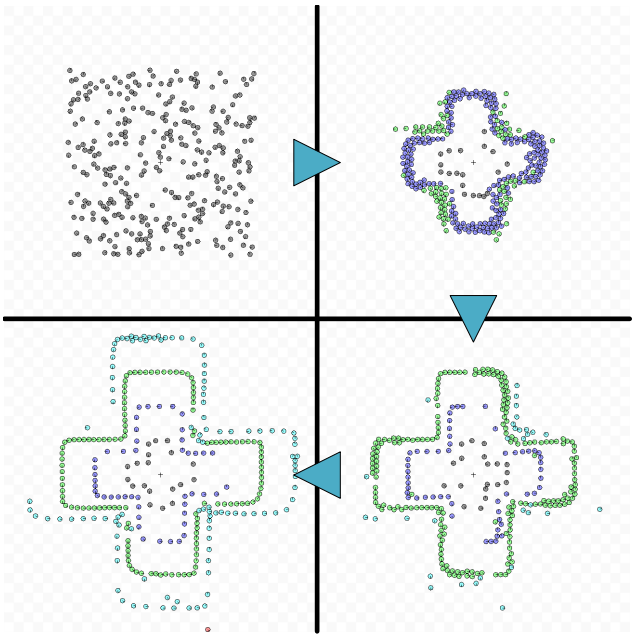


Figure 4: Shows a swarm of robots forming a cross shape. As robots near the inner layers interfere with each other, robots will promote themselves to the outer layers.

$$avg\_orbital\_error = \frac{\sum_{i=1}^{N} \frac{|d_i - t_i|}{E}}{N} \quad (4)$$

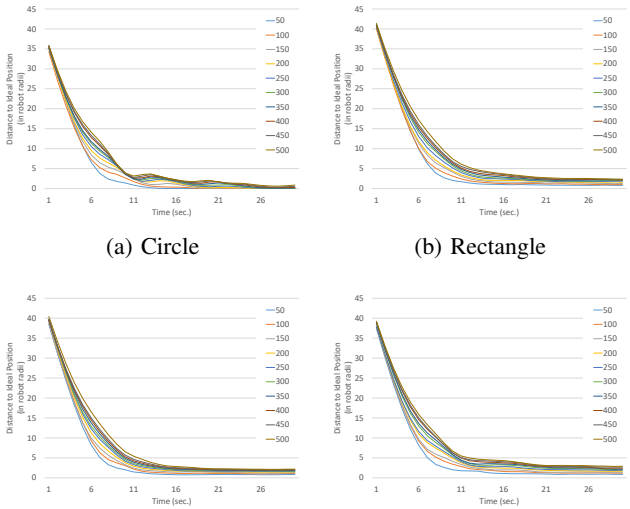
where E is the radius of a robot and N is the number of robots in the system. A non-technical interpretation of this value would be the average failure of the robots to represent the desired formation at the current time, i.e., to what degree the swarm diverges from the ideal formation appearance. We sampled this value every second so we could observe it as it changed over time, and averaged the results of each configuration over the 10 trials.

V. RESULTS AND DISCUSSION

In all trials, robots achieved a stable configuration within 30 seconds. Figure 5 shows the average orbital error for each formation over time, with varying numbers of robots.

Perhaps the most interesting result is that the average orbital error grows very slowly as the number of robots increases. This is an encouraging result, since it suggests that our controller is scalable to many robots and is not severely impacted by interference as more robots are added. While interference is obviously undesirable in multi-robot systems, in our case we harness it to provide useful information that improves the state of the robot swarm as a whole.

The total number of robots in the system also affects the final value of the average orbital error. As the number of robots increases, the final average error increases slightly as well. This is an expected result: it is clear that a larger



(a) Circle (b) Rectangle

(c) Triangle (d) Cross

Figure 5: Average orbital error over time for all four formations, for the first 30 seconds of simulation, for all 10 swarm population sizes.

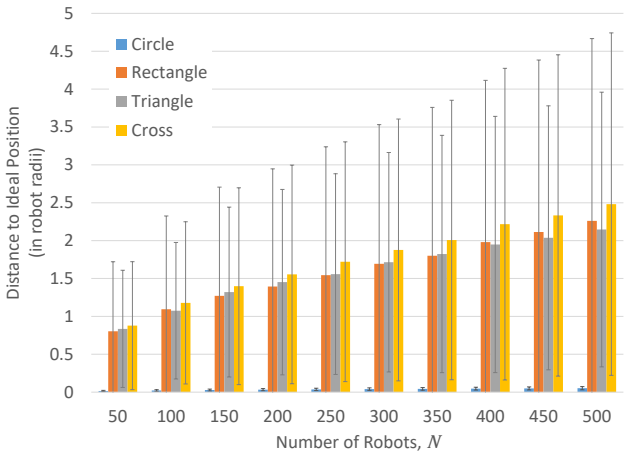


Figure 6: Shows the final average orbital error at the conclusion of each 2 minute simulation, for every formation and value of N. Error bars show standard deviation.

number of robots only makes it more likely that some robot(s) may not be situated in the formation 100% correctly.

It is also possible that higher layers of a formation are more difficult to occupy with precision: robots in outermost layers tend to move quicker because they experience less congestion (Section III-B), and moving at faster speeds makes it difficult to turn hard corners. Notably, the *cross* formation is the most complex (containing 12 hard turns) and suffers the most from this. This effect is also sometimes present in the innermost layer since formation shapes are harder to navigate accurately when they are very small. Interestingly, only the *circle* formation reaches near-zero average orbital error. This is likely because the other three shapes are much less uniform and contain hard angles of at least 90 degrees. This suggests that robots may be able to achieve lower orbital error if the *rectangle*, *triangle*, and *cross* formations are modified to have rounded corners.

The bumps seen in Figures 5a show large numbers of robots simultaneously deciding to promote themselves in order to avoid interference. The same behaviour is present in Figure 5d, but to a lesser degree. The *rectangle* and *triangle* formations do not demonstrate this behaviour as strongly. This is because these formations are irregularly shaped in comparison—while large-scale promotion events still occur, they are not as synchronized as a result.

Figure 6 shows the final values of the average orbital error after 2 minutes. The error for the circle formation is noticeably smaller than those of the other formations, likely because the circle shape is the least complex and is best-suited for an orbit-based controller. Error for the circle formation only shows a slight increase as the number of robots grows. The other shapes are more complex and suffer from higher orbital error. However, as seen in the example screenshots (Figures 3 and 4), these errors do not translate into a strongly noticeable deformation—the shapes being formed are still easily recognizable.

Orbital error appears to grow linearly as the number of robots increases. This is true for all formations in our study, and is an intriguing property of our controller. The fact that our robots rely on interference to provide information allows them to adapt and avoid collisions by promoting themselves to higher layers of the formation. The results shown here suggest that if we were to add even more robots to the system, we would see a linearly proportional increase in orbital error. The finite size of the environment itself would eventually become a limiting constraint.

A point of curiosity is whether or not it is possible using our system to generate formations that feature worse-than-linear performance. We suspect that any formation our system is capable of generating will exhibit linear properties similar to those shown in Figure 6 because of how the formation points are defined (as a function of a robot's orientation relative to a point). It may be the case that linearly-scaling formation error is an inherent property of our algorithm due to the limitations imposed on the types of formations that can be represented this way. However, we have not performed any formal validation in this regard.

A. Limitations

Our choice of initial $g = 0$ for all of the robots had an obvious effect on our results. Since every robot initially attempted to occupy the innermost layer, there was a significant amount of interference until robots began promoting themselves. There would have been much less interference

(and the swarm may have converged more quickly) if g had been chosen differently. For example, initially assigning every robot a random g between 0 and 4 may have improved our results. We chose to use $g = 0$, however, because this makes no assumptions about the size of the swarm, which may potentially range from a handful to thousands of robots. Additionally, our robot controller does not support *demotion*—robot layer IDs can only increase or remain the same. This places limitations on good initial choices of g. For example, an initial $g = 1$ would prevent any robot from ever occupying the innermost ($g = 0$) layer. Future work could include having robots periodically transmit anonymous radio pings, the frequency of which could be used to estimate the swarm size and determine a good distribution of initial g values across all robots. Even if the amount of communication is so overwhelming that no messages can be received uncorrupted, such events could themselves be used as indicators that the swarm is potentially massive [11].

Our results are also likely influenced by the starting arrangement of the robot swarm. Recall from Section IV-A that all robots are initially placed in the inner 60% of the environment. This is obviously advantageous to our controller since the average robot starting position in this case is roughly the formation center, c. Additional experiments could determine how well the system performs when the robots are distributed along one edge of the arena, or when the robots are already perfectly arranged in their correct formation positions but assigned random, zero, or otherwise inappropriate values of g at the onset of the simulation.

Currently, robots in our work require the ability to determine their range and bearing to a central point. While the availability of GPS or motion-tracking systems makes this possible, it would be better to not have to rely on them. (Chen et al. [6] used a light source to provide localization information to their robots.) Possible extensions to this work should include methods that rely on local sensing only (perhaps by localizing to nearby detected robots). However, computing orbit distances based on orientation relative to a point means that certain shapes are impossible to form. Highly concave formations, for example, cannot be created.

VI. CONCLUSION

We have presented a simple controller for self-organizing a robot swarm into concentric shapes. We have demonstrated the generalizability of our approach using four different formations. Simulation results indicate that the system scales well as more robots are added, and that the complexity of the shape and the size of the swarm have a strong influence on how accurately the shape can be represented. Additionally, we have identified several areas in which our controller could be improved. The results we obtained are encouraging and suggest that further research into concentric formation control would be useful.

REFERENCES

[1] X. Yu, L. Liu, and G. Feng, "Distributed circular formation control of multi-robot systems with directed communication topology," in *Control Conference (CCC), 2016 35th Chinese*. TCCT, 2016, pp. 8014–8019.

[2] R. Zheng, Y. Liu, and D. Sun, "Enclosing a target by nonholonomic mobile robots with bearing-only measurements," *Automatica*, vol. 53, pp. 400–407, 2015.

[3] L. Ma and N. Hovakimyan, "Cooperative target tracking in balanced circular formation: Multiple uavs tracking a ground vehicle," in *2013 American Control Conference*. IEEE, 2013, pp. 5386–5391.

[4] J. Saez-Pons, L. Alboul, J. Penders, and L. Nomdedeu, "Multi-robot team formation control in the guardians project," *Industrial Robot: An International Journal*, vol. 37, no. 4, pp. 372–383, 2010.

[5] R. Groß, S. Magnenat, and F. Mondada, "Segregation in swarms of mobile robots based on the brazil nut effect," in *Proc. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4349–4356.

[6] J. Chen, M. Gauci, M. J. Price, and R. Groß, "Segregation in swarms of e-puck robots based on the brazil nut effect," in *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 163–170.

[7] Y. Litus and R. Vaughan, "What can a sunflower teach a robot? efficient robot queuing by reverse phyllotaxis," in *Proc. 12th International Conference on Artificial Life (ALife XII)*, August 2010.

[8] X. Défago and S. Souissi, "Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity," *Theoretical Computer Science*, vol. 396, no. 1, pp. 97–112, 2008.

[9] Z. Chen and H.-T. Zhang, "A remark on collective circular motion of heterogeneous multi-agents," *Automatica*, vol. 49, no. 5, pp. 1236–1241, 2013.

[10] G. Lee, S. Yoon, N. Y. Chong, and H. I. Christensen, "A mobile sensor network forming concentric circles through local interaction and consensus building," *Journal of Robotics and Mechatronics*, vol. 21, no. 4, pp. 469–477, 2009.

[11] D. Goldberg and M. J. Mataric, "Interference as a tool for designing and evaluating multi-robot controllers," in *Aaai/iaai*, 1997, pp. 637–642.

[12] H. Hamann, "Towards swarm calculus: Urn models of collective decisions and universal properties of swarm performance," *Swarm Intelligence*, vol. 7, no. 2-3, pp. 145–172, 2013.

[13] Z. Song, S. A. Sadat, and R. T. Vaughan, "Mo-lost: Adaptive ant trail untangling in multi-objective multi-colony robot foraging," in *Proc. 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 1199–1200.