# NightOwl: Self-Localisation by Matching Edges

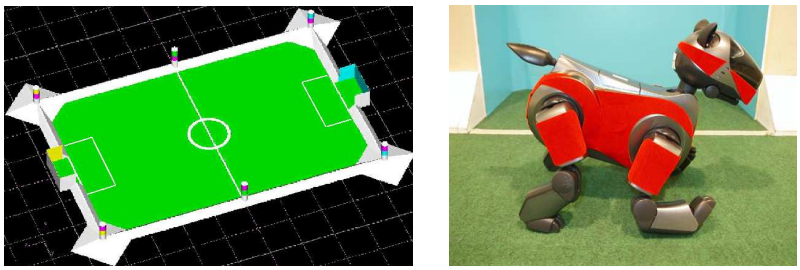
Raymond Sheh[1,2] and Bernhard Hengst[2,3]

[1] Department of Computing, Curtin University of Technology
Perth, WA, 6102, Australia
[2] School of Computer Science and Engineering, University of New South Wales
Sydney, NSW, 2052, Australia
[3] National ICT Australia, University of New South Wales
Sydney, NSW, 2052, Australia
shehrk@cs.curtin.edu.au, bernhardh@cse.unsw.edu.au

UNSW-CSE-TR-0406
February 2004

**Abstract**

A mobile robot must know where it is to act appropriately. An algorithm that allows a robot to accurately localise itself locally using a vision sensor and a map of its environment is described in this paper. The basic idea of this algorithm, called NightOwl, is to match the projected camera image with a map of the environment in a local area in order to find the most likely position and orientation of the camera platform.



(a) (b)

Figure 1: The legged league soccer field with six localisation beacons (a). The Sony ERS-210A robot used in the Four-Legged Soccer League (b).

# 1 Introduction

One of the challenges facing the effective application of mobile robots is that they localise themselves efficiently and accurately in their environment. It is desirable that they do this by using naturally occurring environmental features rather than relying on purpose built navigation aids.

The annual international RoboCup competition provides an excellent test-bed to highlight these needs and foster research. In particular, as of 2003, the Sony Legged League used 6 brightly coloured marker poles or beacons [8] (see figure 1) to assist visual self-localisation of both the robot teams on the soccer field for all competitions. Reliable and accurate localisation is essential in allowing the robot to play soccer effectively. However, beacons are not always visible when playing the game, for example, near the edges of the field and near the goal area [5]. Also, the league intends to progressively remove beacons as RoboCup moves towards its long term goal [7, 6]. Both these reasons provide the motivation behind the research addressed in this paper, namely, to face the new challenge of localising the robots based on naturally occurring features.

The contribution of this paper is an algorithm for matching features in the camera image with those expected in the environment for the purpose of quickly and accurately localising the camera platform. We have called this algorithm NightOwl as it metaphorically allows the robot to "see without beacons", in the dark, so to speak. In RoboCup, some of these features consist of the borders and markings of the RoboCup field, observed through the robot's camera[1].

NightOwl relies on the principle that a unique visual image will precisely determine the location and orientation of a robot camera platform and hence the robot. Theoretically the position of a camera can be determined from only six appropriately matched pixels [2]. In practice, matching unique images to

[1]NightOwl was successfully implemented in both the 2003 RoboCup legged league soccer competition and the localisation challenge without beacons. The University of NSW/National ICT Australia team was placed 1st in the world championships in Padova, Italy for both of these events.

a 3D model of the environment is computationally expensive and there is no guarantee that an image is unique.

During the RoboCup competition, computational power is limited to the 385MHz microprocessor on the Sony Aibo ERS-210A robot. In order to make NightOwl tractable on this platform, we reduce the computational needs of the matching problem by concentrating only on field lines and borders. While these can produce many aliased images, the ambiguity is reduced by considering only the portion of the environment local to the robot. Restricting the search locally in this way further reduces the computational requirements of the algorithm.

In the rest of this paper we will describe the NightOwl localisation method that comprises:

- Efficiently finding points of interest. In this case, the boundary pixels from the soccer field border and markings in the robot camera image.
- Projecting these image pixels onto the soccer field plane using geometry based on the pan and tilt of the camera.
- Matching the projected pixels against a local model of the field with a pre-generated matching lookup table.

NightOwl assumes that an image feature is largely disambiguated in the robot's neighbourhood. A natural extension is to use NightOwl as a front end to global localisation techniques. We conclude with a discussion including this promising research direction.

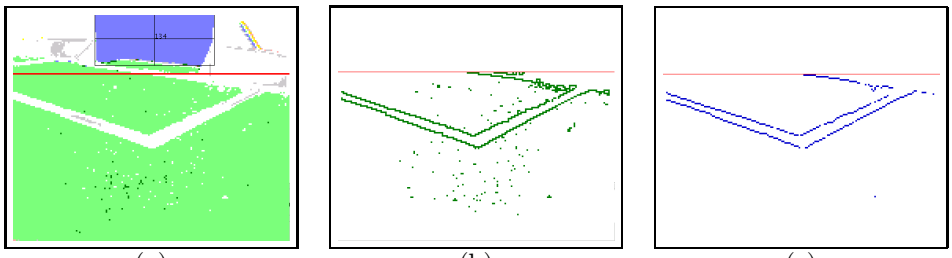
# 2 Boundary Edge Detection

The points of interest correspond to green-white boundaries in the images obtained from the robot's CMOS colour camera. These camera images have a resolution of 176x144 pixels and a colour depth of 16 bits per pixel [10].

For every camera frame provided by the robot's operating system, the existing software infrastructure produces a colour segmented pseudo-image, an example of which appears in figure 2(a), called the *CPlane* [3]. The CPlane is formed by looking up the three-dimensional colour attribute of each pixel in a lookup table containing the colour label corresponding to that colour. This lookup table is machine learnt using, for example, decision trees and based on training samples that are classified manually by colour.

White-green boundaries in the CPlane are found by a morphological edge detector. These boundaries are defined by white pixels with more than a predefined number of eight-nearest-neighbour green pixels. It was found that defining the boundary in this way gave better results than having the boundary lie within the green regions of the image as the matching performed later in the NightOwl system is biased towards the white markings on the field.

This morphological edge detector, whilst effective at finding white-green boundaries, appears to be badly affected by image noise as shown in figure 2(b). The traditional solution of smoothing the input image in order to reduce



(a) (b) (c)

Figure 2: (a) an example CPlane, (b) after processing by a basic edge detector, (c) NightOwl noise reduced edge detector output.

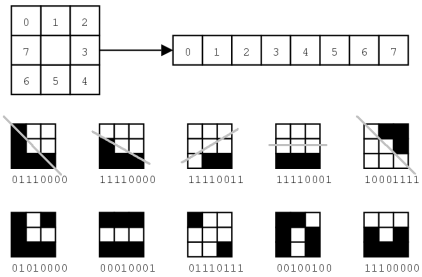



Figure 3: Examples of patterns around a central pixel that support field lines (top row) and that don't (bottom row), with corresponding byte representations.

noise is not an option as the CPlane is effectively a binary input image and the smoothing operator is likely to be too slow to satisfy the time constraints of the application. Testing for several green pixels around a candidate white edge pixel is found to reduce "pepper" noise. However, noise that consisted of several pixels of one colour embedded in another cannot be effectively eliminated in this fashion.

In order to reduce the effect of image noise, the pattern of edge pixels surrounding edge candidates are classified into those that are likely to support well-defined boundaries and those that are likely to be noise, and are kept or discarded accordingly. Well-defined boundaries are defined as those that lie on roughly straight lines that are separated by more than one pixel and which stay constant for at least three pixels as shown in figure 3. The result of the application of this filtering process appears in figure 2(c).

The neighbouring pattern for a given pixel may be classified efficiently by "unrolling" the neighbouring 8 pixels into a byte as in figure 3, in which green pixels are represented as "0" and other colours as "1". This byte is used as an index into a 256-element pre-generated lookup table containing flags for

patterns that are likely to be noise or patterns that are likely to support desired boundaries. Whilst the classifier will often misclassify sharp corner points as noise and discard them, it was found that the omission of these points had a minimal effect on the operation of the NightOwl system.

Further computational savings are obtained by incorporating this filtering stage into the edge detector. The "unrolling" process replaces the neighbouring pixel inspection and summation step and adds negligible overhead as it comprises bit operations. The lookup table replaces the threshold step and may be performed with minimal overhead as the 256 element lookup table has a high likelihood of being loaded into the processor's memory cache. The resulting highly specific local-feature-level edge filter effectively minimises the effect of image noise, classification noise and errors with virtually no noticeable increase in computational or storage overheads[2].

# 3 Location Matching of Boundary Points

Localisation information is extracted from the observed boundary points by matching them to a model of the robot's environment. Using information from the robot's joint angle encoders, the image points are projected onto the field. Accuracy limitations in the angle readings limit the effectiveness of this process to points within about 1.5m of the robot when stationary and 75cm when moving. These projected boundary points, in combination with the robot's last-known position, are matched to a pregenerated field model. The robot's belief position is then shifted in the local region in order to maximise the quality of this match which is an indicator of the probability that the robot is in the location and orientation that produced that match. Thus, the most likely actual location and orientation of the robot in the region around its prior belief position may be determined.

## 3.1 Matching Grid Points

For efficiency, an unnormalised measure of probability is used to position the robot in a particular location and orientation given a set of observed, projected boundary points. This "relative" probability, $p_{rel}^{robot}(pos)$ at a given location and orientation $pos = (x^{robot}, y^{robot}, \theta^{robot})$, is calculated by determining the global co-ordinates $b_{global,i}(pos) = (x_{global,i}^{point}(pos), y_{global,i}^{point}(pos))$ of each observed boundary point $i$ given its projected, robot-relative co-ordinates $b_{local,i} = (x_{local,i}^{point}, y_{local,i}^{point})$ and the robot's position $pos$ as in equation 1.

$$\begin{aligned} x_{global,i}^{point}(pos) &= x_{local,i}^{point}\cos(\theta^{robot}) - y_{local,i}^{point}\sin(\theta^{robot}) + x^{robot} \\ y_{global,i}^{point}(pos) &= x_{local,i}^{point}\sin(\theta^{robot}) + y_{local,i}^{point}\cos(\theta^{robot}) + y^{robot} \end{aligned} \quad (1)$$

[2]Whilst local angle is not used in NightOwl, this edge detector may also determine the local angle of the edge feature with no additional computational expense.

Equation 2 is then evaluated for each projected point where the matching function $M$ is defined in equation 3. This equation finds $p_i^{point}(pos)$, an unnormalised measure of the probability of the point $i$ at global co-ordinates $b_{global,i}(pos)$ corresponding to a feature point $f$, $\{f \in F\}$ where $F$ is the set of all known feature points in the environment. For the soccer environment, the set $F$ contains all the points in global co-ordinates that lie on any location on the field through which a field marking or border passes. It is generated from information about the field geometry.

$$p_i^{point}(pos) = M\left(\min_{f \in F} |f - b_{global,i}(pos)|\right) \tag{2}$$

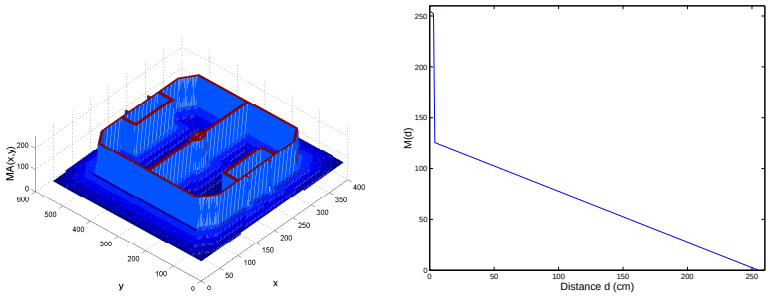
$$M(d) = \begin{cases} MA_{MAX} = 252 & \\ MA_{MAX} - d & , d < 3 \\ \frac{1}{2}\left(MA_{MAX} - d\right) & , d \geq 3 \end{cases} \tag{3}$$

These point probabilities are summed as in equation 4 to yield the total relative probability $p_{rel}^{robot}(pos)$ for a given position $pos$. The process implicitly deals with occlusion, as long as enough boundary points remain visible. This is because the set of positions that can explain a set of observed boundary points at a particular position in the presence of occlusion is always a superset of those positions that can explain the boundary points that would be observed were the occlusion not present.

$$p_{rel}^{robot}(pos) = \sum_I p_i^{point}(pos) \tag{4}$$

The evaluation of $p_i^{point}(pos)$ in equation 2 is performed efficiently via a precomputed two-dimensional lookup table, known as the *matching function array* $MA$ of dimensions $X, Y$ shown in figure 4(a). This lookup table contains one value for every square centimetre of the field, plus a border of 75cm to allow for mis-localisation. Whilst this requires the point positions to be quantised to the nearest centimetre, this accuracy is sufficient for the purpose of RoboCup.

An innovative feature of the matching function $M$ is that it is not directly proportional to distance. Instead, a significantly higher weighting is given to regions very close to known features whilst the gradient is maintained, albeit at a significantly lower weighting, for areas beyond this distance, as shown in figure 4(b) and defined in equation 3. This matching function reduces the effect of noise and resulting outliers in the observed points by favouring positions where large numbers of observed points correspond accurately to model features. In contrast, if a directly proportional matching function were used, an outlying pixel will displace a correctly matched cluster of pixels. A quadratic matching function, with an effect similar to the least mean square error method, would make the problem worse as the outlier would be given even greater weight.




(a) (b)

Figure 4: The matching function array used in the RoboCup competition $MA(x, y)$. (a) the matching score $M(d)$ as a function of distance $d$ from the nearest feature in centimetres, (b) the profile of the matching function in (a) as a vertical cross section perpendicular to an isolated field line or boundary.

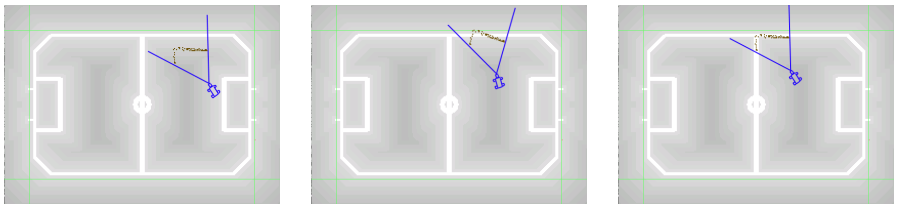
## 3.2 Localising by Maximising the Matching Score

A natural approach to find the most likely position of the robot is to utilise some form of local derivative based gradient ascent over the "probability map", starting from the robot's current position. However, such an approach behaves poorly in this application, where low level local maxima occur frequently, as demonstrated in figure 5. In this example, given a starting position at (a), gradient ascent would terminate at the incorrect local maxima at (b) without searching for a better match or higher maxima in the probability function $p_{rel}^{robot}$ at (c). This problem is especially apparent around goal boxes and other feature-rich areas of the field.

Instead, a grid-based discrete hillclimbing process is performed by sampling $p_{rel}^{robot}(pos)$ numerous times in order to find a satisfactory local maxima. Translational and rotational sampling is carried out around the prior belief position with a spacing of $r = (r_x, r_y, r_\theta)$ and for a number of samples per dimension $m = (m_x, m_y, m_\theta)$. Combinations of steps in each dimension are also taken, resulting in a sampling "cube". The position $pos_{max,1}$ which yields the highest value of $p_{rel}^{robot}(pos_{max,1})$ over all the points thus sampled satisfies equation 5 for $k = 1$ and is used as the new belief position. This sampling and maximisation process may repeat for a number of iterations $K$, each iteration potentially having different values of $m_k$ and $r_k$ for a variety of effects and yielding a most likely position $pos_{max,k}$ satisfying equation 5.

$$p_{rel}^{robot}\left(pos_{max,k} \begin{array}{c} x_{max,k}, \\ y_{max,k}, \\ \theta_{max,k} \end{array}\right) = \max_{\substack{n_x \in [-m_{k,x}, m_{k,x}] \\ n_y \in [-m_{k,y}, m_{k,y}] \\ n_\theta \in [-m_{k,\theta}, m_{k,\theta}]}} \left\{ p_{rel}^{robot} \begin{pmatrix} x_{max,k-1} + r_{k,x} n_x, \\ y_{max,k-1} + r_{k,y} n_y, \\ \theta_{max,k-1} + r_{k,\theta} n_\theta \end{pmatrix} \right\} \tag{5}$$

Figure 5 presents an example of this maximisation process. The initial belief



(a) (b) (c)

Figure 5: Three examples showing the matching of projected image field edge pixels to field edges defined by the matching function array. (a) a poor match given the last estimated position of the robot, (b) a better match found by varying the position and orientation of the robot in a local neighbourhood, (c) the best match found in the neighbourhood. The robot position and orientation are updated to fit this match.

position at (a) results in a sampling grid covering both positions (b) and (c). Whilst standard gradient ascent would yield position (b), the extra sampling performed is able to find a better match at (c) despite the local gradient from (a) not leading to (c). For a given "cube" sampling grid, multiple samples may satisfy equation 5 in which case one is selected at random. Whilst this leaves open the possibility of an incorrect match, this was found to be a rare occurrence.

The values of the parameters $K$, $r_k$ and $m_k$ are crucial in balancing computational tractability, accuracy and the risk of incorrect matches. In the presence of processing time constraints, $K$ and $m_{k=1,2,\ldots,K}$ become severely limited whilst, generally, $r_{k=1,2,\ldots,K}$ has no bearing on processing time. During the 2003 RoboCup Competition, NightOwl was run with $K = 3$ and $m_{1,2,3} = \{5, 3, 3\}$ for all dimensions. With these parameters and the number of boundary points sampled down to 20 points, NightOwl was able to run alongside the other in-game processes at full frame-rate whilst still contributing useful localisation information. In contrast, during the 2003 RoboCup localisation challenge, these parameters were set to $K = 3$ and $m_{1,2,3} = \{19, 5, 5\}$ for all dimensions. With these parameters and the number of boundary points sampled down to 100 points, NightOwl would take around one second to process one frame. During the localisation challenge, NightOwl was called on-demand and not on every frame, hence this processing time was acceptable.

The selection of suitable values for the sample spacing $r_k$ is governed by two conflicting factors. With a wide spacing, a large area may be sampled with fewer points, allowing larger errors to be corrected. However, if the spatial frequency of maxima in the probability function $p_{rel}^{robot}$ becomes greater than the spatial sampling frequency, the desired maxima may be missed altogether. There are two general ways in which the $r_k$ and $m_k$ arrays may be configured.

The first configuration involved similar values between successive iterations for $r_k$ and $m_k$. This enables the NightOwl to correct localisation errors that

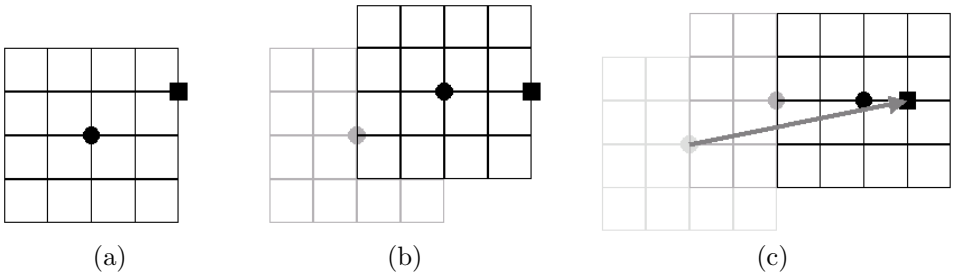



Figure 6: Two dimensional simplified example of an iterative grid-based gradient ascent configured with similarly spaced grids. At each iteration, filled circle marks the start point, filled square marks the point of greatest probability. Note that the third dimension of rotation is not shown in this diagram.

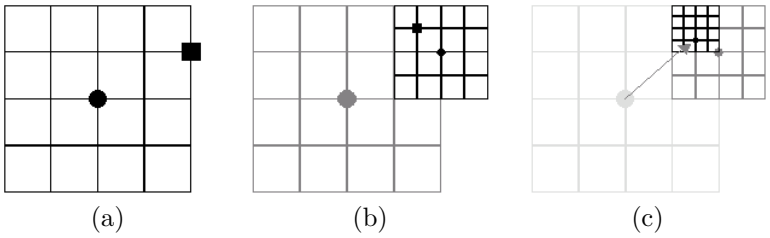
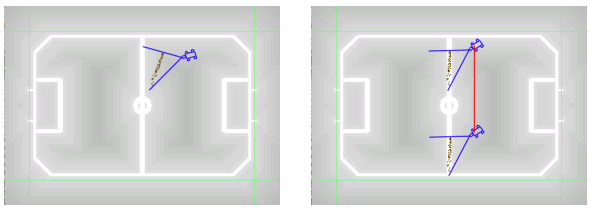



Figure 7: Two dimensional example of an iterative grid-based gradient ascent configured with reduced spacing grids. At each iteration, filled circle marks the start point, filled square marks the point of greatest probability. Note that the third dimension of rotation is not shown in this diagram.

involve movements beyond the range of the first set of sampling, somewhat akin to an extended version of traditional gradient ascent. This is illustrated in figure 6. The aim of this configuration is to allow relatively large corrections to be made to the robot's localisation with relatively few computed points. Accuracy may be limited, especially if no samples happen to fall near the desired local maximum due to too large a spacing.

The second configuration is weighted towards correcting situations where the error in prior localisation is relatively small and aims to maximise the accuracy of the final "snap-in". This configuration is characterised by successive iterations where the spacing between samples $r_k$ reduces such that the range of positions sampled at a given iteration covers an area comparable to the spacing between samples in the prior iteration, as illustrated in figure 7. The aim of this configuration is to allow for as accurate a "snap-in" as possible without having to cover the entire search area with a fine grid.

In any limited-area search, there remains a chance that the resulting data only provides a "snap-in" that is unique in two dimensions out of the three.



(a) (b)

Figure 8: Example of an observation that does not provide information in one dimension. The observation and starting position (a) may indicate any position along the line between the two positions in (b).

Figure 8 provides an example whereby, on observing a single line, NightOwl finds a maxima at the nearest consistent line feature. It is unable to localise along the line. Note that the absence of an observation of the centre circle in this example cannot rule out the possibility that the robot is around the centre circle due to the possibility of total occlusion of the centre circle.

A modified Kalman filter is used by the UNSW/NICTA team in order to track the robot's position. Broadly speaking, this filter combines information from new observations with its existing belief of its current position plus a motion model, in order to form an updated belief of its current position using a form of moving average. It is possible for NightOwl to pass partial information to this filter, in the event that information is lacking in one dimension, as is the case with a single line match such as in figure 8. In the general case, the lack of information along a given dimension, which may not be axis parallel, corresponds to a "ridge" in the distribution of $p_{rel}^{robot}(pos)$ which may be detected by taking many samples of $p_{rel}^{robot}(pos)$ in the region around the final match. However, with the exception of the short corner-boards and centre circle, which are rarely observed in isolation, all lines on the field are axis-parallel. Thus, a much simpler system based on an approximate derivative in the two translational dimensions, taken by only sampling the four translational samples around the final sample, may be utilised. Rotation about the robot centre is not used.

The magnitude of a pseudo-derivative of $p_{rel}^{robot}(pos)$ at the final "snap-in" position $pos_{si}$ is taken, as in equation 6. Normalisation by the number of observed boundary points $I$ is performed to make this value comparable between observations. Dimensions with low derivatives indicate a match that is unlikely to be unique in the local area in that dimension and thus the match in that dimension is not used in the Kalman filter. For efficiency, the distance in each translational dimension $\{\Delta x, \Delta y\}$ over which the falloff gradient is computed is the same as the final sampling grid $\{r_{K,x}, r_{K,y}\}$.

$$\left|\frac{\Delta p^{robot}}{\Delta x}\right|_{si} = \frac{1}{I\Delta x}\left(p_{rel}^{robot}(x_{si}, y_{si}, \theta_{si}) - \max\left\{\begin{array}{l} p_{rel}^{robot}(x_{si} - \Delta x, y_{si}, \theta_{si}), \\ p_{rel}^{robot}(x_{si} + \Delta x, y_{si}, \theta_{si}) \end{array}\right\}\right)$$
$$\left|\frac{\Delta p^{robot}}{\Delta y}\right|_{si} = \frac{1}{I\Delta y}\left(p_{rel}^{robot}(x_{si}, y_{si}, \theta_{si}) - \max\left\{\begin{array}{l} p_{rel}^{robot}(x_{si}, y_{si} - \Delta y, \theta_{si}), \\ p_{rel}^{robot}(x_{si}, y_{si} + \Delta y, \theta_{si}) \end{array}\right\}\right) \quad (6)$$

# 4 NightOwl Performance in a RoboCup

During the main competition, the primary goal of NightOwl was to assist in localisation near the left and right sides of the field. NightOwl enabled the UNSW/NICTA robots to approach and dribble balls very close to the sidewall and with a significantly smaller safety margin, thus providing a significant tactical advantage. However, NightOwl's performance was less reliable during the rough and tumble of play in areas of the field where many features appeared close together, such as near the centre circle or goals. In these locations, an incorrect starting position may cause NightOwl to report an incorrect position correction. Therefore, during the main competition, NightOwl was restricted to operation near the sidewalls where it yielded the greatest benefit. This issue may be partially addressed by the multi-hypothesis extensions discussed later.

NightOwl's major application was in the second technical challenge for which the six localisation beacons were removed from the field and the robot required to visit five predetermined co-ordinates on the field. NightOwl was used in situations where the goals, which were used for global localisation, proved to be too inaccurate. NightOwl assisted in achieving a perfect score on one of the locations placed inside one of the goal boxes and contributed to rUNSWift winning this technical challenge.

# 5 Related Work and Future Work

NightOwl alone cannot be used to localise the robot globally when the image feature is aliased in its environment. One solution is to enhance and extend the feature with the objective to make it unique. For example, stitching together multiple views was demonstrated to reduce the aliasing at the expense of matching a greater number of pixels and increased potential inaccuracy due to movement errors. The process of searching for a strong local maxima may also be worth further inspection, with approaches such as Simulated Annealing [4] being possible candidates for more efficient optimisation techniques.

Another approach is to combine NightOwl with an occupancy grid or particle filter based localisation method such as Monte Carlo localisation [11, 9]. The benefit of NightOwl is that it should be able to help the robot to quickly localise globally. It provides a more focused sensor model with a highly concentrate probability mass. For use with Monte Carlo localisation, the grid-based gradient ascent stage may be skipped and the function $p_{pos}^{robot}$ evaluated for the position

of each particle. This value may be used as part of the probability update for that particle. In this approach, $p_{pos}^{robot}$ must be normalised appropriately.

To demonstrate the more general applicability of this RoboCup research, the NightOwl localisation method is being used in an experiment involving the Curtin University Smart House laboratory. The aim of this laboratory is to investigate technologies that can be used to assist the elderly and disabled in living independently in their own homes and consists of a variety of cameras and other sensors plus an Aibo robot. Overhead cameras provide updates to a Monte Carlo localisation filter whilst NightOwl, implemented on an Aibo, supplements the robot's localisation in situations where there are ambiguous observations from the overhead cameras or where the robot is occluded from the overhead cameras. Preliminary results have been promising with NightOwl being able to maintain the robot's localisation even in the presence of large errors in odometry, lengthy periods of occlusion and other moving objects in the scene.

The more complex feature based characteristics of NightOwl would make it an ideal candidate as a front end to a feature based multi-hypothesis localisation method, such as for example in [1]. NightOwl can provide such a localiser with a discrete, limited number of hypotheses, when for example, a corner is seen.

# 6 Conclusion

We have presented an algorithm for extracting robot self-localisation information from a vision sensor. The process entails extracting edges from an observed colour segmented image, projecting them onto the field plane using the current position and matching them against a model of the field. The output is the likelihood of the robot's position. We use a grid-based gradient ascent method to maximise this measure thereby localising the robot.

This algorithm has been developed in the context of RoboCup in 2003 and was used with considerable success in the Sony Legged League for both the competition matches and the localisation technical challenge. The application of NightOwl to the Smart House domain at Curtin University shows promise. We also believe that it has application as a feature location generator for a multi-hypothesis global localisation and tracking system using geometric constraints.

## 6.1 Acknowledgements

We would like to thank the members of the 2003 rUNSWift RoboCup team and in particular Dr Will Uther, for the many useful suggestions in developing the algorithm for RoboCup 2003.

# References

[1] Kai O. Arras, Jose A. Castellanos, and Roland Siegwart. Feature-based multi-hypothesis localization and tracking for mobile robots using geomet-

ric constraints. *IEEE International Conf. on Robotics and Automation (ICRA02)*, 2002.

[2] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, chapter 6. Cambridge University Press, 2000.

[3] Bernhard Hengst, Darren Ibbotson, Son Bao Pham, John Dalgliesh, Mike Lawther, Phil Preston, and Claude Sammut. The UNSW robocup 2000 Sony legged league team. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors, *RoboCup 2000: Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Artificial Intelligence subseries of Lecture Notes in Computer Science*, chapter Champion Teams, pages 64–75. Springer-Verlag, Heidelberg, 2001.

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[5] Andre Olave, David Wang, James Wong, Timothy Tam, Benjamin Leung, Min Sub Kim, James Brooks, Albert Chang, Nik Von Huben, and Claude Sammutand Bernhard Hengst. The unsw robocup 2002 legged league team. *Workshop on Adaptability in Multi-Agent Systems: The First RoboCup Australian Open (AORC-2003)*, 2003.

[6] RoboCup 2003 Organizing Committee. RoboCup 2003 Legged League Challenges, 2003.

[7] RoboCup Federation. Robocup: Objective, 2003.

[8] RoboCup Technical Committee. Sony Four Legged Robot Football League Rule Book, 2003.

[9] Thomas Röfer and Matthias Jüngel. Fast and robust edge-based localization in the sony four-legged robot league. *7th International Workshop on RoboCup 2003, Lecture Notes in Artificial Intelligence*, 2004. to appear.

[10] Sony Corporation. *OPEN-R SDK, Model Information for ERS-210*, 2003. Programmers Manual.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.