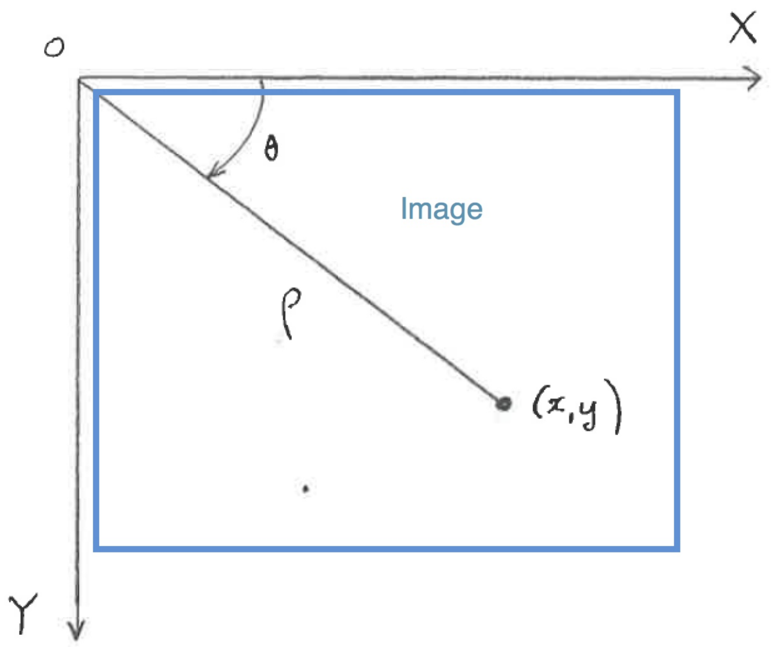
CENG 5436- Computer Vision and Applications

Problem set 2

Important notes: You may use the Matlab or OpenCV's edge operator **but you may not use any Matlab or OpenCV Hough functions or tools**.

1. Load the image file img = 'window.png' posted on the class blackboard. Generate an edge image (window_edge) of img and save it as 'window_edge.png'. Display the edge image (using imshow) for your TA to see and show him/her what method and any threshold value that you used.

2. Implement a Hough Transform method for finding lines using the following coordinate system.



Thus, the pixel at img(row,column) corresponds to the (x,y) coordinates, meaning, x = column and y = row. This pixel should vote for line parameters (ρ, θ) where

$$\rho = x \ \cos\theta + y \ \sin\theta$$
$$\theta = atan2(y, x)$$

a. Write a function hough_lines_votes which computes the Hough Transform for lines and produces the votes accumulator array H, the θ values corresponding to columns of H and ρ values corresponding to rows of H. Your code should conform to the specifications of the Matlab function hough: (documentation on function hough can be invoked by the Matlab command "doc hough")

   ```
   [H, theta, rho] = hough_lines_votes(edge_image, rho_res, theta_res)
   ```

   The final H array should be normalized to [0,255]. Apply this function to the edge image window_edge from problem1, display H as an image for your TA to see.

b. Write a function hough_peaks that finds indices of the votes accumulator array H that correspond to local maxima. This function returns an Nx2 with ρ indices in column 1 and θ indices in column 2, and N is the number of peaks found. See Matlab documentation on houghpeaks function (invoked by Matlab command "doc houghpeaks"). Call your function with the votes accumulator from the step above to find up to 10 strongest lines

   ```
   peaks = hough_peaks(H,10,Threshhold,NHoodSize)
   ```

   Draw a square with 4-pixel sides at each peak and display the image of the peaks for your TA to see.

c. Write a function hough_lines_draw to draw color lines that correspond to the peaks found in the votes accumulator array H. This means you will need to look up values of θ and ρ using the peak indices, convert them to line parameters in Cartesian coordinates so you can use line-drawing functions. Use this function to draw lines on the original grayscale (not edge) image. The lines should extend to the edges of the imge (aka infinite lines):

   ```
   hough_lines_draw(img, outfile, peaks, rho, theta)
   ```

   Show the image to your TA and save it in the file named 'window_lines.png'.

3. Window image with noise
   a. Load the image 'window_w_noise.png' from class blackboard. Use a Gaussian filter to produce a modestly smoothed version of this image. Make σ at least a few pixels wide. Display the original image window_w_noise and its smoothed image side by side for your TA to see.

   b. Using an edge operator of your choosing, create a binary edge image for both the original image window_w_noise_edge and its smoothed version. Display both images for your TA to see.

   c. Now apply your Hough method to the smoothed version of the edge image. Adjust the filtering, edge finding and Hough algorithms to find the lines as best as you can for this test case. Display the Hough lines votes accumulator array H with highlighted peaks. Display the original image 'window_w_noise.png' with lines drawn on them.

4. Load the image 'coins_and_pens.png' from class blackboard
   a. Create a monochrome version of the image using the grayscale function conversion and compute a modestly smoothed version of this image using a Gaussian filter with σ at least a few pixels wide. Display the original grayscale image. Display the smoothed monochrome image

   b. Create an edge image for the smoothed version above. Display the edge image.

   c. Apply your Hough algorithm to the edge image to find lines along the pens. Draw the lines in color on the original monochrome image. Display Hough votes accumulator array image with peaks highlighted. Display the original monochrome image with lines drawn on it. Was there anything special that you had to do to get a good result.

5. Write a circle finding version of the Hough transform
   a. Implement a function hough_circles_votes to compute the votes accumulator array H for a given radius. Using the edge image obtained from Problem 4-b and try calling your function with radius = 20
      Find the circle centers by using the same peak finding function hough_peaks

      ```
      centers = hough_peaks(H,10)
      ```

      Draw and display the circles on the original monochrome image.

   b. Implement a function find_circles that combines the above two steps, searching for circles within a given radius range, and returns circle centers along with their radii:

      ```
      [center, radii] = find_circles(img_edges, [20 50]);
      ```