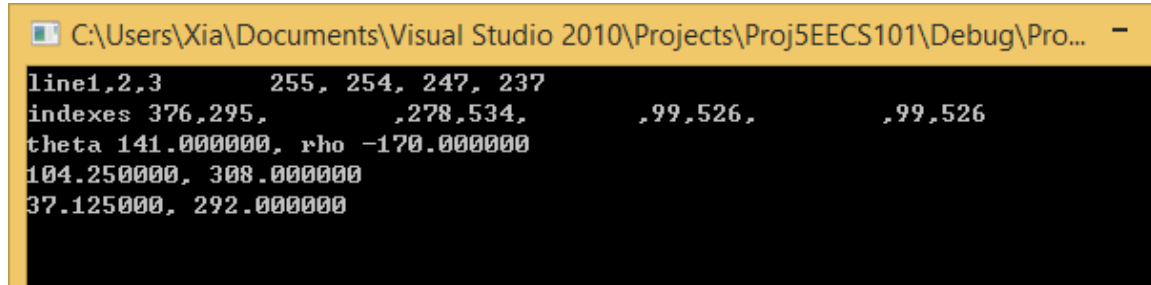


EECS 101 HW5

Results:

Line 1,2,3 got 255, 254,247 votes

Rho in degrees, and Theta values of (141,-170),(104.25,308),(37.125,292)



```
C:\Users\Xia\Documents\Visual Studio 2010\Projects\Proj5EECS101\Debug\Pro...  
line1,2,3 255, 254, 247, 237  
indexes 376,295, ,278,534, ,99,526, ,99,526  
theta 141.000000, rho -170.000000  
104.250000, 308.000000  
37.125000, 292.000000
```

Finding Theta and Rho and implementing voting array:

At every pixel of the binary image, a set of Rho and theta are calculated 480 times, by the step of theta being $\pi/480$. For each theta value, a Rho is calculated based upon theta and its x,y location. Then find the max and min value of Rho, and offset it to be positive starting from 0. Then the range of rho is divided by 640 to get the step size for normalizing it into a voting array of 480x640, just like the image array. Then, a vote is casted by indexing the theta and rho value of multiples of their respective step sizes. Each time if there's a hit, increment the vote[i][j] by 1. Loop thru the entire binary image array.

Then sort thru the voting results, finding most popular votes which are also far away, (10 indexes away). The most popular 3 votes will be the rho and theta for the 3 lines. Then, reverse the process find the actual angle and distance for the curves using the index of the most popular votes. The entire conversion is relied on Hough transform.

The voting array is a 2-D array of size 480x640. First set the popular votes to be 0, which are line1, 2, 3 in my code. Then loop thru the entire vote array, finding the top 3 values of the votes.

Threshold : 190, this is the closest I can get to obtain all 3 line theta and rho in one shot.

Reconstructing:

With the most popular votes, the indexes are then converted back to x,y coordinates system. Once a pair of (x,y) are calculated to be on the line defined by rho and theta, the binary value of the pixel is changed to 255. The process also loops thru the entire 2-D array by inserting all x-values into the inverse Hough transform function.