Artificial Intelligence and Systems Theory Applied to Cooperative Robots: the SocRob Project*

Pedro Lima, Luis Custódio
{pal,lmmc}@isr.ist.utl.pt
Instituto de Sistemas e Robótica, Instituto Superior Técnico – Av. Rovisco Pais, 1 – 1049-001 Lisboa

Abstract - **This paper describes an approach to the design of a population of cooperative robots based on concepts borrowed from Systems Theory and Artificial Intelligence The research has been developed under the SocRob project, jointly carried out by the Intelligent Control and Artificial Intelligence Laboratories at ISR/IST. The acronym of the project stands both for "Society of Robots" and "Soccer Robots", the case study where we are testing our population of four robots. Designing soccer robots is a very challenging problem, where the robots must act not only to shoot a ball towards the goal, but also to detect and avoid static (walls, stopped robots) and dynamic (moving robots) obstacles. Furthermore, they must cooperate to defeat an opposing team. Our current research in soccer robotics includes image processing for object segmentation, recognition and tracking, navigation and behavior-based architectures for real time task execution of cooperating robot teams. Cooperative learning, behavior modeling and distributed planning are topics we have been investigating and plan to apply to soccer robots and other case studies, such as search and rescue robots.**

I. INTRODUCTION

Cooperative Robotics is a modern research field, with applications to areas such as building surveillance, transportation of large objects, air and underwater pollution monitoring, forest fire detection, transportation systems, or rescue after large-scale disasters. In short, a population of cooperative robots behaves like a ``distributed" robot to accomplish tasks that would be difficult, if not impossible, for a single robot. Many lessons important for this domain can be learned from the Multi-Agent Systems field of Artificial Intelligence (AI) concerning relevant topics for Cooperative Robotics, such as distributed continual planning[5], task allocation[7], communication languages or coordination mechanisms[4]. Robotic soccer is a very challenging problem, where the robots must cooperate not only to push and/or kick an object (a ball) towards a target region (the goal), but also to detect and avoid static (walls, stopped robots) and dynamic (moving robots) obstacles while moving towards, moving with or following the ball. Furthermore, they must cooperate to defeat an opposing team. All these are features common to many other cooperative robotics problems. This paper surveys the several research problems addressed by the SocRob project, building on AI concepts a Systems Theory standpoint. In Section II we describe our view of the general problem involving multiple robots that act as a team, cooperating and coordinating their actions to attain the team goal. Needless to say, single-robot ``traditional" research problems are covered, both from the sub-system and the integration standpoints. Natural extensions to cooperative multi-robot teams are also detailed. The problems addressed so far and the solutions we obtained for them are described in Section III. Open problems of interest for the project and clues on how we intend to approach their solution are discussed in Section IV. We end the paper drawing some conclusions in Section V.

II. A GENERAL MULTI-ROBOT COOPERATION AND COORDINATION PROBLEM

Many researchers around the world are designing mobile robots capable to display increasing autonomy and machine intelligence properties. Most groups concentrate in specific subsystems of a robot, such as the planner, the navigator, or the sensor fusion. What usually is missing in their design is a systematic way to glue together all these subsystems in a consistent fashion. Such a methodology, should one be available, would help engineering the mobile robots of the future.

Perhaps the key factor of success for a robot lies on its capability to perceive correctly its surrounding environment, and to build environment models adequate for the task the robot is in charge of, from the information provided by its sensors. Different sensors (e.g., vision, laser, sonar, encoders) can provide alternative or complementary information about the same object, or information about different objects. Sensor fusion is the usual designation for methods of different types to handle the data from the several sensors available and provide improved information about the environment (e.g., about its geometry, color, shape, relevance).

When a team composed of several cooperating robots is concerned, the sensors are spread over the different robots, with the important advantage that the robots can move (thus moving its sensors) to actively improve the cooperative perception of the environment by the team.

* This project has been sponsored over the years by several public and private institutions: Fundação para a Ciência e a Tecnologia, Fundação Calouste Gulbenkian, Agência de Inovação, Mota & Teixeira, Alcatel, Caixa Geral de Depósitos, ICEP - Portugal, FLAD, Público newspaper, ISR/IST, IST

The information about the environment so obtained can be made available and regularly updated by different means (e.g., memory sharing, message passing, wireless communications) to all the team robots, so as to be used by the other sub-systems.

Once the information about the world is available, one may think of using it to make the team behave autonomously and machine-wise intelligently. Three main questions arise for the team:

- Where and which *apriori* knowledge about the environment, team, tasks and goals, and perceptual information gathered from sensors, should be kept, updated and maintained? This involves the issue of distributed knowledge representation adequate to consistently handle different and even opposite views of the world.
- What must be done to achieve a given goal, given the constraints on time, available resources and distinct skills of the team robots? The answer to this should provide a team *plan*.
- How is the actual implementation of a plan handled, ensuring the consistency of individual and team (sub)-goals and the coordinated execution of the plan?

So far, a bottom-up approach to the **implementation** of a cooperative multi-robot team has been followed in the SocRob project, starting from the development of single robot sub-systems (e.g., perception, navigation) and moving towards relational behaviors, comprehending more than one robot.

However, a key point is a top-down approach to system **design**. The design phase establishes the specifications for the system:

- qualitative specifications - concerning formal logical task design so as to avoid deadlocks, livelocks, unbounded resource usage and/or sharing non-sharable resources, and to choose the primitive tasks that will span the desired task space;
- quantitative properties - concerning performance features, such as accuracy (e.g., the spatial and temporal resolution, as well as the tolerance interval around the goal, at each abstraction level), reliability and/or minimization of task execution time given a maximum allowed cost.

To support this top-down design and bottom-up implementation philosophy, suitable functional and software architectures, respectively, must be conceived prior to the development of all the sub-systems.

A. Single-Robot Research Problems

Most of the problems tackled so far within the SocRob project concern the sub-systems of the individual robots composing a team. From our standpoint, relevant topics are:

Functional and Software Architectures: Modern robots should be designed based on a top-down design from specifications to ensure desired performance levels (both qualitative and quantitative). Therefore, the designers should start by specifying a functional architecture which will guide the design of the robot sub-systems in an integrated fashion, i.e., each sub-system is not necessarily designed to optimize its performance but rather aiming at optimizing the overall system performance. Another important issue is to determine, given the desired task space (i.e., the set of tasks that will have to be carried out by the robot in a particular application), the minimal set of primitive tasks that will span that task space. Moreover, the final implementation should be supported on a suitable software architecture designed to allow real-time multi-processing, data sharing and mutually exclusive allocation of shared resources among the robot sub-systems.

Single-Robot Task Planning: Given the primitive task set referred in the previous item, the robot must be able, given the current and past world states (including its own internal state), to compose primitive tasks so as to come up with a plan that carries out a given desired task. There may be more than one plan that accomplishes a task, but the planning system should be able to determine the one that achieves the best performance, based on the available information and prediction horizon.

Single-Robot Task Coordination: Plans must be such that they allow continuous handling of the environment uncertainties and unexpected events. Once a plan is determined, task coordination deals with its execution. Plan execution must, at least, take into account the detection of events, smooth transitions between primitive tasks, synchronization of primitive tasks executed concurrently, mutual exclusion when two or more tasks attempt to access shared resources, iterative estimation of primitive task performance, learning how to improve a plan over time by choosing more convenient algorithms among those available for each primitive task, and so on.

Navigation: The navigation system is an important sub-system of a mobile robot. In many applications one important feature of the navigation system concerns the ability of the robot to self-localize, i.e., to autonomously determine its position and orientation (posture). Using posture estimates, the robot can move towards a desired posture, i.e., by following a pre-planned virtual path or by stabilizing its posture smoothly[1]. If the robot is part of a cooperative multi-robot team, it can also exchange the posture information with its teammates so that appropriate relational and organizational behaviors may be established. In robotic soccer, these are crucial issues. If a robot knows its posture, it can move towards a desired posture (e.g., facing the goal with the ball in between). It can also know its teammate postures and

prepare a pass, or evaluate the game state from the team locations. Most approaches to Navigation determine with high accuracy the posture of the robot with respect to a given coordinate frame. However, this approach is typically resource-consuming, requiring the robot to spend a significant percentage of its processing time with the navigation sub-system disregarding other important sub-systems, such as perception or planning, to name but a few. Furthermore, high accuracy is not always required for navigation purposes. One may be just interested to move closer to an object, rotate to see a given landmark, or move to another region. In those cases, another approach to navigation, known as topological (or relative) navigation, is advisable.

Object Recognition and Tracking: The ability to discriminate and recognize its surrounding objects, to distinguish the relevant ones and to track, among them, those that move, is a major problem for any robot. For soccer robots, this problem is simplified since the relevant objects are distinguished by their colors (e.g., the ball is orange, the goals are blue and yellow). Nevertheless, fast and reliable color segmentation is not a trivial problem and requires some attention too. Furthermore, object detection may be performed by more than one sensor, such as different virtual sensors based on the vision transducer (e.g., mass center, edge detector, color segmentation), sonars, infrared and others. Therefore, sensor fusion arises as an important topic.

B. Cooperative Multi-Robot Research Problems

Functional and Software Architectures: If a team of cooperative robots is involved, the single-robot architectures of each of the team members must be integrated in the overall team architecture. The most usual solutions are

- *centralized*, where one of the robots (or an external machine) processes the data acquired and sent by all the team members, takes all the team decisions and sends commands to the others;
- *distributed*, where local data processing is made at each of the robots but then information is sent to one of them to take the decisions;
- *fully decentralized*, where each robot takes its own decisions based on its own data and on information exchanged with its teammates.

When the population is composed of heterogenous robots, if a robot has to perform a particular task for which it does not have the necessary actuators, it might ask another robot with the adequate skills to carry it out. In the particular case of the ISocRob robotic team, where the robots are homogeneous, examples of cooperative behavior, in terms of sensors and actuators, are the cooperative localization of the ball and the execution of a pass or the decision of which robot should go for the ball.

Multi-Robot Task Planning and Allocation: In the multiple-robot case, plans must take into account the distributed nature of the task at hand. Different tasks must be allocated to the different robots in the team, according to their skills and performance. Plans must also include synchronization and communication among team members involved in the task.

Multi-Robot Task Coordination: The extension of task coordination to a team of multiple robots introduces issues related to knowledge distribution and maintenance, as well as communications and related problems (e.g., noise, protocols, limited bandwidth). Furthermore, communication can be explicit (e.g., through wireless radio-frequency channels) or implicit (e.g., through the observation of teammates actions, should an *apriori* model of the teammates behavior exist). The coordination of a task carried out by a team of cooperating robots involves signaling events detected by one robot which are relevant for some or all of its teammates and/or to exchange information obtained locally by the different robots of the team. Whenever a formation is required, several formation topologies are possible and the one suitable for the task at hand must be chosen as part of the coordination process. Although not inevitable, communications among team members are also required to keep the formation under control.

Distributed World Modeling: A team composed of multiple robots, possibly heterogeneous concerning on board sensing, can benefit from the availability of a world model, obtained from the observations made by the different team members and its on board sensors. This world model can be richer that if it were obtained by a single robot, due to the coverage of a broader area by a more diverse sensors set. It can also be distributed through the teammates, e.g., by keeping in a single robot information which is only relevant locally and by broadcasting information gathered locally butwhich is of interest for the team as a whole. The sensor fusion problem is similar to the single-robot case, with the important difference that the sensor subsets are now independently mobile and can be actively positioned to improve the determination of object characteristics.

III. PROBLEMS ALREADY ADDRESSED

A key issue of the research work developed under the SocRob project is the application of conceptual results to real robots. However, a basic factor to achieve this endeavor is to ensure hardware reliability so as to avoid spending a large percentage of time handling hardware problems. Therefore, even though the original ISocRob team robots were developed at ISR/IST, they were later replaced by 4 Nomadic Super Scout II commercial

platforms, each of them equipped with the following items:

- Two-wheel differential drive kinematics;
- Sixteen sonar sensors radially distributed around the robot, equally spaced;
- Pentium 233MHz based motherboard, 64MB of RAM, 8GB of hard drive (laptop model), one PCI and one PC104 bus connectors;
- m68k based daughter board with three-axis motor controller, sonar and bumper interface, and battery level meters;
- Two 12V batteries, 18Ah capacity.

The following components were added to enable other functionalities, most of them described in the sequel:

- Ultrak KC7500CP color 1/3" CCD camera with a 4mm, F1.2 lens, in the robot front, through one of the sonar transducer openings;
- Omni-directional catadioptric vision assembly: one MicroVideo MVC26C color CCD camera under a 11cm diameter mirror, manufactured to capture the bird's eye view of the soccer field;
- Bt848 based (Zoltrix TVmax) frame grabber board, with S-VHS and Composite video inputs;
- Pneumatic kicking device, based on Festo components, plus two bottles for pressurized air storage;
- Lucent WaveLAN/IEEE Turbo 11Mbps (Silver) wireless Ethernet modem connected through a PC104/PCMCIA bridge;
- Logitech Optical Mouse, to detect when a robot is blocked.

A. Color Segmentation and Object Recognition

A specific application with a graphical interface that allows a user-friendly adjustment of thresholds in HSV (Hue-Saturation-Value) [8] color space to discriminate all the relevant colors (thus, all the relevant objects) was developed to handle the color segmentation problem. Two look-up tables (LUT), one for each camera image, are compiled from the adjusted thresholds and linked to the main code to provide real-time color segmentation. The LUTs return, for each point in HSV space, either a color label or an `unknown` label, if the point is outside any of the regions defined by the thresholds. Typically, only first-order moments (i.e., the mass center) are then used for sets of pixels classified with the same color, so as to determine the object position. We are currently working on using higher-order moments (e.g., inertia moments and axes) as well as other features (e.g., edges) to attain more robust object segmentation.

B. Vision-Based Self-Localization

An algorithm that determines the posture of a robot, with respect to a given coordinate system, from the observation of natural landmarks of the soccer field, such as the field lines and goals, as well as from *apriori* knowledge of the field geometry, has been developed within the SocRob project[9]. Even though the intersection between the field and the walls is also currently used, the wall replacement by the corresponding field lines would not change the algorithm. The algorithm is a particular implementation of a general method applicable to other well-structured environments, also introduced in [9].

The landmarks are processed from an image taken by an omni-directional vision system, based on a camera plus a convex mirror designed to directly obtain the soccer field bird's eye view, thus preserving the field geometry in the image. The image green-white-green color transitions over a pre-determined number of circles centered with the robot are collected as the set of *transition pixels*. The Hough Transform is applied to the set of transition pixels in a given image, using the polar representation of a line[8]:

$$\rho = x_i^t . \cos\phi + y_i^t . \sin\phi$$

where (x_i^t, y_i^t) are the image coordinates of transition pixel p^t and ρ, ϕ are the line parameters. The q straight lines (ρ_1, ϕ_1), ..., (ρ_q, ϕ_q) corresponding to the top q accumulator cells in Hough space are picked and, for all pairs $\{ (\rho_j, \phi_j), (\rho_k, \phi_k), j,k=1, ...,q, j \neq k \}$ made out of those q straight lines the following distances in Hough space are computed:

$$\Delta\phi = \left|\phi_j - \phi_k\right|$$

$$\Delta\rho = \left|\rho_j - \rho_k\right|$$

Note that a small $\Delta\phi$ denotes almost parallel straight lines, while $\Delta\rho$ is the distance between 2 parallel lines. The $\Delta\phi$ and $\Delta\rho$ values are subsequently classified by *relevance functions* which, based on the knowledge of the field geometry, will filter out lines whose relative orientation and/or distances do not match the actual field relative orientation and/or distances. The remaining lines are correlated, in Hough space, with the geometric field model, so as to obtain the robot posture estimate. An additional step must be taken to disambiguate the robot orientation. In the application to soccer robots, the ambiguity is due to the soccer field symmetry. The goal colors are used to remove such ambiguity.

Currently, the algorithm is used by each of the ISocRob team robots to obtain their self-localization during a game in specific states after a pre-determined timeout has expired. The algorithm is part of each robot navigation system, but it is also used by the robot to share information with its teammates regarding team postures and ball location.

A paper on this work won the Engineering Challenge Award of the RoboCup 2000 Workshop.

C. Multi-Sensor Guidance with Obstacle Avoidance

The ability to navigate at relatively high speeds through an environment cluttered with static and dynamic

obstacles is a crucial issue for a mobile robot. Most robotic tasks require a robot to move to target postures adequate to carry out its planned activities.

In robotic soccer, relevant activities include facing the opponent goal with the ball in between or covering the team goal by positioning itself between the ball and the goal, while avoiding the field walls and the other (stopped and moving) robots. Also relevant is the capability to move towards a given posture while avoiding obstacles and keeping the ball (also known as dribbling).

A guidance control method for non-holonomic (differential drive) vehicles, based on odometry reset by the vision-based self-localization algorithm described before was introduced in [10]. The vehicle must be endowed with a sonar ring, used for obstacle avoidance. The odometry is reset at specific robot states, such as (in the case of soccer robots) at restart time or before returning to home position. The algorithm, designated as *Freezone*, can be generally applied to structured indoors environments, provided that visual features can be observed by the self-localization method.

The Freezone algorithm starts by creating a polar diagram centered with the origin of the robot frame, using the distance-to-obstacles information obtained from the 16-sonar ring. Next, it searches in the diagram all the possible candidate directions that will let the robot move safely. The candidates are the "valleys" of the diagram that allow safe passage of the robot without bumping into obstacles. To find "valleys" in the diagram a mask was created. This mask represents an angular interval with a minimum distance to the obstacles that allow the robot to pass between them. Among the candidate directions, the one with the smallest angle w.r.t. the line connecting the origins of the frames representing the robot current and target postures is chosen. The robot is moved towards the center of the selected valley, with linear and angular speeds determined from the analysis of the robot vicinity (nearby obstacles reduce linear speed) and smoothness considerations (the closer to the desired heading, the slower the robot rotates).

An alternative guidance method has been introduced in [3], consisting of a modified potential fields method for robot navigation, especially suited for differential-drive non-holonomic mobile robots. The potential field is modified so as to enhance the relevance of obstacles in the direction of the robot motion. The relative weight assigned to front and side obstacles can be modified by the adjustment of one physically interpretable parameter. The resulting angular speed and linear acceleration of the robot can be expressed as functions of the linear speed, distance and relative orientation to the obstacles. This formulation enables the assignment of angular and linear velocities for the robot in a natural fashion. Moreover, it leads to an elegant formulation of the constraints on angular speed, linear speed and acceleration that enable a soccer robot to dribble a ball, i.e., to move while avoiding obstacles and pushing the ball without losing it, under severe restrictions to ball holding capabilities. It is shown that, under reasonable physical considerations, the angular speed must be less than a non-linear function of the linear speed and acceleration, which reduces to an affine function of the acceleration/speed ratio when a simplified model of the friction forces on the ball is used and the curvature of the robot trajectory is small.

D. Behavior-Based Architectures

The basic functional architecture of the SocRob team is organized in three levels of decision and responsibility, similar to those proposed in [6]: individual, which is responsible for all functionalities that involve only one agent; relational, which is responsible for the relationships between the robot and its teammates; and, organizational, which is responsible for the strategic decisions that involve the team as a whole. The current instantiation of this functional architecture considers that:

- there is, at the **organizational level**, a mapping from the environment state, including the team state, to a tactical decision, resulting in an organizational behavior displayed by the team. The tactics consists of the set of role assignments to each team member. In robotic soccer, basic roles can be Goalkeeper, Defender and Attacker. Only the captain robot will have the organizer enabled. Should the captain "die", the next robot in a pre-specified list will have its Organizer enabled and become thecaptain.
- there are, at the **relational level**, operators which control relations between two or more team members (e.g., to pass a ball, to avoid moving simultaneously towards a ball, to cover a field region while the teammate advances in the field). Any team member has relational operators running. Each operator has a pre-conditions set and, when this set is satisfied, establishes communications with the relational operator(s) of designated teammates, asking them to start a negotiation process which may end up in a coordinated action among this temporary sub-team. As a result, a *relational behavior* is displayed.
- there are, at the **individual level**, operators consisting of single *primitive tasks* or of *composite tasks* (primitive tasks linked by logical conditions on events).

The software architecture is the practical implementation of the functional architecture, which could be done in any programming language and using different software technologies. In the SocRob project, the software architecture was defined based on three essential concepts: micro-agents (μA for short), *blackboard* and *plugins*.

Inspired by the idea of Society of Agents, proposed by Minsky[11] , each functional module of the SocRob architecture was implemented by a separate process, using the parallel programming technology of threads. In this context a functional module is named μA. In the current

implementation of the SocRob architecture there are nine different threads, but only the three most important ones are mentioned here: μA Vision, responsible for processing the data acquired from the cameras, μA Machine, responsible for deciding which behavior should the robot display, and μA Control, responsible for the execution of the corresponding operator.

The concept of threads was chosen to improve module performance and facilitate the information passing throughout the threads. This was accomplished by the blackboard concept (memory space shared by several threads), further sophisticated here by the development of a distributed blackboard, in what information availability is concerned. Instead of being centralized in one agent, the information is distributed among all team members and communicated when needed.

As mentioned before, the decision making involved for each agent is twofold: which behavior should be displayed, and how the operator which displays such behavior is executed. This separation between behavior decision and operator execution allows the μA Machine, the one responsible for behavior decision, to work with abstract definitions of behaviors, and choose among them without knowing details about their execution. So, new operators could be easily added and removed without affecting the existing ones, and these can also be easily replaced by others with the simple restriction of maintaining the name. This was accomplished using the concept of plugin, in the sense that each new operator is added to the software architecture as a plugin, and therefore the μA Control can be seen as a multiplexer of *plugins*. Examples of already implemented operators are: `dribble`, `score`, `go`, `standby`, to name but a few. The same idea of plugins was also used for the μA Vision, as each particular functionality related to vision data is defined as a different plugin, and multiplexed by the μA Vision (e.g., a plugin for the front camera, a plugin for the up camera, a plugin for the self-localization algorithm, etc.).

Both relational and individual operators have been implemented as state machines. For individual operators, the states represent primitive tasks, while the arcs between states (if any) are traversed upon the validation of given logical conditions over events (e.g., `see ball`, `distance < x`). The relational operator state machines are defined similarly, but events include synchronization signals between the state machines running in the sub-team robots. However, the way the functional architecture was conceptualized allows the implementation of these operators and the switching among them using different approaches, as for example AI production systems.

E. Steps Towards Cooperation and Coordination

Each agent of the team will have access to all individual and relational operators. However, a *role* defines which individual and relational operators are enabled and which ones are not. Furthermore, for a given role, there is always a default (initial) individual operator running until it is temporarily replaced by a relational operator which claims a temporary action. Such an action may consist of executing a primitive task or a composite task, in order to comply with the requirements of an established relation. An example of an already implemented relation is when two robots want to go for the ball. As the ball is a scarce resource, and to avoid that robots from the same team would jeopardize their own actions, a relational operator is needed to manage which robot should go to the ball and which one(s) should not. In the current implementation, each robot that sees the ball and wants to go for it uses a heuristic function to determine a fitness value. This heuristic penalizes robots that are far from the ball, are between the ball and the opposite goal and need to perform a angular correction to center the ball with its kicking device. Each robot broadcasts its own heuristic value, and the robot with the smallest value is allowed to go for the ball whereas the others execute a standby behavior.

IV. Problems To Be Addressed

The previous section shows that most of the work done so far has been concentrated on single-robot behavior and primitive task development. This is a natural consequence of our bottom-up approach to implementation. Nevertheless, many design issues have been analyzed in a top-down perspective since the beginning of the project, and several interesting problems remain to be tackled and solved. We will only mention the currently most important ones.

Behavior Modeling: A consistent model for individual and relational behaviors is required to provide a systematic methodology for behavior synthesis and analysis. Finite state automata (FSA) have been used for this purpose up to now. They have the advantage of the availability of several tools for analysis and synthesis in the literature [2] but suffer from limited modeling capabilities. Petri nets [2] extend the modeling capabilities of FSA and provide a more convenient modeling methodology starting from the identification of the system components and events. A wide range of analysis (e.g., concerning boundedness, liveness, stochastic and deterministic time) and synthesis (e.g., concerning admissible marked languages) tools is also available, and the non-decidability of some analysis problems can be overcome with no significant expenses. Furthermore, modularity and system design can be achieved by interconnecting several sub-systems, each modeled as a Petri net. This is particularly convenient to model relational behaviors, where more than one teammate is involved. So, Petri nets are being investigated as an alternative tool for behavior modeling. Behavior switching can also be modeled as discrete-event systems supervision [2], for which there are results

available regarding FSA and Petri nets. Production systems also have modeling characteristics that make them suitable for this purpose. However, further work must be done to study its design and analysis properties.

Distributed Planning: The available behaviors among which switching is possible are currently designed "by hand". However, a more appropriate approach would be to develop a planner capable of periodically (or when invoked) analyzing the world state and providing a new set of individual and relational behaviors appropriate for the current conditions. A suitable approach should be the continuous interleaving of plan generation and execution. Task allocation among the team robots and distributed world modeling are relevant issues to be further investigated under this topic.

Cooperative Learning: One possible way of designing plans which continuously adapt to new situations and are fine tuned to the actual surrounding environment is to use reinforcement learning (RL) algorithms, especially those which guarantee convergence properties [12]. However, learning is usually slow. An envisaged approach that overcomes this problem is to provide plans with alternative paths among which the RL algorithms can learn to switch over time. Cooperative learning arises when a robot takes its decisions from information learned and provided to it by its teammates.

Control as a Game: Modern views of control state the control problem as a game against an adversary (i.e., the disturbances). In the particular case of soccer, there is an actual opponent whose modeled behavior, once estimated (e.g., using Hidden Markov Models), can be used as information for game-playing algorithms, as part of the planning process.

V. CONCLUSIONS

This paper described the SocRob project (on the development of methodologies for analysis, design and implementation of multi-robot cooperative systems), its objectives, past, current and intended future work. One interesting feature of the project is that it enables different approaches to the solution of the problem at hand. This naturally motivates competing research approaches, as well as research on analysis methods to compare the different results. Furthermore, the project fosters education in AI and Robotics related topics, because so many issues must be solved to handle the overall problem. Students from different levels (undergraduate, graduate, post-doctorate) can get involved at different difficulty levels and accomplish project sub-goals. The SocRob project has involved so far 10 undergraduate and 4 graduate (MSc and PhD) students, besides 2 doctorates who have been supervising the project. All these students have participated regularly in RoboCup - The World Cup of Soccer Robots, since 1998. We believe that RoboCup is a very attractive long-term scientific challenge that brings together people from several different scientific fields in an exciting fusion of research, education and science promotion which are actually the driving forces of our project too.

REFERENCES

[1] C. Canudas de Wit and B. Siciliano and G. Bastin (Eds), *Theory of Robot Control*, CCE Series, Kluwer, 1996

[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999

[3] B. Damas and P. Lima and L. Custódio, "A Modified Potential Fields Method for Robot Navigation Applied to Dribbling in Robotic Soccer", *Proceedings of RoboCup-2002 Symposium*, 2002, Fukuoka, Japan

[4] K. S. Decker and V. R. Lesser, "Designing a Family of Coordination Algorithms", Technical Report No. 94-14, Department of Computer Science, University of Massachussets, Amherst, MA01003, 1995

[5] M. E. desJardins and E. H. Durfee and C. L. Ortiz, Jr and M. J. Wolverton, "A Survey of Research in Distributed, Continual Planning", *AI Magazine*, Winter 1999, pp. 13-22

[6] Drogoul and A. Collinot, "Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: A Case Study in Robotic Soccer", *Autonomous Agents and Multi-Agent Systems Journal*, Vol. 1, pp. 113-129, 1998

[7] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999

[8] R. Gonzalez and R. Woods, *Digital Image Processing*, Addison-Wesley, 1992

[9] C. Marques and P. Lima, "A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor", RoboCup-2000: Robot Soccer World Cup IV, P. Stone, T. Balch, G. Kraetzschmar (Eds.), Springer-Verlag, Berlin, 2001

[10] C. Marques and P. Lima, "Multi-sensor Navigation for Soccer Robots, *RoboCup-2001: Robot Soccer World Cup V*, A. Birk, S. Coradeschi, S. Tadokoro (Eds.), Springer-Verlag, Berlin , 2002

[11] M. Minsky, *The Society of Mind*, Touchstone Publishers, 1988

[12] R. Sutton and A. Barto, *Reinforcement Learning*, MIT Press, Cambridge, MA, 1998