Ryerson University
CPS843/CP8307
Introduction to Computer Vision, Winter 2016
Assignment 2 - Model Fitting
Due: Saturday, March 26th, 11:59pm

All programming questions are to be completed in MATLAB. **CPS843 students can work in groups of two**, if they choose, and **CP8307 students are to work individually**. For those working in groups, only submit one copy and clearly indicate the group members.

Finally, do not share code with those outside your group or use code from the web (other than the code instructed to use). We will be checking for copying and reserve the right to give offenders (both copier and source) a zero on this assignment and pursue academic sanctions.

1 Least Squares Fitting of a Plane

1. **[5 points]** Write a MATLAB script that generates 500 data points for a plane, $z = \alpha x + \beta y + \gamma$, with additive Gaussian noise. (HINT: See the MATLAB example in the lecture slides for generating points on a line with additive noise.)

2. **[5 points]** Write a MATLAB script to estimate the parameters for the point set in Q1.1 based on all 500 data points using least-squares fitting. You will need to rewrite the equation of a plane as a non-homogeneous matrix equation, $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ is a vector of unknowns $(\alpha, \beta, \gamma)^\top$.

3. **[5 points]** Print to the screen the absolute error you found between each parameter of the ground truth and estimated planes?

2 RANSAC-based Image Stitching

The goal of this question is to write a simple mosaic/panorama application. A panorama is a wide-angle image constructed by compositing together a number of images with overlapping fields-of-views in a photographically plausible way.

Part A:

[30 points] In this part, you will write code to construct a mosaic based on an affine transformation. The images you will work with are shown in Fig. 1 and can be downloaded here: image 1 and image 2. An example result is shown in Fig. 2. An affine transformation is equivalent to the composed effects of translation, rotation, isotropic scaling and shear. Formally, an affine transformation of an image coordinate, $\mathbf{x}_1$, is given by the matrix equation $\mathbf{x_2} = \mathbf{Tx_1} + \mathbf{c}$. The unknowns are given by the elements in the 2×2 matrix $\mathbf{T}$ and the 2×1 vector $\mathbf{c}$. Rewrite the affine equation as a non-homogeneous matrix equation, $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ is a vector containing the the six unknown elements of $\mathbf{T}$ and $\mathbf{c}$. Since each point correspondence yields two equations and there are six unknowns, a minimum of three point correspondences are required. *(Real mosaics are constructed with a homographic image transformation. This transformation is more general than an affine transformation. Nonetheless, the same basic robust estimation architecture you are implementing in this part of the assignment applies when constructing a homography-based mosaic in Part B.*

Figure 1: Images used to create the Parliament panorama using an affine transformation.

1. **Preprocessing** Load both images, convert to single and to grayscale.

2. **Detect keypoints and extract descriptors** Compute image features in both images. The feature detector and descriptor you will be using is SIFT. Use the publicly available VLFeat library to compute SIFT features. The instructions for setting up VLFeat in MATLAB are available here: instructions. Also, check out the VLFeat SIFT demo. Compute SIFT feature descriptors using: `[f, d] = vl_sift(img);`

3. **Match features** Compute distances between every SIFT descriptor in one image and every descriptor in the other image. You can use this code for fast computation of (squared) Euclidean distance.

4. **Prune features** Select the closest 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.

5. **Robust transformation estimation** Implement RANSAC to estimate an affine transformation mapping one image to the other. Use the minimum number of pairwise matches to estimate the affine transformation. Since you are using the minimum number of pairwise points, the transformation can be estimated using an inverse transformation rather than least-squares. Inliers are defined as the number of transformed points from image 1 that lie within a user-defined radius of ρ pixels of their pair in image 2. You will need to experiment with the matching threshold, ρ, and the required number of RANSAC iterations. For randomly sampling matches, you can use the MATLAB functions `randperm` or `randsample` functions.

6. **Compute optimal transformation** Using **all the inliers** of the best transformation found using RANSAC (i.e., the one with the most inliers), compute the final transformation with least-squares.

7. **Create panorama** Using the final affine transformation recovered using RANSAC, generate the final mosaic and **display the color mosaic result to the screen**; your result should be similar to the result in Fig. 2. Warp one image onto the other using the estimated transformation. To do this, use MATLAB's `maketform` and `imtransform` functions. Create a new image big enough to hold the mosaic and composite the two images into it. You can create the mosaic by taking the pixel with the maximum value from each image. This tends to produce less artifacts than taking the average of warped images. To create a color mosaic, apply the same compositing step to each of the color channels separately.

Figure 2: An example (affine) panorama result using the Parliament images.

Part B:

[10 points] In this part, you will write code for constructing a panorama based on a homography transformation. The images you will work with are shown in Fig. 3 and can be downloaded here: image 1 and image 2. An example result is shown in Fig. 4. You should reuse the code from Part A but swap out the parts that refer to the affine transformation with the homography.

The minimum number of point correspondences to estimate a homography is four. Using a homography yields a set of homogeneous linear equations, $\mathbf{AX} = \mathbf{0}$. The solution to both the system of homogeneous equations consisting of four point correspondences and homogeneous least squares,

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \|\mathbf{AX}\| \tag{1}$$

subject to the constraint

$$\|\mathbf{X}\| = 1, \tag{2}$$

is obtained from the singular value decomposition (SVD) of $\mathbf{A}$ by the singular vector corresponding to the smallest singular value: `[U,S,V]=svd(A); X = V(:,end);`

1. Using your RANSAC-based homography code generate the mosaic using the Egerton Ryerson images and **display the color mosaic result to the screen**.

2. Run your code on an image pair of your own choosing and **display the color mosaic result to the screen**. Make sure the images you choose have significant overlap; otherwise, you will not be able to establish correspondences. Further, for a homography to be valid, the images can either be obtained from rotating in the same place OR from multiple vantage points if the scene is planar or approximately planar.

Figure 3: Images used to create the Egerton Ryerson statue panorama using a homography transformation.

Figure 4: An example (homography) panorama result using the Egerton Ryerson statue images.

Bonus:

1. **[10 points]** Experiment with combining image pairs where establishing correspondence is rendered difficult because of widely varying images sources. These images should have a Ryerson theme. Possible ideas include: (i) combining a modern and a historical view[1] of the same location, such as these ones and (ii) combining images taken from different times of day or different seasons. **Display the result to the screen and indicate this is for the bonus in the MATLAB Command Window**.

2. **[5 points]** Experiment with image blending techniques to remove salient seems between images; see Szeliski (the course textbook) Chapter 9. **Display the before and after blending color mosaic results to the screen and indicate in the MATLAB command window this is for the bonus.**

Submission details

Submit all MATLAB files and images required for the various parts of the assignment to run. Your submission should include a MATLAB script named `a2.m` for the grader to run. The script should break up the assignment with `pause()` commands, so that the grader can press "Enter" to step through all of your figures and written answers. **If your code does not run we cannot mark it.**

[1] The Ryerson archives may be able to assist with obtaining suitable historical imagery.