A Fast Vision Sensor Model: Matching Edges with NightOwl

Raymond Sheh[1,2,3] and **Bernhard Hengst**[2,3]
[1]Department of Computing, Curtin University of Technology, Perth, WA, Australia
[2]School of Computer Science and Engineering, University of New South Wales, Sydney NSW Australia
[3]National ICT Australia, University of New South Wales, Sydney NSW Australia
rsheh@cse.unsw.edu.au, bernhard.hengst@nicta.com.au

Abstract

Accurate localisation is a key requirement for most autonomous robots. Increasingly, mobile robots use camera sensors. We describe an algorithm that uses edge features in a visual image to implement a sensor model for a given environment. The algorithm (NightOwl) was developed to help localise a Sony legged AIBO robot on a RoboCup soccer field, but has more general applicability. The NightOwl sensor model is optimised for efficient use of computing resources, is effective even when edge features are partially occluded and can be used with a variety of Bayes filter localisation methods.

1 Introduction

For the six years from 1998 to 2003 the RoboCup Sony legged league used six brightly colour coded beacons to help localise the AIBO robot soccer players. The longer term intention of the league is to progressively remove this aid and require that the robots use other visual cues for localisation. After all, real soccer players do not need beacons to determine their position on the soccer field.

We describe a method that provides information to localise a robot by interpreting the soccer field lines and field boundaries in a robot's visual sensor. As the rules of the competition require that all processing be performed in real time on each robot, an additional challenge is that the algorithm should keep up with the 25 frames per second data feed rate of the camera. In the presence of other processing requirements, this algorithm has a time budget of 25ms in which to complete processing on each frame.

The main contribution of this paper is an efficient algorithm that provides robot localisation information from edges in an image. The algorithm can tolerate sensor noise and partial edge occlusion from objects (for example, the ball or other robots). We call the algorithm NightOwl, as it allows the robot to see without beacons, in the dark, so to speak.

In the rest of this paper we will explain the RoboCup legged league environment in more detail and describe the role that this algorithm plays in terms of a sensor model. We will describe how field edges are extracted from the visual image and matched against known soccer field lines and boundaries. We then show how this sensor model is used with a particle filter and a Kalman filter for localisation.

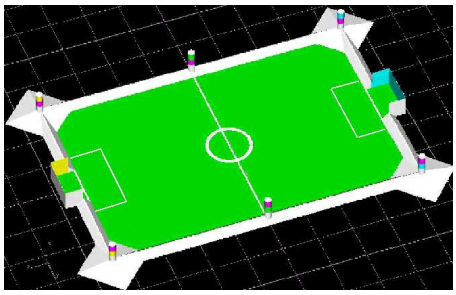
NightOwl was implemented for the 2003 RoboCup legged league localisation challenge in Padova, Italy, that required a robot to move to 5 successive positions on the field without the aid of beacons. The University of New South Wales (UNSW)/National ICT Australia (NICTA) team won this event in competition against 23 other international teams.

2 The RoboCup Legged League

The Sony legged robot league is one of several soccer leagues that currently form the RoboCup competition. In some leagues, the teams are required to design and build the robot hardware themselves. In the case of the Sony legged robot league, all teams use the same robotic platform, manufactured by Sony. The robots operate autonomously, meaning the robot is entirely on its own during the play of the game. Since all teams use the same hardware, the difference lies in the methods they devise to program the robots. Each team in the robot soccer match consists of four robots with the matches being played in two 10-minute halves.

The robots used in the Sony legged league are the standard version of the Sony ERS-210A AIBO entertainment robot (figure 1 (a)). Although designed for the entertainment market, these machines are extremely sophisticated robots, with an on board 385MHz MIPS processor, colour camera, accelerometers, contact sensors, speaker and stereo microphones. Each of the four legs has three degrees of freedom, as does the head. Programs are written in C++ using a PC-based development en-

(a)



(b)

Figure 1: (a) The Sony ERS-210A robot used in the Four-Legged Soccer League in 2003. (b) The legged league soccer field with six localisation beacons .

vironment. Programs are loaded onto a memory stick that is inserted into the robot.

The field used in the Sony legged league measures 3 metres wide by 4.5 metres long (figure 1 (b)). The field has an angled white border designed to keep the ball within the field. The game ball is coloured orange and the two goals are coloured yellow and blue. The field also contains six coloured distinguishable beacons. In front of each goal is the penalty area that only the goalie is allowed to defend. However, more than one attacker is allowed to enter the penalty area.

One of the most difficult tasks in creating the software to control the robots is localisation. That is, by observing landmarks around the field, the robot must be able to estimate its own position within the field and then estimate the positions of the ball and other robots. The source of difficulty in localisation is the inaccuracy of the sensors.The colour CMOS camera is the robot's main sensor. Since the robot only has one camera, distance to a landmark is estimated by its apparent size and other image geometry. With an image resolution of only 176 x 144 pixels, distant objects may occupy only a few pixels, so a difference of one or two pixels can make a big difference to the distance estimate. Noise in the camera image almost guarantees that there will be pixels that are of an incorrect colour. Therefore the size of the object and its distance are always uncertain. So the question is: how can the robot function reasonably reliably with uncertain sensors?

Reliable and accurate localisation is essential in allowing the robot to play soccer effectively. However, beacons are not always visible when playing the game [Olave *et al.*, 2003]. This happens frequently, for example, when the robot faces out of the field near the boundaries, when it faces the goal and generally when keeping its eye on the ball. In addition, the league plans to progressively remove beacons to force the robots to use more natural landmarks for localisation. In 2004 the center two beacons were removed for the first time. Both these reasons motivate the search for new landmark features that may be used for localisation.

In soccer it makes sense to attempt to use the field line markings and edges. Reliable observations of the field by the robot are limited in range to about 75cm and restricted to the field surface by the height of the sidewalls. The field of view of the camera in the direction the robot is facing is limited to 58°. The parts of field lines visible in the robot's camera are highly aliased, meaning that the same observation will be consistent with multiple positions and orientations of the robot on the soccer field.

NightOwl provides a sensor model in a recursive Bayes filter that estimates the position and orientation of a robot recursively in time.

3 Bayes Filters and Sensor Models

We will now provide a brief review of Bayes filters before describing NightOwl.

Let the position and orientation of a robot describe its state. We assume that the robot can be in any of a set of states S. The state of the robot is not directly observable, but we can make probabilistic predictions of the next state of the robot if we know its past state, the effect of state transitions given the robot's actions and the latest sensor measurements. If we assume that state transitions and observations depend only on the current state (the Markov assumption) and that they are described by stationary (time invariant) probability distributions, then the belief in the state of the robot, can be expressed as a Bayes filter recursive update equation as follows [Thrun *et al.*, 2000]:

$$p(s') = \eta p(z|s') \int p(s'|u,s)p(s)ds \qquad (1)$$

Where

- $p(s)$ is probability (or belief) that the robot was in state s prior to taking action u and prior to making observation z. This is referred to simply as the *prior*.

- $p(s'|u, s)$ is the probability of transitioning to state s' from state s given action u. This is referred to as the *motion model*.
- $p(z|s')$ is the probability of making observation z given that the robot is in state s'. This is referred to as the *sensor model*. It is this model that is provided by NightOwl, specialised for the interpretation of soccer field lines and field boundaries.
- η is a normalising constant.
- $p(s')$ is the probability that the robot is in state s' after taking action u and after making observation z. It is the posterior belief in the state of the robot.

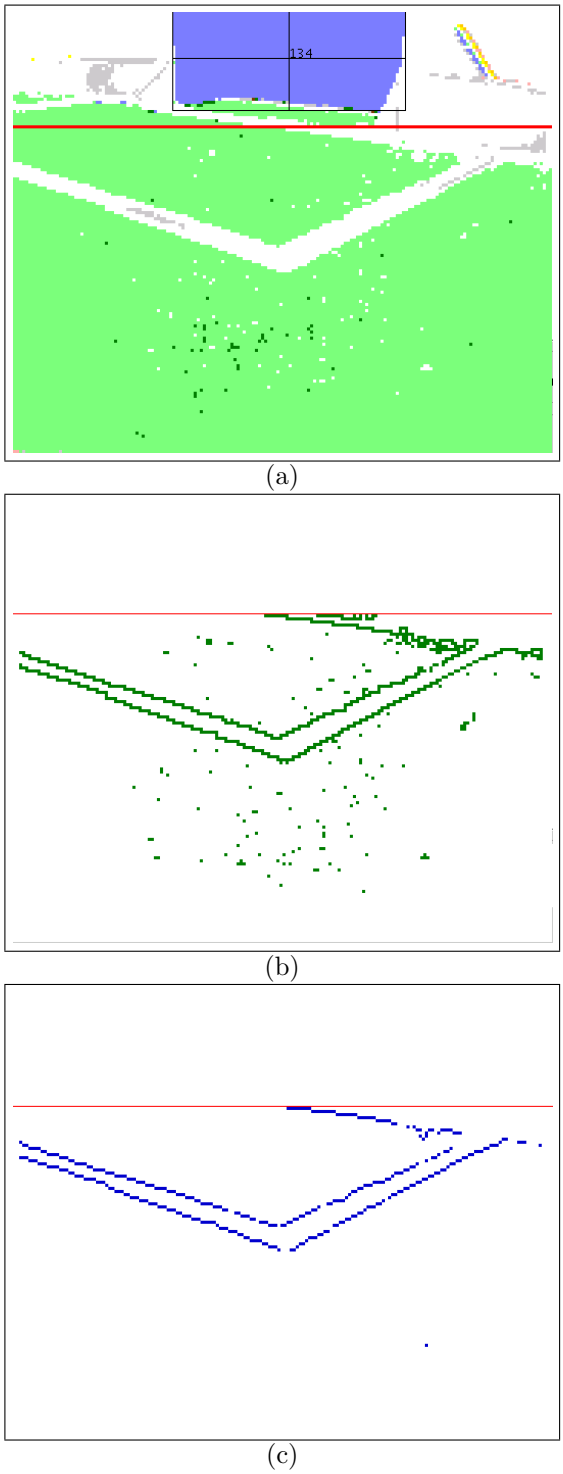
The significance of a Bayes filter is that it is directly related to Kalman filters and particle filters, amongst others [Thrun, 2002], that can localise a robot given a map of the environment. As NightOwl provides the sensor model in the Bayes filter it can be used with both Kalman filters and particle filters to help localise a robot. We will next describe the NightOwl sensor model starting with the detection of edges.

4 Image Edge Detection

Close examination of the 2003 soccer field shows that field lines and boundaries can be identified in an image as edges between green field areas and either the white field markings, the white field borders or the yellow and blue of the goals. We have restricted the NightOwl sensor model at this stage to detect only the predominant green-white edges. The extension to handle edges inside the goals, and other edges for that matter, is left to future work, but should follow directly.

The first step is to identify pixels in the image that belong on the boundary between the green of the field and the white of the field lines and borders.

The images come from the robot's CMOS colour camera at a rate of 25 per second and have a resolution of 176x144 pixels with an effective colour depth of 16 bits per pixel. A colour classified image, known as the *CPlane*, is formed by existing robot vision processes for every incoming camera frame, as a preprocessing step to colour blob formation. Using the existing CPlane for edge detection enables the use of fast binary operations on the image for no additional preprocessing cost. An example of a CPlane is shown in figure 2(a). The CPlane is formed by taking every pixel in the original camera frame and looking up its (Y,U,V) value in a colour table. The pixel is then replaced by the colour label returned from the lookup table. The lookup table is populated using a supervised machine learning method [Hengst *et al.*, 2001] and contains eight colour labels representing the ball, goals, beacons, other robots, the field and markings.



(a)

(b)

(c)

Figure 2: (a) an example CPlane showing a goal and the field, (b) the same image after processing by a basic field line sensitive edge detector, (c) the previous image using the NightOwl noise reduced edge detector.

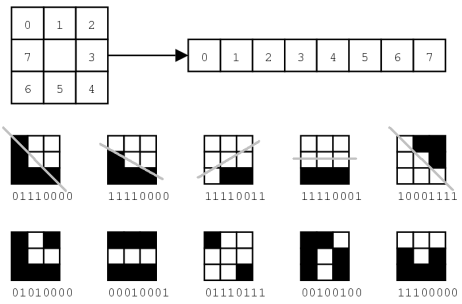


Figure 3: Byte encoding of neighbourhood patterns (top), examples of patterns around a central pixel that support field lines (middle) and ones that do not (bottom), with corresponding byte representations.

White-green boundaries in the image are found by using an edge detector specialised for use with the CPlane. The edge detector is a 3 by 3 kernel that scans the CPlane and identifies edge pixels as white pixels with more than a threshold number of eight-nearest-neighbour green pixels.

An alternative definition of edge points is to select the green side of a white-green edge and favor overestimating the width of observed field lines. However, due to the 1cm working resolution used in the field model, field lines are represented as single-width lines, thus the white side definition of the boundary was chosen.

While testing for several green pixels around a candidate white edge pixel was found to reduce "pepper" noise, this edge detector was still badly affected by image noise as shown in figure 2(b). A Gaussian kernel smoothing operator, and other operations on non-binary images such as a Canny edge detector, will require extra processing which may not be affordable.

An efficient solution is to reduce image noise by classifying the pattern of pixels surrounding edge candidates into those that are likely to support well-defined edges and those that are likely to be noise. The latter are simply discarded. Well-defined edges are defined as those that lie on roughly straight lines such as those shown in the middle of figure 3. The improved result of this pattern matching process is shown 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 the top of 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 edges. While 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 NightOwl. Rather than requiring an increase in processing, the edge templates above instead provided further computational savings. The "unrolling" process replaces the neighbouring pixel inspection and a summation step and adds negligible overhead. It simply comprises bit operations. The lookup table replaces the thresholding operation and is performed with minimal overhead as the 256 element lookup table is likely to be small enough to be loaded into the processor's memory cache.

So far, by leveraging the labelled colour encoding of the CPlane, we have designed an efficient noise resistant field line edge pixel detector.

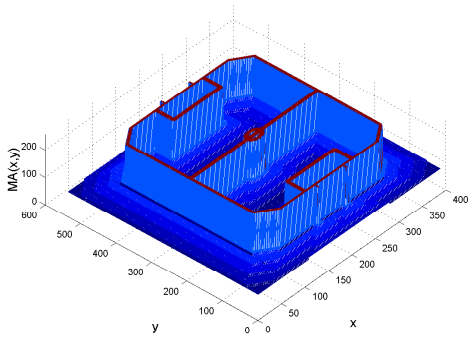
5 The NightOwl Sensor Model

The next task is to construct the sensor model that gives us the probability of an observed edge pixel being on a field line given a map of the soccer field. As identified field edge pixels are in the camera's image plane, we first project them onto the field plane. This field plane provides a common model given any camera pose. The projection is a straight forward geometric calculation given the robot's dimensions, joint angles and the pan and tilt of the head containing the camera. The equations projecting a point in the camera image into the field plane are given in appendix A.

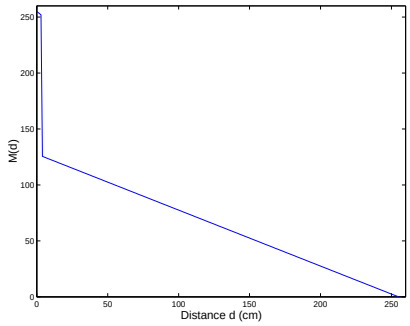
Once the points have been projected, those that lie further than 75cm from the robot are discarded since inaccuracies become too great at that distance. If there are more than a predefined number of remaining pixels, they are linearly sampled to preserve representative numbers of both near and far pixels. During the main competition, 20 pixels were retained whilst during the challenge competition, where the 25ms time budget was relaxed, 100 pixels were retained. Conversely, if there are too few pixels, NightOwl is aborted altogether due to lack of information. For both competitions, a minimum threshold of 20 pixels was used.

A sensor model is now constructed that provides a likelihood measure of an observed field edge pixel given the robot state (position and orientation). This sensor model $p(z_i|s')$ in the Bayes filter of equation 1 is instantiated as the likelihood of the identified edge pixel, i, being part of a field line or border given the state of the robot. This likelihood function is plotted in figure 4. A half cross section through a typical field line in the 3D graph in part (a) is represented in detail in part (b). The function is constructed empirically to give good performance and stored as a lookup table for fast access. As would be expected, it has a bell shaped profile with the highest likelihood near mapped field lines and a sharp drop-off locally at any significant distance from each line.

An innovation in the NightOwl algorithm is the structure of the sensor model. The sensor model is in the form



(a)



(b)

Figure 4: The sensor model showing the likelihood measure that an identified edge pixel is a field line given the position and orientation of the robot. (a) the likelihood score as a function of distance d from the nearest feature in centimeters. (b) a typical profile of the likelihood function in (a) as a vertical cross section perpendicular to an isolated field line or boundary.

of a two-dimensional array with one entry per square centimeter of the field including a border of 50cm around the extremities of the field to allow the hill climbing required for a Kalman filer (described later). The array is pre-computed only once using the five step process described in appendix C.

For a given number of observed field line pixels the individual likelihoods can be integrated multiplicatively, assuming conditional independence between observed pixels as shown in equation 2.

$$p(z|s') = \prod_i p(z_i|s') \tag{2}$$

This is the naïve Bayes assumption [Mitchell, 1997]. However, pixel observations are generally not expected to be independent given a robot's state. In practice, it was found that summation instead of multiplication yielded a speed improvement and added robustness against image distortion and spurious observations. Images tend to be distored due to camera motion, causing straight lines to become hooked at the ends and introducing spurious points some way off the real line. Summation favors matches where many, but not all, edge points coincide with model lines. Multiplication would favor matches where most of the points, including those displaced by image distortion, were as close to a model line as possible.

To summarise, the steps used to determine the probability of an observed edge being a field line, given the pose of the robot, and subject to a normalising factor, are as follows:

- Use the NightOwl image edge detector to find the pixels in the camera image that qualify as field line or field border pixels.
- Given the robot's stance, head tilt and pan, project each of these pixels from the image plane onto the ground plane, relative to the robot (see appendix A).
- Given the robot's estimated position and orientation on the field, use the relative position of each projected pixel to locate it on the map of the field and look up its likelihood of being a field line or field border in the sensor model likelihood function lookup table (figure 4) (see appendix B).
- Integrate the likelihood values for each pixel to obtain an estimate of the probability of observed edges in the image being part of the field markings, given the robot's position and orientation.

Having developed a sensor model for field borders and markings, this model can now be used in any filter derived from the generic Bayes filter. We will next discuss the use of the NightOwl sensor model with a particle filter and describe its adaption to work with a Kalman filter. As the observation of field edges is multi-modal and therefore non-linear over the robot states, our adaption effectively specifies the first stage of an Extended Kalman filter.

6 NightOwl and Monte Carlo Localisation

NightOwl may be used directly with an occupancy grid or particle filter based localisation method such as Monte Carlo Localisation [Thrun *et al.*, 2000; Röfer and Jüngel, 2004]. These multiple hypothesis tracking methods are a good fit for NightOwl as they can accommodate the multi-modal nature of $p(z|s')$. NightOwl is simply evaluated for each belief state as represented by the particle cloud.

NightOwl was used in the Curtin University Smart House Laboratory [Sheh and West, 2004]. The aim of the laboratory is to investigate technologies that can be used to assist the elderly and disabled in living independently in their own homes. It consists of a variety

Figure 5: The Smart House environment as seen from one of the overhead cameras. Robot position for figure 6 marked by the arrow.

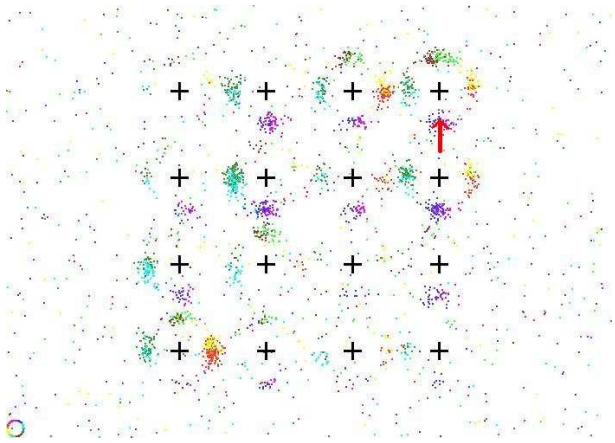


Figure 6: Clusters of particles representing the many possible locations of the robot given an observation taken by the robot in the position indicated in figure 5 and marked by the arrow.

of cameras and other sensors, including an AIBO ERS-210A robot. A Monte Carlo localisation filter is used to track the robot. The motion model comes from the robot's odometry [Hengst *et al.*, 2002] and the measurement update involves sensor fusion from several overhead cameras and NightOwl.

The overhead camera vision is noisy and inaccurate because moving objects and furniture often occlude the robot. In the Smart House laboratory NightOwl makes use of the crosses marked on the ground as its features[1] as shown in figure 5. Whilst the ability of the robot to observe a feature is reliable, the level of aliasing is very high as all these crosses look the same. The result, in the absence of additional information, was a large number of clusters of particles around each cross in the environment model as shown in figure 6. The overhead cameras tended to provide the global localisation signal and the NightOwl signal was effective more in tracking the robot. It was interesting to observe that using only NightOwl and motion updates, it was possible to, over time, globally localise the robot to points consistent with the symmetric pattern of the crosses. This could be done by walking the robot in a large square pattern around the environment, for instance. In this case the particle cloud divided into four sub-clouds consistent with the fourfold symmetry of the environment.

The system was implemented as proof of concept for the Smart House and yielded promising results[2]. NightOwl as the sensor model in the Monte Carlo filter was able to maintain the robot's localisation with large errors in odometry, lengthy periods of robot occlusion and in the presence of other moving objects in the scene.

[1]These crosses are normally used to calibrate the overhead cameras

[2]For further details see [Sheh and West, 2004]

7 Using NightOwl with Kalman Filters

The distribution of $p(z|s')$ can be highly multi-modal and impossible to describe in a clean geometric manner, making it unsuitable for updating a Kalman filter directly. To drive a Kalman filter we require an appropriate linearised approximation of $p(z|s')$ that is a Gaussian distribution.

We will denote this Gaussian approximation as $p^G(z|s')$. In an Extended Kalman Filter (EKF), this approximation consists of finding an estimate of the current state $\hat{s}$ based on the state at the prior time instant transformed by the motion model and then finding a state s^{max} that corresponds to a maxima of $p(z|s')$ closest to $\hat{s}$. The state s^{max} is used as the mean for the Gaussian distribution $p^G(z|s')$. The partial of derivative of $p(z|s')$ is taken along each state dimension at s^{max} to weight the variances used to describe the Gaussian distribution.

In NightOwl, the distribution $p(z|s')$ cannot be described in a way that allows this process to be evaluated in a closed form. However, we can find state s^{max} by hill-climbing to the nearest maximum to $\hat{s}$. This is implemented by recursively estimating the general gradient around $p(z|\hat{s})$ and progressively moving to a more likely state.

The distribution $p(z|s')$ is not only multimodal but "lumpy". In other words the path to a significant local maxima is often littered with local maxima that are undesirable and surrounded by distinct minima. This "lumpiness" is caused by the potentially large separa-

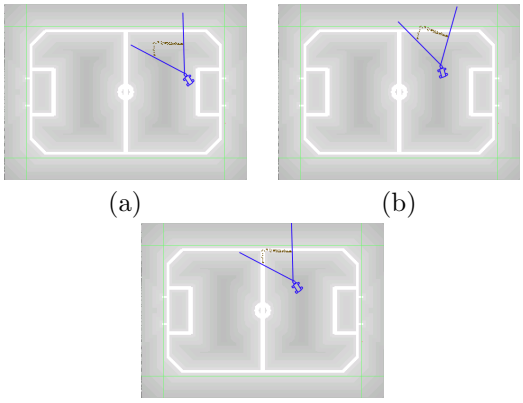


Figure 7: Three examples showing the matching of projected image field edge pixels to field edges. (a) a poor match given the previous $\hat{s}$, (b) a better match found by varying the position and orientation of the robot in a local neighbourhood, (c) the best match found in the neighbourhood, reported as s^{max}.

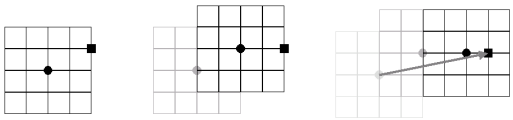


Figure 8: Two dimensional example of an iterative grid-based gradient ascent configured with similarly spaced grids. At each iteration, filled circle marks the start point, filled square marks the point of greatest probability.

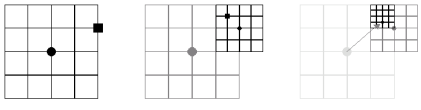


Figure 9: Two dimensional example of an iterative grid-based gradient ascent configured with reduced spacing grids. At each iteration, filled circle marks the start point, filled square marks the point of greatest probability.

tion between pixels in the observations. As a result, simple gradient ascent, one that only concentrates on the local derivative, tends to behave poorly as demonstrated in figure 7. In this example, given a starting position (a), the gradient ascent may terminate at the incorrect local maxima (b), without searching for the better matched $p(z|s')$ in (c). This problem is especially apparent around goal boxes and other feature-rich areas on the soccer field.

Instead, a grid-based discrete hill climbing process is performed by sampling $p(z|s')$ numerous times over a sampling window in order to find a satisfactory local maxima. Translational and rotational sampling is carried out around $\hat{s}$ with a given grid resolution and window size.

$$res =< res_x, res_y, res_\theta >$$
$$win =< win_x, win_y, win_\theta >$$

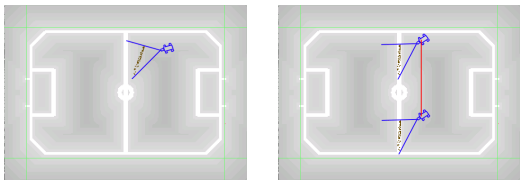
The maximum likelihood state over all the grid points is chosen as the new $\hat{s}$. This sampling and maximisation process is repeated K times, possibly at progressively finer grained resolutions, until the desired accuracy is reached. Figure 8 and 9 show the effect of iterating a fix and increasing resolution grid respectively.

This process of finding the state of maximum likelihood s^{max} close to the initial $\hat{s}$ consists of four steps:

- Evaluate $p(z|s')$ over a window of states around the estimate $\hat{s}$.
- Select the state in that window which maximises $p(z|s')$ and make this the new $\hat{s}$.
- Iterate with the new $\hat{s}$ until the desired accuracy in s^{max} is obtained.
- Estimate the variances in the estimate of s^{max}.

The selection of the sampling resolutions res and window sizes win, and the total number of iterations K is vital in trading off correctness, accuracy and running speed. A window that is too small may not track the overall gradient and result in a match far away from the desired maxima. A resolution that is too low may miss the desired maxima whilst windows that are too large or resolutions that are too fine will result in the algorithm exceeding the available time budget.

For the main competition, which was restricted by the time budget, it was found that a three level scheme ($K = 3$) was suitable. The first iteration was performed with $res =< 15cm, 15cm, 5^\circ >$ and $win =< 5, 5, 5 >$. This allowed the window to cover a large amount of area whilst still keeping the computation tractable. However, because this window was sampled at such a low resolution, it was highly likely that the true maximum will not be sampled. The second and third iterations were performed with resolutions of $res =< 5cm, 5cm, 5^\circ >$ and $res =< 2cm, 2cm, 2^\circ >$ and identical window sizes



(a) (b)

Figure 10: 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).

$win =< 3, 3, 3 >$. The use of windows with these specifications resulted in the ability of the algorithm to find the true maximum s^{max} so long as, at each iteration after the first, s^{max} lay within one sample of the reported $\hat{s}$. An example of this process taking place is provided in figure 7.

When run with these settings during a game, NightOwl was able to complete within its 25ms time budget over 90% of the time. A timer was used to abort the hillclimbing process if the time budget was exceeded. However, with these settings, NightOwl would at times fail to find a significant maxima due to the initial state $\hat{s}$ being too far from the robot's actual position or due to there being too much noise in the camera image. This situation was detected by thresholding the result of the final summation at s^{max} and discarding the result if the average pixel likelihood was too low.

In contrast, during the RoboCup localisation challenge where the time budget was relaxed, the three iterations used

$$res =< 2cm, 2cm, 2^\circ >, win =< 19, 19, 19 >$$
$$res =< 2cm, 2cm, 2^\circ >, win =< 5, 5, 5 >$$
$$res =< 1cm, 1cm, 2^\circ >, win =< 5, 5, 5 >$$

respectively. This larger search grid finds sensor matches to the model more reliably at the expense of extra computational costs. The three iterations take up to two seconds to complete.

With a limited view there is a chance that the resulting data only provides a "snap-in" that is unique in two of the three state dimensions. Figure 10 provides an example where NightOwl is more confident in its position along the field and less so across the field. On observing a single line, a maxima is found at the nearest line feature. It is unable to localise *along* the line. Note that the absence of pixels belonging to the centre circle, in this example, does not rule out the possibility that the robot is near the centre circle due to the possibility of total occlusion of the centre circle.

This information can be reflected in the variance of the Gaussian distribution of $p^G(z|s')$. The lack of information along a given direction corresponds to a "ridge" in the distribution of $p(z|s')$ and is detected as a by-product from the grid samples around s^{max}. Large differences in certain directions will tend to indicate a high degree of uniqueness. This would be reflected in a low variance in that particular direction. If the difference is small it is assumed that the match is not unique and a high variance is applied. In figure 10 for example, the Gaussian distribution corresponding to the match is highly elongated with the long axis running across the field (up and down the page). It was found that whilst this worked for translation, it was less useful for rotation for which we used a fixed variance.

8 NightOwl Performance in RoboCup

During the main competition, the primary benefit 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, 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 such situations, the starting estimated $\hat{s}$ was often so poor that NightOwl resolved the aliased image to the incorrect positions. Therefore, during the main competition, NightOwl was restricted to operation near the sidewalls where it yielded the greatest benefit. This problem was more a result of the simple Kalman filter being limited to maintaining a single belief state. A multi-hypothesis filter would be more appropriate with this sensor model, as was indeed demonstrated in the Curtin University experiment discussed earlier.

NightOwl's performance came to the fore in the technical localisation challenge. For this challenge the six localisation beacons were removed from the field and a robot was required to visit five predetermined coordinates on the field. The robots were scored on the accuracy with which they would localise the x and y dimensions of their pose at each location before proceeding to the next one. The goals were used for the necessary global localisation but are too inaccurate for precise localisation. NightOwl was solely responsible for achieving a perfect score on one of the difficult locations placed inside one of the goal boxes. The UNSW/NICTA team won this technical challenge against the other 23 international teams.

9 Concluding Discussion

The NightOwl sensor model is less efficient when the image features are highly aliased in the environment. This occurs when only a few pixels are observed or when features are very repetitive. One solution is to compound

images to build features that are more unique. For example, stitching together multiple field views, as suggested by Will Uther, was demonstrated to reduce the aliasing at the expense of having to match a greater number of pixels. While there is an accuracy-computation tradeoff, the stitching process also introduces additional inaccuracy due to camera movement.

The process of searching for a strong local maxima in $p(z|s')$ for the purpose of generating a Gaussian distribution $p^G(z|s')$ is worth further investigation, with approaches such as simulated annealing [Kirkpatrick *et al.*, 1983] as possible candidates for more effective optimisation techniques.

The sensor model may be improved by accounting for varying variance in measurements as the image pixels approach the horizon. In practice it was found that the usable edge pixels were limited to about 1.5m from the robot when stationary and 75cm when moving, due to the low camera angle and encoder errors.

One important characteristic of this process is that it degrades gracefully in the presence of occlusion, so long as the edges of the occluding object are not picked up by the edge detector. Reducing the number of observed pixels may add more local maxima to the sensor model. The result may be increased ambiguity but existing maxima will be preserved.

A more complex feature construction would make NightOwl an ideal candidate as a front end to a feature based multi-hypothesis localisation method, such as for example in [Arras *et al.*, 2002]. NightOwl can provide such a localiser with a discrete, limited number of hypotheses, when for example, a corner is seen.

An obvious extension is to detect other distinguishing features, such as the green-yellow or green-blue edges between the field and the goals. With additional sensor model arrays, this is expected to enhance localisation significantly. Of course any environmental feature edge may be modelled and it would be interesting to study a combination of sensor models based on various features.

9.1 Acknowledgements

We would like to thank the members of the 2003 UNSW/NICTA RoboCup team and in particular Dr Will Uther, for the many useful suggestions in developing the algorithm for RoboCup 2003. We would also like to thank the School of Computer Science and Engineering at the University of New South Wales, and National ICT Australia, for their academic and financial support of the team.

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program.

Appendix A

The following equations are used for projecting a point on the camera image (w, u) of the AIBO ERS-210A to a point (x, y) on plane parallel to the field, i.e. the ground plane.

Definition of variables:

hF = vertical height of the front robot shoulders (from pWalk parameters ie 7cm [Hengst *et al.*, 2001])

h = vertical height to the base of the neck (13cm)

$b = hB$ = vertical height of the back robot shoulders (from pWalk parameters 11cm)

s = inter-shoulder distance of the robot (fixed 11.9cm)

A = the tilt angle of the body of the robot to the ground

T = angle of tilt of the robot's head from sensor readings (down positive assumed)

P = angle of pan of the robot's head from sensor reading (right positive assumed)

c = distance from top of neck to camera (fixed)

f = distance from camera to image plane (fixed)

(u, w) = the point on the image to be projected assuming origin is in the middle. u = distance from middle of image (up +ve) to the image point to be projected. w = distance from middle of image (right +ve) to the image point to be projected

n = length of neck from top of shoulders (fixed actual)

p = height of plane (ground plane) that we are projecting to from the field ($p = 0$ for the field plane)

x = sideways distance from camera of projected image point on ground plane y = forward distance from camera of projected image point on ground plane ie (x, y) are the co-ordinates of projected point on ground plan relative to robot neck base.

$A = arcsin((b - hF)/s)$

Effective tilt R = body tilt + head tilt
$R = A + T$

Coordinates of camera point(c)

$x_c = c * \sin(P)$
$yc = n * \sin(R) + c * \cos(R) * \cos(P)$
$z_c = h - p - c * \sin(R) * \cos(P) + n * \cos(R)$

Coordinates of image point (i) to be projected
$x_i = w * \cos(P) + (c + f) * \sin(P)$
$y_i = -w * \sin(P) * \cos(R) + (c + f) * \cos(R) * \cos(P) + (n + u) * \sin(R)$
$z_i = h - p + w * \sin(P) * \sin(R) - (c + f) * \sin(R) * \cos(P) + (n + u) * \cos(R)$

Intersection of ground plane $z = 0$ and of line through points c and i
$k = z_i/(z_i - z_c)$
$x_i \leftarrow x_i + k * (x_c - x_i)$
$y_i \leftarrow y_i + k * (y_c - y_i)$

If $z_i - z_c >= 0$ then point is projected over or at the horizon, so discard.

Appendix B

The equations to transform robot relative pixel (x_i, y_i) in the field plane to environmental pixel coordinate $(x_{s',i}, y_{s',i})$ given the robot state $s' = (x'_s, y'_s, \theta'_s)$ are:

$x_{s,i} = x_i \cos(\theta'_s) - y_i \sin(\theta'_s) + x'_s$
$y_{s,i} = x_i \sin(\theta'_s) + y_i \cos(\theta'_s) + y'_s$

Appendix C

The five step evolution of of the sensor model array MA is computed once at boot-up and kept in memory, unchanged, for the duration of program execution. The final model function is shown in figure 4.

- Zero the array MA.
- Populate entries that correspond with field features with some maximum constant M.
- Set every element in the array that is not equal to M to the value of it's highest neighbour minus one.
- Iterate the previous step until no element changes value. Note that at the end of this step, the value of each element is inversely proportional to its distance from the nearest feature.
- Apply equation 3 to each element in the array.

$$MA[x][y] \leftarrow \begin{cases} MA[x][y] & , d < w \\ \frac{1}{2} MA[x][y] & , d \geq w \end{cases} \quad (3)$$

where d is the distance from the field line and w is a constant that sets the width of the top of the "bell" in figure 4.

References

[Arras *et al.*, 2002] Kai O. Arras, Jose A. Castellanos, and Roland Siegwart. Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints. *IEEE International Conf. on Robotics and Automation (ICRA02)*, 2002.

[Hengst *et al.*, 2001] 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.

[Hengst *et al.*, 2002] B. Hengst, D. Ibbotson, S.B. Pham, and C. Sammut. Omnidirectional locomotion for quadruped robots. In S. Tadokoro A. Birk, S. Coradeschi, editor, *RoboCup International Symposium RoboCup 2001: Robot Soccer World Cup V, Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence*, volume LNAI 2377, pages 368–373. Springer, 2002.

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

[Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, Singapore, 1997.

[Olave *et al.*, 2003] Andres Olave, David Wang, James Wong, Timothy Tam, Benjamin Leung, Min Sub Kim, James Brooks, Albert Chang, Nik Von Huben, Claude Sammut, and Bernhard Hengst. The UNSW RoboCup 2002 Legged League Team. *Workshop on Adaptability in Multi-Agent Systems: The First RoboCup Australian Open (AORC-2003)*, 2003.

[Röfer and Jüngel, 2004] 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.

[Sheh and West, 2004] Raymond Sheh and Geoff West. Visual Tracking and Localization of a Small Domestic Robot. In *RoboCup 2004 Symposium*. Springer, 2004.

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

[Thrun, 2002] Sebastian Thrun. Robotic mapping: A survey. Technical Report CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, 2002.