CSCI-B 657: Computer Vision
Assignment 2 : Image Warping, Matching and Stiching
Final Report

Ishtiak Zaman (izaman@indiana.edu)
Hanfei Mei (hanfei@indiana.edu)
Supreeth Keragodu Suryaprakash(skeragod@indiana.edu)

February 28, 2016

Part I
Image Matching

1 Explanation and Results:

1. The solution for this has been implemented in ***SIFT_match*** function.
   Input: *source image, query image, threshold value 1, threshold value 2.*
   Output: *Returns the number of matching points.*

   Implementation:

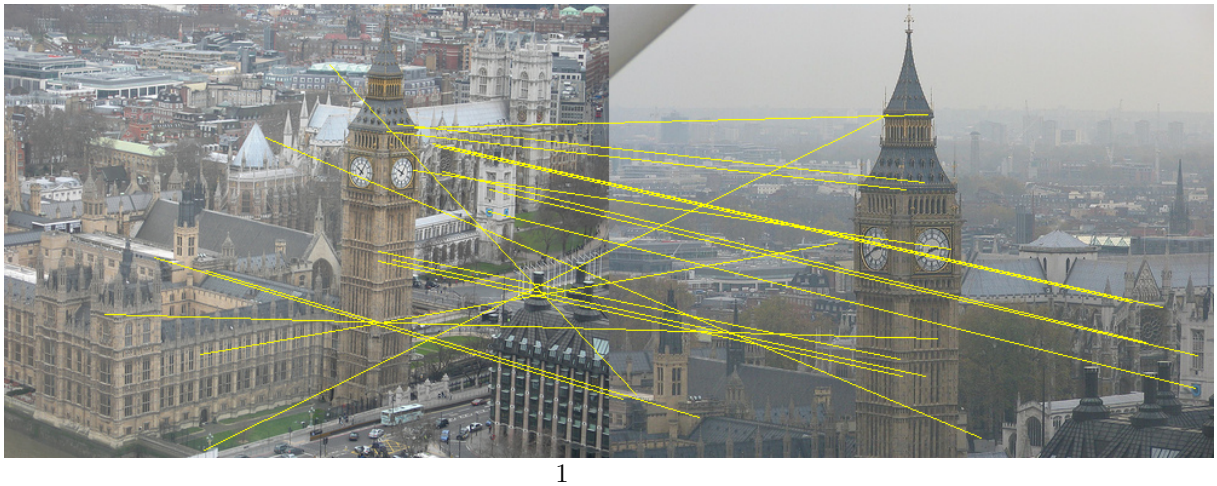
   (a) We are finding the sift descriptor vectors for both the images.

   (b) For each pair of *(s_d, q_d)*, *s_d* is a sift descriptor in source image and *q_d* is a sift descriptor in query image. We'll compute the Euclidean distance for both.

   (c) For each s_d in query image, We'll find the best two matching pairs of *(s_d, q_d)*, compute radio by *closest_distance(s_d,q_d1)/second_closest_distance(s_d,q_d2)*.

   (d) If ratio is less than the threshold, it is regarded as a pair of match points, draw a line in the merged image and return the total number of matched points.

   Output Image:



(Fig 1 : Matching points for bigben_2.jpg and bigben_3.jpg)

How to Run?

Command Line : *./a2 part1 < img1.extension >< img1.extension >*

This will provide you with three options, press 1 and hit enter.

2. The solution has been implemented in the *maximum_key_points* function.

   Input : argument_count, argument_vector, an interger value to provide solution for which problem.

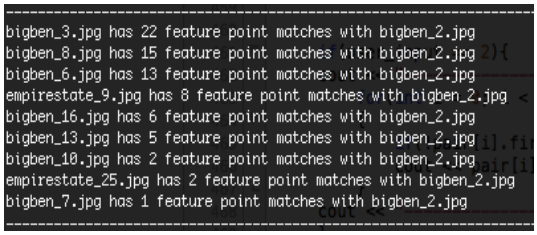
   Output : Prints the result on the console.

   Implementation : We are making use of the function that was explained in the previous question. The output would be the total number of key points matched. We have used a $< string, interger >$ vector to store the total number of match points for each query image with the source image. In the string part of the pair we are storing the file name of the query images. We are then sorting the vector based on 2^{nd} field (which contains integer values) and are displaying the result.

   How to Run?

   Command Line : *./a2 part1 < img1.extension >< img1.extension > ...*

   This will provide you with three options, press 2 and hit enter.



(Fig 2 : Sorted the vector in the increasing order of matched key points wrt source image.)

3. The solution for this question has been implemented in *maximum_key_points* function

   Input : argument_count, argument_vector, an interger value to provide solution for which problem.

   Output : Prints the result on the console.

   Implementation : We will be calculating the matching points in the function that was explained in question 1. The solution to this question has only change one from the previous question i.e. We will be running the source image against all the 100 images provided in part1_images folder provided by the Professor. We are storing the values in a vector as explained in the previous question and are sorting it in the same fashion. We will pick the top 10 images from the vector. We will caculate the number of correct predictions and the wrong predictions for 10, thereby calculating the precision factor. Formula to find the precision is as below,

$$\mathrm{P} = \frac{correct\_prediction}{correct\_prediction + wrong\_prediction} * 100$$

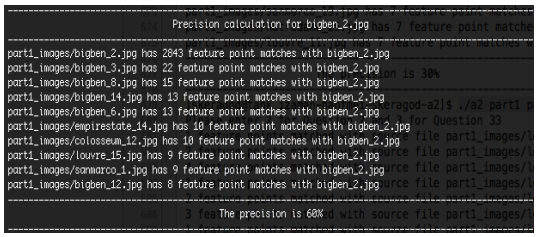
Serial Number	File Name	Precision
1	bigben_2.jpg	60%
2	colosseum_11.jpg	30%
3	eiffel_1.jpg	20%
4	empirestate_10	50%
5	londoneye_12.jpg	30%
6	louvre_10.jpg	20%
7	notredame_1.jpg	30%
8	sanmarco_1	30%
9	tatemodern_11.jpg	10%
10	trafalgarsquare_1.jpg	50%

How to Run?

Command Line : *./a2 part1 < img1.extension >< img1.extension > ...*

e.g. *./a2 part1 part1_images/louvre_10.jpg part1_images/*.jpg*

This will provide you with three options, press 3 and hit enter. You can view the number of keypoints matched with each image on the console as well as the top 10 matched images and the precision value.



(Fig 3 : Output screengrab for Bigben_2.jpg which has a precision of 60%)

4. The solution for this has been implemented in ***SIFT_summary_match*** function.
Input: Source Image, Query Image, k, w.
output: Number of match points in both the images.
Implementation:

   (a) We have used gaussian random to generate k summary vectors. Each vector contains 128 random number which obeys the 0,1 Guassian Distribution.

   (b) Calculate k features $f(v)$ for each descriptor in both source image and query image.

   (c) For each descriptor d in source image, find a few candidates in query Image which has a similar $f(v)$ with $f(d)$.

   (d) For every descriptor in source image, find the first and second closest descriptors in candidates.

   (e) Compute the ratio of the most and second similar candidate descriptor for each descriptor d and if the ratio is less than the threshold, it is regarded as a match point for d.

Running time: it works a little faster than the normal function. A random 50 iterations in *SIFT_match* will take $68.349s$ and *SIFT_summary_match* will take $67.251s$ because of the dimensionality reduction.

(Fig 4: Output of the SIFT_summary_match function for bigben_2.jpg and bigben_3.jpg)

Part II
Image Warping and Homographies

1. **Transformation:** We used inverse warping to get a projection of an image given a 3x3 transformation matrix. During the inverse warping method, when some pixel in the destination image points to a pixel outside of the source image dimensions, we used black pixel for those missing parts. When we need to get a non-integer pixel value from the source image, we used nearest neighbor interpolation.

   Following is an example of transformation of the left image into the right one using the transformation matrix $[[0.907, 0.258, -182], [-0.153, 1.44, 58], [-0.000306, 0.000731, 1]]$. The black area on the right image does not have any correspondences with any part of the source image.

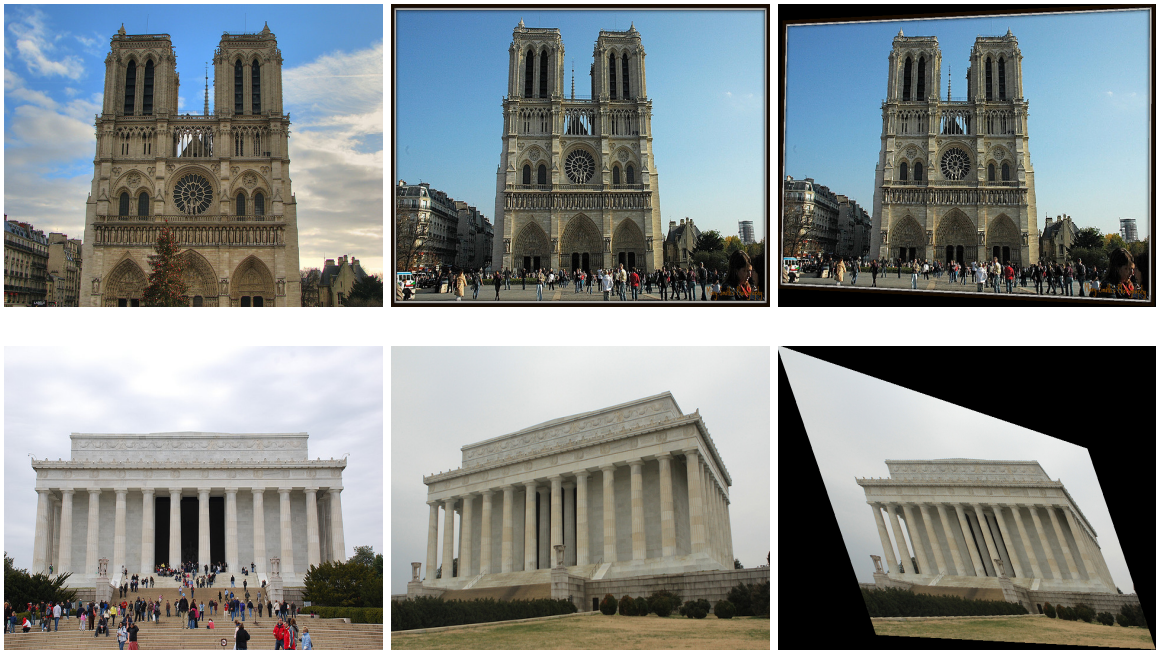
(Fig 5: Depiction of Tranformation)

2. **Homography Estimation using RANSAC:**

   We used feature based alignment to get matches between two images using SIFT library. After getting the matches, we used RANSAC algorithm to maximize our inliers. In the RANSAC, we randomly choose 4 set of matches, get the transformation matrix by solving 8 equations with 8 unknown.

   We used CImg linear solver library to do this task. Afterwards, we applied the transformation and see how many other matches agree with this transformation. We fine tuned our threshold parameters to maximize inliers with minimum false positive. The transformation with highest ratio of inliers is the winning hypothesis and we take that transformation to warp the second image. When the SIFT matches are good, we get a very good inlier ratio that results in a good warping of the image. But when the images are not very similar and the SIFT matches are not very good, we get a poor performance in the RANSAC with a low number of inlier matches. It results in a poor warped image. The performance of the RANSAC heavily depends on the quality of the SIFT matches.

Here are a few examples where the second image was warped into the third image in a way such that it looks like the second image was taken from the camera position of the first image.

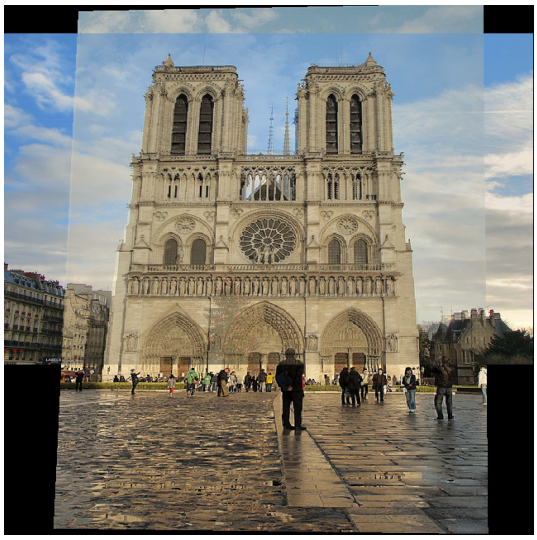


3. *Image Stitching:* After finding the homography transformation between two images, we used this information to make a panorama image by stitching the images together. We warped the second image as if it was taken from the camera position of the first image, and stitches the images together. There are common parts for both the two images, in those places we used mean blending technique to blend the pixels.

   Here are some examples where we have stitched the first and second images to create the third panoramic image.

   Examples :

(Fig 6a: Two images taken from different angles)



(Fig 6b: Stiched Image)

(Fig 7a: Two images taken from different angles. Fig 7b: Stiched Image(below))

How to Run the Code for Part 2?

To execute the part2, You need to provide the command line arguments as below,

./a2 part1 $< img1.extension >< img1.extension > ... < imgn.extension >$

The execution of the above command line will produce warped images viz.

$< img2\_warped.extension >< img3\_warped.extension > ... < imgn\_warped.extension >$

and panorama images viz.

$< img2\_panorama.extension >< img3\_panorama.extension > ... < imgn\_panorama.extension >$.

2 References :

- http://www.cnblogs.com/yeahgis/archive/2012/07/13/2590485.html
- Lecture slides by Prof. David Crandall