Efficient Localization For Robot Soccer Using Pattern Matching

Thomas Whelan, Sonja Stüdli, John McDonald, and Richard H. Middleton*

Department of Computer Science, NUI Maynooth, Maynooth, Co. Kildare, Ireland
Hamilton Institute, NUI Maynooth, Maynooth, Co. Kildare, Ireland
thomas.j.whelan@nuim.ie,sonja.stuedli@nuim.ie,
johnmcd@cs.nuim.ie,richard.middleton@nuim.ie,

Abstract. One of the biggest challenges in the RoboCup Soccer Standard Platform League (SPL) is autonomously achieving and maintaining an accurate estimate of a robot's position and orientation on the field. In other robotics applications many robust systems already exist for localization such as visual simultaneous localization and mapping (SLAM) and LIDAR based SLAM. These approaches either require special hardware or are very computationally expensive and are not suitable for the Nao robot, the current robot of choice for the SPL. Therefore novel approaches to localization in the RoboCup SPL environment are required. In this paper we present a new approach to localization in the SPL which relies primarily on the information contained within white field markings while being efficient enough to run in real time on board a Nao robot.

1 Introduction

In our earlier work [12] we gave some initial thoughts and results for an algorithm based on Cox's algorithm (a form of least squares error pattern matching) for effective robot self localization based on field markings. As we continued to use this algorithm, we became aware of a number of significant shortcomings in the algorithm in the form proposed in [12] and in addition, were able to introduce a number of additional features suggested in this previous work. The main aim of this paper is to address some of the issues highlighted in our previous work and describe our progress with some of the future work mentioned therein [12]. This paper is also intended to function as a stand alone reference document for those who wish to implement this approach to localization themselves.

Previous to any RoboCup competition a complete description of the RoboCup SPL field is provided for competing teams [8]. As can be seen in Figure 1, this is a wholely static environment with a large amount of concise visual information including field lines, goal posts, penalty spots and the centre circle. During soccer matches some dynamic elements do present themselves such other robots,

* This work was supported by Science Foundation Ireland, PI Grant no. 07/IN.1/I1838 and UREKA Site Grant no. 09/UR/I1524.

referees and audience members beyond the pitch boundaries. A typical localization system which may used in a RoboCup environment is described by Röfer et al. in [9].

Some of the challenges associated with using white field markings in localization include the development of robust vision algorithms for line detection and line fitting, as well as techniques for identifying circular field markings and penalty spot type features. Given the small field of view provided by the camera on board the Nao robot, oftentimes only small segments of white field markings are visible in any given image. This ultimately leads to the difficult problem of uniquely identifying a certain field marking from a partial view. An immediate issue with common solutions to this problem is the resultant spaghetti code - large blocks of nested conditional statements tailored specifically to the RoboCup pitch layout are required. The solution we present removes the need to design ad-hoc line detection and identification systems by functioning generically on any combination of simple geometric features.

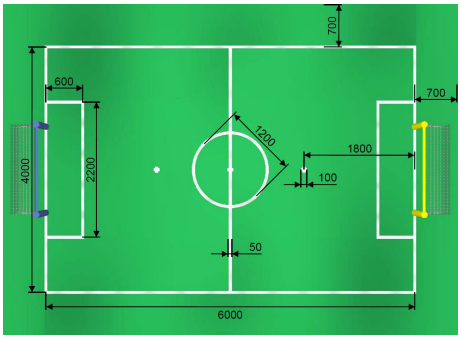


Fig. 1. Standard Platform League Pitch Description [8].

2 Background

Our method for utilising field markings is a specialised extension of matching detected 'line points' to a predefined map of line segments. This algorithm, in the context of laser scanner based analysis, was first used by Cox in 1991 [1]. Cox treated the problem as least-squares linear regression with an analytical solution, successfully demonstrating that this technique was both accurate and practical. Lauer et al. described a similar algorithm in 2006 for robots in the RoboCup Midsize League using points detected on white field markings instead of laser points [5]. They also introduced a new error function with a numerical gradient descent based solution. In 2010, Rath implemented an adaptation of Lauer et al's algorithm for use on the Nao robot in the SPL [7]. This adaptation emulated most of the methods used by Lauer et al. but due to the hardware constraints of the Nao some small changes were introduced.

2.1 Previous Work

In our previous paper we described a modified version of Cox's original algorithm dubbed the Modified Cox Algorithm (MCA) [12]. We presented a number of modifications to the original algorithm including (i) the use of a Voronoi diagram to reduce computational load in determining the nearest field marking to a given point; (ii) an extension of the basic algorithm to include all types of field markings (line segments, circles and single points); (iii) distance based outlier detection; and, (iv) weighted least-squares cost minimization. We also detailed the integration of the MCA with an Unscented Kalman Filter. For the sake of completeness the MCA which we described previously is listed in the following section in its entirety, excluding the Kalman Filter integration. Our new improved Kalman Filter integration is described in detail in Section 3.1.

2.2 Modified Cox Algorithm

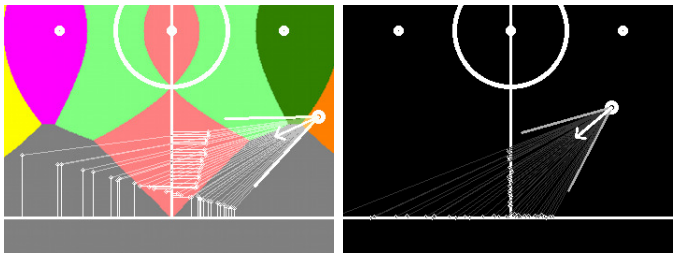
This description is a combination of the original method described by both Cox and Rath and the modifications we presented in our previous paper [1, 7, 12]. Given a set of detected points on white field markings in an image, the basic process of the Modified Cox Algorithm involves 3 main steps;

2.2.1 Point transformation from image to world coordinates Transformation from image coordinates to world coordinates is achieved using typical back projection associated with the extrinsic and intrinsic camera parameters (see for example Figure 2a). In this regard the camera location is based on the geometry of the robot and the current joint sensor readings.

2.2.2 Selecting the closest field marking to each point Before this step is carried out a Voronoi diagram for all white field markings must be precalculated, see Figure 2 (a). This can be done once off during the start up of the robot. Then, the closest white field marking to any point projected into world coordinate space can be determined in $\mathrm{O}(1)$ time.

2.2.3 Finding a correction for the current pose In this final step, a correction to the current robot pose, described by $l_t = (x, y, \theta)^\top$ where x and y describe the estimate of the robot's global position and θ describes the estimated orientation of the robot, is calculated. We wish to calculate $b = (\Delta x, \Delta y, \Delta\theta)^\top$ such that a new estimate, $l'_t = l_t + b$, gives a pose which better matches observed points to white field markings.

The aim of Cox's original algorithm is to minimise the squared distances associated with points on line segments (line points) to their nearest line segment. To achieve this, the problem is linearised into a least-squares linear regression problem and each line segment is treated as an infinite line with orthogonal unit vector $u_i = (u_{ix}, u_{iy})^\top$ and offset r_i such that $u_i \cdot z_i = r_i$ holds for all arbitrary line points z_i on the line.



(a) Error Calculation (b) Distance Minimised

Fig. 2. Example of the Modified Cox Algorithm.

Let the ith transformed line point be $z_i = (z_{ix}, z_{iy})^\top$ and the current position of the robot be $c = (l_{tx}, l_{ty})^\top$. The transformation of each line point z_i can be described as:

$$t(b)(z_i) = \begin{pmatrix} \cos \Delta\theta & -\sin \Delta\theta \\ \sin \Delta\theta & \cos \Delta\theta \end{pmatrix} (z_i - c) + c + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{1}$$

Cox suggests that the correction angle $\Delta\theta$ should be sufficiently small such that we can approximate the transformation to:

$$t(b)(z_i) \approx \begin{pmatrix} 1 & -\Delta\theta \\ \Delta\theta & 1 \end{pmatrix} (z_i - c) + c + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{2}$$

Next, the squared distance of each line point z_i can be found as:

$$d_i^2 = (t(b)(z_i)^\top u_i - r_i)^2 \approx ((x_{i1}, x_{i2}, x_{i3})b - y_i)^2 \tag{3}$$

Where:

$$\begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \end{pmatrix} = \left(\begin{matrix} u_{ix} & u_{iy} \end{matrix} \middle| \, u_i^\top \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (z_i - c) \right) \tag{4}$$

$$y_i = r_i - z_{ix} u_{ix} - z_{iy} u_{iy} \tag{5}$$

Defining the absolute fixed world position of a penalty spot as $s_i = (s_{ix}, s_{iy})^\top$ and $q_i = (q_{ix}, q_{iy})^\top$ as a transformed penalty spot point we have:

$$\begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \\ x_{i+1,1} & x_{i+1,2} & x_{i+1,3} \end{pmatrix} = \left(\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \middle| \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (q_i - c) \right) \tag{6}$$

$$\begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} = \begin{pmatrix} s_{ix} - q_{ix} \\ s_{iy} - q_{iy} \end{pmatrix} \tag{7}$$

Assuming a centre of $(0, 0)$ for the centre circle feature, given a radius of h and a transformed centre circle point $v_i = (v_{ix}, v_{iy})^\top$ we have:

$$\begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \end{pmatrix} = \left(\begin{matrix} \frac{v_{ix}}{||v_i||_2} & \frac{v_{iy}}{||v_i||_2} \end{matrix} \middle| \, \frac{v_i^\top}{||v_i||_2} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (v_i - c) \right) \tag{8}$$

$$y_i = h_i - v_{ix} u_{ix} - v_{iy} u_{iy} \tag{9}$$

Since we expect smaller errors in the location of points close to the robot, we define a diagonal weighting matrix, W, with diagonal elements given as:

$$W_{ii} = \frac{1}{\alpha_i^2 + \eta}, \tag{10}$$

and where α is the relative distance to the point from the robot and η is some small offset value.

Now we can calculate the weighted sum of squared distances for all points z_i, q_i and v_i:

$$E_W(b) = \sum_{i=1}^{n} W_{ii}((x_{i1}, x_{i2}, x_{i3})b - y_i)^2 = (Xb - Y)^\top W(Xb - Y) \tag{11}$$

Where:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{12}$$

The correction $\hat{b}$ that minimises $E_W(b)$ can then be (approximately) solved by:

$$\hat{b} = (X^\top WX + \zeta I)^{-1} X^\top WY \tag{13}$$

where ζ is a small positive constant to avoid singularity occuring in (13). Finally a new pose is given by:

$$l'_t = l_t + \hat{b} \tag{14}$$

The results of this process can be seen in Figure 2b.

3 Extensions & Modifications

The main short comings of our previous implementation were localization performance and computational performance. We showed that the system was at most on par with three other localization systems evaluated and not significantly better by any measure [12]. Computational performance was also undesirably low for use on the Nao system, considering that localization is just one of many components that ideally must run at 30Hz during a soccer game. In this section we describe the extensions and modifications we made to the original system that aided in overcoming these issues.

3.1 Unscented Kalman Filter Integration

Initial tests with the algorithm described in [12] did not show good performance, and the CPU requirements were excessive. For these reasons, the algorithm we used in the RoboCup 2011 competition (including the technical challenge) underwent a major redesign. This redesign reduced the number of sigma points

used and the number of iterations in the local search, and also added some extra computations to give a better estimate of the errors in the predictions from Cox's algorithm. The algorithm used is described below.

1. **Generation of a set of sigma points:**
   The sigma points were calculated in a similar way to those of [3, 11]. We have $n_x = 3$ state variables in the filter. In our case we selected the primary weight as $w_0 = \frac{1}{\sqrt{2}}$, and the remaining weights are selected as:

$$w_\ell = \sqrt{\frac{1 - w_0^2}{2n_x}}; \quad \ell = 1, ...2n_x. \tag{15}$$

   We denote the current state estimate by $x_{k|k-1}$ and the ℓth column of the square root of the covariance matrix as $(p_{k|k-1})_{(\ell)}$. The sigma points, $\mathcal{X}_\ell$, are then calculated as:

$$\mathcal{X}_\ell = \begin{cases} x_{k|k-1} : \ell = 0 \\ x_{k|k-1} + s_\sigma (p_{k|k-1})_{(\ell)} : \ell = 1...n_x \\ x_{k|k-1} - s_\sigma (p_{k|k-1})_{(\ell)} : \ell = n_x + 1...n_x \end{cases} \tag{16}$$

   where the scale factor for the sigma points is given by $s_\sigma = \sqrt{\frac{n_x}{1-w_0^2}}$.
2. **Use each sigma point as an initial value for the Modified Cox Algorithm:**
   The Modified Cox Algorithm (MCA) described in Section 2.2 can be thought of as a local search for a good fit between the observed points and the known field markings. We use the sigma points in (16) as initial values for the MCA and perform a single step correction for each. The result of this update is a new estimate, $\hat{\mathcal{X}}_\ell$, of a possible robot location, together with the weighted sum of residual errors, $J_\ell = \sum_i W_{ii} e_i^2(\hat{\mathcal{X}}_\ell)$, and a covariance of the estimate, $var\hat{\mathcal{X}}_\ell$, computed as described below in (21).
3. **Discard points with excessive error, or other problems:**
   The result of a single correction to one of the sigma points may not give a good fit to the data. This may be because a single iteration is insufficient to be close to convergence. It may also occur due to being close to a local minimum. We therefore select a threshold, $\bar{J}$, and ignore any results where the residual errors are too large, namely, $J_\ell > \bar{J}$. Other checks used are to test if the MCA has enough valid points to process and that the resultant estimated robot position is not too far off the pitch. Note that in the following equations, discarding a point is equivalent to setting the corresponding weight to zero.
4. **Adjust sigma point weights according to the residual errors and renormalize:**
   Given the initial set of weights, w_ℓ, we first adjust these according to the residual errors:

$$\tilde{w}_\ell := w_\ell / J_\ell. \tag{17}$$

We then renormalize the adjusted weights as follows:

$$\hat{w}_\ell := \tilde{w}_\ell / \sum_k \tilde{w}_k \tag{18}$$

5. **Check for sufficient valid estimates:**
   For the algorithm to generate a valid measurement, we require that a sufficient number of sigma points had valid MCA results (in our case, this was set to 3), otherwise the entire set is ignored.
6. **Recombine Sigma Points into a single combined estimate:**

$$X = \sum_\ell \hat{w}_\ell \hat{\mathcal{X}}_\ell \tag{19}$$

$$var(X) = \sum_\ell \hat{w}_\ell \left[(\hat{\mathcal{X}}_\ell - X)(\hat{\mathcal{X}}_\ell - X)^T + var\hat{\mathcal{X}}_\ell \right] \tag{20}$$

7. **Use the combined estimate as a linear Covariance Intersection KF Measurement Update:**
   The combined estimate, (19), is a linear function (in fact the identity) times the localization state variables. It is therefore straightforward to use linear Kalman Filter covariance intersection updates (e.g. [10, 4]) to perform a measurement update. This update includes standard features such as outlier detection and kidnapped robot detection.

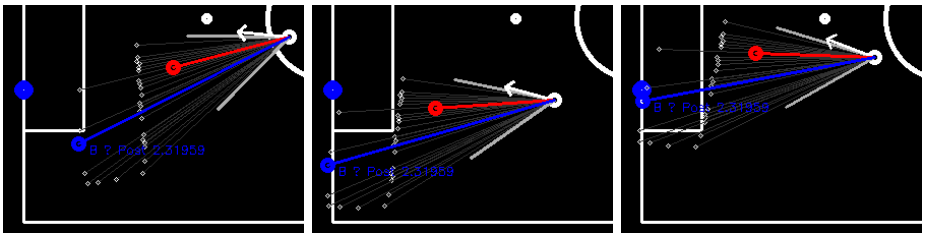
3.2 Modified Cox Algorithm

A number of additions and alterations have been made to the MCA. These have been divided into two categories: (i) Algorithmic Enhancements, concerned with the information used and produced by the algorithm; and, (ii) Computational Optimisations, concerned with the way in which the algorithm processes input and produces an output.

3.2.1 Algorithmic Enhancements

Super-Weighted Posts - The goal posts are one of the key sets of landmarks on the SPL pitch. They are generally quite easy to identify and are the most significant cue for localization. Having previously ignored any goal post information in the MCA optimisation, we now include goal posts in the form of super-weighted single points, akin to penalty spot type features.

Being color coded, the posts on the SPL field are inherently less ambiguous than points on white field markings. In line with this, post points are weighted 30 times higher than white field marking points in the optimisation but not in the error calculation in order to avoid inconsistencies in error values between frames including posts and those not. When both posts of a single color goal are visible a robot's position and orientation can be accurately determined using



(a) Before MCA (b) After Without Posts (c) After With Posts

Fig. 3. Example of Post Point Inclusion in MCA.

simple triangulation. In this scenario, both post points are matched to the known fixed positions of the posts in the world model for the MCA algorithm. More often than not however, only one post is visible due to optical occlusion by other robots. When this occurs the single visible post is ambiguous. Given that the MCA is a local search to begin with, an ambiguous post is matched to the fixed position of the nearest post in the world model when the perceived post point is transformed into world coordinates.

An example is shown in Figure 3 of the kind of effect this feature has on the algorithm. In situations where the robot is mislocalized badly or a poor measurement to a post is perceived, there is concern for ambiguous posts being matched incorrectly and furthering corruption of the localization estimate. Typically in this scenario the system is more reliant on the Kalman Filter discussed in Section 3.1, which would normally have a high uncertainty when mislocalized and as such will have a large spread of sigma points and be more likely to throw out badly matched MCA updates.

Calculation of MCA Variance - In order to improve the integration of MCA updates in the Kalman Filter the variance of the calculated correction is recorded, for each sigma point. This is especially useful in scenarios where only points on co-linear field markings are detected. The variance of the translation and rotation is derived from the diagonal from the inverted component P of the final correction calculation listed as (13) in Section 2.2.3:

$$P = (X^\top W X + \zeta I)^{-1} \sigma_n^2 \tag{21}$$

The noise variance measurement is taken as $\sigma_n^2 = 0.01$.

3.2.2 Computational Optimisations

Point Sampling - Given that the MCA is highly dependent on matrix methods, a large amount of attention was given to these methods when optimising the technique for computational performance. Notably, for two matrices of dimensions $m \times p$ and $p \times n$ the run time complexity of standard matrix multiplication

is $\mathrm{O}(mnp)$. The only input of varying size to the MCA is the number of points on white field markings. Owing to the complexity of the multiplication method it was observed that the execution time scaled badly. As a result, the number of white field marking points used in the MCA is capped at 30. If there are more points than this detected in an image 30 points are selected at random for use in the optimisation. The same 30 points are used for all 7 sigma points.

Matrix Operation Optimisation - As mentioned in Section 2.2.3, a diagonal matrix W is used to weight the optimisation. Rather than using naïve standard matrix multiplication in the evaluation of equations which involve the diagonal matrix W a more efficient multiplication method is used. This alternative method skips the summation step of standard matrix multiplication, which when used on a diagonal matrix would result in many redundant sums of zero. For two matrices (one diagonal) of dimensions $m \times p$ and $p \times n$ this alternative method runs in $\mathrm{O}(mn)$ time as opposed to $\mathrm{O}(mnp)$.

Another concern when optimising matrix operations is matrix chain multiplication. When presented with a sequence of matrices we wish to determine the most efficient way to multiply these matrices together, given that the parenthesization of matrix multiplication changes only the number of operations and not the result. Evaluating (13) in Section 2.2.3 in standard left to right order involves roughly 6000 operations when 30 points are used. Solving the matrix chain multiplication problem on this equation results in a decrease of the number of operations to approximately 3900. This represents a performance increase of 35%. The order of multiplication was determined using a dynamic programming approach [2]. The resulting parenthesization of (13) is:

$$\hat{b} = ((X^\top(WX)) + \zeta I)^{-1}(X^\top(WY)) \tag{22}$$

One final optimisation involves the precalculation of point weights. Rather than recalculating the weights for each sigma point, the weights of all points are calculated before hand as they do not change between sigma points. Overall a performance increase of 60% was recorded per sigma point with the MCA, derived from an execution time reduction of 1ms to 0.4ms per point.

In the table below execution time differences are shown for each of the described computational optimisations. The time given represents the increase in execution time of the entire MCA if that feature isn't implemented in the computation.

Optimisation	**Time Saved (ms)**
Reduction from 13 to 7 sigma points	6.1
Maximum of 30 line points	3.1
Diagonal Multiplication	1.2
Chain Multiplication	1.0
Weight Precalculation	0.1

3.3 White Field Marking Point Filtering

One concern highlighted by Rath was the issue of false positive points detected on white features on the RoboCup pitch [7]. This was less of a concern for Lauer et al. due to the fact that the robots used in the Midsize League do not contain white features. The Nao robot, used in the SPL, is almost entirely white and as a result false positive white field marking points are prevalent when color reliant detection algorithms are used.

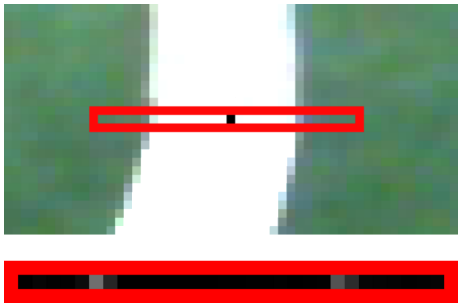


Fig. 4. Result of a Horizontal First Difference Operation Around a Point.

In order to combat this we have implemented a filter based on edge detection principles that reliably removes false positives. When a set of points are detected in an image a local edge check is performed either horizontally or vertically depending on which scan orientation a point was detected with. As can be seen in Figure 4 there should exist a very obvious single maximum in a simple gradient estimate on both sides of a point. Inspecting both sides of a point for this maximum is a trivial process. A threshold can then also be applied to accept points with a large gradient.

Calculation of the gradient estimate is carried out using Intel MMX SIMD instructions in order to maintain good computational performance. The estimated gradient value for a given pixel (i, j) in the raw YUV image F is calculated using only the Y component as $|F_y(i, j) - F_y(i + 1, j)|$ for horizontally detected points and similarly as $|F_y(i, j) - F_y(i, j + 1)|$ for vertically detected points. These values can be calculated for 7 pixels in one go by using the technique described below.

3.3.1 MMX Gradient Estimation A two pass technique is required to calculate the full first difference for a set of pixels using this method. The 64-bit MMX register is divided up into 8 unsigned integers each with a range of [0 - 255]. Given that the Y component of each pixel also has a range of [0 - 255] we are unable to use the MMX registers in a signed format. Thus, using the horizontal direction as an example, we calculate $F_y(i, j) - F_y(i + 1, j)$ and $F_y(i + 1, j) - F_y(i, j)$ separately using saturated arithmetic (clipping at 0 and 255) and sum the result together.

When processing points on white field markings however, the expected gradient direction is known and as a result we can skip first difference calculation in one direction.

Fig. 5. Sample Y Component Values at Point Surrounding

1. Populate an MMX register with the Y component values of the pixels around the edge of the detected point (at most 8). The pixels shown in Figure 5 would translate to an MMX register as:

140	142	144	160	235	240	240	240

2. Depending on the expected gradient direction, based either on horizontal or vertical orientation and whether inspecting the edge going from green to white or vice-versa, shift a copy of the register left or right by 8 bits. In the example case, we shift right:

0	140	142	144	160	235	240	240

3. Using saturated MMX arithmetic, subtract the shifted register from the original, yielding the first difference for 7 pixels (we ignore the end value):

	140	142	144	160	235	240	240	240
-	0	140	142	144	160	235	240	240
=	-	2	2	16	75	5	0	0

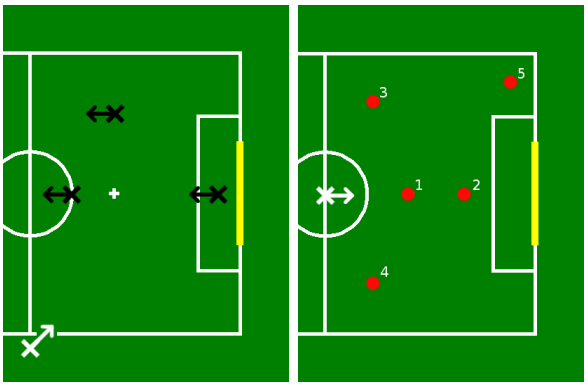
The result of this calculation can then be inspected for a single maximum (with leniency) above a preset threshold. The MMX implementation described above was measured to be 2.6 times faster than a standard C++ implementation of the same process. Figure 6 shows an example of the results of this process. Occasionally good points are filtered out, particularly those that appear on features which are small in the image. Loss of such distant points is only a minor inconvenience due to the fact that they are weighted quite lightly in the MCA optimisation.

4 Tests & Results

The performance of the newly modified MCA based localization system was evaluated in 3 different tests along with 3 other localization systems. The three tests were:

(a) Color Classified Image (b) Points Before Filtering (c) Points After Filtering

Fig. 6. Example of Detected Points With / Without Edge Filtering.



(a) Target Ready Positions (b) Ball Placement Points

Fig. 7. Test Scenario Set-Ups. Initial Position and Orientation Drawn in White.

1. Ready Positions. As shown in Figure 7 (a). For this test, the robot is placed in the initial position and commanded to walk to each of the positions marked in black as soon as a correct estimate of the initial position is reported by the localization system. This test was repeated 3 times for each position and once on each side of the field, bringing the total number of runs to 18.
2. Open Goal Shots. As shown in Figure 7 (b). In this test the robot is placed in the initial position and commanded to take shots on an open goal, again only when an accurate estimate of the initial position is reported. The ball is first positioned at the location labeled '1'. After each attempted shot on the goal the ball is placed at the next position in the labeled sequence. This test was carried out once on each side of the field giving a total of 10 attempted shots.
3. Goalless Open Goal Shots. This test is identical to the previous test except both color goals are removed from the field before the robot begins. Initially all goal posts are on the field to allow the robot acquire an estimate of its initial position. Once the robot acquires the correct initial position all goal

posts are removed. As before this test was carried out once on each side of the field giving a total of 10 attempted shots.

The localization systems compared are as follows:

- **Kalman Filter + MCA:** The new localization system presented in this paper including MCA updates, post updates, line and corner updates.
- **Particle Filter + White Field Marking Points:** An experimental implementation developed for testing purposes. A basic Particle Filter using 100 particles using post updates, line updates, corner updates and a new experimental update that uses certain functions of the MCA. This update involves steps 1 and 2 in Section 2.2 followed by evaluation of the error magnitude of the projected points. The optimisation in step 3 is not carried out. Instead, the error before the optimisation is used to weight particles.
- **Particle Filter + Lines:** A basic Particle Filter using 100 particles, post updates, line updates and corner updates.
- **Kalman Filter + Lines:** The same Unscented Kalman Filter used with the MCA in this paper using only post updates, line updates and corner updates.

4.1 Localization Performance

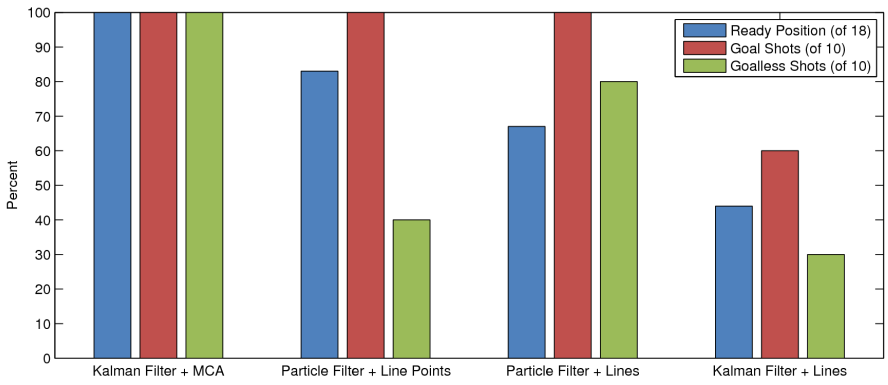


Fig. 8. Localization System Test Results.

Figure 8 shows the results of all 3 tests. For the Ready Position test, the percentage of times the robot successfully reached the target position is given. For each Ready Position test, the robot's final resting position and orientation was manually recorded. The robot was deemed unsuccessful if it either left the field or halted with a position which was greater than 30cm from the target location

or an orientation which was greater than 15 degrees from the target orientation. In some cases the robot would never halt completely and oscillated around some final location, however there was no penalty for this behaviour if the position was correct.

In the goal shot tests the success criteria was a lot simpler. The percentage represents the number of times the robot successfully kicked the ball towards the goal with an accurate position estimate. Kick line up issues were accounted for by monitoring the robot's position estimate throughout the tests. Successful goals where the robot's estimate was incorrect were not counted.

4.2 Computational Performance

The execution time of all four localization systems was monitored throughout the 3 tests. It should be noted that a small amount of debugging functions were enabled during the testing of all 4 systems and as a result execution times without any debugging functions may in fact be slightly lower. The execution time of the PF + Points algorithm is significantly longer than the 3 other algorithms tested because although it lacks the optimisation in the full MCA, it requires the repeated projection of all line points for each of the 100 particles.

Algorithm	KF + MCA	PF + Points	PF + Lines	KF + Lines
Avg Time (ms)	6	10	4	3
Max Time (ms)	7.7	20	5	3

5 Conclusion

In this paper we have described a number of modifications and extensions to our original MCA implementation. Many aspects of the system have been looked at to address some of the issues and future work discussed in our previous paper. The results include: (i) revised and more computationally efficient Kalman Filter integration; (ii) usage of post information in the MCA to improve localization performance; (iii) MCA variance calculation to improve Kalman Filter interaction; (iv) point sampling for improved computational performance; (v) matrix method and chain multiplication optimisation for computational performance; and, (vi) efficient field marking point filtering for reduced false positive points.

Computational performance was previously a significant issue with the system but is clearly no longer a concern. The localization performance achieved with the enhancements listed above is superior to the previous version of the MCA and all other localization systems tested. As a testament to the system's high performance it was successfully demonstrated as the core part of RoboEireann's Open Challenge demonstration at RoboCup 2011, "Localisation without goal posts". The demonstration was voted 1st place out of 20 other presentations.

5.1 Future Work

Extensions to the proposed algorithm are needed to better deal with cases where the robot gets lost. This may occur due to a range of circumstances such as:

(i) when the robot falls over; (ii) when the robot's locomotion is restricted, particularly when it is attempting to perform a rapid turn, but is blocked from being able to execute the turn; (iii) when the robot is moved by the game referees (for example in relation to either a local game stuck; or if the robot is penalized).

At present, the overall algorithm is slow to respond to these 'kidnapped robot' type situations, and further algorithm development is needed in this area. Multiple model Kalman filtering [6]; the ability to perform a larger number of MCA updates (with a larger number of sigma points) and the ability to perform multiple iterations of the local search may all improve the re-localization performance significantly. However, at least on the current hardware and with the current versions of the algorithm, speed improvements are crucial to permit these features.

References

1. I. Cox. Blanche - An experiment in guidance and navigation of an autonomous robot vehicle. *Robotics and Automation, IEEE Transactions on*, 7(2):193 –204, April 1991.
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Section 15.2: Matrix-chain multiplication. In *Introduction to Algorithms, Second Edition*, pages 331–339. MIT Press and McGraw-Hill, 2001.
3. S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401 – 422, Mar. 2004.
4. S. Julier and J. Uhlmann. Using covariance intersection for SLAM. *Robotics and Autonomous Systems*, 55(1):3–20, 2007.
5. M. Lauer, S. Lange, and M. Riedmiller. Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization. In *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020, pages 142–153. Springer Berlin / Heidelberg, 2006.
6. M. Quinlan and R. Middleton. Multiple Model Kalman Filters: A Localization Technique for RoboCup Soccer. In *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 276–287. Springer Berlin, 2010.
7. C. Rath. Self-localization of a biped robot in the RoboCup domain, Master's Thesis. Institute for Software Technology, Graz University of Technology, 2010.
8. RoboCup Technical Committee. RoboCup Standard Platform League (Nao) Rule Book. `http://www.tzi.de/spl/pub/Website/Downloads/Rules2011.pdf`.
9. T. Röfer, T. Laue, and D. Thomas. Particle-filter-based self-localization using landmarks and directed lines. In *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 608–615. Springer, 2006.
10. S. Stüdli. Kalman Filtering approach for Localisation in RobotSoccer, Master's Thesis. Hamilton Institute, NUI Maynooth & Institute for Control, Swiss Federal Institute of Technology (ETH), Zurich, 2011.
11. R. Van Der Merwe and E. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *IEEE International Conference on Acoustics Speech and Signal Processing*, volume 6, pages 3461–3464. Citeseer, 2001.
12. T. Whelan, S. Stüdli, J. McDonald, and R. H. Middleton. Line Point Registration: A Technique For Enhancing Robot Localization in a Soccer Environment. In *Proc. RoboCup Symposium*, July 2011.