ECE549 / CS543 Computer Vision: Assignment 3

Instructions

1. Assignment is due at **11:59:59PM on Monday April 6, 2020**.
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 Q3(b), Q3(e) you can present your responses 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	03/20/2020	Creation.

1. **Stitching pairs of images [30 pts][1].** We will estimate homography transforms to register and stitch image pairs as discussed in lecture 12.

   Test Images: We are providing a image pairs that you should stitch together, and some starter code. See `MP3_Q1_starter.ipynb` and folder `data/Q1/stitch`. We have also provided sample output, though, keep in mind that your output may look different from the reference output depending on implementation details.

   (a) Detect feature points in both images and extract descriptor of every keypoint in both images. We will use SIFT descriptors from OpenCV library (<=3.4.2). You can refer to this tutorial for more details about using SIFT in OpenCV.

   (b) Compute distances between every descriptor in one image and every descriptor in the other image. We will use `scipy.spatial.distance.cdist(X,Y,'sqeuclidean')`.

   (c) **Putative Matches [5 pts].** 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. In your report, display the putative matches overlaid on the image pairs.

   (d) **Homography Estimation and RANSAC [20 pts].** Implement RANSAC to estimate a homography mapping one image onto the other. Describe the various implementation details, and report all the hyper-parameters, the number of inliers, and the average residual for the inliers (mean 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.
   Hints: 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, i.e., $A = USV$ (and not USV^T) and `V[-1,:]` gives the right singular vector corresponding to the smallest singular value.

   (e) **Image Warping [5 pts].** Warp one image onto the other using the estimated transformation. 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). You may find `ProjectiveTransform` and `warp` functions in `skimage.transform` useful. Include the stitched image in your report.

   (f) **Extra Credit [5 pts].**
      - Extend your homography estimation to work on multiple images. `data/Q1/extra_credits` contains three sequences with three images each, that you could attempt to stitch together. Alternatively, feel free to acquire your own images and stitch them.
      - 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. Another idea is to try to register images with a lot of repetition, or images separated by an extreme transformation (large rotation, or scaling). To make stitching work for such challenging situations, you may need to experiment with alternative feature detectors or descriptors, as well as feature space outlier rejection techniques such as Lowe's ratio test.
      - Implement bundle adjustment or global nonlinear optimization to simultaneously refine transformation parameters between all pairs of images.
      - Learn about and experiment with image blending techniques and panorama mapping techniques (cylindrical or spherical).

[1] Adapted from Lana Lazebnik.

2. **Fundamental Matrix Estimation, Camera Calibration, Triangulation [35 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 14, and lecture 15.

   Test Images: We are providing two image pairs (`library` and `lab` in `data/Q2/`), and some starter code in `MP3_Q2_starter.ipynb`, that we will work with.

   (a) **Fundamental Matrix Estimation [10 pts].** Given point matches, your task is to fit a fundamental matrix. You will implement and compare the normalized and the unnormalized algorithms as discussed in lecture 15. For each algorithm and each image pair, report the estimated fundamental matrix, the residual (mean squared distance (in pixels) between points in both images and the corresponding epipolar line), and visualization of epipolar lines and corresponding points. We have provided some starter code.
   Note: 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. You should use the least square setup that involves solving a homogeneous system.

   (b) **Camera Calibration [10 pts].** For the `lab` pair, calculate the camera projection matrices by using 2D matches in both views and 3D point coordinates in `lab_3d.txt`. Once you have computed your projection matrices, you can evaluate them using the provided evaluation function (`evaluate_points`). The function outputs the projected 2-D points and residual error. Report the estimated 3×4 camera projection matrices (for each image), and residual error.
   Hint: The residual error should be < 20 and the squared distance of the projected 2D points from actual 2D 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) **Camera Centers [5 pts].** Calculate the camera centers using the estimated or provided projection matrices for both pairs. Report the 3D locations of all the cameras in your report. **Hint:** Recall that the camera center is given by the null space of the camera matrix.

   (d) **Triangulation [10 pts].** Use linear least squares to triangulate the 3D position of each matching pair of 2D points using 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. Include snapshots of this visualization in your report. Also report the residuals between the observed 2D points and the projected 3D points in the two images, for each pair. **Note:** You do not need the camera centers to solve the triangulation problem. They are used just for the visualization.

   (e) **Extra Credits [5 pts].** Use your putative match generation and RANSAC code from the last question 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.

[2] Adapted from Lana Lazebnik and James Hays.

3. **Single-View Geometry [40 pts]**[3]**.** The goal of this assignment is to implement single-view camera calibration in lecture 14.

   Test Images: We are providing starter code (`MP3_Q3_starter.ipynb`), and an image (`data/part3/eceb.png`) that we will work with.

   (a) **Vanishing Points [5 pts].** Estimate the three major orthogonal vanishing points. Assume that the vertical vanishing point is for the Y-direction, the horizontal vanishing point to the left is the Z-direction, and the horizontal vanishing point to the right is the X-direction. Starter code provides an interface to mark out lines. Mark at least 3 lines in each orthogonal direction. Compute their intersection (the vanishing point) using least squares. In your report, plot the VPs and the lines used to estimate them on the image plane (starter code has some plotting functions). Also report the pixel coordinates for the vanishing point.

   (b) **Horizon [5 pts].** Estimate the horizon, using the vanishing points for the horizontal lines. Plot the ground horizon line on the image, and report its parameters in the form $ax + by + c = 0$, where parameters are normalized such that: $a^2 + b^2 = 1$.

   (c) **Camera Calibration [15 pts].** 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, and report the values you find.

   (d) **Rotation Matrix [5 pts].** Compute and report the rotation matrix for the camera.

   (e) **Fronto-parallel Warps [10 pts].** Use the estimated camera parameters and the camera rotations to compute warps that show the XY, YZ and ZX planes (corresponding to the front, side and top views of the ECE building). To do this: compute the appropriate rotation that will transform the world coordinates such that the axis of projection becomes the world Z, X and Y axis respectively. Use this rotation to estimate a homography that will be used to compute the output view. Apply the homography to generate the 3 fronto-parallel views and save them. Show your work (how you obtain the necessary rotation matrices, and homography transforms), and the generated views.

   (f) **Extra Credits [5 pts].** Attempt to fit lines to the image and estimate vanishing points automatically either using your own code or code downloaded from the web.

[3] Adapted from Lana Lazebnik.