The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Decentralized Lattice Formation Control for Micro Robotic Swarms

Grigoris Lionis and Kostas J. Kyriakopoulos

Abstract— **In this paper we discuss an algorithm for distributed formation of a lattice structures using a team of mobile robots. Lattice structures can be used for solving task allocation problems, as well as for utilizing cooperative swarm like motion for the agents. We are specifically dealing with the problem of constructing a lattice decentralized with the agents having only a limited sensing radius and a possibly imperfect communication channel that make "agree and go" methods unsuitable for solving the problem. The proposed algorithm is based on reducing the 2D problem to an 1D problem, i.e. moving on a curve that connects all the lattice points without intersecting itself. The algorithm is decentralized and is exact in the sense that all agents will converge to the lattice structure.**

I. INTRODUCTION

The nature of micro robotic systems is directly responsible for the new problems that we face with such systems, compared to traditional large scale robotic systems. Micro robots need special algorithms that use little power, little computations, little communications etc, while, the motion of most micro robots is even more constrained than the motion of a traditional unicycle. Moreover, another set of problems encountered when dealing with micro robots is the large scale problems arising from the large numbers of micro robot necessary, typically found in many envisioned micro robot applications.

In the recent literature, a great deal of attention has been given to the control of a large number of agents, with various different ways of looking at the problem. To name a few different approaches, in [4], [2] the authors study the stability of robot formations, while in [9] the authors study the controllability of leader-follower formations. The feasibility of formations has been studied in a number of works, for example [7]. Other issues related with the control of large number of agents have also been examined in the literature. For example in [1] the aggregation problem is studied, while in [3] the dispersion problem is studied.

In this paper we focus on a slightly different problem, on the problem of constructing a formation when the agents do not know a priori their part in the formation. We will focus on a lattice formation. The automated construction of a formation is a challenging problem, as both an allocation problem and a motion problem have to be solved. What makes the problem more challenging is the limited resources available to the micro robots. The agents do not have the computational/communication capabilities for first solving the allocation problem and then start moving, but the problems have to be solved concurrently with minimal computations. The choice of a rectangular lattice structure is not a random one, but it is dictated by the applications, as in a grid formation the topology is easily quantified and a number of problems regarding communications etc can be easily stated and solved. To summarize, we are presenting a decentralized algorithm that converges the agents to a rectangular grid structure. This work is heavily built on our previous work [6], where the proposed algorithm of helical target assignment was first presented. In our current work we provide explicit bounds for the completion of the task as well as a formalization of the algorithm.

Similar problems are also treated in the literature. In [10] the authors study a generalized version of the problem we are dealing with, but without considering collision avoidance in their scheme, while in our algorithm collision avoidance is included by construction. In [8] the authors study a similar problem focusing mainly on complexity issues. Their proposed solution to the concurrent solution of the task assignment problem and to the motion problem resembles closely to our own. We include collision avoidance to the design of the controller, and we use a helical ordering of the targets, as opposed to a circular ordering. We focus on the design of control laws that meet by design the collision avoidance principle. The addition of collision avoidance in an existing control scheme is not always feasible, as, especially in dense environments, collision avoidance maneuvers can interfere with the higher level controller and break down the controller. Finally, our controller is specifically constructed with micro agents in mind, and as a result limited computational and memory requirements are necessary.

II. PROBLEM FORMULATION

We are given a set of N mobile agents, randomly dispersed over the workspace $\mathscr{W}$. The workspace is assumed to be 2D and the agents are considered to be rigid objects on the plane $\mathscr{W}$. The state of agent i is described by

$$q_i = [x_i \; y_i \; \theta_i]^T$$

where (x_i, y_i) are the coordinates of the center of the agent, and θ_i is the orientation of the agent We denote with $p_i = [x_i \; y_i]^T$ the position of agent i. The kinematics of a microrobotic agent could vary considerably, but we will stick to a model rich enough to allow us to solve the problem, but

This work is partially supported by the European Commission through contract FP6 IST 2002 507006 ISWARM: Intelligent Small World Autonomous Robots for Micro-Manipulation, and by the Greek State Scholarship Foundation, IKY

Grigoris Lionis is a PhD Student in the School of Mechanical Engineering, National Technical University of Athens, 15700 Zografou, Greece glion@mail.ntua.gr

Kostas J. Kyriakopoulos is Faculty of School of Mechanical Engineering, National Technical University of Athens, 15700 Zografou, Greece kkyria@mail.ntua.gr

close enough to the kinematics of a micro robot. We assume a micro robot that has 3 modes of motion,

- Mode 1: Move forward
- Mode 2: Rotate Right
- Mode 3: Rotate Left

This kinematic setup is a strict subset of the motion envelope of a unicycle or an omnidirectional vehicle, and can also be approximated with a -theoretically- arbitrary precision by the kinematic structure of a class of micro robots that move on discrete curvature sets [5].

We assume that the radius of the bounding circle of the robots is known, and is equal to r_b, where the bounding circle is defined as the smallest circle that fully encircles the robot. A collision between two robots is -for simplicity- perceived to happen when the distance between the robots is less than twice the bounding distance, i.e. robots i, j collide when

$$d(q_i, q_j) = (x_i - x_j)^2 + (y_i - y_j)^2 < 4 \cdot r_b^2$$

where $d(q_i, q_j)$ is defined as the distance between agents i, j. We can now state the problem: The initial position of the N robots is random, within $\mathscr{W}$, with no collisions. (Figure 1)

$$d(q_i^{initial}, q_j^{initial}) \geq 4 \cdot r_b^2$$

The goal is to position all the agents on a grid like structure contained in $\mathscr{W}$ (Figure 2). The grid is defined as a set of points in $\mathscr{W}$ as following

$$G = \{(x,y) \in \mathscr{W} | x = x_c + i \cdot \beta, y = y_c + j \cdot \beta\}, i, j \in Z$$

where β is the grid size, and (x_c, y_c) is the center of the grid. The final position of the agents has to comply with the following rules

- $p_i \in G, \forall i$, i.e. all the agents to be on the grid
- $\theta_i = \theta_j, \forall i, j$, i.e. all agents share the same orientation
- $\forall i \ s.t. \ p_i \in G \ \exists j \ s.t. \ p_j = q \ \forall q \in G \ s.t. \ d(q, q_c) < d(p_i, q_c)$ where $p_c = (x_c, y_c)$ is the center of the grid and d denotes the Euclidean distance. If an agent is on a grid point, the all grid points closer to the center of the grid must also be occupied, i.e. the grid must be completely filled. This is a crucial requirement, as the grid structure will be used to transmit messages in the swarm, and as a result a void in the grid undermines the message propagation capabilities.

Moreover, the transition to the final positions has to obey the kinematics of the agents and the non-collision requirements.

Obviously, the grid size has to be large enough to accommodate the agents without collisions. So, it must hold that

$$\beta \geq 2 \cdot r_b$$

A. *Communication and Sensing*

The decentralized solution of this problem clearly necessitates some sort of intra agent communication, either explicit or implicit. We will assume that the agents are equipped with both communications schemes, and more precisely , with a communication system capable of transmitting messages within a communication radius r_m and with a sensing subsystem capable of identifying other robots and their position within a sensing radius r_s. Moreover, to model aspects of the reality, we will assume that the communication system is not flawless, but rather, that there is a possibility of dropping a message during a transition. On the other hand, the sensing system is -in reality- much more robust, as it only has to sense the other robots as well as their position. We shall assume that $r_s > \beta$, that is that grid to be formed is inside the sensing radius of the robots.

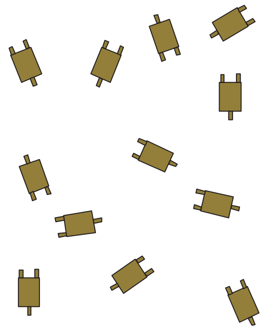
Fig. 1. Initial Random Configuration of a number of agents.

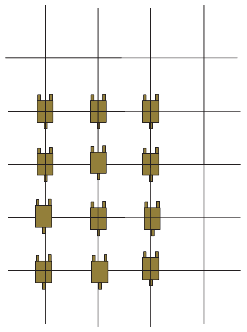
Fig. 2. Final Grid Structured Configuration.

III. Think and Go approach

An obvious approach to solving the aforemention problem decentralized would be the "think and go" approach, in which the robots first start exchanging messages as to where to go and then, after the messages exchange has been completed, the robots start moving in a decentralized way to reach the goals that have been assigned to them via cooperation. This approach, would use in essence, a decentralized agrement protocol in which the position of the agents would be agreed upon via messages exchanges. We opted not to use this approach for two main reasons

- There is no guarantee that the swarm is fully connected. Given that the number of agents N is not known a priori it is difficult to ensure that the final system will be singly connected, and moreover, the algorithms that accomplish this might result in redundant motion for the robots.
- The inherit difficulties in communicating complex signals between the robots. As the communication is not flawless, the transmission of a large number of messages is not straightforward and it is difficult to bound the probabilities of erroneous message propagation.
- Finally, the cost of transmitting messages is -sometimes-large, and increases disproportionably with the number of the agents.

Finally, the general problem of moving the agents inside $\mathscr{W}$ has a complexity that sharply increases with the number of the agents, especially when the agents move completely independently, in the sense that the destinations of the agents, and hence the trajectories, are completely independent. We have a somewhat different problem to resolve, as we are concurrently trying not only to move the agents, but also to allocate them in appropriate positions. The additional requirement - allocate the robots to the final positions- can be used as an advantage. In particular, we will solve the allocation problem taking into account -with an implicit but immediate way- the trajectories towards the targets, i.e. we will solve both these problems concurrently and the complexity issues raised above for the case of independent motion of the agents, will no longer hold in our case.

IV. Proposed Solution

The proposed solution is intuitively simple. We transform the 2D problem into a 1D problem, by constructing a path on the workspace connecting all the target points, in a way that each point is crossed only once, and with the final point being the center of the grid. Our proposed path is depicted in Figure 3. In a sense, we create a potential function on the workspace that guides the robots to take over all the points possible, respecting the constraints specifying that voids must be be avoided, and avoiding all collisions and deadlocks between the robots.

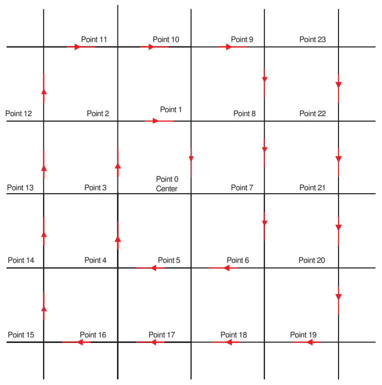

Fig. 3. Grid Point Numbering. The red arrows correspond to the motion of an agent while it goes to grid points of lower number.

A. Grid Point Hierarchy Construction

We shall examine in detail how to construct the numbering scheme on the grid, a scheme that is more than a simple numbering, as it defines a hierarchy of grid point, in the sense which must be filled first. Moreover, the careful construction of this hierarchy is essential for the correctness of the algorithm. Finally, as we shall discuss in more detail, the same technique can be extended to cover other shapes/collection of points, but the automatic construction of the appropriate grid point hierarchies is not straightforward in the general case.

The numbering of the grid points is straightforward. We introduce a matrix marking the grid points, matrix G, where $G_{ij}=0$ corresponds to the case where grid point (i,j) has not been numbered, while otherwise $G_{ij}=n$ corresponds to the case where grid point (i,j) is the n-th grid point. We allow the coordinates of G to attain negative values as well[1] We start from the center of the grids, numbered 0. We move upwards, for one grid point and we number the point as point 1.We proceed left for one grid point, and label the grid point, point 2. We start moving downwards for two grid points, and we label the two grid points 3 and 4 respectively. A finite state machine description of the algorithm is depicted in Fig. The algorithm has 4 internal states :

- Up : Moving up the grid, and successively numbering grid points, as long as the grid point on the left is already numbered.
- Left : Moving left on the grid, and successively numbering grid points, as long as the grid point below is already numbered.
- Down : Moving down on the grid and successively numbering grid points, as long as the grid point right is already numbered.
- Right : Moving right on the grid and numbering grid points, as long as the grid point above is already numbered.

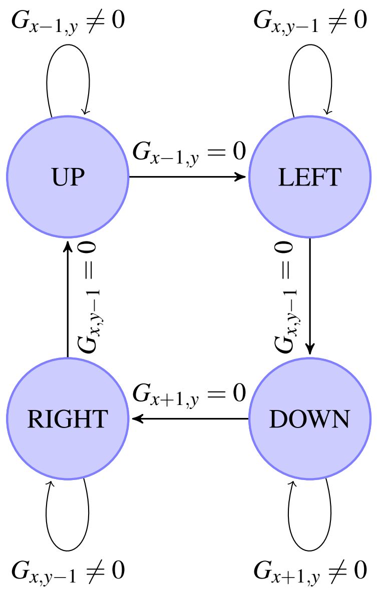

Fig. 4. Grid Point Numbering Algorithm. Starting from the zero grid point, corresponding to the center of the grid, this algorithm successively numbers ALL the grid points moving up, left, down, right and so on. The X and Y correspond to the integer coordinates on the grid.

This algorithm will number ALL the grid points on the grid, and will assign all the integers necessary to these grid points, as the algorithm ”moves” continually on the boundary of the part of the grid already numbered. No grid points are left in the internal of the numbered points. Furthermore, after this construction, one can easily establish the inverse function, from the coordinates of the grid point, to the grid point number.

B. Main Control Algorithm

After constructing the grid hierarchy, we are in position to state the main algorithm, which is intuitively extremely simple. Each agent hops from a grid point with index i to the grid point with index $i-1$, provided that this grid

[1]Obviously, a trivial remedy is available by defining G^* $s.t.G^*_{ij}=G_{i+m,j+n}$ where m is the minimum (negative) value of i and n of j.

TABLE I

GRID NUMBERING ALGORITHM STATES

State	Action
UP	$y \mapsto y+1$
	$N \mapsto N+1$
	$G_{xy} \mapsto N$
DOWN	$y \mapsto y-1$
	$N \mapsto N+1$
	$G_{xy} \mapsto N$
LEFT	$x \mapsto x-1$
	$N \mapsto N+1$
	$G_{xy} \mapsto N$
RIGHT	$x \mapsto x+1$
	$N \mapsto N+1$
	$G_{xy} \mapsto N$

point, with index $i-1$ is free. This extremely simply control primitive results in a complete coverage of the grid, assuring no collisions and no deadlocks and no livelocks.

- Complete coverage : It is easy to prove that this algorithm has only one stable point, a point where the grid points are covered, in the sense given above. Assuming that the system has stopped. This means that for all agents, the grid position with a lower number is taken over by another agent. Therefore, all grid positions with a number less than or equal to the maximal grid position occupied are also occupied. So the grid is covered.
- No collisions: In all moves, the agents trajectories do not intersect by construction, and therefore no collisions are possible.
- No Livelocks : To show that no livelocks are possible, it suffices to see that the sum of the numbers taken by the agents cannot increase, but only decrease. Therefore, it is not possible to return to a previous state of the system, hence no livelocks are possible.

Algorithm 1 Descent Grid

```
$T \Leftarrow 0$
$N \Leftarrow$ GetCurrentGridPoint
while $N > 1$ AND $T \leq N$ do
    $N \Leftarrow$ GetCurrentGridPoint
    if $N-1$ is free then
        MoveRobot($N-1$)
        $T \Leftarrow 0$
    else
        $T \Leftarrow T+1$
        Wait($T_{tr}$)
    end if
end while
EXIT
```

C. Algorithm Initialization

The algorithm described in the previous section is an algorithm guiding the agents towards the requested configuration, given that the agents already are located on the grid. It is therefore necessary to construct an initialization step that can guide all agents onto arbitrary grid points. A key idea to accomplish this task is to construct an one to one correspondence between the position of the agents and the grid points, i.e. to construct a decentralized grid point allocation function that assigns to each grid point one and only one agent. To do this Fig. it suffices for the agents to have a minimum distance from each other greater than $\sqrt{2}\beta$. If the agents have this minimum intra-agent distance, it is not possible for two agents to have the same proximal grid point, but rather, the agents will necessarily have different proximal grid points.

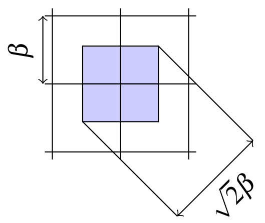

Fig. 5. Part of the workspace nearest to a specific Grid Point. The diameter of this area is $\sqrt{2}\beta$.

1) Agent Dispersion: Of course, as we stated in the formulation of the problem, the initial configuration of the agents is more or less random, and we cannot assume that the condition of a minimal intra agent distance is automatically satisfied. Rather than this, as a first step, we shall force the agents to be repelled from one another as to ensure that their intra agent distance becomes larger than $\Delta = \sqrt{2}\beta + \varepsilon$ To accomplish this subtask we employ the controller found in [3]. This controller works completely decentralized, with the sensing requirements to be limited only to a knowledge, for each agent, of the surrounding agents and it can, therefore be applied to our case. The controller stabilizes the agents from any initial configuration to a configuration where the intra-agent distance is bigger than a set distance. So, for our case, it suffices to set the desired intra-agent distance to Δ. Obviously, for this controller to work, the following condition must be met $\Delta \leq r_s$ that is the sensing distance must be larger than the dispersion distance. This condition trivially satisfies the condition of the grid size being less than the sensing distance, a condition necessary for the grid controller.

D. Complete Algorithm

We can now summarize the structure of the complete controller, and it is graphically depicted in Fig. First the dispersion controller is used, to ensure that the agents are properly dispersed inside the working area. The second step of the overall controller is for each agent to go to the nearest grid point. Obviously, as the intra-agent distance is larger than the diameter of the basin of attraction of each grid point, it is not possible for two agents to have the same grid point as a target, nor it is possible for two agents to cross, as the trajectories of the agents are straight lines between the initial position and the nearest grid point, and these trajectories are completely included inside the basin of attraction of each grid point. Finally, the control law that forces the agents to descent the grid is used, and this guarantees that the agents will form a grid with the requested properties.

E. Time Bounds

An important question is the question of how much time is necessary for the whole process to converge. First, it is important to notice that as the overall control law is not continuous, it can -in principle- have a finite convergence time. Moreover, as the whole idea behind this controller is to from a grid in order to do other more useful stuff, it is necessary to bound the time necessary for the convergence of the controller. We shall only give bounds to the two final parts of the overall control strategy, as the first part is taken out of the literature, and is guaranteed to work in finite time.

We shall denote as T_2 the time necessary for the the agents to complete the second part of the control scheme. This can be broken down as following

$$T_2 \leq T_2^{max} = \underbrace{\frac{\pi}{\omega}}_{\text{Orientation}} + \underbrace{\frac{\sqrt{2}/2\beta}{u}}_{\text{Motion}}$$

The first term corresponds to the time necessary for the agents to reorient towards the goal, while the second term corresponds to the time necessary for moving towards the goal. The orientation time is bounded by the time necessary for an agent to turn for half a circle, as by choosing the best way of turning the agent needs less than π rotation to reorient to any direction. The motion time is bounded by the time necessary to move from the point of the basin of attraction of the grid point farthest away from the grid point, which is equal to half the diameter of the basin of attraction of the grid point. So the time necessary for completing this part of the controller is bounded. Obviously, this part of the controller will remain active for at least T_2^{max} to ensure that all agents have completed their rotation.

We denote as T_3 the time necessary for all agents to traverse through the grid, i.e. the time necessary for the completion of the grid, or equivalently the time necessary for the system to converge. We shall assume -for simplicity in the derivation of these bounds- that all agents move synchronously, i.e. that all agents have a global clock that allows them to synchronize their motion. We have to note that this assumption is only used to establish an appropriate bound on the necessary time for the convergence and has nothing to do with the convergence properties of the scheme. We merely use the well known fact that synchronous systems are significantly easier to analyze than asynchronous systems. So we shall assume that if agent i is located in grid point k while agent j is located in grid point $k-1$ while grid point $k-2$ is free, then first agent j starts moving towards grid point $k-2$ while agent i remains still. While during the motion of agent j agent i "sees" that grid point $k-1$ is free, even though agent j has not reached yet grid point $k-2$, nevertheless agent i does not move, as it waits for the next move window. If we also assume that grid point $k-3$ is also free, then agents j and i will start moving concurrently towards grid points $k-3$ and $k-1$ respectively.

As discussed above, all grid points are numbered. We denote as function $g : N^+ \to N^+$ the function that gives the grid point associated with each agent, for a given time step. Then, we can introduce quantity S with $S = \sum_i g(i)$ that is S is the sum of the number of all grid points on which there is an agent. This function will allow us to bound the time necessary for completing step 3. Assume that we have N agents and that the agents furthest away from the center is on grid point M. Then we have that $S^{initial} \leq M + (M-1) + (M-2) + ...(M-N+1)$ since the maximum value that S can have corresponds to the case where all agents are stack to the maximal values of the grid. We also have that $S^{final} = 0+1+2+...+(N-1)$ which corresponds to the case where all agents are in their final positions, and therefore all grid positions from 0 to $N-1$ are taken. So we have that

$$\Delta S \leq \underbrace{N \cdot M - (1+2+...+(N-1)}_{S_{max}^{initial}} - \underbrace{(1+2+...+(N-1)}_{S^{final}}$$
$$= N \cdot M - N \cdot (N+1) = N \cdot (M-N-1)$$

Given that in each time step the system reduces S for at least one -at least one agent moves from one position to another- we have that the maximal number of time steps possible is equal to ΔS. The time necessary for each time step is given, using the same line of reasoning used for establishing T_2 as $\delta t = \frac{\pi}{\omega} + \frac{\beta}{u}$ and the time necessary for completing the third step of the control scheme is equal to

$$T_3 \leq \Delta S \cdot \delta t = N \cdot (M-N-1) \cdot \delta t$$

thus proving that this algorithm is $\mathcal{O}(MN)$. This is an upper bound, based on the simplifying assumption that at every time instant only a single agent moves. In reality, even in the extreme case where the agents start moving stacked on the grid points with higher index, the agents will move concurrently and the actual running time will be less. It is a subject of current research to quantify more precisely the complexity of the scheme. M is trivially bounded by the size of $\mathscr{W}$, although the exact connection between them is not trivial.

V. Simulations

In order to depict the overall controller behavior, as well as to verify via simulations that the controller is accurate and correct we present a simulated run of the controller, resulting in the automatic, decentralized construction of a grid. The simulation concerns a flock of 30 robots, that converge to a structure grid around the origin. As this simulation is concerned with the inner workings of the controller, we use circular agents, to focus on the switching controller and not on the details of how the agents move. The simulated run is depicted in figures 6 to 13. In Figure 6, the beginning of the simulation is shown. The agents are dispersed in the work area and the controller starts working, while Figure 13 represents the final structured grid. The intermediate figures (7,8,9,10,11,12) depict various instances of a semi constructed grid, with the agents still moving.

VI. Conclusion

We have presented a grid formation control scheme for a swarm of robots. Grids, and in general structured formations, are of major importance in swarm robotics, as several issues regarding coordination, communication etc can be solved much easier on grids. In this paper we focus on the "preproblem" of how to construct a grid using a decentralized controller with minimal communication between the agents -in a sense only implicit communication is exchanged- This problem is basic when some properties have to be ensured in a formation. Moreover, this problem is a special case of a wider class of problems where a set of positions has to be covered by a set of agents. The presented algorithm can be used to fulfil these tasks. The algorithm has a simple structure as it has been designed for use in microrobots with minimal resources. Finally, the presented algorithm is guaranteed to work in bounded time.

The proposed scheme is completely decentralized apart from the initial knowledge of where to form a grid of specific size (i.e. the center point of the grid and the grid size), and for this paradigm to work, the agents have to be fully aware of their position (i.e. equipped with a GPS-like system) An interesting extension would be the drop of these two working assumptions (i.e. knowledge of where to form and knowledge of what to form) possibly using some kind of communication scheme for agreeing on these data decentralized.

Further more, we would like to extend this work towards different directions. First, an important problem with this algorithm lies within its cooperative nature : All agents are assumed to work flawlessly, and in this case the robustness of the algorithm is fulfilled. What would happen in a different case. Another important question would be to address a possibly heterogeneous swarm of robots. How would such an assignment algorithm work in that case? Finally we would like to extend this algorithm to cover more general structures, and to possibly find some criteria of what structures can be dealt with, using the proposed setup.

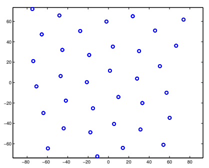
Fig. 6. T=0

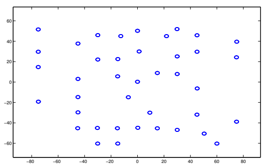
Fig. 7. T=15

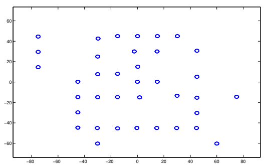
Fig. 8. T=35.

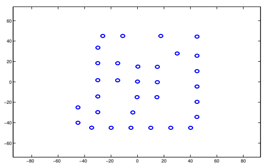
Fig. 9. T=55

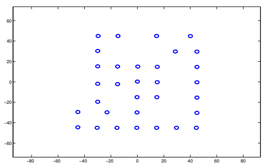
Fig. 10. T=75.

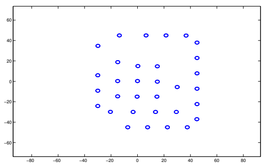
Fig. 11. T=95

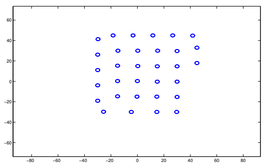
Fig. 12. T=115.

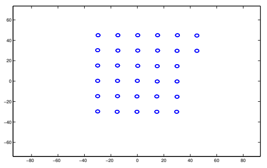
Fig. 13. T=200

References

[1] Dimos Dimarogonas and Kostas Kyriakopoulos. Connectedness preserving distributed swarm aggregation for multiple kinematic robots. *IEEE Transactions on Robotics*, 24(5):1213–1223, 2008.

[2] Dimos V. Dimarogonas and Karl H. Johansson. On the stability of distance-based formation control. *Proccedings of the 2008 Conference on Decision and Control*, pages 1200–1205, 2008.

[3] D.V. Dimarogonas and K.J. Kyriakopoulos. Inverse agreement algorithms with application to swarm dispersion for multiple nonholonomic agents. *Proccedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 367–393, 2008.

[4] A. Jadbabaie H.G. Tanner and G. J. Pappas. Stable flocking of mobile agents. *Proccedings of the 2003 IEEE Conference on Decision and Control*, pages 2010–2021, 2003.

[5] Grigoris Lionis and Kostas Kyriakopoulos. Pwm control for a microrobot moving on a discrete curvature trajectory set. *Proccedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2324–2329, 2007.

[6] Grigoris Lionis and Kostas J. Kyriakopoulos. Iswarm D6.3 Technical Report. Internal Project Puplication, March 2007.

[7] G. J. Pappas P. Tabuada and P. Lima. Motion feasibility of multi-agent formations. *IEEE Transactions on Robotics*, 21:387–392, 2003.

[8] Stephen L. Smith and Francesco Bullo. Target assignment for robotic networks:asymptotic performance under limited communication. *Proccedings of the American Control Conference*, pages 1155–1160, 2007.

[9] Herbert Tanner. On the controllability of nearest neighbor interconnections. *Proccedings of the 2004 Conference on Decision and Control*, pages 2010–2015, 2004.

[10] M. M. Zavlanos and G. J. Pappas. Distributed formation control with permutation symmetries. *Proccedings of the 2007 IEEE Conference on Decision and Control*, pages 2894–2899, 2007.