Due: 25th Feb 2016 11:55 PM Total points: **20**

1 Understanding RANSAC

In this part of the assignment, you will implement the RANSAC algorithm discussed in class for two different sets of problems. In the first problem, you will have to find the line that fits the given 2-D data the best using RANSAC algorithm. Note that you will have to choose the parameters for the algorithm carefully in order for it to work. In the second problem, you will have to find the affine parameters that best describe the transform from a set of points in the first image to the corresponding set of points in the transformed image, again by making use of the RANSAC algorithm, this time however on 6-D data instead of 2-D data. This is explained in detail later

1.1 Ransac for Line Fitting

4 points

To acquaint yourself with RANSAC, you will implement the RANSAC algorithm for fitting a line for the 2-D data provided to you in the mat file 'LineData.mat'. Before you start, you are advised go through the notes once again so that you are clear about the algorithm. You can also check out the wiki page on this topic which has pseudo-code for the algorithm. Be sure to include a 2-D data with your colored best-fit line along with answers to the following questions:

1. What value will you choose for the initial number of points s and why?
2. What would be an appropriate choice for the threshold t?
3. What should be the value of the number of samples N so that with probability $p = 0.9999$, at least one sample is free from outliers?

1.2 Ransac for Affine Fitting

6 points

An affine transform can be be viewed as a combination of some of the more basic transforms such as translation, rotation, scaling and shear. Affine transforms preserve straight lines and ratios of distances between points lying on a straight line. Mathematically, it can be described by the following equation:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} \times \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

where (u_x, u_y) is a point in 2-D space, (v_x, v_y) is its corresponding affine transformation and $a_1, \ldots, a_6$ are the affine parameters.
For instance with a_1, $a_5 = 1$ and a_2, $a_4 = 0$, the above equation becomes:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & a_3 \\ 0 & 1 & a_6 \end{bmatrix} \times \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

which gives $v_x = u_x + a_3$ and $v_y = u_y + a_6$ which corresponds to a simple translation. Similarly, you can verify that

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \end{bmatrix} \times \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

and

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \end{bmatrix} \times \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

corresponds to scaling by s_x and s_y in the x and y directions and pure rotation by angle θ around the origin respectively.

You are given an image 'CastleOriginal.png' (which we shall refer to as I_1) and its corresponding Affine transformed image 'CastleTransformed.png' (I_2). Moreover, you are also given a set of points (u_{x_i},u_{y_i}) chosen from the first image and its corresponding set of points (v_{x_i}, v_{y_i}) from the second image in the mat file 'AffineData.mat'. These points have been carefully chosen using an algorithm known as the Spatially Invariant Feature Transform (SIFT). This algorithm is widely used to identify and describe local features in an image.

You are expected to find the affine parameters $a_1, \ldots, a_6$ using point correspondences and the equation described above. (Hint: One can combine the 6 equations corresponding to the 3 point-correspondences and write it as a single equation $Ax = b$ where A is a 6×6 matrix containing the 3 randomly chosen entries for the points from image 'Lena.png', x is a 6×1 matrix containing the affine parameters $a_1, \ldots, a_6$, and b is a 6×1 matrix containing the corresponding points from the image 'CastleTransformed.png. Now one can invert this equation to obtain the affine parameters.) Continue with the RANSAC algorithm.

Now use these estimated parameters to transform the original image of the castle (I_1) using the Matlab function **imtransform** to get the new transformed image of castle (I_{new}). How does I_new compare with I_2? Put both these images in your report along with the affine parameters you found and answers to the following questions:

1. What value will you choose for the initial number of points s and why?

2. What would be an appropriate choice for the threshold t?

3. What should be the value of the number of samples N so that with probability $p = 0.9999$, at least one sample is free from outliers?

2 Distance Transforms and Chamfer Matching

In this part of the assignment, you have to implement the L_1 distance transform algorithm and later use it for the purpose of Chamfer matching.

2.1 Distance Transform

6 points

A distance transform (as related to images) is a technique used to determine "closeness" to a given structure of interest. Specifically, it is a map from an image of such a structure (generally binary) to an image which contains as pixel values the "distance" from each respective pixel location to the "nearest" structure pixel. How distance is quantified determines which structure pixel is closest and what the distance to that pixel is. Possible measures include Euclidean distance (L_2 norm), Manhattan distance (L_1 norm), Chessboard distance (L_1 norm), etc. In this assignment, we will be using the L_1 distance transform: for two given points (x_1, y_1) and (x_2, y_2) it is defined as $|x_1 - x_2| + |y_1 - y_2|$.

In order to compute a distance transform, you will need a structure of interest. One such structure is an edge map of a cow. You will need to compute the Canny edge map yourself, but the cow is provided in 'cow.png'. You can use Matlab's **edge** function, or you can use the code you wrote for an earlier assignment. With the edge map, you are now ready to compute the L_1 distance transform. How you go about doing this is up to you, but you can refer to the following link for ideas `http://www.cs.cornell.edu/courses/cs664/2008sp/handouts/cs664-7-dtrans.pdf`. Remember to initialize your algorithm correctly.

1. Compute the Canny edge map image of the cow and display the result.

2. Compute the L_1 distance transform image of the cow with your own code and display the result.

Matlab also has a function to compute distance transforms called **bwdist**. With this function, you can specify the distance metric you intend to use while computing the distance transform.

3. Compute the distance transform of the cow edge map with three different metrics using **bwdist**. Comment on the differences between these. Under what circumstances would the various distance metrics be useful?

2.2 Chamfer Matching

4 points

Now, write a Matlab script to correctly position the edge template ‘template.png’ on the cow in ‘cow.png’. To do this exhaustively for translation requires probing the distance transform with the template at different positions and calculating the Chamfer distance for each.

1. What is the minimum Chamfer distance for the given template?
2. Is there a better way to search for the minimum chamfer distance rather than doing an exhaustive search? Suggest an alternative.
3. Display the final image of the cow with the template correctly superimposed on it.

Submission Instructions

Every student must submit following 2 files:

- An organized report submitted as a PDF document. The report should describe the implementation, issues (problems encountered, surprises), and an analysis of the test results (interpretation of effects of varying parameters, different image results). Intermediate and final results must be provided.
- A ZIP file containing the necessary codes.

The heading of the PDF file should contain the assignment number and topic. Also, attach a photo of yourself at top-left of the PDF along with your name and department.

Late Submission Policy

Assignments are expected to be submitted on the due date. Each student gets a total of 3 late days that can be used however you wish. For examples, all 3 days can be used towards 1 assignment or 1 day late for 3 assignments or other combinations. Late submissions beyond that will be penalized as below:

- One day late will be penalized 25% of the credit.
- Two Days late will be penalized 50%.
- Submissions more than 2 days late will not be considered for credit.

I will be ruthless in enforcing this policy. There will be no exceptions

Collaboration Policy

I encourage collaboration both inside and outside class. You may talk to other students for general ideas and concepts but the programming must be done independently. For mid-term and final examination there will be no collaboration permitted.

Plagiarism

Plagiarism of any form will not be tolerated. You are expected to credit all sources explicitly. If you have any doubts regarding what is and is not plagiarism, talk to me.