CALIFORNIA STATE UNIVERSITY, LONG BEACH

College of Engineering
Department of Computer Engineering and Computer Science
Dr. Thinh V. Nguyen
Spring 2007
CECS-553/653: Machine Vision
PROJECT 3 AND EXTRA CREDIT PROJECT

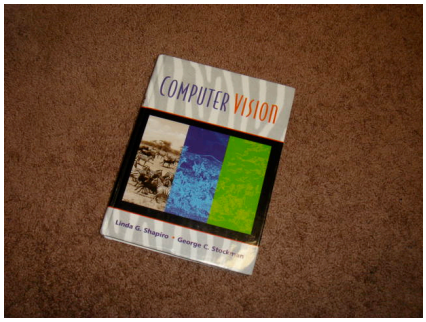
Name: _____Foss, Shannon_____
Last, First

- **Dates:** Date assigned Tuesday April 24, 2007. Date due: 10:15PM on Thursday May 17, 2007 (for regular project) and 11:59PM on Sunday May 27, 2007 (for extra credit projects). Late submissions will receive penalty at 10% per day. This project is worth 50% of the project grade.

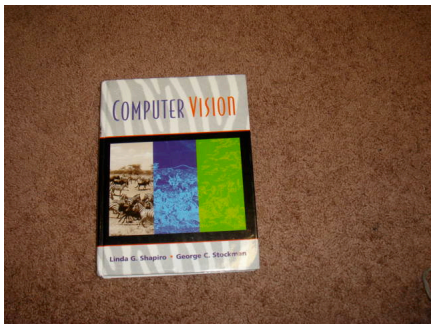
- **Objectives**: The objectives of this project includes: (1) to study image segmentation and 2D matching, (2) to perform Hough transforms to detect lines, and (3) to determine motion vector field and transformation parameters.

- **Project Description:**

  Write a computer program to read in an image named "image.bmp" where *image* is the input to the program and is selected from the image database. Save the image in grey level as "image_grey.bmp" For this project, use only two image files: "Book_1" and "Book_2" as follows:



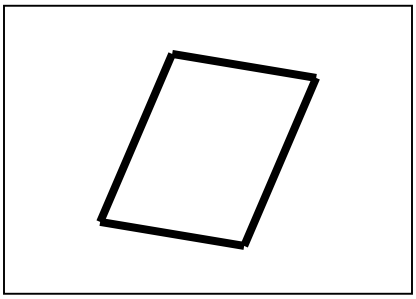
Book 1

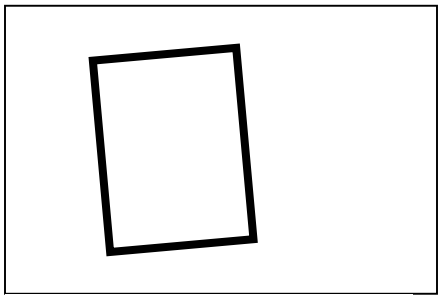


Book 2

Perform the following operations:

1) Edge detection of the book outline: Develop an algorithm to detect the edge of only the outline of the book. You can use either color or grey level image.

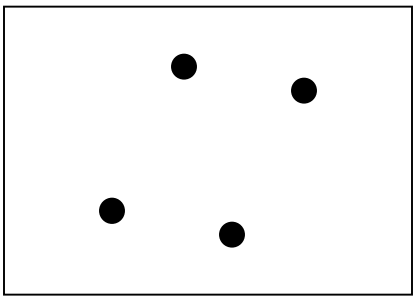

Outline Book 1


Outline Book 2

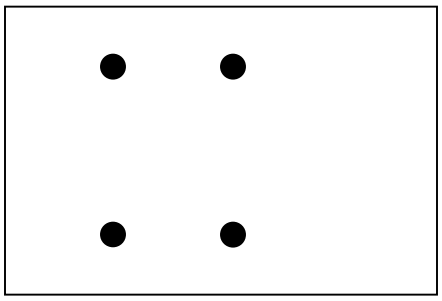
Use any edge detection algorithm as appropriate. The objective is to have clean edges for reliable line detection and motion field determination (for the extra credit project)

2) Line Detection using Hough Transform. Apply the Hough Transform for line detection on the detected edge image for Outline_Book_1 and Outine_Book_2. Show the line detection in the Hough space for both images. Scale the Hough space properly for display purposes. Select proper quantization levels for angle and distance. Discuss results.

3) Group project or Extra Credit 1 for individual project: Corner point detection (for extra credit project, this is equivalent to 50% of project grade). Develop an algorithm to detect the corner points of the book outline. The algorithm may be applied to the original color/ grey level images, or to the edge detected images, or to the Hough space. Discuss results.

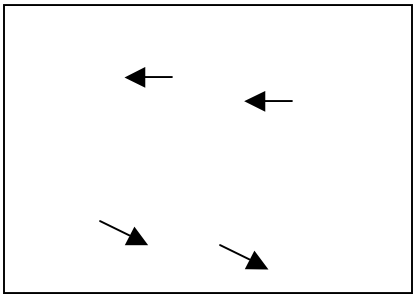


Outline Book 1



Outline Book 2

4) Group project extra credit or Extra Credit 2 for individual project: Point correspondence, motion vector field, and RST transformation (equivalent to 50% project grade). Develop algorithms to (a) determine the correspondence of the corner points; (b) compute the motion field using two images; and (c) determine the RST transformation parameters. Discuss results.



Motion vector field

- **Project Report:** Follow the required format. Attach these two sheets as the cover sheet for the report. Attach printouts of the results for the above images. Use "insert pictures" in WORD to insert the images within the text. Show the pictures in small sizes (about 1/8 to 1/16 of a page). Discuss the results and specific implementations

- In general, the algorithms should be computationally efficient and reliable. In addition, do not exploit any *a priori* information such as color of the background, shape of the book, etc. Certain reasonable assumptions may be employed such as object rigidity, motion constraints, etc.

END OF DOCUMENT

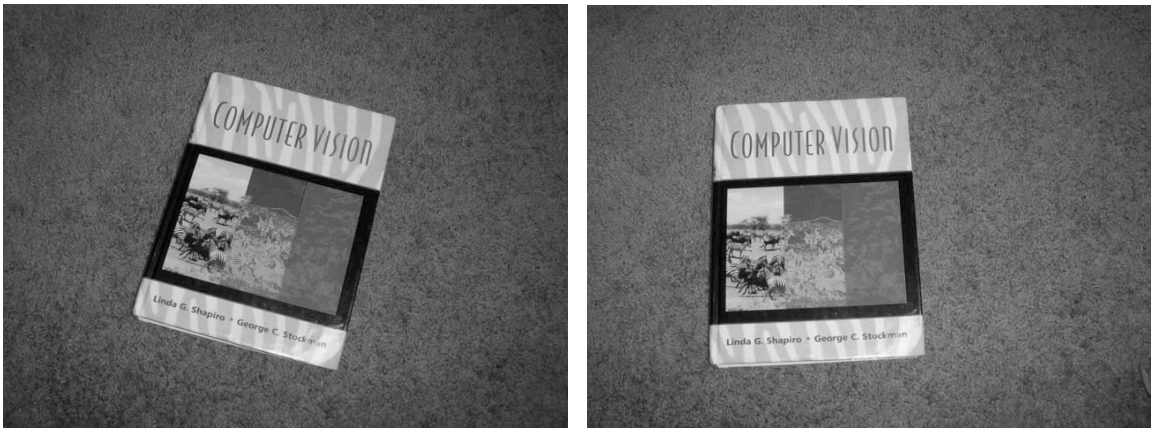
Shannon Foss
May 27, 2007
CECS 553
Project 3 + Extra Credit

Edge Detection and Hough Transform
Additional: Corner Detection

In this project we attempted to produce an algorithm to find the absolute edges of an object contained within a color image. Then after the edges are found, we would apply the Hough Transform to create an intensity image representing the most probable r and θ values of the lines.

Introduction

The images shown in Figure 1, both of which are a picture of a single book in slightly different positions that is relatively near the center of the image space, are to be used to determine the boundary edges of the object. Usually detecting the edges of an object is very easy, the simple application of an edge detecting operator mask will normally find the edges quite well. However, distinguishing between an object's boundary edges and the edges detected within the object, which are merely a change between colors or areas of light and dark, can sometimes be difficult.



(a) (b)

Figure 1: (a) Book1_grey.bmp – Image 1, a book near the center of the image turned slightly clockwise. (b) Book2_grey.bmp – Image 2, the same book, oriented vertically.

Once the outside edges of the object are found and an image of the boundaries has been made, the Hough Transform may be applied to the image to find the locations of the lines. This is accomplished by an accumulator array tabulating the data as each point in the image is evaluated. The product of which results in an image of the accumulator array's data where the brightest spots in the image are the most likely locations of the lines.

Edge Detection

When detecting edges we normally are trying to find places where there is a difference between areas, usually between light and dark, or patches of color or texture. However, when we are trying to detect the boundaries of an object, there may be places within the object where there are detectable edges. For

example, Figure 2 has several areas where there are detectable edges along with its actual boundaries which could make it difficult for the computer to determine which is which.

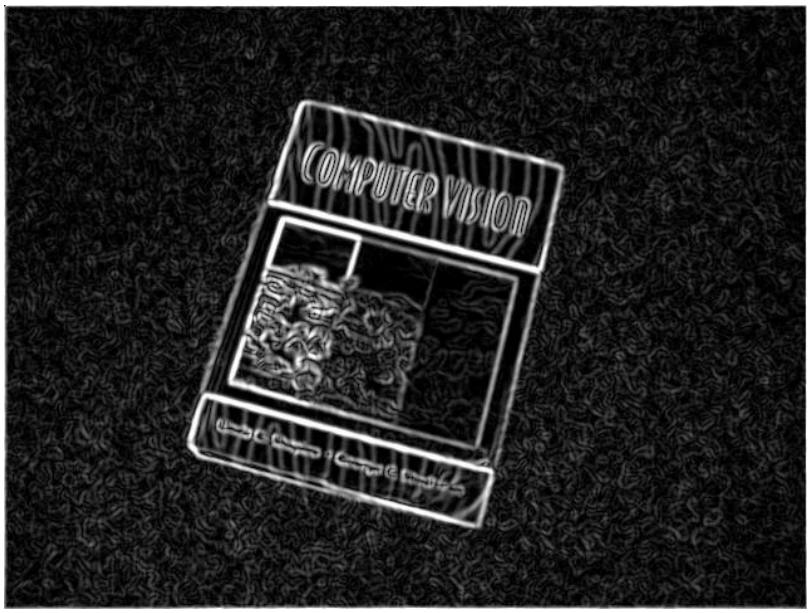


Figure 2: Application of Sobel's operator to the image Book1_grey.bmp.

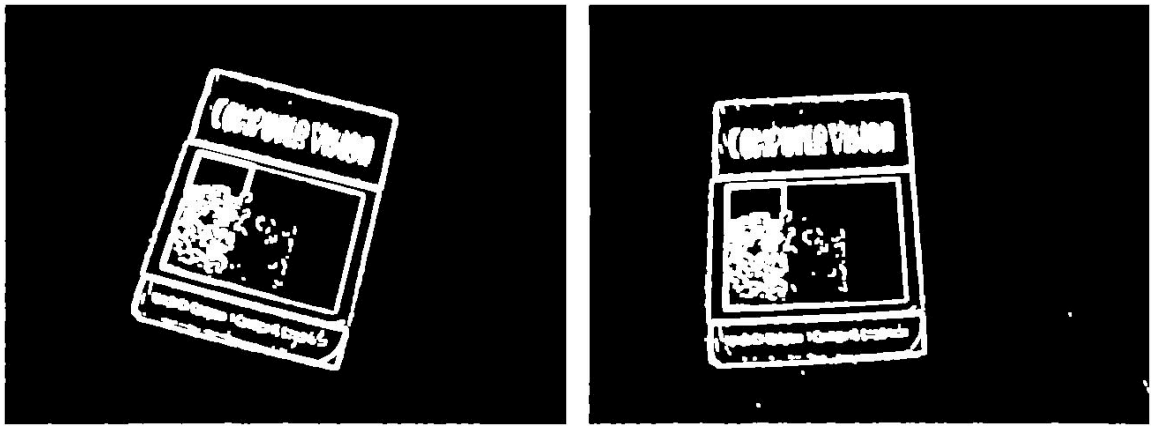
$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

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

Equation 1: Sobel's Operator

To account for this, a few simple things may be done to the image when processing for edge detection. Firstly, to get good edges without a lot of extra noise, a Gaussian filter is applied to the grey level image; this will blur the image enough so that sharp gradients are picked up less often by the Sobel operator. Next, the edge detection operator mask is applied to the blurred image, in this case Sobel's operator (Equation 1) was used because it comes up with brighter edges. After that, the image is then thresholded to eliminate any extra noise and to be sure that the edges that are picked up come out clearly. Finally, dilation is applied to thicken up the edges and to fill in any gaps resulting from the thresholding.

From Figure 3 (a) in comparison with Figure 2, it can be seen that the background has been totally or near completely eliminated because of the Gaussian blur and the thresholding. Due to the type of algorithm that is used to detect the outer boundaries, it is imperative that the background be, as much as possible, eliminated.

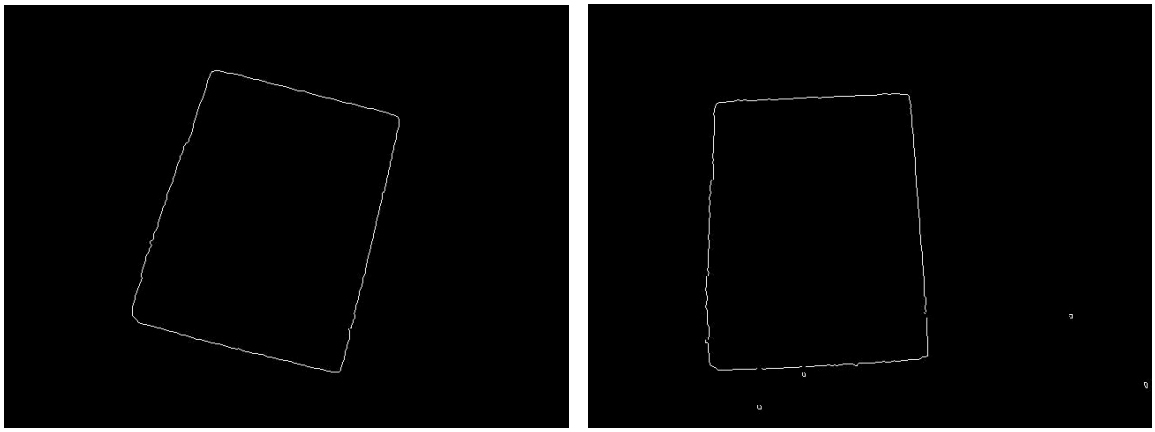


(a) (b)

Figure 3: (a) Book1_Edge.bmp & (b) Book2_Edge.bmp – Application of Sobel's operator after a Gaussian blur is applied, both images are then thresholded.

After these edges have been derived, the application of a simple algorithm may be applied to find the outer boundaries of the object. For this, we have to make a few assumptions about the image, one, there is an object in the image, and two, that there is only one closed contour object within the image. Once we assume these to be true we can easily find the boundary edges.

The algorithm that is applied to generate the outer boundary edges consists of iterating through each line of the image, once vertically, and once horizontally. Each line is examined for the first and last visible pixel of the object, and these locations are stored in a new array that is to be displayed as a new image (Figure 4). This is an incredibly simple algorithm, however it works quite well as long as there is little to no noise outside of the object's boundaries.



(a) (b)

Figure 4: (a) Book1_Boundary.bmp & (b) Book2_Boundary.bmp – Results of the application of the boundary finding algorithm.

Hough Transform

The Hough Transform is an iterative algorithm that finds the location of a line in an image. It goes through each pixel in the border image, and it applies the equation:

$$r = x\cos\theta + y\sin\theta$$

Equation 2: Equation of a line using r and θ.

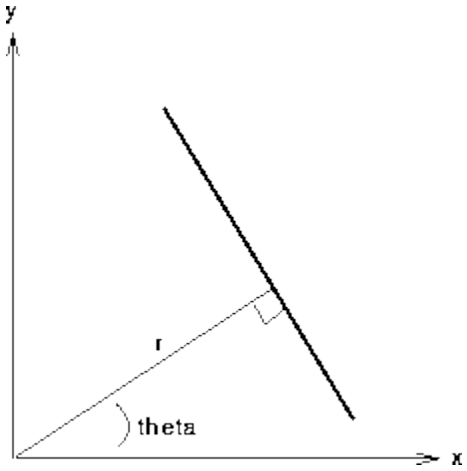
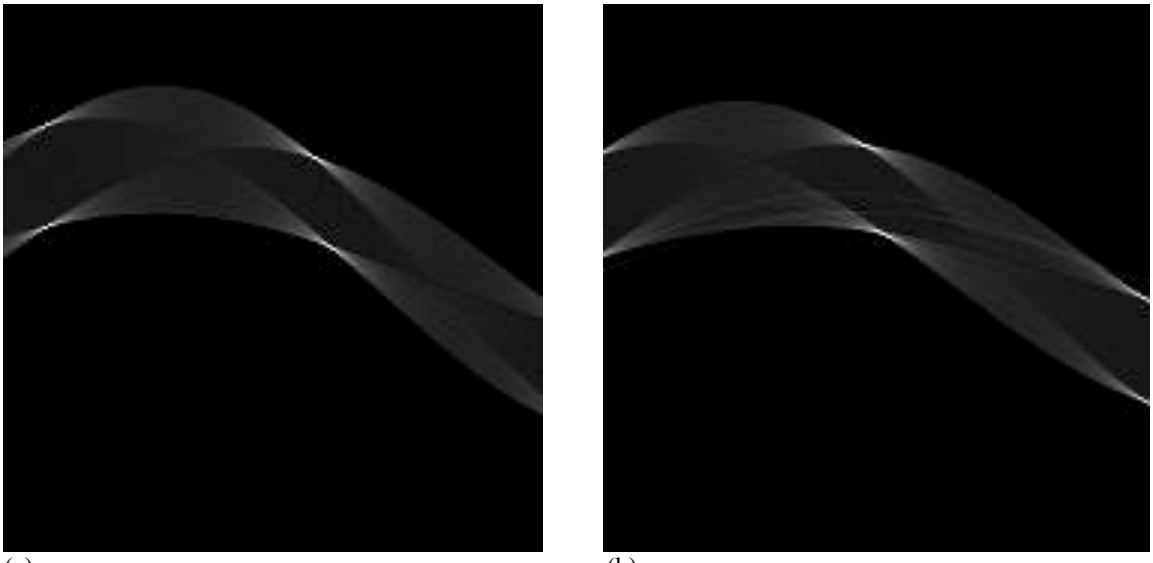


Figure 5: right - Representation of the values of r and θ in relation to the line. r is the perpendicular distance from the origin to the line, and θ is the angle from 0°.

using the x and y values from the border image array and iterating through different θ values to come up with different values of r that satisfy the equation of the line. The values for r and θ are used as indices for a new array whose values are incremented each time an occurrence of r and θ are found.

The more times an occurrence of r and θ are found, the higher the value of that element will be. This means that when it is displayed as an image, the locations of the line will be displayed as brighter points in the image.

The Hough Transform works well even when there is a lot of noise in the image being analyzed. As long as there are clear lines in the image the results will still show bright clear points marking where the line locations are at, this is because there is still so many points in the line being analyzed that they outweigh any rogue points that might misconstrue the resulting data.



(a) (b)

Figure 6: (a) Book1_Hough.bmp & (b) Book2_Hough.bmp – Resulting images from application of Hough Transform.

Figure 6 shows the results of the Hough Transform on the images of the book, the horizontal axis represents θ going from 0° to 179° and since the values for r can range from negative to positive values, the vertical axis is split by 0 midway up. The bright points in the image are the locations where the values received the most increments, meaning that these points are the most likely values for a line. Figure 6 (a) shows four bright points corresponding to the four lines in its boundary image. Figure 6 (b), which looks as though it is just a small phase shift from Figure 6 (a), also shows four points corresponding to the four lines in its boundary image as well, though it may look like six bright spots in the image, it is actually just four, because we have to remember that when evaluating values of θ that are greater than or equal to 180° they have equivalent reference angles It can also seen that the magnitudes of the extra bright spots are equivalent distances from the r=0 point on the vertical axis, so the extra bright spots in Figure 6 (b) are really just continuations of the same ones.

Conclusion

The edge detection I used to create these results was simple but seems capable of producing good results for most single object images with a relatively simple background. The Hough Transform was not difficult to implement either, and it comes up with rather good results that are quite accurate. In all it seems that simple algorithms that get the job done usually work well enough and get pretty good results.

Extra Credit – Corner Detection

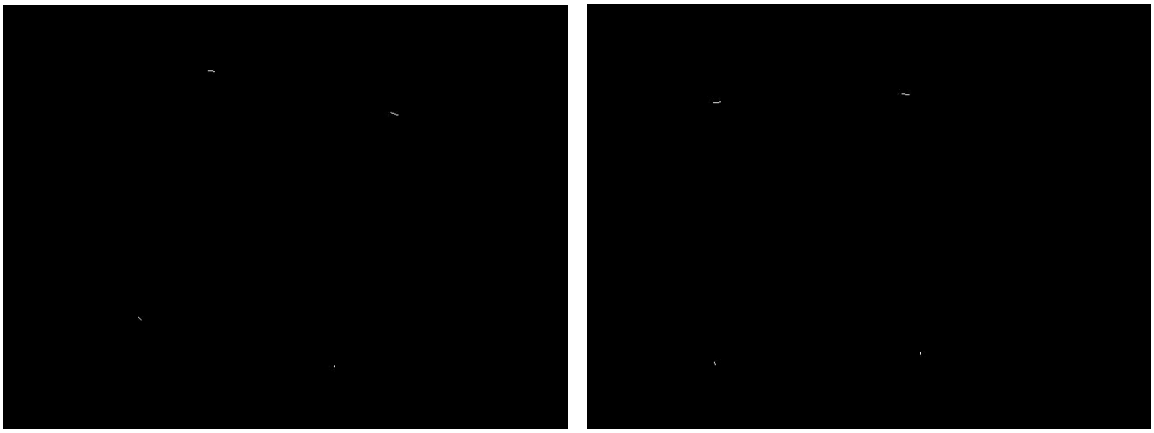
Corner Detection

To detect the corners I used a few large masks that would find locations of corners when applied to the border line image (Figure 4). It finds these corners by detecting where a line comes into the mask on one side and where it leaves on a perpendicular side. The mask consists of a pixel in the middle of one edge and a pixel anywhere along the perpendicular edge, this allows for corners of different angles to be detected as well. To detect the other four corners, similar mirror image masks are used.

						1
1	1	1	1	1	1	

Figure 7: Example of a mask: A line comes in through the right side and can leave anywhere on the lower side. This mask will easily detect top right hand corners.

The larger the mask was the better it did in detecting corners, usually masks that were smaller than 7x7 or 9x9 picked up too many rough edges and created too much noise.



(a) (b)

Figure 8: (a) Book1_Corners.bmp & (b) Book2_Corners.bmp – this shows the four corners on each image that were detected by the mask.

For being so simplistic, the mask did quite well in finding the corners of the book.