Tim Herrmann
EECS 332
MP6
Hough Transform

Programming Language:

- Python 2.7.6 with the following dependencies
  - matplotlib
  - numpy
  - cv2
  - pylab

Input:

Image – grayscale or color
Two floats representing the quantization step size for r and theta.
Example: "./hough.py input.bmp .5 .005"

Output:

A grayscale image showing only the lines detected by the algorithm.
A grayscale image showing the detected lines superimposed on the original image

Method:

Step 1: Read the image into a grayscale format
Step 2: Apply a Canny edge detector to identify the image pixels that are considered edges.
Step 3: Initialize an empty 2D "accumulator" array. The length of first dimension is twice the length of the image diagonal divided by specified quantization value. The length of the second dimension is pi divided by the second quantization value from the input.
Step 4: For each pixel identified as an edge pixel:

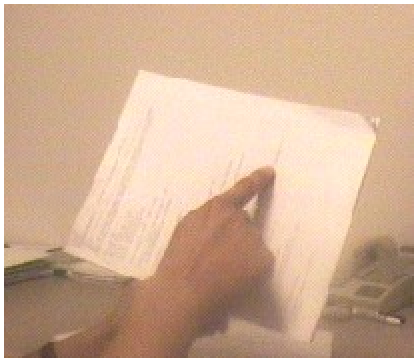
- starting at theta = 0, calculate r = x*cos(theta) + y*sin(theta)
- to account for negative values, add the length of the image diagonal
- increment the accumulator array bin that corresponds to the theta and r just calculated
- increment theta by the specified quantization value.

Step 5: Find Critical Intersections. I apply a threshold to the accumulator array to only keep the largest bins.
Step 6: Find maxima: To avoid plotting several lines of nearly identical slope and intercept, I analyze the accumulator array for local maxima. I do this by analyzing the arrays in 10x10 sub arrays and eliminating all bins that are not the maximum for that particular sub array. I overlap sub arrays to ensure that only the true maxima remain.
Step 7: Convert the remaining accumulator bins back to cartesian coordinates and plot using the equation "y = mx+b"

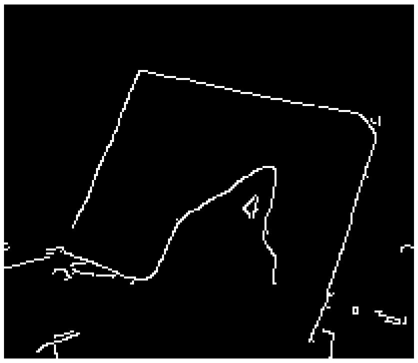
Examples and Analysis:

Input.bmp – r step size: .5 - theta step size: .005

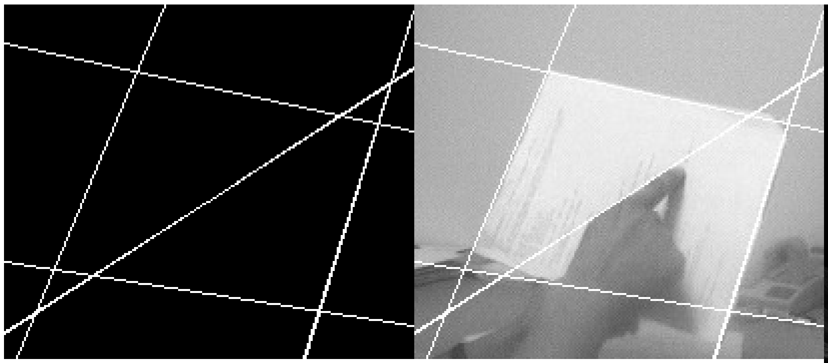
Original Image



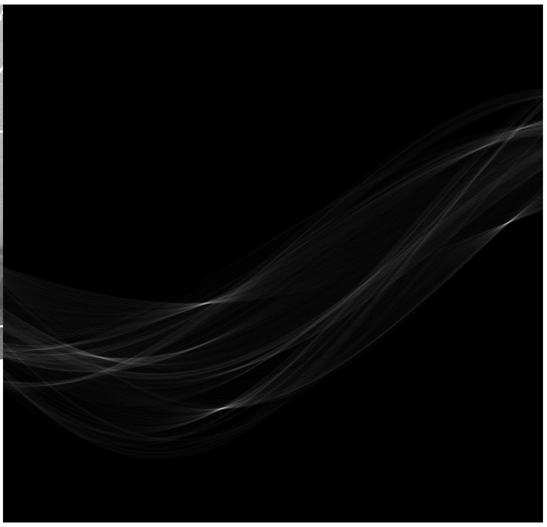
Edges



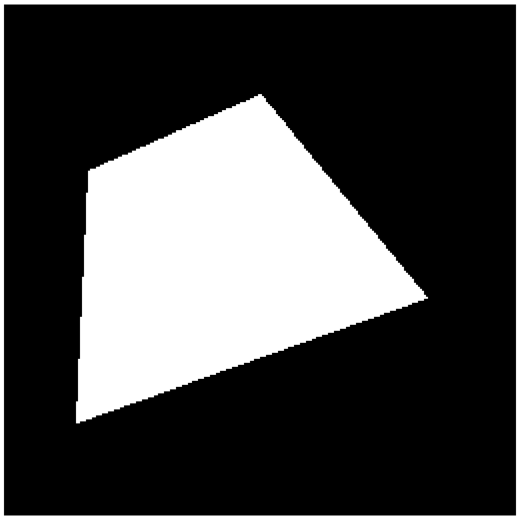
Lines

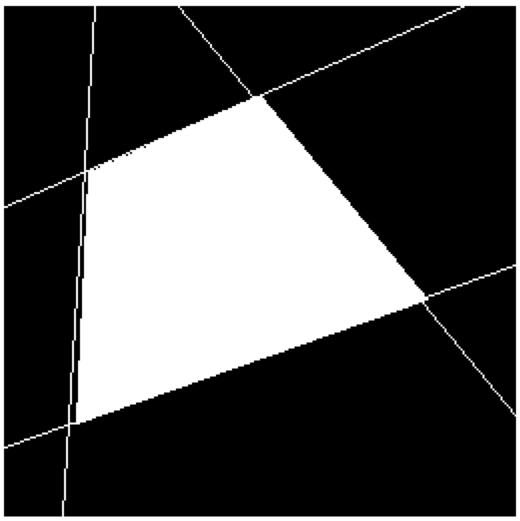


Visualization of Accumulator Array

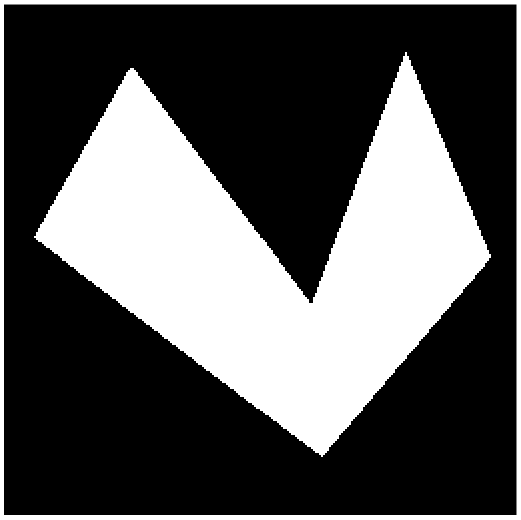


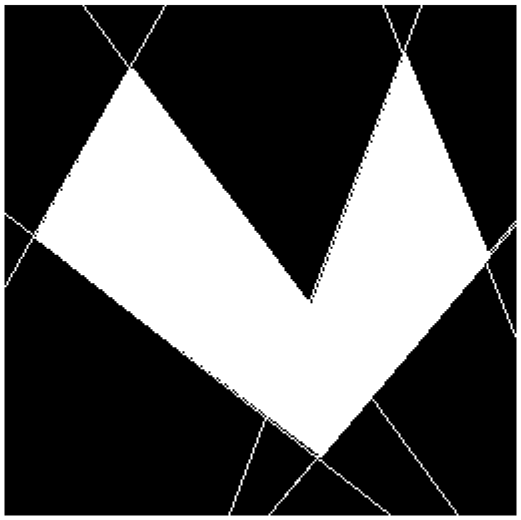
Test.bmp – r step: 1 theta step: .01

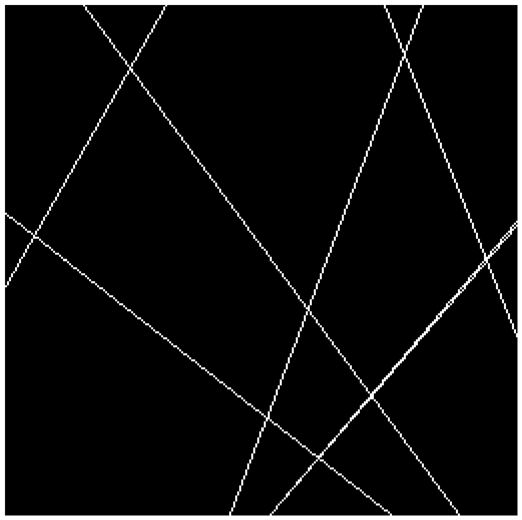




Test2.bmp   - r step:

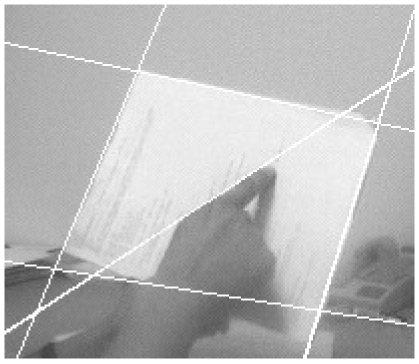




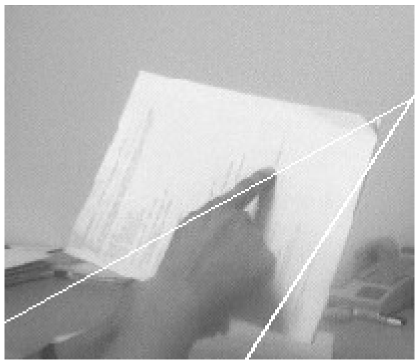


Quantization Comparison:

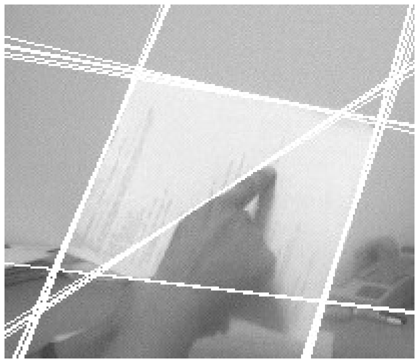
r = .5   theta = .005



r = 1   theta = .5



r = .1   theta = .001



When the step sizes are too large, lines are missed and tend to be shifted and angled slightly from where they should be. Ont he other hand, when the step sizes are very small, the local maxima get spread out and multiple lines are detected for each true line. Also, the program takes exponentially longer to run.