Programming Assignment #1; Due Date:09/24/2019

Submit your code via CANVAS. Also submit a TYPED REPORT that contains, concise discussion on your observations from the results and any reasoning necessary to explain your approach. The report should contain, results that are arranged to show the input image and the output images, if and when applicable. Try to minimize your page usage by displaying at least 3 images per row on a page. All your image displays must have a caption.

1. (15 pts.)**Geometric transformations:** Write a MATLAB program that will rotate an image by 30 and 60 degrees respectively about the center and display the rotated image; You should NOT use the rotate function in MATLAB but write your own code for performing the rotations. You may validate your output via a comparison with the output of the MATLAB rotate function output. Perform and explain whatever necessary operations you have to in order to display a meaningful image. Show the results by displaying the input and output images using appropriate MATLAB display functions.

2. (25 pts.) **Planar projective distortion:** The goal here is to remove planar perspective distortion which was discussed in class at length. Take an image of a wall of the CISE building using your digital camera. Make sure that you take the image from a sufficiently oblique view point in order for the image to contain enough distortion. Develop MATLAB code to, (i) remove this planar perspective distortion using the linear algorithm (called DLT) discussed in class; (ii) have some fun with the perspective image of the wall of the CISE building by generating two new perspectives from the same camera center. Note that the mapping between two perspectives is a homography.

   For solving (i) above, you will need to identify four points in the image (see Figure 2.4 in Chapter 2 of Hartley & Zisserman posted on the class website), which you can do by using a utility called 'xv' on any linux/unix platform or you may use the code provided below. The following code segment will output the 2D coordinates of the clicked image points:

```
function XY2D=Get2DPoints(ImageFileName, NumberOfPoints)

XY2D=[];
[Img, Col]=imread(ImageFileName, 'jpg'); %%% assuming the imagefile is a jpg file.
image(Img); drawnow; hold on;

for i=1:NumberOfPoints
 [x, y]=ginput(1);
 v=[x;y];
 plot(x, y, 'r*');
 XY2D=[XY2D v];
end
return;
```

   You will submit your code for both (i) and (ii). Further, for (i), you will turn in the input image that depicts the perspective distortion and the output image after distortion removal. For (ii), you will turn in the newly generated perspectives. Note that, once you compute and apply the homography to the whole image, you may need to use some interpolation technique such as a bilinear interpolation scheme. You can find information online about bilinear interpolation.

3. (40 pts.) **Camera Calibration:** In this assignment, you will implement the camera calibration algorithm described in class (see slides). Recall that in order to calibrate a camera, you need a set of correspondences between 3D points and their 2D image points. The image provided for you in this assignment is shown in Figure 1 (see link below). In this scene, there are three mutually perpendicular planes, and each plane has been painted with a chess-board pattern such that *each side of the square is one inch long.* The center of the world frame is located at the intersection of the three planes, and the three coordinate axes can be inferred from the image using the labels assigned to the planes shown in Figure 1. The image can be downloaded at `http://www.cise.ufl.edu/class/cap5416fa19/images/Image.JPG`.

   For this assignment, you need to identify n number of scene points and figure out their 3D coordinates using the world frame defined above. You are free to choose the n scene points except those nine points marked as test points shown in Figure 1. Of course, you will also need to determine the 2D image coordinates of the chosen points. For this latter part, you can use 'xv' or a quick solution would be to write a simple graphical user interface (GUI) that will allow you to figure out the image coordinates of a point on the image by clicking it.

   What to turn in

   You should try three values of $n = 6, 12, 18$ for the number of correspondences used in calibration. For each value of n, you will estimate one set of camera parameters in the form of a 3×4 camera $\mathcal{M}$. To estimate the camera internal and external parameters you will need to use an RQ factorization. Given any $(3,3)$ matrix A, you can achieve the RQ factorization using Givens rotations by doing $AQ_xQ_yQ_z = R$, where Q_x, Q_y, Q_z are $(3,3)$ rotation matrices about the x, y and z- axis respectively. Hence $A = RQ$, where $Q = {Q_x}^t {Q_y}^t {Q_z}^t$.

   In this assignment, the accuracy of the calibration will be evaluated by the nine test points. The idea is to compute the *average projection error*: for each test point $\mathbf{P}_i$, let's denote (x_i, y_i) its corresponding point on the image. By computing $\mathcal{M}_n\mathbf{P}_i$, you will obtain the image point (x_i', y_i') of $\mathbf{P}_i$ according to the camera matrix $\mathcal{M}$. If $\mathcal{M}_n$ models exactly the image projection process, then $x_i = x_i', y_i = y_i'$, which is generally not the case. Therefore, it is sensible to define the error (for the point $\mathbf{P}_i$) as

$$\mathbf{E}(\mathbf{P}_i) = \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2}.$$

   And for the set of nine test points $\mathbf{P}_1, \cdots, \mathbf{P}_9$, one way to evaluate the accuracy of the calibration is to calculate the average projection error:

$$\mathbf{E} = \frac{\mathbf{E}(\mathbf{P}_1) + \ldots + \mathbf{E}(\mathbf{P}_9)}{9}$$

   The following items should be turned in (along with a printout of your MATLAB code): for each n,

   (a) The eleven camera parameters that include the five intrinsic parameters $(\alpha_x, \alpha_y, s, t_x, t_y)$ and the extrinsic parameters namely, the rotation matrix and translation vector.

   (b) The average projection error, $\mathbf{E}$.

   Suggestions

   The MATLABfunction **eig** computes the eigenvalues and eigenvectors of a square matrix. You can use the MATLAB function **help** to learn more about the inputs and outputs of **eig** (and other functions). For code to output coordinates of clicked image points, see problem # 2 above.

4. **Fourier Transforms (20 pts):** You are given two images A and B, of yourself and your friend. *You can capture these images using your cell phone camera.* (i) Write a MATLAB program to compute the Fourier transform magnitude only and phase only images of A and B respectively. When creating magnitude only image, you zero out the phase and when creating phase only image, you set the magnitude to unity. Display your results as follows - top left: image A, top middle: magnitude-only reconstruction, top right: phase-only reconstruction. Similar arrangement for image B. *The term reconstruction here refers to the result of applying the inverse Fourier transform.* (ii) (a) Form a new image by compositing the magnitude of the FT of A and phase of FT of B and taking the IFT of this composite to obtain the reconstruction. (b) Now, repeat this experiment by interchanging the roles of A and B. Display your results as follows - top left: original image A, top right: reconstructed image from (ii)(a) and bottom left: original image B, bottom right: reconstructed image from (ii)(b).

   Finally, write a succinct report of your observations on the quality of reconstruction from magnitude-only and phase-only reconstructions as well as the two reconstructions from the compositing operations. Give precise reasoning justifying the structural detail observed in each of the reconstructions, thereby signifying the relative importance or the lack thereof of magnitude and phase components of the FT.