Comparison of the Roberts, Sobel, Robinson, Canny, and Hough Image Detection Algorithms

John Burnham, Jonathan Hardy, Kyle Meadors

Image Processing Group
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, MS 39762
434 Simrall, Hardy Rd.

{jab2, jeh3, kam1}@ece.msstate.edu

ABSTRACT

This paper presents a comparison and evaluation of the Roberts, Sobel, Robinson, Canny, and Hough image detection algorithms based on their ability to detect red squares on a black and white background. Inspired by the Institute of Electrical and Electronics Engineers (IEEE) Southeastcon student hardware competition, the algorithms are tested on an image database comprised of gray scale images taken from a test platform containing red squares on a black and white background. The success of each algorithm is based on its accuracy in detecting the red squares within the images from the database. The results of the algorithms are then compared statistically to determine which one is the best suited for this application.

I. INTRODUCTION

The motivation for this paper was inspired by the IEEE student hardware competition. Southeastcon is the annual technical conference of IEEE Region 3. Its purpose is to bring together both professional and student electrical engineers from the southeastern part of the United States and the rest of the world [1]. Among the many technical sessions offered at the conference is the student hardware design competition. For 1998 competition, an autonomous robot was to be designed that would seek and deactivate infrared lights located at the four corners of a square playing surface. As a deterrent, “mines” are placed on the playing surface that penalize the robotic team. The rules concerning the playing surface and the mines are as follows:

“The competition will take place on a 8′ x 8′ flat black playing surface. All other parts of the playing surface will also be flat black (Rust-Oleum Flat Black, #7776). The playing surface will be surrounded by 6" high walls which will be painted flat black. A rectangular grid will be painted on the surface with parallel lines being 8" apart and each line being 1/2" wide using gloss white (Rust-Oleum Gloss White, #7792). The first white line in both the

horizontal and vertical directions will be centered at a distance of 8" from the wall. A 8" square will be painted about each fixed mine. The square will be painted such that its lines are perpendicular to the grid lines which they cross. The lines will be pained dark red and will not cover any existing white lines. The starting square will be designated as one of the red mine squares located closest to the wall. Only one square will be chosen for the competition.

Each mine will occupy an intersection of two white lines. The mines will use an optical sensor to detect the presence of the vehicle. A 1/2" diameter circle of Plexiglass will be centered above the sensors. The Plexiglass will be mounted flush with the playing surface. The fixed mines are located as follows:

4 mines each located at the intersection of the third line away from each wall.

4 mines each located at intersection of the center lines and the first line away from each wall." [1]

A possible solution to the problem posed by the mines would be the design of an image detection system. The system would include some type of camera and image detection software. Images from the playing surface would be captured by the camera and then processed by the software. The software would determine the presence, if any, of a red mine within the image. This information would then allow the robot to adjust, if necessary, its direction on the playing surface so as to avoid the penalizing mines. It is the image detection software which is the interest of this paper. We wish to investigate the possibility of using some standard image detection algorithms, the Roberts, Sobel, Robinson, Canny, and Hough, for implementation with the design robot. While this is not necessarily the best solution to the problem of mine avoidance, conclusions from this paper will likely indicate whether or not an image detection system is feasible for this project.

II. THEORY

This section of the paper will introduce some background into image detection and give detailed explanations of the algorithms themselves. Also, information concerning the frequency response of the algorithm's masks will be given.

A. Image Detection

The first step in analyzing images is the separating, or segmenting, of the objects within the image. Segmentation algorithms allow for distinctions to be made between two or more objects.[3] Segmentation is based upon two concepts: similarity and discontinuity. If an image is converted to gray scale, i.e., colors are separated into distinct shades of gray, a boundary of an object can be noted by a sudden change in the gray level. This discontinuity in the image could be either an isolated point or a line or edge of an object. It is the purpose of a segmentation algorithm to accurately locate these discontinuities. Segmentation algorithms can be divided into three separate types based on the discontinuities that they locate: point detection, line detection, and edge detection.[2]

Point detection is the simplest of the detection techniques but provides the least information. A point will have a drastic change in gray value from its neighbors. Therefore, if a pixel's value differs from those of its neighbors by more than some threshold amount, it can be considered a point.[2]

Line detection is a more complicated process. It involves finding pixels that are likely to be

parts of lines and testing them to see if they are part of a common line. One such process, the Hough Transform, is described in the Algorithm section below.[2]

Edge detection is the attempt to find discernible changes in contrast in two dimensions. This approach is most appropriate for this particular project, and thus most of the algorithms in this project fall into this category.[2.

Most segmentation algorithms use a mask on the image's pixels for the detection of a discontinuity. Each pixel and its neighboring pixels have its gray level value multiplied by a mask value. The sum of these values represent that point's mask response. An example could be the following:

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \bullet \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix} =$$

$$R = m_1 p_1 + m_2 p_2 + \ldots + m_9 p_9 = \sum_{i=1}^{9} m_i p_i$$

Figure 1 Example of a mask response.

Here, m_i is the mask value for a pixel, and p_i is the gray level value for a pixel. R is the mask response for the pixel which the mask is centered around (p_5). By sweeping this mask across the image row by row, a new array is created. It is the same size as the original image but contains the values of the mask response instead of the pixel value. These mask response values can then be compared to a minimum threshold value to determine which pixels are more likely to be part of an edge. This threshold can be adjusted to vary the selectivity for edge pixels, allowing a user to "tune" the algorithm for optimal performance for a given picture. [2]

Consider the example mask responses shown in Figure 2 and Figure 3 as examples. Let the threshold value be 20. Figure 2 shows the mask response for a point that is continuous. The image matrix represents the pixel values of a 3x3 portion of an image. Since the outside values of the mask response (summed to -16) cancel the center value (+16), no point of discontinuity is observed.

Point Detection Mask	Image	Mask Response
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} -2 & -2 & -2 \\ -2 & 16 & -2 \\ -2 & -2 & -2 \end{bmatrix}$

Figure 2 Mask response for a continuous set of pixels.

However, Figure 3 demonstrates a mask response for a point of discontinuity. In this case, the center value of the mask response (+64) and the outside edge values (-16) sum together for a value of +48. Since this value is greater than 20, the center pixel in the image

matrix is a point of discontinuity.

Point Detection Mask	Image	Mask Response
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 8 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} -2 & -2 & -2 \\ -2 & 64 & -2 \\ -2 & -2 & -2 \end{bmatrix}$

Figure 3 Mask response for a discontinuous set of pixels.

B. Algorithms

Based on the uniformity of the grid design on the board, it was decided that several conventional segmentation algorithms would work well to find the borders of the red mine field. Many such algorithms have been developed, mainly differing in the masks used to determine the chance of a pixel being an edge pixel. The four edge detection algorithms used in this project are some of the more widely known image detection algorithms and were chosen based on their different strengths and weaknesses.

The **Roberts Operator** is one of the oldest and simplest edge detection algorithms to implement. It uses two 2x2 matrices to find the changes in the x and y directions:[9]

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1}$$

$$G_x \qquad G_y$$

To determine whether the pixel being evaluated is an edge pixel or not, the magnitude of the gradient is calculated as follows:

$$|G| = \sqrt{G_x^{\,2} + G_y^{\,2}} \tag{2}$$

If the calculated magnitude is greater than a minimum threshold value (set based on the nature and quality of the image being processed), the pixel is considered to be part of an edge. The direction of the edge's gradient, perpendicular to the direction of the edge itself, is found with the following formula:

$$\alpha = \operatorname{atan}\left(\frac{G_y}{G_x}\right) \tag{3}$$

The small size of the masks for the Roberts Operator make it very easy to implement and quick to calculate the mask responses. However, these responses are also very sensitive to noise in the image.

The **Sobel Operator** uses two masks to find vertical and horizontal gradients of edges. The masks for the Sobel are as follows[8]:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{4}$$

$$G_y \qquad G_x$$

The formula for finding the magnitude of the response and the angle of the gradient is the same as for those in the Roberts Operator. Because the masks are 3x3 rather than 2x2, the Sobel is much less sensitive to noise than the Roberts, and the results are more accurate. The drawbacks of using the Sobel operator are that effects of an edge are spread out over a 3x3 pixel area, and the computation of $|G|$ is fairly involved. Therefore, in practice, $|G|$ is often

approximated as the following:

$$|G| = |G_x| + |G_y| \quad (5)$$

The **Robinson Operator** is similar in operation to the Sobel operator but uses a set of eight masks, four of which follow:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \quad (6)$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

The other four are simply negations of these four, and thus the computation is simplified. The magnitude of the gradient is the maximum value gained from applying all eight masks to the pixel neighborhood, and the angle of the gradient can be approximated as the angle of the line of zeroes in the mask yielding the maximum response. This algorithm increases the accuracy of $|G|$ and a but requires more computation than both the Roberts and Sobel algorithms[6].

The **Canny Edge Detector** has its basis in a slightly more visual approach. If one considers a one-dimensional step edge change in contrast and then convolves that edge with a Gaussian smoothing function, the result will be a continuous change from the initial to final value, with the slope reaching a maximum at the point of the original step. If this continuous slope is then differentiated with respect to x, this slope maximum will become the maximum of the new function again at the point of the original step.

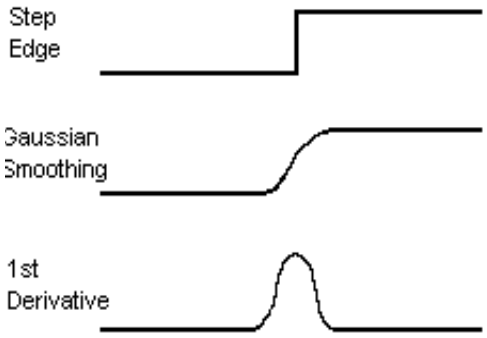

Figure 4 Steps in 1-D Canny edge detector.

Because the derivative of the convolution of the Gaussian and the image is the same as the convolution of the derivative of the Gaussian and the image,

$$D[Gauss(x, y) \otimes Img(x, y)] = D[Gauss(x, y)] \otimes Img(x, y) \quad (7)$$

a mask can be created that represents the first derivative of the Gaussian. The maximums of the convolution of the mask and the image will indicate edges in the image. This process can be accomplished through the use of a two-dimensional Gaussian function or the combination of a one-dimensional Gaussian function in both the x- and the y-directions. The values of the differentiated Gaussian mask depend on the choice of sigma in the Gaussian

equation:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x}{2\sigma}}$$
$$G'(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x}{2\sigma}} \quad (8)$$

The computational intensity of the Canny Edge Detector is relatively high, and the results are usually post-processed for clarity. However, the algorithm is very effective in processing noisy data or images with fuzzy edges.[11][12]

The **Hough Transform** is ideal for line detection if little is known of the location of an image but its shape can be represented by a mathematical formula. By considering the equation of a line, $y = mx + c$, all possible lines that could pass through a single point in a image, (x',y'), can be represented in the form of $y' = mx' + c$. If (x',y') are considered fixed, then m and c are now the variables in what is called the parameter space. If (x',y') lie on line *AB*, then every point of line *AB* will have a common point of intersection in the parameter space, (m',c'). With this relationship between the image space and the parameter space established, the Hough Transform can be applied. By considering the maximum and minimum values of both *m* and *c*, an array of *H(m,c)* can be created with all elements initialized to zero. For every available point in the image space, the gradient is computed. If the gradient exceeds a defined threshold, then all elements of *H(m,c)* that pertain to that line are incremented. The local maxima of the array now represent the points of a line in the image.[3]

C. Frequency Response

The following image shows the frequency response of the Dy Sobel mask.

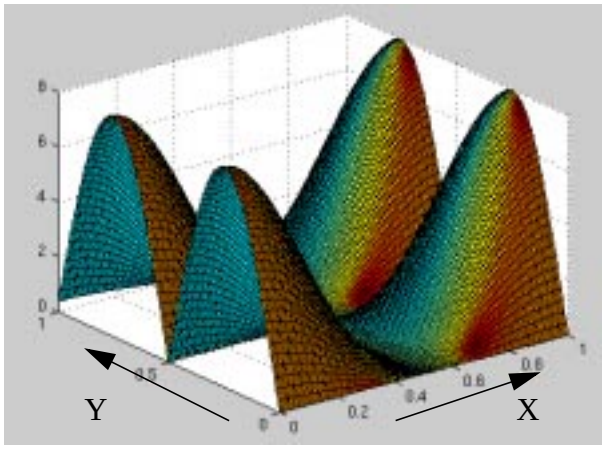

Figure 5 Frequency Response for Sobel Dy

By following the curve along the y-frequency direction, one can see that the response rises sharply with frequency from 0 to about Fs/4, indicating a differentiator effect, then declines again through Fs/2, showing an attenuation of relatively high frequencies. This shows that the filter should respond strongly to line changes in the y direction, but filter out some of the higher frequency noise in the image. Likewise, if one follows the curve along the x-frequency direction, it is apparent that the response in this direction is zero, given no change in y.

The frequency response of the Dx Sobel mask

shows a similar operation. The response along

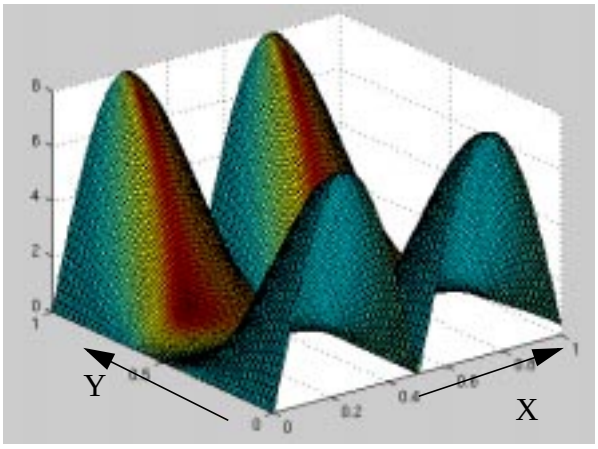

Figure 6 Frequency Response for Sobel Dx

the y-frequency direction's edge shows a zero response. The response along the x-frequency direction's edge shows the differentiation of the low- to mid-frequency signal components of an edge, and the attenuation of high-frequency signals.

The frequency responses of the diagonal masks in the Robinson algorithm are shown below. The effects of the filtering are the same

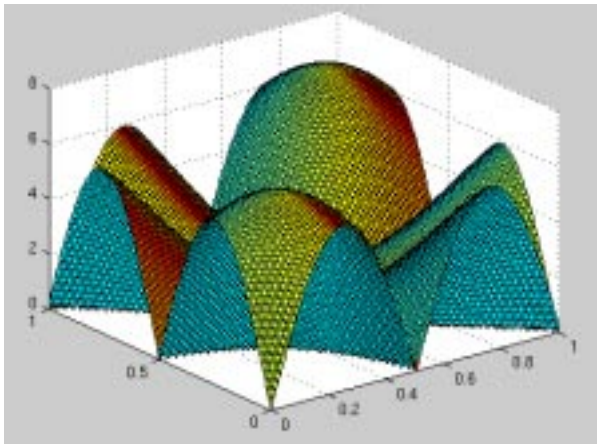

Figure 7 Frequency Response for Robinson at 45 degrees

as above, with the angles of the responses differing accordingly. The first image shows the mask with zeros at a 45 degree angle. Following a path from the origin of graph to the opposite corner, corresponding to equal increases in x and y, yields the same differentiation and attenuation pattern seen before. The same pattern can be seen in the second diagonal frequency response graph by following a path from (1,0) to (0,1), which corresponds to a line in the image at 135 degrees.

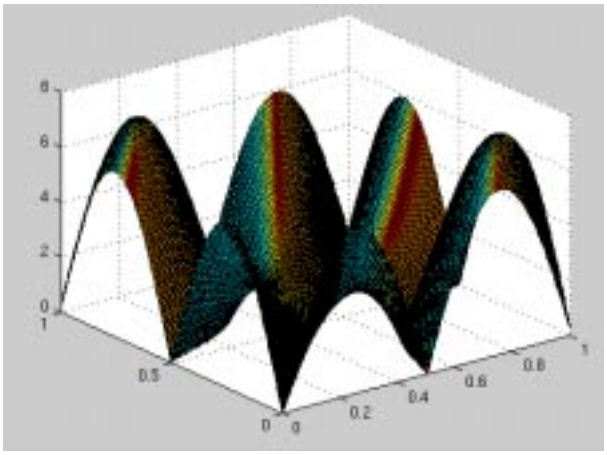

Figure 8 Frequency Response for Robinson at 135 degrees

The next two images are the frequency responses of the Roberts 2x2 masks. It quickly becomes apparent that these masks actually respond best to diagonal edges, rather than vertical and horizontal as indicated in the research. It can also be seen from examining the frequency range of the lobes that these masks will be more susceptible to high-

frequency noise, as expected.

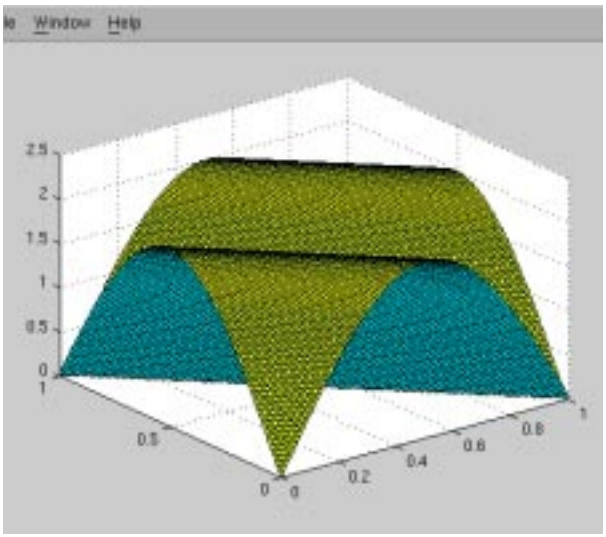

Figure 8 Frequency Response for Roberts Dy

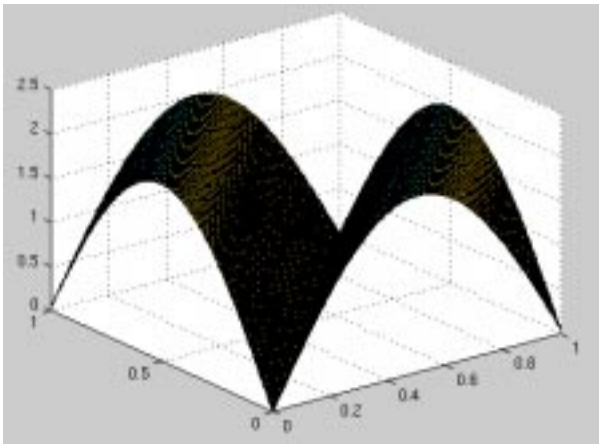

Figure 9 Frequency Response for Roberts Dx

Next, the frequency of the Canny Dy and Dx masks are shown. Again, the differentiation characteristics can be seen in the shape of the lobes. However, with the Canny masks, this action occurs on a much lower frequency range, and the high frequency attenuation is very pronounced, with nearly complete attenuation from about Fs/5 to Fs/2. This provides much less sensitivity to noise than any of the other approaches, while the steep differentiation still detects the lower frequency components of the edges.

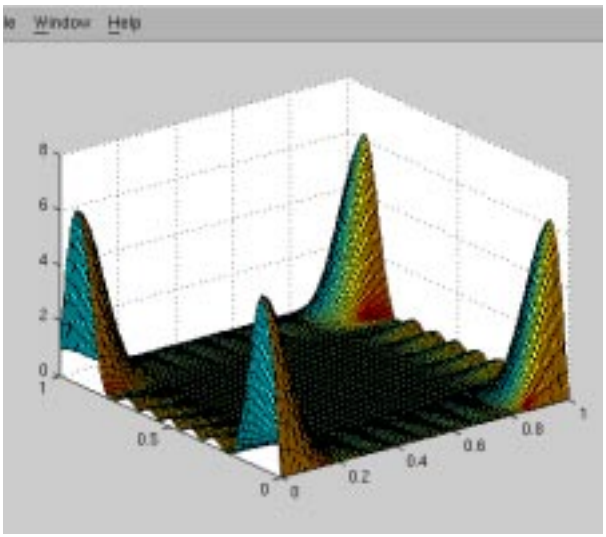

Figure 10 Frequency Response for Canny Dy

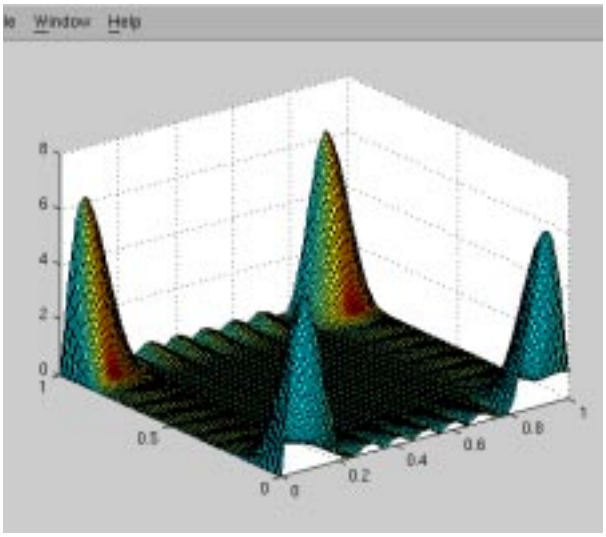

Figure 11 Frequency Response for Canny Dx

The final four images demonstrate the effects of varying x and sigma in the calculation of the

Gaussian functions for the Canny. In the first image, the filter width is changed from 10 to 5. This introduces more high-frequency components, basically because the image is not being smoothed out as much in the convolution. The second image shows the effect of a filter width of 15. The higher frequencies are almost completely stopped, leaving only the very lowest frequencies in the edge to be differentiated. The third image shows the effect of leaving the filter width 10 and changing sigma from 3 to 1. It is obvious that setting the value of sigma too low compared to the filter width produces undesirable effects. For the last image, the value of sigma was set to 5. The ripple in the higher frequencies is diminished considerably,

and the differentiation peaks are very sharp.

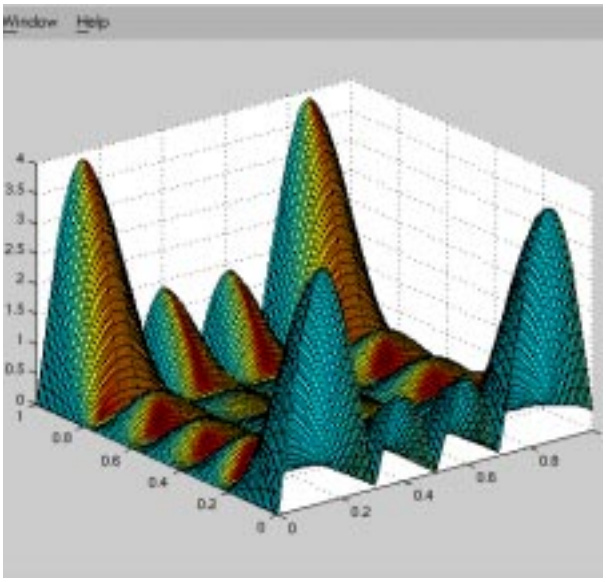

Figure 12 Frequency Response for Canny Dx with width 5

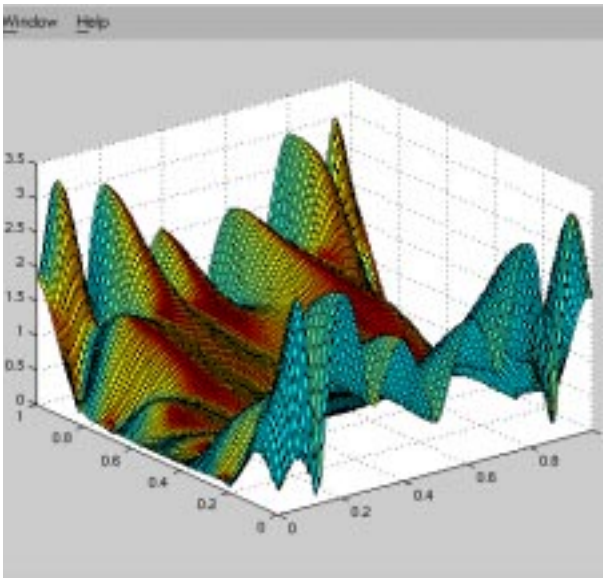

Figure 14 Frequency Response for Canny Dx with sigma 1

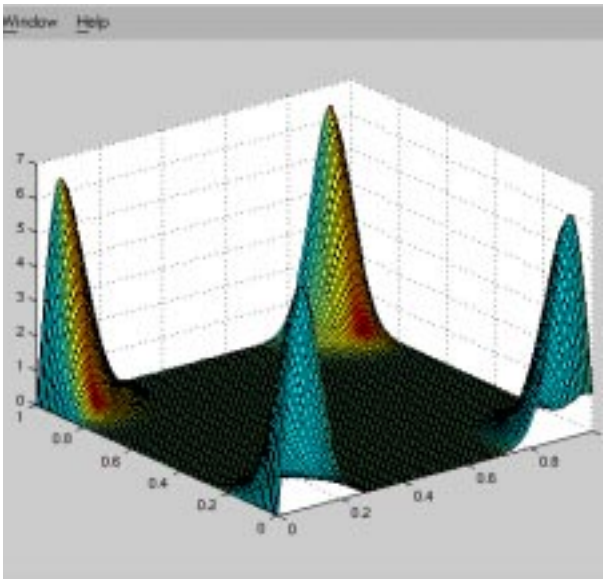

Figure 13 Frequency Response for Canny Dx with width 15

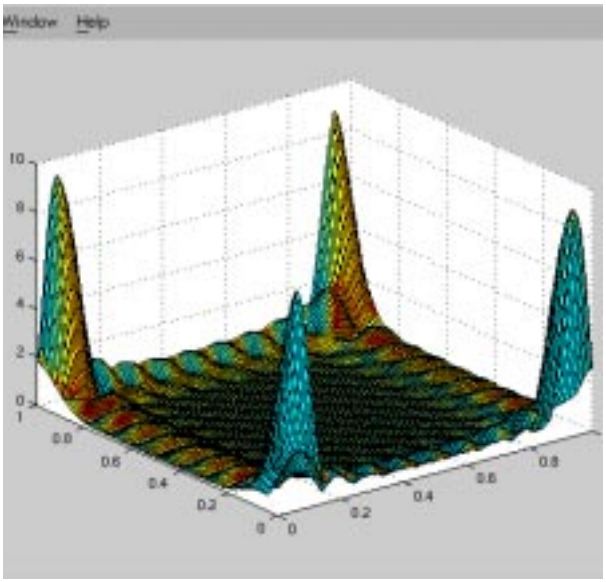

Figure 15 Frequency Response for Canny Dx with sigma 5

The effects of these parameter variations were

seen in practice when testing various values for x and sigma in the Canny implementation. For smaller sigma values, there was a slight response to noise in the image, and increasing the value of sigma reduced this response. Also, the noise was reduced if the width of the filter was increased.

III. IMPLEMENTATION

The implementation of our project was divided into three sections: completion of a comprehensive image database, development of algorithms and edge detection software, and testing of algorithms on images from database.

A. Image Database

Before the image database could be constructed, the test platform had to be designed, and some hardware and image capture software had to be constructed so that images could be taken from the platform.

First, a mobile test platform was constructed according the specifications of the hardware competition rules. After finishing the platform, attention was turned to the task of capturing images. Images could be captured in either color or gray scale. Gray scale was chosen because of the relatively low complexity of the images. Each image would ideally have only three colors: black, white, and red. Even when noise and image capturing inaccuracy are included, the colors should be different enough to be easily separated. With so few colors in the images, color images would introduce unnecessary information. Gray scale images would be easier to process and would give results comparible to color images.

For the actual capturing of the images, a CCD gray scale camera was connected to an image capture card in a PC. The software provide with the image capture card allowed the storage of the images to Windows Bitmap format. The images were converted from Windows Bitmap to PGM using shareware image conversion software. The reason for converting the format of the images will be explained later in the paper.

Once we were able to take images from the test platform, the database was constructed. To derive an accurate conclusion from our results, a comprehensive database was to be constructed. A total of 80 images was included in the database. Sixty of those images were used as training data for the algorithm, and the remaining 20 were reserved for performance evaluation of the algorithms.

The images were divided equally into two main groups: images taken under controlled lighting conditions and images taken in ambient lighting conditions (Table 1). Forty unique images of a specified orientation were taken under controlled lighting, and then another 40 images were captured under ambient lighting. For the controlled lighting conditions, the CCD camera was shrouded from the fluorescent lights. This provided an image with nearly equal brightness level at all points. For the images taken under ambient light, no special considerations were made to alter the brightness level of the image.

The contents of the images were also varied for a more complete evaluation. 3 sets of images were taken with an individual “mine” covering 1/4 or more, 1/16, and 1/64 of the camera’s viewing area, respectively. Each set of images was further divided into a subset based on the angle of the mine with respect to the camera, θ. The subset had 4 images taken at $\theta = 0^{\circ}$, $\theta = 45^{\circ}$, and $0^{\circ} < \theta < 45^{\circ}$, respectively. The final four images taken contained no portion of a red square. This made a total of 40 different image orientations.

With images taken under both ambient and controlled lighting, a total of 80 real images were included in the image database. In separating the testing images, one image of each type was used for a total of 20 test images.

% of Mine Within Image	Angle of Square	Number of Images (Ambient/ Non-ambient)
>=1/4	$\theta = 0^\circ$	4/4
	$\theta = 45^\circ$	4/4
	$0^\circ < \theta < 45^\circ$	4/4
1/16	$\theta = 0^\circ$	4/4
	$\theta = 45^\circ$	4/4
	$0^\circ < \theta < 45^\circ$	4/4
1/64	$\theta = 0^\circ$	4/4
	$\theta = 45^\circ$	4/4
	$0^\circ < \theta < 45^\circ$	4/4
no square	n/a	4/4
Total Images		80

Table1 Image Database Breakdown.

B. Software

The implementation of the algorithms and edge detection software consisted of three steps: reading in and writing out of images in PGM binary format, development of the Roberts, Sobel, Robinson, and Canny edge detection algorithms, and development of the Hough Transform algorithm. All code was written in C++ and compiled under the Solaris 2.51 using a g++ compiler.

1. PGM Format

In order to use these algorithms on an image, the format of the pixels and other information pertinent to the image itself had to be known. Otherwise, the algorithms would not be able to recognize the images. Rather than attempt to use the images in Windows Bitmap format, these images were converted to gray scale PGM format. This decision was made for a several reasons. One nice feature of PGM is its relatively simple format. Because it has very little header information, accessing the actual image data in a PGM file is greatly simplified. Another consideration dealt with the demonstration aspect of our project. For the demonstration, the TCL script programming language was selected. A drawback to Tcl is its inability to display Windows Bitmap images. It could, however, read PGM. With the image format set, the images could now be read and processed.

2. Edge Detection Algorithms

Once images were readable, we began implementing the various processing algorithms. The first three, the Roberts, Sobel, and Robinson, were fairly simple. As the image array was scanned pixel by pixel and row by row, the mask for each algorithm was multiplied by each pixel and its neighbors. The iterations began and ended one pixel in from each of the edges to avoid using neighbor pixels that were outside the image array. The summation of these products of course became the mask response for pixels. For the Roberts and Sobel algorithms, the *Dy* and *Dx* mask responses were calculated for each pixel, and

the sum of their absolute values was placed into a new array representing the pixel response magnitude. For the Robinson algorithm, all four mask responses were calculated, and the maximum of these four responses was placed into the response magnitude array. Once the entire image array had been processed, the magnitude array was thresholded to filter out any small responses and written out to a new PGM image file.

The Canny algorithm was implemented differently than the previous three algorithms. The first step was the calculation of the values for both the Gaussian smoothing and the derivatives of the Gaussian. It was decided to allow the values of *x* and *sigma* be set dynamically by the user, and so the values being multiplied by the pixel values could vary. A two-sweep 1-dimensional approach rather than a 2-dimensional single convolution of a mask was used. So once the Gaussian values were computed, the image array was smoothed in the *x*- and *y*-directions. These smoothed values were placed separately into two intermediate arrays. The arrays were then convolved with their orthogonal Gaussian derivatives, i.e., the *x*-smoothed array was convolved with the *y*-direction derivative values. This produced the *Dy* and *Dx* magnitude arrays, where the subscript *y* or *x* is the same as the direction of the Gaussian derivative used. These magnitude arrays were then processed to find their maxima, indicating which *x,y* locations contained likely edge pixels. The value of these maxima were then written out to a new PGM image file.

3. Hough Algorithm

The output of the edge detection algorithms indicated pixels that were likely edges. However, the pixels still had to be grouped together to form an edge. For this step, the Hough Transform was used.

The Hough algorithm was implemented by reading in the output image from one of the edge detection algorithms. To insure that only pixels that had a high likelihood of being edge pixels were considered, a minimum threshold was set to filter out very low response pixels from the input image. An accumulator array was created with a height of 180 and width equal to the maximum value calculated for *r* in the line equation:

$$r = x\sin\theta + y\cos\theta \qquad (8)$$

The height of 180 corresponds to the number of angle values that would be processed. The thresholded input array was then swept through, and any time a non-zero pixel value was encountered, the *r*-value was calculated for all integer values of θ between 0 and 180. These *r*-values were made into integers, and the accumulator pixels with indexing $[\theta][r]$ were incremented by 1. Any input pixels that shared the same line would increment the same r,θ pair, so the accumulator array locations that were maximums would likely represent lines in the original input image. Once the input image array had been processed, the accumulator array was searched for local maxima, and the r,θ values at these points were used to calculate *x,y* endpoints for these likely lines.

One problem with the Hough line detector is that is can not distinguish between line segments that terminate within the image and line segments that traverse the entire image. Therefore, the endpoints calculated from the r,θ pairs ran from one side of an image to the other, regardless of whether their representative line segments actually did or not. We tried to deal with this overshooting problem by splitting the original image into smaller sections and running the Hough transform on these sections. For each section, the line segments contained and the corresponding local endpoints were found. In

any endpoint in one section was sufficiently close in angle and x,y proximity to an endpoint from another section, the two segments were combined into one. Another problem with the Hough is that it sometimes doesn't detect shorter line segments in the image. This is because there are few pixels along that line to contribute to the accumulator array. It was hoped that splitting the image would solve this problem because the short segment would probably be a significant part of a smaller section. The approach met with some limited success in avoiding the overshoot problem by not detecting edges past where they actually terminated. However, poor local representation of the line significantly increased the variance in the angles detected. Also, the sensitivity to noise, smoothed over by the larger magnitude of the global Hough transformation, became a factor by increasing false edge detection. Finally, parts of the line segments were occasionally omitted in the output from this algorithm. On examining these effects, this approach was abandoned.

C. Testing Procedure

In evaluating the algorithms, the success of each algorithm was determined by its robustness in detecting edges. The evaluation of the algorithms was done statistically two ways. One method of evaluation was based on a point method. A correctly identified edge was given a +1 point, and a incorrectly identified or undetected edge was given a -1 point. A successful edge detection was the placing an edge within 5 pixels of its actual location. The point totals were summed together for a final point value. The second method of evaluation was an edge detection percent error. The number of correctly identified edges was divided by the total number of edges providing the percent error in detection.

Each edge detection algorithm required some thresholding arguments. All of the algorithms used a upper and lower input threshold. The Roberts, Sobel, and Robinson also used a output threshold. The Canny did not need an output threshold but did require a filter width and sigma value. The Hough used an input and output threshold value. During the training of the algorithms, different values for each of the arguments were used to find optimum values. Different values for the same argument were often needed on different images.

IV. RESULTS

The results of our project our listed in Table 3 and Table 4. Table 3 indicates the point totals for the algorithms. Table 4 gives the percentage error of the edge detection.

The filename represents the contents of the image. The format for the filename is as follows:

LXXYYn4

where:

L is the lighting conditions - a= ambient, c= controlled

XX is the percentage of the mine in the image - 04=1/4, 16=1/16, 64=1/64, 0=no square

YY is the angle of the mine - a0=0 deg., a4=45 deg., ao=0<theta<45 deg., 00=no square

Filename	Max Possible Points	Points Awarded for Algorithms			
		Roberts	Sobel	Robinson	Canny
a0000n4	1	1	1	1	1
a04a0n4	3	1	3	1	-1
a04a4n4	2	-3	1	1	2
a04aon4	3	-1	-2	-2	-1
a16a0n4	2	-2	-1	-2	-2
a16a4n4	2	-2	-2	-3	2
a16aon4	1	0	-1	-3	1
a64a0n4	2	-2	1	-2	-1
a64a4n4	2	-3	-2	-2	0
a64aon4	2	-1	-1	-3	2
c0000n4	1	-1	1	-1	1
c04a0n4	3	1	3	3	3
c04a4n4	2	1	2	2	2
c04aon4	5	-4	-4	0	3
c16a0n4	2	-4	0	0	0
c16a4n4	2	-4	0	-5	2
c16aon4	2	-2	0	-1	0
c64a0n4	2	-3	-6	-5	0
c64a4n4	1	-3	-2	-4	-4
c64aon4	2	-5	-3	-4	0
Total	**42**	**-37**	**-12**	**-29**	**10**

Table 2 Analytical Results

Lighting Conditions	Roberts % Error	Sobel % Error	Robinson % Error	Canny % Error
Ambient	74.9%	65.9%	69.8%	34.8%
Controlled	84.6%	53.0%	68.0%	30.0%
Average % Error	**79.75%**	**59.45%**	**68.9%**	**32.4%**

Table 3: Percent Errors

	Add./Sub.	Mult./Div.	Comparisons	Higher Order	Trig Function	Avg Time
Roberts	7*W*H	W*H	0	2*W*H	W*H	1.83s
Sobel	51*W*H	5*W*H	W*H	4*W*H	W*H	2.89s
Robinson	53*W*H	8*W*H	W*H	4*W*H	0	3.21s
Canny	181*W*H + 20*filterW	20*W*H + 32*filterW	2*W*H	2*W*H + 4*filterW	W*H	10.19s

Table 4: Analytical Results

V. CONCLUSIONS

In conclusion, the four edge detection algorithms, namely the Roberts, Sobel, Robinson, and Canny, have been evaluated in their effectiveness to detect red mines on a black and white background using an inexpensive gray scale CCD camera. As summarized in Table 3, the algorithm with the least percent error is the Canny edge detection algorithm. Table 4, however, demonstrates that the Canny also has the greatest execution time, on the order of 10 seconds for a 320x240 8-bit gray scale image. Even though it appears to have the most desirable results, the Canny is not particularly suited for realtime applications due to the enormous latency associated with it. This tradeoff is true for all the algorithms tested, in that the performance is inversely proportional to the execution time. As the execution time nears a realistic value for realtime operation, the performance of the algorithm drops to unacceptable levels.

Furthermore, it was observed that certain algorithms had inherent strengths and weaknesses. For example, the Roberts algorithm was fast, but generally found edges only on the 0 and 90 degree axis. The Sobel algorithm was able to find edges on the 0 and 90 degree axis, along with the 45 and 135 degree axis, but the execution time was slower than the Roberts. For an additional increase in execution time, the Robinson was capable of detecting an increased number of random-angle edges. The Canny appeared to be the best algorithm for the edge detection requirements of this application, but suffered from the worst execution time.

Therefore, it has been determined that, due to the performance versus accuracy tradeoffs involved, none of the four standardized edge detection algorithms tested are very well suited for this particular realtime application. This, however, does not rule out the possibility of designing a hybrid algorithm based on the principles discovered in this comparison which might be better suited for realtime operation.

VI. REFERENCES

[1] IEEE SouthEastCon 1998 Hardware Competition. Http://www-ece.engr.ucf.edu/secon98, 1997.

[2] Gonzales, Rafael C., Richard E. Woods. *Image Segmentation*. Addison-Wesley, Reading Massachusetts, 1992.

[3] Ballard, Dana H., Christopher M. Brown. *Computer Vision*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[4] Dougherty, Edward R., Charles R. Giardina. *Image Processing - Continuous to Discrete, Volume 1*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[5] Boyle, R. D., R. C. Thomas. *Computer Vision, A First Course*. Blackwell Scientific Publications, Oxford, 1988.

[6] Haralick, Robert M., Linda G. Shapiro. *Computer and Robot Vision*. Addison Wesley, Reading, Massachusetts, 1991.

[7] Fairhurst, Michael C. *Computer Vision for Robotic Systems, An Introduction*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1988.

[8] Russ, John C. *The Image Processing Handbook*. CRC Press, Boca Raton, Florida, 1995.

[9] Young, Tzay Y., King-Sun Fu. *Handbook of Pattern Recognition and Image Processing*. Academic Press, San Diego, California, 1986.

[10] Pitas, Ioannis. *Ditigal Image Processing Algorithms*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[11] *Edges: The Canny Edge Detector*, Http://www.icbl.hw.ac.uk/marble/vision/low/edges/canny.htm.

[12] *Basic Image Processing Demos*, Http://robotics.eecs.berkelee.edu/~mayi/imgproc/

[13] *Canny Edge Detection*, Http://www.icbl.hw.ac.uk/marble/vision/low/edges/canny.htm

[14] Http://trantor.cse.psu.edu/~jphelps/cse485/project2.html

[15] Http://www.cse.psu.edu/~caro/cse485/p2/project2.html

[16] Http://robotics.eecs.berkeley.edu/~mayi/imgproc/

Comparison of the Roberts, Sobel, Robinson, Canny, and Hough Image Detection Algorithms

John Burnham, Jonathan Hardy, Kyle Meadors

Image Processing Group
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, MS 39762
434 Simrall, Hardy Rd.

{jab2, jeh3, kam1}@ece.msstate.edu

ABSTRACT

This paper presents a comparison and evaluation of the Roberts, Sobel, Robinson, Canny, and Hough image detection algorithms based on their ability to detect red squares on a black and white background. Inspired by the Institute of Electrical and Electronics Engineers (IEEE) Southeastcon student hardware competition, the algorithms are tested on an image database comprised of gray scale images taken from a test platform containing red squares on a black and white background. The success of each algorithm is based on its accuracy in detecting the red squares within the images from the database. The results of the algorithms are then compared statistically to determine which one is the best suited for this application.

I. INTRODUCTION

The motivation for this paper was inspired by the IEEE student hardware competition. Southeastcon is the annual technical conference of IEEE Region 3. Its purpose is to bring together both professional and student electrical engineers from the southeastern part of the United States and the rest of the world [1]. Among the many technical sessions offered at the conference is the student hardware design competition. For 1998 competition, an autonomous robot was to be designed that would seek and deactivate infrared lights located at the four corners of a square playing surface. As a deterrent, “mines” are placed on the playing surface that penalize the robotic team. The rules concerning the playing surface and the mines are as follows:

“The competition will take place on a 8′ x 8′ flat black playing surface. All other parts of the playing surface will also be flat black (Rust-Oleum Flat Black, #7776). The playing surface will be surrounded by 6" high walls which will be painted flat black. A rectangular grid will be painted on the surface with parallel lines being 8" apart and each line being 1/2" wide using gloss white (Rust-Oleum Gloss White, #7792). The first white line in both the

horizontal and vertical directions will be centered at a distance of 8" from the wall. A 8" square will be painted about each fixed mine. The square will be painted such that its lines are perpendicular to the grid lines which they cross. The lines will be pained dark red and will not cover any existing white lines. The starting square will be designated as one of the red mine squares located closest to the wall. Only one square will be chosen for the competition.

Each mine will occupy an intersection of two white lines. The mines will use an optical sensor to detect the presence of the vehicle. A 1/2" diameter circle of Plexiglass will be centered above the sensors. The Plexiglass will be mounted flush with the playing surface. The fixed mines are located as follows:

4 mines each located at the intersection of the third line away from each wall.

4 mines each located at intersection of the center lines and the first line away from each wall." [1]

A possible solution to the problem posed by the mines would be the design of an image detection system. The system would include some type of camera and image detection software. Images from the playing surface would be captured by the camera and then processed by the software. The software would determine the presence, if any, of a red mine within the image. This information would then allow the robot to adjust, if necessary, its direction on the playing surface so as to avoid the penalizing mines. It is the image detection software which is the interest of this paper. We wish to investigate the possibility of using some standard image detection algorithms, the Roberts, Sobel, Robinson, Canny, and Hough, for implementation with the design robot. While this is not necessarily the best solution to the problem of mine avoidance, conclusions from this paper will likely indicate whether or not an image detection system is feasible for this project.

II. THEORY

This section of the paper will introduce some background into image detection and give detailed explanations of the algorithms themselves. Also, information concerning the frequency response of the algorithm's masks will be given.

A. Image Detection

The first step in analyzing images is the separating, or segmenting, of the objects within the image. Segmentation algorithms allow for distinctions to be made between two or more objects.[3] Segmentation is based upon two concepts: similarity and discontinuity. If an image is converted to gray scale, i.e., colors are separated into distinct shades of gray, a boundary of an object can be noted by a sudden change in the gray level. This discontinuity in the image could be either an isolated point or a line or edge of an object. It is the purpose of a segmentation algorithm to accurately locate these discontinuities. Segmentation algorithms can be divided into three separate types based on the discontinuities that they locate: point detection, line detection, and edge detection.[2]

Point detection is the simplest of the detection techniques but provides the least information. A point will have a drastic change in gray value from its neighbors. Therefore, if a pixel's value differs from those of its neighbors by more than some threshold amount, it can be considered a point.[2]

Line detection is a more complicated process. It involves finding pixels that are likely to be

parts of lines and testing them to see if they are part of a common line. One such process, the Hough Transform, is described in the Algorithm section below.[2]

Edge detection is the attempt to find discernible changes in contrast in two dimensions. This approach is most appropriate for this particular project, and thus most of the algorithms in this project fall into this category.[2.

Most segmentation algorithms use a mask on the image's pixels for the detection of a discontinuity. Each pixel and its neighboring pixels have its gray level value multiplied by a mask value. The sum of these values represent that point's mask response. An example could be the following:

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \bullet \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix} =$$

$$R = m_1 p_1 + m_2 p_2 + \ldots + m_9 p_9 = \sum_{i=1}^{9} m_i p_i$$

Figure 1 Example of a mask response.

Here, m_i is the mask value for a pixel, and p_i is the gray level value for a pixel. R is the mask response for the pixel which the mask is centered around (p_5). By sweeping this mask across the image row by row, a new array is created. It is the same size as the original image but contains the values of the mask response instead of the pixel value. These mask response values can then be compared to a minimum threshold value to determine which pixels are more likely to be part of an edge. This threshold can be adjusted to vary the selectivity for edge pixels, allowing a user to "tune" the algorithm for optimal performance for a given picture. [2]

Consider the example mask responses shown in Figure 2 and Figure 3 as examples. Let the threshold value be 20. Figure 2 shows the mask response for a point that is continuous. The image matrix represents the pixel values of a 3x3 portion of an image. Since the outside values of the mask response (summed to -16) cancel the center value (+16), no point of discontinuity is observed.

Point Detection Mask	Image	Mask Response
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} -2 & -2 & -2 \\ -2 & 16 & -2 \\ -2 & -2 & -2 \end{bmatrix}$

Figure 2 Mask response for a continuous set of pixels.

However, Figure 3 demonstrates a mask response for a point of discontinuity. In this case, the center value of the mask response (+64) and the outside edge values (-16) sum together for a value of +48. Since this value is greater than 20, the center pixel in the image

matrix is a point of discontinuity.

Point Detection Mask	Image	Mask Response
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 8 & 2 \\ 2 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} -2 & -2 & -2 \\ -2 & 64 & -2 \\ -2 & -2 & -2 \end{bmatrix}$

Figure 3 Mask response for a discontinuous set of pixels.

B. Algorithms

Based on the uniformity of the grid design on the board, it was decided that several conventional segmentation algorithms would work well to find the borders of the red mine field. Many such algorithms have been developed, mainly differing in the masks used to determine the chance of a pixel being an edge pixel. The four edge detection algorithms used in this project are some of the more widely known image detection algorithms and were chosen based on their different strengths and weaknesses.

The **Roberts Operator** is one of the oldest and simplest edge detection algorithms to implement. It uses two 2x2 matrices to find the changes in the x and y directions:[9]

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1}$$

$$G_x \qquad G_y$$

To determine whether the pixel being evaluated is an edge pixel or not, the magnitude of the gradient is calculated as follows:

$$|G| = \sqrt{G_x^{\,2} + G_y^{\,2}} \tag{2}$$

If the calculated magnitude is greater than a minimum threshold value (set based on the nature and quality of the image being processed), the pixel is considered to be part of an edge. The direction of the edge's gradient, perpendicular to the direction of the edge itself, is found with the following formula:

$$\alpha = \operatorname{atan}\left(\frac{G_y}{G_x}\right) \tag{3}$$

The small size of the masks for the Roberts Operator make it very easy to implement and quick to calculate the mask responses. However, these responses are also very sensitive to noise in the image.

The **Sobel Operator** uses two masks to find vertical and horizontal gradients of edges. The masks for the Sobel are as follows[8]:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{4}$$

$$G_y \qquad G_x$$

The formula for finding the magnitude of the response and the angle of the gradient is the same as for those in the Roberts Operator. Because the masks are 3x3 rather than 2x2, the Sobel is much less sensitive to noise than the Roberts, and the results are more accurate. The drawbacks of using the Sobel operator are that effects of an edge are spread out over a 3x3 pixel area, and the computation of $|G|$ is fairly involved. Therefore, in practice, $|G|$ is often

approximated as the following:

$$|G| = |G_x| + |G_y| \quad (5)$$

The **Robinson Operator** is similar in operation to the Sobel operator but uses a set of eight masks, four of which follow:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \quad (6)$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

The other four are simply negations of these four, and thus the computation is simplified. The magnitude of the gradient is the maximum value gained from applying all eight masks to the pixel neighborhood, and the angle of the gradient can be approximated as the angle of the line of zeroes in the mask yielding the maximum response. This algorithm increases the accuracy of $|G|$ and a but requires more computation than both the Roberts and Sobel algorithms[6].

The **Canny Edge Detector** has its basis in a slightly more visual approach. If one considers a one-dimensional step edge change in contrast and then convolves that edge with a Gaussian smoothing function, the result will be a continuous change from the initial to final value, with the slope reaching a maximum at the point of the original step. If this continuous slope is then differentiated with respect to x, this slope maximum will become the maximum of the new function again at the point of the original step.

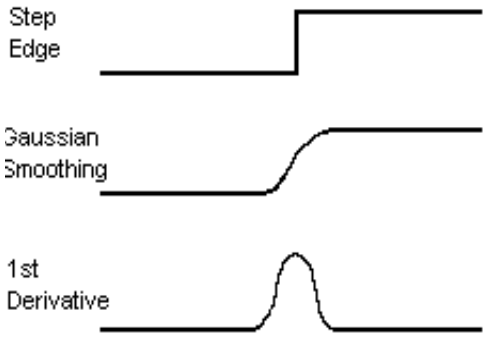

Figure 4 Steps in 1-D Canny edge detector.

Because the derivative of the convolution of the Gaussian and the image is the same as the convolution of the derivative of the Gaussian and the image,

$$D[Gauss(x, y) \otimes Img(x, y)] = D[Gauss(x, y)] \otimes Img(x, y) \quad (7)$$

a mask can be created that represents the first derivative of the Gaussian. The maximums of the convolution of the mask and the image will indicate edges in the image. This process can be accomplished through the use of a two-dimensional Gaussian function or the combination of a one-dimensional Gaussian function in both the x- and the y-directions. The values of the differentiated Gaussian mask depend on the choice of sigma in the Gaussian

equation:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x}{2\sigma}}$$
$$G'(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x}{2\sigma}} \quad (8)$$

The computational intensity of the Canny Edge Detector is relatively high, and the results are usually post-processed for clarity. However, the algorithm is very effective in processing noisy data or images with fuzzy edges.[11][12]

The **Hough Transform** is ideal for line detection if little is known of the location of an image but its shape can be represented by a mathematical formula. By considering the equation of a line, $y = mx + c$, all possible lines that could pass through a single point in a image, (x',y'), can be represented in the form of $y' = mx' + c$. If (x',y') are considered fixed, then m and c are now the variables in what is called the parameter space. If (x',y') lie on line *AB*, then every point of line *AB* will have a common point of intersection in the parameter space, (m',c'). With this relationship between the image space and the parameter space established, the Hough Transform can be applied. By considering the maximum and minimum values of both *m* and *c*, an array of *H(m,c)* can be created with all elements initialized to zero. For every available point in the image space, the gradient is computed. If the gradient exceeds a defined threshold, then all elements of *H(m,c)* that pertain to that line are incremented. The local maxima of the array now represent the points of a line in the image.[3]

C. Frequency Response

The following image shows the frequency response of the Dy Sobel mask.

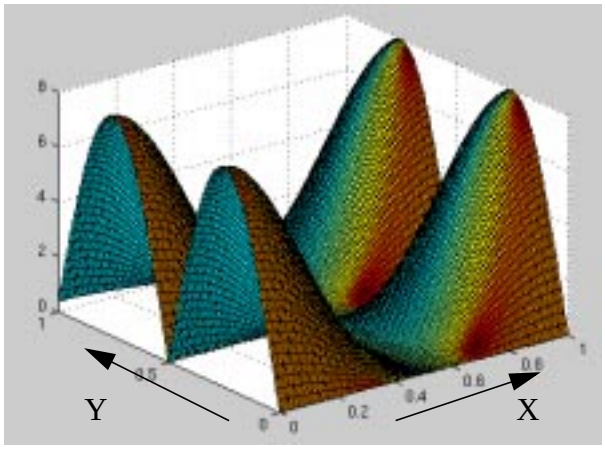

Figure 5 Frequency Response for Sobel Dy

By following the curve along the y-frequency direction, one can see that the response rises sharply with frequency from 0 to about Fs/4, indicating a differentiator effect, then declines again through Fs/2, showing an attenuation of relatively high frequencies. This shows that the filter should respond strongly to line changes in the y direction, but filter out some of the higher frequency noise in the image. Likewise, if one follows the curve along the x-frequency direction, it is apparent that the response in this direction is zero, given no change in y.

The frequency response of the Dx Sobel mask

shows a similar operation. The response along

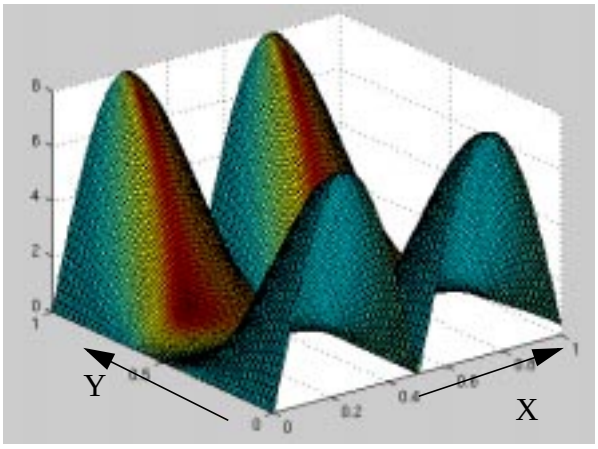

Figure 6 Frequency Response for Sobel Dx

the y-frequency direction's edge shows a zero response. The response along the x-frequency direction's edge shows the differentiation of the low- to mid-frequency signal components of an edge, and the attenuation of high-frequency signals.

The frequency responses of the diagonal masks in the Robinson algorithm are shown below. The effects of the filtering are the same

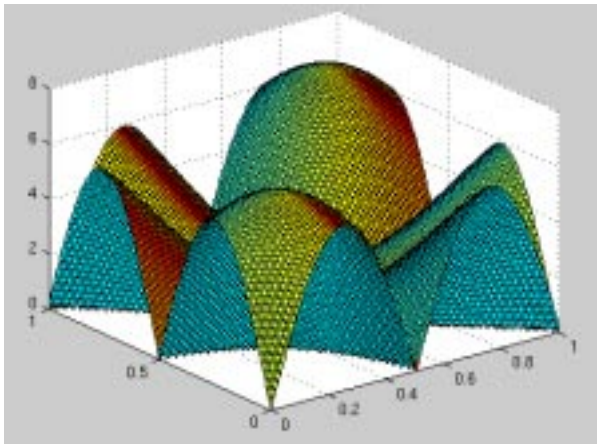

Figure 7 Frequency Response for Robinson at 45 degrees

as above, with the angles of the responses differing accordingly. The first image shows the mask with zeros at a 45 degree angle. Following a path from the origin of graph to the opposite corner, corresponding to equal increases in x and y, yields the same differentiation and attenuation pattern seen before. The same pattern can be seen in the second diagonal frequency response graph by following a path from (1,0) to (0,1), which corresponds to a line in the image at 135 degrees.

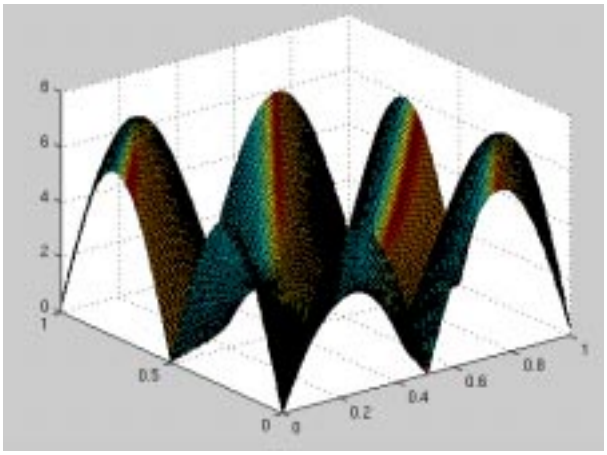

Figure 8 Frequency Response for Robinson at 135 degrees

The next two images are the frequency responses of the Roberts 2x2 masks. It quickly becomes apparent that these masks actually respond best to diagonal edges, rather than vertical and horizontal as indicated in the research. It can also be seen from examining the frequency range of the lobes that these masks will be more susceptible to high-

frequency noise, as expected.

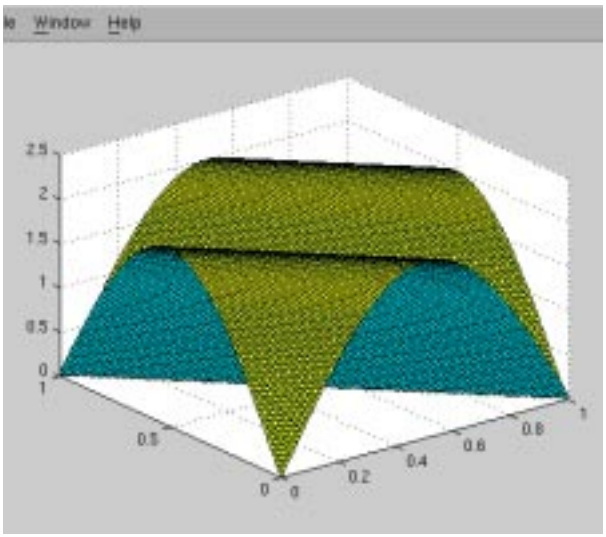

Figure 8 Frequency Response for Roberts Dy

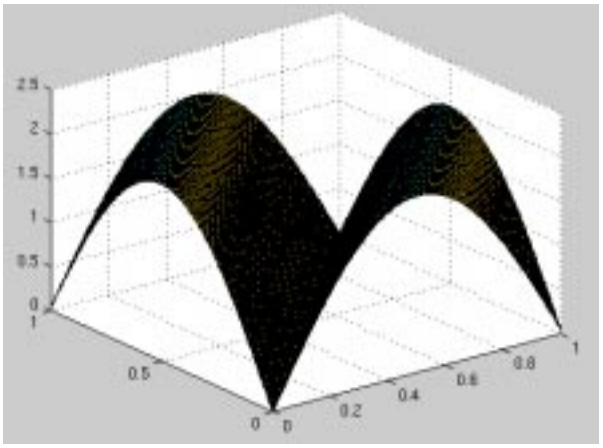

Figure 9 Frequency Response for Roberts Dx

Next, the frequency of the Canny Dy and Dx masks are shown. Again, the differentiation characteristics can be seen in the shape of the lobes. However, with the Canny masks, this action occurs on a much lower frequency range, and the high frequency attenuation is very pronounced, with nearly complete attenuation from about Fs/5 to Fs/2. This provides much less sensitivity to noise than any of the other approaches, while the steep differentiation still detects the lower frequency components of the edges.

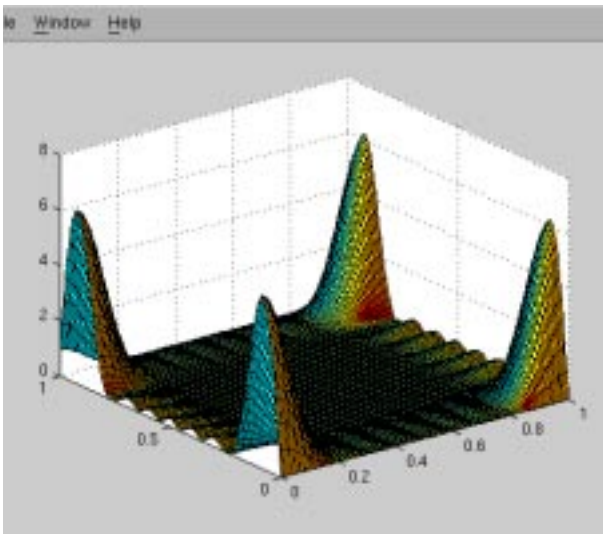

Figure 10 Frequency Response for Canny Dy

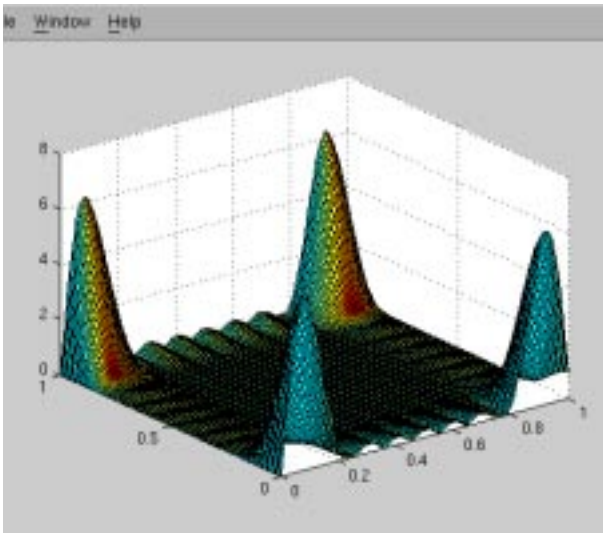

Figure 11 Frequency Response for Canny Dx

The final four images demonstrate the effects of varying x and sigma in the calculation of the

Gaussian functions for the Canny. In the first image, the filter width is changed from 10 to 5. This introduces more high-frequency components, basically because the image is not being smoothed out as much in the convolution. The second image shows the effect of a filter width of 15. The higher frequencies are almost completely stopped, leaving only the very lowest frequencies in the edge to be differentiated. The third image shows the effect of leaving the filter width 10 and changing sigma from 3 to 1. It is obvious that setting the value of sigma too low compared to the filter width produces undesirable effects. For the last image, the value of sigma was set to 5. The ripple in the higher frequencies is diminished considerably,

and the differentiation peaks are very sharp.

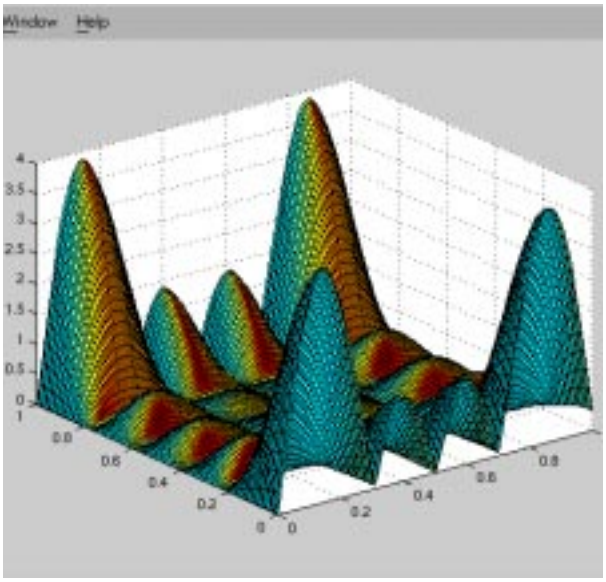

Figure 12 Frequency Response for Canny Dx with width 5

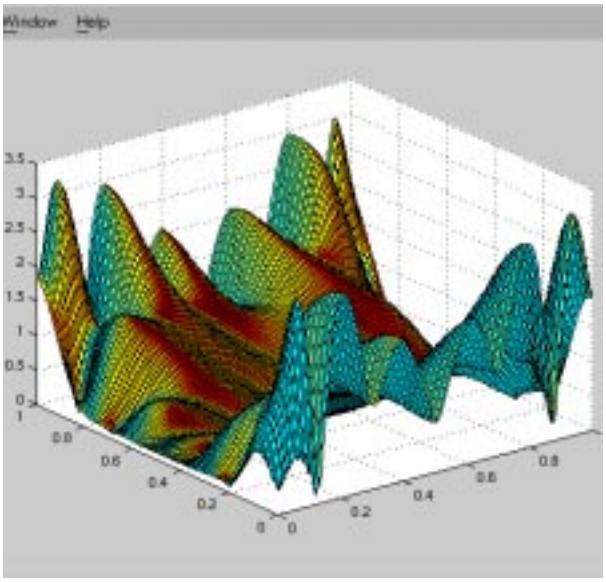

Figure 14 Frequency Response for Canny Dx with sigma 1

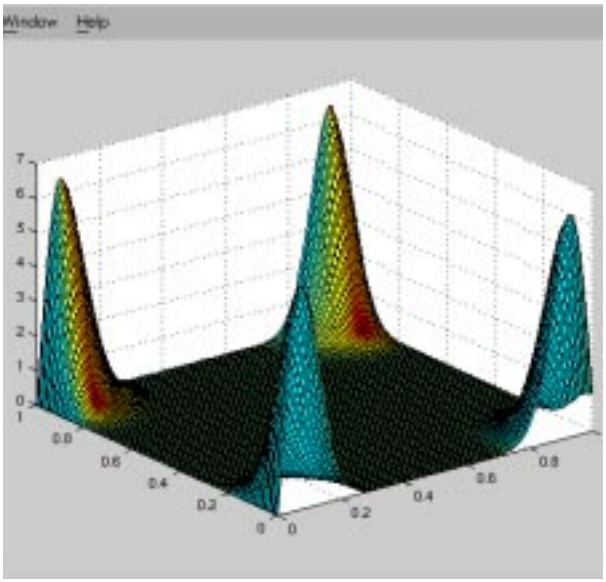

Figure 13 Frequency Response for Canny Dx with width 15

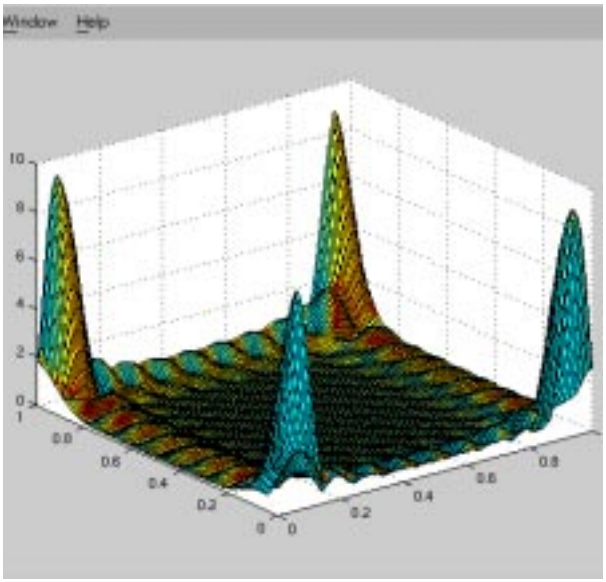

Figure 15 Frequency Response for Canny Dx with sigma 5

The effects of these parameter variations were

seen in practice when testing various values for x and sigma in the Canny implementation. For smaller sigma values, there was a slight response to noise in the image, and increasing the value of sigma reduced this response. Also, the noise was reduced if the width of the filter was increased.

III. IMPLEMENTATION

The implementation of our project was divided into three sections: completion of a comprehensive image database, development of algorithms and edge detection software, and testing of algorithms on images from database.

A. Image Database

Before the image database could be constructed, the test platform had to be designed, and some hardware and image capture software had to be constructed so that images could be taken from the platform.

First, a mobile test platform was constructed according the specifications of the hardware competition rules. After finishing the platform, attention was turned to the task of capturing images. Images could be captured in either color or gray scale. Gray scale was chosen because of the relatively low complexity of the images. Each image would ideally have only three colors: black, white, and red. Even when noise and image capturing inaccuracy are included, the colors should be different enough to be easily separated. With so few colors in the images, color images would introduce unnecessary information. Gray scale images would be easier to process and would give results comparible to color images.

For the actual capturing of the images, a CCD gray scale camera was connected to an image capture card in a PC. The software provide with the image capture card allowed the storage of the images to Windows Bitmap format. The images were converted from Windows Bitmap to PGM using shareware image conversion software. The reason for converting the format of the images will be explained later in the paper.

Once we were able to take images from the test platform, the database was constructed. To derive an accurate conclusion from our results, a comprehensive database was to be constructed. A total of 80 images was included in the database. Sixty of those images were used as training data for the algorithm, and the remaining 20 were reserved for performance evaluation of the algorithms.

The images were divided equally into two main groups: images taken under controlled lighting conditions and images taken in ambient lighting conditions (Table 1). Forty unique images of a specified orientation were taken under controlled lighting, and then another 40 images were captured under ambient lighting. For the controlled lighting conditions, the CCD camera was shrouded from the fluorescent lights. This provided an image with nearly equal brightness level at all points. For the images taken under ambient light, no special considerations were made to alter the brightness level of the image.

The contents of the images were also varied for a more complete evaluation. 3 sets of images were taken with an individual “mine” covering 1/4 or more, 1/16, and 1/64 of the camera’s viewing area, respectively. Each set of images was further divided into a subset based on the angle of the mine with respect to the camera, θ. The subset had 4 images taken at $\theta = 0^{\circ}$, $\theta = 45^{\circ}$, and $0^{\circ} < \theta < 45^{\circ}$, respectively. The final four images taken contained no portion of a red square. This made a total of 40 different image orientations.

With images taken under both ambient and controlled lighting, a total of 80 real images were included in the image database. In separating the testing images, one image of each type was used for a total of 20 test images.

% of Mine Within Image	Angle of Square	Number of Images (Ambient/ Non-ambient)
>=1/4	$\theta = 0^\circ$	4/4
	$\theta = 45^\circ$	4/4
	$0^\circ < \theta < 45^\circ$	4/4
1/16	$\theta = 0^\circ$	4/4
	$\theta = 45^\circ$	4/4
	$0^\circ < \theta < 45^\circ$	4/4
1/64	$\theta = 0^\circ$	4/4
	$\theta = 45^\circ$	4/4
	$0^\circ < \theta < 45^\circ$	4/4
no square	n/a	4/4
Total Images		80

Table1 Image Database Breakdown.

B. Software

The implementation of the algorithms and edge detection software consisted of three steps: reading in and writing out of images in PGM binary format, development of the Roberts, Sobel, Robinson, and Canny edge detection algorithms, and development of the Hough Transform algorithm. All code was written in C++ and compiled under the Solaris 2.51 using a g++ compiler.

1. PGM Format

In order to use these algorithms on an image, the format of the pixels and other information pertinent to the image itself had to be known. Otherwise, the algorithms would not be able to recognize the images. Rather than attempt to use the images in Windows Bitmap format, these images were converted to gray scale PGM format. This decision was made for a several reasons. One nice feature of PGM is its relatively simple format. Because it has very little header information, accessing the actual image data in a PGM file is greatly simplified. Another consideration dealt with the demonstration aspect of our project. For the demonstration, the TCL script programming language was selected. A drawback to Tcl is its inability to display Windows Bitmap images. It could, however, read PGM. With the image format set, the images could now be read and processed.

2. Edge Detection Algorithms

Once images were readable, we began implementing the various processing algorithms. The first three, the Roberts, Sobel, and Robinson, were fairly simple. As the image array was scanned pixel by pixel and row by row, the mask for each algorithm was multiplied by each pixel and its neighbors. The iterations began and ended one pixel in from each of the edges to avoid using neighbor pixels that were outside the image array. The summation of these products of course became the mask response for pixels. For the Roberts and Sobel algorithms, the *Dy* and *Dx* mask responses were calculated for each pixel, and

the sum of their absolute values was placed into a new array representing the pixel response magnitude. For the Robinson algorithm, all four mask responses were calculated, and the maximum of these four responses was placed into the response magnitude array. Once the entire image array had been processed, the magnitude array was thresholded to filter out any small responses and written out to a new PGM image file.

The Canny algorithm was implemented differently than the previous three algorithms. The first step was the calculation of the values for both the Gaussian smoothing and the derivatives of the Gaussian. It was decided to allow the values of *x* and *sigma* be set dynamically by the user, and so the values being multiplied by the pixel values could vary. A two-sweep 1-dimensional approach rather than a 2-dimensional single convolution of a mask was used. So once the Gaussian values were computed, the image array was smoothed in the *x*- and *y*-directions. These smoothed values were placed separately into two intermediate arrays. The arrays were then convolved with their orthogonal Gaussian derivatives, i.e., the *x*-smoothed array was convolved with the *y*-direction derivative values. This produced the *Dy* and *Dx* magnitude arrays, where the subscript *y* or *x* is the same as the direction of the Gaussian derivative used. These magnitude arrays were then processed to find their maxima, indicating which *x,y* locations contained likely edge pixels. The value of these maxima were then written out to a new PGM image file.

3. Hough Algorithm

The output of the edge detection algorithms indicated pixels that were likely edges. However, the pixels still had to be grouped together to form an edge. For this step, the Hough Transform was used.

The Hough algorithm was implemented by reading in the output image from one of the edge detection algorithms. To insure that only pixels that had a high likelihood of being edge pixels were considered, a minimum threshold was set to filter out very low response pixels from the input image. An accumulator array was created with a height of 180 and width equal to the maximum value calculated for *r* in the line equation:

$$r = x\sin\theta + y\cos\theta \qquad (8)$$

The height of 180 corresponds to the number of angle values that would be processed. The thresholded input array was then swept through, and any time a non-zero pixel value was encountered, the *r*-value was calculated for all integer values of θ between 0 and 180. These *r*-values were made into integers, and the accumulator pixels with indexing $[\theta][r]$ were incremented by 1. Any input pixels that shared the same line would increment the same r,θ pair, so the accumulator array locations that were maximums would likely represent lines in the original input image. Once the input image array had been processed, the accumulator array was searched for local maxima, and the r,θ values at these points were used to calculate *x,y* endpoints for these likely lines.

One problem with the Hough line detector is that is can not distinguish between line segments that terminate within the image and line segments that traverse the entire image. Therefore, the endpoints calculated from the r,θ pairs ran from one side of an image to the other, regardless of whether their representative line segments actually did or not. We tried to deal with this overshooting problem by splitting the original image into smaller sections and running the Hough transform on these sections. For each section, the line segments contained and the corresponding local endpoints were found. In

any endpoint in one section was sufficiently close in angle and x,y proximity to an endpoint from another section, the two segments were combined into one. Another problem with the Hough is that it sometimes doesn't detect shorter line segments in the image. This is because there are few pixels along that line to contribute to the accumulator array. It was hoped that splitting the image would solve this problem because the short segment would probably be a significant part of a smaller section. The approach met with some limited success in avoiding the overshoot problem by not detecting edges past where they actually terminated. However, poor local representation of the line significantly increased the variance in the angles detected. Also, the sensitivity to noise, smoothed over by the larger magnitude of the global Hough transformation, became a factor by increasing false edge detection. Finally, parts of the line segments were occasionally omitted in the output from this algorithm. On examining these effects, this approach was abandoned.

C. Testing Procedure

In evaluating the algorithms, the success of each algorithm was determined by its robustness in detecting edges. The evaluation of the algorithms was done statistically two ways. One method of evaluation was based on a point method. A correctly identified edge was given a +1 point, and a incorrectly identified or undetected edge was given a -1 point. A successful edge detection was the placing an edge within 5 pixels of its actual location. The point totals were summed together for a final point value. The second method of evaluation was an edge detection percent error. The number of correctly identified edges was divided by the total number of edges providing the percent error in detection.

Each edge detection algorithm required some thresholding arguments. All of the algorithms used a upper and lower input threshold. The Roberts, Sobel, and Robinson also used a output threshold. The Canny did not need an output threshold but did require a filter width and sigma value. The Hough used an input and output threshold value. During the training of the algorithms, different values for each of the arguments were used to find optimum values. Different values for the same argument were often needed on different images.

IV. RESULTS

The results of our project our listed in Table 3 and Table 4. Table 3 indicates the point totals for the algorithms. Table 4 gives the percentage error of the edge detection.

The filename represents the contents of the image. The format for the filename is as follows:

LXXYYn4

where:

L is the lighting conditions - a= ambient, c= controlled

XX is the percentage of the mine in the image - 04=1/4, 16=1/16, 64=1/64, 0=no square

YY is the angle of the mine - a0=0 deg., a4=45 deg., ao=0<theta<45 deg., 00=no square

Filename	Max Possible Points	Points Awarded for Algorithms			
		Roberts	Sobel	Robinson	Canny
a0000n4	1	1	1	1	1
a04a0n4	3	1	3	1	-1
a04a4n4	2	-3	1	1	2
a04aon4	3	-1	-2	-2	-1
a16a0n4	2	-2	-1	-2	-2
a16a4n4	2	-2	-2	-3	2
a16aon4	1	0	-1	-3	1
a64a0n4	2	-2	1	-2	-1
a64a4n4	2	-3	-2	-2	0
a64aon4	2	-1	-1	-3	2
c0000n4	1	-1	1	-1	1
c04a0n4	3	1	3	3	3
c04a4n4	2	1	2	2	2
c04aon4	5	-4	-4	0	3
c16a0n4	2	-4	0	0	0
c16a4n4	2	-4	0	-5	2
c16aon4	2	-2	0	-1	0
c64a0n4	2	-3	-6	-5	0
c64a4n4	1	-3	-2	-4	-4
c64aon4	2	-5	-3	-4	0
Total	**42**	**-37**	**-12**	**-29**	**10**

Table 2 Analytical Results

Lighting Conditions	Roberts % Error	Sobel % Error	Robinson % Error	Canny % Error
Ambient	74.9%	65.9%	69.8%	34.8%
Controlled	84.6%	53.0%	68.0%	30.0%
Average % Error	**79.75%**	**59.45%**	**68.9%**	**32.4%**

Table 3: Percent Errors

	Add./Sub.	Mult./Div.	Comparisons	Higher Order	Trig Function	Avg Time
Roberts	7*W*H	W*H	0	2*W*H	W*H	1.83s
Sobel	51*W*H	5*W*H	W*H	4*W*H	W*H	2.89s
Robinson	53*W*H	8*W*H	W*H	4*W*H	0	3.21s
Canny	181*W*H + 20*filterW	20*W*H + 32*filterW	2*W*H	2*W*H + 4*filterW	W*H	10.19s

Table 4: Analytical Results

V. CONCLUSIONS

In conclusion, the four edge detection algorithms, namely the Roberts, Sobel, Robinson, and Canny, have been evaluated in their effectiveness to detect red mines on a black and white background using an inexpensive gray scale CCD camera. As summarized in Table 3, the algorithm with the least percent error is the Canny edge detection algorithm. Table 4, however, demonstrates that the Canny also has the greatest execution time, on the order of 10 seconds for a 320x240 8-bit gray scale image. Even though it appears to have the most desirable results, the Canny is not particularly suited for realtime applications due to the enormous latency associated with it. This tradeoff is true for all the algorithms tested, in that the performance is inversely proportional to the execution time. As the execution time nears a realistic value for realtime operation, the performance of the algorithm drops to unacceptable levels.

Furthermore, it was observed that certain algorithms had inherent strengths and weaknesses. For example, the Roberts algorithm was fast, but generally found edges only on the 0 and 90 degree axis. The Sobel algorithm was able to find edges on the 0 and 90 degree axis, along with the 45 and 135 degree axis, but the execution time was slower than the Roberts. For an additional increase in execution time, the Robinson was capable of detecting an increased number of random-angle edges. The Canny appeared to be the best algorithm for the edge detection requirements of this application, but suffered from the worst execution time.

Therefore, it has been determined that, due to the performance versus accuracy tradeoffs involved, none of the four standardized edge detection algorithms tested are very well suited for this particular realtime application. This, however, does not rule out the possibility of designing a hybrid algorithm based on the principles discovered in this comparison which might be better suited for realtime operation.

VI. REFERENCES

[1] IEEE SouthEastCon 1998 Hardware Competition. Http://www-ece.engr.ucf.edu/secon98, 1997.

[2] Gonzales, Rafael C., Richard E. Woods. *Image Segmentation*. Addison-Wesley, Reading Massachusetts, 1992.

[3] Ballard, Dana H., Christopher M. Brown. *Computer Vision*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[4] Dougherty, Edward R., Charles R. Giardina. *Image Processing - Continuous to Discrete, Volume 1*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[5] Boyle, R. D., R. C. Thomas. *Computer Vision, A First Course*. Blackwell Scientific Publications, Oxford, 1988.

[6] Haralick, Robert M., Linda G. Shapiro. *Computer and Robot Vision*. Addison Wesley, Reading, Massachusetts, 1991.

[7] Fairhurst, Michael C. *Computer Vision for Robotic Systems, An Introduction*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1988.

[8] Russ, John C. *The Image Processing Handbook*. CRC Press, Boca Raton, Florida, 1995.

[9] Young, Tzay Y., King-Sun Fu. *Handbook of Pattern Recognition and Image Processing*. Academic Press, San Diego, California, 1986.

[10] Pitas, Ioannis. *Ditigal Image Processing Algorithms*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[11] *Edges: The Canny Edge Detector*, Http://www.icbl.hw.ac.uk/marble/vision/low/edges/canny.htm.

[12] *Basic Image Processing Demos*, Http://robotics.eecs.berkelee.edu/~mayi/imgproc/

[13] *Canny Edge Detection*, Http://www.icbl.hw.ac.uk/marble/vision/low/edges/canny.htm

[14] Http://trantor.cse.psu.edu/~jphelps/cse485/project2.html

[15] Http://www.cse.psu.edu/~caro/cse485/p2/project2.html

[16] Http://robotics.eecs.berkeley.edu/~mayi/imgproc/