Introduction to Computer Vision and Image Processing
CSE 473/573 Project 2

Due Date: Oct. 23, (Wed.) 3 P.M., 2019

1. RANSAC Algorithm (5 points)

The goal of this task is to fit a line to a given set of points using RANSAC algorithm, and output the names of inlier points and outlier points for the line. Specifically, there are in total 8 points, and you need to find a line to best fit these points. After finding this line, you can use the distance threshold t to determine which points are inliers and which points are outliers. That is, a point is inlier if the perpendicular distance of this point to the fit line is smaller than the threshold t; otherwise, it belongs to outliers. It is recommended to record the two initial points for each iteration, such that you will not start from this two points in next iteration. The maximum iterations is $k = 100$, and the program must be finished within 1 minute; **otherwise, your grades will be low**. There is only one answer, as you can traverse all cases within 100 iterations as long as you do not repeat to select the initial two points. You are required to implement all the operations in "solution" function. The output should be a txt file which contains the names of inlier points and outlier points in the first row and second row, respectively. The specific format of the output has been clearly explained in the code.

2. Image Stitching (5 points)

The goal of this task is to stitch two images together to construct a panorama image. There is one solution for your reference. First, you need to find keypoints (points of interest) in the given images using corner detector, e.g., Harris detector. Then, you can use SIFT or other feature descriptors to extract features for these keypoints. Next, you should match the keypoints between two images by comparing their feature distance. After having the matched point pairs, you are able to compute the homography matrix using RANSAC algorithm. Finally, you can use the homography matrix to stitch the two given images. Besides the above method, you can also use any other approach to achieve the image stitching. Following is an example of image stitching: (Figure 3 is the expected result, and Figure 4 is the failed case.)

Figure 1: Left Image

Figure 2: Right Image

Figure 3: Successful example

Figure 4: Failed example

3. Guidelines

- For **ALL** students whose code raise "RuntimeError", your grades will be 0.
- Compress the two python files, i.e., "task1.py", "task2.py", the two given images and the folder "results" into a zip file, name it as "UBID.zip" (replace "UBID" with your eight-digit UBID, e.g., 51399256) and upload it to UBLearns before the due date. The zip file you upload should not contain files other than the five aforementioned files.

- For task 1, **Do Not** import any library or APIs provided by numpy and opencv.
- For task 2, You can use APIs provided by numpy and opencv, **except for** APIs that have "stitch" or "Stitch"in their names, e.g., "cv2.Stitcher.create()". For opencv 3, you may encounter errors when using SIFT feature descriptor. It is recommended to install opencv-contrib-python by "pip install opencv-contrib-python" or other ways, and then use "cv2.xfeatures2d.SIFT_create()". These are common issues, please first try to solve them by yourself when you encounter these errors.
- For runtime: Both of the two tasks are finished within 5 seconds in our laptops. If your runtime is too long, please check your code or consider to reduce the complexity of your program.