Ball sensor fusion and ball interception behaviours for a Robotic Soccer Team

João Silva, Nuno Lau, João Rodrigues and José Luís Azevedo

IEETA / Department of Electronics, Telecommunications and Informatics
University of Aveiro, Portugal
{a25385,nunolau,jmr,jla}@ua.pt

Abstract. This paper presents some of the work developed in the CAMBADA team at the sensor and information fusion level, as well as developments at the behaviours level. Specifically, the ball sensor and information fusion techniques, to improve the position and velocity reliability, are described, as well as interception behaviours based on that reliable velocities, allowing for different approaching behaviours to the ball, are depicted. The described work improved the team performance, allowing it to distinctively achieve the 1st place in the Portuguese robotics open Robótica2008 and in the world championship RoboCup2008.

1 Introduction

Robotic soccer is nowadays a popular research domain in the area of multi robot systems. RoboCup[1] is an international joint project to promote artificial intelligence, robotics and related fields that includes several leagues, each one with a different approach, some only at software level, others at hardware, with single or multiple agents, cooperative or competitive [1]. The CAMBADA team was created to participate in the middle size league of robotic soccer, which requires the integration of several technologies in order for the robots to play soccer, such as autonomous agents architectures, multi-agent cooperation, strategy acquisition, real-time reasoning, control and sensor fusion.

In the context of RoboCup, the "middle size league" (MSL) is one of the most challenging. In this league, each team is composed of 4 to 6 robots with maximum size of 50x50cm base, 80cm height and a maximum weight of 40Kg, playing in a field of 18x12m. The rules [2] of the game are similar to the official FIFA rules, with required changes to adapt for the playing robots. Each robot is autonomous and has its own sensorial means. They can communicate among them through a wireless network and with an external computer acting as a coach. This coach computer has no sensor of any kind, it only knows what is reported by the playing robots. The agents should be able to evaluate the state of the world and make decisions suitable to fulfil the cooperative team objective.

CAMBADA, *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*, is the Middle Size League Robotic Soccer team from Aveiro University.

[1] http://www.robocup.org/

The project started in 2003, coordinated by the IEETA[2] ATRI[3] group. It involves people working on several areas for building the mechanical structure of the robot, its hardware architecture and controllers and the software development in areas such as image analysis and processing, sensor and information fusion, reasoning and control.
Since its creation, the team has participated in several competitions, both national and international. Each year, new challenges are presented, and new objectives are defined, always with a better team performance in sight. The team achieved the 1st place in the national competitions Robótica2007 and Robótica2008, the 5th place in the world championship RoboCup2007 and the 1st place in the world championship RoboCup2008.

This paper includes a brief description of the CAMBADA software agent in Section 2 to contextualize the described work. Having the objective of developing new behaviours to intercept the ball based on a prediction of its future position rather than moving to its current position, one felt the necessity to first improve the reliability of the estimation of the ball position and velocity. For that, improvements on the integration of ball information and sensor fusion of ball data were necessary, which are described in Section 3. The developed behaviours are presented in Section 4 and some conclusions and comments on the developed work are presented in Section 5.

2 CAMBADA

The CAMBADA team is currently composed of 6 robots (Fig. 1.a) with a general architecture described in [3,4,5] and also a description of the physical structure is present in [6].

The software system in each robot is distributed among various computational units. High level functions are executed on a PC, while low level functions are executed on microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [3,7,8] has been adopted. The RTDB is a data structure used by the robots to share their world models. It is updated and replicated in all players in real-time. The high-level processing loop starts by integrating perception information gathered locally by the robot, namely, information coming from the vision system and odometry information coming from the low-level layer. This information is afterwards stored in the RTDB. The next step is to integrate the robot local information with the information shared by team-mates, disseminated through the RTDB. The data stored in the RTDB is then used by another set of processes that define the specific robot behaviour, by generating commands that are sent down to the lowlevel control layer [6].

The tasks performed by the high-level loop are described in Fig. 1.b. As shown in the diagram, the integrator module is responsible for getting the sensorial data and building the representation of the state of the world. This representation is then used by the behaviours to define their set points and by the higher level layer. The roles and decision modules use the behaviours and act according to the conditions of the state of the world. The behaviours are the ones responsible for defining the commands to be sent down to the actuators of the robot.

[2] Instituto de Engenharia Electrónica e Telemática de Aveiro - Aveiro's Institute of Electronic and Telematic Engineering

[3] Actividade Transversal em Robótica Inteligente - Transverse Activity on Intelligent Robotics

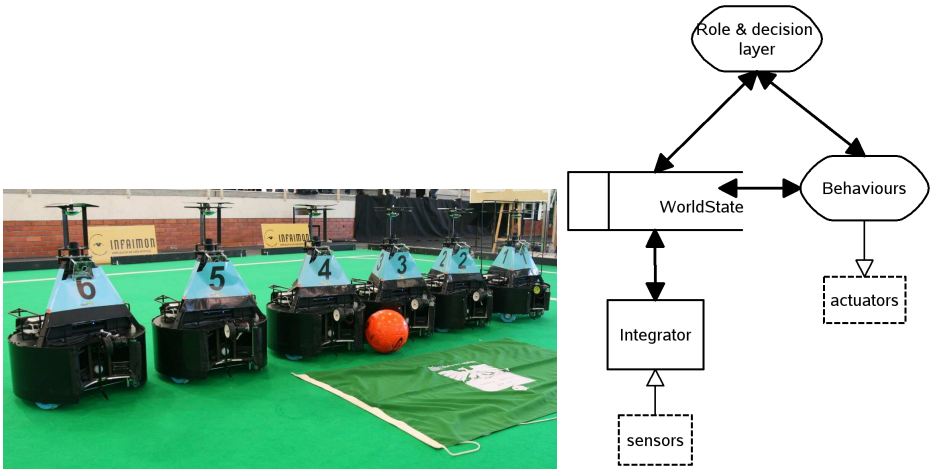


Fig. 1. Left, **a)**: Picture of the team robots; Right, **b)**: High-level tasks diagram.

The work described was done at the three lower levels of the diagram, the integrator, the worldstate and the behaviours.

3 Integration

The integrator module is responsible for updating the world state representation. For that task it may use the values stored in the previous representation, the current sensor measures (eventually after pre-processing) that has just arrived, the current actuator commands and also information that is available from other robots sensors or world state. This is essentially an information fusion problem. The most common methods to tackle information fusion are based on probabilistic approaches, including Bayes rule, Kalman filter and Monte Carlo methods [9]. Within RoboCup several teams have used Kalman filters for the ball position estimation [10,11,12,13]. In [13] and [12] several information fusion methods are compared for the integration of the ball position using several observers. In [13] the authors conclude that the Kalman reset filter shows the best performance.

In our case, a Kalman filter was chosen to refine the estimation of the ball position. The integrator module is the first task to be executed by the software agent, and starts by getting the RTDB vision information. It then updates the information provided by the low-level micro controllers. This update includes getting the odometry values and calculating the displacements since last cycle for x, y and angular components and getting the current battery and ball engaged sensor status. A verification of the coach computer is made, by reading the coach information on the RTDB. The coach computer defines the team colour and the goal colour. Also, the start and stop commands are given by the coach computer. The team mates information is read from the RTDB and the obstacle vector in the worldstate is updated by the current vision obstacle information.

The localisation algorithm (based on the Tribots team algorithm [14])then uses the vision information of the field lines and the odometry displacements to estimate the agent position on the field. After getting the position, a linear regression is used to estimate the agent velocity. Since the line based localisation can be ambiguous due to symmetry of the field, a verification is needed. This verification is made by getting the robot orientation from an electronic compass and estimate the orientation error. Afterwards, the integration of the ball position and velocity is done, the state of the game is updated and a free path for the opponent goal is estimated.

All the information available from the sensors in the current cycle is kept in specific data structures (Fig. 2), for posterior fusion and integration, based on both the current information and the previous state of the world.

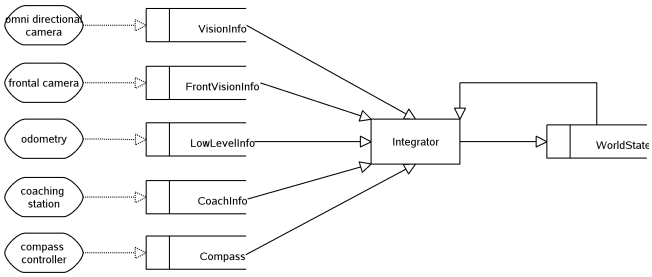


Fig. 2. Integrator functionality diagram.

3.1 Ball position estimation

The information of the ball state (position and velocity) is, perhaps, the most important, as it is the main object of the game and is the base over which most decisions are taken. Thus, its integration has to be as reliable as possible. To accomplish this, a Kalman filter implementation was created to filter the ball position estimates given by the visual information.

It is assumed that the ball velocity is constant between cycles. Although that is not true, due to the short time variations between cycles, around 40 milliseconds, and given the noisy environment and measurement errors, it is a rather acceptable model for the ball movement. Thus, no friction is considered to affect the ball, and the model doesn't include any kind of control over the ball. Therefore, given the Kalman filter formulation (described in [15]), the assumed state transition model is given by

$$X_k = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} X_{k-1}$$

where X_k is the state vector containing the position and velocity of the ball. Technically, there are two vectors of this kind, one for each cartesian dimension (*x*,*y*). This velocity is only internally estimated by the filter, as the robot sensors can only take measurements on the ball position. After defining the state transition model based on the ball

movement assumptions described above and the observation model, the description of the measurements and process noises are important issues to attend.

The measurements noise can be statistically estimated by taking measurements of a static ball position at known distances. The standard deviation of those measurements is used to calculate the variance and thus define the measurements noise parameter. In practice, the measurements of the static ball were taken while the robot was rotating over itself, to simulate movement and the trepidation it causes, so that the measurements were as close to real game conditions as possible. Some of the results are illustrated in Fig. 3.

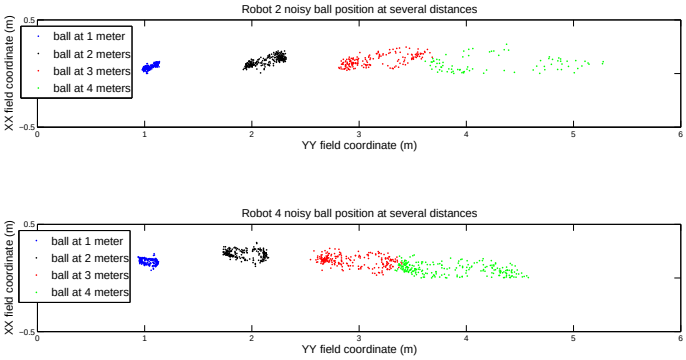


Fig. 3. Noisy position of a static ball taken from rotating robots.

A relation between the distance of the ball to the robot and the measurements standard deviation is modeled by the 2nd degree polynomial best fitting the data set in a least-squares sense (Fig. 4). A 1st degree polynomial does not fit the data properly, and assumes negative values for positive distance, which is not acceptable. Given the few known points, a 3rd degree polynomial would perfectly fit all 4 of them. However, these known points are also estimated and thus cannot be taken as exact. For that reason, a curve that would exactly fit them is not desirable.

As for the process noise, this is not trivial to estimate, since there is no way to take independent measurements of the process to estimate its standard deviation. The process noise is represented by a matrix containing the covariances correspondent to the state variable vector.

Empirically, one could verify that forcing a near null process noise causes the filter to practically ignore the read measures, leading the filter to emphasise the model prediction. This makes it too smooth and therefore inappropriate. On the other hand, if it is too high, the read measures are taken into too much account and the filter returns the measures themselves.

To face this situation, one had to find a compromise between stability and reaction. Given the nature of the two components of the filter state, position and speed, one may consider that their errors do not correlate.

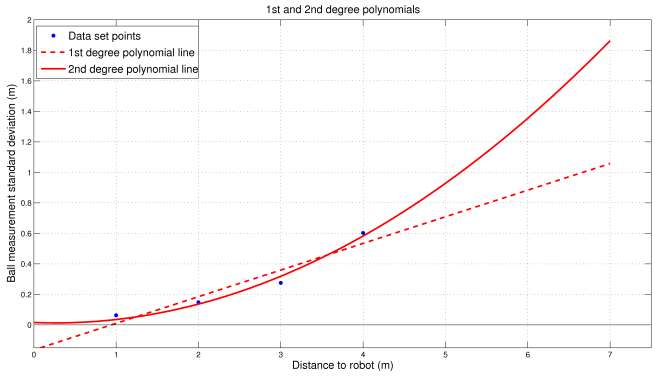


Fig. 4. Representation of the standard deviation value for variable distance to the robot. Data set points as blue dots. 1st degree polynomial as dashed line, 2nd degree polynomial as solid line.

Because we assume a uniform movement model that we know is not the true nature of the system, we know that the speed calculation of the model is not very accurate. A process noise covariance matrix was empirically estimated, based on several tests, so that a good smoothness/reactivity relationship was kept (Fig. 5).

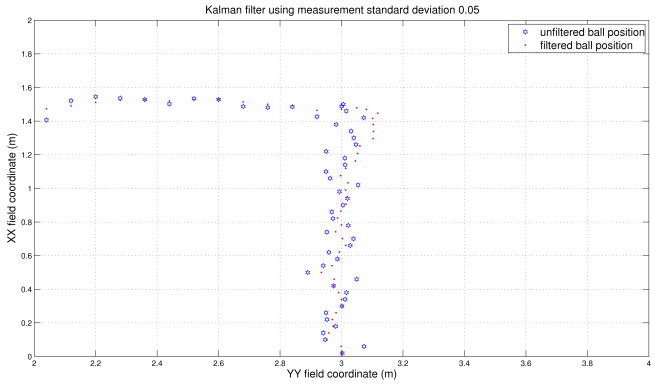


Fig. 5. Kalman filter over theoretical noisy positions of the ball. The unfiltered ball positions are the blue hexagrams, filtered positions are the red dots.

In practice, this approach proved to improve the estimation of the ball position. Figure 6 represents a capture of a ball movement, where the black dots are the ball positions estimated by the robot visual sensors and thus are unfiltered. Red stars represent the position estimations after applying the Kalman filter. The ball was thrown against the robot and deviated accordingly and the robot position is represented by the black star in its centre and its respective radius. It is easily perceptible that the unfiltered positions are affected by much noise and the path of the ball after the collision is deviated from the real path. The filtered positions however, seem to give a much better approximation to the real path taken by the ball.

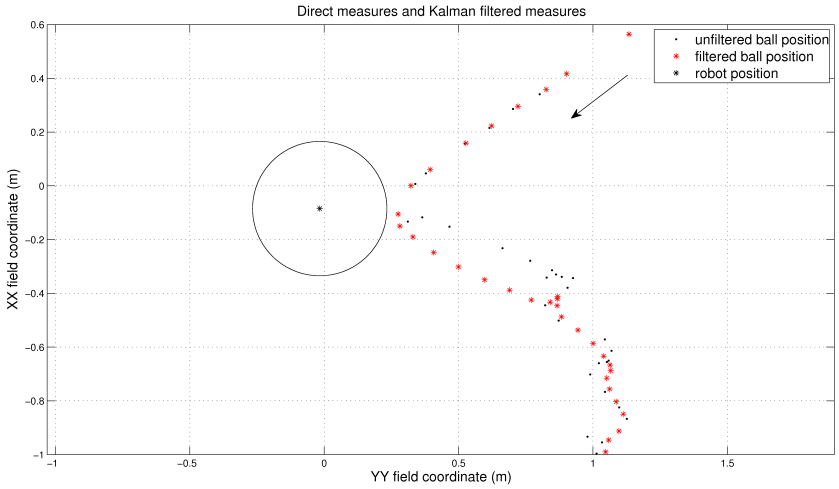


Fig. 6. Plot of a ball movement situation. See text for details.

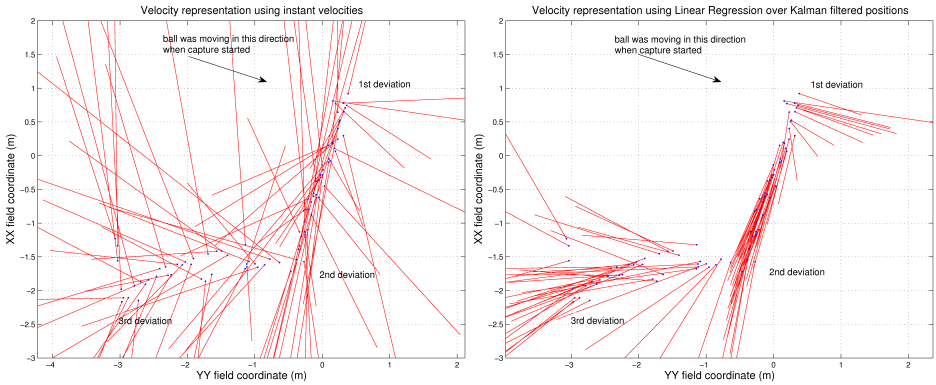
Using the filter *a-priori* estimation, a system to detect great differences between the expected and read positions was possible to implement, allowing to detect hard deviations on the ball path.

3.2 Velocity estimation

The calculation of the ball velocity is a feature becoming more and more important over the time. It allows that better decisions can be implemented based on the ball speed value and direction. Assuming the same ball movement model described in the previous section, constant ball velocity between cycles and no friction considered, one could theoretically calculate the ball velocity by simple instantaneous velocity of the ball with the first order derivative of each component $\frac{\Delta D}{\Delta T}$, being ΔD the displacement on consecutive measures and ΔT the time interval between consecutive measures. However, given the noisy environment it is also predictable that this approach would be greatly affected by that noise and thus its results would not be satisfactory (as it is easily visible in Fig. 7.a).

To keep a calculation of the object velocity consistent with its displacement, an implementation of a linear regression algorithm was chosen. This approach based on linear regression ([16]) is similar to the Tribots team [17] velocity estimation. By keeping a buffer of the last *m* measures of the object position and sampling instant (in this case buffers of 9 samples were used), one can calculate a regression line to fit the positions of the object. Since the object position is composed by two coordinates *(x,y)*, we actually have two linear regression calculations, one for each dimension, although it is made in a transparent way, so the description in this section is presented generally, as if only one dimension was considered.

When applied over the positions estimated by the Kalman filter, the linear regression velocity estimations are much more accurate than the instant velocities calculated by $\frac{\Delta D}{\Delta T}$, as visible in Fig. 7.b.



a) **b)**

Fig. 7. Velocity representation using: Left, **a)**: consecutive measures displacement; Right, **b)**: linear regression over Kalman filtered positions.

In order to try to make the regression converge more quickly on deviations of the ball path, a reset feature was implemented, which allows deletion of the older values, keeping only the *n* most recent ones. This reset results from the interaction with the Kalman filter described above, which triggers the velocity reset when it detects a hard deviation on the ball path.

Although in this case the Kalman filter internal functioning estimates a velocity, the obtained values were tested to confirm if the linear regression of the ball positions was still needed. Tests showed that the velocity estimated by the Kalman filter has a slower response than the linear regression estimation when deviations occur. Given this, the linear regression was used to estimate the velocity because quickness of convergence was preferred over the slightly smoother approximation of the Kalman filter in the steady state. That is because in the game environment, the ball is very dynamic, it constantly changes its direction and thus a convergence in less than half the cycles is much preferred.

The results achieved by the presented work served as a base for the development of the new behaviours presented in the next section.

4 Behaviours

The behaviours define the velocity to apply on the wheels, through 3 attributes *velX*, *velY* and *velA*, for velocity's *x*, *y* and *angular* components, respectively. Given the specificity of each different behaviour, they inherit the responsability of calculating the values of the velocity by its own means. Also, the control of both the kicker and the grabber is kept by the behaviours.

For the presented behaviours, the velocities are calculated based on the need to get to the defined target position.

The need for a way to approach the ball other than moving directly towards its instantaneous position became evident since it was verified that the robot describes a

completely unnecessary arc to get to the ball and in most cases would have to chase it back, loosing much time and strategic advantage.

Avoiding this kind of situations was a need, and thus the interception algorithms described below were created.

4.1 Passive interception

One approach to intercept the ball is to passively "wait" for it. This approach can be advantageous in situations where the robot should not go for the ball, but rather intercept its path and then, if the ball reaches it, move forward with it. The used approach is based on the ball velocity direction. The ball speed vector is extended as a line segment with origin in the ball. That line represents the ball path and the closest point of the path line to the robot position is calculated; geometrically, the perpendicular point is always the nearest. The calculated point is the one where the robot should go to in order to intercept the ball path (Fig. 8).

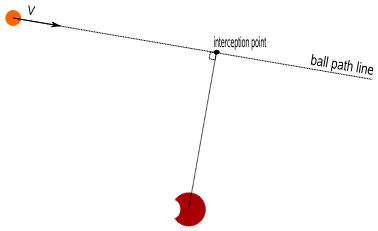


Fig. 8. Illustration of the passive interception behaviour. The ball velocity vector *V* is extended as a line defining its path. The robot calculates the closest point over the line, relatively to itself, and defines it as the interception point.

In the practical experiment represented in Fig. 9, the ball moves from right to left in a practically straight line. The ball positions are represented by blue hexagrams and the robot positions by red dots. The calculated interception point is represented by a small black circle, in this case, near (0.2, -1.7). It is visually verifiable that the interception point is, as expected, the perpendicular interception of the ball velocity direction line and the robot velocity direction line.

The results obtained by this approach were effective, although this behaviour does not guarantee the robot gets to the interception point before the ball in minimum time, as it only evaluates its direction and goes to the path. For that, another behaviour was defined, the active interception.

4.2 Active interception

The active interception appears naturally from the passive one, as in certain situations one is interested in getting in front of the ball in minimum time, and necessarily has to evaluate, besides the velocity direction, the speed value to calculate an interception point where the robot can get to before the ball. The algorithm used to calculate the interception point assumes a constant robot speed. Considering that unknown coefficients

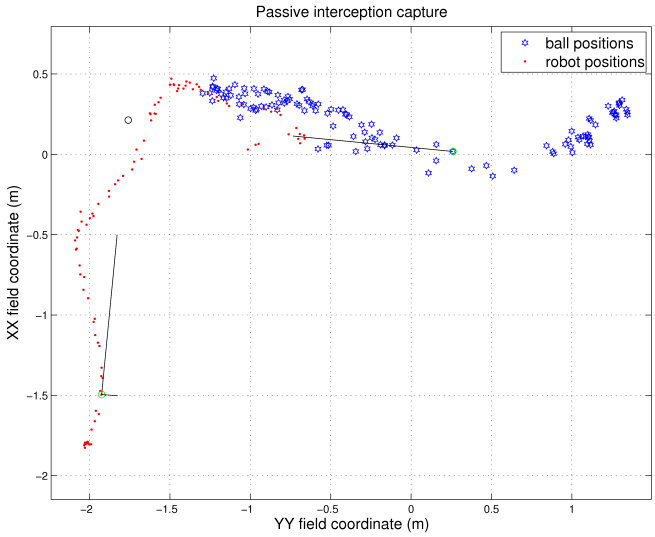


Fig. 9. Plot of a capture of a passive interception (the presented positions are estimated by the robot). See text for details.

of friction, slidings on the wheels, and other external factors affect the robot speed, the constant usually assumed is a bit conservative, currently 1.0 m/s.

The idea is to evaluate the ball line path and estimate a point over that line where the robot can get before the ball does (Fig. 10).

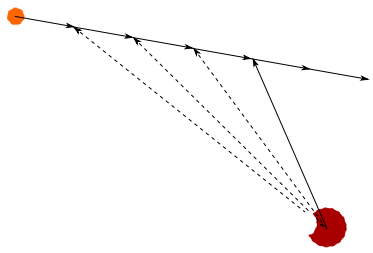


Fig. 10. Illustration of the active interception behaviour. At a given point over the ball path line, it is estimated that the robot can get there before the ball, and it becomes the interception point.

In the practical experiment represented in Fig. 11, the ball moves from top centre to bottom centre with a sightly curved path. The calculated interception point is represented by the small black circle near (0.2, -0.1). Given the variations in speed evaluation and the conservative assumed robot speed, the robot path shows that the robot goes ahead of the ball and then goes back to catch it. For plot interpretation, one must keep in mind that the hexagrams and dots represent the centre of both the ball and the robot; each of these objects have a radius, and thus, when the robot has the ball engaged, the dots are necessarily separated by around 25cm.

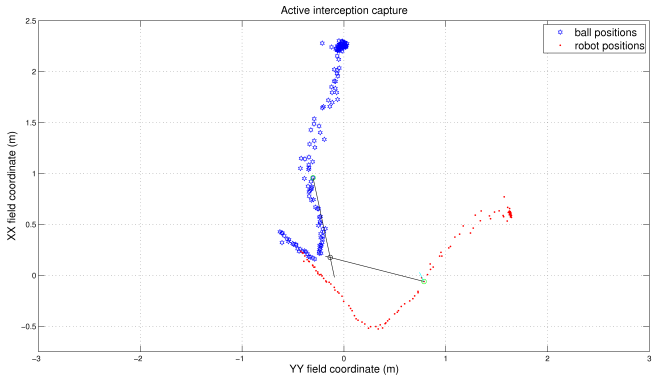


Fig. 11. Plot of a capture of an active interception (the presented positions are estimated by the robot). See text for details.

This interception behaviour takes advantage of a well refined ball velocity estimation, and proved to be quite effective.

5 Conclusion

Reliable estimations of the ball position and velocity are an important aspect to develop, since high level decisions can and should be based on the way the ball is moving.

The chosen techniques for information and sensor fusion proved to be effective in accomplishing their objectives. The Kalman filter allows to filter the noise on the ball position and provides an important prediction feature which allows fast detection of deviations of the ball path. The linear regression used to estimate the velocity is also effective, and combined with the deviation detection based on the Kalman filter prediction error, provides a faster way to recalculate the velocity in the new trajectory.

At the behaviours level, the implementation of the interceptions and the approach point calculation are the first step for “predictive” features on the CAMBADA robots, as they allow for movement ahead of the ball, instead of going directly to it. However, it was only possible to make them effective due to the improvement of the ball velocity accuracy.

The new passive interception behaviour brought a new dynamic to the game, as it was used in real game situations during the Robótica2008 games and RoboCup2008 games.

The active interception behaviour was not tested in real game situations yet, but can provide a good solution for future needs of the team strategy.

The described work improved the team performance, allowing it to distinctively achieve the 1st place in the Portuguese robotics open Robótica2008 and the 1st place in the RoboCup2008.

Acknowledgments

This work was partially supported by project ACORD Adaptive Coordination of Robotic Teams, FCT/PTDC/EIA/70695/2006.

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. Proceedings of the first international conference on Autonomous agents (1997) 340–347
2. MSL Technical Committee 1997-2008: Middle Size Robot League Rules and Regulations for 2008 (2007)
3. Almeida, L., Santos, F., Facchinetti, T., Pedreiras, P., Silva, V., Lopes, L.: Coordinating distributed autonomous agents with a real-time database: The CAMBADA project. In: Proc. of the ISCIS, Springer (2004)
4. Silva, V., Marau, R., Almeida, L., Ferreira, J., Calha, M., Pedreiras, P., Fonseca, J.: Implementing a distributed sensing and actuation system: The CAMBADA robots case study. In: Proc. of the 10th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2005. Volume 2. (2005)
5. Azevedo, J., Cunha, B., Almeida, L.: Hierarchical distributed architectures for autonomous mobile robots: A case study. In: Proc. of the 12th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2007. (2007) 973–980
6. Azevedo, J.L., Lau, N., Corrente, G., Neves, A., Cunha, M.B., Santos, F., Pereira, A., Almeida, L., Lopes, L.S., Pedreiras, P., Vieira, J., Martins, D., Figueiredo, N., Silva, J., Filipe, N., Pinheiro, I.: CAMBADA'2008: Team Description Paper (2008)
7. Santos, F., Almeida, L., Pedreiras, P., Lopes, L., Facchinetti, T.: An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among Mobile Autonomous Agents. In: Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS 2004. (2004)
8. Pedreiras, P., Teixeira, F., Ferreira, N., Almeida, L., Pinho, A., Santos, F.: Enhancing the Reactivity of the Vision Subsystem in Autonomous Mobile Robots Using Real-Time Techniques. In: RoboCup Symposium: Papers and Team Description Papers, Springer (2005)
9. Durrant-Whyte, H., Henderson, T.: Multisensor Data Fusion. In: Springer Handbook of Robotics. Springer (2008)
10. Lauer, M., Lange, S., Riedmiller, M.: Modeling Moving Objects in a Dynamically Changing Robot Application. In: Advances in Artificial Intelligence. Springer (2005) 291–303
11. XU, Y., JIANG, C., TAN, Y.: SEU-3D 2006 Soccer Simulation Team Description (2006)
12. Marcelino, P., Nunes, P., Lima, P., Ribeiro, M.I.: Improving object localization through sensor fusion applied to soccer robots. In: Proc. of the Robótica2003. (2003)
13. Ferrein, A., Hermanns, L., Lakemeyer, G.: Comparing Sensor Fusion Techniques for Ball Position Estimation. In: Proc. of the RoboCup 2005 Symposium. Springer (2005)
14. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: an efficient and accurate approach for robot self-localization. In: RoboCup Symposium, Springer (2005)
15. Bishop, G., Welch, G.: An Introduction to the Kalman Filter. In: Proc of SIGGRAPH, Course 8. Number NC 27599-3175, Chapel Hill, NC, USA (2001)
16. Motulsky, H., Christopoulos, A.: Fitting models to biological data using linear and nonlinear regression. GraphPad Software Inc. (2003)
17. Lauer, M., Lange, S., Riedmiller, M. In: Modeling Moving Objects in a Dynamically Changing Robot Application. Springer (2005) 291–303