Georgia Institute of Technology
George W. Woodruff School of Mechanical Engineering
ME6406 Machine Vision (Fall 2021)
Assignment #2: (ME6406A Due **Wednesday October 6th 2021, 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 10% deduction from the homework score, and will NOT be accepted 24 hours after due date.

1. Template Matching

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			
Features	1	2	3
X	0	7	3
Y	0	4	8

Parameters
$(x_d, y_d) = (6, 7)$
$\theta = 60^\circ$
$k = 0.8$

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

c. Polygon matching:
Write a MATLAB program to perform template matching on the template and target as shown in Table 2 and Figure 1 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	6	8	5	-3
Y	0	2	6	8	5
	Target				
	a		b		c
x	2.28		10.621		9.545
y	16.28		10.318		15.576

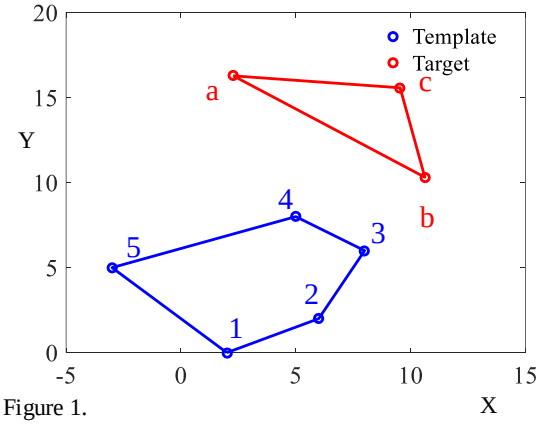

Figure 1.

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 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]. ($E = \sum_{i=1}^{k} \sqrt{(x_{ti} - x_i)^2 + (y_{ti} - y_i)^2}$)

2. Feature Points Detection

a. Use ***rho-theta signature*** method discussed in class, implement using MATLAB, to locate all the corners of the object in 'HW2.png'. Plot the $\rho\theta$ signature for locating the corners. Show the coordinates of the corners with respect to the coordinate system as shown.

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

c. Repeat (a) with ***curvature*** method, which is to locate the 'peaks' of the curvature along the boundary. Plot the graph that you used to locate the corners. Show the coordinates of the corners with respect to the coordinate system as shown.

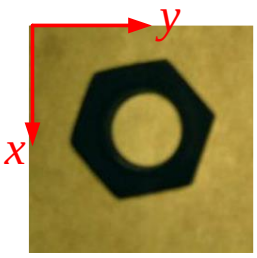

Figure 2. HW2.png

Figure 3. Camera.png

3. Hough Transform

a. Show that

$$\begin{bmatrix} x_o & y_o \end{bmatrix}^{\mathrm{T}} = \upsilon \begin{bmatrix} g_x & g_y \end{bmatrix}^{\mathrm{T}}$$

where $\upsilon = \left(x g_x + y g_y \right) / \left(g_x^2 + g_y^2 \right)$.

b. With the parameters (x_o, y_o), write a MATLAB program to perform Hough transform (**Foot-of-normal parameterization**) on image 'Camera.png' to find the yellow edge line:

1) Convert RGB image to gray level image and followed by the binary edge image (edge pixel: 1, others: 0). Suggested Matlab functions: rgb2gray.m, edge.m.
2) Apply the Sobel operation on the binary edge image in (1). Suggested Matlab functions: imfliter.m.
3) Determine the accumulator matrix for (x_0, y_0).
4) Find (x_0, y_0) and the equation of the yellow edge of the camera.

c. Use Hough Transform on the points of inner boundary to find the circle in HW2.png.

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

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.