ECE549 / CS543 Computer Vision: Assignment 3

Instructions

1. Assignment is due at **???**.
2. Course policies: http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/policies.html.
3. Starter code, and data for this assignment is available at http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp3.zip.
4. Submission instructions:
   (a) A single `.pdf` report that contains your work for Q1, Q2 and Q3. For Q1(a) you can present your responses in the space provided on page 2 (directly typing on PDF, or via LaTeX, or by hand-writing on a print out and scanning it). For rest of the assignment, your response should be electronic (no handwritten responses allowed). You should respond to the questions 1, 2, 3 individually and include images, plots and metrics as necessary. Your response in the PDF report should be self-contained. It should include all the output you want us to look at. You will not receive credit for any results you have obtained, but failed to include directly in the PDF report file. PDF file will need to be submitted to https://www.gradescope.com (Entry Code: **MVZDNW**), and you will need to tag your PDF with where your response to each of the question is.
   (b) You also need to submit code for all questions in the form of a single **`.zip`** file. Code will need to be submitted to compass2g.
   (c) The LaTeX source is available: http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp3-latex.zip.
   (d) We reserve the right to take off points for not following submission instructions.

Change log

v0	02/24/2020	Creation.

1. **Stitching pairs of images [30 pts][1].** The goal of this assignment is to implement homography to register pairs of images in lecture 12.

   Test Images: We are providing eight images (four along with sample outputs for reference) that you should report results on. See folders `stitch−data` and `stitch−ref−output` at http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp3.zip. Keep in mind, though, that your output may look different from the reference output depending on your threshold, range of scales, and other implementation details.

   Steps:

   (a) Load both images, convert to double and to grayscale.

   (b) Detect feature points in both images. We provide Harris detector code you can use. Alternatively, feel free to use the blob detector you wrote for Assignment 2.

   (c) Extract local neighborhoods around every keypoint in both images, and form descriptors simply by "flattening" the pixel values in each neighborhood to one-dimensional vectors. Experiment with different neighborhood sizes to see which one works the best. If you're using your Laplacian detector, use the detected feature scales to define the neighborhood scales. Optionally, feel free to experiment with SIFT descriptors. You can use the OpenCV library to extract keypoints through the function cv2.xfeatures2D.SIFT_create().detect and compute descripters through the function cv2.xfeatures2D.SIFT_create(). This tutorial provides details about using SIFT in OpenCV.

   (d) Compute distances between every descriptor in one image and every descriptor in the other image. In Python, you can use scipy.spatial.distance.cdist(X,Y,'sqeuclidean') for fast computation of Euclidean distance. If you are not using SIFT descriptors, you should experiment with computing normalized correlation, or Euclidean distance after normalizing all descriptors to have zero mean and unit standard deviation.

   (e) Select putative matches based on the matrix of pairwise descriptor distances obtained above. You can select all pairs whose descriptor distances are below a specified threshold, or select the top few hundred descriptor pairs with the smallest pairwise distances.

   (f) Implement RANSAC to estimate a homography mapping one image onto the other. Report the number of inliers and the average residual for the inliers (squared distance between the point coordinates in one image and the transformed coordinates of the matching point in the other image). Also, display the locations of inlier matches in both images.

   (g) Warp one image onto the other using the estimated transformation. In Python, use skimage.transform.ProjectiveTransform and skimage.transform.warp.

   (h) Create a new image big enough to hold the panorama and composite the two images into it. You can composite by averaging the pixel values where the two images overlap, or by using the pixel values from one of the images. Your result should look similar to the sample output. You should create color panoramas by applying the same compositing step to each of the color channels separately (for estimating the transformation, it is sufficient to use grayscale images).

   Tips and Details:

   - For RANSAC, a very simple implementation is sufficient. Use four matches to initialize the homography in each iteration. You should output a single transformation that gets the most inliers in the course of all the iterations. For the various RANSAC parameters (number of iterations, inlier threshold), play around with a few "reasonable" values and pick the ones that work best.
   - Homography fitting calls for homogeneous least squares. The solution to the homogeneous least squares system AX=0 is obtained from the SVD of A by the singular vector corresponding to the smallest singular value. In Python, U, s, V = numpy.linalg.svd(A) performs the singular value decomposition and V[len(V)-1] gives the smallest singular value.

   Extra Credits:

---

[1] Adapted from Lana Lazebnik.

(a) Extend your homography estimation to work on multiple images. You can use this `stitch−multiple` at the data folder, consisting of three sequences consisting of three images each. Alternatively, feel free to acquire your own images and stitch them.

(b) Experiment with registering very "difficult" image pairs or sequences – for instance, try to find a modern and a historical view of the same location to mimic the kinds of composites found here. Or try to find two views of the same location taken at different times of day, different times of year, etc. Another idea is to try to register images with a lot of repetition, or images separated by an extreme transformation (large rotation, scaling, etc.). To make stitching work for such challenging situations, you may need to experiment with alternative feature detectors and/or descriptors, as well as feature space outlier rejection techniques such as Lowe's ratio test.

(c) Implement bundle adjustment or global nonlinear optimization to simultaneously refine transformation parameters between all pairs of images.

(d) Learn about and experiment with image blending techniques and panorama mapping techniques (cylindrical or spherical).

Grading Checklist:

- Describe your solution, including any interesting parameters or implementation choices for feature extraction, putative matching, RANSAC, etc.
- For the image pair provided, report the number of homography inliers and the average residual for the inliers (squared distance between the point coordinates in one image and the transformed coordinates of the matching point in the other image). Also, display the locations of inlier matches in both images.
- Display the final result of your stitching.

2. **Fundamental Matrix Estimation, Camera Calibration, Triangulation [30 pts][2].** The goal of this assignment is to implement fundamental matrix estimation to register pairs of images, as well as attempt camera calibration, triangulation in lecture 13 and lecture 14.

Test Images: We are providing two image pairs (library and lab) that you should report results on. See folder `multiview_data` at http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp3.zip.

Steps:

(a) **Fundamental matrix estimation**. Load each image pair and matching points file using the provided sample code [???]. Add your own code to fit a fundamental matrix to the matching points and use the sample code to visualize the results. You need to implement and compare the normalized and the unnormalized algorithms. For each algorithm and each image pair, report your residual, or the mean squared distance in pixels between points in both images and the corresponding epipolar lines.

(b) **Camera calibration**. For the lab pair, calculate the camera projection matrices by using 2D matches in both views and 3-D point coordinates given in lab_3d.txt in the data file. Refer to relevant lecture for the calibration method. Once you have computed your projection matrices, you can evaluate them using this sample function, which will provide you the projected 2-D points and residual error. (Hint: For a quick check to make sure you are on the right track, empirically this residual error should be < 20 and the squared distance of your projected 2-D points from actual 2-D points should be < 4.) For the library pair, there are no ground truth 3D points. Instead, camera projection matrices are already provided in library1_camera.txt and library2_camera.txt.

(c) **Calculate the camera centers** using the estimated or provided projection matrices for both pairs.

(d) **Triangulation**. Use linear least squares to triangulate the 3D position of each matching pair of 2D points given the two camera projection matrices. As a sanity check, your triangulated 3D points for the lab pair should match very closely the originally provided 3D points in lab_3d.txt. For each pair, display the two camera centers and reconstructed points in 3D. Also report the residuals between the observed 2D points and the projected 3D points in the two images.

---

[2] Adapted from Lana Lazebnik and James Hays.

Tips and Details:

- For fundamental matrix estimation, don't forget to enforce the rank-2 constraint. This can be done by taking the SVD of F, setting the smallest singular value to zero, and recomputing F.
- Lecture 13[???] shows two slightly different linear least squares setups for estimating the fundamental matrix (one involves a homogeneous system and one involves a non-homogeneous system). You may want to compare the two and determine which one is better in terms of numerics.
- Recall that the camera centers are given by the null spaces of the matrices. They can be found by taking the SVD of the camera matrix and taking the last column of V.
- You do not need the camera centers to solve the triangulation problem. They are used just for the visualization.

Extra Credits:

(a) Use your putative match generation and RANSAC code from Part 1 to estimate fundamental matrices without ground-truth matches. For this part, only use the normalized algorithm. Report the number of inliers and the average residual for the inliers. Compare the quality of the result with the one you get from ground-truth matches.

Grading Checklist:

- For both image pairs, for both unnormalized and normalized fundamental matrix estimation, display your result (points and epipolar lines) and report your residual.
- For the lab image pair, show your estimated 3x4 camera projection matrices. Report the residual between the projected and observed 2D points.
- For both image pairs, visualize 3D camera centers and triangulated 3D points.

3. **Single-View Geometry[30 pts][3].** The goal of this assignment is to implement algorithms for single-view in lecture 15.

   Test Images: We are providing one image that you should report results on. See folder `singleview_data` at http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp3.zip.

   Steps:

   (a) You will be working with the above image of the North Quad (save it to get the full-resolution version). First, you need to estimate the three major orthogonal vanishing points. Use at least three manually selected lines to solve for each vanishing point. This Python code provides an interface for selecting and drawing the lines, but the code for computing the vanishing point needs to be inserted. For details on estimating vanishing points, see Derek Hoiem's book chapter (section 4). You should also refer to this chapter and the single-view metrology lecture for details on the subsequent steps. In your report, you should:
   - Plot the VPs and the lines used to estimate them on the image plane using the provided code.
   - Specify the VP pixel coordinates.
   - Plot the ground horizon line and specify its parameters in the form $a * x + b * y + c = 0$. Normalize the parameters so that: $a^2 + b^2 = 1$.

   (b) Using the fact that the vanishing directions are orthogonal, solve for the focal length and optical center (principal point) of the camera. Show all your work.

   (c) Compute the rotation matrix for the camera, setting the vertical vanishing point as the Y-direction, the right-most vanishing point as the X-direction, and the left-most vanishing point as the Z-direction.

   (d) Estimate the heights of (a) the CSL building, (b) the spike statue, and (c) the lamp posts assuming that the person nearest to the spike is 5ft 6in tall. In the report, show all the lines and measurements used to perform the calculation. How do the answers change if you assume the person is 6ft tall?

---

[3] Adapted from Lana Lazebnik.

Extra Credits:

(a) Perform additional measurements on the image: which of the people visible are the tallest? What are the heights of the windows? etc.

(b) Compute and display rectified views of the ground plane and the facades of the CSL building.

(c) Attempt to fit lines to the image and estimate vanishing points automatically either using your own code or code downloaded from the web.

(d) Find or take other images with three prominently visible orthogonal vanishing points and demonstrate varions measurements on those images.

Grading Checklist:

- Derivation and output of every step mentioned above.