Turkish Journal of Electrical Engineering and Computer Sciences

Volume 8 | Number 1 Article 1

1-1-2000

Knowledge-Based Navigation for Autonomous Road Vehicles

MURAT EKİNCİ

FRANCHES W.J.GIBBS

BARRY T. THOMAS

Turk J Elec Engin, VOL.8, NO.1 2000, © TÜBİTAK

Knowledge-Based Navigation for Autonomous Road Vehicles

Murat Ekinci
Department of Computer Engineering, Karadeniz Technical University,
61080, Trabzon, TURKEY
e-mail: ekinci@eedec.ktu.edu.tr
Franches W.J. Gibbs, Barry T. Thomas
Department of Computer Science, University of Bristol
Bristol BS8 1UB, UK

Abstract

This paper presents a computer vision system for an autonomous road vehicle (ARV) that is capable of negotiating complex road networks including road junctions in real time. The ultimate aim of the system is to enable the vehicle to drive automatically along a given complex road network whose geometric description is known. This computer vision system includes three main techniques which are necessary for an ARV: a) road following, b) road junction detection, c) manoeuvring at the road junction. The road following algorithm presents a method of executing a number of algorithms using different methods concurrently, fusing their outputs together into an accurate road model. A model-based object approach is used for detecting the road junctions in images. In this approach two sequence processes are performed. They are to find a boundary between the candidate road junction surface and the current road surface, and to locate the road junction surface. A multi-camera vision-based re-orientation mode was used to guide the ARV during the manoeuvring process at the road junction. In the re-orientation mode, the position of the ARV with respect to the road junction is determined by the "bootstrap" process. The results are presented for real road stretches and intersection images and performed on the experimental autonomous road vehicle in real time.

Key words: *Computer vision, image processing, autonomous road vehicle navigation, real-time computing, fusion.*

1. Introduction

In the research community, an increasing number of projects aim towards efficient means to support a driver to navigate a vehicle on a road network. The ability to guide a vehicle automatically on a road network is regarded as a desirable goal in itself as well as an intermediate objective towards a competent driver support system. Much work in the content of general outdoor road following has aimed to navigate the autonomous road vehicle (ARV) without requiring explicit knowledge of the environment [1, 2]. The VaMoRs-Project [3, 4, 5] and the VITA-Project [6, 7] in Germany, the PVS-Project [8] in Japan and the CMU-Navlab-Project [2] in the USA are well-known representative projects which demonstrate automatic

vehicle guidance in experimental vehicles. All of these projects use machine vision as the main information source about the vehicle's local environment.

Some work on model-based autonomous vehicle navigation in in door environments has also been done when explicit and accurate geometrical descriptions of the environment are available [9, 10]. The geometrical knowledge of the environment is used to remove any ambiguity in the path, thus greatly reducing the requirement for more robust object or path recognition, to plan a route and then manoeuvre the vehicle.

One of the key problems in ARV navigation is that it is very difficult to represent out door environments. The problem is especially acute for a general navigation system in which the details of the road network are not determined prior to implementing the navigation system.

Over the past eight years research at Bristol University has focused on producing modular hardware and software units so that the information from many algorithms can be combined to produce a robust vision system to navigate the autonomous road vehicle in a complex road network which includes road junctions.

This paper describes recent developments conducted by a team at the University of Bristol into computer vision system for an autonomous road vehicle that is capable of negotiating complex road networks in real time. This paper includes mainly three subjects. First (in Section 2), the more robust road following algorithm which four road following algorithms' outputs together into an accurate road model is presented. Second, a road junction detection algorithm based on a model-based object approach is described (in Section 3). Third, (in Section 4), a multi-camera vision-based re-orientation mode, which is used to guide the vehicle during the manoeuvring at a road junction, is explained.

Using the test vehicle and also the real-time system as the test platform, the more robust road following algorithm was tested by itself, while the road junction detection and manoeuvring algorithms were combined and tested together. Detection of a road junction was successfully achieved while the vehicle was navigated on a road by a road following algorithm which is explained by [11]. Manoeuvring at the road junction was also performed after the road junction detection algorithm detected the road junction. When there is high-level knowledge of the environment, the road following algorithm developed in Section 2 can be easily adapted to road junction detection and manoeuvring algorithms to navigate the ARV along a known complex road network [12].

2. Road Following Procedure

Different research institutes have also been pursuing road following algorithms; Thorpe et al. [13], Sharma and Kuan [14], and Turk et al. [15] segment the road from colour images, Davis et al. [16] use search windows to follow the road edges, and Dickmanns and Graefe [17] follow the edges using controlled correlation in search windows.

This section firstly considers several independent road following algorithms developed at Bristol University, and describes their continuing development to the point where all can operate at near real time when implemented on specially developed vision hardware. A robust real-time road following system is developed around these individual road following algorithms by implementing them in parallel on the vision hardware and fusing their outputs into a global road estimate. Finally the complete system is demonstrated in real-time scenarios; firstly using video footage of a road, and secondly by installing the system as the navigation unit in an autonomous vehicle and safely negotiating a test track. The overall vehicle-based system is further enhanced by the inclusion of odometric data into the fusion of the individual road following algorithms.

2.1. Road Following Algorithms

Four road following algorithms were developed for the Autonomous Road Vehicle (ARV): Road Edge Detector (Image plane), Road Edge Detector (World plane), Road Segmentation and White Line Boundary Follower.

Road Edge Detector (Image plane) - This algorithm, described in greater detail in Morgan et al. [18], models the road edges as two approximately parallel curves in the real-world plane. For ease of analysis these curves are represented by second order polynomials such that

$$X = C_0 + C_1 * Z + C_2 * Z^2 \tag{1}$$

where the (X,Z) plane is the ground plane, which is assumed to be flat with the X axis parallel to the vehicle's rear axle. (0,0) is the point perpendicularly below the camera. There are two modes of operation: a bootstrap mode and a real-time mode.

The bootstrap mode works on the assumption that the vehicle is correctly positioned on the road and that the strongest edges on the left and right are the road edges. Passes of the edge detector are made at various heights across the image until a consistent set of edge segments has been found on each side of the image. To fit the model, the points are projected onto the real-world horizontal plane and the curves are then fitted using a least squares method.

The real-time mode uses the current model to guide the search for the edges. This consists of transforming points from the previous frame's model back into the image plane and searching in the locality of the model for points in the current frame. The edges are detected using an oriented differential filter which is selected from a set of eight templates according to the predicted direction of the edge. The points found are then transformed into the world plane and the model is fitted, as in the bootstrap mode.

Road Edge Detector (World plane) - This algorithm is described in Lotufo et al. [19]. The image is subsampled, then transformed into the real-world horizontal plane. In the transformed image the road edges should now be almost vertical, and thus only a one-dimensional edge detector is needed to locate them. The current steering angle is used in the transformation to ensure that the relevant edges are indeed vertical. Naturally, other edges are found as well; the two edges that determine the road are the pair which have the most points, are parallel, and are approximately the road width apart.

Road Segmentation - The road surface is found using its grey level and texture. A two-dimensional feature space is built for each image row using the grey level for one dimension and a texture measure (a count of micro-edges generated using a Roberts operator) for the other dimension. A sample patch is taken from directly in front of the vehicle, and then for each row the correct number of points (determined for each row by an approximate road width) closest to the sample in the feature space are classified as road.

Whilst the feature vector approach does correctly classify the majority of pixels in the scene, there are still some errors caused by the following:

- Objects in the scene having either a similar grey level or texture measure.
- Junctions, sudden changes in road width, or passing cars causing a dramatic change in the number of visible pixels of the correct texture.
- An injudicious seed point having been chosen.

Significant errors in the pixel classification process cause the segmentation to fail; however, small errors can be overcome by fitting a road model to the pixels classified as road. This will at the very least give a clearly defined result as opposed to a region with ragged edges.

As the processing is done in the image plane, a simple image plane model is used to group the majority of the classified pixels together. The model used is a trapezium that would, when transformed to the horizontal road plane, have two parallel sides a road width's distance apart representing the left and right road edges. The trapezium is fitted over the classified pixels by maximizing a cost function C

$$C = \frac{(nl + r + nr)}{total \quad area} \tag{2}$$

where r is the number of classified road pixels within the trapezium, and nl and nr are the number of non-road pixels on the left and right of the trapezium respectively. The search area for the best fit is reduced greatly by only searching within the locality of the previous frame's trapezium model. Figure 1 shows a sample image with the best-fit trapezium model superimposed using black lines.

Figure 1. Individual results ready to be fused.

White Line Follower - The white line is modelled as a quadratic curve in the real world, as used in the image plane road edge detector above. An estimate of the white line's position in the frame (i.e., the white line found in the previous frame) is used to direct the search. A set of n horizontal strips is taken from the image at distance $Z_i (0 <= i < n)$ away from the vehicle such that the centre of strip i at distance Z_i is given by using Z_i in the estimated white line equation. The strips are separated by a constant distance in the real-world plane, and each is 1m wide. Each strip is transformed into the real world so that the width of the white line remains constant in all strips, and is then processed with a correlation mask designed to find a bar, 10 cm wide, of high intensity sandwiched between areas of low intensity. If the white line lies within the strip then a well-defined peak occurs in the output. Because the search is performed with a correlation mask and not an edge detector, false points are rarely found.

The white line model is then fitted to the (X,Z) points found in the strips using a least squares method. If no peak is found in a strip then the strip's data is ignored. To fit the model to the data, it is desirable to have as many strips as possible returning points. The number of points found depends on the following: the number of strips used, the accuracy of the estimate, the clarity of the white line, and the duty cycle of the white line (duty cycle is the ratio of the length of the white line segments to the length of the gap between them). The accuracy of the estimate is directly related to the algorithm's cycle time, and hence there is a direct trade-off between the number of strips used and the accuracy of the estimate. For a low duty cycle very few points are found, but even as few as three are sufficient to fit the model as false points are rare.

Each of these algorithms is implemented individually and demonstrates a degree of robustness; however, each can, in certain situations, produce false information or no information at all. When they produce no data they can reboot themselves, but when they follow false data, there is no way for the individual algorithms to know that the results they provide are misleading. Algorithms following false data will eventually provide no results and therefore reboot themselves, but this would be unsatisfactory for an autonomous vehicle operating in real time on these results. The solution to this is to run all the algorithms (and any others that become available) in parallel, fusing their outputs together to generate a consistently reliable road model.

2.2. Fusing the Algorithms

The fusion task takes from each road following algorithm a road estimate and a weight that represents how good that algorithm considers the road estimate to be. It then fuses all the estimates together into a fused road using a weighted averaging process. This is demonstrated on two representations of the road, one low level, the other high. The methods used by the algorithms to apply the weights, and the two road representations used, are described below.

2.2.1. Determining the Weights

The road following algorithms were each designed with the intent of operating independently as the sole navigation algorithm of an autonomous road vehicle. If more than one of these algorithms were to be concurrently implemented, the controlling algorithm would receive simple road position estimates from each algorithm with no way to discern between them. In the production of the road estimates, each algorithm discards relevant data that could be used to gauge a self-assessed goodness of the road estimate. From the road-finding engine of each algorithm, a weight is generated that can then be used as a score of how good the algorithm considers its road estimate to be. The weights returned by each algorithm are between 0 and 1. Both fusion tasks take the weights from all the algorithms and normalize them such that the sum of the weights from the n algorithms is 1 (as shown below).

$$\sum_{j=1}^{n} w_j = 1 \tag{3}$$

Road Edge Detector (Image Plane) - In the original implementation of this algorithm, values from the least squares fitting process were used to gauge the goodness of the algorithm. However, while giving an accurate reflection of how well the points themselves from the edge fit the line model, this does not give any indication of how well the points represent the side of the road. Thus a different approach was taken whereby the score given to an edge reflected the strength of the response from the edge detector, along with the consistency.

The scores from the convolution of the edge detector through the search windows are accumulated for each search window in which an edge is found. They are averaged and then scaled such that the final score lies in the range $(0 \leq w \leq 1)$. To generate the scaling factor, the maximum range of values' output from the edge detector was considered. The image range is 0 to 255, so the range of output for the edge detector is -2040 to 2040. However, the contrast at the road edges is by no means so great, so using the range 0 to 255 is unreasonable. By inspection it was found that the average grey level difference at a good edge was 80 or more, and that a poor edge had a difference in the region of 40. Using these extremes a weight between

0 and 1 is generated for the edge detector output when its value is in the range 1000 to 2000 (representing 3 runs of the edge mask for each search window). Thus an edge generated from edge points lying on different but close edges could in the previous weighting system have received a high weight but now will only do so if the edges have like-signed gradients (both + or -).

Road Edge Detector (World Plane) - If the transformation angle used in generating the planview image is approximately correct, then, of the candidate edges, the actual left and right edges will be approximately vertical and have a length close to the full height of the planviewed grid. They will be approximately one road's width apart. These two measures are used to generate a weight for the final pair of edges decided upon by the algorithm. The weighting function is

$$w = \frac{\frac{n_l}{N} + \frac{n_r}{N}}{2} \left(1 - \frac{|roadwidth - W|}{roadwidth}\right) \tag{4}$$

where w is the weight allocated; n_l is the number of points in the vertical direction in the left edge; similarly n_r is the number of points in the right edge. N is the maximum possible number of points in an edge; $roadwidth$ is the current road width in metres, unadjusted by the current results of the planview algorithm. W is the average width between the left and right edges found by the planview algorithm.

The weight for the planview algorithm is therefore calculated according to a consistency on road width and on the length of the new edges found.

Road Segmentation - Measuring the success of a feature space classification is difficult, so the closeness of fit of the trapezoidal road model to the segmented area is used as a means to weight the texture algorithm. The cost equation (equation 2) used to fit the trapezium to the segmented data provides a value that is used as the weight. Hence the weight is a measure of how un fragmented the texture classification is.

White Line Follower - The advantage of using a correlation mask for feature detection is that the results are reliable. Therefore, the weight is given using the assumption that any points found in the search strips are correct. Therefore the weight is given purely as a function of the number of points found. If N search strips are used, then for a dashed line, if half of them find the white lines, the weight is set to maximum. The weight is

$$w = \frac{2n}{N} \tag{5}$$

where n is the number of strips that have registered the white line. A function of the error between the points and the fitted curve could be included in the weighting function, but since not one false positive arose in initial tests this was not implemented.

2.2.2. Pixel Voting

The **pixel voting strategy** works in the image plane, and avoids the issue of choosing to work with a single road model, hence avoiding the unnecessary clipping of data. The fusion process operates in a horizontal **voting band** which contains a finite range in front of the camera through which the road must cross (assuming the camera is approximately correctly placed). The actual near and far values of the voting band were chosen by experimentation; the values found to be successful were within the following ranges: $(5m < Z_{near} < 10m)$ and $(20m < Z_{far} < 35m)$. Within the voting band, each pixel is voted for by giving it a value P whose value lies in the range 0 to 1, where 1 represents definite road, and 0 represents definite non-road.

Figure 2. Pixel Voting Result. The different grey levels within the voting band of the Pixel-Based Fusion show the strength of belief that that a pixel is in the road. The final road edges displayed on the image are obtained by thresholding the voting space.

P is assigned as follows:

$$P_{i,j} = \frac{\sum_{k=1}^{M} W_k}{\sum_{k=1}^{N} W_k} \tag{6}$$

where M is the number of algorithms that have the pixel at (i,j) in their road estimate, N is the total number of algorithms, and W_k is the weight (lying between 0 and 1) assigned by the k^{th} algorithm.

As the algorithm is intended for use in real time the voting area is subsampled to a more manageable size. The road following algorithms vote for pixels in the image as follows:

- White Line Follower: votes for any pixels within a half road width on either side of its line in the world plane. This limit is converted to the image plane.
- Texture Segmentation: votes for any pixels lying within its image plane trapezoidal road model.
- Edge Tracking code (planview and image plane): vote for pixels lying within one road width to the left of the right edge and similarly within one road width to the right of the left road edge.

Since each road following algorithm provides a vote for a solid mass of pixels, the resultant voting area is not fragmented (unless two or more algorithms disagree by more than half the road width), with those pixels having the highest vote being in the centre of the voting space and the pixel vote values decreasing outwards. A threshold value is then used to determine at which point the change in pixel values represents a road/non-road boundary. Increasing the threshold value will reduce the area in the image classed as road. All pixels whose vote value is over the threshold value are then labelled road, and the resultant road block can be used to generate a steering command for vehicle guidance. The threshold value was chosen from experimentation. The edges of the designated road area are stored by the fusion process by converting stored points to the real world where they have quadratic curves fitted to them. The saving of the points and curves is for analysis purposes only.

Figure 2 shows this fusion method's output given the input road estimates shown in Figure 1. The belief that each pixel in the horizontal voting band is road is shown by the pixel's brightness. The final road

edges, obtained using thresholding on the voting space, are shown by the dashed line. The original road image is shown above and below the voting band, and within the voting band where the belief is zero.

Figure 3. Model Voting Result (Individual results of four algorithms have been fused).

2.2.3. Model Voting

This strategy requires each of the slave road following algorithms to fit their own road data to a standard road model. The global road model represents the road as a pair of real-world quadratic curves, as used in the edge track code in section 2.1. The fusion algorithm accepts from each slave road following algorithm three curve coefficients representing the left road edge and three coefficients similarly representing the right edge. For the two edge tracking algorithms the data is already in this format, though for the texture segmentation and the white line follower a small amount of extra processing is needed. The texture algorithm takes the four corners of its image plane trapezoidal road image, converts them into the real-world plane, and then uses the two points available on each edge to generate a straight line equation respectively. The real-world quadratic equation modelling the white line's path along the road is modified as follows to generate left and right road edge estimates: the equation for the line, given by

$$X = C_0 + C_1 Z + C_2 Z^2 \tag{7}$$

is used to generate

$$X_{left} = \left(C_0 - \frac{road\_width}{2} \right) + C_1 Z + C_2 Z^2 \tag{8}$$

$$X_{right} = \left(C_0 + \frac{road\_width}{2} \right) + C_1 Z + C_2 Z^2 \tag{9}$$

The model fusion algorithm thus takes N input curves and generates a resultant fused curve. The fused curve's coefficients are calculated as follows:

$$F_i = \sum_{j=1}^{N} \frac{W_j C_{i,j}}{K} \tag{10}$$

where F_i is the fused curve coefficient (i being 0, 1, or 2), j is the algorithm's index, W its weight, $C_{i,j}$ its curve coefficient, K the sum of the weights, and N is the number of algorithms being fused together.

Although the left and right edges are treated as independent entities in the fusion process, their inherent relationship is taken account of in the slave road following algorithms themselves, and in the conversion from their own road models to the global road model prior to fusion. Figure 3 shows the results from the model voting strategy when applied to the road estimates shown in Figure 1.

2.3. Results on Road Following Algorithms

Firstly, both methods (pixel voting and model voting) were tested off-line on a SUN4 workstation. A sequence of 25 images was taken from a video of a road filmed using a video camera mounted on the roof of a car travelling at approximately 20 mph. The road chosen includes bends, junctions (3 on the left, 2 on the right), and a section where there is dirt on the road causing the white line to be obscured and the texture to be irregular. However, to include all of these hazards in 25 frames, the frames were grabbed at one-second (approximately 9m) intervals. The road following algorithms were designed to operate at a higher frame rate than this; the longer gap between frames here makes them less stable and, thus, demands a better performance from the fusion tasks than would be needed under real-time operation. [20]

For each image in the sequence, points were manually plotted on the left and right edges and real-world curves were fitted to these points. These manually created results were used to assess the success of the fusion methods. Note that the manually created results are shown in Figures 4-top by the broken line marked with plusses (+) and crosses (x), for right side and left side respectively.

Figures 4-top-left and -top-right show the results from the pixel voting method and the model voting method respectively, each plotted against the manually created results. The graphs show the x values in the image plane of the left and right edges at a distance of 10m in front of the vehicle, as shown in Figure 4-bottom. While the sequence of images was captured by a video camera mounted on the top of the vehicle, the vehicle was being operated by a driver.

At Z=10m in front of the vehicle, 10 pixels in the x direction is about 14cm. The average absolute error at Z=10m is shown below:
Pixel Voting left: 28 pixels
Pixel Voting right: 44 pixels
Model Voting left: 20 pixels
Model Voting right: 38 pixels

From Figures 4-top-left and -top-right it is clear that the left edge, which is the most important edge for a vehicle driving in the left lane, is followed very closely, and that at no point in the test run were both edges lost.

To test the two fusion methods on a real-time set of images, the code was ported to specially designed hardware consisting of Transputer-based frame stores, each algorithm running as an independent task on a separate frame store. A control process, running on another frame store, is permanently, waiting for results from the road following algorithms, and each time it receives a result it runs the fusion process, using the most recent data from each algorithm. Due to the sometimes unpredictable nature of a road scene, it is possible for one algorithm to differ from the others (and hence from the fused estimate) and yet still be more accurate than the others which are either following misleading information or simply slow to react. Consequently, commanding a slave to reboot immediately when it strays too far from the fused estimate would be a premature reaction and could actually cause the system as a whole to lose track of the road. To

overcome this, a record is kept by the control loop of the number of consecutive failures each road following algorithms has, only sending a reboot command when a certain number of consecutive failures has occurred. The reboot message contains the current fused road for the rebooting algorithm to seed from during re-initialization. Since only blatantly erring algorithms are to be pinpointed, an error in width in excess of 15% of the dynamically calculated full road width is used as the criterion for failure. This value was chosen by experimentation.

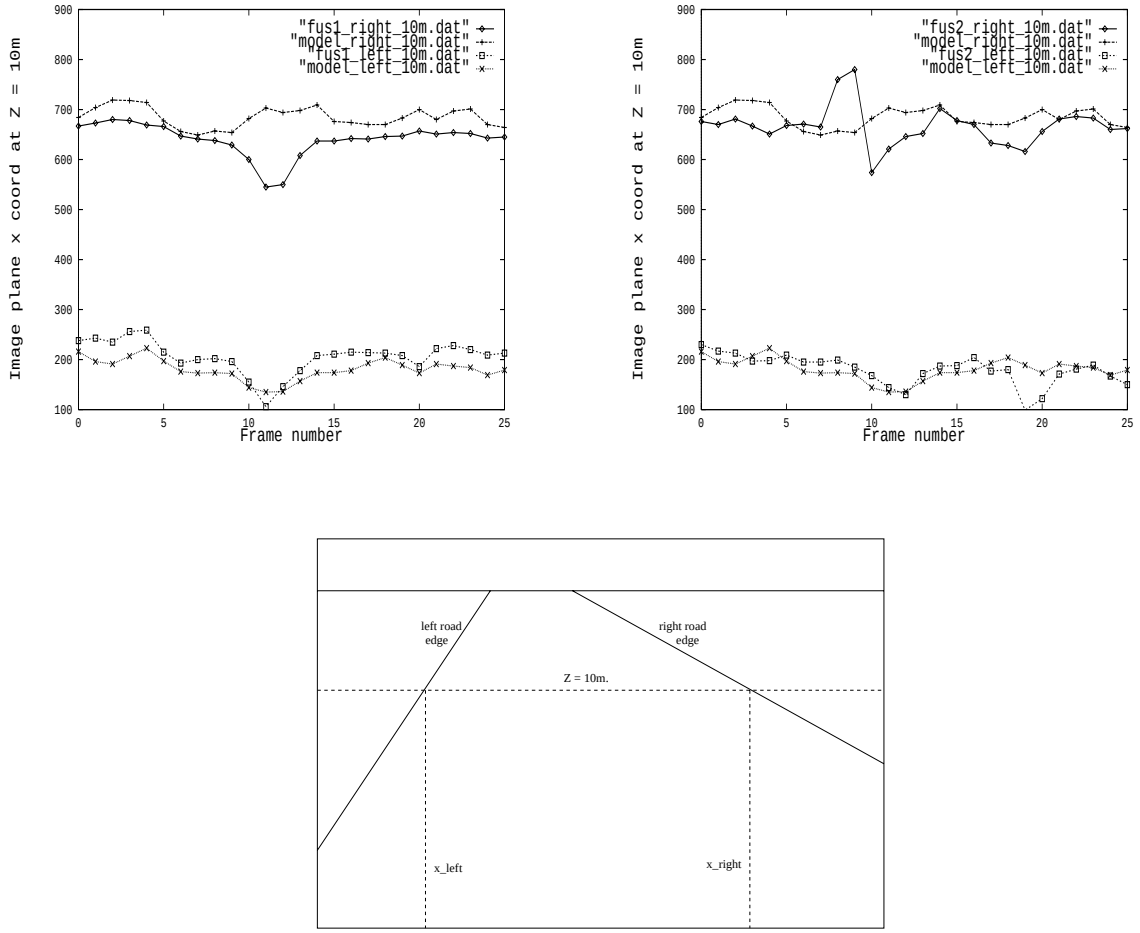


Figure 4. Top-Left: Pixel Voting vs Manually Created at 10m for left and right over 25 frames, off-line. The manually created results are shown by the broken line marked with pluses (+) and crosses (x), for right side and left side respectively. Top-Right: Model Voting. Bottom: Diagram showing how the x values used in the top-left and top-right graphs above were obtained.

The real-time implementation on video sequences was tested with a video of a one-mile stretch of varied road. The stretch was two lanes wide throughout, and included bends, junctions on both sides of the road, a central white line whose duty cycle varies considerably, and different road textures. At the road junctions, there were no indicative road markings at the road edges representing the missing kerb. The whole stretch was followed by both fusion methods; both were considerably more reliable than any of the road following algorithms when they were run individually (more details and results on image sequences in [21]).

The vehicle is used to demonstrate the road following system, and to test the theory of fusing the

vehicle feedback into the road estimate. Since the processing speed of the fusion module in the video trials was approximately 3Hz, the effective difference between consecutive results is about the same as the difference between the images in the off-line sequence used to test the fusion processes above. The model voting strategy that performed so well in the video trials was used for the vehicle trials.

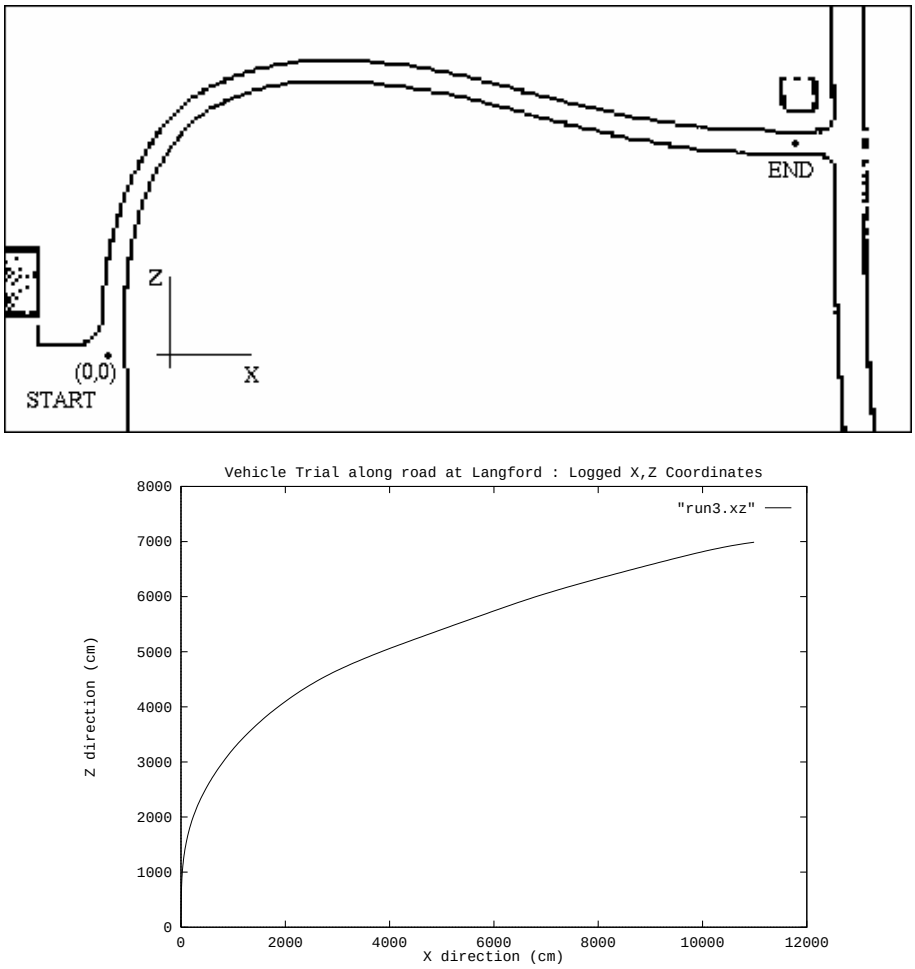


Figure 5. Top: Map of Test Track; Bottom: Logged Positional (X,Z) data

The vehicle trials took place on a stretch of private road owned by Bristol University, approximately 200m long. Unlike the type of road the individual road following algorithms were originally developed on, this road is a single lane track with no road markings and with fences on either side. The only alteration made to the road following software in compensation of this was a halving of the expected road width, and consequently the success achieved in the vehicle trials demonstrates the versatility of the road following system.

Two implementations of the road following system were tested:

1. Using the same system as the one successful in the video trials i.e., not using any data made available by the vehicle itself.

2. Using the vehicle's odometry as an extra sensor and fusing its extrapolated road edges into the road estimate. Also, an approximate positional estimate is maintained by using the history of positional updates.

A video recorder was used to capture the images from the camera mounted on the vehicle along with the overlain fused road edges. Both runs were successful and samples from the video. A brief study of these images reveals no discernible difference between the two runs, and thus there is no immediate evidence that fusing the odometry feedback into the road estimate improved the system.

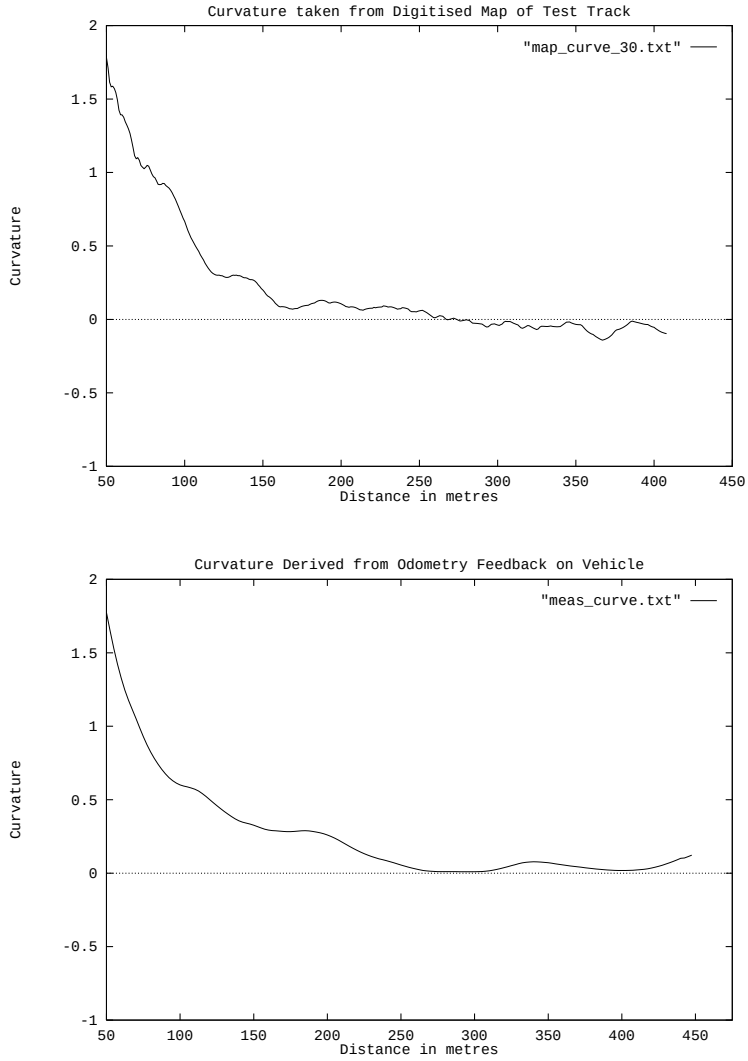


Figure 6. Top: Road Curvature Measured from the Map of the Test Track; Bottom: Road Curvature Calculated using the Vehicle's Odometric Feedback

The inclusion of odometric data into the fusion system does, however, improve the robustness of the system; this is supported by considering the way the odometric data is used, and by considering the logged positional data.

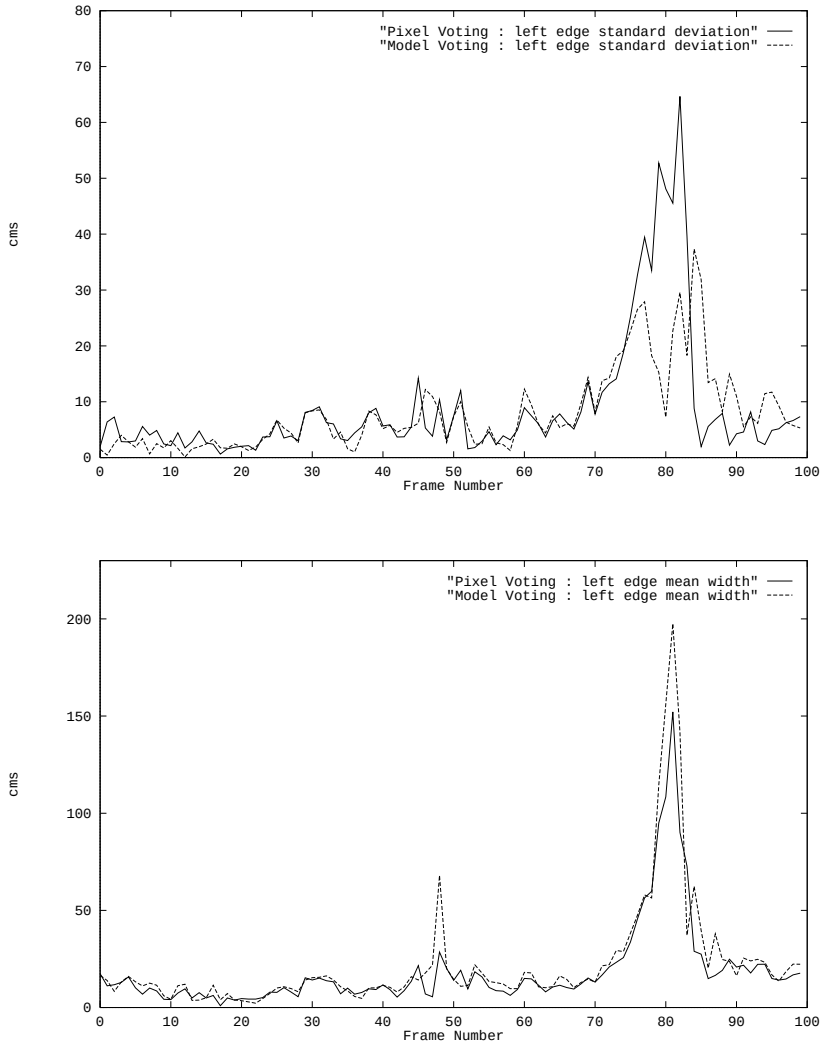


Figure 7. The standard deviation and average width between the fused left edge and the actual left edge, using the pixel voting strategy and the model voting strategy.

The positional data recorded during the vehicle run using odometry is shown in Figure 5 alongside a map of the test track. There is a clear correspondence between the two, although the recorded global position is inaccurate, and this inaccuracy grows with distance travelled. The graph of positional data was produced from measurements made at the same speed as the fusion cycle (about 2 to 3Hz) and thus consists of several hundred measurements. Each of these measurements has an associated error, but if each measurement is considered individually, then that error is small and constant. This is demonstrated in Figure 6 where the curvature of test track calculated using the recorded positional data is plotted below a similar plot of curvature measured from the digitized map of the test track. The similarity between these graphs, in Figure 6, shows that the calculated curvature is a good approximation to the real curvature irrespective of distance travelled. The noise on the curve taken from the map is caused by the low spatial resolution of the digitized map. Therefore, using each individual measurement to extrapolate the previous fusion process's edges produces reasonably good approximations to the new road edges visible from the vehicle's new position. Fusing these extrapolated edges into the new road estimate can therefore only enhance the robustness of the

system and help guard against the (rare) situation when all the vision tasks fail.

Graphs comparing the pixel voting strategy with the model voting strategy are shown in Figure 7. From the graphs it can be seen that the two fusion strategies have similar results; the mean difference and the standard deviation seem to disagree as to which fusion strategy is better. They are effectively equal given that the Y-axis units are in centimetres and thus the differences are minuscule in terms of whether or not the road has been found; and for all 100 images in the sequence, both fusion strategies have indeed successfully located the road.

3. Road Junction Detection

To traverse a road network, an autonomous road vehicle (ARV) must be able to detect road junctions and manoeuvre at road junctions. Over the past few years some studies on road junction detection have been undertaken. The van VaMoRs [22] uses a search process which is specifically targeted at the area beyond the road edge. Arrangements of features indicative of a road junction ahead are searched for in the targeted area. Both edge-and area-based components are utilized when searching for road junction candidates. The IITB [23, 24] uses a straight line model representing the boundaries of the junction and assumes that a gap and a corresponding straight line segment represents the junction. Their algorithms may not recognize the road junction on both sides while the current road continues straight ahead. The SCARF [25] system has detected unstructured road junctions but its algorithm processing time is very long and so it may be unsuitable for real-time applications. In the YARF [26] system, the road junction detection process depends on a gap in the white line on the road surface. That is, every gap in the white line on the road surface is directly accepted as a road junction. In the ALVINN, a virtual camera is used to detect intersections [27]. When the vehicle arrives at an intersection area, the virtual camera is oriented towards the intersection. Then the system deals with information for a single lane road which will be presented at the intersection. The intersection detection algorithm depends on the virtual camera position being directed towards the intersection. The system still had problems navigating onto the branches. In the FMC [28] system, the candidate road edges in gradient images are firstly located and constraints are used to determine the road location. Then breaks in the road edge to allow for lost road edge are smoothed. The algorithm assumes that a section of undetected road edge is a result of the road edge actually being broken. When fitting the intersection model on the undetected road edge, the road junction edges which are other features of the intersections were ignored by the algorithm.

In this section, a computer vision system based on multi-cameras for an ARV that is capable of negotiating complex road networks including road junctions in real time is presented. To detect road junctions, a model-based approach is used. In this approach, two main steps are performed. The first step is to find a boundary between the current road and the start of the road junction. The second step is to perform a localization process on the road junction surface and edges. In this second step, two different low-level image processes are fused using a model voting strategy. The two low-level image processes used are region growing segmentation and the detection of the road junction edges in edge segments produced using a Sobel operator.

To manoeuvre at an intersection, a vision-based multi-camera re-orientation mode using a “bootstrap” process has been developed. The “bootstrap” process, which is used to determine the location of the initial driving corridor and the position of the vehicle with respect to the road junction, is also vision based.

3.1. Road Junction Model

The road junction model for road junction detection algorithm contains the following values:

Road junction model : { B, $I_x, I_y, \Theta, l_{jn}, l_{road}$ } as shown in Figure 8.

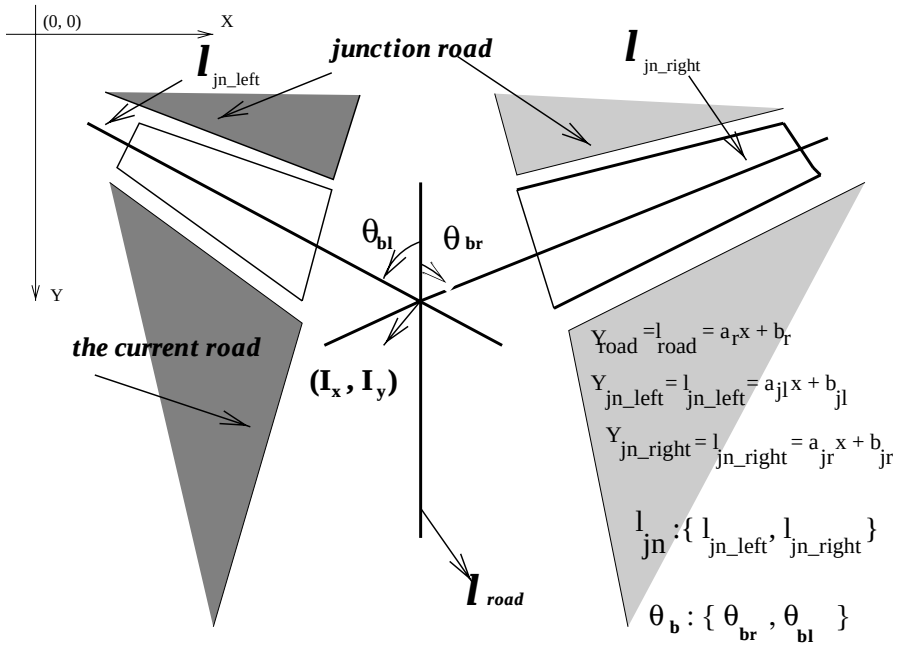


Figure 8. The local geometric model of the junction road.

B is number of branches in the road junction model.

(I_x, I_y) is the intersection at the common point of the mid lines which pass through the branches, and through the current road.

Θ_b is the angle between the mid line of the current road and each of the mid lines of the branches.

l_{jn} is the equation of each mid straight line of the branches.

l_{road} is the equation of the mid straight line of the current road.

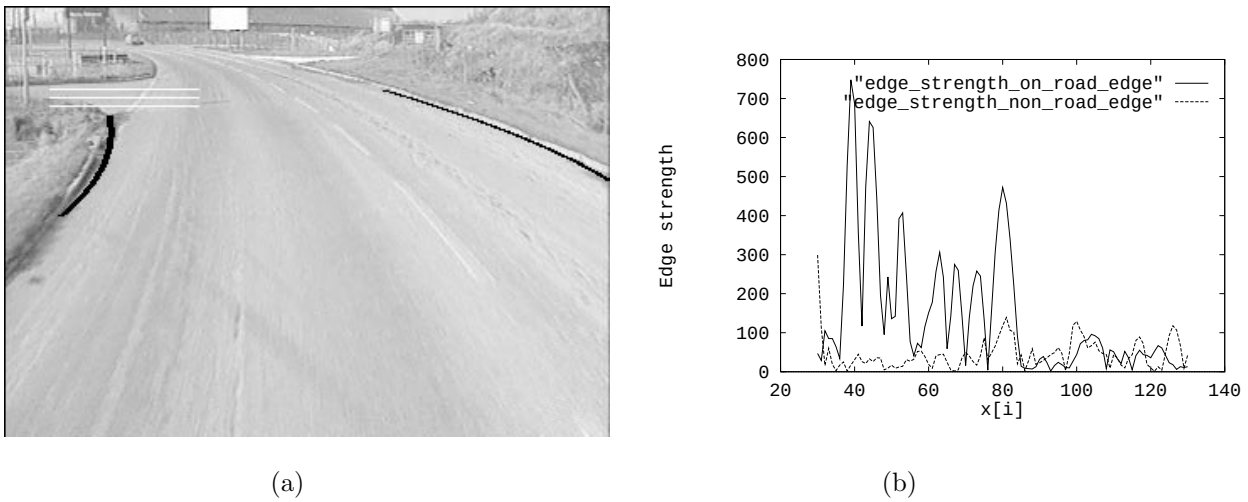
3.2. The Road Junction Procedure

To determine the position of the road junction, two main steps are performed. The first step is to find a boundary between the current road and the start of the road junction based on information from the map database. The second step is to locate the road junction surface using a localization process. The map database provides some data about the expected road junctions like the expected road junction is on the left, there is no junction marking on the boundary, the expected road junction intersects at a right angle to the current road, etc. Prior knowledge may be necessary for the improvement of the road junction model in order to more correctly apply the model with the results produced by low-level processes.

In the first step, there is a need to develop two different algorithms, one for junctions with road markings along the boundary between the current road and candidate road junction and another for junctions without such markings. An earlier study has addressed the case where the road markings are on the boundary [29]. In this study there will be no road markings along the length of the boundary. Consequently, this first step is to cue the road junction surface localization process, which is performed as a second step in our algorithm.

3.2.1. Finding the Boundary

The process to find the boundary between the current road and the start of the road junction is to detect, where there is no road edge in the current frame, where a road edge would be expected based on information from previous frames. It is assumed that the variation in the positions of road edges from frame to frame are small. Three steps are involved in this process. In the first step, the edge detection process is used to determine that the length of the boundary without road edge is suitable for a minimum width of a junction road.



(a) (b)

Figure 9. a) Typical road junction b) Output of the edge detector

In the second step, texture structures using an edge-detector process on small areas of the candidate broken road edge are determined. The small areas of the candidate broken road edge used are shown in Figure 9a as the areas between the three white lines located in the area of the road junction. The results of this process applied to the area of the top white line in Figure ha, are shown in Figure 9b. In Figure 9b, the x[80] point shows a road edge point in the image. To the right side of the x[80] point the texture structure of the current road surface, as found using an edge detector process, is shown, while to the left side the texture structure of the junction road surface is shown where there exists a junction road. In Figure 9b the unbroken line shows the texture structure around the road edge when there is no junction road surface. The broken line shows the texture structure on the white line at the top of the three white lines in the road junction in Figure 9a. The texture structure represented by the unbroken line around the x[80] point does not have similar characteristics and it says that there is no junction road surface at that line position. The other texture structure represented by the broken line around the x[80] point has similar characteristics and it suggests that there is a junction road surface at that line position.

In the third step, average intensity values are determined around the broken road edge using the same areas of this image as used in the previous texture structure process. When the road edge has been broken as shown in Figure 9a, the texture structure and average intensity values on both sides of the broken road edge are the same. Thus, the result from these processes is the identification of a length of boundary, texture structure and intensity value which could possibly be due to a road junction based only on road edge information.

3.2.2. Locating the Road Junction Surface

Results of a region growing segmentation process and an edge detection process using edge segments are fused to locate the candidate road junction surface. These two low-level image processes are run in parallel on the hardware system [30].

In the region growing segmentation process, a small area of the current road surface is used to approximate the average road surface intensity. This is shown as four small closed black squares in Figure 10. Note that these black squares are at the mid point of the candidate road junction area in the real world. To verify that the small areas selected from the image are road surface, two other small areas are selected from the road surface at locations near the vehicle. These are shown by the black squares in Figures 10. If the average of the four small areas selected do not have intensity characteristics similar to those of the two larger squares selected then it must be assumed that in at least one case the road surface has not been located. The process of area selection is then repeated until the road surface is located. The average intensity value of the current road surface will be used in a region growing segmentation process to identify the candidate road junction surface. Each average intensity value of 2x2 pixel regions, in 360 by 256 pixels resolution of the images, on the candidate road junction area is compared with the average intensity value of the current road.

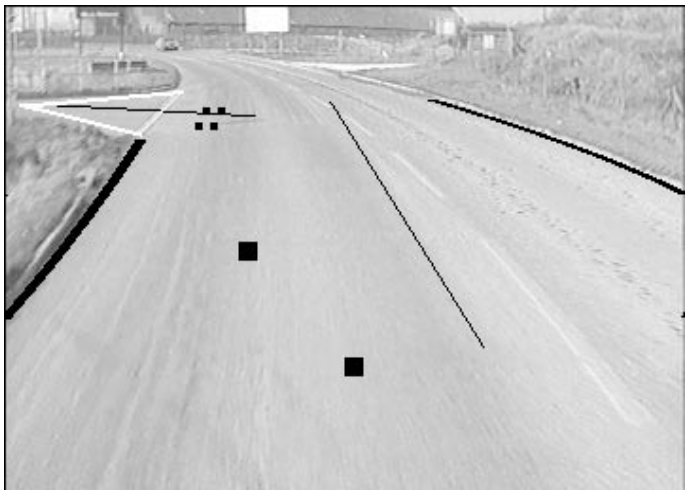


Figure 10. Region growing segmentation results.

All adjacent average intensity regions similar to the current road average intensity are joined by a process making use of positional information. The similar average intensity 2x2 pixels regions in the column nearest the current road surface are first joined starting at the mid point of the column. The top and bottom points of the column are noted. This process is then repeated for the next adjacent column of similar average intensity 2x2 pixel regions. This continues until all of the similar average intensity regions are joined. Thus, a list of top and bottom column positions is generated. This list is then used to fit two straight lines to the junction road edges by a least squares method. An example is shown in Figure 10; the fitted lines are white. When the road junction surface has brightness different from its own road surface, this low level image process would not produce data which is used the following fusion process. Therefore the results of the fusion process depend on other low level image processing results.

The other low level image process is to determine the road junction edges. The junction road edges are detected as edge segments using a horizontal convolution mask. The road junction edges above and

below are detected. This is achieved by a filter mask to detect edge segments which correspond to the road junction edges in images. This filter is of the form

$$[-1 \quad -2 \quad -4 \quad -2 \quad 0 \quad 2 \quad 4 \quad 2 \quad 1]$$

It is a first derivative of a Gaussian, which gives the optimal response to step edges [31]. The width of the filter mask is set to give the best response for the poor contrasting edges encountered and the integer values are powers of two so as to avoid multiplication by using a bit shift operator for faster processing time.

Thus a set of edge segments for the top and bottom of the candidate road junction in the image is obtained. A model of the candidate road junction is then developed by fitting a straight line function to the edge segments for both sides of the road junction using a least squares method. The results of edge detection using edge segments are shown as white lines in Figure 11.

Figure 11. Results of edge segments process using edge detector.

In the second step, the equations of the middle lines which pass through the candidate road junction surface are derived from the road junction surface data. The middle line of the current road is derived from the equations of the current road edges. These are shown as black lines in the middle of the roads and in the middle of the road junction in Figures 10-11. The results of the two different processes (region growing process in Figure 10, edge detection using edge segments in Figure 11) are transformed into real-world coordinates, taking the camera's perspective into account [1] with the transformation equations [32]. These results in the real-world coordinates are fused using a model voting strategy. In the fusion process, weight parameters of each process are determined by the road junction surface area and the point and angle of intersection of the lines that pass through the current road and the candidate road junction surface [29].

As a result of each individual process to determine the road junction surface the model is fitted to its data in the form of two real-world straight lines and a weight. The model's straight line coefficients are then calculated as follows:

$$F_i = \sum_{j=1}^{n} w_j * a_{ji} \tag{11}$$

where F_i is the fused straight line coefficient (i being 0, 1), j is the algorithm's index, w its normalized

weight, a_i its straight line coefficient and n is the number of algorithms being fused together. The result of this fusion process is shown in Figure 12.

If these data produced above are within specified limits, then it is decided that a road junction exists at this point and the algorithms will track the position of the road junction. Otherwise, the algorithms will restart automatically for the next frame.

Figure 12. Results of fusion process using model voting strategy.

After the position of the road junction is determined, its position is tracked relative to the position of the ARV until the start of the manoeuvre process to turn the ARV towards the road junction. In addition, the parameters of the road junction such as the road and road junction width and intersection angle are updated in every frame during the tracking process. The road edges are represented by the coefficients of second order equations and are tracked by the road edge tracking algorithm [1].

4. Multi-Camera Based Manoeuvres at a Road Junction

4.1. Camera Arrangement

A TV camera is used as an input device for a machine vision system which comprises a lens system, a light sensitive sensor area, and appropriate amplification electronics. According to the camera orientation, the field of view is determined. The resolution of the sensor area is limited due to interlacing and motion blur. Therefore only every second horizontal line can be used but the resulting depth resolution decreases with an increasing field of view. This forces a trade-off between conflicting requirements – a large field of view and a high depth resolution [24].

There are two common solutions to this dilemma. One is used in the VaMoRs-Vehicle of the Universitat der Bundeswehr in Munich, Germany [33]. They use two cameras with lenses of different focal length; both are fixed together on a pan-and-tilt platform to supply the requested field of view and depth resolution to guide a vehicle on a highway. Another solution is to use a set of cameras fixed to the vehicle. In reference [26], they experiment with this configuration. With the ever decreasing expenses for electronic devices we favour the second approach.

Our experimental vehicle is equipped with three cameras fixed on the top of the vehicle at the front [11]. One camera is headed along the longitudinal axis of the vehicle, and the other two are oriented leftward

and rightward, respectively. The cameras are calibrated with respect to the vehicle. The frame grabbing of the cameras are synchronized, so we can change between the appropriate viewing direction without losing a single image. For our task of navigating the vehicle autonomously on private ground with its narrow roads and sharp curvatures, this configuration was adequate.

4.2. Manoeuvring

Roads on private premises have to cope with severe limitations regarding available space and therefore one can often find perpendicular junctions with very small radii. This challenging situation requires a junction-turning manoeuvre that uses all available information from the map and from the image sequence gathered by the camera. This combination results in a junction-turning manoeuvre totally controlled by machine vision without any blind phases.

Some research studies on the manoeuvring at the road junction have been undertaken. In [23, 24], the steering wheel angle during the manoeuvring is provided from the tracking of the crossing straight line. When the straight line is missed, the manoeuvring process may be broken. In this study, the steering wheel angle is obtained by determining the direction of the road junction without needing to track the road junction edge. In [22], the necessary angle data for the steering wheel is produced by the active camera control algorithm which depends on the road junction edges. The features of the active camera, especially with regard to supplying the requested field of view, represent a considerable expense when developing the system. A set of cameras fixed to the vehicle is used in this study to obtain similar features of the active camera for manoeuvring at the road junction. In this work, the necessary angle for the steering wheel of the vehicle during the manoeuvring at the road junction is produced by determining the direction of the road junction using a re-orientation mode in real time.

To explain the manoeuvring process in this study, the turning-left manoeuvre in the vicinity of a road junction will be described in more detail. The vehicle is guided by tracking the left and right lane boundaries. When the road junction is determined by image processing and map information, the relevant boundaries of this junction (white lines in Fig. 12) are used to follow the position of the road junction with respect to the vehicle in the tracking module. Tracking starts in the images of the centre camera and is switched to the left camera automatically as the vehicle approaches the intersection. The manoeuvre at the road junction is started in the image sequence of the left camera and is switched back to the centre camera as the turning vehicle has reached a predefined orientation towards the goal orientation. During the manoeuvre process, a re-orientation mode is used to select the appropriate camera and to produce the steering command for the vehicle control.

4.2.1. The Re-orientation Mode

The re-orientation mode was designed to orient the vehicle on the road junction such that the camera is pointing along the road junction with the vehicle correctly placed within the road junction's bounds. The vision process in this mode is performed on the images which are produced by the multi-camera based vision system. In the multi-camera based vision system, the basic step is to determine which camera is active at the moment. Then the results of the processes are implemented according to the active camera position on the vehicle.

When the vehicle approaches the intersection, a central controller module selects the left camera (for left side intersection) automatically, and then the manoeuvre process is started. Examples are shown in Figure 13a-b. Figure 13a is obtained from the left camera and Figure 13b is obtained from the centre

camera at the same time.

(a) (b)

Figure 13. a) A view from the left camera b) A view from the centre camera at the sametime.

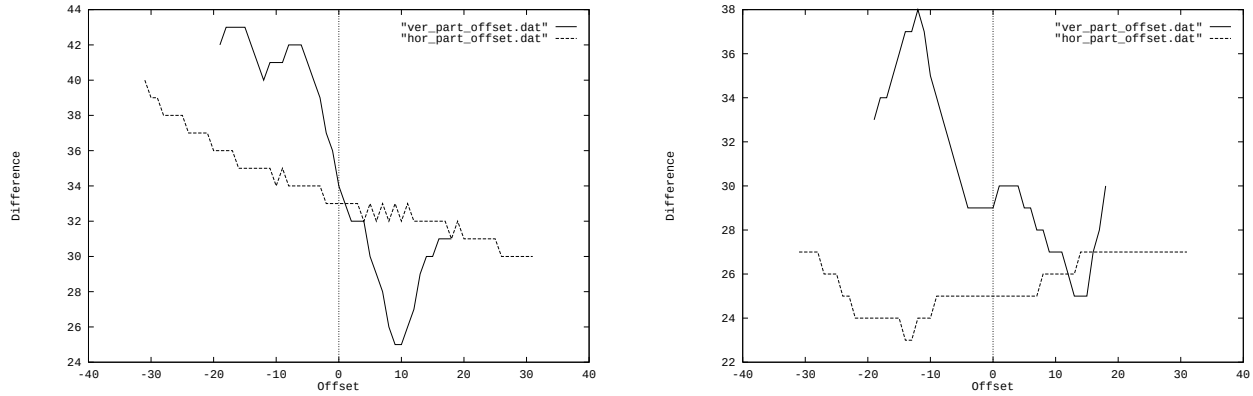
To control the vehicle during the manoeuvre process at the road junction, a control command is produced which depends on a vanishing point of the road junction in the image. The vanishing point of this section of the road junction, V_p, is determined first and this indicates the direction of the road junction with respect to the vehicle. For instance, in the top images in Figure 17a, the black cross indicates the detected vanishing point for the particular road. For the vanishing point to be found, the vertical line in the image, where the horizon may be located, is calculated using the camera parameters and the result of the matching process following.

Four sub-images, two from the horizontal bands (in Fig. 15b) and two from the vertical bands (in Fig. 15a), are sampled in the image as shown by the black and white samples in Figures 15a-b. For both horizontal sampled masks, a set of vertical image values is chosen in front of the vehicle that are equally spaced in the real-world. The lower grid's horizontal image values are then calculated to give a predetermined value of horizontal samples. Using the same real-world sampling dimensions, the upper horizontal image sample locations are calculated. Using the same spacing gives a wider spread of horizontal samples and thus a wider sample is generated. For both vertical sampled masks, two parts of the image, which are equally spaced in the real world, are chosen for the matching process [34, 11].

The four sampled sub-images are transformed into real-world coordinates, taking perspective into account [32]. The sub-sampling is correlated across the length of the other sub-image on the same bands and gives an output showing the match between the two sub-images as shown in Figures 14a-b. The matching process is computed as

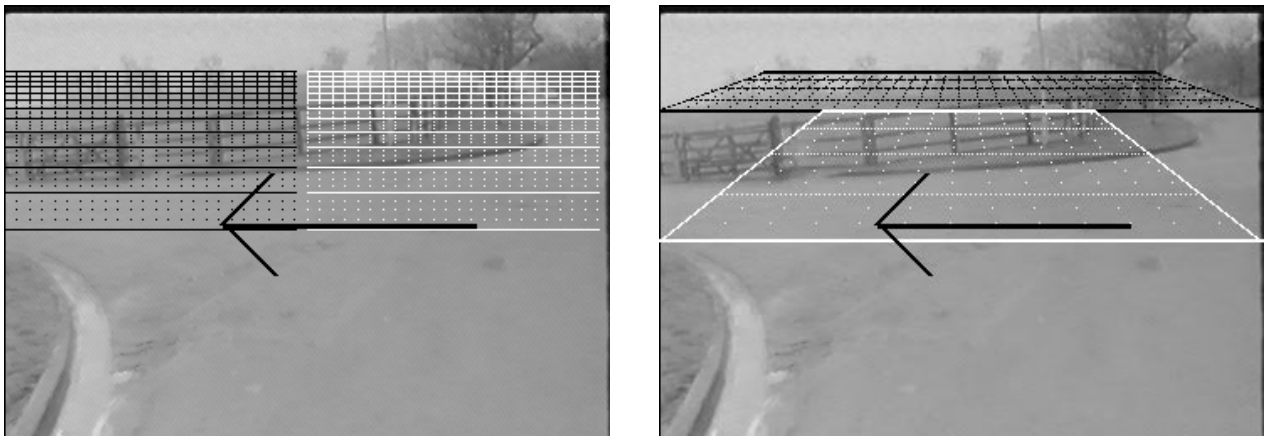
$$\Delta[d] = \sum_{y=0}^{H_L} \sum_{x=0}^{W_L} |i_L[x][y] - i_H[x+d][y]| \tag{12}$$

where $\Delta[d]$ is the difference between the two sampled images i_H and i_L at offset d, H_L and W_L are the height and width of the lower sampled image for horizontal bands or a sampled image near the junction road way for vertical bands respectively.



(a) (b)

Figure 14. Matching of the two vertical and two horizontal sample images, a) For Figure 14a b) for Figure 15a.

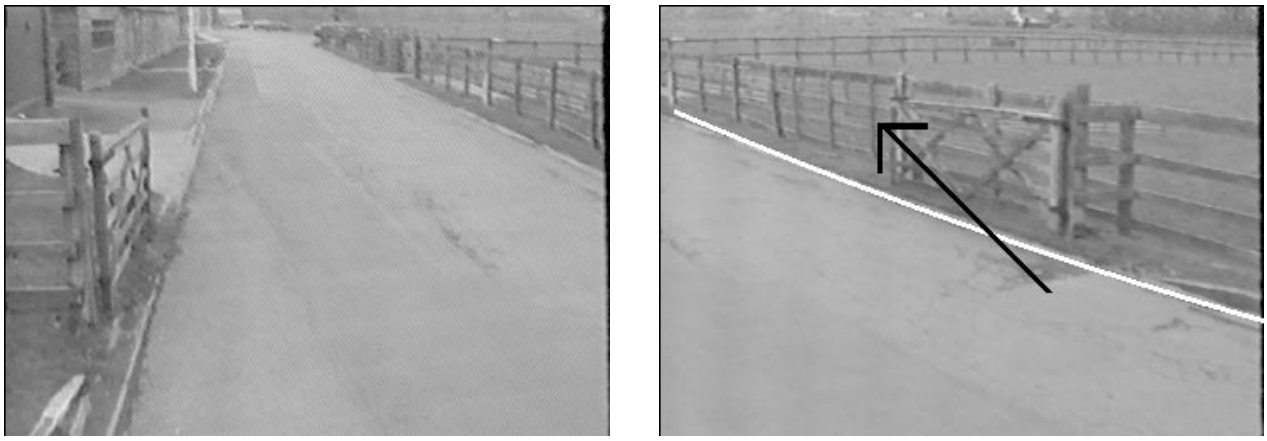


(a) (b)

Figure 15. a) Vertical bands on the image, b) Horizontal bands on the image.

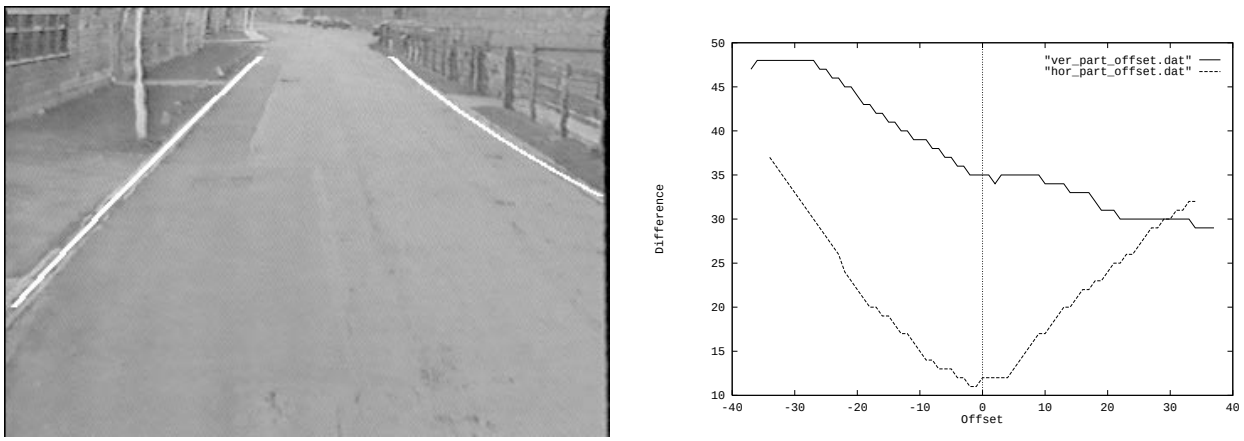
The minimum of the matching function indicates the best match. The best match shows the position of the vanishing point in the images. In Figure 14, the broken lines and unbroken lines show the matching process on the horizontal bands and on the vertical bands respectively. Figure 14a shows the results of the matching process for images 15a-b. When comparing the two matching results in the graphs, the lower minimum of the two indicates the overall best match. From the results (Figure 14a) the best match achievable is shown to be by vertical matching, which is represented by the unbroken line. The best match, or minimum, is for an offset value, which indicates that the vanishing point lies far to the left. In other words, in Figures 15a-b, the vanishing point is out of range of the horizontal matching and the best match is given by the vertical matching. Therefore the vehicle will be turned to the left with the maximum angle.

Having identified the vanishing point as described above, the angle between the vanishing point and the centre of the image (depending on which camera is currently active) gives the bearing the vehicle has with respect to the road junction. This angle is used to turn the vehicle as shown by the black arrows in Figures 15 and 16b.



(a) (b)

Figure 16. a) A view from the left camera, no longer used b) Centre camera is active again at the next frame.



(a) (b)

Figure 17. a) An image, produced by the centre camera, after the vehicle is oriented on the road junction. b) Matching of the two vertical and two horizontal sample images.

In the subsequent images, the vanishing points enters the scene and the best match is taken from the horizontal matching as shown in Figure 14b. The results in Figure 14b are for the matching of the two vertical and horizontal sampled images in Figure 16a. When the best match is taken from horizontal matching as shown in Figure 14b, then the centre camera is selected automatically by the central controller. Then the process to determine the vanishing point in image produced by the centre camera is repeated to orient the vehicle on the road junction as shown in Figure 16b. In the Figure 16b image which is produced by the centre camera, the vanishing point is out of range of the horizontal matching and the best match is given by the vertical matching. Therefore the image processing monitors the resulting motion, computing the new vanishing point from an image sequence which is produced by the centre camera until the vanishing point lies approximately centrally in the image as shown in Figure 17b.

The algorithm assumes that one of the junction road edges is visible until the vanishing point lies roughly centrally in the image as in Figures 15 and 16. When this occurs it is assumed that both road junction edges are then visible, as shown in Figure 17a, and the main road tracking process then restarts as explained in Section 2.

4.3. Experimental Results

The algorithms were tested on a golf buggy within the University campus and on vision hardware systems consisting of transputer-based frame stores [35]. Different sequences, which include road junctions on the left and right hand side, have been captured from three video cameras mounted on the roof of a car travelling on public roads in the Bristol area and within the University campus. To test the algorithms on a real-time set of images, the code was ported to specially designed hardware consisting of transputer-based frame stores, with each algorithm (road following, road junction detection and following and manoeuvre) running as a parallel task on separate frame stores as shown in Figure 18. A control process, running on another frame store, is permanently waiting for results from the road following and road junction detection and following algorithms. For each frame, the control process receives the results and gives the final decision using the most recent data from each algorithm above. The real time implementation was tested on sequences of road images including road junctions. These sequences were collected from a video camera as explained above.

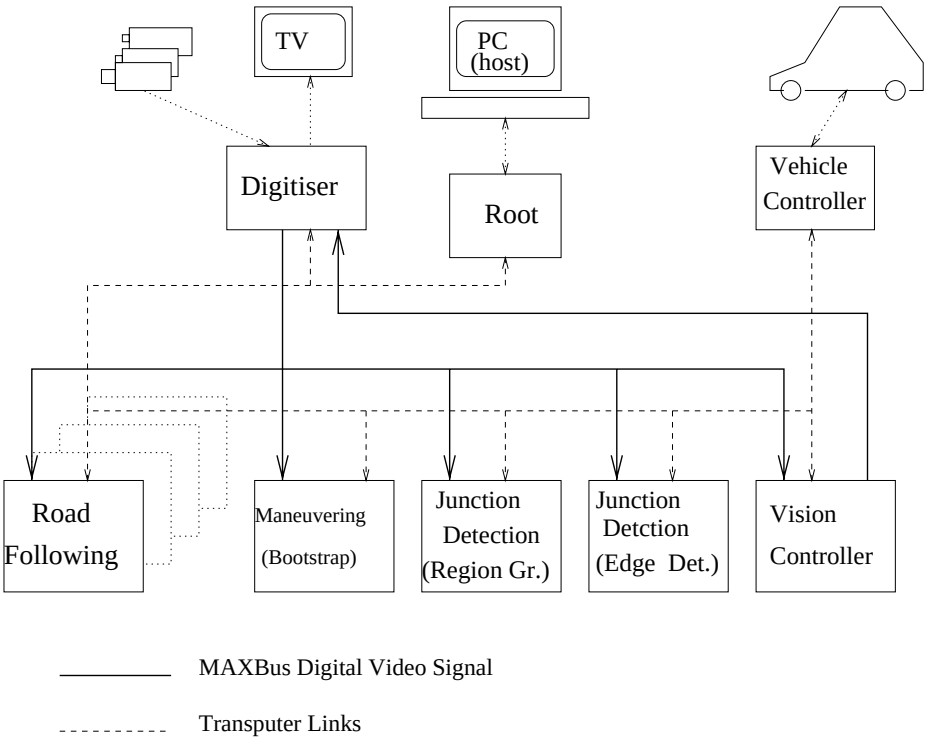


Figure 18. System architecture for the autonomous road vehicle.

In the real-time process, the processing time to detect the position of the road junction in sequence images is 165-195 milliseconds. A manoeuvre trajectory with the turning angle derived from the results of the image processing is updated with a rate of about 10 Hertz. These processes times were obtained on the hardware system consisting of five frame store boards and one digitizer board. Each frame store board consists of one 20 MHz T805 transputer frame grabber with a memory cycle time of 150 ns. The digitizer board consists of video A/D and D/A converters for capturing and displaying images. All programmable options within the module are conFigured by an on-board T225 transputer, which responds to commands from a master transputer [4]. The algorithms were programmed in ANSI C and compiled on a B004 transputer PC board running the Inmos Transputer Development System.

Figure 19. Navigating sequence in complex road network includes road junction.

The parallel architecture is entirely modular and capable of real-time analysis, control and overall supervision of the vehicle. All modules in the architecture are managed and interfaced with each other using a single technology, the Inmos transputer [36].

The algorithm was tested on our experimental vehicle on a private road network owned by the University of Bristol. In the test processes the vehicle was successfully navigated on the private road network. Figure 19 shows a time sequence in the test processes for the autonomous road vehicle navigating a complex road network which includes a road junction.

5. Conclusions

The algorithm in Section 2 was tested by itself and the algorithms in Sections 3 and 4 were combined for testing. Both tests were performed using the vision hardware running on the vehicle within the university campus. Figure 18 shows an arrangement of the system architecture that has been used for experimentation with road navigation software. Each vision module is placed on its own frame store and accesses the same raw digital video data. A road following algorithm based on the road edge detector in the image plane explained in Section 2.1 was placed within the vision system at the test process for the algorithms in Sections 3 and 4 as shown in Figure 18. If it is desired to combine the algorithm in Section 2 with the algorithms in Sections 3 and 4, the algorithm in Section 2 can be easily installed onto the vision modules which are shown by broken line rectangles in Figure 18.

Four near real-time road following algorithms were implemented and tested, firstly in an off-line situation on a sequence of pre-recorded road images, and then on a video of the road. All demonstrated a reasonable degree of success, but there were situations where each algorithm failed.

The features tracked by the algorithms are

- Road texture.
- Road edges. Two techniques are used; one in the image plane, one in the world plane.
- Central white line.

These features are not always present in a road scene (or, in the case of the texture, may either be irregular, due to shadows, or similar to the surroundings especially in the case of junctions) and so relying on any one as a road navigator would not be satisfactory.

To provide a consistently successful navigation system in real time without a priori knowledge of every nuance of the target road, the algorithms are implemented concurrently and their results are fused together to provide a global road estimate.

The fusion algorithms fuse the incoming road data into a global world-plane road model, where the left and right road edges are modelled as second order quadratic equations. This road model is considered sufficiently robust and versatile that it can be used to model a wide variety of roads, and is also a compact representation that can be easily stored for later analysis.

The spatial fusion techniques considered, the pixel voting strategy and the model voting strategy, both demonstrated a high degree of success. They both use the same fusion method, a weighted averaging approach, and are thus fully dependent on the self-assessed weights provided by the road following algorithms. A comparison of the two algorithms, given in the Table, implies that for a real-time vision-based road following algorithm fusion technique, the model voting strategy is superior.

Table Comparison of the Pixel Voting Strategy with the Model Voting Strategy.

Pixel Voting Fusion	**Model-Based Fusion**
A priori Threshold needed	No A priori Data needed
Uses high level road models from individual road following algorithms to vote into a low level pixel-based voting area. Data is unnecessarily clipped here, especially for the texture segmentation.	Keeps rate of change of data abstraction constant throughout road following process.
Fast execution time	Slow execution time

Despite the fact that both the fusion algorithms are successful over the test image sequence, there are other issues that should be considered, and these are highlighted in Table. In theoretical terms, the model-based fusion technique is better since it fuses road models to a single road model in a single step whereas the pixel-based fusion takes the road models, uses them to generate low level pixel data, and fuses the pixel data, then generates a final road model. To skip the first model fitting stage and hence avoid unnecessary data clipping, the pixel-based fusion could benefit from taking road data from the individual road following algorithms in pixel format, but this would only suit the texture algorithm and would cause vast amounts of data to be transferred. However, the individual algorithms are all model based and so it is intuitive that the fusion is performed on the road models. Since it is likely that other road following algorithms designed at a later date will also be model based it will be simpler to integrate them into the system if the fusion is performed at the road model level. A final point is that the system is aimed at real-time road following, so fusing small quantities of high-level data is considerably more practical than fusing large quantities of low-level data.

A model-based approach using a fusion process based on a model voting strategy was described to detect road junctions in real time in Section 3. The algorithm to detect the road junction in real time was tested on different both side road junction sequence images captured by the video camera and on the test vehicle. The road junction detection algorithm consists of two sequential algorithms. They find the boundary between the road and road junction and locate the road junction surface. The advantages of this algorithm are that the search time to find the road junction is decreased, and a more robust road junction surface location is obtained using a fusion process clarified above. Different types of road junction were

successfully tested. If there is any large obstacle near the road junction, and if it is not allowed to obtain road junction information (both edges and surface), the algorithm will not detect this type of road junction unless high level information on the navigation environment is used.

A multi-camera vision-based re-orientation mode developed for manoeuvring at road junctions was explained in Section 4. During the manoeuvre at the road junction, the vehicle was controlled by machine vision without any blind phases. Control commands on the vehicle during the manoeuvre process are updated at a rate of about 10 Hertz. The advantage of the manoeuvring algorithm at the road junction is to present less expense than [22] and a more reliable and faster execution time than [24]. These are achieved by using parallel processing on the frame store boards which were in the vehicle during its navigation on the road. If a part of the road junction is seen in the image during the manoeuvring process, robust results are obtained by using this low-level data directly. On the other hand, good results were also achieved using high-level information processing in the manoeuvring process.

The more robust road following algorithm was tested separately, while the road junction detection and manoeuvring at the road junction algorithms were combined and tested on the golf buggy. The golf buggy was successfully navigated in a complex road network including a road stretch and road junction. An example of the navigating sequence in the complex road network including a road junction is shown in Figure 19.

The work in this paper was performed to navigate an autonomous road vehicle in a real-world road network which includes a road junction in a fairly constrained environment. To enhance the road junction detection algorithm, it should be able operate in more challenging environments within real traffic (including cars, pedestrians, traffic signs etc.) and with more extensive interaction with higher level knowledge.

References

[1] Thomas B. T., Dagless E. L., Milford D. J., Morgan A. D., "Real-time Vision-Guided Navigation". Engineering Application of Artificial Intelligence, **Vol. 4**, pp. 287-300, 1991.

[2] Thorpe C., Hebert M. H., Kanade T., Shafer S. A., "Vision and Navigation for the Carnegie-Mellon Navlab". IEEE Trans. on Pattern Analysis and Machine Intelligence, **Vol. 10**, No. 3, May 1988, pp. 362-373.

[3] Dickmanns E. D., Behringer R., BrudigamC., Dickmanns D., Thomanek F., Holt V. v., "An All-Transputer Visual Autobahn-Autopilot/Copilot". Proc. 4th Int. Conference on Computer Vision ICCV '93, 11-14 May 1993, Berlin/Germany; IEEE Computer Society Press; pp. 608-615.

[4] Dikmanns E. D., "Performance Improvements for Autonomous Road Vehicles", Int. Conference on Intelligent Autonomous Systems (IAS-4), Karlsruhe, March 1995.

[5] Maurer M., Behringer R., Furst S, Thomanek F., Dickmanns E. D., "A Compact Vision System for Road Vehicle Guidance", IEEE Proceeding of 13th International Conference on Pattern Recognition, **Vol.3** pp. 313-317, Vienna Austria, August 1996.

[6] Ulmer B., "VITA- An Autonomous Road Vehicle (ARV) for Collision Avoidance in Traffic". Proc. Intelligent Vehicle '92 Symposium, Detroit/MI 29 June 1992, pp.36-41.

[7] Franke U., "Real-time 3D Road Modelling for Autonomous Vehicle Guidance", Proceeding of the 7th Scandinavian Conf. on Image Analysis, pp. 316-323, Aalborg, Denmark, August 1991.

[8] Hattori A., Hosaka A., Taniguchi M., Nakano E., "Driving Control system for an Autonomous Vehicle Using Multiple Observed Point Information". Proc. Intelligent Vehicles '92 Symposium, Detroit/MI, 1992 pp.

[9] Cox I. J., "Blanche -An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle", IEEE Trans. on Robotics and Automation, **Vol. 7**, No. 2, pp. 193-204, April 1991.

[10] Leonard J., Durrant-Whyte H. F., Cox I. J., "Dynamic Map Building for an Autonomous Mobile Robot", The International Journal of Robotics Reseacrch, **Vol. 11**, No. 4, August 1992.

[11] Ekinci M., "Computer Vision Applied to the Navigation of an Autonomous Road Vehicle in Complex Road Networks". PhD Thesis, Department of Computer Science, University of Bristol, 1997.

[12] Ekinci M., and Thomas B. T., "Model-Based Autonomous Road Vehicle Navigation in Complex Road Networks", 2nd International Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey, August, 1998.

[13] Thorpe C., Herbert M., Kanade T., Shafer S., "The New Generation System for CMU Navlab". Vision-Based Vehicle Guidance, ed. Ichiro Masaki, Springer-Verlag, Chapter 2, 1992.

[14] Sharma U. K., Kuan D., " Real-Time Model-Based Geometric Reasoning For Vision-Guided Navigation". Machine Vision and Applications, 2, 31-44, 1989.

[15] Turk M. A., Morganthaler D. G., Gremban K. D., and Marra M., " VITS - A Vision System for Autonomous Land Vehicle Navigation". IEEE Trans. Pattern Analysis and Machine Intelligence, 10, no 3, 342-361, 1988.

[16] Davis L. S., DeMenthon D., Dickinson S., and Veatch P., "Algorithms for Road Navigation". Vision-Based Vehicle Guidance, ed. Ichiro Masaki, Springer Verlag, Chapter 3, 1992.

[17] Dickmanns E. D., Graefe V., "Applications of Dynamic Monocular Machine Vision". Machine Vision and Applications 1, no 4, 241-261, 1988.

[18] Morgan A. D., Dagless E. L., Milford D. J., and Thomas B. T., " Road Edge Tracking for Robot Road Following". Proc Fourth Alvey Vision Conference, University of Manchester, 178-184, 1988.

[19] Lotufo R. A., Dagless E. L., Milford D. J., and Thomas B. T., "Road Edge Extraction using a Plan-View Image Transformation". Proc Fourth Alvey Vision Conference, University of Manchester, 185-190, 1988.

[20] Gibbs F. W. J. and Thomas B. T., "The Fusion of Multiple Image Analysis Algorithms for Robot Road Following". 5th Int. Conference on Image Processing and its Applications, Heriot-Watt University, U.K., 394-399, 1995.

[21] Gibbs F. W. J., "Vision Based Autonomous Road Following". PhD Thesis, University of Bristol, 1996.

[22] Muller N., Baten S., " Image Processing Based Navigation with an Autonomous Car". Int. Conference on Intelligent Autonomous Systems (IAS-4), Karlsruhe, March 1995.

[23] Struct G., Geisler J., Laubenstein F., Nagel H. N., Siegle G., "Interaction Between Digital Road Map Systems and Trinocular Autonomous Driving". Intelligent Vehicles '93 Symposium, July 1993, Tokyo, Japan.

[24] Struct G., Geisler J., Laubenstein F., Nagel H. -H., Siegle G., " Multi-camera Vision-Based Autonomous Manoeuvring at Road Junctions". Proc. Intelligent Vehicle '94 Symposium, October 1994, Paris/France.

[25] Crisman J. D., and Thorpe C. E., "SCARF: A Color Vision System that Tracks Roads and Intersections". IEEE Trans. on Robotics and Automation, 9:1, Feb. 1993.

[26] Kluge K. and Thorpe C. E., "Intersection Detection in the YARF Road following System". Int. Conference on Intelligent Autonomous Systems (IAS-3), Pittsburg, P.A. February, 1993, pp. 145-154.

[27] Jochem T. M., Pomerlau D. A., Thorpe C. E., "Vision-Based Neural Network Road and Intersection Detection and Traversal". Proc. of IROS 95, Vol. 3, pp. 344-349, USA, 1995.

[28] Kushner T. R. and Puri S., "Progress in Road Intersection Detection for Autonomous Vehicle Navigation". SPIE Mobile Robots II, vol.852, pp.19-24, 1987.

[29] Ekinci M. and Thomas B. T., "Navigating an Autonomous Road Vehicle in Complex Road Networks". 2nd Asian Conference on Computer vision, **Vol. 1**, pp. 229-233, Singapore, December, 1995.

[30] Ekinci M. and Thomas B. T., "Road Junction Recognition and Turn-offs for Autonomous Road Vehicle Navigation". 13th Int. Conf. on Pattern Recognition, **Vol.3**, pp. 318-322, August, 1996, Vienna, Austria.

[31] Canny J., "A Computational Approach to Edge Detection". IEEE Trans. on Pattern Analysis and Machine Intelligence, **Vol.8**, No. 6, November, 1986.

[32] Campbell N. W., Pout M. R., Priestly M. D. J., Dagless E. L., Thomas B. T., "Autonomous Road Vehicle Navigation". Engineering Application of Artificial Intelligence, **Vol. 7**, No. 2, pp.177-190, 1994.

[33] Behringer R., Holt V. v., Dickmanns D., "Road and Relative Ego-State Recognition". Proc. Intelligent Vehicles'92 Symposium, pp. 385-390, Detroit/MI, 1992.

[34] Ekinci M. and Thomas B. T., Multi-Camera Vision-Based Navigation of Autonomous Road Vehicles in Complex Road Networks". ISCIS XI The 11th International Symposium on Computer and Information Sciences, November 6-8, 1996, Antalya, Turkey.

[35] Dagless E. L., *et al.* "Image Processing Hardware and Algorithms". Microcomputer '94, Proc. of the 5th School computer Vision and Graphics, Feb. 1994, Zakopane, Poland.

[36] Priestly M. D. J., "The Initialisation and Control of a Visually Guided Autonomous Road-Following Vehicle". PhD Thesis 1994, University of Bristol.