A Coordination Mechanism for Swarm Navigation: Experiments and Analysis

(Short Paper)

Leandro Soriano Marcolino
VeRLab – Vision and Robotics Laboratory
Computer Science Department - UFMG - Brazil
soriano@dcc.ufmg.br

Luiz Chaimowicz
VeRLab – Vision and Robotics Laboratory
Computer Science Department - UFMG - Brazil
chaimo@dcc.ufmg.br

ABSTRACT

We present an algorithm that allows swarms of robots to navigate in environments containing unknown obstacles, moving towards and spreading along 2D shapes given by implicit functions. Basically, a gradient descent approach augmented with local obstacle avoidance is used to control the swarm. To deal with local minima regions, we use a coordination mechanism that reallocates some robots as "rescuers" and sends them to help other robots that may be trapped. The main objective of this paper is to analyze the performance of this algorithm in terms of its completion rate and communication requirements as the number of robots increases. For this, a series of simulations are presented and discussed.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*

General Terms

Algorithms, Experimentation

Keywords

Swarms, Multi-Robot Coordination

1. INTRODUCTION

The use of large groups of robots, generally called *swarms*, in the execution of complex tasks has received much attention in recent years. Inspired by their biological counterparts, these systems employ a large number of simpler agents to perform different types of tasks. Swarms of robots may bring performance gains, by dividing the work among the team, and also increased robustness, by having robots with redundant capabilities and dynamically reconfiguring the team in case of failures. But, on the other hand, swarms must execute in a completely decentralized fashion and using limited communication. Hence, new algorithms to control and coordinate these very large groups of robots must be developed.

We recently proposed a coordination mechanism that allows swarms of robots to overcome local minima regions while navigating to a specific goal [5]. Basically, robots that reach the goal can be reallocated as rescuers and retrace their paths to help teammates that may be stuck in local minima. In this paper, we revisit this mechanism, carefully analyzing its performance in terms of its completion rate and communication requirements as the number of robots increases. Based on this analysis, we discuss some key aspects and propose improvements to the mechanism that would make it more efficient and robust.

Cite as: A Coordination Mechanism for Swarm Navigation: Experiments and Analysis (Short Paper), Leandro Marcolino, Luiz Chaimowicz, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.),May,12-16.,2008,Estoril,Portugal,pp.1203-1206.

Motion planning and control for large groups of robots has received a great deal of attention in recent years. A general approach in the literature is to control robots in a decentralized way, trying to mix gradient descent with local repulsion forces [1]. Unfortunately, as in regular potential field approaches, the presence of obstacles and local repulsion forces among the robots may cause convergence problems, mainly when robots are required to synthesize shapes. Hsieh and Kumar [3] 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 [6]. But these approaches may be hard to compute in real time and may not be applicable to all types of environments.

Besides motion planning, certain types of tasks may require a greater level of coordination to be executed. When dealing with swarms, coordination mechanisms have to scale to tens or hundreds of robots. Scalable approaches for the coordination of large groups of agents (not necessarily robots) have been proposed in [2, 4, 7] among others. Scerri et al. [7] for example, present an algorithm called LA-DCOP that uses tokens to encapsulate task information. These tokens circulates through subsets of agents in order to find a suitable agent for task execution.

In our approach, instead of restricting the environment or developing complex controllers and coordination mechanisms, we rely on the composition of a gradient descent controller and a simple coordination mechanism to navigate swarms in environments containing unknown obstacles. As will be shown in Section 3, this approach provides a good completion rate and is scalable to large numbers of robots.

2. CONTROL AND COORDINATION

The overall mission to be executed by the swarm can be stated as follows: robots must move towards and spread along a goal region in an environment containing unknown obstacles. For doing this, they are controlled by a gradient descent augmented with local repulsion forces, as detailed in [5].

The composition of these forces can lead to local minima regions. 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. 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.

Our coordination is based on a mode switching mechanism, generally known in robotics as dynamic role assignment [8]. A robot can switch between different modes (or roles) during 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.

These modes can be better modelled by a finite state machine (FSM), in which the states and edges represent respectively the modes and the possible transitions between them. In the mechanism presented in this paper, the FSM for each robot is shown in Figure 1. It is composed of five different modes: *normal*, *trapped*, *rescuer*, *attached* and *completed*.

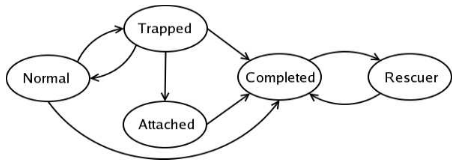


Figure 1: Finite state machine showing the possible modes and transitions for each robot.

All robots start in the *normal* mode. A *normal* robot performs a gradient descent trying to reach the goal while avoiding obstacles. A *normal* robot switches its mode to *trapped*, if it considers itself trapped in a local minima region. This transition is determined by the variation in the robot's position over time. If its position does not change significantly during a certain amount of time, it becomes *trapped*. A *trapped* robot may switch back to *normal* if its position changes considerably again. Sometimes, due to the resultant forces in the controller, robots may switch to *trapped* even if they are not in a local minima region. These "false-positives" do not compromise the performance of the algorithm since *trapped* robots also perform gradient descent.

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.

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. The number and frequency of rescuer robots are set empirically and may vary depending on the total number of robots and characteristics of the environment. The algorithm used to determine which robots become rescuers is explained in [5]. In order to minimize memory requirements, 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.

A *trapped* robot keeps sending messages announcing its state. When a *rescuer* listens to one of these messages, 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, creating a powerful communication chain. Finally, a robot will change its mode to *completed* when it reaches the target. As mentioned, *completed* robots may become *rescuers* according to the coordination tokens.

3. SIMULATIONS

A series of simulations were performed to better observe the performance of the proposed mechanism. Basically, we would like to analyze two main aspects: (i) the effectiveness of the algorithm in terms of its completion rate, i.e., the percentage of robots that are able to reach the target, and (ii) the scalability of the mechanism, mainly in terms of its communication requirements. Two scenarios were used in this analysis (Figure 2). The first one is composed by a square target surrounded by four u-shaped obstacles. The second scenario has a large u-shaped obstacle between the initial position of the robots and the target. Each simulation was executed 10 times and the arithmetic mean of the results was calculated.

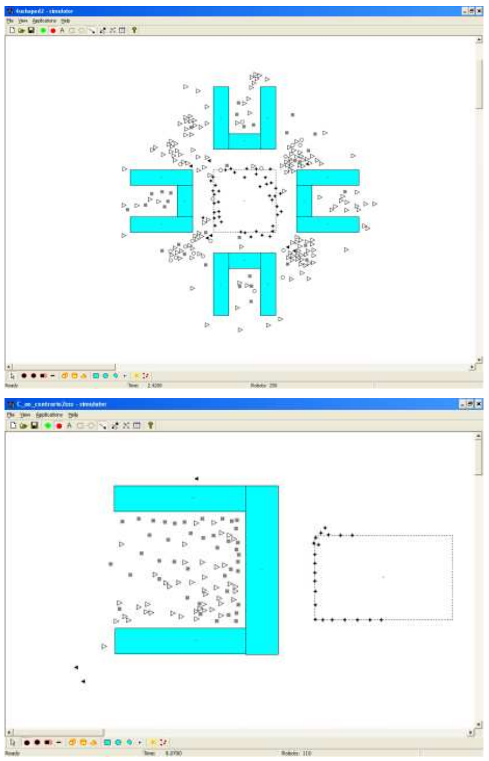


Figure 2: Two scenarios used in the simulations.

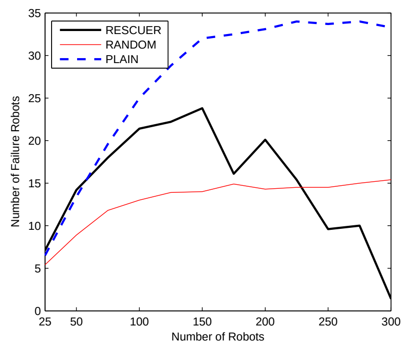
3.1 Effectiveness

To analyze the effectiveness of the algorithm in overcoming local minima, we performed a series of simulations where we measured the number of "failure robots", i.e., robots that were not able to reach the target, as we increased the total number of robots. We compared the performance of the proposed algorithm (from now on called RESCUER Algorithm) with two others: a PLAIN Algorithm, in which robots do not have any strategy to escape local minima and a RANDOM Algorithm, in which random forces are applied to *trapped* robots. Basically, in the RANDOM Algorithm, when a robot is in the *trapped* state, it is subjected to a force in a randomly chosen direction, for a randomly chosen time to try to escape the local minima region.

The graph on the top of Figure 3 shows the number of "failure robots" for the first scenario. For the RESCUER Algorithm, the number of failures increases for a small number of robots, than it tends to decrease as the total number of robots increases. The initial increase is related to the fact that, with fewer robots, the local repulsions in some of the u-shaped obstacles are not strong enough to free some robots to become rescuers and help the others. Despite this fact, the percentage of failure tends to decrease as the number of robots gets higher, indicating the rise of the quality of convergence as we increase the number of robots. The performance of RANDOM was better when less than approximately 250 robots were used. For more than 250 robots, the algorithm could not improve the completion rate anymore, while the results of the RESCUER Algorithm continued to get better. Compared to the others, the PLAIN algorithm had a very low quality of convergence, as it cannot do anything to overcome local minima situations.

In the second scenario, the benefits of the proposed algorithm are more clear. In the second graph of Figure 3, we can see the total number of robots that were unable to reach the target for this scenario. It is interesting to note that the three algorithms performed poorly with a small number of robots, but as the number of robots increases (more than 100 robots), the RESCUER Algorithm has an excellent performance, obtaining almost 100% of completion rate.

The differences found in those results can be explained by the different characteristics of the two scenarios. In the first scenario, there are a larger number of local minima regions distributed in the environment, but the size of those regions are small. Thus, the number of robots that get trapped in those regions is smaller and, even those that get trapped, can escape with the application of random forces, mainly considering that *trapped* robots apply stronger repulsion forces between them. Also, since the local minima regions are spread in the environment, a larger number of *rescuer* robots is necessary to find and help the ones stuck in local minima. That explains why in this scenario it is necessary a higher number of robots for the RESCUER Algorithm to perform better. On the other hand, the second scenario has just one very large local minima region. So robots cannot escape easily applying only random forces since these forces only work adequately for robots that are close to the border of the u-shaped obstacle. Moreover, since *trapped* robots are in the same region, a single *rescuer* robot could save all the robots. In this case, the chain of communication created by the *attached* robots is very effective. When a *trapped* robot receives a message, this message is quickly spread among the swarm. This explains the sudden improvement in the proposed algorithm when a certain number of robots is used.



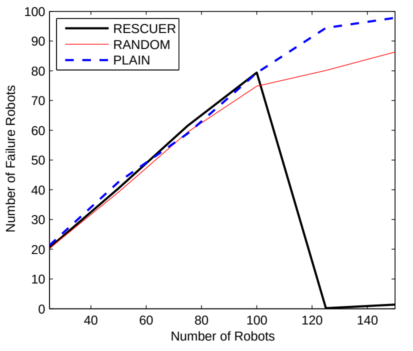


Figure 3: Number of failure robots for the first (top) and second (bottom) scenarios.

3.2 Communication

The coordination mechanism used in this paper relies on explicit communication between robots. When dealing with swarms of robots, it is specially important to see if the communication requirements do not compromise the scalability, i.e., if the communication needs of the algorithm do not increase unboundedly as the size of the swarm grows.

To study this important aspect of the algorithm, we performed simulations measuring the total number of exchanged messages as the number of robots increases. We ran simulations in the first scenario setting communication range to 20 times the radius of the robots. The results can be seen in the graph of Figure 4. Basically, there are two types of messages in the algorithm: "status messages", that are sent by *trapped* robots asking for help, and "path messages" that are sent by the rescuers upon receiving a status message. We use two y axis (status on the left, path on the right) since they are one order of magnitude different.

As mentioned, status messages are sent by *trapped* robots asking for help, thus, its variation is proportional to the number of robots that get stuck in local minima regions (shown in Figure 3). The exchange of path information depends on the number of *rescuer* and *trapped* robots. It can be observed in Figure 4 that it increases linearly with the number of robots. Besides that, it can also be observed that the number of status messages is a lot higher than the number of messages with path info. Status messages are generally very small, with few bytes representing the robot status. On the other hand, the size of the path messages depends on the number of waypoints stored by the robots. We performed some experiments using the same scenario that showed that this number is not very high (average of 230 waypoints) and decreases as the number of robots in-

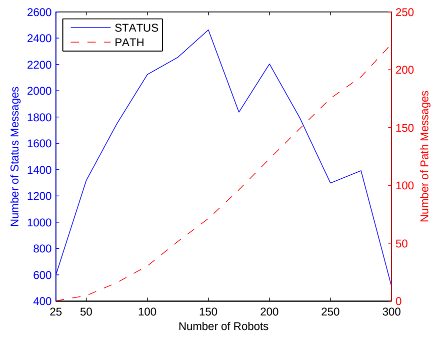


Figure 4: Number of messages sent (first scenario).

creases. Moreover, after eliminating redundant waypoints, the total number is less than 20, which does not compromise bandwidth.

We also performed some experiments in which we fixed the number of robots in 250 and varied the communication range to observe how it affects the amount of messages sent. The results are shown in Figure 5. It is interesting to note that the number of messages with path information increases with a small communication range, but tends to stabilize after the range reaches 20 times the radius of the robot. It is also interesting to see that the number of messages with status decreases as the range of communication gets higher. As *trapped* robots are rescued more effectively with a higher communication range, the amount of time that they waste sending status messages decreases, therefore decreasing the total number of status messages sent by the robots.

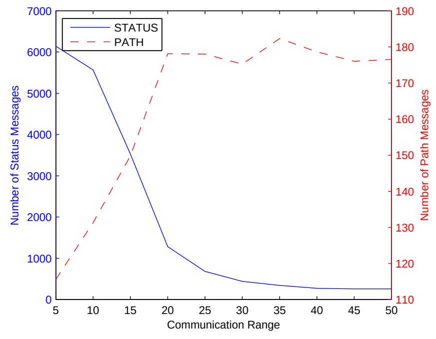


Figure 5: Total number of messages sent for a varying communication range.

4. CONCLUSION

From the results of the previous section, we can draw a very important conclusion: the proposed algorithm performs better for larger teams. As the number of robots increases, two important aspects that improve the performance can be observed: firstly, the local force interactions between robots get more intense, causing robots to spread inside the local minima and even forcing some of them out of this region. Secondly, a larger number of robots can be reallocated as rescuers and help teammates to complete the mission. Another important result is that the communication requirements of the algorithm do not compromise scalability. It was shown that the amount of path messages increases linearly with the number of robots while the number of status messages decreases as fewer robots get stuck. Besides that, communication is performed locally (only among neighbors) and the overall size of each message is very small. These two results, along with the fact that the algorithm is completely decentralized and do not need detailed information about the environment, makes it very suitable for this type of navigation, in which the swarm must converge and spread along a specific target in an environment containing unknown obstacles.

The experiments and analysis performed in this paper also helped us to identify some important issues in the proposed approach that may be better addressed. Firstly, we noticed that sometimes robots get caught in a “traffic jam”, when several of them converge simultaneously to the same locations. So the development of a mechanism for congestion control would be important. Another point is that robots must know their global position in order to compute the gradient forces and store waypoints. This is a strong assumption, but it is generally accepted when simulating swarms of robots. To address this issue in real experiments, we developed a framework for swarm localization in indoor environments. Robots are tagged with geometrical markers and a group of overhead cameras is used in a distributed way to localize and uniquely identify the robots. This framework was successfully used in some proof-of-concept experiments presented in [5]. The development of algorithms for swarm navigation that do not require global localization is particularly difficult and has become an important topic of research. Finally, to better validate the proposed algorithm, it is necessary to perform experiments using a large group of real robots. We are currently performing some experiments using a group of seven robots and the results seem very promising.

5. REFERENCES

[1] R. Bachmayer and N. E. Leonard. Vehicle networks for gradient descent in a sampled environment. In *Proc. of the 41st IEEE CDC*, pp. 112–117, 2002.

[2] N. Correll, S. Rutishauser, and A. Martinoli. Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures. In *Proc. of 10th ISER*, 2006.

[3] M. A. Hsieh and V. Kumar. Pattern generation with multiple robots. In *Proc. of the IEEE ICRA*, 2006.

[4] M.-W. Jang and G. Agha. Dynamic agent allocation for large-scale multi-agent applications. In *Proc. of the Workshop on Massively Multi-Agent Systems (MMAS)*, pp. 19–33, 2004.

[5] L. S. Marcolino and L. Chaimowicz. No robot left behind: Coordination to overcome local minima in swarm navigation. In *Proc. of the IEEE ICRA*, 2008.

[6] L. C. A. Pimenta, A. R. Fonseca, G. A. S. Pereira, R. C. Mesquita, E. J. Silva, W. M. Caminhas, , and M. Campos. On computing complex navigation functions. In *Proc. of the IEEE ICRA*, pp. 3463–3468, 2005.

[7] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proc. of AAMAS*, pp. 727–734, 2005.

[8] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241-273, 1999.