ECE661: Homework 5
Fall 2018

Deadline : Thursday, October 11, 2018 , 1:30 pm

Turn in typed solutions via Blackboard. Additional instructions given at [1]

1 Introduction

This is a homework whose basic goal is for you to gain experience with automatic computation of a 3x3 homography and its subsequent refinement by a *nonlinear* least-squares algorithm like Levenberg-Marquardt.

Obviously, before you can invoke the *nonlinear* least-squares refinement of the homography values, you must invoke the RANSAC algorithm for outlier rejection and a *linear* least-squares algorithm for creating an initial estimate of the homography.

You will do the work described above in the context of what is known as image mosaicing, which consists of "stitching" together a sequence of overlapping photos of a scene to create a single *panoramic* photo. Note that there must exist significant overlap between the adjacent images in the sequence of photos used for mosaicing for the technique to work.

You will need to estimate the homographies between the successive images in the sequence using the interest points in the overlap regions for setting up the correspondences. It is important to realize that there is NO manual delineation of the overlap areas. If your RANSAC code is working well, the system will figure out automatically where the overlap areas are in two adjacent images. Let's say you have five images in a sequence and let's label them 1, 2, 3, 4, and 5. Your code will need to estimate the homographies automatically between 1 and 2, between 2 and 3, between 3 and 4, and finally, between 4 and 5. Subsequently, you can use these homographies to project all the images into a common frame of reference — perhaps the frame of reference of the middle photo.

2 Tasks

1. Collect a **minimum of 5** photos by standing in a fixed location and turning through some angle as you take each successive shot. Ensure that there is sufficient overlap between successive photos, since otherwise you will not be able to establish good correspondences between the successive images.

2. Extract interest points from all the images. For the interest points, you can use either SIFT, or SURF, or even Harris that you coded up for the previous homework.

3. Using a similarity threshold that you consider appropriate, construct the set of all possible correspondences $(\mathbf{x}, \mathbf{x}')$ for each pair of adjacent images in your photo sequence. As you know from Lecture 11, some of these correspondences will be inlier correspondences, whereas others will be outlier correspondences.

4. Focusing initially on just one pair of adjacent images and using all the correspondences collected for that pair, now write code for automatically estimating the homography between the two images. This will involve three steps: (1) Using the RANSAC algorithm to reject the outlier correspondences; (2) Using one of the Linear Least-Squares methods of Lecture 10 to optimally estimate the homography from just the inlier correspondences; and (3) Refining the homography calculated in the previous step with the Levenberg-Marquardt Nonlinear Least-Squares method presented in Lecture 12.

   Proceed to the next step only if your Automatic Homography Calculator (AuHoCa) is working well.

5. Apply your AuHoCa (Automatic Homography Calculator) to every pair of adjacent photos in your sequence.
6. Use the pairwise homographies calculated by your AuHoCa to project all the images into the coordinate frame of the central image in the sequence. [You will run into some interesting computational issues when you carry out these projections. When you project the pixels of an image A into the frame of reference of image B, the projected coordinates will not correspond to the pixel locations of the latter image. You can use bilinear interpolation for a weighted assignment of the projected pixel to its four immediate neighbors.]

2.1 NOTES:

You must write your own code for the Linear Least-Squares and the RANSAC algorithms. Implementing Linear Least-Squares is pretty trivial: For a system of homogeneous equations like $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{0}}$, all you have to do is to carry out an SVD of the matrix $\mathbf{A}$ to yield $U \cdot D \cdot V^T$ and retain from the V matrix the last eigenvector for the solution you want.

Implementing RANSAC is only marginally more difficult. All you have to do is to *randomly* select, say, five or six correspondences at a time from all the correspondences for a given image pair, construct a tentative homography from the selected correspondences, and use that homography to project all the domain-image interest points into the range image. Now count the number of inlier correspondences, that is, the number of correspondeces for which the projected points in the range-image are within δ-distance of the actual corresponding point. The tentative homography that yields the largest inlier set is the solution you want. After you have discovered the solution, refine it using *all* of the correspondences in the inlier set.

Regarding the Levenberg-Marquardt (LM) algorithm for the final nonlinear refinement of the homographies returned by the Linear Least-Squares step, you have a choice here: You can use an open-source implementation of LM; or, **if you really want to impress your instructor**, create your own implementation. If you go the latter route, you will find helpful Professor Kak's open-source implementation of LM [II].

Make sure you draw lines to indicate the selected correspondences.

3 Submission

1. Turn in a typed pdf of your report via Blackboard.
2. Your pdf must include
    - A good description on your implementation of RANSAC alogirthm, the least squares method to estimate homographies, and the Levenberg-Marquardt algorithm.
    - A description of the steps involved in your image mosaicing method, with relevant equations.
    - The extracted correspondences between sets of adjacent images.
    - The outliers and selected inliers for at least two pairs of images.
    - The final output mosaic.
    - The parameters that you chose for the experiments.
    - Your source code.

References
[I] `http://engineering.purdue.edu/RVL/ECE661_2018/`
[II] `https://pypi.org/project/NonlinearLeastSquares/1.1.1/`