Fast and Robust Road Segmentation and Obstacle Map Generation for Autonomous Navigation

F.P. Senekal

Council for Scientific and Industrial Research, South Africa

fsenekal@csir.co.za

Abstract

The ability to detect and navigate drivable road surfaces is an important research area in autonomous navigation for use in autonomous vehicles. In this paper, a probabilistic computer vision algorithm for segmentation of tarred road surfaces is developed. Using a calibrated camera, a projection of a local obstacle map is then laid over the segmented image and an estimate is made of the likelihood of drivable region in each occupancy cell.

The algorithm is both fast (can be implemented in real-time systems) and robust (road surfaces are segmented well). The method was tested on a set of test images captured from a camera mounted on an autonomous vehicle. Good classification results are achieved, making it possible to use the algorithm and the resulting obstacle map in conjunction with global and local path planning algorithms to achieve autonomous navigation.

1. Introduction

The development of *intelligent autonomous vehicles* has recently gained much interest under researchers in the field of robotics. The research interest has perhaps been brought about by a series of competitions organised by the Defense (sic) Advanced Research Projects Agency (DARPA) in the United States. These competitions (2004 Grand Challenge, 2005 Grand Challenge and 2007 Urban Challenge) required teams to construct a vehicle that could autonomously sense its environment, navigate and plan its movements, under off-road conditions (240km in the Mojave Desert during the 2004 and 2005 competitions) or even under simulated traffic conditions, requiring vehicles to obey a limited set of traffic regulations and negotiating with other traffic for right of way (2007). The competitions generated much interest from top universities and vehicle manufacturers and have been successfully completed during the 2005 and 2007 challenges.

Even prior to the DARPA challenges, the EUREKA Prometheus Project received more than a billion dollars in research funding from the European Commission in the period 1987 to 1995. It succeeded to construct autonomous vehicles that could achieve speeds up to 175km/h on the German Autobahn and drove up to 158 km without human intervention.

Problems in the field of sensing and navigation for autonomous vehicles would thus largely seem solved. However, one of the primary sensors used for sensing is laser scanners (especially during the DARPA challenges). Laser scanners are able to construct an obstacle map in a local area around the vehicle. They are relatively fast and relatively successful in their application. There are however a number of shortcomings. Laser scanners are active sensors, requiring them to emit electromagnetic energy, which may raise safety and legal concerns. They typically obtain measurements along a single scanning dimension, which may mean less data to analyse. Furthermore, they are more expensive than passive sensors. For these reasons, there is quite a bit of research interest in using passive sensors (cameras) to analyse a scene image for use in autonomous vehicles.

There are two general approaches to determining a traversable road region in a digital image – road segmentation and lane detection. In *road segmentation*, the objective is to analyse image content to determine the properties of the projected image of the road that will distinguish it from other content in the image. Various approaches have been tried, such as analysis based on appearance cues [1], analysis based on structure and motion cues [2] various segmentation techniques such as watershed segmentation [3] and a variety of classifiers such as neural networks [4].

In *lane detection*, the objective is to use the painted lane markings or road boundaries as image cues in determining the location of the road. Different lane extraction methods have been tried, such as the Hough transform [5], straight line approximation [6], correlation-based detection [7] and inverse perspective mapping [8].

Road segmentation and lane detection are challenging computer vision tasks, since it need to be robust under various environmental conditions, lighting conditions, scene clutters and variable contrast. Clutter can be due to shadows, reflections, tire skid marks, oil drops, surface wears, dirt, occlusions by vehicles, pedestrians and other objects, etc. In addition, although it may conceptually be easy to characterise roads and lanes as having a specific colour or texture, these concepts are challenging to define in the computer vision community and to implement in a software algorithm.

For our purposes, we are interested in developing an algorithm that does not only perform well, but also executes fast, enabling implementation in a real-time system.

The algorithm needs to be seen in context as part of a larger vision system and autonomous vehicle platform. The CSIR is currently developing an autonomous vehicle, named the CSIR Autonomous Rover (CAR). The CAR implements a number of systems, such various vision and positional sensors, vision system, localisation system, route planning, local planning, actuation, etc. The vision system performs various tasks, such as road segmentation (as discussed in this paper), lane detection, traffic sign recognition [9], stereovision, structure from motion and object recognition based on appearance cues. The objective of the vision system is to produce an obstacle map, which can be used by the route and local planning systems to determine a safely traversable path for the vehicle.

Section 2 discusses the method that was developed to solve the problem of fast road segmentation and obstacle map generation. In Section 3, the results are discussed and the computational performance of the algorithm characterised. Section 4 provides suggestions for future work. The article is concluded in Section 5.

2. Method

The objective of the method presented here is to construct a local obstacle map in front of the autonomous vehicle, given a digital image captured from a camera mounted on the vehicle (shown in Figure 1). Such a map consists of various cells, each cell indicating whether the associated area is part of the road surface or not.

Figure 1: *Camera mounted on an autonomous vehicle.*

The obstacle map is created in a three-step process. Firstly, the digital image is analysed to determine which parts of the image are likely to be part of the road surface. This is expressed in the form of a probability map, where each pixel in the digital image has an associated probability of being road. Secondly, a three-dimensional planar grid is projected onto the two-dimensional digital image. This projection makes assumptions about the image content, such as that the environment of the autonomous rover is mostly flat. It also requires knowledge about the camera properties, such as how it is mounted (the tilt angle and height above the ground) and its intrinsic calibration properties. In the third step, the probability of being road surface over each region in the two-dimensional image that corresponds to a projected three-dimensional cell is calculated. Cells with probabilities above a certain threshold are marked as being road surface in the final obstacle map; those below the threshold are marked as non-road. The process is visualised in Figure 2.

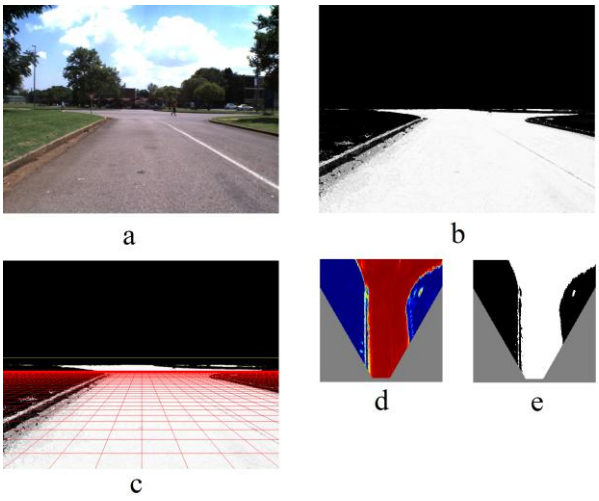

Figure 2: *Visualisation of steps taken during road segmentation and obstacle map generation. (a) Source image. (b) Probability map (light areas indicate high probability and dark areas low probability). (c) Projection of a grid onto the probability map (red lines indicate grid cells, yellow line indicate the horizon). (d) Probabilistic obstacle map (red represents higher probability, blue represents lower probability, grey not in view). (e) Thresholded obstacle map.*

2.1 Determining Road Probability

The first step is to determine a probability associated with each pixel as being road. We seek an approach that is probabilistic in nature, yet fast enough for real-time implementation. In this method, we build a Bayesian classifier, based on the colour and position of a pixel. Assuming that the position and colour is independent, we can write

$$P(R \mid r,g,b,x,y) = P(R \mid r,g,b)P(R \mid x,y), \qquad (1)$$

where $P(R|r,g,b,x,y)$ is the conditional probability that a given pixel is road, given its colour and position, $P(R|r,g,b)$ is the conditional probability that a pixel is road given its colour ((r,g,b) representing its colour coordinate), and $P(R|x,y)$ is the conditional probability that a pixel is road given its coordinate (x,y).

2.1.1 Conditioning on Position

We observe that the horizon line in the scene will have a projection onto the image. Given the assumptions and setup described in Section 2.2, the horizon line will project onto a single scan line, y_ω in the image. Clearly all pixels above the y_ω will not be road pixels, while those below it will have some probability of being a

road pixel. We express the probability of being a road pixel, given its position (x, y) in the image as

$$P(R \mid (x,y)) = \begin{cases} 0 & \text{if } y < y_\omega \\ 1 & \text{if } y \geq y_\omega \end{cases}. \quad (2)$$

We could try to characterise the probabilities of pixels below y_ω as being road pixels based on their position, rather than assigning a constant probability value of 1. Such a characterisation could work well given that a mathematical model of the road can be constructed that is predictive of the probabilities of being road at a certain position in the image and will be investigated in future work. For our purposes, we found Equation 2 to lead to satisfactory results.

2.1.2 Conditioning on Colour

Our objective here is to construct a probability estimate for a pixel as being road or not based on its colour. Thus we seek a probability density function describing the distribution. One approach is to construct a parametric distribution model, for example modelling the colour distribution as a three-dimensional Gaussian distribution. However, to speed up the process we opted to use a colour lookup table, which is in essence a non-parametric approach.

Colour images typically represent pixels with three channels (red, green and blue), each channel intensity being at one of 2^L levels. L is typically 8, which means that a single pixel can take one of 2^{24} values (roughly 16 million). We could construct a colour lookup table that specifies a probability for each of the 2^{24} values. However, this would require enough training data to ensure that such values are statistically significant.

To lower the training data requirements, we reduce the number of intensity levels of each level to 2^K. For our purposes, we use $K = 4$, resulting in 2^{12} (4096) unique values. This reduction is achieved by simply truncating the least significant bits of each colour channel. It is worth noting that if the lookup table is represented as a linear memory representation (rather than a three-dimensional array), the appropriate index can be determined very efficiently from the original colour coordinate through simple bit-shift and binary-or operations. The reduced lookup table is also much less memory intensive (being $1/2^{L-K}$ in size of the original).

The probability associated with each index (colour) in the lookup table is now simply calculated from training data as the proportion of times that the specific colour is found as part of the road, to the total number of times that colour is found. Let an image in a set of training images be represented by $I^{(i)}$ and the associated overlay by $O^{(i)}$, where

$$O^{(i)}(x,y) = \begin{cases} 1 & \textit{iff the pixel is a road pixel} \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

Then

$$P(R \mid (r,g,b)) = \frac{\sum_{i=1}^{N}\sum_{x=0}^{W-1}\sum_{=0}^{H-1} O^{(i)}(x,y)\delta^{(i)}(x,y)}{\sum_{i=1}^{N}\sum_{x=0}^{W-1}\sum_{=0}^{H-1} \delta^{(i)}(x,y)}, \quad (4)$$

where

$$\delta^{(i)}(x,y) = \begin{cases} 1 & \text{if } \mathrm{I}^{(i)}(x,y) = (r,g,b) \\ 0 & \text{otherwise} \end{cases}, \quad (5)$$

N is the number of training images and W and H are the width and height of the image respectively. When translating to computer code, Equations 3-5 is efficiently implemented as a simple loop through the training data, indexing two arrays that keep track of the number of times a colour was observed and the number of times that colour was observed as part of a road pixel, followed by a final division calculation. Of course, the lookup table is only constructed once during the training phase.

2.1.3 Smoothing Filters

Given equations (1), (2) and (4), we could calculate the road probability for any input image. In practise, the input image may need to be smoothed before the probability calculations are applied.

For example, under normal daylight conditions, a tarred surface would have a couple of areas reflecting light that is projected as white spots in the image. Since the white is not characteristic of the typical grey associated with a road surface, their associated probabilities would be lower, resulting in an effect akin to spot noise in the probability map. To counter this effect, a smoothing filter is applied to the input image, thereby reducing the effect of such occurrences.

In our work, we apply a box filter, which simply determines the average value in an $N \times N$ area centred on the pixel. Typical applications of a box filter require N^2 computations (N^2 – 1 additions and 1 division). To speed up computation, we use integral images. For an image I, the integral image II is defined as

$$II(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i,j). \quad (6)$$

The integral image can be computed efficiently using only two additions per pixel, as described by:

$$\begin{aligned} &s(0,y) = I(0,y), \forall y \in [0, H-1] \\ &s(x,y) = s(x-1,y) + I(x,y) \\ &\forall x \in [1, W-1],\ y \in [0, H-1] \\ &II(x,0) = s(x,0) \forall x \in [0, W-1] \\ &II(x,y) = s(x,y) + II(x,y-1) \\ &\forall x \in [0, W-1],\ y \in [1, H-1] \end{aligned}. \quad (7)$$

The integral image has the property that the sum of values over any rectangular area can be computed in constant time (three additions and thus the average in four operations). The sum of values of an image patch in an image from top left coordinate (x_1, y_1) to bottom right coordinate (x_2, y_2) can be computed from its integral image as

$$\begin{aligned} S = &II(x_2, y_2) + II(x_1 - 1, y_1 - 1) \\ &- II(x_1 - 1, y_2) - II(x_2, y_1 - 1) \end{aligned} . \quad (8)$$

The classical approach to implementing a square $N \times N$ box filter requires about N^2WH operations, where the implementation using an integral image requires about $6WH$ operations. It is clear that there is a computational saving for a filter as small as 3×3 and a substantial saving for larger filters.

2.2 Grid Projection

Consider the camera mount setup depicted in Figure 3. The camera is mounted at a height H above the ground (not to be confused with the image height) at a tilt angle α, such that its optical axis is pointing in the direction of the arrow in the figure. Typically, α would be close to 90° for a forward-facing camera. We assume that there is no roll around the optical axis, such that an imaginary line on the ground plane at a distance Z (relative to origin O) would project onto a single scan line in the image. If the effect of the roll is not negligible, it can be incorporated into the model, at the cost of increasing the computational complexity of further steps in the method.

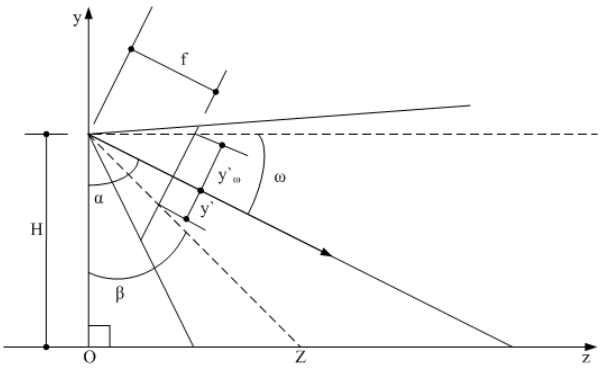

Figure 3: *Camera Mount Setup (Side View).*

We would like to determine the scan line y onto which such an imaginary line at distance Z on the ground plane would project. We calculate

$$\begin{aligned} \beta &= \tan^{-1}(\frac{Z}{H}) \\ y` &= f \tan(\alpha - \beta), \\ y &= y_c + y` \end{aligned} \quad (9)$$

where f is the focal length of the camera (which for convenience sake is expressed as a number of pixels), y_c is the y-coordinate where the optical axis intersects the image plane and $y`$ is the offset relative to y_c. For convenience sake, we assume that the optical axis is orthogonal to the image plane.

We are also interested in the scan line y_ω associated with the horizon line. We could use the above formulas with $Z \to \infty$; however it could be calculated directly using

$$\begin{aligned} \omega &= \frac{\pi}{2} - \alpha \\ y`_\omega &= f \tan(\omega), \\ y_\omega &= y_c - y`_\omega \end{aligned} \quad (10)$$

where $y`_\omega$ is the offset relative to the y_c. Assuming there is no roll, we can calculate the x-coordinate of the projection of a point (X, Z) on the grid as

$$\begin{aligned} x` &= \frac{X}{Z} f \\ x &= x_c + x` \end{aligned} , \quad (11)$$

where x_c is the x-coordinate where the optical axis intersects the image plane (see Figure 4).

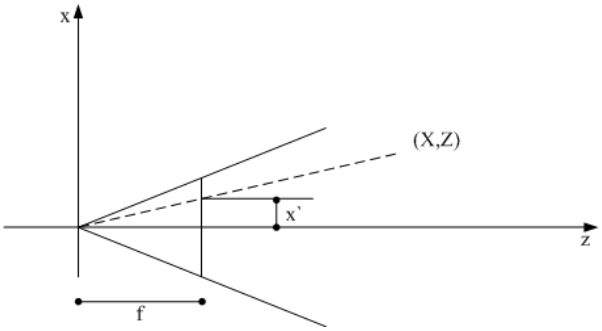

Figure 4: *Camera Mount Setup (Planar View).*

In general, a coordinate (X, Z) in the plane at $Y = 0$ would thus project onto

$$(\frac{X}{Z} f + x_c, f \tan(\alpha - \tan^{-1}(\frac{Z}{H})) + y_c) \quad (12)$$

in the image according to our model. The model thus requires x_c, y_c, f, α and H to be determined through a calibration process.

2.3 Determining the local obstacle map

Given equation 12, we can determine the projection of any coordinate in a planar surface in front of the camera in the image. We now superimpose an aligned grid on the planar surface, with equal spacing in the X and Z direction. The size and resolution of the grid could be set depending on the requirements for the local obstacle map.

For a given grid configuration, we pre-compute the projections of the various grid line intersections ((X, Z) coordinate pairs). Note that at a specific distance Z, all projections will have the same y-value. This is true

since we have assumed there is no roll in the camera orientation. We also pre-compute whether the projection actually falls within the boundaries of the image, since the coordinate we are interested in, may not actually be within the field of view of the camera.

Any square grid cell will project onto a trapezium in the image. The trapezium is defined by six coordinates – two defining the *y*-coordinate of the two parallel lines, and four defining *x*-coordinates of the four corners – all of which has been pre-computed. We can also pre-compute whether any grid cell will be fully or partially out of view in the image, and flag it as such.

For every grid cell, we now determine the average probability over the projected trapezium using the calculated probability overlay. Since the two parallel lines of the trapezium fall onto two unique scan lines, the average over the trapezium can be simply computed by stepping from scan line to scan line, computing the starting and ending *x*-coordinates using linear interpolation, and keeping a running sum of the average values and count of the number of pixels visited, followed by a final division. (Strictly speaking can the number of pixels being visited also be pre-computed. One can even devise a scheme where each pixel is back-projected onto a grid cell and the index stored, thereby eliminating the interpolation requirement (akin to the process in ray-tracing applications). It is worth pointing out however, that the further away from the camera (and depending on the grid resolution), neighbouring cells may be projected onto a single scan line, thus limiting such an approach. However, the integrity of the local obstacle map could be questioned then in any case).

3. Results and Discussion

3.1 Dataset

For purposes of testing the success of the algorithm, a dataset was captured from a camera mounted on an autonomous vehicle. The vehicle was put under human control, and driven on a road network during good daylight conditions. A total of 6000 images were captured, having a resolution of 900×680 pixels each.

Of the 6000 images, every 200th image was taken and manually labelled into road and non-road areas, resulting in a total of 30 training images. A selection of four of the 30 images is shown in Figure 5, showing the variety of road textures, colours, lane markings, shadows, white-washing, rubber marks and other conditions that occur. The figure also shows the corresponding probability maps when the algorithm developed in this paper is applied to the images.

The process of labelling the images is quite labour intensive, since every pixel has to be marked as either road or non-road, limiting the amount of training data that can be generated. It is also susceptible to subjective assignment, especially close to the regions between road and non-road segments. As such a classification system that is 100% accurate is not possible. The dataset used in [1] and [2] is a bit more extensive and is being acquired for testing purposes.

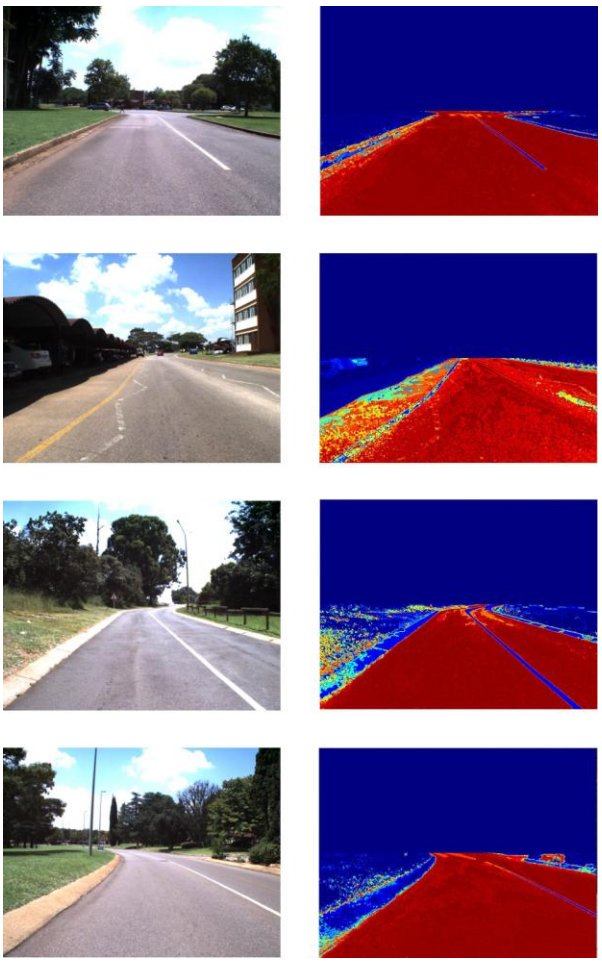

Figure 5: *Examples of road images captured from a camera mounted on a moving vehicle (left) and the resulting probability maps obtained by applying the algorithm suggested in this paper (right) Red represents higher probability and blue represents lower probability.*

3.2 Results and Discussion

In this section, we will report on the results obtained for road segmentation. A series of tests were conducted, ranging the filter size and probability threshold parameters. Box filters with dimensions 1×1, 3×3, 5×5, 7×7, 9×9, 11×11 and 13×13 were tested. Simultaneously the probability threshold parameter was evaluated over the interval [0, 1].

The training images were first used to train the colour distribution as discussed in Section 2.1.2. The training process constructed the lookup table used to index the probability associated with each colour. The lookup table is then used on the same dataset to determine the performance of the system. Although this process may introduce unfair bias into the system, we believe that such bias is minimal, since the process was applied to a set of similar (but unlabelled) images, which presented similar results when visually interpreted. Future tests using a larger number of training images will however provide a fairer assessment of the system.

To characterise the system, the number of true positives (*tp*), false positives (*fp*), true negatives (*tn*)

and false negatives (*fn*) over the entire dataset was calculated. As performance measures we use accuracy ((*tp* + *tn*) / (*tp* + *tn* + *fp* + *fn*)), precision (*tp* / (*tp* + *fp*)) and sensitivity (*tp* / (*tp* + *fn*)). The results for all filter dimensions achieve peak accuracy at a probability threshold of about 0.45 to 0.55.

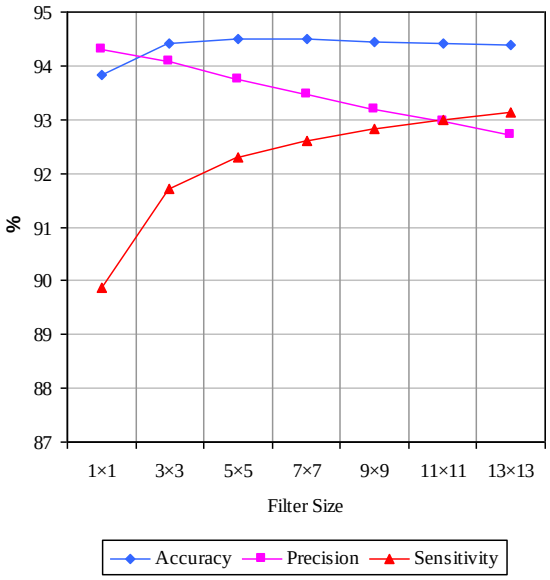

Figure 6: *Accuracy, precision and sensitivity as a function of filter size at a probability threshold of 0.5.*

The accuracy, precision and sensitivity for different filter dimensions are plotted in Figure 6. A constant probability threshold of 0.5 is used. Peak accuracy of 94.5% is achieved for dimensions of 3×3 and higher. The effect of using the box filter is clearly visible in the figure, improving from 93.8% for a 1×1 box filter (unfiltered image). The box filter thus has the effect of smoothing areas which may be susceptible to noise or small reflections, as discussed in Section 2.1.3.

It is interesting to observe the change in precision and sensitivity as a function of filter dimensions. For larger filter dimensions there is an increase in sensitivity but a decrease in precision. This means that for larger filter dimensions, there is an increase in the number of false positives and a decrease in the number of false negatives. This provides a convenient performance trade-off which needs to be evaluated in terms of the requirements of the system.

Given that the system makes a type I error (stating that an area is road in the case that it is not, i.e. more false positives), an obstacle map may be constructed indicating that a non-road cell is a road-cell and hence traversable. This is a serious error that may lead to an accident. In the case of a type II error (stating that an area in not road even if it is, i.e. more false negatives), the system will possibly mark some road cells as non-road. This will lead to the system simply avoiding such cells when traversing a path, or in the worst case, not being able to move. Clearly, type I errors are far more severe than type II errors and a system with as few as possible false positive errors is preferred. Thus, smaller filter dimensions are preferred over larger filter dimensions. This can be explained quite intuitively by observing that for larger filter dimensions, small detail may be smoothed and thus lost. There is also considerable smoothing at the road/non-road boundaries, where possible errors may be made. A 3x3 or 5x5 filter size seems to be the best choice.

The usability of the system lies in the obstacle map that is created. Type I errors in the obstacle map are obviously to be avoided. Due to the averaging effect of the map over a number of pixels in the region, the obstacle map has the effect of smoothing small errors, at the risk truly neglecting small obstacles. Depending on the mechanical structure and robustness of the vehicle, this may or may not be a concern. More serious is the occurrence of Type I errors due to larger obstacles, which will cause damage or accidents. The typical case is that such objects have similar colour characteristics as the road, making them indistinguishable to the classifier. Such objects typically protrude above the ground and due to the projective nature in which the obstacle map is generated, will clutter the scene and cover many more grid cells than usual. However, the covered grid cells will typically be further away than the cells at the feet of the object where it meets the road surface. For example, the cells closest to the camera for the projection of a vehicle are typically the ones associated with the wheels of the vehicle. The wheels of a vehicle typically are black, with a low probability of being classified as road. Although cells further away may thus create Type I errors, they might not actually be reachable as part of path calculated by a local planner.

3.3 Performance

In this section, we discuss the performance aspects of the algorithm. We assume that the digital image on which the algorithm is applied has a resolution of *W*×*H*. As discussed in Section 2.2, we are only interested in the portion of the image below the projected horizon line. For the purposes of this discussion, *H* will be the effective height, i.e. the number of horizontal scan lines below the projected horizon line (these scan lines form the effective region). Strictly speaking, the region is actually narrower, since we are only interested in the portion of the image below the projection of the furthest end of the overlaid grid.

The road probability is calculated by means of a lookup table. The index into the lookup table can be calculated very efficiently using bit-shift and binary-or operations based on the colour coordinate of the pixel. In addition, two memory references are required (one for the actual colour lookup and one for the assignment). This step requires a single pass through the image and could even be implemented on a parallel processing architecture.

As discussed in Section 2.1.3, the smoothing filter requires constant time (about 6*WH* simple operations) and is completed in a single pass through the image. Efficient parallel processing algorithms also exist to implement box filters.

The steps during grid projection (Section 2.2) require a calibrated camera. However, all the calculations can be pre-computed, given a constant tilt

angle. In the case that the grid projection is calculated in real-time based on tilt angle information provided by an external sensor, the coordinates of the projected grid intersection lines need be to computed. Given an $m \times n$ grid, this requires the calculation of $(m+1)n$ x-coordinates and the calculation of $n+1$ y-coordinates. Since m and n is typically much less than W and H, this would require no more complexity than is required for a single pass through the image.

The determination of the local obstacle map (Section 2.3), require a that the average over the projected trapezium associated with each grid cell be computed. This means that every pixel covered by the trapezium is visited once. Pixels associated with trapezium boundaries are thus typically visited twice. In the far field, when the two parallel lines associated with the trapezium are projected onto a single scan line, each pixel could be used in more than two averaging calculations. Given the back-projection scheme suggested in the final paragraph of Section 2.3, each pixel will only be visited once. The thresholding step is executed as part of the calculation associated with each grid cell. In general, the time complexity of this step is roughly equal to the number of pixels in the effective region.

The method presented in this paper thus requires three passes through the effective region (four under the condition of changing tilt angle). During each of these passes, only a few simple operations are done, which means that the algorithm executes very fast, achieving real-time speed for fairly large images.

4. Future Work

One possible way of improving the road segmentation, is to incorporate a probabilistic approach based on texture, rather than colour. Since texture may be a better distinguishing feature for a road surface than colour, especially with regard to greyish objects that may be visible in a scene, it is expected that fewer false positives will be generated. An additional idea is the incorporation of a multi-scale approach, where texture probabilities over multiple scales are used during classification.

Another area that requires improvement is the detection of lane markings. Lane markings are falsely rejected as being part of the road surface, since they are not characterised well based solely on colour (more accurately, the training data provides many examples where the same colours are used in a different context than a road surface, hence the associated probabilities are lower). The multi-scale texture-based approach may solve this problem.

Finally, the algorithm is sensitive to the tilt angle of the camera relative to the ground plane. Although it does not have a mayor effect in the near field, a fraction of a degree change in tilt angle may have a large effect in the far field. In the absence of a sensor in the system that could directly measure tilt, techniques need to be developed to stabilise the image and determine tilt angle relative to a feature in the far field.

5. Conclusions

A technique to segment a road surface from a digital image using a probabilistic approach was presented. Using a calibrated camera, a projection of a local obstacle map is laid over the segmented image and an estimate is made of the likelihood of drivable region in each occupancy cell.

The technique performs reasonably well and has the advantage that it is fast enough that it can be implemented on a real-time system. When used in conjunction with global and local path planning algorithms, the obstacle map can be used to achieve autonomous navigation in an autonomous vehicle.

6. Acknowledgements

The research conducted and reported on in this paper was funded by the Council for Scientific and Industrial Research (CSIR), South Africa, under the CSIR Autonomous Rover (CAR) project.

7. References

[1] J. Shotton, M. Johnson and R. Cipolla, "Semantic texton forests for image categorization and segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2008*.

[2] G.J. Brostow, J. Shotton, J. Fauqueur and R. Cipolla, "Segmentation and recognition using structure from motion point clouds", in *ECCV*, pp. 44-57, 2008.

[3] S. Beucher, M. Bilodeau and X. Yu, "Road segmentation by watershed algorithms", in *Proceedings of PROMETHEUS workshop*, Sophia-Antipolis, 1990.

[4] C. Fernandez and W. Bonner, "Texture and neural network for road segmentation", in *Proceedings of the IEEE Intelligent Vehicles Symposium 1995*, pp. 344-349, Detroit, MI, USA, 1995.

[5] P.S. Liou and R.C. Jain, "Road following using vanishing points", *Computer Vision Graphics and Image Processing*, Volume 39, pp. 116-130, 1987.

[6] C. Thorpe, M.H. Herbert, T. Kanade and S.A. Shafer, "Vision and navigation for the Carnegie-Mellon Navlab", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 10, Issue 3, pp. 362-373, 1988.

[7] C.J. Taylor, J. Malik and J. Weber, "A real-time approach to stereopsis and lane-finding", in *Proceedings of the IEEE Intelligent Vehicles Symposium 1996*, pp. 207-212, 1996.

[8] D. Pomerlau, "RALPH: Rapidly adapting lateral position handler", in *Proceedings of the IEEE Intelligent Vehicle Symposium 1995*, Detroit, MI, USA, pp. 90-95, 1995.

[9] F.P. Senekal, "Traffic sign detection and classification using colour and shape cues", in *Proceedings of the Nineteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pp. 131-136, Cape Town, South Africa, 2008.