CIMCA 2004 Proceedings/ISBN :1740881885 M. Mohammadian (Ed.)
12-14 July 2004, Gold Coast - Australia

Monocular Vision as a Range Sensor

T.Taylor[1], S.Geva[2], and W.W.Boles[3]
[1]School of Information Systems,
[2]School of Software Engineering and Data Communications,
[3]School of Electrical and Electronic Systems Engineering,
Queensland University of Technology, Brisbane, Australia
E-mail: {t.taylor, s.geva, w.boles}@qut.edu.au

Abstract

One of the most important abilities for a mobile robot is detecting obstacles in order to avoid collisions. Building a map of these obstacles is the next logical step. Most robots to date have used sensors such as passive or active infrared, sonar or laser range finders to locate obstacles in their path. In contrast, this work uses a single colour camera as the only sensor, and consequently the robot must obtain range information from the camera images. We propose simple methods for determining the range to the nearest obstacle in any direction in the robot's field of view, referred to as the Radial Obstacle Profile. The ROP can then be used to determine the amount of rotation between two successive images, which is important for constructing a 360º view of the surrounding environment as part of map construction.

1 Introduction

The most fundamental problem facing an autonomous robot using vision is obstacle detection – a point that is often made by researchers [1,2]. However, despite over 30 years of research, there is no agreement within the Computer Vision community on the best approach to achieve this task (see [3] for a review of progress to date), and a precise definition of obstacle detection is surprisingly difficult [4].

Once a robot can detect obstacles and navigate reliably, it can construct a map of its environment. Mapping can be done using a well-established technique such as occupancy grids [5]. This approach was originally designed for sonar data, and has been used successfully, e.g. [6]. Sonar data takes the form of range sweeps, i.e. radial distance and angle measurements to obstacles relative to the centre of the robot, which can be produced by a ring of sonar devices. However, the signals require suitable surfaces to bounce off; the range data often suffers from noise and multiple reflections; and the maximum range can be quite limited.

Using vision, i.e. a camera, as a range sensor is one way to overcome the problems with sonar. Other advantages of vision over sonar are that it can detect small obstacles; it is a passive sensor (lower power, undetectable); and it can distinguish between obstacles based on colour.

A great deal of work has been done on stereo vision, e.g. [7], but for reasons of size and cost this research is based on a single colour camera, i.e. monocular vision. Stereo cameras can be used to obtain depth information using disparity maps, and differences between the two images can also be used to detect obstacles. However, with monocular vision different approaches have to be used to obtain range information.

Although navigation using vision has been addressed in the past, e.g. [8], vision is not commonly used on its own but usually in conjunction with other devices such as odometers or passive infrared sensors. In the current research, the robots do not have odometers; therefore

vision must be used to perform localization as well as the ultimate goal of mapping the surrounding environment.

For vision to be used as a range sensor that is suitable for map construction, there are two fundamental tasks to be performed:

1. Locate the obstacles in the image (Ground Plane Determination); and
2. Place the obstacles on a map using real-world coordinates (Calculate Range Data).

Locating obstacles is basically a process of image segmentation, which is a non-trivial problem in its own right. Having identified the obstacles, placing them on a map requires transforming pixel coordinates in the image back into real-world coordinates. These two tasks are addressed in the following sections, but only form part of an overall program of research for collaborative mapping by multiple robots using vision.

2 Ground Plane Determination

In its simplest form, obstacle detection is the process of distinguishing an obstacle from the floor (sometimes referred to as the ground plane in the literature). It is not necessary to understand what is seen in order to avoid obstacles – simply distinguishing between the floor and all other objects (even moving humans) is sufficient.

Many obstacle detection methods have been proposed, each with their own advantages and disadvantages. In the early 1990s, Horswill [9] developed a robot, called Polly, which navigated using vision. This widely-cited work has been the basis for many robots since then.

Polly operated in a restricted environment with a uniform floor colour and had a built-in map. The robot had a monochrome camera, and used a resolution of only 64 x 48 pixels. Although this sounds small, it turns out that sometimes too much information is actually a disadvantage, both in terms of processing power requirements and because the additional detail at higher resolutions tends to obscure the similarity of floor pixels. Polly had a problem with obstacles of a similar colour to the floor because it could not distinguish these isoluminant edges due to their similar greyscale values. The robot also had "bump" and sonar sensors, so it did not rely totally on vision.

Horswill went on to develop a second-generation robot called Frankie [10]. In this case an edge detector was used to identify obstacles. Frankie could therefore handle some small degree of texture, but there were still problems with shiny floors because the robot would treat reflections as obstacles.

The constraint of a uniform colour for the floor was relaxed somewhat and turned into a constraint on the frequency components of the floor texture. Because it is a spatial frequency, the constraint threshold is affected by the height of the camera from the floor, with higher off the floor being better. The robots in this paper are quite small, and the camera is very close to the floor which accentuates any carpet texture. Cheng and Zelinsky [11] took the concept of the Polly vision system, but used a very large grid size so that carpet texture would be averaged out.

In another similar approach to Polly, Howard and Kitchen [8] developed Robot J. Edgar which used a simple one-dimensional edge detector to locate the first non-floor pixel in each of 30 vertical columns in the image at a rate of 10Hz. Due to noise, as many as 10% of the boundary points in the resulting map exhibited errors. Consequently, the measurements from three frames

were combined to produce reliable obstacle maps. A manual calibration process was used to build a table for mapping from image coordinates to real-world coordinates.

Lorigo *et al* [12] extended Horswill's work and refined the method for segmenting the floor pixels from obstacle pixels by using a colour image and a sum of absolute differences between histograms of a reference area and the rest of the pixels in the image. The approach used three "vision modules" – brightness gradients (edges), Red and Green (from RGB) and Hue and Saturation (from HSV). An obstacle boundary map was constructed by each of these modules, and the results fused together using smoothing and median values. However, the image was still only 64 x 64 pixels, and a sliding window of 20 x 10 pixels was used in computing the obstacle boundaries. These Polly-derived systems fail if the floor is textured.

In contrast, Lourakis and Orphanoudakis [1] developed a system that used texture to advantage to extract the floor from successive images. The system is based on a ground reference, and detects obstacles that protrude from the ground using a registration process.

Ulrich and Nourbakhsh [2] developed a system that was intended to be used in a variety of environments. The system used a resolution of 320 x 240 in colour and ran at up to ten frames per second. The process involved transforming into HSI (Hue Saturation Intensity) colour space and classifying pixels by comparing the Hue and Intensity to values of a histogram in a reference area. Hue is notorious for wild variations at low illumination, and so "invalid" values of Hue and Saturation, where Intensity was too low, were ignored.

Segmentation of colour scenes for Robot Soccer is a common topic in the literature. Bruce *et al* [13] present a scheme using the YUV colour space that allows them to track several hundred regions of 32 colours at a resolution of 640 x 480 and a frame rate of 30 Hz using a 700 MHz PC. The obvious disadvantage of this method is that it must have 32 predefined colours. Das *et al* [14] claim that YUV works better than RGB for segmentation.

With these systems as background, we have developed our own segmentation scheme for determining the ground plane from an image. The objective is to provide a more robust method of segmenting the images and the approach uses edges as well as the colour of the floor.

2.1 Locating Obstacles

Using video data, a free space (floor) map is constructed from the Red, Green, Intensity and Hue image attributes using an algorithm referred to as a "Flood Fill". It is basically a Region Growing method, but rather than just looking for pixels that satisfy some similarity measure, i.e. thresholding, the algorithm also allows the pixel values to vary across the image using a simple moving average. This reduces the number of mismatches due to gradients from non-uniform illumination and specular reflections.

In addition to segmenting the image using colour information to identify obstacles, a Canny edge detector is run on the intensity image to obtain an edge map. It is well known that edges play an important role in human perception, and we have found edges to be good indicators of the presence of obstacles. Hue is the poorest indicator of all, partially because it has no representation for black or white, which are common colours. Also, hue is very unreliable in low light situations where it can fluctuate widely.

Although ground plane determination is basically a segmentation problem, Martin [15] found that a simple Sobel edge detector was the best way to discriminate between the floor and obstacles. He selected several different algorithms and then applied a genetic approach to allow the best choice of algorithms to evolve. This reinforces the importance of edges.

The "Flood Fill" algorithm works as follows:

- The colour image is smoothed using an averaging filter. This eliminates some of the "salt and pepper" pixels that commonly occur in video images.
- For each of the image attributes (R, G, H, I), the pixel in the centre of the image immediately in front of the robot is used as a starting point.
- The program then scans outwards across the baseline comparing pixel values. A moving average is maintained so that the colour of pixels can vary across the image. Matching pixels are marked as free space. If the difference between two successive pixels exceeds a threshold, then the scan stops. (In effect this operates as a colour edge detector.)
- Next, a scan, using the same moving average matching process, is performed upwards from each baseline pixel that has been marked as part of the free space. This extends the free space upwards until the colour difference becomes too great.
- Finally, the free space map is masked by the edge map. The assumption here is that the floor is of uniform colour, so any edge that is encountered whilst scanning upwards is the base of an obstacle sitting on the floor.

This process "floods" through the image filling in the "floor map" wherever the pixels are similar to the immediate foreground in front of the robot. Most other approaches use simple thresholds, whereas the "flood fill" allows the colour of the floor to vary across the image.

3 Obtaining Range Data from Images

Camera images are two-dimensional and the process of capturing an image loses the depth information and introduces a so-called Perspective Mapping. All the points in three dimensional space along a ray of light traced from the camera lens will map to the same pixel in the image. Of course, occlusion prevents the camera from seeing past the nearest obstacle along the ray.

The classical method for turning an image back into a three dimensional map, referred to as Inverse Perspective Mapping, involves an analysis based on homogeneous coordinates and the perspective transformation matrix. The transformation matrix can be determined through analysis using an idealised pinhole camera and knowledge of the focal length of the camera lens. Unfortunately, the focal length is unknown in our case.

In the case of a digital camera, the pixel grid can be used to advantage. Bertozzi *et al* [16] derived formulae for use with a stereo system, but their result is fairly complex. Cheng and Zelinsky [11] presented an analysis for a monocular camera. However, the camera was tilted to the extent that the entire Field of View (FOV) of the camera intersected the floor. This is not the case in general. Furthermore, there appears to be a mistake in the derivation in that the maximum screen x and y coordinates should be one less than the screen resolution.

The camera FOV is illustrated in the diagram below (Figure 1). It shows the top and side views of a robot (not to scale). Notice that there is a blind area immediately in front of the robot. Also, the top of the camera image in this diagram is above the true horizon (the horizontal line

through the centre of the camera). In general, the horizon is not at the centre of the image because the camera is tilted downwards to improve the visibility of the foreground.

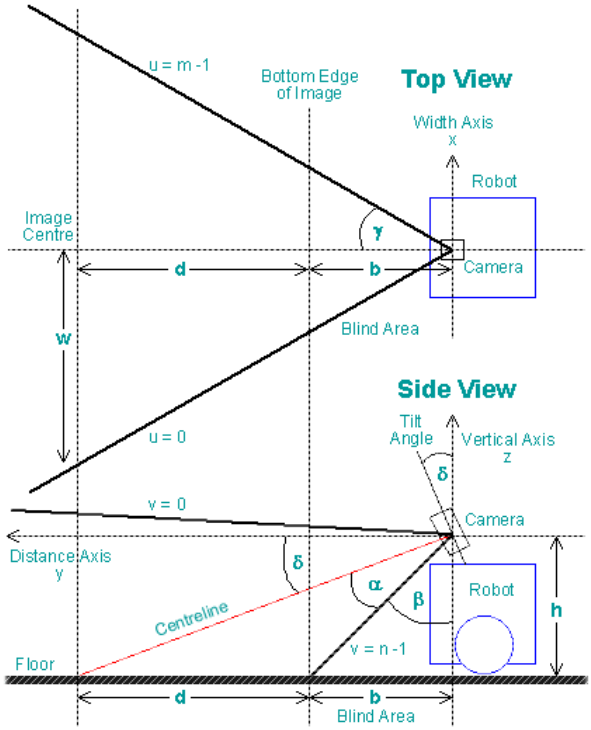

Figure 1 - Field Of View

In the FOV diagram, α is one-half of the vertical FOV; γ is one-half of the horizontal FOV; and δ is the camera tilt angle. (γ and α are related through the aspect ratio of the camera, which is usually 4:3 for conventional video cameras.) If the image resolution is m by n pixels, then the values of the image coordinates (*u*,*v*) will range from 0 to (m-1) and 0 to (n-1) respectively.

Consider rays from the camera to points on the ground corresponding to successive scanlines in the camera image. Each pixel in this vertical column of the image corresponds to an angle of 2α/(n-1). Similarly, pixels in the horizontal direction correspond to an arc of 2 γ/(m-1).

The following relationships can be easily determined from the diagram:

$$\alpha + \beta + \delta = 90^{\circ} \quad (1)$$

$$\tan(\beta) = b / h \quad (2)$$

$$\tan(\alpha + \beta) = (b + d) / h \quad (3)$$

$$\tan(\gamma) = w / (b + d) \quad (4)$$

The values of b, d and w can be measured, although not very accurately (to within a few millimetres) by placing a grid on the ground, and h can be measured directly.

For any arbitrary image vertical coordinate, *v*, the distance along the ground (y axis) can be calculated using the following formula (Equation 5). Note that, by convention, the vertical coordinates, *v*, in an image actually count downwards from the top. This affects the formula.

$$y = h \tan(\beta + 2\alpha (n - 1 - v) / (n - 1)) \quad (5)$$

If α is sufficiently large, then eventually *y* will progress out to infinity, and then come backwards from minus infinity. (This is a result of the tan function, but in geometrical terms it means that a ray through the camera lens intersects the ground behind the camera.) On the other hand, a larger tilt angle, δ, will reduce β so that *y* will never reach infinity, i.e. a ray corresponding to the top of the image will hit the ground.

Having calculated *y*, the *x* value corresponding to the *u* coordinate is:

$$x = y \tan(\gamma (2u - m + 1) / (m - 1)) \quad (6)$$

This is the distance along the *x* axis, which is the measured at right angles to the centre line of the robot and can therefore be positive or negative. Notice that *x* depends on both *u* and *v* because of the perspective transformation.

Given the four parameters (b, d, h and w), these calculations only need to be performed once for a given camera geometry, and a lookup table of corresponding (*x*,*y*) values can be constructed for all values of (*u*,*v*) in the image. (Note that the *y* value will approach infinity, so that above a certain *v* value it becomes irrelevant.) This makes re-mapping of image pixels for the inverse perspective mapping very quick, and the boundary line between the floor and the obstacles can easily be drawn onto a map using Cartesian coordinates with a known scale.

Similarly, a table can be constructed to map pixel locations into polar coordinates, (*r*,θ). Mapping the obstacle boundary into this polar space produces the Radial Obstacle Profile, ROP. The advantage of the ROP as a representation is that rotations of the camera are equivalent to a linear sliding of the ROP to the left or right, making it easy to predict what the ROP should look like after the robot has rotated on the spot, e.g. as part of performing a 360º sweep.

4 Experimental Results To Date

The algorithms discussed above have been implemented on two different robotic platforms:

- Hemisson (made in Switzerland by K-Team)
- Yujin Soccer Robot (made in Korea by Yujin Corporation)

The photos in Figure 2 below show the two robots.

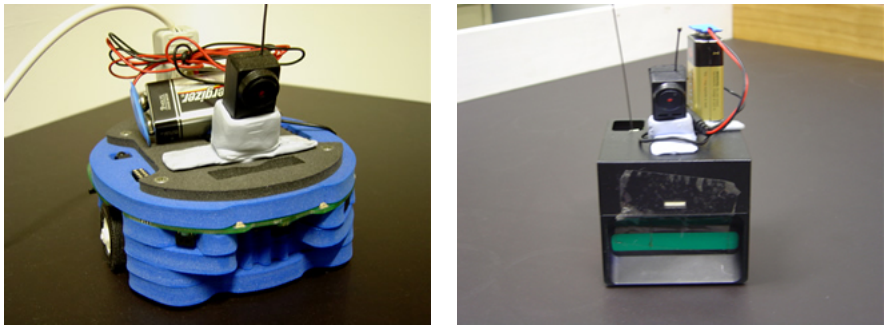

Figure 2 - Hemisson and Yujin Robots with Wireless Cameras

In the pictures above the cameras are visible on top of the robots, along with the 9V batteries, which give some idea of the size of the robots. The Hemisson robot is controlled via an "umbilical" serial cable, whereas the Yujin robot has an on-board wireless modem.

The robots are quite small – less than 10cm in diameter – and have two direct-drive wheels so they can turn on the spot. They roam around custom-built playing fields (similar to robot soccer fields, but without the lines on the floor).

Swann miniature wireless colour cameras are used for vision because they are light and relatively cheap. These have a resolution of 320 x 240 pixels but a Field of View of only 60º. Processing takes place on a PC equipped with an off-the-shelf USB video capture device connected to the wireless receiver. Cheap hardware introduces some challenges which require more ingenuity to solve, but this is a necessity when operating on a tight budget.

The sample screen shot below shows a robot's view of its "world" (on the left in Figure 3). Notice that there is considerable variation in colour across the floor and obstacle surfaces. Applying the Flood Fill algorithm results in the corresponding Floor Map as shown on the right.

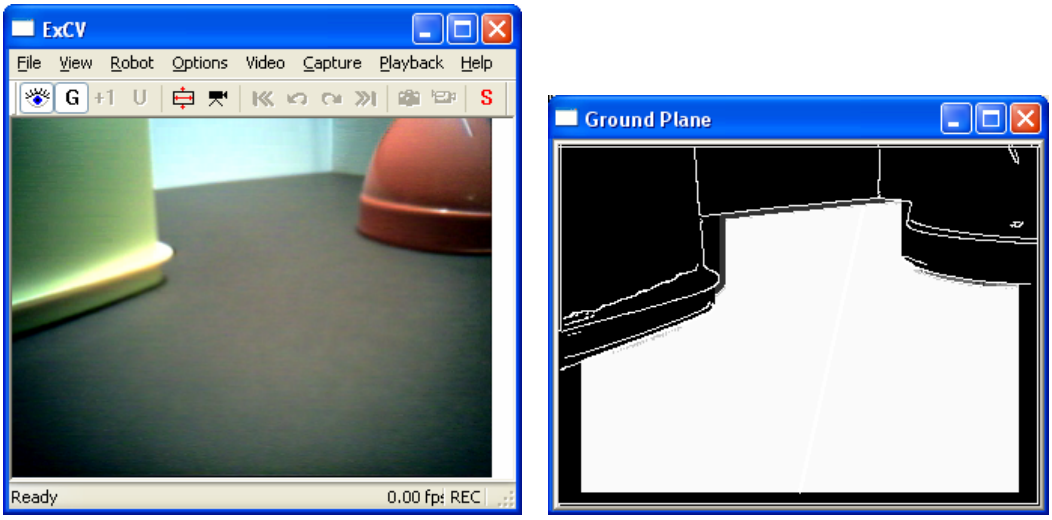

Figure 3 – Robot View of the world and the computed Floor Map

The set of image coordinates making up the floor boundary can be extracted from the Floor Map. Converting these via lookup tables results in the Radial Obstacle Profile, and the Inverse Perspective Map as shown in the diagram below (Figure 4) for a different robot view.

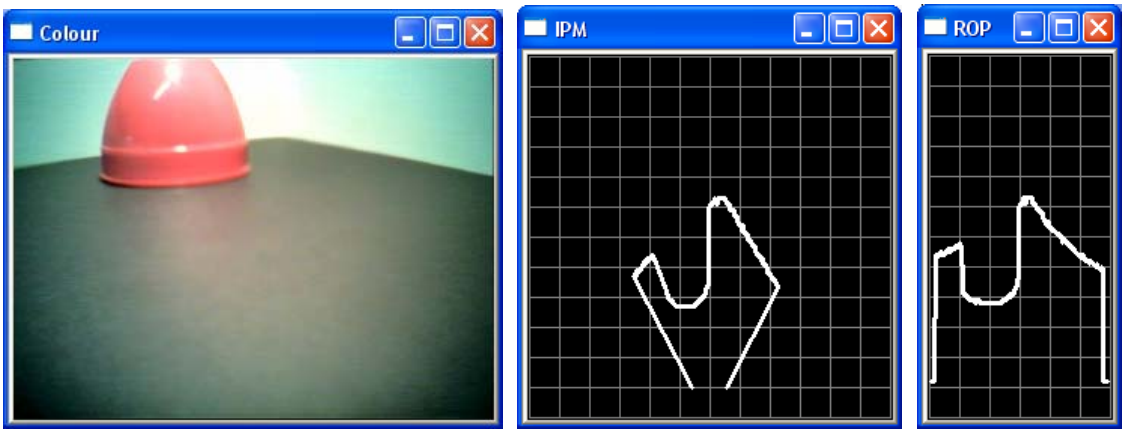

Figure 4 – Camera Image with Inverse Perspective Map (IPM) and Radial Obstacle Profile (ROP)

The IPM shows a top-down view of the real world, which is intuitively easy to understand. The robot is located at the bottom centre of the diagram and the grid markings are 10cm apart. The area inside the curve is the visible region, which is why the two sides angle outwards at 60º

because this corresponds to the FOV. Note that the large object in the centre-left throws a "shadow" because the camera cannot see behind it. The other straight edges are walls.

The ROP is a plot of the range to the nearest obstacle versus the angle within the field of view. The horizontal axis is in degrees, with markings every 10 degrees. (The camera has a 60º FOV and is located at the bottom centre of the diagram.) The vertical axis is in centimetres with 10cm markings. This diagram gives a somewhat distorted picture of the floor boundary, but the advantage is that scrolling the ROP curve to the left or the right corresponds to rotating the camera viewpoint.

Pre-defined behaviours are available for the robots based on the time required to move a given distance or rotate through a specified angle. By issuing a series of turn commands, it is possible to build a 360º range sweep of the surrounding area by combining the ROP information at each step. Automating this procedure is still work in progress.

For localization and mapping, it is essential for the robot to know exactly how far it has turned. Unfortunately, the Hemisson does not execute turns very reliably. The Yujin robot turns are quite precise due to its speed-controlled motors. Comparison of successive ROPs should help to determine how far the robot has turned.

4.1 Analysis of Results and Difficulties Encountered

By far the greatest difficulty in this process is recognising the floor in an image. Segmentation suffers from the "perceptual colour constancy" problem, i.e. what we humans perceive as being the "same" colour can in fact be a wide range of different pixel values. Flood Fill sometimes stops short if the illumination gradient is too great, but it performs better than the thresholding approach as used by many other researchers.

Humans are also very good at "filling in the gaps" when it comes to edge detection, but most algorithms create broken lines under poor illumination or with low quality video images. If the colour of an obstacle is similar to the floor, there will not be a solid edge and the Flood Fill can "leak" into the obstacle.

If the robot gets too close to an obstacle, then the template region used in the Flood Fill is actually part of an obstacle, and then the matching is based on an obstacle, not the floor. This results in the robot attempting to climb walls and push obstacles around, both of which are highly undesirable. To overcome this, when the robot is initialised, it "looks around" by turning 90º to the left and then to the right in order to determine the average colour of the floor as a starting point for comparisons. This also allows it to be used on floors of different colours.

The cameras suffer from several problems: Radial distortion; "Halo Effect"; and Adaptation. Radial distortion in the camera lens results in errors of around half a centimetre at the bottom corners of the image, but tens of centimetres at the top edge. This could be accounted for in the lookup tables, but we have not implemented this at the present time.

Visible in both Figure 3 and Figure 4 is the "halo effect" – the central part of the image is much brighter than the outer edges. This is due to the lens limiting the light reaching the corners of the image sensor. As a consequence, the edges of the images are frequently dark or almost black, which causes problems with them not being recognised as the floor.

The camera automatically adjusts the exposure and white balance. If the robot is left staring at the same scene for too long, the camera will adapt and the brightness of the image can vary considerably over time. Brightly coloured objects can affect the image, e.g. a large green object viewed for a long period of time causes the image to develop a reddish tinge.

Although the approach given above for Inverse Perspective Mapping is simple to implement, it is difficult to obtain the four parameters with sufficient accuracy to provide a reliable mapping over the whole image. Small variations in the parameters result in large variations at the image extremities. In particular, near the top of the image the y coordinate starts to approach infinity, and a single pixel corresponds to an enormous change in the y coordinate value. The mapping therefore becomes quite unreliable. For this reason, a limit has to be placed on the maximum range that the camera can "see". A limit of 60cm has been selected because the error in the estimated range increases rapidly after that. Note that in terms of a "range sweep" this is not a hard constraint, i.e. it does not indicate the presence of an obstacle, but rather an uncertainty in the range value. This uncertainty will be incorporated later when performing the mapping.

At this stage, the computation time required for processing images is quite high – over a second on a 450MHz Pentium III. However, this will become less of a problem over time as computers get faster. In the meantime, the approach that will be adopted is to only make point-wise linear motions, i.e. the robots will have a choice of only two types of motions – move forward by a small amount (5 or 10cm) or turn through a small angle (30º left or right). Because they will not have to make decisions whilst in motion, there is no need for real-time performance and the time factor becomes irrelevant.

5 Conclusions and Future Research

The Flood Fill algorithm works well even when the illumination is not ideal, as is the case in typical office environments. Like most algorithms, Flood Fill has some tuning parameters. The optimal values for these settings depend on the floor colour and the lighting conditions. Further work is required to automate this process.

Inverse Perspective Mapping using the algorithm outlined above allows fast conversion of pixel coordinates to real-world coordinates for the construction of a Radial Obstacle Profile. It also extends previous work to the case where the camera can see to "infinity". However, the mapping becomes unreliable towards the top of the image.

The Radial Obstacle Profile proposed in this paper provides an easy way to determine the range to the nearest obstacle in any direction in the robot's field of view. The amount of rotation between two images can be determined by comparing the two ROP curves because they are simple linear arrays. Future research will extend the use of the ROP to a 360º "range sweep".

Colour information in the image could be used to improve the accuracy of comparing successive ROPs in order to determine the turn angle. At present this is not done, but it is proposed to incorporate this information in future work.

Ultimately, the information gained from the methods in this paper will be used as input to a Simultaneous Localization and Mapping (SLAM) algorithm and the robots will be able to construct maps of their local environments using only monocular vision.

References

[1] M. I. A. Lourakis & S. C. Orphanoudakis, Visual Detection of Obstacles Assuming a Locally Planar Ground, *3rd Asian Conference on Computer Vision*, Hong Kong, 1998.

[2] I. Ulrich & I. Nourbakhsh, Appearance-Based Obstacle Detection with Monocular Color Vision, *17th National Conference on Artificial Intelligence*, Austin, Texas, 2000.

[3] G. N. DeSouza & A. C. Kak, Vision for Mobile Robot Navigation: A Survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, 237-267, 2002.

[4] Z. Zhang, R. Weiss & A. R. Hanson, Obstacle Detection Based on Qualitative and Quantitative 3D Reconstruction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, 15-26, 1997.

[5] M.C. Martin & H.P. Moravec, Robot Evidence Grids, *Technical Report CMU-RI-TR-96-06*, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.

[6] W. Elmenreich, L. Schneider & R. Kirner, A Robust Certainty Grid Algorithm for Robotic Vision, *IEEE International Conference on Intelligent Engineering Systems*, Opatija, Croatia, 2002.

[7] D. Murray & J. J. Little, Using Real-Time Stereo Vision for Mobile Robot Navigation, *Autonomous Robots*, Vol. 8, 161-171, 2000.

[8] A. Howard & L. Kitchen, Fast Visual Mapping for Mobile Robot Navigation, *IEEE International Conference on Intelligent Processing Systems*, Beijing, China, 1997.

[9] I. D. Horswill, Polly: A Vision-Based Artificial Agent, *11th National Conference on Artificial Intelligence*, Washington, D.C., 1993.

[10] I. D. Horswill, Visual Collision Avoidance by Segmentation, *DARPA Image Understanding Workshop*, Monterey, California, 1994.

[11] G. Cheng & A. Zelinsky, Real-Time Visual Behaviours for Navigating a Mobile Robot, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Osaka, Japan, 1996.

[12] L. M. Lorigo, R. A. Brooks & W. E. L. Grimson, Visually-guided obstacle avoidance in unstructured environments, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Grenoble, France, 1997.

[13] J. Bruce, T. Balch & M. Veloso, Fast and inexpensive color image segmentation for interactive robots, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, 2000.

[14] A. K. Das, R. Fierro, V. Kumar, B. Southall, J. Spletzer, & C. J. Taylor, Real-time vision based control of a nonholonomic mobile robot, *IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001.

[15] M. C. Martin, Genetic Programming for Robot Vision, *From Animals to Animats 7: The Seventh International Conference on the Simulation of Adaptive Behavior*, Edinburgh, UK, 2002.

[16] M. Bertozzi, A. Broggi & A. Fascioli, Stereo inverse projection mapping: theory and applications, *Image and Vision Computing*, Vol. 16, 585-590, 1998.