Princeton University

COS429 Computer Vision

Problem Set 3: Stitching a Panorama

Figure 1: Princeton Panorama

Panoramic images are used to portray wide scenes that cannot be captured entirely within any single shot. In this problem you will develop your own automatic algorithm for creating panoramas that you can then show-off to your friends.

To create a panoramic image from two overlapping photos we need to map one image plane to the other. Since in general we do not know how to relate the position and orientation of the two camera views, we will use image features techniques discussed in class to recover the underlying mapping. First, we will identify key points in both images, and match between those points to find correspondences. From the correspondences we can compute a transformation that maps one set of points to the other. Once we have the transformation, we can render the images in common coordinate system, and merge them to generate the final result.

For key points and point matching, we will use the SURF descriptor, one of the most commonly used image descriptor in recent years in computational imaging. (The reason that we didn't use SIFT here is that MATLAB don't provide a SIFT implementation due to patent issues.)

For good quality panoramas, the transformation between the images need to be as accurate as possible. Yet, image descriptors and feature matching are both rather noisy processes: the descriptors are subject to image noise and compression artifacts, and not all presumed correspondences are true correspondences due to descriptor error and ambiguities in the matching (See Figure 4). Incorrect matches will insert error to our estimation and can adversely affect the result.

To make our algorithm robust, we will use the RANSAC algorithm discussed in class, a method for estimating a parametric model from noisy observations. You can refer to the lecture notes and Szeliski's book for details on the algorithm.

Here are the steps for you to create a panoramic image by using your own implementation (and of course, some handy MATLAB functions)!

Figure 2: Input Images

Problem 1 *Preprocessing*

Most feature descriptors in computer vision only work with gray scale images. Therefore, we need to convert color image to gray scale for feature extraction. To do this, you can use the MATLAB function `rgb2gray`. Copy the lines of code you wrote in the report as well.

Problem 2 *Detecting Keypoints*

Next, we need to detect key points in order to match two images. Run MATLAB function `detectSURFFeatures` on the two images to detect SURF features and find corresponding key points. Copy the lines of code you wrote in the report as well (separate the code into answers for different questions).

Problem 3 *Extracting Descriptors*

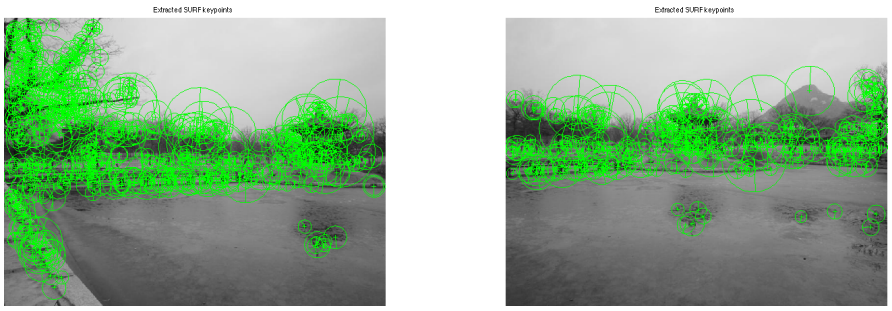

Figure 3: SURF Descriptors

Use the MATLAB function `extractFeatures` to extract feature vectors (descriptors) on each keypoint. Visualize corresponding feature points and add the figure to your report. Visualize the extraction results and include a figure to your report. Copy the lines of code you wrote in the report as well.

Problem 4 *Matching Features*

Next we need to look at pairs of features, compute the distance between them then find the matching. You can use the MATLAB function `matchFeatures` to achieve this goal. Visualize

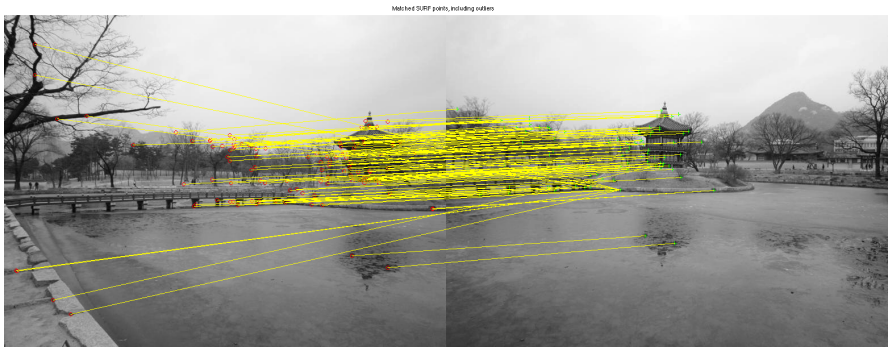

Figure 4: Initial matchings with errors

the matching results and include a figure to your report. Copy the lines of code you wrote in the report as well.

Problem 5 *RANSAC to Estimate Homography*

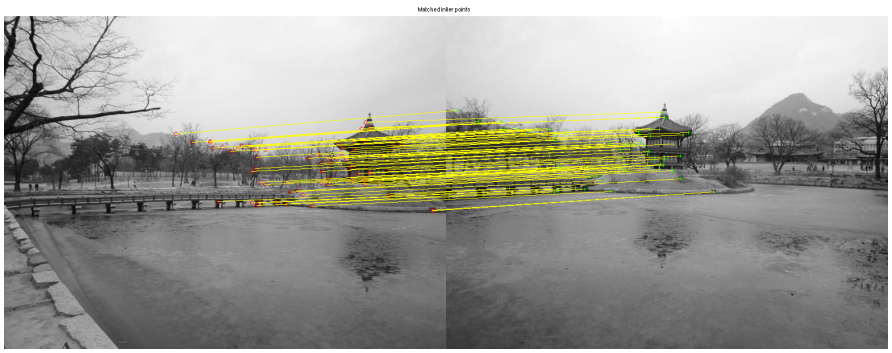

Figure 5: Inliers of the homography transformation

Next, we will use RANSAC to exclude the outliers and compute the homography. Use MATLAB function `estimateGeometricTransform` to estimate the homography. Visualize the matching results and include a figure to your report. Copy the lines of code you wrote in the report as well.

Problem 6 *Stitching Panorama*

Once you have the Homography transformation matrix, you will need to warp images and composite two images to make a panorama. You should map the pixels in the warped image to pixels in the input image so that you don't end up with holes in your image. We have provided the code for you to warp and paste the first image. Read the code and write similar codes to warp and paste the second image in order to produce the panoramic image. Use Matlab function `imwarp`, `vision.AlphaBlender`, and `step` to overlay the second image on top of the first one. You don't need to take care of blending here. Add the resulting panorama to your report. Copy the lines of code you wrote in the report as well.

Problem 7 *Take Your Own Pictures for Princeton Campus*

Figure 6: Stiching Result

Take two pictures on our beautiful campus by yourself, run the code to stitch them together. Include the original two pictures and your stitching result in your report.

Extra Credit [optional]

To get extra credits, try the following techniques to improve your results for panoramic stitching:

- Now our results look reasonable but not perfect since you can see an obvious boundary where you stitched these two images. You could try alpha blending to merge the overlapping region instead of just copying and pasting the intensity value from one single image.

- Actually you can also try something smarter. First, use Graph Cut to find a optimal seam to cut out regions in two image that you will be using for stitching, then apply Poisson Blending to seamlessly blend two wrapped images.

- Extend the algorithm to handle n (> 2) images, and run it on your own photos, or photos you found on the web.

- Implement a system to combine a series of photographs into a $360^\circ \times 180^\circ$ panorama using equirectangular projection as shown in Figure 7(a). Refer to `http://en.wikipedia.org/wiki/Equirectangular_projection`.

- Convert the panorama into a stereographic projection (example in Figure 7(b)). This looks like a small planet for outdoor images with a sky. Refer to `http://en.wikipedia.org/wiki/Stereographic_projection`.

- Reconstruct the 3D geometry of the panorama. Refer to `http://panocontext.cs.princeton.edu`

What to submit: You need to submit two files: one PDF file for the report that contains your name, Princeton NetID, all the pictures taken and text to answer the questions; one

(a) equirectangular projection

(b) stereographic projection

Figure 7: Panorama Examples

ZIP file (not RAR or any other format) that contains all source code for your system, and a "ps3.m" file that takes no parameter as input and run directly in Matlab to generate the results reported in your PDF file. Both the PDF and ZIP file should be named using your Princeton NetID underscore cos429ps3. As an example using my account, they should be named "xj_cos429ps3.pdf" and "xj_cos429ps3.zip". To **verify your result** and **detect plagiarism** to make sure there is no cheating, we will use an automatic program to run your code and compare your code with other students' (including both this year and all previous years) and public available implementations (e.g. from Google, Bing, Github). Therefore, please follow the file format to make our grading job easier. Failure to follow these rules will result in losing your grade.