Color Vision for Road Following *

Charles Thorpe and Jill Crisman
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213

March 14, 1988

Abstract

This paper discusses the research conducted at CMU in several topic areas: increasing the dynamic range of current cameras to better handle outdoor scenes, detecting roads in heavily shadowed scenes and in rapidly changing illumination, and identifying intersections. This research is necessary to add new capabilities to our road following vision system. We built a new system, using the results from our research, that could identify roads and intersections in shaded scenes. We also transferred some of our algorithms to the WARP, an experimental supercomputer developed at CMU, to test the capabilities of current parallel machines.

1 Introduction

Our long-term objective is to create color vision systems capable of finding and following roads under all conditions. Our ambitions include modeling and perceiving dirt roads, city streets, and expressways; handling sunny days as well as rain, snow, or dark of stormy night; and both driving on familiar, well-mapped roads and exploring new areas without the aid of a map.

In our navigation system, the vision module is just one of many modules. The vision module is responsible for detecting roads in the camera images. The pilot module uses the road position to plan a path for the vehicle to travel. A map of the world is updated by the map update module. And the helm module is responsible for sending driving commands to the vehicle. This paper is concerned only with the vision module and does not discuss the functions of the rest of our navigation system.

Our previous research [1] has built the PARK I system capable of driving our Navlab robot van, shown in Figure 1, on a narrow, twisting, asphalt path under a variety of lighting conditions, with no a priori map knowledge but with a known road model. During the past year, we have concentrated on new capabilities in four areas:

- Handling extremes of lighting (greater dynamic range than the capabilities of our cameras)
- Less reliance on a fixed road model, to allow us to track roads that change and curve
- Use of a map to predict intersection locations and shapes, allowing us to recognize intersections and successfully navigate a road network
- Processing speed, using the CMU Warp supercomputer

Section 2 of this paper discusses our PARK I system, including both a description of processing and a discussion of the points we decided to improve. Section 3 presents specific vision research towards our objectives. Several topics were researched in isolation, some of which were used in a PARK II system. In section 4, we describe the PARK II system incorporating the new vision algorithms, and its performance. This section also discusses the WARP implementation of our PARK I system. And the final section lays out the plan for our next round of research tasks.

*This research is sponsored by the Strategic Computing Initiative of DARPA, DoD, through ARPA Order 5351, and monitored by the US Army Engineer Topographic Laboratories under contract DACA76-85-C-0003, titled "Road Following".

2 Previous Work

2.1 The PARK I Demo System

The PARK I demo system digitizes images and tries to find the best straight road in the image, using color classification and a Hough voting scheme. This is done in five phases: System Interface, Pre-processing, Labeling, Interpretation, and Model Update. The data and control flow between the phases is shown in Figure 2. The interface phase tells the pre-processing phase when to digitize an image. The pre-processing phase transforms the camera data into reduced resolution color and texture images. These images are used by the labeling phase to produce a road probability image, in which each pixel describes how well the color and texture of that image pixel match known colors and textures of road pixels. This probability image is passed to the interpretation phase that finds the best straight road in the image. The road location is back-projected onto the ground plane and passed to the navigation system. The road location in the image plane is given to the model update phase where the vision module's idea of road and grass colors are updated. Each of these phases is described in more detail below.

System interface The interface between the vision module and the navigation system defines the function of the vision module. The navigation system tells perception where the vehicle should be when it takes the next picture. The vision system continuously polls the navigation system for the vehicle position, and digitizes an image when the vehicle is closest to the specified location. The image is then processed to determine the road location in the image.

The required output of the vision system is the location of the road on the ground plane. The ground is assumed to be locally flat, with the vehicle sitting on the same plane as the road in the image. The coordinate frame of the vision output is defined by the vehicle location when the image is digitized as shown in Figure 2. The road is assumed to be straight and of known width. It can therefore be represented as a rectangle on the ground. Furthermore, this rectangle can be described completely by its center line. The location of this rectangle on the ground plane can be described by two parameters, (x, θ): the x position of the center line as it crosses the x axis, and the angle at which the center line crosses this axis. These two parameters are the description of the road that the vision module returns to the system.

Pre-processing The pre-processing phase transforms the camera data into images that can be processed by the labeling phase. In our case, labeling cannot process the full size images from the camera due to time constraints. Therefore, the pre-processing phase first reduces the color images from (480 X 512) to (30 X 32) pixels. This is done in a series of averaging reduction stages that create a pyramid of reduced resolution color images.

Preprocessing also calculates a texture value for each pixel in the (30 X 32) image, to give the labeling phase another cue besides color. Texture is calculated by counting edges found by a high-resolution Robert's operator, normalized by local image intensity and large-scale edges. Normalizing by intensity assures that the texture operator will have similar responses to similar textures regardless of the illumination of the textured region. Normalization by large-scale edges reduces the response of the texture operator to shadow edges. Grassy regions and trees tend to have many local variations, which produce many local edges, which in turn produce a strong response in the texture operator. Road surfaces are usually more evenly colored, and evoke less response from the texture operator.

At the end of this phase the four (30 X 32) images representing red, green, blue, and texture, are passed to the labeling phase.

Labeling The goal of labeling is to label each pixel as either road or grass and to record a confidence that the pixel is correctly labeled. We compute the label and confidence by comparing each pixel's color and texture with road and grass appearance models. We have multiple models for road and for grass. For example, since sunny and shaded road have very different colors, we need a separate color model for each illumination. Moreover, since colors change gradually as the illumination changes from sunny to dark shadows, we have extra models to represent these intermediate colors. We found that the system ran well using four road models and four grass models.

Labeling uses these color models in a standard pattern recognition technique. At each pixel, a likelyhood is calculated for each model, and the model with the highest likelyhood is selected. The probability of this model is recorded in the road-probability image. Positive probabilities in this image signify that one of the road models had this highest probability, while negative pixels signify one of the grass models. The road-probability image is used by the interpretation phase.

Interpretation The interpretation phase is responsible for transforming the road probabilities into a (x, θ) interpretation. This ground plane interpretation can be transformed into the image plane resulting in an equivalent set of parameters: c, the column position where the road centerline crosses the horizon row, and ϕ, the angle with which the centerline crosses the horizon row. These two image plane parameters, (c, ϕ), are used in a Hough space interpretation scheme. Each road pixel votes, using its probability, into the Hough accumulator, for all road shapes that contain that road pixel. Each grass pixel votes, using its negative probability, against all road shapes that contain that grass pixel. The best road location in the image is given by the coordinates of the peak of the Hough space. This location is passed directly to the model update phase, and is transformed to the ground plane and sent to the navigation system interface for path planning and vehicle guidance.

Model Update The model update phase is responsible for adjusting the statistical color model of the different road and grass models. This adjustment is necessary when the illumination conditions are changing, or if the road or grass colors change. The texture model is not adjusted, since texture remains constant regardless of the lighting or the road or grass colors. After the road is found by the labeling phase, we know exactly which pixels in the image are road pixels and which are grass pixels. Although we know the road/grass type of each pixel, we use only those road and grass pixels that are not near the road edge for updating the model. This prevents the corruption of the color model due to errors of modeling curved roads with straight road models. For each of the road and grass regions, we perform a nearest mean clustering technique for all of the data in the road and grass regions. This technique groups color data into a set of models where the color data in each individual model is described by color means and covariances. The clustering technique selects the set of models whose mean values are the most separated, and whose clusters are the most compact. The means and variances of the resulting classes are then used by the labeling phase to label the next image.

2.2 Lessons Learned from the PARK I Demo System

Under many conditions, the PARK I system was rather robust. On days when the light was fairly constant the vision module worked perfectly almost all of the time. In the times that the solution was imperfect, it was not far from the correct solution, so that the Navlab drove on the edge of the road instead of down the center. However, there were some limitations with this system that needed to be addressed.

Increased Dynamic Range The need for an increased dynamic range in the camera is especially apparent on bright sunny days, when the vehicle is in the sunlight and the vision module is processing images containing dark shadows. In these dark shadows, the pixels have such low values that the road and grass pixel values are indistinguishable. Likewise, when the vehicle is in the shade, sunlit regions of the image are saturated, and again, road and grass are indistinguishable. We need some way of increasing the dynamic range of the perception system to handle these difficult scenes.

Changing Illumination The labeling phase depends on knowing the colors of road and grass from the previous image. At the end of each step, we determine new road and grass colors, so that the system can adapt to changing conditions. When illumination changes gradually or when the system slowly enters shadowed regions from sunny regions, the system can update the color model and successfully track the road. In this sense, the system is capable of adapting to the environment. However, when the illumination changes rapidly, as when the sun goes behind a cloud, the color model used for labeling can be inaccurate, and the road can be misclassified. Therefore, we want reliable methods of distinguishing road from grass regions that do not solely rely on color models derived from the previous image.

Intersection Detection As in any vision system, the interpretation phase needs a model for the system to match in the image. In the PARK I system, the model is implicitly coded as a straight road. We needed to add models and algorithms to interpret intersections and curving roads.

3 Research and Algorithm Development

To address the problems discussed above, and to expand our road following capabilities, we researched in the following topics independently:

- As an attempt to increase the dynamic range of the cameras, we developed a two camera pre-processing phase.
- Unsupervised learning was tried for dealing with rapidly changing lighting conditions.
- Two different ideas for recognizing intersections, a convolution interpretation scheme and a region searching algorithm, were developed.

The algorithms that were successfully developed and that met speed and accuracy requirements were integrated into a PARK II demo system.

3.1 Two cameras

Since the dynamic range of available video cameras is not large enough for bright sunny days with dark shadows, we are trying a two camera system with fixed iris settings. One of the cameras has its iris open to peer into the shaded areas of the scene. The other camera has a closed iris for the sunny regions. A closed iris image has an average intensity less than the open iris image. We refer to it as the *dark* image, which is used for looking into sunlit areas. Likewise, the image from the camera with the open iris is called the *bright* image, which is used for looking into shadows. By combining the bright and dark images, we can indirectly increase the dynamic range of the sensor.

To test this idea, we replaced the pre-processing phase of the PARK I system with a new pre-processing phase that digitizes two color images and produces one reduced resolution color image and a texture image. These images receive some of their pixel data from the bright image and some of their pixel data from the dark image. This combined image is fed into the labeling phase.

First the digitized images are reduced to form bright and dark color image pyramids. Next each image pyramid is run separately through the texture subroutine producing a bright and dark texture image. Then, the two color images are combined and the two texture images are combined. For each pixel in these combined images, we select a value either from the dark image or from the bright image. This is better than trying to do any sort of pixel-level averaging or combination since these techniques would distort color values of the images. So the combination step chooses, for each pixel, which image to use, and gets the red, green, blue, and texture values from that image. It also produces a mask image that records from which image, bright or dark, each pixel was selected.

For selecting which pixels should be placed in the combined images, we use a simple thresholding technique. We first apply a threshold to each pixel in the dark image. If its pixel value is less than the threshold value, then the data in the dark image is inadequate. Therefore, this combined pixel is copied from the bright image. If the dark image pixel value is greater than the threshold, then this pixel is usable, and is copied from the dark image into the combined image. A more sophisticated algorithm could perhaps improve system performance by dividing the image into sunny and shaded pixels instead of just dark and bright pixels. However, separating sunny from shaded may be just as difficult as the original problem of separating road from non-road.

By running this combination step, and carefully setting the threshold parameters, we are able to succeed in increasing the dynamic range of cameras to handle images containing dark shadows and bright sunlight. Unfortunately, these parameters need to be individually set for the conditions on each day that we run the vehicle. Therefore, we would like this threshold to adapt automatically. Even with the correct threshold, there are problems with the colors changing within a shadow and with rapid and large illumination changes, such as the sun going behind a cloud. Sunlit areas are fairly evenly illuminated, and the same surface type produces approximately the same image colors. But shaded areas are illuminated by reflections from the sky, from clouds, and from surroundings like trees and buildings. The color of the illumination, and thus the color of the image pixels, changes noticeably depending on the color of the reflecting objects. More research, including fundamental work in optics of materials, is needed to be able to properly classify road and nonroad under all illumination conditions.

3.2 Unsupervised Learning Labeling

When the lighting conditions change rapidly, our PARK I system fails since it relies on color models from the previous image. Because illumination can change rapidly and unpredictably outdoors, models calculated from a previous image may not correspond to the colors in the next image. We would like to build a labeling phase which does not rely on color models from the previous image.

The input to this labeling module is the same as the PARK I system, except, of course, there is no input color model. The output should be a a set of regions in the image, a collection of which would completely and exclusively cover the road in the image. Our basic approach is to use a standard clustering algorithm to group regions that have similar colors [2].

To run this clustering algorithm, the number of classes in the image must be pre-selected. First the pixels of the image are divided randomly into each of the classes. The mean color value is computed for each class. Next each pixel is labeled as belonging to the class whose mean value is closest to its pixel value. The mean color is then recomputed for each class. The sequence of *label by closest mean* and *calculate mean* is repeated until few or no pixels change their class. In the first labeling step, the majority of pixels change their class, however, in the second labeling step, only a minority of the pixels change. In successive steps, the number of pixels that change classes continues to decrease. If run long enough, this will converge until no pixels change; in practice, three iterations is adequate. Now all of the adjacent pixels, having the same class label, can be grouped into regions using a connected components algorithm.

The biggest problem with this procedure is that the number of regions found in the image is quite large. Searching for a road shape becomes impractical if there are too many regions. Simple methods for reducing the number of regions, such as shrinking and growing or merging small regions, either still left too many regions or distorted the region contours to too great an extent. For unsupervised labeling to be applicable to real time road following, we need to improve both the labeling and the searching methods (see Section3.4 below). This segmentation system is not directly applicable in our PARK I system, since its interpretation phase depends on road/grass labeled pixels. There are no semantic labels associated with the unsupervised learning scheme. We are in the process of developing interpretation methods for this scheme.

3.3 Interpretation using Convolution

One of the main reasons for the robustness of the PARK I system was the road model fitting. We used a Hough transform to calculate the globally most likely position of the road, regardless of local misclassifications.

We want to retain that robustness, but extend the class of shapes from just straight roads, in the earlier system, to also include curved roads and intersections. For these more complicated shapes we replaced Hough transforms with convolution, using a template mask in the shape of the predicted road or intersection.

Each road segment or intersection is modeled, based on either map data or previous scene interpretations, as a set of polygons. The vehicle planning system chooses the most likely set of roads and intersections in the field of view, and passes those shape descriptions to the vision module. The vision module does not directly search for those shapes in the image. As the distance changes between the vehicle and an intersection, for instance, perspective projection will change not only the location of the intersection in the image but also its size and shape. So instead the pixels from the image are projected onto the ground, and the search for the predicted shape is carried out in ground coordinates.

The first input to this system is a probability image which contains positive probabilities for road and negative probabilities for grass. The magnitudes of these probabilities correspond to the confidence that that pixel is actually road or grass. The second input is a shape description of the expected road or intersection in the scene. The output of this system is the most likely location of the specified road shape.

First, this algorithm back projects the road probability image onto the assumed ground plane. Then it convolves a mask formed from the description of the road or intersection with the ground plane probability image. For finding a straight road, only the x dimension and the angle of the road need to be found to describe the position of the road, since the appearance does not change along the y direction. However, to locate an intersection, the (x, y, θ) position of the intersection on the ground plane are needed.

To test this idea, we replaced the interpretation module of the PARK I system. We provided test models for the roads and intersections by selecting the borders of the road in out test images. These points are back-projected onto the ground plane, giving us a perfect model of the intersection seen in the image. The results are very good, matching to the road in almost all situations. Matching with distorted models, or with models picked from a different view of the same scene, gave less accurate but still satisfactory results.

3.4 Region Searching Interpretation

The main motivation for this algorithm is to apply road information, such as edge shape, color, road shape, and predicted location, to help decide the location of the road in the image. This scheme frees us from the exact shape description that is required by the convolution interpretation scheme discussed above. In real navigational situations, an exact model, in general, is unknown. This algorithm is an attempt to interpret road and intersection scenes using general constraints, without knowing the exact shape of the roads.

The input to this algorithm is a list of regions. Each region includes the following information: road/grass label, road probability, size, neighboring regions, and polygonal approximations. The output is the collection of regions that make the *best* road. Best, in this case, is evaluated using geometric and heuristic information. We search over various combinations of regions in the image to see which collection forms the best road. Each collection of regions is called the candidate road. An exhaustive search over all candidate roads would be much too expensive, so we use a *Hill Climbing* algorithm to constrain the search space.

This algorithm is tested by replacing the interpretation phase of the PARK I system. In the PARK I system, the labeling phase labels pixels in the image. This algorithm needs these pixels to be grouped into regions, so we added a region growing step to the labeling phase to test this algorithm.

Region Growing Before we can apply the region searching technique, we first need to convert the classification and probability images into a list of regions. This is done in the following steps:

1. Use a connected components algorithm for region extraction.
2. Merge each small region into its most similar neighbor.
3. Approximate the remaining regions with polygons, retaining neighbor information.
4. Calculate descriptors, such as size and road/grass probability, for each region.

Region Searching Algorithm Searching starts by considering the initial candidate road, C_0 consisting of the regions that were labeled as road in the input image. We first evaluate the cost, ν of the initial candidate C_0. The algorithm proceeds by either adding neighboring grass regions to the candidate road or deleting road border regions from the candidate road. A neighboring grass region touches the candidate road, and the road border regions touch at least one grass region. The algorithm can be described as follows:

1. *Try expanding road* by evaluating the costs of all the candidate roads that can be formed by adding one neighboring grass region to the initial candidate road, C_i. Remember which of the grass regions G, when added to the initial candidate, gave the lowest cost of ν_G.
2. Likewise, *try shrinking road* by evaluating the costs of all of the candidate roads that can be formed by deleting one of the road border regions from the initial candidate road, C_i. Remember which of the road regions R, when deleted from the initial candidate, gave the lowest cost of ν_R.
3. If ν_R is lower than ν_G and ν, then removing one region from the road improves the road model fit. $C_{i+1} = C_i - R$ and $\nu = \nu_R$. Go to 1.
4. If ν_G is lower than ν_R and ν, then adding a grass region to the road improves the model fit. $C_{i+1} = C_i + G$ and $\nu = \nu_G$. Go to 1.

5. Else ν is lower than ν_R and ν_G, switching the label of any one region will not lower the cost, so **exit**.

The final result region C_{final} is the collection of regions in the image that form the best road.

Evaluation of Candidate Roads The confidence of a road interpretation can be determined by a combination of the following constraints. Notice that each constraint tries to enforce a particular attribute.

- The *confidence* measure comes from color classification of the pixels, and their resulting road or grass probabilities. It prefers regions that have high road probabilities.
- The *edge straightness* metric prefers candidate roads whose edges form straight lines.
- The *road width* constraint reinforces candidate roads that have the correct width.
- The *parallelness* measure supports candidate road that have parallel edges.
- The *prediction* constraint prefers candidate roads that are spatially close to the prediction.

Any deviation from the ideal for each of these features adds to the cost. We form a weighted linear combination of these costs to calculate the total cost of the candidate road, ν. The candidate road that is selected by the region searching algorithm will have the best compromise between all of these constraints. This cost function can be easily expanded if additional constraints are needed and available.

Results This approach is very promising and powerful. When given good predictions, it successfully classified all our test images except the single worst shaded intersection image. Its main drawback, and the reason it was not incorporated into our demo system, is that it does not guarantee real-time performance. In clean scenes, relatively few regions are produced by pixel labeling and region growing, and the initial candidate road is nearly correct. Then the search proceeds quickly, goes through few iterations of generating and evaluating candidate roads, and is quite efficient. In more complex situations, such as dappled sunlight or leaves scattered on the road, it is possible to produce literally hundreds of separate regions. Searching over all those regions is not very efficient. Merging small regions into their larger neighbors helps with search time, but at the expense of distorting region boundaries and possibly corrupting the solution. This approach requires, and merits, further work before it is practical in real-time road following, and to enable it to not to rely on good predictions.

4 Demo Systems

During the course of the year, we built two demo systems to test new capabilities of vision and the rest of the Navlab navigation system. The first system was a reimplementation of the PARK I system using the WARP machine. The second system was built to test some of the new algorithms discussed above.

4.1 Warp Demo System

The PARK I system required about 10 seconds to complete one vision cycle on a Sun 3/160. This is much slower than we would eventually like. Therefore, we wanted to run the PARK I system on the prototype wire wrapped WARP machine, to get an idea of the speed we could get from the current supercomputer technology. A prototype WARP was installed on the Navlab for this purpose [3]. The WARP is an experimental high speed computer consisting of

- an array of four cell processors that can perform parallel computations,
- an interface unit to transfer data to and from the cell array,
- three 68020 processors to perform higher level data processing and sequencing, and
- a host computer that links high level programs to the WARP machine.

The WARP group provided software support to make this demo possible. Programs for the WARP machine are specified in an Ada-like language called W2 [4] [5]. The W2 compiler produces code that runs in parallel on the cells of the Warp. The warp_call() interface allows a C program running on a Sun host to call W2 functions on the parallel processing cells, and automatically handles all data transfer and format conversions.

We divided the vision program into modules that were consistent with the function of the WARP:

- Roberts' gradient operator
- Texture operator
- Color and texture classification
- Statistics collection
- Statistics class adjustment
- Hough space voting

All of these modules, except for the Hough module, were implemented by dividing the input images into column swatches. Each of the cells of the WARP array was responsible for one part of the image data. The results were then merged when they were read from the array. For the Hough space module, the Hough space accumulator was divided among the cells by column swatches. The input image was passed to all of the cells. Therefore, each cell would look at each pixel in the input image and vote for all the road locations within the range of angles described by its swatch of the Hough accumulator.

Recoding our algorithms for the Warp succeeded, and even on code that was not designed for parallel processing we achieved a speedup of 2.5, from 10 seconds per image to 4. We learned several practical lessons:

- Designing for parallelism in the first place would have saved us much more time. The Warp is much better at some things, such as regular computations. It is not so fast at others, such as conditional branching. Efficient use of the WARP requires designing algorithms with these constraints in mind.
- Even with the computing power of the Warp, standard coding optimizations such as unrolling loops and folding constants are still important time savers.
- The prototype Warp did not support constructs such as variable loop bounds. So we had to pad data to maximum sizes, thereby wasting computation.

The production version of the Warp, which we have since mounted on the Navlab, helps solve those problems.

4.2 Park System II Vision Module

The PARK II demo system tested some of our experimental algorithms and tested the new PC WARP. Specifically, we wanted a system that used the two camera pre-processing phase and the convolution interpretation algorithm. The vision system would then be capable of recognizing intersections in even strong shadows.

An outline of this system can be seen in Figure 3. This system is based on the PARK I system with some important changes. The pre-processing phase is replaced with the two camera system that was discussed in section 3.1. The labeling phase of the PARK I system then processes the combined images. The interpretation phase of this system is the convolution interpretation scheme discussed in section 3.3. The location of the road is then reported to the navigation system through the new interface phase, which is modified to transfer intersection descriptions. The polygonal model of the road or intersection is passed to the model update phase, which updates the bright and dark color models used for classifying the next image. The more detailed description of the modifications to each of the modules is given below.

Calibration We expected to need calibration between the dark and bright cameras. By mounting the cameras as close to each other as possible, and by careful bore sighting, we were able to get nearly perfect alignment. For the ranges of interest, and at reduced resolution, typical errors between the two cameras were less than a pixel.

Calibrating the geometry between a camera and the vehicle became more important for this system. In the PARK I system, vision only reported locations of straight roads, so calibration of distances along the road was not crucial. In the PARK II system, vision had to report intersection locations as well, which required careful calibration of all camera parameters. We developed and used a calibration scheme that uses two different views of a grid of spots to deduce camera geometry [6].

Resolution To identify intersections at reasonable distances, the (30 X 32) resolution of the road probability images was not high enough. The branches of the intersection were typically smoothed over by the image reduction. Therefore, we increased the image size to (60 X 64). We also developed a subroutine to digitize from both of the cameras almost simultaneously by digitizing at reduced resolution into separate portions of the frame buffer on successive camera frames. Our input color images were therefore (240 X 256) rather than (480 X 512).

Convolution Interpretation The convolution algorithm is used for the interpretation phase of this system. The navigation system predicts the shape of the road or intersection in the field of view, or of multiple possible objects if vehicle position is imprecisely known. Each shape is matched to the road seen in the image. The object with the best matching shape is returned as the identified object, and its best fit position in the image is projected onto the ground plane and returned to the navigation system. This shape description is also passed to the model update phase.

The location of a road can be described with two parameters, namely (x, θ). To describe the location of an intersection we need three parameters (x, y, θ). Unfortunately, when the dimension of the convolution increases, the computation increases exponentially. In order to reduce the computation time for intersections, we attempted to use an on board gyrocompass to localize the θ dimension. Then the intersection convolution reduces back to two dimensions, (x, y). The navigation system reads the gyro information, properly rotates the intersection model before passing it to the interface phase. The intersection model is matched to the image data, and the properly translated model is returned to the navigation system.

Results The system ran successfully through the three intersections on our test course. Each of the intersections had different shapes and two of the intersections were in shaded areas of the site. The first intersection was located on a curved section of the path and the branch exiting on the left was comprised of a different material and was narrower than the two other branches. The second intersection was Y shaped located at the crest of a hill. The third intersection was a T intersection where the approaching branch, the upright of the T, was angled.

Two of the results of our system are shown in figure 4, one showing the algorithm for heavily shaded road interpretation, and the other showing intersection detection. Each result is shown in a cluster of four images. The top two are the bright and dark images from the cameras. The lower left is the combined image. The result of the segmentation is shown in the lower right. The position of the detected road or intersection is drawn on top of the dark image in the upper left.

We made many runs, including runs in a variety of lighting conditions and longer runs than any of our previous systems made. The navigation system used perceived roads not only for navigation but also for map building, and used intersections as landmarks. The final maps showed all of the bends and curves in our test course, and correctly located the intersections. Along with our successes, we also discovered areas for further work.

- **Shape descriptions** It is impractical to depend on exact shape descriptions of intersections. Moreover, if the shape descriptions are significantly incorrect, not only is the calculated location somewhat off, but the statistics update phase may incorrectly update its model of road and nonroad colors, leading to misclassifications in later scenes.
- **Assumed ground plane** The ground plane assumption is unusable for the second intersection of our test site, where the intersection lies on the top of a hill. The perceived shape of the intersection changes dramatically as the vehicle pitches forward at the top of the hill.
- **Gyro problems** We had difficulty using the gyro to orient the intersections. This was the first system to attempt using the gyro on the Navlab. The device would drift in orientation to the point that dead reckoning of the vehicle location provided better orientation information than did the gyro. Therefore, when matching the intersection models to the intersections, we would often receive models incorrectly oriented from the navigation system. Future systems will have to have better heading information, or will have to search for intersections in x, y, and θ.

5 Future and Continuing Work

Our current research focuses on five areas: increasing camera dynamic range to an even greater extent, dealing with shadow fringes, using approximate shape models, getting improved ground plane models from direct 3-D sensors, and improving unsupervised classification labeling.

Because our two cameras each had a fixed iris setting, large changes in illumination could cause problems. When the sun went behind a cloud, in some instances even the *bright* image was not bright enough to see into the shadows. We are investigating computer controlled irises and computer controlled digitizer gains to try to expand further the dynamic range of the vision system.

Shadow fringes in conjunction with changing illumination also provide problems. The pixels that are most often misclassified are those at the fringes of shadows, where the illumination changes from bright to dark. These mixed pixels may not belong unambiguously to any one color class. In images where there is a sharp shadow line, this causes problems in only a few pixels. In other images, however, with dappled sunlight filtering through trees, a large portion of the image is composed of shadow fringe, and many pixels may be misclassified. We are working on methods to detect shadow fringes and discard those unreliable labels.

In the current system, an exact intersection model is required by the interpretation phase of the vision system. However, exact models are not realistic in real navigational situations. They are also impractical even if the exact shape is measured. Therefore we would like to continue our research into the region searching interpretation method. A parallel effort in our project has developed an expert system image interpreter that finds roads and intersections in very difficult scenes and which has only weak shape models [7]. This expert system, written in OPS, takes up to a half hour per image. If we can extract the most powerful heuristics from this system and combine them with our experiments in region-based road finding, we can perhaps build a powerful yet still not too slow system.

We would also like to improve our ground plane assumption in the vision system. In order to construct a ground surface, we need three-dimensional information about the ground surface. Another effort in our project is using a scanning laser range finder to build 3-D terrain maps. We are looking at methods for merging this 3-D data with our color images and using the combined data to give true intersection or road shape and location.

We are also continuing our work on the unsupervised classification method. This labeling algorithm should be able to handle changing illuminations. The remaining problems are that the labeling scheme results in far too many regions to be reasonably processed in real-time. We hope to develop some better interpretation methods or region reduction algorithms.

6 Conclusions

Our road following algorithms have made significant improvements, but there remains much work before we achieve our goal of reliable, autonomous, road following. Specifically we believe the following:

1. Using 2 cameras with different apertures helps significantly with the problem of small dynamic range. We continue to work on problems of rapidly changing illumination, and on changes of color at shadow fringes and within shadows.
2. Convolving an intersection template mask with classified image pixels, projected onto the ground plane, works for recognizing intersections. We continue to research methods for finding intersections without exact shape models.
3. Region based road finding methods can help by using not just individual pixel probabilities by other geometry, such as edge straightness an parallel edges. Such heuristics are powerful but time consuming.
4. Unsupervised classification has promise for producing cleaner regions for region based methods. Since it does not rely on matching with known color models, unsupervised clustering correctly models the colors in each image, and produces cleaner region boundaries.
5. Other non-color factors are important, such as fusion with three-dimensional data to get accurate ground shape models and planning for parallel implementation.

References

[1] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer. Vision and navigation for the Carnegie-Mellon Navlab. *Annual Review of Computer Science*, 2:521–556, 1987.

[2] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., 1973.

[3] M. Annaratone, E. Arnould, T. Gross, H. T. Kung, M. Lam, O. Menzilcioglu, and J. Webb. The Warp computer: architecture, implementation, and performance. *IEEE Transactions on Computers*, C-36:1523–1538, December 1987.

[4] Monica Sin-Ling Lam. *A Systolic Array Optimizing Compiler.* PhD thesis, Carnegie-Mellon University, May 1987.

[5] T. Gross and M. Lam. Compilation for a high-performance systolic array. In *Proceedings of the SIGPLAN 86 Symposium on Compiler Construction*, pages 27–38, ACM SIGPLAN, June 1986.

[6] K. Gremban, C. Thorpe, and T. Kanade. Geometric camera calibration using systems of linear equations. *IEEE International Conference on Robotics and Automation*, 1988.

[7] T. Fujimori and T. Kanade. *Knowledge-Based Interpretation of Outdoor Road Scenes.* Technical Report, Department of Computer Science, Carnegie-Mellon University, 1988. In preparation.

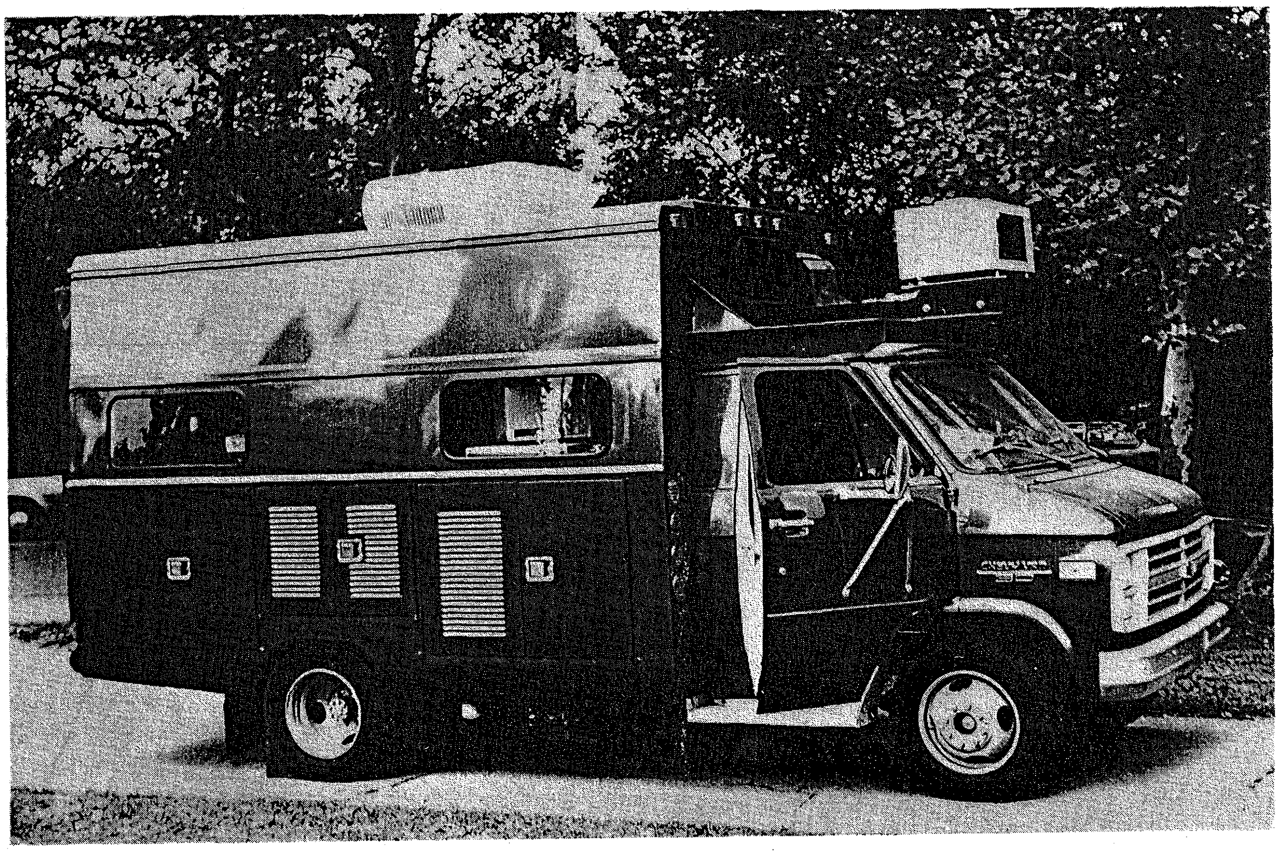
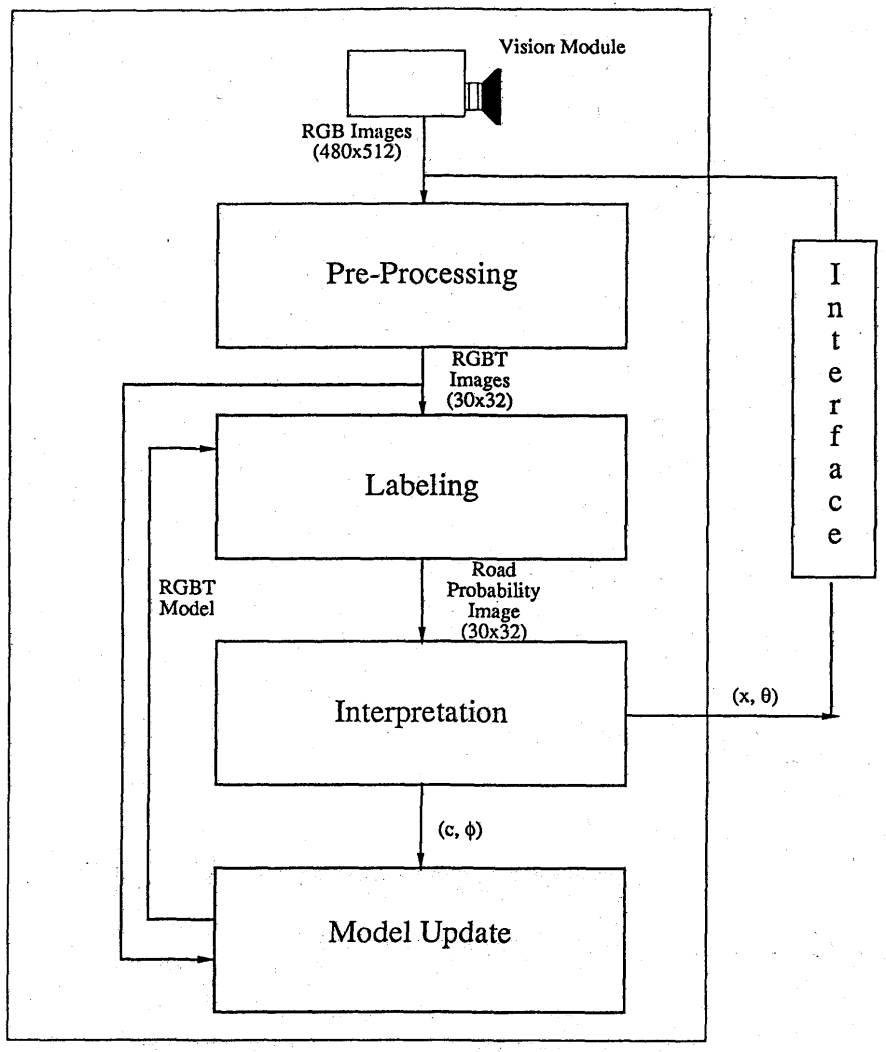


Figure 1: The Navlab Vehicle



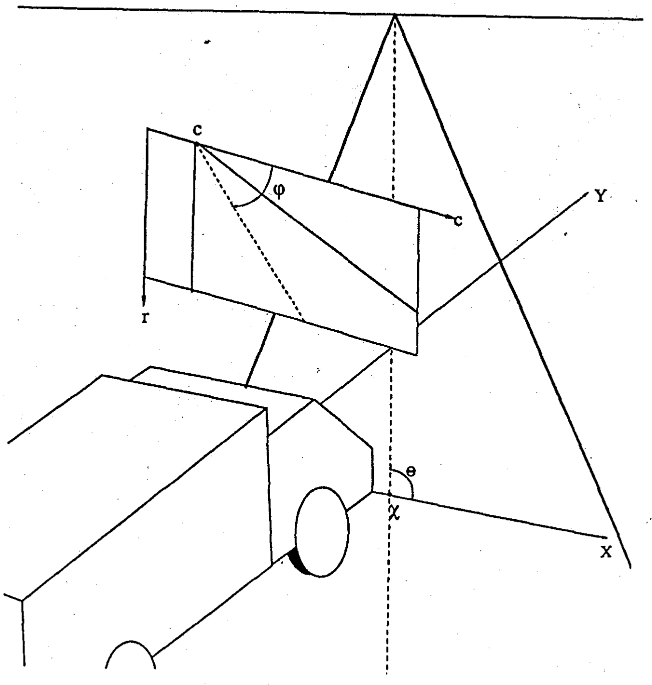


Figure 2: The PARK I System

and the Vehicle Coordinate Frame

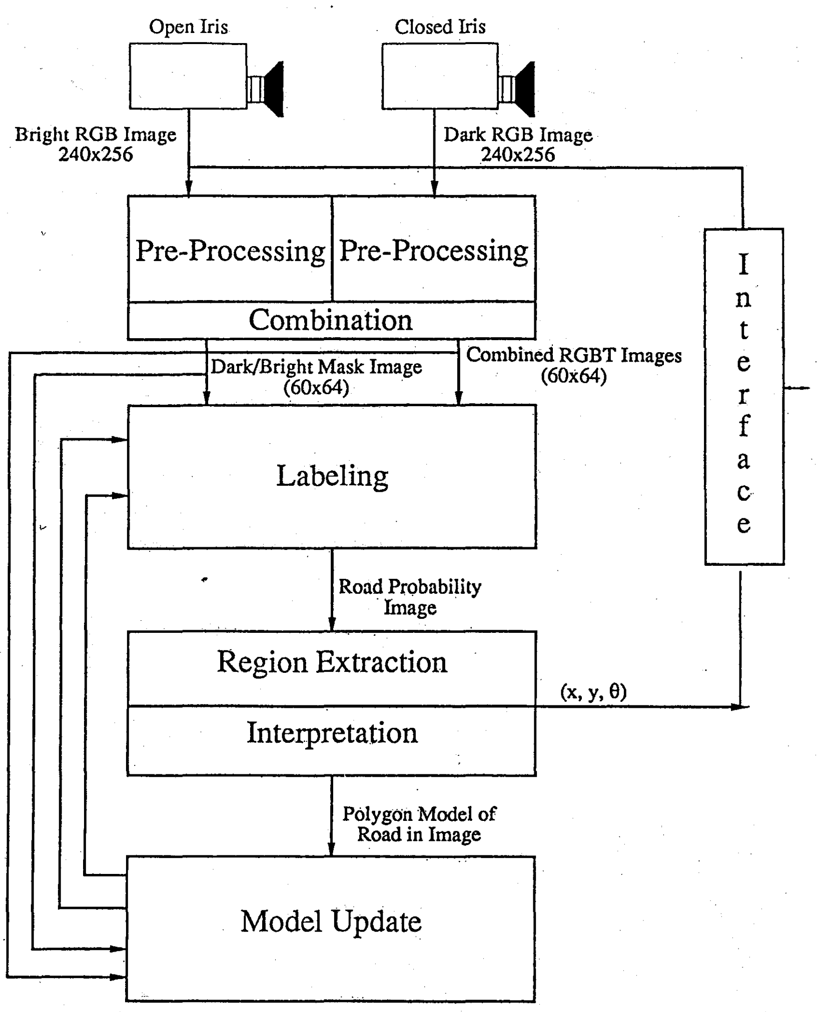


Figure 3: The PARK II System

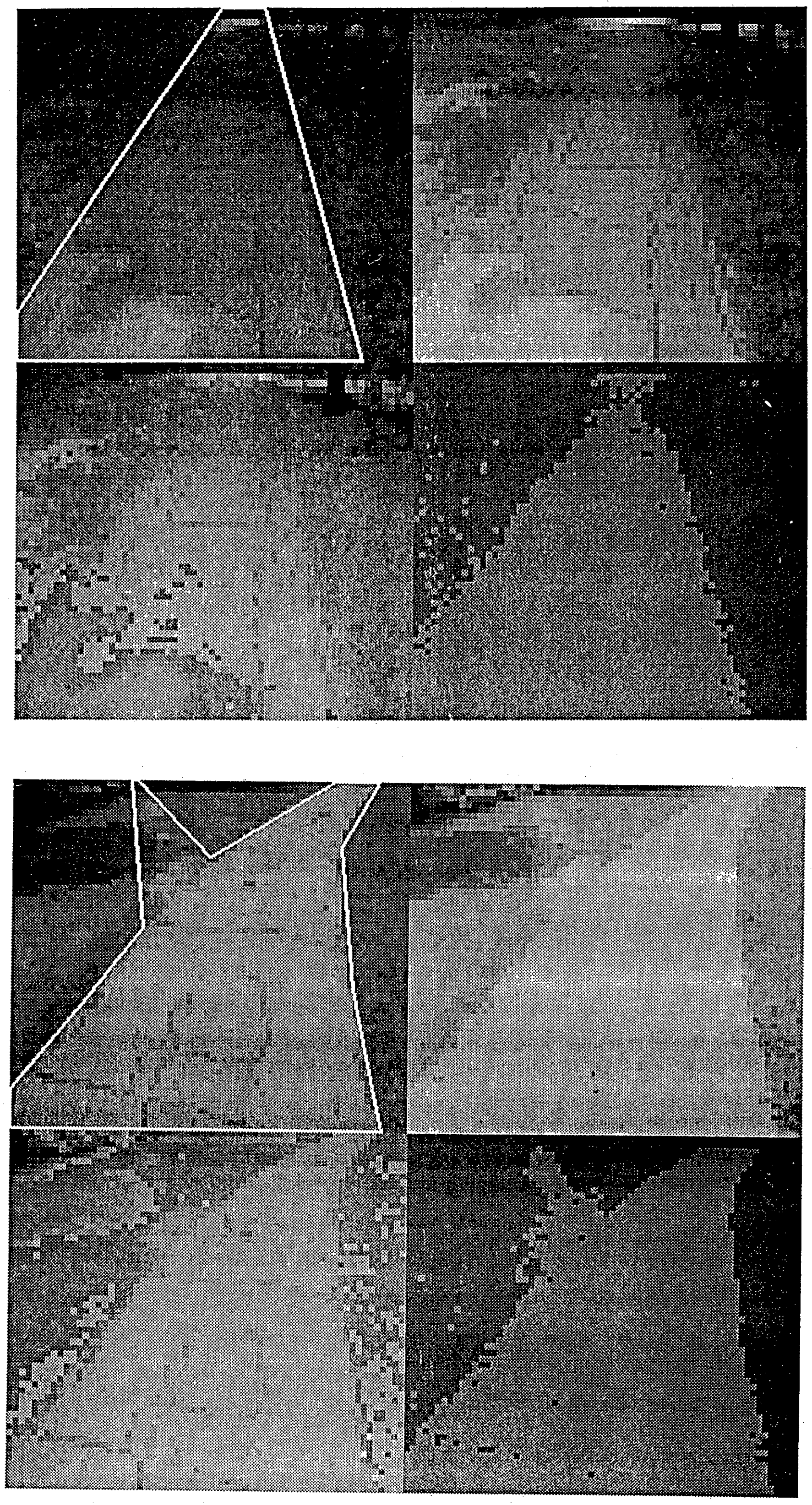


Figure 4: Results of PARK II System