A Distributed Scalable Approach to Formation Control in Multi-Robot Systems

Iñaki Navarro, Jim Pugh, Alcherio Martinoli, and Fernando Matía

Abstract A new algorithm for the control of formations of mobile robots is presented. Formations with a triangular lattice structure are created using distributed control rules, using only local information on each robot. The overall direction of movement of the formation is not pre-established but rather results from local interactions, giving all the robots a common, self-organized heading. Experiments were done to test the algorithm, yielding results in which robots behaved as expected, moving at a reasonable speed and maintaining the desired distances among themselves. Up to seven robots were used in real experiments and up to forty in simulation.

1 Introduction

A robot formation can be defined as a group of robots that moves as a collective maintaining pre-determined positions and orientations among its members. Having a pre-defined global shape is not always a requirement, and sometimes only relative positions between the members are defined.

A different but related problem to formation is flocking, where robots move as a group but the flock shape and the robots' relative positions are not strictly enforced, allowing robots to shift within the group. The first work in artificial flocking was a computer graphic animation of a group of birds by Reynolds [18]. With just three simple local rules, the artificial birds were able to move as a cohesive group: *Collision Avoidance*, avoiding collisions with other flock mates; *Velocity Matching*,

Iñaki Navarro and Fernando Matía
Intelligent Control Group, Universidad Politécnica de Madrid, José Gutiérrez Abascal 2, 28006 Madrid, Spain. e-mail: {inaki.navarro,fernando.matia}@upm.es

Jim Pugh and Alcherio Martinoli
Distributed Intelligent Systems and Algorithms Laboratory, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland. e-mail: {jim.pugh,alcherio.martinoli}@epfl.ch

matching velocity with nearby flock mates; and *Flock Centering*, trying to stay close to neighboring flock mates.

There are many applications in which robot formations can play an important role and be an advantage as compared to a single robot. For instance, formations of robots organized in a lattice can act as sensor arrays, allowing them to collect rich spatial information about their environment. Thus, formations can be very useful in search tasks, especially those in which the spatial pattern of the source can be complex as in the case of sound [17] or odor [8]. They can also be quite useful in mapping tasks (see multi-robot mapping in [9]). Redundancy in the measurements of both the environment and the relative positions among the robots might allow them to build more accurate maps than those generated using individually operating robots.

Robot formations in which only relative positions of nearby robots are known are classified by [2] into two groups: *Leader-referenced* and *Neighbor-referenced*. The first corresponds to those formations in which there exists a leader followed by one or more robots that may also act as leaders for further robots in the formation [6, 11, 14, 4]. Follower robots try to maintain a certain distance and angle to their leaders. The main disadvantage of this type of architecture is that it does not scale well with an increasing number of robots since the position error propagates cumulatively from leader to follower.

In the second type of formation, robots react to the positions of their neighboring robots, creating among the robots a regular pattern made up of squares, rectangles or triangles [3, 7, 12, 20, 21, 22]. No leaders are designated, and in principle the global shape of the formation is not pre-specified, and is determined solely by local interactions.

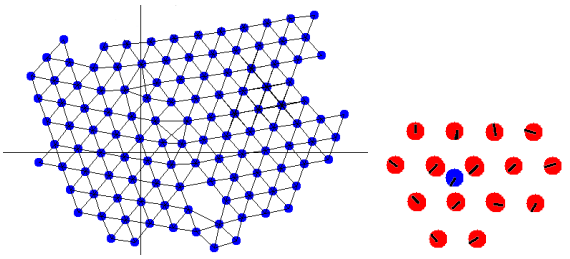
Although a lot of effort has been recently dedicated to stability of formations (see for instance [23, 5]), we do not consider stability analysis in this work and instead focus on the experimental validation of a fully distributed algorithm using real robots.

Some desirable characteristics that a formation may have are scalability in the number of robots, the ability to create different shapes, and avoiding obstacles at the group level. In order to have scalability in the number of robots, local sensing and communications and a decentralized controller are necessary [19].

The related work to the new algorithm presented in this paper is explained in Section 2. The newly developed algorithm is described in Section 3. A description of the set of experiments performed is given in Section 4, and their results are shown in Section 5. Finally the conclusions and future work can be found in Section 6.

2 The Physicomimetics Framework

The developed algorithm is based on the *Physicomimetics Framework* (PF) by Spears [20, 21, 22], that allows for the creation of a self-organized formation by using control laws inspired by physics. The controller is fully decentralized; each



(a) (b)

Fig. 1 (a) Triangular lattice formation result of PF. (b) Clustering phenomenon.

robot perceives the relative positions of its neighbors and reacts to attractive or repulsive virtual forces given by these positions, making the system scalable with the number of members. Using these forces, robots can position themselves to form triangular lattices.

The bulk of the work in PF has been done in simulation, treating robots as volumeless particles responding to the dynamics $\mathbf{F} = m\mathbf{a}$. Each of them is defined by a position $\mathbf{x}$, a velocity $\mathbf{v}$ and a mass m. Time is divided into time steps Δt. At each time step, every particle is driven by a perturbation $\Delta\mathbf{x}$, given by the current velocity, $\Delta\mathbf{x} = \mathbf{v}\Delta t$. Velocity is also modified at each time step by the force F that acts on every particle, $\Delta\mathbf{v} = F\Delta t/m$, depending on the relative positions of nearby robots. Only distance and bearing to neighbors is required.

In order to create a lattice distance R, an attractive force is applied when the inter-particle distance is greater than R, and a repulsive force when it is smaller than R. Force is given by $F_{ij} = \frac{K}{r^p}$ with $F_{ij} \leq F_{max}$, where F_{ij} is the force between two particles i and j with inter-particle distance r, and K and p are control parameters. Each robot is driven by the sum of all the forces of the neighboring robots that are at a distance $r \leq 1.5R$. This value is chosen since in a perfect triangular lattice distance R, nearest neighbors that are not part of the same triangle are $\sqrt{3}R$ away. Thus, by applying the sum of the neighboring robot forces to each virtual particle, a triangular formation such as that seen in Fig. 1a arises from random starting positions of the robots. The resulting lattice formation created is not perfect; holes may appear in the lattice, not all the triangles are equilateral, and lattices with different orientations may be produced.

One problem with the formation created is that, in some instances, particles are too close to each other. This clustering phenomenon occurs because one particle is stuck in a local minimum with respect to the forces from neighboring robots; at the same time, the particle's influence on its neighbors is not sufficient to force them to move and escape this deadlock situation. In Fig. 1b a schematic representation of the phenomenon is depicted. In real robot applications, clustering could be a major problem since robots could collide.

By just reacting to the neighboring robots, the formed lattice will remain stationary. In order to produce motion, PF adds a force towards a common goal that makes every robot move in the same direction. This violates the local property of the control law, since the goal can be distant, and possibly not seen by every robot.

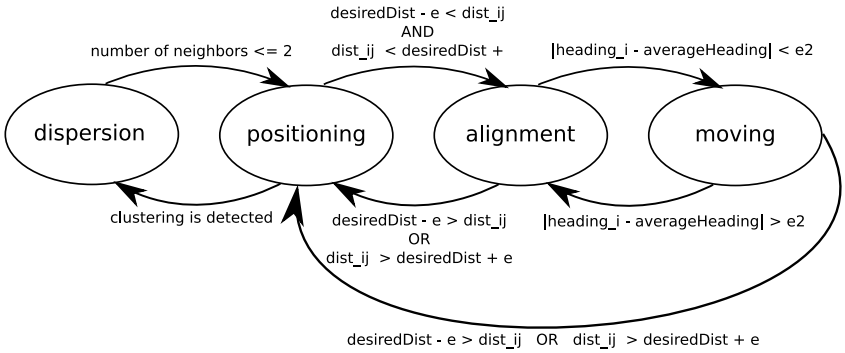


Fig. 2 FSM representation of the proposed algorithm.

In PF, robots are considered to be holonomic vehicles. In [21], it is explained that PF is used with nonholonomic robots, by taking a large Δt (in their case 22 s) and performing a rotational movement followed by a translational within a large time window, robots are able to achieve positions specified by a high-level control law assuming holonomicity. However, such a workaround would not be practical for most of the potential real-world applications.

3 Algorithm

As previously mentioned, PF is taken as a basis for the presented algorithm. The two major limitations of a PF-based algorithm are concerned with the assumption of robot holonomicity and lack of recipes for potential clustering phenomena. The new control algorithm solves both problems while simultaneously removing the need for a global goal in order to make the formation move.

The proposed algorithm must be able to function with nonholonomic robots; this means that once robots are in the proper positions, they must align themselves so that the formation can move as a whole in the same direction. Thus, the algorithm is implemented as a Finite State Machine (FSM) with the following four states: *Positioning*, *Alignment*, *Moving*, and *Dispersion*. States are innate to individual robots, so different robots may be in different states at the same time. The *Positioning* state works in a similar way to the PF, resulting in a triangular lattice. Once each robot considers itself well-positioned with respect to neighboring robots, it enters into the *Alignment* state and aligns its heading with its neighbors. When it is has achieved proper alignment it can enter into the *Moving* state that moves the group in a common direction. While in the *Positioning* state, each robot continuously evaluates whether it is stuck in a cluster. If this is the case, the robot enters into the *Dispersion* state that allows it to get out from this deadlock situation. The FSM and its state-to-state transitions are shown in Fig. 2.

3.1 Positioning State

Robots in this state react to attractive or repulsive forces generated by their neighbors which eventually result into a virtual velocity. The reaction force to each one of the neighboring robots is given by:

$$\mathbf{F}_{ij} = \frac{K}{distance_{ij}^2}\mathbf{u}_{ij} \tag{1}$$

where K is a parameter, $distance_{ij}$ is the distance between robot i and its neighbor j, and $\mathbf{u}_{ij}$ is the unitary vector pointing from one robot to the other. The force $\mathbf{F}_{ij}$ is attractive if $distance_{ij} > desiredDistance$, where $desiredDistance$ is the distance between two well-positioned robots forming a triangular lattice. Otherwise the force is repulsive. Only nearby robots at a $distance_{ij} < 1.3 * desiredDistance$ are considered neighbors. The resulting force is given by the sum of all the forces from the neighboring robots:

$$\mathbf{F}_i = \sum_j \mathbf{F}_{ij} \tag{2}$$

At each cycle, the application of this virtual force $\mathbf{F}_i$ modifies the virtual velocity:

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\mathbf{F}_i}{m}\Delta t \tag{3}$$

which is translated by the *low-level controller* to set the motor speeds of the robot.

3.2 Alignment State

In this state the robot aligns with neighboring robots in order to get a common heading within the group. Robots reach the *Alignment* state when the following conditions have been fulfilled:

- For every neighboring robot (robots with $distance_{ij} < 1.3 * desiredDistance$)

$$desiredDistance - e < distance_{ij} < desiredDistance + e \tag{4}$$

  where e is the tolerance error in the positioning. Once a robot has reached the *Alignment* state, a larger value of e is used.
- A given robot must have at least two neighbors in order to enter the *Alignment* state.

The *Alignment* algorithm works as follows. Each robot calculates the average heading of its neighboring robots that are already in the *Alignment* or *Moving* states. Those in the *Positioning* state might be turning to position themselves, so their heading is not considered. A proportional controller using the difference between personal heading ($heading_i$) and the average heading ($averageHeading$) is applied:

$$v_{angular_i} = K_3 * (averageHeading - heading_i) \tag{5}$$

The linear velocity is set to 0.

$$v_{linear_i} = 0 \tag{6}$$

All robots in the *Alignment* state apply the same controller, so by simply matching the heading locally a global common heading eventually arises at the group level.

In principle robots might detect the distance and direction to their neighbors using an on-board relative positioning system as described in [15]. While it is more difficult to determine the heading of their neighbors, robots can couple relative positioning with communication in order to get this information [16]. Communication is also necessary to know the state of the neighboring robots.

Once robots meet *Positioning* state and *Alignment* state conditions, they enter into the *Moving* state. The alignment condition is that the error in the heading with respect to the average heading of its neighbors must be under a certain level (e_2): $|heading_i - averageHeading| < e_2$.

3.3 Moving State

In this state robots should share a common heading. Thus, by applying a virtual force in the forward direction, they will move together. In addition, drift within the formation could result in some positioning error with respect to the neighbors so the sum of the forces $\mathbf{F}_{ij}$ used in the *Positioning* state is also applied:

$$\mathbf{F}_i = K_4 \sum_j \mathbf{F}_{ij} + K_5 \mathbf{F}_{Forward_i} \tag{7}$$

where K_4 and K_5 are two constants. As in the *Positioning* state, a local control law (see Section 3.5) translates this $\mathbf{F}_i$ force into motor speeds. In this manner, local rules applied by every robot result in a common, global movement of the complete formation. The group velocity is determined by $K_5 \mathbf{F}_{Forward_i}$.

3.4 Dispersion State

Clustering phenomena occur when robots are stuck in the middle of the lattice and have insufficient force to push the other robots around them. This phenomenon is due to local minima of the inter-robot forces and can be solved by adding stochastic noise to the controllers, as in [1]. It can be minimized by setting the appropriate values in the control parameters, but it is not guaranteed that it will never happen. Experimentally it was observed that depending on the initial configuration of the robots, this phenomenon was more or less likely to take place, with higher chances

for larger formations. It also appears when a robot is introduced into the middle of an already formed lattice, since the force that it creates on its neighboring robots is not sufficient to modify the created structure, and the robot finds itself in a deadlock.

In order to overcome the clustering phenomenon, a solution was chosen which first identifies and then solves the problem. Each of the robots maintains a variable that represents an estimation of the probability of being in a cluster. When a robot identifies itself as being part of a cluster it enters into the *Dispersion* state, leaving the formation; this is accomplished by navigating to the border of the formation and not allowing the other robots to react to its position. As result, the cluster disappears and the robot rejoins the formation at the border, entering into the *Positioning* state.

The probability of being part of a cluster is increased if the average distance to the closest neighbor is below a certain threshold and keep decreasing over successive time steps, otherwise this probability is decreased. Every step the robot verifies whether it is in a cluster by randomly choosing a value between 0 and 1 and comparing this value with the probability of being in a cluster. This randomized process represents a noisy perturbation on the controller as proposed in [1].

The process of detecting the clustering randomly reduces the probability of two nearby robots detecting the clustering at the same time and leaving the cluster simultaneously, which results in a hole in the formation.

3.5 Low-Level Controller

A given virtual velocity, generated by both the *Positioning* and *Moving* algorithms, must be translated into wheel movements. The *low-level controller* is designed for mobile robots with differential drive configuration.

The controller is inspired by a similar one described by Hsu [10], but augmented to allow backwards movement. The angle (θ_i) and magnitude ($|v_i|$) of the virtual velocity are the inputs for getting these two velocities:

$$v_{linear_i} = K_1 * |v_i| * cos(\theta_i) \tag{8}$$

$$v_{angular_i} = \begin{cases} K_2(\theta_i + \pi), & \text{if } \theta_i < -\pi/2 \\ K_2\theta_i, & \text{if } \pi/2 > \theta_i > -\pi/2 \\ K_2(\theta_i - \pi), & \text{if } \theta_i > \pi/2 \end{cases} \tag{9}$$

The sum of the linear and angular velocities is translated to motor speeds taking into account the kinematics of differential drive robot as follows:

$$s_{motor-right_i} = v_{linear_i} + B * v_{angular_i} \tag{10}$$

$$s_{motor-left_i} = v_{linear_i} - B * v_{angular_i} \tag{11}$$

where B is half the distance between the two wheels, $s_{motor-right_i}$ is the speed of the right motor, and $s_{motor-left_i}$ is the speed of the left motor.

Any other controller based on the inverted kinematic model could be used instead of the proposed one.

4 Experiments

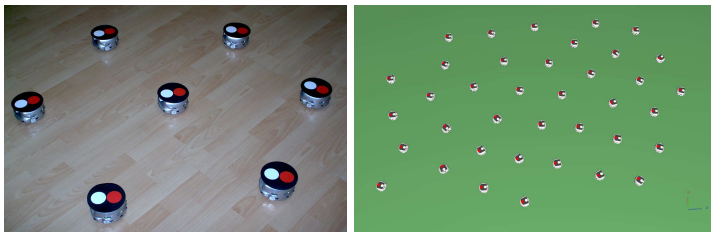
In order to test the algorithm up to seven Khepera III robots were used. These robots have an on-board Linux system with wireless LAN communications, 2 motors in differential drive configuration, a ring of 9 infrared sensors, and 5 ultrasound sensors. The size of these robots is 12 cm in diameter.

Experiments were first done using the Webots simulator [13], with a realistic model of the robots. This allowed us to perform the experiments in a fast way, tuning the different parameters easily. Experiments in simulation were performed working with formations of 40 robots.

The simulator has a built-in relative positioning system that gives information about the distance and direction to neighboring robots within line-of-sight. It also allows the exchange of information between them. This simulates an omnidirectional infrared relative positioning system, such as the one presented in [15], which allows for the simultaneous exchange of messages and assessment of range and bearing information.

In our real robot experiments, in order to detect the positions of nearby robots and exchange information between them, an overhead camera tracking system was used to emulate on-board relative positioning. A camera was connected to the ceiling looking down at the arena and linked to a computer. The computer used a tracking tool which sent the information on the positions of the robots via UDP packages using the wireless LAN system. This system has an error of up to 1 cm in position and 1 degree in angle. Robots received these packages and calculated the relative positions to their neighbors taking into account occlusions and maximum communication range. In addition Gaussian noise was added to more accurately emulate a real on-board relative positioning system. The values for the noise were chosen according to the previous characterization of a relative positioning system in [15], using a standard deviation of 0.1 radians for the bearing and 10% of the distance for the range. This emulated system could be replaced with a recently developed system which offers comparable performance. The positioning information was updated approximately every 300 ms (a delay due to the image acquisition and processing). This value is quite high and according to [16] it might represent the bottleneck of the system, since robots cannot move very quickly without risk of oscillation.

We ran 30 simulated experiments and 20 real robot experiments with an inter-robot desired distance of 60 cm. The 20 real-world experiments were carried out using seven Khepera III robots that were placed in a small cluster. Another set of 20 experiments were done in Webots simulation with a model of the Khepera III robot and the same starting poses as the real-robot experiments. The last 10 experiments correspond to simulations with 40 robots with initial positions in a square lattice



(a) (b)

Fig. 3 Two formations: (a) seven Khepera III robots, (b) forty Khepera III robots in simulation.

Table 1 Results on average for the different type of experiments

	Mean Position Error (m)	Mean Angle Error (rad)	Group Speed (m/s)	Time Positioning	Time Alignment	Time Moving
7 Real Robots	0.052	0.283	0.011	36.8 %	27.9 %	35.3 %
7 Simulated Robots	0.043	0.123	0.018	26.5 %	55.3 %	18.2 %
40 Simulated Robots	0.056	0.488	0.009	51.2 %	28.2 %	20.6 %

and random headings. An image of seven real Khepera III in formation can be seen in Fig. 3a, while in Fig. 3b, a larger simulated formation is shown.

5 Results

The 50 experiments we ran worked quite well, since robots were eventually able to create a triangular lattice formation that was able to move as a whole in an emergent direction decided upon by all robots.

For each of the experiments, four metrics were analyzed: mean position error (averaged over the different robots), mean heading error (averaged over the different robots), speed of the center of mass of the formation (group speed) and the internal state of the robot (*Positioning*, *Alignment* or *Moving*). *Dispersion* state was not differentiated and considered as *Positioning* state. The averages over multiple runs of the first three values as well as the percentage of time in each internal state are given in Table 1 for the different experiments. They are determined by taking the values after some initial starting time, once the formation is already created. This time was set manually to 40 s in the case of experiments with seven robots, and 100 s for experiments with 40 robots.

As can be seen, the average values do not change much between real robots and simulation in the case of 7 robots. Both mean position error, and group speed are quite similar. Group speed has a value of around 1 cm/s, which (taking into account that the maximum enforced speed for a robot is 3 cm/s) is a good result. The mean position error is about 5 cm, 8.3% of the desired distance of 60 cm, which is fairly

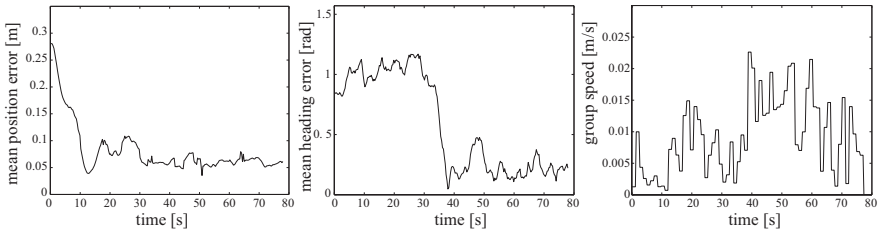


Fig. 4 Evolution of mean position error, mean heading error and group speed in an experiment using seven real robots.

impressive considering the dynamic nature of the system and the 10% range error from relative positioning. The percent of time that on average robots stay in each internal state shows differences between reality and simulation. We suspect this may be due to imperfections in the simulation model. The communication between real robots is asynchronous, with delays and lost packets, while communication in the simulator is synchronous and with no delays or losses. A better communication model would likely give closer results.

Experiments with 40 robots have similar results to those with just seven. Mean position error, internal state and (more importantly) group speed are quite similar, demonstrating empirically that the algorithm can scale well with increasing numbers of robots, thanks to their control laws that are distributed and local. The mean angle error is larger in the case of 40 robots, as well as the time in the *Positioning* state; the reason for this might be that, because of propagation of error in bearing measurements and the high convergence time, all the robots are not heading in the exact same direction. However this does not seem to be a major problem since the formation does not split.

The workings of the algorithm can be understood better by observing the graphs of Fig. 4 that show the average values of the mean position error, mean heading error, and group speed over time for an experiment with seven real robots. In the initial phase of the experiment, the mean position error and mean heading error are quite big and most of the robots are in the *Positioning* state. After some time (around 15 seconds) the mean position error decreases significantly, but the mean heading error remains quite large. The explanation for this is that most robots are already well placed but still need to align themselves. After around 40 seconds, the mean heading error also decreases, indicating that the robots are aligned and beginning to move, as can be seen in the group speed graph. Once robots start moving the group speed and mean heading error have oscillations. This is due to heading errors gradually increasing while moving, requiring the formation to periodically stop and re-align. When it is properly aligned, it starts moving again. This pattern can be seen in all of the experiments done, both in simulation and with real robots. Some videos of real world and simulation experiments can be found at http://138.100.76.231/inaki/dars08/index.html

6 Conclusions and Future Work

The algorithm for the control of robot formations presented here appears to work reasonably well according to our experimental results. A formation with an internal triangular lattice structure is created, with an external shape which is not predefined. Working with seven real robots, we observed that a formation emerges, just using local rules, local sensing and minimal communication, and that the formation is able to agree on a common direction of movement with no global information. Experiments in simulation with 40 robots show the scalability with respect to the number of robots in the formation. There is no evidence that the algorithm will not work for larger groups of robots although its performances might be gracefully decreasing with large swarms. Both group speed and mean error in position have reasonable values in simulation and real experiments.

Scalability can be an interesting property in certain applications such as distributed search and mapping. In addition, the absence of any leader and the use of many robots make the formation tolerant to individual robot failures.

The controllers could be improved using fuzzy controls to eliminate the discrete changes in behavior produced when jumping from one state to another in the FSM. Stability properties of the formation should be analyzed. Another possible improvement would be deciding the direction of movement not by all the members of the formation, but by those that have more relevant information. For instance, robots on the periphery of the formation could have more information about the environment and could lead the rest in the right direction, without any need for communication just by making the others copy their headings. In this way, obstacle avoidance at the group level could be implemented.

7 Acknowledgements

I. Navarro is sponsored by Madrid Region with a Ph.D. grant (FPI-2005), that allowed him to develop this work at EPFL. I. Navarro and F. Matía are partially sponsored by Spanish Ministry of Science (ROBONAUTA project DPI2007-66846-C02-01). J. Pugh has been sponsored by a Swiss NSF grant (contract Nr. PP002-116913).

References

1. Arkin, R.C.: Motor schema-based mobile robot navigation. The International Journal of Robotics Research **8**, 92–112 (1989)
2. Balch, T., Arkin, R.: Behavior-based formation control for multi-robot teams. IEEE Transactions on Robotics and Automation **14**(6), 926–939 (1998)
3. Balch, T., Hybinette, M.: Social potentials for scalable multi-robot formations. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 73 – 80 (2000)

4. Das, A., Fierro, R., Kumar, V., Ostrowski, J., Spletzer, J., Taylor, C.: A vision-based formation control framework. IEEE Transactions on Robotics and Automation **18**(5), 813–825 (2002)
5. Fax, J., Murray, R.: Information flow and cooperative control of vehicle formations. IEEE Transactions on Automatic Control **49**(9), 1465–1476 (2004)
6. Fredslund, J., Mataric, M.J.: A general algorithm for robot formations using local sensing and minimal communication. IEEE Transactions on Robotics and Automation, Special Issue on Multi Robot Systems **18(5)**, 837–846 (2002)
7. Fujibayashi, K., Murata, S., Sugawara, K., Yamamura, M.: Self-organizing formation algorithm for active elements. In: SRDS '02: Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02), pp. 416–421. IEEE Computer Society (2002)
8. Hayes, A.T., Martinoli, A., Goodman, R.M.: Distributed odor source localization. IEEE Sensors Journal **2**(3), 260–271 (2002). Special Issue on Artificial Olfaction, H. T. Nagle, J. W. Gardner, and K. Persaud, editors
9. Howard, A.: Multi-robot mapping using manifold representations. In: IEEE International Conference on Robotics and Automation, vol. 4, pp. 4198–4203 (2004)
10. Hsu, H., Liu, A.: Multiagent-based multi-team formation control for mobile robots. Journal of Intelligent and Robotic Systems **42**, 337–360 (2005)
11. Kostelnik, P., Samulka, M., Janosik, M.: Scalable multi-robot formations using local sensing and communication. In: Proceedings of the Third International Workshop on Robot Motion and Control, 2002. RoMoCo '02, pp. 319– 324 (2002)
12. Martinson, E., Payton, D.: Lattice formation in mobile autonomous sensor arrays. In: E. Sahin, W. Spears (eds.) Swarm Robotics Workshop, no. 3342 in Lecture Notes in Computer Science, pp. 98–111. Springer-Verlag, Berlin Heidelberg (2005)
13. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems **1**(1), 39–42 (2004)
14. Naffin, D.J., Sukhatme, G.S.: Negotiated formations. In: International Conference on Intelligent Autonomous Systems, pp. 181–190 (2004)
15. Pugh, J., Martinoli, A.: Relative localization and communication module for small-scale multi-robot systems. In: IEEE International Conference on Robotics and Automation, pp. 188 – 193 (2006)
16. Pugh, J., Martinoli, A.: Small-scale robot formation movement using a simple on-board relative positioning system. In: O. Khatib, V. Kumar, D. Rus (eds.) International Symposium on Experimental Robotics, vol. 39, pp. 297–306. Springer Tracts in Advanced Robotics, Rio de Janeiro, Brazil (2006)
17. Pugh, J., Martinoli, A.: Inspiring and modeling multi-robot search with particle swarm optimization. In: IEEE Swarm Intelligence Symposium, pp. 332–339 (2007)
18. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics **21**(4), 25–34 (1987)
19. Sahin, E.: Swarm robotics: From sources of inspiration to domains of application. In: E. Sahin, W. Spears (eds.) Swarm Robotics Workshop, no. 3342 in Lecture Notes in Computer Science, pp. 10–20. Springer-Verlag, Berlin Heidelberg (2005)
20. Spears, W., Spears, D., Hamann, J., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots **17**, 137–162 (2004)
21. Spears, W., Spears, D., Heil, R., Kerr, W., Hettiarachchi, S.: An overview of physicomimetics. In: E. Sahin, W. Spears (eds.) Swarm Robotics Workshop, no. 3342 in Lecture Notes in Computer Science, pp. 84–97. Springer-Verlag, Berlin Heidelberg (2005)
22. Spears, W.M., Heil, R., Spears, D.F., Zarzhitsky, D.: Physicomimetics for mobile robot formations. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1528–1529. IEEE Computer Society (2004)
23. Tanner, H., Jadbabaie, A., Pappas, G.: Stable flocking of mobile agents, part i: fixed topology. In: Proceedings. 42nd IEEE Conference on Decision and Control, vol. 2, pp. 2010–2015 (2003)