Due on Monday, October 7, 2019 at 11:59 pm

Answer all problems following the instructions. Please include the written and programming portions in a zipped folder titled "Lastname_firstname_hw2.zip"

1 Written Problems

Problem 1. *(Homogenous Coordinates, Points, and Lines*

1. Show that the line $y = mx + b$ can be written in homogenous coordinates as the following: $\mathbf{l}^T\mathbf{x} = 0$, and show the parameters for homogenous line $\mathbf{l}$ and homogenous point $\mathbf{x}$.
2. Give at least two homogeneous representations (x, y, z) for the point $(3, 5)$ on the 2D plane, one with $z > 0$ and one with $z < 0$.
3. Take two homogenous lines, $\mathbf{l}_1, \mathbf{l}_2$. Show that $\mathbf{l}_1 \times \mathbf{l}_2$ is the point of intersection between the two lines.
4. Consider the two lines $x + y - 5 = 0$ and $4x - 5y + 7 = 0$. Compute the point of intersection in homogenous space, and then convert it into standard Cartesian space.
5. Take a line with standard form $ax + by + c = 0$. Write the general equation for the intersection point for any parallel line to this line. Where does this point lie in Cartesian space?
6. Consider two homogenous points $\mathbf{x}_1, \mathbf{x}_2$. Write (and show justification) an expression for the homogenous line $\mathbf{l}$ that goes through both points.

Problem 2. *(2D transformations)*

1. Derive a single homogenous $3x3$ matrix for rotation of angle θ around a point (a, b). You must show steps to justify the derivation (not merely giving the answer).
2. A square has vertices $p_1 = (1, 1)$, $p_2 = (2, 1)$, $p_3 = (2, 2)$ and $p_4 = (1, 2)$. Calculate the new vertices of the square after a rotation about p_2 through an angle of 45 degrees.
3. Construct the 2D transformation (one matrix) to reflect across an arbitrary line $ax+by+c = 0$. [Hint: First consider $b \neq 0$. One should translate and rotate to align the line with the x-axis, reflect across the x-axis, and then restore the line back to its normal place. The final matrix should be only in terms of a, b, c and θ as the angle of the line. Show that this matrix also satisfies the special case when $b = 0$ as well.

Problem 3. *(Affine + second order warp)* Consider an image transform that maps (x, y) to (x', y') via the following equations:

$$x' = ax + by + t_x + \alpha x^2 + \beta y^2, y' = cx + dy + t_y + \gamma x^2 + \theta y^2$$

.

Given two images with point-correspondences (x, y) to (x', y'), derive a technique to solve all parameters in closed-form. What is the minimal number of points needed to solve this? Note: you don't have to actually solve this problem, just setup the way to solve it.

2 Coding Assignments

Problem 4. *2D Transformations on Images*

1. Construct an image of 300×300 pixels, with a white irregular quadrilateral (i.e. four unique corners) approximately size 50×50 pixels is centered in the picture on a black background. Perform the following operations: (1) Translate the image by $(30, 100)$ and (2) Rotate the image by 45 degrees using OpenCV commands about the original center. Display the rotated quadrilateral.
2. Come up with a feature detector and descriptor to accurately match the four corners of the quadrilateral in the two pixels. **You cannot use any built-in feature detectors/descriptors to do this step.** This could include implementing Harris Corners from scratch and coming up with your own descriptor. However, there might be simpler features that will give you the matching you need.
3. Once you have established point correspondences with your feature detector/descriptor, pick a subset of them (how many do you need?), and then construct a linear least squares optimization to solve for θ and t_x, t_y. You should recover back the translation and rotation in the first part.

Problem 5. *Image Stitching*
See next page

Assignment: Image Stitching

Adapted from Jean Ponce, Ivan Laptev, Cordelia Schmid and Josef Sivic

Due on **October 7th**, 2018

1 Introduction

The goal of the assignment is to automatically stitch images acquired by a panning camera into a mosaic as illustrated in Figures 1 and 2 below

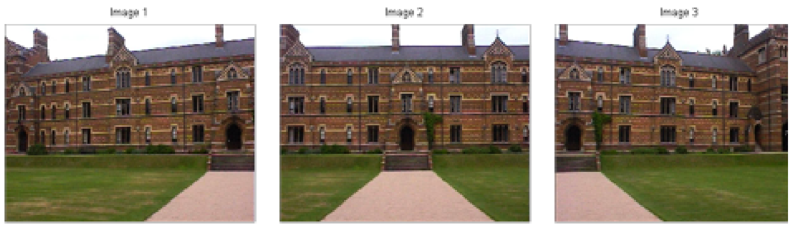

Figure 1: Three images acquired by a panning camera.

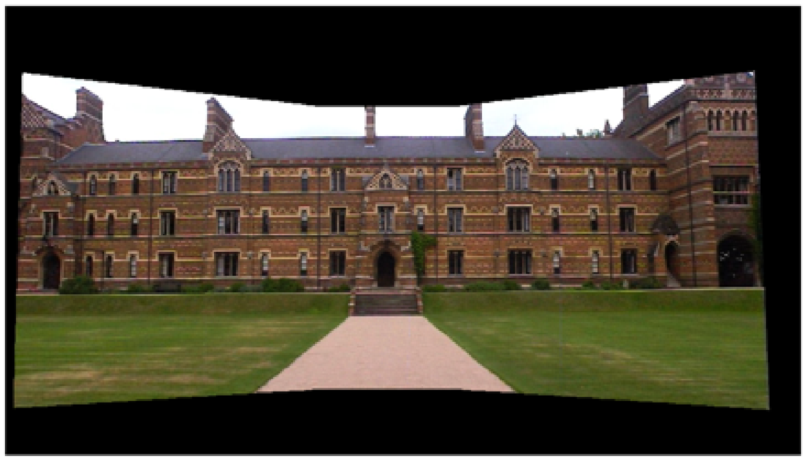

Figure 2: Images stitched to a mosaic.

2 Algorithm outline

1. Choose one image as the reference frame.
2. Estimate homography between each of the remaining images and the reference image. To estimate homography between two images, we use the following procedure:
   (a) Detect local features in each image (using SIFT).
   (b) Extract feature descriptor for each feature point.
   (c) Match feature descriptors between two images.
   (d) Robustly estimate homography using RANSAC.
3. Warp each image into the reference frame and composite warped images into a single mosaic.

3 Tips and detailed description of the algorithm

The algorithm will be described on the example of stitching images shown in figure 1.

1. Choosing the reference image. Choosing the middle image of the sequence as the reference frame is the preferred option, as it will result in mosaics with less distortion. However, you can choose any image of the sequence as the reference frame. If you choose image 1 or 3 as the reference frame, directly estimating homography between images 1 and 3, will be difficult as they have very small overlap. To deal with this issue, you might have to "chain" homographies, e.g. $H_{13} = H_{12} * H_{23}$.

2. Estimating homography.
   (a) Match feature descriptors between two images. That is, you will need to find pairs of features that look similar and are thus likely to be in correspondence. We will call this set of matched features "tentative" correspondences.

As the first step, use Euclidean distance to compute pairwise distances between the SIFT descriptors. Then, you need to perform "thresholding". For thresholding, use the simpler approach due to Lowe of thresholding on the ratio between the first and the second nearest neighbors. This is described in Section 5 in "Multi-Image Matching using Multi-Scale Oriented Patches" by Brown et al.[1], (see helpful resources below). Consult Figure 6b in the paper for picking the threshold. Note: You can ignore the fast indexing described in section 6 of the paper. You can visualize the tentative correspondences between two images by displaying the feature displacements. For example, to visualize tentative correspondences between image 1 and 2: (i) show image 1, (ii) show detected features in image 1 (display only region centers as points, do not worry about the regions' scale), (iii) show displacements between detected features in image 1 and matched features in image 2 by line segments. This is illustrated in Figure 3.

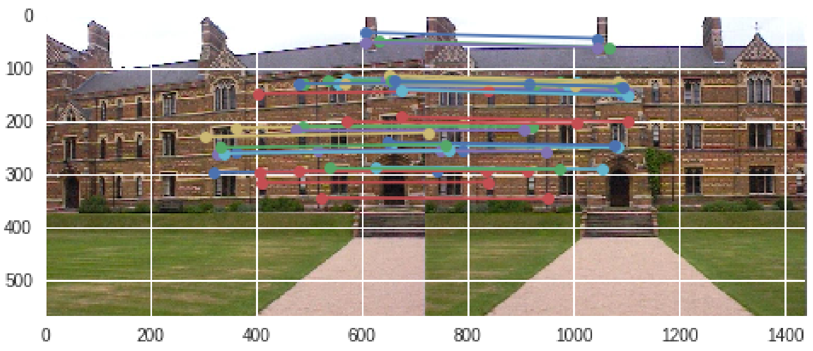

Figure 3: Correspondences between the inliers between left and center images found by RANSAC. Similarly you can show tentative correspondences.

(b) Robustly estimate homography using RANSAC. Use a sample of 4-points to compute each homography hypothesis. You will need to write a function of the form:

H = computeH(im1_pts, im2_pts)

where, again, im1_pts and im2_pts are $2 \times n$ matrices holding the (x,y) locations of $n(= 4)$ point correspondences from the two images and H is the recovered 3×3 homography matrix. In order to compute the entries in the matrix H, you will need to set up a linear system of n equations (i.e. a matrix equation of the form $Ah = 0$ where h is a vector holding the 9 unknown entries of H). The solution to the homogeneous least squares system $Ah = 0$ is obtained from the SVD of A by the right singular vector corresponding to the smallest singular value. For more details on homography estimation from point correspondences see a note written by David Kriegman [2].

For RANSAC, a very simple implementation performing a fixed number of sampling iterations is sufficient. 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. You should display the set of inliers as illustrated in figure 3. Finally, after you find the set of inliers using RANSAC, don't forget to re-estimate the homography from all inliers.

3. Warping and compositing. Warp each image into the reference frame using the estimated homographies and composite warped images into a single mosaic. In Python, you can make use of the OpenCV function cv2.warpPerspective.

4 Test Images

Implement the stitching algorithm for the three images given to you: keble_a.jpg, keble_b.jpg and keble_c.jpg, which are the left, center and right images respectively.

5 What to hand in

You should prepare a brief report including the following:

1. Show the final mosaic of the provided test images (similar to figure 2).
2. Show the visualization of the set of tentative matches and the set of inliers for one image pair (similar to figure 3).
3. The report be submitted on Canvas in .pdf and should not be inside a zipped folder. Also upload relevant codes to Canvas.

6 Helpful resources

1. M. Brown, R. Szeliski and S. Winder. Multi-Image Matching using Multi-Scale Oriented Patches. International Conference on Computer Vision and Pattern Recognition (CVPR2005), pages 510-517 `http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Papers/MOPS.pdf`

2. A note on computing homography by David Kriegman `http://cseweb.ucsd.edu/classes/wi07/cse252a/homography_estimation/homography_estimation.pdf`

3. A comprehensive treatment of homography estimation can be found in chapter 4 of Multiple View Geometry in Computer Vision by R. Hartley and A. Zisserman, Cambridge University Press, 2004.

4. Nice Slides by Alyosha Efros on mosaicing `http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Lectures/mosaic.ppt`