Robot Formations as an Emergent Collective Task using Target-following Behavior

Diego Ariel Bendersky and Juan Miguel Santos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Cdad. Universitaria, Pabellón I
(1428) Cdad. de Buenos Aires, Argentina
{dbenders,jmsantos}@dc.uba.ar

Abstract

Robot formations imply the establishment and the maintenance of a predetermined geometric shape by a group of robots. In this work, we achieve formations using robots with proximity sensors of short detection range and no inter-robot communication equipment. This fact turns the synthesis problem into a hard one compared to those cases where the robot has information about the absolute location of the other robots, or has sensors with a higher range limit. The robot formations emerge from an individual behavior called *target-following* which we synthesize using Reinforcement Learning (RL). We propose a task decomposition technique based on an action space transformation that allows us to reduce considerably the time needed for learning and overcomes some difficulties that arise with the use of RL for this particular problem. Preliminary results support the feasibility of this method for the synthesis of robot formation behaviors of different shapes.

Keywords: Robot-formations, Target-following-behavior, Reinforcement-Learning, Task-decomposition, Action-space-transformation

1 Introduction

Robot formation is a multi-robot application that has been used extensively as a testbed for different control algorithms. It consists in the establishment and the maintenance of a predefined geometric shape while moving. From classical AI to behavior-based, from genetic algorithms to neural networks, from distributed to centralized approaches, several control algorithms were tested using this application, because it is simple, yet it allows great flexibility in the definition of the task. Furthermore, it can be used to test different aspects of the control algorithms. Also, robot formations impose different problems and challenges to the developer depending on the sensing and acting facilities of the robots. Global Positioning Systems, centralized control, inter-robot communication and precise sensors, all make the problem easier to solve, and allow greater level of sophistication in the resultant behavior. The goal of this project is to obtain robot formation behaviors of different shapes using miniature autonomous robots with infra-red (IR), low range proximity sensors as the only sensing equipment and no inter-robot communication. At a first glance, the fact that only IR sensors are available makes the problem harder to solve compared with the other cases mentioned (GPS or high range sensors). The robot has high chances of loosing its target and thus a single bad chosen action on a single robot can break the formations. Also, IR sensors are less accurate than other kind of sensors and they are sensible to the color of the target, its covering material and the ambient light. They are also less uniform, in the sense that two different sensor units may return different values for the same stimuli. But IR sensors have several advantages too. They are fast, small and cheap, they require very little computing power and, despite of their imprecision, they can work on a variety of conditions, even in cases where other sensors, such as cameras, do not work

at all (for example, in a poorly illuminated room). Without central control and inter-robot communication, formations emerge as collective behaviors from the activities of each robot. To this purpose, we develop an individual behavior called *target-following*. This behavior consists in the detection of a mobile target and the maintenance of a bounded distance from it over time.

To synthesize this behavior, we use Reinforcement Learning (RL). This robot learning technique is suitable for this problem because: 1) there is no easy mapping between the information given by the IR sensors and the state variables needed to develop a model of the problem (position of the robots or distance and angle), and 2) the amount of sensor information involved and the complexity of the situation and action spaces turn very difficult to get a representative base of examples needed for the use of supervised learning. But using RL in this problem has some limitations. As the scope of the sensors is limited, the probability of approaching a mobile target during the exploration phase is low, and therefore the probability that the robot experiences actions with an associated reward is low too. Since the robot needs the target to be within its sensing scope to try actions that allow it to maintain a bounded distance, the limitation mentioned above constraints the use of RL for synthesizing the target-following behavior. To overcome these limitations, we have considered the use of task decomposition. We propose a task decomposition method based on a well-known action space transformation that converts the actions of the robot, expressed with a velocity for each wheel, into an action expressed with linear and angular velocities:

$$\begin{aligned} v_{linear} &= (v_{right} + v_{left})/2 \\ v_{angular} &= (v_{right} - v_{left})/L \end{aligned} \qquad (1)$$

where v_{left} and v_{right} are the velocities on the left and right wheels of the robot; v_{linear} and $v_{angular}$ are the linear and angular velocities, respectively, and L is the distance between the wheels of the robot. Therefore, based on this new representation, a natural way to decompose the target-following behavior is to think of it as two concurrent ones: the first one maintains the distance between the follower and the target within a given range (*keep-distance*), and the second one keeps the angle between the two robots close to the desired value (*keep-angle*). This method allows us to synthesize *target-following* in very short time lapses. The resultant behavior was tested as an individual robot task under different conditions and proved to be successful. Preliminary results with 4 robots showed that it can be used by a group of robots to do formations of different shapes.

The rest of the paper is organized as follows. In section 2 we review previous work on robot formations and task decomposition. Section 3 shows a detailed description of our decomposition approach and how it overcomes the problems of RL. We show in section 4 the experimental results on the synthesis of *target-following*, its use as an individual behavior and preliminary results on robot formations. Finally, section 5 gives conclusions and future work.

2 Previous Work

2.1 Robot Formations

As we have mentioned, a variety of approaches have been used to obtain robot formations. In this section we mention the works that are more relevant to our project. Fredslund and Mataric [1] proposed the use of local sensing for the synthesis of robot formation behaviors of different shapes (such as column, row, diamond, triangle, etc). In their work, each robot has to follow another robot (its friend) at a given distance and with a given angle, a behavior similar to *target-following*. To achieve this, each robot measures the distance and angle to its target with a laser sensor and a color camera with pan capabilities. The camera computes the angle to the target and the laser gives the distance to it. Another work related to robot formations corresponds to Balch [2]. He presents several strategies, named unit-center-referenced, leader-referenced and neighbor-referenced. The neighbor-referenced strategy has an approach similar to the previous mentioned work: each robot has to maintain a relative position to another robot of the group, which acts as its target. He implemented a motor schema [3] architecture on Nomad 150 robots. In his implementation, each robot estimates its own absolute position using shaft encoders (odometry) and transmits it to its follower robot through a wireless network. Another architecture implemented in this work

is based on a DAMN Arbiter architecture [4]. In this case, each robot obtained the absolute position of its target with the aid of DGPS (differential GPS).

The camera used in the first cited work demands computing processing power to analyze the images, and is more sensitive to light than IR sensors. The laser is more accurate and can sense longer ranges, but it is expensive, bigger, and it requires absolute precision for the steering movements, not needed if a bundle of IR sensors is used. Shaft encoders, used in the second cited work, are sensible to small changes in the arena surface. These errors are accumulative and degrade the behavior performance over time if the robot position estimate is not corrected regularly with some other sensor. Also, transmitting data over wireless networks introduces delays and complexity in the behaviors not present if only local sensing is used. For a more extensive survey on different robot formation approaches, we refer the reader to the work by Fredslund and Mataric [1].

2.2 Task Decomposition

The use of task decomposition is a very common strategy to deal with behavior synthesis of complex tasks. A survey of task decomposition methods for robot behaviors is presented by Jonas Karlsson in [5]. A well-known method for task decomposition is the subsumption architecture [6]. In this method, the designer has to separate the task into smaller ones and develop the relationships among the subtasks by hand (top-down approach). This job is time consuming even for experimented developers and leads to a trial-and-error method, since the only way to assure that a given decomposition solves the task is to test it on a real environment. Other methods for task decomposition are based on the development of a complete behavior through the arbitration of individual and predefined tasks (bottom-up approach). The arbitrator must decide which subtasks should be executed at any time, based on the current situation. In [7] Mataric uses RL to develop an arbitrator for cooperative behaviors (foraging). Maes and Brooks [8] use a method based on preconditions over each subtask, where a subtask is executed if all its preconditions are met. Other popular arbitration technique is Arkin's motor schemas [9]. In this method, each subtask is implemented as a potential field. For every situation, a force vector is obtained from each task. The action that the robot performs is given by a weighted sum of all these vectors. The weights for each task at each situation can be learned and adapted over time.

Our task decomposition method differs from the other mentioned methods in: 1) the developer does not need to decompose the task by hand; 2) there are no parameters to adjust for the recomposition of the original task; and 3) an arbitrator is not necessary, since the subtasks are executed concurrently.

3 Synthesis of the Target-following Behavior

3.1 Formal Description of the Problem

3.1.1 Definition of Robot Formations

A group of robots are in formation if they can achieve and maintain a predefined geometrical shape while moving with a specified direction. Different formation shapes can be considered. The formations that are most mentioned in the literature are column, row, diagonal and triangle.

3.1.2 Definition of Target-following

The target-following behavior will be formalized in a parameterized way that allows the specification of different formations. We say that the follower is following the target if

$$||p_f - p_t|| = d \pm \epsilon_d \text{ and } \alpha(p_f - p_t) = \theta \pm \epsilon_\theta. \quad (2)$$

where p_f and p_t are the positions of the follower and target robots respectively and $\alpha(p)$ is the angle of the vector p (i.e. the angle component of the vector p when expressed in polar coordinates). d and θ are the optimal distance and angle between the two robots, and ϵ_d , ϵ_θ their respective maximum errors. This definition is static; it only takes into account the instantaneous position of the robots and not the direction or the velocity of their movements. With different values for the optimum angle parameter, we can create different shapes for the formations. For example, column formation emerges if we use $\theta = 0°$. For diagonals, we

should use $\theta = 45°$, and for columns $\theta = 90°$. For triangle shapes, we should use different parameters for different robots: half of the robots must obey the formula with $\theta = -45°$ and the other half with $\theta = 45°$. In a first stage we will use the definition with $\theta = 0°$. This gives us a simple scenario to test *target-following* as an individual behavior (it forces the follower robot to always align its front with the target). Also, the formation emerged (column formation) is the simplest and easier to test.

3.1.3 Task Decomposition

As stated on the introduction, our approach comprises an action space transformation: the actions, originally expressed by an independent velocity for each wheel (v_{left}, v_{right}), are transformed in tuples with linear and angular velocities $(v_{linear}, v_{angular})$ using equation 1. We then create an independent subtask for each component of the tuple: one, called *keep-distance*, maintains the distance to the target within a bounded range, and another, called *keep-angle*, maintains the two robots aligned. Both subtasks will be active and executed concurrently all the time. Thus, the action to be carried out by the robot will be the combination of the action of each subtask. *Keep-distance* and *keep-angle* are both developed as behaviors. The inputs for both behaviors are the raw values of the IR sensors. The outputs are the distance to move in straight line (for *keep-distance*) and the angle to rotate (for *keep-angle*). As we have used RL for the synthesis of both behaviors, we had to define a reinforcement function (RF) for each task. The RF used for the learning of *keep-distance* is

$$kdrf(s_0..s_n) = \begin{cases} 1 & \text{if } |s_{front} - 0.5| < \sigma_1 \\ 0 & \text{if } \sigma_1 \leq |s_{front} - 0.5| \leq \sigma_2 \\ -1 & \text{if } \sigma_2 < |s_{front} - 0.5| \end{cases} \tag{3}$$

where $0 \leq \sigma_1 \leq \sigma_2 \leq 1$, and s_{front} is the sensor that should be aligned with the target. If the angle that the robots should maintain lies between two sensors, then its average should be taken. For example, for column formations, $s_{front} = |(s_2 + s_3)/2|$. For diagonal formations, $s_{front} = |(s_0 + s_1)/2|$ (see figure 2 for a diagram of the robots, with the position of each sensor). These equations define five areas from the values of the sensors. The robot receives positive reinforcement if the value of the front sensor lies inside the central area (medium distance), zero if it lies near it, and negative if it is too big (collision) or too small (far away). See figure 1.

For *keep-angle*, the definition of the function varies with the shape of the formation, and the place of each robot in it. At this moment, we have tested two reinforcement functions: the first one forces the follower robot to align its front with the target, producing column formations when used by multiple robots. The second reinforcement function tested, allowed us to make preliminary studies on diagonal formations. Notice that our robots have 8 sensors.

$$kacolrf(s_0..s_n) = \begin{cases} 1 & \text{if } (s_2 \geq s_i \vee s_3 \geq s_i) \wedge \\ & \quad |s_2 - s_3| \leq \epsilon \\ 0 & \text{if } (s_1 \geq s_i \vee s_4 \geq s_i) \vee \\ & \quad ((s_2 \geq s_i \vee s_3 \geq s_i) \wedge \\ & \quad |s_2 - s_3| > \epsilon) \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

$$kadiagrf(s_0..s_n) = \begin{cases} 1 & \text{if } (s_0 \geq s_i \vee s_1 \geq s_i) \wedge \\ & \quad |s_0 - s_1| \leq \epsilon \\ 0 & \text{if } s_2 \geq s_i \vee \\ & \quad ((s_0 \geq s_i \vee s_1 \geq s_i) \wedge \\ & \quad |s_0 - s_1| > \epsilon) \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

where $0 \leq i \leq n$ and $0 \leq \epsilon \leq 1$.

In the first equation, if the front sensors have both high values (one has the highest and the difference with the other is low) then the robot will receive a positive reinforcement. If the sensor with maximum value is near the front (or the two front sensors have very different values), then the robot will receive a null reinforcement. If the sensor with the maximum value is a lateral one or is on the back of the robot, then the robot will receive a negative reinforcement. The second equation differs from the first one in the use of the left sensors instead of the front ones.

Once we have obtained these two mappings, we use the following transformation to obtain the action needed for *target-following*:

$$\begin{aligned} v_{linear} &= \frac{dp}{t} \\ v_{angular} &= \frac{\alpha \frac{\pi}{180}}{t} \end{aligned} \tag{6}$$

where d is the result of *keep-distance*, α is the result of *keep-angle*, t is the time needed to perform the actions, and p is the distance (in mm) of one wheel encoder unit.

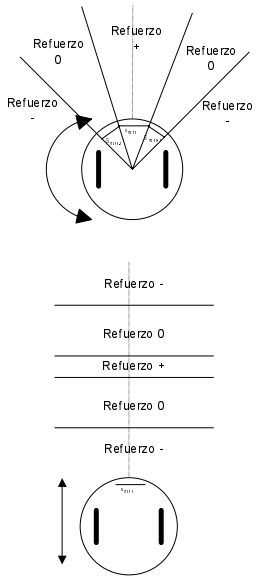

Figure 1: Definition of *keep-distance* and *keep-angle* as deduced from the reinforcement function.

The transformation from $(v_{linear}, v_{angular})$ to (v_{left}, v_{right}) can be derived from equation 1.

$$\begin{aligned} v_{left} &= v_{linear} - L/2\ v_{angular} \\ v_{right} &= v_{linear} + L/2\ v_{angular} \end{aligned} \tag{7}$$

In summary,

$$a = (v_{left}, v_{right}) = (\frac{k_1 d - k_2 \alpha}{v_p}, \frac{k_1 d + k_2 \alpha}{v_p}) \tag{8}$$

where $k_1 = \frac{p}{t}$, $k_2 = \frac{L}{2}\frac{pi}{180}\frac{1}{t}$ and v_p is the velocity of one wheel velocity unit.

In the following section we analyze the consequences and advantages of task decomposition, and we describe the environment used for the learning and the clustering technique used for the representation of the q-function.

3.2 Task Decomposition and the Learning Environment

The task decomposition we implement in this work has the advantage that both, the state space and the action space, will be smaller compared to the case in which no decomposition is used. The action space is smaller because v_{linear} represents a forward or backward movement which implies the same speed for the two wheels, and $v_{angular}$ represents a rotation which implies same magnitude but opposite signs for the wheel velocities. Since the learning time for RL algorithms grows with the size of the situation/action space [11] a considerable speed-up is gained with this space reduction. Another method used to speed-up learning in RL is the design of simplified learning sessions [12]. Although our task involves following a mobile target, we propose a learning environment with a non-mobile target robot, fixed at a position that is optimal for the formation we want to learn. For example, for column formations, the target robot is placed with its back aligned with the follower's front, and for diagonal formations, the target is placed at 45° from the follower. With this environment, the task can be learned faster because: 1) the situation space is reduced, 2) it is easier for the follower robot to obtain positive rewards during learning, and 3) the learning process involves more immediate rewards. The state space is reduced because, in *keep-distance*, only the sensors in the target robot direction receive non-zero input and in *keep-angle*, the center of the follower is always at the same distance from the center of the target. By setting the environment in this way, the target is most of the time inside the follower's sensor scope. Therefore, the follower has more opportunities to experience new situations and actions, and thus it will experience more rewards over time. Besides, these rewards are, in general, immediate because the goal state can be reached from almost any other state by executing one or two actions, speeding-up the convergence of the RL algorithm. Additionally, if the target was moving during the learning phase, different situations that require different actions could be perceived as the same situation by the robot, introducing a problem known as perceptual aliasing [13]. This happens because the inputs of the robot are the instantaneous values of the proximity sensors, and not their changes over time. But the simplification of the learning environment has its drawbacks too. It restricts the set of states the robot can reach while learning, imposing an a-priori exploration bias, and affecting the performance of the obtained behavior [14]. For example, while learning *keep-distance* for column formations, the robot will never reach a situation with maximum values on the left sensors, although

this situation can be obtained in the execution of *target-following*. In our approach, we have attenuated or suppressed this bias influence in the obtained behavior with the use of a clustering technique over the situation space (see next section).

3.3 Neural Network Approach to the Representation of the Q-function

Clustering techniques are always a big help for reinforcement learning because they provide generalization, eliminate redundancy and speed up the learning. But in this case, clustering is mandatory because large parts of the state space are not explored during the learning. In this work, we use a modification of RBF (radial basis functions) algorithm, called QRBF. This algorithm was proposed by Santos [15]. It provides a clustering technique for the space formed by the tuples situation-action-q value. The activation region for each cluster is defined by a Gaussian function. The clusters are adjusted during the learning according to the situation-action pairs visited by the robot. If these pairs fall outside the activation region of all the clusters, a new one is created. In short, the network acts as an associative memory: to look for the best action, a partial tuple with a situation and a q-value (the maximum possible) is presented to the network, and a complete tuple, with an optimal action, is given in return. This cluster represents a trade-off between the goodness of the action (its associated q-value) and the similarity between the situation represented by the cluster and the situation obtained by the robot. To update the network while learning, the observed situation and the action actually carried out (that may be different from the optimal action due to the exploration) are presented, and the nearest cluster is updated with the reinforcement value obtained by this pair. We encourage the reader to refer the work of Santos for a detailed description of this algorithm and its motivations.

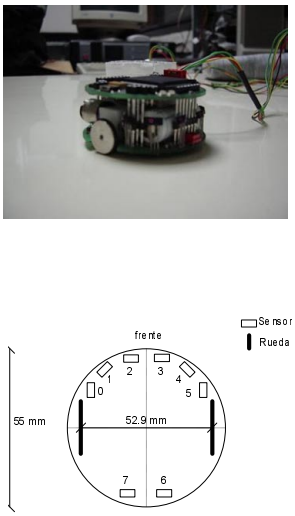

Figure 2: The Khepera robot has two front, two rear and four front-lateral sensors.

4 Experimental Results

4.1 Robot and Environment Description

For all the experiments we used Khepera mobile robots. These are miniature robots with a diameter of 55 mm. They have eight infra-red transmitter/receiver pairs along its perimeter (although not equidistant) which can act as proximity and light sensors with a 10 bit resolution. They can detect obstacles up to a distance of 50 mm, but the shape of the function for these sensors (i.e. sensor value vs. distance) varies with the ambient light and the color and material of the obstacle. They have two independent wheels with motors that can take speeds of up to 1 m/s (with 7-bit resolution). See figure 2. More information on this robot can be found in [16].

In the following sections we present implementation details for the learning and results for several test of the robot-following behavior. In section 4.4 we show preliminary results for column and diagonal formations of 4 robots.

4.2 Learning of Robot-following Behavior

As explained on section 3, we placed the two robots aligned and at a medium distance (between 0.5 and 2 cm) at the beginning of the

learning of both *keep-angle* and *keep-distance*. During learning, the actions carried out by the robot were uniformly chosen from an interval centered on the action that maximizes the q-value for the current situation. The size of this interval is reduced linearly over time. In this way, at the beginning almost any action could be carried out, providing a good source for exploration. At the end only optimal actions were chosen (exploitation). We use a QRBF neural network for the representation of the q-function. The output of the network (a number between 0 and 1) was transformed to an angle from -90° to 90° for keep-angle behavior and to a distance from -8 mm to 8 mm for keep-distance behavior. The parameters of the networks were adjusted by hand (QRBF parameters for learning keep-distance behavior were: $\eta_a = 0.1$, $\eta_s = 0.01$, $\eta_q = 0.5$, $acceptance = 0.1$, $\sigma^2 = 0.3$ and $\gamma = 0.3$; and for keep-angle behavior were $\eta_a = 0.1$, $\eta_s = 0.01$, $\eta_q = 0.7$, $acceptance = 0.08$, $\sigma^2 = 0.6$ and $\gamma = 0.5$. Consult references for the meaning of these parameters). For the keep-distance behavior, the parameters of the reinforcement function were adjusted by hand, analyzing the proportion of positive rewards during the learning. The values actually used were: $\sigma_1 = 0.3$ and $\sigma_2 = 0.4$. For *keep-angle*, the parameter ϵ was set to 0.4. We carried out 6 different learning sessions for both *keep-angle* and *keep-distance*. Each learning session consisted in 300 iterations. It took 3 minutes approximately to execute each *keep-distance* session and 5 minutes for *keep-angle*. Figure 3 shows the best action for each situation. Both the situation and the action is expressed in mm for *keep-distance* and in degrees for *keep-angle*.

4.3 Target-following Behavior Results

To test the target-following behavior, we carried out different sets of experiments. In all of them, we placed the two robots aligned at 30 mm from each other, pointing in the same direction. On each series, the target robot followed a particular trajectory with a fixed velocity, always lower than 30 mm/s. Trajectories used where: circular paths (of different radius), rectangular paths and random paths (straight movements with random rotations at fixed intervals). We performed 4 to 6 experiments for each set,

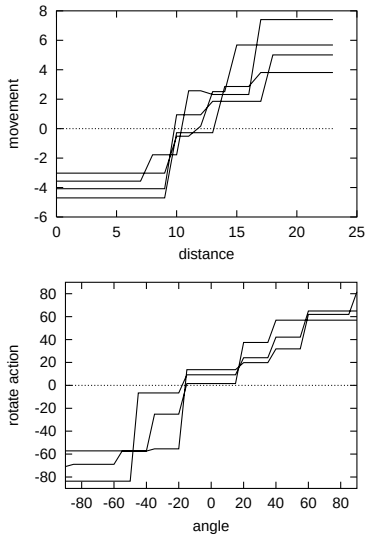

Figure 3: *Keep-distance* and *keep-angle* results. On the top figure, the x axis represents the distance to the target, and the y axis the movement executed by the robot (in mm). On the bottom figure, the x axis represents the angle to the target, and the y axis the best action expressed in degrees.

measuring the value of the two front sensors over time. Figure 4 shows the trajectories of both the target and the follower for each test set. For each robot, a line is drawn every 100 msec. These lines are parallel to the axis of the wheels (i.e. perpendicular to the direction of the movement). The length of the line has no relation with the diameter of the robot, which is much bigger. The darker lines represent the predator robot and the lighter ones, the pray.

Table 2 shows, for each set of experiments, the percentage of the time that the value of the front sensors produces low, medium and high values. With circular paths of big diameter (low rotation angle), the predator follow almost exactly the trajectory of the pray. With higher rotation angles, the predator performs shorter movements and in-place rotations. In the other two examples, we can see that the movements of the predator are, in general, smoother than the movements of the pray. When performing the same experiments with higher velocities (more than 35-40 mm/s), at some point of the trajectory the follower robot lost its target. We could obtain the correct behavior multiplying the velocity of the follower by a predefined factor, but in this case the quality of the

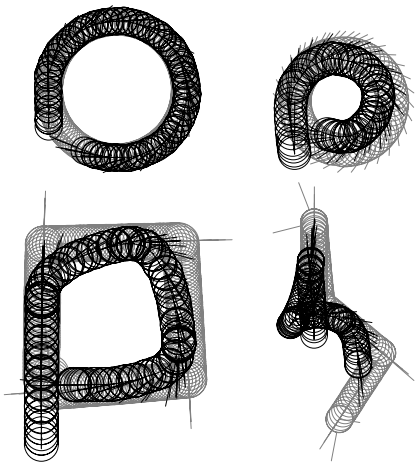

Figure 4: Target-following: Trajectories followed by both robots in different test cases. The light circles represent the target and the dark circles represent the follower.

Set	< 0.1	$0.1-$ 0.2	$0.2-$ 0.8	$0.8-$ 0.9	> 0.9
1	1%	9%	89%	1%	0%
2	8%	5%	81%	4%	2%
3	12%	9%	63%	6%	10%
4	18%	11%	53%	5%	13%

Table 1: Distribution of values for the front sensors in *target-following* tests.

behavior was lower (for example, the follower made oscillatory movements in some parts of the trajectory).

4.4 Formation Results

To test robot formations, we put 4 robots in a column. The same trajectories used in the previous section were used in one of the robots (the leader) and the target-following behavior was used to control the other 3 robots. Preliminary results show that the 4 robots achieve to maintain the formation when the leader followed a circular path with a very low velocity. We can see the results of this experiment in table 3. This table shows the percentage of the time that the front sensors take different values. From this table, we can see that, although all the robots succeed to maintain the formation, the quality of the behavior is lower for the robots on the back of the formation.

Figure 5: Trajectories for column and diagonal formations of 4 robots.

Robot	< 0.1	$0.1-$ 0.2	$0.2-$ 0.8	$0.8-$ 0.9	> 0.9
1	0%	2%	98%	0%	0%
2	1%	9%	87%	1%	2%
3	3%	23%	66%	3%	5%
1	6%	5%	88%	1%	1%
2	16%	8%	71%	0%	4%

Table 2: Distribution of values for the relevant sensors on each robot of the formation. The first series is for column formation on circular paths, and the second series is for diagonal formation on straight walk.

The other formation geometry we tested was diagonal shape. In this case we re-learn the target-following behavior, but using other reinforcement functions (see section 3.1). The results are preliminary. The experiment consisted in 3 robots that, starting in column, accommodate themselves diagonally while moving forward, following the leader. Table 3 shows the results (in this case the lateral sensors were considered, instead of the front sensors). Although the robots succeeded to reach their goal in this very simple experiment, the quality of the behavior obtained was lower. The robots performed excessive oscillatory movements while moving, failed to deal with a large set of situations, and lost their target easily.

5 Conclusions and Future Work

The aim of this project is to obtain formations of different shapes using miniature robots with limited sensing capabilities and

no communication equipment. With these limitations, formations should arise from individual behaviors. We synthesized a behavior called *target-following*, decomposing it in two simpler behaviors: *keep-distance* and *keep-angle*. Our method of decomposition and re-composition of the task does not need intervention from the developer, and it is guaranteed that, if the basic behaviors solve their respective subtasks, then the complete task will be solved too. This method speeded-up the learning time considerably: in all cases, the synthesis of the behavior did not take more than a few minutes.

After that, we use the target-following behavior to control several robots, in order to obtain formations. We successfully obtained a column formation of robots following a circular path, and a diagonal formation of robots moving forward. These preliminary results support the feasibility of this approach, and show that robot formations of different shapes can be obtained using low-range sensors and no inter-robot communication.

Several aspects of this method can be improved in order to obtain more robust and flexible behaviors. The behavior obtained was very sensible to the velocity of the target. It worked well with low velocities, but we had to increase the velocity response (multiplying the velocity module by a factor greater than to one) to allow the robot to follow a faster target, at the expense of a degradation of the quality of the behavior. The diagonal target-following behavior obtained has several problems and did not work very well. We are now analyzing the reasons of this fact and we are testing several ways to improve the behavior. Finally, we are investigating different approaches to deal with other aspects of real world formations, such as obstacle avoidance.

References

[1] J. Fredslund and M. Mataric: A General, Local Algorithm for Robot Formations. Transactions on Robotics and Automation, special issue on multi-robot systems (in press).

[2] T. Balch: Behavior-based Formation Control for Multi-Robot Teams. IEEE Transactions on Robotics and Automation, Vol. XX, No. Y, 1999.

[3] R. Arkin: Motor schema based mobile robot navigation. International Journal of Robotics Research, vol. 8, no. 4, pp. 92-112, 1989.

[4] J. Rosenblatt: Damn: a distributed architecture for mobile navigation. Working Notes AAAI 1995 Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents, Palo Alto, CA, March 1995.

[5] J. Karlsson: Learning to Solve Multiple Goals. University of Rochester, NY, 1997.

[6] R. Brooks and J. Connell: Asynchronous distributed control system for a mobile robot. Proceedings of SPIE, vol. 727, pp 77-84, 1986.

[7] M. Mataric: Reward functions for accelerated learning. Proceedings of the Eleventh International Machine Learning Conference, Morgan Kaufmann Publishers, inc., 1994.

[8] Maes, Pattie and Brooks: Learning to Coordinate Behaviors. Proceedings of AAAI-90. pp. 796-802, 1990.

[9] R. Arkin: Integrating behavioral, perceptual and world knowledge in reactive navigation. Robotics and Autonomous Systems, vol. 6, pp. 105-122, 1990.

[10] R. Arkin: Behavior based robotics. MIT Press, Cambridge, MA, 1998.

[11] P. Kaelbling and A. Moore: Reinforcement learning: a survey. Journal of Artificial Intelligence Research. 4, pp. 237-285, 1996.

[12] A. Bonarini: Anytime learning and adaptation of structured fuzzy behaviors. Special Issue of the Adaptive Behavior Journal, no. 5, 1997.

[13] R. Matuk and J. M. Santos: The Clustering aliasing problem in Reinforcement Learning for robots. Proceedings of the Fifth European Workshop on Reinforcement Learning, pp. 33-35, Utrecht, The Netherlands, 2001.

[14] C. Touzet and J. M. Santos: Reinforcement Function Design and Bias for Efficient Learning in Mobile Robots. Proceedings of the World Congress

on Computational Intelligence, FUZZ-IEEE'98, pp. 153-158, Anchorage, Alaska, 1998.

[15] J. M. Santos: Contribution to the study and the design of reinforcement functions. PhD Thesis, Universidad de Buenos Aires and Université d'Aix-Marseille III, 1999.

[16] F. Mondada, E. Franzi and P. Ienne: Mobile robot Miniaturization: A Tool for Investigation in Control Algorithms. Proceedings of the Third International Symposium on Experimental Robotics, Kyoto, Japan, 1993.