Fast Visual Mapping for Mobile Robot Navigation

Andrew Howard and Les Kitchen
Department of Computer Science
University of Melbourne
Parkville, Victoria 3052
Australia

Abstract — **In this paper we describe a vision-based mapping system for mobile robot navigation that is fast, simple and requires very modest computational resources. The system comes in two parts: a visual range subsystem and a map building subsystem. The visual range subsystem acts as a kind of 'virtual range sensor': it takes as input a stream of images and produces as output a stream of range-and-bearing measurements, comparable to that produced by a scanning laser range finder. The map building subsystem then fuses this data stream into a coherent grid-based representation of the environment. The system has been implemented and extensively tested on Robot J. Edgar, a small mobile robot that roams the corridors of an unmodified office building.**

I Introduction

One of the key difficulties facing any mobile robot is obtaining accurate, relevant and timely maps of the environment it inhabits. Over the years, a great many sensing modalities have been applied to this task, including vision-based systems such as geometric modelling, stereopsis and structured lighting, and non-vision systems such as sonar, radar and time-of-flight laser range-finders [1]. Unfortunately, vision-based systems have generally been distinguished by low speed and enormous computational requirements, whilst non-vision systems have been distinguished either by poor resolution (in the case of sonar) or high cost (in the case of radar and laser range-finders). In this paper, we describe a vision-based mapping system that is fast, simple and requires relatively modest computational resources. The system comes in two parts: a visual range subsystem and a map building subsystem. The visual range subsystem acts as a kind of 'virtual range sensor', which takes as input a stream of images and produces as output a stream of range-and-bearing measurements. The map building subsystem then fuses this data stream into a coherent grid-based representation of the environment, known as an 'occupancy map' [2]. The motivation for this design arises from the observation that very good environment maps can be obtained by combining scanning laser range finders with grid-based data fusion techniques [3]. In our system, the laser range finder is replaced with a vision-based virtual range sensor that produces comparable output. The system uses standard vision components – a camera, frame grabber and computer – and as such is low cost. For example, the results presented in this paper where obtained using a inexpensive monochrome camera fitted with a plastic lens, a 10-year-old frame grabber and 33MHz 486-based computer.

In order to make the visual mapping problem computationally tractable, we have made use of certain assumptions about the environment the robot inhabits (these assumptions are detailed in Section II). While these assumptions hold true for most indoor environments, they are violated in natural, outdoor environments. The visual mapping system presented in this paper is therefore only suitable for use in indoor environments. In addition, we exploit the fact that the output of the visual range subsystem does not need to be entirely error free: during the map building process, measurements from multiple viewpoints are combined in such a way that errors tend to get 'washed out'. As a result, the visual range subsystem can afford to take a 'dumb-but-fast' approach to image processing, in which some degree of accuracy is sacrificed for the sake of speed.

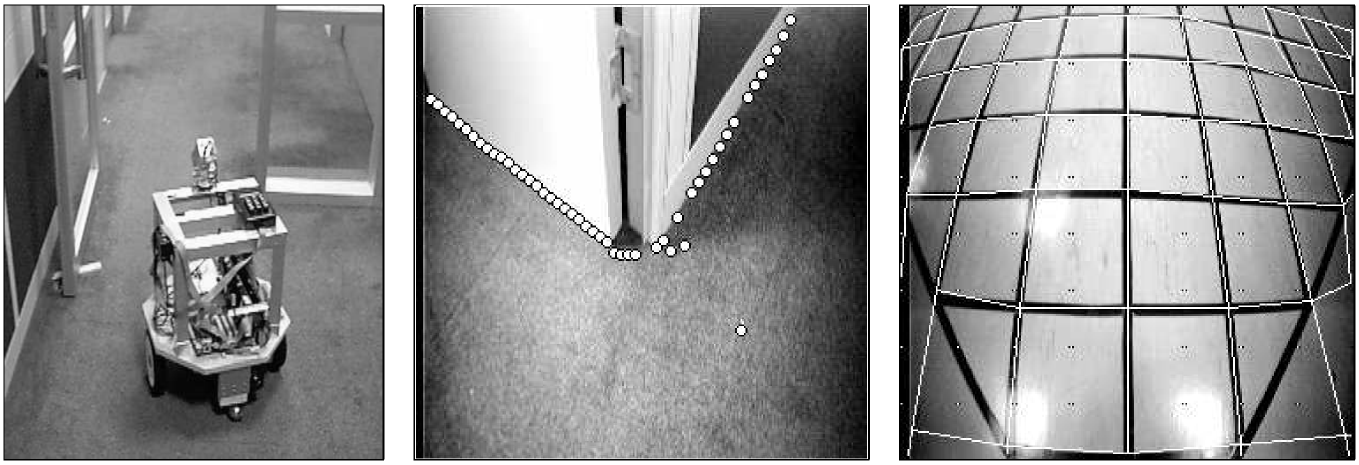
The visual mapping system described in this paper has been implemented on Robot J. Edgar, a small mobile robot that roams the corridors of an unmodified office building. Robot J. Edgar is shown in Figure 1. The robot uses the generated occupancy maps for both obstacle avoidance [4] and landmark recognition. Note that in addition to its single camera, Robot J. Edgar can be equipped with a rotating sonar head. The range data from this head can be fused by the same map building subsystem [5]. Generally, however, the data rate from the sonar sensor is so low that it can be discarded without any loss of performance. Robot J. Edgar displays remarkably robust behaviour, having seen many hours of service without significant failure of the visual mapping system.

The structure of the paper is as follows. In Section II we make explicit the assumptions we have made about the robot's environment. In Sections III and IV we describe, first in general terms and then in detail, the two subsystems that make up the visual mapping system. Finally, in Section V, we present results from experiments with Robot J. Edgar.

II Properties of the Environment

The visual mapping system described in this paper was designed for use in indoor environments. As has been noted previously [6], such environments generally have a number of properties, which we will list as *assumptions*:

- *Ground plane assumption.* There exists a well defined ground plane.



(a) (b) (c)

Fig. 1: (a) Robot J. Edgar. (b) Image of a doorway taken with Robot J. Edgar's camera. The while circles mark the location of the carpet boundary. Note the mislocated boundary point, due to a scrap of paper on the floor. (c) Calibration image. The dark lines of the calibration pattern are placed at 20 cm intervals on the floor. The white lines show the calculated calibration pattern.

- *Projection assumption.* The ground is covered by carpet (or some similar floor covering), such that all carpeted areas are traversable by the robot. Conversely, all non-carpeted areas are obstacles.
- *Texture assumption.* The carpet has a relatively even texture.

The ground plane assumption recognises that indoor environments rarely have uneven floors; rather, they are made up of one or more well defined planes connected by lifts or stairs. This assumption allows the environment to be modelled by a set of two-dimensional maps.

The projection assumption allows us to transform the very difficult problem of locating arbitrary three-dimensional obstacles into the much simpler problem of locating non-carpet areas. Of course, this assumption does not always hold true. For example, a piece of paper placed on the carpet will be treated as an obstacle, when strictly speaking, it is not. Similarly, the boundary between dissimilar floor coverings must be regarded as an obstacle. This does lead to some difficulties on Robot J. Edgar, which cannot detect the doorways connecting carpet-covered corridors with linoleum covered laboratories.

Another possible violation of the projection assumption arises whenever overhanging objects, such as tables and chairs, are present. For example, the area beneath a table or a chair may be carpeted, but this space is only traversable if the robot is shorter than the table. It is apparent that the projection assumption is really an assumption about the combined robot/environment system. The height of Robot J. Edgar is such that it will fit comfortably beneath an ordinary table, so the projection assumption is not violated in this case. The robot is *not* short enough to pass beneath chairs, however, leading to a violation of the projection assumption. Fortunately, this particular violation makes no functional difference (i.e. it does not affect robot behaviour). Chair legs *will* be detected as obstacles, and since they are too closely spaced for the robot to fit between, the robot will never attempt to move under a chair.

The final assumption is the texture assumption. This is used by the visual range subsystem to simplify the task of classifying carpet and non-carpet areas. In areas with highly textured floor coverings, such as tiles or patterned carpet, the visual range subsystem will fail.

III Visual Range Subsystem

A Overview

In principle, the task of the visual range subsystem is to determine the range-and-bearing of any obstacles in the robot's environment. In practice, what the visual range subsystem really determines is the location of the *carpet boundary.* While the former task is conceptually difficult and computationally intractable, the latter task is quite straight-forward. Furthermore, if the projection assumption noted in the previous section holds, the tasks are equivalent.

Given an image, the visual range subsystem must first locate the carpet boundary *in the image.* Many techniques could be applied to this problem, such as texture analysis or region growing, but we have chosen to employ an edge-based method for the sake of speed and simplicity. The method works as follows. Consider Figure 1, which is an actual image acquired by J. Edgar's camera. Inspecting any column of pixels in this image, it is apparent that the variation in pixel intensity in the carpeted areas is quite low. At the carpet boundary, however, there is a sharp discontinuity. This boundary can therefore be located by applying a one-dimensional edge filter to each pixel column, starting from the bottom, and recording the location of the first significant edge. Note that this procedure *does not* work if we apply the edge-filter to pixel *rows*; here, the edges may correspond either to the carpet boundary or to an occluding surface, such as a doorframe. On pixel columns, the procedure will always work, so long as there are no overhanging objects and the texture assumption holds true.

Note also that the bottom row of pixels in the image must correspond to carpet. If, for example, the robot is extremely close to an obstacle, such that there is no carpet visible, the location of the boundary will be incorrectly determined. As a result, this method tends to work best when the camera is tilted downwards, looking at the carpet directly adjacent to the robot.

Given the location of the carpet boundary *in the image*, the next step is to determine the location of the carpet boundary *in the world*. If the ground plane assumption holds, then the geometry of the camera dictates that each pixel in the image corresponds to exactly one point on the ground plane. Therefore, it is possible to establish (through a calibration procedure) a function that maps each point in the image to a point on the ground plane. These points will naturally be camera-centered. Given the location of the carpet boundary in the image, this function can be applied to determine the camera-centered coordinates of the carpet boundary.

The overall behaviour of the visual range subsystem is to produce, for each input image, a set of camera-centered points describing the location of the carpet boundary. There will, of course, be errors is the data due to noise in the image or occasional features in the carpet. However, since the map building subsystem uses a Bayesian data fusion scheme to combine data from multiple viewpoints, such errors tend to get 'washed out'. Our test-bed, Robot J. Edgar, typically experiences error rates of the order of 10% (that is, 10% of boundary points are not correctly located) and must combine measurements from at least three frames to produce reliable maps.

B *Algorithm*

Finding the entire floor boundary on a full 512×512 or 256×256 image is computationally expensive, so the visual range system settles for finding a limited number of boundary points. A number of pixels columns are used to 'sample' the image. The following algorithm is applied to each of the selected columns:

1. Apply a one dimensional edge filter, starting from the bottom of the image.
2. Record the image location the first significant edge.
3. Using a lookup table, convert the image location to camera-centered coordinates.

Note that the rate at which boundary points are found depends purely on the structure of the input images (close boundaries will be found more quickly than distant boundaries) and on the processing power available to perform the edge filtering process. For example, the visual range subsystem could inspect 1 image frame per second, and find 200 boundary points, or it could inspect 10 frames per second, and find 20 boundary points in each. In either case, the rate at which boundary points are located is the same – 200 points per second. To ensure that the information being generated by the visual range system is up-to-date, we generally choose to process a large number of frames per second (10 frames per second is typical), and extract only a small number of points from each. With a 33Mhz 486 PC, rates of 320 boundary points per second can be easily achieved, with enough processing power left over to attend to other tasks, such as map building.

The one dimensional edge filter that is applied to each column is chosen for its speed. In general, for some filter F, the filter response $R(n)$ of the n^{th} pixel on in a column can be calculated as follows:

$$R(n) = \sum_{i=-\delta}^{+\delta} F(i) I(n+i). \tag{1}$$

$I(n)$ is the intensity of the n^{th} pixel. The particular filter used has the following form:

$$F(i) = \frac{1}{2\delta} \begin{cases} -1 & -\delta \leq i < 0 \\ 0 & i = 0 \\ +1 & 0 < i \leq +\delta \end{cases} \tag{2}$$

For this filter, it is straight-forward to show that the response at $R(n+1)$ can be calculated from the response at $R(n)$ as follows:

$$R(n+1) = R(n) - \frac{1}{2\delta}\{I(n) - I(n+1) + \tag{3}$$
$$I(n-\delta) + I(n+\delta+1)\}. \tag{4}$$

Thus, instead of re-evaluating the filter response for each point (which involves inspecting 2δ pixels), the filter response can be computed incrementally (inspecting at most 4 pixels for each point). Use of this incremental technique yields a significantly faster algorithm.

The first pixel for which the absolute value of $R(n)$ exceeds some preset *filter threshold* will be classified as a boundary point. Appropriate values for this threshold, together with the filter size, must be determined experimentally.

Once a boundary point has been located in the image, the location of the corresponding boundary point in the world must be determined. The visual range subsystem makes this determination using a lookup table and linear interpolation. Specifically, a lookup table lists the camera-centered coordinates of each of a number of equally spaced image points. Intermediate points points are calculated from these values using linear interpolation. The lookup table is constructed during a calibration process, in which an image of a known calibration pattern is used to estimate the lookup table entries. Figure 1 shows a typical calibration image, in which the estimated calibration pattern (calculated from the lookup table entries) is superimposed on the real calibration pattern. The variation between these patterns is never greater than 2cm, which is sufficiently accurate for mapping purposes. Note that we have chosen not to use the standard pinhole camera model, as this allows for greater flexibility in lens selection. Robot J Edgar, for example, uses a wide-angle lens to which the pinhole model does not apply.

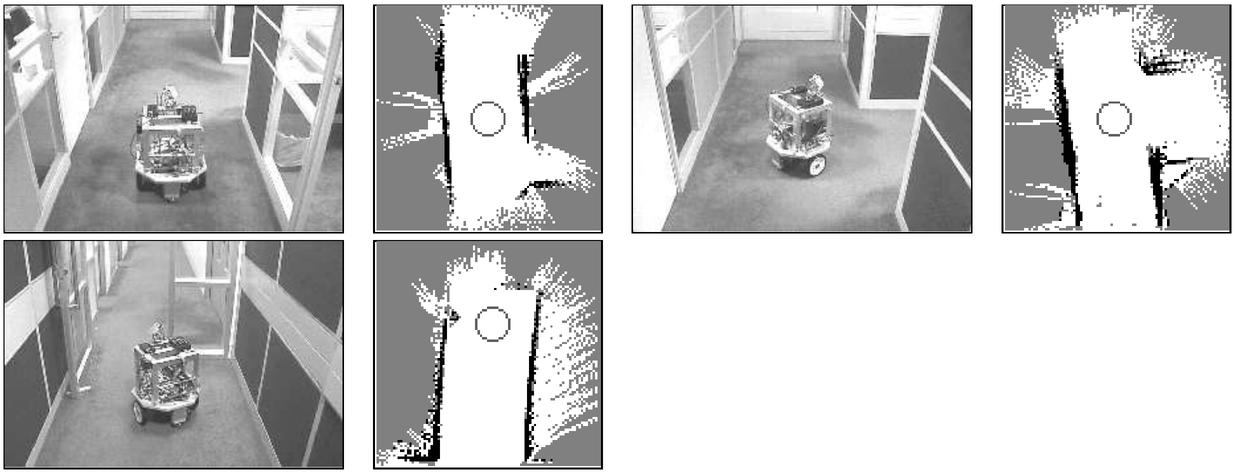


Fig. 2: Occupancy maps generated at a number of different locations. Occupied areas are indicated by dark pixels, unoccupied areas by light pixels and unknown areas by gray pixels.

IV Map Building Subsystem

A Overview

The map building subsystem fuses the data from the visual range subsystem to form an *occupancy map* of the robot's environment [2, 7]. The occupancy map is a grid-based map in which each cell has some value indicating the probability that it is occupied, i.e. that there is some obstacle at that location. A fast log-likelihood formulation of Bayes Law is used to fuse the incoming data to arrive at a single probability value for each cell. The map building subsystem may construct either a *global* map, which is fixed to the environment and through which the robot moves; or it may construct a *local* map, which is fixed to the robot, and around which the environment moves. The only difference between these two maps is the way in which data from multiple viewpoints is combined. In either case, however, the correspondence between different viewpoints is determined using odometric data. Figure 2 shows some typical occupancy map constructed by Robot J. Edgar.

The procedure for updating the map is as follows. Given a set of points that describes the location of the carpet boundary in camera-centered coordinates, the map building subsystem must first transform these points into map coordinates. The specific transformation used will depend upon the nature of the map being constructed. For global maps, the transformation must consider the robot's absolute pose, wherease for local maps, only changes in pose need to be considered. With a set of points in map coordinates, the map building subsystem can update the relevant cells within the map. Cells that lie along the carpet boundary are updated to reflect the fact that these cells are probably occupied; cells that lie *between the robot and the carpet boundary* are updated to reflect the fact that they are probably unoccupied (otherwise the boundary would not be where it is); and cells that lie beyond the boundary are ignored – the data carries no information about the occupancy state of these cells. The logic of this update scheme is exactly the same as that for updating the map with sonar or laser range finder data [2, 5].

The log-likelihood formulation of Bayes Law (defined in the next section) can be thought of as a simple evidence accumulator: every measurement indicating that a cell is occupied increases the cell's accumulator, whilst every measurement indicating the opposite decreases the accumulator. The fusion process can be 'tuned' by selecting different increment and decrement values: big values indicate high confidence in each measurement, small values indicate low confidence. If, for example, the visual mapping subsystem produces very noisy data, low values should be chosen. The robot will then have to make a great many measurements before it can be confident as to whether or not a particular cell is occupied. One consequence of this is that the robot will have to move quite slowly, lest it crash into an obstacle before it has decided whether or not the obstacle is real. On the other hand, if the error rates from the visual range subsystem are low, the increment and decrement values can be large, few measurements will be required to determine whether or not a particular cell is occupied, and the robot can move much faster.

B Algorithm

The occupancy map is implemented as a two-dimensional array in which each cell represents some small region of the environment. We typically let each cell represent a region 4cm square. The occupancy value of all cells is initially set to zero, which corresponds to a prior probability of 0.5 (see below). Given a carpet boundary point from the visual range subsystem, the map update algorithm is as follows:

1. Using the robot's odometrically determined pose, determine the map coordinates of both the robot and the carpet boundary point.
2. Increment the occupancy value of the cell corresponding to the carpet boundary point.

3. Decrement the occupancy value of all the cells between the robot and the carpet boundary point.

Note that, strictly speaking, only those cells that lie in the region that can be seen by the camera should be updated. That is, areas that are too close to the robot to be seen by the camera should not be updated. Such an update scheme can be implemented in a straight-forward manner by making use of the camera calibration data to determine which cells lie within the camera's field-of-view at any given time, and only updating those cells.

This very simple algorithm is based on a log-likelihood formulation of Bayesian data fusion [8]. In this formulation, the occupancy value of a cell interpreted as the logarithm of the likelihood that the cell is occupied, i.e.

$$L(o) = \log \frac{p(o)}{p(\neg o)}. \quad (5)$$

Where $L(o)$ is the occupancy value and $p(o)$ is the probability that the cell is occupied. While probabilities are bounded on the domain $[0, 1]$, log-likelihoods take values on the domain $[-\infty, +\infty]$, with 0 corresponding to a probability of 0.5. With this definition, it is straight-forward to show that the standard Bayesian update rule, which gives the probability that a cell is occupied, given some measurement m:

$$p(o|m) = \frac{p(m|o)}{p(m)} p(o), \quad (6)$$

becomes

$$L(o|m) = L(m|o) + L(o). \quad (7)$$

The advantage of this rule is that cells can be updated with a single addition, rather than multiplication; no normalisation is required. Positive values of L indicate that there is some evidence that the cell is occupied; negative values indicate that there is some evidence that the cell is unoccupied.

V Experiments

The visual mapping system described in this paper has been implemented on Robot J. Edgar. This is a small mobile robot with twin drive wheels, a 33MHz 486-based computer, a monochrome camera with wide-angle lens, a pan head, a frame grabber and miscellaneous supporting hardware. The robot is entirely autonomous and has sufficient battery power to operate continuously for over an hour. The visual mapping system is written in C++. A typical set of performance characteristics are as follows:

Visual range subsystem	
Input rate	10 frames/second
Output rate	320 points/second
Processing time (ave)	50 ms/frame
Map building subsystem	
Map size	384×384 cm
Map resolution	4 cm
Processing time (ave)	< 0.02 ms/point

At this frame rate, the visual mapping system is using about 50% of the total CPU capacity, leaving ample time for other subsystems to execute. In separate experiments [4], we have found that the visual mapping subsystem can support reliable autonomous obstacle avoidance at speeds of up to 30cm/s, which is the maximum speed achievable by Robot J. Edgar.

Figure 2 shows a typical set of occupancy maps generated during a global navigation experiment in the corridors of an unmodified office building. Doorways and intersections are clearly resolved in these map. Note that some maps have a slight 'bending' effect. This is a result of cumulative drift in the odometry data used to compute the robot's pose. No attempt has been made to correct this effect, since Robot J. Edgar's navigation systems are immune to such errors [9].

VI Conclusion

The raw occupancy maps produced by the visual mapping system described in this paper provide an excellent source of data for both low-level obstacle avoidance problems and high-level landmark recognition problems. The system has in fact been used for both purposes on Robot J. Edgar, where it has worked remarkably well. In the future, we hope to explore some alternative visual ranging algorithms that do not rely on the texture assumption, and are therefore applicable to a somewhat wider range of environments, such as those with tiled-surfaces.

References

[1] R A Jarvis, "A perspective on range finding techniques for computer vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, 1983.

[2] Alberto Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception", in *Proceedings of the Sixth Conference on Uncertainty in AI.* July 1990, Morgan Kaufmann Publishers, Inc.

[3] R A Jarvis and Alan Lipton, "Go-2:- an autonomous mobile robot for a science museum", in *Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision*, 1996, pp. 260–266.

[4] Andrew Howard and Les Kitchen, "Vision-based obstacle avoidance", Tech. Rep. 35, Department of Computer Science, University of Melbourne, 1997.

[5] Andrew Howard and Les Kitchen, "Generating sonar maps in highly specular environments", in *Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision*, 1996, pp. 1870–1874.

[6] Ian Horswill, "Polly: a vision-based artificial agent", in *Proceedings AAAI-93*, 1993.

[7] Hans Moravec, "Sensor fusion in certainty grids for mobile robots", *AI Magazine*, pp. 61–74, Summer 1988.

[8] James O. Berger, *Statistical Decision Theory and Bayesian Analysis, Second Edition*, Springer-Verlag, 1985.

[9] Andrew Howard and Les Kitchen, "Navigation without localisation: a reactive network approach", in *Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision*, 1996, pp. 873–877.