Line Point Registration: A Technique for Enhancing Robot Localization in a Soccer Environment

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, stuedlis@ee.ethz.ch,
johnmcd@cs.nuim.ie, richard.middleton@nuim.ie

Abstract. The Standard Platform League (SPL) provides an environment that is essentially static; with the exception of other robots and the audience, the area in which a robot is expected to localise itself is quite favourable. However, a large number of the predefined landmarks in the given world model can be perceived as ambiguous in many scenarios, with the prime example being field line markings. In this paper a technique is presented that implicitly disambiguates these detected field line objects in order to use them for localization purposes.

1 Introduction

The environment in which a robot in the SPL must localise in is very well defined; pitch dimensions, field markings and landmark objects are fully described previous to any competition [11]. As a result we are reliably presented with a constrained environment. While other robots and audience members may introduce some dynamics into the environment, in general it is static and rich in predefined information.

1.1 Basics of Localization

In order to localise within this fully defined world model, we need only identify a small number of the known fixed features before we can employ a number of traditional methods for maintaining robot localization. A typical localization system used in humanoid robotic soccer would make use of either a Kalman filter[8] or a Particle filter[12,4]. Either of these filters use odometry and visual information to provide an accurate estimate of a robot's global position. Speaking in even simpler terms, only two known field landmarks are needed for triangulation of one's position. The goal posts are a perfect example of two such landmarks[1]. However, it is not always possible to maintain two goal posts in a robot's field of vision. Either of the filters listed previously are designed to account for this and aided by reasonable odometry measures can give a moderately accurate estimate of a robot's position in such scenarios. However, given

T. Röfer et al. (Eds.): RoboCup 2011, LNCS 7416, pp. 258–269, 2012.

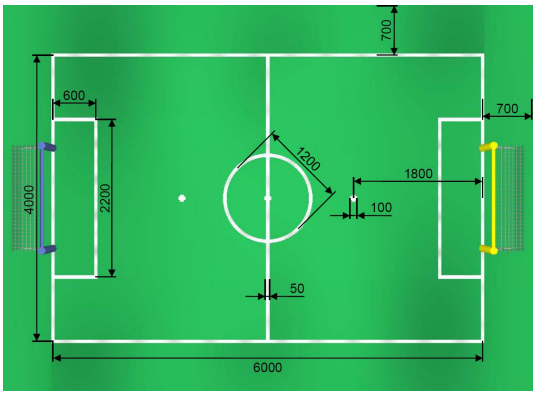


Fig. 1. Given Standard Platform League pitch description in rule book [11]

the relatively fast pace of a robotic soccer game, it is desirable to be accurately localised at all times.

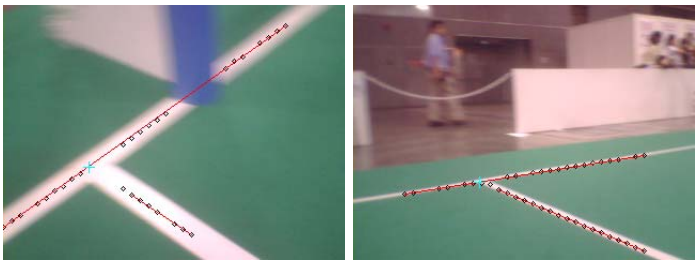
The remaining known landmarks that exist on the field are the field line markings. As can be seen in Figure 1 these include a large number of corners (line intersections) and a centre circle. Considering the number of these landmarks one would expect to be able to triangulate on a position in the majority of view points. However, this is where our two main issues arise.

1.2 Computational Load

The first issue, albeit slightly more accessible than the second, is associated with computational performance. A typical method for the detection of lines in an image is to perform an operation such as the Hough Transform and extract lines from Hough Space[3]. Unfortunately this operation is quite expensive and cannot be realistically used in realtime on the current SPL hardware. A more basic approach, given the constrained environment, is to scan a given image for green-white-green transitions in order to detect what may be points on a field line. Then, clustering can be performed on these detected line points that should yield line segments within the image. This approach does yield good results, as shown in Figure 2, but still requires a significant amount of computation. For example, this technique makes up 50% of the processing time of the vision component of our SPL system.

1.3 Ambiguity

The second issue associated with field line markings is the ambiguity of corners and curves. While many heuristics can be applied to attempt to uniquely identify a detected corner, there are scenarios where it is impossible unless information about a robot's currently estimated position is used. This can lead to a nasty feedback loop - if the estimate is incorrect, a field line landmark may be



(a) Close lines (0.5 - 1.5m) (b) Far lines (1.5 - 3.0m)

Fig. 2. Results of simple line point and line segment detection on close and far away field lines

incorrectly identified which would further the corruption of the current position estimation. A curve, the centre circle to be specific, is also ambiguous. From as many as 4 different view points any of these field line landmarks may appear the same. This presents us with a significant ambiguity problem.

The work presented in this paper provides a solution which removes the need for the initial stage of line detection and explicit disambiguation. However, it does make use of the information provided by processing an image containing field lines. In effect, by combining Cox's algorithm with a Kalman Filter based system, the aim is to rapidly and continually use the algorithm to keep close to the best fit. In other words, disambiguation on our proposed system is a consequence of continually maintaining an estimate of the range of possible robot locations. From this range of possible locations, Cox's algorithm is used to enhance the accuracy of estimation, and thereby narrow the range of possibilities.

2 Background

The process of matching detected points to a predefined map of line segments was first detailed by Cox in 1991 [2]. A robotic system was described that used a laser scanner to detect points on solid objects in the surrounding. These detected points were then subsequently matched in a local search fashion to an *a priori* map of the environment, producing an estimated pose. 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 field lines instead of laser points [7]. They introduced an alternative error function for minimisation that was more robust to outliers when compared to the squared error function. Noisy distance estimates were cited as the motivation behind this. They also introduced gradient descent as an alternative optimisation method in place of the least-squares linear regression used by Cox, due to the form of their new error function. This evoked a requirement to calculate the gradient for translation and orientation at each position before descending towards a minimum. *RPROP* was then used to solve

the minimisation task [10]. Also in line with Cox's original suggestion, they used a simplified application of the Kalman filter in the form of a stochastic weighted averaging approach for smoothing of the estimated pose.

In 2010, Rath implemented an adaptation of the algorithm for use on a Nao robot in the SPL [9]. This adaptation included most of the methods used by Lauer et al. but due to the hardware constraints of the Nao some small changes were introduced. The most significant modification was the pre-calculation of gradients with respect to the translation for a 5×5cm grid. The motivation behind this was to reduce computational load, the result being that only the gradient for the orientation would need to be calculated at run time.

In 2009, Inam [4] presented a related method where a Particle filter could be used in conjunction with an image database. The matching technique described by Inam, unlike the previous examples, functions in image space rather than world space and uses correlation in place of an optimisation. It therefore uses substantially more computational power than required for the algorithm presented here.

2.1 Cox's Algorithm

Given a set of detected line points in an image, the basic process of Cox's algorithm involves 3 main steps, as outlined by both Cox and Rath; This description is taken mainly from [9].

2.1.1 Line 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. In this regard the camera location based on the geometry of the robot and the current joint sensor readings.

2.1.2 Selecting the Closest Line for Each Point

This is carried out for each transformed line point by calculating the shortest Euclidean distance from each line segment in the world model to the point, and selecting the one with the shortest distance.

2.1.3 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 line points to field line markings.

The aim of Cox's Algorithm is to minimise the squared distances associated with each point to its closest line segment that we calculated in 2.1.2. 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 points z_i on the line.

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:

$$x_{i1} = u_{ix} \tag{4}$$

$$x_{i2} = u_{iy} \tag{5}$$

$$x_{i3} = u_i^\top \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (z_i - c) \tag{6}$$

$$y_i = r_i - z_{ix} u_{ix} - z_{iy} u_{iy} \tag{7}$$

Now we can calculate the sum of squared distances for all points z_i:

$$E(b) = \sum_i ((x_{i1}, x_{i2}, x_{i3}))b - y_i)^2 = (Xb - Y)^\top (Xb - Y) \tag{8}$$

Where:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{9}$$

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

$$\hat{b} = (X^\top X)^{-1} X^\top Y \tag{10}$$

And finally a new pose is given by:

$$l_t' = l_t + b \tag{11}$$

3 Extensions to the Algorithm

Lauer et al. described a method for matching detected field line points to a global map of curves that overcame the two main shortcomings of Cox's original method. (i) The restriction to straight lines, and (ii) the lack of robustness against distance measure outliers. In 2010, Rath described an adaptation to the method introduced by Lauer et al. that functioned on a smaller, less powerful system - Aldebaran's Nao. However, this adaptation was not without a cost; it required the pre-calculation of the gradients for translations. This resulted in what was effectively a drastically down-sampled representation of all possible translation gradients, from an almost continuous (limited only by computation precision) 6×4m area, to a discrete 120×80 size area, with each component of the area measuring 5×5cm. The hardware used by Lauer et al. contained a 1GHz CPU, which was more than adequate for evaluating gradients for all three components of the optimisation online. However implementing this full online gradient computation on Nao hardware is impractical, which justifies Rath's pre-calculated gradient look up table.

Here we present a set of adaptations to Cox's original algorithm that solve both the non-straight line and distance measurement error limitations without the need for gradient calculation or significant offline computation or quantization, while still maintaining reasonable performance for online execution onboard an Aldebaran Nao. Given the relatively small size of the SPL pitch, it is desirable to maintain as high a precision as possible when estimating a robot's pose.

3.1 Voronoi Diagram

In his paper [2] Cox suggests the use of a Voronoi diagram to simplify the determination of the closest line to a given point, but did not implement one himself. Lauer et al. do not mention any preprocessing done to speed up the determination of closest lines and in the approach taken by Rath, the pre-calculated gradient look up table essentially removes the need for a Voronoi diagram.

On-the-fly calculation of the closest field marking feature taking a naive brute force approach is $\mathrm{O}(m)$, where m is the number of features. Pre-calculation of the closest line at each point in the form of a Voronoi diagram provides a look-up of $\mathrm{O}(1)$, a much more desirable complexity. According to the description of the field markings described in the SPL rule book [11], we pre-calculate a Voronoi diagram of precision 1×1cm for the entire pitch, as shown in Figure 3.

Then, the closest line to the transformed line points as described in Section 2.1.1 can be found in a look-up table.

3.2 Inclusion of All Field Markings

The mathematics described by Cox only supports straight line segments as part of the optimisation process. While straight line segments do make up 88.5% of the white field marking area on the SPL pitch there is a lot of useful information in the penalty spots and the centre circle, as shown in Figure 1.

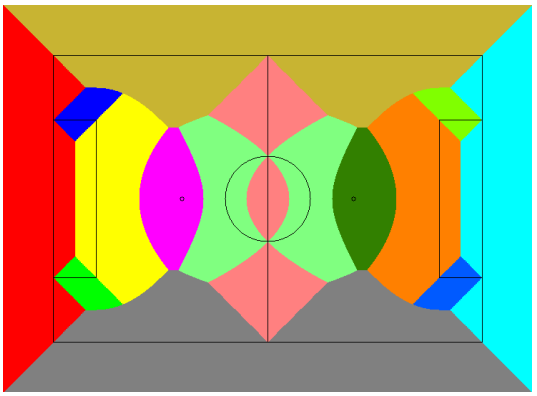


Fig. 3. Voronoi diagram for field markings

3.2.1 Inclusion of Points

In order to include penalty spots in Cox's original algorithm we must devise a method to include single points. Points can be included by modifying the entries made for Equations 4 through 7 in Section 2.1.3 in matrices X and Y:

$$\begin{pmatrix} x_{i,1} & x_{i,2} & x_{i,3} \\ x_{i+1,1} & x_{i+1,2} & x_{i+1,3} \end{pmatrix} = \left(\begin{array}{cc|c} 1 & 0 & \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (c - z_i) \\ 0 & 1 & \end{array} \right) \tag{12}$$

$$\begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} = \begin{pmatrix} s_{ix} - z_{ix} \\ s_{iy} - z_{iy} \end{pmatrix} \tag{13}$$

Where $s_i = (s_{ix}, s_{iy})^\top$ describes the coordinates of the point, or penalty spot, in world coordinates.

3.2.2 Inclusion of Circles

In order to include circular curves or the centre circle in particular, similar modifications are required for Equations 4 through 7 in Section 2.1.3. Namely, to include line points on a circle with radius R:

$$r_i = R \qquad u_i = \frac{1}{\sqrt{z_{ix}^2 + z_{iy}^2}} \begin{pmatrix} z_{ix} \\ z_{iy} \end{pmatrix} \tag{14}$$

3.3 Accounting for Distance Outliers

Lauer et al. and Rath used a normalising distance error function for minimising, said to be more robust to distance outliers than the squared error function of Cox's approach. In order to introduce some tolerance for such errors we added a

weighting to each detected line point based on its relative distance. The weight for line point i is defined as:

$$W_i = \frac{1}{d^2 + \eta} \tag{15}$$

Where W is a diagonal matrix, d is the relative distance to the point and η is some small offset value.

Another concern for Cox's approach is the issue of singularity occurring during the pseudo inversion, particularly when no line intersections are visible. This is accounted for by adding some small value ζ in the form of a diagonal matrix before carrying out the inversion. The final modified version of Equation 10 is then:

$$\hat{b} = (X^\top W X + \zeta I)^{-1} X^\top W Y \tag{16}$$

4 Kalman Filter Integration

To stabilise the pose estimate given by the modified version of Cox's algorithm the output is smoothed with a specialised Kalman Filter. The unscented transform [13,5] is used, so that the uncertainty of the robot's position estimate is taken into account for the initial pose estimate given by the modified algorithm. Then from the output of the modified algorithm we compute an estimate of the position of the robot and an associated covariance matrix. Afterwards a normal Kalman filtering approach can be applied.

The unscented transform uses a set of points called sigma-points χ_i and associated weights w_i to represent a probability distribution; in our case the pose estimation. The sigma-points and their weights have to fulfill the following characteristics:

1. The weights must sum to unity.
   $\Rightarrow \sum_{i=0}^{n} w_i = 1$

2. The weighted sum must be the mean value of the distribution.
   $\Rightarrow \sum_{i=0}^{n} w_i \chi_i = \mathbf{x}_{k|k-1}$

3. The weighted square error sum must be equal to the covariance of the distribution.
   $\Rightarrow \sum_{i=0}^{n} w_i (\chi_i - \mathbf{x}_{k|k-1})(\chi_i - \mathbf{x}_{k|k-1})^T = P_{k|k-1}$.

There are various selections of sigma-points which fulfill these characteristics. For this implementation a symmetric sigma-point set similar to that of Julier and Uhlmann [5] is chosen with:

$$\chi_0 = \mathbf{x}_{k|k-1}$$
$$w_0 = \frac{1}{n_x + 1}$$

$$\chi_i = \mathbf{x}_{k|k-1} + \frac{1}{\gamma}\sqrt{n_x + 1}(p_{k|k-1})_{(i)}$$
$$w_i = \gamma\frac{1}{2(1+\gamma)(n_x+1)}$$

$$\chi_{i+n_x} = \mathbf{x}_{k|k-1} - \frac{1}{\gamma}\sqrt{n_x + 1}(p_{k|k-1})_{(i)}$$
$$w_{i+n_x} = \gamma\frac{1}{2(1+\gamma)(n_x+1)}$$

$$\chi_{i+2n_x} = \mathbf{x}_{k|k-1} + \gamma\sqrt{n_x + 1}(p_{k|k-1})_{(i)}$$
$$w_{i+2n_x} = \frac{1}{2(1+\gamma)(n_x+1)}$$

$$\chi_{i+3n_x} = \mathbf{x}_{k|k-1} - \gamma\sqrt{n_x + 1}(p_{k|k-1})_{(i)}$$
$$w_{i+3n_x} = \frac{1}{2(1+\gamma)(n_x+1)}.$$

The modified version of Cox's algorithm is then applied at each sigma-point, delivering a representation of the distribution of the measurement data. This means that the algorithm is executed thirteen times; once for each sigma-point. In addition to an estimated position of the robot Υ_i, the modified algorithm also computes a goodness of fit between the detected line points and their closest lines ξ_i. These measures are used to modify the weights of the sigma-points, according to:

$$\hat{w}_i = \frac{w_i}{\xi_i^2}. \tag{17}$$

As the weights must sum up to one, they are normalized after modification.

Then these new sigma-points are used to calculate the measurement:

$$\mathbf{y}_k = \sum_{i=0}^{n} \hat{w}_i \Upsilon_i \tag{18}$$

and its square root of the covariance matrix, where HT denotes the Householder Triangularization:

$$R_k = \mathrm{HT}\left(\left[\sqrt{\hat{w}_0}(\Upsilon_0 - \mathbf{y}_k), \ldots, \sqrt{\hat{w}_n}(\Upsilon_n - \mathbf{y}_k), r_0\right]\right). \tag{19}$$

These two values are then used for a measurement update of a Kalman filtering approach. In our case a Covariance Intersection as described by Julier and Uhlmann [6] is used.

5 Results

5.1 Localization System Tests

To fully test the system, the performance of various localization algorithms are compared in moderately realistic game scenarios. The localization algorithms compared are as follows:

- **UKF:** The Unscented Kalman Filter algorithm used by the RoboEireann team in the 2010 Competition.
- **Particle Filter:** A basic particle filter, with 100 particles, implemented for testing purposes.
- **CI (line detection):** A Covariance Intersection alternate to the UKF based on standard vision measurement data, that is, using post measurements, corner points, T points, centre circle etc.
- **CI (Cox's algorithm):** A Covariance Intersection algorithm, using goal post measurements and integrated with Cox's algorithm as described in Section 4.

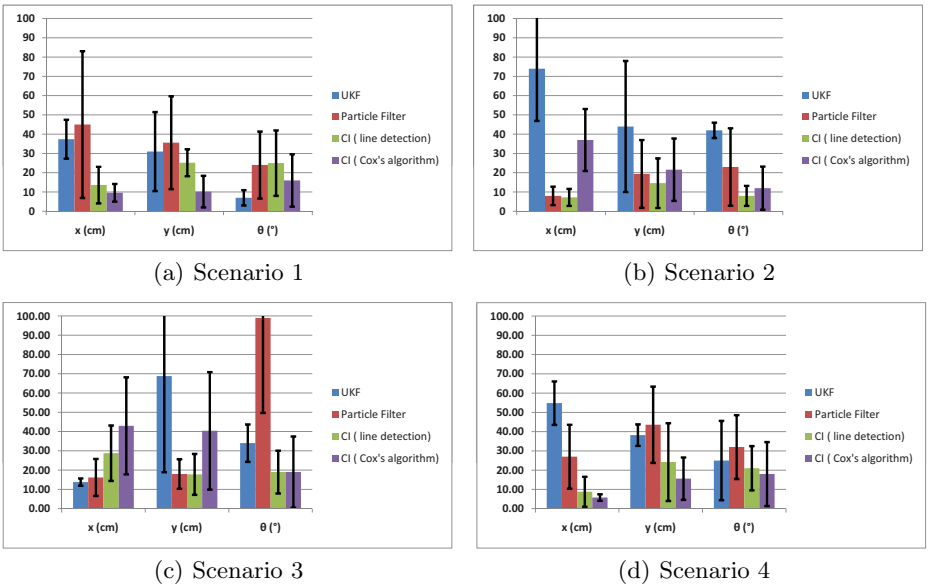
For each of the localization systems, tests were conducted in four different scenarios. In each of these, there is a start location and orientation and a destination location and orientation. Once the robot reaches its final location, the error in this location was manually recorded. (The coordinate frame is a standard Cartesian system with origin at the centre of the pitch, x-axis directly towards the centre of the yellow goal). The scenarios are described in the Table 1.

Table 1. Scenarios used for localization tests

	Starting Position			Destination		
Scenario	$x(m)$	$y(m)$	$\theta(^\circ)$	$x(m)$	$y(m)$	$\theta(^\circ)$
1	0.0	-2.0	135	-1.2	0.0	180
2	-2.4	-1.1	0	-0.7	0.7	0
3	0.0	2.0	-135	-2.8	0.0	0
4	0.0	0.0	0.0	-1.0	1.0	-135

The first scenario denotes a situation where an attacker must move from the sideline, to a point facing the goal at the penalty spot. The second scenario describes a situation where a robot in the back corner of the pitch must move up to the opposite side of the field, just before the half way line. The third scenario describes a goal keeper moving to position and the fourth is where a robot on the center circle, must turn around, and move to an attacking position slightly to one side of the pitch.

To minimize variability due to changing lighting conditions, the vision system camera calibration was adjusted between each set of trials. The results presented in Figure 4 give data from 5 repetitions of a test in each scenario.



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 4. Localization error results for tests in four different scenarios. For each scenario 5 tests are conducted, and statistics on the errors are computed. The error bars in the figures indicate standard error distance, and the vertical scale is either cm or degrees.

The implementation chosen for the inclusion of the modified version of Cox's algorithm in the Kalman filter took an iterative approach. Given the size of the pitch and the prevalence of local minima it was decided that each execution of the algorithm would be capped in translation and rotation, reducing the potential for overstepping a local minimum valley in the error minimisation space. This approach turned out to be quite computationally intense, taking between 8 and 30ms per frame, due to the necessity for multiple iterations at each of the 13 sigma-points.

6 Conclusion

In this paper we have considered a number of extensions to a line point registration based algorithm, Cox's algorithm, for utilizing field markings in localization. A number of modifications to the basic algorithm have been examined. These include: (i) use of a Voronoi diagram to reduce computational load in determining the nearest field mark; (ii) extension of the basic algorithm to include all types

of field markings (lines, circles and points); (iii) distance based outlier detection; (iv) weighted least square cost minimization; and, (v) integration with unscented Kalman Filter based localization.

The performance of the algorithm achieved so far is similar to or better than any of the other algorithms tested. It also shows promise for further developments, such as modification to learn robot pose from the data. There is also potential in the future to use the algorithm in a non-iterative fashion and more cleverly chose integration points with the Kalman filter, solving any costly computation issues. It would also be interesting to see if the algorithm can be integrated with particle filter approaches, though generally, the algorithm itself is too expensive to be run once for each particle.

References

1. Cañas, J.M., Puig, D., Perdices, E., González, T.: Visual goal detection for the RoboCup Standard Platform League. In: X Workshop on Physical Agents, WAF 2009, Cáceres, Spain, pp. 121–128 (2009)
2. Cox, I.: Blanche - An experiment in guidance and navigation of an autonomous robot vehicle. IEEE Transactions on Robotics and Automation 7(2), 193–204 (1991)
3. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM 15, 11–15 (1972)
4. Inam, W.: Particle filter based self-localization using visual landmarks and image database. In: Proceedings of the 8th IEEE International Conference on Computational Intelligence in Robotics and Automation, CIRA 2009, pp. 246–251. IEEE Press, Piscataway (2009)
5. Julier, S., Uhlmann, J.: Unscented filtering and nonlinear estimation. Proceedings of the IEEE 92(3), 401–422 (2004)
6. Julier, S., Uhlmann, J.: Using covariance intersection for SLAM. Robotics and Autonomous Systems 55(1), 3–20 (2007)
7. Lauer, M., Lange, S., Riedmiller, M.: Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 142–153. Springer, Heidelberg (2006)
8. Middleton, R.H., Freeston, M., McNeill, L.: An application of the extended Kalman filter to robot soccer localisation and world modelling. In: Proceedings of the IFAC Symposium on Mechatronic Systems (2004)
9. Rath, C.: Self-localization of a biped robot in the RoboCup domain. Master's Thesis. Institute for Software Technology, Graz University of Technology (2010)
10. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: IEEE International Conference on Neural Networks, vol. 1, pp. 586–591 (1993)
11. RoboCup Technical Committee: RoboCup Standard Platform League (Nao) Rule Book, `http://www.tzi.de/spl/pub/Website/Downloads/Rules2010.pdf`
12. Röfer, T., Laue, T., Thomas, D.: Particle-Filter-Based Self-localization Using Landmarks and Directed Lines. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 608–615. Springer, Heidelberg (2006)
13. Van Der Merwe, R., Wan, E.: The square-root unscented Kalman filter for state and parameter-estimation. In: IEEE International Conference on Acoustics Speech and Signal Processing, vol. 6, pp. 3461–3464. Citeseer (2001)