2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

No Robot Left Behind: Coordination to Overcome Local Minima in Swarm Navigation

Leandro Soriano Marcolino and Luiz Chaimowicz.

Abstract— **In this paper, we address navigation and coordination methods that allow swarms of robots to converge and spread along complex 2D shapes in environments containing unknown obstacles. Shapes are modeled using implicit functions and a gradient descent approach is used for controlling the swarm. To overcome local minima, that may appear in these scenarios, we use a coordination mechanism that reallocates some robots as "rescuers" and sends them to help other robots that may be trapped. Simulations and real experiments demonstrate the feasibility of the proposed approach.**

I. Introduction

The use of large groups of robots in the execution of complex tasks has received much attention in recent years. Generally called *swarms*, these systems employ a large number of simpler agents to perform different types of tasks, oftentimes inspired by their biological counterparts. In general, swarms of robots must perform without a designated leader and using limited communication. Due to these challenges, new algorithms to control and coordinate these very large groups of robots have been developed.

In this paper, we present navigation and coordination methods that allow swarms of robots to converge and spread along complex 2D shapes in environments containing obstacles. We build on our previous work in which implicit functions and gradient descent techniques were used to synthesize shapes and patterns in obstacle free environments [1]. Here, along with gradient descent, robots are repelled by locally sensed obstacles using a potential field approach. As expected, this can lead to the appearance of local minima compromising the convergence. To overcome this, we rely on multi-robot coordination: some robots become rescuers and retrace their paths to help others stuck in local minima.

The general area of motion planning for large groups of robots has been very active in the last few years. One of the first works to deal with the motion control of a large number of agents was proposed for generating realistic computer animations of flocks of birds (called *boids*) [2]. Basically, local interactions among neighboring agents create an emergent behavior for the whole flock. In robotics, these interactions can be considered as a special case of the potential field approach [3], in which robots are attracted by the goal and repelled by obstacles and other robots. In swarms, attractive forces are generally modeled through the gradient descent of specific functions [4], [5]. Unfortunately, as in regular potential field approaches, the presence of obstacles and local repulsion forces among the robots may cause convergence problems in general gradient descent approaches, mainly when robots are required to synthesize shapes. Hsieh and Kumar [6] are able to prove convergence properties and the absence of local minima for specific types of shapes and environments. Also, special types of navigation functions can be used to navigate swarms in cluttered environments [7]. But these approaches may be hard to compute in real time and may not be applicable to all types of environments.

Other approaches to navigate large groups in obstructed environments consist in treating the swarm as a simpler entity with a smaller number of degrees of freedom and then perform the motion planning for this entity. The work presented in [8], for example, models a group with a deformable shape and uses a *Probabilistic Roadmap* to plan for this shape. Belta et al. [9] show how groups of robots can be modeled as deformable ellipses, and presented decentralized controllers that allowed the control of the shape and position of the ellipses. This approach was extended in [10] with the development of a hierarchical framework for trajectory planning and control of swarms. A hierarchical approach was also used in [11] in which planning was simplified by dynamically grouping robots using a sphere tree structure. A related work, that investigates coordination mechanisms for boundary coverage with swarms is presented in [12].

In this paper, instead of restricting our environment, developing complex controllers, planners, and navigation functions, or relying on random movements to escape local minima, we use the composition of simple controllers and decentralized coordination to allow swarms of robots to navigate and synthesize patterns, overcoming local minima in environments containing unknown obstacles. This is the main contribution of this work.

This paper is organized as follows. Next section presents the approach used to generate the implicit functions, the controllers used to navigate the swarm, and an example of a complex shape synthesized in an environment containing obstacles. Section III explains the coordination methodology used to overcome local minima and shows simulations of a local minima scenario. Section IV presents some experiments performed with a couple of *e-puck* robots that demonstrate the feasibility of the proposed approach. Finally, Section V brings the conclusion and directions for future work.

The authors are with the Vision and Robotics Laboratory (VeRLab), Computer Science Department, Federal University of Minas Gerais, Brazil. *e-mails:* {*soriano,chaimo*}*@dcc.ufmg.br*

This work is partially supported by PRPq–UFMG, Fapemig (CEX-1097/05) and CNPq (305127/2005-5, 504915/2007-0). The authors would like to thank Renato Garcia and Pedro Shiroma for their help with the localization system.

II. Synthesizing Shapes

As stated in the previous section, we want to control a very large group of robots to converge and spread along complex shapes in environments containing obstacles. As in [1], robots spread along a 2D curve S given by implicit functions of the form $s(x,y)=0$. This implicit function can be viewed as the zero isocontour of a 3D surface $f=s(x,y)$ whose value is less than zero for all points (x,y) that are inside the S boundary and is greater than zero for all points outside the S boundary. By controlling individual robots to perform a gradient descent on f^2, we are able to make the group converge to S. In order to compute the gradient forces, robots must know their global position. This is a strong assumption, but it is generally accepted when dealing with swarm navigation. Besides, advances in localization technologies have been providing affordable and scalable ways to localize large number of robots (see Section IV-A).

To synthesize specific shapes, we consider f as a weighted sum of radial basis functions (RBFs) created interpolating from a set of constraint points. We specify some constraint points $\mathbf{p}_j$ along the shape boundary such that $f(\mathbf{p}_j)=0$ and at least one constraint smaller than zero inside the boundary (to avoid degenerate solutions). Each of these constraints will be the center of one RBF. Then, solving a simple linear system, we determine the weights (w_j) of all the RBFs that comprise the function f:

$$f(\mathbf{q})=\sum_j w_j h(|\mathbf{q}-\mathbf{p}_j|) \qquad (1)$$

where the term $|\mathbf{q}-\mathbf{p}_j|$ is the Euclidean Distance (d) between the point where the function is being evaluated and a constraint $\mathbf{p}_j$. In this paper function h is given by $h(d)=d^2\log(d)$.

For obstacle avoidance, we use a regular potential field approach: if an obstacle is detected by the robot, this obstacle applies a repulsive force that is inversely proportional to the distance between them. The same approach is used for modeling repulsion between robots. In simulation, we consider that the robot's sensing region is determined by a circle with radius r centered at the robot. So, we define two sets O_i and N_i containing, respectively, the obstacles and robots detected inside this region.

Thus, considering a fully actuated robot i with dynamic model given by $\dot{\mathbf{q}}_i=\mathbf{v}_i$ and $\dot{\mathbf{v}}_i=\mathbf{u}_i$, where $\mathbf{q}_i=[x_i,y_i]^T$ is the configuration of robot i, $\mathbf{u}_i$ is its control input and $\mathbf{v}_i$ is the velocity vector, the control law is given by:

$$\mathbf{u}_i=-\alpha\nabla f^2(\mathbf{q}_i)-C\dot{\mathbf{q}}_i-\beta\sum_{k\in O_i}\frac{1}{\mathbf{d}_{ik}}-\gamma\sum_{j\in N_i}\frac{1}{\mathbf{q}_j-\mathbf{q}_i}. \qquad (2)$$

Constants α, β, γ, and C are positive. The first term is the gradient of the square of the RBF function given by equation 1, used to guide the robots toward the specified shape. The second term is a damping force. The third term is the sum of repulsive forces applied by the obstacles ($\mathbf{d}_{ik}$ is the distance vector between robot i and obstacle k). Only the obstacles that are inside robot i sensing region, represented by the set O_i, are considered in the computation of forces. The fourth term computes the repulsive interaction of a robot with its neighbors, represented by the set N_i.

To illustrate this approach, we performed simulations in which a swarm should generate a pattern (letter 'G') in an environment containing obstacles. We used MuRoS, a multirobot simulator that allows us to implement various tasks, test different controllers, and observe the robots in real time. Constants α, β and γ were tuned to balance attractive and repulsive forces adequately and velocities were limited to an acceptable maximum (saturation). The sensing radius r is set to 3 times the size of the robot. Figure 1 shows a sequence of snapshots of robots converging to the target.

III. Overcoming Local Minima

The approach presented in the previous section can lead to situations of local minima. Since robots are attracted by the goal and repelled by obstacles and other robots, they can be trapped in regions where the resultant force is zero or where the force profile leads to repetitive movements (for example, continuous circular movements in a specific region). In general, the local minima regions depend on the shape of the obstacles and on the number of robots. So, it is difficult to model these regions precisely and there are no formal guarantees that the robots will converge to the desired pattern. To overcome this, we use coordination strategies that allow robots to escape from local minima with the help of their teammates. The general idea is to use some of the robots that reach the target as "rescuers". These rescuers will retrace their path looking for other robots that may be stuck in local minima. For this, two additional assumptions are made: a robot must have a small memory in order to save their path and must be able to send messages to the robots in its neighborhood. The amount of storage needed is not high, a few KBytes will suffice in most situations. Also, affordable short range communication mechanisms, such as Bluetooth, are commonly available nowadays. Thus, these assumptions do not compromise the applicability of our methodology.

The strategies developed take advantage of some characteristics of swarms in general: the presence of a large number of members and the possibility of local interactions among them. Since we have a very large group, it is possible to allocate different roles to some of the robots while the others perform the original mission. As an example, we have the rescuers to help other robots. Also, when a large number of robots are trapped, local force interactions allow robots to "push" some teammates out of the local minima and local communication allows rescuers to broadcast free paths to the neighboring robots trapped in regions of local minima. Finally, an important characteristic of using swarms is fault tolerance. The loss of some team members do not compromise the mission as a whole. So the algorithm does not need to be complete: we can have robots that may not be rescued and will be considered "missed in action".

Our coordination is based on a mode switching mechanism similar to the dynamic role assignment presented in [13]. A robot can switch between different modes (or roles) during

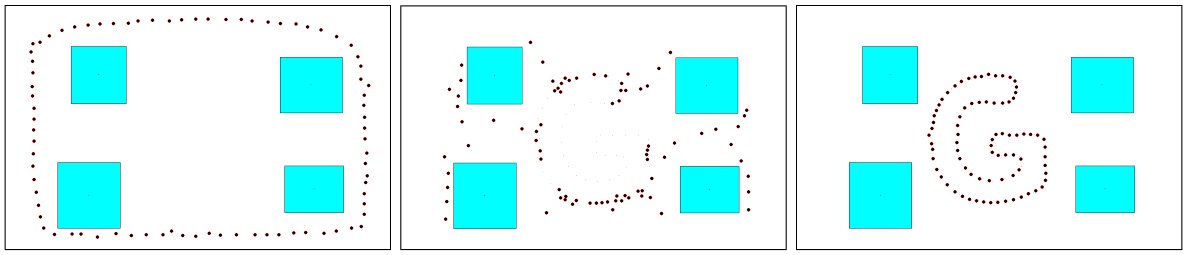

Fig. 1. A group of 80 robots synthesizing a complex shape (letter 'G') while avoiding obstacles. Robots are represented by the small circles.

the execution of the task. Each mode determines a different behavior for the robot and will be executed while certain internal and external conditions are satisfied. The mode switching together with local interactions allow robots to escape local minima and converge to the desired target.

A. Modes

Robots in the swarm can be in one of five different modes during task execution: *normal*, *trapped*, *rescuer*, *attached* and *completed*. These modes can be represented by a finite state machine (FSM), in which the edges represent the possible transitions between different modes. Figure 2 shows the finite state machine used in our coordination mechanism.

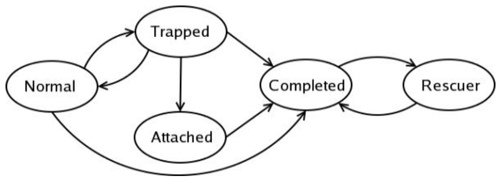

Fig. 2. Finite state machine showing the possible modes and transitions for each swarm member.

A *normal* robot simply behaves as explained in Section II. It performs a gradient descent, following paths that will avoid obstacles and eventually lead to the target. All robots start in the *normal* mode and become *trapped* if they fall in a local minima region. A *trapped* robot acts similarly to a *normal* one, except for the following facts: (i) a *trapped* robot strongly repels another *trapped* robot and this repulsion is stronger than the one between two *normal* robots. As a local minima region tends to attract many robots, the local interactions through these stronger repulsion forces will help some of the robots to escape this region; (ii) *trapped* robots accept messages from *rescuers* or *attached* robots that will help them to escape from local minima and move towards the target. This will be better explained later in this section.

To change its mode from *normal* to *trapped* (and vice-versa), a robot considers the variation of its position over time. If its position do not change much during a certain amount of time, it becomes *trapped*. Since robots can have small or repetitive movements in the local minima area, the transition back *trapped* to *normal* is harder. It only gets back to the *normal* mode with larger variations in its position.

When a robot arrives at the target it may become a *rescuer*. Basically, when moving towards the goal, a robot saves a sequence of waypoints that is used to mark its path. If it becomes a *rescuer* it will retrace its path backwards looking for *trapped* robots. After retracing its path backwards, the robot moves again to the goal following the path in the correct direction. In order to minimize the memory requirements of the algorithm, the robot discards redundant information in the path stored. Therefore, if there is a straight line in the path, ideally only two waypoints will be used. If there is a complicated and narrow curve, the robot will save more waypoints to be able to follow the path.

An important point is to define which and how many robots will become *rescuers*. To control this we use "coordination tokens" [1]. We start with n tokens at the target. Every robot that arrives removes one token. The first robot that does not find any token to remove becomes a *rescuer* and puts m new tokens at the target, with $m < n$. This inequality is important because we do not want the first robots to arrive to become rescuers since they probably started near the target and will not find many *trapped* robots in their path. This procedure is repeated with m tokens until a maximum number τ of *rescuers* are sent. The values of m, n and τ are determined empirically and may vary depending on the total number of robots and characteristics of the environment. With an appropriately large number of robots and correctly specified constants, generally we will have enough *rescuer* robots to achieve a good convergence rate.

A *trapped* robot keeps sending messages announcing its state. When a *rescuer* listens to one of these messages, thereby detecting a *trapped* robot in its neighborhood, it broadcasts its current position and its path. Any *trapped* robot will receive the message if it is within a certain distance from the *rescuer* and there is a direct line of sight between them. After receiving it, the *trapped* robot changes its mode to *attached*.

An *attached* robot will move to the received position and then follow the received path to the goal. An *attached* robot can also communicate with other *trapped* robots, spreading the information about the feasible path to the goal. In this situation, the *trapped* robots will change their status to *attached* and will be able to also spread the information to their neighbors, creating a powerful communication chain. To avoid congestion with many robots converging to the same waypoint, a circular area around the waypoint is considered.

[1]We borrow this term from the multi-agent community where tokens are used to transmit information between agents in a scalable way [14]. Here, tokens are basically distributed counters used to coordinate the robots.

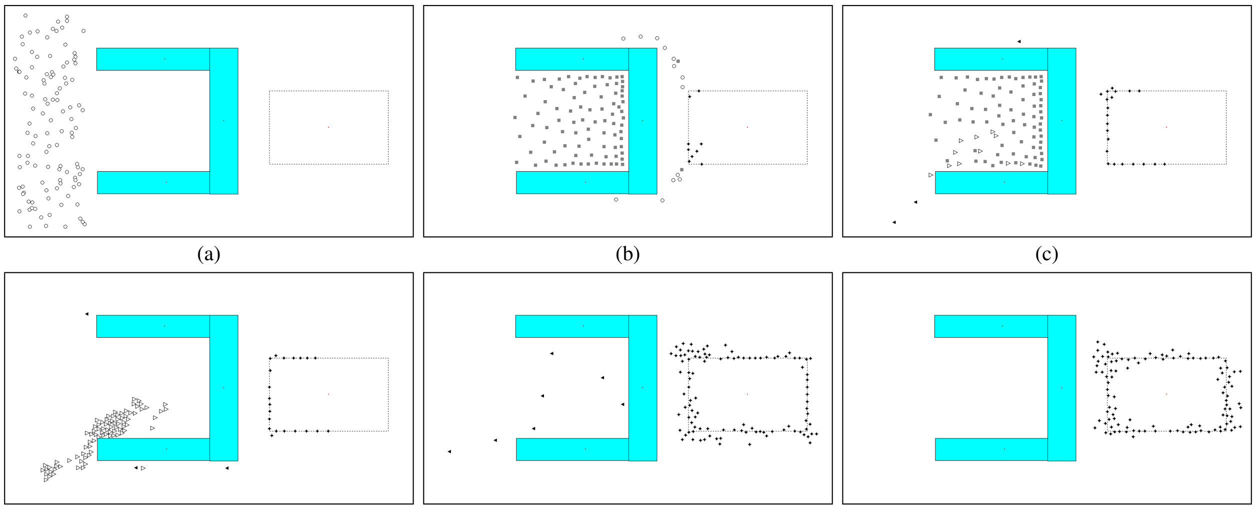
(a) (b) (c) (d) (e) (f)

Fig. 3. A swarm of 110 robots escaping from local minima and converging to the target. The robot shapes represent the different modes.

Finally, a robot will change its mode to *completed* when it reaches the target. In this case, it will spread along the zero isocontour of the implicit function as explained in section II. *Completed* robots will not switch to *trapped* again but may become *rescuers* according to the coordination tokens.

This idea of saving and communicating feasible paths to the target is inspired in the use of pheromones by social insects. A pheromone is any chemical or set of chemicals produced by a living organism that transmits a message to other members of the same species [15]. For example, certain ants leave a trail of pheromones as they return to the nest with food. This trail attracts other ants and serves as a guide, thereby creating an implicit form of communication. In this paper, a phromone inspired approach is implemented using explicit communication due to the difficulties of artificially creating and detecting real pheromones. Instead of being marked on the ground, feasible trails are saved by the robots and transmitted to teammates.

B. Classical Example

To demonstrate the coordination strategy, we modeled a classical local minima scenario: an u-shaped obstacle forming a dead end. We simulated 110 robots in this scenario and were able to successfully achieve convergence, with all robots escaping local minima and spreading along the target. The variables that control the number and frequency of rescuers are set to $n = 12$, $m = 4$, $\tau = 10$.

The initial state of the simulation can be seen in Figure 3(a). The robots are in the left (white circles), in the middle we have an u-shaped obstacle and in the right we have our target (dashed square). In Figure 3(b), some robots have already arrived at the target, but many robots are stuck in the region of local minima. It is interesting to note how the *trapped* robots (gray squares) are spread in this region due to the stronger repulsion forces that exist among them. As mentioned, this enables some of these robots to escape from the local minima. In Figure 3(c), the initial effect of the rescuers can be observed. There are some *attached* robots (right-pointing white triangles) using the information transmitted by the *rescuers* (left-pointing black triangles) to escape the dead end in the bottom left of the obstacle. Others are already moving in the correct direction while the *attached* robots re-transmit the rescuer's information.

Figure 3(d) shows all robots escaping the local minima region, thanks to the communication network created by the *attached* robots that enabled all of them to receive a feasible path. Soon we achieve the situation in 3(e) where most robots are reaching the target region (black diamonds) while there are still some *rescuer* robots that are looking for *trapped* ones and will later move back to the goal (Figure 3(f)).

IV. Experiments

To demonstrate the feasibility of the proposed approach, we performed some experiments with a couple of *e-puck* robots. The e-puck is a small-sized (7cm diameter) differential drive robot that is very suitable for swarm experimentation [16]. Each robot is equipped with a ring of 8 IR sensors that allows proximity sensing and a group of colored leds to indicate robot status. Local processing is performed by a dsPIC microprocessor and a bluetooth wireless interface allows robot to robot communication and remote control. The robot is also equipped with a micro-camera, a 3D accelerometer, speakers and a microphone. Figure 4 shows the robots used in the experiment.

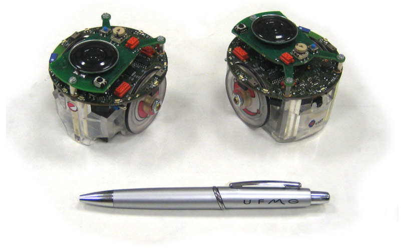

Fig. 4. Pair of *e-puck* robots used in the experiments.

Despite not being executed with swarms of robots, these experiments demonstrate three fundamental robot competencies for the execution of the proposed algorithm: (i) the robot's ability of localizing itself, following the negative gradient to a specific goal, and retracing its path; (ii) the ability of communicating its path to a trapped robot that will follow this path to the goal; and (iii) the capacity of detecting that it is trapped in a local minima.

A. *Localization, Gradient Descent, Path Retracing*

As discussed in Section II, robots navigate to the goal following the negative gradient of an implicit function and avoid obstacles and other robots using repulsion forces. For gradient computation, robots must know their position. In this paper, we use a localization system specifically designed for swarm localization in indoor environments [17]. In this system, robots are tagged with geometrical markers and a group of overhead cameras is used to localize and uniquely identify the robots. The use of geometrical markers makes the system scalable to large numbers of robots while the association of multiple cameras allows the coverage of a larger work area. A modular and distributed software system that may run in different computers is responsible for gathering information from multiple cameras, localize the robots, and transmit this information in real time to the robots. Experiments performed in [17] showed that the system is capable of detecting up to 40 different markers at 25Hz on a single computer, with 1cm accuracy.

Since we are using differential drive robots in these experiments, some changes had to be made in the controller of Equation 2. The resultant acceleration vector generated by the controller is integrated to linear and angular velocities using the approach proposed in [18]. The repulsive potential field generated by the obstacles is also based on [18], using distances measured through the IR sensors (the sensing range is about 4.5 cm). Constants are set to $\alpha = 2$, $\beta = 0.002$, $\gamma = 0.002$, and the maximum velocity is set to 2.05 cm/s.

Figure 5 shows a trajectory performed by one robot using this controller in a environment containing a single obstacle. The implicit function sets the goal region approximately around position (0,0). The start and end points are marked with a triangle and a circle respectively. It can be observed that the robot successfully follows the inverse of the gradient from a initial position to the goal while avoiding obstacles. After reaching the goal, it retraces its path using the saved waypoints. The total number of waypoints in this path is 118 (without discards). Considering that each waypoint needs 64 bits, the total memory required for storing the path is less than 1KB. Notice that due to the robot's differential constraints, the retraced trajectory is not identical to the saved one, but both are sufficiently close for the rescue mission.

B. *Communication and Rescue*

Figure 6 shows a complete run of the algorithm in a scenario containing a local minima. The graphs show the trajectories executed by two robots (robot 1 - solid, robot 2 - dashed). In figure 6(a), robot 1 moves from a start position

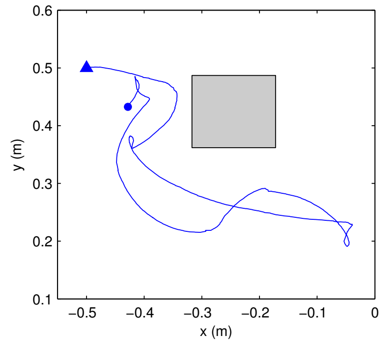

Fig. 5. E-puck's trajectory: the robot moves from a start position (triangle) to the goal and then retraces its path.

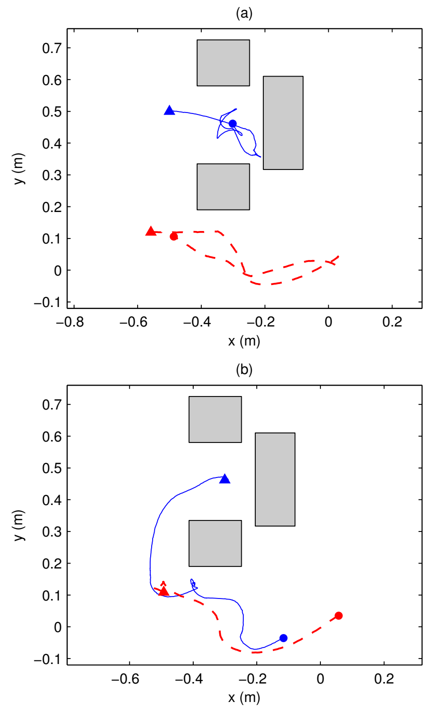

Fig. 6. A complete run of the algorithm. (a) Robot 1 (solid) gets trapped while robot 2 (dashed) reaches the goal and retraces its path. (b) Robot 1 escapes local minima after receiving a message from robot 2. The triangles mark the start of the trajectories.

until getting trapped in a local minimum, while robot 2 reaches the goal around position (0,0) and then becomes a rescuer and retraces its path. Figure 6(b) shows robot 1 escaping from the local minimum and moving to the goal after receiving a message from robot 2 containing a free path. As explained in Section III, robot 1 first moves to the position from where robot 2 sent the message and then follows the received waypoints to the goal. Robot 2, after retracing its path, also follows its saved waypoints to the goal.

Robots communicate using a bluetooth interface. In this experiment, differently from the simulations, robots do not need to "see" each other to communicate, i.e., they exchange messages even when there is no line of sight between them. This leads to an interesting situation: after receiving the

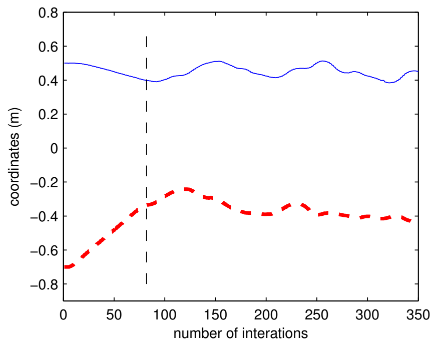

Fig. 7. Robot changes to the *trapped* state at the vertical line when the variation in its recent positions is small.

waypoints, robot 1 changes its status to attached and tries to move towards robot 2 to start the waypoint following. But since there is no direct path between them, robot 1 gets trapped again. This happens 8 times until robot 2 sends a message from a position that can be reached by robot 1. This may explain the movements performed by robot 1 inside the local minima while robot 2 is retracing its path in Figure 6(a).

C. Trapped State Detection

To be able to execute the mode switching presented in Section III, robots must have the ability of detecting that they are trapped in a local minimum. To do this, a robot keeps track of its recent positions. If the distance from its current location to one of these positions is less than a certain threshold, it considers itself trapped. To demonstrate this ability we ran some experiments in which one robot gets trapped in a local minima caused by an u-shaped obstacle, in a scenario similar to Figure 6(a).

Figure 7 shows the robot's coordinates x (dashed) and y (solid) as a function of time. The vertical line marks when the robot changes its status from *normal* to *trapped*. It can be noticed that the robot correctly detects that it is trapped when the variation of both coordinates during a specific amount of time is small. The position variation that activates the *trapped* state must be tuned according to the robot expected velocities, otherwise a robot that is moving very slowly may consider itself trapped. But it is important to mention that these "false-positives" generally do not compromise the algorithm, since, as explained in Section III, robots in the *trapped* state still perform the gradient descent.

V. Conclusion

This paper presented a methodology for controlling and coordinating large groups of robots to navigate and synthesize complex shapes in environments containing unknown obstacles. Shapes are modeled using implicit functions generated interpolating from a set of constraint points. A composition of gradient descent and potential fields is used to guide robots to the target while avoiding obstacles. To overcome local minima, that may appear in these scenarios, we developed a distributed coordination mechanism based on mode switching that reallocates some robots as rescuers and sends them to help the robots that may be trapped. Simulations and real experiments demonstrated that this composition of simple controllers and explicit coordination allowed robots to successfully navigate and synthesize shapes in these environments. We believe that the use of local interactions and task allocation mechanisms with robot swarms opens an interesting path for research and we intend to investigate this further.

References

[1] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling swarms of robots using interpolated implicit functions," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 2498–2503.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics (SIGGRAPH 87)*. ACM Press, 1987, pp. 25–34.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[4] R. Bachmayer and N. E. Leonard, "Vehicle networks for gradient descent in a sampled environment," in *Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02)*, 2002, pp. 112–117.

[5] V. Gazi and K. Passino, "Stability analysis of social foraging swarms," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 1, pp. 539–557, 2004.

[6] M. A. Hsieh and V. Kumar, "Pattern generation with multiple robots," in *Proceedings of the 2006 International Conference on Robotics and Automation (ICRA)*, Orlando, FL, 2006, pp. 2442–2447.

[7] L. C. A. Pimenta, A. R. Fonseca, G. A. S. Pereira, R. C. Mesquita, E. J. Silva, W. M. Caminhas, , and M. F. M. Campos, "On computing complex navigation functions," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3463–3468.

[8] A. Kamphuis and M. Overmars, "Motion planning for coherent groups of entities," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, Louisiana, Apr 2004, pp. 3815–3821.

[9] C. Belta, G. A. S. Pereira, and V. Kumar, "Abstraction and control for swarms of robots," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, 2003.

[10] M. Kloetzer and C. Belta, "Hierarchical abstractions for robotic swarms," in *Proceedings of the 2006 International Conference on Robotics and Automation (ICRA)*, Orlando, FL, 2006, pp. 952–957.

[11] T. Y. Li and H. C. Chou, "Motion planning for a crowd of robots," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003, pp. 4215–4221.

[12] N. Correll, S. Rutishauser, and A. Martinoli, "Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures," in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Rio de Janeiro, Brazil, July 2006.

[13] L. Chaimowicz, M. Campos, and V. Kumar, "Dynamic role assignment for cooperative robots," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, 2002, pp. 292–298.

[14] Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara, "An integrated token-based algorithm for scalable coordination," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005, pp. 407–414.

[15] Wikipedia, "Pheromone — wikipedia, the free encyclopedia," 2007, [Online; accessed 02-September-2007]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Pheromone&oldid=119708944

[16] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli, "Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics," in *Simulation of Adaptive Behavior (SAB-2006), Swarm Robotics Workshop*, ser. Lecture Notes in Computer Science (LNCS), 2007, pp. 103–115.

[17] R. Garcia, P. Shiroma, L. Chaimowicz, and Campos, "A framework for swarm localization," in *Proceedings of VIII SBAI – Brazilian Symposium on Intelligent Automation*, October 2007, (In portuguese).

[18] A. De Luca and G. Oriolo, "Local incremental planning for nonholonomic mobile robots," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994.