

S.P.Q.R. + Sicilia

RoboCup 2005 Report

L. Iocchi, D. Nardi, A. Cherubini, L. Marchetti, V. A. Ziparo

Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”
Via Salaria 113 - 00198 Roma, Italy
E-mail: {lastname}@dis.uniroma1.it
Web: <http://www.dis.uniroma1.it/~spqr/>

1 Introduction

The Italian Robot Team **SPQR+Sicilia** is the result of a joint effort of two Italian research groups:

S.P.Q.R. (Soccer Player Quadruped Robots, but also *Senatus PopolusQue Romanus*): the group of the Faculty of Engineering at University of Rome “La Sapienza” in Italy, that is involved in RoboCup competitions since 1998 in different leagues (Middle-size 1998-2002, Four-legged since 2001, Real-rescue-robots since 2003). The members of S.P.Q.R. are: Luca Iocchi (Team Leader), Daniele Nardi, Vittorio Amos Ziparo, Andrea Cherubini, Simone Elviretti, Francesca Giannone, Stefano La Cesa, Francesco Macrì, Luca Marchetti, Pier Francesco Palamara, Maria Sciannelli.

SICILIA: the group of the Computer Science Department at University of Palermo that was involved in Middle-size RoboCup competitions in 1998-1999. The members of SICILIA are: Rosario Sorbello (Team Leader), Antonio Chella (Team Leader), Salvatore Maria Anzalone, Giuseppe Balistreri, Rosamaria Elisa Barone, Dario Cacca, Alessandro Cimino, Francesco Cinquegrani, Francesco Di Paola, Ivan Di Paola, Laura Mancuso, Luca Maria Benedetto Martorana, Sergio Orlando, Marco Palermo, Giovanni Reina, Carmelo Scozzola

This paper presents the main development efforts of the team in 2005. In the following sections, before addressing the specific issues that have been developed during this year, we will briefly describe the features of the software architecture. Moreover, specific issues regarding the challenges will be reported in the last section.

2 Software Architecture

The main focus of our research group is the realization of a team of cognitive soccer robots. Past experiences at RoboCup has shown the need of an effective software framework for integrating in a modular way the different components that realize the functionalities of the robots.

However, general robot development frameworks are not easy to be ported on the Sony AIBOs, that have an operating system with different concepts with respect to other systems (e.g. Linux) used on other platforms. Moreover, software architectures for AIBO robots that have the required characteristics have been developed and successfully experimented: German Team Architecture and the CMU's Tekkotsu.

Our choice for development of 2005 team has been to adopt the German Team Architecture [6]. This choice has been motivated not only by considering that it has been successfully used by several teams, but also by the fact that its high modularity allows for developing modules that can be shared with RoboCup Four-Legged community in a more effective way.

Our work in the development of 2005 team has been divided in two parts: 1) porting of some of our previously developed modules into the GT architecture; 2) realizing and experimenting new solutions for some other functionalities. More specifically, we adapted the modules developed in the past regarding Deliberation and Coordination, for which we have realized a porting in the new architecture, and realized and experimented new solutions for other modules, in particular for localization and perception.

The overall goal of our development was thus to integrate our previous achievements and current research in a new robust and effective software architecture.

In the rest of this report we will describe the main components that we developed. From a functional point of view, there are five main functional modules: *Perception*, *Localization*, *Deliberation*, *Control*, and *Coordination*. In the next sections we address these components, by describing the solutions we developed.

3 Perception

Image processing is implemented by both using standard techniques for color segmentation, blob formation, and object recognition and developing new modules that are described in the rest of this section.

3.1 Dynamic color segmentation

In order to improve robustness of color segmentation and to avoid long and tedious calibration processes, we have developed a dynamic color segmentation method that is able to provide a non-parametric color segmentation, that thus does not require any kind of calibration. A detailed description of the algorithm can be found in [4].

Figure 1 shows an example of segmentation. From the original image, we compute the color distribution in two mono-dimensional spaces: the Y component of the color space YUV and the H component of the color space HSV. We process each color distribution by applying a clustering algorithm that aims at grouping color components. Middle part of Figure 1 shows an example of such

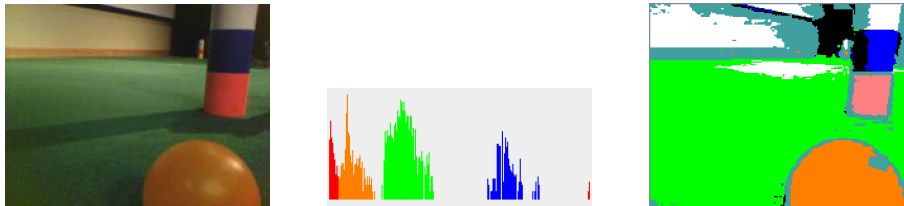


Fig. 1. Dynamic Color Segmentation

clustering for the H component: the X axis represents H values, the Y axis the number of pixels with that value of H, the color of the bar represents the main color that has been associated to each value of H. Since this process is executed on-line the method is very robust in two different situations: 1) variable lights; 2) unknown lighting conditions. Although an extensive evaluation of the method has not been performed yet, we have experimented that is significantly robust on different fields with typical but different RoboCup illumination (i.e. our lab, San Francisco RoboGames, German Open, RoboCup 2005 at Osaka).

The computational resources of the AIBO robots do not allow for a real-time implementation of the described method, i.e. to apply the method at each frame. Nonetheless, it is possible to use the method by computing the color distribution on half-resolution images and by computing a color table every about 15 frames, and then using it for the remaining frames. In this way we obtain an average frame rate of 15 fps, that is suitable for robot play.

Finally, it is important to highlight that a manual well defined color table provides better results with respect to our method. However, we experienced that segmentation obtained by our method is sufficient for object recognition, thus identifying the same elements than a manual color table does.

During RoboCup 2005, we have tested the method not only during the variable lighting challenge, but also in some games (test matches). We check and set parameters the first day, mainly because some colors (carpet and pink colors on markers) were different with respect to the ones in our lab, and never needed to change them during the competition.

4 Localization

Localization on the RoboCup 2005 field is significantly more challenging than in the previous editions. For the last edition of RoboCup, we have developed a method based on Monte Carlo localization, and used it during our match.

4.1 Monte Carlo Localization

Particle Filters such as SIR (Sample Importance Resampling) have been shown to be robust and efficient methods for estimating the state of a dynamic system

[2]. Our approach follows the widely used Monte Carlo Localization method [1], which is supervised by a sensor resetting mechanism such as [5].

The algorithm developed runs in two stage, as usual. In the predict stage of the filter the samples are drawn from the robot motion model. We use the last set of particles and update it using the motion model as proposal distribution.

In the second stage we use the observation to compute the likelihood of samples. In our implementation (see Figure 2) the only eteroceptive sensor used is the built in camera of the robot, so we consider the following features: lines of the field, landmarks, goals.

For all of them we developed an observation model, and we delegate it the evaluation of likelihood. For markers, we consider the angle between it and robot, and the estimated distance from robot. The calculate likelihood can be expressed by:

$$likelihood_{marker} = \exp\left(-\left(\frac{\delta_{angular}^2}{\sigma_{angular}}\right)\right)\exp\left(\frac{\delta_{linear}^2}{\sigma_{linear}measuredDistance}\right) \quad (1)$$

where $\delta_{angular}$ is the difference between calculated and measured angle, δ_{linear} is the linear difference and $\sigma_{angular}$ and σ_{linear} are the respective covariance. The observation model of the goals use the post as marker and don't use the distance. Due to its shape, the posts of a goal can't be detected precisely, so the sigma for the model is more relaxed. The likelihood for single post is:

$$likelihood_{goal} = \exp\left(-\left(\frac{\delta^2}{\sigma}\right)\right) \quad (2)$$

where δ is the difference between calculated and measured angle, and σ is the relative covariance. The lines are extracted in the camera image using the Hough transform. Subsequently, by using the camera transformation and the information provided by the robot joints, we compute the lines local projection. The likelihood of a particular pose depends on the expected set of observations that a robot should see in that particular pose and the perceived items. The likelihood for single line is:

$$likelihood_{line} = \exp\left(-\left(\frac{\delta_{\theta}^2}{\sigma_{theta}} + \frac{\delta_{\rho}^2}{\sigma_{\rho}}\right)\right) \quad (3)$$

where δ_{θ} and δ_{ρ} are the difference on parameters between the observed line and the one from the table.

The final likelihood is:

$$likelihood = likelihood_{marker}likelihood_{goal}likelihood_{line} \quad (4)$$

The method is robust and extensible with respect to the kind of observations considered by the likelihood function. Clearly, its robustness grows with the number of observable items.

4.2 Sensor Resetting

The particle filtering technique has one major drawback: the so called kidnapping problem. When a robot is penalized by the referee, or collides with another robot, the particles evolve inconsistently respect the real pose. A method to resolve this problem was first exposed in [5].

In our implementation we consider the sensor resetting replacing a small amount of particles with low probability with new particles taken from the observation distribution. We calculate the new set of particles using up to three marker observations: with just one marker we calculate a donut-shape distribution, with two or more markers we use triangulation.

For all pose calculated we insert a random gaussian noise to give more chance to new pose to be the correct one.

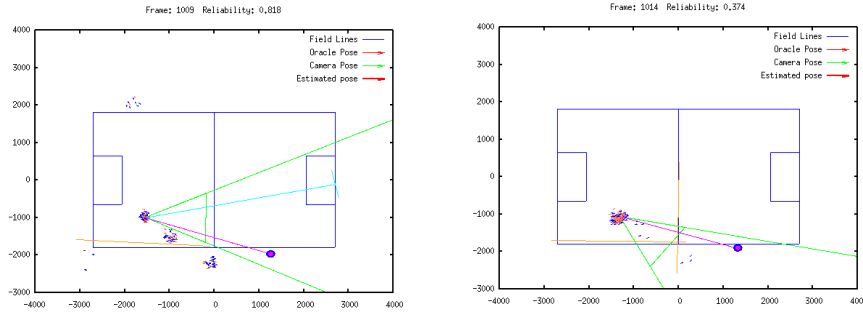


Fig. 2. Monte Carlo based Localization

5 Deliberation

The Deliberation Module aims at driving the decisions of the robot about the actions to be executed, by considering both the current state of the world and the information coming from the Coordination Module. At a higher level the Coordination Module decides which goals should be pursued, while a Plan Executor Module selects (from a library) an appropriate *plan* for execution. Plans represent the means of obtaining goals and are specified in terms of *actions* that contain commands to be delivered to the motion level for effective execution.

5.1 Plan representation and execution through Petri Nets

The behavior of the robot is represented by plans that require a high representation power in order to express the capabilities of the robots.

The structure of AIBO robots allows the execution of concurrent actions, since many mechanical parts of the robots can be controlled independently, e.g. the robots can move their legs and their head at the same time neglecting (from a control standpoint) what other parts are doing. On the other hand it must be taken into account (in a deliberative perspective) the fact that the activity each part is performing will contribute to world evolution by involving its own state transition. Thus, action synchronization is also required. More specifically, it is important to model the following features: 1) sensing actions implementing `if-then-else` or `while` constructs able to determine the behavior of the robot according to the evaluation of a specific property at run-time; 2) concurrency of actions (e.g. moving the head while going to a position); 3) interrupts (e.g. aborting a `go-to-ball` action if the robot can not see the ball); 4) action synchronization between two or more robots. In order to represent the above mentioned features, transition graphs are not adequate, since, for example, concurrency and interrupts can not easily be modeled. Therefore, we adopted a formalism, based on Petri Nets, which is enough expressive to describe plans with concurrent actions (see [7] for further details). Moreover, we developed a plan execution module that can cope with plans described with this formalism. The key idea is to introduce a new *execution* state between the *start* and *end* states of an action. The *execution* state is labeled with *execution conditions*, that must be valid during action execution. Transitions are allowed into and out of this state providing for the capabilities of simply defining action synchronization and recovery from failures. Action concurrency is simply implemented by allowing multiple edges to come out from the same state.

As a difference with other approaches based on extensions of transition graphs, such as XABSL [6], we clearly distinguish action specification and implementation, obtaining a framework which permits easier debugging: first, the semantic is well defined and easily verifiable by automated verification programs; second, we have a high granularity of actions which are grouped by functional properties and physical resources used.

Plans can be generated by an off-line planner or edited by hand. In the latter case we use available open-source tools which can generate an appropriate standard XML format (PNML). If transitions and places are correctly labeled to meet the specification of the Petri Plan the PNML code can be parsed to produce executable representation on the robot (PET files).

Moreover, the tool Jarp¹ has been extended in order to debug existing plans. During plan execution, the robot can produce a log containing the information regarding the deliberative process. This log can be parsed by our tool to view the evolution of the Petri Net Plans, allowing for easily identifying loops or wrong behaviors, and providing a quick and user friendly plan debugging interface.

¹ <http://jarp.sourceforge.net/>

6 Control

The control module is responsible for implementing all motion commands generated by the Deliberation module. This involves computing and managing low-level motor commands for the robot's 20 joints. This issue mainly concerns with walking control, static actions control (i.e. kicks and goalkeeper blocks) and head control.

The walking control issue has been thoroughly addressed by the RoboCup community in recent years, and little improvement can be done as long as optimal speed is concerned. However, we have seen the necessity of developing a different walking pose for the goalkeeper. In fact on the RoboCup 2005 field, where the goal width has increased by 200 mm, the goalkeeper is extremely important. Therefore, our approach is to use a set of parameters (a pose) for the goalkeeper walk which optimizes stability and makes ball-blocking easier, instead of aiming at maximum gait speed. For example, if the goalkeeper walks with the front legs in a wider and lower position than the other robots, it can protect a larger part of the goal with its body.

Since kicks and blocks are also an important element in RoboCup, we developed of new static actions. Our approach is to consider a small and effective set of kicks, addressing real game situations. These kicks are designed in order to optimize power and precision, but also to improve the safety of the robots.

7 Coordination

7.1 Coordination through dynamic role assignment

The Coordination Module is responsible for implementing a distributed coordination protocol for dynamic task assignment according to the current situation of the environment [3]. The approach relies on the definition of a set of roles, that are ordered with respect to importance in the domain, and a set of utility functions, one for each role, expressing the capability of a robot to play this role. The values of the utility functions are computed by all the robots in the team for all the roles and these data are exchanged broadcast through the wireless network. A coordination algorithm is then executed by each robot in order to determine robot-role assignment. This algorithm ensures that every role is assigned to one robot and that the robots assigned to the most important roles are those that are in the best situation (according to the values of the utility functions).

8 Challenges

8.1 Variable Lighting Challenge

The Variable Light challenge was performed by using the dynamic color segmentation method described in Section 3.1. As already mentioned the method not only is robust to variable light conditions, but also allows for being used without any knowledge of the environment (i.e. without any kind of calibration).

References

1. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI99)*, 1999.
2. N. Gordon, D. Salmond, and C Ewing. A novell approach to nonlinear nongaussian bayesian estimation. In *In Proceedings F.*, pages 107–113, 1993.
3. L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa. Distributed coordination in heterogeneous multi-robot systems. *Autonomous Robots*, 15(2):155–168, 2003.
4. Luca Iocchi. Dynamic color segmentation for the robocup environment. Technical report, Dip. di Informatica e Sistemistica, 2005. Available from <http://www.dis.uniroma1.it/~spqr>.
5. S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of International Conference on Robotics and Automation (ICRA'00)*, 2000.
6. T. Röfer et al. German team robocup 2004. Technical report, 2004. Available from <http://www.germanteam.org/>.
7. Vittorio Amos Ziparo. Representation and execution of plans based on petri nets. Technical report, Dip. di Informatica e Sistemistica, 2005. Available from <http://www.dis.uniroma1.it/~spqr>.