Target surrounding solution for swarm robots*

László Blázovics[1], Tamás Lukovszki[2,1], and Bertalan Forstner[1]

[1]Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Budapest, Hungary
{laszlo.blazovics,bertalan.forstner}@aut.bme.hu
[2] Faculty of Informatics, Eötvös Lóránd University, Budapest, Hungary
lukovszki@inf.elte.hu

Abstract. In this paper we present a distributed algorithm, which enables to follow and surround moving objects by a swarm of homogenous robots that only use local sensing. We introduce the multi orbit surrounding problem and present a solution for it. We prove that our solution always guarantees that the robots enclose the target and circulate around them. We also evaluate our solution by simulations.

Keywords: robot swarm, distributed control, convergence

1 Introduction

The cooperative control of biological and artificial intelligence have been being a popular research filed of many scientific researchers. The origin of the research of the *swarm behavior* can be dated to 1987. In this year Reynolds [13] published a method about creating realistic flock motion in a virtual reality. Since this the way of controlling multiple entities has changed significantly. Instead of using a centralized coordinator for controlling the movements of the other entities, Reynolds described and implemented a decentralized control method.

Reynolds achieved a realistic flocking model by the definition of the following three simple rules, which should be individually obtained by each *virtual bird*:

1. Match the velocity of the other members of the swarm.
2. Avoid collision.
3. Avoid getting too far from the others.

By defining these simple rules the computation complexity was dramatically reduced.

In order to control the formation and the direction of the motion of these kind of asynchronous systems many solutions are using potential or gravitational fields to *move* the entities [5]. Besides the low computational complexity it is also oblivious, which is ideal for the desired artificial robots of the swarms.

* This work is connected to the scientific program of the "Development of quality-oriented and cooperative R+D+I strategy and functional model at BUTE" project. This project is supported by the New Hungary Development Plan (Project IDs: TÁMOP-4.2.1/B-09/1/KMR-2010-0002, TÁMOP-4.2.1/B-09/1/KMR-2010-0003).

Our contribution: We introduce the multi-orbit surronding problem, where autonomous robots with limited sensing range have to surround a target and circulate around it. We describe a simple algorithm to solve this problem, where each entity only needs information about the entities in its local environment and the target. We prove the correctness and the convergence of our solution and also provide simulation results.

Outline of the paper: This paper is organized as follows. Section 2 gives an overview of related work. In Section 3 we introduce our multi-orbit surrounding algorithm and mathematical notations. We prove the convergence of the algorithm in Section 4. Section 5 presents our experimental results and our simulation environment. Finally, Section 6 summarizes the work.

2 Related work

There are many other working implementations of swarms which rely on the work of Reynolds like the RAVEN project of the MIT[8] the UAV system [3], or a military solution for UGVs [1]. The main goal of these implementations is to design a stable group of quasi autonomous robots that can keep dedicated formations.

One of the most simple solution is created by Mataric [12]. She defined five easily adoptable rules which comforms to the rules of Reynolds. These five behaviors are the following.

1. *Safe-Wandering:* The ability of a group of agents to move about while avoiding collisions with obstacles and each other.
2. *Following:* The ability of an agent to move behind another retracing its path and maintaining a line or queue.
3. *Dispersion:* The ability of a group of agents to spread out in order to establish and maintain some minimum inter-agent distance.
4. *Aggregation:* The ability of a group of agents to gather in order to establish and maintain some maximum inter-agent distance.
5. *Homing:* The ability to find a particular region or location.

By using these five basic rules more dynamic and complex behaviors can be constructed. However the effectiveness of this solution was shown only through simulations.

From the mathematical perspective there are many approaches which are able to clearly formalize the swarm behavior. Most of that are using the already introduced artificial potential fields [7], [4], [9], [10], [11], in order to define interaction control forces between neighboring entities and the environment itself. The benefit of this formalism was the easy provability of the stability of the overall system.

Gazi et al. [7] have presented a virtual potential field based solution which are able to stabilize swarm of homogenous entities in an n-dimension Euclidean space. Based on this approach Chu [4] has extended this to be able to handle

heterogenous entities. However, by using interactive matrix, the final shape of the swarm will be rough in contrast with the sphere form of Gazi's solution. Neither of these solutions are taking care of those situations where the average velocity of the swarm is not zero.

Leonard and Fiorelli [11] proposed a concept for the above mentioned problem. They introduced virtual leaders, which are moving "independently", while they are followed by the rest of the entities.

Hsieh et al. [10] also used virtual potential fields in their solution in order to keep entities in a dedicated smooth shape "orbit". Once the entities have reached the given trajectory, they are using the tangential speed to avoid inter-agent collisions.

Barnes et al. [2] has presented a similar methodology to the work of Hsieh. However they have been using multiple *weight* functions depending on the distance from the center of the potential filed. We have modified and extended their weight functions in order to decrease the time while the entities are reaching the desired trajectory.

Cohen and Peleg [5] presented an asynchronous algorithm to gather entities at the center of gravity. Their algorithm uses the LCM (Look-Compute-Move) discrete cycle based model to move their robots. They mathematically proved upper and lower bounds on the convergence speed of their solution. Cord-Landwehr et al. [6] described an easy-to-check property of target functions that guarantee convergence and gives upper time bounds. This property holds for the target function in [5] and improves the upper bound on the speed of the convergence.

3 Multi-orbit surrounding

In this section we will present our solution for target surrounding. We have been using a similar potential field based approach as it was introduced and used by Hsieh et al. [10] and Barnes et al. [2]. However in order to achieve a faster and more uniform surrounding process we have been using a more complex potential field.

3.1 The concept

If multiple entities are trying to capture a moving target one of the most relevant tactic is the surrounding. Most animal groups who are hunting in group - like wolf packs - are following this strategy. Therefore, the main idea of our method is to define a trajectory around the target on which the entities should take place. However, in order to assure that the target cannot escape from the circle of the entities - i.e. the circle is not uniformly filled and there a holes on it - , they should move around him. In order to minimize the possibility of inter agent collision we have defined a heading direction on the trajectory. Nevertheless by the introduction of this strict direction rule on the desired trajectory prohibits the use of a usual pincer movement.

In order to solve this issue, we have introduced the *multi-orbit surrounding* mechanism. We have defined multiple orbital trajectories around the target which are moving with it. A simple example can be seen on Fig. 1.

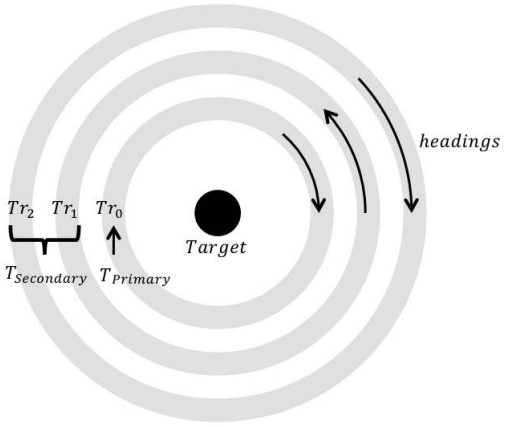


Fig. 1. Three circular trajectory curves and their *headings* around a given target.

Each trajectory has a different heading direction like their neighbors. This serves two purposes. First it enables the possibility of the pincer movement while keeping the probability of inter-agent collision low. Each neighboring trajectory pair has different heading direction, which implies that there should be some distance between them in order to minimize the collision probability. Second it accelerates the surrounding process, i.e. if an inner trajectory contains a hole, an additional robot from the next trajectory can fill it in shortest time.

The basic behavior of the entities between two trajectories is a radial movement around the target.

As it can be seen there are two types of trajectories. The first type is the *primary trajectory* ($T_{primary}$). This is the nearest trajectory around the target. The main goal of the entities is to put themselves into orbit on this trajectory. Whenever an entity has reached this trajectory, it stops radial movement and start to move around the target in the given heading direction.

The second type of trajectories is the *secondary trajectory* ($T_{secondary}$). These trajectories are more distant than the primary. If an entity is passing through one of this trajectories during the surrounding process and it is sensing another entity in front of it, it should put itself onto orbit on the current trajectory. By doing this not only the collision will be avoided but the surrounding process will be accelerated: a robot in the secondary trajectory and a hole in the primary trajectory move towards each other.

However if the entities are not staying on any of the above introduced trajectories, it is not allowed to make tangential movements around the target.

Our solution uses the gradient vectors of the potential field to *move* the entities to the desired direction. The *potential* of the entities are decreasing while they are approaching the desired trajectory. We are calculating with constant velocity.

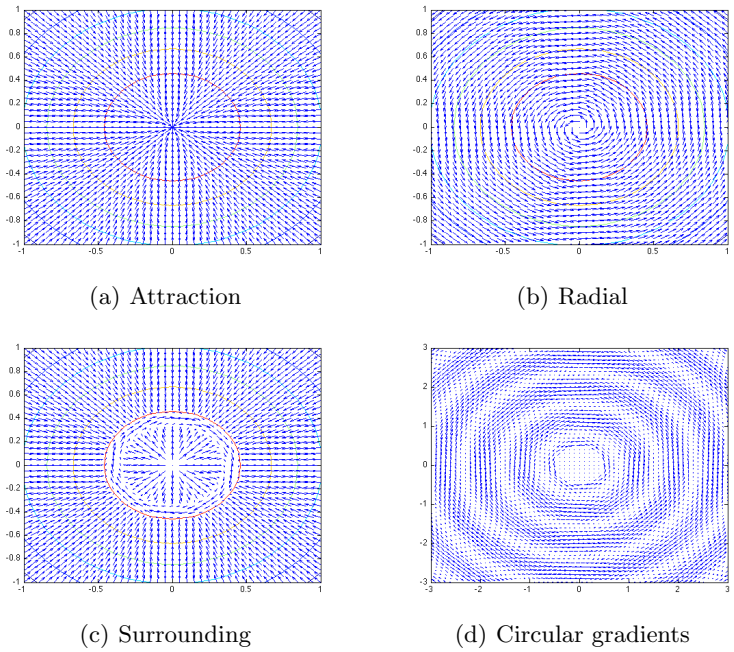
3.2 Potential field based approach

In this section we present the mathematical notations for our potential field based *multi-orbit surrounding* solution.

Let the two dimensional potential function - which generates the potential field - of the target be the following:

$$f_{(x,y)} = e^{-\alpha((x-x_c)^2+(y-y_c)^2)} \tag{1}$$

The $x_c,\ y_c$ are the coordinates of the target, α is a positive parameter.



(a) Attraction (b) Radial

(c) Surrounding (d) Circular gradients

Fig. 2. Different vector fields generated with the gradients

The gradient of this function can be seen on the Figure 2(a) and it is look like as follows:

$$d_x = -2\alpha f(x,y)(x - x_c)$$
$$d_y = -2\alpha f(x,y)(y - y_c) \tag{2}$$

This gradient can be used for a pure attractive function. In order to keep the entities away from the close proximity of the target, a repulsion function should be used. This can be achieved by inverting the direction of the gradients.

$$d_x = 2\alpha f(x,y)(x - x_c)$$
$$d_y = 2\alpha f(x,y)(y - y_c) \tag{3}$$

The next part of our *function gradient set* is the perpendicular gradients. This will be used for the circulation on the trajectories. These gradients can be generated by a simple rotation. See Figure 2(b).

$$d_x = 2\alpha f(x,y)(y - y_c)$$
$$d_y = -2\alpha f(x,y)(x - x_c) \tag{4}$$

In order to limit the attraction and repulsion and perpendicular forces to the desired areas, weight functions should be used. For the attraction and repulsion functions the following weight function is sufficient:

$$W_{rad} = \frac{1}{1 + e^{-\alpha_{out}(r-(R+R_{out}))}} - \frac{1}{1 + e^{\alpha_{in}(r-(R-R_{in}))}} \tag{5}$$

Where R is the radius of the primary trajectory, r is distance function which is rotation invariant, α_{in}, α_{out}, R_{in} and R_{out} are positive parameters. More detailed description can be found in [2].

In order to localize the perpendicular forces to the primary trajectory another weight function should be introduced:

$$W_{tan} = e^{-\alpha_{\perp}(r-R)^2} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{6}$$

Where $\alpha_{\perp}$ is a positive parameter. By combining these two weight function in the following equation of $V = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$, the entities will move until they reach the primary trajectory where they start their orbital motion, as it can be seen on Figure 2(c).

$$V = (W_{rad} + W_{tan})\,\partial x \tag{7}$$

Where $\partial x = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$.

In order to extend this system with our multi-trajectory solution we have generated a new weight function which can be seen on Figure 2(d) and looks like as follows:

$$S_{tr} = (2 - \frac{1}{2 + e^{r}})sin(\alpha_{tr} r) \tag{8}$$

$$S_{out} = \frac{1}{1 + e^{\alpha_{out}(r-(R+R_{out}))}} \tag{9}$$

$$W^{*}_{tr} = S_{out} S_{tr} \tag{10}$$

The S_{tr} component generates the distinct trajectory curves with a slightly increasing tangential speed while the S_{out} limits its gain to the outside of the primary trajectory.

However it is not efficient when every entity start to move along on the secondary trajectories before reaching the desired trajectory. This is only required when they are not able to move directly towards the desired position. This can only happen, when there is another entity in the way, which means that this newly introduced weight is required only in collision avoidance situations.

Therefore we have extended this by adding *proximity* parameters to it:

$$W_{tr} = \sum_{i \in X} W^{*}_{tr} \left(\lambda - \frac{\lambda}{1 + e^{-\alpha_{avoid}(r_{avoid} - R_{avoid})}} \right) \tag{11}$$

The λ is a parameter like the κ, nevertheless it should be minor: $\lambda < \kappa/2$.

The final velocity vector of a given entity looks like as follows:

$$V = (W_{rad} + W_{tan} + W_{tr})\, \partial x \tag{12}$$

4 Analysis of the convergence

In this section we prove that by using of the above introduced surrounding method a swarm of robots will always enclose a given target. For this we use following assumptions.

Assumptions:

1. *The entities are modeled as points.*
2. *There is only one static target at the same time.*
3. *All entities know the position of the target.*
4. *All movements of the entities are divided into discrete steps.*
5. *In each step a robot can move a unit distance in the direction determined by the potential field or stay in place if this movement is prohibited by another robot.*
6. *The distance $d(u, v)$ between each pair of entities u, v must be at least some constant $d_{min} < 1$.*
7. *The distance between neighboring trajectories are 1 unit, i.e. $r_{i+1} - r_i = 1$, $i = 0, 1, ...,$ where r_i is the radius of the i^{th} trajectory T_i.*

8. *The sensing range of the entities is at least $d_{min}+2$. Thus, each entity which could get closer than d_{min} after one time step is within the sensing range.*
9. *The entities are able switch between neighboring trajectories within one time step.*

Remark: For simplicity, we assume a static target. This assumption can be changed for a slowly moving target. In that case the unit velocity of the robots must be relativ to the velocity of the target.

4.1 Convergence

Each entity tries to move straight into the direction of the target. If it is not possible, because it would get too close to another entity, it moves on the trajectory containing their current position. Note that by Assumption 5,6, 7, and 8, all entities that could be closer to an entity v than d_{min} in the next time step and could cause a collision with v, are within the sensing range of v.

We assume that at the beginning the distance between each pair of eintities u and v is at least d_{min}. We prove that the entities always can move, never stuck in deadlock situation and we give a convergence guarantee of the surrounding process.

We say that two entities u and v *prohibit* each others movement if the above rule would cause collision. In that case the entity with lover potential is allowed to move. In order to break ties, we assume that each entity $v \in V$ has a unique ID denoted by $v.ID$ and each entity knows the IDs of the entities within its sensing range[1].

We say that an entity v has a *higher priority* than another entity u, iff $\varphi(v) < \varphi(u) \ \vee (\varphi(v) = \varphi(u) \wedge v.ID < u.ID)$. Thus, we have a straight total order on the entities. If two entities prohibit each others movement, the entity with higher priority is allowed to move.

First we show that each entity can move eithter staight into the direction of the target or on the trajectory around the target. Therefore, the potential of an entity never increases.

Lemma 1. *Each entity v, which is not on the innermost trajetory, can move straight towards the target if v is not prohibited by another entity u with higher priority. Otherwise, v can move on the trajectory around the target in the corresponding heading direction. The potential of v never increases during the surrounding process.*

Proof. First we consider an entity v which is already on the innermost trajectory T_0. We show that v can move on that trajectory and its potential never increases. Since all entities on T_0 (if any) move in the same direction around the target, their distance to each other remains the same, and thus, they do not cause a

[1] For example, the entities within the sensing range of each other exchange their IDs through their communication interface.

collision. Another entity u with a higher potential $\varphi(u) > \varphi(v)$ is only allowed to move to T_0, if it does not cause a collision.

Now we consider an entity v which not in T_0. If v is not prohibited to move straight towards the target by any another entity u with higher priority then it moves towards the target and its potential strictly decreases. Otherwise, by similar argument than above, v can move on the trajectory corresponding to its current position and its potential does not change. □

Now we are able to prove a guarantee of the convergence of the surrounding process.

Theorem 1. *Until the inner trajectories have not been filled with robots (i.e. an inner trajectory T_{in} contains a hole of length at least $2d_{min}$), the overall potential energy of the swarm is strictly monotonically decreasing within $r_{om} \cdot \pi + 1$ steps, where r_{om} is the radius of the outermost trajectory T_{om}.*

Proof. Our rules guarantee that no robot increases its potential. If a robot is not prohibited by other robots, it is moving on a straight line towards the target, until it reaches the innermost trajectory. If a robot do not decreases its potential in a time step, then it is either on the innermost trajectory or it is prohibited to decrease the potential by another robot with higher priority.

Assume that in a time step no robot can decrease its potential. Let T_i be the innermost trajectory which contains a hole of length at least $2d_{min}$. Let r_i be the radius of T_i. Consider the robot v with the highest priority among the robots having strictly higher potential than the robots on T_i. If no such robot exists, then we are done, all inner trajectories are filled. Otherwise, v can only be prohibited to move straight to the direction of the target by robots on T_i. Then v circulates on trajectory T_{i+1}. The robot v and the hole on T_i circulate in opposite direction. Within $r_i\pi + 1$ steps either v can move into the hole on T_i or another robot filled the hole before v. Thus, within $r_i\pi + 1$ steps at least one robot strictly decrease its potential. □

5 Simulation Environment and Validation

In this section we will present our experimental results, which were made by the V-REP Virtual Robot Experimentation Platform, which is a 3D robot simulator with an integrated development environment.

We have created a compact 2D model that is easily extendable into a 3D model. Our virtual entities use infrared distance sensors, however a vision based solution is also usable for simultaneously scan the foreground and track the target. Although our entities are using local sensing only, it is possible to extend it with a communication layer for an extended model.

Besides the scanning of the neighborhood the entities should also track the target itself. Although we have made an assumption that all entities know the position of the target, we have tried to be more realistic in order to ease the integration of our simulation environment into a real implementation.

In Section 5, for simplicity, we have assumed a static target and we remarked that it can be changed for a slowly moving target. Then the unit velocity is relative to the velocity of the target. In our simulations we have a slowly moving target.

We compared our multi-orbit surronding algorithm (Algorithm 1) with another simple algorithm, which we call *baseline* algorithm (Algorithm 2), where the entities move to the left on the trajectory corresponding to its current position, if they are prohibited to move towards the target.

Algorithm 1 Multi-Orbit Surrounding

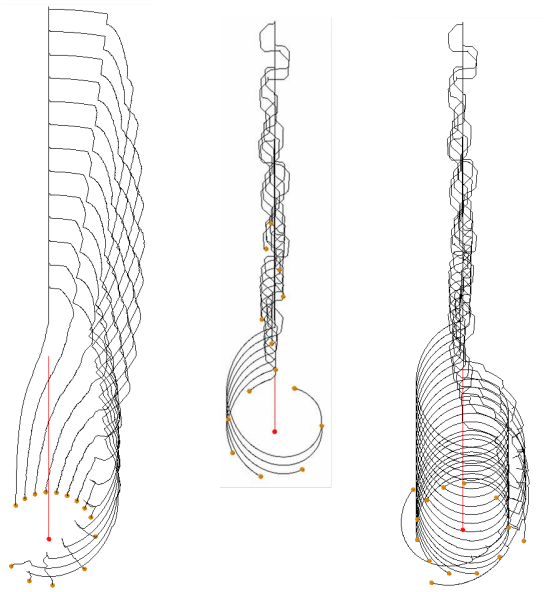
```
loop
  if current position is on T_0 or a neighbor with higher priority prohibits to move
  towards the target then
    Circulate in the corresponding direction around the target
  else
    Homing to the target
  end if
end loop
```

Algorithm 2 Baseline sorrounding

```
loop
  if current position is on T_0 or a neighbor with higher priority prohibits to move
  towards the target then
    move to left
  else
    move forward until the desired trajectory around the target is reached
  end if
end loop
```

In the first simulation scenario the entities are forming a row behind the target which is moving to the opposite direction. Fig. 3 shows the trajectory of the entities during the surrounding process until the the stationary state, where the swarm is circulating around the target. If the entities use an unidirectional surrounding behavior like Algorithm 2, which can be seen in the left image of Fig. 3, instead of our multi-orbit surrounding concept (middle and right image of Fig. 3), the surrounding takes longer.

In our second simulation scenario the start positions of the robots formed a block, i.e. they were distributed in a square area in a grid with d_{min} distance from the neighbors. At the beginning of the simulation many robots was prohibited to move towards the target. Our multi-orbit algorithm handeled this situation easily.



(a) baseline (b) multi-orbit (c) multi-orbit

Fig. 3. Simulated trajectories for the first test scenario, where 15 entities are surrounding a moving object.

In our third scenario the start positions of the robots were chosen randomly from the simulation area. In this scenario almost all robots were able to move straight towards the target until they reached the available innermost trajectory. The results of the simulations are summarized in Table 1.

	Line		Block		Random	
	baseline	multi-orbit	baseline	multi-orbit	baseline	multi-orbit
number of entities	15	15	15	15	15	15
entity velocity	5· v_{target}	5· v_{target}	5· v_{target}	5· v_{target}	5· v_{target}	5· v_{target}
enclosing time (min)	*3:15*	*1:00*	*1:40*	*0:56*	*0:21*	*0:20*

Table 1. Simulation results where 15 entities are surrounding a moving object.

6 Summary

We have defined the multi-orbit surronding problem, where autonomous robots with limited sensing range have to surround a target and circulate around it. We

have described a simple local algorithm to solve this problem. We have proved the correctness and the convergence of our solution and provided simulation results.

References

1. Balch, T., Arkin, R.: Behavior-based formation control for multirobot teams. IEEE Transactions on Robotics and Automation 14(6), 926 –939 (Dec 1998)
2. Barnes, L., Fields, M., Valavanis, K.: Swarm formation control utilizing elliptical surfaces and limiting functions. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39(6), 1434 –1445 (dec 2009)
3. Bürkle, A., Segor, F., Kollmann, M.: Towards autonomous micro uav swarms. Journal of Intelligent & Robotic Systems 61, 339–353 (2011), http://dx.doi.org/10.1007/s10846-010-9492-x, 10.1007/s10846-010-9492-x
4. Chu, T., Wang, L., Chen, T.: Self-organized motion in anisotropic swarms. Journal of Control Theory and Applications 1, 77–81 (2003), http://dx.doi.org/10.1007/s11768-003-0012-4, 10.1007/s11768-003-0012-4
5. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithm in asynchronous robot systems. SIAM J. Comput. 34(6), 1516–1528 (Jun 2005), http://dx.doi.org/10.1137/S0097539704446475
6. Cord-Landwehr, A., Degener, B., Fischer, M., Hüllmann, M., Kempkes, B., Klaas, A., Kling, P., Kurras, S., Märtens, M., Meyer auf der Heide, F., Raupach, C., Swierkot, K., Warner, D., Weddemann, C., Wonisch, D.: A new approach for analyzing convergence algorithms for mobile robots. In: Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011). Lecture Notes in Computer Science, vol. 6756, pp. 650–661. Springer-Verlag, Heidelberg, Germany (Jul 2011)
7. Gazi, V., Passino, K.: Stability analysis of swarms. IEEE Transactions on Automatic Control 48(4), 692 – 697 (Apr 2003)
8. How, J., Bethke, B., Frank, A., Dale, D., Vian, J.: Real-time indoor autonomous vehicle test environment. Control Systems Magazine, IEEE 28(2), 51 –64 (Apr 2008)
9. Hsieh, M., Kumar, V.: Pattern generation with multiple robots. In: Proc. IEEE International Conference on Robotics and Automation (ICRA 2006). pp. 2442 – 2447 (may 2006)
10. Hsieh, M., Loizou, S., Kumar, V.: Stabilization of multiple robots on stable orbits via local sensing. In: Proc. IEEE International Conference on Robotics and Automation. pp. 2312 –2317 (april 2007)
11. Leonard, N., Fiorelli, E.: Virtual leaders, artificial potentials and coordinated control of groups. In: Proc. 40th IEEE Conference on Decision and Control. vol. 3, pp. 2968 –2973 (Aug 2001)
12. Mataric, M.J.: Designing and understanding adaptive group behavior. Adaptive Behavior 4, 51–80 (1995)
13. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph. 21, 25–34 (August 1987), http://doi.acm.org/10.1145/37402.37406