BMVC99

# Poppet: A Robust Road Boundary Detection and Tracking Algorithm

M.B. Wilson, S. Dickson
AMAC, SME, Cranfield University, Cranfield MK43 0AL

**Abstract**

There have been many attempts at following road edge boundaries based upon edge information. Most of this work concentrates on the following of road markings. This paper describes a robust algorithm which can track both unmarked road boundaries and marked road boundaries. The Poppet (Position of Pivot Point Estimating Trajectory) algorithm accumulates edge points in the image and makes a decision on the position of the road boundary based upon the maximum of the accumulated values. The merits of using an edge map which is not thresholded are discussed. Finally, the implementation of Poppet using parallel processors is discussed.

**Keywords**: automated driving aids; edge detection; road edge boundary; boundary detection; road following; parallel processing

## 1 Introduction.

Automated driving and the development of driving aids for use in "smart" cars are areas of research that are very active [8] [6] [1]. A fundamental task in many of the systems developed is the determination and tracking of road boundaries. This paper reports the development of an algorithm (Poppet- Position of pivot Point Estimating Trajectory) that robustly tracks unmarked road boundaries under a wide variety of different conditions.

The problem is to uniquely identify a road boundary in a moving monochrome video image. Finding edges in the image is relatively straight forward using conventional edge detection techniques, but to find and follow the road edges while rejecting the non-road edges, requires secondary processing.

An approach often adopted to the problem of tracking road direction is the identification and following of "white line" road markings. Limiting the problem to white line following eases the requirements, because the white line provides a distinct "white against dark" edge and the line usually follows a curve free from mathematical discontinuities making it a good candidate for mathematical modelling approaches. Chapuis [3], Morgan [4] and Schneiderman [7] are just a few examples of work covering model based methods which can track road markings.

The identification of road boundaries in countryside scenes is more difficult, because edges at these boundaries are less distinct, variable (from grass to sand, for example), do not follow smooth curves and are subject to shadowing and other lighting effects. The inconsistent nature of this type of boundary gives rise to many outlying edges, which, if included in a least squares fit will badly distort the result. For the purposes of detection in this class of road, the road boundary should be the junction of the grass verge with the road surface, however, a small margin of negative error (into the road) but clearly no positive error (into the grass verge) would be acceptable. A white marking on such a road is effectively the acceptable margin of error. If there are any white line markings they should be identified in preference to road boundary.



BMVC 1999 doi:10.5244/C.13.35

# 2 Detection and following of road boundaries

## 2.1 Generation of the edge map

Road edge detection and tracking should identify a boundary that a vehicle must not cross (i.e. somewhere in the negative error margin). The first stage in this identification problem is the generation of an edge map.

The requirements for the edge detector in this application are entirely different from finding a unique edge in the Canny [2], sense i.e one response, localised and the optimisation of noise rejection.

Noise variations are insignificant in this application as the contrast between the road and non-road surface is clearly defined. The exact position of the edge is not important within the error margin discussed above. The most important criteria is to mark definite points close to the road edge within the negative error margin, that is, close to where a white line would be painted on a similar, but marked road. Detected road boundary points represent the road boundary correctly for our requirements. Other detected points are considered as outliers.

Inspection of pictures of road scenes, show that due to defocusing there is little variation across a road edge on a pixel by pixel basis. Therefore an edge detector which looks for edges on a pixel by pixel basis will fail to record any change at these points. If the detector spans several pixels then the variation will be more pronounced. There are two byproducts of this approach:

i) A comb filter is produced which will miss high spatial frequencies

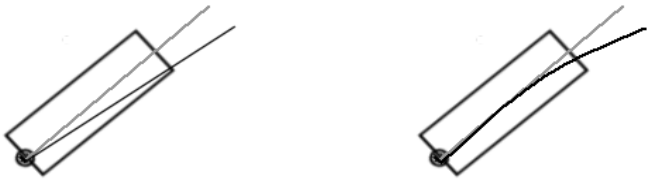
ii) The resultant edge will be broad (multiple response to a single edge)

The road edge will not contain high spatial frequencies so losing pixel by pixel detail will only lose non-road edges, this is obviously an advantage to our aim. In the context of the above discussion ii). is not a particular drawback and is used to our advantage, (as discussed in section 2.3).

## 2.2 Window based Poppet

Once the edge map is generated, it must be processed in order to differentiate road edges from other edges. The detected road edge should be long and continuous and, (assuming that the car is already correctly positioned on a U.K. road), the left hand side of the road should start somewhere in the bottom left of the frame. If the start of the road edge can be positively identified then it is possible to continue to track the rest of the edge. This is the bootstrap phase identified by Waxman et al [9]. It is acceptable to take a long time for the bootstrap, in proportion to other processing, providing that it is robust. The Poppet algorithm (Position of Pivot Point Estimating Trajectory) originally operated a thresholded edge map. It used a rectangular window which rotates about a pivot point. The pivot point was moved to all of the possible positions of the entrance of the road edge within the image frame. Then for each of the individual possible positions and all of the individual rotation angles the number of edge points inside of the pivoting window was counted and stored. The road edge was considered to be contained within the window whose position and angle had maximum edge points. The road edge could then be redrawn using the position and angle information. The principle is shown below in figure 1. The angles and positions are discrete and so a window was necessary not only to tolerate small non-linearities in the road edge, but also to tolerate linear portions of the road edge which did not exactly match the discrete angle. The window encloses edge pixels which would otherwise have been missed. This is shown in figure 2 in which the grey line (parallel to and central to the long sides of the window) represents the closest discrete search angle to the road edge. Figure 2 a) shows the different angle case and figure 2 b) shows the non-linear, (but closer angle) case.

Figure 1: The counting window implementation of Poppet



(a) Window includes road edge with different angle to the discrete angle

(b) Window includes non-linearities in the road edge

Figure 2: Window implementations

## 2.3 Vector based Poppet

The search window used, (described above), must include a constant area if it is not to give different weights to different positions and angles. This involves intensive trigonal-metrical computations. A close equivalence exists between running a window over a single pixel edge compared with running a single line vector through a broader edge. The multiple response of the pixel spanning edge detector provides such a broad edge. See figure 3 below, here the grey line represents the single line vector (of the same discrete angle as the equivalent window). This reduces the generation of the initial search vector by 9 times, (see Results). The operation of this vector based algorithm is explained below.

The local average luminance in our road scene varies not only from image to image ,but also over a single image or frame, due to flare, street lighting, shadows and other car headlights. When the edge map of these scenes is thresholded there is no single optimum threshold value that can be chosen to accommodate these variations. Attempts at automatic thresholding have been unsuccessful. Thresholding not only removes weak edge points from the edge map, it also destroys information about the strength of the edge points. To avoid this information loss the edge map should not be thresholded.

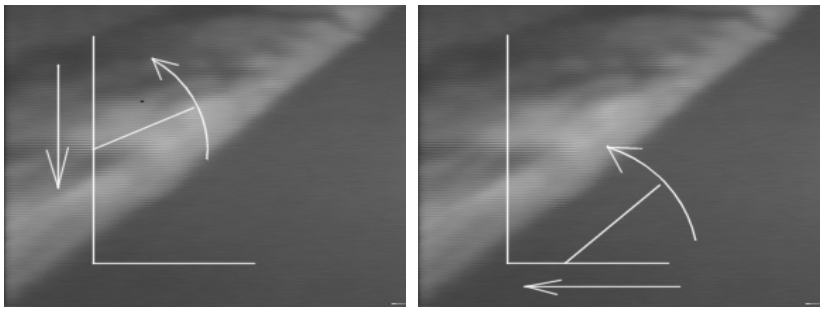
This version of the Poppet algorithm does not use a counting window or a thresholded

(a) Single line vector passes though all edge points of broad road edge of slightly different angle

(b) Single line vector passes through all edge points of non-linear broad road edge

Figure 3: Vector Implementations

edge map. A short vector of selected size is rotated about a pivot point over a range of angles which represent the expected range of angles of the road edge. A border of size 20 pixels around the image is not processed to avoid the inclusion of picture edges and electronic disturbances in the source material. The pivot point is moved over an expected range of intersections of the road edge with the border, firstly down the y axis and then along the x axis. (The order is unimportant to the operation). At each pivot point the vector rotates through the expected angle range. The is shown in fig4 below.



(a) Vector rotates and moves down y axis

(b) Vector rotates and moves along x axis

Figure 4: How Poppet searches

The expected angle range is 20-80 degrees and the expected x and y search range is 120 and 160 pixels respectively. These search angles and distances were selected empirically after the measurements of 50 differing still roadside images.

At each pivot point, for each angle, the value of the edge map is summed for all of the pixels along the vector. These summation values are placed in a two dimensional array; the indices of which are $\theta$ and distance. Two such arrays are constructed. (One for the search along the x axis and the other for the search along the y axis). The two arrays are then searched for the maximum summation element and the two sets of indices corresponding to the maxima are recorded. Then the maximum of the two array's maxima is determined

to select which set of indices is chosen to represent the road boundary. Since the array from which the maximum is selected is known, (x axis or y axis), the indices represent a point, $x_(max), y_(max)$ and an angle, $\theta_(max)$. The selected vector is then redrawn from $x_(max), y_(max)$ and $\theta_(max)$. See figure 5.

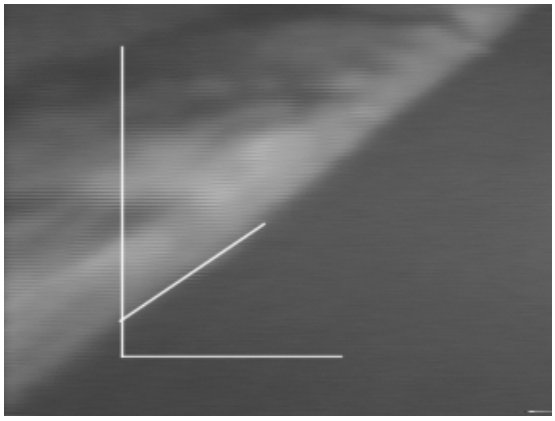
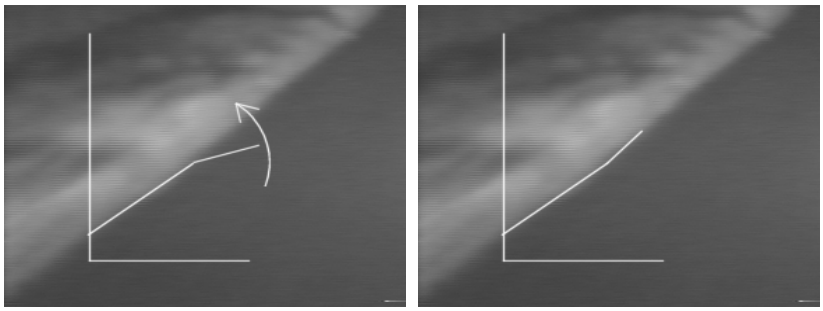


Figure 5: Vector is redrawn corresponding to maximum accumulated edge values

## 2.4 Extended Poppet

In order to track the road edge after the initial search vector has been determined a similar system to the above is used, except that the starting search point is well known since it should be the end of the initial search vector. A shorter vector is therefore pivoted at this point and rotated through a dynamically adjusted angle range (see section 4.3). Figure 6 demonstrates the principle.



(a) A new vector is started which pivots about the end of the old

(b) Again the vector is redrawn at the maximum correspondence

Figure 6: Extended Poppet

This process is continued towards the vanishing point of the road or until confidence measurements, (from the summations, see section 2.7), curtail the search.

### 2.5 Choice of the size of the initial search vector

The size of the search vector was determined after experimenting with a representative sample of road curvatures. The range of curvatures examined excluded the special case of turning around a corner at a road junction. The choice of the size is a compromise between sensitivity to angle and approximating extremes of curvature and processing time. As the size approaches 10 pixels, extending the vector linearly shows gives a predicted angle of the road border that is incorrect by up to 5 degrees. A size of 150 pixels will accurately track a straight road, but the end of the vector will be 20 pixels away from the road boundary on a road exhibiting the extremes of curvature. The processing time increases linearly with the length of this vector. The time taken to process the initial vector occupies 95% of the total processing time. It was found that the choice was not critical within the range of 40 to 100 pixels, consequently a figure of 50 pixels was chosen.

### 2.6 The size of the extended search vector and dynamic adjustment of the search angle

After deciding on the size of the initial search vector, the size of the extended vector was chosen using the same sample as above. The size of the extended search vector was found to be much more critical. As the vanishing point of the road is approached the effects of perspective will bring the number of spurious edge points within range of the extended search vector. A short vector of 10 pixels, for example, is very susceptible to spurious points and will deviate from the true road boundary, while a long vector (40 pixels or more) may end far from the road boundary and provide a rouge starting point for the next vector. The system will then lose its way. To improve the reliability of the extended search the search angle range was limited to reject spurious points. This overcomes, to some extent, the limitation of using a short extended search vector. The search angle range is limited to the angle of the previous vector $\pm 20$ degrees and a choice of 25 pixels for the size of the extended search vector provided maximum reliability on the sample data.

### 2.7 Use of the summation data

The summation data not only encodes the road edge, but includes important information as to the confidence of the road edge. In an extended search vector of length 25 pixels it is reasonable to expect a summation value of 20, (this is equivalent to only 20 detected edge pixels of minimum value). If the summation value is less than this value it is reasonable to have low confidence that this result correctly identifies a road edge. The extended search vector loop is terminated when summation value has dropped below this level and no more extended vectors are drawn. This is visually preferable to limiting the drawing of extended vectors by total maximum line length.

### 2.8 Implementation

After initial development on still images using a P.C., (to determine optimum parameters for lengths of initial and extended vectors and search regions and angles), Poppet was ported to a Matrox Genesis board to enable processing of live video. The Genesis board has a Texas Instruments TMS320C80 processor ('C80) for image manipulation. The 'C80 may be programmed directly if required, but the many built in access functions of the Genesis library allowed the development of a reliable implementation for live video.

## 3 Parallel implementation of Poppet

Poppet, when processing using the Genesis library functions, (which run on the host P.C.), achieved a speed of 4 frames per second. The TMS320C80 has five processors, one float-

ing point "master processor" and four integer parallel processors; which can be used as required. Programming the 'C80 master processor directly, instead of using the host Genesis functions, does not produce much speed improvement . Of course it is possible to considerably speed up the processing by using coarser starting points and search angles, and by taking fewer points on the search vector. This is fine for road marked scenes or scenes with no discontinuities along the search vector, but other scenes will demand the higher level of processing for accuracy. It may be possible to achieve speed up by following a coarse search with a finer search in the search area, but a limit will be reached when the initial coarse search is so coarse that it misses the region of maxima. So, the task of programming the 'C80 to run Poppet on the parallel processors was undertaken. Writing efficient parallel code for Poppet poses a problem because the system needs information from the lower part of the picture, before it can calculate the extended search vectors for the upper part. The conventional parallel processing method of splitting the scene into separate vertical bands and processing them simultaneously is therefore inappropriate. In this implementation each parallel processor is given the same portion of the image and the search angle is divided equally between the parallel processors. Each processor calculates the maximum summation value for the search angle range being used and the master processor calculates the absolute maximum from the four maxima. Poppet also relies heavily on sine and cos functions. Since the parallel processors do not support floating point operations, these functions must be converted to integer, (by multiplication by 100 and rounding) and transfered to a look up table in the parallel processors memory space. If Poppet is to run quickly on the parallel processors, the image information to be processed must reside in the parallel processors' local memory. The size of this memory is limited to 4KB. Information from the edge map must be transfered to this memory in square blocks for efficiency. Therefore the maximum search vector that can be accommodated is $\sqrt{4KB}$ or 63. Such a block must be transferred for each search point in both the x and y axes. The parallelisation of Poppet would be worthless if the time taken for memory transfers was longer than the time saved by dividing the processing by four. The 'C80 has dedicated hardware for the purpose of memory transfer; therefore Poppet can be successfully parallelised. Parallisation improved the processing speed by a factor of 3, giving 12 frames per second.

# 4 Results

Figure 7 shows the initial vector generated by the window based Poppet algorithm. Although it produces successful results it takes 9 times as long as the equivalent vector Poppet algorithm, (90 seconds and 10 seconds respectively on the P.C. still image development platform), and proves no more robust.

Figures 8, 9 and 10 are examples of typical output from the vector based Poppet algorithm and the Retina algorithm. [5] Retina was developed as a improved road boundary follower after reviewing literature on previous techniques and forms part of a PhD Thesis. These examples show Poppet successfully identifying and following marked and non-marked road edges, while Retina runs out of confidence very early in the following process. Figure 11 shows the effect of simply counting thresholded edge points.

The starting vector is identified correctly, but distant extended vectors fail to track. No optimum threshold value could be found for this image. Altering the threshold simply causes the failure to occur in different places or on different road images. Generation of the initial vector is time consuming, since it involves a double "for" loop (firstly for position and secondly for angle), while the extended vectors are generated using an angle for loop only. The "speed up" achieved by parallelisation is greater than three times, giving the promise of real time results if more processors where to be used.

Figure 7: Window based Poppet located initial vector

(a) Poppet (b) Retina

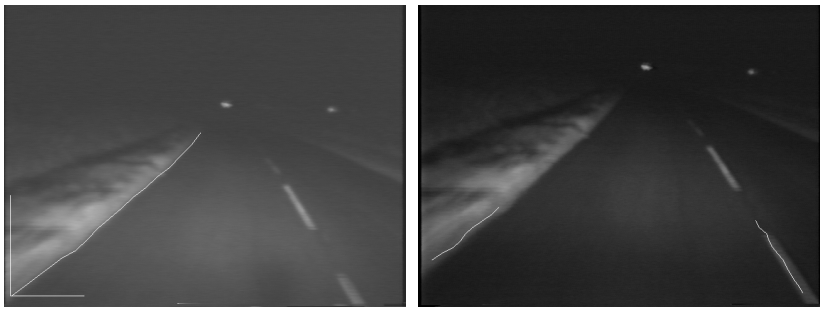
Figure 8: Performances on marked road

## 5 Conclusions

Poppet robustly identifies and tracks road boundaries on unmarked roads and white lines on marked roads, under varying conditions. The importance of late thresholding by linear accumulation of edge points can be clearly seen from the examples. The system is robust, but relies on long processing time. Processing time may easily be shortened by the use of parallel processors or by accepting a compromise in robustness.

## 6 Further work

There should be scope in reducing the time taken to generate the initial vector, possibly by use of coarser searches, followed by shorter finer searches. Temporal tracking should improve the robustness of the system, but with the processing times achieved the variation between successive processed frames is such that there is little correlation between them. However, this situation may alter.



(a) Poppet (b) Retina

Figure 9: Performances on unmarked roads

(a) Poppet (b) Retina

Figure 10: Performances on unmarked road with shadows and trees

## 7 Acknowledgements

We wish to thank Jaguar Cars and Texas Instruments for their continuing support and encouragement.

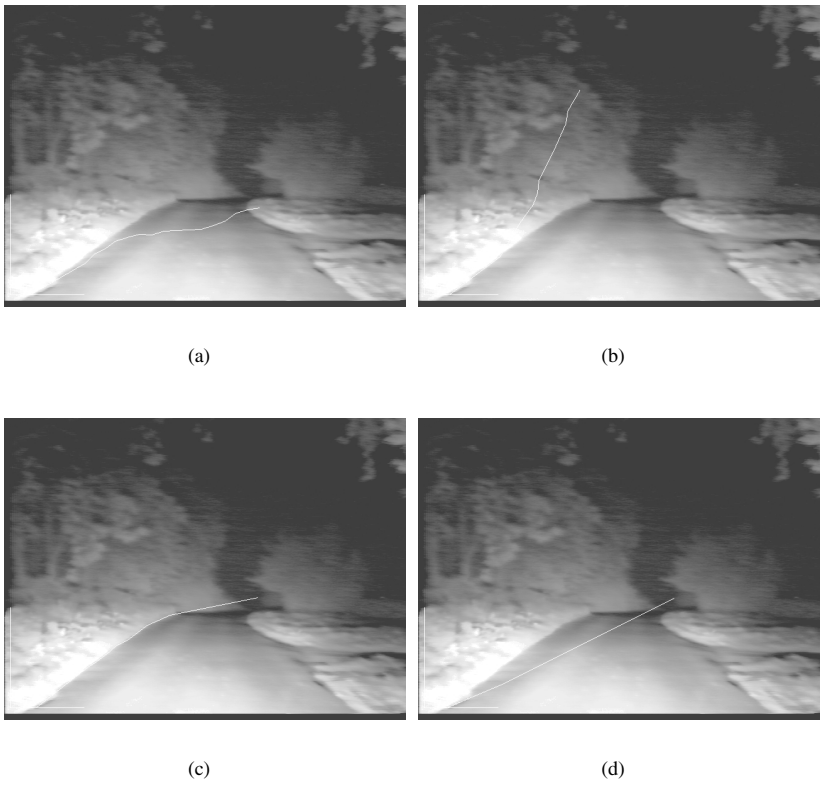
## References

[1] M. A. Brackstone and M. McDonald. Potential benefits of intelligent vehicle control systems. In *Autotech '95 Seminar on Automotive systems*, pages 61–68. London: IME, 1996, November 7-9 1995.

[2] J. Canny. Computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 8(6):679–698, 1986.

[3] R. Chapuis and A. Potelle. Real-Time Vehicle Trajectory Supervision on the Highway. *International Journal of Robotics Reasearch*, 14(6):531–542, December 1995.

[4] A. D. Morgan, E. L. Dagless, D. J. Milford, and B. T. Thomas. Road Edge Tracking For Robot Road Following. *Journal Of Image And Vision Computing*, 8(3):233–240, August 1990.



(a) (b)

(c) (d)

Figure 11: The effects of thresholding

[5] Alexandre Rio. *Investigation of Intelligent Adaptive Image Enhancement to Aid Night Time Driving*. PhD thesis, School of Mechanical Engineering, Cranfield University, England, 1999.

[6] T. Sasayama. Sensors and sensing technology for intelligent vehicle era. In *IME International Conference on Automotive Electronics*, pages 61–67. Bury St Edmunds:IME, 1994, May 17-19 1994.

[7] Henry Schneiderman and Marilyn Nashman. A discriminating feature tracker for vision based autonomous driving. *IEEE Transactions on Robotics and Automation*, 10(6):769–775, 1994.

[8] Steven E. Shladover. Research and Development needs for Advanced Vehicle Control Systems. *IEEE Micro*, 13(1):11–19, February 1993.

[9] A. M. Waxman, J. Lemoigne, and B. Srinivasan. Visual Navigation Of Roadways. In *Proceedings IEEE Int. Conf. On Robotics And Automation St Louis*, pages 862–867, March 1985.