# VIDEO ROAD-FOLLOWING FOR THE AUTONOMOUS LAND VEHICLE

Matthew A. Turk, David G. Morgenthaler, Keith D. Gremban*, Martin Marra

Martin Marietta Denver Aerospace
P.O. Box 179, M.S. H0427
Denver, CO 80201

## Abstract

We describe the vision system for Alvin, the Autonomous Land Vehicle, addressing in particular the task of road-following. The system builds symbolic descriptions of the road and obstacle boundaries using both video and range sensors. Road segmentation methods are described for video-based road-following, along with approaches to boundary extraction and the transformation of boundaries in the image plane into a vehicle-centered three dimensional scene model. The ALV has performed public road-following demonstrations, traveling distances up to 4.5 km at speeds up to 20 km/hr along a paved road, equipped with an RGB video camera with pan/tilt control and a laser range scanner.

## 1. INTRODUCTION

The task of the vision system for a mobile robot is to locate and model the relevant aspects of the world so that an intelligent navigation system can plan appropriate action. For an outdoor autonomous vehicle, typical behaviors include road-following, obstacle avoidance, cross-country navigation, landmark detection, map building and updating, and position estimation. The vision system must provide a description of the world rich enough to facilitate such behaviors.

In May of 1985, "Alvin", the Autonomous Land Vehicle at Martin Marietta Denver Aerospace, performed its first public road-following demonstration. In the few months leading up to that performance, a basic vision system was developed to locate roads in video imagery and send three dimensional road centerpoints to Alvin's navigation system. Since that first demonstration, VITS (for Vision Task Sequencer) has matured into a more general framework for a mobile robot vision system, incorporating both video and range sensors and extending its road-following capabilities to include obstacle detection and avoidance. A second public demonstration in June 1986 showed the improved road-following ability of the system, allowing the ALV to travel a distance of 4.5 km at speeds up to 10 km/hr, handle variations in road surface, and navigate a sharp, almost hairpin, curve. In October 1986 the initial obstacle avoidance capabilities were demonstrated, as Alvin steered around obstacles while remaining on the road, and vehicle speeds up to 20 km/hr were demonstrated on an obstacle-free portion of road. This paper describes Alvin's vision system and addresses the particular task of video road-following. Video obstacle detection and range-based road following and obstacle avoidance are initially discussed in [4,5,12].

The Autonomous Land Vehicle project, part of DARPA's Strategic Computing Program, is described in [2] and [10]. Vision research is proceeding concurrently by a number of groups (see for example [8,9,13,14,15]), as is work in route and path planning. As the ALV is intended to be a national testbed for autonomous vehicle research, various vision systems and algorithms will eventually be implemented. The vision system described in this article is the system currently meeting the perception requirements for testing and formal demonstrations of the ALV system.

## 2. ALV SYSTEM OVERVIEW

Alvin's vision subsystem is an integral part of a larger system, and can affect and be affected by the performance of the system as a whole. Figure 1 illustrates the basic system configuration of the ALV, including the interfaces to the major modules. In the paragraphs following, each of Alvin's major components will be briefly discussed in the context of the interaction as a complete system. See [10] for a more complete discussion of the initial ALV system configuration.

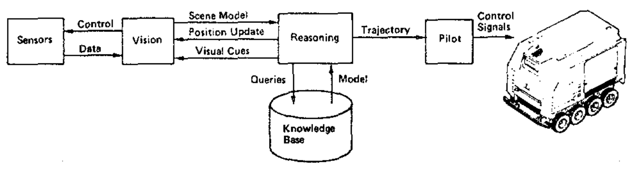


Figure 1. The ALV System Configuration.

The ALV is an all-terrain vehicle with eight-wheel drive, diesel-powered, and hydrostatically driven, with a fiberglass shell to protect the interior from dust and inclement weather and to insulate the equipment inside. The ALV hosts a number of sensors. A Land Navigation System (LNS) provides position and heading information. The primary vision sensor is a color video CCD camera, mounted on a pan/tilt unit that is under the direct control of the vision subsystem. The other vision sensor is a laser range scanner which determines range by measuring the phase shift of a reflected modulated laser beam.

Alvin currently uses a variety of different computers, and the computer architecture has been designed to facilitate the integration of additional machines as necessary. The diverse processing requirements were met by designing a modular multiprocessor architecture. In the "first generation" ALV hardware, VITS is hosted on a Vicom image processor, while the other software

*Current address: Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

The support of the Defense Advanced Research Projects Agency and the Army Engineering Topographic Laboratories under contract DACA76-84-C-0005 is gratefully acknowledged.

subsystems are hosted on an Intel multiprocessor system. VITS communicates with the other subsystems across a dedicated communication channel.

The vision subsystem is composed of three basic modules: VITS, the vision executive, which handles initialization, sets up communication channels, and "oversees" the processing; VIVD, the video data processing unit; and VIRD, the range data processing unit. The task of the vision subsystem in road-following is to process color or range images to produce a description of the road in front of the vehicle. This description is passed to the reasoning subsystem, which uses additional data such as current position, speed, and heading to generate a trajectory for Alvin to follow.

Communication between the vision subsystem and Reasoning takes place in two basic forms: the scene model and the position update. The **scene model** is the output of the vision subsystem after each image frame is processed. The scene model contains a record of Alvin's position and heading at the time of image acquisition, and a description of the road found in the image. The description of the road consists of lists of points denoting left and right edges of the road, as well as points surrounding obstacles. The points are given as 3D positions with respect to the vehicle center of gravity at the time the imagery was acquired. The reasoning subsystem must then transform the road description into a fixed, world coordinate system for navigation.

VITS must know the position and heading of the vehicle at the time of image acquisition to integrate sensor information acquired at different times, and to transform vehicle-centered data into world coordinates. In addition, VITS must be able to predict the location of the road in an image, given its location in the preceding image. These are effected by means of a **position update** message passed from Reasoning to the vision subsystem. The position update contains information on the current vehicle speed, position, and heading. Synchronization of position update and image acquisition is mediated by a position update request.

The Reasoning subsystem is the executive controller of the ALV, and Vision is a resource of Reasoning. At the highest level, Reasoning is responsible for receiving goals from a human test conductor, creating a plan script to accomplish the goals, and coordinating the other subsystems on Alvin to perform the necessary tasks. script.

Because the processing involved in creating a visual description of the environment is beyond the real-time capability of present computers, the scene model is not used directly in the vehicle's control servo loop. Instead, the Navigator (part of the reasoning subsystem) pieces together scene models from the vision system and builds a reference trajectory that is sent to the Pilot for control. The reasoning subsystem accepts a position update request from VITS, generates the appropriate data, and sends a position update to VITS. Upon receipt of a scene model, Reasoning evaluates it and plots a smooth trajectory if the data is acceptable. The new trajectory is computed to smoothly fit the previous trajectory.

The Pilot performs the actual driving of the vehicle. Given a trajectory from Reasoning, the Pilot computes the error values of lateral position, heading and speed by comparing LNS data with the target values specified in the trajectory. The Pilot uses a table of experimentally obtained control gains to determine commands needed to drive the errors toward zero; these commands are output to the vehicle controllers. The vision subsystem has no direct communication with the Pilot.

## 3. VIDEO-BASED ROAD-FOLLOWING

The task of the vision system in a road following scenario is to provide a description of the road for navigation. Roads may be described in a variety of ways, e.g. by sets of road edges, a centerline with associated road width, or planar patches. We have chosen to represent a road by its edges, or more precisely, points in three space that, when connected, form a polygonal approximation of the road edge. The difficulties in extracting the real road boundary directly from the image led us to adopt a segmentation algorithm to first extract the road in the image, track the road/non-road boundary, and then calculate three dimensional road edge points.

VITS currently uses a clustering algorithm to segment the image into road and non-road regions. After producing a binary road image, the road boundaries are traced and transformed from image points into three dimensional road boundary points. The complete cycle time, from digitization to producing a symbolic description of the road, is currently just over 2 seconds. The algorithm is summarized in the following steps, which are discussed in detail in the following sections: (1) digitize the video images; (2) segment road/non-road regions; (3) extract road boundaries by tracing the binary road edges; and (4) transform 2D road edge points to 3D coordinates and build the scene model.

### 3.1. Sensor Control and Image Acquisition

Because of a curving road, vehicle oscillation while traversing the road, or sudden path corrections to center the vehicle, the vehicle's heading may cause a fixed camera to lose all or part of the road from its field of view. Because sampling road pixels is vital to the video segmentation algorithm, losing the road is not acceptable. Two methods compensate for this: panning the camera and *power windowing*. Power windowing, a "software panning" technique, is described in Section 3.2.1.3.

Control of the pan/tilt mechanism is a function of vehicle orientation and desired viewing direction. During road-following, we would like the camera to point "down the road", regardless of the vehicle orientation, keeping the road approximately centered in the image. This requires the vision system to know global position information (provided by the LNS) and relate the vehicle-centered road description to present vehicle location and orientation, and then to calculate and command the desired pan angle. The tilt angle is fixed at approximately 17° below the horizon.

The image processing computer digitizes RGB images directly into memory from the video camera; typically, the images are then blurred to reduce noise. Calibration is performed on the camera before a test run to calculate the exact tilt angle and focal length. Along with the raw images, the *position update* is requested and received from the communication control processor, allowing later conversion from a vehicle-centered road description to world coordinates.

## 3.2. Road Segmentation

Segmentation of natural outdoor scenes is a particularly complex problem [7], but it is simplified when a predominant feature in the scene (i.e. the road) is the main focus of the segmentation. In a real-time, outdoor environment with a mobile robot, road segmentation is complicated by the great variability of vehicle and environmental conditions. Changing seasons, weather conditions, time of day, and man-made changes impact the video segmentation, along with the variable color response of the cameras, the vehicle suspension system, performance of the navigation and control subsystems, and other changes in the vehicle system. Because of these combined effects robust segmentation is very demanding.

The segmentation methods used by VITS are motivated by the hardware supporting it, speed requirements, and assumptions about road and non-road image characteristics. We have proposed and tested various segmentation techniques, all based on knowledge of road characteristics in RGB color space. These techniques and algorithms are described in the following sections.

### 3.2.1. Segmentation Techniques

#### 3.2.1.1. Color Parameter Selection

A plane in RGB space does a very good job under most conditions of partitioning the space into road and non-road regions. If each point in RGB space can be projected onto a line perpendicular to this plane, then, the three dimensional feature space is reduced to one dimension. The projection of image points in RGB space onto a line is equivalent to taking the dot product of every pixel vector with a vector in the direction of the line (normal to the separating plane). This is accomplished by a **tricolor** operation, a linear weighted combination of the red, green, and blue images:

$$I(i,j) = rR(i,j) + gG(i,j) + bB(i,j) \tag{1}$$

$$= (r,g,b) \bullet (R,G,B)$$

The outcome $I$ is a single band "feature enhanced" image. The vector $(r,g,b)$ represents the red, green, and blue components of the tricolor operation. This vector is normal to the plane that separates road and non-road clusters in the RGB space. The orientation of this plane is relatively consistent under given weather and camera conditions. It can often be chosen by hand at the beginning of a run and not modified throughout the operation of VITS. However, we have found that changing weather conditions, seasonal changes, and the camera color response affect the optimal plane orientation; therefore we would like to dynamically choose the plane orientation based on current image parameters.

We have found by experience that the green band is often not needed for a good segmentation, so we can reduce the problem conceptually to finding the slope of a line in two dimensional Red/Blue space, rather than finding the normal of a plane in RGB space. Figure 2(b) shows a *scatter diagram* of the red and blue components of Figure 2(a); this can be thought of as a projection of the RGB space onto the Red/Blue plane, or as a two dimensional histogram. Road pixels cluster nicely, distinct from non-road pixels, and it should be clear that the line drawn in the figure will successfully separate road and non-road clusters and therefore segment the image. This line is the linear discriminant function in Red/Blue space.

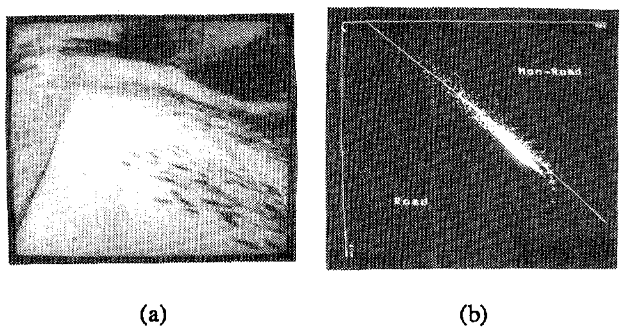


Figure 2. Road image. (a) Original. (b) Red/blue scatter diagram of image. Line in (b) depicts road/non-road boundary.

The slope of the line determines the red and blue components of the tricolor operation, with the green component equal to zero. Because of the consistent "footprint" of the road cluster, the slope of its principal axis is identical to the slope of the desired line. The angle the principal axis makes with respect to the red axis ($\theta$) determines the red, green, and blue color parameters as follows: $(r, g, b) = (\cos\theta, 0, \sin\theta)$.

The method to dynamically choose color parameters, then, proceeds as follows: sample road points in the image, calculating the orientation of the road cluster $\theta$ and then $r$, $g$, and $b$. This provides the normal of the plane in RGB space, or equivalently the line in Red/Blue space, that separates the road and non-road clusters.

#### 3.2.1.2. Threshold Selection

Once the color parameters (i.e. the plane normal or line slope) are known (either calculated or preset), the tricolor operation is performed, creating an image for which each pixel represents the distance (which may be positive or negative) from the original RGB pixel to the plane $rR + gG + bB = 0$, or, equivalently, the distance between a pixel and the origin when both are projected along a line normal to this plane. Choosing a value $\lambda$ with which to threshold the new image, then, is equivalent to translating the separating plane in RGB space.

The resulting binary image is a function of the threshold $\lambda$, which is selected by sampling a population of road pixels. In the original version of VITS, we did a histogram equalization of the feature image (the tricolor result) and used a constant threshold to segment. This assumed that the road occupied a constant percentage of the image pixels from image to image. Because this assumption is not generally true and because the equalization was too expensive with our hardware, we opted for a more robust and faster method of calculating the threshold by sampling road pixels. The road sampling technique is described in the next section.

The original threshold was calculated as the mean of the road cluster plus a constant number of standard deviations. This proved to be very sensitive to the presence of shadows, dirt on the road, potholes, and patches of new tarmac used in road repair; these often caused the calculated road mean to be unreliable. To overcome some of these problems, we chose the *median* of the top M sampled values (typically M = 15), rather than the mean of the whole sample population, as the nominal threshold value. This takes advantage of the knowledge that true road pixels are brighter in the feature image than most of these problem areas (with the nagging exception of dirt).

### 3.2.1.3. Road Sampling and Power Windows

Sampling road pixels in a dynamic environment is not straightforward. Our original implementation sampled at 125 fixed image locations, as shown in Figure 3(a), assuming that the road covered these points. Because the road changes position within the field of view from frame to frame, the sampling windows sometimes fell partially off the road, sampling dirt or grass. Since dirt and grass tend to fall above the road cluster boundary, this upset the calculation of the threshold.

To prevent sampling portions of the scene outside of the road boundary, **power windowing** was developed for road sampling. Instead of sampling at *fixed* image locations, the sampling window is projected onto the *predicted* road position in the image. Because of vehicle motion, this involves projecting a trapezoid representing the boundary of the road found in the previous scene model into world coordinates and then back into the new image plane location. This trapezoid, the prediction of the location of the road in the new image, is now the bounding window for image sampling, given the new position and orientation of the vehicle. This relies on the position update information to do the geometric calculations between the previous and the present vehicle positions. Figure 3(b) shows the sampling window computed as Alvin travels around a curve.

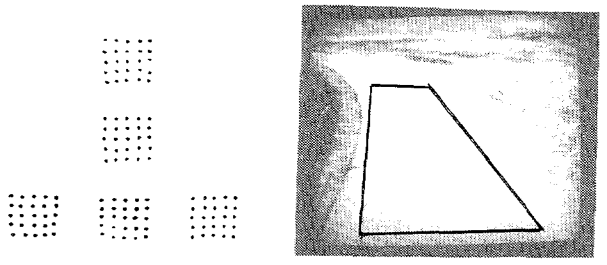

(a) (b)

Figure 3. (a) Original fixed sampling points.
(b) Sampling window around sharp curve, calculated by power windowing module.

Power windowing is used along with or independent of pan/tilt control. Without the pan/tilt mechanism, it gives the ALV a software panning ability; with it, power windowing provides fine adjustment for road sampling. In relatively straight portions of road terrain, power windowing alone is preferred, because of the time involved in panning the camera. Even small angular panning is significant because of the acceleration and deceleration times of the pan/tilt mechanism.

Because Alvin may travel as much as 10 m between successive scene acquisitions at top speeds, the projection of the old road model into the new road image may be small and fill only the lower portion of the image. To compensate for this, we use the speed from the previous position update to extend the top of the window forward so that it reaches a fixed distance in front of the vehicle. A larger sample area reduces the danger of sampling solely on a patch of dirt, shadow or stained road.

### 3.2.2. *Red Minus Blue* Segmentation

The segmentation algorithms are implemented on an image processing computer and take advantage of the computer's frame rate convolution and lookup table operations. The feature reduction is accomplished by a tricolor operation, a weighted sum of the red, green, and blue images, as described above. Typical values for $(r, g, b)$ are $(0.5, 0.0, -0.5)$; hence the name "Red minus Blue".

This segmentation method was originally motivated by noticing that the road appears darker than the dirt on the road shoulder in the red image and brighter than the dirt in the blue image. Since the spectral content of the pavement is "mostly blue" and the dirt bounding the road is "mostly red", subtracting the images became an obvious way to enhance the road/non-road boundary.

A threshold value is chosen from the road statistics to differentiate road and non-road clusters, as discussed above. The resulting image is then thresholded to produce the binary road/non-road image, simply one pass of the image through a lookup table.

### 3.2.3. Color Normalization

Another clustering algorithm involves segmenting a color normalized image, rather than the "Red minus Blue" feature image. In an image of a road with shadows falling on it, for example, the color intensities vary quite a bit within a single surface; intensities from the shaded regions are much smaller than those from the the sunny region of the road. Intuitively, normalizing the color components will allow both shaded and sunny regions to cluster together. This assumes that the ambient illumination is identical or similar in spectral content to the incident illumination of the scene. Gershon *et al.* [6] discuss this assumption and propose a tool that can be used to classify whether discontinuities in an image are due to material changes or shadows.

We have found that using a normalized blue feature image enhances the pavement/dirt boundary and therefore gives a good road/non-road segmentation. The calculation of this feature and the resulting segmentation is described by

$$I'(i,j) = \begin{cases} 1 & \text{if } \dfrac{B}{R+G+B} - \lambda > 0 \\ 0 & \textit{otherwise} \end{cases} \tag{2}$$

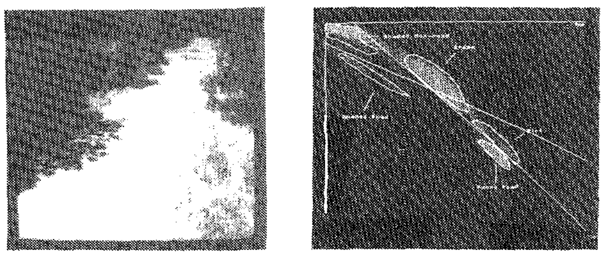
The threshold equation of (2) can be rewritten as

$$\lambda R + \lambda G + (\lambda - 1)B < 0$$

This can also be implemented as a tricolor operation with $(r, g, b)$ equal to $(\lambda, \lambda, \lambda - 1)$. This is equivalent to a plane segmentation of the RGB color space, where the dynamically chosen threshold actually varies the *orientation* of the plane, rather than its translation as in the "Red minus Blue" algorithm. Calculation of the threshold $\lambda$ proceeds as described in Section 3.2.1.2.

### 3.2.4. Shadow Boxing

Segmenting the road using a single threshold on a combined RGB image supposes that the road cluster is the only significant cluster in an RGB half-space. If there are significant non-road regions inside of the half-space defined by the plane normal $(r, g, b)$ and the threshold $\lambda$, they will also be labeled as "road" and perhaps cause faulty scene models to be output. Figure 4 shows a scatter diagram of such a case: a large region labeled "shaded non-road", caused primarily by shadows of bushes and ditches off the road, falls in the *road* half-space. Shadows that fall on the road are close to this region in the scatter diagram; the cluster labeled "shaded road" must be distinguished from the "shaded non-road".



(a) (b)

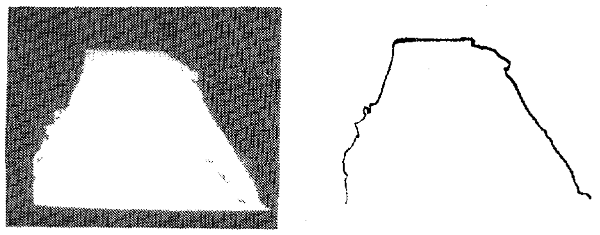
Figure 4. (a) Original image. (b) Scatter diagram of road scene with shadows. Location of threshold line indicates that "shaded non-road" will be segmented as "road".

This could perhaps be solved by segmenting twice, corresponding to the two threshold lines in Figure 4(b), and performing a logical AND of the resulting binary images. The dynamic threshold calculation for the boundary between the sunny and shaded road regions is very sensitive to noise, however, because there is very little information in the shaded regions, even when digitized from a camera with a reasonably good dynamic range. Rather than segmenting complete half-spaces, then, the road regions are bounded by rectangles, and only the *boxed* regions are segmented and labeled as road. This is particularly helpful in conditions with significant shadows; hence the name "Shadow Boxing". Shadow boxing is similar to a dynamic Bayesian classifier [3] with three decision regions and rectangular decision boundaries. The bounding boxes are again motivated by the current hardware, as the segmentation can be implemented quickly by two global lookup table operations per boxed region.

### 3.3. Boundary Extraction

The segmentation algorithms produce a binary road/non-road image. From this image, the road edges are extracted and transformed into three dimensional coordinates to fill in the scene model. The boundary extraction is an edge-tracking process in which the road boundary is found and then traced while keeping track of the image locations.

The initial task is to find the road/non-road boundary in the image. To facilitate easier boundary tracing and to avoid looking for the road on or above the horizon, a false road boundary is added around the image, creating an artificial horizon, as in Figure 5(b) — this prevents following the road edges up into the sky. In order to find an initial boundary, we start in the bottom quarter of the image and step upwards until a boundary is detected. The border is then traced in both directions, using an 8-neighbor non-road, 4-neighbor road connectivity rule, and image coordinates of boundary points are saved. The boundary detection and tracing uses preset "skip factors" in both row and column directions to speed processing; this effectively reduces the image size by the row and column skip factors. The boundary tracking method allows for reasoning on the fly — "bubbles" are properly ignored, and globally non-linear segments, such as corners of intersections, can be detected and noted. When the right or left false boundary is detected, the corresponding road edge is known to be out of the camera's field of view. Figures 5(a) and (b) show the segmented road and the boundary traced for the image of Figure 2(a).



(a) (b)

Figure 5. (a) Binary road image with false boundary added. (b) Road edges.

Once the image coordinates of both right and left road edge points are found, we choose a small number of points (up to ten) on each edge to send to the geometry module to include in the scene model. The row locations of these points are spaced by a quadratic function so that the three dimensional locations of the points will be approximately equal distances apart. Also, rather than choosing distinct image points, local edge points are averaged to smooth the road description.

### 3.4. Three Dimensional Geometry Transformations

Once road edge points are selected in the image, a three dimensional description, the scene model, must be sent to Reasoning for trajectory calculation. This process of recovering the three dimensional information projected onto the two dimensional image plane is the "inverse optics" problem of vision. As Poggio [11] and others have pointed out, this is an under-constrained (formally "ill-posed") problem that requires the introduction of generic constraints to arrive at a unique solution. In our case, such constraints are assumptions about the structure of the environment. We call the various techniques which result from the adoption of such constraints solutions to the *forward-geometry* problem for the ALV.

VITS also uses the results of what we call the *inverse-geometry* problem. The inverse-geometry problem is to determine the location within the image plane of the image of a point whose three dimensional location is known. Unlike the forward-geometry problem, the inverse-geometry problem has an exact solution; no assumptions need to be made in order to constrain the problem and make it well-posed. The combination of the forward-geometry and inverse-geometry processes allow for frame-to-frame registration of features as well as predictions about, for example, the continuation of the road.

The original forward-geometry module used in VITS was essentially a model driven "shape from contour" method developed at the University of Maryland [15], based on calculating the vanishing point of parallel lines projected onto the image plane. Experiments soon showed that assuming a *flat-earth* road model allows for a much faster forward-geometry module and performs adequately for the roads Alvin encounters and the speeds attained through the 1986 demonstrations. While flat-earth geometry is clearly an assumption that is very useful in certain circumstances, it is not accurate enough for all road-following applications. Work is proceeding to incorporate a *hill-and-dale* geometry module that uses a fast "shape from contour" method to solve the forward-geometry problem. The flat-earth and hill-and-dale road models are discussed in the following sections, preceded by a description of the relevant coordinate systems and followed by a discussion of the inverse-geometry solution and its uses within VITS.

### 3.4.1. Flat-Earth Geometry Model

In the *flat-earth* geometry model we assume that the road is planar, and that the plane containing the visible portion of the road is the same plane which is giving support to the vehicle. Thus, the three dimensional location of an edge point found at $(col, row)$ can be determined by finding the point of intersection of the vector from the focal point of the camera through this point with this plane.

The flat-earth geometry model has several advantages over the other forward-geometry models. First of these is its speed; a straightforward calculation gives the three dimensional location for a given image point. Second, this model can be applied to any single image point, even those which are not edges of the road; there is no need for multiple image points as in the vanishing point geometry model [15]. Third, the error in the output three dimensional locations is only a function of the extent to which the flat-earth assumption is violated, and not additionally a function of the goodness of the segmentation.

In practice there are a number of problems which limit the applicability of this technique. First is its sensitivity to inaccuracies in the assumed tilt angle formed by the camera to the road plane. The camera is in a fixed position relative to the body of the ALV, but the body of the vehicle is able to rock forward and backward on the undercarriage. When traveling uphill or downhill, the vehicle body rocks, decreasing or increasing, respectively, the effective tilt angle of the camera, causing parallel road edges to be output as converging or diverging three dimensional edge segments. Because of the problems caused by this rocking motion of the vehicle, we are adding sensors which will measure the angle formed between the vehicle body and undercarriage.

A second problem with the flat-earth model occurs at hills and valleys, which cause the description of the road to diverge and converge, respectively. This problem is addressed by the hill-and-dale geometry model described below.

### 3.4.2. Hill and Dale Geometry Model

The "hill-and-dale" geometry model was developed to address the problems of converging and diverging scene model edges. The essence of this technique is to use the flat-earth geometry model for the two roadway points nearest the vehicle in the image, and then to force the road model to move up or down from the flat-earth plane so as to maintain a constant width.

Let $\mathbf{p}(i,j)$ be the world coordinates of the points which appear in the image as indicated in Figure 6. The first step of the algorithm is to use flat-earth geometry to solve for $\mathbf{p}(0,1)$ and $\mathbf{p}(0,2)$. From this it is possible to compute the road width $W$, where $W = ||\mathbf{p}(0,1)-\mathbf{p}(0,2)||$.

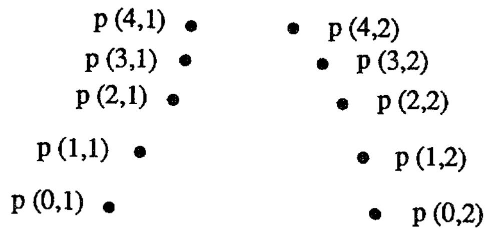


Figure 6. Road edge points to be converted into the vehicle coordinate system.

One way to maintain a constant road width in the scene model is to intersect the successive pair of edge points (i,1) and (i,2) with a different plane than the ground plane. To see this, note that the rays from the camera origin through the image locations for these points are diverging; thus using a plane above the plane defined by $h_0$ will produce a narrower road than using a plane below the plane defined by $h_0$. For each successive pair of edge points (i,1) and (i,2) we can compute the elevation of a plane containing these points, perhaps above or below the assumed ground plane, such that the the road maintains the same width. The elevation is then used in a flat-earth geometry calculation to produce scene models for which the road is of constant width when measured at paired scene model points.

Testing has shown that this algorithm produces more accurate scene models than the flat-earth algorithm on straight or slightly curved roads which go up and down hill when the segmentation is good. However the algorithm is very dependent upon a good segmentation, as a slightly wider road segmentation will cause the road to appear to travel uphill, and a slightly narrower road segmentation will cause the road to appear to travel downhill.

A potentially larger drawback to this algorithm is its behavior near and in curves and intersections. Many curves exhibit banking which this algorithm is unable to reproduce; the apparent location of the lower edge of the banked road will always be too high, and that of the outer edge will always be too low. Also, the selection of opposing pairs of edge points is critical, since the width of the road is measured between these pairs of points. Thus, if the road is curving it is necessary to select pairs of edge points such that the resulting tiles of the road are pie shaped. Finally, it should be noted that the constant width premise of this algorithm is violated at intersections, and may be violated at other roadway areas as well.

We are currently investigating heuristics for selecting matched edge points. DeMenthon [1] uses a "zero-bank" road model which addresses this problem by the introduction of additional constraints.

## 4. DISCUSSION

The ALV public demonstrations in 1985 and 1986 have demonstrated Alvin's road-following and obstacle avoidance capabilities. Successful runs have been made that switched back and forth between video-based and range-based road-following and demonstrated obstacle avoidance based on a fusion of both video and range data. A subset of the vision system under development at the University of Maryland [15] has run Alvin over part of the test area. Current vision efforts are concentrated on speed and robustness of road-following, obstacle detection and location, as well as vision for off-road navigation. Long term research areas include object modeling, landmark recognition, terrain typing, stereo, and motion analysis. Researchers at many groups are currently working on these problems for future ALV application. Their efforts are critical to meeting future demonstration requirements, and to the success of the program in general. Interaction with these groups has influenced our present system to a large degree.

To travel at higher vehicle speeds, the vision system must not only provide scene models more rapidly but also provide longer scene models to allow for the distance needed to stop in case of an emergency or to slow down for a detected obstacle. We are presently investigating methods to more accurately model the road at far distances and to extend calculated road edges based on road history and assumptions about road curvature. Because of limited field of view and accuracy in the range of the current range scanner, we are working on fast methods to detect obstacles at a distance using video data [12].

Of the video road-following algorithms described, the "Red minus Blue" algorithm has proved to be the most dependable so far, and it has been used (at different stages of development) in the formal demonstrations to date. The color normalization algorithm performs well in very sunny conditions when shadows present a problem to "Red minus blue". "Shadow boxing" is designed to deal with shadows and obstacles. It has been tested but not yet used to drive the ALV.

## 5. SUMMARY

Experiments in mobile robot road-following prove the importance of an evolving, robust vision system to model the environment for navigation. Such a system must exhibit intelligent behavior under varying vehicle and environmental conditions: seasonal variation in scene characteristics, diverse and changing weather conditions, unexpected visual information (e.g. obstacles, shadows, potholes), changes in navigation and control systems, and changing sensor characteristics. Particular conditions that have proven difficult to handle are the presence of dirt on the road, spectral reflection when the sun is at a low angle, shadows on the road, and tarmac patches (used to repair road segments). For this reason, research in mobile robot vision is largely an incremental process of hypothesis and test. Analyzing the failure of a particular test is often much more informative than a success.

We have presented the vision system for Alvin, the Autonomous Land Vehicle, discussing in particular the task of video road-following. The system provides an effective level of behavior in both speed and performance. As the processing power of the ALV increases and the performance requirements become more ambitious, the system must become more robust, faster, and more "intelligent". Work is progressing at a number of institutions to meet these goals.

## REFERENCES

[1] D. DeMenthon, "Inverse perspective of a road from a single image," Technical Report CAR-TR-210, Center for Automation Research, Univ. of Maryland, July 1986.

[2] R. J. Douglass, "The Second Generation ALV architecture," *Proc. IEEE International Conference on Robotics and Automation,* Raleigh, April 1987.

[3] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* New York, Wiley-Interscience, 1973.

[4] R. T. Dunlay and D. G. Morgenthaler, "Robot road-following using laser-based range imagery," *Trans. SME Second World Conference on Robotics Research,* Scottsdale, AZ, August 1986.

[5] R. T. Dunlay and D. G. Morgenthaler, "Obstacle detection and avoidance from range data," *Proc. SPIE Mobile Robots Conference,* Cambridge, MA, October 1986.

[6] R. Gershon, A. D. Jepson, and J. K. Tsotsos, "The effects of ambient illumination on the structure of shadows in chromatic images," Technical Report RBCV-TR-86-9, Dept. of Computer Science, Univ. of Toronto, 1986.

[7] A. R. Hanson and E. M. Riseman, "Segmentation of Natural Scenes," In *Computer Vision Systems,* A. R. Hanson and E. M. Riseman (eds.), Academic Press, 1978.

[8] M. Hebert and T. Kanade, "Outdoor scene analysis using range data," *Proc. IEEE International Conference on Robotics and Automation,* San Francisco, pp. 1426-1432, April 1986.

[9] D. T. Lawton, T. S. Levitt, C. McConnell, and J. Glicksman, "Terrain models for an autonomous land vehicle,"

*Proc. IEEE International Conference on Robotics and Automation,* San Francisco, pp. 2043-2051, April 1986.

[10] J. M. Lowrie, M. Thomas, K. Gremban, and M. Turk, "The autonomous land vehicle (ALV) preliminary road-following demonstration," *Intelligent Robots and Computer Vision,* David P. Casasent, Ed., Proc. SPIE 579, pp. 336-350, Sept. 1985.

[11] T. Poggio, "Early vision: From computational structure to algorithms and parallel hardware," *Computer Vision, Graphics, and Image Processing,* Vol. 31, pp. 139-155, 1985.

[12] M. A. Turk and M. Marra, "Color road segmentation and video obstacle detection," *Proc. SPIE Mobile Robots Conference,* Cambridge, MA, October 1986.

[13] R. S. Wallace, "Robot Road Following by Adaptive Color Classification and Shape Tracking," *Proc. AAAI-86.*

[14] R. Wallace, K. Matsuzaki, J. Crisman, Y. Goto, J. Webb, and T. Kanade, "Progress in robot road-following," *Proc. IEEE International Conference on Robotics and Automation,* San Francisco, pp. 1426-1432, April 1986.

[15] A. M. Waxman, J. LeMoigne, L. Davis, E. Liang, and T. Siddalingaiah, "A visual navigation system," *Proc. IEEE International Conference on Robotics and Automation,* San Francisco, pp. 1600-1606, April 1986.