Assignment 1: CS 663, Fall 2021

Due: 18th August before 11:55 pm

Remember the honor code while submitting this (and every other) assignment. You may discuss broad ideas with other students or ask me for any difficulties, but the code you implement and the answers you write must be your own. We will adopt a zero-tolerance policy against any violation.

Submission instructions: Follow the instructions for the submission format and the naming convention of your files from the submission guidelines file in the homework folder. Please see assignment1.zip in the homework folder. For all the questions, write your answers and scan them, or type them out in word/Latex. In eithe case, create a separate PDF file. The last two questions will also have code in addition to the PDF file. Once you have finished the solutions to all questions, prepare a single zip file and upload the file on moodle before 11:55 pm on 6th November. We will not penalize submission of the files till 10 am on 19th August. **No assignments will be accepted after this time.** Please preserve a copy of all your work until the end of the semester. **Your zip file should have the following naming convention:** RollNumber1_RollNumber2_RollNumber3.zip for three-member groups, RollNumber1_RollNumber2.zip for two-member groups and RollNumber1.zip for single-member groups.

1. A function $f(z)$ is said to be linear if $f(az_1 + z_2) = af(z_1) + f(z_2)$, where z_1, z_2 are two scalar values in the domain of f and a is a scalar constant. This definition extends to the case where $\boldsymbol{z}$ is a vector, i.e. $f(\boldsymbol{z})$ is linear if $f(a\boldsymbol{z_1} + \boldsymbol{z_2}) = af(\boldsymbol{z_1}) + f(\boldsymbol{z_2})$. Now, we have seen the formula for bilinear interpolation which expresses image intensities as a bilinear function of x, y (spatial coordinates) in the form $v(x, y) = ax + by + cxy + d$ where a, b, c, d are scalar constants independent of x, y. Is $v(x, y)$ a linear function of x keeping y constant and vice-versa? Prove or disprove. Is it a linear function of $\boldsymbol{z} \triangleq (x, y)$? Prove or disprove. [15 points]

2. Suppose an image is subjected to histogram equalization. Prove that another round of histogram equalization will produce the exact same result as the first round. [15 points]

3. Consider two images I and J whose intensity values (in each location) are randomly drawn from the known probability mass functions (PMFs) $p_I(i)$ and $p_J(j)$ respectively. Derive an expression for the PMF of the image $I + J$. The expression resembles which operation that we are currently studying in class (and will discuss on 13th August during the interaction session)? [15+5=20 points]

4. Read in the images T1.jpg and T2.jpg from the homework folder using the MATLAB function `imread` and cast them as a double array. Let us call these images as J1 and J2. These are magnetic resonance images of a portion of the human brain, acquired with different settings of the MRI machine. They both represent the same anatomical structures and are perfectly aligned (i.e. any pixel at location (x, y) in both images represents the exact same physical entity). We are going to perform a simulation experiment for image alignment in a setting where the image intensities of physically corresponding pixels are different. To this end, do as follows:

   (a) Write a piece of MATLAB code to rotate the second image by $\theta = 28.5$ degrees anti-clockwise. You can use the `imrotate` function in MATLAB to implement the rotation using any interpolation method. Note that the rotation is performed implicitly about the centroid of the image. While doing so, assign a value of 0 to unoccupied pixels. Let us denote the rotated version of J2 as J3.

(b) Our job will now be to align J3 with J1 keeping J1 fixed. To this end, we will do a brute-force search over θ ranging from -45 to +45 degrees in steps of 1 degree. For each θ, apply the rotation to J3 to create an intermediate image J4, and compute the following measures of dependence between J1 and J4:

- the normalized cross-correlation (NCC)
- the joint entropy (JE)
- a measure of dependence called quadratic mutual information (QMI) defined as $\sum_{i_1}\sum_{i_2}(p_{I_1I_2}(i_1,i_2) - p_{I_1}(i_1)p_{I_2}(i_2))^2$, where $p_{I_1I_2}(i_1,i_2)$ represents the normalized joint histogram (*i.e.*, joint pmf) of I_1 and I_2 ('normalized' means that the entries sum up to one). Here, the random variables I_1, I_2 denote the pixel intensities from the two images respectively. For computing the joint histogram, use a bin-width of 10 in both I_1 and I_2. For computing the marginal histograms p_{I_1} and p_{I_2}, you need to integrate the joint histogram along one of the two directions respectively. You should write your own joint histogram routine in MATLAB - do not use any inbuilt functions for it.

(c) Plot separate graphs of the values of NCC, JE, QMI versus θ and include them in the report PDF.

(d) Determine the optimal rotation between J3 and J1 using each of these three measures. What do you observe from the plots with regard to estimating the rotation? Explain in the report PDF.

(e) For the optimal rotation using JE, plot the joint histogram between J1 and J4 using the `imagesc` function in MATLAB along with `colorbar`. Include it in the report PDF.

(f) We have studied NCC and JE in class. What is the intuition regarding QMI? Explain in the report PDF. (Hint: When would random variables I_1 and I_2 be considered statistically independent?) [2+10+2+3+3+5=25 points]

5. Read in the images 'goi1.jpg' and 'goi2.jpg' from the homework folder using the MATLAB `imread` function and cast them as double. These are images of the Gateway of India acquired from two different viewpoints. As such, no motion model we have studied in class is really adequate for representing the motion between these images, but it turns out that an affine model is a reasonably good approximation, and you will see this. We will estimate the affine transformation between these two images in the following manner:

(a) Display both images using `imshow(im1)` and `imshow(im2)` in MATLAB. Use the `ginput` function of MATLAB to manually select (via an easy graphical user interface) and store $n = 12$ pairs of physically corresponding salient feature points from both the images. For this, you can do the following:
```
for i=1:12, figure(1); imshow(im1/255); [x1(i), y1(i)] = ginput(1);
figure(2); imshow(im2/255); [x2(i), y2(i)] = ginput(1);
```
Tips: Avoid selecting points which are visible in only one image. Try to select them as accurately as possible, but our procedure is robust to small sub-pixel errors. Make sure `x1(i),y1(i)` and `x2(i),y2(i)` are actually physically corresponding points. Salient feature points are typically points that represent corners of various structures.

(b) Write MATLAB code to determine the affine transformation which converts the first image ('goi1') into the second one ('goi2').

(c) Using nearest neighbor interpolation that you should implement yourself, warp the first image with the affine transformation matrix determined in the previous step, so that it is now better aligned with the second image. You are not allowed to use any implementation for this already available in MATLAB. Display all three images side by side in the report PDF.

(d) Repeat the previous step with bilinear interpolation that you should implement yourself. You are not allowed to use any implementation for this already available in MATLAB. Display all three images side by side in the report PDF.

(e) In the first step, suppose that the n points you chose in the first image happened to be collinear. Explain (in the report PDF) the effect on the estimation of the affine transformation matrix. [5+5+5+5+5=25 points]