Designing a minimal reactive goalie for the RoboCup SPL

Juan F. García, Francisco J. Rodríguez, Camino Fernández, and Vicente Matellán
Departamento de Ingeniería Mecánica, Informática y Aeroespacial
Escuela de Ingenierías Industrial e Informática, Universidad de León, 24071 León
{jfgars, fjrodl, camino.fernandez, vicente.matellan}@unileon.es

***Abstract*—This paper presents the basic design and implementation of a goalkeeper made according to the regulations of the two-legged Standard Platform League of the RoboCup Federation. The paper describes the perceptive schemas created using the architecture of the TeamChaos-URJC team as well as the action schemes designed to create a minimal reactive goalie. This player was tested in the 2009 German Open international competition. The results obtained there are analyzed and the future works derived from that analysis are presented.**

***Index Terms*—RoboCup, reactive, attention, vision, humanoid schema**

I. Introduction

ROBOCUP (Robotic soccer WorldCup) is an international research and education initiative, which has put forward a standard problem to promote the research on artificial intelligence and intelligent robotics. For this purpose, RoboCup Federation[1] has chosen soccer as a basic domain, and is organizing a robotic soccer World Cup in different categories, leagues and Academic Conferences since 1997.

The work described in this paper has been developed according to the Standard Platform League (SPL) regulations and tested during the German Open 2009[2] in April 2009. In this league all teams use the same hardware platform, the Nao robot (see Fig. 1). This robots are manufactured by Aldebaran Robotics, so the focus of this competition is on the software controlling the robot.

Nao robot is a 21 degrees of freedom humanoid, whose height is 57 cm and its weight is around 4.5 Kg. It has two 30 fps video cameras located in the forehead and in the mouth, each one with a maximum resolution of 640x480, but they cannot be used simultaneously. The switch between cameras takes too long and the field of view is scarcely overlaped so they are not capable of stereo vision.

All control is made onboard using a x86 AMD Geode chip at 500 MHz, 256 MB of SDRAM memory and a standard 1 Gb in flash memory that can be upgraded. It also has WiFi (802.11g) and Ethernet connections. Concerning the sensors, apart from the cameras, it has 2 gyroscopes and 3 accelerometers, 2 bumper sensors in the feet, and 2 ultrasonic sensors in the chest.

Nao operating system is a Linux Embedded version. It can be programmed using a propietary SDK called NaoQi that supplies bindings for C, C++, Ruby, and Urbi. It is also compatible with the robot simulator Webots by Cyberbotics[3].

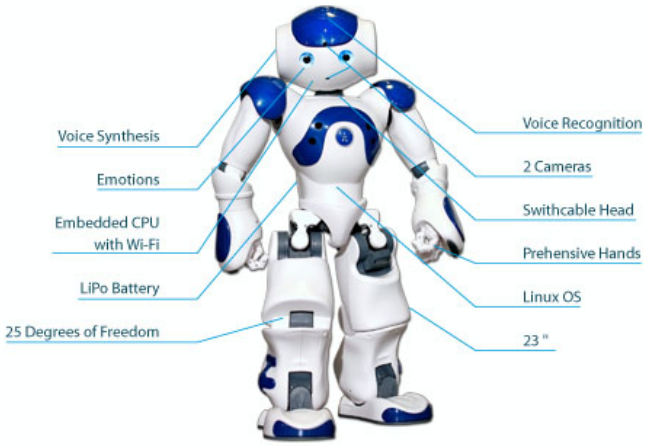

Fig. 1. Nao robot (figure copyrighted by Aldebaran Robotics)

The rest of the paper is organized as follows, in the second section goalkeeper restriction are enumerated. In the third section the architecture used to build the software is explained, both the principles and the software structure are detailed. In the forth section the perception algorithms used are described, and the attention mechanisms are sketched. Finally, in the last section, the results obtained in the RoboCup German Open are analyzed and also the future works envisioned are enumerated.

II. Goalkeeper Restrictions

Using this robot, we needed to implement a goalkeeper in a short period of time since the platform is new and barely a year has passed since migration from AIBO. Of course, the main task of the goalie is of course to prevent the opponent team from scoring. To do this, the goalie has to look for the ball and try to place itself between its goal and the ball. If the ball is close enough, the goalie has to kick it taking care of not doing it towards our goal. In our experience with the previous platform of the SPL, the AIBO [11], there is a critical restriction for this behavior: the robot should not leave the penalty area. If the robot goes far of its goal, the chances of the opponent team to score are very high. This

[1] http://www.robocup.org/
[2] http://www.robocup-german-open.de/en
[3] http://www.cyberbotics.com/products/webots

restriction, combined with the lack of time to port our previous localization code ([14], [15]) to the new platform, forced us to take some design decisions in order to simplify the whole process:

1) Our goalie will use a set of movements limited to moving right and left, like a table soccer player.
2) It will only use indirect odometry to solve self-localisation, that is, will use information about the movement actions taken.
3) The robot is independent and will be a reactive player.
4) We will implement some kind of attention mechanism to filter only the ball and the goal net of the opponent for our goalie to stay aligned to our goal.
5) We want to implement it according to the software architecture of our team (TeamChaos).

Taking into account that Nao Standard League is quite new, the literature is not centered in a special role player but in the Nao players like a team ([3],[4]). For a quickly approach to other Nao goalkeepers, we follow [2] that explains basic schemas to be assigned to differents players and [1] that makes a brief explanation about it.

III. Architecture

Building software to create autonomous behavior in robots is a complex task that cannot be restarted from scratch for every robot and problem. A conceptual architecture has to be used, and this architecture has to be reflected in a software toolkit. Many different architectures have been proposed in the literature to organize the software of automous robots [6]. In this work, we have decided that we will use the architecture implemented in TeamChaos-URJC[4].

This architecture is based on the JDE architecture [7]. Basically, the behavior of a robot is generated by JDE and is a hierarchy of reactive behaviors that are sensitive to goal modulation too (*schemas*). This organization lets the systems react quickly to external estimuli due to low level control loops. High level schemas can select the estimuli to which the low level schemas should react, using the modulation parameters and biasing this way the behavior to goal-oriented movements. Detailed information about JDE, and projects using it, can be found in the web page of the project: `http:/jde.gsyc.es`.

Though inspired in JDE, the current code of the TeamChaos does not use the JDE software development suite. It uses a simplified version where all the schemas inherit from the abstract class `Controller`. All the schemas will have to implement these three methods:

- `Activate`: represents the initialization. This method is used to create the basic mechanisms of the controller. Depending on the level, we can use methods of the lower level, which again will call methods in the lower level till we reach the NaoQi methods.

[4]TeamChaos is a joint effort of the Rey Juan Carlos Univ., Univ. de Murcia, Univ., Rovira i Virgili, and the Univ. of León. We compete together in RoboCup but we do it with different configurations in local and regional events.

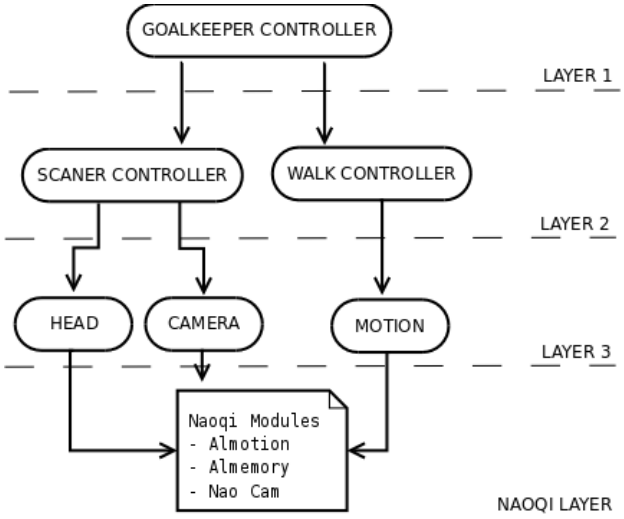

Fig. 2. Hierarchy of controllers implementing the goalie behavior

- `DeActivate`: used when finishing the work of the class. It also has to call the `DeActivate` methods of the lower classes to orderly finish, that is, erasing states created by higher classes, and the movements sequences generated, cleaning them as well.
- `Step`: every time an event is generated, the `step` method has to be invoked in the controller of its level. There are two main steps, one of them related to the scanner controller and the other one connected to the walk controller. Both are independent tasks and it is the data produced in the scanner controller the one which generate a new state in the walk controller.

The architecture of schemas we have designed for implementing the goalie can be seen in figure 2. The top level *GoalKeeper Controller* (GKC), will coordinate the movement of the head when tracking the ball, and the movement of the robot right or left to try to intercept it.

The lower level of figure 2 (Naoqi Layer) is the software API provided by the manufacturer. NaoQi is based on a client-server architecture, where NaoQi itself acts as a server. The modules plug into NaoQi either as a library or as a broker, with the latter communicating over IP with NaoQi. Running a module as a broker, thus, has some IP-related overhead, but also allows the module to be run on a remote machine, which is very useful for debugging.

As we have previously stated, we decided to implement our software over this level, which let us use the hardware in a semi-transparent way, that is, we can send high level movement commands without taking care of implementing a low level waking generator engine.

"Layer 3" in figure 2 is the control of the low level parameters needed to deal with the Naoqi API. It receives values from the upper level, and translates errors and exceptions generated in Naoqi to the upper controller. The main goal of this layer is to let our code be independent of Naoqi API. In the future, if we would like to change, for instance, the walking generator from Naoqi to ours, we just need to change this level.

In "Layer 2" the first controllers, that is, schemas that

will work by themselves, generating the movement of the robot and obtaining and processing images from the camera. These controllers, according to our architecture philosophy, are autonomous, that is, they can deal with errors and they adapt to the current situation while working on its goal.

Finally, in "Layer 1", the coordination between system vision, the ball tracking system and the robot movement is made. This is the highest level and it also has to take care of the initialization of the system, the communication with the game controller, and the finalization of the system.

We have to remember that the big difference between the low level layer (Naoqi) and the top layers is the event subscription. The communications between the naoqi layer and layer three are executed using the primitive functions that Aldebaran provides. All the interchanged information in the other layers is carried out modifiying the appropiate variables.

IV. Perception mechanisms

Processing a stream of video in limited hardware as a robot is a high time-consuming task. In particular, in robotic soccer we need to process images on-board at frame rate, that is, at 25 Hz. In order to achieve it, we have decided to implement some kind of visual attention [5]. Basically, the main idea is that not all areas in a given visual scene are relevant to the task at hand. Therefore by restricting attention to the relevant parts of the scene, the agent can greatly increase its visual processing speed. This intuition is corroborated by work in cognitive science confirming that human vision processing takes advantage of selective attention [13].

In robotic vision, selective attention can take two main forms. One is "gaze control", in which a robot moves its camera so that its field of view is faced towards the important information [10]. That approach is analogous to human eye saccading, but does not address the question of how to process each image, an often time-consuming process.

The other approach to selective attention involves processing only the areas of the image that are likely to have relevant features. Because of the large amount of data in every image, processing each image entirely is difficult to achieve at frame rate, and if possible, it severely limits the amount of time the robot can spend taking care of other issues as localization, trajectory generation, planning, etc.. By restricting its attention to the parts of the image that are most likely to contain the important information, the robot can speed up its image processing. This raises the challenge of identifying the useful areas of the image.

We will take both approaches: we will first move the camera looking for an image which includes orange pixels and then we will process only the orange blobs in it, discarding any other information, to determine if they verify all conditions to be considered a ball.

Next subsections analyze each controller and the ball perception mechanics.

A. Head Controller

The first type of attention is implemented in our goalie by the `Head Controller`. Its mission is to seek for and continiously track the ball. Actions associated to this controller are basically two: movement controlling , and image analysis.

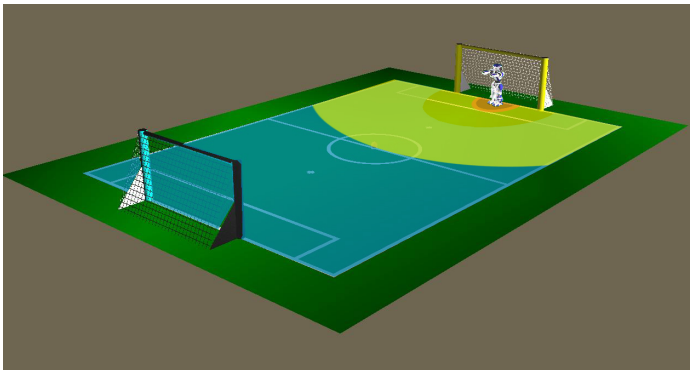

Fig. 3. Field view

We have implemented four types of scanning: high (t_1), medium (t_2), low (t_3) and very low (t_4). All of them but the latter one use the upper camera. The four types use different values of the pitch (*HeadPitch* (HP) and *HeadYaw* (HY). In t_1 the HP is $\frac{\pi}{7}$ $radians$ up, in t_2 0, in t_3 and t_4 $\frac{\pi}{7}$ $radians$ down, using the upper camera for t_3 and the lower camera for t_4 . The controller modifies HY to get a continuous movement left to right. We will not be using t_1 for the release version since we do not need to look for any object which is so high above the horizon, so we will be working with t_2, t_3 and t_4. The scaned area for each of them is represented in Fig. 3: the area closer to the robot corresponds to t_4, the second closer to t_3 and the last one to t_4. Pitch angles and minimum and maximum distance covered for each scan type are represented in Tab. I.

The type of scanning is chosen by the `GKC` by activating the `Scanner Controller` with the adequate parameters. It is also GKC which switches between them if the ball is not located.

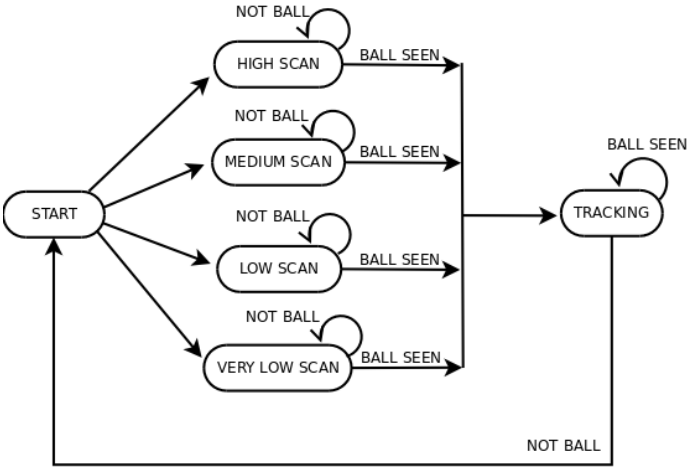

Fig. 4. Scanner state automata

In Fig. 4 we represent the state automata for `Scanner Controller`: upon activation from the `GKC`, a scan type is selected. By default, the chosen one will be Medium Scan. The automata will remain in this state looking for the ball until it

SCANNER MODE	CAMERA	PITCH ANGLE (rad)	MinDISTANCE (m)	MaxDISTANCE (m)
Medium	upper	0	1.63	∞
Low	upper	$\frac{\pi}{7}$	0.51	3.00
Very Low	lower	$\frac{\pi}{7}$	0.03	0.58

TABLE I
HEAD SCANNING TYPES

is found or until the GKC changes the scan type, whatever happens first. After the whole amplitude of the field of view for the current scan type has been checked unsuccessfully the scan type is changed. If the ball was found, the autamata will enter in tracking state, in which it will remain until the ball is lost, which will restart the scan routine.

The goal of the scanning process is to obtain images of different areas of the field. These images have to be analyzed. There are several "interesting things" we have to look for in the image:

- Ball
- Beacons
- Goals
- Players

The number and type of the items we are looking for represent the second type of attention this controller has to deal with. The most important one for a reactive goalie is the ball. As we are designing a reactive goalie we are not using a self-localizing method based on perception, so we do not need to look for other things. In the second version we will use the opposite goal to get aligned, or the field lines of the goal area to estimate the position.

B. Ball Perception

The perception of the ball is based on color segmentation and shape analysis. We can suppose that the only orange pixels will be the ones belonging to the ball. Obviously, this will be in the ideal world, because in the real one there are many noisy points that can look orange, even if the calibration is good. We could add some other restrictions (i.e. an artificial horizon to filter false positives), but previously we need to eliminate the isolated points (that we consider noise).

Basically, for every orange pixel found in the image ($image[i,j]$), we check how many orange pixels are around, and we set an arbitrary percentage threshold to consider it as an isolate point. If the percentage is lower than the threshold, that pixels is not segmented:

$$if\,\frac{OPC}{PC} < minP \text{ then erase}$$

where:

PC: is the number of pixels
OPC : is the number of orange pixels counted
minP : is the threshold

Once we have filtered the isolated points, we can check if the orange pixels look like a circumference, and thus discard orange objects which are nor round shaped (see Fig. 7). It will be done in two consecutive steps: During first step we will calculate the radius of the smallest circle which includes all orange pixels. Then we will analyze the shape, evaluating the percentage of orange points in the circle and, if it is above a given threshold, we will assume we have found the ball.

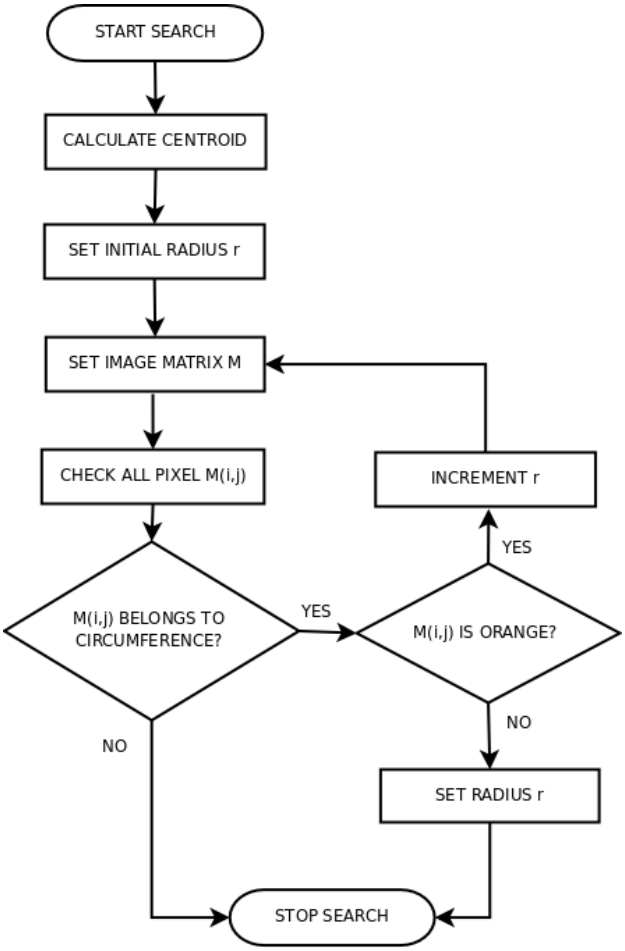

Fig. 5. Step one: circle's radius calculation

Step one: circle's radius calculation.

First, orange points centroid is calculated and an initial radius value is set (Fig. 5). Then, a circumference of the given radius centered at the centroid and its circumscribed image submatrix are set. All the submatrix pixels which belong to the circumference are analyzed until an orange one is found and hence the radius value is increased. The whole process will be repeated until an iteration where no orange pixel appears is reached.

Step two: shape analysis.

Pixels counter (PC) and orange pixels counter (OPC) are set (Fig. 6). Then, a circle of the previously obtained radius centered at the centroid and its circumscribed image submatrix are set. All the submatrix pixels which belong to the circle are analyzed in order to check if they are orange. For every

pixel checked PC is increased, so is OPC in case the given pixel is orange. Once the whole circle has been computed, the proportion of orange points is calculated: OPC/PC. If the result is above a given threshold, the object inside the submatrix is round shaped and can be considered the ball.

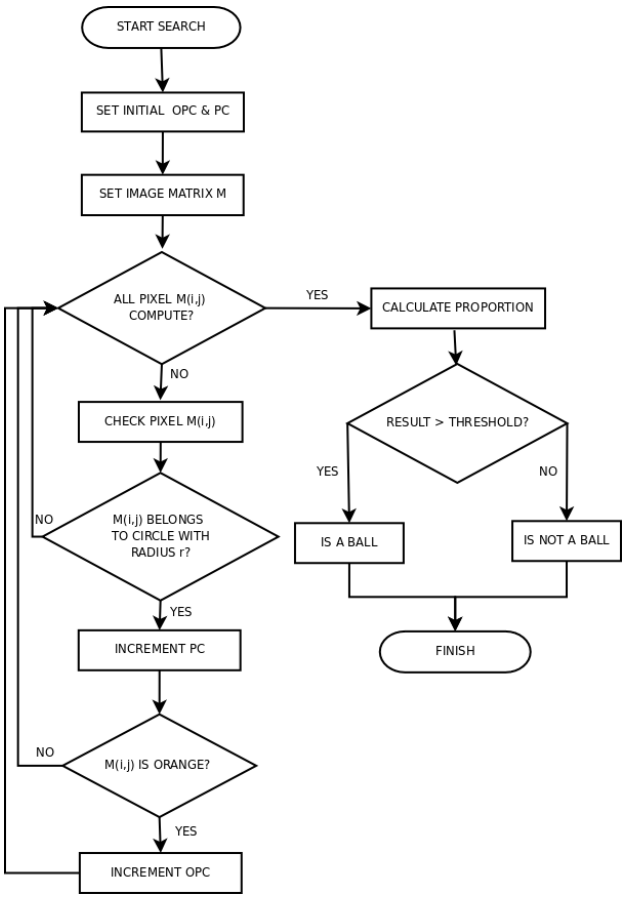

Fig. 6. Step two: shape analysis

Please note that during radius r calculation we are working with circumferences' points while during shape analysis we are working with the circle of radius r. First impression could be that percentage of orange points should be computed at the same time we are looking for the radius value while in first step and that second step is not neccesary. However, that is not possible because we are working with and integer indexed matrix: (i, j) matrix elements, which correspond with image pixels, are integer values instead of reals, and thus most of them will not verify the circumference equation. A possible solution to this problen is the one we have chosen: first we get the radius value, r, and then we get the orange pixels proportion using the circle of radius r instead of working with each isolated circumference.

Once the ball has been found in the current image (see Fig. 8), we estimate its distance to the robot based on the height of the ball divided by the image height (ρ). We have not used any interpolation, we have just consider four different cases with empirical data:

1) $if \rho \geq 30\%$ then $distance \leq 0.40m$
2) $if 30\% > \rho \geq 15\%$ then $0.40m < distance < 1.00m$

Fig. 7. Orange object discarded since it is not round shaped

Fig. 8. Ball found

3) $if 15\% > \rho \geq 5\%$ then $1.00m < distance < 3.00m$
4) $if 5\% > \rho$ then $distance < 1m$

The information about the estimated distance to the ball, and the orientation to it, is obtained by inverse kinematics when the tracking controller has centered it. This information will be used by the other controllers, in particular, the `GKC`, to move the robot: the `GKC` is subscribed to all information relative to the ball (orientation and distance), so that any time one of these parameters changes a control method inside `GKC` is called. The `Scanner Controller` keeps updating all this information, and the `GCK` will take the decision to move or stay in the current position based upon those paremeters values. In the `GKC` section we explain the movement decision a little further.

C. Walk Controller

The `Walk Controller` basically moves the robot laterally. The number of steps is calculated according to the distance and orientation to the ball. The maximum number of steps ($MaxSteps$) is given by the size of the step ($stepD$), and the distance to goal post:

$$MaxSteps = \frac{stepD*10^3}{2^{stepD*10^2-2}}$$

D. GoalKeeper Controller

This is the top level controller that manages the beginning and the end of the automata. This controller will analyze the data obtained by the `Scanner Controller`, basically the position of the ball, and according to it, it will set the parameters of the `Walk Controller`, see Fig. 9. This decision will be based in the angle to the ball (θ_{ball}), and it is a discrete and proportional response:

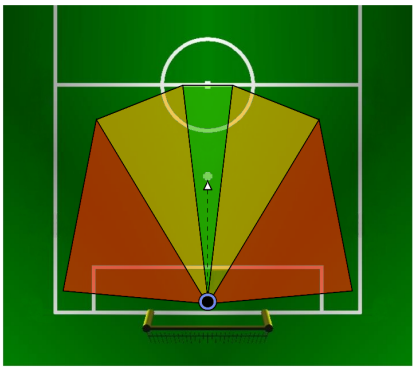

Fig. 9. Ranges for the movement decision

- If $0 < |\theta_{ball}| < \frac{\pi}{18} rad$ the goalie is centered: do not move.
- If $10 < |\theta_{ball}| < \frac{\pi}{6} rad$ the ball is slightly to one side of the goalie: one lateral step.
- If $30 < |\theta_{ball}| < \frac{\pi}{3} rad$ the goalie is really misplaced: two lateral steps.

The movement will be right or left according to the orientation of the ball.

V. Experiments, results and further work

This goalie was tested in the 2009 edition of the RoboCup German Open, unofficially considered the European championship, held in Hannover (Germany) from the 20th to the 24th of April 2009. The goalie was part of the TeamChaos-URJC, one of the two teams from the TeamChaos consortium that entered the competition. We compete as different teams in local competitions in order to test different algorithms, and compete as a unified team in the official RoboCup WorldCup.

The results of the team in Hannover were not satisfactory. Score of the preliminary phase is summarized in table III. We did not qualify for the semifinals (the other team TeamChaos-UMU got it with the same results: 2 points, -2 goals difference). As it can be seen in table III, we only received two goals, one of them while the goalkeeper was inactive due to an internal Game Controller's listener error, so the major problem was that we were not able to score any. But this is a simple analysis. In fact, we were not scored more goals because other teams strikers were also very limited.

The performance of the goalie was poor. It was extremely slow. As shown by table II, it was able to detect the ball in less than 2400ms if a good color calibration had been made, given the ball was in the visual field of the current scan type, and in case it was not changing scan type would consume less than 400ms. Nevertheless, a more elaborated vision technique for tracking could be used, thus, when the ball is lost, information on the last known position could be useful to detect it again. The biggest trouble, however, is that it takes a huge amount of time for the keeper to move, almost 6 seconds per step, as a result of using Naoqi primitives. We have to consider developing our locomotion method.

For the 2009 RoboCup, to be held in Austria in June 2009, two main works have been envisioned:

1) Refactoring the vision system: we plan to eliminate the overheads detected and use faster algorithms. For instance, we will use a modified version of the visual sonar proposed by the CMU team [8].
2) Incorporate stationary actions for intercepting the ball, that is, reactions of the player that do not move its position, but one leg, one foot, or one hand to try to intercept the ball.

We need also to include a self-localisation system, probably we will try to adapt the method [14] developed for the previous league (the one using the Aibo robots), combined with the detection of the field lines of the goal area that has already been implemented in our team [12] for the Aibos.

Acknowledgment

We want to express our gratitude and acknowledgment to all members of our team, in particular to the members of the robotics group of the Rey Juan Carlos university for their support, help, and inspiration of the ideas described in this paper.

The authors would also like to thank the Spanish Ministry of Innovation for its support to this project under the grant DPI2007-66556-C03-01(COCOGROM project).

ACTION	TIME (ms)
Cover all bases	2340
Change angle level (medium - low)	380
Change angle level (low - medium)	380
Stand up after ball seen	2032
Step	5670

TABLE II
EXPERIMENTAL RESULTS

	Humboldt	Dortmund	**TeamChaos-URJC**	HTWK	L3M	GA	GR	GD	Pts	Rank
Nao Team Humboldt	X	0:0	**0:0**	1:2	0:0	1	2	-1	3	3
Nao Devils Dortmund	0:0	X	**1:0**	0:0	1:0	2	0	2	8	2
TeamChaos-URJC	**0:0**	**0:1**	X	**0:1**	**0:0**	**0**	**2**	**-2**	**2**	**4**
Nao-Team HTWK	2:1	0:0	**1:0**	X	3:0	6	1	5	10	1
Les 3 Mousquetaires	0:0	0:1	**0:0**	0:3	X	0	4	-4	2	5

TABLE III
GERMAN OPEN 2009: ROUND ROBIN POOL A

REFERENCES

[1] Andreas Panakos et al *Kouretes 2008, Nao Team Report* Robocup 2008

[2] J. Ruiz-del-Solar, P. Guerrero, R. Palma-Amestoy, M. Arenas,R. Dodds, R. Marchant, L. A. Herrera *UChile Kiltros 2008 Team Description Paper*. Robocup 2008

[3] Todd Hester, Michael Quinlan and Peter Stone *UT Austin Villa 2008: Standing On Two Legs*. Technical Report UT-AI-TR-08-8

[4] Aaron Tay, *Exploration of Nao and its Locomotive Abilities* Exploration of Nao and its Locomotive Abilities,2008

[5] Daniel Stronger and Peter Stone. *Selective Visual Attention for Object Detection on a Legged Robot*, in Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorenti, and Tomoichi Takahashi, editors, RoboCup-2006, Springer Verlag, 2007.

[6] José María Cañas y Vicente Matellán. *From Bio-inspired vs. Psycho-inspired to Etho-inspired robots*. Robotics and Autonomous Systems. Vol.55, Num. 12, pp. 841-850. DOI:10.1016/j.robot.2007.07.010

[7] J.M.Cañas, J. Ruíz-Ayúcar, C. Agüero, F. Martín. *JDE-neoc: component oriented software architecture for robotics*. . Journal of Physical Agents, Volume 1, Number 1, pp 1-6, 2007.

[8] Scott Lenser and Manuela Veloso. *Visual sonar: Fast obstacle avoidance using monocular vision*. In Proceedings of IROS, pp. 391-406, October 2003.

[9] James Bruce, Tucker Balch, and Manuela Veloso. *Fast and inexpensive color image segmentation for interactive robots*. In Proceedings of IROS, Japan, October 2000.

[10] L. Itti, C. Koch, and E. Niebur, *A model of saliency-based visual attention for rapid scene analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1254 - 1259, Nov 1998.

[11] H. Martínez, V. Matellan, M. Cazorla, A. Saffiotti, D. Herrero, F. Martin, B. Bonev, K. LeBlanc. *Robotics soccer with Aibos*. In Proceedings of WAF 2005 (Workshop de Agentes Físicos) Granada (Spain). ppp 69 - 71

[12] David Herrero Pérez, Humberto Martínez Barberá. *Robust and Efficient Field Features Detection for Localization*. RoboCup 2006

[13] Pilar Bachiller, Pablo Bustos and Luis J. Manso. *Attentional Selection for Action in Mobile Robots*. I-Tech, pp. 472, October 2008.

[14] Francisco Martín, Vicente Matellán, José María Cañas y Pablo Barrera. *Localization of legged robots based on fuzzy logic and a population of extended kalman filters*. Robotics and Autonomous Systems. Vol.55, Num. 12, pp. 870-880.doi:10.1016/j.robot.2007.09.006

[15] Renato Samperio, Housheng Hu, Francisco Martín, Vicente Matellán. *A hybrid approach to fast and accurate localisation for legged robots* . Robotica International, Cambridge Journals. Vol. 26, Num. 06, pp. 817-830. DOI:10.1017/S0263574708004414.