Assignment 2: CS 763, Computer Vision

Due: 8th February before 11:00 pm

Remember the honor code while submitting this (and every other) assignment. All members of the group should work on and understand all parts of the assignment. We will adopt a zero-tolerance policy against any violation.

Submission instructions: You should ideally type out all the answers in Word (with the equation editor) or using Latex. In either case, prepare a pdf file. For assignment submission, follow the instructions for arrangement of folders and subfolders as given in `https://drive.google.com/open?id=1xFMfAqbl0QE5tQqpJIyn_uwY_AEPVUAI`. Create a single zip or rar file obeying the aforementioned structure and name it as follows: A2-IdNumberOfFirstStudent-IdNumberOfSecondStudent-IdNumberOfThirdStudent.zip. (If you are doing the assignment alone, the name of the zip file is A2-IdNumber.zip). Upload the file on moodle BEFORE 11:00 pm on 8th February. Late assignments will be assessed a penalty of 50% per day late. Note that only one student per group should upload their work on moodle. Please preserve a copy of all your work until the end of the semester. If you have difficulties, please do not hesitate to seek help from me.

1. In this question, you will calibrate your own camera using a checkerboard object. Print two (or three) copies of the checkerboard image (checkerbox.jpg) provided in the input folder and stick them to two (or three) orthogonal planes (wall corner). Click a picture of the checkerboard from your phone. Camera calibration requires 2D and 3D correspondences. Create a dataset that contains XYZ coordinates of N points marked out on the wall checkerboard and also the XY coordinates of the corresponding points on the image. Now, write a MATLAB program that estimates the 3×4 projection matrix P and then decompose it into the intrinsics and extrinsics. Your code should follow the following steps: **[5+15+5+5 points]**

   - Normalize the data such that the centroid of 2D and 3D points are at origin and the average Euclidean distance of 2D and 3D points from the origin is $\sqrt{2}$ and $\sqrt{3}$, respectively. Find the transformation matrices T and U that achieve this for 2D and 3D respectively, *i.e*, $\hat{\mathbf{x}} = T\mathbf{x}$ and $\hat{\mathbf{X}} = U\mathbf{X}$ where $\mathbf{x}$ and $\mathbf{X}$ are the unnormalized 2D and 3D points in homogeneous coordinates.
   - Estimate the normalized projection matrix $\hat{\mathbf{P}}$ using the DLT method as discussed in the class. Denormalize the projection matrix $\hat{\mathbf{P}}$ ($\mathbf{P} = T^{-1}\hat{\mathbf{P}}U$).
   - Decompose the projection matrix $\mathbf{P} = K[R| - RX_0]$ into intrinsic matrix K, rotation matrix R and the camera center X_o. K and R can be estimated using RQ decomposition.
   - Verify that the projection matrix is correctly estimated by computing the RMSE between the 2D points marked by you and the estimated 2D projections of the marked 3D points. Visualize the points on the image and include them in the report. Also mention why it is a good idea to normalize the points before performing DLT.

2. Consider the image wembley.jpeg in the input folder. It is an image of the famous Wembley stadium captured during a football match. Your task is to find the length and width of the playing area using your knowledge about Homography Transform. The only information you can use is that the dimension of the outer Dee (or box) is always $18yd \times 44yd$. For doing this, you will have to write a function `homography()` that takes as input two point sets corresponding to two images and returns the estimated homography transform, $\mathbf{H}$. Include the code/calculations along with the estimated dimensions in your submission. **[10 points]**

3. Consider the question above. This time, estimate the dimensions using known properties of pinhole projection without using homography transform. Again, you should only exploit the information about the dimensions

of Dee in this question as well. Explain your method in the report and include the code/calculations with your submission. **[10 points]**

4. **Image Stitching**: In this question, you will stitch three images to create a panorama view. For doing this, use the `homography()` function you wrote for Question 2. First, extract matching keypoints for two images using either SIFT or SURF(available in MATLAB). Next, write a function `ransacHomography()` that takes as input two sets of keypoints and a threshold value to return the homography matrix after performing RANSAC on the input keypoints. Finally, stitch the images using the retrieved homographies and display a single image that contains a panorama of the three images. You are provided with four sets of test images in the input directory. Please ensure that you do not crop the images and a complete mosaic is visible. Sample submission output is shown in Fig. 1. Finally, capture images of a planar scene with your phone/camera and test your code on them. Ideally, you should keep some non-overlapping areas to make it more interesting. **[20 + 5]**

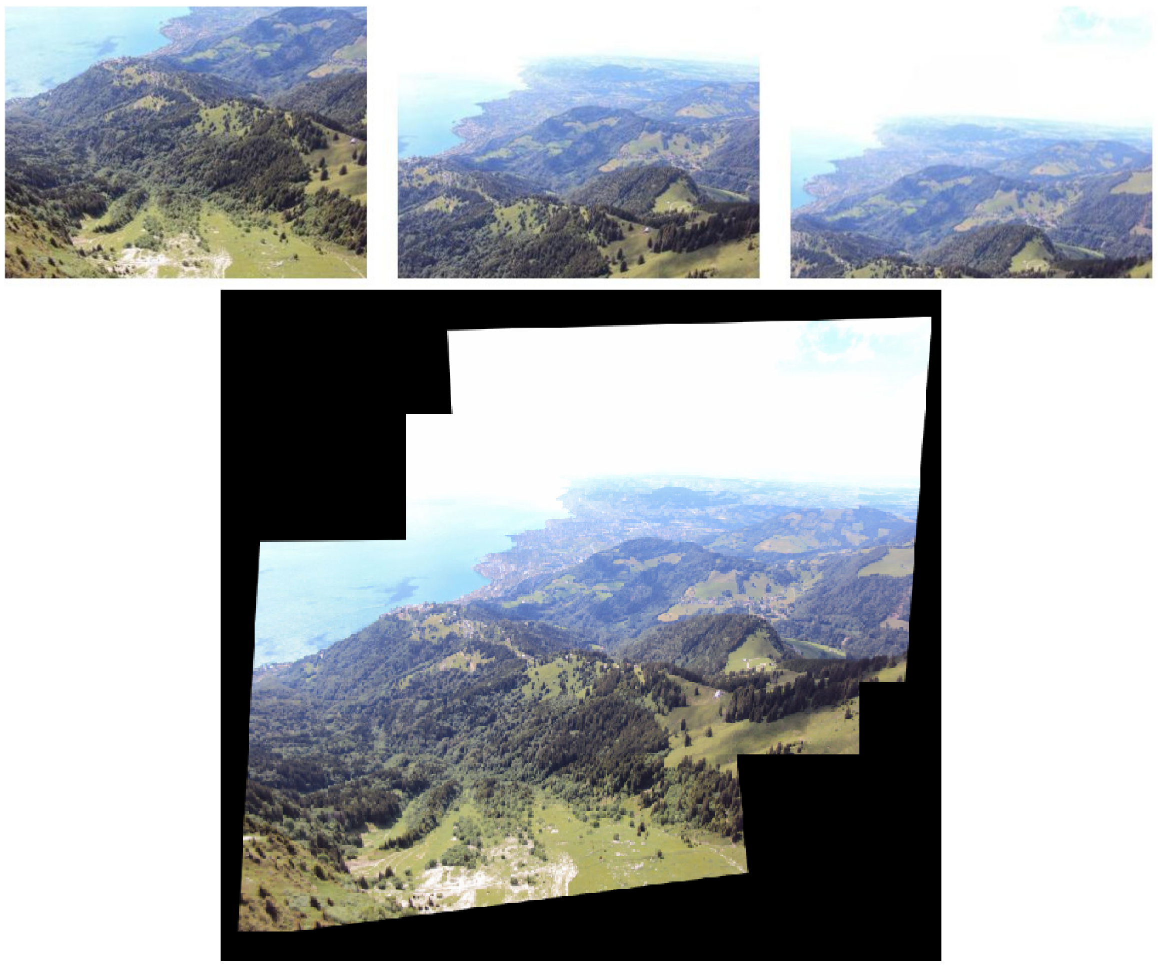

Figure 1: Top row: input images. Bottom row: Stitched image

5. In this task, we will register two pairs of images with each other: (1) The famous barbara image (regarded as a fixed image) to be registered with its negative (regarded the moving image), and (2) a flash image (regarded as a fixed image) and a no-flash image (regarded as the moving image) of a scene. We will use the joint entropy criterion we studied in class as the objective function to be minimized for alignment. Download all required images from the homework folder. Convert all images to gray-scale (if they are in color). Note that the flash image and the no-flash image have different image intensities at many places, and the no-flash image is distinctly noisier. In the beginning you may want to either downsample or work with smaller portions of the flash and no-flash images.

For each of the two cases, rotate the moving image counter-clockwise by 23.5 degrees, translate it by -3 pixels in the X direction, and add uniform random noise in the range [0,8] (on a 0-255 scale). Note that the rotation must be applied about the center of the image. Set negative-valued pixels to 0 and pixels with value more than 255 to 255. Now perform a brute-force search to find the angle θ and translation t_x to optimally align the modified moving image with the fixed image (in each case), so as to minimize the joint entropy. The range for θ should be between -60 and +60 in steps of 1 degree, and the range for t_x should be between -12 and +12 in steps of 1. Compute the joint entropy using a bin-size of 10 for both intensities. Plot the joint entropy as a function of θ and t_x using the surf and imshow commands of MATLAB. Comment on the difference (if any) between the quality of alignment for the first and second pair of images.

Also, determine a scenario (for the first pair of images) where the images are obviously misaligned but the joint entropy is (falsely and undesirably) lower than the 'true' minimum. Again, display the joint entropy as mentioned before. Include all plots in your report.
Finally, perform the same image alignment **[25 points]**