Multi-Robot Formation Control via a Real-Time Drawing Interface

Author(s):
Hauri, Sandro; Alonso-Mora, Javier; Breitenmoser, Andreas; Siegwart, Roland Y.; Beardsley, Paul

Publication Date:
2014

Permanent Link:

https://doi.org/10.3929/ethz-a-010034896 →

Originally published in:

Springer Tracts in Advanced Robotics 92, http://doi.org/10.1007/978-3-642-40686-7_12 →

Multi-Robot Formation Control via a Real-Time Drawing Interface

Sandro Hauri, Javier Alonso-Mora, Andreas Breitenmoser, Roland Siegwart and Paul Beardsley

Abstract This paper describes a system that takes real-time user input to direct a robot swarm. The user interface is via drawing, and the user can create a single drawing or an animation to be represented by robots. For example, the drawn input could be a stick figure, with the robots automatically adopting a physical configuration to represent the figure. Or the input could be an animation of a walking stick figure, with the robots moving to represent the dynamic deforming figure. Each robot has a controllable RGB LED so that the swarm can represent color drawings. The computation of robot position, robot motion, and robot color is automatic, including scaling to the available number of robots. The work is in the field of entertainment robotics for play and making robot art, motivated by the fact that a swarm of mobile robots is now affordable as a consumer product.
The technical contribution of the paper is three-fold. Firstly the paper presents shaped flocking, a novel algorithm to control multiple robots - this extends existing flocking methods so that robot behavior is driven by both flocking forces and forces arising from a target shape. Secondly the new work is compared with an alternative approach from the existing literature, and the experimental results include a comparative analysis of both algorithms with metrics to compare performance. Thirdly, the paper describes a working real-time system with results for a physical swarm of 60 differential-drive robots.

Sandro Hauri
Disney Research Zurich e-mail: shauri@student.ethz.ch

Javier Alonso-Mora
ETH Zurich & Disney Research Zurich e-mail: javiera@disneyresearch.com

Andreas Breitenmoser
ETH Zurich e-mail: andreas.breitenmoser@mavt.ethz.ch

Roland Siegwart
ETH Zurich e-mail: rsiegwart@ethz.ch

Paul Beardsley
Disney Research Zurich e-mail: pab@disneyresearch.com

1 Introduction

The goal of this work is to allow a non-expert user to represent drawings and animations with a robot swarm. Previous work has demonstrated the artistic potential of robot swarms but the focus has been on abstract visual effects. In contrast, this work describes how to create representational images with robots. A user can make a drawing and the robots will adopt a physical configuration and colors to represent the drawing as shown in Figure 1. Or the user can create an animation by (a) creating an initial shape and moving it, or (b) creating a sequence of keyframe drawings, and the robots will move to represent the animation.

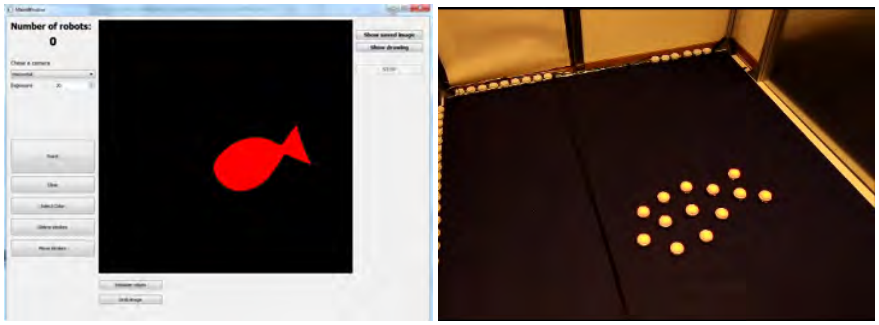

Fig. 1 At left, the user interface for creation of drawings and animations. **At right,** the experimental setup and the robot representation of the fish drawing. Robots around the perimeter are docked in battery chargers, and the robots automatically move between the workspace and the chargers as required.

The motivation for a drawing interface is that it is an intuitive way for a non-expert user to direct a swarm. It is not feasible for a user to explicitly control every robot in a swarm because of the large number of degrees of freedom - 3 DOF for the location of each robot, 2 DOF for velocity (on the ground plane), and 3 DOF for color. Existing flocking algorithms already offer a way for a user to direct a whole swarm, but they do not offer any possibility for representational visual effect. In contrast, our goal with the drawing interface is to allow a user to easily create representational robot art. Our approach has been to build on the power of existing flocking algorithms, and extend them to incorporate shape.

The work belongs in the field of entertainment robotics. The falling price of robots like the ones in Figure 1 makes for an affordable consumer product. For example, the hardware components of a differential-drive robot with IR communication and RGB LED cost as little as $5, although the robot used in these experiments presents a higher cost. Common portable devices such as smartphones would be suitable to provide the user interface. (The system in this paper uses a fixed overhead camera, but a hand-held device with camera is also a feasible approach for controlling the swarm).

The primary contribution of the paper is shaped flocking, a method for formation control. It combines traditional flocking forces with forces arising from a target shape. It draws on the power of flocking to produce collective motion that is visually compelling, and adds the ability to achieve formations with desired shapes.

Flocking is a non-explicit approach to guiding a multi-robot system, with robot motion being emergent from the flocking forces. A completely alternative approach is to do a geometric analysis of a target shape, and explicitly determine the goal positions of individual robots within the shape. Algorithms for such a goal-directed approach, in particular Voronoi coverage, already exist in the research literature. To provide a comparative analysis, we do a head-to-head comparison of shaped flocking and a goal-directed approach [2] in Section 5.

2 Related Work

Outside of robotics, shape constraints have been extensively studied in computer graphics and crowd simulation.

Takahashi et al. [13] achieves smooth transformations between key frames in crowd simulation, using a spectral-based approach, where a transformation is made on the Delauney triangulation from the one key frame to the next. In a next step, a social force field is introduced to avoid collisions. In [5], the complexity of the agent formation is reduced and described only by mean and deviation. It presents a decoupled controller to move and transform a group of robots, but only for simple cases as ellipses and rectangles. In a crowd simulation, Gu [6] samples the outline of the desired shape first and then fills the rest of the shape area using grid points. The agents are assigned to their goal positions by finding correspondences from the current position and the goal deployment. These correspondences are found separately for the outline and the inner part of the shape. The collision avoidance is created by repulsive forces between agents. Rubenstein [10] shows a way to automatically scale the size of the displayed image, so that all robots can fit in the shape. It works without knowing the number of robots in the display, but the robots agree on a factor of the image scale, that is increased, if robots do not fit into the image or reduced, if there is unfilled space around the robot.

A way of controlling coverage in real mobile sensor networks is implemented in [11], where each robot can approximate its weighted Voronoi cell by knowing its local sensor values.

Reynolds famously published a method of simulating the flocking behavior of birds in 1987 [9], using only three behavioral rules to create global bird migration motions. A lot of attention has been given to theoretical and practical research on Reynolds' rules. Olfati-Saber showed in [8], that all three Reynolds rules can be expressed with a simple control law, that uses attractive and repulsive forces between the robots. Additionally to this, forces of the environment can be added, to affect the behavior of the swarm. The control laws are proven to be stable and collision free for simulated point-agents. Flocking with real non-holonomic robots was achieved

by Antonelli [4] by defining a set of rules, how a robot has to react to achieve a specific task, like obstacle avoidance, collision avoidance and approaching the goal. The correct rule is then chosen by a supervisor, which adapts the priority of the rule. In both [3] and [14], shape constraints are created by taking random samples within the wanted shape with a later relaxation to distribute the sample points more evenly. While [14] generally uses a random assignment from agents to those sampling points, [3] minimizes the squared distances of the swarm. In [3], additional constraint at intermediate times can be set and therefore make agents or even the center of the swarm meet exact constraints at specific times. Ho [7] presents a way to control the navigation of a flocking swarm, while navigating through obstacles and along paths with strong curvatures. They assume a flock that already has a given shape, where the connections between the agents is known. The motion of the flock is not rigid, but flexible, which leads to a natural flow of the swarm. Syamsuddin [12] also uses sample points of a skeleton to simulate particles forming a human body. Each particle is attracted by a fixed number of neighboring sample points that are next to each other. Since the particles all start at about the same position and are not real, the assignment and collision avoidance do not matter. The forces are designed in a way, that the particles can follow the sample points without oscillation. A force for obstacle avoidance is also introduced, which allows for simulations including obstacles.

3 Representing Drawings and Animations

This section describes the method for taking a drawing or animation created on the drawing interface and realizing it with robots.

3.1 Representing a Static Drawing

The starting point is the traditional flocking approach defined by Reynolds [9]. Flocking behavior is achieved using three steering rules which define forces between agents -

- Cohesion - steer towards the average position of neighboring agents
- Alignment - steer towards the average heading of neighboring agents
- Separation - steer to avoid close proximity with neighboring agents

The rules can be implemented in different ways, and the formulation here draws on Olfati-Saber [8]. An agent r_i has location q_i and holonomic velocity p_i. It can sense agents within a range r_α. A pair of agents r_i to r_j has a connecting line $\mathbf{n}_{ij}$, and an associated variable a_{ij} -

$$a_{ij} = \begin{cases} 1 & \text{if } \|q_i - q_j\| \leq r_\alpha \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\|q_j - q_i\|$ is the Euclidean distance between agent r_i and r_j.

Olfati-Saber showed that the following formulation for acceleration u_i of agent r_i incorporates all three Reynolds' rules if the robots are sufficiently close -

$$u_i = \sum_{j \in N_i} \phi(\|q_j - q_i\|)\mathbf{n}_{ij} + \sum_{j \in N_i} a_{ij}(p_j - p_i), \tag{2}$$

If the agents are within sensing range, $\phi(\|q_j - q_i\|)$ is a function that drives the two agents to adopt a separation d_α. If the agents are not within sensing range then $\phi(\|q_j - q_i\|)$ is 0. The second term adds a damping force between agents that drives an agent to adopt the same velocity as its neighbors. The result is flocking, an emergent collective motion of all the agents.

Parameter d_α is critical in determining the flocking behavior. Our formulation defines d_α based on characteristics of the target drawing. Assume that the drawing has total area is A and N is the total number of agents. Then -

$$d_\alpha = 2 \cdot c_g \cdot \sqrt{\frac{A}{\pi \cdot N}}, \tag{3}$$

The parameter c_g is related to neighbor connectivity as described in Olfati-Saber.

We augment the traditional steering rules by formulating an additional steering rule called the shape-steering rule, which has the effect of causing the robots to adopt a target shape. The process is -

- The target shape is transformed from screen coordinates of the drawing interface to ground plane coordinates of the physical workspace. The transformation can be done straightforwardly and directly because the aspect ratio of the canvas in the drawing interface is the same as the aspect ratio of the physical workspace.
- For each robot, a circle c_i with radius R is constructed. Subsequent processing for robot r_i depends on whether any of the target shape is contained within c_i.
- If c_i contains none of the target shape, then the nearest boundary point b_i of the target shape is found for r_i, and the steering force is towards b_i.
- If c_i contains some of the target shape, then the center of mass m_i is computed for the parts of the target shape within c_i, and the steering force is towards m_i.

The shape-steering rule causes robots outside the shape to move towards the shape, and it causes robots on the shape perimeter to move within the shape. It has no effect on a robot whose c_i is completely within the shape. As for distribution of robots within the shape, this results from the traditional Reynolds rules because of the choice of d_α in Equation (3).

The combined steering forces determine an acceleration, and thereby a required velocity p_i, at each robot r_i for the next iteration, suitable for holonomic robots. In fact the robots are non-holonomic, and the velocities p_i are modified to account for this as described in the next section.

If the drawing is formed by more than one shape, each of the robots is assigned to one of the shapes and ignores the other ones. The number of robots assigned to each shape can either be proportional to the area or specified by the user.

3.2 Implementation with Differential-Drive Robots

A holonomic velocity defines a preferred velocity for a non-holonomic agent. The preferred velocity is achieved using the velocity mapping method in [1] as follows.

The aim is for the non-holonomic agent to achieve a trajectory within a bounded error of the trajectory of the holonomic agent. The method has two parameters. Parameter T_0 is the time which determines how quickly the non-holonomic agent should achieve the orientation of the holonomic trajectory. Parameter ε is a a maximum tracking error, which enforces the proximity of the non-holonomic agent to the holonomic trajectory.

These two parameters limit the possible set of allowed holonomic velocities, as illustrated by Figure 2. If a holonomic velocity lies outside this set, it is optimally reduced to give the nearest allowed velocity. The allowed holonomic velocity is then transformed to a linear and angular velocity for the non-holonomic agent [1].

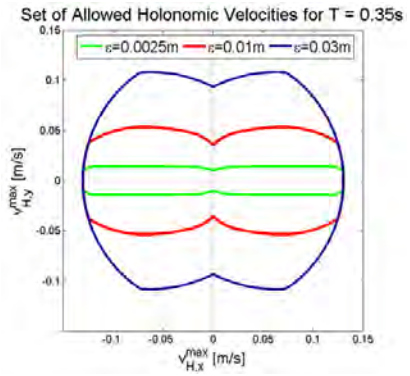

Fig. 2 Set of allowed holonomic velocity (SAHV) for a fixed T_0 and varying ε. A desired holonomic velocity is optimally reduced to the nearest allowed velocity in this set, and then transformed to a non-holonomic velocity and angular velocity.

There is no explicit handling of collision avoidance in this scheme. The Reynolds steering rule for separation prevents robots coming into close proximity, and collision rarely occurs in the real system. But no guarantee of collision-free trajectories is given.

3.3 Representing an Animation

This section describes how to create a robot animation. A user interacts with the drawing interface in the following way -

- Define the first action by creating one or more shapes on the drawing interface. Then repeat the steps below for each new action.
- Define a new action by adding one or more new shapes.
- Define a new action by transforming an existing shape s to a new shape s' using translation or rotation.
- Define a new action by an unconstrained morphing of an existing shape s to a newly drawn shape s'.

In its most basic form, the animation is thus a sequence of drawings that can be input directly to the algorithm in Section 3.1 to drive the robots. But one important modification is needed. In the case that the user creates multiple shape pairs (s_i, s_i') above, it is also possible to specify that the shape pairs maintain an association with the same group of robots g_i. This is useful, for example, for the case of an animated face. The robots for the eyes and mouth should not drift between different facial parts during the animation.

For the case where there are multiple triplets (s_i, s_i', g_i), $i > 1$, the method in Section 3.1 is modified as follows. The Reynolds Separation rule is applied across all of the robots, but the Reynolds Cohesion and Alignment rules, and the new shape-steering rule, are applied separately to each triplet (s_i, s_i', g_i). This achieves the desired result - a user can ensure that the robots for individual components of a drawing maintain their identity as they transition through an animation. And it avoids that robots are freely redistributing across the drawing in a confusing and unwanted way.

4 Comparison with a Goal-Directed Approach

The shaped flocking method in Section 3 is a non-explicit way to compute robot motion, with the motion being emergent from the flocking rules. A completely alternative approach is to do a geometric analysis of a target shape, and explicitly compute the goal positions for robots within the shape. This goal-directed approach is found in [2] and is not a contribution of this paper, but it does provide a basis to assess this work by a comparative analysis.

The goal-directed approach is described next, followed by a general discussion about both approaches.

4.1 Goal-Directed Approach

Given a target shape, a number of goal positions, equal to the number of robots, is determined by computing the Voronoi coverage of the target shape as described in [2]. The goal positions are fixed, and the robots are advanced to the goal positions over multiple iterations. At each iteration, each robot is associated with a goal position using some proximity criteria. Computed trajectories to reach the goals are modified as required for collision avoidance between robots. Each robot is then advanced for that iteration. The process stops when each robot is within a tolerance of its goal.

4.2 Discussion of Shaped Flocking and Goal-Directed Approach

This section addresses the question of why we are investigating shaped flocking given that the existing goal-directed approach could achieve similar goals. The motivation for our work is that flocking is known to be a powerful method for producing aesthetic effects in multi-agent systems. These effects are both visually compelling and they can be readily varied by changing the steering rules. We seek to build on that, while adding the ability to represent target shapes. In contrast, the goal-directed approach is explicit and rigid, and would require new components in order to vary the computed robot motion.

More specifically, flocking has these interesting properties -

- A change in one part of a shape affects only the local robots in a flocking approach. But it requires a complete recomputation of goal positions in the goal-directed approach.
- It is straightforward to add heterogeneous behavior of agents in a flocking approach. For example, some agents can be more attractive or repulsive. Or agents can be grouped like the shape triplets described in Section 3.3.
- Steering parameters in the flocking approach provide an opportunity for automated learning of a range of aesthetic effects.

These arguments suggest that flocking can provide the most flexible basis for producing varied and compelling visual effects.

5 Results

In this section we discuss the execution and the results of the conducted experiments. We first describe the experimental setup and then compare specific measures for the shaped flocking and the goal-directed approach.

5.1 Experimental Setup

The experimental setup is shown in Figure 3. There is a 2m-by-2m workspace for deployment of the robots. An overhead camera views the workspace, and a central computer provides the drawing interface (see Figure 1), tracks the robots, computes the required motion of the swarm, and sends wireless commands to direct the robots. Battery chargers line the perimeter of the workspace as shown in Figure 1, and the robots have the ability to automatically dock for recharging.

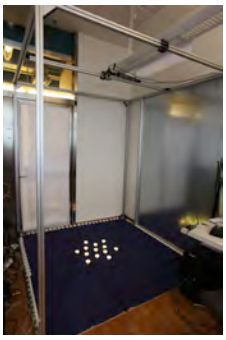

Fig. 3 The experimental setup - robots on the ground plane, overhead camera for tracking, and central computer for processing and sending wireless motion commands to the robots.

The overhead camera used for tracking is fixed at a height of 2.3 m above the deployment plane and can detect both visual and infrared light. The camera has 1600 x 1200 resolution, running at a frame rate of 10 Hz and can localize robots with precision better than 0.01 m.

Figure 4 shows the differential-drive robot used in the experiments. The RGB LED is under a translucent cover. The robot has two wheels driven by geared motors. The motors allow maximum speed of 0.5m/s. The onboard microcontroller runs a fast controller measuring the wheel speed using back-EMF of the motor. Speed commands are transmitted constantly from the host computer through a 2.4 GHz radio with a specific protocol that ensures 10Hz for up to 100 robots.

Figure 4 also shows a close-up of the charging unit. The robot docks by driving directly into the center channel of the metal component on the green strip, so that it touches the springed recharging contacts that are on either side of the channel.

Figure 5 shows examples of robot faces. Figure 6 shows an animation of a tree and Figure 7 of a fish.

5.2 Comparative Analysis

Table 1 provides a comparison of the shaped flocking algorithm and the goal-directed algorithm. Both algorithms can apply to a group of robots starting from arbitrary positions, although convergence results may vary. In these experiments

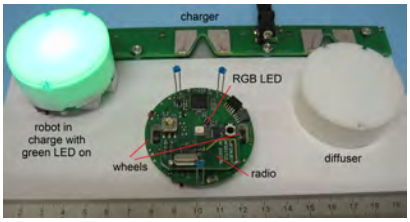

Fig. 4 Differential-drive robot with controllable RGB LED. The robot automatically docks in the charging plate for battery recharging.

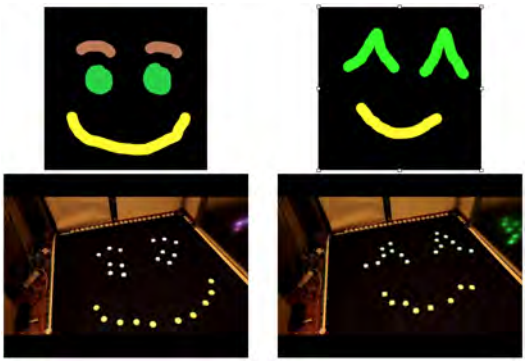

Fig. 5 Smiley faces - at top, the drawings created on the user interface and, at bottom, the corresponding robot configurations. Faces make a good choice for animation with wobbling eyebrows and dynamic smiles.

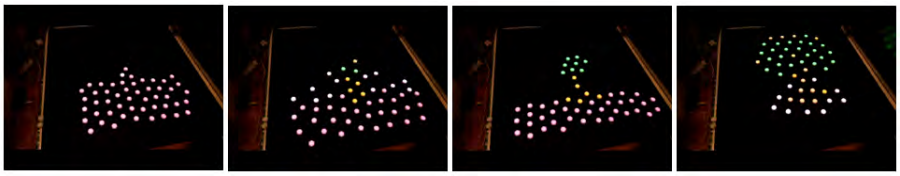

Fig. 6 Tree animation starting with flat earth at left, and growing into a full tree at right.

the robots were randomly distributed in the environment (approximately uniform distribution). The effect of the initial distribution of robots is further described in Section 5.3.

The first three rows display the time to converge to a new shape. Shaped flocking needs longer to converge to its final configuration. This is unsurprising because there are no explicit goal positions, and hence no direct motion toward the final configuration, unlike the goal-directed approach.

The coverage cost is a measure of quality of the distribution of robots through the shape and is given by

$$C = \sum_{u \in U} \min_{i} \|q_u - q_i\| \tag{4}$$

where q_i the location of robot i and U is the set of all pixels u of all patterns, with coordinates q_u.

Shaped flocking presents higher coverage cost. Again this is unsurprising because shaped flocking implicitly achieves low coverage costs via steering rules,

whereas the goal-directed approach explicitly precomputes goal positions by minimizing the coverage cost.

Table 1 Performance metrics for the shaped flocking and goal-directed algorithms.

Tree Animation	Shaped Flocking	Goal-Directed
Time to convergence (keyframe 1)	33.1 s	18.1 s
Time to convergence (keyframe 2)	14.5 s	5.1 s
Time to convergence (keyframe 3)	50.0 s	11.1 s
Final coverage cost (keyframe 1)	2647	2486
Final coverage cost (keyframe 2)	2635	2553
Final coverage cost (keyframe 3)	3241	2938

Figure 8 shows the progression of coverage cost for the animation of the fish in Figure 7. The shaped flocking method typically shows an increase in cost at the start of a new keyframe, which then reduces as the motion proceeds. As in the discussion above, this is a consequence of the implicit nature of flocking, which is not enforcing low cost configurations of the robots during the motion.

Fig. 7 Fish animation showing motion though user defined keyframes. The image series is arranged from left to right and top to bottom.

In summary, the goal-directed approach is superior in time to convergence and in coverage cost. However shaped flocking is still the primary focus of our research because

- It is a flexible basis for future work as argued in Section 4.2.
- Flocking motion is aesthetically appealing in our experiments, although we have not identified a way to quantify this.

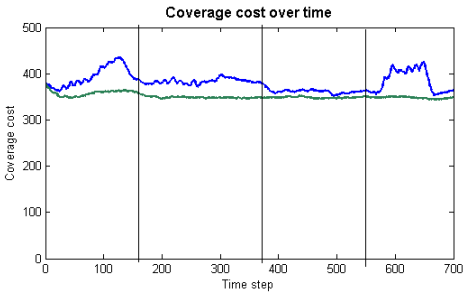

Fig. 8 Coverage cost progression for an animation of a fish. The blue line shows the cost of shaped-flocking, the black line of goal-driven method. Vertical lines indicate the beginning of a new keyframe in the animation.

5.3 Bottlenecks and Convergence to Local Minima

The experiments revealed a limitation in the shaped flocking algorithm for certain types of shape and depending on the initial conditions. Figure 9 shows an example with robotic Kanji. Once within the shape, the flocking rules disperse the robots. But for shapes with a pipe-like form, dispersion is slow i.e. it is a bottleneck problem. This is an issue where goal-direction does not encounter the same problem because goal positions are explicitly computed throughout the target shape. We have observed, as expected from the methods, that the goal-directed approach does present similar convergence results to arbitrary shapes independently of the initial conditions, whereas the shaped flocking algorithm strongly depends on the characteristics of the shape and the initial distribution of the robots (convergence to a local minima). For the latter, initial distributions that are homogeneously distributed over an area containing the shape typically present better results.

This problem requires further research. For example, additional forces toward unoccupied space in the shape could help to disperse the robots faster and drive the robots through bottlenecks. And the scaling strategy in [10] could be used to define optimal distances between robots to ensure better dispersion.

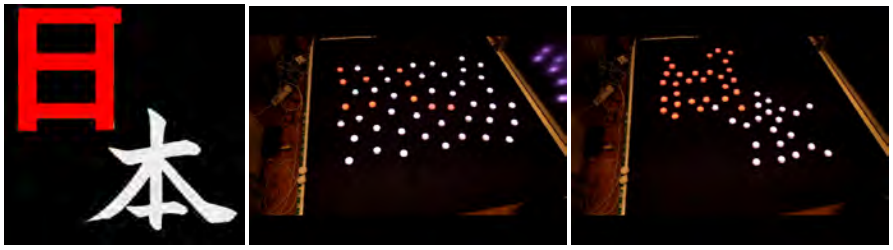

Fig. 9 Illustration of the bottleneck problem. **At left,** the user drawing. **At center,** robot start position. **At right,** the shape is convoluted, and diffusion of robots to achieve the shape is slow.

6 Conclusion

This paper has described shaped flocking, a novel algorithm for representing static and deforming shapes with a multi-agent system. Traditional flocking algorithms produce compelling collective motion in a group of agents. Shaped flocking extends this so that the agents additionally conform to a static or deforming shape. The algorithm was demonstrated in the context of a system that allows a user to do real-time direction of a swarm of robots via a drawing interface, in the field of entertainment robotics.

For comparison, shaped flocking was evaluated against a goal-directed approach. Conceptually the two approaches are different, with shaped flocking relying on emergent behavior to represent a target shape, while goal-direction relies on an explicit geometric analysis of the shape. Both approaches show good representational power and support real-time interaction. Although the goal-based approach shows superior results, we believe that shaped flocking is worth further investigation due to its flexibility and potential for creating robot art.

References

1. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: Proc. Int. Symp. on Distributed Autonomous Robotics Systems (2010)
2. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Siegwart, R., Beardsley, P.: Image and animation display with multiple mobile robots. Int. Journal of Robotics Research **31**, 753–773 (2012)
3. Anderson, M., McDaniel, E., Chenney, S.: Constrained animation of flocks. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '03, pp. 286–297. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
4. Antonelli, G., Arrichiello, F., Chiaverini, S.: Flocking for multi-robot systems via the null-space-based behavioral control. Swarm Intelligence **4**, 37–56 (2010)
5. Belta, C., Kumar, V.: Abstraction and control for groups of robots. Robotics, IEEE Transactions on **20**(5), 865 – 875 (2004)
6. Gu, Q., Deng, Z.: Formation sketching: an approach to stylize groups in crowd simulation. In: Proceedings of Graphics Interface 2011, GI '11, pp. 1–8. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (2011)
7. Ho, C., Nguyen, Q., Ong, Y.S., Chen, X.: Autonomous multi-agents in flexible flock formation. In: R. Boulic, Y. Chrysanthou, T. Komura (eds.) Motion in Games, *Lecture Notes in Computer Science*, vol. 6459, pp. 375–385. Springer Berlin / Heidelberg (2010)
8. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. Automatic Control, IEEE Transactions on **51**(3), 401 – 420 (2006)
9. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph. **21**, 25–34 (1987)
10. Rubenstein, M., Shen, W.M.: Automatic scalable size selection for the shape of a distributed robotic collective. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 508 –513 (2010)
11. Schwager, M., Mclurkin, J., Rus, D.: Distributed coverage control with sensory feedback for networked robots. In: in Proceedings of Robotics: Science and Systems (2006)

12. Syamsuddin, M.R., Kim, J.: Controllable simulation of particle system. In: Proceedings of the 7th international conference on Advances in visual computing - Volume Part II, ISVC'11, pp. 715–724. Springer-Verlag, Berlin, Heidelberg (2011)
13. Takahashi, S., Yoshida, K., Kwon, T., Lee, K.H., Lee, J., Shin, S.Y.: Spectral-based group formation control. Computer Graphics Forum **28**(2), 639–648 (2009)
14. Xu, J., Jin, X., Yu, Y., Shen, T., Zhou, M.: Shape-constrained flock animation. Computer Animation and Virtual Worlds **19**(3-4), 319–330 (2008)