ELSEVIER
IFAC
PUBLICATIONS
www.elsevier.com/locate/ifac

AN APPLICATION OF THE EXTENDED KALMAN FILTER TO ROBOT SOCCER LOCALISATION AND WORLD MODELLING.

Rick Middleton, Michaela Freeston and Leonie McNeill

Newcastle Robotics Laboratory
The University of Newcastle
NSW 2308, Australia
http://www.robots.newcastle.edu.au

Abstract: The extended Kalman filter (EKF) has been used in a range of applications for filtering, smoothing, data fusion and related operations. In robot soccer, both self localisation and world modelling benefit substantially from these features. In this paper, we review the application of the EKF to robot soccer. We pay particular attention to the enhancements, tuning and fixes we needed make to the basic algorithm to ensure adequate performance in the competition.

Keywords: Robotics, Kalman Filtering, Estimation

1. INTRODUCTION

Robot localisation and world modelling describes the process by which an autonomous robot infers information about its own location (Localisation) and the location of other objects (world modelling) from sensor information. This can be expressed as a Markov estimation problem (see for example Fox, Burgard and Thrun (1999). A number of different algorithms have been proposed for such situations including particle filters, Monte Carlo techniques and extended Kalman filters (EKFs) Fox et al (2003), Schmitt, Hanek and Beetz (2003), Rofer and Jungel (2003).

In selecting an appropriate algorithm amongst the many possible, important factors were our background experience with the EKF (Goodwin et al (2001), Fresston (2002)) Note that the limitation of the EKF to unimodal distributions can be overcome by a range of data association algorithms (see for example the text Bar-Shalom and Fortmann (1988)) that have been developed in the past for applications such as radar tracking

In Section 2 we briefly give some background to the Robocup soccer four legged league competition and the software system used by the Newcastle University team (the "NUbots") (Bunting et al 2003). In Section 3 we give a description of the EKF we designed, implemented and used, focussing on some enhancements and novel features of the algorithm used for Robot Soccer. In Section 4 we give some preliminary results from laboratory tests, soccer game competitions, and challenge[1] competitions. Section 5 concludes the paper with discussion including a number of topics for further work.

2. SYSTEM OVERVIEW

2.1 *RoboCup Four-Legged League*

RoboCup is an international competition with the aim: "By the year 2050 develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team"[2]. Our focus here is on the Four Legged League of RoboCup soccer.

The Four Legged league involves four vs four competitions using Sony Aibos. Each half of the game goes for 10 minutes. Competition rules constrain actions such as: duration of 'holding' the ball, obstruction, keeper charging, and defender locations. Penalties involve a robot being removed from play for up to 30 seconds, and then replaced on the centre line. The field of play has dimensions of 270cm x 420cm (see Fig. 1). The objects on the field have predefined colour combinations to aid localisation and world modelling.

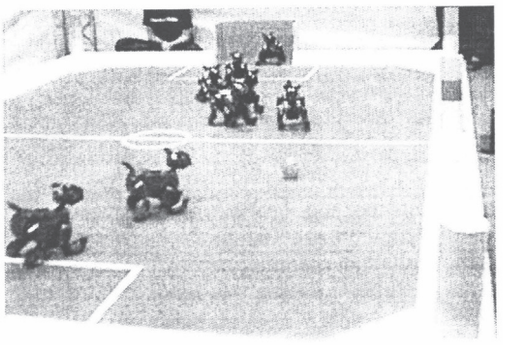

Fig. 1. RoboCup Four-legged League field

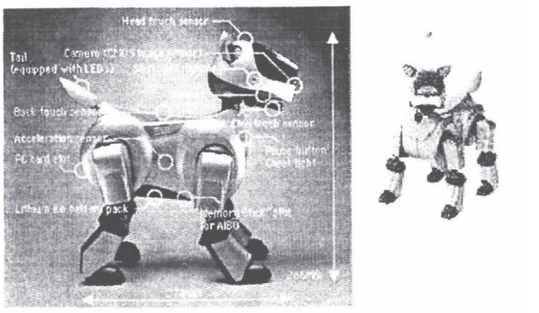

Fig. 2: Sony Aibo – ERS210A

[1] The 'challenges' refer to structured competitive test tasks. Robocup 2003, included a localisation challenge.

[2] *Robocup Official Website*, http://www.robocup.org/

2.2 *Aibo Hardware Overview*

The robots used in the RoboCup Four-legged League in 2003 are Sony Aibo entertainment robots, model ERS210A (see Fig. 2). The robots have several sensors and actuators, which assist them in playing soccer. The most important of these is a colour camera that operates at 25 frames per second. This camera has a field of view of about ± 30° and a resolution of 176 x 144 pixels. The main actuators are the leg (4*3=12) and neck (3) motors, with associated potentiometers.

2.3 *NUbots Software Overview:*

The NUbots software for playing soccer is split into four main modules: Vision, Localisation and World Modelling, Behaviour and Locomotion, as illustrated in Fig. 3:

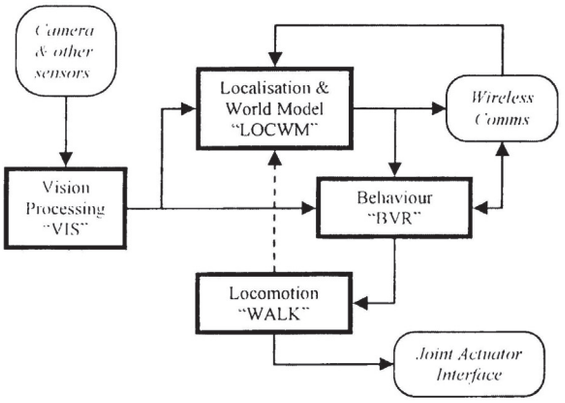

Fig. 3: Software Structures Block Diagram

The Vision Module receives images from the camera, classifies pixels into colours using a table lookup procedure, and forms objects from groups of pixels. Features such as range and bearing are extracted from these objects. Edge detection is also performed on the colour classified image to determine the distance and bearing to certain corners and lines seen in the camera image.

These distance and bearing measurements, plus other information, are used by the Localisation and World Modelling module to determine an estimate of the position of both the robot and of other objects in the field. These estimates are calculated using an EKF (see Section 3), which also utilises measurements from other robots regarding their own position and that of the ball received via wireless communications.

The Behaviour module uses information from Vision and Localisation & World Modelling to determine game strategy and actions. When an action is decided upon, the Locomotion module is called on to carry it out. The Locomotion software converts actions such as walking, turning and kicking into a series of commanded joint angles. The locomotion module also passes information to localisation on the estimated distance moved and angle rotated by the robot. This 'odometry' information allows improved localisation, particularly when there is limited vision of suitable landmarks.

3. AN EXTENDED KALMAN FILTER FOR LOCALISATION

The NUbots team in RoboCup 2003 modelled the motion of the robot itself, and the soccer ball. For this purpose, we use an EKF (see for example Fox, Burgard and Thrun (1999), Welch and Bishop(2001), or Goodwin et al (2001)). We first describe the model structure used for the EKF.

3.1 *Model Structure*

We use a five state model, using world centred coordinates:

$$x^T = \begin{bmatrix} x_r & y_r & \theta_r & x_b & y_b \end{bmatrix} \quad (1)$$

where (x_r, y_r) are the robot's Cartesian co-ordinates, θ_r is the robot's heading and (x_b, y_b) are the ball Cartesian co-ordinates. We chose to represent all these variables in a single vector due to the high degree of correlation between them. For example, the perceived bearing to the ball depends on all five state variables.

From this state definition, we model the time update (i.e. odometry update) as:

$$x_k = x_{k-1} + u_k + w_k \quad (2)$$

where

$$u_k^T = \begin{bmatrix} -d_l \sin\theta_r - d_l \cos\theta_r & d_f \cos\theta_r - d_l \sin\theta_r & d_r & 0 & 0 \end{bmatrix}$$

is the update based on the odometry estimates: (d_f, d_l) are the forward and left component of motion relative to the heading of the robot (θ_r) and d_r is the rotational component of the robot motion; w_k is the 'process noise' component to take account of odometry errors; and k is the time index. Based on this model, the 'Time update equations' implemented were:

$$\hat{x}_k^- = \hat{x}_{k-1} + \hat{u}_k \quad (3)$$

$$P_k^- = P_{k-1} + Q_k \quad (4)$$

where $\hat{u}_k$ is the estimated change in position based on odometry, and the current best estimate of the robot heading; $\hat{x}_{k-1}$ is the previous estimate of the state; $\hat{x}_k^-$ is the new a-priori estimate of the state; and P_k^-, P_{k-1} are the a-priori state covariance and previous state covariance respectively. Q_k is a matrix chosen to capture the covariance of the process noise w_k.

We make use of standard measurements such as range and bearing to a beacon or ball. In addition we also use information from corners of the penalty box, the field centre, and intersections between the centreline and the border. In some of these cases, there may be ambiguity in the feature, although in most cases this ambiguity could be eliminated by incorporating the appropriate sanity checks in vision processing or comparing with the model. In all cases, the measurement data can be expressed as a nonlinear function of the states with noise representing measurement errors:

$$y_k = h(x_k) + v_k \quad (5)$$

The EKF linearises this equation, and for simplicity of implementation, we use the Jacobian as a

linearization, $C_k = \left[\frac{\partial h}{\partial x}\right]_{x=\hat{x}_k^-}$. The appropriate measurement update equations are then:

$$J_k = \frac{P_k^- C_k^T}{C_k P_k^- C_k^T + R_k} \quad (6)$$

$$\hat{x}_k = \hat{x}_k^- + J_k(y_k - h(\hat{x}_k^-)) \quad (7)$$

$$P_k = (I - J_k C_k) P_k^- \quad (8)$$

where R_k is the 'measurement noise covariance' intended to capture the effects of measurement errors in v_k.

Example vision data with no robot motion (apart from head panning) are illustrated in Fig. 4. (Note that "YP" and "GP" refer to Yellow above Pink and Green above Pink beacons respectively.) Clearly there are significant errors, particularly for long range measurements.

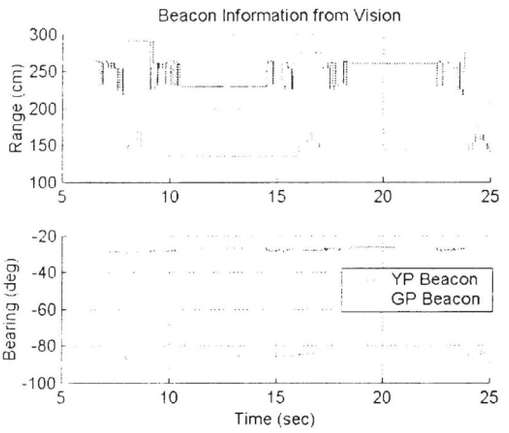

Fig. 4. Example Beacon Information for Localisation

3.2 *EKF Initialisation*

Having selected the state variables and model update equations, to implement an EKF, the user must select a number of parameters. These required parameter selections include initial values for the state estimates and covariance, and also the process and measurement noise covariances. The initial variables were selected as the centre of the field and the initial covariance was made to be large enough to easily cover the field dimensions, and full 360 degree rotational uncertainty. More care is needed with the selection of the process and measurement noise errors.

Selection of Process Noise Covariance, Q_k. It might be expected (heuristically) that given no measurements, the uncertainty in the robot's own position would increase linearly depending on how accurately the robot knew its input (locomotion) data. For this to be the case we should have:

$$\sqrt{P_k^-} = \sqrt{P_{k-1}} + \Delta \quad (9)$$

where Δ is the covariance of the locomotion data. However, according to the standard time update (3.4) we will have:

$$\sqrt{P_k^-} = \sqrt{P_{k-1} + Q_k} \quad (10)$$

and therefore selecting a fixed Q_k will not achieve the desired linear growth in standard deviation. Instead, if we select a variable process noise covariance as:

$$Q_k = 2\Delta\sqrt{P_{k-1}} + \Delta^2 \quad (11)$$

then we do indeed achieve the desired behaviour. For simplicity, and to avoid the need to compute matrix square roots on line, we take a diagonal approximation to (11).

Constraining Positions within the Field. Due to measurement noise, locomotion and linearization errors, the Kalman Filter may not always return Cartesian coordinates (for either the ball or robot) that are within the soccer field. In other words, if we denote the field region by $\mathcal{R}$ in the state space, then we seek to ensure $\hat{x}_k \in R$. The simplest solution to this problem, given that the field is approximately rectangular, is to clip the individual coordinate that is out of range back into the field. This is equivalent to the orthogonal projection:

$$\hat{x}_k' = \arg\min_{x \in R} (x - \hat{x}_k)^T (x - \hat{x}_k) \quad (12)$$

where $\hat{x}_k'$ is the state estimate after projection.

As has been previously suggested, a more insightful solution is to use the fact that there is an estimate of the certainty of the measurements and adjust the robot back onto the field taking into account the uncertainty and correlations in the variables:

$$\hat{x}_k' = \arg\min_{x \in R} (x - \hat{x}_k)^T P^{-1} (x - \hat{x}_k) \quad (13)$$

It is well known that this projection is unique for all possible $\hat{x}_k$ starting points if and only if the region R is convex. Furthermore, the action of the projection is guaranteed to produce an estimate that is closer (in the sense of being most probable, as described by the P^{-1} norm) if and only if the region is convex. Thus the exact projection to the exact field boundary, which is a non-convex shape when the goal areas are included, is not practical. Prior to 2003, we dealt with this problem by implicitly ignoring the goal area as illustrated in Fig. 5(a). We now use the convex hull of the true field region as illustrated below in Fig. 5(b):

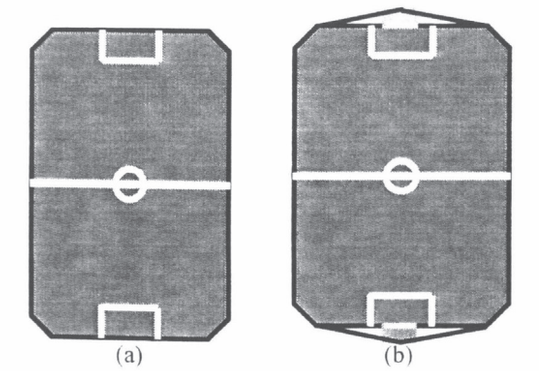

Fig. 5. Field Shapes used for Kalman Filter Estimates Projection

Note that since the region, $\mathcal{R}$, is a convex polygon this can be represented as a finite intersection of half planes, i.e. linear inequality constraints. The appropriate projection can be performed by solving an associated constrained least squares optimisation problem. Solving this constrained optimisation, requires solution of a quadratic program with up to 4 linear inequality constraints active. This can be described by equation (14) where the linear inequality constraints are represented by the half spaces $H_i := \{x : c_i^T x < 1\}; i = 1..n_c$:

$$\hat{x}_k = \arg\min_{x \in H_i, i=1..n_c} \left\{ \left(x - \hat{x}_k\right)^T P^{-1} \left(x - \hat{x}_k\right) \right\} \qquad (14)$$

Equation (14) is a quadratic program to which there are a number of quite efficient iterative solutions. To further simplify computations we approximate this problem by considering only a single linear equality constraint at a time, as described by equation (15):

$$\begin{aligned} &\hat{x}^0 = \hat{x}_k \\ &for\ i = 1...n_c \\ &\quad \hat{x}^i = \arg\min_{x \in H_i} \left\{ \left(x - \hat{x}^{i-1}\right)^T P^{-1} \left(x - \hat{x}^{i-1}\right) \right\} \end{aligned} \qquad (15)$$

Note that although (15) does not guarantee that the estimate $\hat{x}^{n_c}$ satisfies all constraints, it does guarantee convergence 'closer' (in the P^{-1} norm) to both the true parameter estimates, and to the constraint region[3]. It also has the highly desirable property of a very simple algorithm with a guaranteed maximum CPU demand per sample.

Lost Robot Detection (or the 'kidnapped robot problem'). One of the dilemmas recognised by a number of teams is when a robot may undergo a dramatic change due to, for example, human intervention to implement a penalty. Note that this problem is equivalent to a long studied problem in detecting and tracking jumps (see for example Willsky and Jones (1976)). In principle, this type of situation could be detected by examining where there is a conflict between the measurements expected by the current model (prior to measurement update) and the measurements observed. However, this conflict between estimates based on prior data, and current measurements may have two possible root causes: (i) the measurements could be erroneous (i.e. there are rare occasions when, despite the best efforts and sanity checks in vision, objects will be perceived at false locations); or (ii) there may have been major unpredicted changes in the robot or ball location (for example, the most severe case of this is when a robot is penalised, and moved to the centre of the field).

We have implemented a simple heuristic for deciding between these different possibilities. Firstly, determine whether innovations is unusually large (in comparison with the modelled variance). If so, we tentatively treat the measurement as an outlier. After all available measurements have been processed, we check whether in the recent past, several different object types have been treated as outliers. If so, we make a decision that the discrepancy is due to a 'kidnap' problem, rather than occasional measurement outliers. Otherwise, only a small number of different objects have given recent inconsistent measurements, and we treat the measurements as being outliers.

An obvious technique to employ once a 'lost robot' determination has been made is to reset the Kalman Filter uncertainty values. This allows the estimate to rapidly converge to the new position without compromising localisation performance in normal situations.

3.3 Error Sources

The main sources of error that affect the robot localisation Kalman Filter and the ball position Kalman Filter are summarised in the following sections.

Robot Position Error Sources. The errors that contribute to the robot localisation Kalman Filter Model/Process Noise:

- Robot locomotion data errors: Inaccuracy in locomotion data can be attributed to a number of factors including slip on the field surface, individual robot motor differences, and collisions with other objects (walls and other robots). Of the three, the latter is by far the largest source of error.
- Model linearisation error.

The vision errors that contribute to the robot localisation Kalman Filter Measurement Noise include:

- Misclassified camera pixels: Colour similarities (for example, yellow and orange) can cause objects to be 'seen' where there is no object, or pixels at the edge of a colour block not to be recognised. These errors can alter the perceived object size and bearing.
- Granularity (limit of reading): The calculation performed to determine a beacon's distance from the robot is computed as 'constant'/(height of pixels). This granularity in range may give significant errors, in some cases as large as 60cm.
- Camera velocity: Observations of the distortion of objects when the camera is moving fast indicate that each camera frame update is not very fast. Objects can appear to be 'spread' across the frame which distorts the distance and angle readings.

Ball Position Error Sources. The ball position Kalman Filter Model/Input Noise accounts for the fact that there is constant movement of the ball but no knowledge of the input driving this. In addition to the vision errors mentioned already, there are extra vision processing problems involved in measurements of the ball, particularly when the ball is occluded or at close range.

[3] Note that this convergence to a closer position is guaranteed since each step of the iterations has this property.

4.1 Stationary Robot

Most testing of the Kalman Filter was performed on a stationary robot. While not representative of conditions during game-play, the behaviour of the filter on a stationary robot is relatively easy to track, and allows for comparisons between the filter estimates and real world data. It is sufficient for detecting and fixing most errors in the filter. There are two main issues with the performance of the filter on a stationary robot. These are the time taken to reach a reasonable estimate and the stability and accuracy of that estimate.

The performance of a stationary robot was tested by placing a robot in various positions on the field, with no obstructions to vision, such as a ball or other robots. The robot's head was constantly panning from side to side, looking at different objects around the field. Data was streamed via wireless to a computer and stored in a text file. The actual position was measured with a tape measure to the nearest cm.

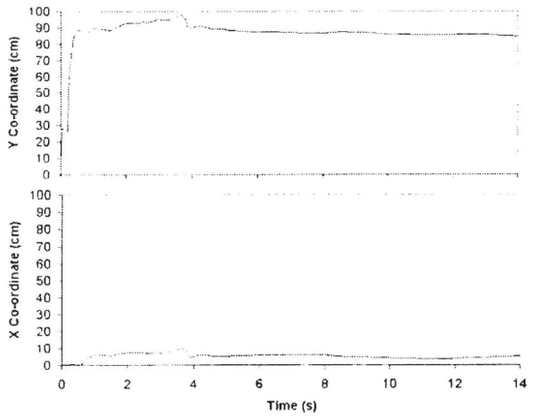

Fig. 6. Time response of robot position estimates

Fig. 6 illustrates a typical time response of the filter for the robot's x and y co-ordinates and heading. In this case, despite large initial error in the position (90cm) the filter reaches an estimate close to the mean value in less than 1 second. In robot soccer this time affects how long it takes a robot to 'find itself' if it becomes lost or is moved (for example if it is penalised). This level and speed of accuracy is more than adequate for robot soccer.

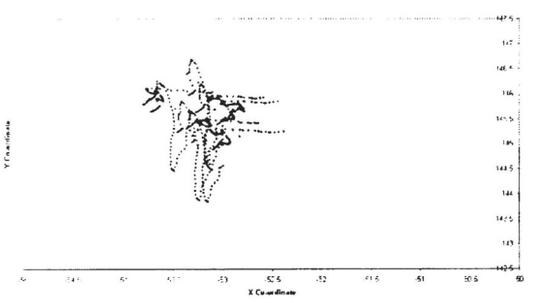

Fig. 7. Stationary Robot estimated position

Fig. 7 shows the amount of variation in the position of the robot in terms of x and y. This data is taken after the filter has had time to stabilise after the value can be seen.

We note that typically, the position estimates are within around 10cms of the correct location, and a few degrees of the correct heading. In both cases this is more than adequate for positioning and strategy decisions in robot soccer. Note that the much higher accuracies demanded for chasing, kicking the ball and other skills are achieved by using relative location directly from vision information, and therefore do not rely on the absolute accuracy of localisation.

4.2 Robot in Motion

A robot in motion is far harder to track than a stationary one, and obtaining an indication of how well the filter is performing while the robot is moving is difficult. Most debugging of a moving robot is done by a visual comparison between the robot's position on the field and its position estimate as indicated by our World Model debug program. This is not particularly accurate, but is sufficient to spot serious errors in the filter.

Providing position estimates for a robot in motion presents problems that are not relevant when it is stationary. The vision data received is lower quality because the motion of the robot has effects on the camera image that can't always be accounted for in processing vision. The moving robot is provided with data on how its position has changed, but this data becomes invalid if a collision (with a wall or another robot), or slippage occurs. Currently locomotion software has no way of knowing if this is occurring. The filter is also forced to respond faster, to keep up with the motion of the robot. This gives the filter less time to reach a relatively stable estimate. All these factors mean that the performance of the filter while the robot is moving is not as good as while the robot is stationary.

A quantitative measure of a robot in motion can be given by programming the robot to walk along a set path. In this test the robot began at (x,y) = (0,160) and walked roughly straight ahead at constant velocity for 10 seconds until it reached (x,y) = (-50,-75).

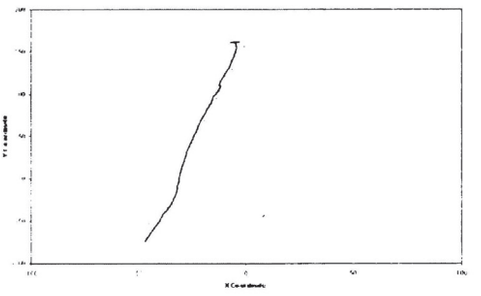

Fig. 8. Robot estimated position (dashed – target track; solid – estimated track)

Fig. 8. shows the Kalman Filter estimate of the path the robot took. This test still does not give access to the exact real world data, but it gives a reasonable idea of the path the robot took and the performance of the Kalman Filter while the robot is in motion. Due to some fairly accurate Locomotion data, the filter response is fairly smooth and relatively close to the actual path taken.

Again from these tests, the distance from the track to the nearest point on the desired track is approximately 10cm, which is consistent with tests conducted under stationary conditions.

4.3 Observations during the Competition

There are several ways in which observation may be used to qualitatively test the performance of the algorithm. The first of these is to observe the robot during game play.

Robot Soccer Play Observations. During a game, errors in localisation or world modelling show up in a range of possible ways. These are described below, with comments on our performance from the 2003 Robocup competition:

- *Kicking the ball in the wrong direction.* This can easily occur if there are large errors in heading. During the entire 2003 competition we had only a single own goal.
- *Goal Keeper Lost.* Since the goal keeper behaviour is designed to keep the goal keeper at a fairly fixed location on the field, it is usually easy to observe if the goal keeper is lost. There were no known incidences of this in the 2003 competition.
- *Defender not correctly aligned.* Whilst attacking, our behaviour sets a single robot as the defender, positioned on the line segment joining the ball and the defending goal. We rarely observed any significant signs of incorrect alignment.

Localisation Challenge Results. The Localisation Challenge for 2003 involved a situation where the simplest landmarks available for localisation were removed, namely the 6 beacons. The robots had to perform localisation and move to defined points on the field, within quite tight tolerances. Our team was one of a minority of teams that was able to find any of the target locations. We came 5th in this challenge, being awarded at least some points at 3 of the 5 targets. For the remaining target points, we were only slightly outside the maximum distance required. This was achieved using essentially the same algorithms that were used for the soccer, with only minor modifications to the algorithms used in the competition itself.

These tests were repeat in a laboratory situation and gave errors of (18,0,21,0,30)cm respectively for the five target points.

5. CONCLUSION

The NUbots system of localisation and world modelling based on the extended Kalman Filter performed well in the Robocup 2003. A large number of enhancements were made compared to early code, such as using an integrated, 5th order model for the robots own location and the ball location. The localisation challenge was performed with essentially no changes to the regular game code for localisation, and we appeared to be 'close' to all required target points (though not close enough to score full points). In an idealised setting, the same algorithm gives positional accuracy of a few cm, and heading accuracy of within a few degrees in self localisation. Qualitatively, the performance during game play showed few discernable flaws in the basic algorithm, which exhibits good basic accuracy, robustness to both outliers, and 'kidnap' problems, whilst retaining low computational cost. Future work may examine extensions of the model to take into account modelling for other robots on the field (team mates and opponents).

6. REFERENCES

Bar-Shalom, Y. and T. Fortmann, *Tracking and Data Association*, Academic Press, 1988

Bunting, J., S. Chalup, M. Freeston, W. McMahan, R. Middleton, C. Murch, M. Quinlan, C. Seysener, G. Shanks, "Return of the NUbots! The 2003 NUbots Team Report", Newcastle Robotics Lab, The University of Newcastle, October 2003 http://www.robots.newcastle.edu.au/publications/NUbotFinalReport2003.pdf

Fox, D., W. Burgard and S. Thrun, "Markov localization for mobile robots in dynamic environments", *Journal of AI Research*, V11, pp391-427, 1999

Fox, D., J. Hightower, L. Liao and D. Schulz, "Bayesian Filtering for Location Estimation", IEEE Pervasive Computing, pp24-33, V2, N3, 2003.

Freeston, L. "Applications of the Kalman Filter Algorithm to Robot Localisation and World Modelling", School of EE&CS, The University of Newcastle, June 2002 http://murray.newcastle.edu.au/users/students/2002/c9806118/main.html

Goodwin, G., R. Middleton, B. Ninness and S. Weller, "Kalman Filters Short Course Notes 2001", CIDAC, The University of Newcastle, University Drive, Callaghan NSW 2308, Australia.

Rofer, T. and M. Jungel, "Fast and robust edge based localization in the Sony four-legged robot league", Proc. Robocup 2003 Int. Sym., Padova, Italy, July 2003.Welch, G. and G. Bishop, "An Introduction to the Kalman Filter", Feb, 2001. Available from http://www.cs.unc.edu/~welch/kalman/.

Schmitt, T., R. Hanek and M. Beetz, "Developing comprehensive state estimators for robot soccer", Proc. Robocup 2003 Int. Sym., Padova, Italy, July 2003.

Willsky, A. and H. Jones, "A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems", IEEE Transactions on Automatic Control, V21, N1, pp108-112, 1976.

Acknowledgements

We would like to thank SONY Corporation; the ARC Centre for Complex Dynamic Systems and Control; School of Electrical Engineering and Computer Science; and the Faculty of Engineering and Built Environment at the University of Newcastle.