Robust, Dependable Model-based Visual Localisation for Mobile Robots

Nick Pears, Steve Crook-Dawkins and John McDermid
Department of Computer Science
University of York
York, YO10 5DD, UK
email: nep@cs.york.ac.uk, steve@cs.york.ac.uk, jam@cs.york.ac.uk

Abstract

We describe the basic structure of monocular model-based visual localisation (MBL) algorithms, which may be used to guide mobile robots in a variety of human environments. For such applications, we address the dependability and safety issues, which are rarely dealt with in the robotics literature. First we investigate various scenarios that may occur in the operation of the standard MBL algorithm, in order to identify the main hazards that may lead to system failure. We then propose both commissioning and operational solutions to minimize the risk of a hazardous situation occurring, with the aim of providing a template for the implementation of a highly dependable MBL system, without a serious compromise on the system availability

1 Introduction

The Robotics and Computer Vision communities have developed various approaches for localising robots using on-board cameras. We have adopted a specific single camera (monocular) approach called "model-based localisation" (MBL), where a 3D model of the environment (connected edge segments) is stored in a database. The camera viewpoint relative to this model can be computed if features from a real image of the environment can be matched to the correct corresponding features of the stored 3D model and if a camera model is known. This approach was pioneered by Lowe [2] and an improved fully projective formulation was developed by Araujo et al [1].

In the case when mobile robots share the same environment as humans and are large enough to present an injury risk, we need to address dependability and safety issues. In particular, visual sensor systems have a high dimensional input space, with a corresponding large variability in input pattern presented to the robot. We need to predict ways in which we may encounter incorrect interpretation of this highly variable sensor data, which may lead to system failure. For example, one can envisage a situation where the MBL system in some sense fails, causing the robot to collide with a stack of shelves in a factory, which could then fall and injure people in the vicinity. Although large robots in human environments should always be equipped with a 'last-resort' mechanical safety mechanism, such as a large flexible bumper, which, when flexed, causes the robot to stop as soon as possible, there is still a residual safety concern with such collisions and so minimisation of the number of these incidents will be desirable in terms of constructing a safe system and a corresponding safety argument. The focus of this paper is as follows: *Our primary aim is to examine the dependability issues of a standard model-based visual localisation algorithm and its ability to guide the robot with minimal risk of an unsafe failure, namely a collision with the mapped environment.* This will be done in the context of maintaining a high system availability; that is our secondary aim is to minimise the time in which the robot is in a halted, fail-safe state.

Note that this does not include system-wide issues, such as the dependability of the sensors, software and computing platform and the dependability of any obstacle avoidance systems. Such systems, particularly human obstacle avoidance, must undergo their own dependability scrutiny and the obviously the way in which the various system components interact must also be examined. Although such issues are outside the scope of this paper, an overall concern is to reduce the level of trust required of the sensors/software/computing system, making it simpler to develop the required level of dependability.

The structure of the paper is as follows: In the following section, we provide context by outlining the structure of a standard MBL algorithm and discussing timing issues, uncertainty bounds and the maintenance of tracking lock in the algorithm. Section 3 then defines how the fail-safe system may work and identifies the main algorithmic hazard in the system. Section 4 investigates various scenarios that may occur in the operation of the standard MBL algorithm, in order to identify typical event sequences that may lead to the generation of either hazards or fail-safe actions. Section 5 proposes both commissioning and operational solutions to minimize the risk of a hazard occurring, with the aim of providing a template for implementation of a highly dependable MBL system, without a serious compromise of the system availability (defined as the time when the robot is not in fail-safe, halted mode) and without placing great demands on the integrity of the implementation of the algorithms.

2 Outline of model-based localisation

In this section, we first overview dual sensor predictor-corrector systems, then we discuss the algorithm structure given in figure 1. Finally we discuss both sensor sampling issues associated with this algorithm and the need to maintain uncertainty bounds on the pose estimate.

2.1 Dual sensor predictor-correctors

Although it is possible to develop MBL systems that use only vision sensors, often we cannot make position estimates at a sufficiently high frequency, due to both the computational intensiveness of the data processing task and the lack of availability of non-occluded reference targets in the camera field of view at all times. As a result, typically such systems are equipped with one or more proprioceptive sensor systems. These sensor systems estimate robot position by integrating the measured robot motion over time. Examples of such sensor systems include odometry systems, using, for example, wheel encoders and inertial navigation systems. The advantage of such systems is that their results are always available and so can be sampled at a relatively high frequency to give an up-to-date estimate of pose (position and orientation). However, the well known problem with such systems, is that their estimates are subject to significant drift over time, making them unsuitable as single sensor systems. Systems which combine internal proprioceptive sensor systems with sensors that can measure position with respect to known external reference points can eliminate the drawbacks of both types of system. The proprioceptive system compensates for the relatively low frequency positional update of the vision system, whilst the drift in the proprioceptive system is corrected by the external reference measurement provided by the vision system. In a sense, one can think of the proprioceptive system predicting the underlying pose state, whilst the vision system periodically corrects this in a process of sensor fusion. The key to fusing the data from both sensor systems is a good understanding of the noise characteristics associated with each sensor system's pose estimate.

2.2 The algorithm structure

The term "model" in this MBL approach refers to the fact that a 3D model of the robot's environment is stored in a database, which can be accessed by the robot localisation system. We are not concerned here with how the world map is acquired or with simultaneous localisation and map building (SLAM) techniques, except that we assume that any errors in the mapping process are small compared to sensor measurement errors (this assumption, of course, needs to be validated in any practical application of MBL). The data stored relates to straight edges in the scene, referenced to some arbitrary, right-handed Euclidean coordinate frame called the world frame. Straight edges are often used as they are strong features in many man-made indoor and outdoor scenes and there are well developed feature extraction techniques to extract such features from raw images. Other possibilities include corners and Lowe's local scale-invariant features [3]. It should be noted that another model is required for MBL systems, namely a model of the *intrinsic* properties of the camera. Such as model defines the way in which 3D objects in a camera centered frame are projected onto the image plane and captured in the pixel-based coordinates of that image plane. A camera model is obtained by performing a camera calibration, which can also remove the effects of radial distortion.

Given the model of the camera and the model of the environment, we only need to know a good estimate of the pose of the camera (and hence robot) with respect to the world frame (this pose is also known as the extrinsic camera parameters) in order to predict where the modelled world edges are likely to appear in the image. This "project world model" process is shown at the top of figure 1. Note that in the case of a 'fixed camera' mobile robot confined to a ground plane, the pose will only have 3 DOF.

As shown in figure 1, a raw image is fed into the system, from which a simple standard edge detector, such as the Sobel operator is used to detect edgels, that is, pixels on a sufficiently large intensity gradient in the (possibly smoothed) image. The projected model edges and image edgels are then matched using the edge matcher module, shown in figure 1. This module searches in a direction perpendicular to the model edges in order to detect edgels within a search

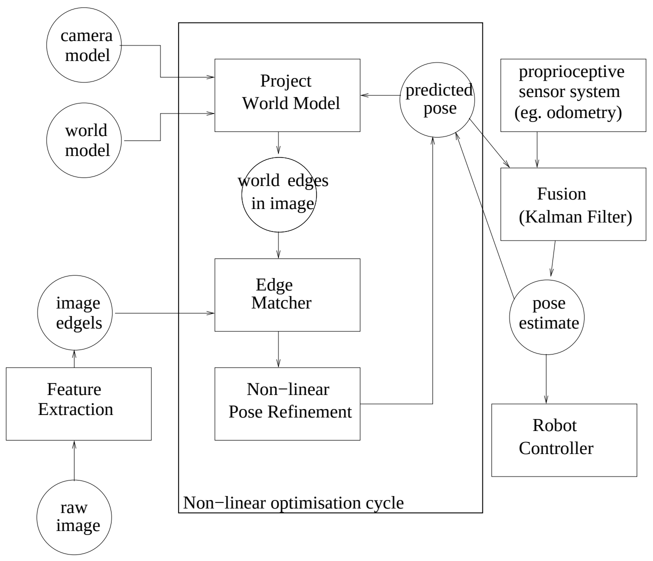

Figure 1: Model-based localisation system

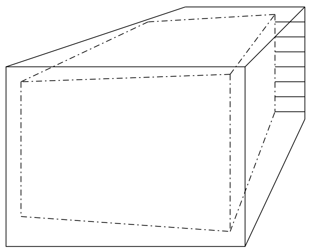

Figure 2: Model (solid line) and edgels (dashed line)

window of the model edge, which have a similar orientation and significant strength. Figure 2 shows an example of a typical model projection from the estimated pose and edgels extracted from a camera at the true pose. Note that the usual graphics techniques of back face culling are employed, so that hidden edges of the model are not generated. Here the estimated distance of the camera to the box shape is closer than the actual distance to the box and examples of the search lines are shown at the back right edge of the model. Obviously correct data-to-model (edgel to edge) association is crucial to the successful operation of MBL and this association features strongly in our dependability discussion.

The question now is, assuming we have correct data-to-model association and we have measured the disparity between data and model, how do we refine the camera pose estimate in order to reduce the overall disparity between model edges and data edgels? Due to the non-linear nature of this problem, an iterative process is required, which forms the core of the MBL algorithm and this is shown by the loop which is boxed in figure 1. For the mathematics of the core algorithm, the interested reader is referred to Trucco and Verri [5] and Araujo et al [1]. In summary, the solution can be cast in the form of a multidimensional Newton-Raphson minimisation problem, where we linearise the non-linear projective effect of pose change on the data-to-model error at each step, update our pose estimate and re-compute the disparity until we reach convergence, or convergence fails (divergence or maximum iterations reached).

If the system converges to a strong match solution, then the measured pose is output from the optimisation loop and passed, along with its measure of uncertainty, to the input of the fusion module, where it is fused using the current estimate of proprioceptive uncertainty to give an update of the estimated camera pose and associated uncertainty, which can be then used to control the speed and steering of robot manoeuvres.

2.3 Timing issues

The large data rate of raw video data will not allow feature extraction and matching to be achieved at this a standard 25Hz or 30Hz frame rate without the use of special purpose hardware. What would be a more reasonable frame processing rate? A typical human walking pace is about 1.5m/s and a reasonable estimate of maximum speed of a mobile robot in a human environment may be 1m/s (less for larger robots). Thus if we would like to maintain position updates for a maximum every 10cm travelled, a frame rate of 10Hz is required. On a standard 2GHz, 512MB PC, using VGA size images (640x480), frame rates of 1Hz are more typical and it may be necessary to pass several proprioceptive sensor pose samples, which are not fused with vision data, directly to the robot controller.

Figure 3 illustrates a situation, where, due to the high visual data rate, proprioceptive sensor sampling is 10 times the visual sensing rate (useful frame rate). In time section A, the edgels are extracted from the image, in time section B, the iterative pose optimisation executes and in some of the remaining time in section C, the new pose estimate and associated uncertainty is computed. Note that in time section C, the fusion of frame 1 data must occur with proprioceptive data up to and including sample 0, which occurs (almost) simultaneously with frame 1 capture. The pose and associated uncertainty at time sample 10, must then be computed using the visually corrected pose and uncertainty at time sample 0 and the retrospective proprioceptive data from time samples 1 to 9. In

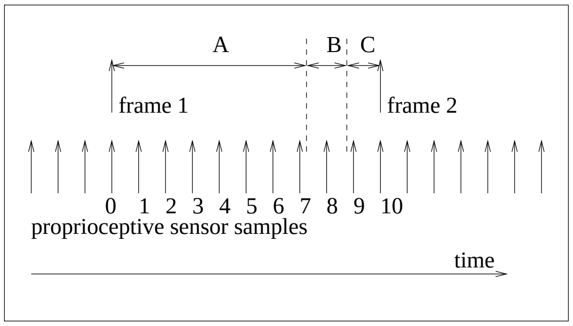

Figure 3: Sensor sampling

this manner, we can deal with the time lag associated with the frame processing time of the MBL system.

2.4 Pose uncertainty estimation

The MBL algorithm in addition to estimating the camera/robot pose state, must also estimate the uncertainty in that state, both for sensor fusion purposes and also to determine whether a collision with the mapped environment is possible. Obviously, the estimate of this uncertainty should be derived from the modelled noise distribution on each sensor system contributing to the pose state estimate.

We would really like the boundary of our uncertainty bounds on robot pose to represent the boundary within which the pose probability is greater than zero and where the pose probability is zero at all points outside this boundary. However, the best models for the probability density distribution of measurements from a sensor are often not finitely bounded. A common model used is the normal or Gaussian distribution, which never actually reaches zero at an arbitrary distance from the mean and so it is common practice to cut the tails of the distribution at n standard deviations from the mean. A typical value for n would be three, giving a 3SD (three standard-deviation) bound that indicates the confidence that the robot is within the uncertainty bounds is 99.7% and 0.3% that it is outside these bounds. This means that even if the error model is very good (i.e. the actual sensor noise distribution is very close to normal), we would still expect to see 3 in 1000 measurements being randomly generated outside the 3SD uncertainty bound, due to random sensor noise.

3 Fail-safe operation and hazards

We envisage an environment, where the allowable uncertainty in position may vary at different positions within the environment. This is intended to reflect the fact that robot pose uncertainty should be significantly less when maneouvering close to any structures

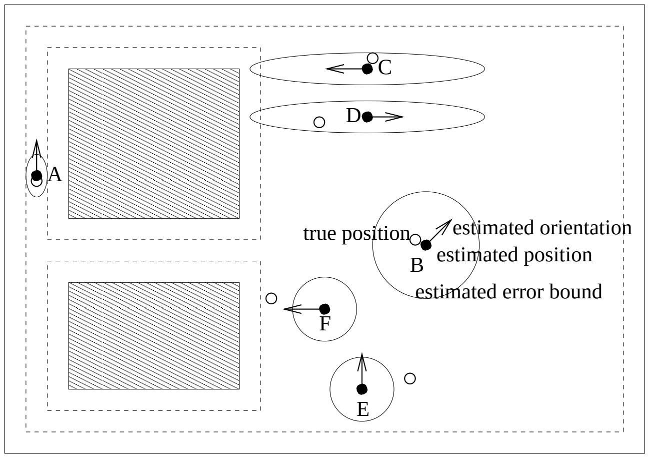

Figure 4: Map showing positional uncertainty

in the environment, than when it is moving across an open space. The simplest scheme may be to say that the uncertainty in robot position should never infringe within a certain safety margin from any mapped structure, shown by the dashed line in figure 4, otherwise the system should fail-safe, which in this case is to stop as quickly as is deemed to be safe (without skidding or losing the robot's load). Figure 4 illustrates six robot estimated positions (solid circles) A, B, C, D, E and F with their associated positional uncertainty bounds (ellipses), velocities (arrows) and true positions (hollow circles). Here the uncertainty bounds of positions A and B (and E and F) are deemed acceptable, whereas those of positions C and D are not. More sophisticated approaches may dynamically alter such boundaries depending on, for example, predicted time to contact, so if robot orientation is known to sufficiently high accuracy, it would be possible to allow the robot in situation D to continue to operate whereas in situation C the robot should still fail-safe. Positions E and F show situations where we have a hazard, that is the true robot position is outside the estimated uncertainty bounds. Again, assuming good orientation estimates, situation E poses no immediate threat of collision, whereas in situation F, the hazard is about to cause a system failure, namely a collision with the environment.

3.1 The mismatched filter hazard

Here we define a hazard as *a physical situation, often following from some initiating event, that can lead to an accident* [4]. The implications of this are that when a hazard exists, no other events are required for an accident to occur, in effect, the hazard is the full precondition for the accident. Whether or not the accident occurs depends on the specific environmental conditions whilst the hazard is present. With this in mind, we have identified the main algorithmic hazard: *The main algorithmic hazard is an over-confident estimate of camera/robot pose, such that the estimated pose is outside the estimated uncertainty bounds associated with that estimated pose.*

In tracking algorithms, such as the Kalman filter, this is known as a "mismatched filter" [6]. The longer such a hazard condition exists, the more probable a collision with the mapped environment becomes. The need to avoid this hazard condition compels us to maintain large uncertainty bounds on the pose estimate. For example, if it was deemed that the sensor noise models were Gaussian, as discussed in section 2.4, then we could make the uncertainty bounds n standard deviations wide, where n is arbitrarily large. However, increasing n yields exponentially diminishing returns in terms of the reduction in the number of hazards and thus we are essentially trading large amounts of system availability (when system is not in a fail-safe, stopped mode) for only marginal improvements in system dependability. This focusses our effort on accurate maintenance of both the pose and the pose uncertainty bounds.

3.2 Recovering tracking lock.

In addition to generating high system availability, maintenance of a low uncertainty in pose has a self-sustaining effect. The smaller the positional uncertainty, the more likely the vision system is to converge, and given that it converges, it is more likely to converge on the correct model-to-data association. This is the concept of maintaining "lock" when tracking: it is easier to maintain lock on the environment model when the uncertainty in pose, particularly orientation uncertainty, is low. However, it has to be assumed that at some point the system will lose lock and this lock will need to be recovered to prevent the system becoming totally unavailable and stuck in fail-safe state. There are two scenarios for recovering tracking lock:

1. When tracking lock is lost, the uncertainty in pose will grow until it infringes the safety-margin, in which case fail-safe will be triggered. The system will then try to reduce the uncertainty in pose by searching for the correct pose over the full uncertainty in pose. As an indicative example, we could sample in steps of 0.2m over the full positional uncertainty and every 1 degree over the full orientation uncertainty. If the positional uncertainty was around $4m^2$ and orientation uncertainty around 10 degrees, we would have around 1000 poses to test, from which we could choose the best model-data match from those that converge. If the system has hit fail-safe at an orientation where there are no visible features, the robot would be stuck and require human intervention to put it back online. One solution to prevent this would be to allow the camera to pan in order to view modelled world features.

2. For small enough uncertainty bounds, we can perform the above search as the robot continues to maneouvre and reduce the uncertainty bounds before the safety-margin is infringed.

4 Scenarios in the system operation

In this section, we systematically consider various scenarios in MBL system operation in order to identify situations when the visual system cannot provide a pose update, situations when the system must fail-safe, and situations where hazards may occur. It is important to note that, unlike figure 1, the flow charts presented in figures 5, 6 and 7 do not represent algorithms, but rather an informal schematic of data, processes and events. For example, at the bottom of figure 6, the system may not know whether a correct data association has occurred and therefore does not know if it is providing a valid output or an invalid output. Similarly in figure 7, the fusion process does not know whether valid or invalid vision data is presented at its input.

4.1 Proprioceptive update

Firstly, we will look at the proprioceptive sensor operation. Figure 5 illustrates the update and uncertainty bound check when the proprioceptive sensor system only is sampled. Due to the integrating nature of this type of sensor system, the uncertainty bounds tend to grow monotonically and at each update we need to check if they infringe upon the safety margin of the mapped environment (the dotted line in figure 4). If they do, then the robot must fail-safe, that is, stop in as short a distance as is deemed safe.

4.2 MBL pose/uncertainty update

Secondly, consider a pose and uncertainty bound update using a captured frame of visual data, as shown in figure 6. The current pose estimate from the latest proprioceptive pose update is used to generate the predicted view using the world model. Note that it may be found that no parts of the modelled environment are visible in the region of the predicted field of view, in which case, no output can be achieved from the MBL visual system. Furthermore, due to poor lighting, fog (if outdoors) or lack of visual features in the actual camera field of view, few or no edgels may be generated. Again, in this case, no output can

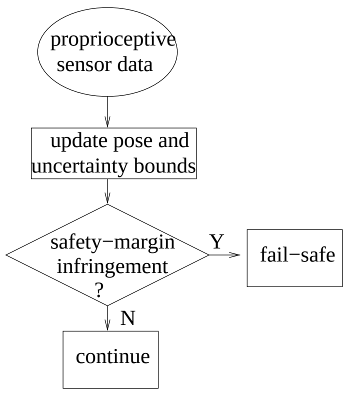
proprioceptive sensor data
update pose and uncertainty bounds
safety−margin infringement ?
Y
fail−safe
N
continue

Figure 5: Data/events in proprioceptive pose update

be generated. Both cases are easily automatically detectable. Given an available model and edgels, the MBL system attempts to make an association between model (edges) and data (edgels), by searching up to a maximum distance in a direction perpendicular to the model edge. The iterative optimiser then alters the MBL system's estimated pose in order to reduce the distance between the model edges and data edgels. If this process fails to converge, then, again, no output can be achieved. If convergence is achieved, then some form of metric should determine how good the match is, which could, for example, be the mean-square error separation of matched edgels to model edges. Again, if this is below a threshold, the system may decide not to use the vision data and no output is achieved. Even if this test is passed, there is no guarantee that a correct model-data association has been made. Only if the association is actually correct is a valid vision output achieved, as shown in figure 6.

4.3 Pose/uncertainty update via fusion

In fusing the pose data from the two sensor systems, we aim to determine what the most probable pose of the robot is and the probability density distribution around that pose (which define the uncertainty bounds). Before we do that, we need to determine whether the two data sets are consistent with each other. To check this, the matching process may use, for example, a thresholded metric such as the Mahalanobis distance, which measures the distance between the two sensor estimates normalised with respect to their covariances. Depending on whether the outputs from the two sensor systems can be fused, the situations that we arrive at illustrated at the bottom of figure 7 and reading from right to left are:

1. The system has maintained "lock", namely the vision system has converged on the correct data-

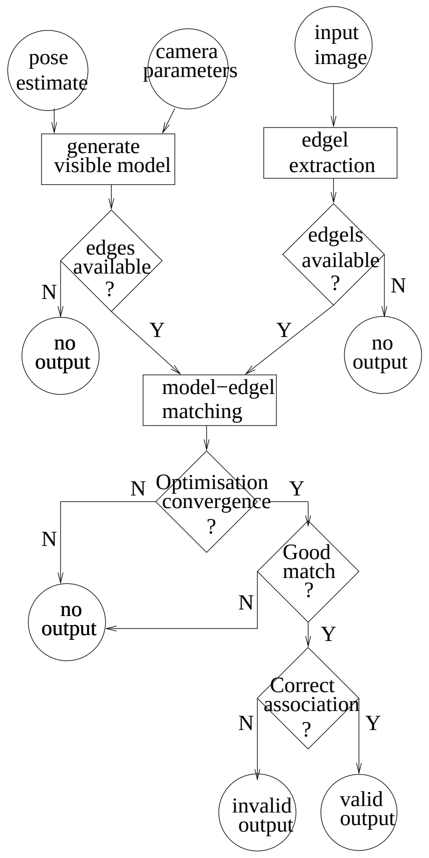

Figure 6: Data and events in vision process

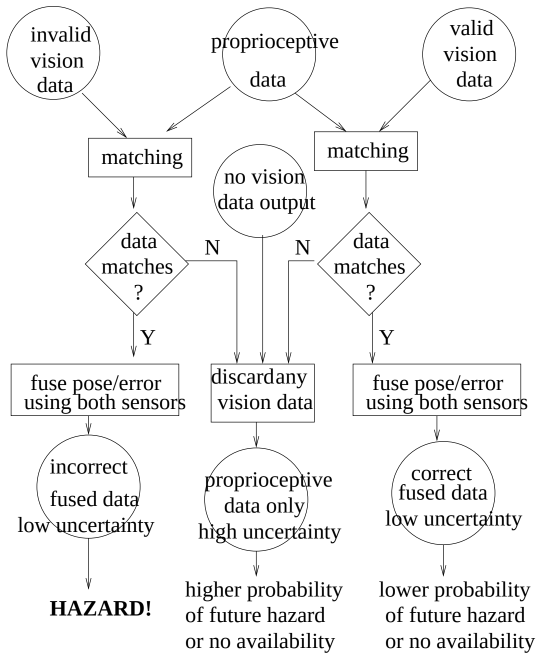
invalid vision data
proprioceptive data
valid vision data
matching
matching
no vision data output
data matches ?
N
N
data matches ?
Y
Y
fuse pose/error using both sensors
discard any vision data
fuse pose/error using both sensors
incorrect fused data low uncertainty
proprioceptive data only high uncertainty
correct fused data low uncertainty
HAZARD!
higher probability of future hazard or no availability
lower probability of future hazard or no availability

Figure 7: Data and events in the fusion process

model association and this matches with the proprioceptive pose estimate. Thus the uncertainty bounds are low and the true robot position is within these uncertainty bounds. This is the situation we hope to be in most of the time.

2. The system has lost "lock" in that there is no pose output from the vision system at all or no pose output that matches with the proprioceptive system pose estimate. Thus the vision pose estimate, if any, is discarded, and relatively high uncertainty in pose remains. Because of this, the system is more susceptible to future hazards and scenarios of no availability due to fail-safe.

3. The system believes that it has locked to the world model and has high confidence in its pose estimate, which is reflected in its low uncertainty bounds. However, it has not correctly associated data with model and the true robot position is outside the uncertainty bounds. This is a hazard and is a full precondition for a collision with structures in the mapped environment.

In conclusion, we note that there may be many situations in which ambiguity exists in terms of the projected model. That is, due to repeated structures in the environment, which are in close proximity to each other, it will be possible for the data edgels to lock onto the wrong part of the world model, such that the match between data edgels and model edges is very good. This *false lock* generates *a mis-matched vision based pose estimate which is consistent with the proprioceptive pose estimate* and thus we are likely to have a true pose which is outside the uncertainty bounds of the pose estimate, i.e. a hazard.

5 Proposals to avoid hazards

A key factor in avoiding the hazard condition of a mismatched filter, is to avoid situations in which there is pose ambiguity due to the appearance of the world model being similar within the uncertainty bounds of the pose. Firstly we propose a means of systematically adding features to the environment in the commissioning phase to remove instances of potential matching ambiguity over the predicted maximum pose uncertainty bounds for a given pose. Secondly we propose an operational procedure to fail-safe if the robot exceeds these predicted *safe uncertainty bounds.*

5.1 Commissioning proposal

If we can keep the uncertainty bounds of pose small, then there is inherently less risk of ambiguity in the environment causing a hazard. Smaller uncertainty bounds can be maintained if we can see parts of the modelled environment at all poses. Thus one obvious proposal in commissioning is to add features to the environment and environment map such that some mapped features can be seen from all poses. The main concept, however, in our commissioning proposal requires us to find a *safe uncertainty bound* for a given pose estimate. The definition of such a bound, which may consist of several disjoint regions, is the bound within which no ambiguity can exist in the viewed environment for a given pose. We suggest two approaches to generate this bound: The first is to search for all regions of ambiguity over all poses. Due to the computationally intensive nature of the task, the search would have to be directed by a number of heuristics. For example, we could cluster parts of the modelled 3D environment into groups with similar 3D structure and then generate possible camera view directions which are cast out into the environment to generate mutually ambiguous pose regions in the environment where the view of the modelled world looks similar. If the actual pose uncertainty bound for a given pose overlapped with one of these regions, then a fail-safe would be required.

A second approach, which significantly reduces the search space, would be to run a Monte Carlo simulation of the the robot and its dual sensor systems operating within the intended map of the environment. The detail of this procedure is as follows:

1. Carefully model errors in the sensor systems and validate these models with real vehicle tests over a range of maneouvres (different turning circles and speeds, different locations within the environment). It is essential that we can show that the sensor models accurately reflect the real sensor data, in order to build a strong safety argument.

2. For each pose, simulate all (typical) paths used to reach that pose and thus determine the maximum pose uncertainty bounds likely to be encountered at that particular pose. The simulation will be a Monte Carlo simulation in that noise will be injected at every update step in the algorithm, where the noise is randomly sampled from the probability distributions modelling each sensor system. An important point here is that, in the simulation at least, we know the true data-to-model association and so given that no association errors occur, we can predict the likely maximum uncertainty bounds for any pose.

3. Shrink the predicted maximum uncertainty bounds at each pose due to any safety-margin infringements.

4. Over the space of possible poses, search for viewed ambiguity within these pose uncertainty bounds and if ambiguity exists, add strong features to environment map to remove the ambiguity.

One can imagine the end result of this process as follows: If we express the pose (x, y, θ) on a 3D graph, then as we move around in this space, a 3D uncertainty ellipsoid (or more complex shape) changes the size and direction of its axes. This ellipsoid at any 3D pose defines the uncertainty bound over which we have checked for and eliminated viewed environment model ambiguity from the operation of the MBL system.

5.2 Operational proposal

We now discuss how to avoid hazards once the system has been commissioned and is operational.

1. If pose uncertainty in any dimension exceeds the largest possible bounds computed in commissioning, then fail-safe, as we have not checked for pose ambiguities outside this range of pose variation.

2. If two different poses within the uncertainty bounds converge with a similar high match score, then fail-safe. Although this is not likely to happen, due to the commissioning stage, there may be an occlusion that makes one part of the environment match to the edgels as well as another part. An alternative to this is to weight the match scores for each pose by the pose probability density at that pose, so that for two identical vision match scores, the pose nearest to the mean of the current pose estimate will be selected. It is recommended that this only be used if there is a significant difference between the values of the two weighted match scores.

3. When in fail-safe state, uncertainty bounds may have grown large due to occlusion by for example a person walking through the camera field of view. If that person remained in the field of view, then the robot may not be able to lock onto the environment model in order to reduce pose uncertainty. In this case the robot would have to wait for a significant change in edgel data before rerunning the search for the best match within the uncertainty bounds. If this never happens, then the robot is effectively stuck in this state and human intervention will be required.

6 Demonstrating safety

To demonstrate safety it would be normal to produce a safety case, comprising a safety argument. Here the safety argument would have three elements:

1. Environment and algorithm definition make false lock (and hence a mis-matched filter) extremely unlikely (see section 5.1).

2. Algorithm is correctly implemented (outside the scope of this paper).

3. System behaves safely when the positional uncertainty is too large (see section 5.2).

This argument would need to be backed up by evidence from commissioning, software development, etc. Thus the approach described here gives the basis for demonstrating system safety.

7 Conclusions

In this paper, we have outlined the structure of model-based visual localisation algorithms and identified various scenarios in the operation of such systems that may lead to hazard conditions. We have then proposed solutions for both the commissioning phase of such systems and the operational phase in order to minimize the risk of the main identified algorithmic hazard, namely an over-confident pose estimate. The basic idea is to generate 'safe uncertainty bounds', within which there can be no ambiguity in the viewed environment. Thus mis-matched filter hazards are avoided and we can maintain a high overall system availability. Our approach also provides a clear and transparent way of demonstrating safety.

References

[1] H Araujo R Carceroni and C Brown. A fully projective formulation to improve accuracy of lowe's pose-estimation algorithm. *Computer Vision and Image Understanding*, 70(2):227–238, 1998.

[2] D. Lowe. Fitting parametrised three-dimensional models to images. *IEEE Trans. Pattern Analysis and Machine Intell.*, 13(5):441–450, 1991.

[3] D Lowe. Object recognition from local scale-invariant features. In *Proc Int. Conf. Computer Vision*, 1999.

[4] D. J. Pumfrey. *The Principled Design of Computer System Safety Analyses.* DPhil Thesis, Department of Computer Science, York University, UK, 1999.

[5] E Trucco and A Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[6] Bar-Shalom Y and Fortmann T E. *Tracking and Data Association.* Academic Press Inc., Boston, USA., 1988.