

Coordination in dynamic environments with constraints on resources

A. Farinelli, G. Grisetti, L. Iocchi and D. Nardi

*Dipartimento di Informatica e Sistemistica
Università "La Sapienza", Roma, Italy*

Abstract

Distributed coordination in multi-robot systems (MRS) is one of the most interesting research issues aiming at improving the performance, the robustness and reliability of a robotic system in accomplishing complex tasks. In this paper, we describe the design and realization of a MRS that takes into account at the same time dynamic role assignment and constraints on resources. The present work extends our previous work in distributed robot coordination by adding in the MRS the capability of synchronizing their actions by exploiting a framework for automatic plan generation and plan execution based on a logical formalism. The proposed method has been applied in the RoboCup environment within the Sony Legged Robot League.

Acknowledgments

Work partially supported by ASI project ARISCOM, by the MIUR Project Knowledge-based mobile Robots, and by CNR project "Sistema multi-agenti con componenti fisse e robotiche mobili".

We thank Vincenzo Bonifaci, Fabio Cottefogle, Fabio Patrizi, Vittorio A. Ziparo for their implementation of the coordination component of the SPQR Legged team.

1 Introduction

Cooperative Robotics is one of the most interesting research areas in Artificial Intelligence and Robotics [8, 17, 16, 3]. The field is rather broad; in the present work we specifically address the issue of distributed coordination in multi-robot systems (MRS). Moreover, we consider MRSs from an engineering perspective, that is to say with the aim of improving the effectiveness of a robotic system both from the viewpoint of the performance in accomplishing certain tasks [8] and in the robustness and reliability of the system [17]. Consequently, we specifically address both the cooperation of multi-robots in order

to perform a global task and coordination to achieve a common goal. More specifically, The design of a MRS performing complex tasks in a dynamic environment must take into account two main aspects: 1) due to the presence of other agents (possibly adversary) in the environment, a dynamic assignment of roles is required for effective performance of the robotic system; 2) the complexity of the task often requires a form of coordination that takes into account constraints on resources that are accessed by the robots.

An analysis of the works in the literature shows that very often cooperation through dynamic task assignment and coordination on the access to shared resources are often treated, separately. The former problem is faced by splitting the tasks the robots have to accomplish, and assigning each robot a role (sub-task), that can change during the global task execution [11, 19, 7]. Distributed approaches are considered more attractive, since they do not place strict constraints on the reliability of communications. While the solutions to dynamic role assignment can be considered quite effective for a number of tasks, they do not directly support the coordination in the access to shared resources, that may be required in many applications.

A second group of works is instead focussed on the techniques for handling the conflicts arising from the attempt to access common resources. Resource conflicts among robots can be solved by combining the plans of each robot, and producing a coordinated plan [2]; using task networks indicating dependencies among the tasks to be executed [15]; or using ad hoc approaches, such as space partition in foraging task [10]. In this works, the possibility of dynamically switching the tasks assigned to each robot is generally not supported.

There are a few proposals [18, 20, 17] that consider both dynamic task allocation and conflict resolution in the same framework and our work has similar goals. Specifically, we propose the design and re-

alization of a MRS that takes into account at the same time dynamic role assignment and constraints on resources. The present work extends our previous work in distributed robot coordination [6] by adding in the MRS the capability of synchronizing their actions by exploiting the framework for plan generation and execution described in [9]. The main feature of the proposed approach is to keep the requirements on each robot to a very abstract set of functionalities. In particular, by phrasing the coordination constraints within the plans of the individual robots we do not need an explicit synchronization, but this is integrated within the information acquisition capabilities of the robots. This feature, makes it easier to implement the approach with heterogenous MRSs.

The proposed approach may be deployed in several application domains in which coordination is needed for an effective realization of the MRS in dynamic environments. In this paper we show its application in the RoboCup environment and in particular within the Sony Legged League competitions. In fact, RoboCup has recently given a significant boost to the work on MRS [16], since, as compared with previous work on MRS, the RoboCup environment is highly dynamic and hostile due to the presence of an opponent team and moreover effective coordination is fundamental in order to perform well in the competitions. Therefore the RoboCup setting provides not only significant challenges in the design of MRS, but also a common field for experimental evaluation of the approaches.

In the Sony Legged Robot League there is a rule (two-defender rule) which prohibits the simultaneous presence of two players in the goal area. This rule introduces a new challenge for coordination, because of the need to synchronize the action of the robots with respect to entering the goal area in such a way as to avoid breaking the rule. In fact, it turns out that this rule gives rise to a scenario, where the dynamic exchange of roles is not sufficient for effective performance, while a specific constraint on the access to a shared resource (the goal area) must be properly taken into consideration.

We have experimented the approach described in this paper in the coordinated task of dynamic role exchanging between the goalkeeper and another player, while properly handling the two-defender rule (that is a constraint on the presence of only one robot in the goal area at any time). The basic intuition underlying the proposed solution is to treat the role assignment as a technique for establishing the goal of each individual player (according to the architecture described in [4]). Moreover, the need to synchronize with other players must be explicitly addressed in the selection and execution of the plan devised by

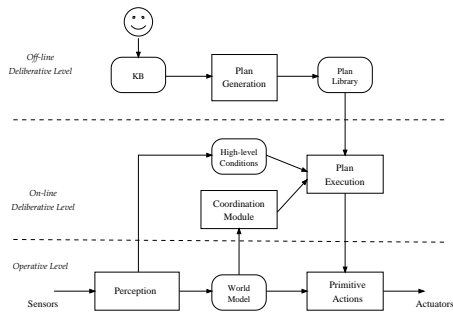


Figure 1: Layered architecture for our robots

the player to achieve the assigned goal (role). To this end we will exploit the framework for plan generation described in [9].

The paper is organized as follows. We first recall the basics of the cognitive architecture and of the distributed coordination of our robots. We then describe the technique to address the problem and its implementation, finally, we discuss its relationship with other approaches.

2 Architecture for Coordination, Plan Generation and Execution

2.1 Plan Generation and Execution

In this section we recall the framework for plan generation and execution already presented in [4]. The framework is based on a layered hybrid robot architecture (see also [13]), displayed in Fig. 1, that has been implemented on several different kinds of robotic platforms, namely Sony AIBOs, Pioneer, and home-made wheeled robots.

The *Operative Level* of the architecture is based on a numeric representation of the information acquired by the robot sensors and of the data concerning the current task, while the *Deliberative Level* is based on a symbolic representation of the information acquired by the robot sensors and of the data concerning the task to be accomplished: the *On-Line Deliberative SubLevel* is in charge of evaluating data during the execution of the task, while the *Off-line Deliberative SubLevel* is executed off-line before the actual task execution.

The deliberative level relies on a representation of the robot’s knowledge about the environment. This knowledge is formed by both a general description of the environment provided by the robot’s designer and the information acquired during task execution. In particular, the world model of this level contains symbolic information corresponding to the data in the geometric world model of the operative level.

The deliberative level is formed by two main compo-

nents:

1. a *Plan Execution Module* that is executed online during the accomplishment of the robot's task and is responsible for executing a plan by coordinating the primitive actions of a single robot;
2. a *Plan Generation Module* that is executed offline before the beginning of the robot's mission, and generates a set of plans to deal with some specific situations.

The reasoning system processes a knowledge base containing a description of the actions to be performed by the robot using an action representation language. In particular, a planner has been developed in order to generate a plan to achieve a given goal in a specific situation. Moreover, the user can also manually design a plan, by using a graphic tool and he/she can validate this plan with respect to the knowledge base by using a plan verification module based on model checking. During the execution of a plan, the plan execution module checks for the conditions that guarantee the applicability of the current plan in the current situation (provided by the high-level state), and, if the current plan is no longer executable, it selects a new plan from the library.

A plan is represented as a *transition graph*, where each node denotes a state, and is labeled with the properties that characterize the state, and each arc denotes a state transition and is labeled with the action that causes the transition. A state represents a situation the system can be in and is characterized by a set of properties which give a (complete) description of the situation. Actions are represented using preconditions and effects. Preconditions are the conditions that are necessary for activating the action and indicate what must be true before the action is executed: they specify circumstances under which it is possible to execute an action. Effects are the conditions that must hold after the execution of the action and characterize how the state changes after the execution of the action.

The actions in the graph can be classified into ordinary (i.e. movement) actions and sensing actions. The former cause changes in the environment, while the latter permit the acquisition of information, in order to let the robot take better decisions. Both actions are relevant to state transition. Sensing actions are associated with conditions to be verified: depending upon the runtime value of these conditions, a different part of the plan will be executed.

In order to be actually executable, a plan requires additional information that is not directly derived from the knowledge base of the robot, because it is

external to the formalism. In particular, the monitor has to know which preconditions have to be verified during the entire execution of the action and which effects determine the state transition. Furthermore, the plan may require modifications in order to cope with possible action failures.

In our setting, the *Plan Generation Module* generates cyclic conditional plans (see [9]) based on the logical formalism. These plans are then automatically extended for dealing with those failure cases that can be automatically adjusted, and the user can possibly modify the plan by using an appropriate graphical tool. After these modifications plan are collected in a Plan Library and they will be selected and executed during the robot mission according to the Coordination Module and the robot internal state.

2.2 Distributed Coordination Protocol

The coordination protocol presented in [6, 5] is based on the dynamic assignment of roles and of the team strategy. The computation for the coordination protocol is distributed and the protocol is robust because it relies on a little amount of transmitted data. The approach has been successfully experimented within our team of heterogeneous robots in the RoboCup Mid-size League.

Each robot has the knowledge necessary to play any role, therefore robots can switch roles on the fly, as needed. In practice any role corresponds to a goal to be achieved by a robot (e.g. going to the ball, defending the goal area, etc.) and dynamic role exchange is a very effective way to select the robot that is in a better situation to accomplish a task. Moreover the assignment is computed by means of a distributed protocol in such a way to be robust to possible network failures.

Dynamic role assignment is achieved by means of *utility values* exchanged among the robots in the team through explicit communication. Such values indicate the usefulness, with regard to the whole team, of the assignment of each role to each robot. This is obtained through the definition of a set of *utility functions* (specific for every role) that every robot evaluates given its current local information about the environment. By comparing these values, each member of the team is able to establish the same set of assignments (robot \leftrightarrow role) to be immediately adopted.

In order to avoid too frequent role switches, the utility functions contain hysteresis, that add a preference for a robot to maintain its current role. In this way role switch is performed only when another robot is really in a better situation of execute the task.

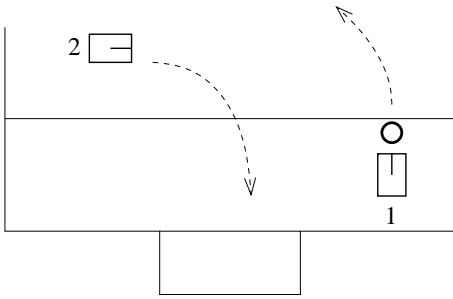


Figure 2: Typical situation for dynamic coordination.

3 Dynamic Coordination with Constraints on Resources

The problem we describe in detail in this section, as an example for explaining our coordination method, is the dynamic exchange of the role of goalkeeper in the Sony Legged League and the application of the two-defender rule.

This problem cannot be easily solved by applying the distributed coordination protocol presented in [5], since dynamic role exchange is not sufficient to respect the two-defender rule. In other words, we want to solve two aspects of coordination: on one hand dynamic role exchange allows for selecting the best robot able to accomplish a task, on the other hand while accomplishing their own tasks the robots must take into account some constraints on shared resources and thus have to synchronize their actions.

The situation presented in Fig. 2 is a typical situation in the Sony Legged League matches in which the goalkeeper (robot number 1) is moving away from its own goal and is approaching the ball to push it away, while robot number 2 is far away from the ball and it cannot help the goalkeeper immediately. In this situation it is more convenient for the team that robot 1 takes the role of attacker pushing the ball toward the opposite goal, while robot 2 goes back to defend its own goal acting as a goalkeeper. However in performing this role exchange the two robots must comply with the two defenders rule, and thus robot 2 can enter the goal area only after robot 1 has left it.

The solution we present in this paper is divided in two parts: 1) we have defined utility functions for dynamic assignment of the roles of goalkeeper and attacker; 2) we have implemented action synchronization in the plan the two robots will execute.

3.1 Utility functions for dynamic role assignment

The utility functions for the role attacker and goalkeeper are defined as functions of information of the

robot about the environment. These functions are designed in such a way that higher values indicate that the robot is able to perform the task associated with this role, while lower values mean that the robot may not be adequate to perform this task. Utility functions are usually determined by a set of experiments aiming at an appropriate configuration of the MRS. Examples of utility functions for the roles *attacker* and *goalkeeper* are reported below. they are mainly based on the position of the robots and of the ball in the field. Other factors may be easily taken into account depending on the implementation of the actions and on game strategies.

The utility function for the attacker is based on the robot position and orientation and its distance to the ball. This function evaluates an estimation of the time needed to reach the ball in the direction of the opponent goal.

$$f_{attacker} = -\alpha trajectory_to_ball - \beta |dir_to_opponent_goal| + \dots + Hysteresis$$

where α and β are positive coefficients to be determined by experiments and that may also be different among robots taking into account heterogeneity in the capabilities of the robots in a team (although it is not this case in the Sony Legged League where robots are all the same). *Hysteresis* is a term that introduce a preference for the robot to maintain its current role.

The utility function for the goalkeeper is based on its position in the field, This function evaluates an estimation of the time needed to reach the goal facing the opponent goal.

$$f_{goalkeeper} = -\alpha dist_to_my_goal - \beta |dir_to_opponent_goal| + \dots + Hysteresis$$

These functions can also be improved by considering the position of other players in the field or the presence of obstacles in the path that must be executed.

3.2 Action synchronization during plan execution

The problem of complying with the two-defender rule is solved by generating a plan in which one robot before entering the goal area must check that it is free (i.e. the other robot has left the area).

This is achieved by adding in the knowledge base of the robot that is taking the role of goalkeeper the following specification for the actions *GotoAreaLine*, *SenseFreeArea*, and *GotoGoal* (given in a simplified notation with respect to the formal one used in [14]): *GotoAreaLine* has precondition *NOT PosGoal* and effects *PosAreaLine*; *SenseFreeArea* has preconditions *PosAreaLine* and effects *FreeArea / NOT FreeArea*; *GotoGoal* has preconditions *PosAreaLine AND FreeArea* and postconditions *PosGoal*.

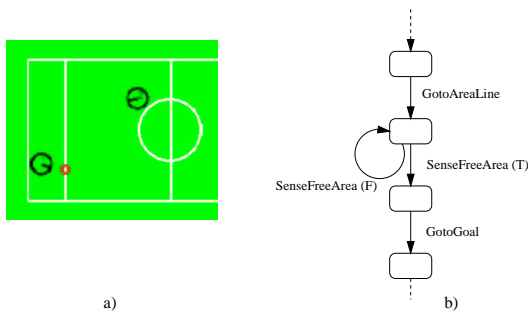


Figure 3: a) Simulator environment. b) Cyclic conditional plan generated by the Plan Generation Module.

The conditions *PosAreaLine*, *FreeArea*, and *PosGoal* represent respectively the robot positioned close (but outside) the area line, the area being free (from robots of its own team), and the robot being in the goal area. Note that the KB includes the specification of other actions and conditions that are not reported here since they are not directly related to the coordination example.

The portion of the plan, of interest for coordination, generated by the automatic plan generation system (see [9]) is represented in Fig. 3b). The plan contains a **while** loop in which the robot waits for the condition *FreeArea* to become true before entering the area. This loop allows for synchronization of the actions between the two plans executed by the robots. In fact, the robot executing the plan in the Figure will not enter the goal area until this is free.

4 Implementation

Previous work on coordination in RoboCup has been carried out mostly in the Mid-size league [5, 12, 22], and in the simulation league [21]. In the past year, we have seen first attempts to coordinate the robots also in the Sony Legged League, however communication through the sound emitters is very unreliable, in particular during the matches. The use of explicit communication in the Sony Legged League (introduced this year) can now take advantage of wireless LAN communication, as in the case of Mid-size league. This makes it feasible to pursue communication-based coordination also in this league. In the following some details on the implementation of our approach in this League and the experiments we have performed are presented.

4.1 Action Implementation

The actions executed by the robots are implemented in the operative layer of our software architecture. In particular, the implementation of the action *SenseFreeArea*, that is of interest for coordination, is ob-

tained by means of explicit communication among the robots. In fact, every robot broadcasts a set of information about its state, including its position to all the other teammates. The execution of the sensing action *SenseFreeArea* corresponds to checking that among the positions received from all the teammates no one is inside the defense area. Notice that the implementation of this sensing action based on explicit communication is adequate for action synchronization among robots, since it is robust in the case of missing communications for a limited amount of time. In fact, when no data from a robot are acquired *SenseFreeArea* is implemented in order to return false, when data are received from other robots, the action evaluates their positions. However, in other cases, it is obviously possible to implement the sensing action used for synchronizing the robots with other techniques (for example vision processing). The current implementation of the sensing action *SenseFreeArea* also relies on the ability of each robot to localize itself, and this could be a limitation in the effectiveness of the approach. However, in practice, for coping with this situation, it is not important to have a precise localization for the robot, but only the ability of detecting when the robot has left the defense area.

4.2 Experiments for goalkeeper-attacker Coordination

The experimental setting we have used has been given by a simulator (see Fig. 3a)), that even though cannot provide a precise characterization of all the aspects that influence the performance of the robot in the real environment, it can provide useful feedback to the design of the coordination system for actual robots. Through this simulator we have verified the intended behaviour of the robots in each of the roles in different scenarios and we have effectively tuned the utility functions. However this tuning will be performed again with real robots. The simulator provides a global view of the environment that in the case of our experiment is the RoboCup soccer field, and shows ball position and robot positions and orientations. With the simulator is possible to let the simulated robot play, changing the environment in order to represent some particular situations of interest. In our experiments we have represented the situation of Figure 2, and we have checked the dynamic role assignment and the execution of the robot plans. More in depth the first robot (robot number 1 in the figure) is initially assigned to a defensive role and its position is inside the goal area, robot number two start position is outside the goal area and is assigned with a non defensive role. When the ball is positioned in front of the robot number 1 a role change occurs: robot number 1 takes the attack role

and begins to push the ball toward the opponent goal escaping the goal area, meanwhile the robot number 2 takes the defender role, and starts to go in the defense position but it does not enter the area until the other robot leaves it, due to the SenseFreeArea sensing action discussed above. The experiment shows that the algorithm for dynamic role assignment successfully exchange the role between the two robots and that the SenseFreeArea action correctly avoids the presence of two robots in the defense area. The method used for the role exchange is also robust with respect to the following unexpected situation: 1) If robot 1 throws the ball toward robot 2 then, due to the dynamic role assignment algorithm, robot 2 becomes the attacker and begins to push the ball toward the opponent goal while robot 1 (which is become defender again) go back to the defense position. 2) If robot 1 is not able to escape from the goal area, then the utility function passes the role of attacker to robot 2; robot 2 starts heading toward the ball while robot 1 goes back to defense position. 3) When three robot are playing, if for some reasons robot 2 violates the two-defender rule and is taken out from the field (according to the RoboCup Legged competition's rules) while robot 1 is pushing the ball, then robot 3 will take the defender role and will go in the defense position while robot 1 keep on pushing the ball.

The mechanism described in this article has been implemented in the Sony Legged robots of the S.P.Q.R. team, in particular we have tested the coordination capability in the technical challenge on coordination of RoboCup (2002) [1]. We are currently working on applying the method also to other situations requiring coordination. In particular, the described approach may be used for ball passing. In this context synchronization is needed in order to execute the pass when the other robot is ready to receive it. Moreover it is also very important that during the execution of their tasks the robots are ready to change their roles if the situation in the environment changes. For example, a typical situation in which a robot may decide to pass the ball, instead of kicking it to the goal, is when the opponent goalkeeper has a good coverage of its own goal. However, during the execution of the ball-passing plan, the robot in possess of the ball should always be ready to kick the ball to the goal if for example the opponent goalkeeper moves away for its position.

5 Discussion

In this paper we have presented an approach to coordination of robots in the Sony Legged League. The characteristics of the matches in the Legged League offer an interesting setting for coordination in which

two aspects must be taken into account at the same time: dynamic role assignment and the presence of constraints to access shared resources. The solution presented in this paper relies on our previous work on coordination of robots in the Mid-size League and on the use of a framework for plan generation and execution that allows for implementing synchronization of the actions of the robot in the team.

The work on distributed coordination often tries to approach the problem of multi robot coordination by splitting the tasks the robot have to accomplish, and assigning each robots a role, that can change during the task execution [11, 19, 7]. An important issue in dynamic role assignment, is whether the roles are assigned to the robots by a central unit or each robot decides its own role. Even if centralized approaches generates optimal solutions they require a very high communication affidability, and are not fault tolerant. Distributed approaches to dynamic task allocation, provide a higher reliability, as example in [11] an interesting distributed task allocation system (MURDOCH) is presented. MURDOCH is based on a publish/subscribe messaging communication among the robots. An interesting point is that the communication is always broadcast, and centered on the resource a robot requires to perform a certain task, the robots just require and give to the teams the resource they need or have, and form dynamically changing subteams for accomplish the tasks. Another interesting approach is to dynamically choose a leader for the robot team that rules the other robots in the task accomplishment [7, 19]. However all those works do not take explicitly into account the synchronization of action among the robotic agents and thus does not provide a solution to the issue of concurrent access to shared resource.

On the other hand, techniques for conflict resolution try to manage the resources for the robot team in order to permit the task to be executed, avoiding delays or deadlocks. Resource conflicts among robots can be solved by combining the plans of each robot, and producing a coordinated plan. For example, in [2] when a robot is executing a plan that makes use of a shared resource it advertises it, and collects from other robots the plans which specify how they intend to use the shared resource, and the permission to generate a coordinated plan. The coordinated plan is then produced and executed. During the execution, the appropriate messages for synchronization are generated. This approach is used for the transshipment of autonomous robots in harbors, airport and marshalling yards.

This work presents a very interesting solution to the issues of resource conflicts, because uses a totally distributed coordination algorithm, and moreover the

architecture is based on a deliberative approach, and thus it is similar to ours. However the testbed environment has significant differences since in the case of [2] the role for each robotic agent are assigned by the central station and thus the task assignment is not distributed. Moreover due to the characteristics of the environment the dynamic exchanging of roles among the robotic agents is not a crucial issue and thus the work does not consider in detail this problems.

A different approach to conflict resolution based on territorial division is used in many domains, such as foraging [10]. In this work, each robot is assigned a territory to accomplish its task, and the territory can be dynamically resized, if malfunctions occurs to some robots. With this approach interferences between robots are minimized, and the task accomplishment improves considerably. However, the solution proposed in this work for conflict resolution seems rather specific.

Another class of works instead are explicitly focussed on both dynamic task allocation and conflict resolution [18, 20, 17, 15]. These works provide methods that can be applied in coordinating MRS in dynamic environments by taking into account constraints in accessing shared resources.

As compared with those works our approach differs in both the automatic generation of plans containing action synchronization and in the mechanism used for dynamic role assignment. In particular in [15] an approach based on the use of task networks indicating dependencies among the tasks to be executed is presented. This work makes use of learning for establishing relationship among tasks and action selection is performed by considering such network. The main difference with respect to our work is that although an implicit dynamic task assignment is entailed from this approach, an explicit protocol of task distribution is not considered. In [18] task reallocation is computed from a Distributed Organizational Task Network specified by the user that takes into account dependencies among actions (or tasks). Dynamic task reallocation is computed in order to minimize total dependencies among tasks. The present work differs from the approach presented in [18] in two aspects: i) action synchronization is implemented with the execution of cyclic conditional plan automatically generated starting from a logical description of the environment; ii) dynamic role assignment is computed during the execution of the robots tasks by taking into account the current state of each robot. The work in [20] describes a dynamic task assignment similar to the one proposed here and a conflict resolution method that is based on the definition of a correlation function between each pair of

modules (tasks). The work is focused on a slightly different problem with respect to the one discussed in this work, namely the conflict among different tasks without considering explicitly the problem of a concurrent access to a shared resource. Thus, in their work the ordering of actions is not explicitly handled, but is inferred by the conflict resolution algorithm. In our approach instead action synchronization is explicitly considered by the *Plan Generation Module* that generates plans containing action synchronization. Finally, in the framework presented in [17] both conflict resolution and dynamic role assignment may be implemented by using a combination of the acquiescence and impatience factors. This work is focussed on the implementation of reactive MRS, while in our approach the deliberative component (i.e. plan generation and execution of high-level actions) has been a primary design choice.

It is worth to notice that in our approach the issues of conflict resolution and dynamic task assignment are addressed in a separate way; more specifically, dynamic task assignment based on utility functions, while the conflict resolution is solved at the symbolic level. The separation of the two aspects in terms of constraints on plans (dynamic role correspond to goals and synchronization to sensing action or information acquisition) has, in our view several advantages. First of all the coordination protocol is simple, robust, already well experimented [6]. In addition synchronization is achieved without requiring an explicit coordination protocol, whose implementation might be demanding especially in the case of of an heterogeneous Multi Robot System. Finally the overall solution is flexible since if we decide to change the coordination protocol no changes need to be done at the symbolic level and vice versa.

We have implemented the framework described in this paper in our Sony Legged team by using a wireless connection. Although we need to evaluate the approach by defining a proper set of experiment, some promising results have been obtained in simulation as well as on real robots.

References

- [1] Robocup official site. <http://www.robocup.org>.
- [2] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi robot cooperation in the martha project. *IEEE Robotics and Automation Magazine* (Published also in the Special Issue on "Robotics and Automation in Europe : Projects funded by the Commission of the European Union, 1997).
- [3] Tucker Balch and Lynne E. Parker, editors. *Robot Teams: From Diversity to Polymorphism*. A K Peters Ltd, 2002.

- [4] C. Castelpietra, A. Guidotti, L. Iocchi, D. Nardi, and R. Rosati. Design and implementation of cognitive soccer robots. In *Proc. of RoboCup Symposium*, 2001.
- [5] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Communication and coordination among heterogeneous mid-size players: ART99. In *RoboCup-2000: Robot Soccer World Cup IV*, 2000.
- [6] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Coordination among heterogeneous robotic soccer players. In *Proc. of International Conference on Intelligent Robots and Systems (IROS'2000)*, 2000.
- [7] L. Chaimowicz, T. Sugar, V. Kumar, and M. F. M. Campos. An architecture for tightly coupled multi-robot cooperation. In *Proc. of IEEE International Conference on Robotics and Automation*, pages pp. 2292–2297, Seoul, Korea, May 2001.
- [8] D. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.
- [9] Alessandro Farinelli, Giorgio Grisetti, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Generation and execution of partially correct plans in dynamic environments. In *Proceedings of 3rd International Cognitive Robotics Workshop (COGROB02)*, 2002.
- [10] M. Fontan and M. Mataric. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 15(5), 1998.
- [11] Brian P. Gerkey and Maja J. Mataric. Principled communication for dynamic multi-robot task allocation. In *Proceedings of the International Symposium on Experimental Robotics*, Waikiki, Hawaii, December 2000.
- [12] J.-S. Gutmann, T. Weigel, and B. Nebel. Fast, accurate, and robust self-localization in the robocup environment. In *RoboCup-99: Robot Soccer World Cup III*, pages 304–317, 1999.
- [13] Luca Iocchi. *Design and Development of Cognitive Robots*. PhD thesis, Univ. "La Sapienza", Roma, Italy, On-line <ftp.dis.uniroma1.it/pub/iocchi/>, 1999.
- [14] Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: logical framework and implementation. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 678–689, 2000.
- [15] D. Jung and A. Zelinsky. An architecture for distributed cooperative planning in a behaviour-based multi-robot system. *Journal of Robotics and Autonomous Systems* 26, pp149-174, 1999.
- [16] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for AI and robotics. In *Lecture Note in Artificial Intelligence*, volume 1395, pages 1–19, 1998.
- [17] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.
- [18] W. M. Shen and B. Salemi. Distributed and dynamic task reallocation in robot organizations. In *Proceedings of the 2002 IEEE Int. Conference on Robotics and Automation*, Washington, DC, May 2002.
- [19] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First results in the coordination of heterogeneous robots for large-scale assembly. In *In Proceedings of the International Symposium on Experimental Robotics (ISER), Honolulu Hawaii*, December 2000.
- [20] E. Uchibe, T. Kato, M. Asada, and K. Hosoda. Dynamic task assignment in a multiagent/multitask environment based on module conflict resolution. In *Proceed. of the 2001 IEEE International Conference on Robotics and Automation*, May 2001. Seoul, Korea.
- [21] M. Veloso and P. Stone. Individual and collaborative behaviors in a team of homogeneous robotic soccer agents. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 309–316, 1998.
- [22] K. Yokota, K. Ozaki, N. Watanabe, A. Matsumoto, D. Koyama, T. Ishikawa, K. Kawabata, H. Kaetsu, and H. Asama. Uttori united: Cooperative team play based on communication. In *RoboCup-98: Robot Soccer World Cup II*, 1998.