MP6 - Hough Transform

Logan Boswell

Overview

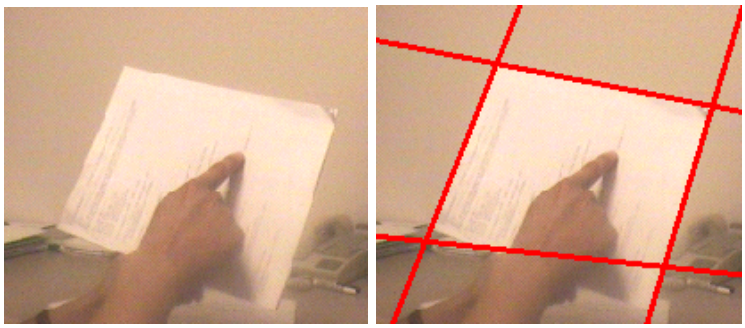
This MP is focused on implementing the Hough Transform. In order to implement this technique, a python function was written that takes in images and identifies prominent lines.

Hough Transform

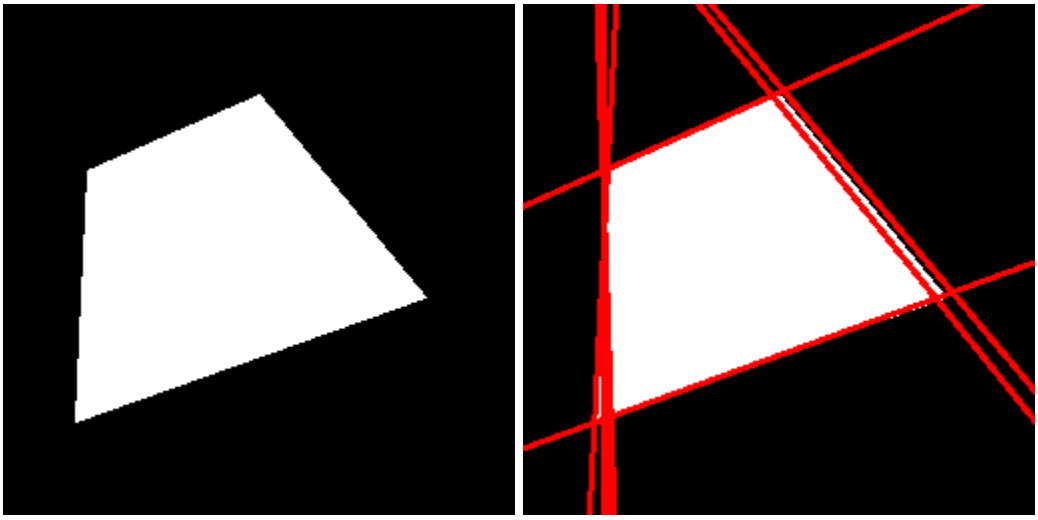
The first step taken in my Hough Transform function is to perform edge detection on an input image. For edge detection, I used my CannyEdgeDetection function from MP5. Next, I set up an accumulator array based on the size of the image, with 181 rows (for the angles in degrees ranging from -90 to 90) and the number of columns corresponding to two times the length of the diagonal of the input image plus 1. With the accumulator matrix set up, the row, column pairs could be mapped to the parameter space (theta, rho). Each edge pixel was mapped to the parameter space by using the current pixel's row, column pair with an array of thetas determined based on a parameter quantization argument in the function to find the corresponding rho values, and incrementing elements of the accumulator array based on the rho and theta values. After all of the edge pixels have been checked, a maximum filter is applied followed by masking to identify significant intersections in the parameter space. These significant intersections were then converted to row, column pairs and used to add lines to the original image and the image with lines is returned.

Results

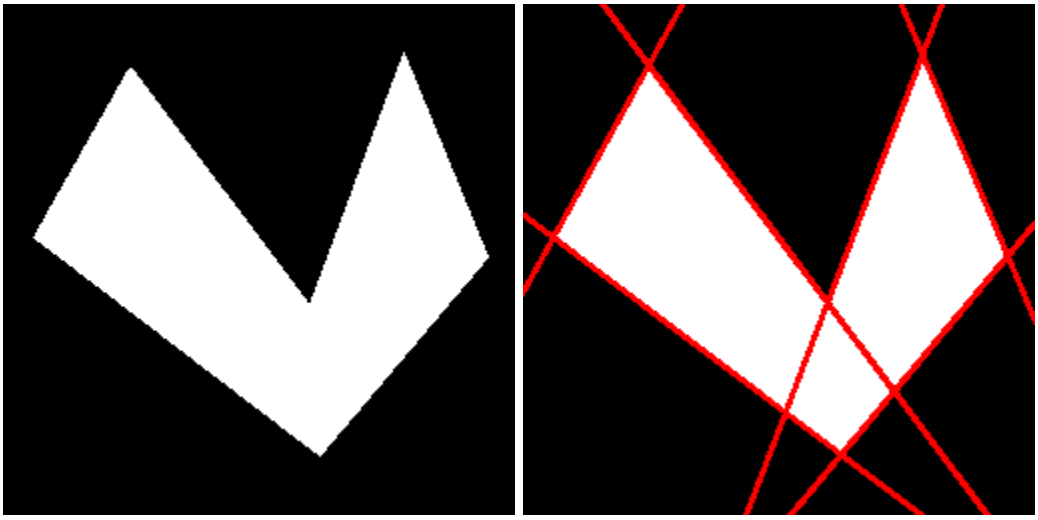
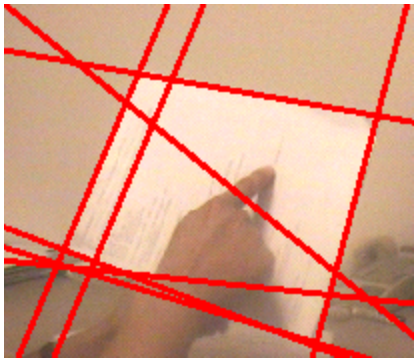
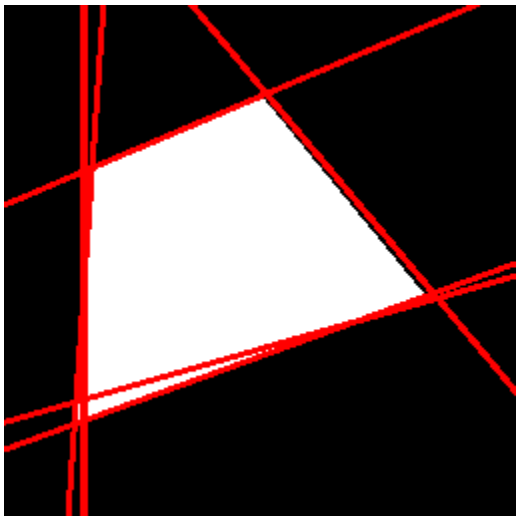
Here is the testing result with input.bmp and N=5, sigma=1, param_quant=180, max_filt_N=15, peaks_thresh_mul=.61, percentageOfNonEdge=.85



Here is the testing result with test.bmp and N=5, sigma=1, percentageOfNonEdge=.94, param_quant=180, max_filt_N=15, peaks_thresh_mul=.21



Here is the testing result with test2.bmp and N=5, sigma=1, percentageOfNonEdge=.94, param_quant=180, max_filt_N=15, peaks_thresh_mul=.22



Here are the images when param_quant is reduced to 50