11-1998

# Robust Course-Boundary Extraction Algorithms for Autonomous Vehicles

Chris Roman
*University of California - San Diego*, croman2@uri.edu

Charles Reinholtz



## Citation/Publisher Attribution

Roman, C. & Reinholtz, C. (1998). Robust course-boundary extraction algorithms for autonomous vehicles. *Intelligent Systems and their Applications, IEEE*, vol.13, no.6 (Nov/Dec): 32-39.
DOI: 10.1109/5254.736000

# Robust Course-Boundary Extraction Algorithms for Autonomous Vehicles

# Robust Course-Boundary Extraction Algorithms for Autonomous Vehicles

**Chris Roman and Charles Reinholtz, Virginia Tech**

*PRACTICAL AUTONOMOUS ROBOTIC VEHICLES REQUIRE DEPENDABLE METHODS FOR ACCURATELY IDENTIFYING COURSE OR ROADWAY BOUNDARIES. THE AUTHORS HAVE DEVELOPED A METHOD TO RELIABLY EXTRACT THE BOUNDARY LINE USING SIMPLE DYNAMIC THRESHOLDING, NOISE FILTERING, AND BLOB REMOVAL. THIS ARTICLE DESCRIBES THEIR EFFORTS TO APPLY THIS PROCEDURE IN DEVELOPING AN AUTONOMOUS VEHICLE.*

AUTOMATION, IN ITS MOST BENeficial form, should relieve humans of dangerous and boring endeavors. This philosophy has found broad application in industrial settings, where the environment is well-structured and cost-to-benefit analysis or safety regulations can justify the expense of automation. With the recent availability of reliable and inexpensive computer-vision systems and microcontrollers, AI system developers are applying automation to more complex and unstructured tasks, such as autonomously guided vehicles. The potential for reducing automobile accidents deaths and injuries is, in itself, a compelling reason to pursue systems that enhance driver performance and minimize errors due to poor judgement or inaccurate driver perception.[1]

In pursuit of such systems, the Society of Automotive Engineers, the Association for Unmanned Vehicle Systems, and Oakland University jointly sponsor the annual Unmanned Ground Robotics Competition. This competition fosters the development of small robotic vehicles that can autonomously navigate an outdoor obstacle course approximately 700 feet long. Continuous or dashed white or yellow boundary lines on grass or pavement define the course, which includes obstacles, a steep incline, and a sandpit. The vehicles must be completely autonomous, meaning that all sensing, computation, control, and power systems must be on board.

## Path following

The major hurdle for a mobile robot's vision system is in ensuring reliable perception, which then guarantees efficient autonomous navigation: "Perception robustness depends essentially upon the reliability of the road-edge extraction algorithm."[2] Furthermore, the system must correctly identify its path under a wide range of light and weather conditions. The computer-vision-based path-following problem is critical for many mobile-robot applications.[1,3,4] In a typical roadway-following situation, an algorithm's performance is judged based on its speed and consistency. Approaches based on sophisticated edge-detection methods or complex transforms frequently limit controller update rates, which leads to navigation errors. In our experience, it is better to use computationally simple algorithms running at higher speeds. Furthermore, we find that intensity-based line-finding approaches are simpler and more robust than any edge-detection method.

Our task thus far has centered on developing a consistent method of extracting white or yellow lines from a grass or asphalt background. Painted using flat latex paint, the lines are approximately 4-inches wide. The grass or asphalt background can vary widely in appearance, with the grass ranging from a


1094-7167/98/$10.00 © 1998 IEEE

uniform green, to spotted green and yellow, to completely brown (see Figure 1), and the asphalt from high-contrast black to low-contrast gray. In each instance, the line or background's quality will likely change as the vehicle traverses the course. Beyond the lines' pure physical quality, we must also account for the effect of weather conditions on system performance. Rain or glare from direct sunlight can significantly change the lines' apparent quality and increase the line-extraction procedure's difficulty.

## Computer-vision system

The overall computer-vision system, implemented on a mobile robot we call Christine, includes the hardware used to capture images of the vehicle path and the software written to process the image. The basic hardware consists of a single high-resolution black and white charge-coupled device (CCD) camera with an auto-iris, 3.5 to 8 mm zoom lens, a Current Technologies frame grabber with onboard digital signal processor, and a Pentium 200-MHz personal computer. As the sidebar shows, the camera mounts to an adjustable rail to allow viewing at various angles and heights from approximately four to six feet above the ground. The vision system's hardware section provides an 8-bit black and white image that is stored on the frame-grabber board. We set up the frame grabber to capture 512 × 512-pixel images.

Once the frame grabber obtains an image, the system manipulates it using both the DSP functions resident on the frame-grabber board and in the PC memory buffer. The original image (512 × 512 pixels) reduces to two regions of interest (ROIs), which are of variable size and move dynamically around the screen following the line in the image. Reducing the processing region's size decreases both the image-processing time and the time required to transfer the image from the frame-grabber image buffer to the host PC memory. Analyzing only the portion of the visual field needed for effective navigation reduces execution time.[2,4] Figure 2 shows a typical course as viewed from the CCD camera and as a human observor would see it.

**Boundary-detection techniques.** The vision system's primary goal is to determine the path-boundary location in the image-coordinate system. Two popular methods for this task are edge detection and pixel-intensity analysis. In either case, the algorithm's success depends on its ability to

- find and locate a line in the image,
- eliminate noise or ignore noise and reflections (large bright specular regions caused by the sun), and
- reduce false detection and reject obstacles that might be confused with the course boundary.[4]

These criteria are similar to those John Canny developed to design and evaluate the performance of optimal edge-detection methods.[5,6] To select a method for boundary identification, we must determine which approach is best suited to operate efficiently and reliably under the given conditions.

For roadway-following problems, two distinguishing characteristics of the boundary line are the presence of the line edges and the line's overall intensity compared to the background. Therefore, either edge-detection or pixel-intensity methods would work for finding course boundaries. Edge-detection methods seek to find abrupt intensity transitions in the image, which represent an edge. An ideal edge is represented by a rapid change in intensity along an image contour, the most obvious being a step edge.[7] Our experience with road-

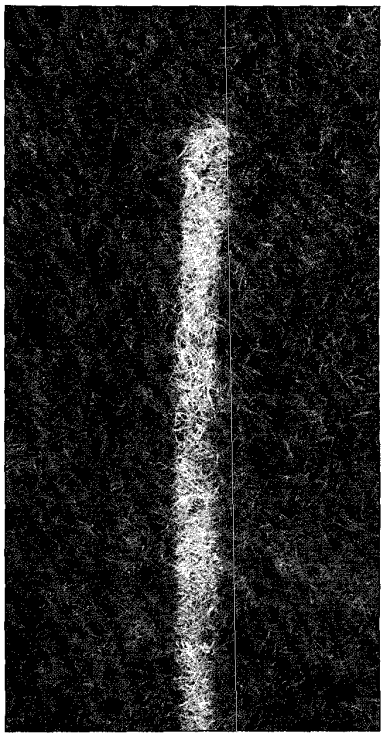

Figure 1. A typical line painted on grass, which shows the different shades of green and brown contained in the grass as well as the varying intensity of the line.

### Christine

Christine was one of three vehicles entered by Virginia Tech in the 1997 Unmanned Ground Robotics Competition (see Figure A). Christine's operational system combines a PC-based vision system and a microcontroller, which performs sensor integration and executes control tasks. The vision system and microcontroller software was developed using standard C and the frame grabber's C libraries. The microcontroller executes the final navigation commands by integrating the preliminary steering angle obtained from the vision system with obstacle information from ultrasonic and tactile sensors. The microcontroller must assign priority to obstacles potentially in the vehicle's path. As a consequence, tactile and ultrasonic sensor information is, in some cases, weighted more heavily than vision in determining the actual steering angle. In effect, the microcontroller performs all onboard navigation by closing the loop around the vehicles desired steering angle and ground speed.

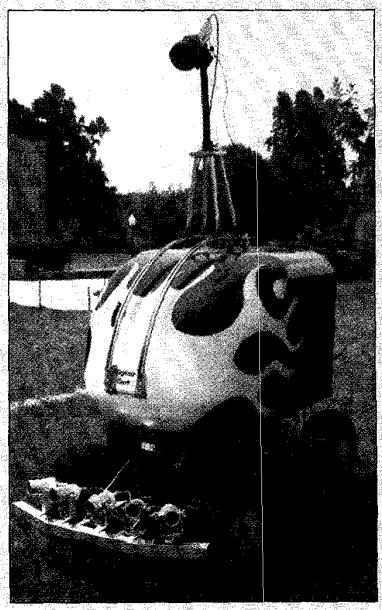

Figure A. Christine.

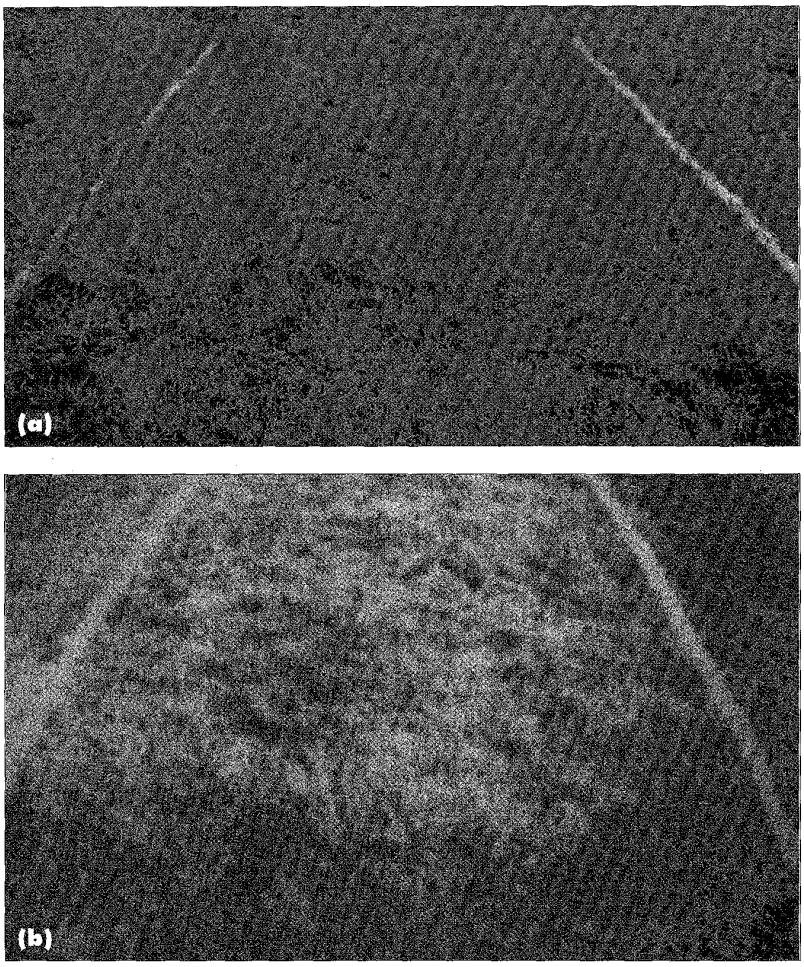


Figure 2. The course (a) as it would appear to a human observer and (b) as seen by the CCD camera.

ways and obstacle courses indicates that line edges are often blurred and poorly defined.

As an alternative to edge detection, we propose a simple intensity-based method that relies on two key assumptions: that the boundary lines will be the most intense, well-structured line-like features in the image, and that the line will occupy a small percentage of the entire image. From our experience, the course's boundary lines in most practical situations fulfill these assumptions. Others working with vehicle roadway navigation have made similar assumptions.[8] We implement this pixel-intensity method by isolating pixels of a specific relative-intensity level using a thresholding operation. A sample image obtained using the vision system shows how we can evaluate the characteristics of the line edge and intensity.

Figure 3 shows the pixel intensities for a single row of pixels obtained by sampling a typical line on a grass background using a 125-pixel-wide ROI. The higher-intensity peak represents the line, and the low-intensity fluctuations result from variation in the grass background and noise. This test image shows a definite intensity difference between the line and image background. It also shows the transition between the lower- and higher-intensity levels that form the line edge. In this situation, the edge is marked by a definite increase in intensity over a range of approximately five pixels. This intensity change would be considered a broad edge, which is typical of a line that has been spray-painted on grass or a rough asphalt surface.[7] Although the edge's quality might change, it is unlikely that the edge will ever appear as a crisp, distinct transition that would characterize an ideal edge.

Aside from the image's edge quality and the separation between the line and background intensity levels, we must consider the amount of random noise in the image. For a line-identification routine to work successfully, it must have reduced sensitivity to noise in the image. Many features in the image cause noise, and this noise has a variety of effects on the boundary-identification algorithm. Very small collections of high-intensity pixels caused by glare and reflection affect image quality most significantly.

**Advantages and shortcomings of edge detection.** Although edge-detection techniques work in many different situations, several drawbacks limit their effectiveness in course- or roadway-following applications. Both intensity-based line extraction and edge-detection algorithms work well when the images contain relatively little noise and the lines are sharp and distinct. The performance difference between the methods becomes apparent when the images contain high levels of noise and varying background intensities. System developers often evaluate the performance of edge-detection algorithms on noisy images assuming an evenly distributed Gaussian white noise superimposed on a step or ramp edge.[5,6] Although many effective edge-detection methods have been developed using this assumption, it is not the best characterization of the noise present in a typical outdoor environment.

In a controlled or more structured environment, the Gaussian noise is a good assumption that can be mathematical characterized and used to design edge-detection methods.[5] The white noise adds no structure of its own to the edge and hence leaves the edge's underlying properties unchanged. In the outdoor environment, however, the image quality usually suffers from disturbances that have structure, such as patchy grass or sections of high specular reflection.

Because simple edge-detection algorithms are essential high-pass filters they are prone to detecting false edges in an environment where the image contains structured disturbances. More sophisticated algorithms that are less prone to identifying false edges are often computationally intensive. The danger of identifying false edges is one of the most critical performance characteristics in edge-detection algorithms.[4-6]

All edge-detection algorithms generate a second image, sometimes called an intrinsic image, that has high intensity in places where the original image had a high rate of intensity change. Following the application of the edge-detection algorithm, the intrinsic image must be evaluated to isolate the strongest responses. This most often involves selecting a threshold to isolate the true edges in the image from the background noise. Selecting

this thresholding value for the intrinsic image complicates the edge-detection procedure, because it must isolate the strongest responses and not introduce false edges. Determining a threshold value for the intrinsic image is difficult because the features that show up in the intrinsic image are directly affected by the smoothing operation performed on the image prior to edge detection.[5,6]

To look at the performance of an edge-detection alogorithm, we use the sobel operator, which is a common edge-detection method frequently used as a basis for comparison.[9] This first-order method is relatively stable in reducing image noise compared to second-order operators.[4] Figure 4 shows the response of the sobel operator on a typical course-boundary image. The edge detector responds to the presence of the line edge in the image, but it also shows a response to typical levels of image noise. To distinguish between the line and the noise in the intrinsic image, a threshold value must be selected by the algorithm. The ability to distinguish the response of an edge from image noise using a thresholding filter depends on the edge being relatively sharp and having a high contrast relative to the background. In the outdoor environment, we find that edge detection does not handle specular reflection well.[1] The reflection of sunlight on grass, for example, often exhibits sharper contrast than the course's edges, which means that these edges are incorrectly considered to be lines.

Processing time for edge-detection algorithms is also an important consideration. The more robust edge-detection algorithms often use large convolution masks to effectively reduce the indication of false edges. The line quality and edge size to be detected govern the size of the convolution mask needed to identify the edge. To identify an edge transition that occurs over four or more pixels requires at least a 5 × 5-pixel convolution mask. Even so, broad edges lead to a much-reduced response. Only a portion of the edge is under the mask at one time, which reduces the apparent change in intensity to a fraction of that which occurs across the entire edge. When considering the presence of a broad edge in a noisy image, the edge detector's response might be no stronger than the response generated from the image noise. In many cases, distinguishing the noise from the broad edge without a sophisticated search of the intrinsic image might become impossible.[7]

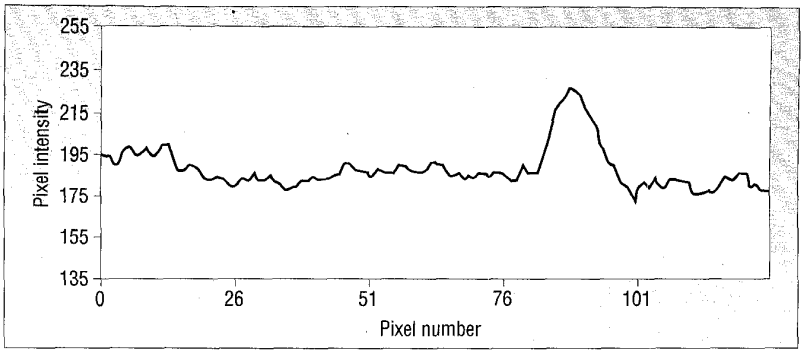


Figure 3. A single row of pixel intensities for the line shown in Figure 1. (Pixel intensity is a relative value from zero to 255. Zero represents black; 255 represents white.)

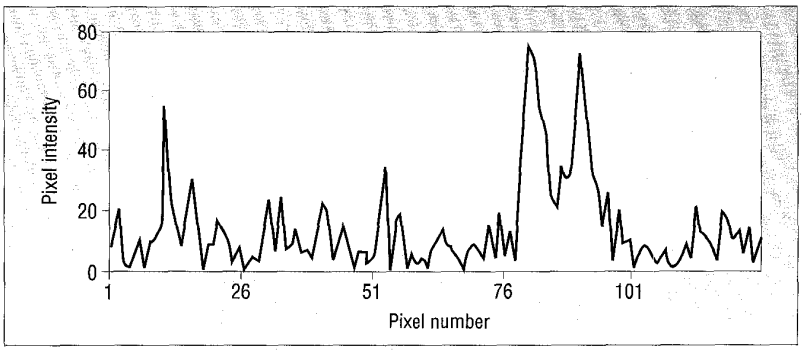


Figure 4. A single row of pixel intensities after a sobel filter. The two highest spikes indicate the line edges. The remainder are noise. (Pixel intensity is a relative value from zero to 255. Zero represents black; 255 represents white.)

## Intensity-based line extraction

The primary assumption made in intensity-based line extraction is that collections of the image's brightest pixels will indicate the course boundaries. In this sense, it is very similar to a road lane where we know that, under almost all conditions, the lane boundaries will be the brightest and most well-structured markings in the immediate environment. Intensity-based boundary extraction attempts to use the image's most stable property as the primary line-identification tool. The histogram is stable, available at low calculation cost, and shows some advantages over edge-detection methods "under especially evil conditions with luminance variation or sun shade or rain drops."[8]

Beginning with the digitized image, the entire line-extraction process takes four steps:

- high-frequency image-noise reduction,
- binary-image thresholding,
- post-extraction noise reduction, and
- position analysis of the extracted line.

**High-frequency image-noise reduction.** Once we have obtained the image, we would like to remove high-frequency noise from it. The image noise comes from spurious, high-intensity pixels that are not part of the line. These pixels are generally the product of single-point, high-intensity light reflection off of the grass or asphalt course surface. Depending on the environmental conditions, the amount of high-frequency noise and spurious disturbances can be quite significant. Because the major intrinsic property of the line in the image is its intensity, we would like to attenuate the intensity of all pixels that do not indicate the line. We can reduce noise using a simple blurring 3 × 3-pixel convolution mask. The size of the convolution mask used depends on the line's relative pixel width. In our case, the 3 × 3 mask worked well for a line typically between 4 and 6 pixels wide. For intensity-based line extraction, this operation has little affect on the ability to locate a line because pixels that occur in groups (such as those contained in the line) are left unchanged. Using such a blurring operation during the edge-detection algorithm will affect the result because the line edges will be "softened," hence complicating the edge-detection process.

**Binary-image thresholding using a histogram.** The line-extraction procedure's most significant operation is the binary thres-

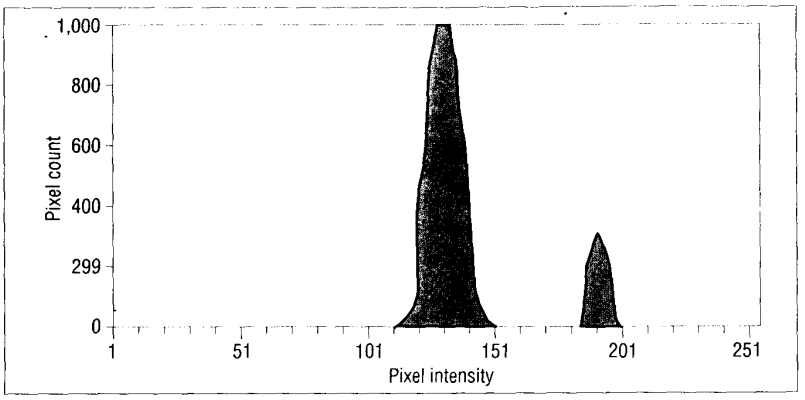


**Figure 5. The ideal histogram for thresholding contains high- and low-intensity pixel collections (Pixel intensity is a relative value from zero to 255. Zero represents black; 255 represents white.)**

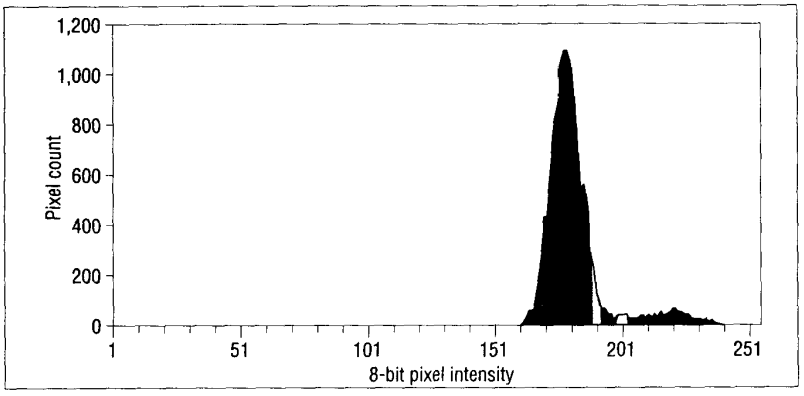


**Figure 6. An actual image histogram with two possible thresholding values highlighted. (Pixel intensity is a relative value from zero to 255. Zero represents black; 255 represents white.)**

hold of the image. This procedure will isolate all pixels in the ROI that have an intensity greater than a specific threshold value. The image's content after the threshold is based on the selection of the thresholding value. Ideally, we would like to have only the pixels that represent the line remaining after the thresholding operation.

In its simplest form, we use the thresholding process to separate a feature in an image based on apparent intensity. In our case, we would like to isolate the line from the remainder of the image by eliminating the lower-intensity image background and leaving the higher-intensity line. A pixel-intensity histogram such as the one in Figure 5 is a simple way to represent the specific intensity levels present in the image. This graph shows what the histogram might look like for an ideal case, which would be a monochromatic high-intensity line on a low-intensity background. In this case, we would select a value between the two intensity spikes on the histogram for the threshold. This selection would completely isolate the line from the background.

Although the actual images recorded with the CCD camera are not bimodal, the intensity-separation principle still applies. The graph in Figure 6 shows an actual histogram for a typical image. Although the separation between the line and the image background is not as distinct as for the ideal case, we can see that the higher-intensity line is represented by a collection of pixels at the scale's higher end. The histogram's colored portions show possible threshold values. If the overall light intensity decreases due to a shadow or cloud, the entire pixel count on the histogram will shift to the left, but it will maintain the same general shape. To smooth out the histogram, adjacent columns can be averaged with minimal loss in resolution. This histogram's shape will also change with the use of optical filters and polarizers. Depending on the environmental conditions, these passive methods can significantly improve our ability to distinguish between the line and the background.

To complete the thresholding procedure using the histogram, we must select a threshold intensity. In a controlled environment, we can do that statically. In this case, we set a single intensity and threshold all images using it. This, however, does not compensate for intensity changes that occur in the image due to environmental changes. Outdoors, operating in natural light, a static thresholding system is subject to many problems and will not generate consistent results. If the image's overall intensity changes, there will either be more noise in the image or fewer pixels. For consistent results, we must use a dynamic method for selecting the threshold intensity to evaluate each image independently.

The dynamic thresholding routine should select the thresholding intensity that separates the line from the image background. The pixels at the histogram's highest end should represent the line in the image. Moving from right to left on the histogram, the pixel intensity decreases and a transition occurs between the higher-intensity line and the majority of pixels composing the image background. To find this transition's edge, we have implemented a simple gradient routine to move across the histogram from right to left and look for a sharp increase in the pixel count between adjacent intensity values. Once it identifies the transition's edge, the routine adds an offset value to the corresponding intensity to determine the actual thresholding intensity. The offset value will move the thresholding intensity away from the transition region and into the higher-intensity line pixels. Although some line pixels will be removed from the image by adding this offset to the threshold, the number of questionable or extraneous pixels that are also removed makes the sacrifice worthwhile.

The parameters used to define this threshold selection routine are the *step* and the *offset* values. Step refers to the gradient between two adjacent intensities and offset refers to the value added to the gradient intensity. Figure 6 showed the histogram with the edge transition highlighted in dark gray and the shifted threshold value in light gray. Figure 7a shows the image thresholded at the value indicated in dark gray—that is, with zero offset. Figure 7b shows the same initial image thresholded with the offset value. Adding the offset clearly improves the processed image's overall quality.

The step and offset parameters must be specified when the algorithm begins. The initial values can be set by the operator or determined automatically by examining several test images during an initialization procedure. While the algorithm is running, the values for step and offset can change dynamically depending on the post-line-extraction image-analysis results.

The intensity-based line-extraction procedure we've described has many advantages over edge-detection techniques. Intensity transitions between different background colors or between shadowed regions of the image do not affect the line-extraction procedure. The procedure can also effectively determine when there is no line in the image. As the vehicle traverses the course, we must consider the possibility that a line might leave the camera's field of view. Because this is a monocular vision system, the inside or outside line might get "lost" while the vehicle makes a tight turn. It is essential to recognize that the line is outside of the field of view rather than identifying noise in the image as a false line.

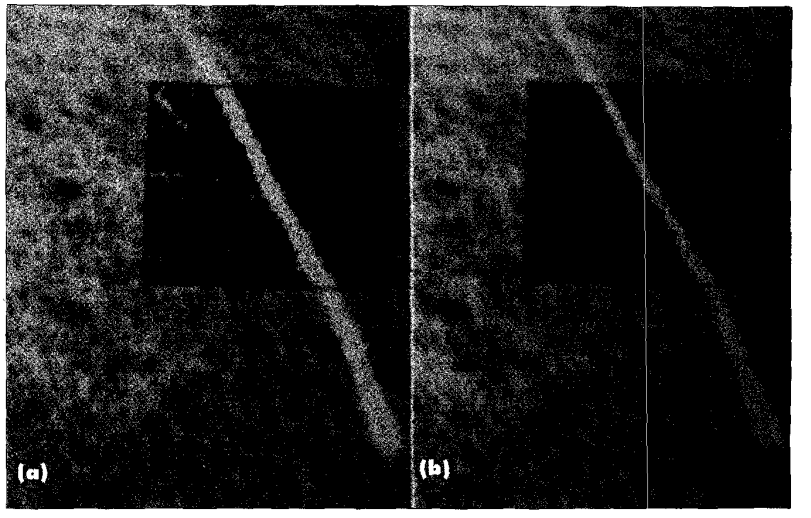

Figure 7. Thresholded images: (a) a noisy line and (b) a clean line.

As we've noted, the intensity-based line-extraction procedure uses an offset value to move the threshold away from the transition region. By counting the number of pixels that occur above the threshold, we can determine if no line is present in the image. The presence of a sufficient number of pixels above the threshold, however, does not guarantee that there is a usable boundary line in the image. A post-line-extraction error analysis will help determine if the remaining pixels actually form a coherent line. It is important to monitor the pixel count and determine if a line could not be in the image. If there is no possibility of a line being in the image, we can abort all subsequent processing to speed up the entire system.

Also, the intensity-based line-extraction method requires only a small amount of computational time to identify the line. In our testing, the intensity-based approach proved to be far more robust than simple edge-detection techniques that ran in a comparable time—and far more efficient than broad edge-masking techniques.

**Post-extraction noise reduction.** Once the binary threshold finishes, the image should contain a collection of points that correspond to the image's most intense continuous regions. The implementation of two post-line-extraction algorithms cleans up the image by removing spurious pixels that were above the threshold value but do not indicate a line. The post-extraction noise reduction is simple for the intensity-based system. The thresholding procedure should leave a collection of pixels of generally consistent shape and density that compose the line. To clean up the image, we implement two filters to remove groups of pixels either too small or large to be part of the line. However, in the case of edge detection, the intrinsic image has much less structure when the original image is very noisy. This complicates the ability to clean up the intrinsic image and extract the line.

First, a simple convolution mask runs over the image to remove pixels that are isolated from large groups of pixels. The second algorithm runs over the image to detect and remove pixel collections that are too large to be part of a line.

This routine successfully removes high-intensity obstacles that appear as large blobs in the image. In our current vehicle, however, the computer-vision system is not responsible for detecting obstacles. Rather, we use an array of ultrasonic range finders placed on the front of the vehicle to determine the presence of obstacles. Having completed these steps, the algorithm counts the pixels in the image and compares the result to a lower and upper bound to determine whether a line may be in the image. We select the bounds in the initialization procedure by evaluating several test images.

**Locating the extracted line in the ground coordinate system.** At this point, the processed image should contain pixels that constitute the course boundary. The next step determines the line's position and orientation in the image. To determine the line's position in the image-coordinate frame, we execute a linear fit of image pixels in the form $y = ax + b$. Using a total least-squares regression,[10] we can minimize the perpendicular error between pixels in the image and the fitted line. This procedure lets us quickly determine the line's position in the image and evaluate the error associated with the fitted line. The processing region's reduced size lets us use a linear fit to approximate the line due to the reduction in apparent curvature in the observation window. This simple fit is also less computationally intensive and less prone to noise than curve approximations using higher-degree polynomials.

In the summation equations to perform this linear fit in the described coordinate frame below, each pixel is determined by its position $(x_n, y_n)$ in the ROI:

$$\bar{x} = Mean(x),\ \bar{y} = Mean(y)$$

$$Sx = \sqrt{\sum (x - \bar{x})^2},\ Sy = \sqrt{\sum (y - \bar{y})^2}$$

$$y = ax + b$$

$$a = \mathrm{Sign}\left(\sum (x - \bar{x})(y - \bar{y})\right)\frac{S_y}{S_x}$$

$$b = \bar{y} - \mathrm{Sign}\left(\sum (x - \bar{x})(y - \bar{y})\right)\frac{S_y}{S_x}\bar{x}$$

Once we determine the fitted line's equation, we can evaluate the actual error associated with the fit and determine if the collection of pixels represents a reasonable line. The acceptable error is bounded by the maximum pixel width of the line in the image. If the error-valuation result is above this bound, we discard the image. This procedure can identify images that have passed all previous tests but contain small collections of pixels that do not lie on a line.

Once we've determined the line's location in the image-coordinate plane, we use an image-plane transform to determine the line's location in the global (ground-referenced) coordinate system. Using a monocular vision system, we cannot extract a full three-dimensional representation of the lines in the

ground-coordinate system without additional information. We obtain this information by assuming that a single plane represents the ground in front of the vehicle and that the camera angle with respect to this plane does not change. Others have made similar assumptions about the structure of the vehicle navigation environment to extract approximate 3D models of the local environment.[3,8] Figure 8 shows the reference used to generate the ground coordinate system and the relevant equations to perform this transform.

Once we've established the line's position in the global frame, the vehicle's navigation routine generates a steering angle for autonomous navigation. This transform depends on a constant value **I**, which represents the distance to the image plane. We can determine the constant by placing an object of known length on the ground at the image's center and measuring its apparent length in the $x$ direction. The image and ground dimensions used in these formula must have the same units, which requires a conversion from pixels to inches in the image plane. This conversion depends on the camera's resolution and the digitized image's aspect ratio.

We make a final line evaluation at this point to determine if the line in the image is acceptable for navigation. Up to this point, a line has passed all pixel-count tests and the error evaluation. To further validate the line, we can compare its current position and orientation with an expected position and orientation determined by the previous images. As a vehicle traverses the course, there are obvious limitations to the change in apparent line position between successive images. By comparing the current line to lines in the previous images, we can determine if the current line represents a physically impossible change in vehicle position. We implemented this check to help discard images that contain a line-like pattern of glare or reflection that is brighter than the course boundary line.

## Vehicle control and navigation

Once we've processed the image, we'll know the positions of the boundary lines relative to the vehicle. At this point, a separate navigation routine can generate a steering angle to follow the course. With the position of the lines known, we can implement many different navigation algorithms independent of the line-identification technique. The navigation routine currently implemented on Christine uses the lines' position and orientation to generate a virtual center of the course. The algorithm predicts the expected course center and generates a steering angle that directs the vehicle toward this point. The steering angle updates with each successive image, provided a navigable line has been extracted. We have also built several safeguards into the algorithm to handle the situation where several successive images fail to produce an acceptable line.

Of course, implementing a successful navigation routine depends on many other factors, including the base vehicle's capabilities, the integration of other sensor information, and the specific operating environment. A significant amount of research has gone into computer-vision-based roadway navigation, and an assortment of viable navigation strategies have been developed.[1–3,8]

We developed the image-processing structure and navigation algorithm presented here using Christine. The intensity-based line extraction has provided a stable computer-vision system capable of working in many different environments. We've tested the system under a variety of conditions, including bright, direct sunlight. In addition to the improved line-extraction method presented here, we've refined the pre- and postextraction processing steps to compensate for some of the specific conditions that traditionally cause navigation problems in outdoor autonomous vehicles. To date, we've found no exceptions to the assumptions made in developing this vision system.

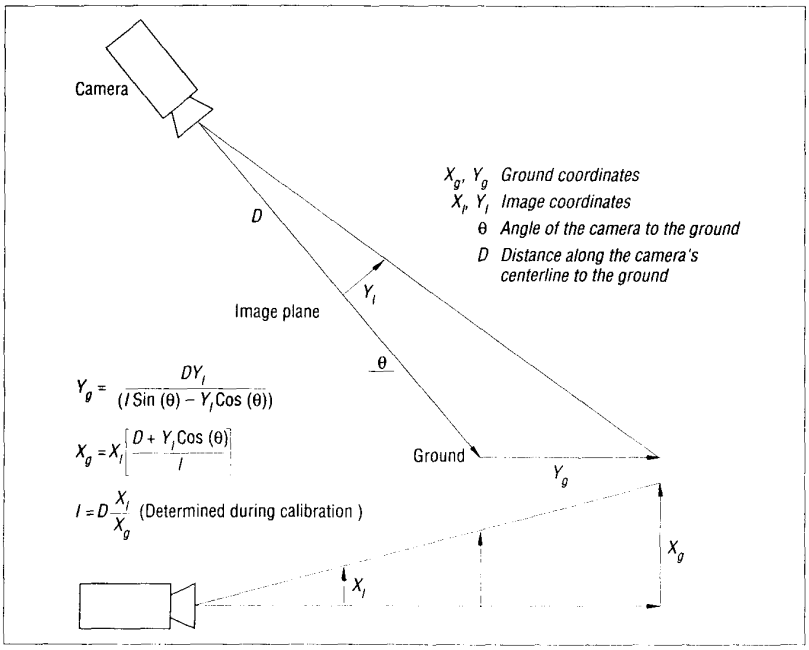


Figure 8. Top and side view of the camera position with image and ground planes.

## Use of passive optical devices for improved algorithm performance

We have also experimented with various optical filters and polarizers to passively remove image noise and improve image quality under specific conditions. We have found the CCD camera to be very sensitive to the radiation in the infrared range. Reflection of infrared light off the grass or asphalt increases the intensity of the background in the image and distorts the intensity separation between the lines and the image background. The use of an IR cutoff filter or hot mirror improves image integrity. To aid in the line-extraction process, we would like to use as many passive optical image-processing techniques as possible. Reducing the image bandwidth before it is digitized reduces the overall software-processing time and the probability of error during subsequent processing.

IN COMPUTER-VISION-BASED NAVIgation of autonomous robotic vehicles, edge-detection techniques seem to be a natural approach to the problem of finding roadway boundaries. Nevertheless, these methods have proven unreliable or computationally intensive and complex when extracting lines painted on grass or rough asphalt. System developers have used standard performance measures to compare the performance differences between edge-detection and intensity-based methods.[5,6] We believe that the simple intensity-based approach to boundary-line extraction we describe in this article is functionally superior to and computationally simpler than edge detection, given the structure of the operating environment. Extensive testing, conducted on Christine, a small autonomous vehicle developed for the Unmanned Ground Robotics Competition, supports this opinion. We continue to develop and refine these algorithms as part of our ongoing autonomous vehicle program. Inexpensive color computer-vision systems and improved data-transport and processor speeds will open many new avenues of investigation. ■

## References


1. Y.-F. Du et al., "ALX: Autonomous Vehicle Guidance for Roadway Following and Obstacle Avoidance," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics*, IEEE Press, Piscataway, N.J., Vol. 1, 1995 pp. 364–370.
2. M.E. Boudihir, M. Dufant, and R. Husson, "A Vision System for Mobile Robot Navigation," *Robotica*, Vol. 12, 1994, pp. 77–89.
3. J.M. Sanchiz, F. Pla, and J.A. Marchant, "A Framework for Feature-Base Motionrecovery in Ground Plane Vehicle Navigation," *Proc. Computer Analysis of Images and Patterns '97*, Springer-Verlag, New York, 1997, pp. 686–693.
4. R. Schuster, N. Ansari, and A. Bani-Hashemi, "A Hierarchical Edge Detector Using the First and Second Derivative Operators," *Proc. SPIE*, Int'l Soc. Optical Engineering, Bellingham, Wash., Vol. 1825, 1992, pp. 230–241.
5. J.F. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. PAMI*, Vol. 8, No. 6,1986, pp. 679–698.
6. K.P. Lam and A. Furness, "K-AVE+GNN+SOBEL = An Affective Highly Parallel Edge Detector Approach," *Proc. SPIE*, Int'l Soc. Optical Engineering, Vol. 3166, pp. 190–198.
7. R. Lewis, *Practical Digital Image Processing*, Ellis Horwood Ltd., Chichester, UK, 1990.
8. K. Yamada and T. Ito, "Image Understanding Based on Edge Histogram Method for Rear-End Collision Avoidance System," *Proc. Vehicle Navigation and Information Systems Conf. '94*, IEEE Press, 1994, pp. 445–450.
9. L.S. Davis, "A Survey of Edge Detection Techniques," *Computer Graphic and Image Processing*, Vol. 4, 1975, pp. 248-270.
10. M.I. Friswell and J.E. Motorshead, *Finite Element Updating in Structural Dynamics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.



**Chris Roman** is a graduate student in the Structural Systems and Control group at the University of California, San Diego. His technical interests include the control of dynamic systems and robotics. He completed two years of autonomous vehicle research while attending Virginia Polytechnic Institute. He received a BS from Virginia Tech and is a member of the American Society of Mechanical Engineers. Contact him at AMES, 9500 Gilman Dr., La Jolla, CA 92093-0413; croman@ucsd.edu.

**Charles F. Reinholtz** is the W.S. White Chair for Innovation in Engineering Education and assistant head of the Department of Mechanical Engineering at Virginia Tech. He is the coauthor of *Mechanisms and Dynamics of Machinery* (John Wiley). He is currently coadvisor to the Autonomous Vehicle Team. He received his BS, MS, and PhD in mechanical engineering from the University of Florida. He is a member of the ASME and ASEE. Contact him at the Dept. of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061-0238; creinhol@vt.edu; http://www.me.vt.edu/ME/peoplefaculty/reinholtz.html.