Range-Video Fusion and Comparison of Inverse Perspective Algorithms in Static Images

DAVID G. MORGENTHALER, STEVEN J. HENNESSY, MEMBER, IEEE, AND DANIEL DEMENTHON

Abstract—**Creating three-dimensional (3-D) descriptions of road boundaries from single two-dimensional images (2-D) of segmented road is a central problem for the autonomous land vehicle (ALV) road-following task. In the past, the computational cost of this 2-D to 3-D inverse perspective problem has been minimized with simplifying assumptions to allow increased vehicle speeds. The paper in part describes how heuristic assumptions may be avoided if actively scanned laser range data are used to determine the 3-space location of features in a 2-D image. An "epipolar plane" approach, commonly used to restrict search spaces in stereo correspondence problems, is used to combine data gathered from the ALV's video camera and an ERIM laser range scanner for accurate 3-D descriptions of roads. An additional objective of this work is to compare various heuristic inverse perspective algorithms with each other and the video/range scanner "fusion" approach, in terms of accuracy, failure situations, and estimated computational speed. The simplest, and earliest used, "flat-earth" algorithm made the assumption that the vehicle was at all times resting in a level attitude on an infinite flat plane. Image points were projected onto this plane. A second approach, the "hill-and-dale" algorithm, permits the plane's elevation to vary such that image pairs of left and right road edge points (derived in a simple way after road segmentation) define a road of constant width. The "zero-bank" algorithm imposes a constant road width constraint to heuristically extracted image pairs of road edge points. The algorithm seeks image pairs of edge points that correspond to true "crossroad" road edges on curving roads. Graphical output describing the performance of the various algorithms on real world scenes will be presented and discussed.**

I. INTRODUCTION

THE AUTONOMOUS LAND VEHICLE (ALV) project was part of DARPA's Strategic Computing Program. During the first three years of this program, a series of successively more ambitious demonstrations was intended to advance and demonstrate the state of the art in image understanding, artificial intelligence, advanced architectures, and autonomous navigation. An overview of the ALV system is given in [1].

Two imaging sensors were available on the ALV, a standard color video CCD camera and a laser range scanner [2] developed by the Environmental Research Institute of Michigan (ERIM). The video camera provides RGB imagery to a process that uses color segmentation techniques [1] to produce an image model of road edge points in row and column camera image plane coordinates. These image locations must then be processed by an inverse perspective algorithm into 3-D coordinates and passed to a navigation system, which then plans and executes a vehicle trajectory down the extracted road. The laser range sensor has been used in systems designed to detect and avoid onroad obstacles [3] and to characterize terrain for offroad navigation [4], [5].

The ALV road following system design has in large part been driven by speed requirements—vehicle speed increased from 5 km/h in 1985 to 10 km/h in 1986 to 20 km/h in 1987. One consideration on the design of the ALV perception system is the speed versus accuracy trade-off of the inverse perspective algorithms. Any heuristic 2-D to 3-D algorithm produces errors when underlying assumptions are violated. The heuristic algorithms used on the ALV always assume good segmentation of the roadway, and further assume some invariant or constraint of the geometry of the road: the road is horizontal and planar [1], the road is of constant width [6], the road has no banking [7], [8], etc. Our experience with these algorithms is that when the assumptions are not grossly violated they tend to produce reasonable behavior near the vehicle, but that errors grow further out from the vehicle. The penalty of operating in inaccurate, more distant, portions of the scene is instability from the navigator, forcing reduced speeds. Additionally, the speed of the algorithm is affected by its complexity, so that even if greater accuracy is supplied, and the "error grows with distance" assumption is true, a faster, less accurate algorithm may provide for higher vehicle speeds. For example, the 1987 ALV road-following system, which achieved the greatest speeds to date, used the "flat-earth" algorithm—the fastest, crudest inverse perspective algorithm —to select speed above accuracy.

A number of inverse perspective algorithms have been developed and several have been used to drive the ALV. The selection of one of these algorithms for use on the vehicle has been based solely on experience with the algorithms. The objectives of the work reported in this

Manuscript received April 1, 1989; revised January 10, 1990. This work was supported by the Defense Advanced Research Projects Agency and the U.S. Army Engineer Topographic Laboratories under contract number DCAC76-84-0005.

D. G. Morgenthaler and S. J. Hennessy are with Martin Marietta Information & Communications Systems, P.O. Box 1260, M.S. XL-4370, Denver, CO 80201-1260.

D. DeMenthon is with the Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, MD 20742.

IEEE Log Number 9037594.

0018-9472/90/1100-1301$01.00

paper are to develop an "exact" inverse perspective transformation using combined range and video data and to compare the performance of the "exact" and heuristic algorithms using real roadway data. It is hoped that this will provide the beginning of a rational basis for algorithm selection.

The algorithms used in this paper reconstruct the three dimensional profile of the road in front of the vehicle from static images up to the point where the road becomes hidden to the sensors. We believe that constructing the road over as large a distance as possible presents several advantages: the reconstructions from several vehicle positions overlap, and can thus be combined for added reliability; and estimations of turns can be made well in advance to enable speed adjustment. This static approach (of obtaining 3-D reconstructions) complements the dynamic approach of Dickmanns and Graefe [9], [10], in which one edge of the road is tracked between multiple monocular images. A Kalman filter uses this data and data of the steering control loop to provide fast and accurate but short range 3-D information.

The presentation of the "exact" fusion algorithm and the heuristic algorithms to be compared will require a description and notation for the various coordinate systems used on the ALV; this is done in Section II. In Section III we present the fusion algorithm. In the fusion algorithm, the three-dimensional locations of an (arbitrary) camera image point, the camera focal point, and the ERIM sensor focal point determine an epipolar plane. As in stereo vision algorithms, where an epipolar plane is used to constrain search for correspondence between two images, we use the epipolar plane to constrain search within the spherical ERIM coordinate system. Having constrained search within the ERIM image to an arc, the correspondence between video pixel and ERIM pixel is easily computed by using the range data of the ERIM pixel.

In Section IV we present the heuristic algorithms that we have chosen to compare, and in Section V we present and discuss qualitative results obtained from these algorithms.

II. Notation, Coordinate Systems, and Transformations

We begin by defining our notation, the various coordinate systems used on the vehicle, and the transformations between these coordinate systems.

A point is a position vector $\vec{p}=(x,y,z)$ specified with respect to some coordinate system. A coordinate system C is defined by its origin O_C and its axes X_C, Y_C, Z_C. Thus $\vec{p}_C=(x_C,y_C,z_C)$ is origined at O_C. Multiple points $\vec{p}_i$, $i=1,\cdots$, may be written as

$$\vec{p}_{i,C}=(x_{i,C},y_{i,C},z_{i,C})=(x,y,z)_{i,C}. \tag{1}$$

A position vector $\vec{p}$ is expressed in homogeneous coordinates as $\vec{p}=(x,y,z,1)^T$. The notation $\vec{p}$ will also be used for a position vector expressed in homogeneous vector form since no confusion arises (see below).

Angles are expressed by specifying an axis (e.g., X_C) about which a notation is performed, and a measure (e.g., α,β,γ), where a positive measure is found by the right-hand rule: the fingers of the right hand indicate the direction of a positive angle when grasping the axis with the thumb pointing in the direction of the axis. An example or an angle specification is (X_C,α).

Homogeneous translation matrices are specified as $T_{\vec{p}_C}$, where $\vec{p}_C$ is the vector specifying a displacement with respect to the C coordinate system. For example, $T_{\vec{p}_{O_V,W}}$ is a translation of the (origin O_V of the) coordinate system V with respect to the coordinate system W. Note that W is associated with the displacement vector $\vec{p}_{O_V,W}$ rather than with the homogeneous translation matrix. Then the positive vector $\vec{p}_W$ corresponding to a position vector $\vec{p}_V$ in the translated coordinate system is given by

$$\vec{p}_W=\boldsymbol{T}_{\vec{p}_{O_V,W}}\vec{p}_V. \tag{2}$$

Basic homogeneous rotation matrices are defined according to the axis about which a rotation occurs:

$$\boldsymbol{R}_{X,\alpha}=\begin{bmatrix}1&0&0&0\\0&\cos\alpha&-\sin\alpha&0\\0&\sin\alpha&\cos\alpha&0\\0&0&0&1\end{bmatrix} \tag{3}$$

$$\boldsymbol{R}_{Y,\beta}=\begin{bmatrix}\cos\beta&0&\sin\beta&0\\0&1&0&0\\-\sin\beta&0&\cos\beta&0\\0&0&0&1\end{bmatrix} \tag{4}$$

$$\boldsymbol{R}_{Z,\gamma}=\begin{bmatrix}\cos\gamma&-\sin\gamma&0&0\\\sin\gamma&\cos\gamma&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \tag{5}$$

where α,β,γ are the measures of the rotations about the X, Y, and Z axes. Homogeneous rotation matrices map vectors expressed in a rotated homogeneous coordinate system D to a reference coordinate system C:

$$\vec{p}_C=\boldsymbol{R}\vec{p}_D. \tag{6}$$

A. Coordinate Systems

The following coordinate systems, shown in Fig. 1 are used on the ALV:

1) A *world coordinate system* is defined by $O_WX_WY_WZ_W$, where Y_W is North, X_W is East, and Z_W is elevation. This coordinate system is the most global reference system used on the ALV.
2) *A vehicle coordinate system* is defined by $O_VX_VY_VZ_V$, where X_V is the direction of forward travel for the vehicle, Y_V is to the left of the vehicle, and Z_V is up with respect to the vehicle. The LNS of the vehicle provides measurements of the vehicle coordinate system's azimuth (ψ), pitch (γ), roll (η), and displacement ($\vec{p}_{O_V,W}$) with respect to the world

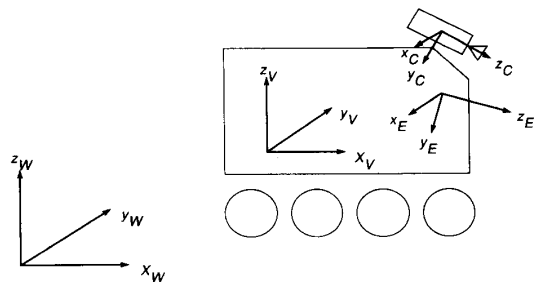


Fig. 1. ALV coordinate systems.

coordinate system. The matrix $M_{V,W}$ is used to transform a position vector $\vec{p}_V$ in vehicle coordinates to a position vector in world coordinates $\vec{p}_W$. It is computed by

$$M_{V,W} = T_{\vec{p}_{O_V,W}} R_{Z_V,\pi/2-\psi} R_{Y_V,\gamma} R_{X_V,\eta}. \quad (7)$$

The world position vector $\vec{p}_W$ corresponding to a vehicle position vector $\vec{p}_V$ is given by

$$\vec{p}_W = M_{V,W}\vec{p}_V. \quad (8)$$

The matrix $M_{W,V}$ is used to transform a position vector $\vec{p}_W$ in world coordinates to a position vector in vehicle coordinates $\vec{p}_V$. It is computed by

$$M_{W,V} = R_{X_V,-\eta} R_{Y_V,-\gamma/2+\psi} T_{-\vec{p}_{O_V,W}} = M_{V,W}^{-1}. \quad (9)$$

The vehicle position vector $\vec{p}_V$ corresponding to a world position vector $\vec{p}_W$ is given by

$$\vec{p}_V = M_{W,V}\vec{p}_W. \quad (10)$$

3) *A camera coordinate system* is defined by $O_C X_C Y_C Z_C$, where O_C is the focal point of the camera, X_C is to the right of the camera, Y_C is down with respect to the camera, and Z_C is in the direction of the optical axis of the camera. Measurements of the camera system available are: the pan angle of the camera (ϕ), the tilt angle (θ), and the camera displacement with respect to the vehicle origin ($\vec{p}_{O_C,V}$). There is no camera roll. The matrix $M_{C,V}$ is used to transform a position vector $\vec{p}_C$ in camera coordinates to a position vector in vehicle coordinates $\vec{p}_V$. The vehicle position vector $\vec{p}_V$ corresponding to a camera position vector $\vec{p}_C$ is given by

$$\vec{p}_V = M_{C,V}\vec{p}_C = T_{\vec{p}_{O_C,V}} R_{Z_C,-\phi-\pi/2} R_{X_C,-\theta-\pi/2}\vec{p}_C. \quad (11)$$

The matrix $M_{V,C}$ is used to transform a position vector $\vec{p}_V$ in vehicle coordinates to a position vector in camera coordinates $\vec{p}_C$. It is computed by

$$M_{V,C} = R_{X_C,\theta+\pi/2} R_{Z_C,\phi+\pi/2} T_{-\vec{p}_{O_C,V}} = M_{C,V}^{-1}. \quad (12)$$

Then the camera position vector $\vec{p}_C$ corresponding to a vehicle position vector $\vec{p}_V$ is given by

$$\vec{p}_C = M_{V,C}\vec{p}_V. \quad (13)$$

4) An *ERIM coordinate system* is defined by $O_E X_E Y_E Z_E$, where O_E is the focal point of the sensor, X_E is to the right of the ERIM sensor, Y_E is down with respect to the ERIM sensor, and Z_E is in the direction of the optical axis of the sensor. Measurements of the ERIM coordinate system available are: the tilt angle of the ERIM sensor (ζ), and the displacement of the ERIM sensor's origin within the vehicle coordinate system ($\vec{p}_{O_E,V}$). There is no ERIM roll or pan, although the orientation of the ERIM coordinate system with respect to the vehicle coordinate system imposes an implicit pan-like rotation of $-(\pi/2)$ about the Z_E axis. The matrix $M_{E,V}$ is used to transform a position vector $\vec{p}_E$ in ERIM coordinates to a position vector in vehicle coordinates $\vec{p}_V$. It is computed by

$$M_{E,V} = T_{\vec{p}_{O_E},V} R_{Z_E,-\pi/2} R_{X_E,-\zeta-\pi/2}. \quad (14)$$

The vehicle position vector $\vec{p}_V$ corresponding to a ERIM position vector $\vec{p}_E$ is given by

$$\vec{p}_V = M_{E,V}\vec{p}_E. \quad (15)$$

The matrix $M_{V,E}$ is used to transform a position vector $\vec{p}_V$ in vehicle coordinates to a position vector in ERIM coordinates $\vec{p}_E$. It is computed by

$$M_{V,E} = R_{X_E,\zeta+\pi/2} R_{Z_E,\pi/2} T_{-\vec{p}_{O_E,V}} = M_{E,V^{-1}}. \quad (16)$$

The ERIM position vector $\vec{p}_E$ corresponding to a vehicle position vector $\vec{p}_V$ is given by

$$\vec{p}_E = M_{V,E}\vec{p}_V.$$

B. Image Transformations

1) ERIM Image Transformations: The ERIM laser scanner produces a 64×256 (rows×columns) image of range values in the scene by actively scanning a modulated laser beam through a 30 degree×80 degree (vertical ×horizontal) field of view. The phase of the beam reflected back from the scene is analyzed to produce the range image. The resulting ERIM data is specified as (c_J, r_J, R_J), where c_J and r_J are the image column and row coordinates of a ray of length R_J. Within the ERIM columns and rows are equally spaced angular increments. We write

$$\rho = c_J k_c \quad (17)$$

$$\omega = -r_J k_r \quad (18)$$

where k_c is a scalar value of radians/column, and k_r is a scalar value of radians/row. Then ρ specifies the rotation of a mirror revolving about the Y_E axis, and ω specifies the rotation of a mirror revolving about the X_E axis. See Fig. 2.

The transformation matrix $M_{S,E}$ used to transform from ERIM image coordinates (c_J, r_J, R_J) to ERIM coor-

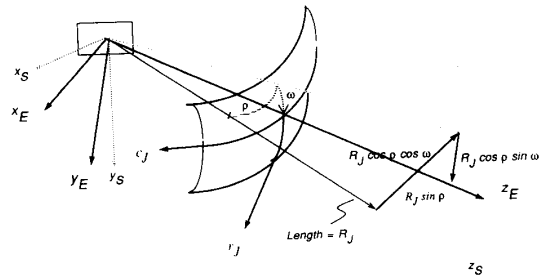


Fig. 2. ERIM image coordinate system and image.

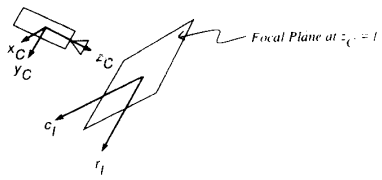


Fig. 3. Camera image coordinate system and perspective transformation.

dinates is defined by

$$\vec{p}_E = \boldsymbol{M}_{S,E}\vec{p}_S = \begin{bmatrix} R_J \sin(c_J k_c) \\ R_J \cos(c_J k_c)\sin(r_J k_r) \\ R_J \cos(c_J k_c)\cos(r_J k_r) \\ 1 \end{bmatrix} \tag{19}$$

where the S coordinate system is as shown in Fig. 2. Also, by referring to Fig. 2 it is easily seen that the inverse solution (i.e., finding c_J, r_J, and R_J from the sensor coordinate $\vec{p}_S$) is

$$\begin{bmatrix} c_J \\ r_J \\ R_J \end{bmatrix} = \begin{bmatrix} (1/k_c)\tan^{-1}\left(\dfrac{x_E}{(y_E^2+z_E^2)^{1/2}}\right) \\ -(1/k_r)\tan^{-1}(y_E/z_E) \\ (x_E^2+y_E^2+z_E^2)^{1/2} \end{bmatrix}. \tag{20}$$

Note that c_J is in units of "columns," r_J is in units of "rows," and R_J is in units of meters.

2) Video Image Transformations: We model the camera image by a perspective transformation with the image plane in front of the camera (i.e., with the image plane along the positive Z_C optical axis). The camera's image plane is modeled as a plane parallel to the camera's $X_C Y_C$ coordinate plane; see Fig. 3. Image coordinates (c_I, r_I) specify the column and row coordinates within the image plane, where $(c,r)_I = (0,0)_I$ is at distance f equal to the focal length along the optical axis of the camera (the Z_C axis). By convention, column values have positive values to the right of the origin (i.e., in the direction of increasing X_C), and row values have positive values toward the bottom of the image (i.e., in the direction of increasing Y_C). Scaling factors a and b are used to convert row and column measures to the units used throughout the rest of the computations (i.e., a is given "columns/meter," and b is given in "rows/meter").

We will use the notation $\boldsymbol{p}_I$ to refer to the 2-D image of a 3-D point $\vec{p}_I$. The perspective transformation matrix $\boldsymbol{P}_I$ maps homogeneous camera position vectors $\vec{p}_C$ into image column-row coordinates:

$$\vec{p}_I = \boldsymbol{P}_I\vec{p}_C = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & s & t \\ 0 & 0 & 1/f & 0 \end{bmatrix}\begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} = \begin{bmatrix} ax_C \\ by_C \\ sz_C + t \\ z_C/f \end{bmatrix} \tag{21}$$

where

$$t = -z_C s + z_C. \tag{22}$$

Because the parameters s and t are functions of z_C the mapping is nonlinear. Dividing the first three terms by the fourth gives

$$\vec{p}_I = \begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} = \begin{bmatrix} \dfrac{fax_C}{z_C} \\ \dfrac{fby_C}{z_C} \\ \dfrac{f(sz_C + t)}{z_C} \end{bmatrix}. \tag{23}$$

The first two coordinates are the $(c,r)_I$ coordinates in the image plane of a projected 3-D point $\vec{p}_C$, and the third component is equal to f whenever (22) holds. Note that (23) provides the image location $\boldsymbol{p}_I$ since we mean

$$\vec{p}_I = \begin{bmatrix} \boldsymbol{p}_I \\ f \end{bmatrix}. \tag{24}$$

Although in general $\boldsymbol{P}_I$ is not invertible (because it is nonlinear), whenever $(0,0,s,t)^T$ is independent of the other rows of $\boldsymbol{P}_I$, $\boldsymbol{P}_I$ is invertible. Retaining the representation of $\boldsymbol{P}_I$ in terms of s and t, the 4 by 4 inverse image perspective transformation matrix is

$$\boldsymbol{P}_I^{-1} = \begin{bmatrix} 1/a & 0 & 0 & 0 \\ 0 & 1/b & 0 & 0 \\ 0 & 0 & 0 & 1/t \\ 0 & 0 & f & -sf/t \end{bmatrix}. \tag{25}$$

$\boldsymbol{P}_I^{-1}$ maps image points $\vec{p}_I$ back into camera coordinates $\vec{p}_C$ providing we express the image position vector $\vec{p}_I$ in homogeneous vector form in terms of a free variable z:

$$\vec{p}_C = \begin{bmatrix} \boldsymbol{p}_I \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_I \\ y_I \\ z \\ 1 \end{bmatrix}. \tag{26}$$

With

$$\vec{p}_C = \boldsymbol{P}_I^{-1}\vec{p}_I \tag{27}$$

we have for $\vec{p}_C$

$$\vec{p}_C = \begin{bmatrix} x_I/a \\ y_I/b \\ f \\ (z - sf)/t \end{bmatrix} \tag{28}$$

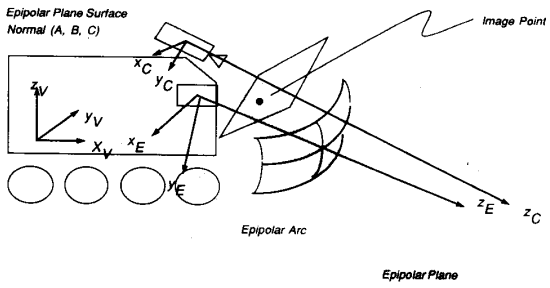


Fig. 4. Fusion geometry.

and in Cartesian form

$$\vec{p}_C = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} \dfrac{tx_I}{a(z - sf)} \\ \dfrac{ty_I}{b(z - sf)} \\ \dfrac{ft}{z - sf} \end{bmatrix}. \tag{29}$$

Solving for z in terms z_C and substituting into the x_C and y_C terms yields the following two equations in three unknowns (x_C, y_C, and z_C):

$$x_C = (x_I z_C)/(af) \tag{30}$$

$$y_C = (y_I z_C)/(bf). \tag{31}$$

Thus the dependence of the inverse mapping on the relationship of (22) is cancelled out, but we are left with the situation where recovering a 3-D point from its image by means of the inverse perspective transformation requires knowledge of at least one of the camera coordinates of the point.

This inverse perspective problem is the subject of the next two sections. In Section III we present an "exact" solution that uses fused range (ERIM) image data and video data. Section IV presents the heuristic methods that we will compare.

III. Combining Video and Range Data

On the ALV, range data from the ERIM sensor can be used to supply a value for z_C (or x_C or y_C) in order to obtain an "exact" solution to (30) and (31). To see how this can be done, we consider Fig. 4, and concentrate on an arbitrary video image point $\vec{p}_{i,W}$; $\vec{p}_{i,W}$ is the three-dimensional position vector of a pixel lying in the video image plane. This point, together with focal points of the video camera and the range sensor, is used to define an epipolar plane. This plane intersects the range sensor's "image sphere" along a spherical arc, which we call the epipolar arc. Clearly the scene point imaged by the video pixel $\vec{p}_{i,W}$ must be imaged by the range sensor somewhere along this epipolar arc. Thus, a one-dimensional search of range image points along this arc can be used to find the range pixel (or pixels) corresponding to the video image point $\vec{p}_{i,W}$. Below, we first solve for the epipolar arc to yield an expression giving the column value of this arc corresponding to any range image row value. Then we discuss the method used for conducting the search along the epipolar arc and the method used to determine a value of z_C for (30) and (31).

Given a camera image point $\vec{p}_{i,W}$ in world coordinates, and the camera and image focal points $\vec{p}_{O_C,W}$ and $\vec{p}_{O_E,W}$ in world coordinates, the epipolar plane is defined by

$$\det \begin{vmatrix} x_W & y_W & z_W & 1 \\ x_{i,W} & y_{i,W} & z_{i,W} & 1 \\ x_{O_C,W} & y_{O_C,W} & z_{O_C,W} & 1 \\ x_{O_E,W} & y_{O_E,W} & z_{O_E,W} & 1 \end{vmatrix} = 0 \tag{32}$$

which is of the form $A_W x_W + B_W y_W + C_W z_W + D_W = 0$, where x_W, y_W, and z_W are world coordinate variables. This world coordinate system based description of the plane can be transformed to a description in ERIM coordinates by applying the world to ERIM transformation $\boldsymbol{R}_{W,E}$ to the plane's normal vector (A_W, B_W, C_W); call the resulting normal vector $\vec{n}_E = (A_E, B_E, C_E)$. Note that since the plane passes through the ERIM origin, the equation of the plane in ERIM coordinates is given by $A_E x_E + B_E y_E + C_E z_E = 0$.

The unit vector corresponding to any column and row of the ERIM image is given by $(x_E, y_E, z_E)^T = (\sin(ck_c), -\cos(ck_c)\sin(-rk_r), \cos(ck_c)\cos(-rk_r))^T$. Then for any unit vector lying in the epipolar plane we have

$$A_E \sin(ck_c) - B_E \cos(ck_c)\sin(-rk_r) + C_E \cos(ck_c)\cos(-rk_r) = 0. \tag{33}$$

Rearranging, we have

$$c = (1k_c)\tan^{-1}[(B/A)\sin(-rk_r) - (C/A)\cos(-rk_r)] = (1/k_c)\tan^{-1}[-(B/A)\sin(rk_r) - (C/A)\cos(rk_r)]. \tag{34}$$

Thus, given any row of the ERIM image we can compute the corresponding column that lies in the epipolar plane.

To perform the search along the epipolar arc, we begin at the bottom row of the range image and search upward within the image. For each point $(r,c)_j$ along the epipolar arc, we use (19), (15), and (8) to determine the three-dimensional location of the imaged point $\vec{p}_{j,W} = (x_{j,W}, y_{j,W}, z_{j,W})$. As we perform the search we look for a pair of epipolar points $(r,c)_j, (r,c)_{j+1}$, such that $z_{j,W}$ is below the ray from O_C through $\vec{p}_{i,W}$ and $z_{j+1,W}$ is above the ray. The intersection of the video line-of-sight ray from O_C through $\vec{p}_{i,W}$ with the scene contour segment whose endpoints are $(x_{j,W}, y_{j,W}, z_{j,W})$ and $(x_{j+1,W}, y_{j+1,W}, z_{j+1,W})$ gives the "exact" solution to the inverse perspective problem. However, due to numerical errors in the computations the video line-of-sight ray and the scene contour segment often do not intersect; we use the midpoint of the shortest line segment connecting the video

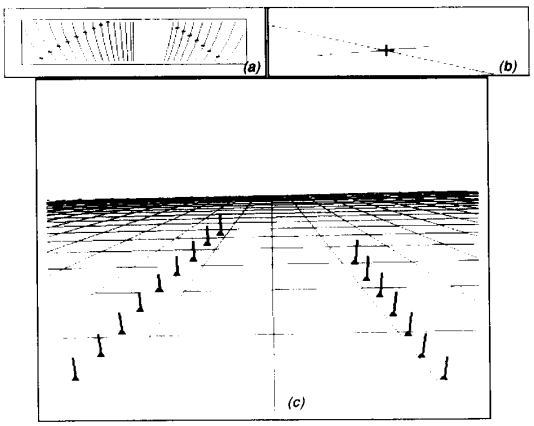


Fig. 5. Results of range/video fusion algorithm showing (a) the epipolar arcs drawn within the ERIM image, (b) profiles showing the camera pixel ray and the elevation derived from the ERIM data along an epipolar arc, and (c) a perspective view of the resulting scene model. See text for a full description of iconography.

line-of-sight ray and the scene contour segment as the solution to the inverse perspective problem.

Fig. 5 shows the results of this fusion algorithm. The figure consists of three windows, which present (a) the epipolar arcs drawn within the ERIM image, (b) profiles showing the camera pixel ray and the elevation derived from the ERIM data along an epipolar arc, and (c) a perspective view of the resulting scene model. Within the perspective view window, each fusion point is represented by a line segment and a triangular "pad," and these are plotted against a grid representation of a horizontal (as determined by the ALV's LNS) plane. The placement of the "pad" indicates the location of the fused point within the plane of the horizontal grid, and the line segment indicates the distance of the fused point above or below the grid. The cross-tics displayed in the epipolar arc window show the road shape in the ERIM field of view. The view is of an up-sloping road that appears above the ideal flat-earth plane.David G. Morgenthaler

IV. Inverse Perspective Algorithms

We have chosen to compare the fusion approach with the following heuristic inverse perspective algorithms: flat-earth [1], hill-and-dale [6], and the modified zero-bank algorithm [8]. Other heuristic inverse perspective algorithms that have been developed, but that we have chosen not to compare, are the vanishing-point algorithm [11] and the zero-bank algorithm [7]. We have chosen not to compare the zero-bank algorithm because we have observed better results from the modified zero-bank algorithm, and we have chosen not to compare the vanishing-point algorithm because, as noted in [7], it is often unstable.

Segmented image regions defining roadways are described within the ALV road-following software as sequences of left and right road boundary points, $\boldsymbol{p}_{R_i}, i = 1, \cdots, N$, and $\boldsymbol{p}_{L_j}, j = 1, \cdots, M$, where the sequences are from the bottom of the image toward the top. For the purposes of performing the inverse perspective computations, a subsampling of these points is usually used. A single road edge may be represented as a list of 100 to 400 edge-points, depending on the resolution used in the border following algorithm and complexity of the segmented road edge. The ALV road-following system typically produces a 10-20 member subsampled list per road edge, and each edge point will usually have a paired opposite edge point in the same row in the image. When the flat-earth inverse perspective algorithm is used the subsampling technique is not critical—the 3-D output is independent of any relationships between the subsampled points. For the flat-earth algorithm we have found it useful to use a subsampling technique on the 2-D road edge list that yields approximately equally spaced 3-D points.

For the hill-and-dale algorithm the subsample of points chosen will affect the behavior of the algorithm. It is assumed by the algorithm that the number of points in the left edge sequence is equal to the number of points in the right edge sequence. Furthermore, a specific 3-D geometric relationship is implied to exist between the points—paired points are also assumed to have the same elevation and to define the width of the roadway. We refer to the pairing of left and right road edge points as "tiling" the roadway.

When the subsampling problem involves selecting paired left and right road edge points for which an implied geometric relationship exists, we must provide a solution to this additional *tiling* problem. Selection of edge point pairs on straight roadways should yield rectangular tiles, while selection of edge point pairs that define the road width on curves should yield pie-shaped tiles. The exact shape of pie-shaped tiles will affect the 3-D locations of road edges output by these algorithms. The tests for the satisfaction of these implied geometric constraints requires knowing the 3-D structure of the world, which is what we are trying to calculate.

The zero-bank inverse perspective algorithms have been developed to compute the tiling using constraints placed on the road model.

A. The Flat-Earth Model

A flat-earth road model allows for a very fast inverse perspective calculation. We have used the flat-earth model on the ALV because its speed allows the vehicle to drive faster—the faster calculation allowed the computation of several scene models, so that the vehicle is never operating in the less accurate, more distant, portion of the scene 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 that is giving support to the vehicle. Thus, the 3-D location of an edge point found in the image 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. If $\vec{p}_{O_C,V} = (x_{O_C,V}, y_{O_C,V}, z_{O_C,V})$ is the position vector of the camera focal point (in vehicle coordinates), and $\vec{p}_{i,V}$ is the position vector of a road edge point (in vehicle coordinates), then $\vec{p}_{O_C,V} - \vec{p}_{i,V} = (x, y, z)$ is the direction vector of the ray (in vehicle coordinates). Some multiple mz is the elevation of the camera above the road: $mz = z_{O_V,C}$. Then the three dimensional coordinates of the road edge point are given by $(mx, my, mz)_V$.

The flat-earth geometry model has several advantages over the other inverse perspective models. The first of these is its speed. Second, this model can be applied to any single image point, even those that 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 heuristic algorithms discussed here that is presently capable of handling intersections or other roadway events that violate ribbon-like roadway constraints. 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 that 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 forward, 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.

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 the other inverse perspective algorithms.

B. The Hill-and-Dale Model

While flat-earth geometry is clearly an assumption that is very useful in certain circumstances, the flat-earth assumption is not accurate enough for many road-following applications. A hill-and-dale geometry model [3] that uses a fast "shape from contour" method was developed as another alternative. While the hill-and-dale model accommodates rising and falling roadways, its output is sensitive to the subsample of roadway edge points used (the tiling problem).

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.

The first step of the algorithm is to use flat-earth geometry to solve for the three-dimensional locations of $\vec{p}_{1,C}$ and $\vec{p}_{2,C}$, the bottom left and bottom right road edge points. From this it is possible to compute the width of the road w, where $w = \|\vec{p}_{1,C} - \vec{p}_{2,C}\|$.

To maintain a constant road width in the scene model, we intersect the rays defined by each successive pair of left and right road edge points with planes parallel to the ground plane but of differing heights. Because these rays for each pair are diverging, an elevated plane will produce a narrower road than a lower plane. For each pair of edge points $\vec{p}_{i,I}$ and $\vec{p}_{j,I}$, we compute the elevation h of an intersecting plane such that the distance between the intersections of the rays defined by these points in the plane is equal to w as follows:

$$h = \sqrt{\frac{w^2}{\left(\frac{x_{i,C}}{z_{i,C'}} - \frac{x_{j,C'}}{z_{j,C'}}\right)^2 + \left(\frac{y_{i,C'}}{z_{i,C'}} - \frac{y_{j,C}}{z_{j,C'}}\right)^2}} \tag{35}$$

where the C' coordinate system is centered at the camera focal point and is orthogonal to the vehicle coordinate system, and the $x_{C'}$, $y_{C'}$, and $z_{C'}$ are the position vectors of the points in the image plane. This roadway elevation is then used in a flat-earth geometry calculation to produce 3-D locations corresponding to the image points.

This algorithm produces more accurate scene models than the flat-earth algorithm on straight or slightly curved roads that go up and down hill. However, the algorithm is very dependent upon good segmentation because 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. 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 that 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 assumption of this algorithm is violated at intersections, and may be violated at other roadway areas as well.

C. The Zero-Bank and Modified Zero-Bank Models

The hill-and-dale model uses the constraint that a road generally keeps an approximately constant width. As mentioned, the *tiling* problem of applying this constraint

is that one must find the pairs of edge points separated by a distance equal to the road width in both straight or curved parts of the road. It appears that to solve the tiling problem the assumption that the road has a constant width constraint is not sufficient. Another constraint must be added for the pairing of edge points to be possible. The *zero-bank* constraint specifies that the road reconstruction may not tilt sideways. Thus, the *zero-bank* algorithms [7], [8] models the road as a space ribbon generated by a center line spine and horizontal line segments of constant length 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." While this is computationally the most costly algorithm, it both addresses the tiling problem and accommodates rising and falling roads.

In the original zero-bank algorithm [7], the road reconstruction method based on these constraints was incremental; a new pair of 3-D edge points could be found if we had already found a neighbor pair of edge points; the road edges were reconstructed incrementally from edge points close to the vehicle to edge points in the distance (the pair of points closest to the vehicle being provided by the flat-earth algorithm). This method was fragile because any increment of construction depends on the previous elements in the chain.

The modified zero-bank algorithm [8] uses the same space ribbon model as the original algorithm, but further assumes that the tangents to the road edges at the end points of cross-segments are approximately parallel. This added constraint is used to find pairs of points (it is on the edges of the road image) that are images of the end points of cross-segments. For points picked on one side of the road image, the proposed algorithm generally finds several matching point candidates on the other side, based on local properties of a road. However among these candidates only one generally corresponds to the global road. A dynamic programming algorithm is applied to reject the candidates that do not correspond to world points of the global road. The modified zero-bank algorithm can be decomposed into the following steps:

1) Image points anywhere on one image edge curve are candidates for being matching points to those on the other image edge curve (two image points are called matching points if they are images of the end points of cross-segments of the 3-D road). Thus, if a_1 and a_2 are matching points and $\vec{a}'_1$ and $\vec{a}'_2$ are the tangent directions to the image edges at these points, the following relation holds:

$$\left[\vec{V}\times(\vec{a}_1\times\vec{a}_2)\right]\cdot\left[(\vec{a}_1\times\vec{a}'_1)\times(\vec{a}_2\times\vec{a}'_2)\right]=0 \quad (36)$$

where $\vec{V}$ is the vertical direction. For edge curves approximated by polygonal lines, the matching point a_2 can be on a line segment, and its position between the end points of the line segment can be expressed by a number between 0 and 1, whereas its tangent vector $\vec{a}'_2$ is constant; or the matching point a_2 can be at an end point of a line segment, with a constant position but with a tangent angle that can be expressed by a number between 0 and 1 within the range of angles of the two adjacent line segments. For each point picked on one image edge, we check for each of the line segments of the other image edge if a matching point belongs to that line segment, i.e., if the expression gives a linear coordinate between 0 and 1 for this line segment. Matching points at the nodes of the polygonal line are found by checking if the expression gives a number between 0 and 1 for the tangent angle.

2) For each point chosen from one edge image, the previous step may give several matching points on the other edge image. One of the reasons is that the images of the edges can be very rough and wiggly. Another reason is that the condition used is only a necessary condition for two points to be matching points in the image of the road. This condition is local—the matching points pairs that are the most globally consistent should be chosen, and the rest discarded. The criteria of global optimization are three-dimensional. From the pairs of matching points, the corresponding three-dimensional cross-segments are found in a fashion similar to the hill-and-dale algorithm. This correspondence is unique if the cross-segments are assumed horizontal and of known constant length. (The constant length is the width of the road, which cannot be defined by this method. It must be obtained from other methods, such as stored data about the road, the flat-earth algorithm applied close to the vehicle, or close-range methods such as stereoscopy or range-video fusion.)
3) The group of matching point pairs corresponding to a single point chosen on one edge is the image of a group of world cross-segments obtained at the previous step, and the world road can go through *at most one* of these cross-segments. If a sequence of points along one road edge is taken, a sequence of groups of cross-segments is obtained, and the world road must go through at most one of the cross-segments of each group, in the same order as the sequence of points chosen on the first road image edge. Each cross-segment can be represented as a node in a graph. A path must be found in the graph that (a) visits each group in the proper sequence and goes through at most one node of each group, and that (b) maximizes an evaluation function that characterizes a "good road." The total evaluation function is the sum of the functions of each of the arcs of the graph. The evaluation function for an arc is the sum of weighted criteria that grade the choices of individual cross-segments and the neighborhood of consecutive cross-segments based on angular considerations.

V. Comparison of Inverse Perspective Algorithms

Given sequences of left and right edge points p_{L_i} and p_{R_i} generated from typical roadway scenes we wish to compare the accuracy of the inverse perspective algorithms. The algorithms we have chosen to compare are the flat-earth, hill-and-dale, modified zero-bank, and the exact solution given by the fusion technique. We have chosen not to present results for the vanishing point algorithm because the instability of the algorithm is such that it cannot be used (on the ALV) without a great deal of modification; and not to present results for the zero-bank algorithm because of the superior performance of the modified zero-bank algorithm.

The comparison was made by acquiring and recording image sets that are representative of the situations encountered on the Martin Marietta ALV test track. The image data set contained straight flat roads, uphill and downhill roads, banked corners, hill-tops and valley-bottoms, and combinations of these. We cannot present complete results of all algorithms on all images due to space limitations. Instead, we have chosen the most illustrative image-and-algorithm combinations, and we describe the salient considerations for the use of each algorithm.

An assessment of the absolute accuracy of these algorithms would require that we obtain ground truth for the road scenes imaged. Obtaining ground truth data by performing a detailed survey of each roadway scene could not be considered because of cost. Instead, our original approach was to use the output of the "exact" fusion algorithm as the absolute basis of comparison; the pitfalls of this approach are discussed below. Thus, rather than making a quantitative assessment of absolute accuracy of the various algorithms, we instead present the outputs of the algorithms so that they can be compared to each other, but not to an absolute ground truth. Finally, an assessment of the accuracy of these algorithms requires selection of a representative set of test images. We have chosen images of roads that exhibit crests, valley, curves, banks, varying widths, and combinations of these.

Fig. 6 shows a comparison of the four algorithms on a curving road scene. As in Fig. 5, fusion-derived edge points are shown as a triangular pad attached to the ground plane, with an attached vector that indicates whether the 3-D point is above or below the ground plane. In Fig. 6(a) flat-earth points are shown as horizontal bars, and by definition they lie in the perspective grid. Fig. 6(b) compares the fusion results with the modified zero-bank results. Zero-bank points are shown as bars lying in the perspective plane, and attached vectors show how far each 3-D point is above or below the plane. The road tiling scheme is critical to the zero-bank algorithm, and these tiles are displayed in 6(b). The hill-and-dale results are compared with fusion in panel 6(c), and the hill-and-dale points are shown as arrows. The arrow head indicates the 3-D location of the point, the base of the arrow lies in the perspective plane, and the arrow's direction indicates whether the point is above or below this plane. An overhead view of all four approaches is shown in 6(d), clearly depicting downrange and crossrange disagreements between the algorithms.

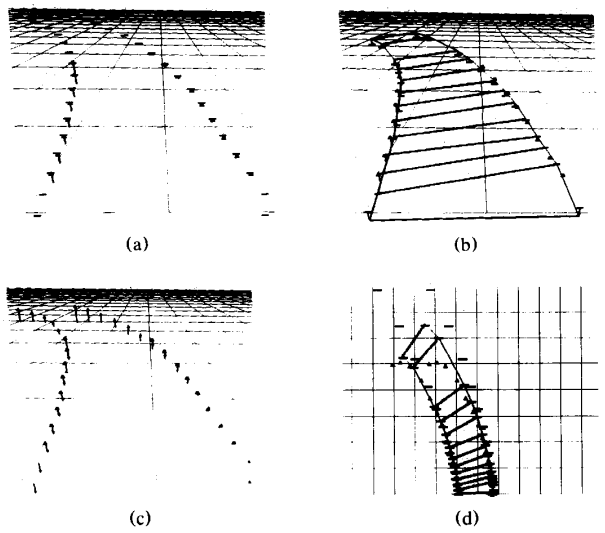


Fig. 6. Results on a curved road scene. (a)–(c) Fusion algorithm compared with: (a) the flat-earth algorithm; (b) the modified zero-bank algorithm; and (c) the hill-and-dale algorithm. An overhead view of all four approaches is shown in (d).

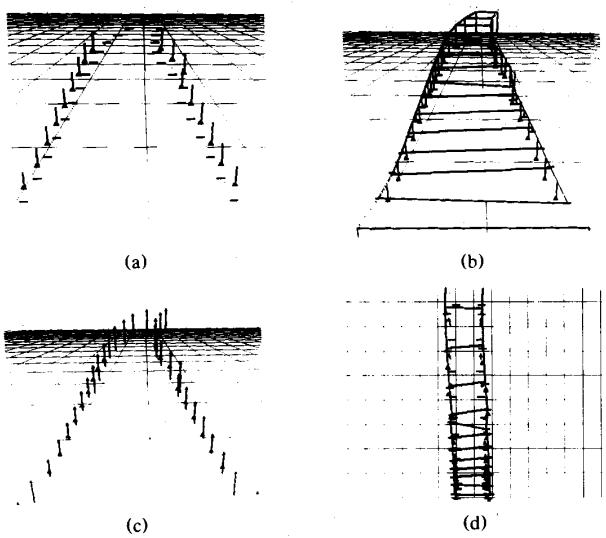


Fig. 7. Results on a slightly curved, up-sloping road scene. (a)–(c) Fusion algorithm compared with: (a) the flat-earth algorithm; (b) the modified zero-bank algorithm; and (c) the hill-and-dale algorithm. An overhead view of all four approaches is shown in (d).

The view is of a sharp curve with moderate banking, and 6(a) shows that the fusion algorithm properly depicts this: the outside (right) edge of the curve is approximately 0.5 m higher than the inside (right) edge. (The perspective grid shows 3-m squares.) Note that the fusion road edges

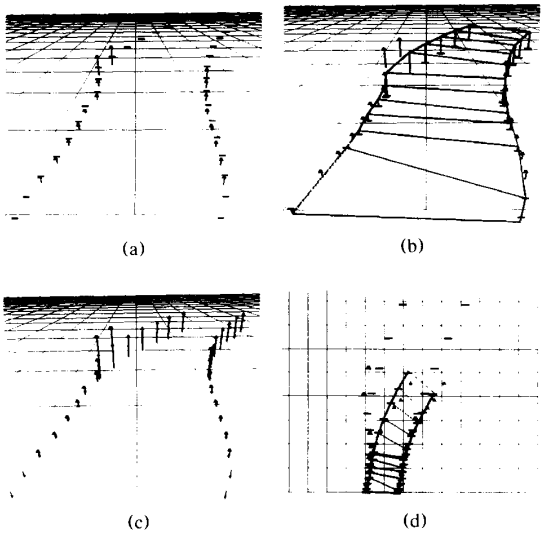


Fig. 8. Comparison of results for a video image with poor segmentation. (a)–(c) Fusion algorithm compared with: (a) the flat-earth algorithm; (b) the modified zero-bank algorithm; and (c) the hill-and-dale algorithm. An overhead view of all four approaches is shown in (d). Notice that the fusion algorithm places two points of the left road edge in a ditch.

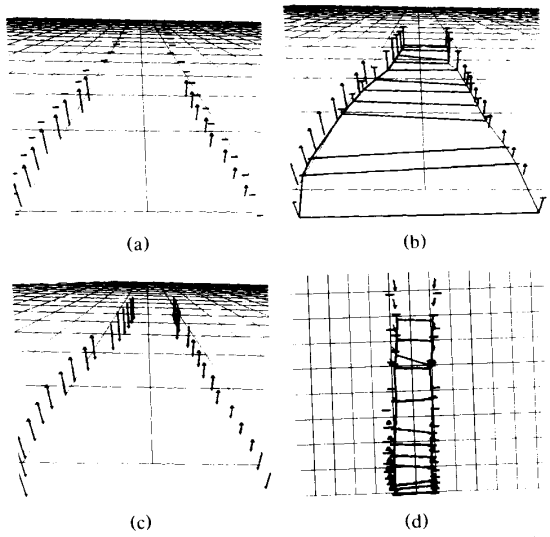


Fig. 9. Comparison of results for a road with a belly-like "dip" in the middle distance. (a)–(c) Fusion algorithm compared with: (a) the flat-earth algorithm; (b) the modified zero-bank algorithm; and (c) the hill-and-dale algorithm. An overhead view of all four approaches is shown in (d).

extend out only 15-20 m from the vehicle due to the limited lookout distance of the ERIM scanner as it is mounted on the ALV. (The bottom of the perspective grid occurs at about 5 m in front of the vehicle.) There is good x, y agreement between fusion and flat-earth in their overlap region, but Fig. 6(d) shows clearly how the flat-earth road edges diverge in the far distance. The modified zero-bank approach only moderately diverges from the fusion approach in the overlap region, primarily showing an elevation discrepancy: it depicts the road edges as higher in space than the fusion points. Fig. 6(d) clearly shows how modified zero-bank maintains constant road width across its derived road tiles and serves to "pull" the curve closer to the vehicle. While the hill-and-dale points agree well in elevation and x, y distance with the fusion points in the overlap region (Fig. 6(c)), the error of assuming a constant road width across an invalid road tile is seen in Fig. 6(d). The last three points are depicted at the same downrange distance from the vehicle, as the plane they are projected to is "lifted" to accommodate a road that appears too wide in the image due to a poor tiling scheme.

Fig. 7 shows a comparison of the various algorithms on a slightly curved, upsloping road. The overhead view shows that all approaches except flat-earth are in substantial agreement. The flat-earth algorithm produces a road that becomes narrower with distance. This is because while travelling uphill, the ALV rocks backward on its suspension, so that the "flat-earth" plane determined by the LNS is actually more inclined than the road. The opposite effect is observed for down-sloping roads, that is, the flat-earth algorithm produces a road that becomes wider with distance. This is because while travelling downhill, the ALV rocks forward on its suspension, so that the "flat-earth" plane determined by the LNS is actually less inclined than the road. The fusion approach (Fig. 7(a)) shows how the road rises radically in the near region. In the modified zero-bank algorithm, the road is initially constrained to lie in the flat-earth plane, but further out in the scene the algorithm properly depicts the rise of the road. Hill-and-dale, not hampered with the road-tiling problem evident in the curving road of Fig. 6, properly shows the lifting road in results almost identical to the modified zero-bank approach.

Fig. 8 shows an example in which the fusion approach gives poor results due to poor segmentation. The last two fusion points on the left side occurred in an over-segmented portion of an image of a curved road. The fusion approach determined that these false road edge points lied distinctly further out and below the actual road plane: it found them on a downsloping portion of terrain on the other side of the banking road "lip." Over the near distance, all algorithms are in close agreement with the fusion points derived from sound segmentation. In the far distance, the three heuristic algorithms tend to behave as they did in the curve depicted in Fig. 6: the flat-earth edges diverge, the hill-and-dale edges "lift" radically to maintain constant width across invalid road tiling, and the modified zero-bank pulls the curve in towards the vehicle.

Fig. 9 shows a comparison of the three algorithms for a road with a belly. The flat-earth algorithm performs poorest; the road is rendered too wide, and the width changes with the relative elevation of the real roadway. The modified zero-bank algorithm agrees with the fusion algorithm in terms of road width, but the zero-bank algorithm places the road much higher near the vehicle (i.e., the

zero-bank output is mostly level with the grid near the vehicle, while the fusion output is well below the grid). This is because the modified zero-bank chooses the road width based on the first tile (notice that the near edge of the first tile is below the grid, while the far edge of the first tile is on the grid), so that a constant elevation offset can be introduced. This introduction of an elevation offset could be mitigated through the use of other information, such as from a previous scene model. However, the net effect of the elevated scene model in this example is that individual edge points are rendered closer to the vehicle by the modified zero-bank than by the fusion algorithm. The overhead view shows that while there are significant elevation discrepancies between the algorithms, their resulting placement of road edges in the perspective grid is comparable.

VI. Conclusion

One objective of the work reported in this paper is to present data that will enable intelligent decisions to be made when contemplating the design of autonomous road-following systems. In order to use the data presented in the previous section, the designer must consider several system engineering issues. We discuss these issues here.

None of the inverse perspective algorithms considered will reproduce roadways in all circumstances (fusion fails when given poor segmentation, flat-earth fails when the earth is not flat, etc.). Thus, in assessing the use of each algorithm we must consider its behavior in all of the expected roadway geometries. Each algorithm may be considered to produce, on average, a usable scene model length. Within this length the data are judged to be sufficiently accurate for navigation. With regard to this issue we find that the usable scene model lengths produced by fusion are short due to the limitations of the ERIM sensor. We would rank the usable scene model lengths produced by hill-and-dale, flat-earth, and modified zero-bank in that order, from shortest to longest. In particular, the modified zero-bank algorithm appears to give the longest usable scene models when considering only 2-D (within the grid) accuracy.

A second system engineering issue to be considered for each algorithm is its resilience to "catastrophic" problems such as poor segmentation. Clearly flat-earth and fusion rank highly with respect to segmentation problems, since the output for each point is independent of other edge points in the video image—errors cannot accumulate. The three heuristic algorithms rank above the fusion algorithm with respect to insensitivity to sensor calibration (both inter- and intra-sensor calibration).

The third system engineering issue to be considered for each algorithm is its computation speed. Certainly it is clear that too long an execution time will preclude the use of an algorithm, even if it is the most accurate. With respect to speed, we rank the algorithms from fastest to slowest (on a serial processor) as flat-earth, hill-and-dale, modified zero-bank, and fusion. Execution times for all of the algorithms are small enough (relative to video image segmentation) that all of these could be used on the ALV, especially with the use of the parallel processors (WARP).

Less clear is the trade-off between execution time and usable scene model length. During 1987 we used the flat-earth algorithm because of its speed, even though we believed that the (unmodified) zero-bank algorithm would produce longer usable scene models. This is because the absolute speed of the flat-earth algorithm permitted production of scene models a such a rate that the vehicle never operated on the distant portions of the scene models—a new scene model took its place before the vehicle traveled more than half of the current one.

In summary, we feel that the fusion algorithm is the algorithm of choice when a range sensor is available and when the data from the range sensor are usable throughout the region of interest (e.g., in our case the range data should be good to about 25 m). The best heuristic algorithm studied is the modified zero-bank; it agreed well with the "exact" output of the fusion approach, but its usable scene model length was not limited by the range of data produced by the ERIM sensor. The flat-earth algorithm can certainly be used successfully, but a greater burden is placed on the navigation system. We would not recommend the hill-and-dale algorithm as presented here.

References

[1] Matthew A. Turk, David G. Morgenthaler, Keith D. Gremban, and Martin Marra, "VITS—A Vision System For Autonomous Land Vehicle Navigation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-10, no. 3, pp. 342–361, May 1988.

[2] M. E. Bair, R. Sampson, and D. Zuk, "3-D imaging and applications," in *Proc. SPIE Intelligent Robotics and Computer Vision Conf.*, Oct. 1986.

[3] T. Dunlay, "Obstacle Avoidance Perception Processing for the Autonomous Land Vehicle," *Proc. IEEE Robotics Automat. Conf.*, Apr. 1988.

[4] M. Daily, J. Harris, D. Keirsey, K. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous Cross-Country Navigation with the ALV," in *Proc. DARPA Knowledge-Based Planning Workshop*, Austin, TX, Dec. 1987.

[5] "Perception offroad software developments," *The Autonomous Land Vehicle (ALV), Phase II, Second Quarterly Scientific and Technical Report*, Sept. 30, 1988.

[6] D. Morgenthaler, "Hill and dale geometry for the ALV," Martin Marietta internal memo, May 1986.

[7] D. DeMenthon, "A zero-bank algorithm for inverse perspective of a road from a single image," in *Proc. IEEE Int. Conf. Robotics Automat.*, Raleigh, N.C., pp. 1444–1449, Apr. 1987.

[8] ____, "Reconstruction of a road by matching edge points in the road image," *Center for Automation Research Tech. Rep. CAT-TR-368*, Univ. Maryland, June 1988.

[9] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision Appl.*, vol. 1, pp. 223–240, 1988.

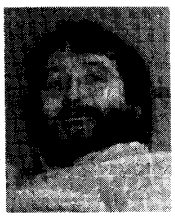
[10] ____, "Applications of dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 241–261, 1988.

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

David G. Morgenthaler received a B.S. in electrical engineering in 1975 from Princeton University, and an M.S. in computer science in

1978 and a Ph.D. in computer science in 1981 from the University of Maryland, College Park. His dissertation was on three-dimensional image processing.

He joined Martin Marietta Denver Aerospace in 1982 as head of a computer vision and image processing research group. He has led computer vision research and development activities on robotic programs for manufacturing, space, and military applications. He was the technical lead on the Autonomous Land Vehicle program, and is now technical lead on a program to develop a real-time model based vision system.



Stephen J. Hennessy (S84–M'84–S'85–M'85) received the B. S. degree in english from Northern Illinois University, Chicago, in 1975 and the M.S. degree in bioengineering from University of Illinois at Chicago in 1985.

He is currently a Senior Engineer at Martin Marietta Information and Communications Systems in Denver, Colorado. From 1986 to 1989 he worked in the Vision Group on the Autonomous Land Vehicle Program, which explored integrated vision systems for on- and off-road vehicle navigation in natural environments. Since 1989 he has worked on sensor fusion problems and the application of model-based vision techniques to SAR (Synthetic Aperture Radar) imagery. His professional interests include Computer Vision and Pattern Recognition.

Daniel DeMenthon was born in Lyon, France, in 1949. He graduated from Ecole Centrale de Lyon in 1972. He also received the Diplome d'Etudes Approfondies degree in applied mathematics from University Claude Bernard, Lyon, in 1973, and the M.S. degree from University of California, Berkeley in 1979.

He is currently a Senior Research Engineer in the Computer Vision Laboratory, Center for Automation Research, University of Maryland. His current research interests are in sensor-based robotics, computer vision and parallel computing.