VITS—A Vision System for Autonomous Land Vehicle Navigation

MATTHEW A. TURK, MEMBER, IEEE, DAVID G. MORGENTHALER, KEITH D. GREMBAN, AND MARTIN MARRA

Abstract—**In order to adequately navigate through its environment, a mobile robot must sense and perceive the structure of that environment, modeling world features relevant to navigation. The primary vision (or perception) task is to provide a description of the world rich enough to facilitate such behaviors as road-following, obstacle avoidance, landmark recognition, and cross-country navigation. We describe VITS, the vision system for Alvin, the Autonomous Land Vehicle, addressing in particular the task of road-following. The ALV has performed public road-following demonstrations, traveling distances up to 4.5 km at speeds up to 10 km/hr along a paved road, equipped with an RGB video camera with pan/tilt control and a laser range scanner. The ALV vision system builds symbolic descriptions of road and obstacle boundaries using both video and range sensors. We describe various road segmentation methods for video-based road-following, along with approaches to boundary extraction and transformation of boundaries in the image plane into a vehicle-centered three dimensional scene model.**

Index Terms—**Autonomous navigation, computer vision, mobile robot vision, road-following.**

I. INTRODUCTION

TO achieve goal-directed autonomous behavior, the vision system for a mobile robot must 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 goal-directed behaviors include road-following, obstacle avoidance, cross-country navigation, landmark detection, map building and updating, and position estimation. The basic vision task is to provide a description of the world rich enough to facilitate such behaviors. The vision system must then interpret raw sensor data, perhaps from a multiplicity of sensor and sensor types, and produce consistent symbolic descriptions of the pertinent world features.

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. A second public demonstration in June 1986 showed the improved road-following ability of the system, allowing the ALV to travel a distance of 4.2 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 speeds up to 20 km/hr were achieved on a straight, obstacle-free road. This paper describes Alvin's vision system and addresses the particular task of video road-following. Other tasks such as obstacle detection and avoidance and range-based road-following are discussed elsewhere [10], [11], [36].

A. A Brief Review of Mobile Robot Vision

SRI's Shakey was the first mobile robot with a functional, albeit very limited, vision system. Shakey was primarily an experiment in problem solving methods, and its blocks world vision system ran very slowly. The JPL robot [32] used visual input to form polygonal terrain models for optimal path construction. Unfortunately, the project halted before the complete system was finished.

The Stanford Cart [25], [26] used a single camera to take nine pictures, spaced along a 50 cm track, and used the Moravec interest operator to pick out distinctive features in the images. These features were correlated between images and their three dimensional positions were found using a stereo algorithm. Running with a remote, time-shared computer as its "brain," the Stanford Cart took about five hours to navigate a 20 meter course, with 20 percent accuracy at best, lurching about one meter every ten to fifteen minutes before stopping again to take pictures, think, and plan a new path. The Cart's "sliding stereo" system chose features generally good enough for navigation in a cluttered environment, but it did not provide a meaningful model of the environment.

Tsugawa *et al.* [34] describe an autonomous car driven

Manuscript received December 15, 1986; revised May 15, 1987. This work was performed under the Autonomous Land Vehicle Program supported by the Defense Advanced Research Projects Agency under Contract DACA76-84-C-0005.

M. A. Turk was with Martin Marietta Denver Aerospace, P.O. Box 179, M. S. H0427, Denver, CO 80201. He is now with the Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.

D. G. Morgenthaler and M. Marra are with Martin Marietta Denver Aerospace, P.O. Box 179, M.S. H0427, Denver, CO 80201.

K. D. Gremban is with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, on leave from Martin Marietta Devner Aerospace, P.O. Box 179, M.S. H0427, Denver, CO 80201.

IEEE Log Number 8820101.

up to 30 km/hr using a vertical stereo pair of cameras to detect expected obstacles, but its perception of the world was very minimal, limiting its application to a highly constrained environment. The "intelligent car" identified obstacles in an expected range very quickly by comparing edges in vertically displaced images. A continuous "obstacle avoidance" mode was in effect, and a model of the world was not needed.

A vision system for a mobile robot intended for the factory floor was presented by Inigo *et al.* [18]. This system used edge detection, perspective inversion, and line fitting (via a Hough transform) to find the path, an *a priori* road model of straight lines, and another stereo technique using vertical cameras, called *motion driven scene correlation*, to detect obstacles. The Fido vision system [33] uses stereo vision to locate obstacles by a hierarchical correlation of points chosen by an interest operator. Its model of the world consists of only the 3-D points it tracks, and it has successfully navigated through a cluttered environment and along a sidewalk. Current work in multisensory perception for the mobile robot Hilare is presented by de Saint Vincent [7], describing a scene acquisition module, using stereo cameras and a laser range finder, and a "dynamic vision" module for robot position correction and tracking world features. Another stereo vision system based on matching vertical edges and inferring surfaces is described by Tsuji *et al.* [35].

The goal of a mobile robot project in West Germany is to perform autonomous vehicle guidance on a German Autobahn at high speeds [8], [22], [29]. The current emphasis is on control aspects of the problem, incorporating a high-speed vision algorithm to track road border lines. The system has performed both road-following and vehicle-following in real-time.

Other mobile robots have been or are being developed that use sensors particularly suited to an indoor environment (e.g., [4], [19]). The project headed by Brooks [2] implements a novel approach to a mobile robot architecture, emphasizing levels of behavior rather that functional modules; much of the current vision work may be incorporated into such a framework.

B. ALV Background

The Autonomous Land Vehicle project, part of DARPA's Strategic Computing Program, is intended to advance and demonstrate the state of the art in image understanding, artificial intelligence, advanced architectures, and autonomous navigation. A description of the project and the initial system configuration is found in [24]. Related vision research is proceeding concurrently by a number of industrial and academic groups, as is work in route and path planning, as well as object modeling and knowledge representation. The ALV project is driven by a series of successively more ambitious demonstrations. The ultimate success of the project depends on coordination among the different groups involved to enable rapid technology transfer from the research domain to the application domain. As the ALV is intended to be a national testbed for autonomous vehicle research, various vision systems and algorithms will eventually be implemented. Some of the current work is briefly described in the remainder of this section.

Vision research areas currently being pursued in relation to the ALV program include object modeling, stereo, texture, motion detection and analysis, and object recognition. An architecture for terrain recognition which uses model-driven schema instantiation for terrain recognition is presented by Lawton *et al.* [23]. Such representations for terrain models will be important for future cross-country navigation. Waxman *et al.* [40], [41] present a visual navigation system that incorporates rule-based reasoning with image processing and geometry modules. The system, developed at the University of Maryland, finds dominant linear features in the image and reasons about these features to describe the road, using bootstrap and feed-forward image processing phases. In the feed-forward phase, previous results are used to predict the location of the road in successive images. A subset of this system has been used to autonomously drive the ALV for short distances. DeMenthon [6] describes an alternative geometry module for the above visual navigation system.

Significant ALV-related work is proceeding at Carnegie-Mellon University (CMU). A review of recent results from the CMU program is presented by Goto and Stentz [13]. Outdoor scene analysis using range data from a laser range scanner is presented by Hebert and Kanade [16], describing methods for preprocessing range data, extracting three dimensional features, scene interpretation, map building, and object recognition. Fusion of video and range data is also discussed. Range data processing has been used on the CMU Navlab to demonstrate obstacle avoidance capabilities. Vision algorithms used for successful outdoor navigation of the CMU Terregator are described by Wallace *et al.* [37]–[39]. The Terregator has achieved continuous motion navigation using both edge-based and color-based sidewalk finding algorithms.

Hughes Artificial Intelligence Center is developing knowledge-based vision techniques for obstacle detection and avoidance using the concept of a *virtual sensor* which blends raw sensor data with specialized processing in response to a request from the planning system [5], [30]. Work at SRI International is focused on object modeling and recognition, and on modeling uncertainty in multiple representations [1].

FMC Corporation and General Dynamics have demonstrated successful transfer of ALV technology to mission-oriented scenarios of mixed teleoperation and autonomous navigation, performed at the Martin Marietta test site in 1986. Kuan *et al.* [20], [21] describe FMC's research in vision-guided road-following. Other university and industrial laboratories which are engaged in vision research related to ALV include Advanced Decision Systems, Columbia University, General Electric, Honeywell Research Center, MIT, University of Massachusetts at Amherst, University of Rochester, and USC. The Proceedings of the February 1987 Image Understanding

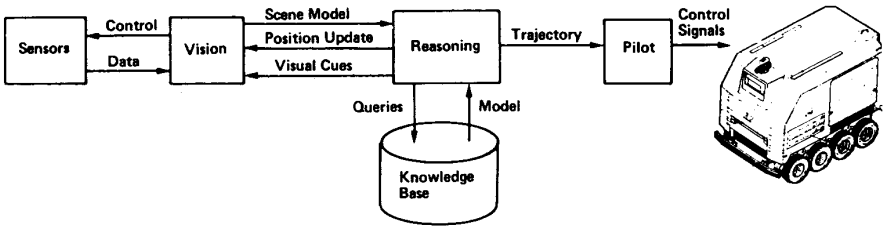


Fig. 1. The ALV system configuration.

Workshop, sponsored by DARPA, contains descriptions and status reports of many of these projects.

The vision system described in this paper (VITS) is the system meeting the perception requirements for testing and formal demonstrations of the ALV through 1986. Section II gives a system overview, briefly describing the various subsystems; it is important to understand the vision system in its context. Video-based road following is discussed in Section III, describing sensor control, road segmentation, road boundary extraction, and geometric transformation to three dimensional world coordinates.

II. ALV System Overview

It is important to view Alvin's vision subsystem as an integral part of a larger system, which can affect and be affected by the performance of the system as a whole. Fig. 1 illustrates the basic system configuration of the ALV, including the interfaces to the major modules. In the paragraphs below, each of Alvin's major components will be discussed in the context of the interaction as a complete system.

A. Hardware Components

The primary consideration behind selection of the hardware components was that Alvin is intended to be a testbed for research in autonomous mobility systems. Consequently, it was necessary to provide Alvin with an undercarriage and body capable of maneuvering both on-road and off-road, while carrying on board all the power, sensors, and computers needed for autonomous operation. In addition, the requirements of autonomous operation directed the selection of sensors and processing hardware.

1) Vehicle: Fig. 2 is a photograph of Alvin. The overall vehicle dimensions are 2.7 m wide by 4.2 m long; the suspension system allows the height of the vehicle to be varied, but it is nominally 3.1 m.

Alvin weighs approximately 16 000 pounds fully loaded with equipment, yet is capable of traveling both on-road and off-road. The undercarriage is an all-terrain built by Standard Manufacturing, Inc. The basic vehicle is eight-wheel drive, diesel-powered, and hydrostatically driven. Alvin is steered like a tracked vehicle by providing differential power to the two sets of wheels.

Alvin's fiberglass shell protects the interior from dust and inclement weather, and insulates the equipment inside. The shell provides space for six full-size equipment racks, as well as room for service access. The electronics within the ALV are powered by an auxiliary power unit.

Fig. 2. Alvin.

An environmental control unit cools the interior of the shell.

2) Sensors: In order to function in a natural environment, an autonomous vehicle must be able to sense the terrain around it, as well as keep track of heading and distance traveled. The ALV hosts a number of sensors to accomplish these tasks.

Alvin's sense of direction and distance traveled is provided by odometers on the wheels coupled to a Bendix Land Navigation System (LNS). These sensors enable Alvin to follow a trajectory derived from visual data or read from a prestored map. The LNS provides direction as an angle from true North, while distance traveled is provided in terms of horizontal distance (Northings and Eastings), and altitude.

Two imaging sensors are currently available on the ALV for use by VITS. The primary vision sensor is an RCA color video CCD camera, which provides 480 × 512 red, green, and blue images, with eight bits of intensity per image. The field of view (38° vertical and 50° horizontal) and focus of the camera are kept fixed. The camera is mounted on a pan/tilt unit that is under direct control of the vision subsystem.

The other vision sensor is a laser range scanner, developed by the Environmental Research Institute of Michigan (ERIM). This sensor determines range by measuring the phase shift of a reflected modulated laser beam. The laser is continuously scanned over a field of view that is 30° vertical and 80° horizontal. The output of the scanner is a digital image consisting of a 64 × 256 array of pixels with 8 bits of range resolution.

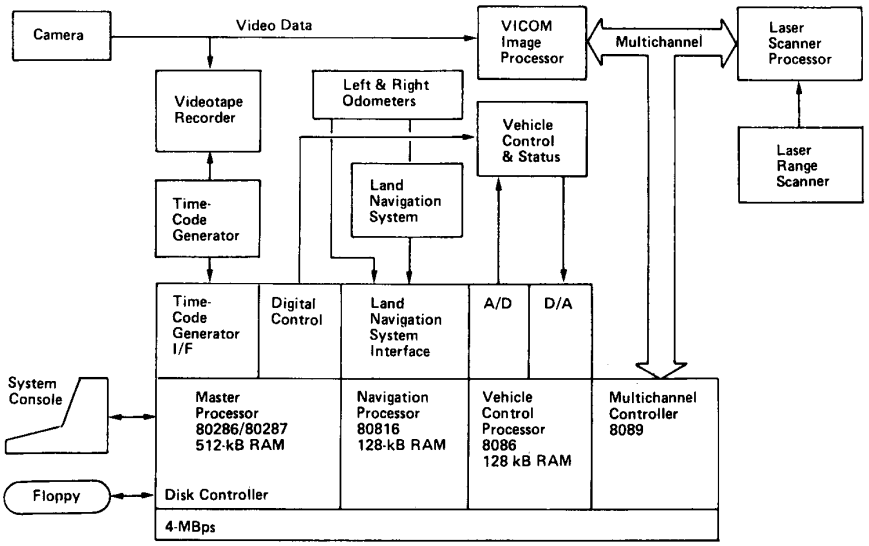


Fig. 3. The first-generation ALV processor configuration.

3) Computer Hardware: Alvin currently uses a variety of computers, resulting from the range of processing requirements of the different software subsystems. The diverse processing requirements were met by designing a modular multiprocessor architecture. VITS is hosted on a Vicom image processor, while the other software subsystems are hosted on an Intel multiprocessor system. VITS communicates with the other subsystems across a dedicated communication channel, while the other subsystems communicate across a common bus. Fig. 3 depicts the processor configuration.

The special capabilities of the Vicom hardware were important to the development of the ALV vision subsystem (VITS). The Vicom contains video digitizers, and can perform many standard image processing operations at near video frame rate (1/30 second). For example, 3×3 convolution, point mapping operations (such as thresholding, or addition and subtraction of constants), and image algebra (such as addition or subtraction of two images) are all frame rate operations. The Vicom also contains a general purpose microcomputer for additional, user-defined operations.

As stated above, Alvin is intended to be a testbed for autonomous systems. In fulfilling this charter, plans have been made to integrate a number of advanced experimental computer architectures in future generations of the ALV system. This will begin with a new architecture in early 1987.

B. Vision

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. Range data processing has been implemented on the ALV, and results of range-based road-following and obstacle avoidance are presented in [10], [11].

The vision system software resides entirely on the Vicom image processor, which also houses a board dedicated to camera pan/tilt control and a board to enable communication with the Intel system. Nearly all of the application code is written in Pascal and uses the Vicom-supplied libraries for accessing high-speed image operations. Some low level control routines have been implemented in Motorola 68000 assembly language.

The responsibility of the vision subsystem in road-following is to process data in the form of video 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 three different forms: the scene model, the position update, and visual cues. A special communication control processor, part of the utilities subsystem, mediates communication between VITS and the other subsystems. The control processor shares memory with VITS, and handles communication by examining the content of key memory locations every 100 ms and modifying them as appropriate.

1) Scene Model: The *scene model*, a description of the observed road, is the output of the vision subsystem after each frame of images is processed. The scene model contains a record of Alvin's position and heading at the time of image acquisition, a description of the road found in the imagery, consisting of lists of vehicle-centered 3-D points denoting left and right road edges, and an optional list of points surrounding an obstacle. The reasoning subsystem must then transform the road description into a fixed, world coordinate system for navigation. VITS may optionally specify the scene model in world coordinates;

this is more efficient when data acquired from multiple sensors or at different times is used to create the scene model.

Since the time needed to compute a scene model is nondeterministic, VITS sets a "scene model ready" flag indicating that a new scene model is ready to be processed. The communication controller examines this flag, and, when set, transfers the scene model to the reasoning subsystem and clears the flag.

Fig. 4 illustrates the format of a scene model. Fig. 5 is an example of a hypothetical road scene and the corresponding scene model.

2) Position Update: 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 (see Section III-B-1-d).

Communication of vehicle motion and position information is effected by means of a *position update* message passed from Reasoning to the vision subsystem. The position update specifies the current vehicle speed, position, and heading. Synchronization of position update and image acquisition is mediated by a position update request. At the time VITS digitized an image, the "position update request" flag is set. When the communication controller finds the flag set, it sends a message to Reasoning which immediately (within 100 ms) generates the required information, builds a position update message, and sends it to VITS.

3) Visual Cues: The reasoning subsystem interfaces to a knowledge base which contains information about the test area. Some of this information can be used by VITS to specify behavior (find road, locate obstacles, pause, resume) or to optimize processing, much as the information on a road map can guide a driver. When Reasoning determines that a visually identifiable feature should be within the field of view, a *visual cue* is sent to VITS enabling vision processing to be modified. In the future, when Alvin's domain becomes more complex, these cues will be used to guide the transition from one road surface to another, from on-road to off-road and vice versa, or to guide the search for a landmark. In the current version of the system, stored knowledge about the shape of the road shoulder has been used to guide a transition between range-based and video-based road-following. Apart from this, the cue facility has been used to date only to notify VITS that the vehicle is approaching a curve (which causes the camera panning mechanism to be enabled) and to send pause and resume commands to VITS.

C. Reasoning

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

```
type scene_model = record
        time: array[1..4] of word; { time stamp }
        count: word; { # of road edge records }
        x,y,psi: real; { vehicle position }
        SM_rec: array[1..10] of record
                        tag: string[2]; { left or right }
                        numpts: word; { # of points }
                        pts: array [1..10]
                              of array[1..3] of real;
                end;
        version: string[10]; {current SW version }
        num: word; { scene model # }
end;
```

Fig. 4. The scene model format.

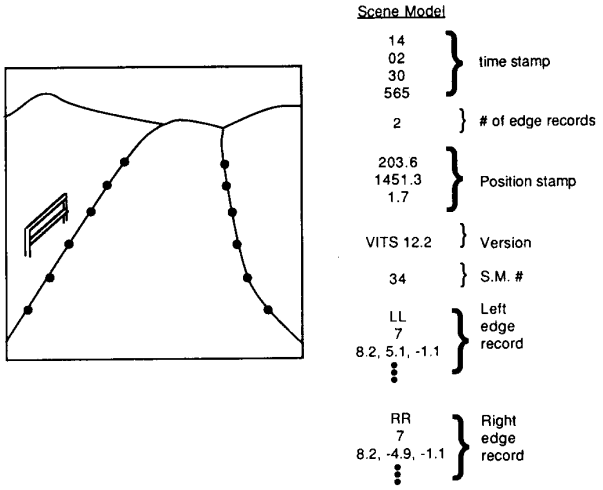


Fig. 5. Road scene and corresponding scene model.

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 back a position update. 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.

Evaluation of scene models is a powerful capability of the reasoning subsystem. Small environmental changes, such as dirt on the road, or the sudden appearance of a cloud, can significantly affect the output of the vision subsystem. Reasoning uses assumptions about the smoothness and continuity of roads to verify data from VITS. Every scene model is evaluated based on the smoothness of the road edges, and on how well they agree with previous edges. A scene model evaluated as "bad" is discarded.

Reasoning creates a new trajectory by minimizing a cost function based on current heading, curvature of the scene model, attraction to a goal, and road edge repulsion. The final trajectory is a sequence of points that lie near the

center of the road. Each point is tagged with a reference speed. The reference speeds are computed so that, if no new scene models are received, the vehicle will stop at the end of the trajectory. The trajectory is then sent to the Pilot.

The reasoning subsystem also interacts with the knowledge base to locate features significant for vision processing. As each new trajectory is generated, the knowledge base is searched to determine if any features are within the field of view of the vehicle. Features that are both within a maximum distance and a maximum angle from the current heading are incorporated into a Visual Cue which is passed to VITS.

D. Knowledge Base

The knowledge base consists of *a priori* map data, and a set of routines for accessing the data. Currently, the map data contains information describing the road network being used as the ALV test track. The map data contains coordinates which specify the location of the roadway, as well as various significant features along the road, such as intersections, sharp curves, and several local road features.

At present, the vision subsystem communicates with the knowledge base through Reasoning.

E. Pilot

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.

III. 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. Road edges are intuitively the most natural representation, since they are usually obvious (to humans, at least) in road images. Often, however, the dominant linear features in road images are the shoulder/vegetation boundaries rather than the road/shoulder boundaries. The difficulties in extracting the real road boundary from the image led us to adopt a segmentation algorithm to first extract the road in the image, track the road/nonroad boundary, and then calculate three dimensional road edge points.

The current video data processing unit (VIVD) uses a clustering algorithm to segment the image into road and nonroad regions. A detailed description of image segmentation by clustering can be found in [3]. After producing a binary road image, the road boundaries are traced and select image points are transformed 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/nonroad regions; 3) extract road boundaries by tracing the binary road edges; and 4) transform 2-D road edge points to 3-D coordinates and build the scene model. Fig. 6 depicts the flow of control in a complete scene model cycle.

A. Sensor Control and Image Acquisition

1) Camera Panning: The position of the road with respect to the vehicle may change due to a curving road, vehicle oscillation, or a sudden path correction. Consequently, the position of the road within the field of view of a fixed camera may change. Because the video segmentation algorithm requires sampling a population of road pixels, two methods were developed to maintain knowledge of the road position from frame to frame: camera panning and *power windowing*. Power windowing, a "software panning" technique, is described in Section III-B-1-d.

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 and relate the vehicle-centered road description to present vehicle location and orientation, and then to calculate and command the desired pan angle. If only one road boundary is detected, then VITS will attempt to pan the camera to the right or left to bring both road edges into view in the next image. The activation of planning is also controlled by cues from the reasoning subsystem that indicate when panning would be useful (e.g., going around a sharp corner), and when it would not be helpful (e.g., passing a parking lot).

In the initial implementation of camera panning, the camera was allowed to assume only three positions, *left*, *mid*, and *right*, with simple rules for switching from one to another based on road location in the image. With this technique we were able to successfully navigate a sharp corner; however, this was not very repeatable. After studying the failure symptoms we rejected this simplistic approach to panning in favor of a technique allowing "continuous" pan positions based on the difference between vehicle orientation and the perceived road orientation, and constraints on minimum and maximum incremental panning and on the maximum absolute pan angles.

We have not yet needed the camera tilting capability. The tilt angle is fixed at approximately 17° below the horizon.

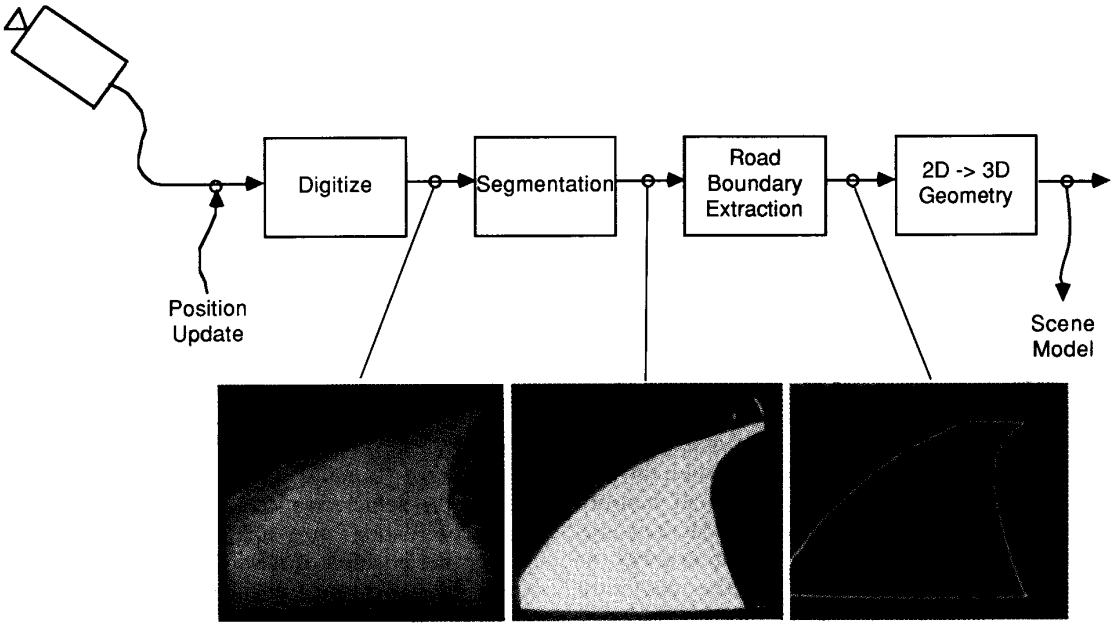


Fig. 6. VITS flow of control.

2) Image Acquisition: The image processing computer digitizes red, green, and blue images, directly into memory from the video camera. Typically, the images are blurred to reduce noise; since convolution is distributive, blurring is performed after the images are combined. Our camera has an optional automatic iris which compensates for global intensity changes. Calibration is performed on the camera before a test run to calculate the exact tilt angle and focal length, and the proper color response.

Along with the raw images, the *position update* is requested and received from the communication control processor. The position update includes a time stamp and the two dimensional position and orientation of the vehicle—this information allows conversion from a vehicle-centered road description to world coordinates.

B. Road Segmentation

There are many techniques used in computer vision systems to segment images into regions of similarity. Segmentation of natural outdoor scenes is a particularly complex problem [15], but it is simplified when a predominant feature in the scene (i.e., the road) is the main focus of the segmentation. *A priori* information is available about roads, as is information from previous frames and vehicle movement.

Road segmentation is rather simple to accomplish on a stored road image, given enough time to experiment and modify parameters. In a real-time, outdoor environment with a mobile robot, however, road segmentation is complicated by the great variability of vehicle and environmental conditions. Changing seasons, weather conditions, time of day, and manmade changes complicate the video segmentation, as do 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. 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). These will be discussed further in Section IV.

The segmentation methods used by VITS are motivated by the hardware supporting it, speed requirements, and assumptions about road and nonroad image characteristics. We have proposed and tested various segmentation techniques, all based on knowledge of road characteristics in color images. Section III-B-1 describes the specific techniques developed for the segmentation algorithms. The algorithms discussed in the following sections are somewhat cryptically called *red minus blue*, *color normalization*, and *shadow boxing*.

1) A Classification Problem: To understand the road segmentation algorithms discussed below, it is best to view road segmentation in the broader context of a general classification problem with only two classes: road and nonroad. A typical classification problem involves five basic steps: feature extraction, feature decorrelation, feature reduction, clustering, and segmentation [3]. Feature extraction is the process of computing the features used to distinguish between classes. Feature decorrelation is formally a multidimensional transformation that results in a new, orthogonal set of features. Feature reduction discards those features that are unnecessary or yield little information. Clustering partitions the feature space into K mutually exclusive regions, and segmentation assigns each pixel in the image to one of the regions.

a) VITS Clustering: The clustering algorithm used

in VITS is a real-time (i.e., abbreviated) solution to the classification problem. Step 1 is trivial. A color video image consists of a 3-tuple of red, green, and blue intensity values at every pixel; thus each pixel is a point (or vector) in RGB space. By using color as the feature space, feature extraction is equivalent to digitization. Other features, such as texture, saturation, and reflectance data from the laser scanner, have been considered but are not presently used.

Feature decorrelation and feature reduction are accomplished in a single step in our algorithm. A single plane in three-space does a very good job under most conditions of partitioning the feature space into road and nonroad regions. The original features are, in effect, combined into a single linear feature whose axis is perpendicular to the separating plane. The projection of image points onto this perpendicular line is equivalent to a dot product of the pixel vector and the normal of the plane. Feature decorrelation and reduction then reduce to the problems of finding the orientation of this plane and projecting image points onto a line normal to the plane; these are discussed in Section III-B-1-b.

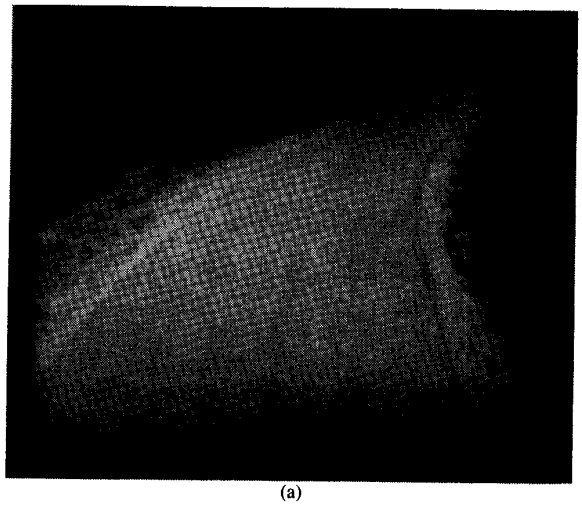
Clustering (step 4) now involves determining the threshold along the "feature line" that separates road from nonroad clusters, discussed in Sections III-B-1-c and III-B-1-d. Segmentation (step 5) is just the creation of a binary road/nonroad image (since $K = 2$ for simple road-following) from the clustering information, labeling each pixel as *road* or *nonroad* according to whether it projects above or below the feature threshold.

b) Color Parameter Selection: The projection of image points in RGB space onto a line is equivalent to taking the dot product of every pixel vector (R, G, B) with a vector in the direction of the line (r, g, b). This is accomplished by a *tricolor* operation, a linear weighted combination of the red, green, and blue images:

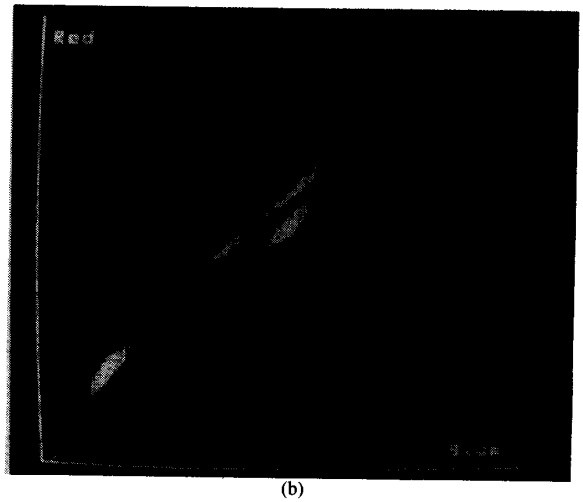
$$I(i,j) = rR(i,j) + gG(i,j) + bB(i,j)$$
$$= (r, g, b) \cdot (R, G, B).$$

The vector (r, g, b) represents the red, green, and blue coefficients of the tricolor operation. When this vector is normal to the plane that separates road and nonroad clusters in the RGB space, the outcome I is the single band "feature enhanced" image. The orientation of the separating 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 thereafter. However, we have found that some segmentation failures have occurred because of a change in the orientation of the separating plane, due to changing weather conditions, seasonal changes, road surface variations, and the camera color response. We have therefore developed a method to dynamically compute the optimal plane orientation based on data from the image currently being processed.

Experience has shown that a good segmentation is achievable without using the green band, 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. (The techniques presented are easily generalized to include the green band, however.) Fig. 7(b) shows a *scatter diagram* of the red and blue components of Fig. 7(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 nonroad pixels, and it should be clear that the line drawn in the figure will successfully separate road and nonroad clusters and therefore segment the image. This line is the *linear discriminant function* in Red/Blue space.



(a)



(b)

Fig. 7. Road image. (a) Original. (b) Red/blue scatter diagram of image. Line in (b) depicts road/nonroad boundry.

The slope of the linear discriminant function determines the red and blue components of the tricolor operation, with the green component equal to zero. In the scatter diagram, the road cluster is consistently an elliptical shape whose principal axis is parallel to the linear discriminant function. Hence the discriminant function can be found by calculating the principal axis of the road clus-

ter. The angle the principal axis makes with respect to the red axis (θ) is defined by the equation [17]

$$\theta = 0.5 \tan^{-1}\left(\frac{b}{a-c}\right) \quad (1)$$

where

$$a = \sum_i (r - \bar{r})^2$$

$$b = 2\sum_i (r - \bar{r})(b - \bar{b})$$

$$c = \sum_i (b - \bar{b})^2$$

and $(\bar{r}, \bar{b})$ is the mean of the cluster. From θ the red, green, and blue color parameters are calculated as

$$(r, g, b) = (\cos\theta, 0, \sin\theta). \quad (2)$$

The method to dynamically choose color parameters, then, proceeds as follows: sample road points in the image, calculating the red and blue means $(\bar{r}, \bar{b})$ and a, b, and c which lead to the calculation of θ 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 nonroad clusters. This automatic color parameter selection technique is yet to be fully integrated into VITS—the current version still uses preset constant color parameters.

c) Threshold Selection: Once the color parameters 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 $rR + gG + bB = 0$. At this point, the image is blurred to reduce noise. Choosing a value with which to threshold the new image, then, is equivalent to translating the plane in RGB space. The segmentation is described by the following equation which makes explicit the use of both color parameters (r, g, b) and the threshold (λ):

$$I'(i,j) = \begin{cases} 1 & \text{if } rR(i,j) + gG(i,j) \\ & \quad + bB(i,j) + \lambda < 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

The resulting binary image I' is a function of the threshold λ, which is selected by sampling a population of road pixels from the current image. 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. This worked well over most of the initial test track, but is not generally true. As an alternative, we opted for a more robust and faster method of calculating the threshold by sampling road pixels in each image. The road sampling technique is described in the next section.

Our original threshold calculation involved finding the mean and standard deviation of the road samples; the 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. (See Section III-B-4 for a proposed solution to this problem.) 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). This normally prevents small shadows, cracks and potholes, and patches on the road from causing erroneous road sampling. The most positive sampled road points define the approximate boundary of the road cluster along the feature line. The median of the top values is used in case spurious nonroad values are sampled, and a constant offset is added to the nominal value to move the threshold just above the cluster boundary. The threshold calculation is critical to a good segmentation; Fig. 8 shows the results of thresholding at different values.

d) 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 Fig. 9(a), assuming that the road covered these points. Because the road can drastically change 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, then, *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 movement, 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 (described in Section II) to do the geometric calculations between the previous and the present vehicle positions. Fig. 9(b) shows the sampling window computed as Alvin travels around a curve.

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. On a straight road the vehicle will often oscillate slightly as it traverses its path, so rather than allowing many small panning motions the pan mechanism is disabled and power windowing takes care of

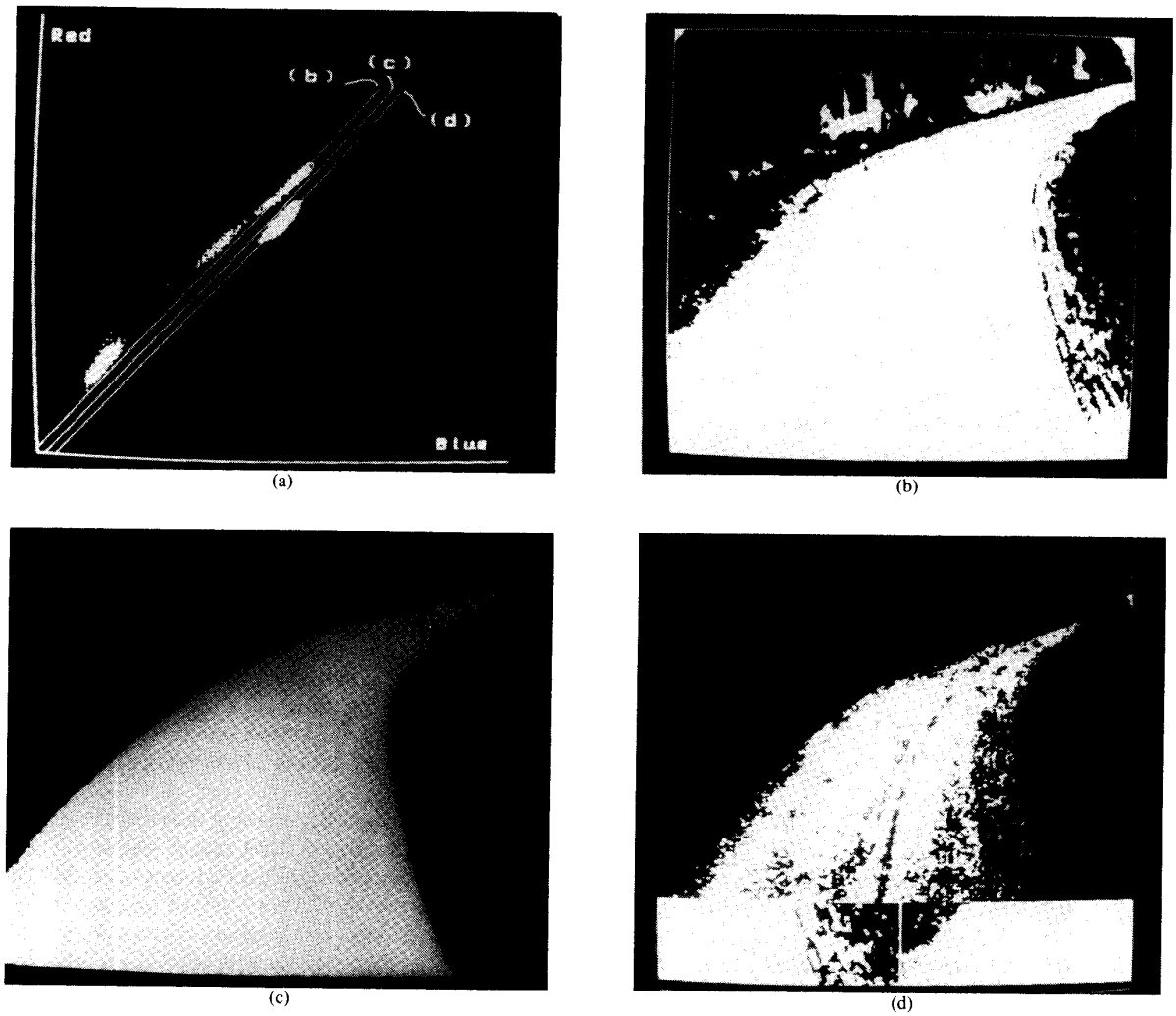


Fig. 8. (a) Scattergram. (b)–(d) Result of thresholding at different values.

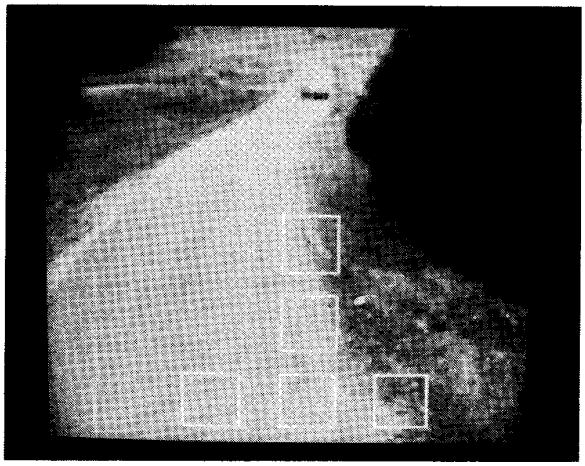
"panning" the sample window. In extreme conditions such as a bad segmentation suggesting a window that is unreasonably small or large, a default window is used, similar to the old fixed image sampling points.

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. The extension of the window also allows us to sample on shadowed pavement, which is helpful in the *shadow boxing* algorithm (Section III-B-4).

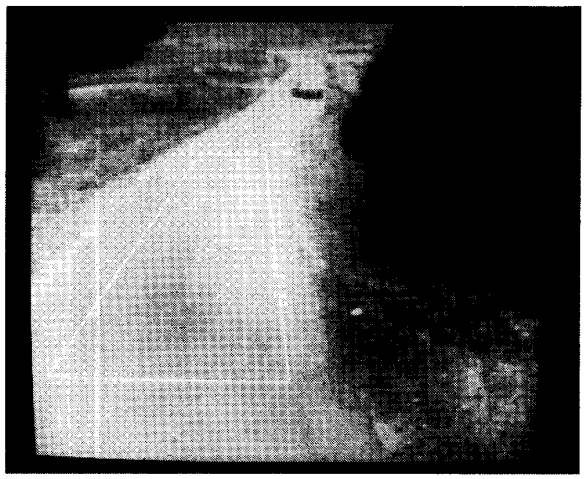
2) Red Minus Blue Segmentation: The segmentation algorithms are implemented on the Vicom image processing computer and take advantage of its frame rate convolution and lookup table operations. The feature reduction is accomplished by a tricolor operation, a weighted sum of the red, green, and blue images. These weights are chosen by hand or more recently calculated based on road image statistics (Section III-B-1-b). Typical values for (r, g, b) are $(0.5, 0.0, -0.5)$; hence the name "Red minus Blue." In practice, for the Martin Marietta test site these values are nearly constant. The dynamic parameter selection described above makes it possible to handle changes in road surface type, for example, during a run.

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/nonroad boundary. (Actually, "Blue minus Red" enhances the pavement, while "Red minus Blue" enhances the dirt!) An edge-based road-following algorithm would best work on the enhanced image rather than a normal intensity or single-band image.

A threshold value is chosen from the road statistics to differentiate road and nonroad clusters, as discussed in



(a)



(b)

Fig. 9. (a) Original fixed sampling points. (b) Sampling window around sharp curve, calculated by power windowing module.

Section III-B-1-c. The resulting image is then thresholded to produce the binary road/nonroad image; this is simply one pass of the image through a lookup table. The steps of the algorithm are depicted in Fig. 10 for two different road scenes.

3) Color Normalization: Another clustering algorithm involves segmenting a color normalized image, rather than the "Red minus Blue" feature image. Color intensities vary quite a bit within the road surface in a road image with shadows; intensities from the shaded regions are much smaller than those from 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.* [12] 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/nonroad segmentation. The calculation of this feature and the resulting segmentation is described by

$$I'(i,j) = \begin{cases} 1 & \text{if } \dfrac{B}{R+G+B} - \lambda > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The threshold equation of (4) can be rewritten as

$$B - \lambda(R + G + B) > 0$$

or

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

This can also be implemented as a tricolor operation followed by a threshold, as described by the equation:

$$I'(i,j) = \begin{cases} 1 & \text{if } \lambda R(i,j) + \lambda G(i,j) \\ & \quad + (1-\lambda)B(i,j) < 0 \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Equation (5) 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 λ proceeds as described in Section III-B-1-c.

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 nonroad regions inside of the half-space defined by the plane normal (r, g, b) and the threshold λ, they will also be labeled as "road" and perhaps cause faulty scene models to be output. Fig. 11 shows a scatter diagram of such a case: a large region labeled "shaded nonroad," caused primarily by ditches and shadows of bushes 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 nonroad."

This could perhaps be solved by segmenting twice, corresponding to the two threshold lines in Fig. 11(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. Another technique that we have considered is to use a nonlinear discriminant function for segmentation. The problem is to find an adequate nonlinear function that reliably corresponds to the bends observed in the boundaries of road features in Red/Blue color space. On inspection of a wide variety of road scenes, no general pattern could be observed.

Rather than segmenting complete half-spaces, then, the road regions may be bounded by rectangles in the scatter

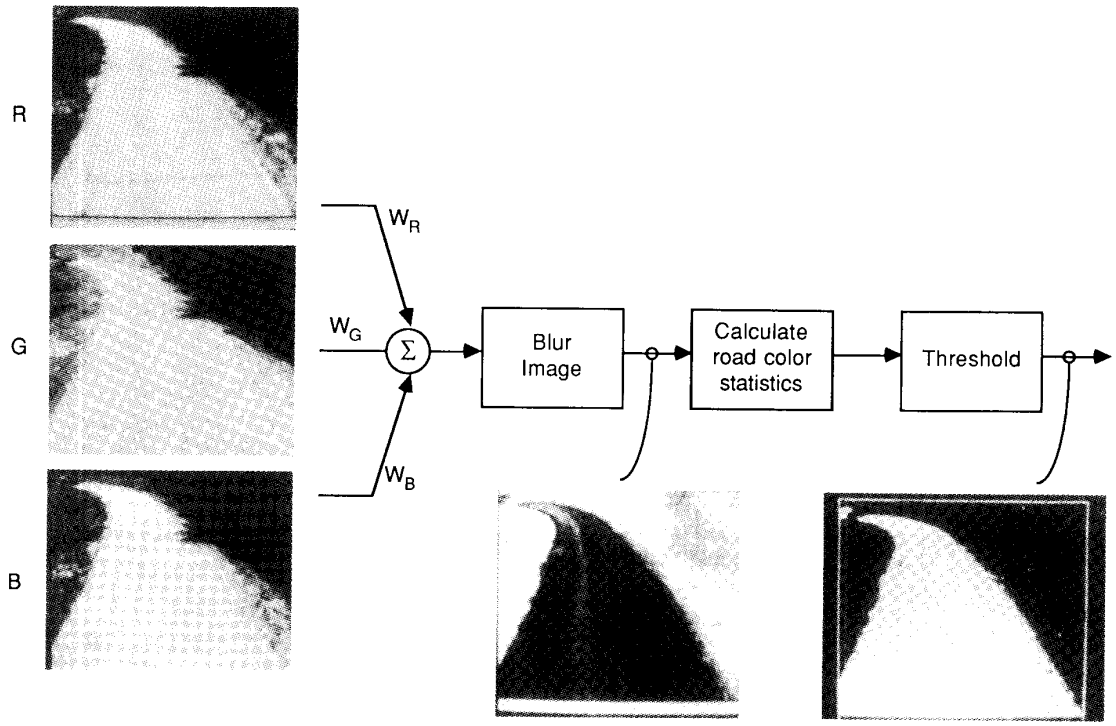
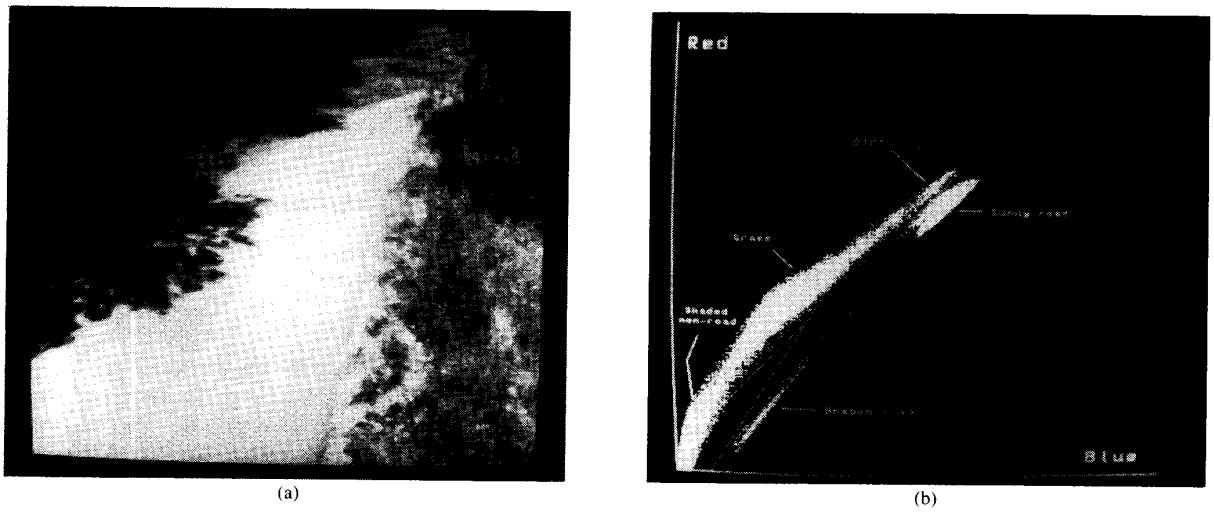


Fig. 10. Incremental results from video segmentation algorithm.



(a) (b)

Fig. 11. (a) Original image. (b) Scatter diagram of road scene with shadows.

diagram, 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."

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. The boxes must be oriented with the axes of the two dimensional feature space, however, so the first step is to perform a rotation of the Red/Blue axes so that the red axis is aligned with the road cluster in the scatter diagram. A rotation of θ about the origin is desired, where θ is defined in (1) as the angle between the red axis and the principal axis of the road cluster in the scatter diagram. As shown in the Appendix, this rotation is performed by replacing the red image with the result of a tricolor operation using $(r, 0, b)$ and replacing the blue image with a tricolor result using $(-b, 0, r)$, where r and b are defined in (2). Fig. 12(a) shows the new scatter diagram with the bounding boxes, from the original in Fig. 11.

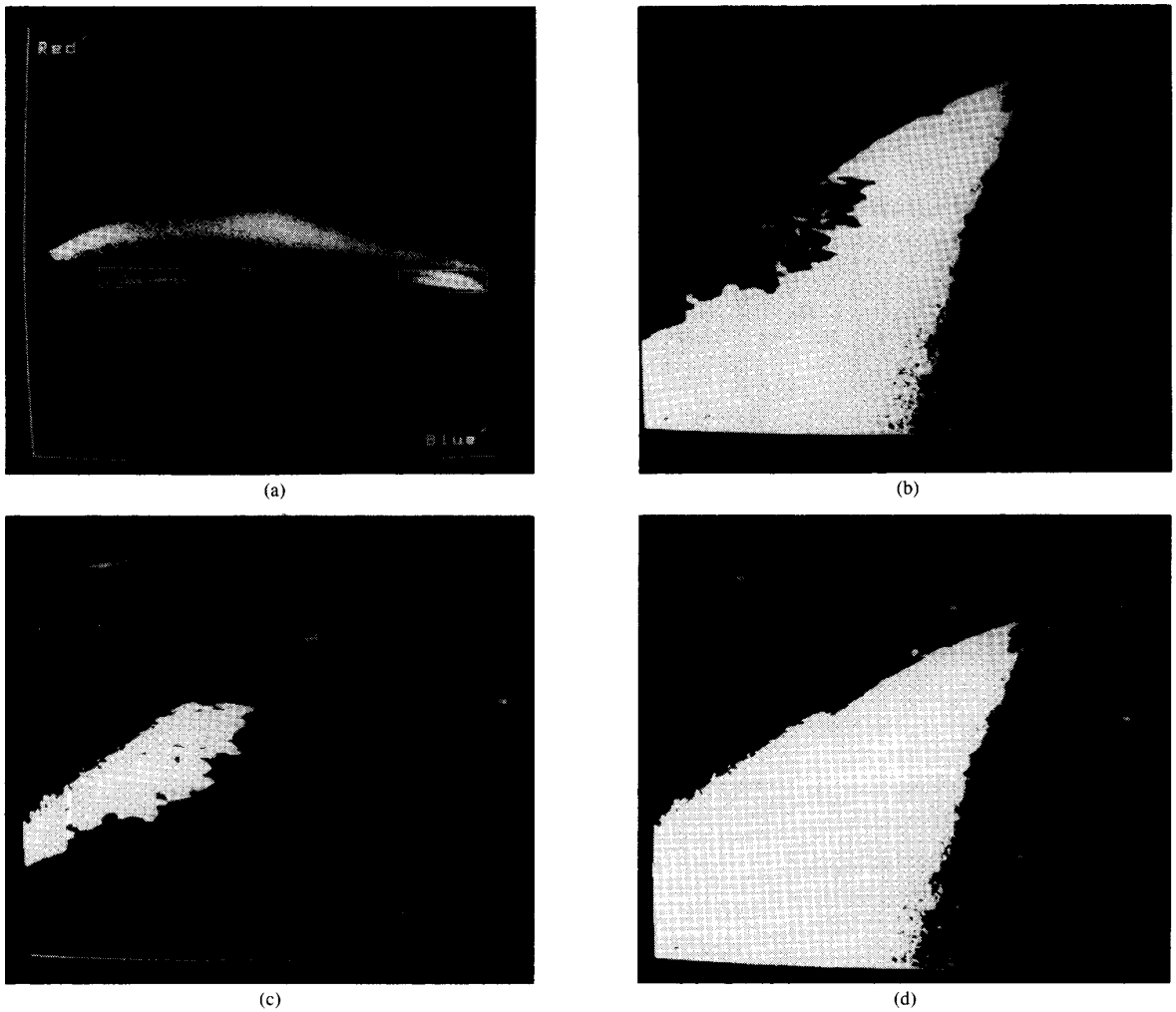


Fig. 12. Scatter diagram from Fig. 11, (a) rotated to align with the road cluster. (b) Sunny road result. (c) Shaded road result. (d) Final road image.

The extents of the bounding boxes in the rotated space are calculated as the road image points are sampled by keeping track of the maximum and minimum sampled values within expected ranges for both shaded road and sunny road regions. The threshold calculation uses only values in the expected range for sampling the road, making that stage less sensitive to noise. Lookup tables are built to perform the segmentation; two binary images are produced from each bounding box, one per axis. For each box, a logical AND of its resulting binary images is performed, giving the road region corresponding to the box. Figs. 12(b) and (c) show the image regions corresponding to the sunny road and shaded road boxes, respectively. The results from each box are then combined with a logical OR operation, resulting in a binary road/nonroad image, as in Fig. 12(d).

Shadow boxing is similar to a dynamic Bayesian classifier [9] with three decision regions and rectangular decision boundaries. In summary, the steps involved in the algorithm are: 1) calculate the color parameters (r, g, b); 2) rotate the Red/Blue space by performing two tricolor operations; 3) sample the road points, keeping track of the extents of the bounding boxes; 4) build the lookup tables; and 5) perform the segmentation for each box by passing the rotated images through the corresponding lookup tables and combining the images with the proper sequence of logical operations.

C. Boundary Extraction

The segmentation algorithms produce a binary road/nonroad 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 pixel locations.

The initial task is to find the road/nonroad boundary in the image. To facilitate easier boundary tracing and to

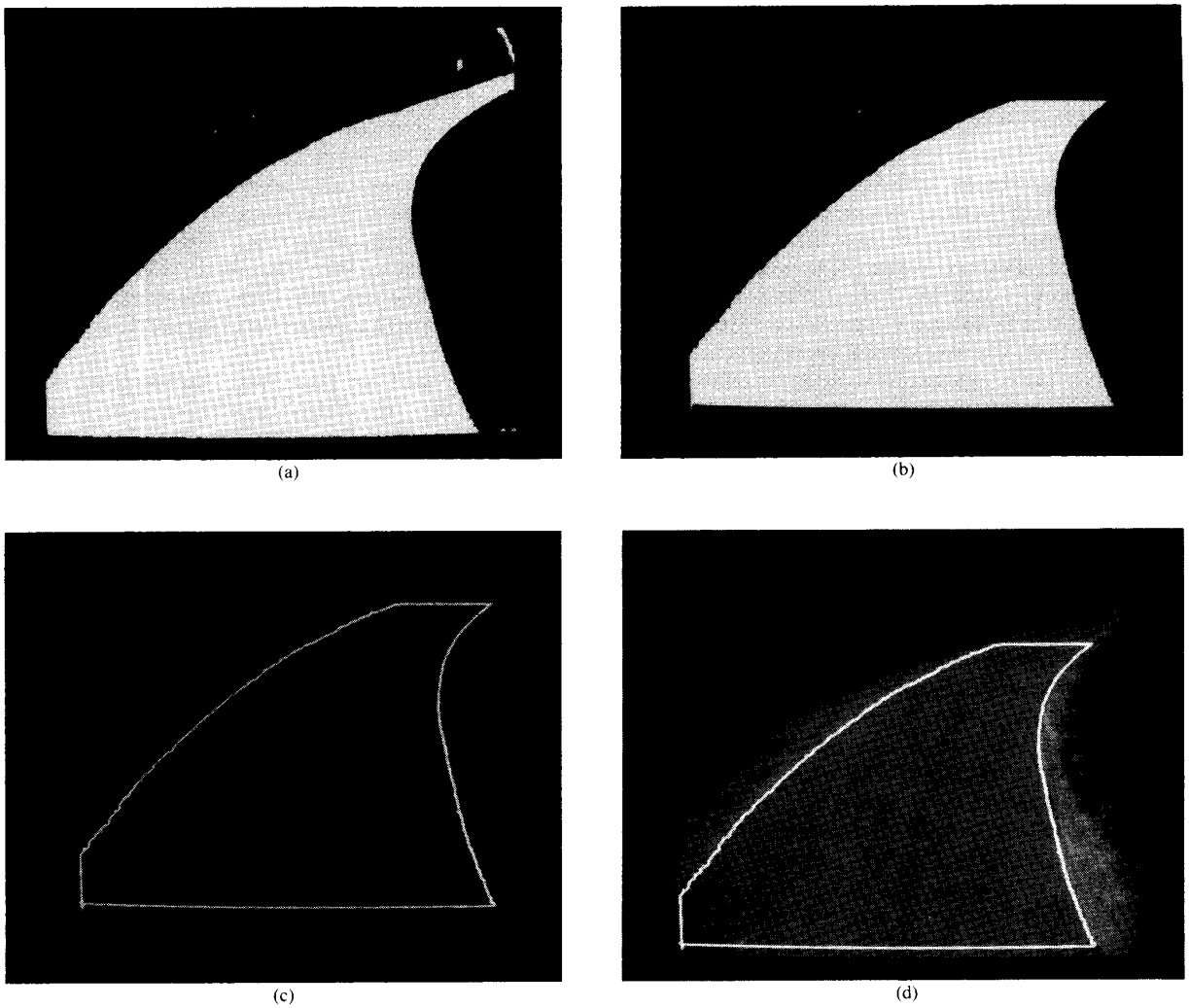


Fig. 13. (a) Binary road image. (b) False boundary added. (c) Road edges. (d) Road edges overlaid on original image.

avoid looking for the road on or above the horizon, a false road boundary is added around the image, creating an artificial horizon and borders, as in Fig. 13(b)—this prevents following the road edges up into the sky or over into the next segment of image memory. 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 nonroad, 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 nonlinear 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. Fig. 13(c) and (d) show the boundary traced for the segmentation of Fig. 7(a).

A completely general approach to boundary extraction is prohibitively expensive with the current hardware. Such an approach, however, would allow more extrensive reasoning about road shape, obstacles in the road, and choosing road sampling regions. Kuan and Sharma [21] have demonstrated model-based reasoning about road boundaries. Future hardware improvements should allow more sophisticated reasoning capability in road boundary extraction.

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 form a polygonal representation of the road edge. Image coordinates of a small neighborhood of edge points are averaged to avoid sending "stray" points. The row locations of these points are spaced by a quadratic function so that the three dimensional locations of the road edge points will be approxi-

mately equal distances apart. These points are then sent to the geometry module for conversion to three dimensional road edge point.

D. Three Dimensional Geometry Transformations

Once road edge points are selected in the image, a three-dimensional description, called 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 [31] 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 road environment. This is the *forward-geometry* problem.

VITS also uses the solution to the *inverse-geometry* problem, determining the location within the image plane 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 [40], 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 better for the roads Alvin encounters and the speeds attained during the demonstrations through 1986. 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 [28] that uses a fast "shape from contour" method to solve the forward-geometry problem.

These various techniques for recovering the three dimensional descriptions are discussed in the remainder of this Section. The flat-earth model is presented first because it is used by the other techniques. Next, the hill-and-dale technique, a slight modification of the flat-earth model, is presented. The *vanishing point* method and a *zero-bank* method, both developed at the University of Maryland, are mentioned for comparison.

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 in the image 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 the ground plane.

The flat-earth geometry model has several advantages over the other forward-geometry models. The 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. This property also implies that this algorithm is the only one of the four discussed here which is presently capable of handling intersections. 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 the vehicle body rocks backwards, decreasing the effective tilt angle of the camera. When traveling downhill the vehicle body rocks forwards, increasing the effective tilt angle of the camera. These changes in the effective tilt angle cause parallel road edges to be output as converging or diverging three dimensional edge segments. If the convergence or divergence is too severe it is difficult to connect road edges from one scene model with those of the next scene model. 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 the vehicle undercarriage. Knowledge of this angle will enable a more accurate forward-geometry transformation.

A second problem with the flat-earth model occurs at inflection points, such as at crests of hills and at bottoms of valleys. These situations cause the description of the road to both converge and diverge within the same scene model. This problem is addressed by each of the geometry models described below.

Examples of both of these problems can be seen in Fig. 14; plots of successive scene models reveal herringbone patterns caused by a combination of a nonplanar road and inaccuracy of the camera tilt angle.

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, as illustrated in Fig. 14. 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 retain a constant road width.

Let $\boldsymbol{p}(i, j)$ be the world coordinates of the points which appear in the image as indicated in Fig. 15. The first step of the algorithm is to use flat-earth geometry to solve for $\boldsymbol{p}(0, 1)$ and $\boldsymbol{p}(0, 2)$. From this it is possible to compute the width of the road W, where $W = \| \boldsymbol{p}(0, 1) - \boldsymbol{p}(0, 2) \|$.

One way to maintain a constant road width in the scene

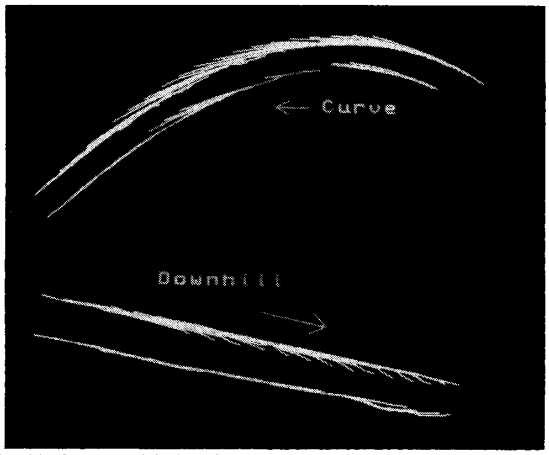


Fig. 14. Scene model plots from a test run/(em road edges viewed from above). Herringbone patterns are seen in the overlap of scene models.

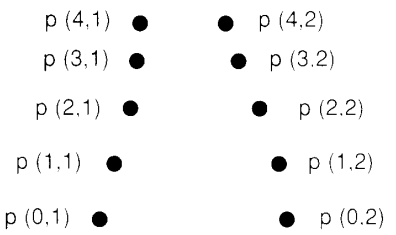


Fig. 15. Road edge points to be converted into the vehicle coordinate system.

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 ground plane will produce a narrower road than using a plane below the ground plane. 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 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, and when the segmentation is good. The algorithm is, however, very dependent upon 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 travel downhill. While this is not particularly important to vehicle behavior when the road is straight, it can cause the distance to a curve to be in error by a significant amount.

A potentially larger drawback to this algorithm is its performance 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. This problem is potentially addressed by the vanishing-point algorithm described below. 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 assumption 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. This is the "tiling problem" of road geometry.

3) Other Geometry Models: The vanishing point geometry model [40] uses flat-earth geometry to determine the location of the nearest visible right and left edge points within the image, and allows for the road to slope uphill or downhill relative to the vehicle ground plane. This algorithm uses constraints based on assumptions of parallel road edge segments, continuity, and local flatness of the road. These assumptions allow the computation of a set of tiles that approximate the road in a viewer-centered coordinate frame. The technique involves solving for the image coordinates of the vanishing point of each pair of matched left and right road edge segments within each tile. This geometry model was used in the initial May 1985 ALV demonstration.

The zero-bank algorithm [6] models the road as a centerline spine and horizontal line segments cutting the spine at their midpoint at a normal to the spine. Modeling the road in this way constrains tiles of the road to be "warped isosceles trapezoids." As in the other algorithms, the three-dimensional locations of the closest point on each edge are found using the flat earth algorithm. In each successive step the three-dimensional location of the next point on one side of the road is parameterized by its distance from the camera. The image location of the paired point on the other side of the road is constrained by the warped isosceles trapezoid as well as the edge curve within the image plane. This leads to a cubic equation in the parameter; when the roots of this equation (if any) are found, the root which yields minimum reasonable slope difference with respect to the previous road direction is kept.

While the zero-bank algorithm is computationally the most costly, it addresses both the tiling problem and the problem of hills and dales. None of the algorithms can reproduce banking in curves, and all but the flat-earth algorithm rely heavily on good segmentation and cannot handle intersections.

IV. Discussion

The ALV public demonstrations in 1985 and 1986 have displayed Alvin's road-following capabilities. Fig. 16 shows a schematic map of the current Martin Marietta test track. In May of 1985, Alvin traversed from points 2 to

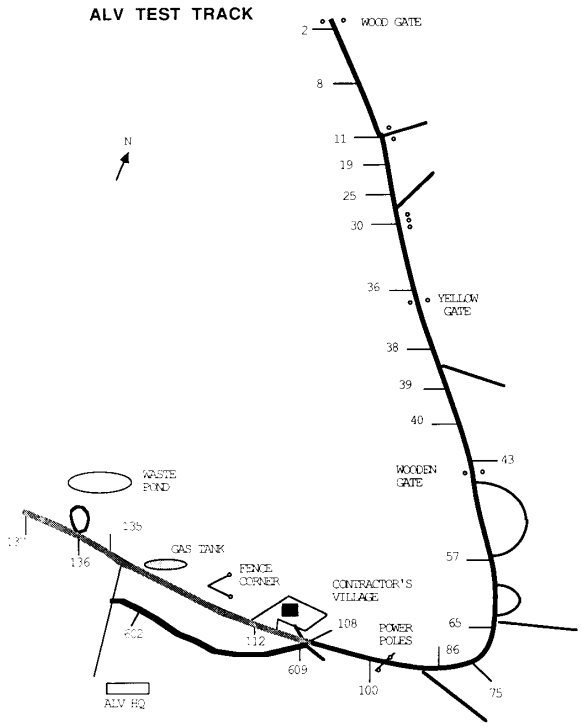


Fig. 16. Martin Marietta ALV Test Track (Denver, CO).

65 (a distance of approximately 1 km) at a speed of 3 km/hr. In June of 1986, Alvin traversed the entire test track (4.2 km) at speeds up to 10 km/hr. Additional capabilities demonstrated in 1986 include:

1) switching back and forth between video-based and range-based road-following
2) obstacle avoidance based on a fusion of video and range data
3) high-speed runs of up to 20 km/hr
4) varying vehicle speed as a function of scene model length
5) slowing to a stop and turning completely around, then traveling back in the opposite direction
6) negotiating a hairpin curve
7) traveling over two different types (and colors) of pavement.

Fig. 17 illustrates some of the various conditions encountered on various portions of the test track. A typical test run over the entire track will involve the acquisition and processing of hundreds of images. These images vary from day to day, since weather, sun angle, shadows, and tire tracks all alter the appearance of the road and its surroundings. To compensate for occasional failures in VITS, the reasoning subsystem builds trajectories so that the vehicle will halt at the end of the most recent scene model if no additional data is received. This convention makes Alvin very forgiving of occasional vision failures, usually causing a slight slowing of the vehicle when the processing of a single image frame fails. During one test run, a cloud passed overhead and changed the road appearance enough so that no road was found in the images. The vehicle ramped down to a stop and waited until the cloud

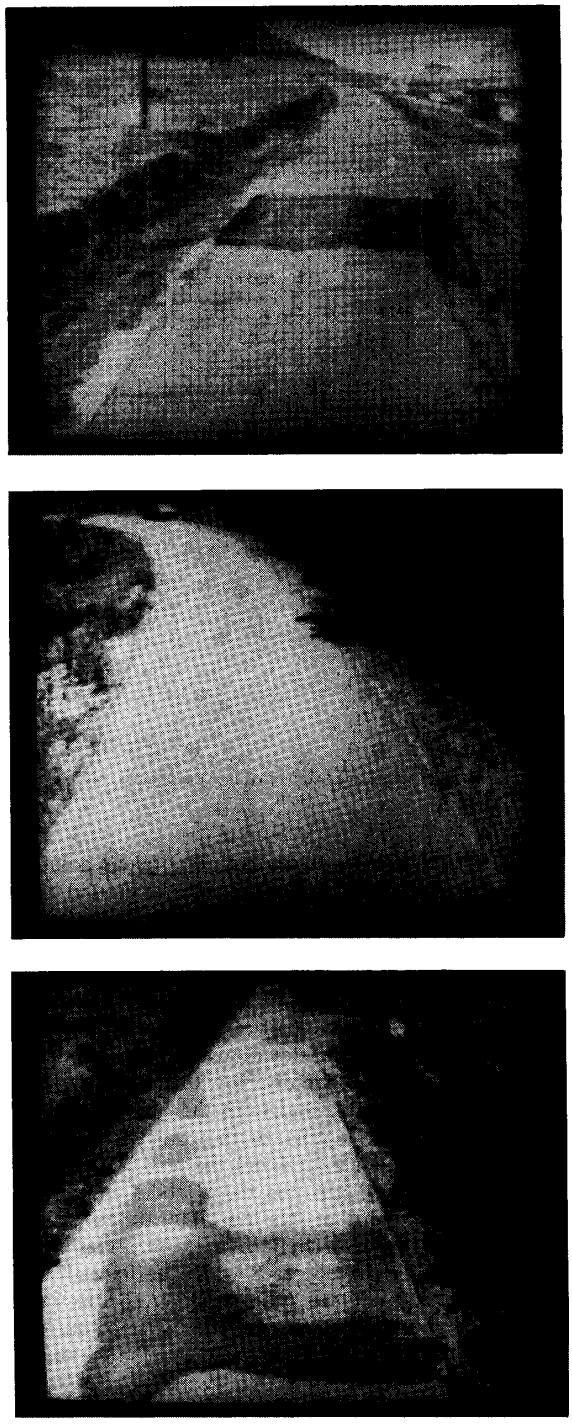


Fig. 17. Representative image of the test track.

passed by and VITS was able to segment the road; Alvin then resumed safe travel.

Current vision efforts are concentrated on speed and robustness of road-following, obstacle detection and location, and 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 produce scene models more rapidly but also provide longer scene models to allow for the distance needed to slow down for a detected obstacle or to stop in case of an emergency. We are presently investigating methods to more accurately model the road at far distances (25 meters and beyond) 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 [36].

To meet the demands of future demonstrations, implementation of a new architecture begin in early 1987. Each subsystem of the ALV will have significantly more processing power available. For the vision subsystem, the second generation hardware consists of two Vicom image processors, a Sun 3/180, and a Warp machine [14]. We are also investigating the use of a FLIR sensor and special sensors for obstacle detection, and other advanced computer architectures (such as the Butterfly and the Connection Machine).

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 in a formal demonstration.

The evolution of the vision system has been largely motivated by failures, caused by deviations from "ideal" road-following conditions. Dirt and tire tracks on the road, unexpected tarmac patches, changing weather conditions (from a momentary cloud overhead to the presence of snow for weeks) and seasonal variations (such as increased shadows and spectral reflection caused by a lower sun angle in the winter months) have all caused failures of the vision system. In analyzing these failures, we have learned much about the vision/navigation interplay, sensor control needs, and the dynamic nature of video parameters.

The Autonomous Land Vehicle is intended not only as a development project to meet specified demonstration requirements, but also as a national testbed for research in image understanding, AI planning, and advanced architectures. As different parts of the system change, we anticipate learning more about how system interaction affects the individual components. A vision system for a mobile robot does not stand alone; we have spent much effort analyzing the vision/navigation interaction to discover the causes of certain vehicle behaviors or failures. Likewise, the algorithm level is not totally independent of the implementation or hardware level. Much of the current vision system has been motivated by the chosen hardware—as the hardware changes, the algorithms may change substantially.

V. Summary and Conclusion

The time is ripe for fruitful research and experimentation in mobile robotics. Early work suffered from a lack of processing power, but with the availability of relatively cheap, fast machines, particularly special-purpose image processing hardware and lots of memory, an important step in mobile robot research is now realizable: experimentation with fully autonomous, continuous motion, self-contained systems. The dynamics of system interaction provides important insight into the development of sophisticated autonomous systems.

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. Our experiments with Alvin have driven the development of such a vision framework. As mobility may be a key to the development of intelligence [27], real-time interaction with the environment is a significant step in the development of intelligent machines. 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.

Appendix
Red/Blue Axes Rotation

We want to show that the rotated coordinate system used for shadow boxing is accomplished by replacing the Red image with the tricolor image with parameters $(r, 0, b)$ and replacing the Blue image with the tricolor image with

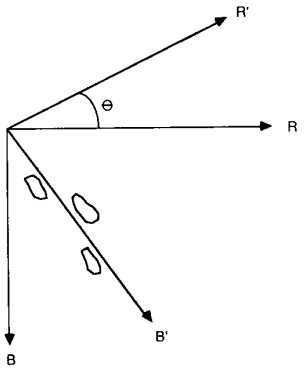


Fig. 18. (R, B) to (R', B') rotation for shadow boxing.

parameters $(-b, 0, r)$. The coordinate systems are shown in Fig. 18, where (R', B') is rotated to align with the major axis of the blob. The transformation is described by:

$$R' = R \sin\theta - B \cos\theta$$
$$= (R, G, B) \cdot (\sin\theta, 0, -\cos\theta)$$
$$B' = R \cos\theta + B \sin\theta$$
$$= (R, G, B) \cdot (\cos\theta, 0, \sin\theta)$$

From (2) in Section III-B-1-b, this is equivalent to

$$R' = (R, G, B) \cdot (r, 0, b)$$
$$B' = (R, G, B) \cdot (-b, 0, r)$$

which describes a pair of tricolor operations. In practice, a *translation* of the coordinate system may also be necessary to avoid negative values.

Acknowledgment

T. Dunlay, S. Hennessy, B. Bloom, J. Bradstreet, Z. Shinno, and G. Arensdorf all contributed to the development of the vision system. The support of the whole ALV team is greatly appreciated.

References

[1] S. Barnard, R. Bolles, D. Marimont, and A. Pentland, "Multiple representations for mobile robot vision," in *Proc. SPIE Mobile Robot Conf.*, Cambridge, MA, Oct. 1986, pp. 143-151.

[2] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robotics and Automation*, vol. RA-2, no. 1, pp. 14-23, Mar. 1986.

[3] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," *Proc. IEEE*, vol. 67, no. 5, pp. 872-884, May 1979.

[4] J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE J. Robotics and Automation*, vol. RA-1, no. 1, pp. 31-41, Mar. 1985.

[5] M. J. Daily, J. G. Harris, and K. Reiser, "Detecting obstacles in range imagery," in *Proc. DARPA Image Understanding Workshop*, Feb. 1987, pp. 87-97.

[6] D. DeMenthon, "A Zero-bank algorithm for inverse perspective of a road from a single image," in *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, Apr. 1987, pp. 1444-1449.

[7] A. R. de Saint Vincent, "A 3D perception system for the mobile robot Hilare," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 1986, pp. 1105-1111.

[8] E. D. Dickmanns and A. Zapp, "A curvature-based scheme for improving road vehicle guidance by computer vision," in *Proc. SPIE Mobile Robot Conf.*, Cambridge, MA, Oct. 1986, pp. 161-168.

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

[10] R. T. Dunlay and D. G. Morgenthaler, "Robot road-following using laser-based range imagery," in *Trans. SME Second World Conf. Robotics Research*, Scottsdale, AZ, Aug. 1986.

[11] R. T. Dunlay and D. G. Morgenthaler, "Obstacle detection on roadways from range data," in *Proc. SPIE Mobile Robot Conf.*, Cambridge, MA, Oct. 1986, pp. 110-116.

[12] R. Gershon, A. D. Jepson, and J. K. Tsotsos, "The effects of ambient illumination on the structure of shadows in chromatic images," Dep. Comput. Sci., Univ. Toronto, Tech. Rep. RBCV-TR-86-9, 1986.

[13] Y. Goto and A. Stentz, "The CMU system for mobile robot navigation," in *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, Apr. 1987, pp. 99-105.

[14] T. Gross, H. T. Kung, M. Lam, and J. Webb, "Warp as a machine for low-level vision," in *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, MO, Mar. 1985, pp. 790-800.

[15] A. R. Hanson and E. M. Riseman, "Segmentation of natural scenes," in *Computer Vision Systems*, A. R. Hansen, and E. M. Riseman, Eds. New York: Academic, 1978.

[16] M. Hebert and T. Kanade, "Outdoor scene analysis using range data," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 1986, pp. 1426-1432.

[17] B. K. P. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.

[18] R. M. Inigo, E. S. McVey, B. J. Berger, and M. J. Mirtz, "Machine vision applied to vehicle guidance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 6, pp. 820-826, Nov. 1984.

[19] M. Julliere, L. Marce, and H. Place, "A guidance system for a mobile robot," in *Proc. 13th Int. Symp. Indust. Robots and Robots7*, Chicago, IL, Apr. 1983.

[20] D. Kuan, G. Phipps, and A. Hsueh, "A real-time road following and road junction detection vision system for autonomous vehicles," in *Proc. AAAI-86*, Philadelphia, PA, Aug. 1986.

[21] D. Kuan and U. K. Sharma, "Model based geometric reasoning for autonomous road following," in *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, Apr. 1987, pp. 416-423.

[22] K. Kuhnert, "A vision system for real time road and object recognition for vehicle guidance," in *Proc. SPIE Mobile Robot Conf.*, Cambridge, MA, Oct. 1986, pp. 267-272.

[23] D. T. Lawton, T. S. Levitt, C. McConnell, and J. Glicksman, "Terrain models for an autonomous land vehicle," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 1986, pp. 2043-2051.

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

[25] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford Univ., Stanford, CA, Sept. 1980. (Reprinted as *Robot Rover Visual Navigation*. Ann Arbor, MI: UMI Research Press, 1981.)

[26] ——, "The Stanford Cart and the CMU Rover," *Proc. IEEE*, vol. 71, no. 7, pp. 872-884, July 1983.

[27] ——, *Mind Children*. Cambridge, MA: Harvard University Press, 1986.

[28] D. Morgenthaler, "Hill and dale geometry," Martin Marietta Internal Memo, May 1986.

[29] B. Mysliwetz and E. D. Dickmanns, "A vision system with active gaze control for real-time interpretation of well structures dynamic scenes," in *Proc. Intelligent Autonomous Systems*, Amsterdam, The Netherlands, Dec. 1986.

[30] K. E. Olin, F. M. Vilnrotter, M. J. Daily, and K. Reiser, "Developments in knowledge-based vision for obstacle detection and avoidance," in *Proc. DARPA Image Understanding Workshop*, Feb. 1987, pp. 78-86.

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

[32] A. M. Thompson, "The navigation system of the JPL robot," in *Proc. Fifth IJCAI*, 1977, pp. 749-757.

[33] C. Thorpe, "FIDO: Vision and navigation for a mobile robot," Ph.D. dissertation, Dep. Comput. Sci., Carnegie-Mellon Univ., Dec. 1984.

[34] S. Tsugawa, T. Yatabe, T. Hirose, and S. Matsumoto, "An automobile with artificial intelligence," in *Proc. Sixth IJCAI*, 1979, pp. 893-895.

[35] S. Tsuji, J. Y. Zheng, and M. Asada, "Stereo vision of a mobile robot: World constraints for image matching and interpretation," in

Proc. IEEE Int. Conf. Robotics and Automation, San Francisco, CA, Apr. 1986, pp. 1594–1599.

[36] M. A. Turk and M. Marra, "Color road segmentation and video obstacle detection," in *Proc. SPIE Mobile Robot Conf.*, Cambridge, MA, Oct. 1986, pp. 136–142.

[37] R. S. Wallace, "Robot road following by adaptive color classification and shape tracking," in *Proc. AAAI-86.*

[38] R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, and T. Kanade, "First results in robot road-following," in *Proc. IJCAI-85.*

[39] R. Wallace, K. Matsuzaki, J. Crisman, Y. Goto, J. Webb, and T. Kanade, "Progress in robot road-following," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, Apr. 1986, pp. 1426–1432.

[40] A. M. Waxman, J. LeMoigne, and B. Scrinivasan, "Visual navigation of roadways," in *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, MO, Mar. 1985.

[41] A. M. Waxman, J. J. LeMoigne, L. S. Davis, B. Srinivasan, T. R. Kushner, E. Liang, and T. Siddalingaiah, "A visual navigation system for autonomous land vehicles," *IEEE J. Robotics and Automation*, vol. RA-3, no. 2, pp. 124–141, Apr. 1987.

Matthew A. Turk (S'81–M'84) received the B.S. degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, in 1982 and the M.S. degree in electrical and computer engineering from Carnegie-Mellon University, Pittsburgh, PA, in 1984.

In 1984 he joined the Advanced Automatic Technology Section at Martin Marietta Denver Aerospace, Denver, CO, where he was involved in computer vision research and development for autonomous vehicles and aerial photointerpretation. He is currently pursuing the Ph.D. degree in computational vision, affiliated with the Media Laboratory at the Massachusetts Institute of Technology, Cambridge.

David G. Morgenthaler received the B.S. degree in electrical engineering from Princeton University, Princeton, NJ, in 1975, and the M.S. and Ph.D. degrees in computer science from the University of Maryland in 1978 and 1981, respectively. His dissertation was on three dimensional image processing.

He joined Martin Marietta Denver Aerospace, Denver, CO, in 1982 as the Computer Vision Unit Head. He has led the computer vision research and development activities on several robotic related programs. He is currently the Software Lead and the Perception Lead on the Autonomous Land Vehicle program.

Keith D. Gremban received the B.S. degree in mathematics and the M.S. degree in applied mathematics from Michigan State University, East Lansing, in 1978 and 1980, respectively. He is currently working toward the Ph.D. degree in computer science and robotics at Carnegie-Mellon University, Pittsburgh, PA.

Since 1980, he has been with the Advanced Automation Technology Section of Martin Marietta, and is currently a visiting scientist at the Robotics Institute, Carnegie-Mellon University. His primary research interests are in the perception and interpretation of natural scenes, and mobile robot navigation. He has worked on several mobile robot projects, including the NAVLAB at CMU, and the ALV at Martin Marietta.

Martin Marra received the B.S. degree in applied math: engineering systems and computer science from Carnegie-Mellon University, Pittsburgh, PA, in 1985.

He worked for the Robotics Institute at CMU between 1983 and 1985 while pursuing his degree. In 1985 he joined the Advanced Automation Technology Section at Martin Marietta Denver Aerospace, Denver, CO, where he was involved in computer vision research and development for the Autonomous Land Vehicle Program.