Team Chaos 2005

H. Martínez[1], V. Matellán[2], M.A. Cazorla[3], A. Saffiotti[4], D. Herrero[1], F. Martín[2], B. Bonev[3], and K. LeBlanc[4]

[1] Department of Information and Communication Engineering
University of Murcia, E-30100 Murcia, Spain
[2] Systems and Communicatiosn Group
Rey Juan Carlos University, E-28933 Madrid, Spain
[3] Department of Computer Science and Artificial Intelligence
University of Alicante, E-03690 Alicante, Spain
[4] Center for Applied Autonomous Sensor Systems
Örebro University, S-70182 Örebro, Sweden

Abstract. "Team Chaos" is a multi-university team which has been competing in the 4-legged robot league of RoboCup since 1999. This paper shortly describes the Team Chaos entry for RoboCup 2005. The most distinctive points of our team are: (i) a general, principled architecture for autonomous systems, (ii) hierarchical fuzzy behaviors for fast incremental development of robust behaviors, (iii) fuzzy landmark-based localization for efficient, fault-tolerant self-localization. The main improvements for 2005 are: (i) improved vision-based perception, (ii) natural landmarks (corners on field) for self-localization, and (iii) improved cooperative perception and behavior.

1 Introduction

"Team Chaos" is a cooperative effort which involves universities in Spain and Sweden. In 2005, the sites of activity are: University of Murcia (coordinating node), University of Alicante and Rey Juan Carlos University from Spain, and Örebro University from Sweden. Team Chaos is a follow-up of Team Sweden, which was created in 1999 and has participated in the 4-legged league of RoboCup ever since. The distributed nature of the team has made the project organization demanding but has resulted in a rewarding scientific and human cooperation experience.

We had two main requirements in mind when we joined RoboCup. First, we wanted our entry to effectively address the challenges involved in managing uncertainty in the domain, where perception and execution are affected by errors and imprecision. Second, we wanted our entry to illustrate our research in autonomous robotics, and incorporate general techniques that could be reused on different robots and in different domains.

Team Leader: Humberto Martínez Barberá (`humberto@um.es`)
Web (includes publications): `http://www.aass.oru.se/Agora/RoboCup/`

2 Architecture

Each robot uses the layered architecture shown in Fiure. 1. This is a variant of the ThinkingCap architecture, which is a framework for building autonomous robots jointly developed by Örebro University and the University of Murcia [9]. The lower layer (implemented in the CMD, or Commander) provides an abstract interface to the sensori-motoric functionalities of the robot. The middle layer maintains a consistent representation of the space around the robot (through the PAM, or Perceptual Anchoring Module), and implements a set of robust tactical behaviours, implemented in a hierarchical manner (in the HBM, or Hierarchical Behavior Module). The higher layer maintains a global map of the field (kept in the GM, or Global Map), and makes real-time strategic decisions based on the current situation (situation assessment and role selection is performed in the HFSM, or Hierarchical Finite State Machine). Radio communication is used to exchange position and coordination information with other robots (via the TCM, or Team Communication Module).

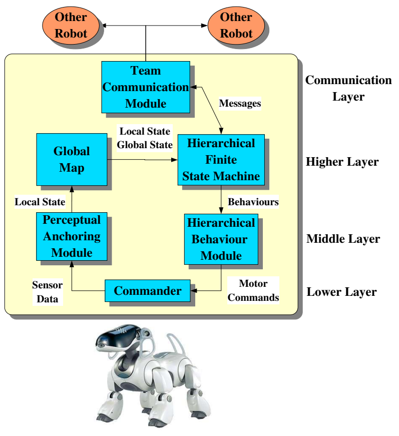

Fig. 1. The variant of the ThinkingCap architecture.

This architecture provides effective modularisation as well as clean interfaces, making it easy to develop different parts of it. Furthermore, its distributed implementation allows the execution of each module in a computer or robot indifferently. For instance, the low level modules can be executed onboard a robot and the high level modules can be executed offboard, where some debugging tools are available. However, a distributed implementation generates serious synchronisation problems. This causes delays in decisions and robots cannot react fast enough to dynamic changes in the environment. For this reason we have favoured the implementation of a mixed mode architecture: at compilation time it is decided whether it will be a distributed version (each module is a thread, Figure 1) or a monolithic one (the whole architecture is a thread and the communication module is another, Figure. 2).

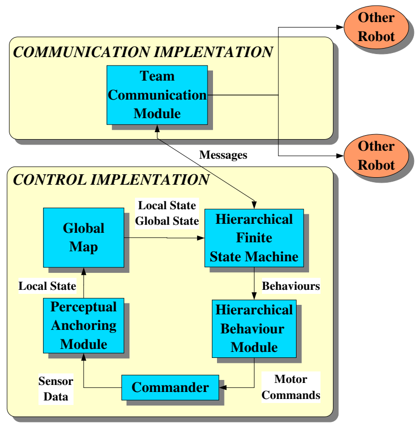
COMMUNICATION IMPLENTATION
Team Communication Module
Other Robot
Other Robot
Messages
CONTROL IMPLENTATION
Local State
Global State
Global Map
Hierarchical Finite State Machine
Local State
Behaviours
Perceptual Anchoring Module
Hierarchical Behaviour Module
Sensor Data
Commander
Motor Commands

Fig. 2. Implementation of the ThinkingCap architecture.

3 Locomotion and Motion control

The Commander module accepts locomotion commands from the HBM in terms of linear and rotational velocities, and translates them to an appropriate walking style. This simplifies the writing of motion behaviors, which are easily portable between different (legged and/or wheeled) platforms. The Commander also implements several types of kicks. This module also controls head movement, and implements scan and look commands.

3.1 Walking style calibration

Our walking styles are based on the code for parametric walk created by the University of New South Wales (UNSW) [3], and we are currently porting the walk created by the Université de Versailles [10]. Two important problems are related with the walking style: (i) finding a good set of parameters that provide a stable walk and high speed; and (ii) finding a good odometry function that allows an accurate localisation.

We identify two different optimization problems. One is the maximisation of the robot's speed. As surfaces' characteristics vary from one place to another, it would be very difficult to find an analytical solution. The other problem is improving the correspondence between commanded and achieved speed. If the difference is high, not only the robot will exhibit an strange behaviour but also it will fool the localisation subsystem. The approach we propose is a parameters optimization method with an empirical evaluation function which evaluates the real speed of the dog. The parametric space has eight dimensions and the evaluation function is quite time consuming, which precludes the use of brute force space search.

We apply a simulated annealing technique for learning an appropriate set of parameters and use a straight line with marks placed at fixed distances. The idea is building a ruler that can be deployed in different arenas. The real speed is measured by counting the ruler markers using the vision system (Figure 3). These techniques will allow the improvement of our locomotion module, which was one the most unreliable module of our system, and allow for a better positioning in the 2005 large field.

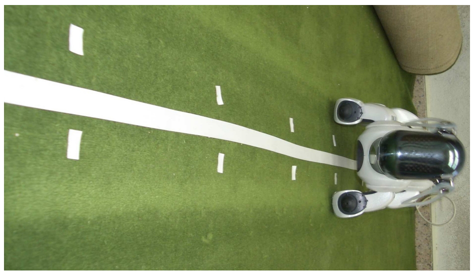

Fig. 3. Ruler for measuring the real speed

3.2 Kicks

Kicking is a fundamental hability for soccer playing. The robot will use it to score a goal, to pass to a teammate, and in our architecture even to block an opponent shot. Basically, we define a kick as a movement sequence that the robot performs when the behavior module (HBM) activates it, and when a given set of preconditions are satisfied. The working schema is as follows:

1. The behavior module (HBM) sends a set of kick, selected among all the available ones, to the motion module (CMD).
2. The CMD module activates these kicks, but it does not performs any of them till the preconditions are satisfied.
3. Each kick is configured with a set of preconditions regarding the relative position of the ball to the robot. The CMD module checks the preconditions for each kick. If the position of the robot and the ball satisfyies the kick conditions, the kick is performed.

Figure 4 shows the precondition (in blue) associated to a particular kick. If this kick is activated by HBM and the ball is in the blue area, this kick will be performed.

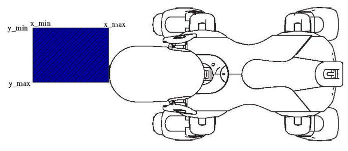

Fig. 4. Kick area condition

We have classified our kicks into two categories:

- **Open loop**. This kind of kick is used for short kicks and they are executed in open loop mode. These kicks cannot be interrupted and they do not use any sensory information. The open-looped kicks we have currently developed are:
  - *LeftLegKick* The ball is kicked with the left leg.
  - *RightLegKick* The ball is kicked with the right leg.
  - *TwoLegKick* The ball is kicked with both legs.
  - *BlockKick* This kick is used for blocking a ball. The robot extends both legs enlarging the blocking area.
  - *LeftHeadKick* The robot uses its head for kicking the ball at its left direction.
  - *RightHeadKick* The robot uses its head for kicking the ball at its right direction.
  - *FrontHeadKick* The robot kicks the ball with its head for push the ball at its front. This kicks is a very effective one.
  - *GetUp* This kick is performed when the robot detects it has fallen down.
- **Closed loop** In this kind of kicks, the robot is getting information from its sensors while it is performing the kick. If a condition is not satisfied (i.e. the ball is not close enough) during the execution, the kick is aborted. These kicks can be divided into several phases. These phases starts and stops depending on the sensory information. Currently we have just one of these kicks.
  - *GrabAndTurn* This kick makes the robot grabs the ball and turns. First, the robot orientates to the target with the ball grabbed. Then, the robot pushes the ball. If the robot lost the ball while it turns, the kick is aborted. This situation is detected by its infrared sensor situated between both legs.

4 Behaviors and Roles

Our behavior system consists of two well separated layers. First one, low level layer, is based on fuzzy logic techniques. The HBM implements a set of navigation and ball control behaviors realized using fuzzy logic techniques, and organized in a hierarchical manner. For instance, the following set of fuzzy rules implement the "GoToPosition" behavior.

```
IF (AND(NOT(PositionHere), PositionLeft))  TURN (LEFT);
IF (AND(NOT(PositionHere), PositionRight)) TURN (RIGHT);
IF (OR(PositionHere, PositionAhead))       TURN (AHEAD);
IF (AND(NOT(PositionHere), PositionAhead)) GO (FAST);
IF (OR(PositionHere, NOT(PositionAhead)))  GO (STAY);
```

More complex behaviors are achieved by combining simpler ones using fuzzy meta-rules which activate concurrent sub-behaviors. Behaviors also incorporate perceptual rules used to communicate the perceptual needs of active behaviors to the PAM.

For the high level layer, we implement a hierarchical finite state machine (HFSM), which is derived from the tool created by the Université de Versailles [10]. We have reimplemented this tool adding some new features: monitoring the current state, reusing states from different HFSM (Figure 11). A HFSM is a set of states, metastates, which are state machines as well, and transitions between states and/or metastates. When the dog is in a state, it executes the code in the state. This code is OpenR, using perception (ball, nets, landmarks, dog position, etc.) both local and global, and shared information from other dogs. We have a mechanism to call low level behavior. This mechanism allow us to call a concrete behavior (for example, GoToBall) and select a kind of kick (KickForward). This kick is active and it will be shoot if low level conditions (ball position) are correct. Thus, several kick can be actived and a low level selection process chooses which of them will be shooted. Transitions in the HFSM are conditions to change between states (ball position, global dog position, etc.).

In the definition of roles, we distinguish between field player and goal keeper. The goal keeper is always the same dog and always will execute the same HFSM. However, field players can play different roles in different game conditions. For example, if a defender goes to goal it changes its role to attacker and another dog should change to defender. This can be easily achieve defining several metastates and sharing information between dogs in order to know when to change. Figure 11 shows an example of the attacker role.

5 Perception

The locus of perception is the PAM, which acts as a short term memory of the location of the objects around the robot. At every moment, the PAM contains the best available estimates of the positions of these objects. Estimates are updated by a combination of three mechanisms: by *perceptual anchoring*, whenever the object is detected by vision; by *odometry*, whenever the robot moves; and by *global information*, whenever the robot re-localizes. Global information can incorporate information received from other robots (e.g. the ball position).

The PAM also takes care of selective gaze control, by moving the camera according to the current perceptual needs, which are communicated by the HBM in the form of a degree of importance attached to each object in the environment. The PAM uses these degrees to guarantee that all currently needed objects are perceptually anchored as often as possible (see [5] for details).

Object recognition in the PAM relies on three techniques: *color segmentation* based on a fast region-growing algorithm; *model-based region fusion* to combine color blobs into features; and *knowledge-based filters* to eliminate false positives. For instance, a yellow blob over a pink one are fused into a landmark; however, this landmark may be rejected if it is, for example, too high or too low relative to the field.

This year, significant improvements have been made to the PAM. Previously, seeds for the growing algorithm were obtained by hardware color segmentation on YUV images, taken directly from the camera. Seeds are now chosen by performing software thresholding on the images, which are first converted to HSV. This allows for a very portable and robust color segmentation, which works even in the face of changing lighting conditions.

In addition to the normal objects in the RoboCup domain, we also detect natural features, such as field lines and corners. Currently we use corners comprised of the (white) field lines on the (green) carpet. Corners provide useful information, since they can be classified by their type, and they are relatively easy to track (given the small field of view of the camera). Natural feature detection is performed using three techniques: *corner detection* based on changes in the direction of the brightness gradient; *color-based filtering* for filtering out corners that aren't on the carpet; and *corner grouping* for associating the detected corners with the natural features for which we are looking ([8]).

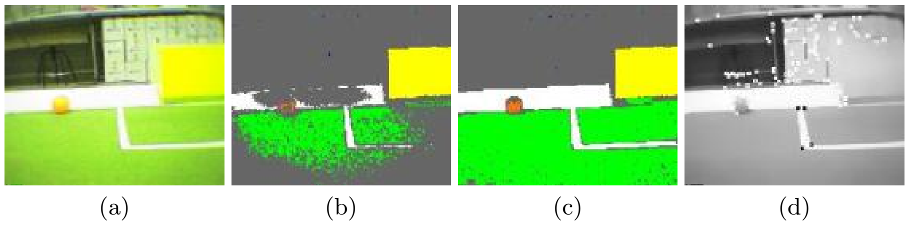

Fig. 5. (a) Raw image. (b) Segmented image (software thresholding). (c) Segmented image after region growing. (d) Gradient-based corner detection (white) and corners after color filtering (black).

The first step in our algorithm is to segment the raw image using a simple thresholding algorithm. The result of this step on a sample image is shown in Fig. 5(b). The next step is to apply the Seed Region Growing (SRG) algorithm, which starts from a number of seed pixels (which are "safe" pixels of each color), and grows the region until a discontinuity in the color space is reached. The details of this algorithm are explained in [7]. The result of the growing step is shown in Fig. 5(c). Corners are detected using a gradient-based method, and a color filter is applied to determine which corners are on the carpet. Fig. 5(d) shows the corners obtained from the gradient-based method (in white), and those which survived the color filtering process (in black).

6 Self-Localization

Self-localization in the 4-legged robot league is a challenging task: odometric information is extremely inaccurate; landmarks can only be observed sporadically since a single camera is needed for many tasks; and visual recognition is subject to unpredictable errors (e.g., mislabeling). To meet these challenges, we have developed a self-localization technique based on fuzzy logic, reported in [1]. This technique needs only qualitative motion and sensor models, can accommodate sporadic observations during normal motion, can recover from arbitrarily large errors, and has a low computational cost. The result of self-localization is used to make strategic decisions inside the HFSM, and to exchange information between robots in field coordinates.

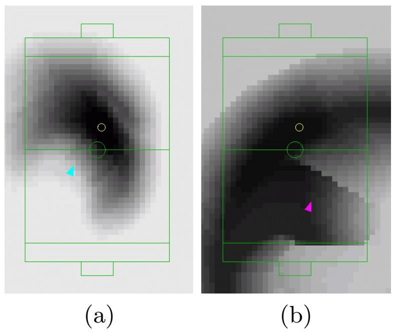

Fig. 6. Self localization grids.

This technique, implemented in the GM module, relies on the integration of approximate position information, derived from observations of landmarks and nets, into a fuzzy position grid — see Fig. 6. To include own motion information, we dilate the grid using a fuzzy mathematical morphology operator. Using this technique, our robots can maintain a position estimate with an average error of approximately $\pm 20\,cm$ and $\pm 10°$ during normal game situations. Localization is done continuously during normal action. Stopping the robot to re-localize is only needed occasionally (e.g. in case of major errors due to an undetected collision).

The extension to our method which includes natural features as landmarks is described in [8]. The extended technique allows the robot to localize using only the nets and natural features of the environment (i.e. corners).

The observation of a landmark constrains the possible robot positions to be on a partial circle around the observed landmark (see Fig. 6 (b)). However, since our feature detector doesn't provide unique identifiers for corners, the possible robot positions induced by a corner observation is the union of several circles, each centered around a corner in the map. This can be seen in Fig. 7. It should be noted that the ability to handle this ambiguity efficiently is a primary advantage

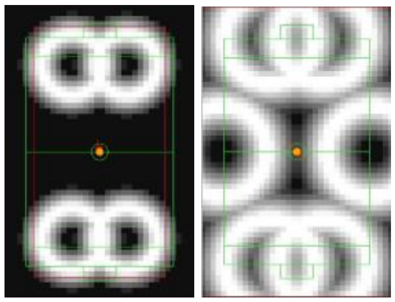
(a) (b)

Fig. 7. Position grid induced by observation of two different types of corners. Due to symmetry of the field, the center of gravity is close to the middle of the field.

of our representation. The data association problem is automatically addressed in the fusion process (Fig. 8). Data association is a difficult problem, and many existing localization techniques are unable to adequately address it.

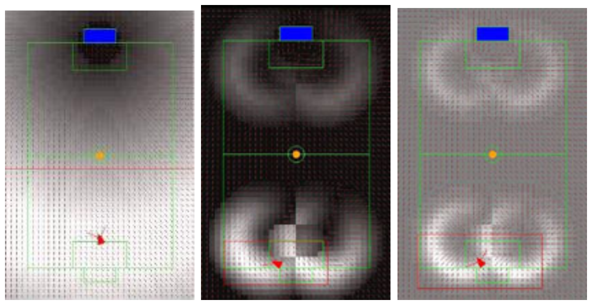
(a) (b) (c)

Fig. 8. Belief induced by the observation of (a) a net, (b) the first feature, and (c) the second feature. Initially the position of the robot is unknown (belief distributed along the full field).

7 Information sharing

We use radio communication to exchange information between robots about the positions of objects in the field, in particular the ball. Information from other robots is fused using an original approach based on fuzzy logic, which is reported in [2]. In our approach we see each robot as an expert which provides unreliable information about the locations of objects in the environment. The information

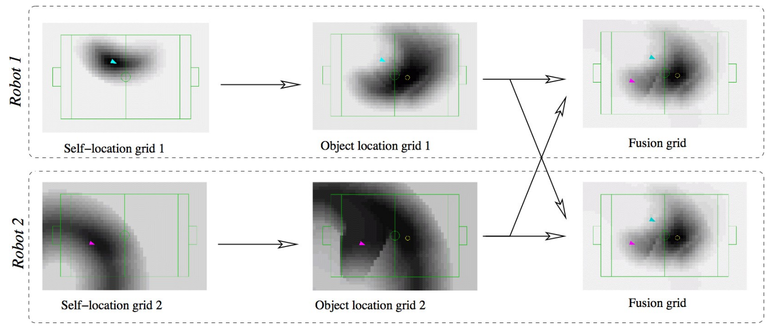

Fig. 9. Schema used for information fusion.

provided by different robots is combined using fuzzy logic techniques, in order to achieve agreement between the robots.

This contrasts with most current techniques, many of which average the information provided by different robots in some way, and can incur well-known problems when information is unreliable. Our method strives to obtain a consensus between data sources, whereas most other methods try to achieve a compromise or tradeoff between sources.

One of the major innovations of this approach is that it consistently maintains and propagates uncertainty about the robots own position into the estimates of object positions in the environment. Moreover, in contrast to many existing approaches, it does not require high accuracy in self-localization.

The schema used for cooperative ball localization is shown in Fig. 9. The self position grids for both robots are shown on the left, the ball location grids are shown in the middle, and the result of the fusion of the ball grids is shown on the right. Darker cells have higher degrees of possibility. The two triangles represent the robot's (defuzzified) estimates of their own positions. The yellow circles show the point estimates of the ball position.

An example of our method can be seen in Fig. 10. The left window shows the ball grid resulting from the sharing of information. The three small circles near the bottom represent the point estimates of the ball position according to each robot (lighter circles) and as a result of the fusion (darker circle). The other two windows show the self-localization grids for robots 1 and 2, respectively. In this example, both robots happen to have a rather poor self-localization, which can be seen from the blurring of the two individual self-grids. Correspondingly, the individual estimates for the ball positions are relatively inaccurate, and quite different from each other. When intersecting the fuzzy sets, however, we obtain a fairly accurate fused estimate of the ball position (left window). Note that just averaging the estimates of the ball position produced by the two robots independently would result in an inaccurate estimate of the ball position.

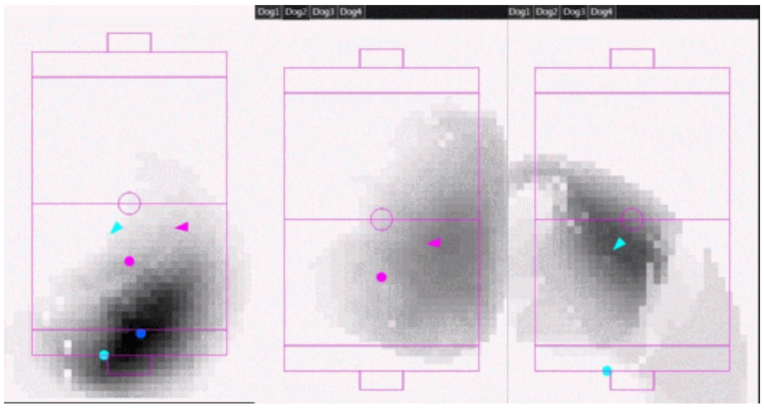

Fig. 10. Combining two imprecise estimates into an accurate one.

8 Support tools

Last year we started the development of an application that incorporates a set of tools for helping the configuration and calibration of the different modules. This application is called Chaos Manager. The underlying technology of this application is the use of the UDP protocol for communicating the robots and the computer. The tool was used last year for color calibration and kick development (Figure 11).

This year we have enhanced the tool to also include walking style parameters calibration, behaviour monitoring, and team performance monitoring (Figure 11). In the first case, the operator can follow the learning process and log all the parameters, in the second case the operator can monitor the current behaviour and role of the robot while playing, while in the third case the operator can know the internal state of each robot (position grid, shared information, and perceived ball position).

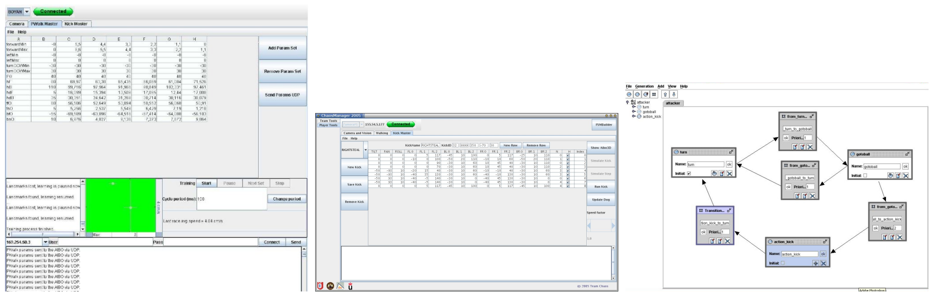

Fig. 11. ChaosManager: a) walking style calibration b) kick definition c) behaviour state machine

9 Conclusion

The general principles and techniques developed in our research have been successfully applied to the RoboCup domain. In particular, fuzzy logic was beneficial in writing robust behaviors, providing reliable self-localization, and achieving effective cooperative perception.

Acknowledgements

Team Chaos gratefully acknowledges the support of the Spanish Ministry of Education and Science (grant CICYT DPI2004-07993-C03-02), Frutas “El Dulze” (Murcia, Spain), the Swedish KK foundation, and the four participating universities.

References

1. P. Buschka, A. Saffiotti, and Z. Wasik. Fuzzy landmark-based localization for a legged robot. *IEEE/RSJ Int. Conf. on Intell. Robots and Systems* (IROS), 2000.
2. J.P. Canovas, K. LeBlanc, and A. Saffiotti. Robust multi-robot object localization using fuzzy logic. *Proc. of the Int. RoboCup Symposium*, Lisbon, PT, 2004.
3. B. Hengst, B. Ibbotson, P. Pham, and C. Sammut. Omnidirectional locomotion for quadruped robots. Birk, Coradeschi, Tadokoro (eds) *RoboCup 2001*, Springer-Verlag, 2002.
4. A. Saffiotti. Using fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(4):180–197, 1997. Online at http://www.aass.oru.se/Agora/FLAR/.
5. A. Saffiotti and K. LeBlanc. Active perceptual anchoring of robot behavior in a dynamic environment. *IEEE Int. Conf. Robotics and Automation* (ICRA), 2000.
6. A. Saffiotti and Z. Wasik. Using hierarchical fuzzy behaviors in the RoboCup domain. In: Zhou, Maravall, Ruan (eds) *Autonomous Robotic Systems*, Springer-Verlag, 2002, pp. 235-262.
7. Z. Wasik and A. Saffiotti. Robust color segmentation for the RoboCup domain. *IEEE Int. Conf. on Pattern Recognition* (ICPR), 2002.
8. D. Herrero-Perez, H. Martinez-Barbera, and A. Saffiotti. Fuzzy Self-Localization using Natural Features in the Four-Legged League. *Proc. of the Int. RoboCup Symposium*, Lisbon, PT, 2004.
9. H. Martínez and A. Saffiotti ThinkingCap-II Architecture. Online at http://ants.dif.um.es/~humberto/robots/tc2/
10. V. Hugel French Team ”Les Trois Mousquetaires”. Online at http://www.lrv.uvsq.fr/research/legged/robocup.php