Assignment2_Final3

May 3, 2022

1 Computer Vision 2022 Assignment 2: Image matching and retrieval

In this assignment, you will experiment with image feature detectors, descriptors and matching. There are 3 main parts to the assignment:

- matching an object in a pair of images
- searching for an object in a collection of images
- analysis and discussion of results

This assignment will have a minimum hurdle of 40%. You will fail if you can not reach the minimum hurdle.

1.1 General instructions

As before, you will use this notebook to run your code and display your results and analysis. Again we will mark a PDF conversion of your notebook, referring to your code if necessary, so you should ensure your code output is formatted neatly.

When converting to PDF, include the outputs and analysis only, not your code. You can do this from the command line using the `nbconvert` command (installed as part of Jupyter) as follows:

```
jupyter nbconvert Assignment2.ipynb --to pdf --no-input --TagRemovePreprocessor.remove_cell_tag
# Or
jupyter nbconvert Assignment2.ipynb --TagRemovePreprocessor.remove_cell_tags='{"remove_cell"}'
```

```
[NbConvertApp] Converting notebook /content/Assignment2_final2.ipynb to html
[NbConvertApp] Writing 25935623 bytes to /content/Assignment2_final2.html
```

Please do try this ccommond early before the last day! As the commend may be a little bit difference depending on your computer and the environment.

This will also remove the preamble text from each question. We will use the `OpenCV` library to complete the prac. It has several built in functions that will be useful. You are expected to consult documentation and use them appropriately.

This being the second assignment, we have provided less strict direction this time and you have more flexibility to choose how you answer each question. However you still need to ensure the outputs and report are clear and easy to read. This includes:

- sizing, arranging and captioning image outputs appropriately

- explaining what you have done clearly and concisely
- clearly separating answers to each question

1.2 Data

We have provided some example images for this assignment, available through a link on the MyUni assignment page. The images are organised by subject matter, with one folder containing images of book covers, one of museum exhibits, and another of urban landmarks. Within each category, there is a "Reference" folder containing a clean image of each object and a "Query" folder containing images taken on a mobile device. Within each category, images with the same name contain the same object (so 001.jpg in the Reference folder contains the same book as 001.jpg in the Query folder). The data is a subset of the Stanford Mobile Visual Search Dataset which is available at

`http://web.cs.wpi.edu/~claypool/mmsys-dataset/2011/stanford/index.html`.

The full data set contains more image categories and more query images of the objects we have provided, which may be useful for your testing!

Do not submit your own copy of the data or rename any files or folders! For marking, we will assume the datasets are available in subfolders of the working directory using the same folder names provided.

Here is some general setup code, which you can edit to suit your needs.

2 Question 1: Matching an object in a pair of images (45%)

In this question, the aim is to accurately locate a reference object in a query image, for example:

0. Download and read through the paper ORB: an efficient alternative to SIFT or SURF by Rublee et al. You don't need to understand all the details, but try to get an idea of how it works. ORB combines the FAST corner detector (covered in week 4) and the BRIEF descriptor. BRIEF is based on similar ideas to the SIFT descriptor we covered week 4, but with some changes for efficiency.

1. [Load images] Load the first (reference, query) image pair from the "book_covers" category using opencv (e.g. `img=cv2.imread()`). Check the parameter option in " cv2.imread()" to ensure that you read the gray scale image, since it is necessary for computing ORB features.

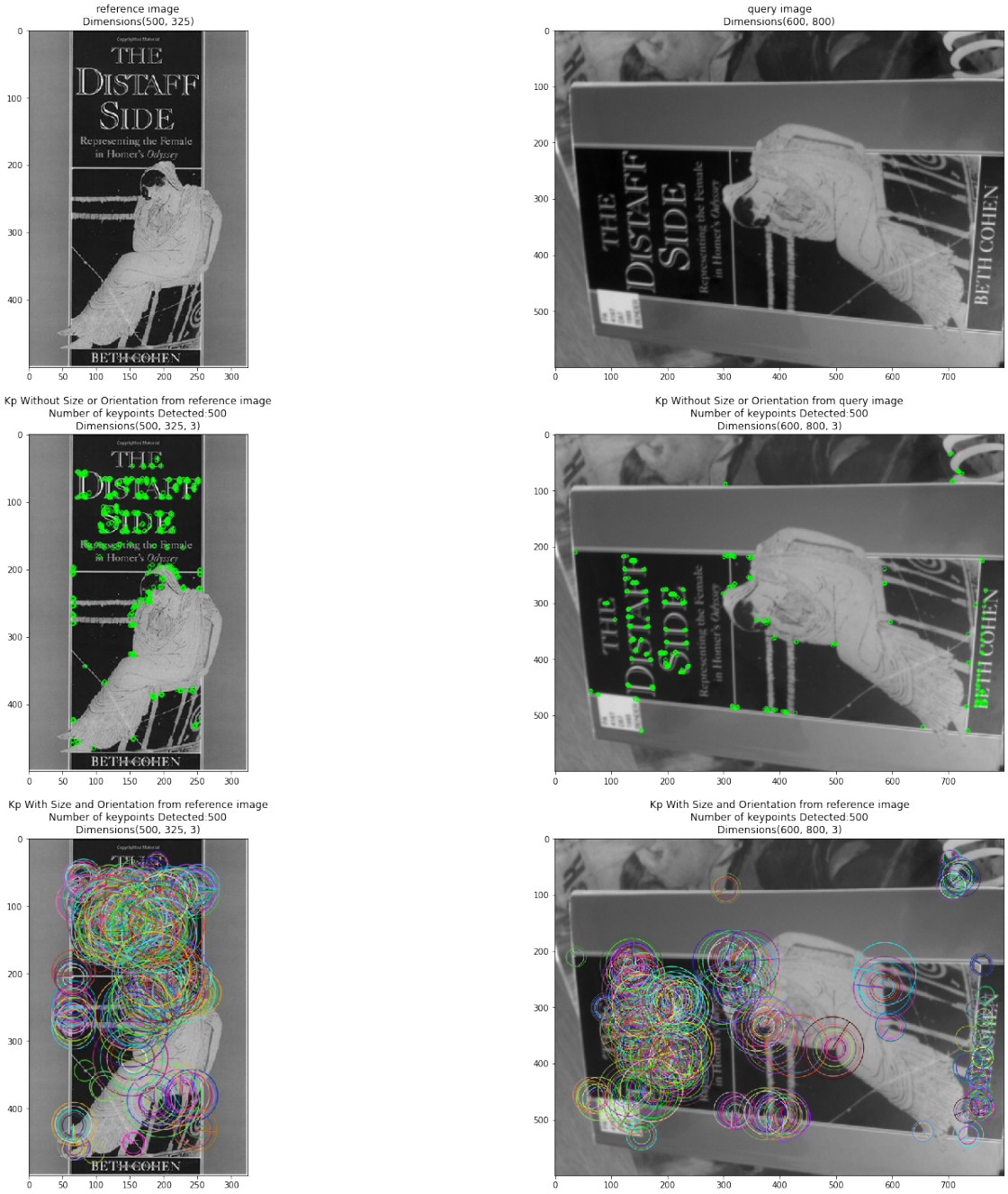
2. [Detect features] Create opencv ORB feature extractor by `orb=cv2.ORB_create()`. Then you can detect keypoints by `kp = orb.detect(img,None)`, and compute descriptors by `kp, des = orb.compute(img, kp)`. You need to do this for each image, and then you can use `cv2.drawKeypoints()` for visualization.

3. [Match features] As ORB is a binary feature, you need to use HAMMING distance for matching, e.g., `bf = cv2.BFMatcher(cv2.NORM_HAMMING)`. Then you are requried to do KNN matching (k=2) by using `bf.knnMatch()`. After that, you are required to use "ratio_test" to find good matches. By default, you can set `ratio=0.8`.

4. [Plot and analyze] You need to visualize the matches by using the `cv2.drawMatches()` function. Also you can change the ratio values, parameters in `cv2.ORB_create()`, and distance

functions in `cv2.BFMatcher()`. Please discuss how these changes influence the match numbers.

Secton 1.1: The following figures will demonstrate the load image process and how the keeypoints been detected for both reference image and query image.



What I observed:

ORB is basically the combination of two algorithms involved FAST and BRIEF where FAST stands

for Features from Accelerated Segments Test whereas BRIEF stands for Binary Robust Independent Elementary Features/ ORB performs as well as SIFT on the task of feature detection (and is better than SURF) while being almost two orders of magnitude faster. ORB detector first uses FAST algorithm, this FAST algorithm finds the key points then applies Harries corner measure to find top N numbers of key points among them, this algorithm quickly selects the key points by comparing the distinctive regions like the intensity variations. The figure above has shown the key points drawn in both the reference image and query image. The simple explanation of key points is that when the human will see the image at that time the features he or she notices in that image, in the similar way when the machine reads the image it sees some points of interest known as Key points. However, FAST features do not have an orientation component and multiscale features. So orb algorithm uses a multi-scale image pyramid. Once the orb has created a pyramid it uses the fast algorithm to detect key points in the image. By detecting key points at each level orb is effectively locating key points at a different scale.

```
--------------------------------------------------------------------------------
----------------------
```

```
Descriptors of reference image [[122  56 163 … 100 166  82]
 [123 115 152 …  30 138  89]
 [114  66 202 … 112 102  10]
 …
 [109 213  49 …  27 196  39]
 [208  30  94 … 128  15 217]
 [ 94 107 202 … 243 171  83]]
Descriptors of query image [[169 125 123 … 254 206 255]
 [118 233 104 …  63  94  78]
 [ 41  48 247 … 128  66 169]
 …
 [252 111 122 … 176 110  74]
 [  3 241 140 …  80 131 253]
 [ 71  54 150 … 100 234   7]]
------------------------------
Shape of descriptor of reference image (500, 32)
Shape of descriptor of query image (500, 32)
```

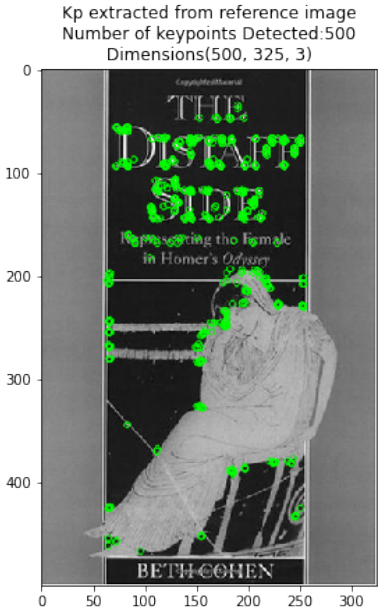
```
--------------------------------------------------------------------------------
----------------------
```

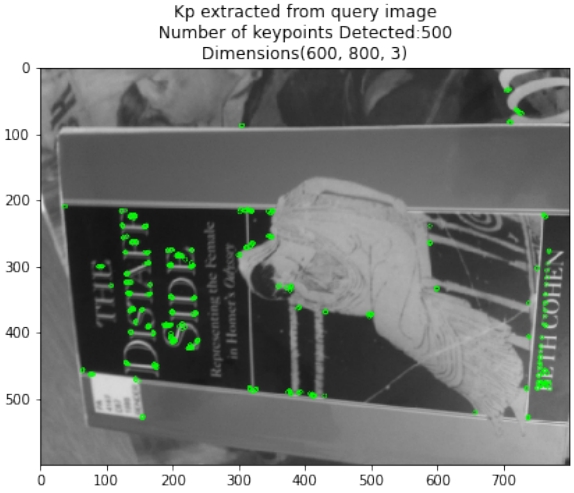
What I observed:

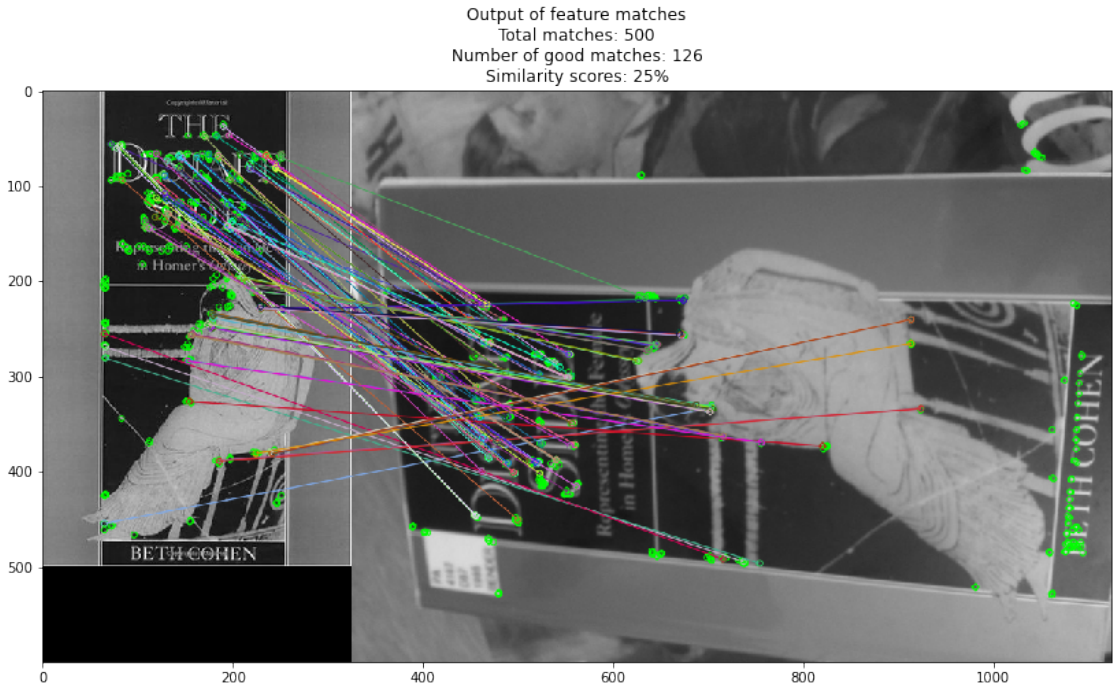
Now the role of the BRIEF algorithm comes, Brief takes all key points found by the fast algorithm and converts them into a binary feature vector so that together they can represent an object. A binary feature vector also known as a binary feature descriptor is a feature vector that only contains 1 and 0. In brief, each keypoint is described by a feature vector which is 128–512 bits string. In

the output shown above, we can see reference image descriptor shape and query image descriptor shape is (500, 32) and (500,32). So, Oriented Fast and Rotated Brief (ORB) detector tries to find 500 features in the image by default, and for each descriptor, it will describe 32 values.

Section 1.2: The following figures will demontrate image matching between two images:



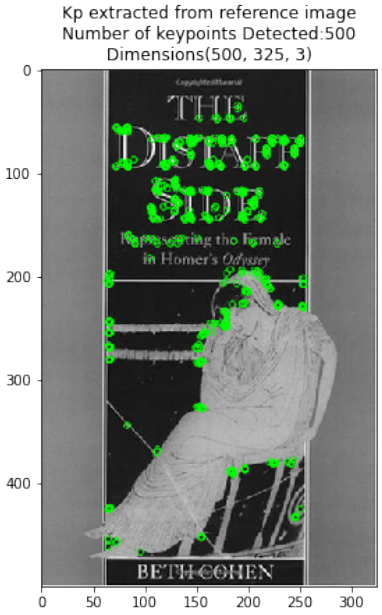


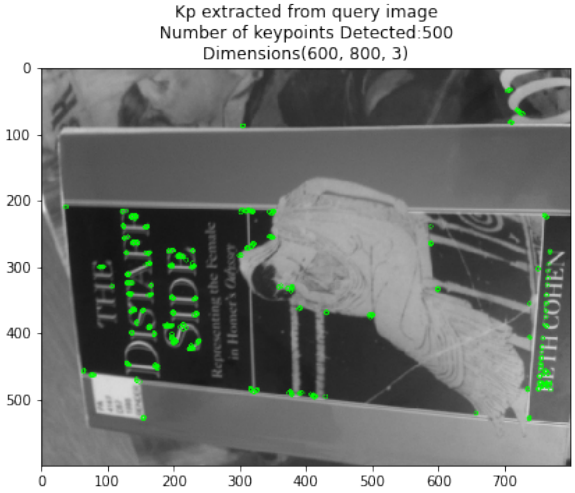


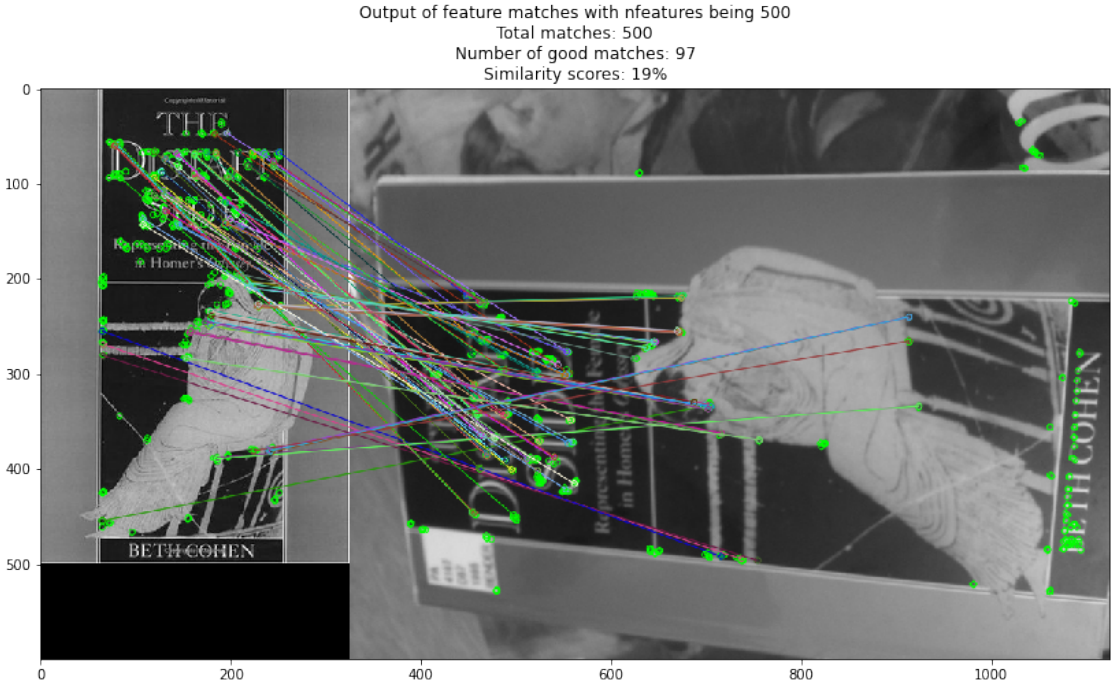
What I observed:

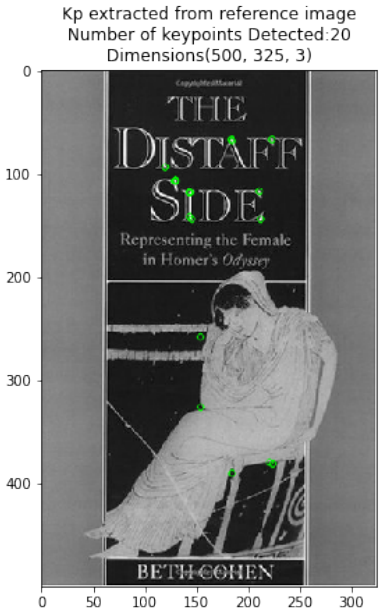
At this point, we already have the features of the reference image and the source image, as well as their descriptors. With these data, we can search for matches between them. One method of matching a feature in the reference image to a feature in the source image is to use the ratio of the distance between the features and return the closest as the best match. For this purpose, Brute Force Matcher is used. Brute Force Matcher takes one descriptor of the first image and matches to all the descriptors of the second image and then it goes to the second descriptor of first image and matches to all the descriptor of the second image and so on. The brute force Matcher function takes two optional parameters. The first one is normType. It specifies the distance measurement to be used. By default, it is cv.NORM_L2. It is good for SIFT, SURF, etc (cv.NORM_L1 is also there). For binary string-based descriptors like ORB, BRIEF, BRISK, etc, cv.NORM_HAMMING should be used, which used Hamming distance as measurement. Now, We can use a Brute Force Matcher to match these descriptors together, then use the ratio test to select good matches from all matches and find how many similarities we are getting. As the figure shows above, when the ratio was set to be 0.8, the good matches selected from all matches were 126 and the similarity score is 25%.

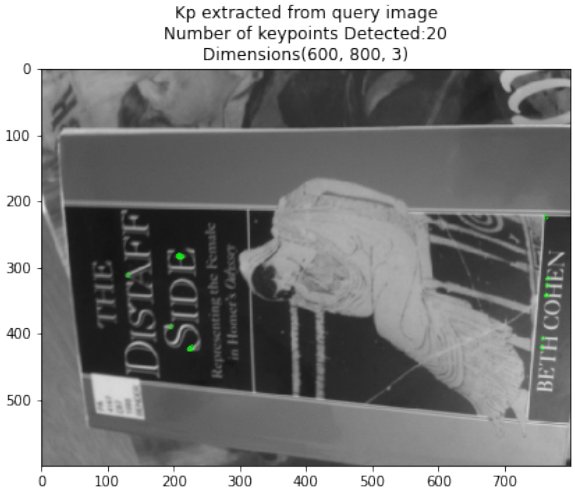
The following figures will demonstrate how the change of nfeatures which is one of the parameters from cv2.ORB_create() have effects on feature matching

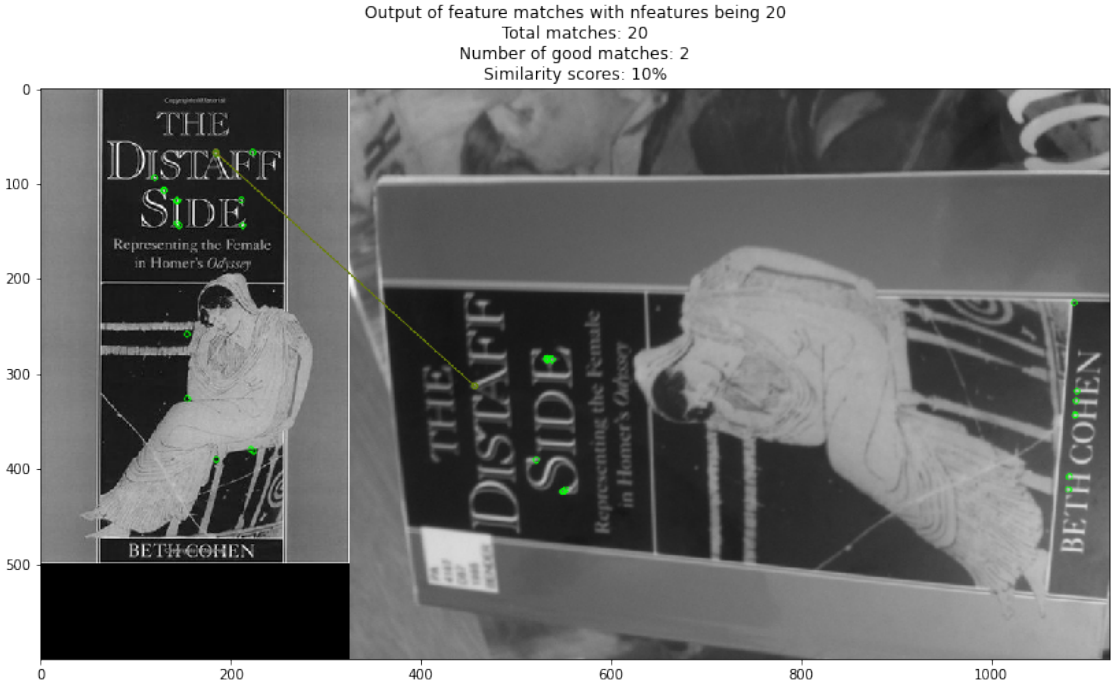


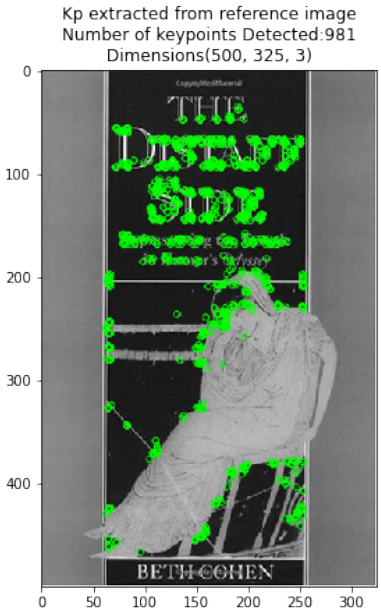


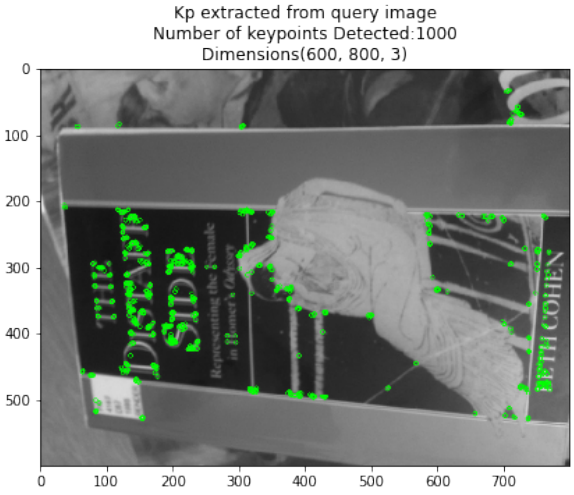


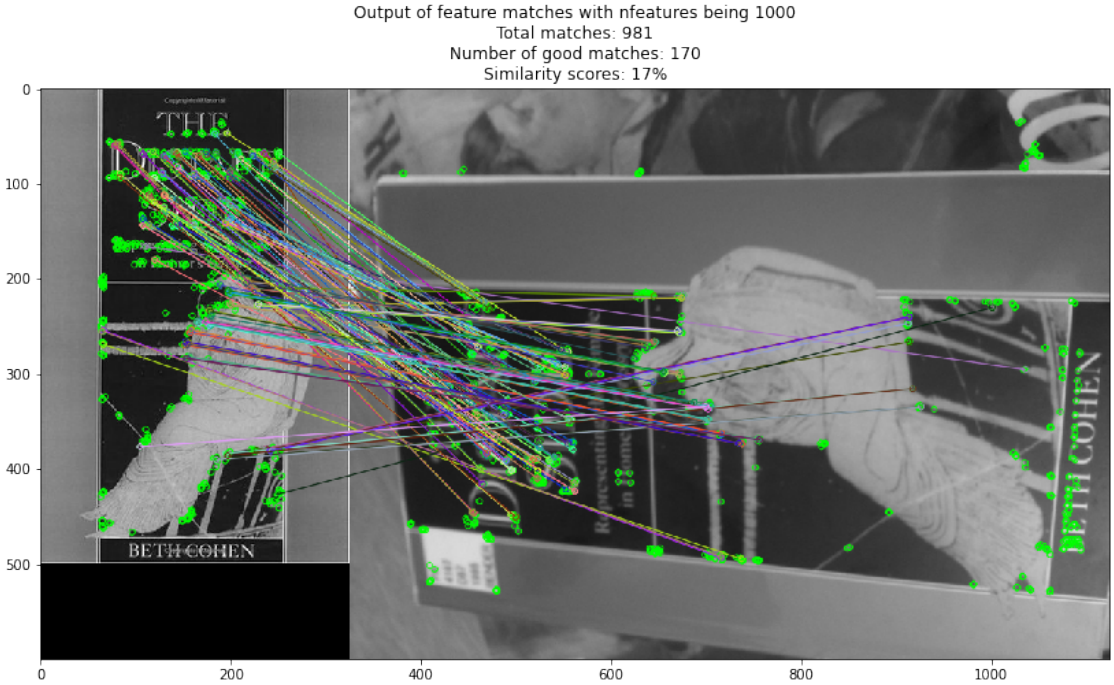

Kp extracted from reference image
Number of keypoints Detected:20
Dimensions(500, 325, 3)
THE DISTAFF SIDE
Representing the Female in Homer's Odyssey
BETH COHEN


Kp extracted from query image
Number of keypoints Detected:20
Dimensions(600, 800, 3)
THE DISTAFF SIDE
BETH COHEN


Output of feature matches with nfeatures being 20
Total matches: 20
Number of good matches: 2
Similarity scores: 10%
THE DISTAFF SIDE
Representing the Female in Homer's Odyssey
BETH COHEN


Kp extracted from reference image
Number of keypoints Detected:981
Dimensions(500, 325, 3)
BETH COHEN

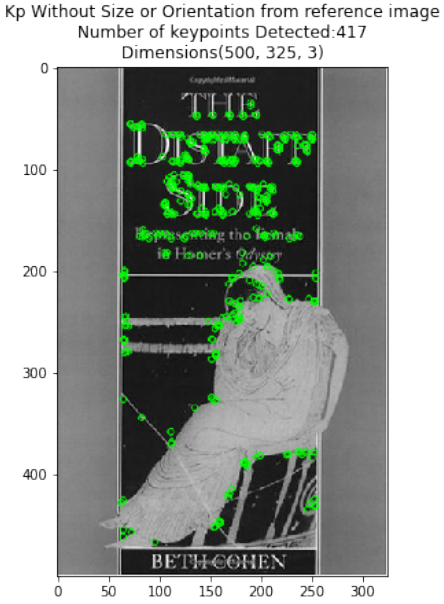

Kp extracted from query image
Number of keypoints Detected:1000
Dimensions(600, 800, 3)
BETH COHEN

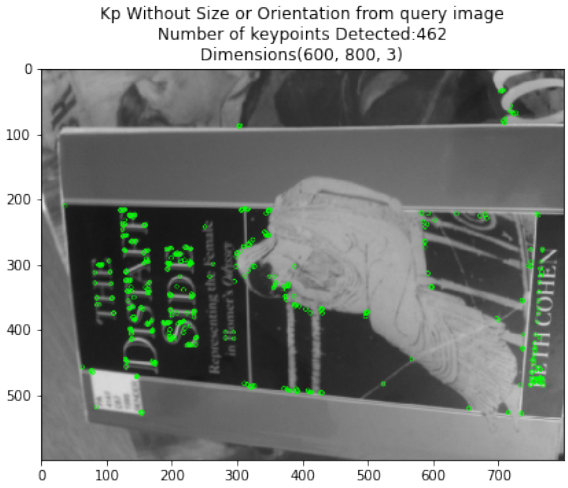

Output of feature matches with nfeatures being 1000
Total matches: 981
Number of good matches: 170
Similarity scores: 17%
BETH COHEN
BETH COHEN

What I observed:

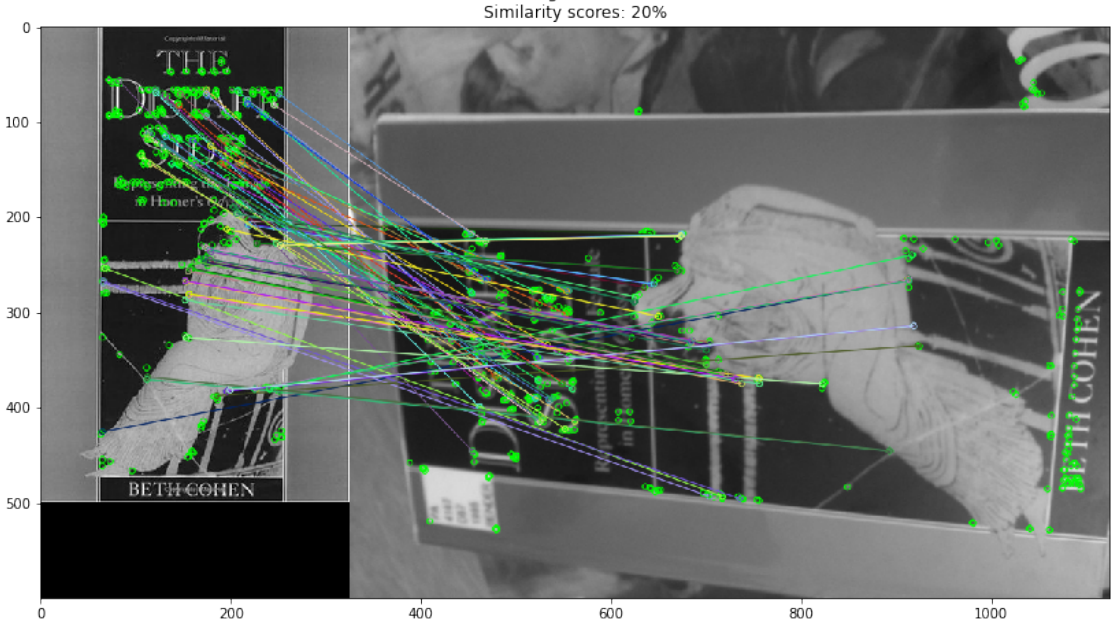
One parameter from cv2.ORB_create() is nfeatures which describes the maximum number of features to retain. Different similarity scores of feature matches can be obtained by varying the number of nfeatures. As the figures show above, three different nfeatures were used which are 500, 20, and 1000, and the similarity scores for each of them are 19%, 10%, and 17% respectively. Hence, a conclusion can be found by analyzing the difference between the 3 figures: the number of nfeatures around 500 will provide the highest similarity for feature matching.

The following figures will demonstrate how the change of scaleFactor which is one of the parameters from cv2.ORB_create() have effects on feature matching

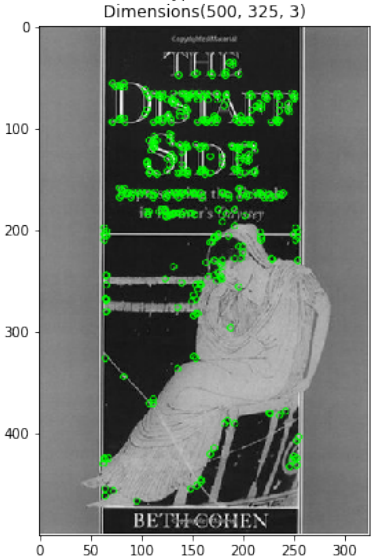




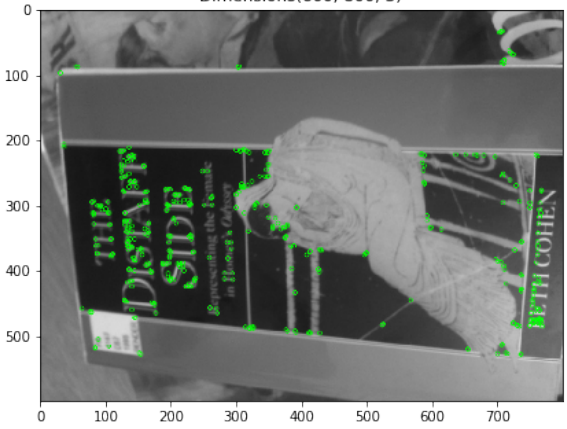
Key points of referenced image with scaleFactor being 1.5
Total matches: 417
Number of good matches: 84
Similarity scores: 20%



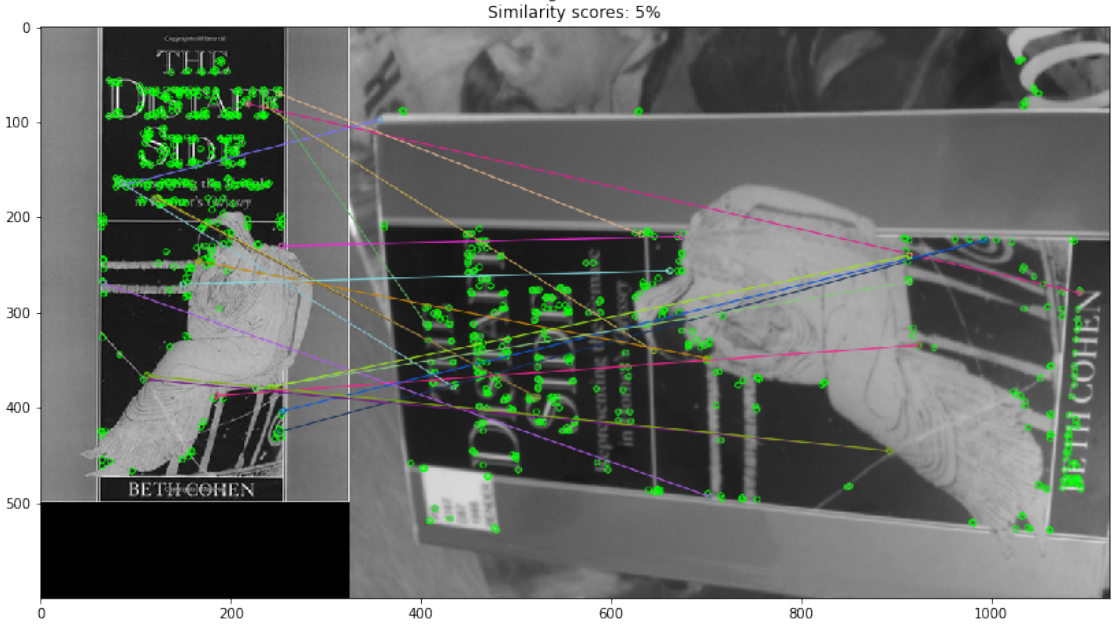
Kp Without Size or Orientation from reference image
Number of keypoints Detected:416
Dimensions(500, 325, 3)



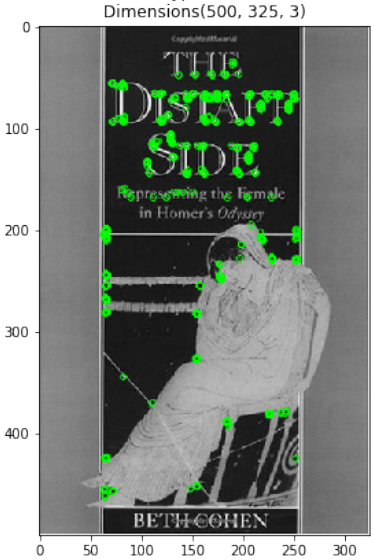
Kp Without Size or Orientation from query image
Number of keypoints Detected:446
Dimensions(600, 800, 3)



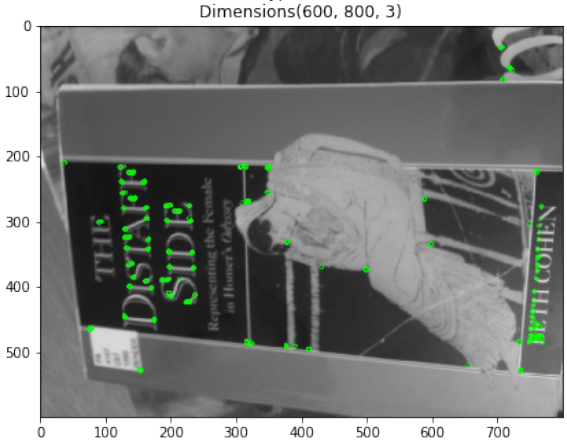
Key points of referenced image with scaleFactor being 2.0
Total matches: 416
Number of good matches: 19
Similarity scores: 5%

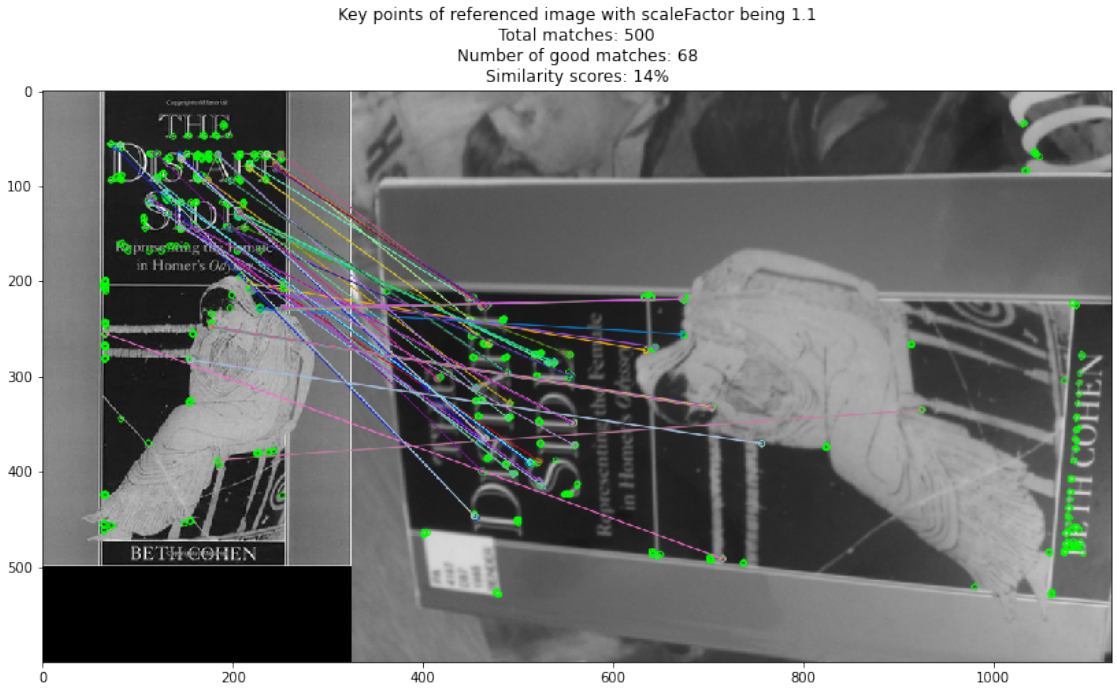


Kp Without Size or Orientation from reference image
Number of keypoints Detected:500
Dimensions(500, 325, 3)



Kp Without Size or Orientation from query image
Number of keypoints Detected:500
Dimensions(600, 800, 3)

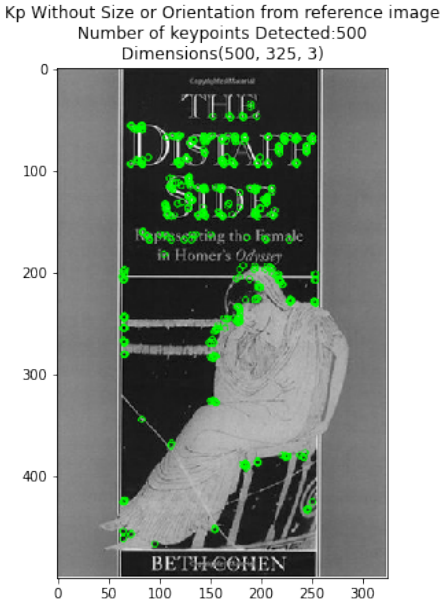


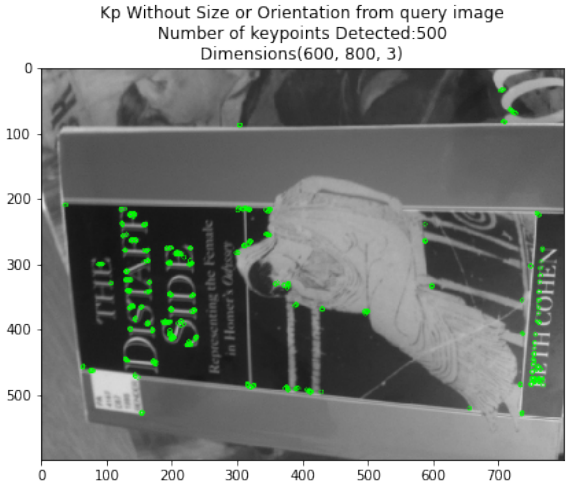


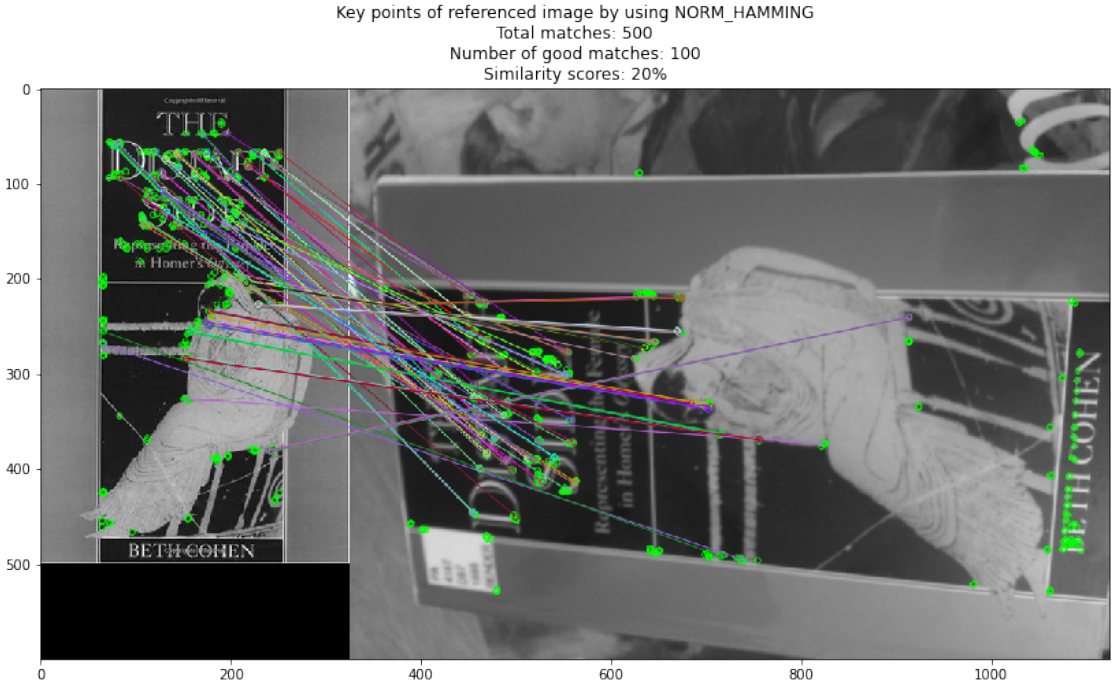
What I observed:

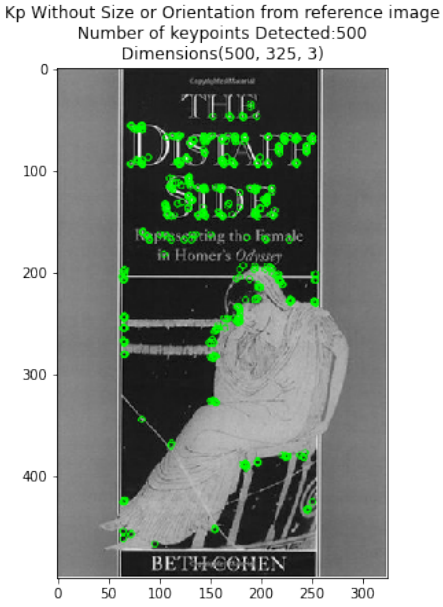
One parameter from cv2.ORB_create() is scaleFactor which describes the Pyramid decimation ratio, greater than 1. scaleFactor==2 means the classical pyramid, where each next level has 4x fewer pixels than the previous, but such a big scale factor will degrade feature matching scores dramatically. On the other hand, too close to 1 scale factor will mean that to cover a certain scale range you will need more pyramid levels and so the speed will suffer. Different similarity scores for feature matches were obtained by varying the scaleFactor. As the figures show above, two different scaleFactor were used which are 1.5 and 2 and 1.1, feature matching scores are 20%. 5% and 14% respectively. These scores verified our assumptions above.

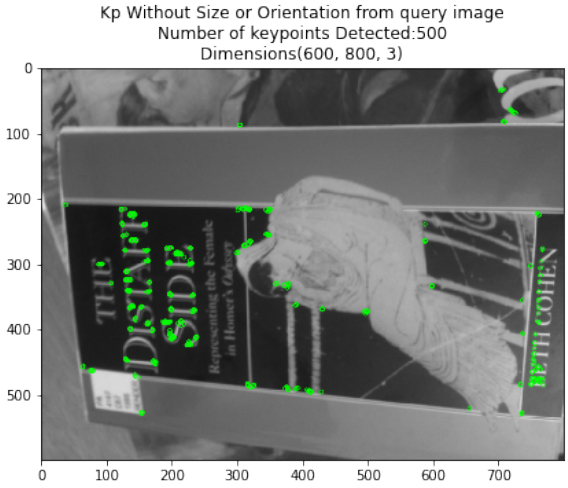
The following figures will demonstrate how the change of matching strategy have effects on image matching

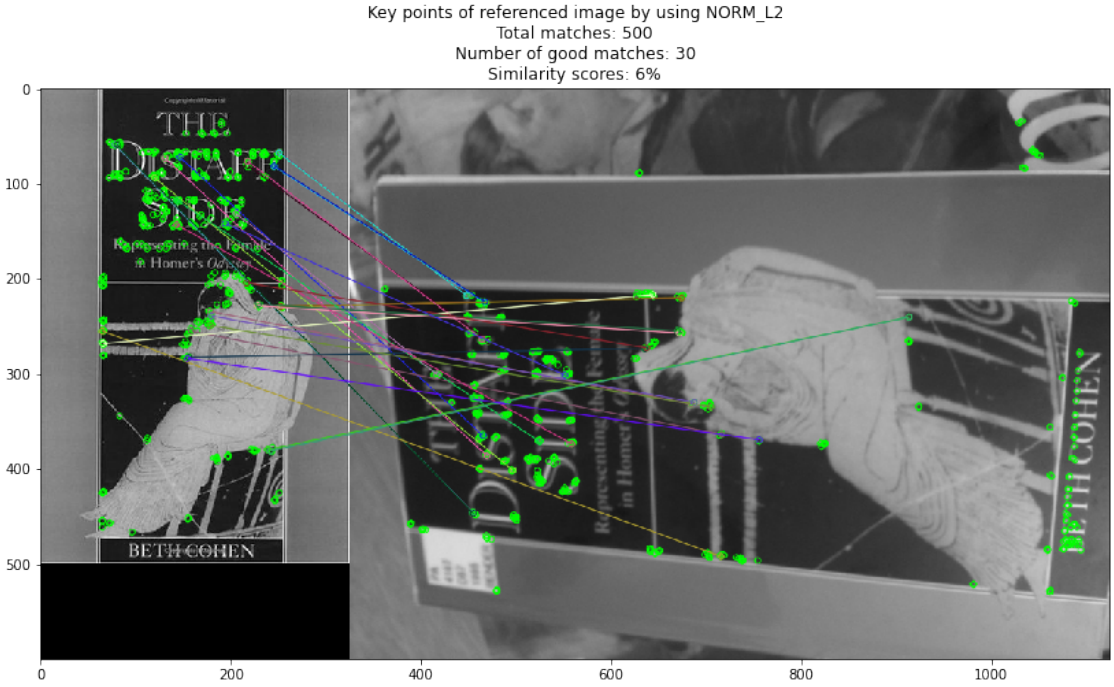

Kp Without Size or Orientation from reference image
Number of keypoints Detected:500
Dimensions(500, 325, 3)
THE DISTANT SIDE
Representing the Female in Homer's Odyssey
BETH COHEN


Kp Without Size or Orientation from query image
Number of keypoints Detected:500
Dimensions(600, 800, 3)
THE DISTANT SIDE
BETH COHEN


Key points of referenced image by using NORM_HAMMING
Total matches: 500
Number of good matches: 100
Similarity scores: 20%
BETH COHEN


Kp Without Size or Orientation from reference image
Number of keypoints Detected:500
Dimensions(500, 325, 3)
THE DISTANT SIDE
Representing the Female in Homer's Odyssey
BETH COHEN


Kp Without Size or Orientation from query image
Number of keypoints Detected:500
Dimensions(600, 800, 3)
THE DISTANT SIDE
BETH COHEN


Key points of referenced image by using NORM_L2
Total matches: 500
Number of good matches: 30
Similarity scores: 6%
THE DISTANT SIDE
BETH COHEN

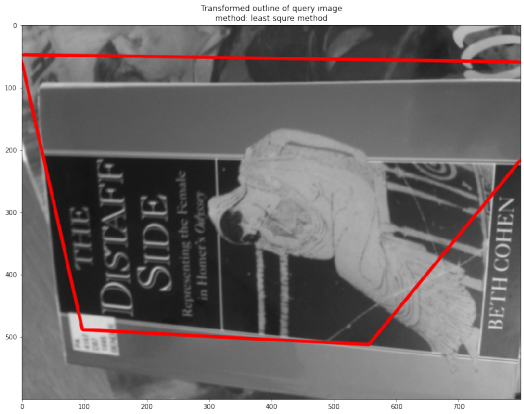
What I observed:

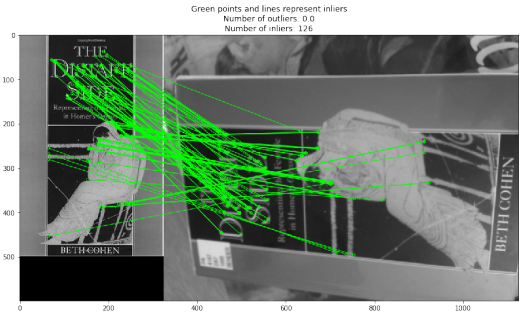
Brute-Force matcher takes the descriptor of one feature in the first set and is matched with all other features in the second set using some distance calculation. And the closest one is returned. The first parameter in cv2.BFMatcher() specifies the distance measurement to be used. As the figures show above, two different distance functions are used which are cv2.NORM_L2 and cv2.NORM_HAMMING and their feature matching scores are 19% and 6% respectively.

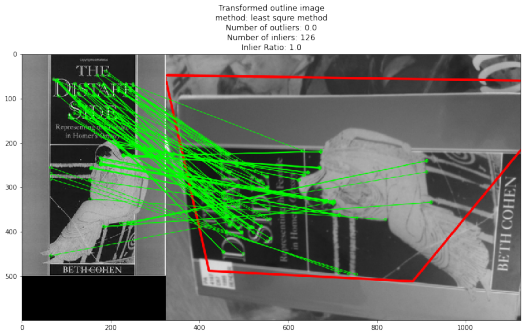
---

3. Estimate a homography transformation based on the matches, using `cv2.findHomography()`. Display the transformed outline of the first reference book cover image on the query image, to see how well they match.
    - We provide a function `draw_outline()` to help with the display, but you may need to edit it for your needs.
    - Try the 'least squre method' option to compute homography, and visualize the inliers by using `cv2.drawMatches()`. Explain your results.
    - Again, you don't need to compare results numerically at this stage. Comment on what you observe visually.

The following figures will demontrate how to use the 'least squre method' option to compute homography


Transformed outline of query image
method: least squre method
THE DISTAFF SIDE
Representing the Female in Homer's Odyssey
BETH COHEN


Green points and lines represent inliers
Number of outliers: 0.0
Number of inliers: 126
BETH COHEN


Transformed outline image
method: least squre method
Number of outliers: 0.0
Number of inliers: 126
Inlier Ratio: 1.0
BETH COHEN

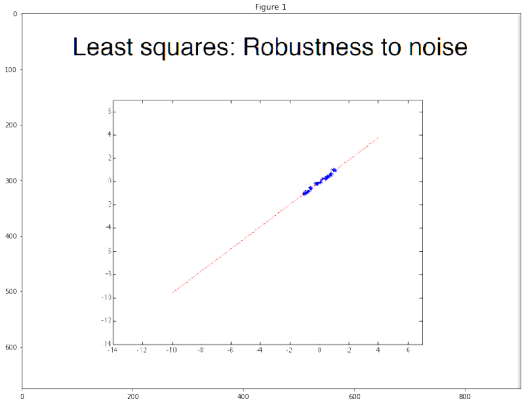


Figure 1
Least squares: Robustness to noise

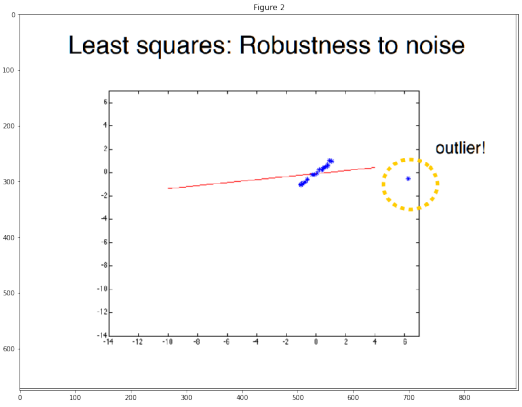
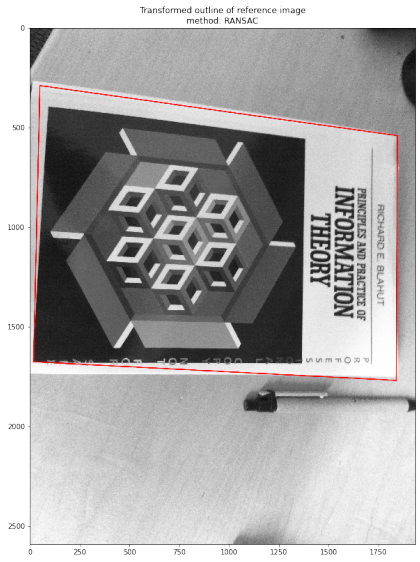


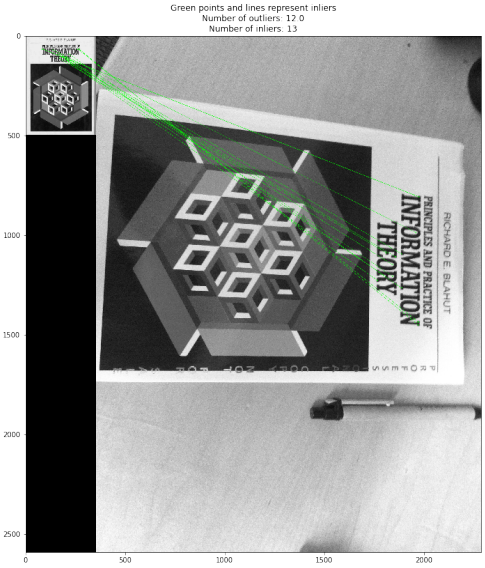
Figure 2
Least squares: Robustness to noise
outlier!

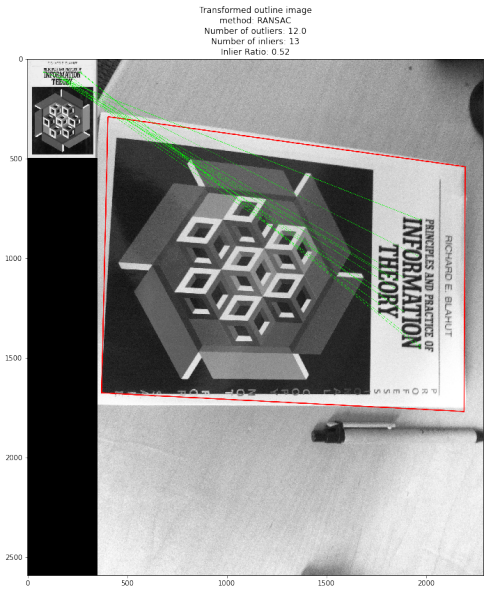
What I observed:

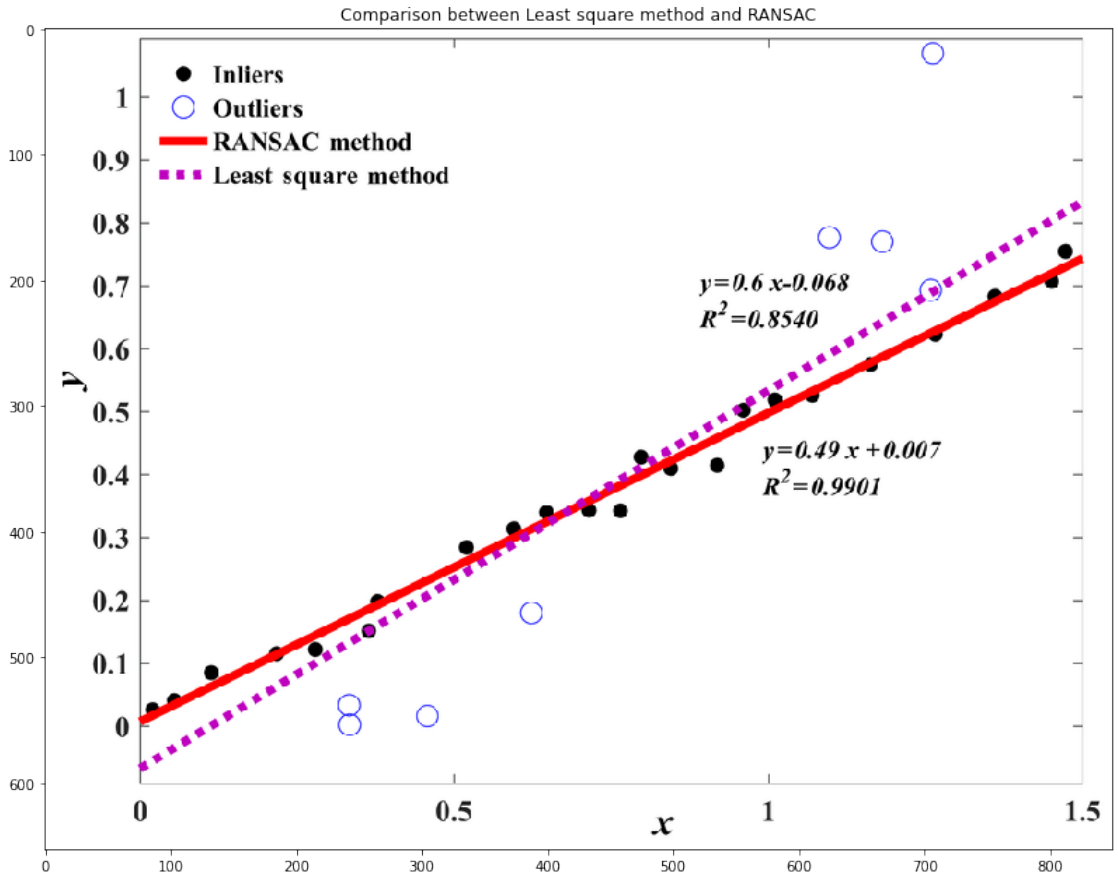
In the figures shown above, the least square method was used to compute homography. The least-squares method is a statistical procedure to find the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve. Least squares regression is used to predict the behavior of dependent variables and it provides the overall rationale for the placement of the line of best fit among the data points being studied. The least-squares methods can work well when the data are noisy. Figure 1 shows the result of a simulation where we used the least square to fit all the red points with a line. However, the least square is in general not very robust to the presence of outliers as the result shown in figure 2 demonstrates. The outlier marked in blue significantly alters the quality of the fitting process. This is also reflected in our book image matching. Because the Least square method is penalized by the squared distances to the line, the outliers can have a huge penalty and this can drive the estimated solution away from the correct solution, that's why the red box cannot exactly cover the query book shape.

The following figures will show the result of trying the RANSAC option to compute homography.


Transformed outline of reference image
method: RANSAC
PRINCIPLES AND PRACTICE OF
INFORMATION
THEORY
RICHARD E. BLAHUT


Green points and lines represent inliers
Number of outliers: 12.0
Number of inliers: 13
PRINCIPLES AND PRACTICE OF
INFORMATION
THEORY
RICHARD E. BLAHUT


Transformed outline image
method: RANSAC
Number of outliers: 12.0
Number of inliers: 13
Inlier Ratio: 0.52
PRINCIPLES AND PRACTICE OF
INFORMATION
THEORY
RICHARD E. BLAHUT

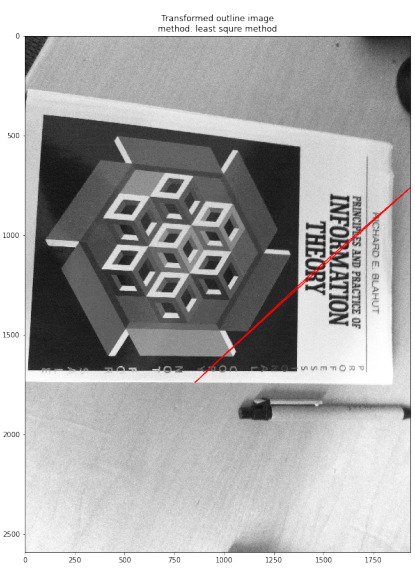


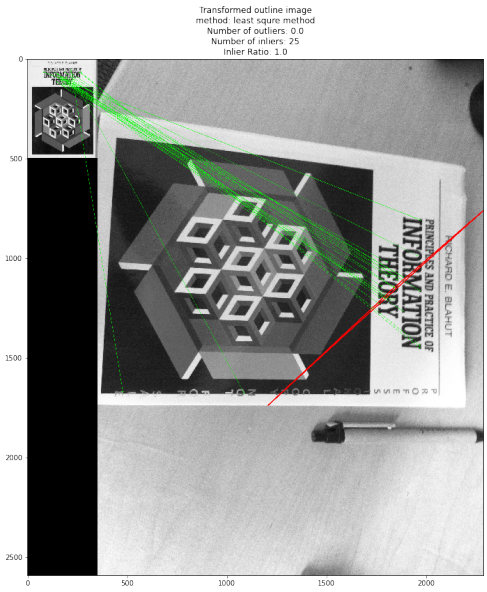
What I observed:

The figures shown above demonstrated the effect of using RANSAC to compute homography. The RANSAC algorithm is a robust data fitting algorithm. Its basic assumption is that a set of data contains a sample data set composed of correct data and a small amount of abnormal data. Iteratively eliminates the process of erroneous data. Applying the RANSAC algorithm to the feature point matching screening can effectively eliminate the error matching points. As we can see from the figures shown above, although the number of inliers is less than the least square method, RANSAC has matched the query book with more accuracy as the red lines have mostly fitted the book cover. RANSAC is an easily implementable method to estimate a model that often works well in practice. However, there are many parameters to tune; it may take a long time to get to the accuracy that you need and requires the inlier to outlier ratio to be reasonable. Often,RANSAC doesn't work well if the inlier/outlier ratio is greater than 50%.

---

6. Finally, try matching several different image pairs from the data provided, including at least one success and one failure case. For the failure case, test and explain what step in the feature matching has failed, and try to improve it. Display and discuss your findings.
    1. Hint 1: In general, the book covers should be the easiest to match, while the landmarks are the hardest.
    2. Hint 2: Explain why you chose each example shown, and what parameter settings were used.
    3. Hint 3: Possible failure points include the feature detector, the feature descriptor, the matching strategy, or a combination of these.

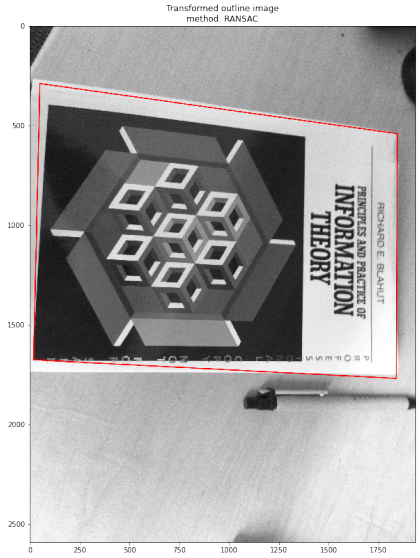


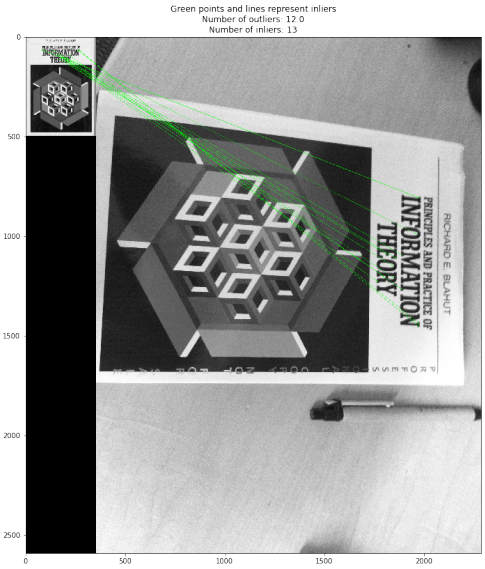


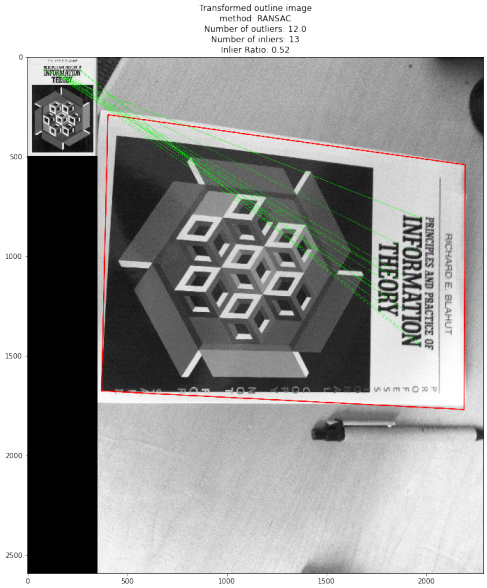
What I observed:

The above Figures demonstrated one failure case from image matching. The maximum number of features to retain has been set to be 500, this is one parameter from cv.ORB_create(). The matching strategy used here is the least square method. As the figures above show, the image matching failed when using ORB to detect image features and using the least square method for

image matching. This method has a high sensitivity to outliers and is likely to overfit data.

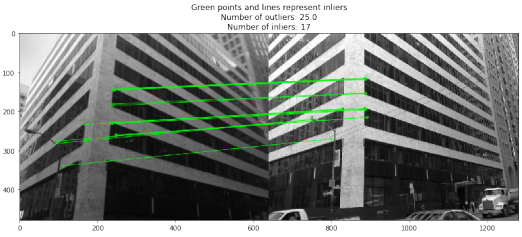


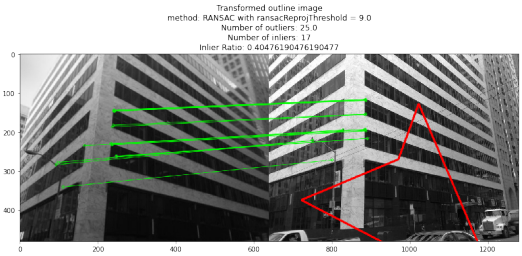


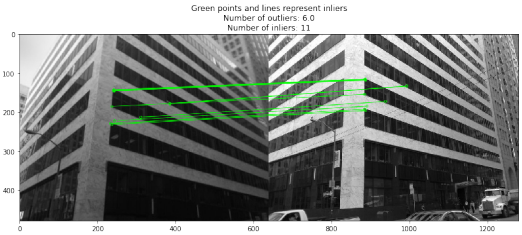


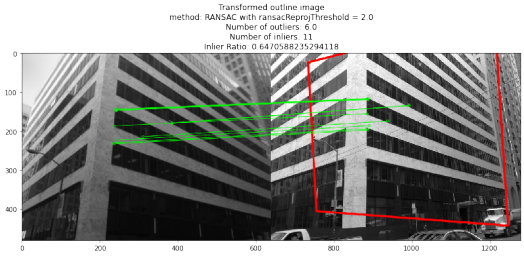
What I observed:

The above figures demonstrated a solution to overcome the previous image matching failure. This algorithm used RANSAC instead of the least square method which can estimate the parameters with a high degree of accuracy even when a significant number of outliers are present in the data set. One drawback of using RANSAC is that there is no upper bound on the time it takes to compute these parameters. When the number of iterations computed is limited the solution obtained may not be optimal, and it may not even fit the data in a good way. In this way RANSAC offers a trade-off; by computing a greater number of iterations the probability of a reasonable model being produced is increased.









What I observed:

In the above figures, the maximum number of features to be extracted is set to be 500 and 10000 respectively, and clearly, the image with 10000 feature points is more accurate. The features to be extracted can be broadly classified into two categories, namely, local and global features. Global features are good, in the context of image retrieval, Object recognition to describe the information of the image but they end up mixing the information of foreground and background. These features are well suited for most of the shape and texture descriptions, but they are affected by partial occlusion and clutter. Thus they are not suitable for the localization of objects spatially. Local features are invariant, they differ from immediate neighbors and allow matching local structures between the images efficiently. Local features may be points, edges, lines, segments, or objects that have specific structures in the image. These features are also referred to as corner points, key points, or feature points. Therefore, more local features would give us more information about the image, hence, more accurate image matching would be.

3 Question 2: What am I looking at? (40%)

In this question, the aim is to identify an "unknown" object depicted in a query image, by matching it to multiple reference images, and selecting the highest scoring match. Since we only have one reference image per object, there is at most one correct answer. This is useful for example if you want to automatically identify a book from a picture of its cover, or a painting or a geographic location from an unlabelled photograph of it.

The steps are as follows:

1. Select a set of reference images and their corresponding query images.
    1. Hint 1: Start with the book covers, or just a subset of them.
    2. Hing 2: This question can require a lot of computation to run from start to finish, so cache intermediate results (e.g. feature descriptors) where you can.
2. Choose one query image corresponding to one of your reference images. Use RANSAC to match your query image to each reference image, and count the number of inlier matches found in each case. This will be the matching score for that image.
3. Identify the query object. This is the identity of the reference image with the highest match score, or "not in dataset" if the maximum score is below a threshold.
4. Repeat steps 2-3 for every query image and report the overall accuracy of your method (that is, the percentage of query images that were correctly matched in the dataset). Discussion of results should include both overall accuracy and individual failure cases.
    1. Hint 1: In case of failure, what ranking did the actual match receive? If we used a "top-k" accuracy measure, where a match is considered correct if it appears in the top k match scores, would that change the result?

Section 1:

10 query images and their corresponding reference images have been selected from book cover image set.


```
results:

Query images matched to Reference images with highest matching scores and same
title:
```


```
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [78, 4]
Highest matching score is: 78
------------------------------
Query image 002 matched to reference image 002
Number of inlier matches found in each case: [55, 6, 5, 4, 4, 4, 0, 0]
Highest matching score is: 55
------------------------------
Query image 003 matched to reference image 003
Number of inlier matches found in each case: [84, 5, 5, 4, 4]
Highest matching score is: 84
------------------------------
Query image 004 matched to reference image 004
Number of inlier matches found in each case: [28, 4, 4, 4, 0, 0]
Highest matching score is: 28
------------------------------
Query image 005 matched to reference image 005
Number of inlier matches found in each case: [47, 4, 4, 4, 0]
Highest matching score is: 47
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [74]
Highest matching score is: 74
------------------------------
Query image 007 matched to reference image 007
Number of inlier matches found in each case: [40, 4, 0, 0]
Highest matching score is: 40
------------------------------
Query image 009 matched to reference image 009
Number of inlier matches found in each case: [60, 9, 4, 0]
Highest matching score is: 60
------------------------------
Query image 010 matched to reference image 010
Number of inlier matches found in each case: [87, 4, 4]
Highest matching score is: 87
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------


Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------
```

```
Query image 008 is not in the dataset
Number of inlier matches found in each case: [10, 0]
The highest matching score 10 is less than threshold 15
------------------------------


--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 10
Total number of matched image is: 9
Total number of query image macthed wrong with reference is : 0
Total number of query image not in the dataset is : 1
Overall accuracy is: 0.9


--------------------------------------------------------------------------------
----------------------
```

Section 2:

10 query images and their corresponding reference images have been selected from landmarks image set.

```
--------------------------------------------------------------------------------
----------------------

results:

Query images matched to Reference images with highest matching scores and same
title:
--------------------------------------------------------------------------------
----------------------

Query image 000 matched to reference image 000
Number of inlier matches found in each case: [17, 8, 6, 5, 5, 5, 4, 4, 4, 4]
Highest matching score is: 17
------------------------------
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [14, 7, 6, 6, 4, 4, 0, 0]
Highest matching score is: 14
------------------------------
Query image 004 matched to reference image 004
Number of inlier matches found in each case: [7, 7, 6, 6, 5, 5, 5, 4, 4, 4]
Highest matching score is: 7
```

```
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [19, 6, 5, 4, 4, 4, 4, 4]
Highest matching score is: 19
------------------------------
Query image 007 matched to reference image 007
Number of inlier matches found in each case: [13, 7, 6, 5, 4, 4, 4, 4, 4, 4]
Highest matching score is: 13
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [21, 5, 5, 5, 4, 4, 4, 4, 0]
Highest matching score is: 21
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------

Query image 002 matched with wrong reference image 006
Number of inlier matches found in each case: [9, 8, 7, 6, 5, 5, 4, 4]
The actual rank of matching is: 3
------------------------------
Query image 003 matched with wrong reference image 008
Number of inlier matches found in each case: [6, 6, 5, 5, 4, 4, 4, 4]
The actual rank of matching is: 7
------------------------------
Query image 005 matched with wrong reference image 009
Number of inlier matches found in each case: [7, 6, 5, 5, 4, 4, 4, 4, 4]
The actual rank of matching is: 3
------------------------------
Query image 009 matched with wrong reference image 008
Number of inlier matches found in each case: [18, 10, 5, 5, 5, 4, 4, 4, 4]
The actual rank of matching is: 2
------------------------------


Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------




--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 10
```

```
Total number of matched image is: 6
Total number of query image macthed wrong with reference is : 4
Total number of query image not in the dataset is : 0
Overall accuracy is: 0.6
```

Section 3:

Top-k accuracy measurement method has been used in order to improve landmamrks image matching accuracy.

```
results:

Query images matched to Reference images by using top 3 method:
```

```
Query image 000 matched to reference image 000
Number of inlier matches found in each case: [17, 8, 6, 5, 5, 5, 4, 4, 4, 4]
Highest matching score is: 17
------------------------------
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [14, 7, 6, 6, 4, 4, 0, 0]
Highest matching score is: 14
------------------------------
Query image 002 matched to reference image 002
Number of inlier matches found in each case: [9, 8, 7, 6, 5, 5, 4, 4]
Highest matching score is: 7
------------------------------
Query image 004 matched to reference image 004
Number of inlier matches found in each case: [7, 7, 6, 6, 5, 5, 5, 4, 4, 4]
Highest matching score is: 7
------------------------------
Query image 005 matched to reference image 005
Number of inlier matches found in each case: [7, 6, 5, 5, 4, 4, 4, 4, 4]
Highest matching score is: 5
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [19, 6, 5, 4, 4, 4, 4, 4]
Highest matching score is: 19
```

```
------------------------------
Query image 007 matched to reference image 007
Number of inlier matches found in each case: [13, 7, 6, 5, 4, 4, 4, 4, 4, 4]
Highest matching score is: 13
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [21, 5, 5, 5, 4, 4, 4, 4, 0]
Highest matching score is: 21
------------------------------
Query image 009 matched to reference image 009
Number of inlier matches found in each case: [18, 10, 5, 5, 5, 4, 4, 4, 4]
Highest matching score is: 10
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------

Query image 003 matched with wrong reference image 008
Number of inlier matches found in each case: [6, 6, 5, 5, 4, 4, 4, 4]
The actual rank of matching is: 7
------------------------------


Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------




--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 10
Total number of matched image is: 9
Total number of query image macthed wrong with reference is : 1
Total number of query image not in the dataset is : 0
Overall accuracy is: 0.9



--------------------------------------------------------------------------------
----------------------
```

The explanation of what I have done:

One application of feature matching is image retrieval. The goal of image retrieval is, given a query image of an object, to find all images in a database containing the same object, and return the results in ranked order. In section 1, 10 query images and their corresponding reference images have been selected from the book cover image set, the algorithm has set the number of feature points extracted to be 500 and the threshold to be 10. The Overall accuracy is 90%, meaning 90% of query images can be successfully retrieved. Similarly, in section 2, 10 query images and their corresponding reference images have been selected from the landmarks image set and the number of feature points was set to be 10000 this time, the threshold was 4. The algorithm report the overall accuracy for the landmarks image set is only 60%. For each failure case presented in the book cover image as well as in the image of the landmark, the actual rank of matching has shown which we can see that in some cases, the correct image matching may sit in the second or third rank. By observing this fact, the TOP K measurement has been used in the algorithm to improve matching accuracy. Top k accuracy is when you measure how often your predicted class(actual matching) falls in the top k values of your softmax distribution. Top-k measurement provides us the ability to find the correct image matching when the actual matching score is not significantly small. Section 3 has clearly demonstrated how the overall accuracy has been improved when the top-k measure has been used. As we can see from the results, the overall accuracy has been improved from 60% to 90%.

---

5. Choose some extra query images of objects that do not occur in the reference dataset. Repeat step 4 with these images added to your query set. Accuracy is now measured by the percentage of query images correctly identified in the dataset, or correctly identified as not occurring in the dataset. Report how accuracy is altered by including these queries, and any changes you have made to improve performance.

Section 2.1:

10 query images and their corresponding reference images have been selected from book cover image set, moreover, 2 extra images from landmarks and museum_paintings have been selected and added to the book cover query set.

`results:`

`Query images matched to Reference images by using top 3 method:`

```
--------------------------------------------------------------------------------
----------------------

Query image 001 matched to reference image 001
Number of inlier matches found in each case: [306, 9, 8, 7, 6, 6, 6, 5, 5, 5]
Highest matching score is: 306
------------------------------
Query image 002 matched to reference image 002
Number of inlier matches found in each case: [191, 8, 8, 7, 6, 6, 6, 5, 4, 4]
Highest matching score is: 191
------------------------------
Query image 003 matched to reference image 003
Number of inlier matches found in each case: [359, 6, 6, 6, 5, 5, 5, 5, 4, 4]
Highest matching score is: 359
------------------------------
Query image 004 matched to reference image 004
Number of inlier matches found in each case: [105, 14, 8, 6, 6, 6, 6, 6, 5, 4]
Highest matching score is: 105
------------------------------
Query image 005 matched to reference image 005
Number of inlier matches found in each case: [385, 6, 6, 6, 5, 5, 5, 4, 4, 4]
Highest matching score is: 385
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [808, 6, 6, 6, 5, 5, 5, 5, 5, 4]
Highest matching score is: 808
------------------------------
Query image 007 matched to reference image 007
Number of inlier matches found in each case: [131, 13, 12, 11, 9, 9, 8, 7, 6, 6]
Highest matching score is: 131
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [55, 7, 7, 6, 6, 6, 6, 5, 4, 4]
Highest matching score is: 55
------------------------------
Query image 009 matched to reference image 009
Number of inlier matches found in each case: [305, 9, 7, 6, 6, 6, 6, 5, 5, 4]
Highest matching score is: 305
------------------------------
Query image 010 matched to reference image 010
Number of inlier matches found in each case: [792, 7, 6, 6, 5, 5, 5, 4, 4, 4]
Highest matching score is: 792
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------
```

```
Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------

Query image landmarks_0000 is not in the dataset
Number of inlier matches found in each case: [9, 7, 7, 6, 6, 6, 6, 5, 5, 4]
The highest matching score 9 is less than threshold 10
------------------------------
Query image museum_paintings_019 is not in the dataset
Number of inlier matches found in each case: [6, 6, 6, 5, 5, 5, 4, 4, 4, 4]
The highest matching score 6 is less than threshold 10
------------------------------


--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 12
Total number of matched image is: 10
Total number of query image macthed wrong with reference image is : 0
Total number of query image not in the dataset is : 2
Overall accuracy is: 1.0


--------------------------------------------------------------------------------
----------------------
```

Section 2.1:

10 query images and their corresponding reference images have been selected from landmarks image set, and 2 extra images from bookcover and museum_paintings have been selected and added to the landmarks query set.

```
--------------------------------------------------------------------------------
----------------------

results:

Query images matched to Reference images with highest matching scores and same
title:
--------------------------------------------------------------------------------
----------------------
```

```
Query image 000 matched to reference image 000
Number of inlier matches found in each case: [20, 8, 7, 6, 5, 5, 4, 4, 4, 4]
Highest matching score is: 20
------------------------------
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [15, 7, 6, 5, 5, 4, 4]
Highest matching score is: 15
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [19, 6, 5, 5, 4, 4, 4, 4, 0]
Highest matching score is: 19
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [18, 6, 5, 5, 5, 4]
Highest matching score is: 18
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------

Query image 009 matched with wrong reference image 008
Number of inlier matches found in each case: [17, 9, 5, 5, 4, 4, 4, 4, 4, 0]
The actual rank of matching is: 2
------------------------------


Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------

Query image 002 is not in the dataset
Number of inlier matches found in each case: [8, 8, 7, 6, 5, 4, 4, 4, 4]
The highest matching score 8 is less than threshold 10
------------------------------
Query image 003 is not in the dataset
Number of inlier matches found in each case: [6, 5, 5, 5, 4, 4, 4, 4, 0]
The highest matching score 6 is less than threshold 10
------------------------------
Query image 004 is not in the dataset
Number of inlier matches found in each case: [8, 6, 6, 6, 5, 5, 5, 4, 4, 4]
The highest matching score 8 is less than threshold 10
------------------------------
Query image 005 is not in the dataset
Number of inlier matches found in each case: [7, 6, 5, 5, 5, 5, 4, 4, 4]
```

```
The highest matching score 7 is less than threshold 10
------------------------------
Query image 007 is not in the dataset
Number of inlier matches found in each case: [9, 7, 7, 5, 4, 4, 4, 4, 4]
The highest matching score 9 is less than threshold 10
------------------------------
Query image book_covers_001 is not in the dataset
Number of inlier matches found in each case: [9, 9, 7, 7, 4, 4, 4, 4, 4, 4]
The highest matching score 9 is less than threshold 10
------------------------------
Query image museum_paintings_019 is not in the dataset
Number of inlier matches found in each case: [8, 6, 6, 5, 4, 4, 4, 4, 4]
The highest matching score 8 is less than threshold 10
------------------------------


--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 12
Total number of matched image is: 4
Total number of query image macthed wrong with reference image is : 1
Total number of query image not in the dataset is : 7
Overall accuracy is: 0.5


--------------------------------------------------------------------------------
----------------------
```

Section 2.2:

More feature points have been extracted and SIFT algorithm has been used to improve the algorithm performance.

```
--------------------------------------------------------------------------------
----------------------

results:

Query images matched to Reference images with highest matching scores and same
title:
--------------------------------------------------------------------------------
----------------------

Query image 000 matched to reference image 000
Number of inlier matches found in each case: [15, 11, 9, 9, 9, 8, 7, 7, 6, 6]
```

```
Highest matching score is: 15
------------------------------
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [9, 9, 9, 8, 7, 7, 6, 5, 5, 4]
Highest matching score is: 9
------------------------------
Query image 005 matched to reference image 005
Number of inlier matches found in each case: [10, 9, 8, 8, 7, 6, 6, 6, 5, 5]
Highest matching score is: 10
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [11, 8, 7, 7, 6, 6, 6, 6, 6, 5]
Highest matching score is: 11
------------------------------
Query image 007 matched to reference image 007
Number of inlier matches found in each case: [13, 8, 7, 7, 6, 6, 6, 6, 6, 5]
Highest matching score is: 13
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [26, 11, 8, 6, 6, 5, 5, 5, 5, 5]
Highest matching score is: 26
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------

Query image 002 matched with wrong reference image 006
Number of inlier matches found in each case: [17, 12, 11, 7, 7, 6, 5, 5, 4, 4]
------------------------------
Query image 003 matched with wrong reference image 009
Number of inlier matches found in each case: [10, 9, 8, 8, 7, 6, 6, 6, 5, 4]
------------------------------
Query image 004 matched with wrong reference image 007
Number of inlier matches found in each case: [10, 8, 8, 8, 7, 6, 5, 5, 5, 4]
------------------------------
Query image 009 matched with wrong reference image 006
Number of inlier matches found in each case: [11, 9, 8, 6, 6, 6, 5, 5, 4, 4]
------------------------------
Query image book_covers_001 matched with wrong reference image 008
Number of inlier matches found in each case: [31, 20, 16, 10, 9, 9, 7, 7, 5, 5]
------------------------------
Query image museum_paintings_019 matched with wrong reference image 007
Number of inlier matches found in each case: [10, 8, 8, 7, 5, 5, 5, 5, 5, 4]
------------------------------
```

```
Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------


--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 12
Total number of matched image is: 6
Total number of query image macthed wrong with reference image is : 6
Total number of query image not in the dataset is : 0
Overall accuracy is: 0.5
```

The explanation of results and what changes made here:

In section 2.1, 10 query images and their corresponding reference images have been selected from the book cover image set, moreover, 2 extra images from landmarks and museum_paintings have been selected and added to the book cover query set. The overall accuracy of the algorithm is 100%, meaning all the query images have been successfully retrieved and all the extra images have been detected as not in the dataset. In section 2.2, 10 query images and their corresponding reference images have been selected from the landmarks image set, and 2 extra images from the book cover have been selected and added to the landmark query set. The algorithm reported that only 50% of images have been successfully retrieved or correctly identified as not in the dataset. By analyzing sections 2,1 and 2.2, the failure cases could have arisen from the insufficient number of key points and the amount of noise present in the image. Therefore, section 3 has demonstrated how the increase of feature points and using SIFT algorithm affect the performance. SIFT helps locate the local features in an image. These key points are scale & rotation invariant. Although SIFT has proven to be very efficient in object recognition applications, it requires a large computational complexity which is a major drawback, especially for real-time applications. Therefore, we have cached intermediate results for reference images. When we first compare the query image to each reference image, the information of the reference image's key points and descriptors has been stored on a map. In this way, when we compare the second query image to all the reference images, we can directly access the reference image's key points and descriptors without needing to compute them again. Through the results above we can see that since the ORB algorithm is more noise sensitive to SIFT algorithm, more images have been successfully retrieved, meaning a small part of failure cases due to a lot of noise present in the image have been fixed.

6. Repeat step 4 and 5 for at least one other set of reference images from museum_paintings or landmarks, and compare the accuracy obtained. Analyse both your overall result and individual image matches to diagnose where problems are occurring, and what you could do to improve performance. Test at least one of your proposed improvements and report its effect on accuracy.

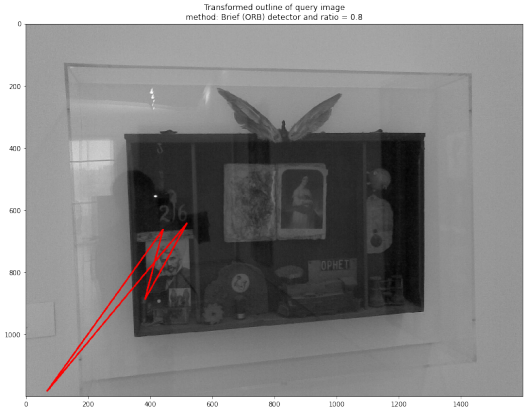
```
--------------------------------------------------------------------------------
----------------------
```

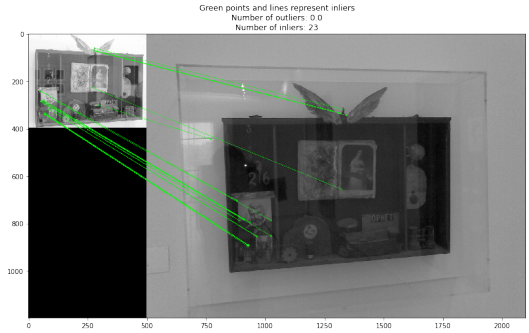
Section 3.1: One additional landmarks reference image and on additional museum_paintings reference image has been added to the reference image data set.

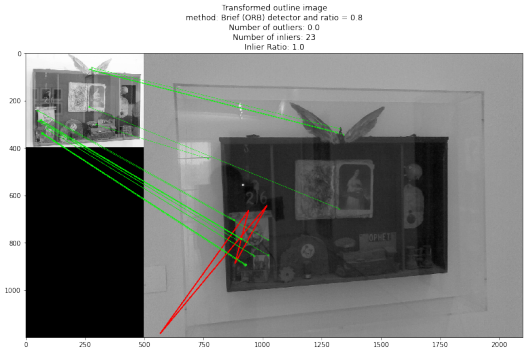
```
--------------------------------------------------------------------------------
----------------------

results:

Query images matched to Reference images with highest matching scores and same
title:
--------------------------------------------------------------------------------
----------------------

Query image 000 matched to reference image 000
Number of inlier matches found in each case: [20, 8, 7, 6, 5, 5, 4, 4, 4, 4]
Highest matching score is: 20
------------------------------
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [15, 7, 6, 5, 5, 4, 4]
Highest matching score is: 15
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [19, 6, 5, 5, 4, 4, 4, 4, 0]
Highest matching score is: 19
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [18, 6, 5, 5, 5, 4]
Highest matching score is: 18
------------------------------


Query images matched with wrong Reference images:
--------------------------------------------------------------------------------
----------------------

Query image 009 matched with wrong reference image 008
Number of inlier matches found in each case: [17, 9, 5, 5, 4, 4, 4, 4, 4, 0]
------------------------------
Query image museum_paintings_024 matched with wrong reference image 008
Number of inlier matches found in each case: [12, 7, 6, 5, 4, 4, 4, 4, 0]
------------------------------
```

```
Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------

Query image 002 is not in the dataset
Number of inlier matches found in each case: [8, 8, 7, 6, 5, 4, 4, 4, 4]
The highest matching score 8 is less than threshold 10
------------------------------
Query image 003 is not in the dataset
Number of inlier matches found in each case: [6, 5, 5, 5, 4, 4, 4, 4, 0]
The highest matching score 6 is less than threshold 10
------------------------------
Query image 004 is not in the dataset
Number of inlier matches found in each case: [8, 6, 6, 6, 5, 5, 5, 4, 4, 4]
The highest matching score 8 is less than threshold 10
------------------------------
Query image 005 is not in the dataset
Number of inlier matches found in each case: [7, 6, 5, 5, 5, 5, 4, 4, 4]
The highest matching score 7 is less than threshold 10
------------------------------
Query image 007 is not in the dataset
Number of inlier matches found in each case: [9, 7, 7, 5, 4, 4, 4, 4, 4]
The highest matching score 9 is less than threshold 10
------------------------------
Query image book_covers_001 is not in the dataset
Number of inlier matches found in each case: [9, 9, 7, 7, 4, 4, 4, 4, 4, 4]
The highest matching score 9 is less than threshold 10
------------------------------
Query image museum_paintings_019 is not in the dataset
Number of inlier matches found in each case: [8, 6, 6, 5, 4, 4, 4, 4, 4]
The highest matching score 8 is less than threshold 10
------------------------------
Query image landmarks_027 is not in the dataset
Number of inlier matches found in each case: [5, 5, 5, 5, 4, 4, 4, 4]
The highest matching score 5 is less than threshold 10
------------------------------


--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 14
Total number of matched image is: 4
Total number of query image macthed wrong with reference image is : 2
Total number of query image not in the dataset is : 8
Overall accuracy is: 0.5
```

The output of one faliur case that affected accuracy by using alogorithm above:







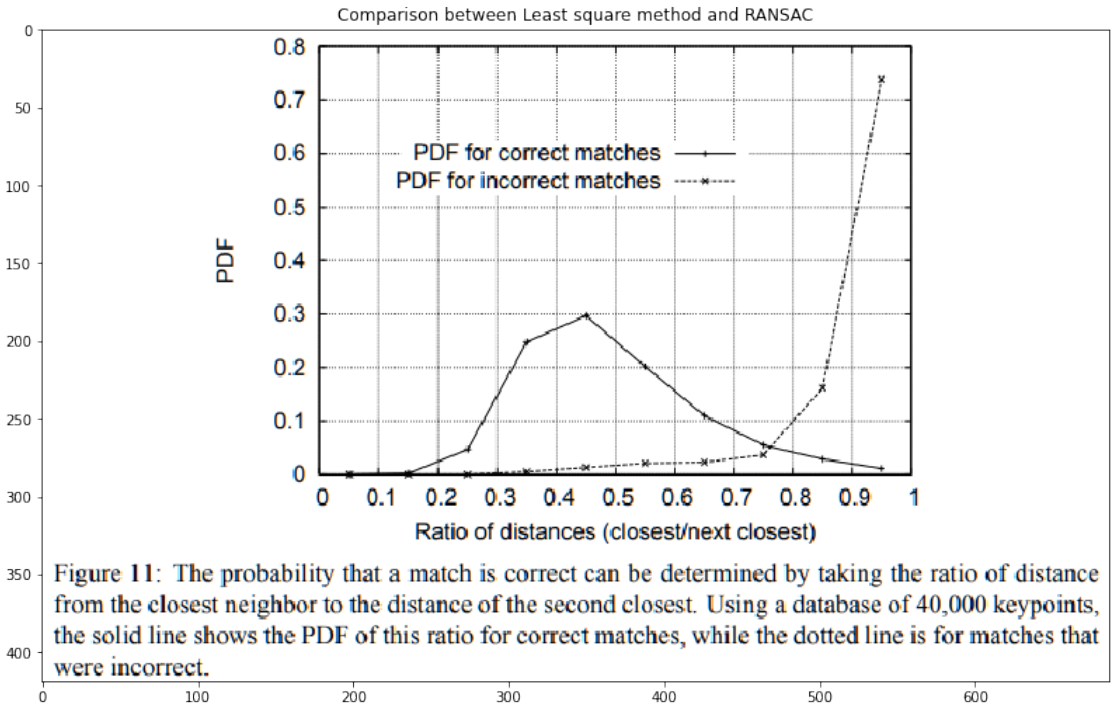


Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

Discussion of the reason for failure:

By observing the above failure case, some nosiy and distinctive distortion are presented in the image. Therefore, in this case, using brief descriptor from ORB and ratio test has failed, due to the insufficient number of feature points detected and selected,and the excessive wrong matches disturbing the iterative parameter estimation process for the RANSAC algorithm. Lowe's ratio test works in this way: each keypoint of the first image is matched with a number of keypoints from the second image. We keep the 2 best matches for each keypoint (best matches = the ones with the smallest distance measurement). Lowe's test checks that the two distances are sufficiently different. If they are not, then the keypoint is eliminated and will not be used for further calculations. In the fowlloing section, an improvement has been proposed by setting different ratio for key points distance check and BELID descriptor and BoostedSSC algorithm have been used to solve the issue present above.

BELID descriptor and BoostedSSC algorithm have been used to improve performance.

```
------------------------------------------------------------------------------
----------------------

results:

Query images matched to Reference images with highest matching scores and same
title:
------------------------------------------------------------------------------
----------------------

Query image 000 matched to reference image 000
Number of inlier matches found in each case: [15, 5, 5, 4, 4]
Highest matching score is: 15
------------------------------
Query image 001 matched to reference image 001
Number of inlier matches found in each case: [10, 6, 5, 4, 4]
Highest matching score is: 10
------------------------------
Query image 004 matched to reference image 004
Number of inlier matches found in each case: [8, 4, 4, 4, 4, 0]
Highest matching score is: 8
------------------------------
Query image 006 matched to reference image 006
Number of inlier matches found in each case: [20, 5, 5, 4, 4]
Highest matching score is: 20
------------------------------
Query image 007 matched to reference image 007
Number of inlier matches found in each case: [9, 6, 5, 4, 4, 0, 0]
Highest matching score is: 9
------------------------------
Query image 008 matched to reference image 008
Number of inlier matches found in each case: [28, 6, 4, 4, 4, 4, 4, 4]
Highest matching score is: 28
------------------------------


Query images matched with wrong Reference images:
------------------------------------------------------------------------------
----------------------

Query image 009 matched with wrong reference image 008
Number of inlier matches found in each case: [15, 5, 0]
------------------------------
```

```
Query images have same title as Reference image but highest matching score is
less than a threshold:
--------------------------------------------------------------------------------
----------------------

Query image 002 is not in the dataset
Number of inlier matches found in each case: [6, 5, 4, 4, 4]
The highest matching score 6 is less than threshold 10
------------------------------
Query image 003 is not in the dataset
Number of inlier matches found in each case: [6, 5, 5, 4, 4]
The highest matching score 6 is less than threshold 10
------------------------------
Query image 005 is not in the dataset
Number of inlier matches found in each case: [5, 4, 4, 4, 4, 4]
The highest matching score 5 is less than threshold 10
------------------------------
Query image book_covers_001 is not in the dataset
Number of inlier matches found in each case: [6, 4, 4, 4, 4]
The highest matching score 6 is less than threshold 10
------------------------------
Query image museum_paintings_019 is not in the dataset
Number of inlier matches found in each case: [5, 5, 5, 4, 4, 0]
The highest matching score 5 is less than threshold 10
------------------------------
Query image landmarks_027 is not in the dataset
Number of inlier matches found in each case: [7, 4, 4, 4, 0]
The highest matching score 7 is less than threshold 10
------------------------------
Query image museum_paintings_024 is not in the dataset
Number of inlier matches found in each case: [6, 6, 5, 4, 0, 0]
The highest matching score 6 is less than threshold 10
------------------------------


--------------------------------------------------------------------------------
----------------------

Total number of image matched is: 14
Total number of matched image is: 6
Total number of query image macthed wrong with reference image is : 1
Total number of query image not in the dataset is : 7
Overall accuracy is: 0.7142857142857143


--------------------------------------------------------------------------------
----------------------
```

Your description of what you have done, and explanation of results, here

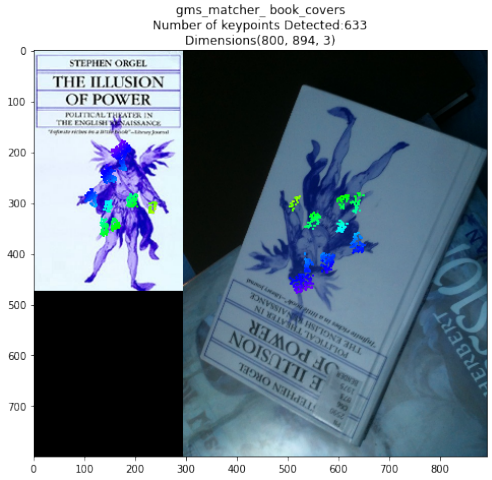
Local image representations are designed to match images in the presence of strong appearance variations, such as illumination changes or geometric transformations. In this section, we have introduced BELID, an efficient real-valued descriptor. In BELID we use the BoostedSSC algorithm to discriminatively select a set of features and combine them to produce a strong description. BELID achieves execution times close to the fastest technique, ORB with an accuracy similar to that of SIFT. Specifically, it provides accuracy better than SIFT in the patch verification. By using BELID we can see that more images are successfully retrieved and more images are correctly identified as not in the dataset. For the ratio part, 0.75 has been selected rather than 0.8. Based on Lowe's ratio test, the match with the smallest distance is the "good" match, and the match with the second-smallest distance is the equivalent of random noise, a base rate of sorts. If the "good" match can't be distinguished from noise, then the "good" match should be rejected because it does not bring anything interesting, information-wise. So the general principle is that there needs to be enough difference between the best and second-best matches. By conducting the experiment of using BELID descriptor and BoostedSSC algorithm, as well as choosing appropriate ratio value, the overall algorithm's accuracy has been improved from 0.5 to 0.71.

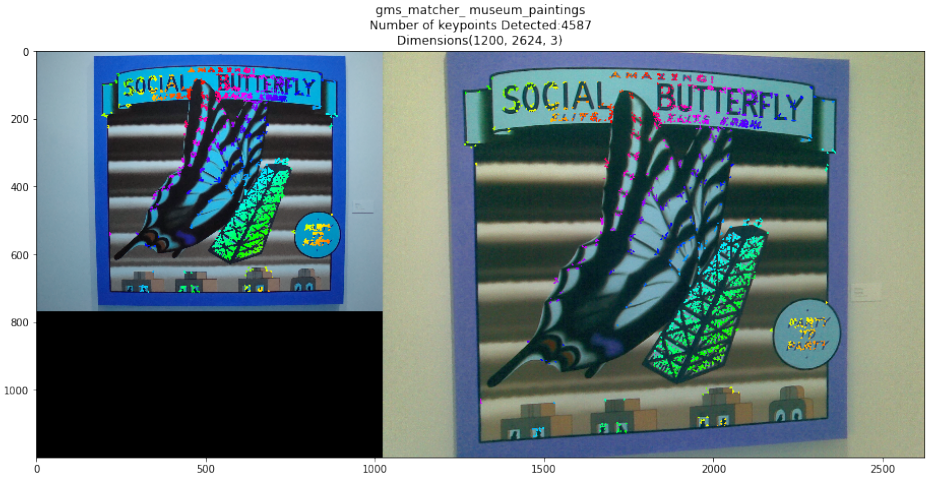
4 Question 3 (10%)

In Question 1, We hope that `ratio_test` can provide reasonable results for RANSAC. However, if it fails, the RANSAC may not get good results. In this case, we would like to try an improved matching method to replace the `ratio_test`. Here, the `gms_matcher` is recommened. You need to implement it and save results of 3 image pairs (you can select any image pairs from the dataset), where you new method is better than 'ratio_test'.

1. Hint 1: `cv2.xfeatures2d.matchGMS()` can be used, but you need to install the opencv-contrib by `pip install opencv-contrib-python`
2. Hint 2: You do not need use KNN matching, because GMS does not require second nearest neighbor.
3. Hint 3: You need to change the parameters in `cv2.ORB_create()` for best results. See the setting in Github.
4. Hint 4: If your are interested in more details. Read the paper "GMS: Grid-based Motion Statistics for Fast, Ultra-robust Feature Correspondence", and the Github "https://github.com/JiawangBian/GMS-Feature-Matcher".

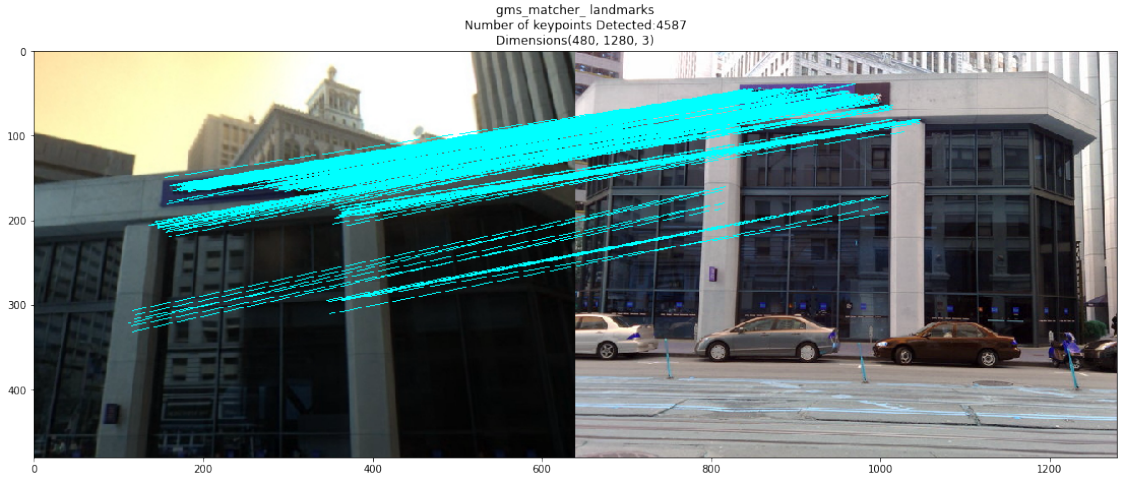
```
Found 265 matches
Found 633 matches
Found 4587 matches
```

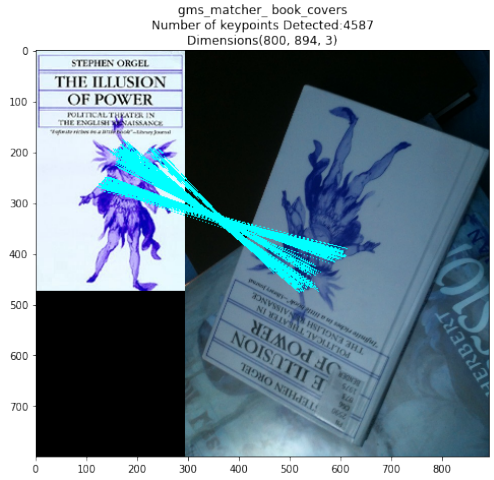


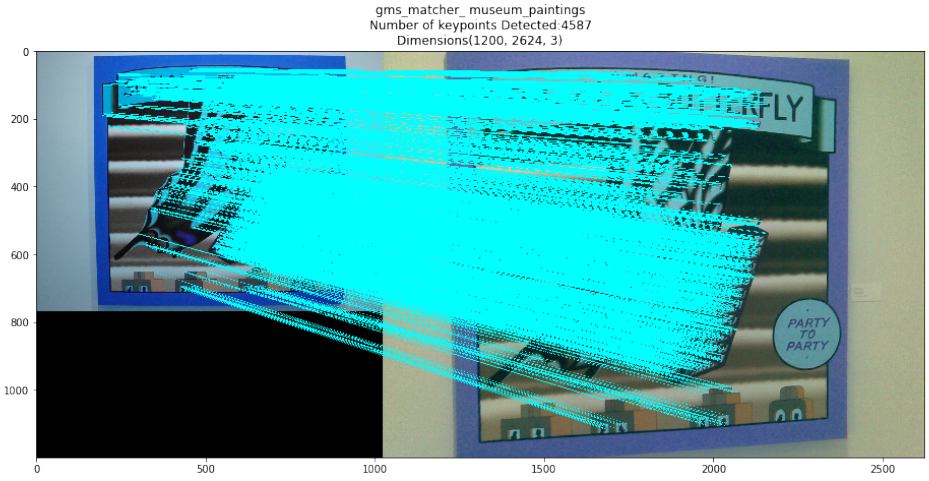



```
Found 270 matches
GMS takes 0.0031027793884277344 seconds
Found 339 matches
GMS takes 0.002628326416015625 seconds
```

```
Found 4196 matches
GMS takes 0.003221750259399414 seconds
```







Your results here

The existing methods like ORB and SIFT take a long computational time, limiting their use in real-time applications. The above method attempts to separate true correspondences from false ones at high speed. The method (GMS) grid-based motion Statistics, incorporates the smoothness constraint into a statistic framework for separation and uses a grid-based implementation for fast calculation. GMS is robust to various challenging image changes, involving viewpoint, scale, and rotation. It is also fast, e.g., takes only 1 or 2 ms in a single CPU thread, even when 50K correspondences are processed. This has important implications for real-time applications.

5 Question 4: Reflection Questions (5%)

1. Describe the hardest situation you faced during the first two assignments. And how you overcome it? (3%)
2. How do you plan to finish the assignment to meet tight deadline? (2%)

1. The hardest situation I faced during the first two assignments is I was not sure how to implement the knowledge I learned from lectures or workshops into the Assignment. Often, I could understand the concepts of the course materials, but to convey the concepts and actually use them to implement an algorithm can be very tricky. To overcome this, I've done a lot of research online and read some papers that are related to the question in the assignment. Moreover, I frequently ask Yifan and other tutors questions and gain many insightful feedbacks from them. This helped me a lot with my assignment.
2. I tend to finish the assignment as soon as I can. If I faced some problems that I cannot overcome, I would ask the tutor for assistance. If the problem is not related to the rest of the question, I can hold the problem and continue the rest of the assignment first, and finally come back to the problem. This way, I can finish the assignment to meet a tight deadline.