Chinmay Chinara
USC-ID: 2527237452
E-mail: chinara@usc.edu

Due date: 10-06-2017

CSCI-677: HOMEWORK #3

Due date: 10-06-2017

Table of Contents

Chinmay Chinara
USC-ID: 2527237452
E-mail: chinara@usc.edu

SIFT Features

Discussion

Salient point extraction is very much required in image processing and computer vision tasks specifically such as object recognition, video tracking, navigation, gesture recognition and image stitching, we need a means to extract meaningful features that can describe the image. These kinds of feature extraction requirements are where SIFT come into the picture. These extracted features are required to be scale invariant and rotation invariant to provide robustness. SIFT is such an algorithm that extract important local features in an image that are scale and rotation invariant.

Object categorization and localization is sometime required to enhance an image. The biggest challenges in object classification are introduced due to problems such as different camera positions, illumination differences, internal parameters and variation within the object which has not been solved till date and is a big area of research that is still going on.

SIFT was proposed by David G Lowe. To implement this, OpenCV was used that contains SIFT detectors and extractors. The overview of SIFT algorithm is as under:

➔ Scale Space using Gaussian Kernel construction.

➔ Now, we compute DoG (Difference of Gaussian) – an approximation of LoG (Laplacian of Gaussian)

Given a Gaussian blurred image,

$$L(x, y, \sigma) = G(x, y, \sigma) \times I(x, y) \quad \text{➔ (1)}$$

where,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} exp^{\frac{-(x^2+y^2)}{\sigma^2}} \quad \text{➔ (1)}$$

Now, we have difference of Gaussian blurred images at scales σ and kσ .

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad \text{➔ (3)}$$

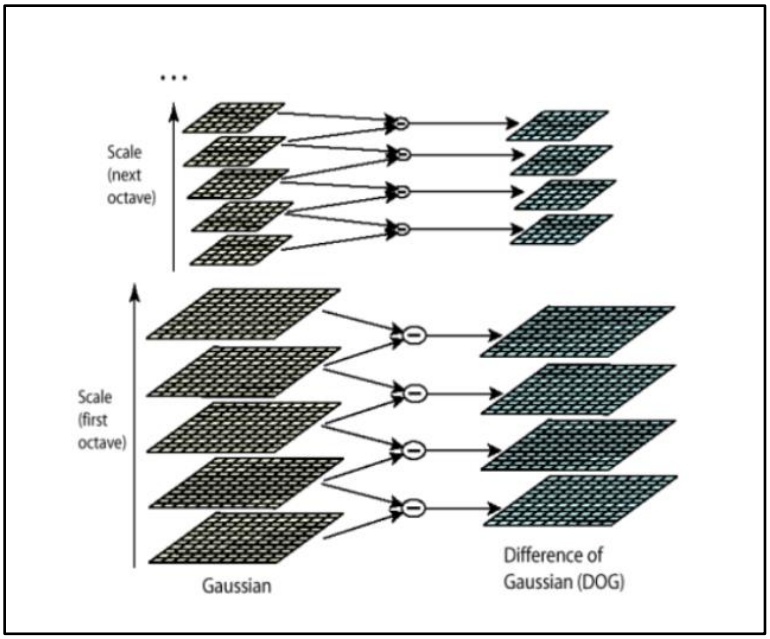

Figure a: Blurred images at different scales and DoG computation (SIFT)

- ➔ We now must locate the extrema of DoG. This can be done by scanning through each DoG image and then identifying the maximum and the minimum.
- ➔ Taylor Series expansion to localize the sub-pixels needs to be used.
- ➔ Filtering out low contrast points using scale space values at previous locations and filter edge responses using Hessian Matrix.
- ➔ Assignment of key point orientations by creation of histogram of local gradient directions.
- ➔ Building the key point descriptors, wherein each key point has a location, orientation and scale.
- ➔ Find blurred image of nearest scale and then sample the points around the key point. Now, rotate the gradients & co-ordinates by the orientation computed previously.
- ➔ Divide the region into sub-regions and create a histogram for each sub-region with several bins.

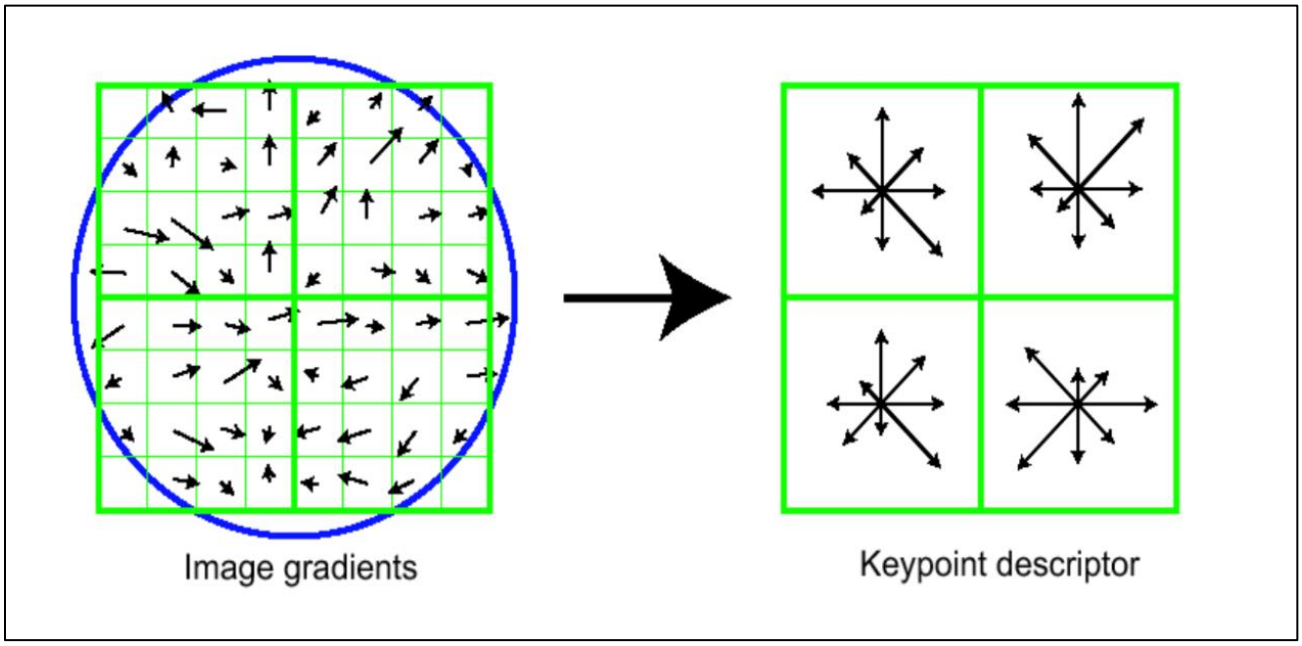

Figure b: Gradient orientation and normalization tricks (SIFT)
Source: http://3.bp.blogspot.com/-2Lw3DxApZrw/VKBKMeAwTnI/AAAAAAAANyU/7IQxfszsclc/s1600/sift_pic.png

- ➔ Once we select the key point orientation, the feature descriptor is then calculated as a set of orientation histograms on 4 × 4 pixel neighbor hoods.
- ➔ The orientation data is obtained from the Gaussian image which is closest in scale to the key point's scale.
- ➔ Histograms contain 8 bins, and descriptors contains array of 4 histograms around the key point and this leads to a SIFT feature vector with (4 × 4 × 8 =) 128 elements. This vector is then normalized to nullify the changes in illumination.
- ➔ Using SIFT algorithm, we obtain scale and rotation invariance, view point variance by using scale space and illumination changes.
  For programming this, we use OpenCV built-in class for SIFT as shown below in the steps involved in the code section.

Lowe's ratio test:

This is a very critical aspect of keypoint matching proposed by Lowe in his paper [1]. The best candidate match for each keypoint was found by identifying its nearest neighbor in the database of keypoints from training images. The nearest neighbor is defined as the keypoint with minimum Euclidean distance for the invariant descriptor vector.

However, many features from an image will not have any correct match in the training database because they arise from background clutter or were not detected in the training images. Therefore, it would be useful to have a way to discard features that do not have any good match to the database. A global threshold on distance to the closest feature does not perform well, as some descriptors are much more discriminative than others. A more effective measure was obtained by comparing the distance of the closest neighbor to that of the second-closest neighbor. If there were multiple training images of the same object, the second-closest neighbor was defined as being the closest neighbor was known to have come from a different object than the first, such as by only using images known to contain different objects. This measure performed well because correct matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve reliable matching. For false matches, there were several other false matches within similar distances due to the high dimensionality of the feature space. The second-closest match can be thought as providing an estimate of the density of false matches within this portion of the feature space and at the same time identifying specific instances of feature ambiguity.

Matches for which the nearest neighbor was a correct match have a PDF that is centered at a much lower ratio than that for incorrect matches. For our object recognition implementation, we reject all matches in which the distance ratio is greater than 0.7, which eliminates most of the false matches while discarding very few correct matches.

RANSAC:

RANSAC or Random Sample and Consensus algorithm is a learning technique to estimate parameters of a model by random sampling of observed data. Given a dataset whose data elements contain both inliers and outliers, RANSAC uses the voting scheme to find the optimal fitting result. Data elements in the dataset are used to vote for one or multiple models. The implementation of this voting scheme is based on two assumptions: that the noisy features will not vote consistently for any single model (few outliers) and there are enough features to agree on a good model (few missing data). The RANSAC algorithm is essentially composed of two steps that are iteratively repeated:

In the first step, a sample subset containing minimal data items is randomly selected from the input dataset. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest sufficient to determine the model parameters.

In the second step, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the fitting model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the effect of noise.

The set of inliers obtained for the fitting model is called consensus set. The RANSAC algorithm will iteratively repeat the above two steps until the obtained consensus set in certain iteration has enough inliers.

The input to the RANSAC algorithm is a set of observed data values, a way of fitting some kind of model to the observations, and some confidence parameters. RANSAC achieves its goal by repeating the following steps:

1. Select a random subset of the original data. Call this subset the hypothetical inliers.
2. A model is fitted to the set of hypothetical inliers.
3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the consensus set.
4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.
5. Afterwards, the model may be improved by re-estimating it using all members of the consensus set.

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size. In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.

Steps involved in the code:

1. Read the detection and target images
2. Initiate the SIFT detector
3. Find the keypoints and descriptors with SIFT for both the images
4. Find the orientation of all the keypoints
5. Initialize Brute Force Matcher with the default parameters
6. Use knnMatch() function of Brute Force Matcher to get the best 'k' matches. k=2 so that Lowe's ratio test could be applied
7. Perform Lowe's ratio test to capture the good matches with a threshold of 0.7
8. Set condition that at least 10 matches need to be found
9. Compute the homography with RANSAN fitting
10. Find the inliers and the outliers
11. Draw the final sift matching between the two images

Results and Conclusion:

The table below shows all the values that were calculated from the entire process:

Table 1: All data calculated

QUERRY / DETECTION IMAGE	TRAIN / TARGET IMAGE	QUERRY IMAGE FEATURES	TARGET IMAGE FEATURES	GOOD MATCHES: LOWE'S RATIO TEST	TOTAL NUMBER OF INLIERS (CONSISTENT WITH HOMOGRAPHY)
image_1.jpg	image_3.jpg	3239	1721	409	358
	image_4.jpg	3239	1612	235	171
	image_5.jpg	3239	1660	559	525
image_2.jpg	image_3.jpg	2830	1721	199	168
	image_4.jpg	2830	1612	213	190
	image_5.jpg	2830	1660	48	35

Image-1 and Image-3 matching

Figure 1: image_1 and image_3 keypoints (location) and orientation (direction)

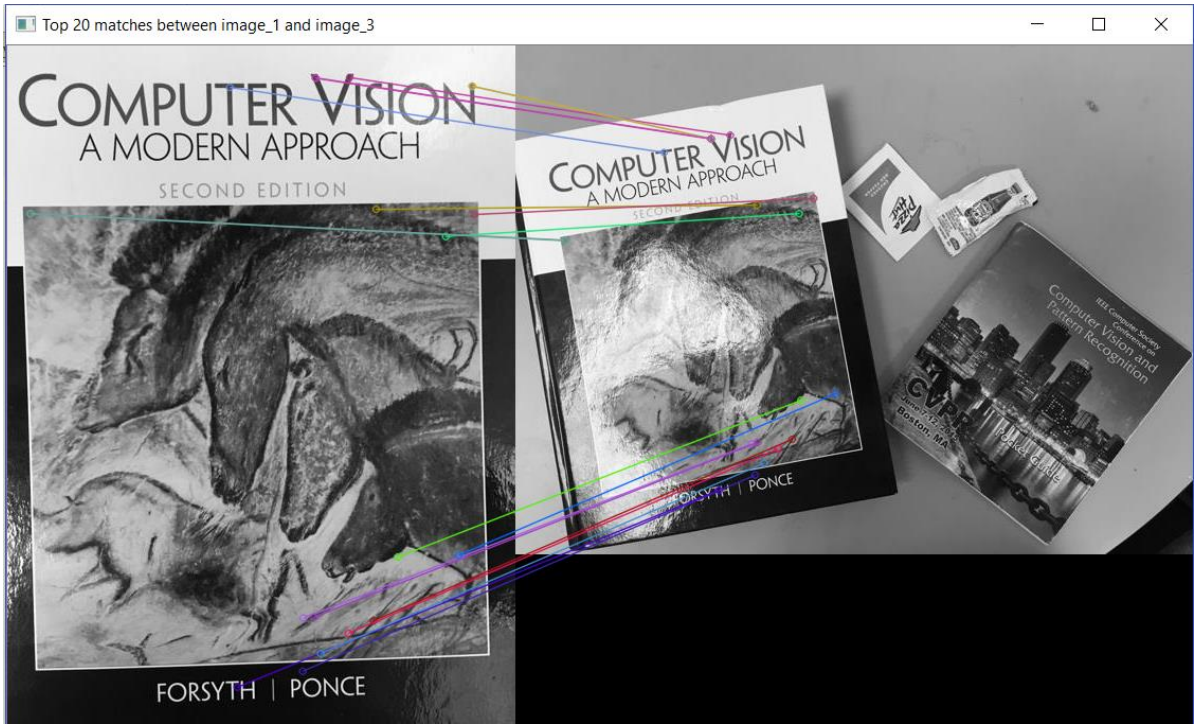

Figure 2: image_1 and image_3 top 20 matches before RANSAC operation

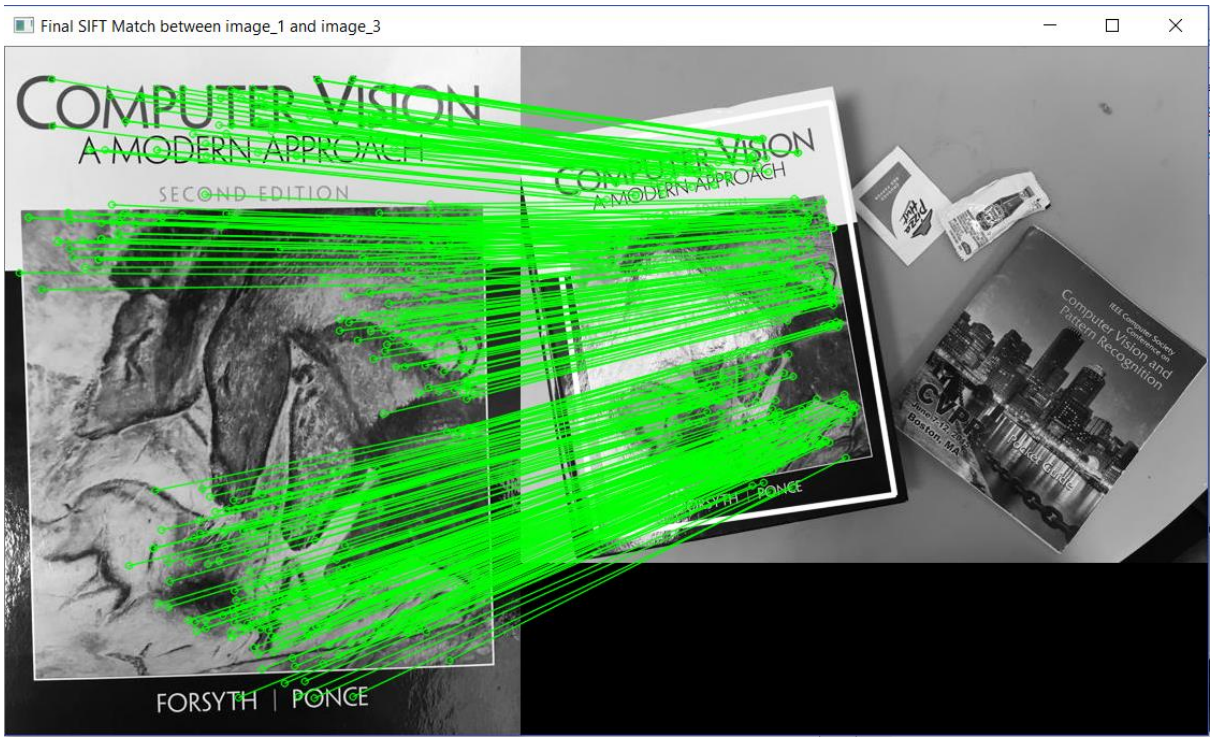

Figure 3: image_1 and image_3 top 10 (or more) matches after homography applied

```
The features found in querry image image_1: 3239
The features found in train image image_3: 1721
The total number of good matches are: 409
The total number of inliers (good matches providing correct estimation and total numbers consistent
 with the computed homography)) are: 358
The homography matrix when image_1 is matched with image_3:
[[  5.41725339e-01   7.31361376e-02   2.22427987e+01]
 [ -1.02199045e-01   5.34159852e-01   9.96019450e+01]
 [ -4.96807467e-05  -5.45282959e-05   1.00000000e+00]]
```

Figure 4: image_1 and image_3 outputs and homography matrix

Image-1 and Image-4 matching

Figure 5: image_1 and image_4 keypoints (location) and orientation (direction)

Figure 6: image_1 and image_4 top 20 matches before RANSAC operation

Figure 7: image_1 and image_4 top 10 (or more) matches after homography applied

```
The features found in querry image image_1: 3239
The features found in train image image_4: 1612
The total number of good matches are: 235
The total number of inliers (good matches providing correct estimation and total numbers consistent
 with the computed homography)) are: 171
The homography matrix when image_1 is matched with image_4:
[[ -2.53966212e-01   5.32010312e-01   2.93132132e+02]
 [ -6.05266641e-01  -3.08883919e-01   4.76679231e+02]
 [  2.59303390e-05  -2.07043507e-04   1.00000000e+00]]
```

Figure 8: image_1 and image_4 outputs and homography matrix

Image-1 and Image-5 matching

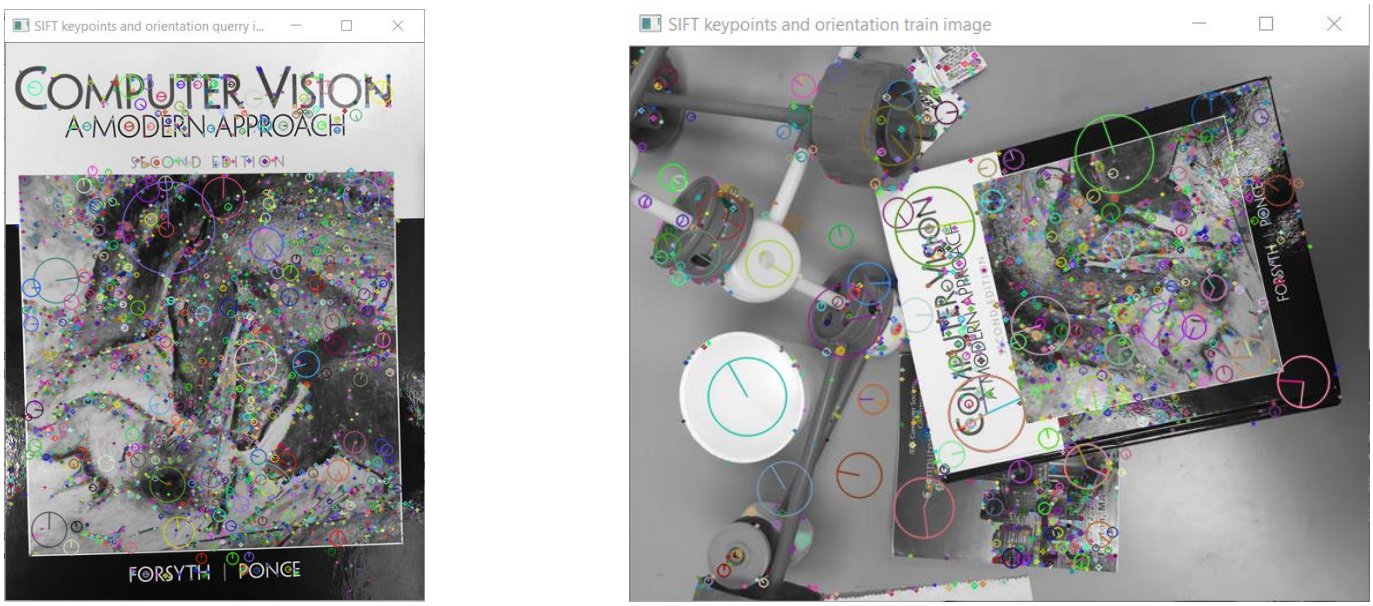

Figure 9: image_1 and image_5 keypoints (location) and orientation (direction)

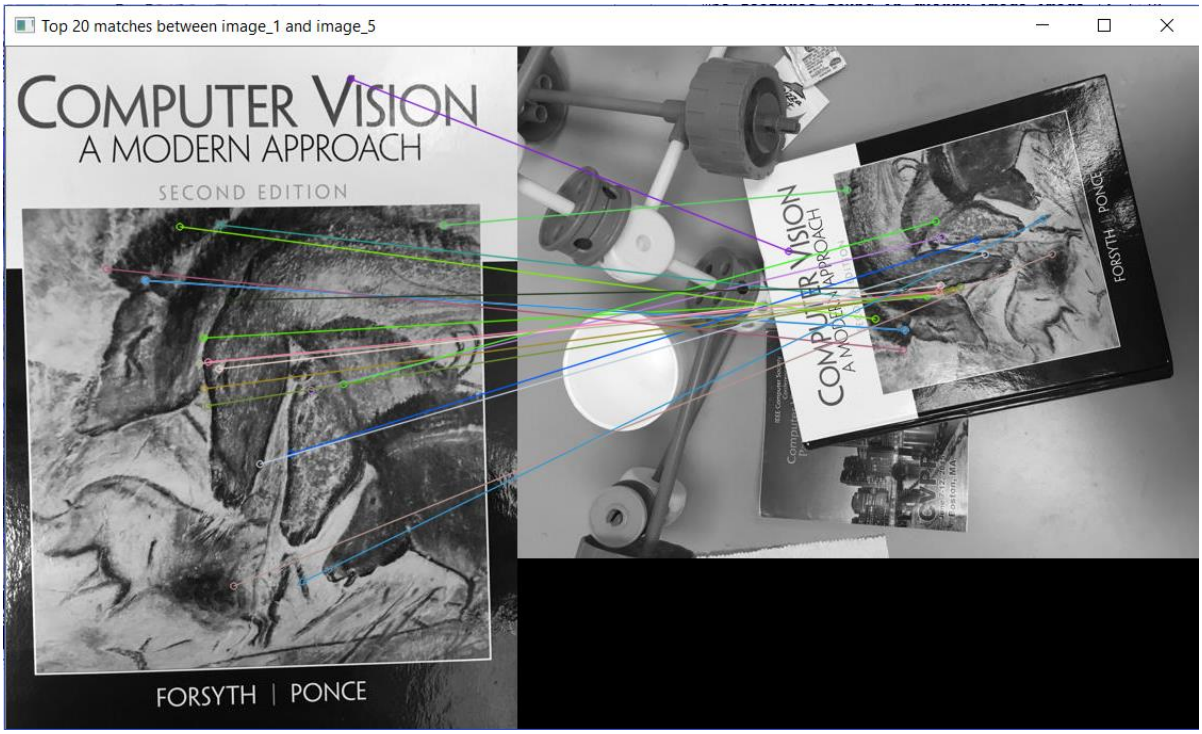

Figure 10: image_1 and image_5 top 20 matches before RANSAC operation

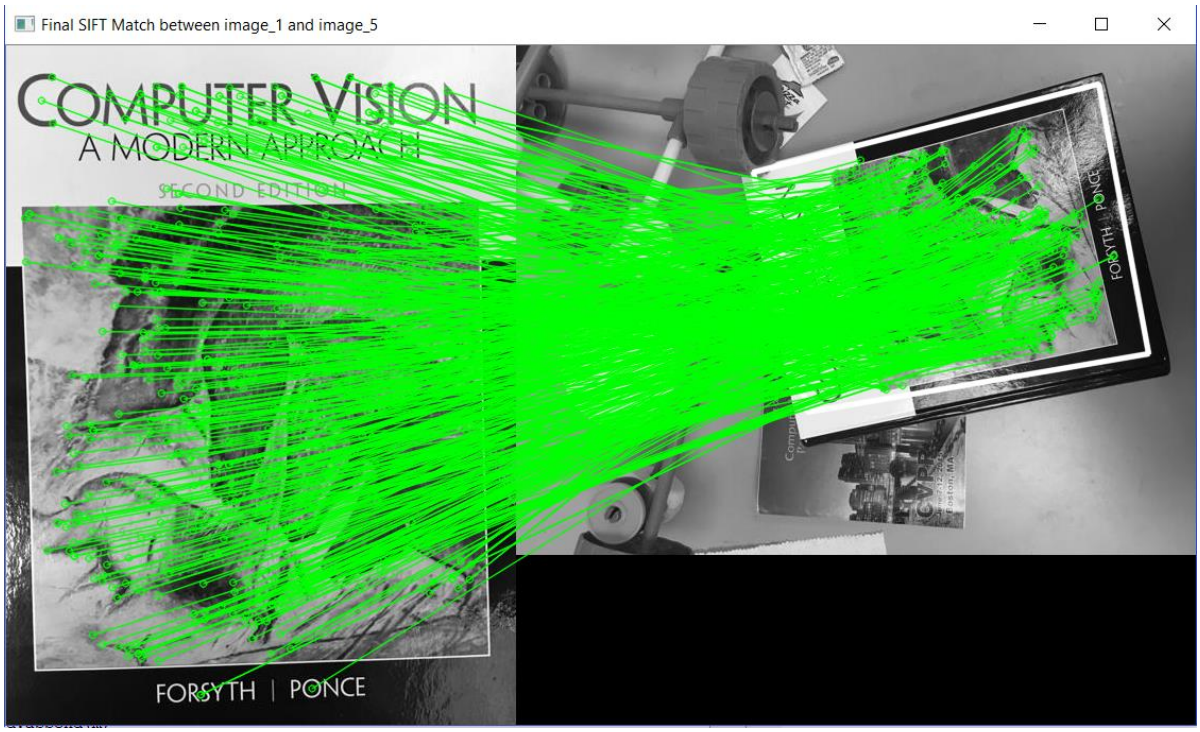

Figure 11: image_1 and image_5 top 10 (or more) matches after homography applied

```
The features found in querry image image_1: 3239
The features found in train image image_5: 1660
The total number of good matches are: 559
The total number of inliers (good matches providing correct estimation and total numbers consistent
 with the computed homography)) are: 525
The homography matrix when image_1 is matched with image_5:
[[ -1.17656513e-01   4.07257641e-01   2.74460994e+02]
 [ -4.76901373e-01  -1.32967113e-01   3.45312111e+02]
 [ -5.21688676e-05  -1.60795795e-04   1.00000000e+00]]
```

Figure 12: image_1 and image_5 outputs and homography matrix

Image-2 and Image-3 matching

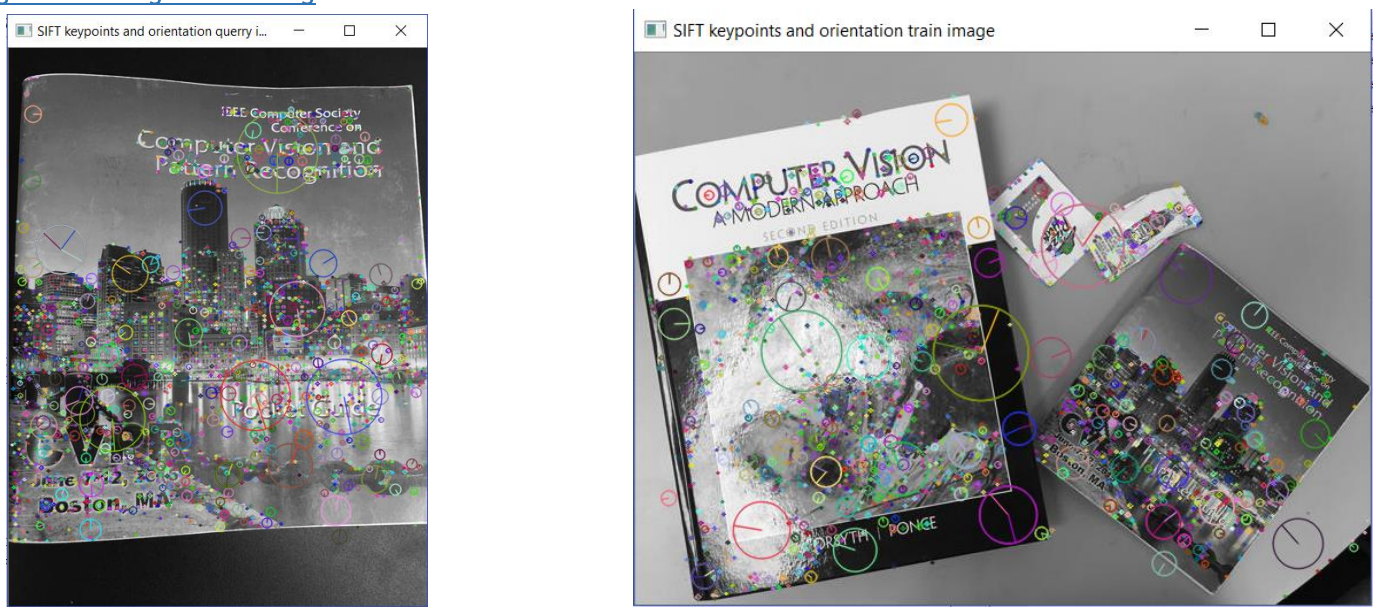

Figure 13: image_2 and image_3 keypoints (location) and orientation (direction)

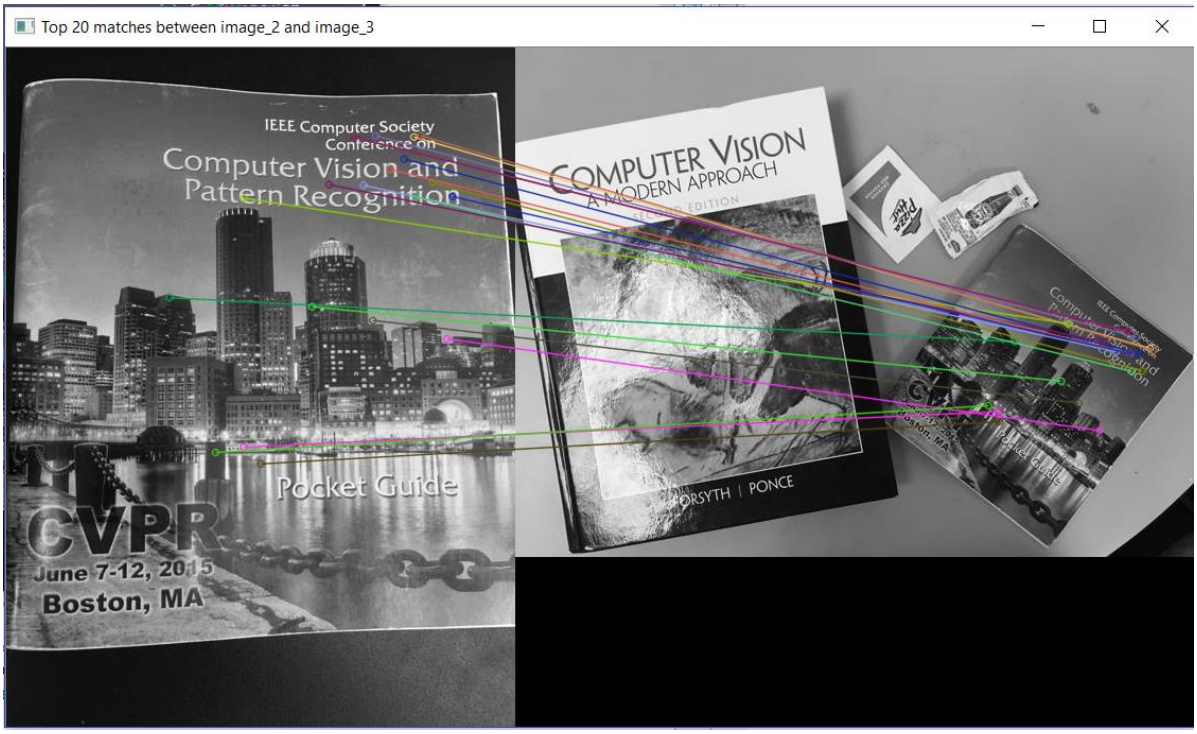

Figure 14: image_2 and image_3 top 20 matches before RANSAC operation

Figure 15: image_2 and image_3 top 10 (or more) matches after homography applied

```
The features found in querry image image_2: 2830
The features found in train image image_3: 1721
The total number of good matches are: 199
The total number of inliers (good matches providing correct estimation and total numbers consistent
 with the computed homography)) are: 168
The homography matrix when image_2 is matched with image_3:
[[  2.56646167e-01  -1.62871509e-01   4.82775517e+02]
 [  2.11945890e-01   4.32366045e-01   1.50071215e+02]
 [ -1.88733968e-04   2.47327095e-04   1.00000000e+00]]
```

Figure 16: image_2 and image_3 outputs and homography matrix

Image-2 and Image-4 matching

Figure 17: image_2 and image_4 keypoints (location) and orientation (direction)

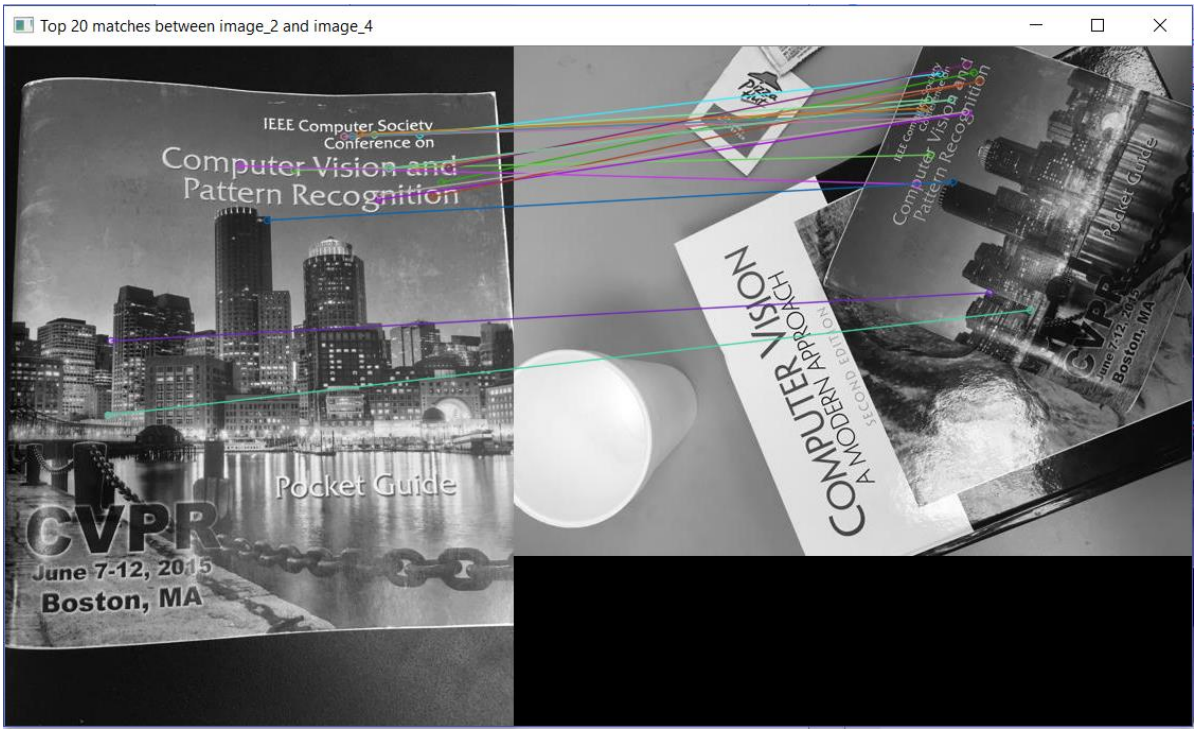

Figure 18: image_2 and image_4 top 20 matches before RANSAC operation

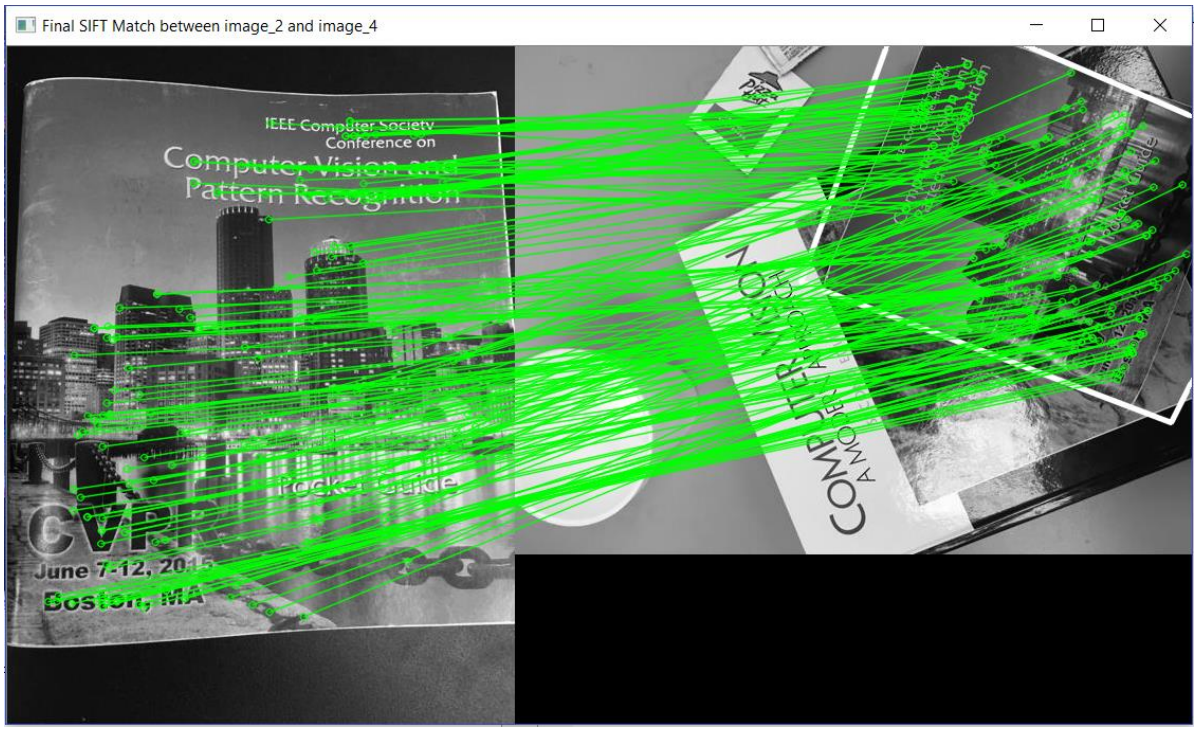

Figure 19: image_2 and image_4 top 10 (or more) matches after homography applied

```
The features found in querry image image_2: 2830
The features found in train image image_4: 1612
The total number of good matches are: 213
The total number of inliers (good matches providing correct estimation and total numbers consistent
 with the computed homography)) are: 190
The homography matrix when image_2 is matched with image_4:
[[  1.39345962e-01   6.04892700e-01   2.70781238e+02]
 [ -5.61280234e-01   2.40700750e-01   2.22532009e+02]
 [ -1.79029430e-04   9.36431317e-05   1.00000000e+00]]
```

Figure 20: image_2 and image_4 outputs and homography matrix

Image-2 and Image-5 matching

Figure 21: image_2 and image_5 keypoints (location) and orientation (direction)

Figure 22: image_2 and image_5 top 20 matches before RANSAC operation

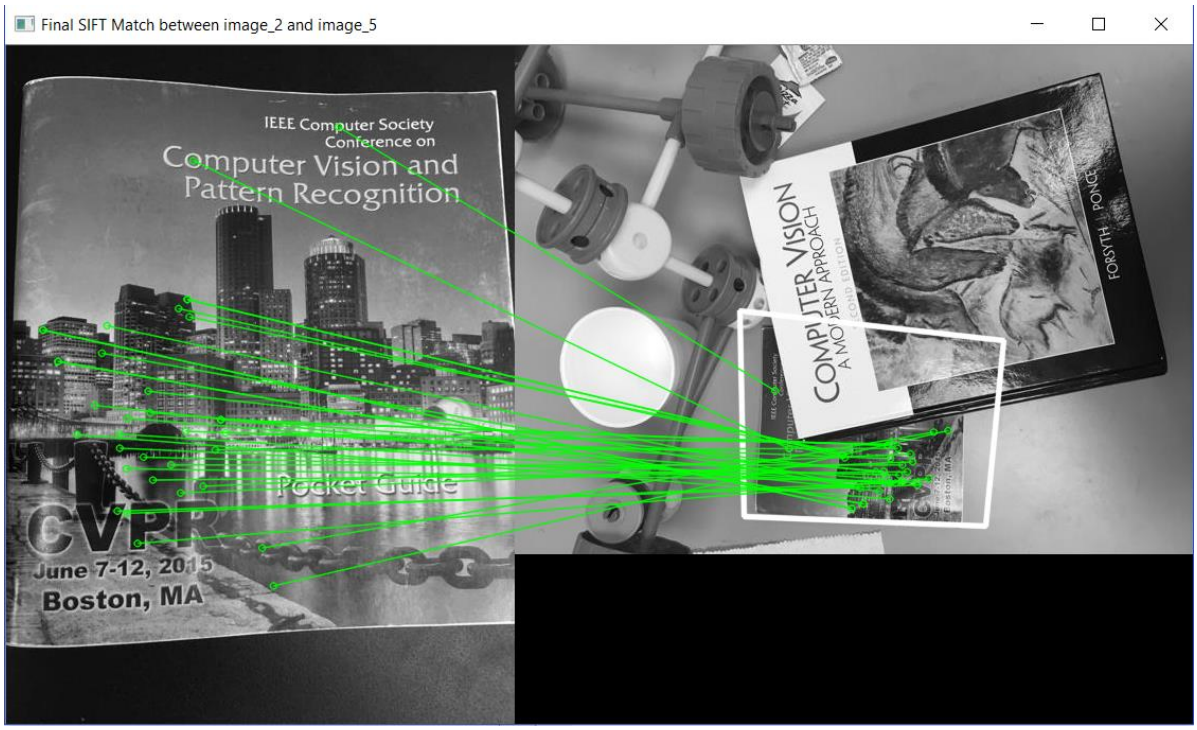

Figure 23: image_2 and image_5 top 10 (or more) matches after homography applied

```
The features found in querry image image_2: 2830
The features found in train image image_5: 1660
The total number of good matches are: 48
The total number of inliers (good matches providing correct estimation and total numbers consistent
 with the computed homography)) are: 35
The homography matrix when image_2 is matched with image_5:
[[ -5.05865434e-02   4.21500711e-01   2.16788853e+02]
 [ -4.54640393e-01   8.16911808e-02   4.44576644e+02]
 [ -1.97468455e-04   1.48361941e-04   1.00000000e+00]]
```

Figure 24: image_2 and image_5 outputs and homography matrix

From the above images, the following conclusions can be drawn:

1. The total number of features calculated for all the images are shown in the images above along with the orientation. Each of these features (or keypoints) are calculated based on different object variations in different parts of the image like slant in text content, size of image parts, etc. and certain salient features.
2. SIFT gave a very good feature mapping of the image to be detected with the various target images. For all the images, various features were detected, but SIFT estimated just the right features matching the detection image to the clutter in the target image. This can be seen in the images above.
3. Before homography was calculated, in the images where top 20 points are shown, some feature matching points got lost is the target image clutter and gave some degree of error. However, this error level is very less due to the use of knnMatch() of the brute force matcher followed by applying the Lowe's ratio test, which reduced the matches to the best possible estimates based on the distance. The threshold of the distance ratio was kept as 0.7 (which is a standard as per [1] that gives the best match).

4. Once the homography was calculated using RANSAC, the minute matching errors above got fully suppressed as RANSAC helped in giving the best fit. On applying homography with RANSAC, the total number of inliers (i.e. the good matches giving the right estimation of features matched) was found and on plotting those the correct feature match was achieved. So, from the total good matches approximately 90% of the matches fell under the inliers and 10% were outliers (the wrong matches). All these can be very clearly seen in the images above.
5. Table-1 and the output window snapshots show all the data for each of the image pairs stating the total number of features, the top 20 matches before RANSAC, the total good matches, the inliers and the homography matrix.

References

[1] http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf#page=20
[2] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html
[3] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

Index

System configuration in which all the training and testing was done

Operating System	Windows 10 Home 64-bit
Processor (CPU)	Intel® Core™ i7-7700HQ CPU @ 2.80GHz
Processor (GPU)	NVIDEA GDDR5 6GB

Development Environment

Python 3.5 with PyCharm 3.5 and OpenCV 3.2