To be presented at the IEEE Intelligent Vehicles '93 Conference, July 14-16, 1993, Tokyo, Japan.

Real-Time Visual Processing for Autonomous Driving

Marilyn Nashman and Henry Schneiderman

National Institute of Standards and Technology; Robot Systems Division; Bldg. 220, Rm. B127; Gaithersburg, MD 20899
e-mail: nashman@cme.nist.gov; hws@cme.nist.gov

ABSTRACT

In this paper, we describe a visual processing algorithm that supports autonomous road following. The algorithm requires that lane markings be present and attempts to track the lane markings on both lane boundaries. There are three stages of visual processing computation: extracting edges, matching extracted edge points with a geometric model of the road, and updating the geometric road model. A fourth stage computes a steering command for the vehicle based on the updated road model. All processing is confined to the 2D image plane. No information about the motion of the vehicle is used. This algorithm has been used as part of a complete system to drive an autonomous vehicle, the High Mobility Multipurpose Wheeled Vehicle (HMMWV). The system has been used to successfully drive the vehicle on roads within the grounds of the National Institute of Standards and Technology (NIST) at speeds up to 90 km/h as well as in simulation on a wide variety of video taped road scenes. It performs robustly for video tapes of both highways and rural roads. The algorithm runs at a sampling rate of 15 Hz and has a worst case latency of 139 milliseconds (ms).

1. Introduction

There has been increasing interest in the development of autonomous vehicles in recent years. Interest has included high-speed driving on highways, urban driving, and navigation through less structured off-road environments. The primary challenge in autonomous driving is the development of perception techniques that can cope with the variability of outdoor conditions and road appearances in any of these environments. Roads can be smooth and well marked, riddled with cracks and potholes, or not marked at all. Shadows, glare, varying illumination, dirt or foreign matter, other vehicles, rain, snow, etc. also affect road appearance.

Perception for autonomous driving has been approached with a wide variety of vision based techniques. Among the methods used are statistical classification methods, feature tracking methods, image flow methods and neural network based methods. Statistical classification methods [1], [2], [3], [4], [5], [6], [7], [8] have been applied to the road perception problem. These methods share a similar paradigm. This approach involves classifying each pixel in the scene as either road or non-road using classical techniques of supervised or unsupervised statistical classification [22]. Road shape is usually then determined by finding the closed region that contains the highest concentration of "road" pixels.

Feature tracking is another perception method used in autonomous driving systems. These methods track prominent features (e.g. lane markers) from image to image. Systems which use feature tracking include [9], [10], [11], [12], [13], [14], [15]. Image flow techniques are described in [16] and neural networks in [17].

We use a feature tracking method. Our processing consists of three stages of visual computation:

1) Extracting edge point position and orientation.
2) Matching extracted edge points to the road model.
3) Updating the road model.

A steering command is computed based on the updated road model in a fourth stage of processing. This command is used by the vehicle in an autonomous driving situation. The sequence of operations is repeated for each new image. The input to Stage (1) (Figure 1) consists of scenes of a driver's view of the road ahead as the vehicle is driven. Stages (2) and (3) require an explicit geometric model of the road. Stage (2) attempts to match the extracted edge points obtained from stage (1) with the road model. Stage (3) updates the geometric model of the road using the matched edge points.

We choose not to reconstruct the 3D scene at this level of processing. Although 3D information is necessary to command navigation of the vehicle, we believe there are advantages to avoiding the transformation from 2D to 3D in the feature tracking feedback loop as shown in Figure 2. In this way, our approach differs from that taken by other feature trackers (e.g. [9], [10], [11], [12], [13]). Their approaches convert the matched features from 2D to 3D before updating the road model. The 3D road model is then backprojected into 2D for the matching process. By updating the model in 2D, our feature tracking algorithm is unaffected by any errors, approximations, or assumptions that might be incurred in doing 3D reconstruction and backprojection. In 3D reconstruction, assumptions are often made about flat roads and small angles, etc. 3D reconstruction also requires camera calibration which can introduce errors.

Our method for updating the model is different from other road following approaches in the manner in which we combine data temporally. As in other approaches, we use a form of recursive estimation to update the parameters of our geometric road model. However, by restricting our representation to the 2D image plane, we are able to combine both spatial and temporal information in one estimation formulation. Under this formulation, the weight assigned to the data from each image implicitly depends on the number of data points matched between the image and the model. If the lane marking momentarily disappears, few edge points will match the model and the weight of this data will be relative-

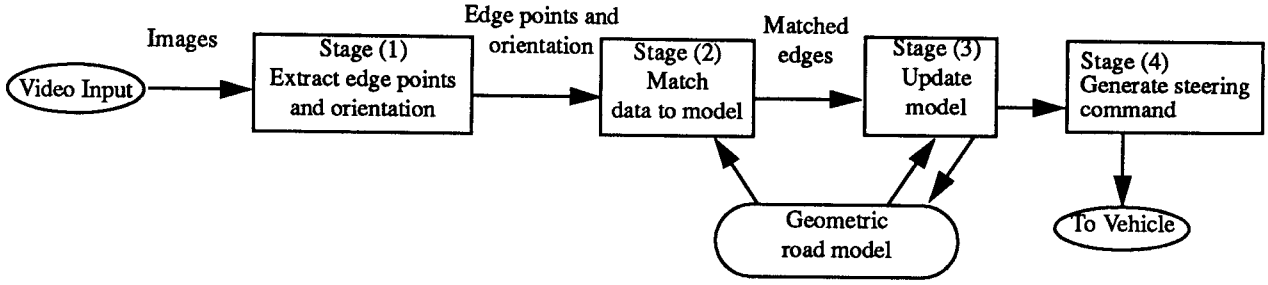

Figure 1. Processing Overview

ly insignificant compared to data from an image in which lane markings are visible.

Other feature trackers do not seem to differentiate in image confidence when combining images temporally. They do not seem to distinguish between images where lane markers are strong and images where lane markers are weak. This will adversely affect performance in situations where lane marker visibility is momentarily weak. For example, the method used in [9], [10], [11] first computes least squares estimates of a set of geometric parameters using only the data from the current image. These parameters are then smoothed over time using a Kalman filter. In using the Kalman filter, the weighting of new data is controlled by the relative choices of the model covariance and the measurement variance. [9], [10], [11] do not explain how these covariances in their Kalman filter are modeled or chosen, or even if they are chosen to vary as a function of the image. If they are chosen as constants, all images would receive equal weighting.

Section 2 discusses our road following algorithm. Section 3 describes our hardware and development environment. Section 4 describes system performance.

2. Road Feature Tracking Algorithm

In this section we describe our road feature tracking algorithm in detail. In 2.1 we describe our geometric representation of the road. In 2.2, we describe how the model is initialized to a road scene. In 2.3 we describe the edge extraction algorithm. In 2.4 we describe the algorithm that matches edge points with the road model. In 2.5 we describe the algorithm that updates the road model.

2.1. Road Model

We model the road using the left and right lane boundaries in the lane of travel. Physically, these boundaries correspond to the white or yellow lane markers painted on the road. Lane markers may consist of either solid or striped lines. We represent each of these lane boundaries by a quadratic model (equation 1) in the image plane:

$$x = a_1 + a_2 y + a_3 y^2 \qquad (1)$$

The parameters, a_1, a_2, a_3, govern the shape, position and orientation of the lane marker as it is viewed in the image.

2.2. Initial Conditions

Our algorithm requires an initially accurate model of the road. The initial model is established by a teleoperator who manually positions models of both lane markers to align them with the lane boundaries in the image. On the visual display, the models of the lane markers are represented to the teleoperator as graphic overlays on the video image. In this way, the teleoperator establishes the initial values of the parameters a_1, a_2, a_3 for both quadratics.

2.3. Edge Extraction

The first processing step performs edge extraction on the input scene (stage (1) in Figure 1). In order to enhance the contrast of the edges formed by the lane markers on the road, we place a yellow filter in front of the camera lens. The filter is designed for spectral transmission of wavelengths from 510 nanometers (nm) into the infrared. The effect is to intensify the contrast of the yellow and white markers against the road.

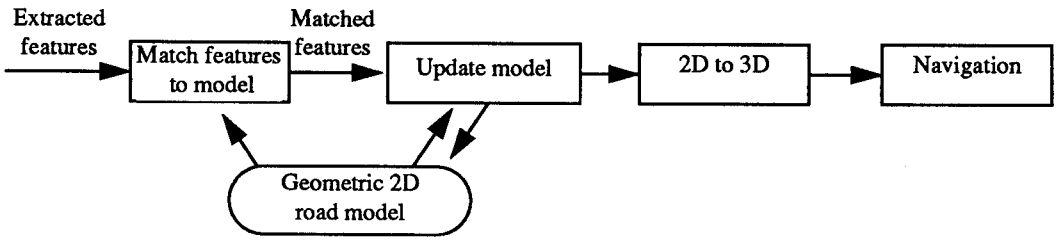

Figure 2. Feature Tracking in 2D

For every point in the image, edge magnitude and edge orientation are computed using a two-dimensional *3 X 3* spatial gradient operator. The direction, θ, of each point in the image is defined to be perpendicular to the direction of the gradient of the intensity function *f(x,y)* at that point:

$$\theta = \text{atan}\frac{(\nabla yf)(x,y)}{(\nabla xf)(x,y)} + \frac{\pi}{2} \qquad (2)$$

The magnitude of each edge pixel, *mag*, is given by:

$$mag = \sqrt{(\nabla xf)^2(x,y) + (\nabla yf)^2(x,y)} \qquad (3)$$

Using a non-maximum suppression algorithm, those edge pixels whose magnitude is greatest in the direction across the edge are selected as edge points. A description of the non-maximum suppression edge extraction algorithm can be found in [21]. A binary edge image is produced by thresholding the edge points. The threshold level is set to a value which removes weak edges and edges caused by camera noise. The output from this processing stage consists of a list of the image coordinates of all edge points above the threshold value and the orientation of these points. It should be noted, at this stage of operation, no effort is made to distinguish road edges from other edges present in the input image. Execution of this algorithm is completely data-driven.

To reduce the amount of data processed by algorithms in stages (2) and (3) in Figure 1, we exclude all edges that fall outside a window of interest. This window of interest is chosen to include the entire portion of the visible road but to exclude, as much as possible, the rest of the image (e.g. the hood of the vehicle, trees, grassy shoulders, houses, etc.). Figure 3a is a typical image of a road viewed from a camera mounted on a vehicle. Figure 3b is a window of interest. Figure 3c represents the results of masking the original road scene with the window of interest. During execution, the lateral position of the window of interest shifts in order to keep it centered on the road. In addition to centering, the shape of the window of interest changes as a function of the current road curvature. We are currently using seven masks: one mask representing zero road curvature (figure 3b), three masks representing increasing road curvature to the left, and three masks representing increasing road curvature to the right. All masks are generated off-line but are instantiated in real-time for the actual image processing. Our mask selection algorithm changes masks when one of the lane marker models intersects either of the vertical boundaries of the current mask. For example, if a lane marker intersects the left boundary, the mask giving the next larger increment of curvature to the left is chosen.

2.4. Edge Matching

In this stage of processing we match the edge data against the existing model of the road. The purpose of this edge matching algorithm is twofold. The first purpose is to associate edge points with the appropriate lane marker. The second purpose is to eliminate edge points that do not seem to be associated with either lane marker. For example, shadows, pot holes, or other vehicles can appear in the selected window and will contribute to the edge information. We wish to exclude this "spurious" edge data from the road model update computation.

The edge matching algorithm compares each edge pixel to the model of each lane marker. An edge pixel is either accepted or rejected depending upon its similarity to the model. The labelling process is based on two criteria. The first criterion is the two-dimensional spatial proximity of an edge point to the model. The second criterion is the similarity of direction of the edge point with the angular orientation of the model.

To facilitate this process, the quadratics representing each lane marker are approximated by a set of consecutive line segments. This is achieved using a simplified version of the iterative endpoints algorithm [22]. The conglomerate of these lines is used as the model in the matching procedure. The first step in this procedure compares the edge direction of the candidate edge point with the angular direction of each of these model lines:

$$|\theta_{model} - \theta_{data}| < \delta \qquad (4)$$

If this angular disparity is within an acceptable range,

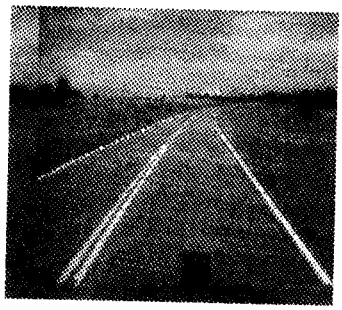

Figure 3a. Road Scene

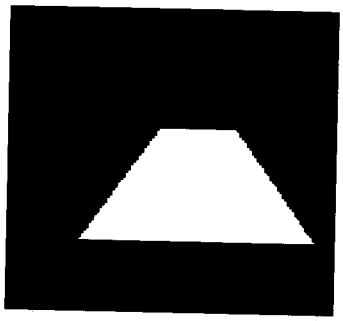

Figure 3b. Window of Interest

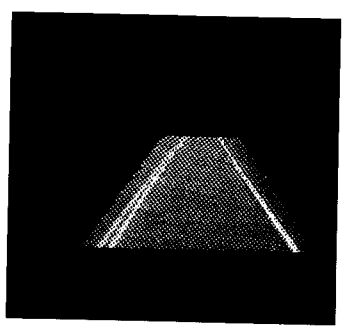

Figure 3c. Window of Interest applied to road scene

δ, for any model line, the distance d is computed between the point at image coordinate (x_i, y_i) and each model line:

$$d = \frac{A_k x_i + B_k y_i + C_k}{(A_k^2 + B_k^2)^{1/2}} \quad (5)$$

Where $(A_k x + B_k y + C_k = 0)$ is the equation for the k^{th} line in the model.

The minimum of these distances is used to determine if that point is less than a distance threshold, ζ, from the model. The point is labelled as belonging to the model when both the spatial proximity and orientation conditions are satisfied. Under these matching criteria, an edge point will rarely be associated with both lane markers except possibly at the vanishing point of the road.

2.5. Road Model Update

Each of the two quadratic lane marker models is updated separately. The parameters of each model a_1, a_2, a_3 in equation (1), are updated by an exponentially weighted recursive least squares computation (see [23]) using the matched edge points as input data.

In this estimation method, the estimated model parameters are based not just on the current image, but on data acquired over the entire sequence of previous images. The matched edge data from any one image alone may not be sufficient to obtain an accurate model of the lane markers. This data may be contaminated by noise or the edges due to actual lane markers may be too weak to be detected. In either case, the estimate of a lane marker model can be improved by using data over a sequence of images.

The success of exponentially weighted recursive least squares is based on the assumption that the appearance of the road changes gradually over a sequence of images. There are, however, limits at which the assumption of continuity fails. We must therefore choose a method of weighing new data with respect to old in order to achieve a compromise between responsiveness and robustness. For example, if new data is weighted relatively heavily, the algorithm will be very responsive to changes in the road. However, the algorithm will also be more susceptible to the ill-effects of noise and sparse data. On the other hand, if new data is weighted less heavily, the algorithm will be more robust in the presence of noise, but more inert in responding to actual changes in the image of the road.

In exponentially weighted recursive least squares, the trade-off between new and old data is controlled by specifying the value of the *exponential weighting factor*, λ (also called the *forgetting factor*). The weight assigned to each image is:

$$\lambda^{n-m} \quad (6)$$

$0.0 < \lambda < 1.0$
n is the current time
m is the time the image was sampled

For example, if $\lambda = 0.5$, all edge points in the current image, time m=n, have a weight of *1.0*. All edge points in the image read at time $m = n - 1$ have a weight of *0.5*; edge points from time $m = n - 2$ have a weight of *0.25*, etc. Values of λ anywhere in the range $0.5 < \lambda < 0.75$ produced acceptable tracking for our road scenes.

The least squares problem is formulated as follows. To determine values of a_1, a_2, a_3 in equation (1) which will provide the best least squares fit to a batch of N data points (x_i, y_i), the least squares residual in x is minimized:

$$J = \sum_{i=1}^{N} [x_i - (a_1 + a_2 y_i + a_3 y_i^2)]^2 \quad (7)$$

In recursive least squares, the data includes the current image as well as all previous images. The data from previous images is weighted by increasing powers of the exponential weighting factor. Therefore we solve for a_1, a_2, a_3, by minimizing the exponentially weighted recursive least squares residual:

$$J = \sum_{i=1}^{N_j} [x_{j,i} - (a_1 + a_2 y_{j,i} + a_3 y_{j,i}^2)]^2 + \quad (8)$$

$$\lambda \sum_{i=1}^{N_{j-1}} [x_{j-1,i} - (a_1 + a_2 y_{j-1,i} + a_3 y_{j-1,i}^2)]^2 +$$

$$\lambda^2 \sum_{i=1}^{N_{j-2}} [x_{j-2,i} - (a_1 + a_2 y_{j-2,i} + a_3 y_{j-2,i}^2)]^2 + \ldots$$

j - Time at which image was sampled
N_j - Number of matched edges points in image j

Each summation represents the data from one image. In this residual, the weight of each image j is also implicitly a function of the number of edge points matched, N_j. An image that matches many edge points will carry more weight than an image which matches few edge points. Therefore, if a lane marker momentarily disappears, few edge points will match the model and the estimate will not be greatly perturbed. Also, since the variance of a least squares estimate decreases as the number of data points increases (see [19], [20]) we are in effect giving more weight to data in which there is a higher confidence.

To efficiently solve equation (8) for a_1, a_2, a_3 such that the residual is minimized, we use the square root information filter (SRIF) algorithm[24]. The SRIF provides an efficient, numerically stable, closed form solution to the least squares problem. It is also a recursive algorithm. That is the model is updated as new data becomes available without having to explicitly store old data. The algorithm also has the advantage in that it is "recursive in batches." Whereas other recursive estimation algorithms combine new information one measurement at a time, this algorithm can efficiently combine multiplies of measurements at once. Since

each image yields a batch of edge points, this algorithm is well suited for our problem. The SRIF algorithm is described in [24][25].

3. Hardware and Developmental Testbed

Our development environment consists of a Sun SPARC2 workstation, a Pipelined Image Processing Engine (PIPE), a VME-based multiprocessor system[1], and a VHS video cassette recorder (VCR).

Our image data was collected from a camera mounted on the hood of a HMMWV [35] aimed to capture the driver's view of the road ahead. For simulation purposes, recordings were made of both highway scenes and rural roads as the vehicle travels at speeds varying between 40 kilometers per hour (km/h) and 88 km/h The HMMWV was occasionally driven in an erratic fashion (weaving back and forth, etc.) to create challenging image sequences. For autonomous driving, live camera input from the camera mounted on the HMMWV hood is used.

Image data is read into PIPE either from the live camera or the VCR playback mode. The incoming images are digitized to provide 8-bit grayscale images that are 242x256 pixels in size. Edge extraction is performed on the images in PIPE which then converts the information from an image format to a symbolic list. The corresponding edge direction values are mapped onto the memory of one of the microprocessors via a specialized PIPE-VME interface board.

The remaining processing is divided among microprocessors in the VME backplane. Most computations -- communication with the PIPE, edge matching, updating the model, and computing a graphical overlay -- are pipelined. The model updates for each lane marker are computed in parallel on separate processors. All inter-processor communication is done through semaphored global memory. For a detailed description of our software engineering practices refer to [36].

The display process provides graphic overlays of the window of interest, the geometric model of the lane boundaries and the computed lane center on the live video image. These graphic overlays are used for debugging purposes and to provide a qualitative measure of performance. A Matrox VIP 1024 board is used to implement the graphic overlays on the video signal.

All program development for the VME-based multiprocessor system is done on a Sun SPARC2 workstation. All code on this system is written in the Ada programming language. Program development for PIPE is done on a personal computer using the PIPE graphical programming language, ASPIPE [37].

1. Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the materials or equipment identified are necessarily best for the purpose.

4. System Evaluation

We tested the system and algorithms described using both live camera input and input from videotaped sequences of road scenes We have integrated and tested the algorithm with the steering control algorithm described in [35] on the HMMWV and have successfully demonstrated this closed loop autonomous driving system on the NIST grounds. We have achieved autonomous driving at speeds of 90 km/h under weather conditions varying from sunny conditions to heavy rain.

In simulation, on a limited access multilane highway, the algorithm successfully maintained tracking over a 3 mile section of road. The vehicle was travelling in the right lane at approximately 88 km/h The lane markings consisted of a dashed line on the left and a solid line on the right. Tracking was maintained while other cars passed in the adjacent lane. Tracking was lost when the vehicle changed lanes to exit.

Again in simulation, the algorithm successfully tracked over a distance of approximately 2 miles on a four lane local road. The vehicle was driving in the left lane at a speed of approximately 65 km/h The lane markings consisted of a solid line on the left and widely spaced stripes on the right. The algorithm was robust in maintaining tracking through two intersections in which the lane markings disappeared. Tracking was also maintained while driving beneath an underpass and over two bridges. The pavement texture and color changed from a dark asphalt to a light cement on the bridges. On other portions of this road, the algorithm could not always maintain tracking through intersections in which the lane of travel split into two lanes (a turning lane and a lane for going straight). Also tracking could not be maintained when the vehicle changed lanes.

On a two lane rural road, tracking was maintained over a distance of 1.5 miles. The vehicle travelled at speeds between 40 and 65 km/h The lane markings consisted of a double solid line on the left and a single solid line on the right. Tracking was robustly maintained on travel up and down hills with on-coming traffic, through sharp curves, through moderate shadows, and through four intersections in which lane markings disappeared. Tracking was temporarily lost when the vehicle travelled through a sharply curved portion of road that was shadowed by a heavily wooded area.

The image sampling rate of our system is 15 Hz and the worst case latency is 139 milliseconds (ms). Edge extraction was performed every 66.7 ms. The number of edge points extracted varies from scene to scene and the processing times for the algorithms in stages (2) and (3) varies depending on the number of data points present. For a representative road scene containing approximately 300 edge points, the edge matching is performed in 21 ms and the road model update is performed in 51 ms. The graphic overlay process, which is not part of the feedback loop, is updated in approximately 5 ms.

5. Conclusion and Future Work

We have described a system of algorithms that robustly follows roads that one might expect to find on state highways. We assume that the lane boundaries are well marked with either solid, double, or dashed lines. All visual processing is done in two dimensional image coordinates. Processing is performed in sequential stages: extracting edges; matching edge points to the road model; updating the model of the road; and computing a steering command. Computation time for the image processing algorithms is reduced by using knowledge of the road curvature to mask out non-road information. The exponentially weighted recursive least squares algorithm used to update the road model operates in both a spatial and temporal domain. The system update rate is 15 Hz.

We have integrated our algorithm with the navigation system of the HMMWV [35] and performed unmanned driving on the NIST grounds. In the near future, we plan to test our autonomous system on state highways.

References

[1] C. 1, M. Hebert, T. Kanade, S. Shafer. "Vision and Navigation for the Carnegie-Mellon Navlab." IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 10, No. 3. May, 1988.

[2] C. Thorpe, M. Hebert, T. Kanade, S. Shafer. "Toward Autonomous Driving: The CMU Navlab." IEEE Expert. August, 1991.

[3] J. Crisman, C. Thorpe. "Color Vision for Road Following." In *Vision and Navigation: The Carnegie Mellon Navlab.* Kluwer. Norwell, MA. 1990.

[4] M. Turk, D. Morgenthaler, K. Gremban, and M. Marra. "VITS - A Vision System for Autonomous Land Vehicle Navigation." IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 10, No. 3. May, 1988.

[5] D. Kuan, U. K. Sharma. "Model Based Geometric Reasoning for Autonomous Road Following." IEEE conference on Robotics and Automation. 1987. pp 416 - 423.

[6] D. Kuan, G. Phipps, and A. Hsueh. "Autonomous Robotic Vehicle Road Following." IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 10, No. 5. September, 1988.

[7] Xueyin Lin and Shaoyun Chen. "Color Image Segmentation Using Modified HSI System for Road Following." IEEE Conference on Robotics and Automation. 1991. pp 1998 - 2003.

[8] Jill D. Crisman and Charles E. Thorpe. "UNSCARF, A Color Vision System for the Detection of Unstructured Roads." IEEE Conference on Robotics and Automation. 1991.

[9] E. Dickmanns, A. Zapp. "A Curvature-based Scheme for Improving Road Vehicle Guidance by Computer Vision." SPIE Vol. 727. Mobile Robots, 1986.

[10] B. Mysliwetz, E. Dickmanns. "Distributed Scene Analysis for Autonomous Road Vehicle Guidance." SPIE Vol. 852. Mobile Robots II, 1987.

[11] E. D. Dickmanns and B. D. Mysliwetz. "Recursive 3-D Road and Relative Ego-State Recognition." IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 14, No. 2. Feb., 1992.

[12] A. Waxman, J. LeMoigne, L. Davis, B. Srinivasan, T. Kushner, E. Liang, T. Siddalingiah. "A Visual Navigation System for Autonomous Land Vehicles." IEEE Journal of Robotics and Automation. Vol RA-3, No. 2. April, 1987.

[13] K. Kluge and C. Thorpe. "Explicit Models for Road Following." In *Vision and Navigation: The Carnegie Mellon Navlab.* Kluwer. Norwell, MA. 1990.

[14] R. Lotufo, A. Morgan, E. Dagless, D. Milford, J. Morrissey, B. Thomas. "Real-time Road Edge Following for Mobile Robot Navigation." Electronics and Communications Engineering Journal. Vol. 2, No. 1. February, 1990.

[15] S. Kenue. "Lanelok: Detection of Lane Boundaries and Vehicle Tracking Using Image-Processing Techniques - Part I: Hough-Transform, Region-Tracing and Correlation Algorithms and Part II: Template Matching Algorithms." SPIE Vol. 1195. Mobile Robots IV, 1989.

[16] D. Raviv and M. Herman. "A New Approach to Vision and Control for Road Following." Proceedings of the IEEE Workshop on Visual Motion. Princeton, NJ. October, 1991.

[17] D. Pomerleau. "Neural Network Based Autonomous Navigation." In *Vision and Navigation: The Carnegie Mellon Navlab.* Kluwer. Norwell, MA. 1990.

[18] M. Hebert, I. Kweon, T. Kanade. "3-D Vision Techniques for Autonomous Vehicles." In *Vision and Navigation: The Carnegie Mellon Navlab.* Kluwer. Norwell, MA. 1990.

[19] C. Helstrom. *Probability and Stochastic Processes for Engineers.* Macmillan Publishing Company. New York, 1984.

[20] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* 2nd Ed. Magraw-Hill. New York, 1984.

[21] A. Rosenfeld, A. Kak, *Digital Picture Processing,* Volume 2, Second Edition. Academic Press, 1982.

[22] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons. New York, 1973.

[23] G. Goodwin, K. Sin. *Adaptive Filtering, Prediction and Control.* Prentice-Hall. Englewood Cliffs, New Jersey, 1984.

[24] G. Bierman. *Factorization Methods for Discrete Sequential Estimation.* Academic Press. New York, 1977.

[25] C. Lawson and R. Hanson. *Solving Least Squares Problems.* Prentice-Hall. Englewood Cliffs, New Jersey, 1974.

[26] G. Golub and C. Van Loan. *Matrix Computations.* 2nd Ed. John Hopkins University Press. Baltimore, 1989.

[27] G. Strang. *Linear Algebra and Its Applications.* 3rd Ed. Harcourt Brace Jovanovich. Orlando, Florida. 1988.

[28] B. Noble and J. Daniel. *Applied Linear Algebra.* 3rd Ed. Prentice-Hall. Englewood Cliffs, New Jersey. 1988.

[29] J. Albus, H. G. McCain, R. Lumia. "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)." NIST Technical Note 1235. Gaithersburg, MD, July, 1987.

[30] K. Chaconas, M. Nashman. "Visual Perception Processing in a Hierarchical Control System." NIST Technical Note 1260. Gaithersburg, MD, March, 1989.

[31] J. Fiala. "Manipulator Servo Level Task Decomposition." NIST Technical Note 1255. NIST, Gaithersburg, MD, October, 1988.

[32] L. Kelmar. "Manipulator Servo Level World Modeling." NIST Technical Note 1258. NIST, Gaithersburg, MD, March, 1989.

[33] A. Wavering. "Manipulator Primitive Level Task Decomposition." NIST Technical Note 1256. NIST, Gaithersburg, MD, October, 1988.

[34] J. Albus, M. Juberts, S. Szabo. "A Reference Model Architecture for Intelligent Vehicle and Highway Systems." Isata 25th Silver Jubilee International Symposium on Automotive Technology and Automation. Florence, Italy. June, 1992.

[35] S. Szabo, H. Scott, K. Murphy, S. Legowik, R. Bostelman. "High-Level Mobility Controller for a Remotely Operated Land Vehicle." Journal of Intelligent and Robotic Systems. Vol. 5, pp 63-77. 1992.