DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

proposal for

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

submitted to fulfill the semester project requirement for

EE 4773/6773: Digital Signal Processing

October 3, 1997

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

John Burnham, Jonathan Hardy, Kyle Meadors

Image Processing Group
The Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Box 6176
Mississippi State, Mississippi 39762
Tel: 601-325-8335
Fax: 601-325-3149
email: {jab2, jeh3, kam1}@ece.msstate.edu

I. ABSTRACT

This project will compare the Roberts, Sobel, Robinson, Canny, and the Hough image detection algorithms based on their ability to detect red squares on a black and white background. An image database consisting of actual grey-scale images taken from a test platform containing red squares on a black and white background will be constructed for the evaluation. The success of each algorithm will be based on its accuracy in detecting the red squares within the images from the database. The results of the algorithms will be then compared statistically to determine which one is the best suited for this application.

II. INTRODUCTION

Background

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 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 Theory 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 the 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:

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \bullet \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{bmatrix} = R = m_1 p_1 + m_2 p_2 + \ldots + m_9 p_9 = \sum_{i=1}^{9} m_i p_i$$

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]

Theory

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 each pixel being an edge pixel.

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

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

$$G_x \qquad\qquad G_y$$

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

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

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 said to be part of an edge. The direction of the gradient of this edge, perpendicular to direction of the edge itself, is found with the following formula:

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

The small size of the masks for the Roberts Operator make it very easy 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]:

$$\underset{G_x}{\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}} \qquad \underset{G_y}{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}}$$

The formula for finding the magnitude of the response and the angle of the gradient are 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:

$$|G| = |G_x| + |G_y|$$

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

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

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. 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,

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

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

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-x^2}{2\sigma^2}} \qquad G'(x) = \frac{-x}{\sqrt{2\pi}\sigma^3}e^{\frac{-x^2}{2\sigma^2}}$$

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 elements all *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]

III. PROJECT SUMMARY

The goal of this project is to compare the effectiveness of the Roberts, Sobel, Robinson, Canny, and Hough image detection algorithms on images taken from the IEEE Hardware Competition test platform.[1] The test platform from the competition rules will be constructed. The test platform is 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]

Since this project in its entirety is not necessarily a solution to the IEEE competition, the other competition rules will not be of concern. Only the test platform will be taken from the rules. A graphic representation of the test platform is shown in Figure 1.

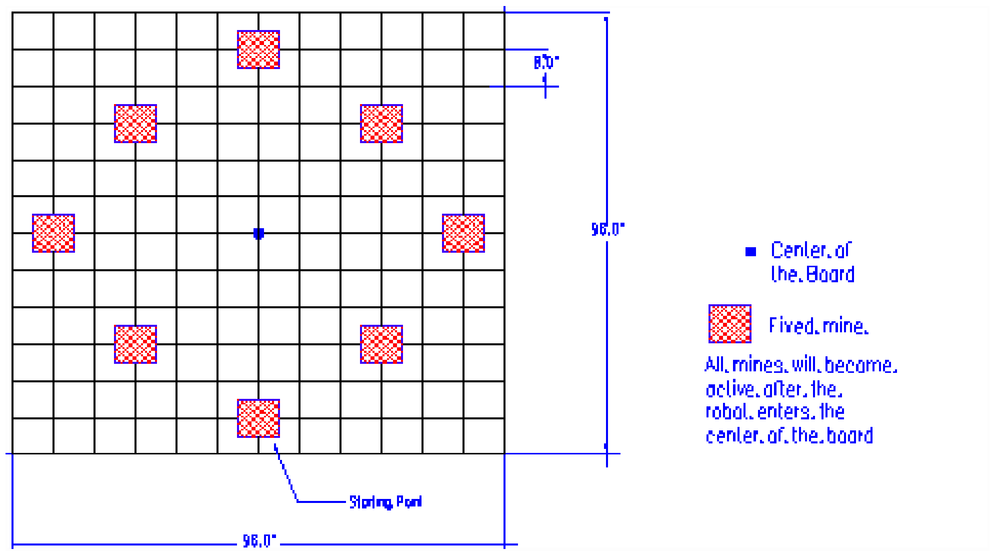

8.0"
96.0"
Center of the Board
Fixed mine
All mines will become active after the robot enters the center of the board
Starting Point
96.0"

Figure 1

The project itself can be divided into four major sections: image capturing and database construction, algorithm implementation, data evaluation, and project demonstration.

Image Capturing and Database Construction

As the algorithms are being implemented, a database of images will be constructed. The first images created will be synthetic. Using the test platform guidelines as specified in the IEEE Hardware Competition, actual sized images will be created in a generic image design program.

Synthetic images provide ideal color values for each pixel and are perfect for early training data. The majority of the database images will be images taken from the actual test platform. Of these real images, 75% will be used as training data with the other 25% reserved for test data. The images will be captured in BMP format by a CCD gray-scale camera and a video capture card, fed to a computer and then saved and categorized. Also, the number and location of each square edge within an image will be recorded.

The real data images can be divided equally into two main groups: images taken under controlled lighting conditions and images taken in ambient lighting conditions. 40 unique images of a specified orientation will be taken under controlled lighting and then ambient lighting. For the controlled lighting conditions, the CCD camera will be surrounded by an enclosure with light sources. The light sources will provide an image with a nearly equal brightness level at all points. The ambient light will be fluorescent overhead lights. For the images taken under ambient light, no special considerations will be made to alter the brightness level of the image.

The contents of the images will also be varied for a more comprehensive evaluation. 3 sets of images will be 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 will be further divided into a subset based on the angle of the mine with respect to the camera, θ. The subset will have 4 images taken at $\theta = 0^\circ$, $\theta = 45^\circ$, and $0^\circ < \theta < 45^\circ$, respectively. The final four images will be taken containing no portion of a red square. This makes 40 different image orientations. With each image taken under both ambient and controlled lighting, a total of 80 real images will be in the image database.

A layout of the database is in Table 1.

Table 1: Image Database Breakdown

Lighting Conditions	Percentage of Mine	Angle of Square	Number of Images
controlled	>=1/4	$\theta = 0^\circ$	4
		$\theta = 45^\circ$	4
		$0^\circ < \theta < 45^\circ$	4
	1/16	$\theta = 0^\circ$	4
		$\theta = 45^\circ$	4
		$0^\circ < \theta < 45^\circ$	4
	1/64	$\theta = 0^\circ$	4
		$\theta = 45^\circ$	4
		$0^\circ < \theta < 45^\circ$	4
	no square	n/a	4

Table 1: Image Database Breakdown

ambient	>=1/4	$\theta = 0^\circ$	4
		$\theta = 45^\circ$	4
		$0^\circ < \theta < 45^\circ$	4
	1/16	$\theta = 0^\circ$	4
		$\theta = 45^\circ$	4
		$0^\circ < \theta < 45^\circ$	4
	1/64	$\theta = 0^\circ$	4
		$\theta = 45^\circ$	4
		$0^\circ < \theta < 45^\circ$	4
	no square	n/a	4
Total Images			80

Algorithm Implementation

The algorithms will be written in C++ and compiled under Solaris 2.51 using a gnu compiler. In the early development of the algorithms, synthetic data will be used for training because of its ideal pixel values. As algorithm implementation progresses, actual images will be used in place of the synthetic images. Approximately 75% of the images from the data base will be used for algorithm qualification.

Algorithm evaluation and project demonstration are detailed in the next two sections of the proposal.

IV. EVALUATION

The evaluation of the project will involve comparing the results of the different algorithms. Since 25% of the database will be reserved for test data, a total of 20 images will be evaluated. The success of each algorithm will be statistically based on its robustness in detecting red square edges in the 20 images.

The number, length, and location of each square edge for every database image will be recorded. When an algorithm processes an image, the output of the algorithm will be another image containing only the edges found within the image. Each output image will be post-processed using a modified edge-following algorithm to produce a file containing a list of the endpoints of all the line segments contained in the image. Several such line-segment construction algorithms will be tested to find the most appropriate one for the project. A program will be written to determine the number, length, and location of every edge in the output image. These results will be compared to the actual values. A correctly identified algorithm edge will closely match the length and location of the actual edge. For every correctly identified edge, a +1 point will added to the algorithm's point total. Incorrectly identified or undetected edges will result in a -1 point for the algorithm. After all 20 test images are evaluated, the algorithm point total will be normalized by the maximum possible points, i.e., the total number of square edges in the test images. This evaluation process will be repeated for each algorithm. The final ranking of the algorithms will be based on the normalized point total.

V. PROJECT DEMONSTRATION

The demonstration will use a graphic user interface (GUI) running from a Sun workstation. An example the GUI layout is shown in Figure 2. While the functionality of the GUI will likely remain the same, the aesthetics of the layout are subject to change.

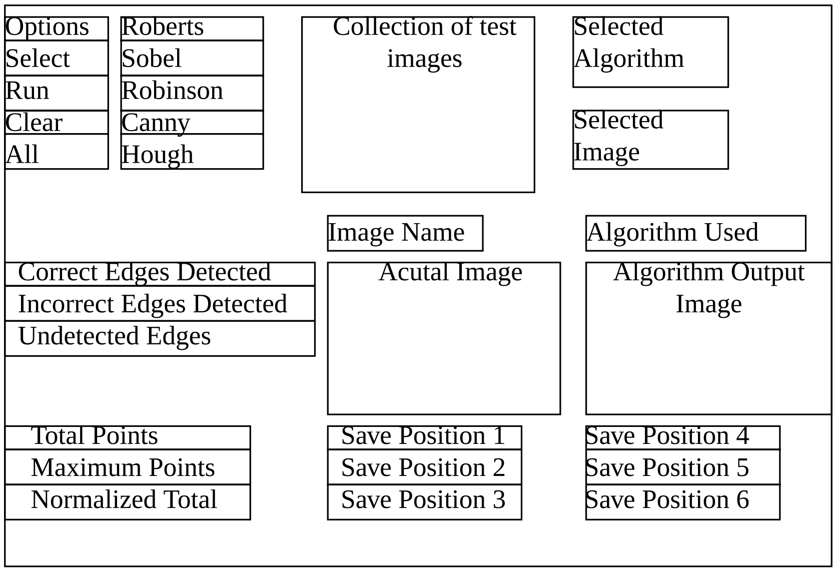


Figure 2 GUI Layout

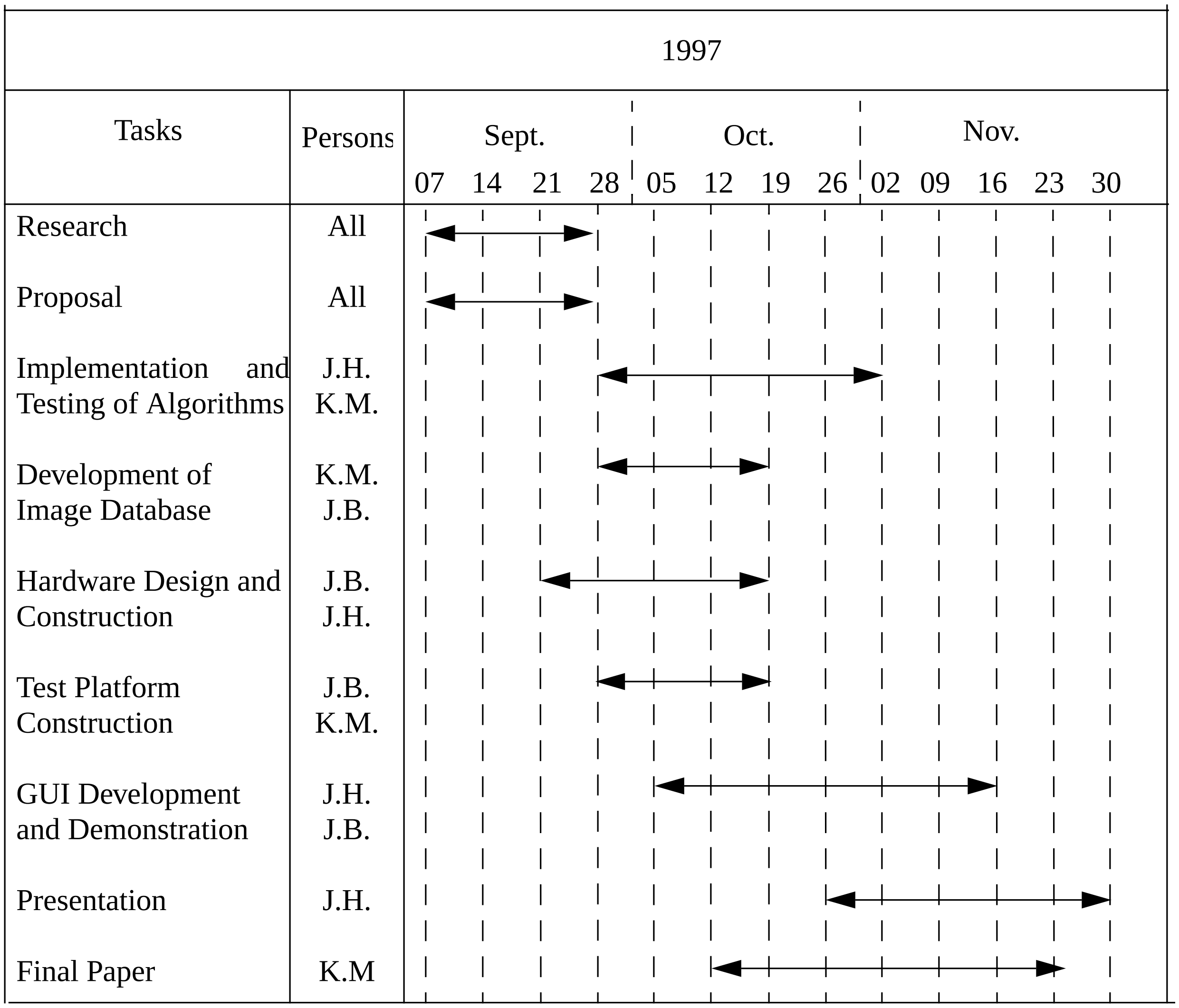
The demonstration will be user interactive. A test image can be selected, and any of the algorithms can be used to evaluate it by using the Select button on the GUI. The selected image and algorithm will appear in the Selected Image and Selected Algorithm sections, respectively. To execute the algorithm on the image, the Run button is used. The actual image and the algorithm output image are displayed along with number of correct edges detected, incorrect edge detected, and undetected edges. The point total, maximum points, and normalized total is also displayed. The point total, maximum points, and normalized total can each be saved by dragging the value into one of the six Save spaces. The value will remain in the Save space until the Save position is cleared.

The entire test image database can also be evaluated by one algorithm. After selecting the algorithm, the All button would then be pushed. The results of the evaluation would be displayed in the point section. The normalized value could then be saved, and another algorithm could be chosen to evaluate all of the test images. The two normalized values could then be compared.

The image database evaluation may be replaced by an interactive image capturing demonstration. A robot with the necessary image capturing paraphernalia will be placed on the test platform. From the GUI, the robot will be maneuvered around the test platform. The image from the camera will be displayed to the GUI. The user then may capture any desired image on the test platform for evaluation. The captured image would be evaluated in the same method as the database images. The interactive image capturing will not be done if difficulty arises in successfully implementing it to the demo.

VI. SCHEDULE

A schedule for the major tasks in this project is shown in Figure 3. The initials of the lead person(s) for each task are listed beside their respective tasks. However, all project members will in some manner be involved with each task.



Tasks	Persons
Research	All
Proposal	All
Implementation and Testing of Algorithms	J.H. K.M.
Development of Image Database	K.M. J.B.
Hardware Design and Construction	J.B. J.H.
Test Platform Construction	J.B. K.M.
GUI Development and Demonstration	J.H. J.B.
Presentation	J.H.
Final Paper	K.M

Figure 3 Task Schedule

VII. 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/.