CIS581, Computer Vision
Project 3, Automatic 2D image mosaic
Due November 18, 4:30pm

Overview

This project is to be done individually.

This project focuses on image feature detection, feature matching and image mosaic techniques. The goal of this project is to create an image mosaic or stitching, which is a collection of small images which are aligned properly to create one larger image. We will follow technique outlined in the following papers, which are available on the course website:

“Multi-image Matching using Multi-scale image patches”, Brown, M.; Szeliski, R.; Winder, S. CVPR 2005

”Shape Matching and Object Recognition Using Shape Contexts”,S. Belongie, J. Malik, and J. Puzicha. PAMI 2002: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/

Function Specifications

This project requires you to write a number of functions. The specifications for each have been included on the Project page of the course website under the ”Project-3 Function Specifications” link. Please make sure your code follows those exactly.

Submission

Compress your files into a ZIP file named ”3_<penn_user_name>.zip”, and email it to cis581.Fall2014@gmail.com with the subject ”Project 3”. Please submit it using your

UPenn email address. The zip file should include:

- your .m files for the five (5) required Matlab functions (`anms.m`, `feat_desc.m`, `feat_match.m`, `ransac_est_homography.m`, `mymosaic.m`)
- any demo .m script(s) for reproducing/showing your results
- any additional .m files with helper functions for your code
  - this includes any third party code that you used, eg for harris corner detection
- the input images you used
- the resulting stitched image mosaic
- a PDF showing your results and describing any bonus features of your implementation. By results, we mean 1) the final stitched mosaic (which should already be included as a standalone image) and also some intermediary results showing 2) corner detection results of an image in red dots, 3) adaptive non-max suppression of the image (used to show corner detection results) in red dots, 4) final matching results (right before stiching) of all pairs of images used in mosaic in blue dots with outliers in red dots. If you used third party code, make it clear. Note that make PDF as simple as possible.

Task 1: CAPTURE IMAGES

For this project, you need to capture multiple images of a scene, which you will use to create image mosaic. In general, you should limit your camera motion to purely translational, or purely rotational (around the camera center).

Bonus points will be given for interesting selection of images.

Task 2: AUTOMATIC CORRESPONDENCES

You need to implement the following steps.

1. Detecting corner features in an image. Recommand to use Matlab `cornermetric` with proper arguments. But, you could follow method outlined in lecture note on Interest Point Detector to build your own image corner detector (for example derived from edge detector). You can probably find free “harris” corner detector on-line, and you are allowed to use them. Software links are provided on the course website.

2. (`anms.m`) Implement Adaptive Non-Maximal Suppression. The goal is to create an uniformly distributed points. See section 3 of the reference paper, as well as lecture notes.

3. (`feat_desc.m`) Extracting a Feature Descriptor for each feature point. You should use the subsampled image around each point feature. Dont worry about rotation-invariance just extract axis-aligned 8x8 patches. Note that it's extremely important to sample these patches from the larger 40x40 window to have a nice big blurred descriptor. Don't forget to bias/gain-normalize the descriptors.

   As a bonus, you can implement the Geometric Blur features.

4. (`feat_match.m`) Matching these feature descriptors between two images. Remember to use the trick to compare the best and second-best match (take their ratio).

TASK 3: RANSAC and IMAGE MOSAIC

Not all the matches computed in Task 2 will be correct. One way to remove incorrect matches is by implement the additional step of RANSAC (see lecture notes).

1. (`ransac_est_homography.m`) Use a robust method (RANSAC) to compute a homography. Use 4-point RANSAC as described in class to compute a robust homography estimate.

   Recall the RANSAC steps are:

   (a) Select four feature pairs (at random), p_i, p_i^1

   (b) Compute homography H (exact). Code is available for this on the Project page.

   (c) Compute inliers where $SSD(p_i^1, Hp_i) < thresh$

   (d) repeat step a-c

   (e) Keep largest set of inliers

   (f) Re-compute least-squares H estimate on all of the inliers

2. (`mymosaic.m`) Produce a mosaic by overlaying the pairwise aligned images to create the final mosaic image. Also, check check Matlab functions, e.g `imtransform` and `imwarp`. If you want to implement `imwarp` (or similar function) by yourself, you should apply bilinear interpolation when you copy pixel values. As a bonus, you can implement smooth image blending of the final mosaic.

FAQ

Q: Suppose I have the matrix cimg which contains corner strengths for each pixel. Do I need to set a threshold to get corners candidates? And then I only need to calculate radius for each corner candidate. Or I need to calculate radius for all pixels?

A: Ultimately you want corners that are at distinctive locations and well distributed throughout the image. If you don?t threshold at all then in areas that are relatively con- stant (say a plane white wall) you?ll get some corners due to noise that won?t end up being matches anyway since the location was essentially random. If you threshold too much ini- tially, however, you might only get corners in a few strong regions and cut out fainter but justified corners. Since for the panorama the overlap between images (and hence corners that can correctly match between two images) is a small region near one of the boundaries, you want to be sure that some corners respond in all portions of the image. The anms will then prune the corners that survived the thresholding to a manageable set that is also nicely distributed.

Q: I was having a problem getting the feat desc to account for the window size at the edges. I tried scaling the size of the window based on the location of the corner, but this would obviously not match with the $P_{(64 \times n)}$ matrix output.

A: You could try zero-padding those corners. Doing so makes it more likely that the corner won?t find a match or will find a close ssd score to other corners that got zero padded. You could also remove corners within 20 pixels of the edge from consideration so that all of your features can fit a window around them. Your region of overlap will probably be large enough to get plenty of matches past the 20 pixels on the boundary. Additionally assuming pixels near the boundary in one image aren?t also near the boundary in the other images (some instances where this assumption isn?t true) then they will likely be poor matches and get not make it past your threshold for matching anyway.

Q: In RANSAC, the part of the algorithm that calls for $SSD(p_i^1, Hp_i)$, I am assuming this is talking about the locations of the features and not the feature descriptors since the ransac_est_homography doesn?t see the descriptors. At any rate, how is this SSD computed and why don?t we just use euclidean distance?

A: Yes, its equivalent to euclidean distance. SSD on the coordinates would be the euclidean distance squared. There is no point taking a square root because we have to set an arbitrary threshold anyway.

Q: For the mymosaic function, are there any assumptions that can be made about the order of the images passed into the function? Meaning, can I assume that the order that the images are passed in is the order I should be stitching? Or do we need to compare every image with each other to find the ones that match best?

A: Assume that the order they come in is the order they stitch together.