Visual Processing for Autonomous Driving

Henry Schneiderman and Marilyn Nashman
Robot Systems Division
Building 220, Room B127
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

In this paper, we describe a visual processing algorithm we have developed 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 computation: extracting edges, matching extracted edge points with a geometric model of the road, and updating the geometric road model. All processing is confined to the 2-D image plane. No information about the motion of the vehicle is used. This algorithm has been implemented and tested using video taped road scenes. It performs robustly for both highways and rural roads. The algorithm runs at a sampling rate of 15 Hz and has a worst case latency of 132.8 milliseconds (ms). The algorithm is implemented under the NASA/NBS Standard Reference Model for Telerobotic Control System Architecture (NASREM) architecture and runs on a dedicated vision processing engine and a VME-based[1] microprocessor system.

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.

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 are reliable under the extreme variability of outdoor conditions in any of these environments. Roads can vary tremendously in appearance. Some are smooth and well marked while others are riddled with cracks and potholes or are not marked at all. Shadows, glare, varying illumination, other vehicles, rain, snow, etc. also affect road appearance.

Perception for autonomous driving has been approached with a wide variety of vision-based techniques. Classical pattern classification techniques have received much attention. These methods usually use some combination of color and spatial cues to label all pixels in the image as either road or non-road. Various techniques are then used to determine a boundary that best separates the road from the rest of the scene. Examples of these methods include [1], [2], [3], [4], [5]. Another popular method is to use feature tracking. These methods track prominent features (e.g. lane markers) from image to image. Systems which use feature tracking include [6], [7], [8], [9], [10], [11]. Other approaches include image flow based methods, [12], and artificial neural networks in [13]. In addition to vision-based perception techniques, in [1], [14], a 3D laser range finder is used for outdoor navigation.

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

1) Extracting edge point position and orientation.

2) Matching extracted edge points to the road model

3) Updating the road model

A video taped recording provides the input to Stage (1) (Figure 1). This video consists of road scenes of a driver's view of the road ahead as a vehicle is driven along both highways and rural roads. Stages (2) and (3) require an explicit geometric model of the road. Stage (2) attempts to

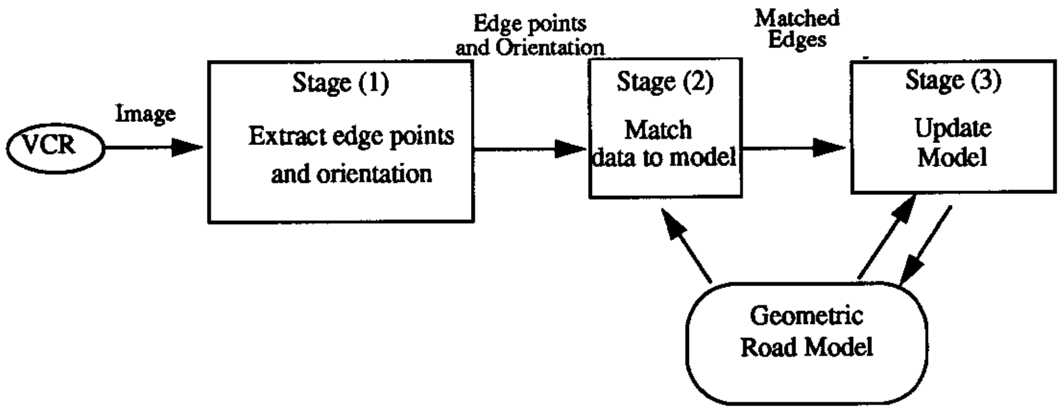

Figure 1. Processing Overview

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 a 3D model of the road is necessary to command navigation of the vehicle, we believe this transformation from 2D to 3D should be independent of the feature tracking feedback loop as shown in Figure 2. In this

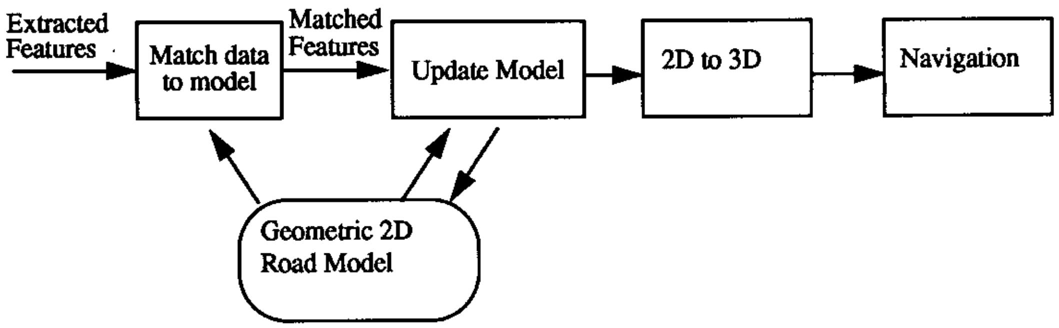

Figure 2. Feature Tracking in 2D

way, our approach differs from that taken by other feature trackers (e.g. [6], [7], [8]). 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 as shown in Figure 3. By updating

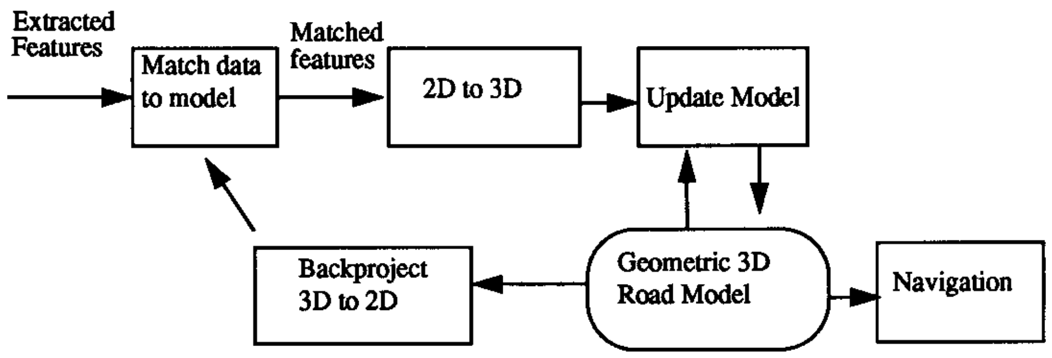

Figure 3. Feature Tracking in 3D

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.

Our method for updating the model is also unique. We use an exponentially weighted recursive least square fit (described in section 2.5) to update the parameters of our geometric road model. The appeal of this method is in its weighting of data. The weight assigned to the data from each image 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 relatively insignificant compared to data from an image in which lane markings are visible. In addition, since the variance of a least squares estimate decreases as the number of data points increases (see [15], [16]) we are in effect giving more weight to data in which there is a higher confidence.

Other approaches do not seem to account for the confidence of data when combining data temporally. For example, the method used in [6] and [7] 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. The weight assigned to these parameters when they are smoothed seems to be independent of the quality of their least squares fit. In the Kalman filter formulation, the weighting of new data is controlled by the relative choices of the model covariance and the measurement covariance. [6] and [7] do not explain how these covariances are modeled or chosen, or even if they are chosen to vary as a function of time. If they are chosen as constants, an image yielding a weak edge would be given the same weight as one yielding a strong edge. This can result in a spurious data point carrying significant weight when lane markers disappear (either between stripes or at an intersection).

In Section 2, we describe our road following algorithm. Section 3 contains a description of our hardware and development environment. Section 4 describes the performance of our system. Section 5 summarizes our experiments and presents future development plans.

2. Road Feature Tracking Algorithm

In this section we describe our road feature tracking (following) 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. They may also be solid or striped lines. We represent each of these lane boundaries by a quadratic model (equation 1) in the image plane:

$$x = a_1 + a_2 y + a_3 y^2 \tag{1}$$

The parameters, a_1, a_2, a_3, govern the shape of this model.

2.2. Initial Conditions

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

2.3. Edge Extraction

The first processing step performs edge extraction on the input scene (stage (1) in Figure 1). 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:

$$\theta = \operatorname{atan}\frac{(\nabla_y f)(x,y)}{(\nabla_x f)(x,y)} + \frac{\pi}{2} \qquad (2)$$

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

$$mag = \sqrt{(\nabla_x f)^2(x,y) + (\nabla_y f)^2(x,y)} \qquad (3)$$

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 [17]. 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. We have empirically found a threshold value of 8 in a grey level range of 0 to 255 to be effective for our application. 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 4a is a typical image of a road

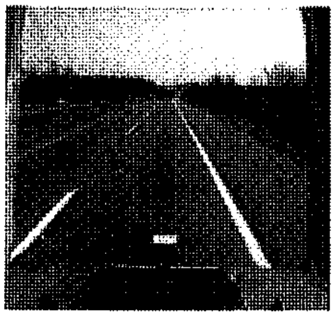
Figure 4a. Road Scene

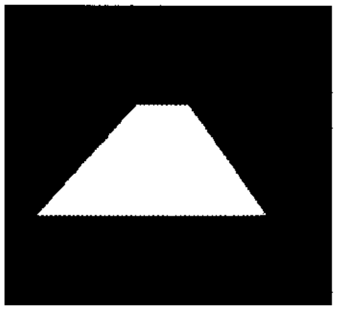
Figure 4b. Window of Interest

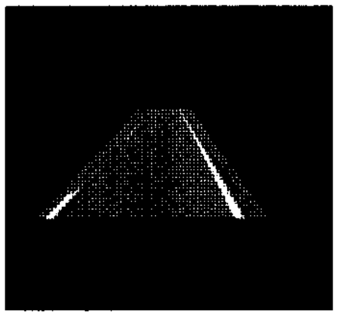
Figure 4c. Window of interest applied to road scene

viewed from a camera mounted on a vehicle. Figure 4b is a window of interest. Figure 4c 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 center of the road. In addition to centering, the shape of the window of interest changes as a function of the predicted road curvature. We are currently using seven masks: one mask representing zero road curvature (figure 4b), 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. 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:

$$|\theta_{model} - \theta_{data}| < \delta \tag{4}$$

If this angular disparity is within an acceptable range, δ, for any model line, the distance *d* is computed between the point at image coordinate *(x_i, y_i)* and each model line:

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

where $(A_k x + B_k y + C_k = 0)$ is the general form for the kth 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. If an edge point is chosen as belonging to one quadratic model, it is not considered as a match candidate for the second quadratic model. In this way, we insure that any given point belongs to at most one model.

2.5. Road Model Update

The parameters of the road model, equation (1), are updated with an exponentially weighted recursive least squares computation using the matched edge points as input data. It should be emphasized that in this estimation method, the fit is based on data acquired over a sequence of images rather than just the current image. Earlier experiments involved fitting the road model separately to each image. The frequent inclusion of spurious data and sparse real data often resulted in poor fits. By combining data over a sequence of images, the spurious data tends to average and the real data reinforces itself resulting in a better fit. Probability theory calls this phenomena the *law of large numbers* [15], [16].

The success of this method 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 *forgetting factor*, λ. The weight assigned to each data point is then given by

λ^{n-n_o} where

$0.0 < \lambda < 1.0$

n is the current time

n_o is the time the data was sampled.

For example, if $\lambda = 0.5$, all edge points in the current image, time n_0, have a weight of 1.0. All edge points in the image read at time n_{-1} have a weight of 0.5; edge points from time n_{-2} have a weight of 0.25, etc.). Empirically we found the values of λ in the range $0.5 < \lambda < 0.75$ produced robust tracking for our road scenes.

The least squares problem of fitting a quadratic to a set of data is formulated as follows. For a set of data points (x_i, y_i) for $i = 1$ *to* N where N is the total number of data points, we wish to determine a_1, a_2, a_3 such that J, the residual in x, is minimized:

$$J = \sum_{i=1}^{N} [x_i - (a_1 + a_2 y_i + a_3 y_i^2)]^2 \tag{6}$$

Since our data set includes edges from all previous images, where images are weighted exponentially, the residual we minimize is given by:

$$J = \sum_{i=1}^{N_j} [x_i - (a_1 + a_2 y_i + a_3 y_i^2)]^2 + \lambda \sum_{i=1}^{N_{j-1}} [x_i - (a_1 + a_2 y_i + a_3 y_i^2)]^2 + \lambda^2 \sum_{i=1}^{N_{j-2}} [x_i - (a_1 + a_2 y_i + a_3 y_i^2)]^2 + \dots \tag{7}$$

Each summation represents the data from one image. In this residual, the weight of image j is also a function of the number of edge points matched, N_j. An image that yields a good match with the road will exert a larger weight than one with a poorer match. Since the variance of a least squares estimate decreases as the number of data points increases (see [15], [16]) we are in effect giving more weight to data in which there is a higher confidence.

To efficiently solve equation (7) for a_1, a_2, a_3 such that the residual is minimized, we use the square root information filter (SRIF) algorithm. The SRIF is an efficient, numerically stable, closed form solution to the least squares problem. It is also an iterative algorithm. Data from previous images is not explicitly stored but is summarized in the previous estimate and its covariance. After each image, this algorithm updates the estimate and a square root of its information matrix. (The information matrix is the inverse of the covariance matrix). The SRIF is described in [18], [19].

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, and a VHS video cassette recorder (VCR). Figure 5 shows the breakdown of processing across hardware. In this figure, the large gray rectangles represent distinct software modules. Each of these modules is labelled by its functionality (SP = sensory processing, WM = world modeling) and level within the NASA/NBS Standard

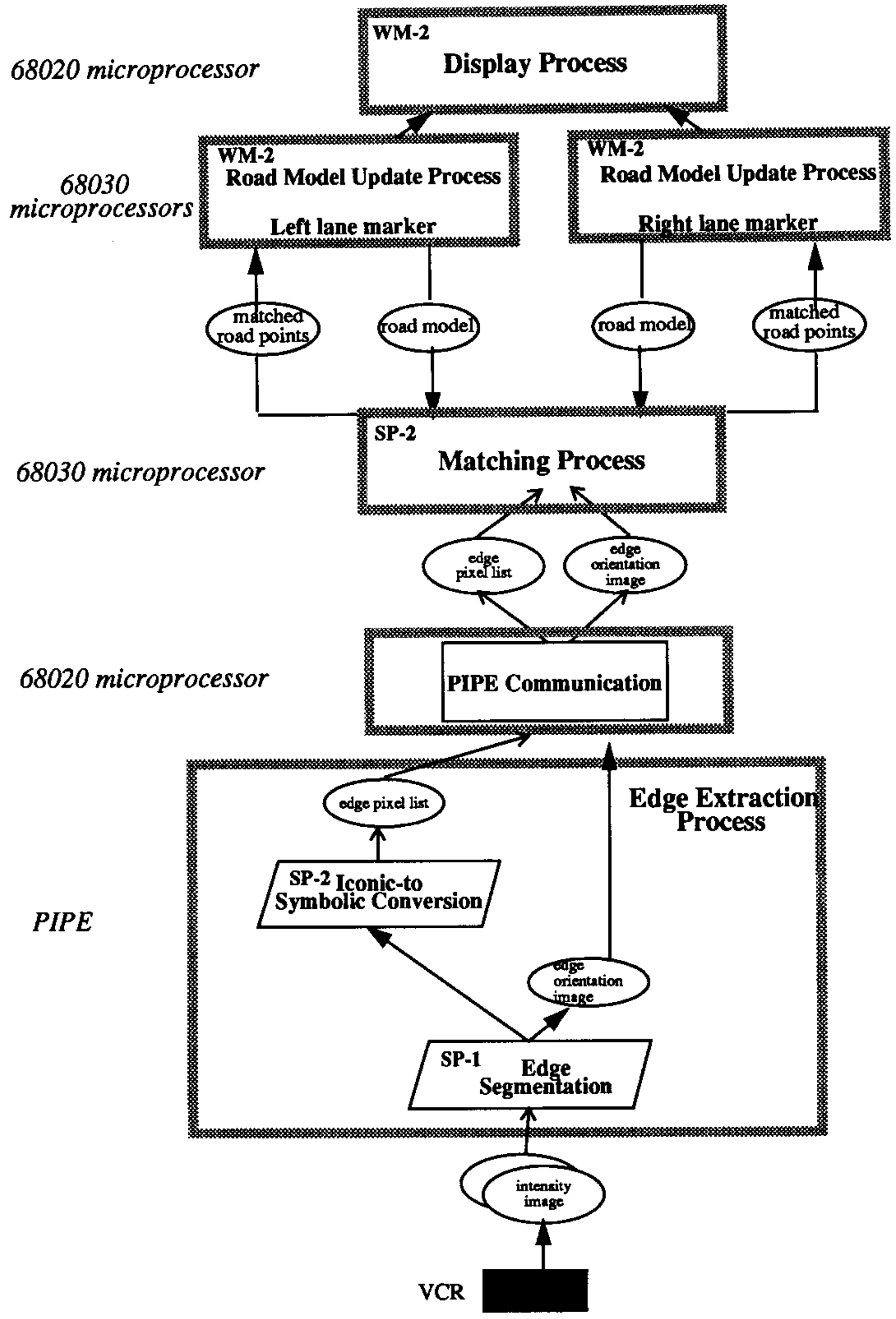

Figure 5. Distribution of processing in hardware

Reference Model for Telerobot Control System Architecture (NASREM) [20]. The system described in this paper is contained in a larger, multi-purpose implementation of NASREM, [21], [22], [23], [24]. Also, a complete control system architecture proposed for intelligent vehicles is found in [25].

Our image data was recorded with a camcorder mounted on the hood of a U.S. Army High Mobility Multipurpose Wheeled Vehicle (HMMWV) [26] aimed to capture the driver's view of the road ahead. Recordings were made of both highway scenes and rural roads as the vehicle travels at speeds varying between 40 kilometers per hour (k.p.h.) and 88 k.p.h. The HMMWV was occasionally driven in an erratic fashion (weaving back and forth, etc.) to create challenging image sequences.

The recorded data is read into PIPE through 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. The Iconic-to-Symbolic Mapper (ISMAP) stage of PIPE then converts information from an image format to a symbolic list and is used to store the binary edge image as a list of pixel positions. In addition, the corresponding edge direction values are stored in the ISMAP iconic buffer where they are mapped onto the memory of one of the microprocessors via a specialized PIPE-VME interface board. The edge extraction and symbolic mapping operations are pipelined. They are indicated by black parallelograms in Figure 5.

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 [27].

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 (not shown in figure) 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 [28].

4. System Evaluation

As mentioned earlier, we tested the system and algorithms described using input from videotaped sequences taken with a camcorder mounted on the HMMWV vehicle as it traveled on highways, local roads, and back roads.

On a limited access multilane highway, interstate I-270 in Maryland, the algorithm successfully maintained tracking over a 3 mile section of road. The vehicle was travelling in the right lane at approximately 88 k.p.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 and while the vehicle weaved back and forth in the lane of travel. Tracking was lost when the vehicle changed lanes to exit.

On a four lane local road, Great Seneca highway in Gaithersburg, Maryland, the algorithm successfully tracked for a distance of approximately 2 miles. The vehicle was driving in the left lane at a speed of approximately 65 k.p.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, Quince Orchard road in Gaithersburg, Maryland, tracking was maintained over a distance of 1.5 miles. The vehicle travelled at speeds between 40 and 65 k.p.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 132.8 milliseconds (ms). Edge extraction was performed every 66.4 ms. The number of edge points extracted varies from scene to scene and the processing times for the algorithms in stages (2) and (3) will vary depending on the number of data points present. For a representative road scene containing approximately 300 edge points, the edge matching is performed in 20 ms and the road model update is performed in 30 ms. The graphic overlay process 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; and updating the model of the road. 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 hope to integrate our algorithm with the navigation system of the U.S. Army High Mobility Multipurpose Wheeled Vehicle (HMMWV) [26] and to perform unmanned driving using the entire system in the near future. We also plan to enhance our algorithm by incorporating geometric constraints such as road width consistency and by incorporating knowledge of vehicle motion. Future research directions include the investigation of pattern classification methods using color and spatial cues similar to those used by [1], [2], [3], [4], [5], in addition to edge extraction, for robustly determining road boundaries.

Acknowledgments

We would like to give special thanks to Thomas Wheatley for his help. We would also like to thank all members of the Robot Systems Division at the National Institute of Standards and Technology (NIST) for their helpful suggestions and comments.

References

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

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

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

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

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

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

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

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

[20] 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.

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

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

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

[24] A. Wavering. "Manipulator Primitive Level Task Decomposition." NIST Technical Note

1256. NIST, Gaithersburg, MD, October, 1988.

[25] 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.

[26] 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.

[27] J. Fiala. "Note on NASREM Implementation." NIST Internal Report 89-4215. Gaithersburg, MD, December, 1989.

[28] Aspex Inc. "PIPE--An Introduction to the PIPE System." New York, 1987.