# Model and Behavior-Based Robotic Goalkeeper

Hans Lausen, Jakob Nielsen, Michael Nielsen*, and Pedro Lima

Instituto de Sistemas e Robótica – Instituto Superior Técnico
Av. Rovisco Pais, 1 – 1049-001 Lisboa, Portugal
{gruber,fisker,miguel}@control.auc.dk, pal@isr.ist.utl.pt
http://socrob.isr.ist.utl.pt

**Abstract.** This paper describes the design, implementation and test of a goalkeeper robot for the Middle-Size League of RoboCup. The goalkeeper task is implemented by a set of primitive tasks and behaviors, coordinated by a 2-level hierarchical state machine. The primitive tasks concerning complex motion control are implemented by a non-linear control algorithm, adapted to the different task goals (e.g., follow the ball or intercept the ball). One of the top level behaviors regularly determines the robot posture from local features extracted from images acquired by a catadioptric omni-directional vision system. Most robot parameters were designed based on simulations carried out with the Hybrid Automata Matlab/Simulink toolbox CheckMate. Results obtained with the actual goalkeeper are presented and discussed.

## 1 Introduction and Overview

In robotic soccer, namely in the Middle-Size League (MSL) of RoboCup, goalkeepers are interesting robots, due to the potential behavior richness they can exhibit. Indiveri [1] introduces an elegant solution for a goalkeeper based on a non-linear state-feedback control algorithm. However, little is said about the coordination of a larger set of behaviors rather than the one corresponding to ball following and blocking. Jamzad *et al* [2] describe a very efficient goalkeeper with changing shape, but they mostly concentrate on its mechanical design. Menegatti *et al* [3] were the first to introduce in RoboCup a multi-behavior goalkeeper which usually moves on an arc in front of the goal and tries to intercept the ball when a shot is headed at its goal. Nevertheless, they use an ad-hoc model for both motion control and the overall behavior coordinator.

The coordinated execution of a robotic task is one of the key features for an autonomous robot. The robot resources (e.g., sensors, actuators, shared memory, CPU) required to accomplish a given task must be properly managed and articulated with the different behaviors composing the task.

In the RoboCup MSL, one of the four allowed players is the goalkeeper. A good goalkeeper should switch among different behaviors to fulfill its role in the team.

* The three first authors are with Aalborg University and did their work at ISR/IST under a SOCRATES grant.

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 169–180, 2004.

In this paper, the design, implementation and test of a a goalkeeper is described. The goalkeeper task is implemented by a set of primitive tasks and behaviors, coordinated by a 2-level hierarchical state machine. The primitive tasks concerning complex motion control are implemented by a non-linear control algorithm, adapted to the different task goals (e.g., follow the ball or intercept the ball), as detailed in Section 2. One of the top level behaviors, described in Section 3, regularly determines the robot posture from local features extracted from images acquired by a catadioptric omni-directional vision system. Most robot parameters were designed based on simulations carried out with the Hybrid Automata Matlab/Simulink toolbox CheckMate, as covered in Section 4. Results obtained with the actual goalkeeper are presented and discussed in Section 5. Conclusions and future work are discussed in Section 6.

The RoboCup MSL *ISocRob* team consists of four *Nomadic Scout II* robots, endowed with an omnidirectional camera and a front camera, both *Philips TouCam Pro* web-cams. The goalkeeper's kicker and cameras assembly, shown in Figure 1-a) is different from its teammates, namely due to a general 90° relative rotation of those hardware components and a larger surface for the kicking device.

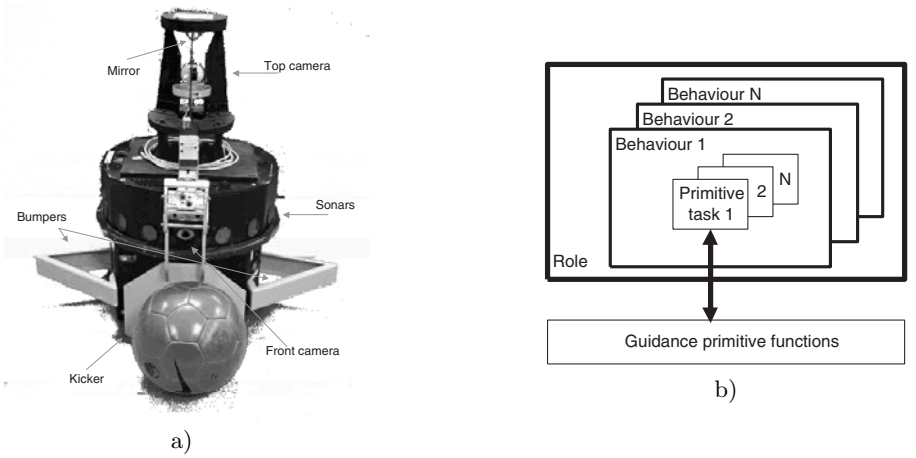



**Fig. 1.** ISocRob's a) goalkeeper robot; b) functional architecture

ISocRob's functional architecture is based on the concepts of roles, behaviors, primitive tasks and guidance primitive functions[4]. Figure 1-b) shows how the four elements interact with each other. Each robot is assigned a role, which consists of one or more behaviors. Behaviors are implemented using one or more primitive tasks, and each primitive task uses the guidance primitive functions to interact with the lowest level of hardware on the robot. At the top level, a role is assigned to each robot. In this case, the goalkeeper role is assigned. The role is filled by executing one of the role's behaviors, according to the current robot + environment state. Each behaviors is assigned to a state of a state machine

whose arcs are traversed whenever some logical condition associated to the robot + environment state or the lower-level state machine becomes true. A behavior is executed by running a number of primitive tasks, coordinated by the lower-level state machine, where each state represents a primitive task and arcs are again traversed when logical predicates over state variables become true.

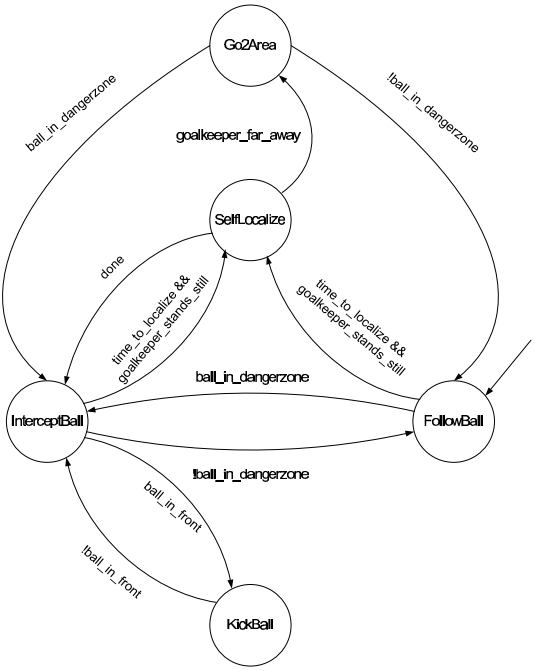



**Fig. 2.** The goalkeeper state machine

The state machine of the implemented goalkeeper is depicted in Figure 2. There are five behaviors in the state machine. The `Go2Area` behavior is implemented by one of the navigation algorithms introduced in previous papers[5]. `KickBall` is trivial. The other three behaviors (`InterceptBall`, `SelfLocalize`, and `FollowBall`) are detailed in the remaining sections of the paper.

The robot is in `FollowBall` behavior if the ball is out of the danger zone specified in Figure 3-a). The ball is not considered as a big threat, but the goalkeeper has to be able to handle a shot from the distance. The goalkeeper follows the ball while tracking an arc with adjustable radius and centered with the goal. The principle of the defensive arc is illustrated also in Figure 3-a). The `r_min` and `r_max` parameters correspond to the minimum and maximum radius of this arc. This adjustment of the radius is dependent of the distance between the ball and the goal. When the ball is near the border of the danger zone, the radius of the arc is at its minimum. When the ball is at the center line or further away, the radius is at its maximum. The robot assumes the behavior `InterceptBall`


**Fig. 3.** Concepts of danger zone, maximum and minimum defensive arcs. a) `FollowBall`; b) `InterceptBall`

when the ball is in the danger zone in Figure 3-a) and moving towards the goal as seen in Figure 3-b). In this case, the goalkeeper is not following the arc anymore but rather moving on a straight line in front of the goal, trying to intercept an incoming ball.

Transitions between behaviors occur only when the associated predicates, shown in the state diagram of Figure 2, become true.

## 2 Motion Control

Both the `FollowBall` and the `InterceptBall` behaviors pose trajectory tracking and posture stabilization problems [6]. Nevertheless, the trajectories to be tracked will be different, therefore each behavior will have its own control algorithm, which will be derived in following subsections. Before that, we will take a brief look at the robot kinematics, required to provide the terminology for the rest of the section.

### 2.1 Kinematics

The goalkeeper is based on a differential-drive robot, that can be described by the same kinematic equations as the unicycle vehicle. The unicycle is a nonholonomic system with no slippage assumed. Let $q = (x, y, \phi)^T$ be the vector that describes the goalkeeper posture in configuration space. The first-order kinematic model for the unicycle is given by [7]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} u\ \cos(\phi) \\ u\ sin(\phi) \\ \omega \end{bmatrix} \tag{1}$$

Where $u$ is the linear velocity and $\omega$ is the angular velocity. Furthermore $u$ and $\omega$ are the control inputs.

### 2.2 FollowBall Behavior

Moving on an arc in front of the goal is a trajectory tracking problem. Among the possible solutions for this problem, we have chosen the algorithm described in [1],

used in its original form to implement the `FollowBall`. The goal of the control design is to track and follow the arc in front of the goal until an equilibrium point (in front of the ball) is reached for the goalkeeper, as well as to achieve asymptotic stability for that equilibrium point.

Consider the kinematic model from equation 1. If the position is given in polar-coordinates an equivalent kinematic model would be:

$$\begin{bmatrix} \dot{r} \\ \dot{\alpha} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} u\ \cos(\alpha) \\ \omega\ -\ \frac{u}{r}\ sin(\alpha) \\ \frac{u}{r}\ sin(\alpha) \end{bmatrix} \tag{2}$$

Where the angle $\alpha$ is given as shown on Figure 4-a).

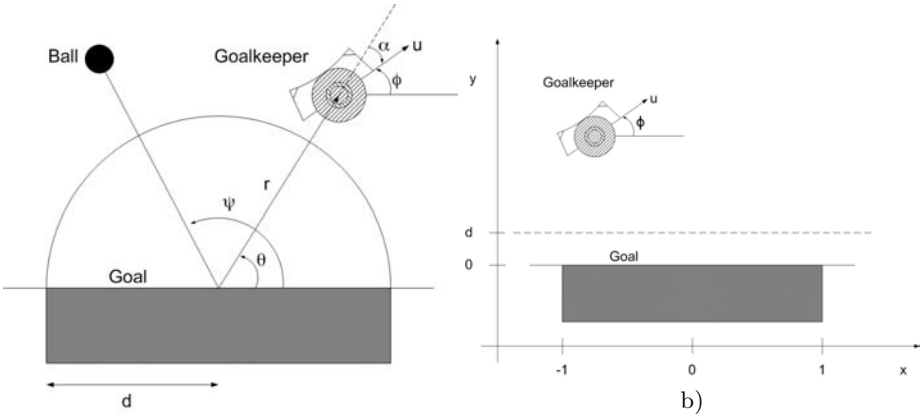



**Fig. 4.** The goalkeeper posture in polar coordinates: a) `FollowBall` scenario; b) `InterceptBall` scenario

In the figure, the goalkeeper is on the arc when $r = d$. Furthermore the angle $\alpha$ should be equal to 90° when the goalkeeper is moving on the arc in order to keep the front towards the ball. These two requirements are expressed in the following error vector:

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} d - r \\ \frac{\pi}{2} - \alpha \end{bmatrix} \tag{3}$$

To achieve asymptotic stability a control law for $u$ and $\omega$ must be derived in such a way that the time derivative of an appropriate Lyapunov function $V(e)$ becomes negative definite [8, 9]. The approach for the choice of $\omega$ is to cancel some of the undesirable terms which makes it difficult to determine the nature of $\dot{V}(e)$. The following control law for $\omega$ guarantees aymptotic stability[9]:

$$\omega = \frac{u}{r}\ \sin(\alpha)\ -\ \gamma\ \left(\frac{\pi}{2} - \alpha\right)\ -\ h\ u\ \left(d - r\right)\ \frac{\cos(\alpha)}{\frac{\pi}{2} - \alpha} \quad ,\ h > 0\ , \gamma > 0 \tag{4}$$

### 2.3 InterceptBall Behavior

If the ball moves inside the danger zone the goalkeeper switches its behavior to `InterceptBall`. When running this behavior, the goalkeeper should defend the goal on a straight line in front of the goal. It is most likely that the posture of the goalkeeper is on the arc, when the upper-level state machine switches to `InterceptBall`. The control algorithm should therefore not only be able to track the straight line, but also to make the goalkeeper able to converge to the line from any posture on the arc. This sounds similar to the trajectory tracking problem of the previous subsection, for a different trajectory. It is therefore chosen to re-derive the control algorithm used to control the angular velocity in `FollowBall` so that the equilibrium point is now on a straight line in front of the goal. Since the reasoning for the design already has been given in the previous subsection the control algorithm design will not be as detailed here. However, it is important to underline that this derivation is one of the original contributions of this work.

Consider again the kinematic model from equation 1. The straight line to be tracked, at a distance $d$ of the goal line, as well as the posture variables $x$, $y$ and $\phi$ are shown in Figure 4-b). When the goalkeeper moves on the straight line in front of the goal, the two requirements $y = d$ and $\phi = 0$ are satisfied. The requirements are expressed in the following error vector:

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} d - y \\ \phi \end{bmatrix} \tag{5}$$

A Lyapunov candidate function is introduced:

$$V(e) = \frac{1}{2} \left( h\, e_1^2 \; + \; e_2^2 \right) \quad = \frac{1}{2} \left( h \left( d - y \right)^2 \; + \; \phi^2 \right) \quad , h > 0 \tag{6}$$

To establish if asymptotic stability is feasible the derivative of $V(e)$ is obtained:

$$\dot{V}(e) = h \left( d - y \right) \; u \; \sin(\phi) \; + \; \phi\, \omega \quad , h > 0. \tag{7}$$

Consider the control law:

$$\omega = -\gamma \; \phi \; - \; h \left( d - y \right) \; u \; \frac{\sin(\phi)}{\phi} \quad , \; h > 0 \; , \gamma > 0 \tag{8}$$

Applying the control law yields $\dot{V}(e) = -\gamma \; \phi^2 \quad , \gamma > 0$, which is a negative definite function. Since the control law is derived following the same line of thought as for the control algorithm of `FollowBall`, a proof for asymptotic stability will not be given here.

A control law for the linear velocity $u$ was designed independent of the control law for $\omega$. Based on the good results from `FollowBall`, it was chosen to use a P-controller for the linear velocity. The control law is given as $u = K_p * (\psi - \theta)$, where $\psi$ is either the angle to the ball or the angle to the predicted interception point $x_{intercept}$, and $\theta$ is, as in `FollowBall`, the angle between the center of the goal and the goalkeeper. Depending on the direction and velocity of the ball, an interception point between the trajectory of the ball and the defending line may be used to determine the angle $\psi$.

## 3 Vision-Based Self-localization Using Local Features

The motion control algorithms of the previous section rely on good estimates of the robot posture. However, and especially for a robotic goalkeeper, frequently subject to bumps from other robots, odometry alone does not provide such a reliable estimate, as it strongly degrades over time. Therefore, one must reset it regularly with a more accurate estimate. The solution used in this work consisted of using goalkeeper local visual features (e.g., the goal and the posts) and a vision-based algorithm to provide such an estimate.

We have used ISocRob's omni-directional catadioptric vision system for this purpose. An image acquired by this system is shown in Figure 5. Since this is a norm-preserving omni-directional catadioptric system (i.e., distances on the field are preserved on the image) it is relatively simple to determine the distance from the goalkeeper to surrounding objects by processing the images. Furthermore the goal posts will always point towards the position of the robot in the (center of the) image.

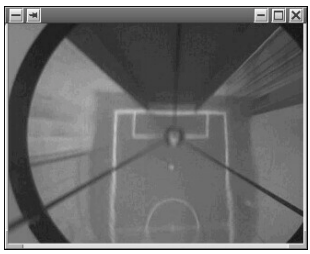


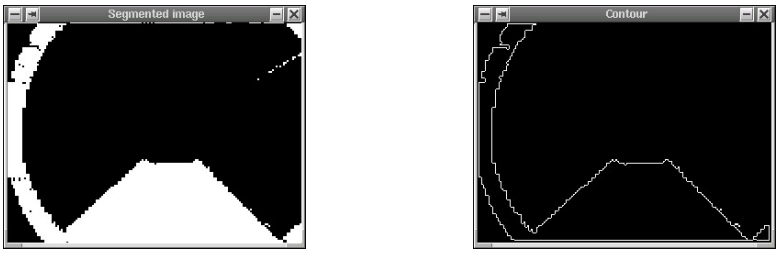
**Fig. 5.** Omni-directional catadioptric vision image taken by the goalkeeper

Under normal operation the goalkeeper should be near its own goal at all times. So a good object to detect from the image will be the goal. Extracting the goal posts as a feature from the image, will reveal the position of the robot with respect to them. Then the posture of the robot can be estimated by simple geometry. The algorithm runs in situations while the robot is standing still.

The image processing algorithm is split in four stages, briefly detailed in the sequel.

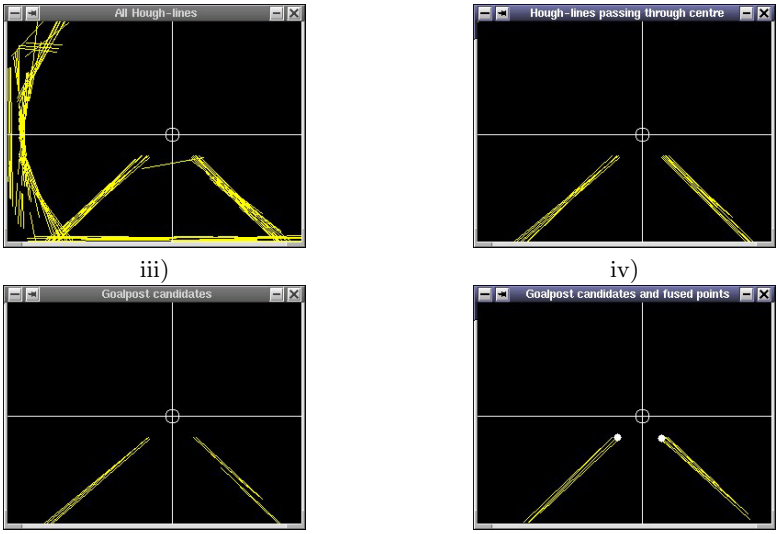
**Stage 1: Image Processing** — to retrieve the contour of the goal from the image two steps are required (see Figure 6): i) segmentation of the goal in the image (identifying the pixels with the same color as the defending goal and disregarding all other pixels); ii) edge-detection revealing the contour of the goal in the image.

**Stage 2: Feature Detection Using the Hough Transform** — to detect the goal posts in the image containing the contour of the goal the following three steps are performed (see Figure 7): iii) Hough Transform of the image; iv) line filtering: all straight lines obtained in the previous step which do not point to the center of the image will be filtered out; v) goal post detection:




i) ii)

**Fig. 6.** Steps of image processing stage 1: i) Goal segmentation; ii) edge-detection




iii) iv)

v) vi)

**Fig. 7.** Steps of image processing stage 2: iii) Hough Transform; iv) line filtering; v) goal post detection; vi) goal posts position estimates

this is accomplished by comparing the angles of all the lines in a histogram, and detecting the two angles most likely to be the angles of the goal posts. Afterwards two arrays of one or more lines are created containing the line(s) of each goal post.

**Stage 3: Estimation of Goal Post Positions** — If, as a result from the previous steps, there were only one line describing the goal post, the position of the goal post would be considered as the line endpoint closest to the robot. In the case there are more lines, an estimate of the line endpoint is obtained by first sorting the line endpoints, and then using a Bayesian data fusion algorithm whose details we will omit here.

**Stage 4: Geometric Calculations** — knowing the positions of the two goal posts with respect to the robot, and the positions of the goal posts in the field, it is possible to determine the posture of the robot using simple geometry. Again, due to limited space, the details will be omitted.

The image processing algorithm was implemented in C, using the OpenCV library developed by Intel.

## 4 Experimental Setup and Simulations

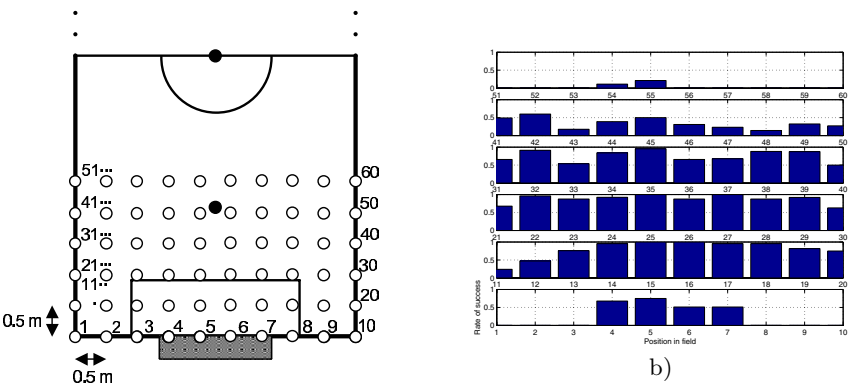
Simulations were designed to determine the "optimal" values for the danger zone distance from the goal line $d_{dz}$ and the radius $r$ of the defensive arc for the `FollowBall` behavior. The purpose is to determine the configuration that will lead to the highest number of saves by the goalkeeper when the state machine includes the `FollowBall` and `InterceptBall` behaviors only. Since both the discrete state (`FollowBall` or `InterceptBall`) and the continuous state of the robot as well as of the ball (i.e., their position and velocity) will matter for this purpose, the goalkeeper was modeled as a hybrid automaton [10], resulting from the composition of the goalkeeper and ball hybrid automata. The hybrid automaton was implemented in Simulink using CMU's CheckMate toolbox. Three tests were performed. In the first test, several values of $d_{dz}$ used for a danger zone as wide as the field. In the second test, the best values for $d_{dz}$ were taken and the danger zone width varied. Finally, for the best pair determined in the previous tests, several values for $r$ were tested. In each of the three tests, shots were fired from different locations in the field, each shot aiming towards the goal. For every different setup for the danger zone or the radius of the defensive arc, 625 shots were fired against the goal. The speed and the direction of the shots differed. In five groups of 125 shots each, the speed was set to 2m/s, 2.5m/s, 3m/s, 3.5m/s, 4m/s respectively. The goalkeeper was initially at the center of the goal, on the defensive arc.

From these tests, the best results (64.6% of goals saved) were obtained for a danger zone with the field width, $d_{dz} = 4.5$ m and $r = 0.6$ m. A constant radius was also used for the defensive arc, instead of `r_min` and `r_max` referred in Section 1. The failure rate of 35.4% is mainly due to the relatively slow top speed achieveable by the goalkeeper ($1m/s$) and the high speed of some of the simulated shots.

## 5 Experimental Results

### 5.1 Self-localization

To test the performance of the self-localization algorithm, two issues were considered: the success rate of the algorithm and the error of the estimated postures. The robot was placed with the same orientation (90° in the field frame, where $x$ points towards the opponent goal $z$ points upwards and $y$ is such that a right-handed frame is obtained) in 60 different positions in front of the goal,




a) b)

**Fig. 8.** Self-localization results: a) goal-keeper field positions; b) estimation error at different field positions

as illustrated in Figure 8-a). In each position the algorithm was run until 20 estimates of the posture were determined (in some it was not possible to get 20 estimates though).

As explained earlier the algorithm does not return any posture if one of several conditions is not met (e.g. if only one goal post is found in the image). With this in mind, the success rate for the algorithm is defined as the ratio between algorithm runs which are successful (a posture is found) and the total algorithm runs. The success rate was evaluated in each of the 60 positions in the field. The results are plotted in Figure 8-b. The success rate when the robot is on the goal line outside the goal posts is zero, since the goal color is not visible from here, and it is needed for posture disambiguation. It is also noticeable that the success rate is very low in positions 2 m and further away from the goal. The algorithm has the best success rate in positions 1 and 1.5 meters away from the field end line, and in general there are more successes in positions in front of the goal.

Concerning the posture estimation errors, in general errors are larger (around 20 cm) in the $x$ direction than in the $y$ direction (around 10 cm). Orientation errors are typically below 10°. Unfortunately, large outliers can sporadically occur, especially on the $x$ coordinate. Therefore, a test is made which accepts an odometry reset only if the new value does not differ from the current odometry estimate for more than a given threshold. The algorithm average run time is of approximately 0.4 s.

### 5.2 Motion Control

The results regarding the motion control cover the the goalkeeper performance while tracking the defensive arc, shown for two different speeds in Figure 9-a), under the `FollowBall` behavior, and when attempting to intercept a ball heading

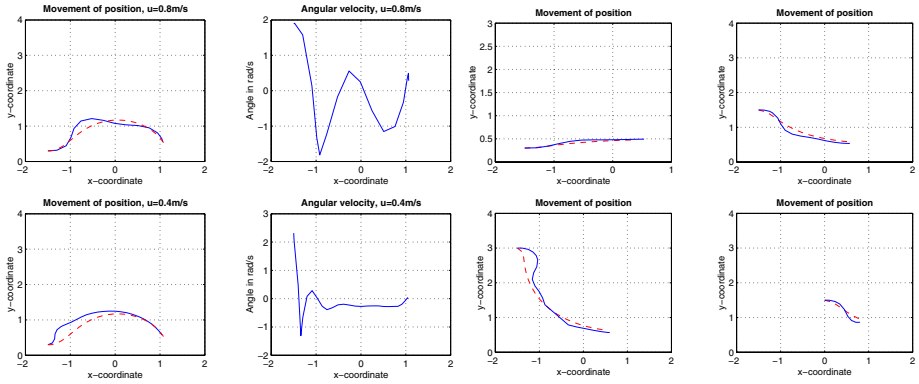



**Fig. 9.** a) Results of simulated and actual robot motion for the `FollowBall` behavior. The goalkeeper moves from its starting position to the arc and stops as it reaches the line between the ball and the center of the goal. The solid line shows the movement in goal coordinates logged from odometry and the dashed line is the simulated motion. The radius for the defending arc is 1.2 m. The parameters used for the control algorithm were $d = 1.2$, $h = 4$, $\gamma = 2$, $K_p = 1.5$ and $u_{max} = 0.8\frac{m}{s}$. b) Results of simulated and actual robot motion for the `InterceptBall` behavior under four situations where the goalkeeper has different start positions. The circle in (2, 1) represents the ball. The line from the ball to the origin illustrates a shot towards the goal. The defending line is placed at $d_{dz} = 0.5$. Starting position for the goalkeeper in the figures from the left to right is (-1.5, 0.3), (-1.5, 1.5), (-1.5, 3) and (0, 1.5).The dashed line is the simulated trajectory and solid is the logged trajectory from odometry

towards the goal, starting from four initial positions, under the `InterceptBall` behavior, as shown in Figure 9-b).

In general, the tests made show a good performance when following the ball, even though some differences from the simulated results were found, essentially due to unmodeled dynamics in the simulations. Among those, the critical problem is the almost unstable convergence to the arc at higher speeds. Regarding ball interception, a problem arises due to the usage of a constant value for $d_{dz}$, instead of a variable one, e.g., equal to the current $x$ coordinate of the goalkeeper. Due to speed limitation, for high ball speeds the robot tends to "open" the goal while moving towards the intercept line, since it first rotates of 90°.

## 6 Conclusions and Future Work

This paper presented an integrated design, implementation and test of a robotic goalkeeper which included hybrid automata modeling of the behavior coordinator, non-linear motion control for trajectory tracking and posture stabilization and vision-based self-localization. The results are very promising, but further work needs to be done, namely to improve the reaction speed, to use a variable intercept line and include further behaviors, such as leaving the goal to face

an opponent robot carrying a ball. In the latter case, inspiration on work concerning behavior coordination with smooth and conflict-free transitions between behaviors [11] will be considered.

## References

1. Indiveri, G.: *An Introduction to Wheeled Mobile Robot Kinematics and Dynamics.* Slides for lecture given at RoboCamp, Paderborn (Germany) (2002)
2. Jamzad, M., Chitsaz, H., et al: A goalkeeper for middle size robocup. RobCup-2000: Robot Soccer World Cup IV (2001)
3. Menegatti, E., Nori, F., Pagello, E., Pellizzari, C., Spagnoli, D.: Designing an omnidirectional vision system for a goalkeeper robot. RobCup-2001: Robot Soccer World Cup V (2001)
4. Lima, P., Custódio, L., Ventura, R., Aparício, P.: A functional architecture for a team of fully autonomous cooperative robots. RobCup-99: Robot Soccer World Cup III (1999)
5. Marques, C., Lima, P.: A multi-sensor navigation system for soccer robots. RobCup-2001: Robot Soccer World Cup V (2002)
6. de Wit, C.C., Siciliano, B., (Eds)., G.B.: Theory of Robot Control. Springer-Verlag, London, UK (1996)
7. Oriolo, G., Luca, A.D., Vendittelli, M.: *WMR Control via Dynamic Feedback Linearization: Design, Implementation and Experimental Validation.* IEEE Transactions on Control Systems Technology (2002)
8. Khalil, H.K.: Nonlinear Systems. Prentice-Hall (2002)
9. Indiveri, G.: *On the Motion Control of a Nonholonomic Soccer Playing Robot.* RobCup-2001: Robot Soccer World Cup V (2002)
10. van der Schaft; Hans Schumacher, A.: An Introduction to Hybrid Dynamical Systems. Springer (1999)
11. Uchibe, E., Kato, T., Asada, M., Hosoda, K.: Dynamic task assignment in a multiagent/multitask environment based on module conflict resolution. In: IEEE Intern. Conf. on Roboticas and Automation. (2001) 3987–3992