2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Implicit robot coordination using Case-Based Reasoning behaviors

J.M. Peula, C. Urdiales, I. Herrero and F. Sandoval

Abstract— **Multi-robot systems (MRS) are a very active and important research topic nowadays. One of the main problems of these systems is the large number of variables to take into account. Due to this, robot behaviors are sometimes learnt instead of calculated via analytical expressions. A typical learning mechanism, specially for biomimetic robots, is Learning from demonstration (LfD). This paper proposes a LfD approach for implicit coordinated navigation using combination of Case-Based Reasoning (CBR) behaviors. During a training stage, CBR is used to learn simple behaviors that associate positions of other robots and/or objects to motion commands for each robot. Thus, human operators only need to concentrate on achieving their robot's goal as efficiently as possible in the operating conditions. Then, in running stage, each robot will achieve a different coordinate navigation strategy depending on the triggered behaviors. This system has been successfully tested with three Aibo-ERS7 robots in a RobCup-like environment.**

I. INTRODUCTION

Multi-robot systems (MRS) have been applied in a wide range of different scenarios. The reason is that MRS can perform a given task faster and more effectively [1] than a single robot. However, coordination between robots can be difficult to accomplish.

Although there is a wide variety of coordination strategies [2], they can usually be divided into two groups [3]: explicit coordination [4] and implicit coordination [5]. In explicit coordination, robots communicate with each other in order to explicitly coordinate their actions –normally via deliberative planning–. On the other hand, in implicit coordination each robot acts on its own taking into account the existence of the others and even sharing information between them –where they are or what they see–, but without explicitly making decisions together. In this case, coordination is normally achieved in the form of an emergent behavior, arisen from the combination of simpler ones [6]–[8]. This kind of coordination is easier to achieve through training, as it only requires to train a robot simultaneously, while the others act on their own.

Independently of the strategy, coordinated systems are based normally on hierarchical hybrid planning in order to deal with the complexity of the environment in real time. Specifically, most works on legged robots rely on behavior based architectures [9], which are typically built in a bottom-up way. The key to these approaches is how to develop these low level behaviors and how to arbitrate them.

Low level behaviors can be achieved using analytical equations or via behavior learning. Analytical equation is a typical solution that provides the same behavior for similar input instances. However, in order to get different or specific responses it is necessary to rewrite the equations, what can be difficult if they depend on a large number of variables. Another option is to learn the desire behavior [10]. The main advantage of this approach is that in order to get different responses, it is only necessary to train again the system or to mix different trainings.

Thus, learning approach can be a more flexible and easier way to perform different, specific responses – comparing to the use of analytic equations–. Although there are different machine learning techniques, one of the most typical ones is Learning from demonstration (LfD) [10]. This technique associates state/action pairs and makes possible to train a robot behavior from examples provided by a supervisor. LfD includes a variety of machine learning strategies [32] such as case-based learning, decision-tree learning, bayesian learning, etc. Each strategy has advantages and drawbacks, however we have chosen Case-Based Reasoning (CBR) because it is very intuitive for humans since it is tightly related to the way humans reason.

This paper presents a LfD-based approach to develop complex coordinated navigation using simple learnt behaviors. Thus, a complex behavior such as get a ball and score avoiding the opponents, is divided in simple behaviors and trained independently. These simple behaviors associate positions of other robots and/or objects to motion commands while training. Robot training is made via teleoperation, which has the main advantages [7], [11]: i) no explicit kinematics model of the robots is required, ii) systematic sensor and mechanical errors are implicitly absorbed by the model, and iii) the state/action pairs are directly recorded without need of any conversion function. Hence, during training CBR couples human's commands at a given circumstance with the robot perception. Thus, if robot perception includes positions of other robots and/or objects, it will learn what the person does in that situation and, implicitly, coordinate navigation.

Thus, this approach is suitable for complex coordinated behaviors that are difficult to train or to implement analytically but easy to divide into the combination of simple behaviors that are easier to train than to implement analytically. Moreover, it is specially interesting in case of legged or heterogeneous robot teams, as kinematics and errors are implicitly absorbed thanks to the teleoperation training.

The paper is structured as follows. First, robot localization algorithm is explained in section II. Then, sections III-A and III-D presents behavior definition and learning for each basic behavior. Once simple behaviors have been trained, they are combined to perform an emergent coordinated behavior –

J.M. Peula, C. Urdiales, I. Herrero and F. Sandoval are with Department of Tecnología Electrónica, E.T.S.I, Telecomunicación, Universidad de Málaga, Campus de Teatinos S/N, 29071 - Málaga (Spain). peula, acurdiales, nhr, fsandoval@uma.es

section III-E–. Finally, section IV shows the experiments using three Aibo-ERS7 robots and section V presents our conclusions and future work proposal.

II. Visual localization

In robot coordination and navigation problems the robot usually requires to know its relative position with respect to the target, obstacles and other teammates in the environment. There are many different localization algorithms depending on the problem to be solved and, more specifically, on the existence of artificial landmarks. In our case, our test environment is a RoboCup-like soccer field. In RoboCup competition, specially when using legged robots, localization is usually visual as the field includes color landmarks at known positions that the robot may use for global localization. Thus, our localization algorithm is based on artificial visual landmark detection [7] [12].

The most frequent localization techniques lately are based either on using probabilistic techniques such as *Kalman filters* [13] or *particle filters* [14] to better estimate the robot position. We have chosen a particle filter, that is a probabilistic method to estimate the state of a system at a certain time t based on current and past measurements. The probability ($p(X_t|Z_t)$) of a system being in the state X_t, given prior measurements ($Z_t = z_0, ..., z_t$), is approximated by a set of N weighted particles ($S_t = \{x_t^{(i)}, \pi_t^{(i)}\}, i = 1...N$). Each particle x_t describes a possible state with an associated weight ($\pi_t^{(i)}$), which is proportional to the likelihood that the system is in this state. Thus, the current location (state) of the robot is modeled as the density of a set of particles whose weights depend on the landmarks observed by the robot.

III. Coordination by learning

The main objective of this paper is to achieve implicit coordinated navigation between different robots using LfD. As commented by different authors [6]–[8] a complex behavior can be modeled via the combination of simple ones. The main advantage of the use of basic behaviors is that they are less sensitive to sensor errors, more generic, easier to develop and test and adapt better to most situations and environments [15]. If more efficient and complex behaviors are required, the system can be hybridized via a higher level control layer (e.g. [16]) to modulate a proper triggering sequence.

The main issue to achieve a complex coordinated behavior is how to divide it into simple behaviors. In our soccer problem, we will use an example with two attackers from one team that need to coordinate to score in the opponent's goal while avoiding the opponent's team defender. Analytically, the task could seem easy to accomplish. However, We have choose this example because in a two-on-two situation, if both teams have a good coordination algorithm/training, each team advantage will depend more on the robots itself – i.e: one robot is faster– and in hazard factors than in the coordination itself – i.e: a two-on-two situation can become a one-to-one situation if each defender covers one attacker–. Hence, despite the numeric advantage this example shows the importance of coordination. Thus, for solving this example, the attackers have to decide who is going to get the ball, to avoid interfering each other. Once one has the ball, it has to move to the enemy's goal and avoid the defender to score. If the defender is effectively blocking it, it has to pass the ball to its teammate, that should be far from them and in an strategically good position. Once we have decided what the robots have to do, we have to divide it in simple tasks. For example get ball, avoid obstacle, go to the sideline, move in parallel... However, there are a lot of potential problems that robots have to solve, depending on the relative positions of all three robots at a time, as well as their own kinematics and dynamics, mechanical and sensor errors, etc. Hence, low level behaviors are trained using CBR by a person using a joystick to control each robot separately. The main advantage of this approach is that people is naturally good at adapting and improvising. For example, if a robot is limping due to a mechanical problem, the human controller will adapt and correct the trajectory to achieve the required goal in a natural way. Besides, people are also good at dealing with many issues at the time, including the rest of robots and objects in the environment. After supervised training, all robots may run in standalone mode, where the case-base returns the most appropriate learnt case depending on the situation.

Next sections present simple behavior division, codification into CBR cases, training necessary for each behavior and how behaviors are combined and triggered to achieve the expected complex behavior.

A. Simple behavior definition

In this section we show how to divide our example behavior (coordinating two attackers against one defender to score a goal) into simpler ones and train them. First, an important stage in our approach is to determine correctly the necessary basic behaviors. These behaviors should be as generic as possible, so they can be reused in many situations. Hence, instead of mentioning a ball, goal or teammate, our simple behaviors consider an object, that can be a goal, an obstacle or both depending on the behavior itself.

After analyzing the global behavior to be acquired, we noted that we needed the following basic ones:

- Get object: the target of this behavior is just to get to an object (ball, enemy goal, etc.) from the current position.
- Avoid object: the target of this behavior is to avoid a close object (objects, teammates or opponents) in the moving direction.
- Avoid collision: the target of this behavior is to avoid imminent collisions with obstacles (objects, teammates or opponents). This is an emergency behavior that overcomes all the rest.
- Go to sideline: the target of this behavior is to guide the robot through the field sideline to approximate to an objective (enemy's goal for example).
- Move in parallel: the target of this behavior is to guide the robot so that it moves in parallel (keeping its distance) with an object (for example, a teammate).

- Defend object: the target of this behavior is to protect an object from another object (for example, defend goal from an opponent).

A correct combination of these behaviors should return the appropriate complex behavior for each circumstance. From the explained behaviors, the simplest ones –i.e. *avoid collision*– are analytically implemented, whereas the more complex ones are trained, as explained below.

B. Case-Based Reasoning

LfD includes a variety of machine learning strategies [17] such as: case-based learning, decision-tree learning, bayesian learning or evolutionary learning. Each of these learning strategies has advantages and drawbacks. However, we have chosen CBR because it is very intuitive for humans since it is related to the way we reason. In addition to it, CBR has been chosen over other well known techniques such as Artificial Neural Networks [18] because in CBR knowledge is stored in a plain case-base, what makes possible to understand why the robot is acting in a specific way.

CBR is a reasoning, learning and adaptation technique, derived from the theory of the Dynamic Memory [19], that solves problems by retrieving and adapting past experiences, called cases, and proposing solutions for present problems [20]. The essence of CBR is based on the assumption that *similar problems have similar solutions*.

CBR has been applied to different tasks [21] such as classification, diagnosis or planning. In the area of robotics, CBR has been used typically for high level planning [22] such as behavior and action selection [23], [24]. In this paper, however, CBR is used for learning low level behaviors.

The typical CBR cycle, or 4R format, consists of four steps –*retrieve*, *reuse*, *revise* and *retain*–. In our case, however, only two of them – *retrieve* and *reuse* – are used in the CBR cycle. Revise and retain steps have been skipped because it can be a bit difficult to check if the output response corresponds to the expected one if it depends on the average output of several behaviors, running in parallel, that can also adapt their outputs respect to trained one. In any case, and independently of the CBR cycle used, before running CBR in standalone mode it is necessary to define inputs and outputs for the CBR cases and train it.

C. CBR case definition

CBR associates in a case a specific object(s) configuration to the user's command for a particular situation. Hence, CBR fits perfectly the idea of a low level behavior coupling each stimuli —robots' or objects' positions— and action —motion commands— as input/output respectively. At reactive level, it is extremely important to correctly define inputs and outputs for each behavior. In our case, the output for all behaviors is the same –the motion command that the robot has to execute– but the input instance is different for almost all behaviors. Table I summarizes inputs for each proposed behavior and Fig.1 shows a graphical representation of the variables used in the case input instance.

	CBR case inputs			
	In_1	In_2	In_3	In_4
Get object	O_ρ	O_θ		
Avoid object	O_ρ	O_θ		
Go sideline	$\theta_{ref} - R_\theta$	R_Y		
Move in parallel	$\theta_{ref} - R_\theta$	$O_X^2 - R_X$	$O_Y^2 - R_Y$	
Defend object	$O_X^1 - R_X$	$O_Y^1 - R_Y$	$O_X^2 - R_X$	$O_Y^2 - R_Y$

TABLE I

CBR CASE INPUTS FOR EACH CBR BEHAVIOR

(R_X, R_Y, R_θ) being the absolute cartesian coordinates of the robot and its orientation, (O_ρ, O_θ) the relative polar coordinates of the object respect to the robot position, (O_X^N, O_Y^N) the absolute cartesian coordinates of the object N and θ_{ref} the reference angle for the robot to move in that direction –0° for moving to the right and 180° to the left–. All distances are measured in centimeters and angles in degrees. It can be observed that *get object* behavior only depends on the relative polar position –respect to the robot– of the object to get (ball, goal, etc.) whereas *defend object* behavior depends on the relative cartesian position –respect to the robot– of the object to defend –object 1– and the object that *attacks* –object 2–. Note that robots share their positions calculated via a particle filter as commented in section II.

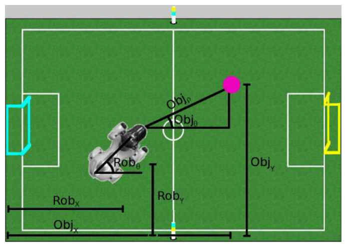

Fig. 1. Representation of the variables used as inputs in CBR cases.

D. Simple behavior learning

As commented, CBR acquires stimuli/perception pairs that gather the experience of a human. In order to acquire this information, we propose a supervised CBR training. We have divided our training stage in two steps: information gathering and CBR case-base extraction.

During the information gathering stage, a human teleoperates the robot to achieve the desired basic behavior –i.e. *get object*– while information –the current joystick command and the input instance– is stored each 100ms. This learning stage needs to be performed for each behavior to acquire. Time required for each training depends on the training itself, but in general it takes 5-10 minutes. During each training the robot learns how the human behaves in each situation taking into account, implicitly, the robot specifics (dimensions, kinematics, dynamics, sensor errors...). Obviously, the human should try to cope with all expected situations in each behavior training. It should be noted that the training depends on the trainer. Thus, the better the supervisor skill the better the training will be. However, as most of behaviors are very simple, non-optimal trainings will only result in more erratic emergent behavior. Thus, the most important issue, at this point, is to define each behavior as simple and clear as possible to avoid redundancy, ambiguity and noise in acquired information. Fig.2 shows the different trajectories perform during each of the proposed behaviors' trainings in our example. Although the more the robot is trained, the better the acquired behavior, these trajectories have proven to be enough to achieve the desired ones.

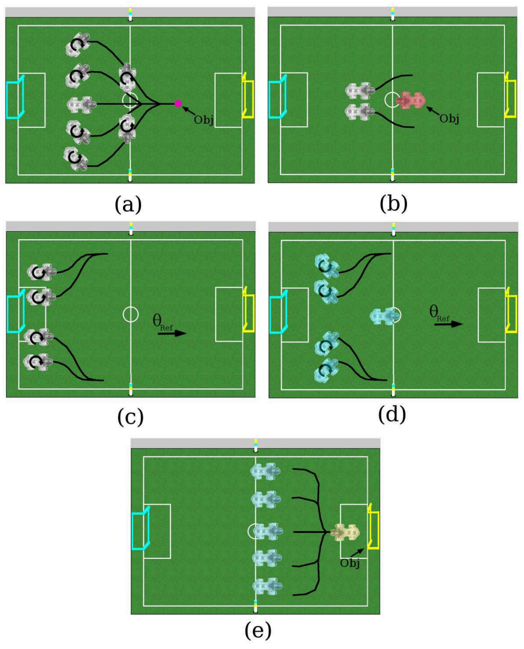

Fig. 2. Trajectories perform during training sessions for each of the behaviors: a) get object, b) avoid object, c) go sideline, d) move in parallel and e) defend object.

It can be observed in Fig.2.a that the robot is trained to reach the object –in this case a ball– from different positions. Fig.2.b shows how the robot is trained to avoid an object –in this case a robot– in the moving direction. Fig.2.c shows a positioning training, in which the robot is trained to go to the opponent field area following the field sideline. In Fig.2.d we can observe how robot is trained to move in parallel and at a certain distance from a reference object –a robot–. And finally, Fig.2.e shows how the robot is trained to defend an object –goal– by blocking another object –an enemy robot– to prevent it from reaching the first object –goal–.

After training, gathered information is processed to extract CBR cases and generate the different CBR case-bases –one for each behavior–. Then, each extracted case-base is filtered and clustered. During filtering, cases whose output –motion command– is null are removed to prevent local minima (the robot stops and never moves again). During clustering, cases with similar input and output instances are removed and cases with similar input and different outputs are marked for supervised verification. This step is specially important because clustering: i) avoids cases in which the same input has different outputs –avoiding oscillations between cases related to similar input instances coming from different training– and ii) optimizes CBR so that number of cases is bounded. It has to be noted that CBR cases are extracted from our system log, that saves each 100 milliseconds. Thus, after filtering, many cases are repeated or too similar. For example, in the *get object* behavior, our CBR case-base has over 600 cases after training, but only 50 cases after clustering.

The clustering algorithm we use is very simple: it just browses the case-base in order, chooses the first case (C_0) and removes from the database all cases whose Chebyshev distance to the chosen one (Eq. 1) is lower than a threshold. After that, it chooses the next case in the database and searches again for similar cases, repeating this operation till the end of the case-base. The distance threshold for inputs is set heuristically to 5, what means –approximately– that cases in which objects' positions are within a range of 10 centimeters are considered to be trained in the same position and so removed from the case-base. The distance threshold for outputs is set heuristically to 25% of the maximum value.

$$D = \max_{i=1}^{N_L}(|\vec{C_0}(i) - \vec{C_n}(i)|) \tag{1}$$

N_L being the length (number of components) of vectors to check $\vec{C_0}$ and $\vec{C_n}$.

E. Simple behavior combination

At this point, we have trained via supervised learning all required low level behaviors that, combined, will provide a coordinated navigation behavior. Thus, it is necessary to determine how these behaviors are combined and which are the inputs for each behavior –as input can be any robot, goal or ball–.

As commented at the beginning of section III, we use an upper layer, statically configured at robot startup, that triggers behaviors according to the configured conditions. This upper layer is an analytical algorithm that only checks a set of conditions and, depending on the result, launches the corresponding low level behaviors. This layer also calculates the final motion command sent to the robot –obtained as the weighted average of all low level behaviors' outputs– and establishes the corresponding weight for each behavior. Weights for all behaviors are set to the same value, except for *avoid object* and *collision behaviors* that, when launched, gain control of the robot for security reasons. We have chosen to work like this because this upper level behavior is actually easier to define in an analytical way and less dependent on the specifics of each robot as long as low level behaviors are working properly.

Our upper layer allows different triggering combinations: behaviors in sequence, in parallel or a combination of both. For example, if we want the robot to search for the ball and then to score a goal, it is just necessary to configure this layer to run in sequence the *get object* behavior twice having as input the ball in the first behavior and the goal in second one. Hence, this upper layer also determines which input is sent to each low level behavior. For example, if we want the previously commented behavior to be combined with obstacle avoidance, we need to run in parallel the *avoid object* behavior with the other ones, setting the obstacle to avoid –for example the closest robot– as input and the obstacle to be in the moving direction as triggering condition. For parallel behaviors, the combined motion command sent to the robot is the weight average of all outputs from all enabled behaviors, where their weight will depend on the situation to solve. In the previous example, if there is an obstacle in the moving direction at 50 cm or less –triggering condition–, the weight of the *avoid object* behavior grows and it gains control until the obstacle is farther than 50 cm or no longer in the moving direction. In any other case, the weight of this behavior is set to 0.

Next section shows different combinations of the low level behaviors tested with the robots and the results obtained in each case for the proposed example.

IV. Experiments and Results

This section briefs results from several runs of the proposed example to check the performance of the system. The goal of presented tests is to prove that combination of learnt behaviors provides different efficient coordinated behaviors as expected. Our test environment is a RoboCup soccer field –scaled to 2 x 1.5 meters– and all involved robots localize themselves using particle filters and visual landmarks localization. All robots in a team share their positions between them via WiFi. Tests have been performed with ERS-7 Aibo robots (no simulations).

In order to test coordinated navigation, we have tried two different behavior combinations involving three robots: two from the same team (R1 and R2) and another from the rival team (R3). All of them include an *avoid collision* analytical behavior that is launched in case of imminent collision to stop the robot and, if the robot has the ball, to steer it in order to pass the ball to the nearest teammate. As commented, since this behavior is quite straightforward, it is programmed instead of learnt.

In the test in Fig.3, robot R1 is configured to run the *avoid object* behavior (input: nearest robot) and the *move in parallel* behavior (input: nearest teammate). Robot R2 is configured to run in sequence the *get object* behavior (input: ball), and, after reaching the ball, *get object* behavior (input: opponent's goal). Finally, R3 is stopped and works as an obstacle this time.

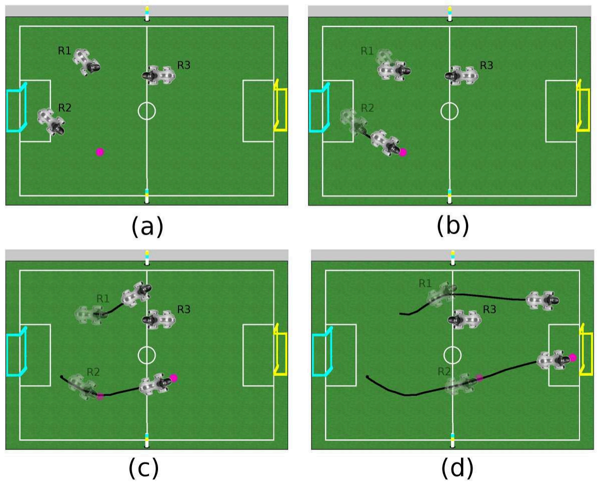

Fig. 3. Snapshots of test 1.

In Fig.3.b we can see how R2 (behavior: *get object*, input: ball) goes straightly to the ball, while R1 (behavior: *move in parallel*, input: R2) tries to be in parallel with R2. Fig.3.c shows how R2 (behavior: *get object*, input: opponent's goal) now goes to the opponent's goal while R1 (behavior: *avoid object*, input: R3) avoids R3, which is in its moving direction. Finally, Fig.3.d shows how R2 gets the opponent's goal and R1 returns to its position in parallel respect to R2. The results from this test were similar for any departure position of the attacking robots and defender, proving that behaviors acquired from the training paths in Fig.2 are generic enough for our tests.

In the second test (Fig.4), robots R1 and R2 are configured to run several behaviors at a time: i) *avoid object* (input: nearest robot); ii) *move in parallel* (input: nearest teammate) if the robot does not have the ball nor is the robot closest to the ball; iii) *get object* (input: ball) if the robot does not have the ball nor is the closest robot to the ball; iv) *go sideline* (input: enemy's goal reference angle) if robot has the ball; and v) *get object* (input: enemy's goal) if robot has the ball and the goal distance is lower than 1 meter. Finally, R3 only runs the *defend object* behavior (inputs: its own goal and the nearest enemy with the ball) if an enemy robot has the ball.

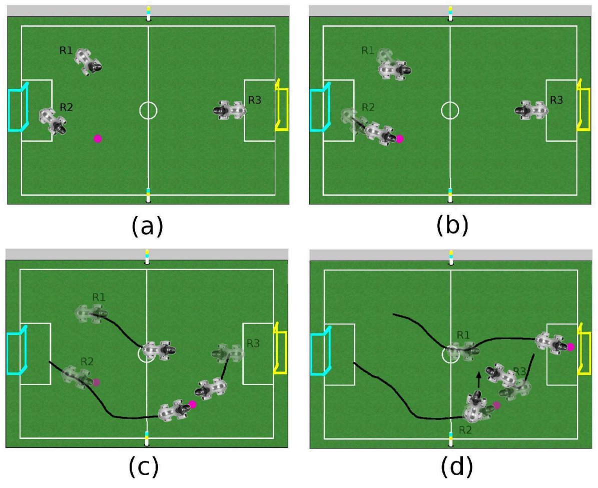

Fig. 4. Snapshots of test 2.

In Fig.4.b we can see how R2 goes straightly to the ball while R1 begins to move in parallel with R2 while R3 does nothing. Fig.4.c shows how R2 now goes to the field sideline (after getting the ball), R1 moves in parallel with R2 and R3 moves to block R2 (nearest enemy with the ball). Finally, Fig.4.d shows how R2 stops due to imminent collision with R3 –blocking R2–. Then R2 passes the ball to R1, and R1 moves to the goal. R3 tries to stop R1 in its way to the goal, but it manages to avoid R3. If we change the relative position of the robots, but keep the high level strategy, we achieve different coordinated behaviors, but the same global outcome. In this test, attacker team got its goal in approximately 90 % of runnings. In the cases in which the team did not get the enemy goal it was due to wrong robot localization or non-refined behaviors.

These two tests shows how robots can perform a coordinated behavior by combining a set of simple learnt ones and how this global behavior can be modulated depending on what low level behaviors are simultaneously triggered and how they are sequenced.

V. Conclusions and Future work

This work has presented a LfD approach to achieve emergent coordinated navigation via combination of low level learnt behaviors. During supervised training, these behaviors are learnt via CBR from a trainer to benefit from human adaptation skills. Advantages of this approach are that: i) dynamics, kinematics and sensory errors can be implicitly absorbed; and ii) some behavior specifics can be acquired even though they are not easy to codify into an analytical expression. The system has been tested with three Aibo-ERS7, using particle filter and visual landmark localization, in a RoboCup-like field.

Navigation is implemented at low level, by associating provided motion commands with the position of other robots and/or objects. The input instance is loosely defined on purpose to allow generalization for different tactics and problems. CBR is used to store these stimuli/perception pairs through supervised learning while training each of the low level CBR behaviors. After training of the desired low level behaviors, they are combined (in parallel, in sequence or both) by a high level layer to perform more complex behaviors that result in an emergent coordinated navigation.

Tests show that it is possible to obtain different trajectories depending on enabled behaviors and how they are combined.

One of the main advantages of the proposal is that it is fairly easy to add new robots and/or sensors to the system by just adding new basic behaviors. Besides, the system can be exported to different robots without major changes as long as we train them for a while. Another advantage is that global behavior is easy to modify by changing the combination of low level behaviors used. Whereas low level behaviors are not that easy to predict, experiments have proven that the global emergent behavior typically follows the guidelines imposed by the high level layer.

Future work will focus not only on testing this approach with more robots and more complex behaviors, but also on replacing the analytical upper layer by a CBR-based one, so that a given robot could actually learn how a human combines low level behaviors at each situation and time instant to achieve a more complex one. Thus, we could achieve dynamic and more adaptable behavior triggering to fit different situations without explicit planning.

VI. Acknowledgements

This work has been partially supported by the Spanish Ministerio de Educacion y Ciencia (MEC), Project TEC2011-29106-C02-01.

References

[1] A. Zhu and S. X. Yang, "A survey on intelligent interaction and cooperative control of multi-robot systems," in *Control and Automation (ICCA), IEEE International Conference on*, 2010, pp. 1812–1817.

[2] F. Almeida, N. Lau, and L. P. Reis, "A survey on coordination methodologies for simulated robotic soccer teams," in *RoboCup Symposium*, 2010.

[3] J. López, D. Pérez, and E. Zalama, "A framework for building mobile single and multi-robot applications," *Robotics and Autonomous Systems*, vol. 59, no. 3, pp. 151–162, 2011.

[4] Y. Kim and M. A. Minor, "Distributed kinematic motion control of multi-robot coordination subject to physical constraints," *The International Journal of Robotics Research*, vol. 29, no. 1, pp. 92–109, 2010.

[5] M. Dekker, P. Hameete, M. Hegemans, S. Leysen, J. Oever, J. Smits, and K. Hindriks, "Hactarv2: An agent team strategy based on implicit coordination," in *Programming Multi-Agent Systems*, ser. Lecture Notes in Computer Science, L. Dennis, O. Boissier, and R. Bordini, Eds. Springer Berlin Heidelberg, 2012, vol. 7217, pp. 173–184.

[6] J. M. Peula, C. Urdiales, I. Herrero, M. Fernandez-Carmona, and F. Sandoval, "Case-based reasoning emulation of persons for wheelchair navigation," *Artificial Intelligence in Medicine*, vol. 56, no. 2, pp. 109 – 121, 2012.

[7] I. Herrero, C. Urdiales, J. M. Peula, I. Sanchez-Tato, and F. Sandoval, "A guided learning strategy for vision based navigation of 4-legged robots," *AI Commun.*, vol. 19, no. 2, pp. 127 – 136, 2006.

[8] J. C. Mogul, "Emergent (mis)behavior vs. complex software systems," *Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems*, vol. 40, no. 4, pp. 293 – 304, 2006.

[9] F. Mantz, P. Jonker, and W. Caarls, "Behavior-based vision on a 4 legged soccer robot," in *RoboCup 2005: Robot Soccer World Cup IX*, 2006, vol. 4020, pp. 480–487.

[10] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[11] J. M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato, and F. Sandoval, "Pure reactive behavior learning using case based reasoning for a vision based 4-legged robot," *Robotics and Autonomous Systems*, vol. 57, no. 6-7, pp. 688–699, 2009.

[12] Z. Liu, N. Jiang, , and L. Zhang, "Self-localization of indoor mobile robots based on artificial landmarks and binocular stereo vision," in *Proceedings of the 2009 International Workshop on Information Security and Application (IWISA09)*, Nov. 2009, pp. 226–231.

[13] S. Chen, "Kalman filter for robot vision: A survey," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 11, pp. 4409–4420, 2012.

[14] R. Luo and W. L. Hsu, "Autonomous mobile robot localization based on multisensor fusion approach," in *Industrial Electronics (ISIE), IEEE International Symposium on*, 2012, pp. 1262–1267.

[15] C. Urdiales, E. Perez, J.Vazquez-Salceda, M.Sanchez-Marre, and F. Sandoval, "A purely reactive navigation scheme for dynamic environments using case-based reasoning," *Autonomous Robots*, vol. 39, no. 5, pp. 67–78, 2006.

[16] C. Urdiales, A. A. Bandera, A. E. J. Perez, A. A. Poncela, and A. F. Sandoval, "Hierarchical planning in a mobile robot for map learning and navigation," *Autonomous Robotic Systems - Soft Computing and Hard Computing Methodologies and Applications*, vol. 3, pp. 165 – 188, 2003.

[17] M. Abramson and R. Mittu, "Learning and coordination: An overview," in *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, 2011, pp. 343–350.

[18] S. Saeedi, L. Paull, M. Trentini, and H. Li, "Neural network-based multiple robot simultaneous localization and mapping," *Neural Networks, IEEE Transactions on*, vol. 22, no. 12, pp. 2376–2387, 2011.

[19] R. Schank, *Dynamic memory*, N. Y. C. U. Press, Ed. New York: Cambridge University Press, 1982.

[20] R. Lopez De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, *et al.*, "Retrieval, reuse, revision and retention in case-based reasoning," *The Knowledge Engineering Review*, vol. 20, no. 3, pp. 215–240, 2005.

[21] S. Begum, M. Ahmed, P. Funk, N. Xiong, and M. Folke, "Case-based reasoning systems in the health sciences: A survey of recent trends and developments," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 4, pp. 421–434, 2011.

[22] S. Ontañon, K. Mishra, N. Sugandh, and A. Ram, "Learning from demonstration and case-based planning for real-time strategy games," in *Soft Computing Applications in Industry*, ser. Studies in Fuzziness and Soft Computing, B. Prasad, Ed. Springer Berlin Heidelberg, 2008, vol. 226, pp. 293–310.

[23] R. A. Bianchi and R. L. de Mántaras, "Case-based multiagent reinforcement learning: Cases as heuristics for selection of actions," in *Proceedings of 19th European Conference on Artificial Intelligence*, 2010, pp. 355–360.

[24] R. Ros and A. J. L. Arcos, "Acquiring a Robust Case Base for the Robot Soccer Domain," *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1029 – 1034, 2007.