3
Algorithms for Road Navigation

LARRY S. DAVIS, DANIEL DEMENTHON, SVEN DICKINSON, AND PHILIP VEATCH

3.1 Introduction

This paper provides a summary of the research conducted at the University of Maryland during the past five years on problems associated with visual navigation of ground vehicles. This research has been driven by a variety of scientific and engineering goals, including

1. the identification of principles of organization for autonomous navigation systems,
2. the identification of fundamental scientific problems that must be addressed in the course of designing and developing visual navigation systems, and
3. the implementation of prototype visual navigation systems that operate in the real world (ideally in real time) and demonstrate progress toward the solution of a specific problem in visual navigation.

Our research has focused primarily on the development of a complete system for visual navigation of roads and road networks (described in detail in Refs [22] and [10]). We call this system **MARF** for **MAryland Road Follower**. MARF was developed as part of the Autonomous Land Vehicle (ALV) program sponsored between 1984 and 1988 by the Defense Advanced Research Projects Agency. It was able to visually navigate a ground vehicle over unfamiliar roads. If *a priori* information was available in the form of a map, then MARF was able to use that information to navigate through intersections whose geometries were coarsely specified in the map. More important than the actual navigation problem that MARF attempted to solve was the organization of the system. That organization emphasized the importance of **focus of attention** (both to minimize computation time and to make maximal use of accumulated expectations), **explicit representation of visual search strategies**

The support of the Defense Advanced Research Projects Agency (ARPA Order No. 6350) and the U.S. Army Engineer Topographic Laboratories under Contract DACA76-88-C-0008 is gratefully acknowledged.

(to support extensibility and ease of modification), and **sensor integration at the symbolic level**.

An important dimension in the design of autonomous vehicles is the decomposition of the system into modules. The most prevalent decomposition is a functional one (referred to by Brooks [4] as a **horizontal** decomposition). So, for example, our road following system has modules for image processing, sensor control, inverse perspective, etc. Brooks [4] argues that such a functional decomposition will lead to systems with very complex interfaces between the modules that are, additionally, very difficult to modify or extend. Our experience with the initial implementation of our road following system certainly supported Brooks contentions; however, our last implementation, based on a set of interacting blackboards with explicit representations for objects and processes (described in detail in Dickinson and Davis [10]), allowed us to overcome many of these difficulties and, perhaps, could form the basis for the design of even more general systems. We describe this system in some detail in Section 2.

The problem of recovering an explicit model for the three-dimensional geometry of the road is especially important when the sampling interval for the vehicle (time taken to acquire and process a frame of data) is large compared to the speed of the vehicle. The most straightforward approach to reconstructing the geometry of a road once its boundaries or lane markings are detected is to assume that the road is flat and that one can measure (using either an inertial sensor or a range sensor) the orientation and height of the camera with respect to the road plane. However, small errors in the estimation of these parameters, or small deviations of the road from flatness, can result in extremely large errors in the recovered three-dimensional road coordinates. If the road contains a curve, and the distance to the curve is either over- or underestimated because of such errors, then the vehicle might be driven off the road. In Section 3, we describe a monocular inverse perspective algorithm developed by DeMenthon [8]. This so-called "zero-bank inverse perspective algorithm" was tested extensively using reference road reconstructions obtained by data fusion between range images and video images from the Martin Marietta ALV [17].

While the ultimate goal of autonomous road following systems is to navigate in the presence of other moving objects, it is also important to be able to navigate around stationary obstacles on the road. The most straightforward approach for identifying road obstacles is based on range images and involves comparing the observed height of points from the range image against the predicted height of road points. This approach suffers from two serious problems:

1. Small errors in estimating the geometry of the road can lead to very large errors in estimating the heights of road points.
2. Even if the road geometry is known, small errors in estimating the orientation of the range sensor with respect to the road can lead to very large errors in height estimation.

In Veatch and Davis [21], we proposed an obstacle detection algorithm based on comparing the observed range derivatives of road pixels against the predicted derivatives. We showed that such an approach is much less sensitive to errors than algorithms based on height comparison or other similar approaches. Section 4 contains a description of the obstacle detection algorithm and experimental results.

3.2 Maryland Road Follower

While the development of MARF involved the solution of many difficult technical problems (image analysis algorithms for detecting road boundaries and markings in color images of roads, robust inverse perspective algorithms for road geometry reconstruction, algorithms for obstacle detection in range imagery, etc.), we would like to focus here on the organization of the system at a high level and describe the representations adopted that allow MARF to reason about what to look for, which sensors to use, and how to recover from failure.

Figure 3.1 contains a road image taken from the Martin Marietta ALV at their Denver, Colorado, test site. From images such as these, MARF constructs a partial model of its three-dimensional environment. This *scene model* contains the objects visually identified by MARF and forms the basis for planning a path through the environment.

For road following, the scene model contains objects from which the location of the road can be determined. Most simply, the scene model might contain the locations of the left and right boundaries of the road. However, it would also be possible to navigate the vehicle based on locating the right boundary and the centerline, the centerline only, the location of a ditch that is known to run parallel to the road at a given distance from the road, etc.

Generally, there are many cues in the environment that contain direct or

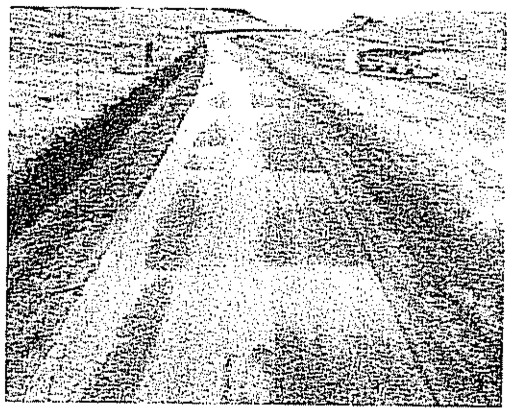


FIGURE 3.1. Typical ALV road image. Reprinted with permission from *IEEE Transactions on Robotics and Automation* ("A Flexible Tool for Prototyping ALV Road Following Algorithms" by Sven Dickinson and Larry Davis, 6(2), 1990). © 1990 IEEE.

indirect evidence concerning the location of the road. MARF must have some basis for deciding which objects in the environment to search for and how to search for them. These decisions can be based on information including the recent history of object detection (i.e., the right road boundary might have been robustly detected recently because of high color contrast and so is a good candidate for the current road tracking method), the current contents of the scene model (i.e., what has already been successfully detected in the current frame and where), and information from a road map.

Detection, verification, and accurate delineation of these objects are themselves complex tasks involving sensor control, computationally demanding sensor data processing algorithms, and fusion of either raw data or analyses from multiple sensors Methods for performing such tasks evolve as the road following problem becomes better understood. We would like to increase the class of objects that the vehicle can use to determine its motion, include new sensors on the vehicle, embed new sensor data processing algorithms into the system, and modify and extend the navigation strategies employed by the vehicle. The successful evolution of such a system depends on the ability of its control structure and knowledge representations to accommodate such changes.

In the system to be described, the scene model is represented as a network of frames, with each frame corresponding to a class of objects and encapsulating the relevant information pertaining to that class. The control structure used to construct the network is based on a system of communicating production systems that implement a structured blackboard. The blackboard is partitioned into regions, each of which corresponds to a specific class of frames and contains the rules that define the attributes of the class. The system promotes modularity and maintainability through a structured object representation and a structured control scheme.

3.2.1 System Overview

A scene model is constructed in MARF through the cooperation of two modules:

1. the scene model planner (Planner) is responsible for deciding what objects to search for and where in the image to search for them, and
2. the scene model verifier (Verifier) is responsible for sensor control and sensor data processing required to verify the scene predictions made by the Planner.

The Planner determines its sequence of object predictions based on knowledge about the current scene, the history of processing of recent frames, and the goals associated with the current navigation task. The Verifier controls the acquisition and analysis of data acquired from the sensors. For example, if requested to locate the image of the road's left boundary at a distance of 10–15 m in front of the vehicle, it would choose an appropriate sensor based on map information and its current history of road boundary detection, choose a sensor data processing algorithm, determine the appropriate pointing direction for the sensor based on estimates of the vehicle's position in the world and its motion, acquire the sensor data, and process it. All of these decisions are arrived at by the application of a rule-based system to structured databases of facts and conjectures about the world, the vehicle, and its sensors.

Here, we would like to present a detailed description of the Planner, and we would specifically like to illustrate how the explicit representation of visual search strategies in the Planner makes it relatively straightforward to experiment with competing strategies.

The Planner is implemented as a frame whose slots point to the modules with which the Planner communicates. There are currently slots for

1. the scene model,
2. the a priori road map,
3. the local navigation task, and
4. the Verifier.

The unique feature of this framework is that it inherits the capabilities of a YAPS [1], a production system providing a rule database, a factual database, and a conflict resolution strategy. The Planner's principal goals are to choose what objects in the world to search for, deduce the location of the road from the detection of some subset of these objects (the Verifier may fail to detect some of the objects sought by the Planner), and decide when to terminate processing of a frame.

In our current implementation, the only local navigation task is to follow the road. A road map is available that specifies the geometry of the road network at a coarse level (i.e., where road intersections occur and how many roads meet at those intersections). Detailed information about the road geometry between intersections is not provided.

In our original road following system [22], a fixed search strategy was employed to detect the road. It operated by predicting where in a video image the left (and right) boundaries at a fixed distance in front of the vehicle would appear. This prediction was based on its accumulated three-dimensional model of the road geometry and an estimate of the vehicle's position in that model. The system (1) placed windows in the image surrounding those points, with the size of the window heuristically determined by estimates of the prediction error; (2) applied simple image processing algorithms to those windows (based on edge detection and line extraction) to locate the road boundary; and (3) placed subsequent windows in the image that overlapped the previous windows by a fixed percentage, oriented along the extrapolated direction of the road. Because the strategy was implemented directly in code, it was very difficult to modify or extend.

In MARF, these strategies are represented explicitly in the Planner as rules, making it straightforward to specify the conditions under which a strategy should be employed, what new strategy to invoke should the current one (or ones, since several could be pursued in parallel) fail, etc. We illustrate this with a simple example.

Since the road patches seen in successive images taken from the vehicle have a large intersection, one might imagine that for straight roads, at least, it is not necessary to reprocess all those parts of the current image that correspond to road segments identified in previous frames. For example, we might want to experiment with a search strategy that, after having detected 10 m of straight road, would *skip over* the image of the next 10 m of road, thus saving the time required to process those parts of the image containing the skipped 10 m (of course, in order to do this, we must either have determined the three-dimensional geometry of that 10-m patch from the analysis of previous frames or we must make some simplifying assumptions, such as the road is straight and flat over that 10-m patch).

Among the rules that collectively define the road patch search strategy, the following rule defines the search strategy as disconnected:

```
(defp define-disconnected-search-strategy
      (hypothesized object -object)
      (goal (define search strategy for road patch -object))
test(null (<- object 'search-strategy))
    (cond ((not (null (<- object 'priori-road-straightness)))
    (>= (<- -object 'priori-road-straightness)
    MIN-ROAD-STRAIGHTNESS)))
—>
    (<- -object 'set-search-strategy 'disconnected)).
```

The antecedent of the rule (the conditions preceding the —>) is a conjunction of conditions that must be satisfied in order for the consequent (the expression following the —>) to be executed. The first antecedent expression matches a road patch hypothesis created by the Planner. The second expression represents the current goal of the Planner; in this case, the Planner is attempting to define the search strategy of the road patch hypothesis. The next two expressions, called test clauses, specify further conditions that must be met; the search strategy must be previously undefined and the prior road straightness must exceed the value of MIN-ROAD-STRAIGHTNESS. The symbol <- indicates message passing between objects; for example, in the first test condition, a message is passed to the road patch hypothesis, bound to the variable-object, requesting the value of the search strategy attribute. If both of these conditions are met, then the search strategy is defined as disconnected.

A second rule defines the search location of the road patch hypothesis:

```
(defp define-disconnected-search-location
    (hypothesized object -object)
    (goal (define search location for road patch -object))
test(null (<- -object 'search location))
```

```
    (eq (<- -object 'search strategy) 'disconnected)
-->
    (<- -object 'set-search-location
    (list (<- (<- * yaps-db * 'scene-model)
    ':predict-extended-left-feature-seed
    MAX-EXTENDED-SEARCH-DISTANCE)
    (<- (<- * yaps-db * 'scene-model)
    ':predict-extended-right-feature-seed
    MAX-EXTENDED-SEARCH-DISTANCE)))).
```

In this rule, the consequent defines the search location as a value resulting from sending two queries to the scene model, requesting points extrapolated from the left and right road patch segments, respectively, of the last patch in the scene model.

Suppose, now, that we wanted to change this strategy by having the vehicle look further and further ahead as it identifies more and more road patches. So, if the strategy initially attempts to skip 5 m of road, then if it were successful, it might next try to skip 10 m of road. To accommodate this new "dynamic search strategy," we add the following rule to the Planner:

```
(defp define-dynamically-disconnected-search-strategy
      hypothesized object -object)
      (goal (define search strategy for road patch -object))
test(null (<- -object 'search strategy))
    (cond ((not (null (<- -object 'prior-road-straightness)))
    (>= (<- -object 'prior-road-straightness)
    MIN-ROAD-STRAIGHTNESS)))
    (eq (<- (<- (<- * yaps-db * 'scene model)
    ':retrieve-most-recent-road-patch) 'search-strategy)
    'disconnected)
-->
    (<- -object 'set-search-strategy 'dynamically-disconnected)).
```

In the test clauses, we check that the search strategy of the road patch hypothesis is undefined, and we make sure that we have accumulated a sufficient amount of straight road in the scene model. In addition, we check that the previously verified road patch was verified using the disconnected search strategy. If all of these conditions hold, the search strategy is defined to be dynamically disconnected. To define the search location for this new strategy, we add the following rule:

```
(defp define-dynamically-disconnected-search-location
      (hypothesized object -object)
      (goal (define search location for road patch -object))
test(null (<- -object 'search-location))
    (eq (<- -object 'search-strategy) 'dynamically disconnected)
-->
    (<- -object 'set-search-location
    (list (<- (<- * yaps-db * 'scene-model)
```

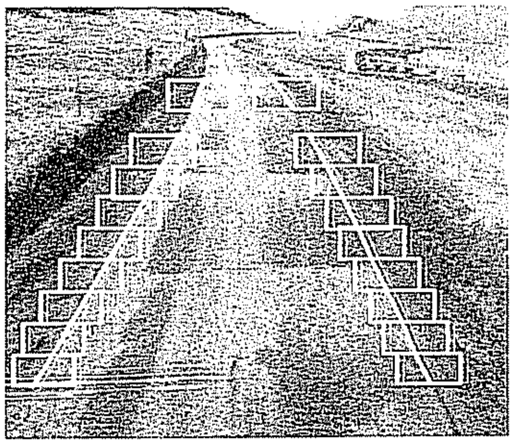


FIGURE 3.2. Tracking a straight road. Reprinted with permission from *IEEE Transactions on Robotics and Automation* ("A Flexible Tool for Prototyping ALV Road Following Algorithms" by Sven Dickinson and Larry Davis, 6(2), 1990). © 1990 IEEE.

```
': predict-extended-left-feature-seed
MAX-EXTENDED-SEARCH-DISTANCE)
(<- (<- yaps-db 'scene-model)
':predict-extended-right-feature-seed
MAX-EXTENDED-SEARCH-DISTANCE))))
```

In this rule, the rule consequent defines the search location to be MAX-EXTENDED-SEARCH-DISTANCE from the last road patch in the scene model. By controlling the value of this parameter, we can control the extent of road skipped by the road detection strategy.

We illustrate the results of these search strategies with an image from the Martin Marietta test site. Figure 3.2 is an image of a straight segment of road. Initial search windows are placed near the bottom of the image based on the predicted location of the road in the image. Using these initial search windows, the connected search strategy is invoked until 10 m of road have been inserted into the scene model. At that time, the disconnected search strategy is invoked, and 10 m of road are skipped in the image. Since the road is straight, the search strategy is successful in identifying the road boundaries in the next pair of windows. With approximately 20 m of straight road in the scene model, the disconnected search strategy is again invoked; however, when the three-dimensional search location of the next road patch is mapped to the current image, the search windows are out of bounds (off the top of the image).

3.3 Recovery of Three-Dimensional Road Geometry

We now present an algorithm for reconstructing the road shape from a single image, providing the three-dimensional profile of the road in front of the vehicle, often up to the point where the road becomes hidden. Reconstructing the road over a large distance presents several advantages. The reconstructions from several video frames can be overlapped, and the evidence from each reconstruction can be combined for added reliability. The road reconstruction can be registered to a stored map of the road network and contribute to locating the position of the vehicle on the map. Finally, a system that makes estimations of turns well in advance can adjust its speed accordingly. This long-range observation of the road does not preclude the use of a shorter range road analysis in the control loop of the vehicle steering [11].

Once the images of the road boundaries are computed, road reconstruction can be posed as a "shape from contour" problem. The problem is, of course, underconstrained—an infinite number of world curves can correspond to each road image boundary unless additional assumptions are made about the three-dimensional geometry of the road.

The simplest assumption one can make is that the earth is flat, and that the vehicle is supported on the ground plane. The three-dimensional location of any image point is then simply the intersection of the line of sight through this point and the known ground plane. Reconstruction from the flat earth model is very fast, and it is not sensitive to slight misplacements of the road boundaries in the image. However, it is very sensitive to errors in estimating the camera tilt angle with respect to the ground plane. While the rocking of the vehicle on its suspension can be measured, changes of ground slopes of several degrees within the field of view are common. If the camera is situated at a height H above the ground and sees a point at a distance L from the vehicle (Figure 3.3), then if the ground plane is overestimated by an angle ε, the estimated value L' of L will be

$$L' = L/[1 - (L/H)\tan\varepsilon].$$

For a camera mounted on a vehicle with $H = 3.5$ m, a world point 30 m in front of the vehicle will be located 55 m in front of the vehicle if the ground plane angle is overestimated by 3° and at 21 m if it is underestimated by 3°—an error range of over 100%.

In an attempt to overcome the limitations of the Flat Earth method, authors have added constraints to the road model, assuming that a road generally keeps an approximately constant width [20] or that directions of road edges may be parallel and may be deduced from their vanishing points

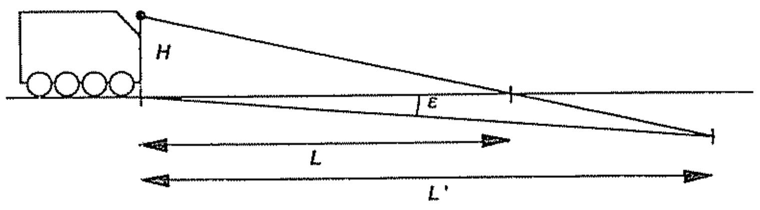


FIGURE 3.3. Error in distance estimates due to ground slope in the Flat Earth geometry model.

[16, 19]. A problem with applying these constraints is that one must find *which* pairs of points are separated by a distance equal to the road width and *which* pairs of points have parallel edge directions in straight or curved parts of the road. We call the problem of locating the correct pairs of points in the image the *matching point problem*.

The constant road width constraint is not sufficient. We find that another constraint must be added for the reconstruction to be possible. We have chosen the *zero-bank* constraint, specifying that the road does not tilt sideways. A road model combining constant width and zero bank was originally suggested in Ozawa and Rosenfeld [18].

In previous work, we had developed an incremental road reconstruction method based on these constraints [7] in which a new pair of edge points could be found if we had already found a neighboring pair of edge points; the road edges were reconstructed incrementally from edge points close to the vehicle to edge points in the distance. This method was fragile because any increment of construction depended on the previous elements in the chain.

This incremental method used a discrete approach. Incremental road reconstructions based on a differential approach can be found in Kanatani and DeMenthon [12] and Kanatani and Watanabe [14]. An interesting alternative to the global dynamic programming optimization proposed in the present paper can be found in Kanatani and Watanabe [13].

3.3.1 Summary

The proposed algorithm can be decomposed into the following steps:

1. In a preliminary step, not detailed here, appropriate image processing techniques have isolated the two curves of the edges in the image, and a polygonal approximation has been found for each edge curve.
2. Picking image points anywhere on one image edge curve, we are able to find the points that are candidates for being matching points on the other image edge curve. (Two image points are called matching points if they are images of the endpoints of cross segments of the 3-D road). This matching is made possible by making reasonable hypotheses about the shape of the road, which add enough constraints to make the problem solvable. Specifically, the road is modeled as a space ribbon defined by a centerline spine and horizontal cross segments of constant length cutting the spine at their midpoints at a normal to the spine. We further assume that tangents to the ribbon edges at endpoints of cross segments are approximately parallel (Section 3.3.2). We find an expression that must be satisfied by the two image points located on the facing image edge curves and the tangents to the edge images in order for the two points to be matching points (Section 3.3.3). If a_1 and a_2 are matching points and $\mathbf{a}'_1$ and $\mathbf{a}'_2$ are the tangent directions to the image edges in these points, the following relation holds:

$$[\mathbf{V} \times (\mathbf{a}_1 \times \mathbf{a}_2)]\ [(\mathbf{a}_1 \times \mathbf{a}'_1) \times (\mathbf{a}_2 \times \mathbf{a}'_2)] = 0,$$

where **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 endpoints of the line segment can be expressed by a number between 0 and 1, whereas its tangent vector $\mathbf{a}'_2$ is constant; or the matching point a_2 can be at an endpoint 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 (Section 3.3.8). For each point chosen from one image edge, we check for each of the line segments of the other image edge to determine if a matching point belongs to that line segment, that is, if our expression gives a linear coordinate between 0 and 1 for this line segment. Then, we look for matching points at the nodes of the polygonal line by checking that the expression gives a number between 0 and 1 for the tangent angle.

3. 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 and we must still choose the matching points pairs that are the most globally consistent and discard the other pairs. The criteria of optimization are three dimensional (Section 3.3.9); thus, at this step of the algorithm, from the pairs of matching points, the corresponding three-dimensional cross segments must be found. This correspondence is unique if the cross segments are assumed horizontal[1] and of known constant length (Section 3.3.2). The constant length is the width of the road, and it cannot be defined by this method. The assumed road width is a scaling factor in the reconstruction, whereas the optimization is based on angular considerations, which are independent of scaling. For driving a vehicle, the road width must eventually be obtained from other methods, such as stored data about the road, the Flat Earth method, or close-range methods, such as stereoscopy or time-of-flight ranging.
4. 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 (Section 3.3.9). 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 visits each group in the proper sequence and goes through at most one node of each group and that maximizes an

[1] The vehicle reconstructs horizontal cross segments on the basis of its knowledge of the vertical direction, and therefore, it must be equipped with a vertical direction sensor.

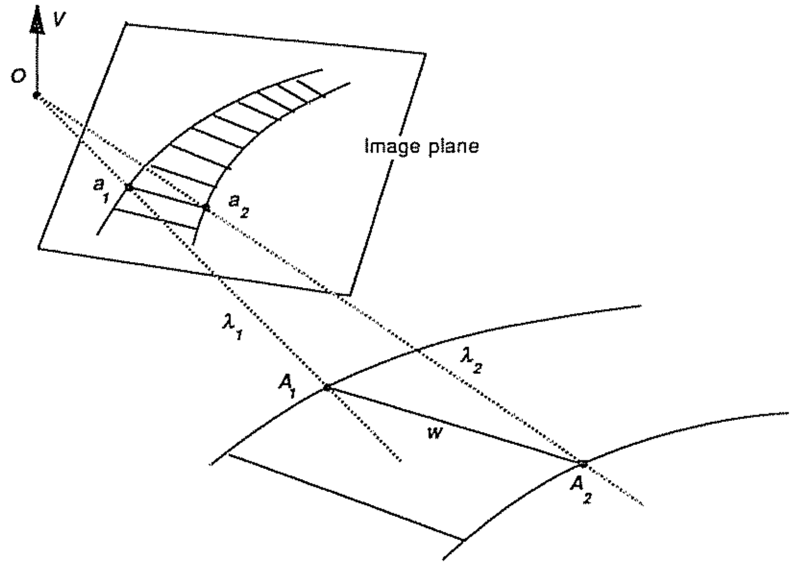


FIGURE 3.4 Reconstructing positions of world railroad ties from their images. Reprinted with permission from *IEEE Transactions on Robotics and Automation* ("Reconstruction of a Road by Local Image Matches and Global 3D Optimization" by Daniel DeMenthon and Larry Davis). © 1990 IEEE.

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, which grade the choices of individual cross segments and the neighborhood of consecutive cross segments based on angular considerations.

3.3.2 The Matching Point Problem

Consider the image of a railroad track and its railroad ties and assume that some appropriate image processing techniques have reduced the images of the rails to curves and the images of the ties to line segments between these curves (Figure 3.4). The positions of the endpoints of the tie segments on the curves of the rail are the matching points in the image. The reconstruction of the shape of the railroad track in 3-D space uses the matching points and is straightforward once three hypotheses are made.

1. The width w of the railroad track is constant and known.
2. The coordinates of the vertical unit vector **V** are known in the camera coordinate system.
3. The railroad ties are approximately horizontal.

Note that the last hypothesis does not mean that the railroad itself should be horizontal. Similarly, the stairs in a spiral staircase have horizontal step edges, but the ruled surface defined by these step edges is far from horizontal.

Consider two matching points a_1 and a_2, the endpoints of the image of a tie. The corresponding vectors from the viewpoint O to these image points will be denoted by $\mathbf{a}_1$ and $\mathbf{a}_2$. The corresponding world points A_1 and A_2 are defined by

$$\mathbf{A}_1 = \lambda_1 \mathbf{a}_1, \qquad \mathbf{A}_2 = \lambda_2 \mathbf{a}_2$$

since world points and their images are on the same line of sight.

The world line segment is assumed horizontal; the two parameters λ_1 and λ_2 are then related by

$$\lambda_2 = m\lambda_1,$$

with

$$m = \mathbf{a}_1 \cdot \mathbf{V}/\mathbf{a}_2 \cdot \mathbf{V}.$$

The requirement that the distance between $\mathbf{A}_1$ and $\mathbf{A}_2$ be equal to the width w completely constrains the parameters:

$$\lambda_1 = w/(\mathbf{a}_1^2 + m^2\mathbf{a}_2^2 - 2m\mathbf{a}_1 \cdot \mathbf{a}_2)^{1/2} \tag{3.1}$$

Thus, the two curves of the rails in the scene can be, in general, uniquely reconstructed from their images up to a scale factor if the ties are assumed horizontal and of constant length. Problems occur only if the railroad image crosses the horizon, as noted in Kanatani and Watanabe [13]. In this case, the ties are horizontal on the horizon line, and their range cannot be determined, as can be seen from the previous equations.

Consider now the problem of reconstructing a *road* from its image, once some appropriate image processing techniques have isolated the curves corresponding to the road edges in the image. Now, of course, we do not have the images of railroad tie segments to help us. The method we propose involves first finding the endpoints of line segments that correspond to images of railroad tie segments and then doing the 3-D reconstruction of the endpoints of the images of these segments by the method just described for the railroad. We call these world segments corresponding to railroad ties cross segments, and their endpoints opposite points. The images of these points are the matching points. The main problem of road reconstruction from an image can then be stated: Given a point on one edge of the road image, where is the matching point on the other edge?

We choose a road model similar to the railroad model: the road is modeled as a space ribbon generated by a centerline spine and horizontal cross segments of constant length cutting the spine at their midpoints at a normal to the spine. This modeling gives cross segments the properties of railroad ties.

Cross segments are horizontal, that is, perpendicular to the vertical (on the ALV the vertical was detected by trim sensors).
Cross segments have constant length (the road width).
Cross segments are perpendicular to both road edges, that is, locally perpendicular to the centerline of the road.

Saying that cross segments are normal to both road edges means that they are normal to the tangents to the edges at their endpoints. Note that this does not generally mean that the tangents to opposite points are parallel. In DeMenthon [8], however, we show that assuming the tangents to opposite points to be approximately parallel is a reasonable assumption in most configurations. This assumption considerably simplifies the recovery of opposite points from the image. It is added to our road model and used in the following section.

3.3.3 Conditions for Two Image Points to Be Matching Points

Consider a world road defined by two 3-D curves E_1 and E_2 and the road image defined by two image curves e_1 and e_2. Assume that two opposite points A_1 and A_2 on road edges E_1 and E_2 have been found. Their images are a_1 and a_2 (Figure 3.5.), and the following properties are dictated by the world road model.

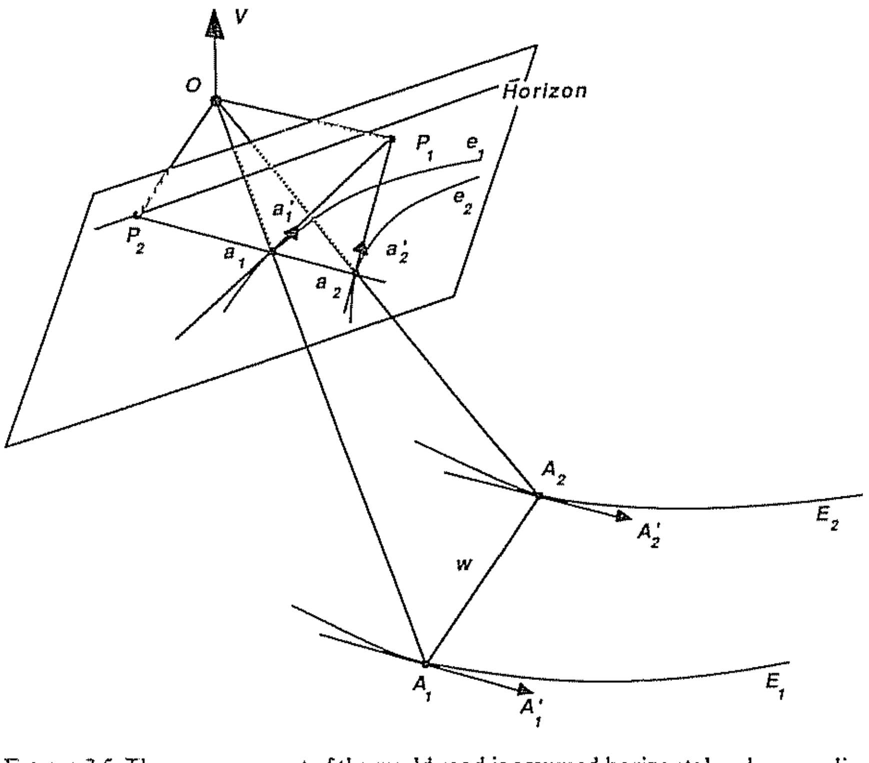


FIGURE 3.5 The cross segment of the world road is assumed horizontal and perpendicular to the tangents at its endpoints. The tangents are assumed parallel. A condition satisfied by the matching points in the image that also involves the image tangents and the vertical direction is deduced. Reprinted with permission from *IEEE Transactions on Robotics and Automation* ("Reconstruction of a Road by Local Image Matches and Global 3D Optimization" by Daniel DeMenthon and Larry Davis). © 1990 IEEE.

1. The segment A_1A_2 is horizontal.
2. The tangents to the road edges at A_1 and A_2 are perpendicular to A_1A_2.
3. The tangents to A_1 and A_2 are approximately parallel.
4. The tangent $\mathbf{a}'_1$ to the image edge e_1 at a_1 is the image of the tangent $\mathbf{A}'_1$ to the world edge E_1 at A_1; the tangent $\mathbf{a}'_2$ to the image edge e_2 at a_2 is the image of the tangent $\mathbf{A}'_2$ to E_2 in A_2. This is a general property of projected curves and tangents.

In deriving the following consequences, we make use of the property that the direction of the intersection of two planes is perpendicular to the normals of each plane and can be obtained by the cross product of the two normals.

3.3.4 Directions of Tangents to Opposite Points

If a_1 and a_2 are matching points and $\mathbf{a}'_1$ and $\mathbf{a}'_2$ are the tangents to the image edges at these points, the direction of the corresponding world tangents is

$$(\mathbf{a}_1 \times \mathbf{a}'_1) \times (\mathbf{a}_2 \times \mathbf{a}'_2).$$

Proof: If a_1 and a_2 are images of opposite points, the world tangents to the world edges are parallel. Since the images of the world tangents are $\mathbf{a}'_1$ and $\mathbf{a}'_2$, the world tangents lie on the planes $(O\mathbf{a}_1, \mathbf{a}'_1)$ and $(O\mathbf{a}_2, \mathbf{a}'_2)$, respectively. These planes are not parallel since they share the point O, and they do not coincide. Since the tangents are parallel, they must be parallel to the intersection of these planes. The direction of this intersection is given by the previous expression.

3.3.5 Direction of a Cross Segment

If a_1 and a_2 are matching points and $\mathbf{V}$ is the vertical vector, the direction of the world cross segment is $\mathbf{V} \times (\mathbf{a}_1 \times \mathbf{a}_2)$.

Proof: A_1A_2 belongs to a horizontal plane since it is horizontal. Since a_1a_2 is the image of A_1A_2, A_1A_2 also belongs to the plane (O_1, Oa_2). This plane is generally not horizontal. Thus, the direction of the segment A_1A_2 is given by the intersection of a horizontal plane with the plane (Oa_1, Oa_2). The normal to the horizontal plane is the vertical vector $\mathbf{V}$. The direction of the normal to the plane (Oa_1, Oa_2) is given by the cross product $(\mathbf{a}_1 \times \mathbf{a}_2)$. Thus, the direction of A_1A_2 is given by $\mathbf{V} \times (\mathbf{a}_1 \times \mathbf{a}_2)$.

3.3.6 Matching Condition

If a_1 and a_2 are matching points and $\mathbf{a}'_1$ and $\mathbf{a}'_2$ are the tangent directions to the image edges in these points, the following relation holds:

$$[\mathbf{V} \times (\mathbf{a}_1 \times \mathbf{a}_2)] \cdot [(\mathbf{a}_1 \times \mathbf{a}'_1) \times (\mathbf{a}_2 \times \mathbf{a}'_2)] = 0 \qquad (3.2)$$

Proof: If a_1 and a_2 are images of opposite points, the direction of the cross segment A_1A_2 is perpendicular to the direction of the parallel tangents.

3.3.7 Local Normal to the Road

If a_1 and a_2 are matching points and $\mathbf{a}'_1$ and $\mathbf{a}'_2$ are the tangents to the image edges at these points, the local normal to the world road has the direction given by

$$\mathrm{N} = [\mathrm{V} \times (\mathbf{a}_1 \times \mathbf{a}_2)] \times [(\mathbf{a}_1 \times \mathbf{a}'_1) \times (\mathbf{a}_2 \times \mathbf{a}'_2)]. \qquad (3.3)$$

Proof: The local planar patch of the world road is defined by A_1A_2 and by the parallel tangents at A_1 and A_2. The direction of the normal to this plane is the cross product of the directions of the cross segment and the tangents.

To summarize, when a point a_1 and the tangent $\mathbf{a}'_1$ to the road image are given, Equation (3.2) becomes an equation that must be satisfied by the coordinates of a_2 and the slope of the tangent to the edge in a_2 in order for a_2 to be a matching point to a_1. We can also find the direction of the normal along the corresponding world cross segment A_1A_2.

3.3.8 Search for a Matching Point of a Given Image Point

If a point a_1 is chosen on one edge image and if the other edge image is a polygonal line, the matching point a_2 can be located on one of the line segments of the polygonal line or at one of the vertices between the segments. All the line segments and all the vertices are checked, because a single point a_1 can have several matching point candidates due, for example, to edge irregularities. Other reasons are considered in DeMenthon [8]. For each line segment and for each vertex, the equations developed in the next two subsections are applied.

Search for a Matching Point on a Line Segment

Assume that the segment being considered is the segment p_2q_2. The matching point a_2 is on this segment if

$$\mathbf{a}_2 = p_2 + \lambda \mathbf{p}_2\mathbf{q}_2, \qquad (3.2)$$

with λ between 0 and 1. The point a_2 must also, with its tangent to the edge, satisfy Equation (3.2). The tangent $\mathbf{a}'_2$ to the edge image in a_2 is approximated by the vector $\mathbf{p}_2\mathbf{q}_2$. We replace $\mathbf{a}'_2, \mathbf{a}_2$ by their values $\mathbf{p}_2\mathbf{q}_2$, and $\mathbf{p}_2 + \lambda\mathbf{p}_2\mathbf{q}_2$ in Equation (3.2), and transform cross product combinations into dot products by the well-known identity

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c}.$$

The resulting value for λ is

$$\lambda = -\frac{(\mathbf{V} \cdot \mathbf{a}_1)(\mathbf{K} \cdot \mathbf{p}_2) - (\mathbf{K} \cdot \mathbf{a}_1)(\mathbf{V} \cdot \mathbf{p}_2)}{(\mathbf{V} \cdot \mathbf{a}_1)(\mathbf{K} \cdot \mathbf{p}_2\mathbf{q}_2) - (\mathbf{K} \cdot \mathbf{a}_1)(\mathbf{V} \cdot \mathbf{p}_2 q_2)}, \qquad (3.4)$$

where $K = (\mathbf{a}_1 \times \mathbf{a}'_1) \times (\mathbf{p}_2 \times \mathbf{q}_2)$. If λ is between 0 and 1, the intersection is between the endpoints of line segment p_2q_2, and the value of λ specifies the position of a_2 on p_2q_2. The search also takes place among the vertices between the line segments.

SEARCH FOR A MATCHING POINT AT A VERTEX

We can think of a point q_2 linking two line segments p_2q_2 and q_2r_2 as a point at which the slope of the tangent to the edge changes from the slope of the segment p_2q_2 to the slope of the segment q_2r_2. An approach similar to the previous subsection is followed. A matching point a_2 is at the vertex q_2 if

$$\mathbf{a}_2' = \mathbf{p}_2\mathbf{q}_2 + \mu(\mathbf{q}_2\mathbf{r}_2 - \mathbf{p}_2\mathbf{q}_2), \tag{3.2}$$

with μ between 0 and 1. For this point to be a matching point to a_1, it must also satisfy Equation (3.2). This produces the following value for μ.

$$\mu = -\frac{(\mathbf{M}\cdot\mathbf{q}_2)(\mathbf{n}_1\cdot\mathbf{p}_2\mathbf{r}_2) - (\mathbf{n}_1\cdot\mathbf{q}_2)(\mathbf{M}\cdot\mathbf{p}_2\mathbf{r}_2)}{(\mathbf{M}\cdot\mathbf{q}_2)[\mathbf{n}_1\cdot(\mathbf{q}_2\mathbf{r}_2 - \mathbf{p}_2\mathbf{q}_2)] - (\mathbf{n}_1\cdot\mathbf{q}_2)[\mathbf{M}\cdot(\mathbf{q}_2\mathbf{r}_2 - \mathbf{p}_2\mathbf{q}_2)]}, \tag{3.5}$$

where $\mathbf{n}_1 = \mathbf{a}_1 \times \mathbf{a}_1'$ and $\mathbf{M} = \mathbf{V} \times (\mathbf{a}_1 \times \mathbf{q}_2)$.

If the resulting value of μ is between 0 and 1, a matching point a_2 to the point a_1 is located at the vertex q_2.

In the following, we provide a detailed description of how the sets of all matching points for a point on one road boundary edge are determined. We then describe the dynamic programming optimization algorithm and illustrate its applications to some simple examples.

3.3.9 Dynamic Programming Road Reconstruction

Given a point from one road boundary edge, the shape-from-contour algorithm identifies several potential matching points from the other road boundary edge. This group of matching point pairs is the image of a group of world cross segments, and the world road can pass through at most one of these cross segments. If a sequence of points along one road edge is taken, then a sequence of groups of cross segments is obtained, and the world road must pass through at most one of the cross segments from each group. Once an optimization criteria is defined, it is then straightforward to find the "optimal" path through the set of groups of cross segments. Consider a directed acyclic graph (DAG) in which the nodes at level i correspond to the cross segments constructed from the matching pairs for the ith boundary point on one road edge, and arcs connect all pairs of nodes on consecutive levels. We seek a path through this DAG from a level 0 node to a level n node, where there are $n + 1$ levels in the DAG. We append a special node to each level, a null node, to allow us to skip over a level that for numerical reasons, for example, might yield no correct cross segments. We next define a merit function on the arcs of the DAG. Let $A = A_1A_2$ be a cross segment from a node at level k and B_1B_2 be a cross segment from level $k + 1$. Then, the merit function for the arc connecting A_1A_2 to B_1B_2 is the weighted sum of three criteria C_i defined as follows.

1. The local normal to A (Equation 3.3) should be nearly vertical, so we set C_1 to be the dot product between the vertical and the normal to A.
2. The slope of the patch containing the two cross segments must be close to vertical. We define that slope to be $A_1B_2 \times A_2B_1$, and we define C_2 to be the dot product between the unit vector in the direction of this cross product and the vertical.
3. The average direction of the two cross segments must be perpendicular to the line joining their midpoints; C_3 measures how well this condition holds.

When the value of any one of these criteria falls below a threshold, the arc is eliminated from the DAG.

The dynamic programming algorithm then simply maintains a set of optimal paths through k levels and extends them to the $(k + 1)$ level by considering the nodes one at a time at the $(k + 1)$ level and retaining only the path from the kth level that is extended with maximal overall merit to the currently considered node on the $(k + 1)$ level. So, at any stage of the optimization, we are only retaining as many paths as there were nodes on the previously considered level (in practice, no more than 4 or 5). The null nodes allow us to skip levels; arcs to null nodes are assigned sufficiently low merit values to deter the optimization from ignoring good cross segments.

3.3.10 Experimental Results

Here, we present the results of road reconstruction using synthetic data only. Additional experimental results on real road images are presented in DeMenthon [8]. Figure 3.6 shows a top and side view of a synthetically generated road. The nominal road width is 4 m, and the road centerline profile (see the side view) is an element from a sinusoid from a crest to a trough. The road slope is modified by varying the sinusoid amplitude. The term *road slope* refers to the slope at the midpoint of the straight segment between the two turns. For the synthetic road, the road slope is $H/38.9$, where H is the difference in meters between the lowest and highest points of the road. In the top view shown in Figure 3.6, the road has a short straight segment, then takes a 45° right turn and then a left turn separated by a short straight segment. The camera's position, orientation and parameters, also shown in the figure, were taken equal to those of the ALV camera at Martin Marietta.

A benchmark was developed for measuring the performance of the proposed matching point algorithm and other road reconstruction algorithms. A reconstructed road is called *navigable* if the tracks of a vehicle of known width (2 m in our examples) following the centerline of the reconstructed road stay within the edges of the actual road over the entire reconstruction and never cross these edges. This requires that no cross segment is shifted sideways by more than one quarter of its width. Notice, that nonnavigable reconstructions are still usable if they are not too far from the actual road. Indeed, a sufficiently fast processor could generate reconstructions in a fixed coordinate

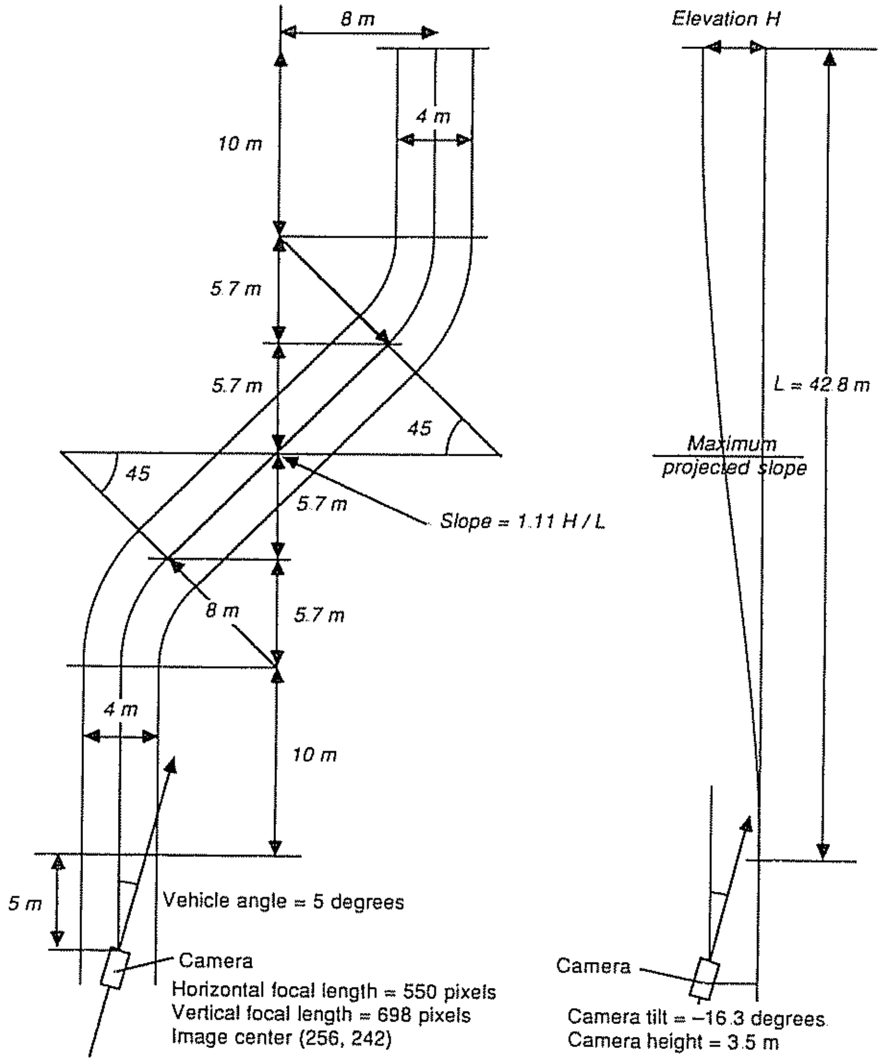


FIGURE 3.6. Synthetic road geometry with two 45° turns and camera position. (a) Top views. (b) Side view showing the $\frac{1}{4}$-period sinusoidal profile of the centerline over the length L. Reprinted with permission from *IEEE Transactions on Robotics and Automation* ("Reconstruction of a Road by Local Image Matches and Global 3D Optimization" by Daniel DeMenthon and Larry Davis). © 1990 IEEE.

system quickly enough such that the composite reconstruction from a sequence of frames is navigable, even though constituent reconstructions may not be. With this in mind, a reconstruction is called *usable* if the centerline of the reconstructed road stays within the edges of the actual road. In other words, a usable reconstruction is a reconstruction in which no cross segment is off the actual road by more than one half of its width. Considering a large number of synthetically generated roads with random variations in their geometry, percentages of navigable and usable reconstructions are computed and characterized by statistics such as the percentage of usable reconstruction.

Specifically, random variations are introduced about the nominal values of road width (4 m) and the road bank (0°). The random width and bank variations are described by Gaussian distributions, and several standard deviations for the Gaussian were employed in the experiments. Five road slopes were chosen: −10%, −5%, 0%, 5%, 10%. Forty roads were produced for 25 combinations of slopes and standard deviations, and the results for these 40 roads were averaged to yield the points plotted in the following graphs.

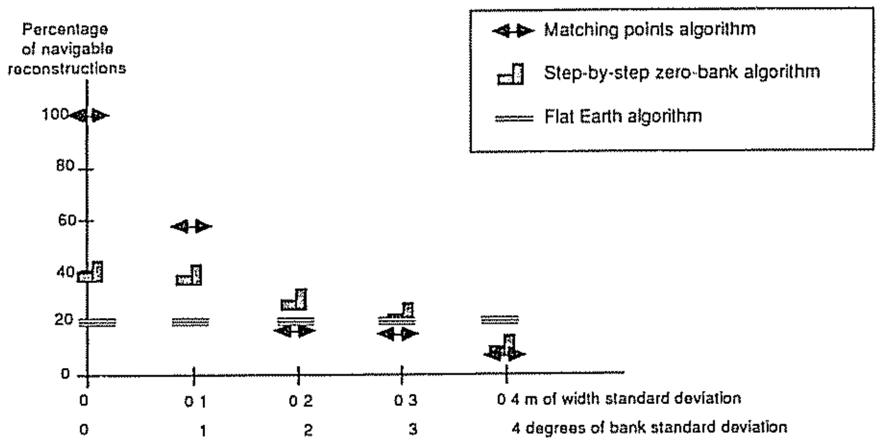
In Figure 3.7, results are shown for the matching point algorithm. The algorithm produces 100% navigable roads when the road has no irregularities of width and bank, regardless of slope. The rate drops to 5% for the maximal width and bank. Other diagrams display how much navigable length is recovered in reconstructions that are completely navigable and in reconstructions that are only partially navigable. Comparisons have also been conducted between the matching points algorithm, the original iterative zero-bank algorithm, and the new matching points algorithm. Not surprisingly, the Flat Earth model gives 100% reconstruction success if the road *is* flat, but cannot produce navigable or usable road reconstructions once any road slope is allowed. On synthetic data, the principle advantage of the matching points algorithm (its ability to recover from local catastrophic errors caused by, for example, image processing errors in localizing road boundaries) is not much in evidence, although it still led to a higher percentage of usable reconstruction. Details are available in DeMenthon [8], and DeMenthon and Davis [9].

3.4 Detection of Stationary Obstacles on Roads

Here, we provide a description of the obstacle detection algorithm proposed by Veatch and Davis, and we present some experimental results. A more complete description of the implementation (that addresses problems of mixed pixels) can be found in Veatch and Davis [21].

What is an obstacle? Generally, an obstacle is a region that a vehicle cannot or should not traverse. Avoiding regions that a vehicle is physically capable of traversing but for some reason should not go (such as not driving the wrong way down a one-way street) would require a level of artificial intelligence that is beyond the scope of this work.

Excluding places that a vehicle can go but should not, one is left with regions that can be defined by their shape and material properties. Rocks, street signs, and steep slopes are all obstacles whose defining characteristics are their shapes. Swamps and ice patches on the other hand may have sufficiently flat surfaces for navigation but their material properties make them



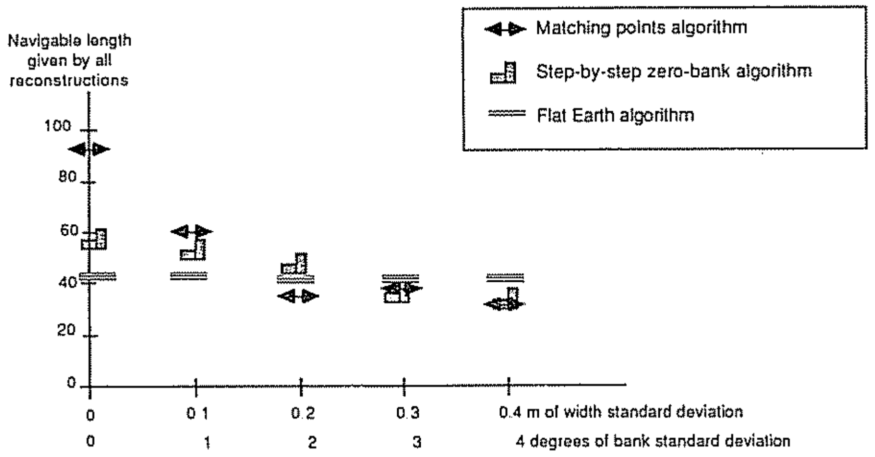
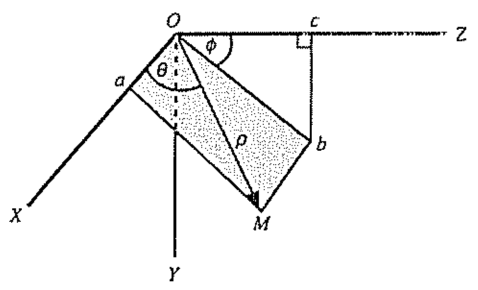


FIGURE 3.7. Comparison of the matching point algorithm with the step-by-step zero-bank algorithm and the flat earth algorithm. Results for different road slopes were averaged In this figure, a navigable reconstructed road is such that a 2-m-wide vehicle following its centerline will not cross the edges of the actual road. (a) Percentage of navigable reconstructions for various combined width and bank variations. (b) Navigable reconstructed road length.

obstacles for a land vehicle that is not specially equipped. Although material properties are important for determining navigability, they are not readily measured by current remote sensing devices on autonomous vehicles. Here, we make the simplifying assumption that regions can be adequately categorized by their geometry alone.

Given the presumption that obstacles will be defined by their shape, the next issue is how a region's geometry can best be determined by an autonomous vehicle. Perceiving geometry is essentially a depth perception problem. Approaches for creating depth (range) images generally fall into three



ρ = range
ϕ = vertical scan angle
θ = horizontal scan angle

FIGURE 3 8 Range image coordinate system.

categories: duplicating human visual ranging methods, contriving lighting methods, such as structured light sensors, and applying direct, active ranging technologies.

Direct range sensing methods do not provide insight into human visual understanding, but they are superior to indirect methods for creating fast, accurate range images. The ALV project used the Environmental Research Institute of Michigan (ERIM) [15] range scanner. Figure 3.8 illustrates the spherical coordinate system (θ, ϕ, ρ) that naturally describes range images. The scanner, which was mounted on the ALV, approximately 9 ft above the ground, is at the origin (O) of the system. The positive Y axis points directly down toward the ground. The positive Z axis points out in the direction that the ALV is currently traveling. The length of the line segment OM is the range (ρ) to the point M.

The right triangle Ocb is in the YZ plane. Angle cOb forms the vertical scan angle ϕ. Each row in a range image is taken from a plane that contains the X axis and is ϕ degrees beneath the Z axis. The rectangle $OaMb$ is in this plane. Angle aOM forms the horizontal scan angle θ. Each column in a range image corresponds to a particular θ. This geometry results in the following relationships:

$$x = \rho \cos(\theta), \tag{3.6}$$

$$y = \rho \sin(\theta) \sin(\phi), \tag{3.7}$$

$$z = \rho \sin(\theta) \cos(\phi). \tag{3.8}$$

The 64 rows of the image are at equally spaced values of ϕ, and the 256 columns are at evenly spaced values of θ. An ERIM range image has a 30° vertical field of view in which ϕ goes from approximately 6° to 36°. The 80° horizontal field of view extends from a θ of 130° to a θ of 50°. Although the total magnitudes of the fields of view are fixed, the orientations can be altered either by internal controls in the scanner or by moving the external platform on which the scanner is mounted.

The ERIM scanner has a vertical sampling interval of 0.3125° and a horizontal sampling of 0.46875°. Since the laser beam has an angular divergence of 0.5°, a scene is densely sampled. This removes the need for sophisticated interpolation techniques, such as those proposed by Boult and Kender [3] or Choi and Kender [5] for sparse range data.

Multiple objects at various ranges may occur within the 0.5° solid cone that forms a single pixel's field of view. The signal that returns to the scanner will indicate a range that is a complex average of all the ranges encountered within the cone. For example, if half of a cone intercepts a tree and the other half travels on to the ground, then the returning signal would yield a value that is somewhere between the distance to the tree and the most distant ground that is within the cone. This is called the mixed pixel problem. A strategy for avoiding range errors resulting from mixed pixels is presented in Vealch and Davis [21].

Because of the ambiguity effect, output range values are all between 0 and 64 ft. They are quantized into three-in. units, so that the final output of the ERIM scanner is a 64×256 array of 8-bit values ranging from 0 to 255.

The fastest of the ALV obstacle detection algorithms, range differencing, simply subtracts the range image of an actual scene from the expected range image of a flat plane. While rapid, this technique is not very robust. Small errors in the orientation of the scanner or a mild slope in the land will result in false indications of obstacles. We propose using the first derivatives of the range with respect to the vertical and horizontal scan angles as an improved, fast obstacle detector.

3.4.1 The Range Derivative Algorithm for Obstacle Detection

When deciding if a surface is an obstacle or not, the pertinent feature is the change in height across the surface. If the change is too rapid, then the surface is unnavigable. A surface normal contains the necessary information on the change in height, but calculating surface normals is computationally intensive. The surface normal at a point is a function of $\partial y/\partial x$ and $\partial y/\partial z$. Simply calculating the slope, $\partial y/\partial z$, would provide significant information concerning a surface's navigability. However, computing the slope directly from a range image is not much easier than calculating a surface normal. What can be done very quickly, though, is finding $\partial \rho/\partial \theta$ and $\partial \rho/\partial \phi$. The following derivation first shows how $\partial \rho/\partial \phi$ can be closely linked to $\partial y/\partial z$ and then how $\partial \rho/\partial \theta$ can be a measure of $\partial y/\partial x$. Using our knowledge of how range derivatives reflect changes in height across a surface, we can then design a rapid obstacle detection algorithm.

The differential of a function $y(\phi, \rho, \theta)$ can be written as

$$dy = \frac{\partial y}{\partial \phi} d\phi + \frac{\partial y}{\partial \rho} d\rho + \frac{\partial y}{\partial \theta} d\theta. \tag{3.9}$$

If θ is held constant so that the $d\theta$ term is 0, then Equation (3.9) applied to Equation (3.7) gives

$$\Delta y = \rho \sin \theta \cos \phi \, \Delta\phi + \sin \theta \sin \phi \Delta\rho, \tag{3.10}$$

where the infinitesimal terms dy, $d\phi$, and $d\rho$ have been replaced by their finite Δ equivalents. In a similar fashion, Equation (3.8) can be differentiated to yield

$$\Delta z = -\rho \sin \theta \sin \phi \, \Delta\phi + \sin \theta \cos \phi \, \Delta\rho. \tag{3.11}$$

Dividing Δy and Δz yields

$$\frac{\Delta y}{\Delta z} = \frac{\rho \cos \phi \, \Delta\phi + \sin \phi \, \Delta\rho}{-\rho \sin \phi \, \Delta\phi + \cos \phi \, \Delta\rho} = \frac{[(\Delta\rho/\rho)(\tan \phi/\Delta\phi) + 1]}{[(\Delta\rho/\rho)(1/\Delta\phi) - \tan \phi]}. \tag{3.12}$$

If ϕ is held constant, then Equation (3.9) becomes

$$\Delta y = \sin \theta \sin \phi \, \Delta\rho - \rho \sin \phi \cos \theta \, \Delta\theta, \tag{3.13}$$

and Equation (3.6) can be differentiated to obtain

$$\Delta x = \cos \theta \, \Delta\rho - \rho \sin \theta \, \Delta\theta. \tag{3.14}$$

Dividing Equation (3.13) by Equation (3.14) and regrouping yields

$$\frac{\Delta y}{\Delta x} = \frac{[(\Delta\rho/\rho)(\tan \theta/\Delta\theta) \sin \phi - \sin \phi]}{[(\Delta\rho/\rho)(1/\Delta\theta) - \tan \theta]}. \tag{3.15}$$

Excluding the terms in Equations (3.12) and (3.15) that we know a priori, we see that the changes in height in the x and z directions are a function of $\Delta\rho/\rho$. If we used some approximation of ρ, we would have a direct relationship between the easily calculated $\Delta\rho$ for a fixed θ or ϕ at a pixel and the slopes at that pixel. Our experiments with real range data suggest that the following is an adequate approximation:

$$\rho \approx H/\sin \theta \sin \phi, \tag{3.16}$$

where H is the height of the range scanner above the ground. Equation (3.16) comes from substituting H for y in Equation (3.7). In hilly terrain, this approximation is probably not adequate, but it works well for many scenes, and a table will show that the derivative algorithm that uses this approximation is less sensitive to orientation errors than other algorithms of similar simplicity and speed.

Using Equation (3.16), we can calculate what $\Delta\rho$ would be at each pixel if the slopes were zero. The difference between this predicted $\Delta\rho$ and the actual $\Delta\rho$ found in a range image is a measure of the actual slope. Large differences between predicted and actual $\Delta\rho$'s will be formed by edges of objects as well as surfaces with steep slopes. Thresholding the absolute values of these differences yields pixels that are likely to be on obstacles.

One could, of course, simply threshold the actual $\Delta\rho$'s without first subtracting the expected $\Delta\rho$'s and assume that large $\Delta\rho$'s indicate surfaces that

TABLE 3.1. Comparison of obstacle detection algorithms for sensitivity to rotation errors.[a]

Perturbation	Magnitude of errors θ	ϕ	Height	Range
3 degree horizontal	0.8	1.5	5.1	24.7
	0.3	0.4	1.6	6.4
3 degree roll	3.7	13.6	21.2	103.3
	1.1	2.8	6.0	23.1
3 degree vertical	1.1	17.3	25.4	123.7
	0.3	13.8	25.4	98.5
3 degrees in each	9.7	53.5	72.2	351.4
	2.6	21.3	38.9	150.7

[a] © 1990 IEEE

have steep slopes and hence are not navigable. This approach, however, would severely reduce one's capability to detect obstacles. A perfectly flat surface will yield a $\Delta\rho$ of about 10 if it is 60 ft away, but the same surface at a range of 10 ft only has a $\Delta\rho$ of about 0.3. This wide range in $\Delta\rho$'s leaves any thresholding algorithm in a bind. Small threshold levels would find nearby obstacles, but more distant flat surfaces would be falsely labeled as obstacles. Conversely, larger threshold levels would hide significant obstacles that are near the range scanner. What is needed is a variable threshold setting. This approach points out another way of looking at the range derivative algorithm: we are, in essence, creating a variable threshold that changes across an image based on expected $\Delta\rho$'s. While this simplistic view is a useful description, the derivative algorithm is founded on the mathematical relationships between $\Delta\rho$ and a surface's slopes and is not a randomly chosen heuristic for setting variable threshold levels.

Table 3.1 reveals the advantages of this approach over height or range prediction. The table contains two entries for each combination of algorithm and perturbation of road orientation (horizontal, vertical, and roll). The top entry is the largest absolute value in the entire image and represents a worst case scenario. In many scenes, however, the road will be near the center of the image's horizontal field of view, and large errors at the periphery are not critical. This is captured by the bottom entry, which is the largest absolute error within the central 30° of the image.

Several important trends emerge from Table 3.1. The ϕ derivatives were insensitive to all four rotational perturbations. When the entire image was considered, the maximum ϕ errors for each rotation were always at least 25% less than the maximum height difference errors. Within the central 30° of the horizontal field of view, the maximum ϕ derivative errors were 45–75% less than the maximum height errors. The range difference algorithm was very sensitive to all rotations. In several instances, the range difference errors were a full order of magnitude larger than the derivative errors. These results clearly show that the derivative algorithms are more robust under rotational uncertainties than either the height difference or range difference algorithms.

3.5 Conclusion

This paper provided a summary of the research conducted at the University of Maryland on problems associated with visual navigation of ground vehicles. We focused on algorithms for three modules of the system, the scene model planner, the road reconstruction module from video images, and the obstacle detection module. One of the roles of the scene model planner is to intelligently locate windows of focus of attention in video images. Strategies are represented explicitly as rules, which are easy to modify and extend. For example, some rules would locate the windows farther apart in the image when the road is straight than when it is curved. The road reconstruction module uses points detected in these windows as road edges to build three-dimensional road models. Image points corresponding to endpoints of cross segments of the road are found, and these cross segments are reconstructed. A dynamic programming algorithm weighs the mutual consistency of the cross segments and rejects the cross segments that do not contribute to a consistent road. The obstacle detection module uses images from the vehicle range scanner. It detects local changes of heights in the scene by directly interpreting gradients of range along the range image rows and columns. This method is more direct than approaches based on computing surface normals in the scene and is robust under rotational and vertical perturbations.

Over the past few years, our research has shifted from road navigation to more general navigation problems, with a long-term practical emphasis on cross-country navigation. This has led us to consider many new and interesting problems in the design and organization of autonomous systems. Preliminary results of our research on two navigation systems—RAMBO and Medusa—can be found in Davis et al. [6] and Aloimonos [2].

References

[1] Allen, E. (1983). "YAPS: Yet Another Production System." University of Maryland Computer Science Technical Report 1146, December.

[2] Aloimonos, J. (1990). "Purposive and Qualitative Active Vision." *Proc. Image Understanding Workshop. September 1990.*

[3] Boult, T. B., and Kender, J. R. (1985). "On Surface Reconstruction Using Space Depth Data." *Proceedings Image Understanding Workshop. Miami Beach. Florida, December 1985*, 197–208.

[4] Brooks, R. (1985). "A Robust Control System for a Mobile Robot." *IEEE Transactions on Robotics and Automation* **2**, 14–23.

[5] Choi, D. J., and Kender, J. R. (1985). "Solving the Depth Interpolation Problem with the Adaptive Chebyshev Acceleration Method on a Parallel Computer."

Proceedings: Image Understanding Workshop. Miami Beach, Florida. December 1985, 219–223.

[6] Davis, L S., DeMenthon, D., Bestul, T., Harwood, D., Srinivasan, H V., and Ziavras, S (1989). "RAMBO: Vision and Planning on the Connection Machine." *Proc. Image Understanding Workshop. May 1989.*

[7] DeMenthon, D (1987). A Zero-Bank Algorithm for Inverse Perspective of a Road from a Single Image." *IEEE International Conference on Robotics and Automation*, 1444–1449.

[8] DeMenthon, D (1988). "Reconstruction of a Road by Matching Edge Points in the Road Image." University of Maryland Center for Automation Research Center Technical Report 368, June.

[9] DeMenthon, D., and Davis, L. S., (1980). "Reconstruction of a Road by Local Image Matches and Global 3D Optimization." *IEEE International Conference on Robotics and Automation*, 1337–1342.

[10] Dickinson, S., and Davis, L (1990). "A Flexible Tool for Prototyping ALV Road Following Algorithms." *IEEE Transactions on Robotics and Automation* **6** (2), 232–242.

[11] Dickmanns, E. D., and V Graefe, V (1988). "Dynamic Monocular Machine Vision" and "Applications of Dynamic Monocular Machine Vision." *Machine Vision and Applications*, International Journal, Springer-Verlag International, Berlin and New York.

[12] Kanatani, K., and DeMenthon, D (1988). "Reconstruction of 3D Road Shape from Images: A Computational Challenge." Center for Automation Research Technical Report CAR-TR-413, December.

[13] Kanatani, K., Watanabe, K (1989). "Computing 3D Road Shape from Images for ALV: A Challenge to an Ill-Posed Problem." Computer Science Technical Report CS-89-5, Gunma University, June.

[14] Kanatani, K., and Watanabe, K. (1980) "Reconstruction of 3D Road Geometry from Images for Autonomous Land Vehicles." *IEEE Transactions on Robotics and Automation* **6** (2), 127–132.

[15] Larrowe, V (1986). "Operating Principles of Laser Ranging, Image Producing (3D) Sensors." Environmental Research Institute of Michigan, Ann Arbor, March.

[16] Liou, S., and Jain, R. (1985). "Detecting Road Edges Using Hypothesized Vanishing Points." Robot Systems Technical Report 18-85, Department of Electrical Engineering and Computer Science, University of Michigan.

[17] Morgenthaler, D., Henessy, S., and DeMenthon, D (1990). "Range-Video Fusion and Comparison of Inverse Perspective Algorithms." *IEEE Trans. on Systems. Man. and Cybernetics*, Special Issue on Unmanned Systems and Vehicles, November–December.

[18] Ozawa, S., and Rosenfeld, A. (1986). "Synthesis of a Road Image as Seen from a Vehicle." *Pattern Recognition* **19**, 123–145.

[19] Thorpe, C., Hebert, M., Kanade, T., and Shafer, S. (1988). "Vision and Navigation for the Carnegie–Mellon Navlab." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**, 362–373.

[20] Turk, M., Morgenthaler, D., Gremban, K., and Marra, M. (1988). "VITS–A Vision System for Autonomous Land Vehicle Navigation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**, 342–361.

[21] Veatch, P., and Davis, L. (1990). "Range Imagery Algorithms for the Detection of Obstacles by Autonomous Vehicles." *Computer Vision, Graphics and Image Processing* **50**, 50–74.

[22] Waxman, A., LeMoigne, L., Davis, L., Srinivasan, B., Kushner, T., Liang, E., and Siddalingaiah, T. (1987). "A Visual Navigation System for Autonomous Land Vehicles." *IEEE Transactions on Robotics and Automation* **2**, 124–142.