Cooperative Motion Coordination Amidst Dynamic Obstacles

Stefano Carpin*[1] and Lynne E. Parker[2]

[1] Intelligent Autonomous Systems Laboratory
Department of Electronics and Informatics
The University of Padova, ITALY
[2] Center for Engineering Science Advanced Research
Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, Tennessee, U.S.A.

Abstract. The cooperative leader following task for multi-robot teams is introduced and discussed. We describe the design and implementation of a distributed technique to coordinate team level and robot level behaviors for this task, as well as a multi-threaded framework for the implementation of a heterogeneous multi-robot system. This approach enables robots to remain in formation as they deal with other obstacles that may appear within the formation. We describe how the robot behaviors are realized and scheduled. The proposed approach has been run and validated on a team of robots performing in both indoor and outdoor environments.

1 Introduction

In this paper we address the problem of *leader following* in the case of a heterogeneous multi-robot team. This task requires the robots to move in a *linear* pattern, each following the previous robot, with the first robot either following a human operator, being teleoperated, or going through a predetermined path. While others have previously studied this leader following behavior, our focus is specifically on enabling the team to perform in a *robust* way, so that the team is able to correctly operate even if some external undesired or unforeseen event occurs. Examples of unexpected events include an obstacle in the way or one or more of the robots failing.

We propose a distributed policy based on explicit communication that allows this goal to be achieved at the team level. We introduce a multi-threaded structure employed on different robots, and we also give the details about the tracking techniques. This framework allows us to abstract the coordinated tracking process from the low-level sensor details. This paper is organized as follows: related work is discussed in Section 2, while our approach in described in Section 3. Section 4 briefly outlines how the various sensors are used for leader following and obstacle avoidance. Finally, Section 5 presents experimental results, with conclusions offered in Section 6.

* Research performed while visiting Oak Ridge National Laboratory.

2 Related Work

The task of pattern formation and formation marching has gained a lot of attention in the last years, and is one of the challenging issues in multi-robot research [10]. In [2] a real implementation on a team of outdoor robots is discussed. A class of reactive behaviors that implement formations is introduced and tested on a team of military unmanned ground vehicles performing in an outdoor terrain. While this work is similar to ours in terms of the behavior based approach and the outdoor operating scenario, the main differences are that we do not use a global positioning system (e.g., GPS) and we also deal with a heterogeneous system in which robots are equipped with different sets of sensors.

Potential field approaches are also widely used (see for example [11,12,2]). Robots move by being attracted to their desired position in the pattern and being repulsed from obstacles and other robots. In this context one of the main problems is dealing with systems that operate in environments that exhibit significant dynamics, since the group has to promptly react to unforeseeable circumstances that can emerge inside or outside the team itself [4]. In [5] the idea of moving a team by means of a leader that conducts the rest of the robots is discussed and some simulation results are shown. The problem of designing control laws for this kind of problem is discussed in [9]. One of the main issues is related to the design of *distributed* control schemes – i.e., schemes where each robotic agent acts on the basis of local decisions [3,7,13] as opposed to *centralized* schemes where decisions are made by a unique subject, which has a global view of the situation, and are then communicated to robotic agents [1]. In constrast to much of this previous research, our research explicitly addresses issues of maintaining formation in a leader-following application while also avoiding obstacles that may unexpectedly appear within the formation.

3 Distributed Control Architecture

We address the following problem: given a set of possibly heterogeneous robots arranged in a linear pattern, design a strategy so that if the first moves in an unknown environment, the others follow while at the same time avoiding obstacles that may appear within the formation. We call this task *Cooperative Leader Following*. In our approach, we distinguish between *team-level* behaviors and *robot-level* behaviors. At the team level we use a situated automaton [8] approach, with the team seen as a finite state automaton whose inputs come from the environment. The transitions between the three group-level behaviors are shown in Figure 1. The behaviors at the team level are *Team-Follow*, *Team-Wait*, and *Team-Recover*. When the team is in *Team-Follow*, every robot (except the first) will follow its local leader. The *Team-Wait* behavior is executed when the team is waiting for some event to happen. This is the case when, for example, a moving object approaches a robot, so that it is not safe to keep moving. In this case all robots stop to avoid breaking the formation. The *Team-Recover* behavior is executed when the team is trying to recover from a *wait* condition. Since not all robots are involved in a collision danger situation, when the team is performing this behavior single robots will execute different behaviors, with some robots trying to go around obstacles, and others simply performing regular following at a reduced speed.

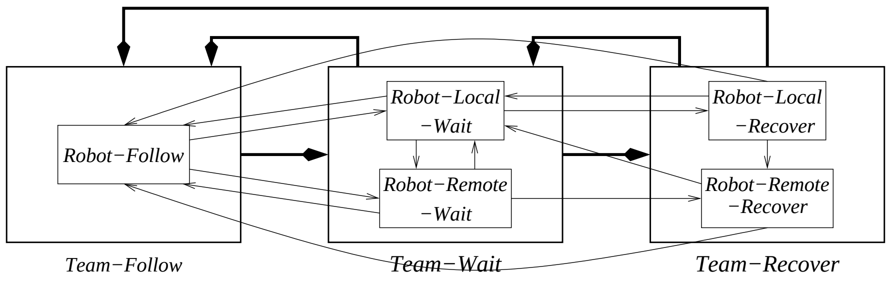

Fig. 1. Team-Level behaviors (bold lines), Robot-Level behaviors (thin lines) and their transitions.

The introduction of the *Team-Wait* behavior is motivated by the assumption that the team will not operate in an interference-free environment, but rather in an unknown, possibly unstructured, environment shared with other moving entities, such as humans or other robots. In this scenario, it could happen that a moving object approaches the robot, so that it is necessary for it to stop to avoid a collision. The robot first *waits* for a certain amount of time for the obstacle to go away. If this time is exceeded, then the robot attempts to circumnavigate the obstacle – i.e., to *recover* from the situation and to resume the *follow* behavior. The wait stage is introduced because recovering is a difficult task, and it is preferred to execute it only when there is no alternative. Of course, when a robot is waiting, other robots should wait too, to keep the formation together. From the above discussion it is evident that some sort of *explicit* communication is necessary to gain the desired team level behavior, especially for large team sizes.

At the single robot level we design a set of five behaviors that, locally executed, give the team level behaviors previously described. They are:

- **Robot-Follow**. The robot is following its leader.
- **Robot-Local-Wait**. The robot is waiting because an obstacle does not let it move safely.
- **Robot-Remote-Wait**. The robot is waiting because one or more other robots are in *Robot-Local-Wait*.
- **Robot-Local-Recover**. The robot is trying to recover from a *Robot-Local-Wait* situation. This means that it is trying to overtake an obstacle while keeping track of its leader.

- **Robot-Remote-Recover**. The robot is following its leader but at a reduced speed, so that if its follower is doing a *Robot-Local-Recover* behavior, it will be easier for the robot to keep tracking while overtaking the obstacle.

At the team level the switching between different behaviors is triggered by explicit communication, while at the single robot level the triggering comes both from sensors and from communication.

One of the goals of this work is to develop a *scalable* framework for the distributed control of the fleet motion, which operates independently of the size of the team. Our approach is based on the use of two counters, *Wait* and *Recover*, to keep track of the number of robots that are performing *Robot-Local-Wait* and *Robot-Local-Recover*. If both those counters are 0, then the team is performing *Team-Follow* (i.e., every robot is performing *Robot-Follow*). When a robot senses a dangerous situation it increments both counters and starts a timer. When the timer expires it decreases the Wait counter and when the dangerous situation does not hold anymore it decreases the Recover counter. Other robots decide what to do on the basis not only of the information they sense, but also on the basis of counters' values. If Wait is greater than zero and no local dangerous situation is detected the behavior is *Robot-Local-Wait*. If Wait is zero but Recover is positive *Robot-Local-Recover* is executed. Table 1 illustrates this approach. S_n is the state of the automata at time n, while W_n and R_n are the values of the counters at time n. The input of the automata, I_n, can be one of the following:

- **LWB** (Local Warn Begin): sensors provide information that something is too close to keep following
- **LWE** (Local Warn End): sensors provide information that a previously detected dangerous situation does not hold anymore
- **LTO** (Local Time Out): the timer started with LWB expired so it is necessary to switch from waiting to recovering

In order to have a distributed implementation, the Warn and Recover counters are not allotted to a unique entity, but are rather shared among all the robots – i.e., every robot has its own copy. For this reason four different messages are sent, namely two for updating Warn (increase and decrease) and two for updating Recover. Messages are anonymously sent in a broadcast fashion. In this way, for both sending and receiving, robots do not need to know the number of members of the team. Thus the approach is independent of the size of the team itself.

4 Sensor Processing Techniques

To test the proposed framework, we implemented this approach on a heterogeneous team of three to five mobile robots. The robots are heterogeneous

S_n	W_n	R_n	I_n	S_{n+1}	W_{n+1}	R_{n+1}
Follow	0	0	LWB	Local Wait	1	1
Follow	1	1	-	Remote Wait	1	1
Local Wait	n	1	LWE	Remote Recover	n-1	0
Local Wait	1	1	LWE	Follow	0	0
Local Wait	n	m	LWE	Remote Wait	n-1	m-1
Local Wait	n	m	LTO	Timer Elapsed	n-1	m
Timer Elapsed	0	m	-	Local Recover	0	m
Timer Elapsed	n	m	LWE	Remote Wait	n	m-1
Local Recover	0	m	LWE	Remote Recover	0	m-1
Local Recover	0	1	LWE	Follow	0	0
Local Recover	1	m	-	Timer Elapsed	1	m
Remote Wait	n	m	LWB	Local Wait	n+1	m+1
Remote Wait	0	m	-	Remote Recover	0	m
Remote Wait	0	0	-	Follow	0	0
Remote Recover	n	m	LWB	Local Wait	n+1	m+1
Remote Recover	n	m	-	Remote Wait	n	m
Remote Recover	0	0	-	Follow	0	0

Table 1. Transitions of the automaton based on messages coming from sensors and on the value of Warn and Recover counters. The state *Timer Elapsed* is used when a robot's timer elapsed, but the robot has to wait because of W_n is greater than 0. In that case its behavior is *Robot-Local-Wait*.

in that they are equipped with different sensor suites and have different mobile platforms. We developed a multi-threaded software architecture in which each sensor is handled by a separate thread that uses its data to obtain information about tracking and navigation. Each thread then sends its output (e.g., the polar coordinates of the point to track) to a *Decisor* thread which, on the basis of the desired behavior and sensory input, drives the robot. The Decisor first merges the provided points to track and then decides how to move on the basis of a set of *fuzzy rules*. A separate thread handles explicit communcation. Each robot is equipped with sonar sensors that provide distance readings within four meters. Three of the robots are equipped with a SICK PLS (Proximity Laser Scanner) laser range finder. Also, three robots have a Sony camera mounted on a pan-tilt unit. Each robot is equipped with wireless Ethernet, which allows them to communicate over the standard TCP/IP protocol. Finally, we note that even though these robots also have a Differential Global Positioning System (DGPS), we did not use it in this research because of its frequent unavailability (e.g., due to indoor operations, satellite obstructions, etc.).

In our approach, we use the available sensors that are most appropriate for each navigation subtask. Our experimental trials involving sonar-based versus laser-based (or camera-based) tracking have clearly shown that the laser and camera sensors are much more accurate than the sonar beyond a short distance. Thus, the laser scanners and cameras are used for robot tracking. However, the laser and camera sensors do not have 360 degree coverage of the space around the robot, whereas the sonars are positioned around the entire robot periphery. Thus, the sonars are used to detect obstacles that approach the robot from directions unseen by the laser and camera sensors, and for short-range obstacle detection.

4.1 Using laser PLS sensor to track the leader

The SICK PLS sensor provides readings with a scan angle of 180 degrees and an angular resolution of 0.5 degrees. Distances returned under 15 meters are considered reliable. Starting from this sensor data, the tracking routine removes spikes and averages across samples to obtain a smoother sequence (see Figure 2). From the smoothed sequence, the routine then extracts all the local minima in the sequence and tries to match each one with the previous tracked minima[1]. The one which is closest is considered to be the new local minima to track and will be tracked in the next step. The routine returns the polar coordinates (distance, ρ, and angle, θ) of the minima to track. This is the information that the thread dealing with the laser will provide to the Decisor module. The chosen local minima is the one which minimizes the quantity:

$$d = \sqrt{(\rho_c \cos\theta_c - \rho_t \cos\theta_t)^2 + (\rho_c \sin\theta_c - \rho_t \sin\theta_t)^2}$$

where (ρ_c, θ_c) are the polar coordinates of the candidate local minima and (ρ_t, θ_t) are the coordinates of the minima currently being tracked. This formula gives the distance on the cartesian plane between the candiate local minima and the currently tracked minima. In addition, during the local minima search, the routine verifies that no obstacle is too close to the robot. If this is not the case, the LWB condition is raised and the corresponding LWE condition is issued when the obstacle moves away.

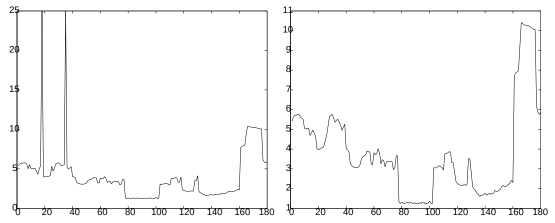

Fig. 2. Raw readings from the PLS sensor and a smoothed sequence (note the difference in the two scales).

[1] It is assumed that at the beginning all the robots are arranged in a line pattern, so that at the first scan cycle the current position of the leader is roughly known.

4.2 Using CCD camera to track the leader

Three of our robots are equipped with a CCD camera mounted on a pan-tilt unit. The associated framegrabber returns a color image in the RGB coordinate space of 120×160 pixels. Robots equipped with a camera have to detect and follow in real time a similar robot in their field of view. The designed approach for detecting the position of the leader robot is as follows:

- *Color segmentation* is accomplished by defining a range of colors that match the red color of our robots, discarding all other pixels.
- *Averaging (Smoothing)* is achieved using a neighborhood averaging technique in which pixel color is updated by the eight surrounding pixels. If four or more are red it is set to red, otherwise it is set to white (white pixels are ignored in the following steps).
- *Blob detection* is done by checking the boundaries of twenty-five pixels. When a red pixel along the boundary of a region is found, the region is flagged as a possible hit.
- *Object assignment* gives a different label to each connected component, using the iterative algorithm by Haralick [6].
- *Object selection* decides which of the objects should be tracked. This is done by comparing the center of mass of every distinct object with the position in the image plane of the previous object being tracked (in a similar way to what is done in the laser routine, as outlined in Section 4.1).
- *Proximity estimation* gives a rough estimation of the distance to the robot, based on the dimension of the blob being tracked, and of its position on the image plane.

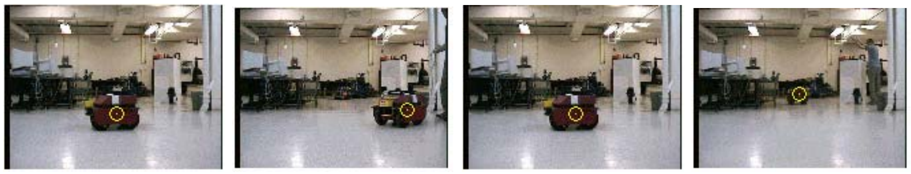

Fig. 3. Results of visual tracking approach. Small white circles indicate locations of robots detected by the visual tracking algorithm.

Thus, as for the laser, the camera-handling thread gives a position to track in polar coordinates (i.e., distance and direction). Figure 3 shows examples of the detection of robot position using this approach.

4.3 Using sonar to detect obstacles

Every robot is equipped with sonar sensors. In our experiments they have been used for obstacle avoidance only, and not for tracking, because from a number of trials it became evident that crosstalk and environmental conditions were such that tracking could be obtained only within a small distance range. Instead, they proved to be highly effective for obstacle avoidance. The thread which deals with sonars identifies the LWB and LWE conditions, as for the laser, and in addition it produces a vector whose direction and intensity indicate the direction and distance to obstacles. Since readings from sonars come in polar coordinates (ρ_i, θ_i), the cartesian components x_r and y_r of the resulting vector are calculated as:

$$x_r = \sum_{i=1}^{n} f(\rho_i) \cos\theta_i \qquad y_r = \sum_{i=1}^{n} f(\rho_i) \sin\theta_i$$

where n is the number of samples returned by the sonar sensor and f is the function plotted in Figure 4.

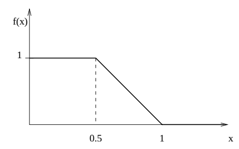

Fig. 4. Sonar weight function, where x is in meters. The x axis is directed along the heading of the robot and the y axis is perpendicular and positive to the left.

5 Robot Team Experiments

The proposed framework has been implemented and tested in both indoor and outdoor environments using teams of three to five robots. Figure 5 shows the robots performing these behaviors in an outdoor grassy environment, while Figure 6 shows the robots performing these behaviors in an outdoor gravel environment. Figure 7 gives an example of the robot state changes that occur to maintain formations when the robots encounter obstacles within their formation. In this figure, all three robots are initially in the *Robot-Follow* behavior. Then, at time $T0$, Robot 1 encounters an obstacle that puts it into the *Robot-Local-Wait* behavior, causing the other two robots to enter the *Robot-Remote-Wait* behavior. At time $T1$, after waiting a period of time for the obstacle to leave but with the obstacle still in the way, Robot 1 enters the *Robot-Remote-Wait* behavior, causing the other two robots to enter the *Robot-Remote-Recover* behavior. At time $T2$, Robot 3 itself then encounters an obstacle, causing it to go into the *Robot-Local-Wait* behavior. When that obstacle does not move, Robot 3 enters the *Robot-Local-Recover* behavior at time $T3$. However, since Robot 1 had not yet completed moving around its obstacle, it also enters the *Robot-Local-Recover* behavior at this time. Robot 2 enters the *Robot-Remote-Recover* behavior. Then, at time $T4$, Robot 1 successfully passes its obstacle and moves to the *Robot-Remote-Recover* behavior to wait on Robot 3 to complete the bypass around its obstacle. At time $T5$, Robot 3 completes its obstacle bypass, and all robots return to the *Robot-Follow* behavior. The introduction of the *wait* behavior proved to be effective in reducing the number of recover stages, where it is more difficult to both go around an obstacle and to keep track of the leader.

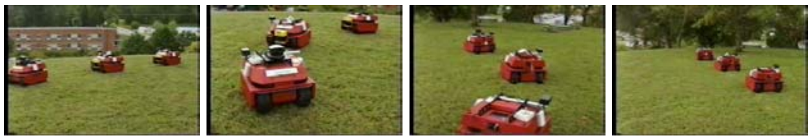
Fig. 5. Results of approach implemented on robots operating in an outdoor grassy environment.

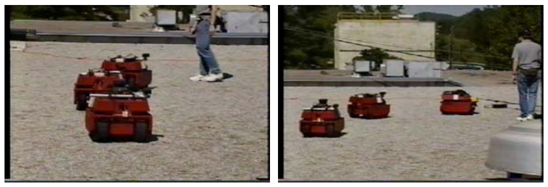
Fig. 6. Results of approach implemented on robots operating in an outdoor gravel environment.

6 Conclusions

A distributed technique for the coordinated motion in a linear pattern of a multi-robot team has been illustrated. The framework, based on explicit anonymous broadcast communcation, is fully scalable with the size of the team and deals with communcation failures. This approach allows a team of robots to remain in a linear formation even when dynamic obstacles appear in the path. The proposed approach has been implemented and validated on a heterogeneous multi-robot team performing in both indoor and outdoor environments.

Acknowledgments

The authors thank Hunter Brown for his work in developing the visual tracking algorithm. This research was supported in part by the Italian Ministry of Education Scientific Research and in part by the Engineering Research Program of the Office of Basic Energy Sciences, U. S. Department of Energy. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U. S. Government purposes. Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the U.S. Dept. of Energy under contract DE-AC05-00OR22725.

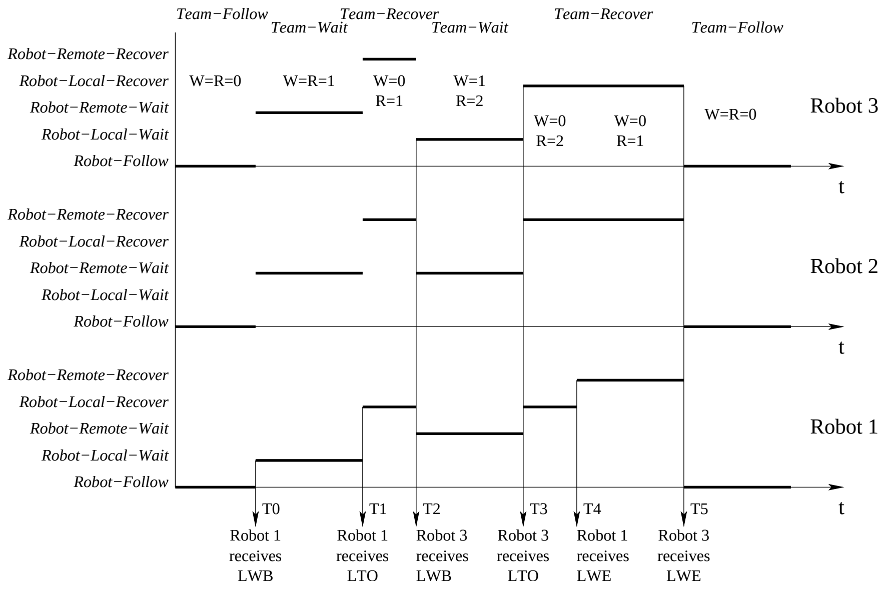

Fig. 7. An example of local behavior scheduling in the case of a three robot team.

References

1. T. Arai, J. Ota. "Dwarf intelligence - A large object carried by seven dwarves". *Robotics and Autonomous Systems*, 18(1-2):149-155, 1996.
2. T. Balch, R.C. Arkin. "Behavior-based formation control for multirobot teams". *IEEE Transactions on Robotics and Automation*, 14(6):926-939, 1998.
3. G.Beni, S. Hackwood. "Coherent Swarm Motion under Distributed Control". *Proc. 1992 Symp. on Distributed Autonomous Robotic Systems*, 39-52.
4. D.C. Brogan, J.K. Hodgins. "Group Behaviors for Systems with Significant Dynamics". *Autonomous Robots*, 4(1):137-153,1997.
5. J.P. Desai, V. Kumar, J.P. Ostrowski. "Control of changes in formation for a team of mobile robots". *Proc. of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1556-1561.
6. R.M. Haralick. "Some Neighborhood Operators". In *Real-Time Parallel Computing Image Analysis*. M. Onoe *et al.* (Eds.), Plenum, New York, 1981.
7. P.J. Johnson, J.S. Bay. "Distributed Control of simulated autonomous mobile robot collectives in payload transp.". *Autonomous Robots*, 2(1):43-63, 1995.
8. L. Kalebling, S. Rosenschein. "Action and Planning in Embedded Agents". In *Designing Autonomous Agents*, ed. P. Maes, MIT Press, Cambridge, MA, 1990.
9. L.E. Parker. "Designing Control Laws for Cooperative Agent Teams". *Proc. 1993 IEEE Int'l. Conf. on Robotics and Automation*, Vol.3, pp. 582-587.
10. L.E. Parker. "Current State of the Art in Distributed Autonomous Mobile Robots". In L.E. Parker, G. Bekey, J. Barhen (Eds.), *Distributed Autonomous Robotics Systems 4*, pp. 3-12, Springer, 2000.
11. J.H. Reif, H. Wang. "Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots". K. Goldberg (Ed.) *Algorithmic Foundations of Robotics*, AK Peters, 1995, pp. 331-336.
12. F.E. Schneider, D. Windermulth, H.L. Wolf. "Motion coordination in formations of multiple mobile robots using a potential field approach". In L.E. Parker, G. Bekey, J. Barhen (Eds.), *Distributed Autonomous Robotics Systems 4*, Springer, 2000, pp. 305-314.
13. H. Yamaguchi, T. Arai, G. Beni. "A distributed control scheme for multiple robotic vehicles to make group formations". *Robotics and Autonomous Systems*, 36(4):125-147, 2001.