CSCI-B 657: Computer Vision
Assignment 2
Final Report

Alan Wu (alanwu@umail.iu.edu)
Bhavik Shah (bbshah@indiana.edu)
Shou Qiuwei (qiuwshou@umail.iu.edu)

February 28, 2016

1 Instructions

1. Part 1 of the assignment

   - To run Part 1 without optimization type the following command
     `./a2 part1 query.jpg <directory of images>`
   - The output is the list of images appended to each other with lines drawn for matching sifts point pairs. The file names are <inputfile>_combined_sift.png. We display the top 10 results as a program output.
   - The ranked output

```
[bbshah@tank bbshah-alanwu-qiuwshou-a2]$ ./a2 part1 a2-images/part1_images/bigben_2.jpg
a2-images/part1_images/
Time Elapsed:152 seconds
matches size:99
Displaying top 10 results:
0) matches: 2845 file:a2-images/part1_images/bigben_2.jpg
1) matches: 45 file:a2-images/part1_images/trafalgarsquare_20.jpg
2) matches: 43 file:a2-images/part1_images/tatemodern_6.jpg
3) matches: 35 file:a2-images/part1_images/tatemodern_14.jpg
4) matches: 35 file:a2-images/part1_images/sanmarco_1.jpg
5) matches: 32 file:a2-images/part1_images/tatemodern_4.jpg
6) matches: 29 file:a2-images/part1_images/bigben_14.jpg
7) matches: 27 file:a2-images/part1_images/tatemodern_2.jpg
8) matches: 26 file:a2-images/part1_images/notredame_5.jpg
9) matches: 26 file:a2-images/part1_images/tatemodern_11.jpg
10) matches: 26 file:a2-images/part1_images/tatemodern_24.jpg
```

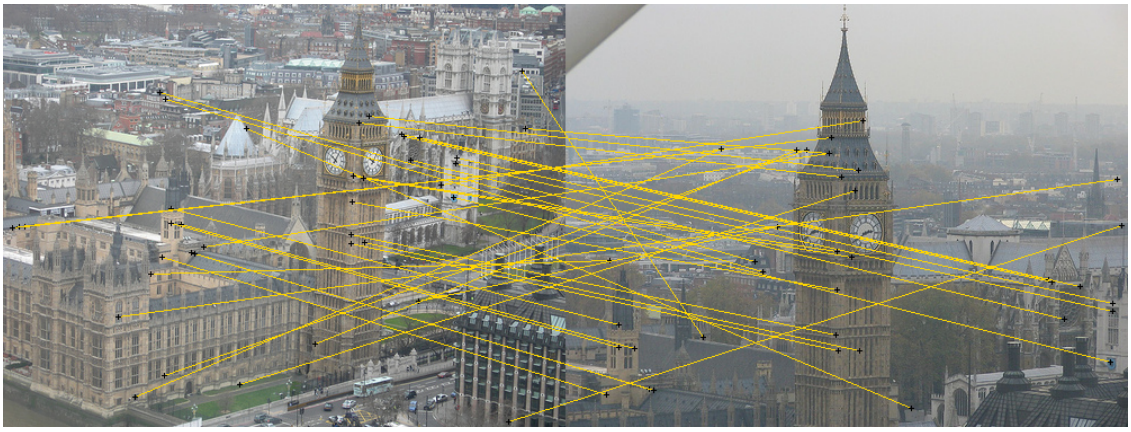

Figure 1. Matching SIFT points without optimization.

- To run the fast heuristic based SIFT comparison, use the command line
  `./a2 part1fast  query.jpg <directory of images>`
- The output is the list of images appended to each other with lines drawn for matching sifts point pairs. The file names are <inputfile>_sift_projection.png. We display the top 10 results as a program output.
- The ranked output

```
[bbshah@tank bbshah-alanwu-qiuwshou-a2]$ ./a2 part1fast a2-images/part1_images/bigben_2.jpg
a2-images/part1_images/
Time Elapsed:104 seconds
Displaying top 10 results:
0) matches: 2845 file:a2-images/part1_images/bigben_2.jpg
1) matches: 4 file:a2-images/part1_images/tatemodern_6.jpg
2) matches: 3 file:a2-images/part1_images/empirestate_15.jpg
3) matches: 3 file:a2-images/part1_images/bigben_14.jpg
4) matches: 3 file:a2-images/part1_images/empirestate_14.jpg
5) matches: 3 file:a2-images/part1_images/trafalgarsquare_20.jpg
6) matches: 2 file:a2-images/part1_images/colosseum_12.jpg
7) matches: 2 file:a2-images/part1_images/sanmarco_22.jpg
8) matches: 2 file:a2-images/part1_images/colosseum_5.jpg
9) matches: 2 file:a2-images/part1_images/empirestate_9.jpg
10) matches: 2 file:a2-images/part1_images/bigben_13.jpg
```

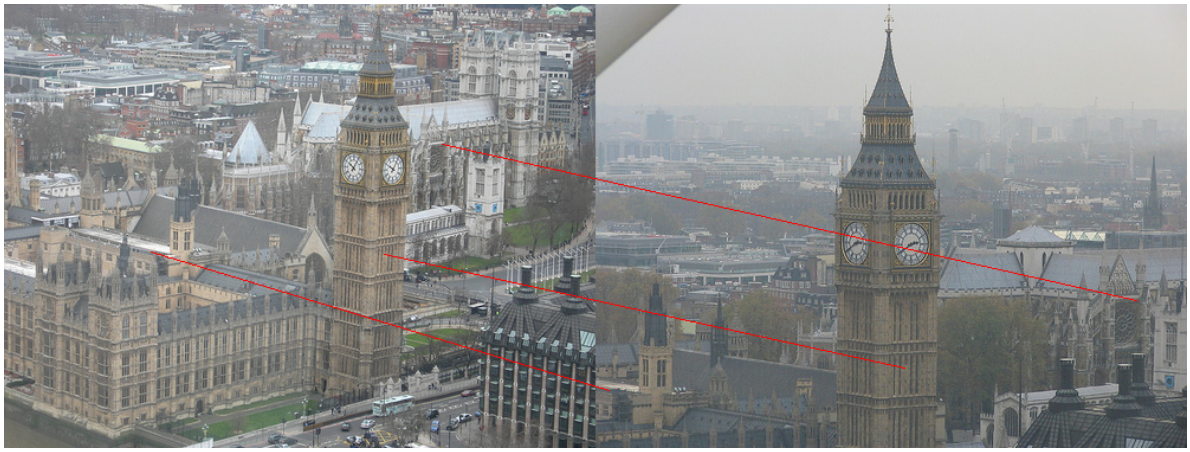

Figure 2. Matching SIFT points with optimization.

2. Part 2 of the assignment

- To produce a similar image to Figure 2 for Part 2.1, enter the following on the command line
  `./a2 part2.1 <image.jpg>`
- The output file name will be ¡filename¿-warped.png. Here is our result after implementing our inverse warping algorithm:

Figure 3. Warped image.

- Instructions to run Part 2.2.
  The image sequence warping application uses function <warp_image>. To create warped images with respect to the first cameras coordinate system, enter the following on the command line per assignment directions:
  (a) run the script do_part2.2 will generate two commands for wrapping the images under seq1 and seq2 directory
  (b) `./a2 part2.2 img_1.jpg img_2.jpg  img_n.jpg`
  (c) The above command should produce warped images called two images for each pairs, img_1_warpped.png, img_1_blend.png img_n_warpped.png, img_n_blend.png
  (d) Below are some sample results:

Figure 3. RANSAC image.

2 Assumptions and design decisions

1. Assumptions

   - We do not need to optimize memory usage
   - For Part 1.4, after finding a suitable k value (number of random vectors) and w value (quantizer) that produced similar results as the full SIFT, we fixed those values and applied them to all the comparisons per Part 1.3 part1fast.
   - For Part 2.1, we assume it is okay to hand-calculate the inverse matrix and hard code the values into our program
   - For Part 2.2 we use the coordinate of the first image (query image) as the first cameras coordinate system. And we calculate inverse projection matrix that maps query image into input image. Then we copy the pixel value from all input images into query image.

2. Design decisions

   - For both Parts 1.1 and 1.4, we use the ratio of the nearest match distance to the second nearest match distance to determine whether matches qualify. We assume that SIFT matches resulting from the ratio are good matches (although theres sometimes outliers that make their way in). For Part 1.3, we rank the images by the number of SIFT matches for the distance and ratio thresholds we had set. The dilemma we faced was how to rank according to good matches, but we found that using the ratio and applying a selective threshold have yielded a high percentage of good matches, thus validating our ranking system.

- For Part2.2, we set the number of iteration to 10000 make sure we can get good hypothesis. Also we use the 0.6 ratio threshold for matching sift descriptors, because we want to make sure that each feature will be distinctive. Also we create the blend version of warping images.

3 Part 1.3 Precision

See design decisions above for system of ranking. We achieved roughly 30% precision for a full SIFT match vs. also 30% precision for a fast, but abridged, SIFT match. Although the precision was comparable between the two algorithms, the fast SIFT match, however, did not take less run time than the full SIFT matching.

4 Results

- Part 1.4: We found that the optimal values for w, k, and the number of trials to be 130, 3, and 1, respectively. This yields similar performance results as the full SIFT.
- Part 2.2 : Using 0.6 for ratio threshold and 10000 for iteration produces good wrapped image.

5 References

- Inverse determinant calculation on Wikipedia
- CImg library definition website ¡CImg.eu¿
- class notes
- Discussed 2.1 with Vishesh Tanksale, Cyril Shelke