Georgia Institute of Technology
George W. Woodruff School of Mechanical Engineering
ME6406 Machine Vision (Fall 2023)

Assignment #2: (ME6406Q Due Saturday, **October 7th** , 2023**, 23:59pm EDT**)

All programs should be written using MATLAB. Solutions must be consolidated into a **single pdf file** (including all results and an explanation of results) and a **zip file** (including all m-files used for the results). Solutions must be submitted electronically through **Canvas**. Late solutions will be penalized at a 10% deduction from the homework score and will NOT be accepted 24 hours after the due date. Without a signed honor code, your exams and assignments will NOT be graded.

1. <u>Hough Transform</u>

   a. Show that the foot-of-normal parameters $\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \upsilon \begin{bmatrix} g_x \\ g_y \end{bmatrix}$ where $\upsilon = \dfrac{xg_x + yg_y}{g_x^2 + g_y^2}$.

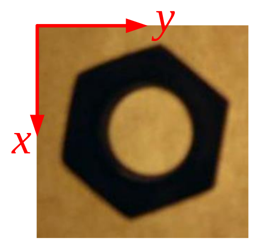

Fig. 1 HW2.jpg

   b. With the equation derived in Part 1*a* to solve the parameters (x_o, y_o), write a MATLAB program to perform the Hough transform **based on the foot-of-normal parameterization** on the image 'HW2.jpg' to find the edge lines:

   1) Convert the RGB image to a gray-level image followed by obtaining the binary edge image (edge pixel: 1, others: 0) and by applying the Sobel operation to get the gradient, so that the image can be characterized by a set of edge points with known gradient components (x_i, y_i; g_x, g_y).

      Suggested Matlab functions: *rgb2gray.m*, *edge.m*, *bwboundaries.m*, *imfliter.m*.

   2) Traverse through the edge pixels and calculate x_o and y_o. Determine the Hough matrix H. You may use *houghpeaks.m* to detect the peaks in H. Suggested MATLAB function: *histcounts.m*/*histcounts2.m*

   3) From the peaks in H, determine the equations that characterize the straight edges in 'HW2.jpg'.

   Note: The Matlab commands *hough.m*, *houghlines.m* are not allowed.

   c. Use Hough Transform to detect the circle in 'HW2.jpg'. Compare your results with MATLAB function *imfindcircle.m*.

2. <u>Feature Points Detection</u>

   a. Use the ***rho-theta signature*** method discussed in class and implement MATLAB to locate all the corners of the object in 'HW2.jpg' (Fig. 1). Plot the $\rho\theta$ signature to locate the corners. Show the coordinates of the corners with respect to the given coordinate system.

   b. Repeat Part 2*a* with the ***median length*** method. Pick an appropriate *N* (length of the neighbor). Plot the median length as a function of the path. Show the coordinates of the corners with respect to the given coordinate system.

   c. Repeat Part 2*a* with the ***curvature*** method, which is to locate the 'peaks' of the curvature along the boundary. Plot the graph that you use to locate the corners. Show the coordinates of the corners with respect to the given coordinate system.

3. <u>Template Matching</u>

a. Forward transformation: Write a MATLAB function to perform the transformation for three template points with the parameters as shown in Table 1. Show and plot three points before and after transformation. The transformed points are accurate to three decimal places.

Table 1

Template				Parameters
Features	1	2	3	$(x_d, y_d) = (8, 5)$
X	0	6	2	$\theta = 30^\circ$
Y	0	3	7	$k = 1.2$

b. Pseudo inverse method: Write a MATLAB function to perform the pseudo-inverse method on the template and transformation points in Part 3*a* to find parameters (k, θ, x_d, and y_d). Check these values in Table 1.

c. Polygon matching: Write a MATLAB program to perform template matching on the template and target as shown in Table 2 and Fig. 2 to identify the match points, and determine the transformation parameters; namely scale k, orientation θ, and displacement (x_d, y_d).

Table 2

	Template				
	1	2	3	4	5
X	2	5	5	4	1
Y	1	2	4	5	3

	Target		
	a	b	c
x	9.697	8.566	6.869
y	6.96	8.091	5.263

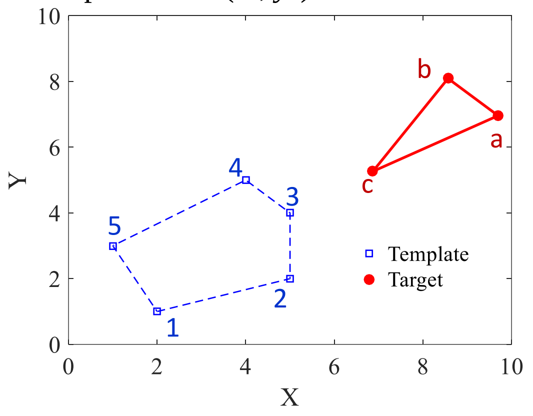

Fig. 2

Note 1: Given n points, there are $\binom{n}{3} = \frac{n!}{(n-3)!3!}$ unique triangles. Use Matlab function *nchoosek*( ) to compute all possible combinations of vertices to form unique triangles.

Note 2: You may find the Matlab function *sort*() useful to arrange the order of vertex in a triangle by the corresponding length.

Note 3: As there are many triangles to be matched, your program must automate the matching process to avoid tedious manual matching.

Note 4: Determine the 'best' match by finding the one that yields the smallest 'matching error' defined by Eq. (11) in [Lee, *et al.*, 1992]:

$$E = \sum_{i=1}^{k} \sqrt{(X_{ti} - X_i)^2 + (Y_{ti} - Y_i)^2}$$

Reference:

Lee, K-M. and S. Janakiraman, "A Model-based Vision Algorithm for Real-Time Flexible Part-feeding and Assembly," Paper number: MS 92-211. *SME Applied Machine Vision Conf.*, June 1-4, 1992, Atlanta, GA.