REAL TIME LOCALIZATION IN FOUR LEGGED ROBOCUP SOCCER

Adrian Martinez-Gomez*
gomado@gmail.com

Alfredo Weitzenfeld*
alfredo@itam.mx

*Instituto Tecnológico Autónomo de México (ITAM) – Computer Engineering Department
Río Hondo #1, San Angel Tizapán, CP 01000
Mexico City, Mexico

ABSTRACT

In this paper we present the system architecture for our Four Legged RoboCup Soccer Team – Eagle Knights. We describe in detail the real-time localization module where a correction filter has been added in improving accuracy while keeping real-time performance. We show and analyze results.

KEYWORDS: localization, four-legged, robocup, autonomous, vision.

1 INTRODUCTION

RoboCup [Kitano 1995] is an international effort to promote AI, robotics and related field primarily in the context of soccer playing robots. In the Four Legged League, two teams of four robots play soccer on a relatively small-carpeted soccer field [RoboCup 2004].

We began this project in January 2004 and our first participation in an official competition was at the U.S. Open 2004 celebrated in New Orleans.

This paper presents an overview of the system architecture used by our team, Eagle Knights, describing in detail our Localization module. We use a very simple and efficient localization module with enough accuracy for the task at hand. The basic algorithm is refined by the use of corrective filters in achieving the desired accuracy while performing in real time.

Our approach produces reliable results without using probabilistic methods, such as Kalman Filter [Middleton 2004] and Monte Carlo Algorithm (MCL) [Lastra 2004].

2 SYSTEM ARCHITECTURE

The Eagle Knights Four Legged system architecture is shown in figure 1.

Acknowledgements: This work has been supported by collaboration project NSF-CONACYT 42440, UC MEXUS-CONACYT, LAFMI-CONACYT and "Asociación Mexicana de Cultura, A.C.".

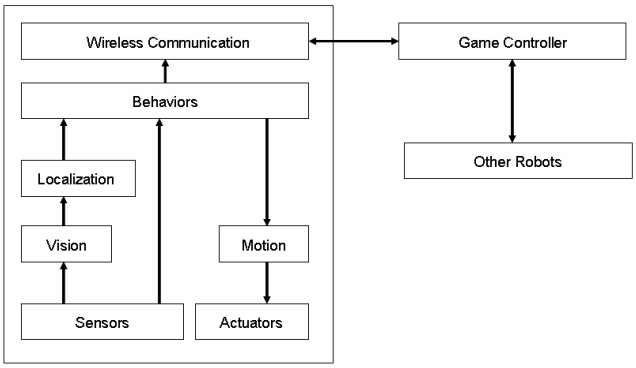

Fig 1. The system architecture includes the Sensors, Actuators, Motion, Localization, Behaviors and Wireless Communication modules.

The architecture includes the following modules:

- **Sensors.** This module receives information from the sensors. We are particularly interested in vision and motor position feedback.
- **Actuators.** In addition to individual head and leg motor control, the system includes actuators for turning off and on head LEDs.
- **Motion.** This module control robot movements, such as walk, run, throw the ball, turn to the right or left, move the head, etc. It receives commands from the behavior module with output sent to the corresponding actuators, representing individual leg and head motor control.
- **Vision.** The vision module receives a raw image from the camera, the main system sensor, and performs segmentation over the image. It then recognizes objects in the field, such as goals, ball, landmarks and other players.
- **Localization.** This module performs the processing necessary to obtain a reliable localization of the robot in the field. In order to obtain a good localization it is desirable for the robot to perceive at least two marks, either goals or landmarks. Localization is described in more detail in Section 3.
- **Behaviors.** This module makes all decisions affecting higher-level robot actions. It takes input from the

sensors and localization systems and sends commands to the motion and actuators module.

- **Wireless Communication.** This module receives commands from the external Game Controller.

2.1 Sensors

Sensory information from the color camera and motor position feedback is the main input to the system. The raw camera image is passed directly to the vision module while information received from motor positions are used in making certain movement decisions at the behavior module, such as when the robot is on its back.

2.2 Actuators

The system includes a number of actuators, in particular legs and head motors. In addition there are actuators responsible for turning off and on head and tail LEDs. Motor commands are received from the motion module while LED control is received from the behaviors module to indicate the action that the robot is performing. Additionally, this module shows the state of the Game Controller in the head LEDs of the AIBO.

2.3 Motion

The motion module is responsible for robot motion control, including, walking and kicking among other set of movements, both legs and head movements. We have developed a number of different routines depending on team roles. For example, the goalie has different motions in contrast to other team players. This also applies to different head kicks and movements in general.

2.4 Vision

The vision system is used image segmentation from the AIBO built-in Color Detection Table (CDT). While this segmentation approach produces relatively good results, we have started to use instead raw image input from the camera, doing complete software segmentation ourselves. We perform calibration in conjunction with either the CDT or our new software segmentation module. Calibration uses external segmentation configuration files.

Our AIBO vision system builds from our existing Small Size team vision system [Martínez-Gómez, 2004], which is relatively efficient and reliable, doing real time color and object recognition in the field.

Different color thresholds are selected during calibration to accommodate varying light conditions. We take initial photos of objects of interest and then manually select colors that we want to distinguish upon. At the end of this task, we test the segmentation on real images to test if our calibration algorithm is working properly.

When segmentation is complete, we process same color region formations. Then, a Run Length Encode (RLE) algorithm is applied compressing the image without information lost allowing generation of contiguous color regions.

After color regions are obtained, the system is ready to recognize and identify objects. Objects in the field must fulfill certain requirements in order to allow a certain degree of confidence that the region we are comparing is really the object that we are interested in. For example, the ball must have green in some adjacent area. A similar criterion is used to identify goals. The identification of landmarks is a little more complicated, requiring a number of color regions to be adjacent to each other.

After objects have been identified, a data structure is generated containing object positions as well as additional object characteristics.

2.5 Behaviors

The behavior module receives information from the sensors and the Localization System and sends its output to motion and actuator modules executing the instructions to control the AIBO. Our team has two types of player roles at this point: Attacker and Goalie. Each one has a different behavior that depends on the ball's position and the referee box (Game Controller).

Goalie behavior is described by a state machine as shown in Figure 2:

- **Initial Position**. This is the initial posture that the robot takes when it's turned on.
- **Search Ball**. The robot searches for the ball.
- **Reach Ball**. The robot walks towards the ball
- **Kick ball**. The robot kicks the ball out its goal area.
- **Search Goal** . The robot searches for the goal.
- **Reach goal** . The robot walks toward its goal.

Attacker behavior is described by a state machine as shown in Figure 3:

- **Initial Position**. This is the initial posture that the robot takes when it's turned on.
- **Search Ball**. The robot searches for the ball.
- **Reach Ball**. The robot walks towards the ball
- **Kick Ball**. The robot kicks the ball towards the goal.
- **Explore Field**. The robot walks around the field to find the ball.

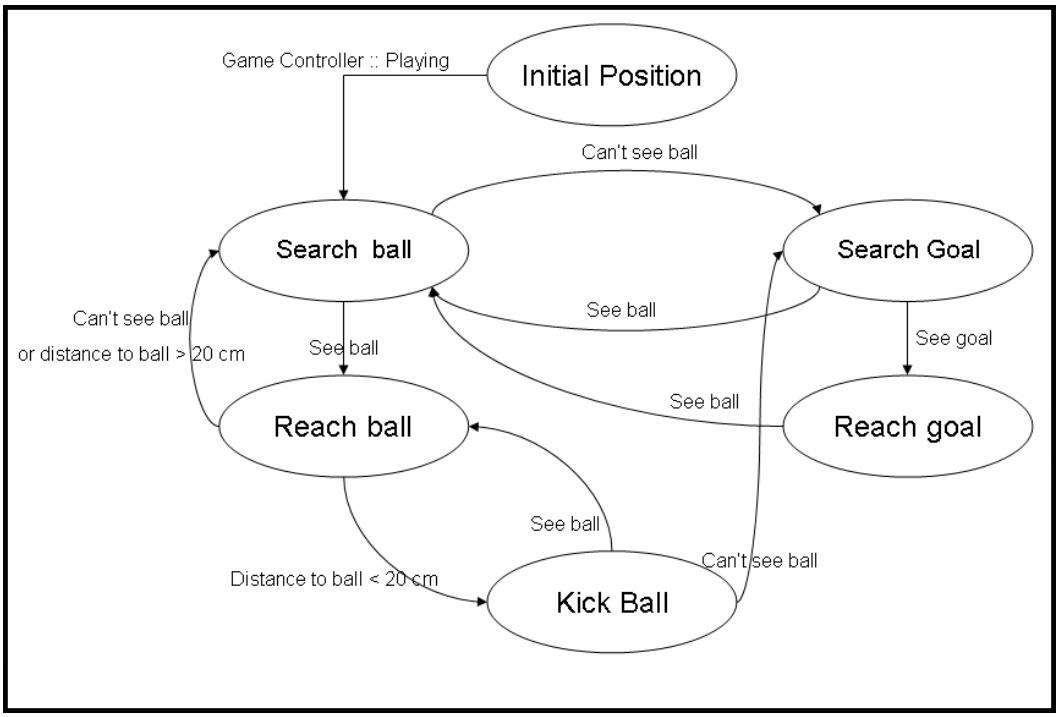

Fig 2. Goalie State Machine.

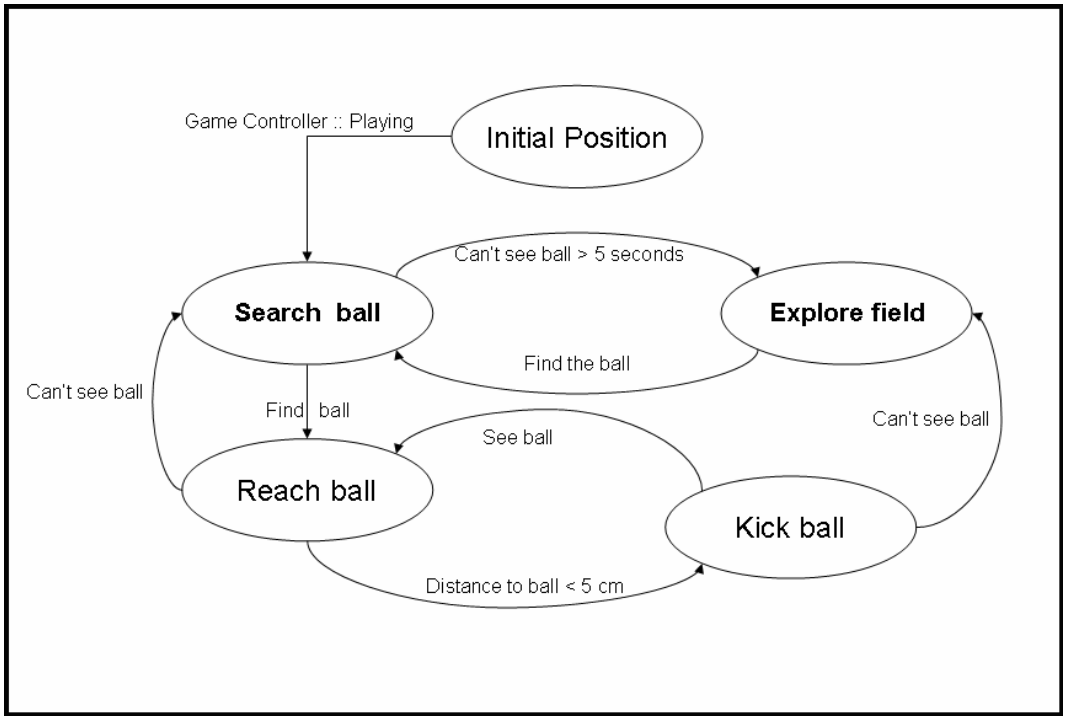

Fig 3. Attacker State Machine.

2.6 Wireless Communication

This module receives commands from the Game Controller and passes them to the Behaviors module. It can use either TCP or UDP protocol.

3 LOCALIZATION SYSTEM

A critical part of playing soccer is being able to localize in the field in an efficient and reliable way. An important challenge is to achieve real time processing when calculating actual distances to objects in the field seen by the robot.

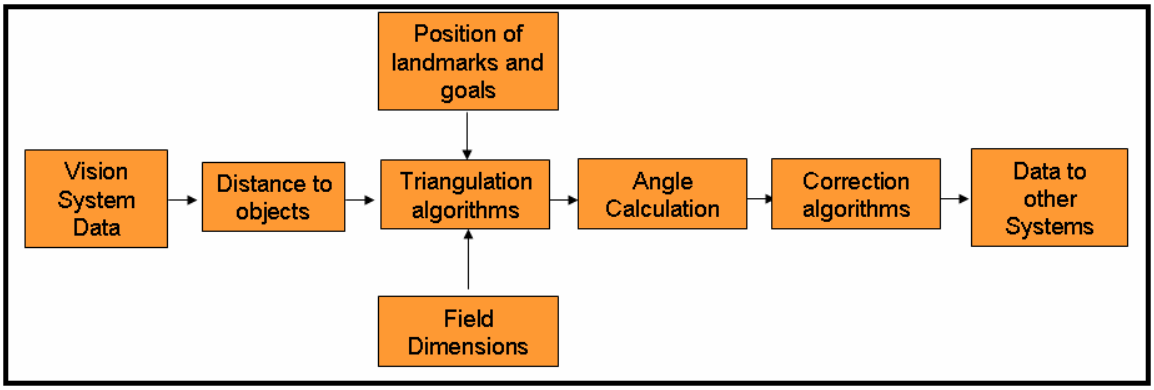

Fig 4. Localization System block diagram.

Our localization system is decomposed into a number of blocks, as shown in Figure 4. Input to the localization system comes from the Vision System in the form of objects identified with their corresponding sizes and centroid positions. Localization requires a triangulation algorithm taking account the position of the landmarks and goals, and field dimensions. With these parameters, the algorithm produces a coordinate estimate of the position of the robot. Then a robot orientation is computed and finally, a correction algorithm improves the estimation of the position and orientation of the robot.

3.1 Field Conditions

The Four Legged League field has dimensions of 6 x 4 meters. It has four landmarks and two goals. Each landmark has a different color combination that makes it unique. The landmarks are divided in tree zones, as shown in Figure 5. Zone C is always white, while zones A and B are a combination of pink and either cyan or yellow. The position of the landmarks in the field is shown in the figure 6.

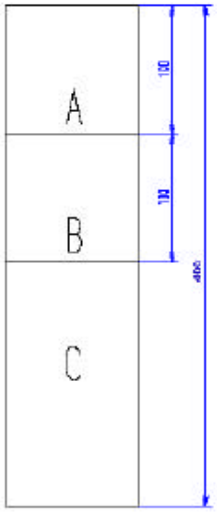

Fig 5. Field landmark color subdivisions.

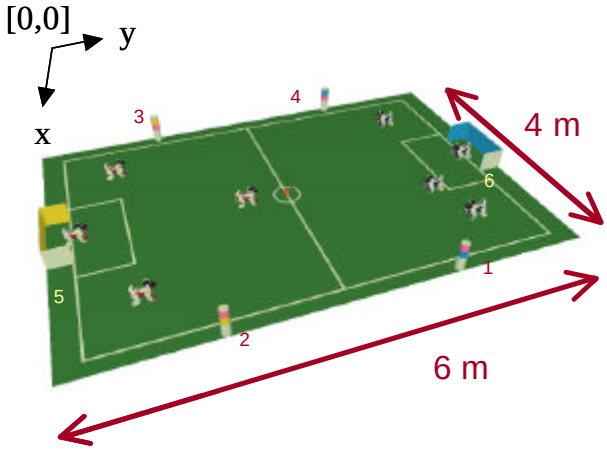

Fig 6. The Four Legged League field.

Landmark and goals positions are shown in Table 1 where [0,0] corresponds to upper left corner in the field (white line corner) with positive x and y pointing towards inside the field.

Table 1 - Landmark and goal positions in the field.

Object	Id Object	X	Y
Goal 1	5	180	-35
Goal 2	6	180	575
Landmark 1	1	370	405
Landmark 2	2	370	135
Landmark 3	3	0	135
Landmark 4	4	0	405

3.2 Distance to objects

The first step in localizing is to obtain the distance between identified objects and the AIBO. It is important that the robot can distinguish at least two marks when it starts. After making several experiments, we have noticed that the best way to obtain the precise distance towards the objects is by using a mathematical relation that takes as parameter the object area and returns as result the distance to the object. In order to obtain this relation, we took a large number of measurements at different distances from the object that we are interested in. In order to make the system more reliable, we took measures each five centimeters. The distance range we use goes from 15 centimeters to 4 meters. Beyond four meters it is very difficult to notice a difference between object and noise.

After making these measurements we noticed that the relation of the object area with the distance behaved like an inverse exponential function. This is independent of object measured, requiring only an offset for the particular function. We made measurements of this relation with distinct head position of the robot. The results of these measurements are shown in figure 7.

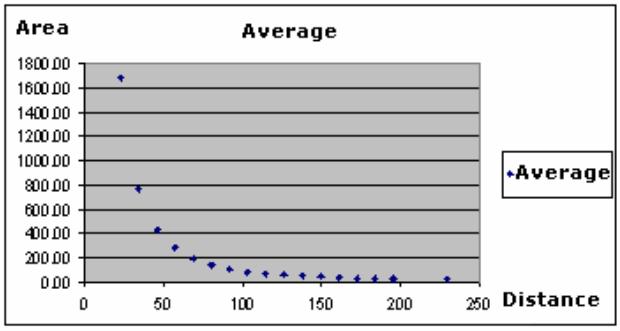

Fig 7. Relation area/distance for landmarks.

Figure 8 shows the area versus distance function together with the resulting standard deviation for this function.

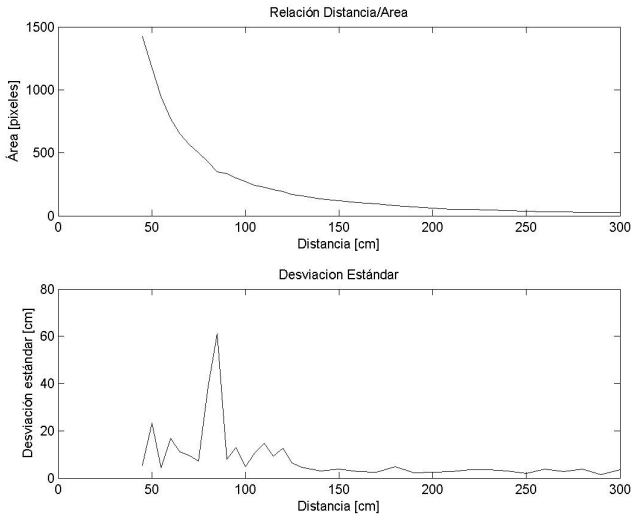

Fig 8. Upper diagram shows relation between area (Y axis) and distance (X axis), while lower diagram shows the resulting standard deviation.

We chose an interpolation function to match our data. Using Matlab we calculated the coefficients for the cubical segments of the interpolation (splines). We chose that type of interpolation because it is very precise and the differences between different points from the samples were very smooth and almost imperceptible. Also, with this method, we just have to calculate the coefficients offline and load them in memory when the robot starts playing. When we want to calculate a distance to an object, we just have to evaluate a polynomial expression with the appropriate coefficients and according to the following equation:

$$s(x) = ax^3 + bx^2 + cx + d$$

We calculated the cubical function coefficients and tested them using different distances. After making some adjustments we obtained an average error of less than 5 centimeters. Another advantage of using this method is that with minor changes, we can use the coefficients from any position in the field, because the relation between area and distance is the same anywhere. Results from these computations are shown in Table 2.

Table 2 - Results of the interpolation function.

Real Distance [cm]	Computed Distance (average) [cm]	Error [cm]
35	40	5
50	52.07	2.07
65	67.47	2.47
80	82.76	2.76
95	96.98	1.98
110	112.70	2.70
125	124.54	0.45
140	140.39	0.39
155	151.46	3.53
170	170.66	0.66
200	204.63	4.63
230	225.69	4.31
260	250.73	9.26
290	277.49	12.51

3.3 Triangulation Algorithm

Following distance computation we apply a triangulation method from two marks to obtain the position of the robot on the field. Triangulation allows obtaining a very precise position of the robot in a two dimensions plane. If a robot sees one landmark and can calculate the distance to this landmark, the robot could be anywhere in a circumference with origin in the landmark, and radio equals to the distance calculated, as shown in Figure 9.

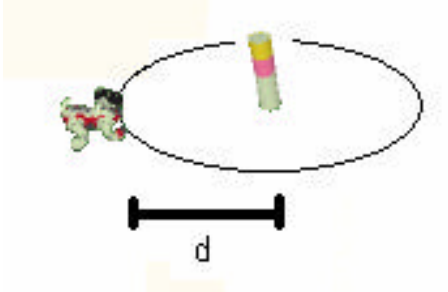

Fig 9. Triangulation algorithm with one landmark.

This kind of calculation is not sufficient for the robot to compute its actual position. So, if the robot can recognize two landmarks, by applying this algorithm, its actual position is obtained from the intersection of the two circumferences, as shown in Figure 10. Note that the robot could be in one of two intersection points in the circumferences.

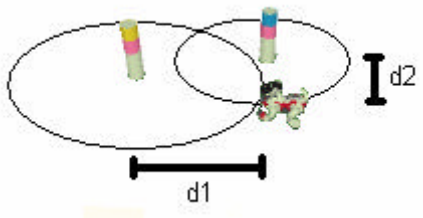

Fig 10. Triangulation with two landmarks.

Thus, we use a third location reference to define a unique point. This third reference is the field condition where one of the two intersection points is always outside the field, as shown in Figure 11.

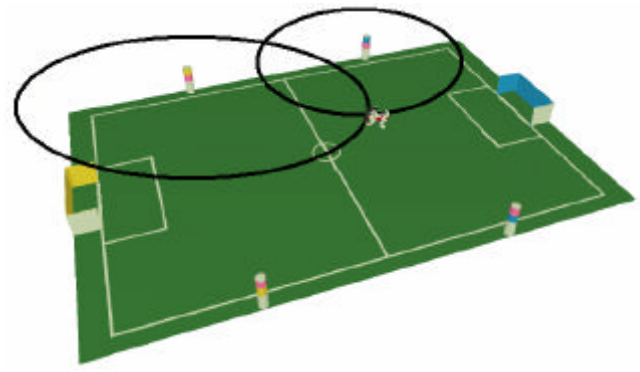

Fig 11. Triangulation with field conditions.

In Table 3 we show the results of the triangulation algorithm. To test the algorithm we put the robot somewhere in the field. Then we computed the average distance obtained from multiple measurements followed by an average error calculation. Note the large difference between the true position and the computed average.

Table 3 - Results of triangulation algorithm.

Real Position [cm]	Average [cm]	Error [cm]
(50,80)	(80.06,139.28)	(30.06,59.28)
(160,90)	(163.86, 104.44)	(3.86, 14.44)
(265,80)	(278.08,108.39)	(13.08,28.39)
(65,185)	(95.04, 195.87)	(30.04, 10.87)
(170, 190)	(175.25, 196.10)	(5.25, 6.10)
(280, 210)	(301.64, 222.44)	(21.64, 12.44)
(70,290)	(80.93, 303.52)	(10.93, 13.52)
(186,300)	(195.75, 313.52)	(9.75, 13.52)
(285,300)	(301.17, 315.51)	(16.17, 15.51)
(65,380)	(68.53, 414.64)	(3.53, 34.64)
(165,400)	(180.04, 421.87)	(15.04, 21.87)
(290,350)	(312.87, 362.08)	(22.87, 12.08)

3.4 Angle Calculation

Once we find the robot position we need to find its orientation to complete localization. We refer to two vectors whose origin is the robot location and the end points are the coordinates of the marks that we use as references for the triangulation, as shown in Figure 12.

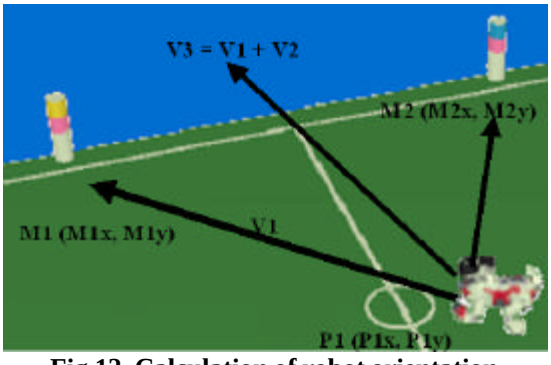

Fig 12. Calculation of robot orientation.

Table 4 shows computed orientation angles in relation to single and multiple object references.

Table 4 - Robot orientation.

Reference 1	Reference 2	Angle
Landmark 1		30°
Landmark 2		330°
Landmark 3		210°
Landmark 4		150°
Goal (5)		270°
Goal (6)		90°
Landmark 1	Landmark 2	0°
Landmark 1	Goal (6)	60°
Landmark 2	Goal (5)	300°
Landmark 3	Landmark 4	180°
Landmark 3	Goal (5)	240°
Landmark 4	Goal (6)	120°

The relative robot orientation in relation to the field is shown in Figure 13.

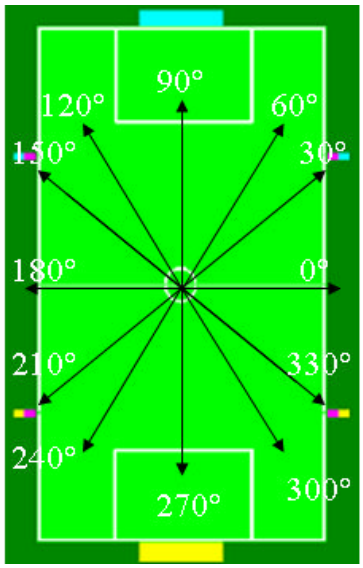

Fig 13. Angle reference.

3.5 Correction Algorithms

While testing our algorithm in real time with the robot walking, we noticed that in many occasions our data was not consistent between two contiguous frames. Thus, we used a correction computation algorithm taking historical data from positions already calculated by the robot in obtaining the average of these measurements. With this correction filter, we were able to reduce the maximum error variance from close to 20% to 30% [Martínez-Gómez 2005].

When computing the triangulation, we reduce the variation of the output signal for the triangulation algorithm by using the following average filter function:

$$y[i] = (x[i-1] + x[i]) / 2$$

After applying this correction, we apply a historical average filter. This filter stabilizes the output signal and considerably decreases the error obtained in the triangulation algorithm. We considered the last 25 data

measurements in order to obtain the historical average output. The frame rate of the AIBO camera (ERS-210) is 25 fps. So, we have a delay of only one second for this computation. We use the following function to compute the historical average:

$$s(x) = \frac{\sum_{i=0}^{n} x(i)}{n}$$

Figure 14 shows sample output from this filter correction. Our original signal produced variations of approximate 10%. By applying this filter we managed to reduce this variation to less than 3%.

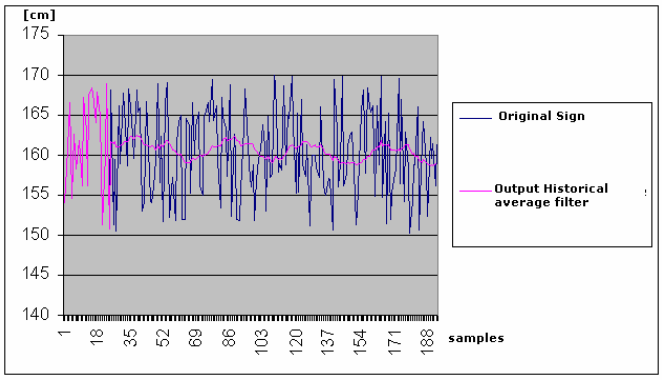

Fig 14. Historical Average Filter.

When we calculate the distance to the goals we found some additional errors. In particular, when the AIBO is near the goal, it stops seeing the complete goal. Thus, the relation area/distance does not provide good scaling. To minimize this error, we made some adjustments in order to estimate a reliable distance to the goal. These adjustments are shown in the Table 4.

Table 4 - Correction for the goal.

Estimate Distance [cm]	goal % seen	Correction factor	Coefficient	Result
65	105%	0	105%	68.25
75	85%	0	100%	75
76	72%	0	100%	76
89	55%	25%	80%	71.2
112	40%	25%	65%	72.8
132	30%	25%	55%	72.6
169	20%	25%	45%	76.05
229	12%	18%	30%	68.70
279	7%	18%	25%	69.75
334	4%	18%	22%	73.48

After applying this correction algorithm, we tested the system an obtained better results without affecting the performance of the system. These results are shown in Table 5.

Table 5 - Final results for the localization system.

Real Position [cm]	Average [cm]	Error [cm]
(50,80)	(69.6, 124.8)	(19.6,44.8)
(160,90)	(160.27, 92.95)	(0.27,2.95)
(265,80)	(270.83, 100.35)	(5.83, 20.35)
(65,185)	(81.76, 188.81)	(16.76, 3.81)
(170, 190)	(170.47, 193.41)	(0.47, 3.41)
(280, 210)	(294.86, 216.79)	(14.86, 6.79)
(70,290)	(75.09, 293.53)	(5.09, 3.53)
(186,300)	(189.9, 303.53)	(3.90, 3.53)
(285,300)	(289.36, 303.19)	(4.36, 3.19)
(65,380)	(65.63, 413.74)	(0.63, 33.74)
(165,400)	(170, 415.22)	(5.33, 15.22)
(290,350)	(305.8, 355.25)	(15.80, 5.25)

4 CONCLUSIONS

We just presented the system architecture for the Eagle Knights Four Legged team with special emphasis on our real time localization system.

The system calculates the distance with the help of an interpolation function producing very good results and in real time. The algorithms used allowed us to estimated a reliable position for the robot without using probabilistic methods like other teams do.

We are currently incorporating localization information as part of our game playing strategy while adapting the algorithm to specific regions in the field to produce better qualitative game playing results as opposed to costly numerical accuracy.

More information can be found in http://robotica.itam.mx/.

REFERENCES

[Kitano 1995] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In Proceedings of the IJCAI-95 Workshop on Entertainment and AI/ALife, 1995.

[Lastra 2004] Raúl Lastra, Paul Vallejos, and Javier Ruiz-del-Solar. Integrated Self-Localization and Ball Tracking in the Four-Legged Robot Soccer League. Proc. 1st IEEE Latin American Robotics Symposium - LARS 2004, México City, México, Oct. 28-29, 2004, pp. 54-59.

[Martínez-Gómez, 2004] Martínez-Gómez, L.A., and Weitzenfeld, A., 2004, Real Time Vision System for a Small Size League Team, Proc. 1st IEEE-RAS Latin American Robotics Symposium, ITAM, Mexico City, October 28-29.

[Martínez-Gómez 2005] Martínez-Gómez, J.A. Design of the Localization System for autonomous robots. Undergraduate Thesis. School of Engineering, ITAM, 2005.

[Middleton 2004] Richard H. Middleton, Michaela Freeston, and Leonie McNeill. An application of the extended Kalman filter to robot soccer localisation and world modelling, 3rd IFAC Symposium on Mechatronic Systems, 2004.

[RoboCup 2004] RoboCup Technical Committee. Sony Four Legged Robot Football League Rule Book. May 2004.