Progress in Robot Road-Following

R. Wallace, K. Matsuzaki, Y. Goto,
J. Crisman, J. Webb, T. Kanade

Robotics Institute, Carnegie-Mellon University

Abstract

We report progress in visual road following by autonomous robot vehicles. We present results and work in progress in the areas of system architecture, image rectification and camera calibration, oriented edge tracking, color classification and road-region segmentation, extracting geometric structure. and the use of a map. In test runs of an outdoor robot vehicle, the Terregator, under control of the Warp computer, we have demonstrated continuous motion vision-guided road-following at speeds up to 1.08 km/hour with image processing and steering servo loop times of 3 sec.

1. Introduction

Research in robot navigation on roads is part of the Autonomous Land Vehicle Project (ALV) at Carnegie-Mellon University. Broadly, our work is aimed at creating autonomous mobile robots capable of operating in unstructured environments. To this end, our research program involves a variety of sensors, programs and experimental robot vehicles. This paper is focused on recent progress in detection of and navigation on roads, using a TV camera as our sensor and a six-wheeled outdoor autonomous robot, the Terregator [7], as our test vehicle. We present results and work in progress in the areas of system architecture, image rectification and camera calibration, oriented edge tracking, color classification and road-region segmentation, extracting geometric structure. and the use of a map.

For robot navigation of roads, we use a single television camera as our primary sensor. In this application, the monocular TV camera is considered superior to ranging sensors such as laser scanners or sonar for three reasons. First, roads we are interested in following do not necessarily have prominent 3-dimensional features at their shoulders; most often there is no depth discontinuity between the road surface and the surrounding roadside. Second, we have developed one steering strategy that servos the vehicle based on measurements in the image plane itself, rather than on measurements in a world coordinate frame. Third, we have so far relied on a *local ground plane* assumption, that the ground around the vehicle is locally planar, so that any time we *do* need to transform image points to world coordinates, the transformation is trivial.

To attain the broad goals of our project, we have split the research into two efforts. The goal of the first effort is to develop a road-following system which uses a map to navigate around a highly structured and visually simple network of sidewalks on the CMU campus. The goal of the second effort is to develop vision routines for road-following in a less structured and visually more complex environment in a nearby park.

[1]Currently, this project is funded in part by Carnegie Mellon University, by the Office of Naval Research under contract number N00014-81-K-0503, by the Western Pennsylvania Advanced Technology Center, by Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539, and by Denning Mobile Robotics, Inc. Richard Wallace thanks NASA for supporting him with a NASA Graduate Student Researchers Program Fellowship Grant.

2. Sidewalk Navigation

The sidewalk environment at CMU is a network of mostly straight concrete pathways joined at intersections of various shape. The sidewalks have fairly uniform color and texture and are always surrounded by well-groomed grass, giving them consistent high-contrast edges. The goal of our research in this environment is to develop algorithms for geometric reasoning, shape-matching and navigation with a map.

2.1 Map and Blackboard

The overall system architecture to which a vision-based road-following subsystem interfaces is a *blackboard* [5], a shared memory structure containing a local map of the robot's environment. Other sensing processes, such as those interpreting range data, and other knowledge-based processes, such as those updating the local map, are also tied to the blackboard.

2.1.1 Dialogue Model

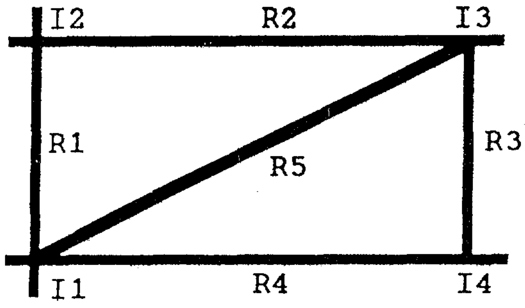
The road-following subsystem consists of four modules; Vision, Map, Navigator, and Motion Control. These modules communicate with each other by sending and receiving tokens through the Blackboard. In selecting this decomposition of our system into modules, we followed the principle of *information hiding*. The Vision module contains expertise needed for extracting features from images. The Map module knows the structure of the robot's environment and its position. The Navigator is responsible for planning paths. The Motion Control module insures that the vehicle executes navigation commands. Thus each module has a different domain of expertise. For example the Vision module does not know the robot's map or route. That information is kept hidden and is used only by the Map module to make predictions to the Vision module.

Communication between the various modules looks like a dialogue. Figure 1 shows the dialogue model of the road-following subsystem. This model reflects the information hiding principle of the design. In the example, the Map hides information from the vision module, except for the facts which are relevant for the current scene. The Map tells the Vision module only about the *predictions it makes for the current scene.*

With map data, the Map module produces the token, Predicted Object, which shows what the Vision system shall see. For example, a Predicted Object can be a road or an intersection. Using Predicted Object, Vision sees and makes the token, Detected Object, which shows the shapes of objects in front of the vehicle. Using Detected Object, the Map decides the vehicle's Current

CH2282-2/86/0000/1615$01.00 © 1986 IEEE

Position. Using Current Position and the map data, the Navigator supplies the token, Motion Command, which tells how to drive the vehicle. Using Motion Command, the Motion Control drives the vehicle.



User: Robot is at road R1, 3 meters from I1.
Navigate to R3, 2 meters from I3.

Map: Vision will see straight road and cross-type intersection. The color in the left is ... Detect them.

Vision: Ok. I found them. Their shapes are ...

Navigator: Drive on it 2.5 meters and turn to right 90 degrees.

Motion Contol: Ok. I drive. (vehicle moves)

Map: Vision will see straight road. The color on the left is ... Detect it.

Figure 1: Dialogue Model of Map Interface

In the road-following subsystem, two kinds of coordinate systems, World Coordinate and Vehicle Coordinate, are used. World Coordinate is an absolute coordinate. The map data is written with World Coordinate. The Vehicle Coordinate frame, which is fixed on the vehicle, is used by Vision to represent Detected Object, because it does not know where the vehicle is. Coordinate transformation is done when necessary.

2.1.2 Predictions

The map module supplies *predictions* to the vision module. The map data consists of two kinds of maps, a topological map and a geometrical map. The topological map stores the topology of roads and intersections. The geometrical map stores the shapes of roads and intersections.

With these map data, the Map predicts the kinds, the shape and the image features of objects which shall be seen in a camera view. The purpose of detecting objects is to navigate the vehicle. The detail of an object shape is trivial and therefore, not necessary for navigation. The Map creates *interest segments*, which are the primary edge line segments of roads and intersections. The interest segments are enough for Map to decide the vehicle's Current Position and the object shape necessary for navigation. They are likely to be the edge segments most easily detected by Vision, and therefore are included in the Predicted Object. An interest segment is also a key for matching. We discuss this in detail below.

2.2 Extracting Geometric Structure

Our Autonomous Land Vehicle has to not only follow single road, but also to detect an intersection and turn into one of the intersecting roads. In this case accurate shape of roads and an intersection has to be extracted. This is difficult because variations in camera view and imaging conditions result in variations in the shapes detected. Furthermore there are many factors which make it difficult to detect a road shape, such as cracks, dust, gaps between concrete slabs. They are not noise but physical substance, therefore even if region classification is done perfectly, they possibly remain. To solve these problems, we implemented two procedures. First, the image is processed to eliminate these disturbing factors and to reproduce the road region. After that, using knowledge from map, interest segments, which are key to decide an position of an intersection, are found.

2.2.1 Reproducing the Road Region

To eliminate the disturbing factors, two phase image processing is done; extracting high-confidence road regions and then connecting them.

The result of region segmentation includes four types of segments: 1)actually road and classified as road, 2)actually not road and classified as not road, 3)actually road but classified as not road, 4)actually not road but classified as road. At the first image processing phase, the program selects a conservative classification threshold so that only ideal road surface is classified as road. This result includes much type 3 region but little type 4 region, and region classified as road is confidently road. Then, to cover type 3 region, we did a combination of reducing resolution and expansion/contraction of image.

The expansion/contraction method is known as a good method to eliminate gaps or small holes, but calculation time is long when the size of defects are large and large number of expansion/contraction is needed. We have to use this method in real time during vehicle running. So, we reduced resolution before expansion/contraction. This method absorbs several pixels into one pixel, and decides the the new pixel value by a threshold on the proportion of original pixels classified as road to nonroad. We use a reduction ratio of 8*8 to 1 pixel followed by 1 or 2 iterations of expansion/contraction. This obtained both sufficient shape estimates and quick calculation.

2.2.2 Polygon Fitting

To recognize an intersection from the reproduced shape, we fit a polygon to the intersection contour. Shape analysis based on polygon is much quicker than one based on whole pixels or run-length data. The processing includes following steps.

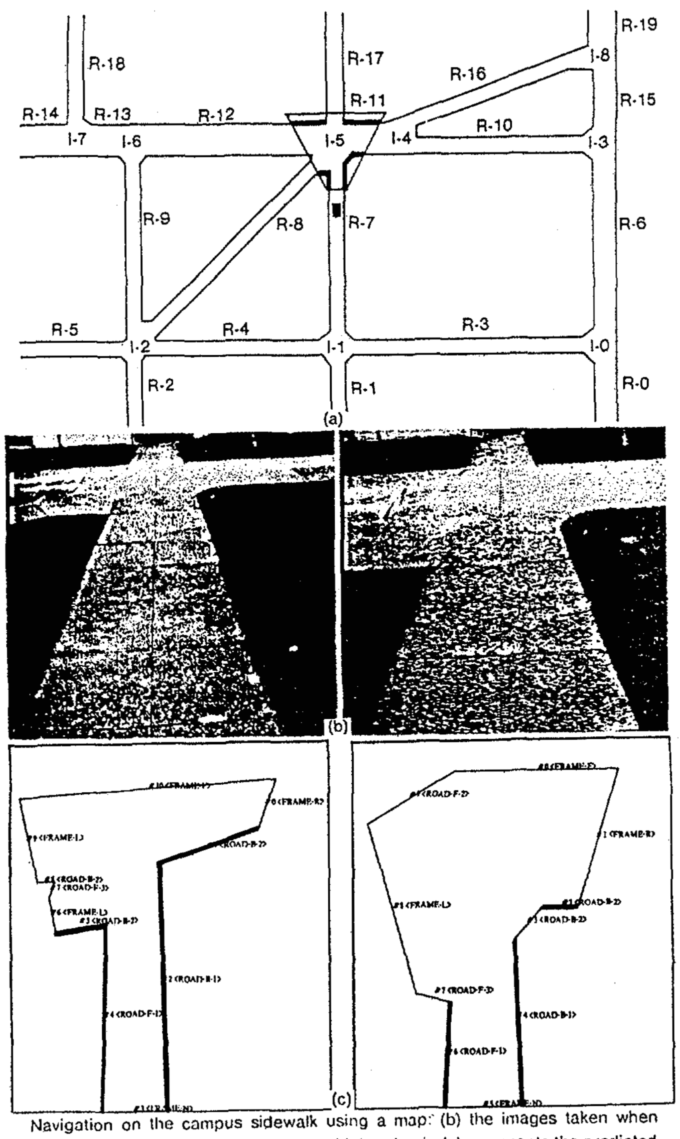
1. **Extracting Straight Line.** Most of roads imaged are straight but if they include curves, these can be represented as a set of segmented straight lines. So, we apply a polygonal approximation to original precise polygon to extract major straight components. The tolerance is set so that the interest segments can be picked up well.

2. **Labeling Lines.** We have developed a program which labels lines. At first,this program identifies viewing frame edge lines by searching lines which are contained in the Coordinate of viewing frame. Second, this program classifies lines by angle and gives same labels for the similar angle lines. The Map module produces also the description of interest segments which shows the segment attribute and the relationship between segments. Using this description, this program can match the classified lines to the predicted interest segments easily. The list showing the detected segments and their correspondence to the predicted is returned to the Map module. Understanding of whole geometric structure is done by the Map in next map matching step.

2.2.3 Map matching

With the result of the Vision module and the object prediction, the Map module can know the names and the shapes of the detected objects. In order to estimate the vehicle current position, the Map module selects crossing lines in the detected objects and corresponding lines in the map data, and calculates coordinates transformation which can match them. In this stage, when only straight portion of the road is in the view frame, the measurement from the Vision module' can constrain the vehicle position and orientation only perpendicular to the road. In such case, the location along the road is calculated using the vehicle motion. The positional error which might accumulate along the path will be corrected as the vehicle approaches to the intersection and can see the road edges in multiple orientations.

Figure 2 shows a result of CMU campus sidewalk run. Along the vehicle approaches an intersection, the vision module detects different parts of road contour which are predicted as major line segments by the map module.



Navigation on the campus sidewalk using a map: (b) the images taken when approaching intersection I-5. The trapezoidal region in (a) represents the predicted view of the Vision. (c) the results of road region extraction of the images in (b). The images are rectified into the map coordinates from the image coordinates. The edges matched with prediction are indicated by bold lines.

Figure 2: Navigation on Campus Sidewalk using Map

3. Park Road Following

Our park environment contains a 1 kilometer curving asphalt path part of which is always illuminated directly and part of which is shaded by trees. The path itself varies in texture from mostly smooth and featureless to cracked and pot-holed, and in color from blue-gray to black. The shoulder around the path consists mostly of grass, but there are also some sections of dirt and rock. Seasonally, both road and shoulder are obscured by leaves, snow or ice. Trees and their shadows are also present. The main goal of our research in the park environment is to develop vision algorithms capable of steering the vehicle reliably in this unstructured environment.

3.1 Road-Edge Following

We have developed a technique for tracing the edges of a road using an oriented edge detector. Like the tracker discussed in [9] our algorithm begins with an estimate of the start position from which is the edge is to be traced. Unlike that tracker, ours integrates or smooths the edge along the edge direction. Integrating the signal along the direction of the edge has the effect of smoothing and reducing noise content. Then, the position of the edge is localized by matching an ideal step edge model with the one-dimensional cross-section.

Oriented edge detection operators have been explored in computer vision, with perhaps the best results found in [2]. We chose an oriented operator since it is more reliable than an unoriented one. For example, if the road in the image is oriented at 45 degrees, then a conventional edge detector will find gradually sloping intensity values, see figure 3. However, if the same detector is oriented at 45 degrees, then the oriented detector would see a sharp change in intensities, and therefore, the edge location is detectable. We have implemented edge operators at a number of different orientations so that we can obtain a reliable response regardless of the orientation of the road in the image.

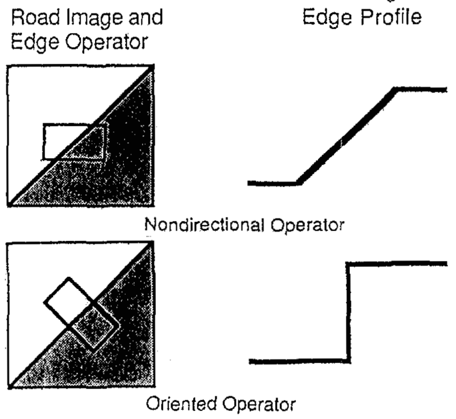


Figure 3: An Oriented Edge Operator

3.2 Implementation

The edge tracer constructs a list of road edge points in an image given a position (r_0, c_0) and orientation, θ_0 of a road edge. The oriented edge operator integrates the signal along its columns. If the operator does not align with the image columns, then it selects pixel values nearest to the position of its columns for the summation. This one dimensional result of the edge operator is called the *edge signature or edge profile.*

Then a new road edge point, (r_p, c_p), is predicted to lie a distance from (r_0, c_0) at an angle of θ. A search window is created centered at (r_p, c_p), oriented at the angle θ. The edge operator creates an edge profile in the search window. The road edge, (r_i, c_i), is determined to be where the an ideal step edge and the window profile have the best correspondence. The orientation of the road is recalculated by θ = arctan2$(c_i - c_{i-1}, r_i - r_i)$. This algorithm is iterative if $(r_i, c_i) \rightarrow (r_{i-1}, c_{i-1})$. This process is repeated until the search window falls outside of the image bounds.

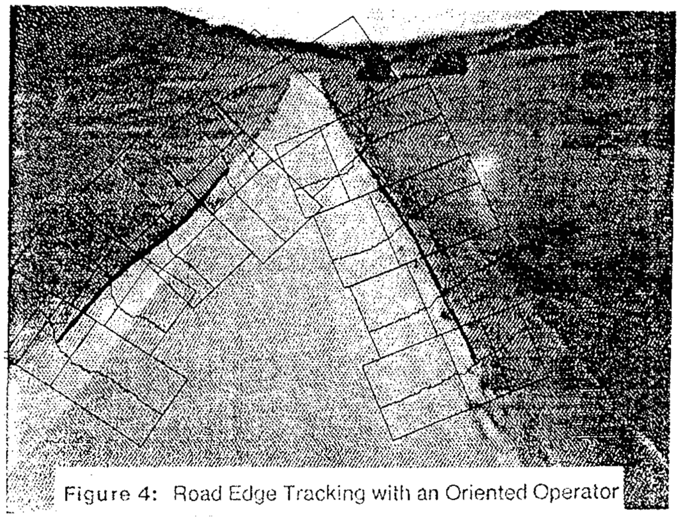


Figure 4: Road Edge Tracking with an Oriented Operator

3.2.1 Results

The edge tracer has been tested on 480 X 512 grey level images. The dimensions used for the search window were 64 rows by 128 columns. Figure 4 shows a typical result of the edge tracer. The initial position is given near the bottom of the image and the oriented edge detector proceeds upward in the image. The larger boxes outline the search windows, and the smaller, inner boxes show the positions of best correlation. The edge profiles are shown inside the search windows.

We have developed a vehicle driver system based on oriented edge tracing. The initial position and orientation of the left and right road edges are input to the system and used for the first iteration of the oriented edge tracer. After finding the road edges in the image, they are back-projected to the ground plane. The vehicle motion between images is used to locate the previously found road edges relative to the vehicle. Then the previous edges are projected in the new image. These edge locations are used for the position and orientation estimations required for the edge tracer. The 3D projection of the road edges also allow the right and left road edges to be tested for parallelism and proper separation.

This system works well on images where there is a fair amount of contrast between the road edge and the road shoulder. We have been able to drive our vehicle quite reliably on gently curving roads. However, we have had difficulty when the edge of the road lies close to obstacles or when shadows lie on the road. The edge tracer can locate a road edge point in under one second. The system can drive the vehicle at speeds up to 0.3 meters/sec.

We are currently working on testing the road edges found by the edge tracer for geometrical consistency. If the right and left edges of the road are not parallel and the proper width apart, then the system must decide which edge should be used to drive the vehicle. Measures of evaluation based on the height, width, smoothness, and consistency are currently being tested. If these measures are reliable, the system should be able to evaluate its performance.

3.3 Road-Region Segmentation

The second major approach to road feature detection is region segmentation. This differs from the edge-based procedure in that the road itself is extracted, rather than its contours. As we mentioned earlier, the edge information can be used to verify and localize the region hypothesis. Region classification is based on assignment of region labels to all pixels in an image, where the assignment depends on properties of that pixel such as brightness, texture and color around that pixel. Our work is focused on color classification.

3.4 Color

Early in our work on visual detection of roads we recognized the importance of utilizing color vision sensors. We found in black-and-white images of our test site that the perceived intensity of the asphalt road differed very little from the intensity of the surrounding grass, although the color was very different. Gray-level histograms of the images were either very flat, or they had peaks caused by shadows and highlights rather than road or nonroad features. Histogram-based segmentation techniques and edge operators failed for the same reason. We considered texture energy measures to segment road and grass, since the grass has more edges per unit area, but the noise introduced into the images by an inferior TV transmission system confounded attempts to measure high-frequency texture information. Even in the presence of high spatial frequency image noise color information is retained.

3.4.1 Pixel Classification

In color images each pixel (x, y) has an associated color vector $(R(x, y), G(x, y), B(x, y))$. The set of all possible (R,G,B) values forms a color cube RGB. The RGB cube can be divided in various ways so that pixels having certain color vector values can be classified as road or nonroad. A simple region classification involves selecting a sample road region and grass region from a training image, and using the average values $(\mu R_{road}, \mu G_{road}, \mu B_{road})$ and $(\mu R_{grass}, \mu G_{grass}, \mu B_{grass})$ as ideal feature points in RGB space. If the covariance matrices Σ_{road} and Σ_{grass} are also measured then the colors can be modeled as trivariate normal distributions (TVNDs). The result of a TVND model is to divide color space into regions separated by quadratic surfaces. Figure 5 shows a result of classifying a sequence of rectified road images from the park site.

3.4.2 Color variation

Unfortunately the color of road and shoulder do not remain constant from one image to the next. Variation in color arises for a variety of reasons, such as illumination changes (e.g. shadow versus direct illumination) and material changes (e.g. dry asphalt versus wet, green grass versus yellow). Additionally, our test vehicle is equipped with a TV broadcast station, through which images are transmitted to a fixed-based computer. The chromatic component of the TV signal varies depending on such factors as the position of the robot vehicle with respect to the TV receiver.

We have begun to explore the use of adaptive color models to reduce the problems arising from color variation.

3.4.3 Shadows and normalized color

Shadows cause many of the failures of our vision system. Edge-based schemes for detecting road edges are fooled by high-contrast shadow edges, as shadow edges often have a greater brightness-to-darkness ratio than material edges. Even region classification schemes based on color are confounded by shadows

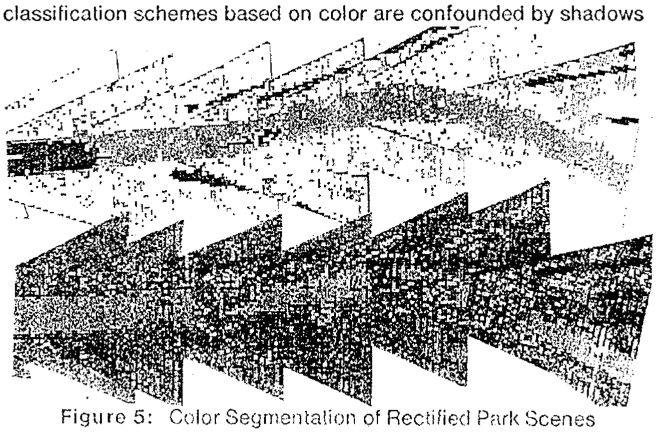


Figure 5: Color Segmentation of Rectified Park Scenes

because images of objects in shadow contain color values clustered around different points in *RGB* space.

Consider an object imaged with color C_1 in a sunlit part of the scene and color C_2 in a shadowed region. To a first approximation, $C_1 = kC_2$ for some constant k. This is because the object reflects the same color in shadow, it is just imaged at a different intensity. Thus a preprocessing step is to *normalize* all the color vectors of an image, by transforming each point $(R(x, y), G(x, y), B(x, y))$ into $(r(x, y), g(x, y), b(x, y))$ such that

$$r = R/(R+G+B), \quad g = G/(R+G+B), \quad b = B/(R+G+B).$$

Then all the color points lie on the plane $R+G+B=1$.

Although the transformation from *RGB* to *rgb* is sufficient for erasing shadows in many cases, it is not always successful. There are two factors limiting its usefulness. First, the dynamic range of a TV camera is not very large (a maximum brightness:darkness ratio of 7:1) compared with film (a maximum brightness:darkness ratio of 20:1) or the human eye (a maximum brightness:darkness ratio of at least 1000:1). Thus TV images containing of shadowed regions may have splotches of maximum bright or dark, in which all spatial detail and color information is lost. Color normalization will not work in these areas. The second factor is less important, but easier to work around. Nonshadow areas in our outdoor road scenes are illuminated by direct sunlight, which has a more-or-less constant spectral distribution. Shadowed regions are illuminated by skylight and by sunlight reflected off surrounding objects (such as tree leaves and tree trunks in our case). Thus the reflected color of a shadowed part of a region is not quite the same as the color reflected from that part of the region in direct sunlight. In practice the difference is small enough not to matter for our classification techniques.

Color normalization reduces the dimensionality of color classification to two, in which case a bivariate normal distribution is used as a color feature model.

3.5 Image Rectification

We have implemented programs for nonlinear warping of an perspective of a road to transform it into a view like what we would see if we were flying over the road and looking down on it. This transformation, called *image rectification*, produces a map-like image in which the structure of the road is made explicit. The result is an image which is in vehicle coordinates and can be used for camera calibration, debugging of ground-plane operations, detection of ground-plane features, and display of planned robot paths.

3.5.1 Definition

Figure 6 shows the process of image rectification. It is most easily described by considering a rectangular grid projected onto the ground plane. Grid points can be considered as pixels of the rectified image. Rectification consists of back-projecting the grid-points in the ground plane to the original image, in order to see what intensity value should be placed at that point. Once the back-projection is computed, it is stored as a lookup table so that subsequent images can be rectified quickly.

Figure 7 shows the process of image rectification for a wide-angle fish-eye lens. This lens is superior to a standard reflex lens (which we usually model as a pin-hole) for imaging the road, because the road always remains in view even when the vehicle makes sharp turns off the centerline. The point $(-1, i_A, j_A)$ on the ground plane is first projected onto the unit sphere centered at the origin, then perpendicularly to the image plane which is tangent to the sphere at $(0, 0, 1)$. The overall transformation is

$$(i_C j_C) = (-1, i_A)/\sqrt{1+i^2_A+j^2_A}$$

where A is the rectified image and C is the original image.

This transformation is more useful if it can be done quickly: we anticipate carrying out this transformation on the CMU Warp

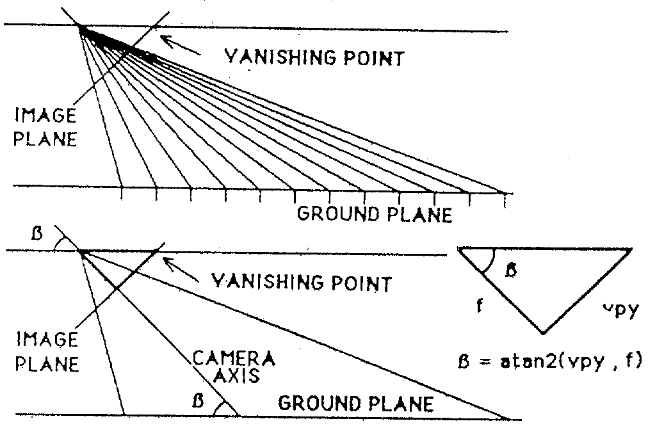


Figure 6: Image Rectification for Pin-Hole Lens and Determination of Camera Tilt

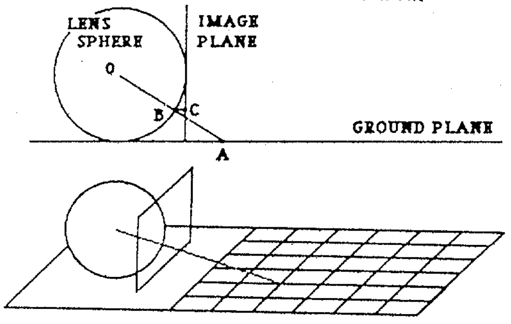


Figure 7: Image Rectification for Fish-Eye Lens

3.5.2 Camera calibration

The image rectification process (for the pin-hole lens model) can be used for camera calibration. By "camera calibration" we mean deriving the necessary parameters for transforming image points to the local ground plane around the vehicle. By intersecting a pair of lines in the ground plane around the vehicle a point on the horizon (vanishing line) can be detected. Note that the *actual* horizon need not be in view, only a pair of lines in the local ground plane. In fact, the lines need only lie in any plane parallel to the ground plane, except the planes containing the camera axis. In practice we use a pair of forward-pointing straight metal poles bolted to the side of the Terregator as a calibration "hood ornament". We hand-select these points from a calibration image.

Once the horizon line is known, the tilt of the camera is easily derived as in figure 6 Given the tilt β of the camera and an estimate of the camera focal length f, the transformation from ground plane points to image points is obtained directly as in figure 6.

A second aspect of camera calibration is determining the x and y scale factors for the image, where x indicates distance along an axis parallel to the vehicle forward direction and y is distance along an axis parallel to the wheel rotation axes. To measure these parameters, we place meter sticks on the ground plane in camera view, digitize and rectify a test image, and then measure the lengths of the meter sticks along the x and y dimensions.

3.6 Warp Runs

In test runs of an outdoor robot vehicle, the Terregator, under control of the Warp computer, we have demonstrated continuous motion vision-guided road-following at speeds up to 1.08 km/hour with image processing and steering servo loop times of 3 sec.

3.6.1 Warp Hardware Description

The Warp machine has three components: the Warp processor array, or simply Warp, the interface unit, and the host, as depicted in Figure 8. We describe this machine only briefly here; more detail is available separately [1]. The Warp processor array performs the bulk of the computation:in this case, low-level vision routines [2]. The interface unit handles the input/output between the array and the host. The host has two functions: carrying out high-level application routines and supplying data to the Warp processor array.

The Warp processor array is a programmable, one-dimensional systolic array, in which all cells are replicas of each other. Data flow through the array on two data paths (X and Y), while addresses and systolic control signals travel on the Adr path (as shown in the Figure 8). The Warp cells are specialized for floating-point operations. The data path of a Warp cell is depicted in Figure 9. Each cell contains two floating-point processors: one multiplier and one ALU [8]. These are highly pipelined; they each can deliver up to 5 MFLOPS each. This performance translates to a peak processing rate of 10 MFLOPS per cell or 100 MFLOPS for a 10-cell processor array. To ensure that data can be supplied at the rate they are consumed, an operand buffer is dedicated to each of the arithmetic units, and a crossbar is used to support high intra-cell bandwidth. Each input path has a queue to buffer input data. A 4K-word memory is provided for resident and temporary data storage.

As address patterns are typically data-independent and common to all the cells, full address generation capability is factored out from the cell architecture and provided in the interface unit. Addresses are generated by the interface unit and propagated from cell to cell (together with the control signals). In addition to generating addresses, the interface unit passes data and results between the host and the Warp array, possibly performing some data conversion in the process.

The host is a general purpose computer. It is responsible for high-level application routines as well as coordinating all the peripherals, which might include other devices such as the digitizer and graphics displays. The host has a large memory in which images are stored. These images are fed through Warp by the host, and result images from Warp are stored back into memory by the host. This arrangement is flexible. It allows the host to do tasks not suited to Warp, including low-level tasks, such as initializing an array to zero, as well as higher level tasks, such as processing a histogram to determine a threshold.

3.6.2 Warp Road Following Algorithm

The Warp-implemented road following algorithm is very simple, but proved to be remarkably robust. The algorithm is region-based; it searches for the road as a bright region in the blue spectrum of a color image. A 100 x 512 band of the image is taken about halfway down the image. The algorithm then works as follows:

1. **Blue Filter.** The color image is filtered by digitizing only the blue band. Blue was chosen because blue is a strong component in the color of the roads we are driving on (asphalt and concrete), but less strongly a component of the background (generally grass).

2. **Edge-preserving smoothing.** This is a smoothing operation which avoids smoothing across edges. It is the algorithm EGPR in the Spider subroutine library [4], implemented on Warp. The algorithm takes a 5 x 5 window around each pixel and chooses nine subwindows in the 5 x 5 window. The subwindow with smallest variance is chosen, and the central pixel is replaced by the mean of this window. Two passes of this algorithm are executed. The effect is to remove noise from the image, especially noise from the poor quality of the TV reception in some cases.

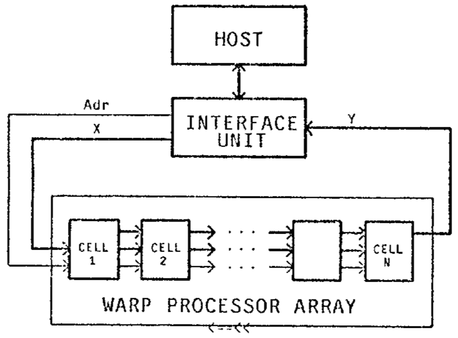


Figure 8: Warp machine overview

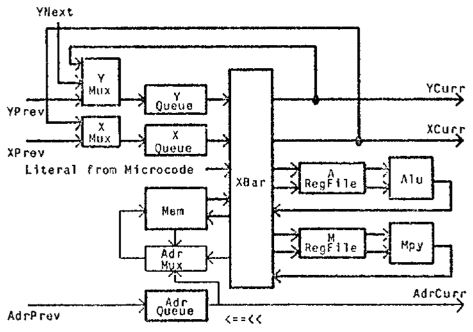


Figure 9: Warp cell datapath

3. **Histogramming.** A standard histogram is taken on the Warp machine.

4. **Threshold selection.** The histogram is used by the Sun 120 to select a threshold. The threshold is selected by starting at the 50th percentile level in the histogram and then finding a local minimum by comparing adjacent 3-element averages of the histogram.

5. **Binarization.** A gray value table translation table is constructed by the Sun using the threshold, and the image in binarized using this table on Warp.

6. **Region smoothing.** The resulting binary image is once again subjected to two passes of edge-preserving smoothing. The idea here is to remove small cracks in the road, and to eliminate small regions of ones in the background. Edge-preserving smoothing was chosen for this step instead of a more traditional operation, like shrinking and growing, because the edge-preserving filtering program was available while the (simpler) binary operator program was not.

7. **Blob detection.** At this point the road is a region of ones surrounded by a background of zeroes. Ten scan lines, taken ten rows apart, are taken from the image and each is examined to find the longest continuous sequence of ones. Each scan line thus defines a left and right road edge. The left and right edges are

averaged together individually to find the estimated road edges. An earlier approach was to find the left edge by finding the first long sequence of ones moving to the right from the left side of the image and the right edge similarly. This did not work as well as the second approach, since the vehicle tended to steer into the center of forks in the road.

8. **Steering.** Our servoing strategy is to steer the vehicle to keep the center of the road centered in the image. Basically we start with a large (512 x 512) image array and reduce it as quickly as possible to a point *(x, y)*. This is the point considered to be the center of the road some fixed distance in front of the vehicle. It is also the point to which the vehicle steers. Assuming that the center of the image is the point *(0, 0)*, the steering command is to turn left or right at some $d\tau/dt = \gamma x$ where γ is a gain constant related to the distance ahead imaged and to vehicle speed. $d\tau/dt$ is rate of turn of the vehicle (giving path curvature) in degrees per second. See [6]for details.

3.6.3 Hardware Configuration

In addition to programming an efficient road following algorithm on Warp, we have made improvements in our video transmission system and vehicle interface that have increased the reliability of our system and further reducted time between image digitizations. Time reductions between in the image processing cycle increase the servo rate of the vehicle steering control loop, and enable the vehicle to drive at higher speed.

We chose to digitize the image of the blue band only, in order to obtain the highest possible contrast between the test road and the surrounding grass in the image. Since grass absorbs almost all blue light and the asphalt road reflects a lot of blue light, the TV image in the blue band shows a very bright road surrounded by very dark grass. The blue filtering of the signal is tied to the particular road on which we are testing the vehicle. The next step in hardware configuration improvement is to selectively digitize the red, green and blue bands and to combine them using our Matrox frame buffers and the Warp.

4. Conclusion

We have presented a comprehensive view of a vision-based road-following system for an autonomous vehicle. Various parts of this system exist and have been tested both off-line on "canned" images and during real-time tests using the Terregator.

An overall picture of our system can be seen by considering the path of a single image through the entire processing loop. First, the Map module announces a set of predictions for the current scene, knowing the vehicle's position. The Vision module then dynamically applies color and texture segmentation techniques to extract the predicted road region. An oriented edge tracker uses the geometry of the extracted road region and the predicted interest segments to either localize the position of the road or reject the region and report failure. If road or intersection region detection is successful, the Navigator is alerted and generates a steering plan from the road region. If not successful, the Vision system halts and signals the blackboard so that another module (or person) to take control. The steering plan is received by the low-level motion control module, which interfaces to the vehicle's gyros and shaft encoders and executes the steering strategy. Timestamps on data carried through the entire system enable the vehicle to be controlled in real time, with old steering plans aborted as the Navigator creates new ones. To work for continuous motion road-following even at the slowest speed the Terregator has run in any road-following experiment (10 cm/sec) the entire processing loop must complete every 10 seconds.

Warp has proved to be a useful high-speed processor for vision tasks. An important advantage of Warp over other image processing computers is its floating-point capability. Many of the processes we have discussed, such as image rectification, color segmentation, and oriented edge tracking, are implemented as floating-point algorithms and can run efficiently on Warp. Using the Warp, we have already demonstrated one efficient and robust road-following algorithm.

5. Acknowledgements

We would like to thank the Civil Engineers Red Whittaker, Chuck Whittaker, Francois Bitz, Steve Berman and Kai Lee for developing and maintaining the Terregator vehicle. Thanks to Mike Blackwell and Kevin Dowling for hardware, system interface and video support. Chuck Thorpe managed much of the effort and Hans Moravec provided support and encouragement.

Currently, this project is funded in part by Carnegie-Mellon University, by the Office of Naval Research under contract number N00014-81-K-0503, by the Western Pennsylvania Advanced Technology Center, by Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539, and by Denning Mobile Robotics, Inc. Richard Wallace thanks NASA for supporting him with a NASA Graduate Student Researchers Program Fellowship Grant.

References

1. Arnould, E., Kung, H.T., Menzilcioglu, O. and Sarocky, K. A Systolic Array Computer. Proceedings of 1985 IEEE International Conference on Acoustics, Speech and Signal Processing, March, 1985, pp. 232-235.
2. Gross, T., Kung, H.T., Lam, M. and Webb, J. Warp as a Machine for Low-level Vision. Proceedings of 1985 IEEE International Conference on Robotics and Automation, March, 1985, pp. 790-800.
3. Kung, H.T. and Webb, J.A. Global Operations on the CMU Warp Machine. Proceedings of 1985 AIAA Computers in Aerospace V Conference, American Institute of Aeronautics and Astronautics, October, 1985.
4. Electrotechnical Laboratory. *SPIDER (Subroutine Package for Image Data Enhancement and Recognition).* Joint System Development Corp., Tokyo, Japan, 1983.
5. Stentz, A., Shafer, S., Thorpe, C. An Architecture for Sensor Fusion in an Autonomous Land Vehicle. forthcoming.
6. Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W., Kanade, T. First Results in Robot Road Following. Proceedings of IJCAI 85, August, 1985.
7. W. Whittaker. Terregator - Terrestrial Navigator. Carnegie-Mellon Robotics Institute, 1984.
8. Woo, B., Lin, L. and Ware, F. A High-Speed 32 Bit IEEE Floating-Point Chip Set for Digital Signal Processing. Proceedings of 1984 IEEE International Conference on Acoustics, Speech and Signal Processing, 1984, pp. 16.6.1-16.6.4.