# Eagle Knights 2008: Standard Platform League

Alonso Martínez[1], Carlos Ramos[1], Carlos Rivera[1], Gibran Ríos[1], Emir Herrera[1] and Alfredo Weitzenfeld[1]

[1] Robotics Laboratory Lab, ITAM, Rio Hondo 1,
01000 DF, Mexico
alfredo@itam.mx

**Abstract.** In this paper we present the system architecture for our latest Standard Platform RoboCup Soccer Team – Eagle Knights 2008. This TDP corresponds to Eagle Knights 2008 participation in the NAO league, yet description are still based on previous AIBO robot software development. We describe the system architecture: Sensors, Vision, Motion, Actuators, Kinematics, Behaviors, Wireless Communication, and Localization. We provide a more extensive description of part of our localization algorithm.

**Keywords:** standard-platform league, robocup, localization.

## 1 Introduction

RoboCup is an international effort to promote AI, robotics and related field primarily in the context of soccer playing robots. In the Standard Platform League, two teams of four robots play soccer on a relatively small carpeted soccer field [1]. In Figure 1 we show an image of the Eagle Knight's NAO robot.

**Fig. 1**. Eagle Knight's NAO robot.

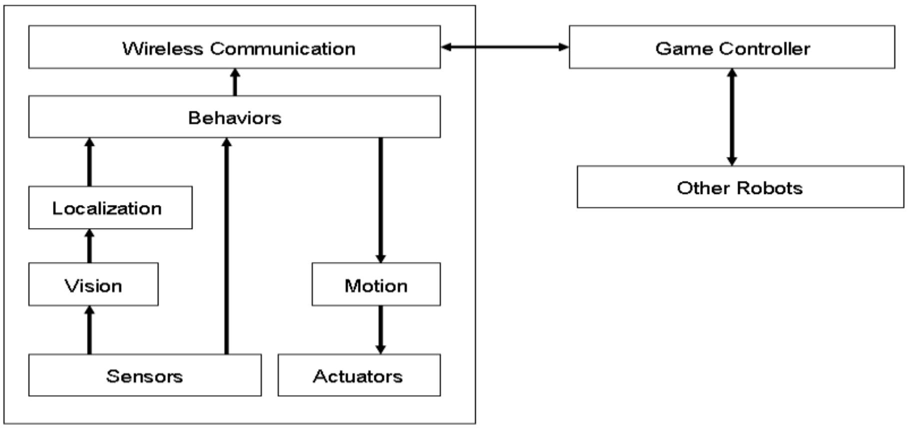


**Fig. 2**. Standard Platform System Architecture.

The standard platform league architecture, depicted in Figure 2, consists of the following modules:

- **Sensors**. This module receives information from from the color camera and motor position feedback. 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.
- **Vision**. The vision system recognizes objects in the field according to their color. Image regions are calibrated according to the 7 colors of interest assigned to objects in each image. The complete calibration and segmentation process is described in detail [2].
- **Motion**. This module is responsible for robot motion control, including in particular walking and kicking. 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.
- **Actuators**. This module is responsible for turning off and on the head and tail LEDs. This module receives commands from the behavior module to indicate the particular action being currently performed by the robot. This module also displays the state of the Game Controller.
- **Behaviors**. The behavior module receives information from sensors and localization, sending output to robot actuators. In defining our team robot behaviors, we specify three types of players: Attacker, Defender and Goalie. Each one has a different behavior that depends on ball position and Game Controller.
- **Wireless Communication**. This module receives commands from the Game Controller and passes them to the Behaviors module accepting connections using either TCP or UDP protocol. This module is used to cooperation between robots for have information about the state of the game and the world.

- **Localization**. This module makes all the processing necessary to obtain a reliable localization of the robot in the field. In order to localize, our current model requires the robot to perceive at least two marks, either goals or landmarks.

In addition to soccer playing, we have used our robots in other research projects, including biorobotics [3] and human-robot coaching [4].

## 2 Localization

An important aspect of soccer playing is being able to localize in the field in an efficient and reliable way. Localization involves computing distances to known objects, use of a triangulation algorithm to compute exact positioning, calculation of robot orientation angles, and correction of any computation errors.

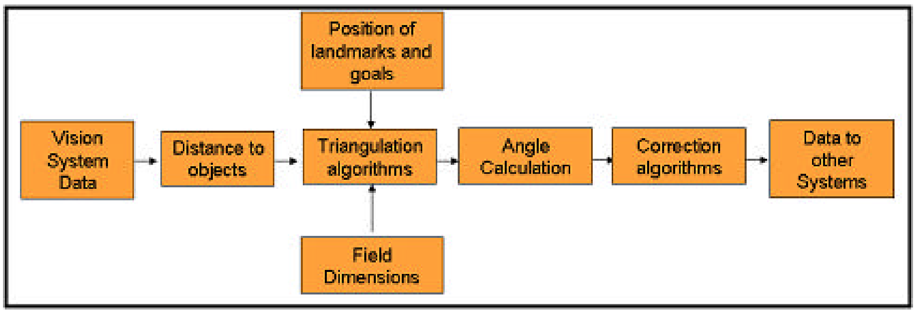


**Fig. 3.** Localization system block diagram.

A block diagram for our basic localization algorithm is described in Figure 3 consisting of the following modules:

- **Vision system data** where objects already recognized and identified are output.
- **Distance to objects** computing local distance from robots to objects.
- **Triangulation algorithms** computing precise localization obtained from external goals and landmarks. Note the 2008 rules do not include any external landmarks other than goals and field lines.
- **Position of landmarks and goals** consisting of external landmarks, currently only goals and field lines.
- **Field dimensions** corresponding to current field size.
- **Angle calculation** computes robot orientation in the field.
- **Correction algorithms** performs error correction on basic localization.
- **Data to other systems** sends resulting localization to the behavior module in the robot as well as other robots.

## 2.1 Distance to objects

Initially, each robot computes distances between identified objects and itself. We use a simple cubic mathematical relationship to calculate distances that takes as parameter the preceived object area and returns as a result the distance to the object. In order to obtain this relationship, we take a large number of measurements at different distances to the object of interested. The distance range used varies from 15 centimeters to 4 meters. Beyond four meters it becomes very difficult to distinguish between objects and noise.

Figure 4 shows the area versus distance function together with the resulting standard deviation for this function.

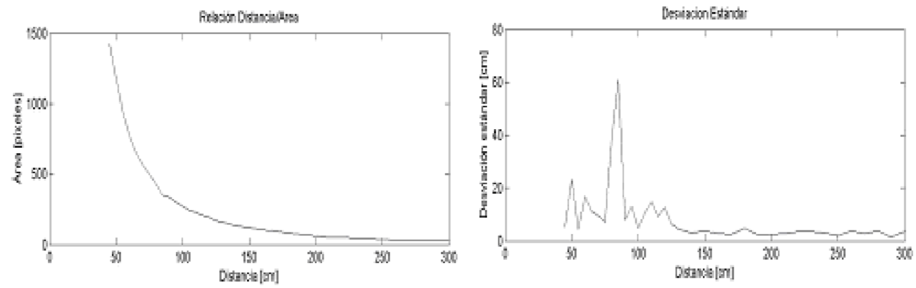


**Fig. 4.** Upper diagram shows relation between area (Y axis) and distance (X axis), while lower diagram shows the resulting standard deviation.

Using Matlab we initially calculate the coefficients for the cubical segments of the interpolation function (splines) to match data. We calculate the coefficients offline and then load them into 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 according to the following equation:

$$s(x) = ax^3 + bx^2 + cx + d \quad (1)$$

We calculated the cubical function coefficients and tested them using different distances. Results from these computations are shown in Table 1.

**Table 1**. 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 |

## 2.2 Triangulation Algorithm

Following distance computation we apply a triangulation method from two marks to obtain the position of the robot on the field. Triangulation results in 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 equal to the distance calculated. While a single is not sufficient for the robot to compute its actual position, recognizing two landmarks can already help compute a specific location from the intersection of two circumferences. In Table 2 we show the results of the triangulation algorithm.

**Table 2**. 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) |

To test the algorithm we put the robot in an arbitrary position 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.

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.

## 2.3 Correction Algorithms

While testing our algorithm with a moving robot, we noticed that in many occasions our data was not consistent between two contiguous frames. To fix the problem we added a correction algorithm taking historical data from positions already calculated by the robot in obtaining the average of these measurements. We reduced the variation of the output signal for the triangulation algorithm by using the following average filter function:

$$s(x) = \frac{\sum_{i=0}^{n} x(i)}{n} \qquad (2)$$

Figure 5 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%. See [5] for more details.

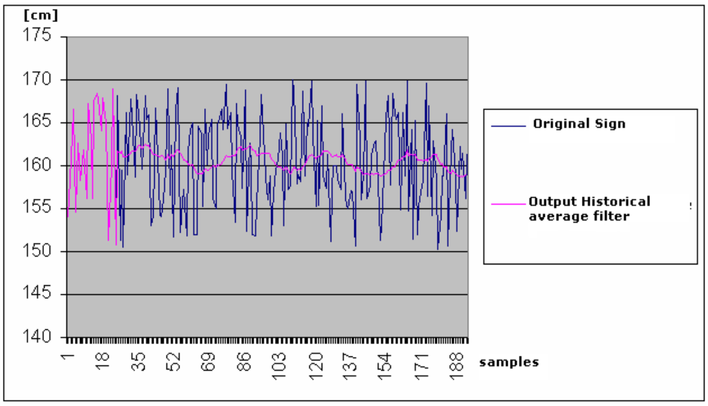


**Fig. 5.** Historical Average Filter.

After applying this correction algorithm, we tested the system and obtained good results without affecting the performance of the system. These results are shown in Table 3.

**Table 3.** Final results for the localization system.

| Region | Real Position [cm] | Average [cm] | Error [cm] |
|---|---|---|---|
| 1 | (50,80) | (69.6, 124.8) | (19.6,44.8) |
| 2 | (160,90) | (160.27, 92.95) | (0.27,2.95) |
| 3 | (265,80) | (270.83, 100.35) | (5.83, 20.35) |
| 4 | (65,185) | (81.76, 188.81) | (16.76, 3.81) |
| 5 | (170, 190) | (170.47, 193.41) | (0.47, 3.41) |
| 6 | (280, 210) | (294.86, 216.79) | (14.86, 6.79) |
| 7 | (70,290) | (75.09, 293.53) | (5.09, 3.53) |
| 8 | (186,300) | (189.9, 303.53) | (3.90, 3.53) |
| 9 | (285,300) | (289.36, 303.19) | (4.36, 3.19) |
| 10 | (65,380) | (65.63, 413.74) | (0.63, 33.74) |
| 11 | (165,400) | (170, 415.22) | (5.33, 15.22) |
| 12 | (290,350) | (305.8, 355.25) | (15.80, 5.25) |

Note that errors were computed by field regions, where the complete field was divided into twelve similarly sized areas, as shown in Figure 6. In some regions errors were larger due to changes in illumination. Yet, when compared to the 20cm approximate robot size, worst errors were a bit less than a full body length. Current work focuses in dividing the field into differently sized regions depending on required localization precision.

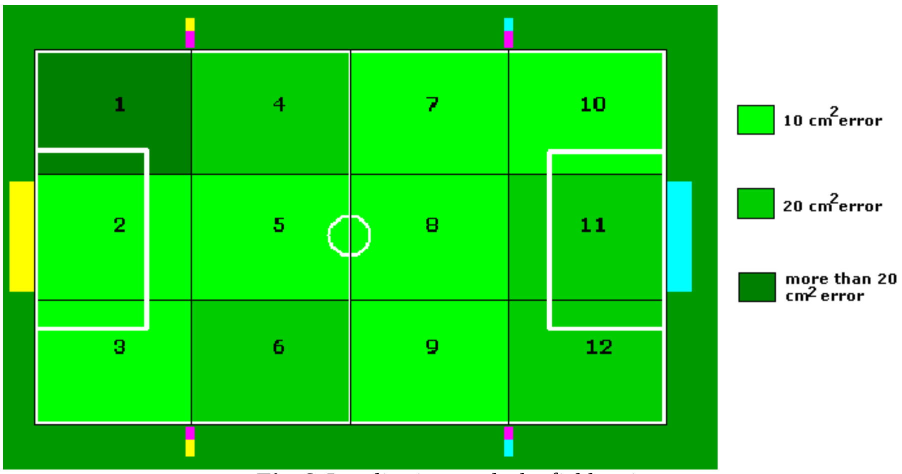


**Fig. 6.** Localization results by field region.

## 3 Conclusions

We have presented the system architecture for the Eagle Knights Standard Platform team with special emphasis on our real time localization system. The system calculates distances with the help of an interpolation function producing 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.

Our team started competing in 2004 and has since then continuously participated in regional or world events. This work is part of broader research we are pursuing in the robotics laboratory at ITAM. One of the related areas is that of human-robot interaction in the context of social cognition where we are using soccer coaching as the application domain. More information can be found in http://robotica.itam.mx/.

**Acknowledgments.** This work is supported by French-Mexican LAFMI, UC MEXUS CONACYT, CONACYT grant #42440, and "Asociación Mexicana de Cultura, A.C." in Mexico.

## References

1. RoboCup Technical Committee. "RoboCup Standard Platform League (Nao) Rule Book." Feb 2008, Standard Platform League Web Site, http://www.tzi.de/spl
2. 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 28-2.
3. Flores Ando, F., and Weitzenfeld, A., 2005, Visual Input Compensation using the Crowley-Arbib Saccade Model, Proc. International Conference on Advanced Robotics ICAR, Seattle, USA, July 17-20.
4. Weitzenfeld A, Ramos C, and Dominey PF, Coaching Robots to Play Soccer via Spoken-Language, RoboCup Symposium 2008, July 14-20, Suzhou, China (accepted for publication).
5. Martínez-Gómez, J.A, Weitzenfeld, A., 2005, Real Time Localization in Four Legged RoboCup Soccer, Proc. 2nd IEEE-RAS Latin American Robotics Symposium, Sao Luis, Brasil, Sept 20-23.