Multi-cue Vision, Novel Architectures, and High Integrity Concepts for Dependable Robots

Nick Pears, Jim Austin and John McDermid
Department of Computer Science
University of York
York, YO10 5DD, UK
email: nep@cs.york.ac.uk, austin@cs.york.ac.uk, jam@cs.york.ac.uk

Abstract

We describe three research areas at the University of York, UK, which impact on the reliability of robot operation in human environments. These are (i) how the use of multiple visual-cues, such as corners, lines, colours and textures can improve the adaptability and robustness of mobile robot visual navigation, (ii) how the use of novel, neurally motivated, architectures can improve system dependability, and (iii) how concepts from high integrity, or safety critical, systems engineering can help solve dependability problems. Firstly, we give a description each of these separately, and subsequently we conclude the paper by discussing how the ideas are related and how they may be integrated in a dependable robotic system.

1 Multi-cue computer vision

Our work in computer vision is particularly aimed at highly robust visual navigation (using a standard CCD camera) for mobile robots in indoor environments. Our focus on indoor environments means that planar regions in the scene will be common. In particular, floors which are planar to some approximation is a fundamental assumption. Apart from this ground planarity requirement, we impose no further restrictions and ultimately aim to be able to navigate in a broad range of indoor scenes. This is a challenging problem, since, as the vehicle moves around, the various visual cues that aid navigation disappear and reappear in the robot's visible environment. This has motivated us to use multi-cue systems to improve dependability, where, initially, we are looking at corner points, edges, colour and texture. Of these cues, corner points are the most fundamental in our system as they can be used to recover scene structure in near real-time.

In this section, we will describe a method of mobile robot navigation using visual ground plane detection. Corner points are tracked through an image sequence and grouped into coplanar regions using a method which we call an H-based tracker. This allows us to detect ground plane patches and the colour within such patches is subsequently modelled. These patches are then grown by colour classification to give a ground plane segmentation, which is fundamental to our navigation system. In remainder of this section, we detail the current operation of this system, but firstly we lay down some guiding principles for building dependable visual navigation systems.

1.1 Guiding principles

Our guiding principles for a framework for dependable visual navigation are as follows:

- Navigational information is bound up in a diverse range of visual cues, many of which are not always available due to scene variation. There must be a correspondingly diverse set of visual processing modules which make explicit this information.
- Wherever possible, modules must also provide a measure of how they think they are performing (The standard linear way of doing this is to provide a covariance matrix associated with a state).
- Minimal assumptions must be made concerning scene context and scale. Thus early processing modules must employ methods that can operate at multiple scales (eg wavelets, scale space, pyramid methods).
- The system should employ visual cues that compliment each other in terms of the constraints they provide for solving particular navigational tasks.
- When there are insufficient constraints to disambiguate a situation, multiple hypotheses must be maintained. This means that data must be processed using a set of possible models and a mechanism must be in place to monitor how well each

hypothesised model is performing, and to choose the most appropriate response.

- The system should degrade gracefully. As cues (or constraints) disappear, the robot's behaviour should degrade gracefully.
- Only task relevant information should be made explicit. For example, it is not necessary to build a full Euclidean reconstruction of the environment, when navigating across a scene.
- Within the framework, manual threshold selection should be eliminated and a minimal set of thresholds should be selected automatically. This could be based on the output of early visual processing, such examination of the spatial frequency content of the image.
- Overall, there must be means of understanding image context and using this context, the framework should be able to adapt in order to integrate visual cues appropriately.

Currently our multi-cue system only employs corner and colour cues in order to automatically segment the ground plane, with no *a priori* colour model. We are in the process of adding edge and texture cues to the system. The following sections describe how the system works, but first we outline some theory, which enables us to test whether tracked corner points are coplanar.

1.2 Background theory

(i) Planar projective invariants:

One approach used to detect coplanar points is the direct use of projective invariants, as exemplified by [8]. This uses the fact that if we have four collinear points in the scene, say a, b, c, d, then a ratio of ratios of distance (the cross ratio) is a projective invariant. This fact can be extended to five points in a general position on a plane, since, using projective constructions, we can get two sets of four collinear points, which are invariant if and only if the original five points are coplanar. Figure 1 shows this construction. The point groupings l, d, k, a and l, c, i, b give the two invariants as:

$$I_1 = \frac{d_{lk} d_{da}}{d_{la} d_{dk}}, \quad I_2 = \frac{d_{li} d_{cb}}{d_{lb} d_{ci}} \tag{1}$$

(ii) Plane to plane projectivity

Early work on exploiting coplanar relations has been presented by Tsai and Huang [9], Longuet-Higgins [5] and Faugeras and Lustman [2]. We summarise the coplanar relation as follows: If a set of corner features in the scene lie in a plane, and they are

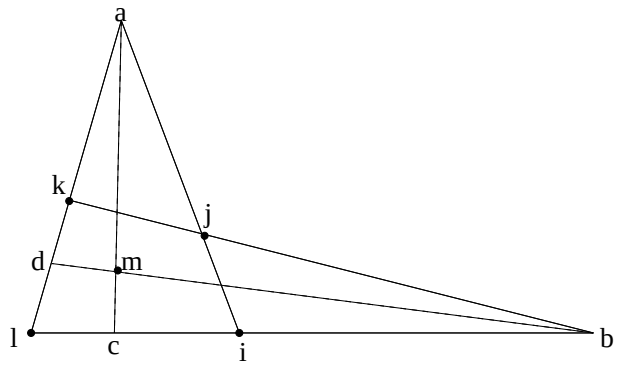

Figure 1: Projective construction for 5 point planar invariant

imaged from two viewpoints, then the corresponding points in the two images (separated by k frames) are related by a plane-to-plane projectivity or homography, **H**, such that:

$$\lambda \mathbf{x_i} = \mathbf{H} \mathbf{x_{i-k}} \tag{2}$$

where $\mathbf{x}$ represents a homogenous image coordinate $(x, y, 1)^T$, **H** is a 3 by 3 matrix representing the homography and λ is a scalar. Since this equation is valid up to a scale factor, **H** has only eight degrees of freedom, and it is normal practice to choose λ such that element h_{33} in **H** is set to unity. Eight degrees of freedom requires that we have four corresponding coplanar features in general position (no three collinear), since each pair of corresponding points then provides two independent constraints, and **H** can be determined by standard linear methods.

Equation 2 suggests a method of grouping corner features into coplanar sets. Namely, if we can select a set of four coplanar corresponding point pairs which are in a sufficiently general configuration in both images (each point is unique and no three are collinear), then **H** can be computed and used to check whether other points in the scene lie in the same plane.

1.3 Navigation using an H-based tracker and colour cues

In our system, we aim to use primarily **H** relations to detect the ground plane and planar projective invariants to help bootstrap this process. We call our system an *H-based* tracker. We now give a top down description of our algorithm.

H-based tracker algorithm

We first run an initialisation stage where we

1. Detect corners using a standard (Plessey-Harris) corner detector.

2. Track these points over n frames using a Kalman filter, with a standard motion model (of velocity) and cross correlation to determine matches.

This generates a reasonable disparity between corresponding corner points in frame 1 and frame n before attempting to estimate H. In subsequent frames we search for correspondences between frame i and frame $i-n$. Thus from frame $n+1$, we run the H-based tracker. The key modification from the basic tracker used in the initialisation stage is that two process models are employed in the state prediction and data association stages of the tracker. The first stage is the standard motion model used in the initialisation stage. The correspondences generated from this allows bootstrapping of the ground plane **H** by testing a population of putative **H** matrices. This is a sample consensus approach similar to RANSAC [3], but the samples are not selected randomly. H matrices can then be used as a model to predict and associate measurements in an iterative manner. To summarise these steps we

1. Bootstrap the system by computing a population of putative **H** matrices for the corner points which have their vertical component of image motion in a downwards direction (i.e. are below the horizon line).

2. Select the dominant **H** model i.e. that which verifies the largest number of corner associations. The corners points that are verified are deemed *inliers*.

3. Recompute **H** by applying orthogonal least squares to the inliers.

4. Retest the data associations of corner points to tracks using the least squares estimate of **H** to get an updated set of inliers.

5. Iterate around the previous two steps until the number of inliers stabilises.

6. Check that the coplanar points extracted are ground plane points by computing the plane normal.

It is possible to remove all of these coplanar corners, and repeat the whole procedure to find further significant co-planar corner groupings in the scene. Indeed, it may be necessary to do this if we find that the dominant plane can not the ground plane due to the computed plane normal. In subsequent frames, we simply sample from the group of points that are deemed to be in the ground plane and choose a suitable selection of basis points to compute a new **H**.

We now describe our method of combining corner and colour cues to extract a first estimate of the navigable image region. At present, our implemented method is fairly basic, but our results illustrate how effective the general technique of combining cues can be. Our algorithm is as follows:

1. Corner points are tracked and classified as either *on the ground plane* or *off the ground plane*, using the H-based tracker described above.

2. Ground plane corner points are then grouped into one or more *ground plane patches*. These are collections of ground plane points where the distance to the nearest neighbour ground plane point within a patch is below a threshold.

3. A bounding polygon for these corner points defines an image region in which the colour space of the ground plane is modelled. (Currently we use a simple bounding ellipse in normalised colour space.)

4. Thus the region(s) classified as the ground plane (i.e. within the bounding polygons) can then be grown by classifying small image regions as either *ground plane colour* or *not ground plane colour*. This process only needs to operate on the image below the extracted horizon line.

1.4 Ground plane extraction results

Figures 2 and 3 illustrate the H-based tracker and colour region growing processes respectively. (For clarity, figure 2 only shows the strongest corner feature within a 20 by 20 pixel window.) The corners marked with a cross have been matched to previous positions, as shown by their trailing lines, and have been used to estimate the **H** matrix by orthogonal least squares. Other crosses, which are also inside the bounding polygon, are corners not used in the **H** matrix estimation, but whose correspondences lie within the matching ellipse associated with this **H** matrix, and so are deemed to lie on the same plane.

Once the bounding polygon has been extracted, the colour space of the ground plane is sampled, and a region growing algorithm expands the polygon to edges in the image where there is a change in colour. Figure 3 highlights the final ground plane region extracted from this technique. Note that we can ignore regions segmented above the extracted horizon line, which is shown, along with the epipole, in the upper half of figure 2. Notice how the ground plane detected can

Figure 2: Tracked and grouped corners

Figure 3: Ground plane region extraction

extend into regions there are no corners due to the texture gradient of the imaged carpet. Obviously, the technique works well in this particular case, because the ground plane has sufficient corner features, and the colour space of the ground plane is unimodal (i.e. homogenous). However, in further work we aim to develop a range of techniques, a selection of which can be automatically deployed depending on the image context.

2 Novel architectures

The second idea we describe, which is relevant to dependable robots, is the use of fine-grained fault-tolerant architectures. Here at York, we have been developing systems based on neuraly plausible architectures, that is systems whose design is directed by the low level operation of biological computation systems. These systems appear to have certain advantages for dependable operation.

A major feature of neural architectures is the use of threshold logic (TL) rather than conventional logic systems found in current computer systems. These gates operate by summing the inputs to the gate, and applying a threshold to determine the output. In such a system, a logical AND operation is implemented by applying a threshold equal to the number of inputs to the gate, and a logical OR operation is implemented by applying a threshold of 1. This enables the systems to implicitly tolerate failures and continue to function under failure. In the case of an AND gate, if an input to a gate fails, the gate can still operate if the threshold is reduced.

Clearly, an approach to robustness that operates at the gate level, rather than at higher levels of abstraction, may result in higher overall dependability, and better use of resources, since errors are masked at the earliest possible stage of processing. To use TL computer systems new architectures and algorithms are needed to exploit the advantages that they provide. Work at York has developed neural architectures for image analysis [6], that are capable of fault tolerance [1], and which exploit the advantages of TL.

3 Safety critical issues

The third and final research area we discuss is the use of safety critical analysis techniques to robotic applications.

If a robot were to be used in a safety critical application, for example when it operates in a human environment where it can potentially cause injury, is necessary to prepare a safety case to demonstrate that it is sufficiently dependable. This poses several challenges. First, with software-based systems, it is normal to require software to behave predictably, which conflicts with the flexibility needed in robotic applications. Second, it is hard to define requirements, and to characterise what is meant by safe behaviour, as the robot must be able to operate in any environment and to recognise and avoid hazardous situations, such as unprotected drops. To address these issues requires the design to be altered so the robot is "aware" of hazards, and to program in "safe" behaviours in the presence of hazards. The safety case then needs to show that the robot will (with a high degree of certainty) adopt a safe behaviour in the face of uncertainty in the interpretation of sensory information.

3.1 Approach to Demonstrating Safety

There are many strategies for demonstrating safety, e.g. showing that a new design is "no worse than" an existing system. We have developed techniques for arguing about safety, including documenting these common forms of argument as patterns [4]. However, with novel systems such as autonomous robots, we cannot adopt such a strategy. Instead we need to argue from first principles, and show that the robot will behave safely under all circumstances. This involves showing validity of the requirements, and that the implementation meets those requirements.

With the sorts of robots being considered requirements are problematic. The most practical approach seems to be to ascribe to actions, or sequences of planned actions, within specific scenarios, a probability which indicates the likelihood that the actions are hazardous. In broad terms we can then classify planned action sequences as

- Almost certainly hazardous.
- Almost certainly safe.
- Uncertain status.

The requirements are then for the robot to plan a sequence of actions, for example mobile robot maneuvers, which is safe to an acceptably high degree of probability. In the course of executing its maneuvers, the robot may make new observations, which make previously safe planned maneuvers, hazardous. In this case, new routes and maneuvers must be planned, and their safety evaluated, and dynamically re-evaluated in the light of new observations. Ultimately the robot must stop if it cannot find a safe route. The design must therefore be modified to incorporate this form of "fail safe" architecture.

Interpretation of a robot's sensory data is always subject to some degree of uncertainty, and the threshold for classification of actions as hazardous (for example, maneuvering near drops, steep slopes, and areas around humans), should be biased to minimise false negative "hazardous" classifications, at the expense of an increase in the number of false positive classifications. Note that there will normally be an acceptable level of failures, e.g. one every 10,000 hours of operation, and this will need to be used to guide the extent to which the classification algorithms need to be biased.

In practice, biasing the algorithms to avoid hazards means that the robot is more likely to stop (become unavailable). Some trade-off studies will be necessary in order to show that there is a satisfactory balance between availability and the likelihood of unsafe movement of the robot. To support this, statistical analysis will be needed on the type of algorithms described in section 1 (modified to provide the required bias), and the distribution of features in the environments where the robots will be used. This analysis will be used to evaluate the probability of unsafe misclassification, i.e. classifying an area as safe when it is not.

The design will also be required to be fault-tolerant, as outlined in section 2, so the algorithms will need to be modified so that the robot stops when there is insufficient remaining operational hardware to classify scenes with acceptable accuracy. Showing that the requirements have been implemented correctly is largely a software engineering issue, but is rather non-standard due to the nature of the requirements and algorithms. However it is further complicated by the need for the algorithms to be fault-tolerant. Here we need to assess the designs to ensure that the "fail safe" bias, adopted to handle the uncertainties in assessing the environment, are not affected by the strategies for tolerating hardware failures. This will require failure-directed verification activities, e.g. testing to show that the algorithms behave the same way under failure as they do under when all the sensors are working correctly. Overall, the safety case strategy will be based on an assessment of the probability of erroneous classification due to the uncertainties in the algorithms, and the likelihood of uncontrollable hardware failures.

3.2 Broader Issues

The tradition in industries developing safety critical systems is to introduce new technologies progressively, e.g. using novel materials in low criticality components on aircraft before using them for major wing components. This is intended to reduce the risk that imperfect analysis of the design will allow through residual design faults which permit unsafe behaviour. A similar approach would be required for the form of robot described in this paper.

In fact we have already assumed that it is acceptable for the robot to stop, under uncertainty or failure. In itself, this is a "low risk" application. It is prudent to try out the approach on such systems before using the technology in more demanding applications, e.g. where the robots have to continue operating (albeit at greater risk of hazardous behaviour) after failures. However the first stage is to try out the classes of design modification outlined above on a robot used in a non-critical application in order to show that the suggested strategy is valid.

4 Conclusions

We have been developing novel Computer Vision techniques, which can be used to enable robots to navigate in unfamiliar territory. These algorithms will need to be adapted to bias the system to fail safe if such a robot is to be used in safety critical applications. We have also considered approaches to implementing these algorithms, and noted the need for fault-tolerance mechanisms to be introduced.

We have discussed possible approaches to showing the safety of such systems, and noted some of the general issues in introducing such new technology into critical applications.

The need to show safety may be seen as an imposition, but it is possible that it could be used more constructively. We have previously demonstrated that it is possible to use safety cases as an active tool in safety monitoring [7] for relatively conventional systems. It would be interesting to explore the possibility of doing the same with a robot, using the safety case to define a "safe envelope" in which the robot could operate. This would be an interesting avenue of research for the future.

References

[1] G Bolt and J Austin. Operational fault tolerance of the adam neural network system. In *Second Int. Conf on Neural Networks*, pages 285–289, 1991.

[2] O Faugeras and F Lustman. Motion and structure from motion in a piecewise planar environment. *Int. Journ. Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[3] M A Fischler and R C Bolles. Random sample consensus: apardigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[4] T P Kelly and J A McDermid. Safety case patterns-reusing successful arguments. In *IEE Colloqium on Understanding Patterns and their Application to System Engineering, London, IEE*, 1998.

[5] H C Longuet-Higgins. The reconstruction of a plane surface from two perspective projections. *Proc. Royal Society London*, B227:399–410, 1986.

[6] C Orovas and J Austin. Associative cellular neural networks for pattern recognition. In S Wermter and R Sun, editors, *Hybrid Neural Systems*. Springer, 2000.

[7] Y Papadopoulos and J A McDermid. Safety-directed monitoring using safety cases. *Int. Journ. of Condition Monitoring and Diagnostic Engineering Management, COMADEM International-UK*, 1(4):5–15, 1998.

[8] D Sinclair and A Blake. Quantitative planar region detection. *Int. Journal of Computer Vision*, 18(1):77–91, 1996.

[9] R Tsai and T Huang. Estimating three-dimensional motion parameters of a rigid planar patch. *IEEE Trans. Acoustics, Speech and Signal Processing*, 29(6):1147–1152, 1981.