Assignment2

May 7, 2021

1 Computer Vision 2021 Assignment 2: Image matching and retrieval

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

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

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
```

This will also remove the preamble text from each question. We will use the `scikit-image` library again to complete the prac. It has several built in functions that will be useful. You are expected to consult documentation and use them appropriately. If you are experienced and want to use functions from other libraries (e.g. OpenCV) that is OK too, but the prac is designed to work with skimage functions so I recommend starting with those.

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!

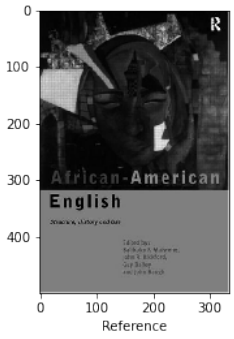
Do not submit your own copy of the data or rename any files or folders! For marking, we will assume the datasets are available in subfolders of the working directory using the same folder names provided.

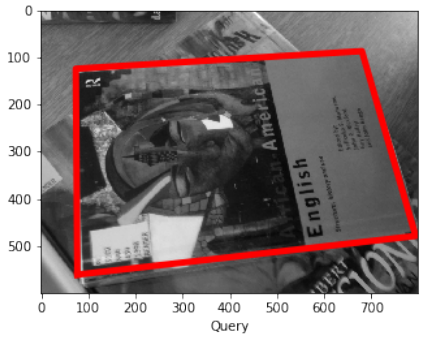
Here is some general setup code, which you can edit to suit your needs.

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

2 Question 1: Matching an object in a pair of images (60%)

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



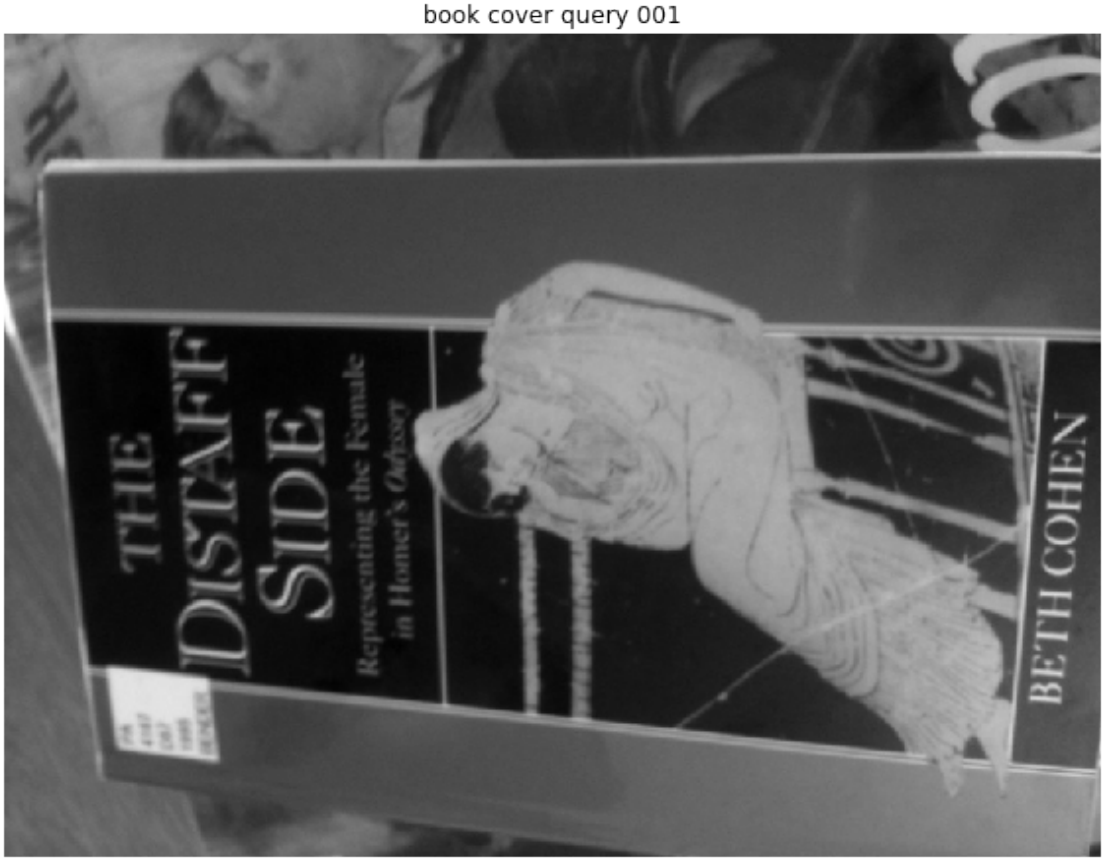


0. Download and read through the paper ORB: an efficient alternative to SIFT or SURF by Rublee et al. You don't need to understand all the details, but try to get an idea of how it works. ORB combines the FAST corner detector (covered in week 3) and the BRIEF descriptor. BRIEF is based on similar ideas to the SIFT descriptor we covered week 3, but with some changes for efficiency.

1. Load the first (reference, query) image pair from the "book_covers" category. Calculate ORB features in each image using `skimage.feature.ORB()` and match the features using `skimage.feature.match_descriptors()`.

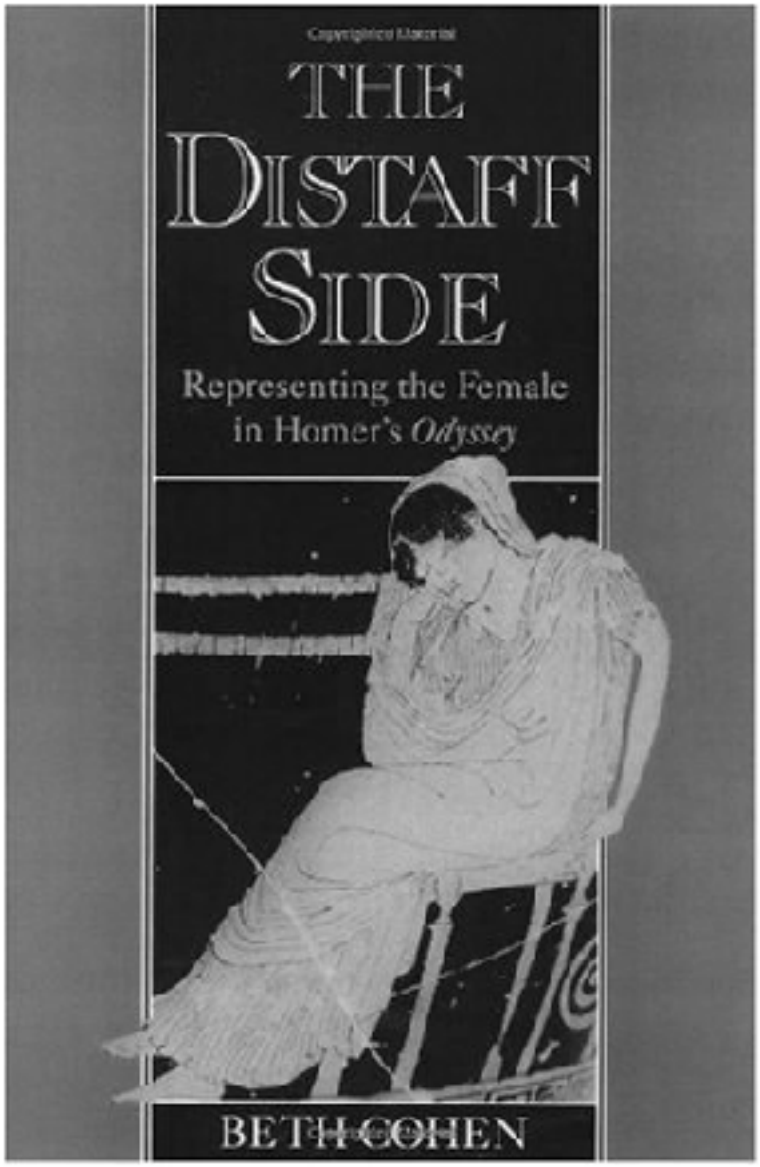
2. Try changing the parameters for the ORB feature and descriptor matching to improve the matching result. You can use `skimage.feature.plot_matches()` to see the matches visually, in addition to checking the output of `match_descriptors`. At this point, you are not expected to measure matching accuracy numerically. Instead, it is enough to point out errors or differences in performance that you notice by changing settings.

1. Hint 1: read the documentation for `ORB()` and `match_descriptors()` carefully to understand what the parameters do. More detail on the ORB feature is also available in the original paper.
2. Hint 2: Display your results so that the changes you discuss are clearly visible.
3. Hint 3: Does `match_descriptors(ref, query)` give the same result as `match_descriptors(query, ref)`?



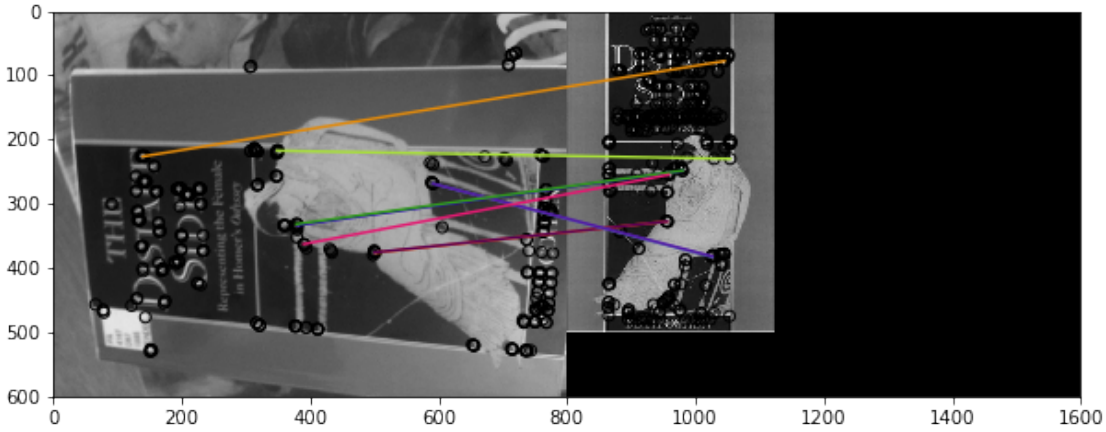

```
Query image shape:  (600, 800)
```

book cover reference 001

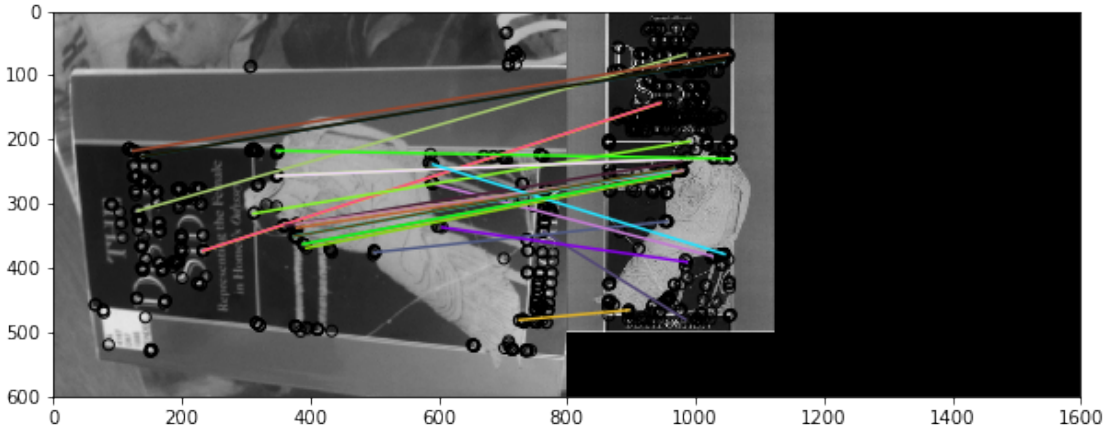


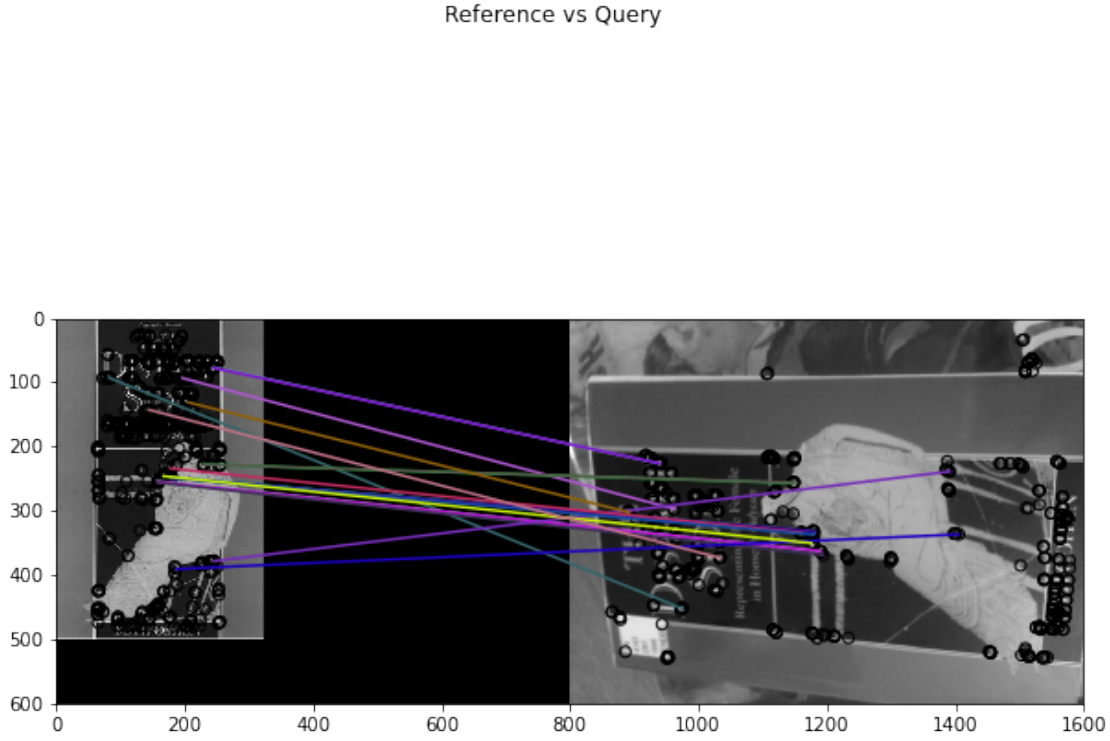

```
Reference image shape:  (500, 325)
```

Query vs Reference



Query vs Reference





ORB is a combination of oFAST(Orientated FAST keypoints detector) and rBrief(Rotated BRIEF descriptor). ORB uses FAST to detect keypoints (more than n_keypoints) and uses Harris corner measurement to order the keypoints and the top n_keypoints are selested after the ordering, then compute rBRIEF features.BRIEF performance is similar to SIFT, but it uses binary tests to train a set of classification trees, which is much faster than SIFT.To recover the loss of the steer BRIEF and reduce correlation among the binary tests, a learning method is introduced into the BRIEF to choose a good subset of binary tests, which forms the rBRIEF.ORB has three steps: feature point extraction,generating feature point descriptors and feature point matching.

ORB() function has parameters including: n_keypoints, fast_n, fast_threshold, harris_k, downscale, n_scales. n_keypoints is the number of best keypoints to be returned according to the Harris corner response. fast_n is the minimum number of consecutive pixels out of 16 pixels on the circle that should all be either brighter or darker with a threshold with respective to the test pixel, in this case it's 7, and it's get through testing fast_threshold is the threshold mentioned above, in this case it's 0.1, the value is get through testing and tuning. harris_k is the sensitivity of seperating the corner from edges,in range from 0 to 0.2. Small values of k result in detection of sharp corners.In this case, a median o.1 is chosen, because sharp corner and wide corner are the same amount. downscale is the downscale factor for image pyramid. Default value 1.2 is chosen so that there are more dense scales which enable robust scale invariance for a subsequent feature description. n_scale is the maximum number of scales in pyramid, and a number of 8 is chosen.We don't need too many scales, because the reference and the query image don't have a large difference in scales.(600,8000) and (500,325)

match_descriptor() function has parameters:descriptors1, descriptors2, metric(optional), p,

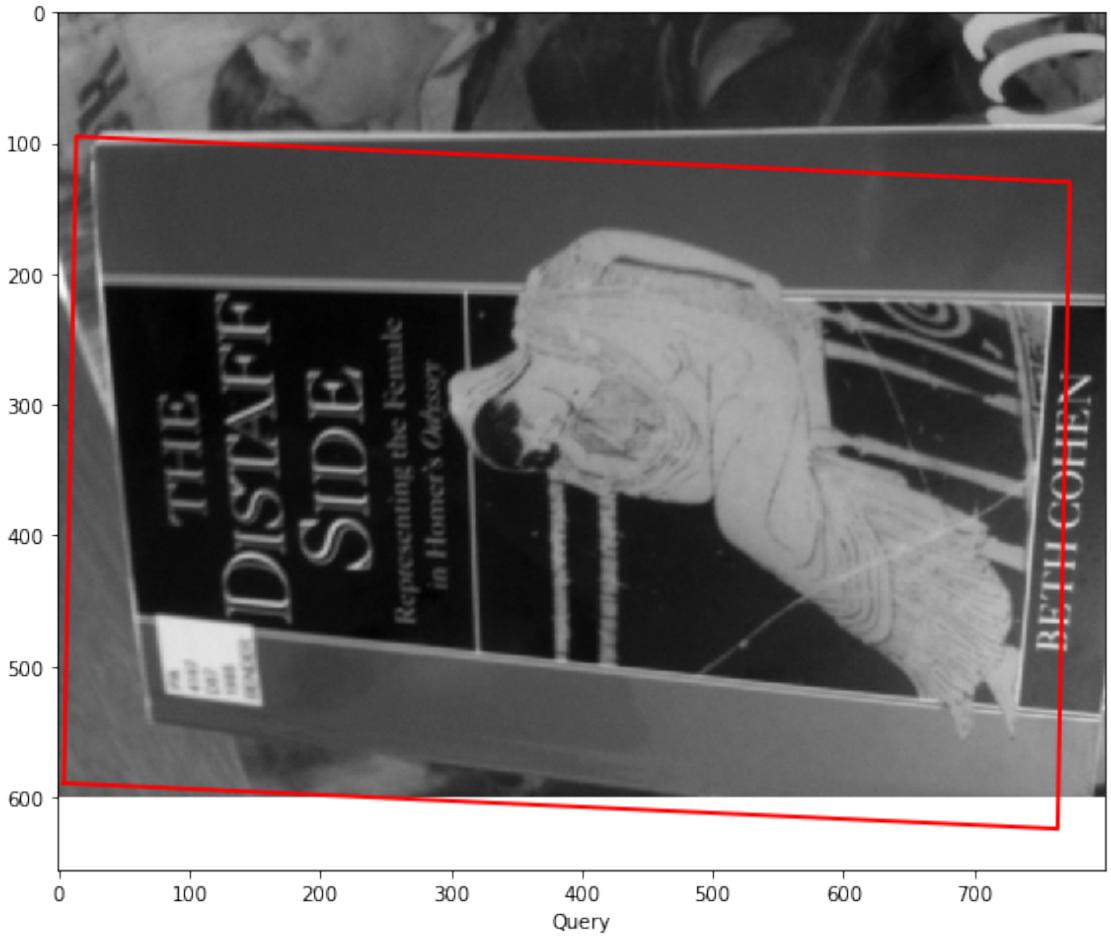
max_distance, cross_check, max_ratio. Descriptor1,2 are the descriptor of keypoints in image1 and image2. Metric is the metric to compute the distance between two descriptors, and Hamming distance is used for binary descriptors automatically, which is used in this case. Max_distance is the maximum allowed distance between descriptors of two keypoints in separate images to be regarded as a match. Cross_check: a matched pair (keypoint1, keypoint2) is returned if keypoint2 is the best match for keypoint1 in second image and keypoint1 is the best match for keypoint2 in first image.max_ratio is the maximum ratio of distances between first and second closest descriptor in the second set of descriptors. It's useful to filter ambiguous matches between the two descriptor sets. The unique match has a lower ratio.The choice of this value depends on the statistics of the chosen descriptor, e.g., for SIFT descriptors a value of 0.8 is usually chosen.In this case, we used rBRIEF, which is similar to SIFT and through testing and parameter tuning, a max_ratio of 0.75 is found to be perfect for this setting.

In the parameters tuning process, I found the n_keypoints and fast_threshold in ORB and the max_ratio in match_descriptors are two useful parameter to be tuned. After testing and tuning,from visual observation,the matches are almost all corrrect matches.

Even through, the matches are correct, but from the "Reference vs Query" graph, it shows the match_descriptors(ref, query) will not give the same result as match_descriptors(query, ref), because the matches are about different pixel locations. For each descriptor in the first set, the match_descriptor() function will find a the closest descriptor in the second set using brute force method. Therefore, the match pairs will only contains the matches corresponding to the keypoints in the first set.

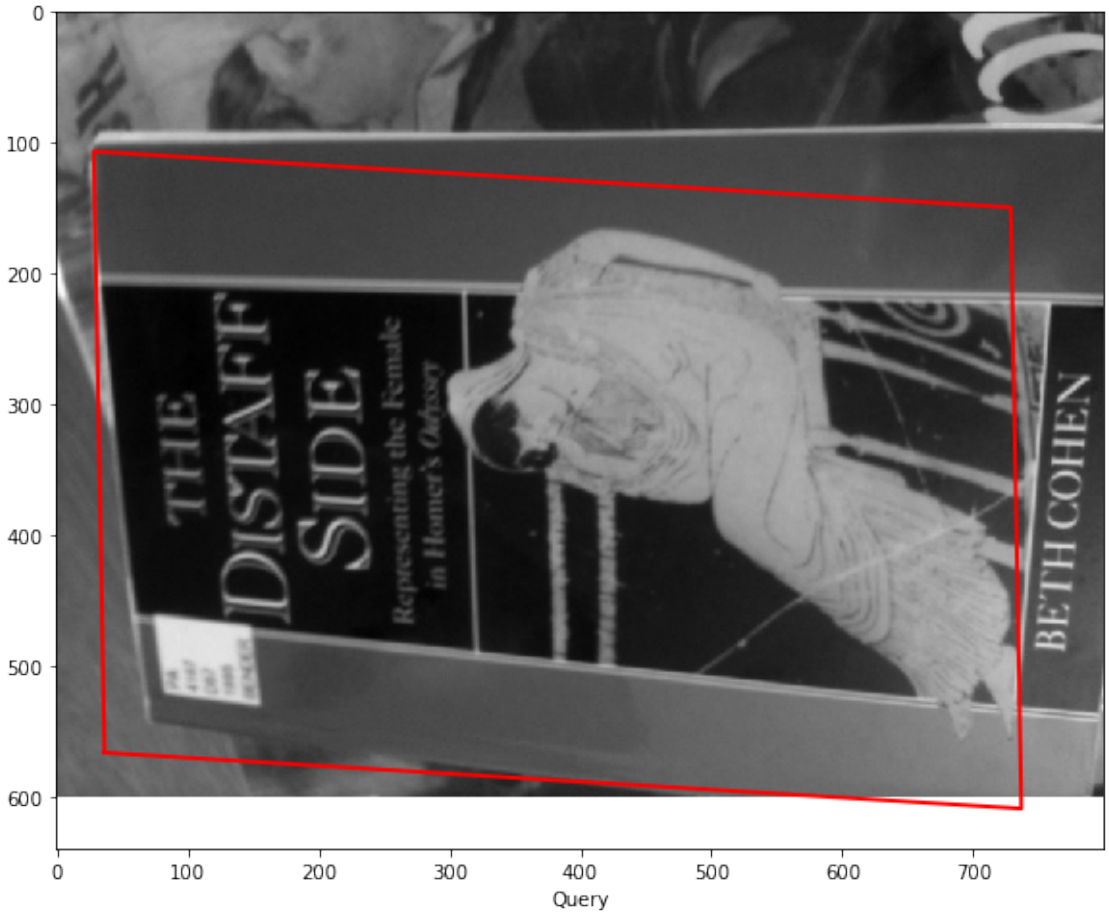
3. Estimate an affine transformation based on the matches, using `skimage.transform.estimate()`. Display the transformed outline of the first reference book cover image on the query image, to see how well they match. Repeat for a 2D projective transformation. Explain your results.

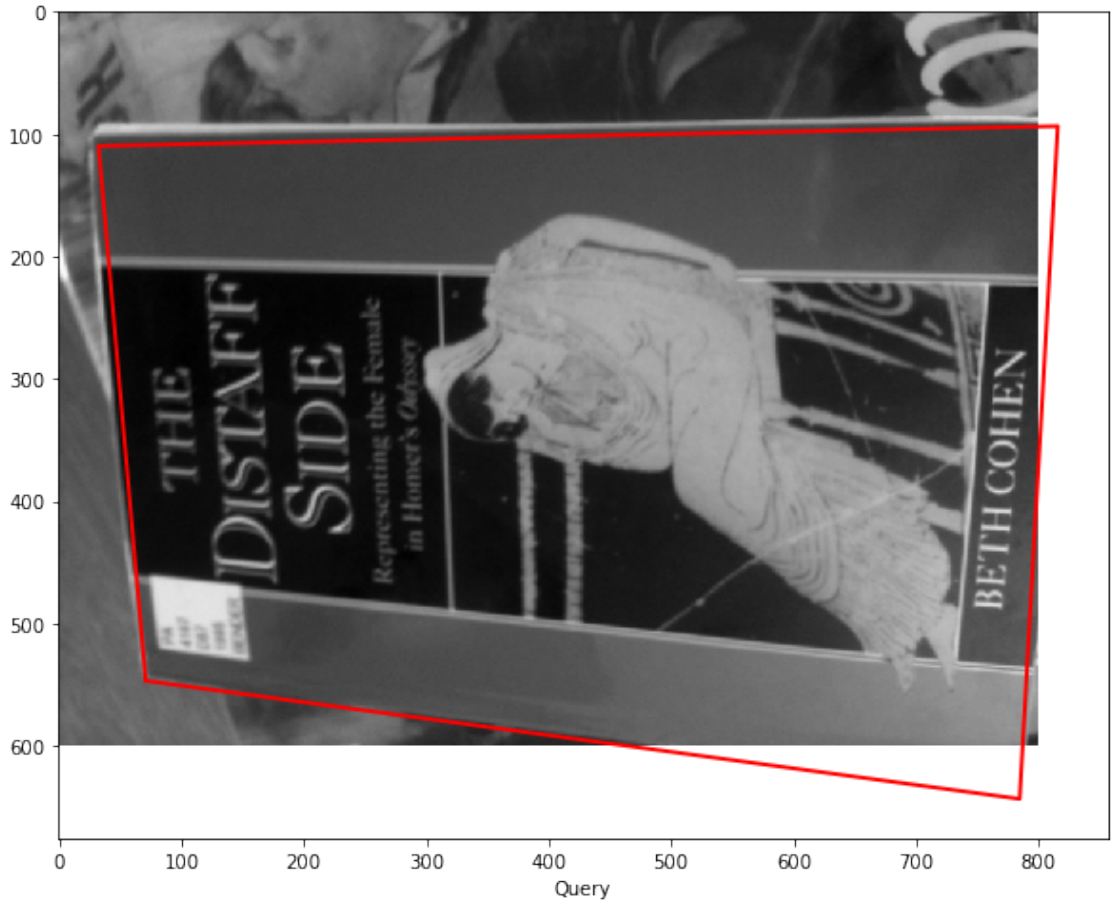
    - We provide a function `draw_outline()` to help with the display, but you may need to edit it for your needs.
    - Again, you don't need to compare results numerically at this stage. Comment on what you observe visually.



From the outlined query image above, it is observed that the outline of the book is roughly outlined, but with an extend of skewness and out of frame.This is because there are wrong matches causing the skewness. Least square is not robust, the result can be very skewed and distorted, however, using ransac can deal with the case of low true positiive rates and give a robust rusult like the section below.

4. Use RANSAC to estimate an affine transformation and eliminate outlier matches using `skimage.measure.ransac()`. Repeat for a projective transformation. Experiment with different settings and display the effect on the estimated location of the object.


THE
DISTAFF
SIDE
Representing the Female
in Homer's Odyssey
BETH COHEN
0
100
200
300
400
500
600
700
Query



From the pictures above,using ransac algorithm to estimate the affine transform has better outline performance than the normal linear least sqaure method above.RANSAC randomly samples several matches, and use this set of matches to estimate the desired transform using linear least square. Then repeat step 1 and 2 enough times to reach a high accuracy. We can get the one with the highest consensus from these transformation estimation, and the matches agree with it. Therefore, even with a low true positive rate, RANSAC still can provide robustness.

The projective transformation estimation using RANSAC has a better outline performance than the affine transformation, this is because the projective transformation is the actual transform of this book cover.Affine transform is a combination of similarity transform and sheer, while the perspective transformation is affine transformation with extra perspective, which is the same case as the transform situation of this book cover.Affine outlined the correct location of the book cover, but it's slightly out of frame, because there is no perspective in affine transform. The outline using projective transformation almost correctly aligned with 3 book corners, but the book's bottom left corner is not in the image and that causes the mismatch in the bottom left book corner.Overall, using RANSAC, we can gain much more robustness than linear sqaure method.Affine transform is not enough to describe the transform of the book cover, but with elements to descibe the perspective i.e. using projective transform estimation to outline the book can give rise to a really good object matching performance.

5. Assume that the inliers found by RANSAC are true matches, and the rest are false matches. How many true positive and false positive matches were found by `match_descriptors()` in part 2? Answer for at least two different cases you tried in part 2.

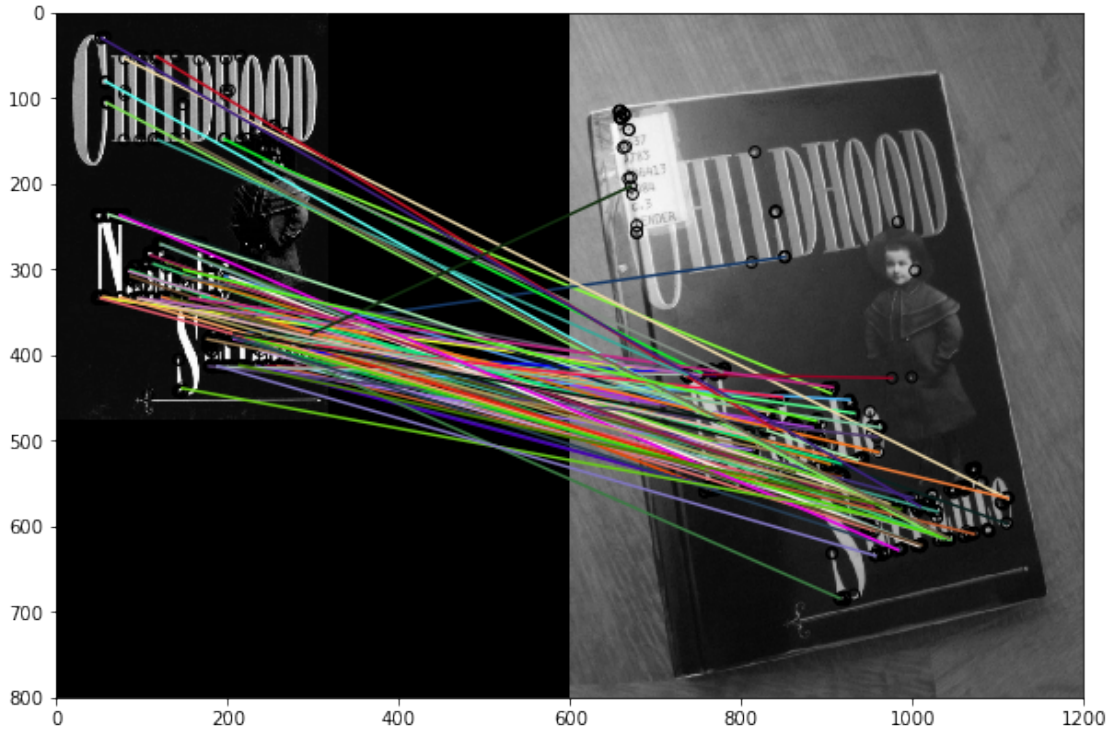
```
Case 1 TP (inlier number): 6
Case 1 FP (outlier number): 13
Case 1 True Positive Rate:  0.32
Case 2 TP (inlier number): 0
Case 2 FP (outlier number): 9
Case 2 True Positive Rate:  0.00
Case 3 TP (inlier number): 6
Case 3 FP (outlier number): 13
Case 3 True Positive Rate:  0.32
```

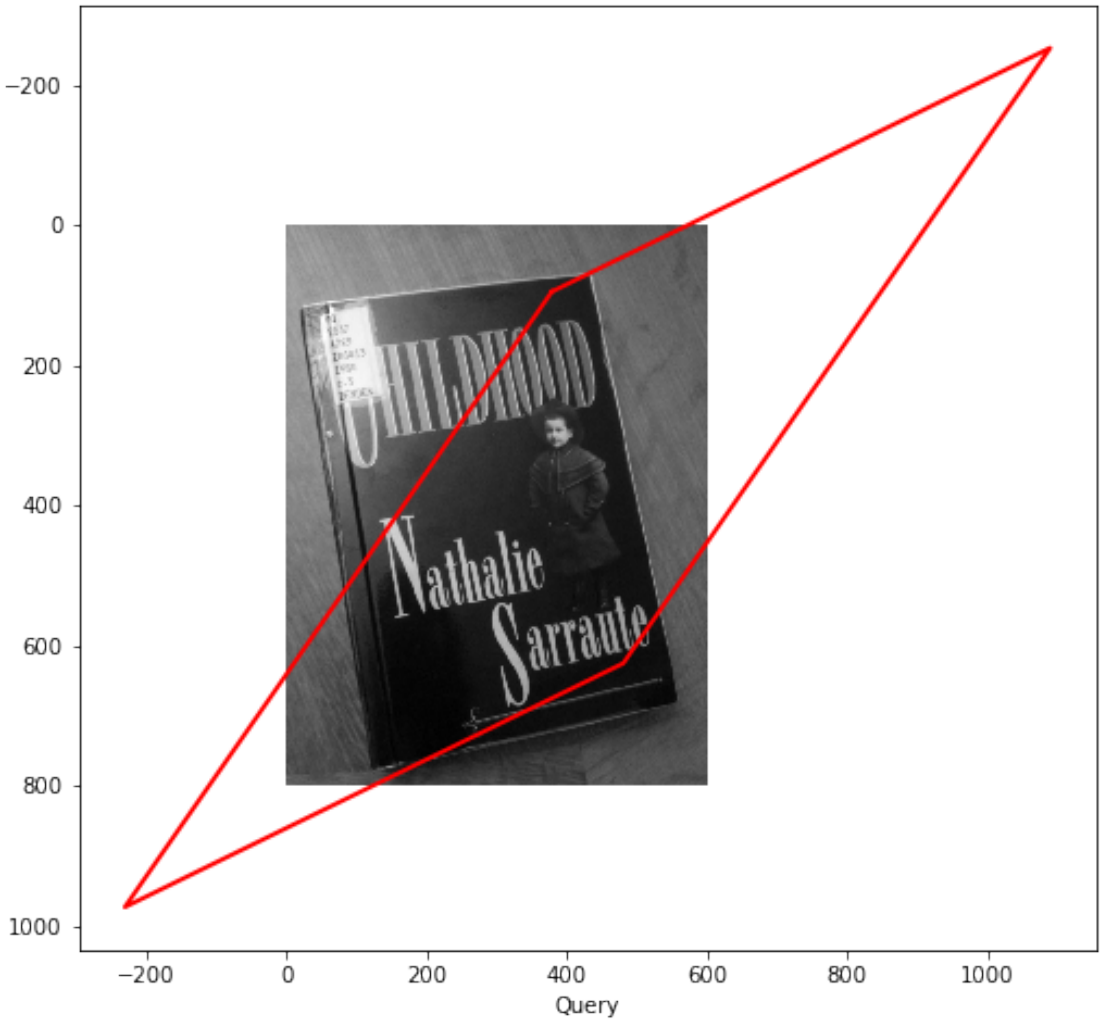
Case 1: well-tuned ORB; Case 2: early testing setting ORB; Case 3: match_descriptors(ref, query) From the data obtained, it showed that the TP rate in case 1 and 3 are same, 32%. Even match_descriptors(ref, query) and match_descriptors(query,ref) gives different matches, but the quality of the matches are roughly at the same level. For case 2, which has relatively poor ORB parameter settings, only has 9 matches, including 0 TP and 9 FP, while in case 1 amd 3, they both have 19 matches. The proper matches are very few so that it's hard for RANSAC to iterate and find the accurate transform. Therefore, both the quality and the quantity of the matches are crucial to RANSAC.

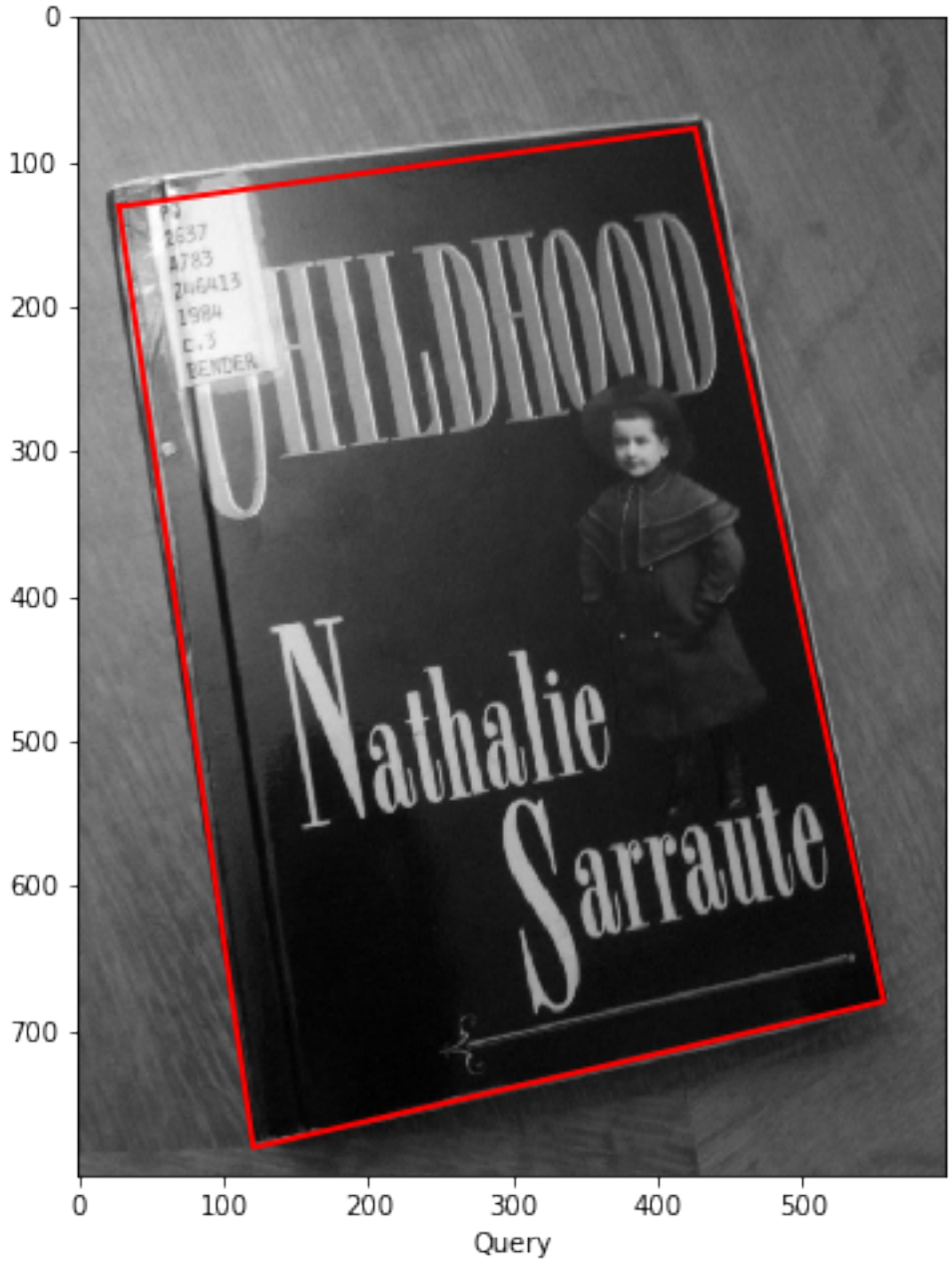
6. Finally, try matching several different image pairs from the data provided, including at least one success and one failure case. For the failure case, test and explain what step in the feature matching has failed, and try to improve it. Display and discuss your findings.
    1. Hint 1: In general, 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.
    4. Hint 4: If the feature detector or descriptor is causing failures, the `skimage.feature` module contains several other detectors (e.g. Harris corners) and descriptors for you to try.

    5. Hint 5: After loading, resize the query image so that it is approximately the same size as the reference image (within a factor of 2, say). Although the features we use are somewhat scale invariant this should improve your results. You are free to apply any other pre-processing steps you find useful, but tell us what you have done and why!

```
TP (inlier number): 32 ,FP (outlier number): 41 ,True Positive Rate:  0.44
```


Query vs Reference
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200

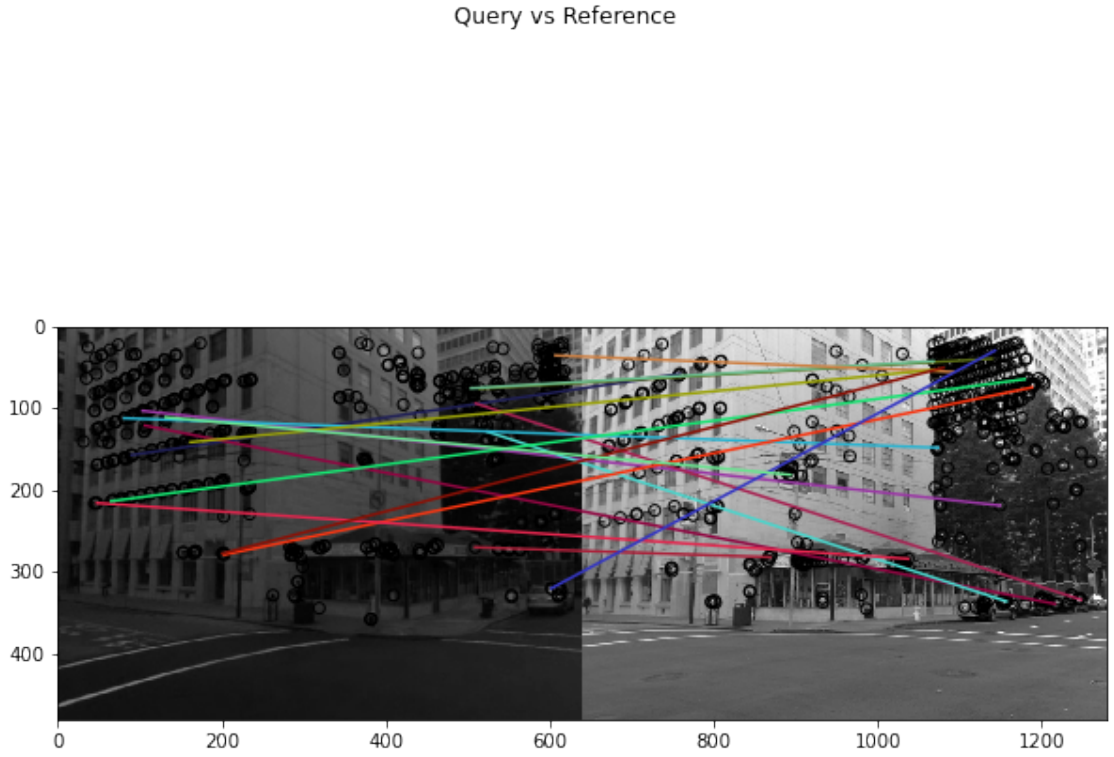

−200
0
200
400
600
800
1000
−200
0
200
400
600
800
1000
Query
CHILDHOOD
Nathalie
Sarraute

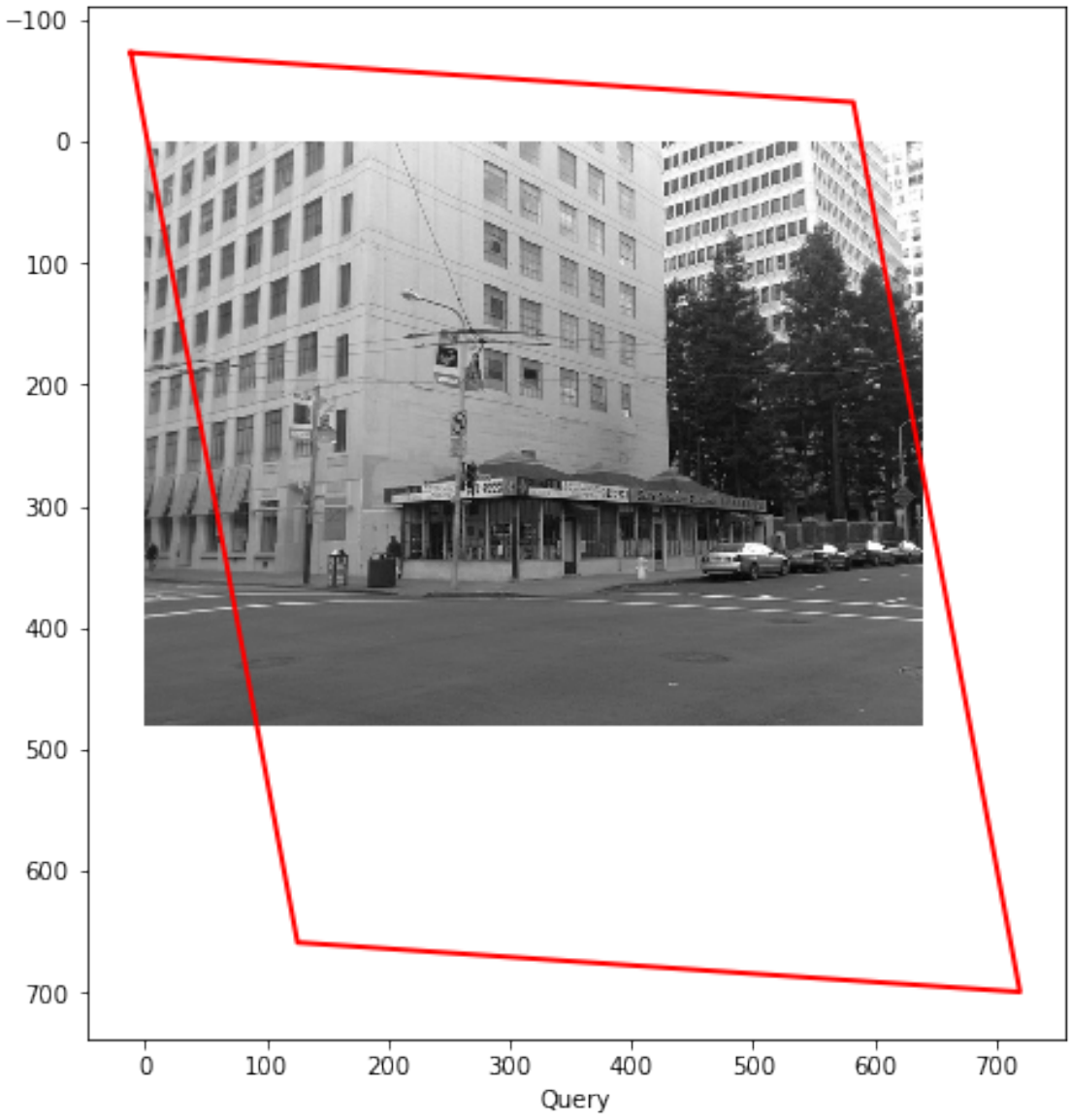


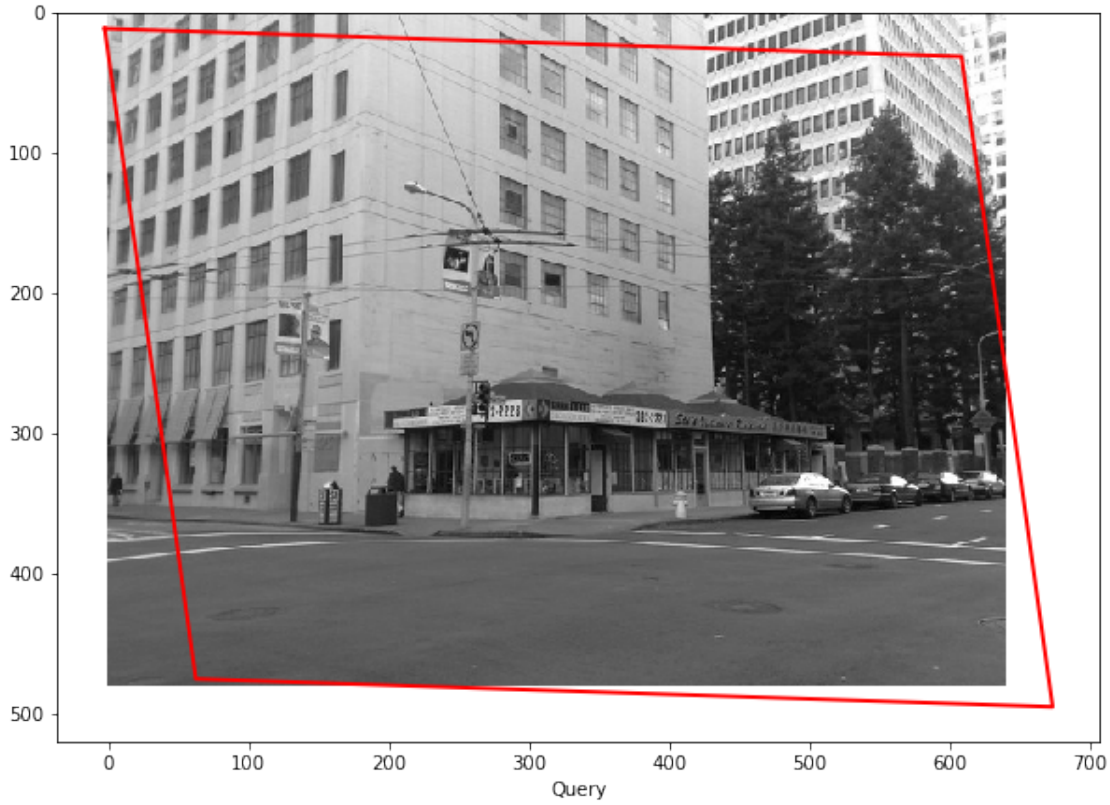
As shown in the image above, the book cover 39 after RANSAC is a very successful case. The book cover is corectly outlined, but only with a slight of unfitness on the top edge of the book cover. This case has 32 TP, 41 FP and a true positive rate of 44%. The reason why this book cover is chosen is the whole object is inside the image, which avoided the issue in image 001. Furthermore, the background of the object is simple and not distractive to the corner feature detector, so that the corner extractor can focus on the object. Due to its plain background, the keypoints detected are all within the book cover, which is exactly what we want. Besides, the book cover has a good amount of corners, because of the existance of the letters. However, there are similar corners caused the mismatch, such as the top corner of the letter 'C 'in 'Childhood' mismatched the top

corner of second ‘a’ in ‘Sarraute’, but from the plot match plot, it is shown that a large majority of the matches are correct. In order to reduce the number of mentioned ambiguious matches, the parameter of max_ratio in match_descriptor was incereased to solve this, but the outcome isn’t ideal, because when this parameter was tuned up, many correct matches disappeared as well. With the reduce number of matches, the final outcome from RANSAC lossed the robustess and couldn’t provide a good outcome.Therefore, the max_ratio is kept 0.8 to maintain the overall performance.The number of matches are important for RANSAC, so without lower the match standaerd, we increased the number of keypoints to increase the number of potential matches in the first place, thus n_keypoints in ORB setting is set to 500. The parameter settings are: ORB(n_keypoints=500,fast_n=7,fast_threshold=0.1,n_scales=7,downscale=1.2,harris_k=0.07) and match_descriptors(descriptors_2q, descriptors_2r,metric=‘hamming’,max_ratio=0.8).

```
Query image shape:  (480, 640)
Reference image shape:  (480, 640)
TP (inlier number): 4 FP (outlier number): 13 True Positive Rate:  0.24
```

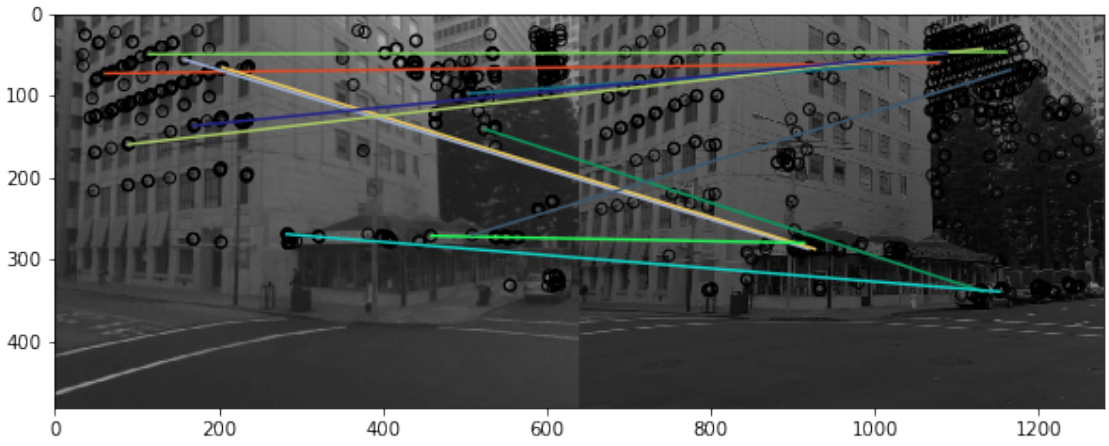


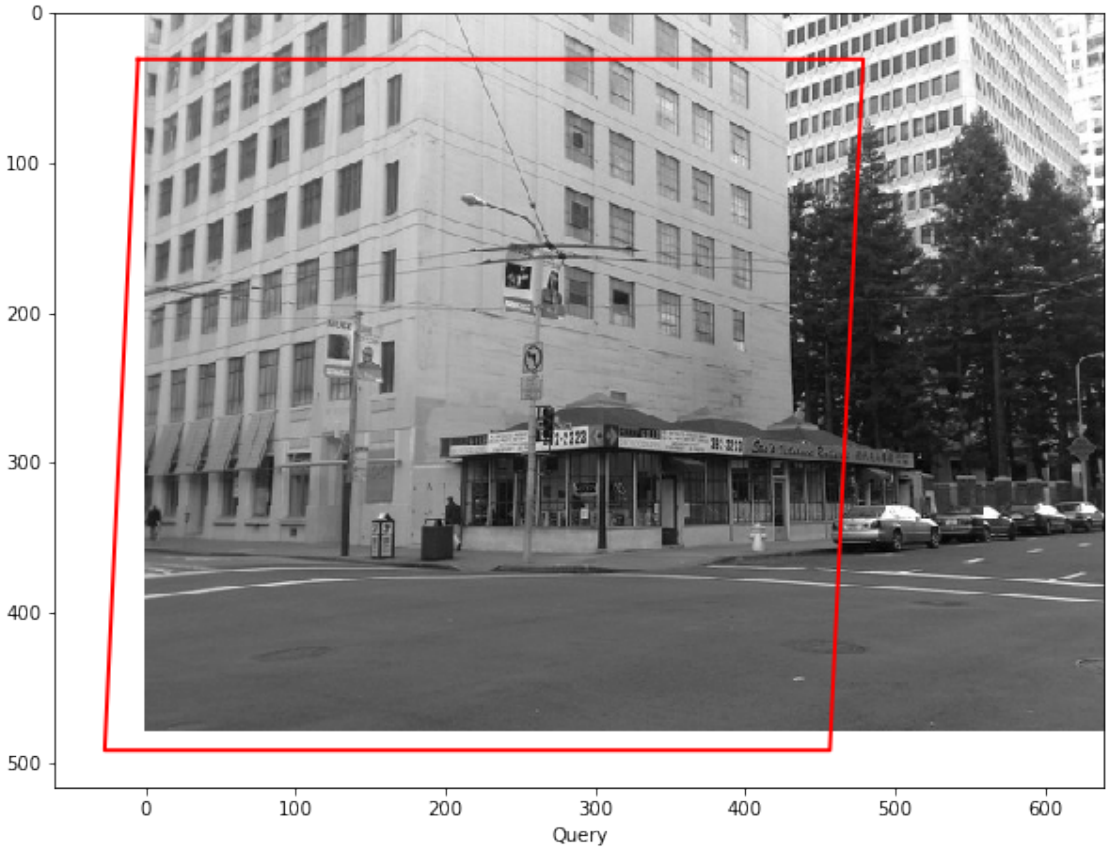

−100
0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700
Query

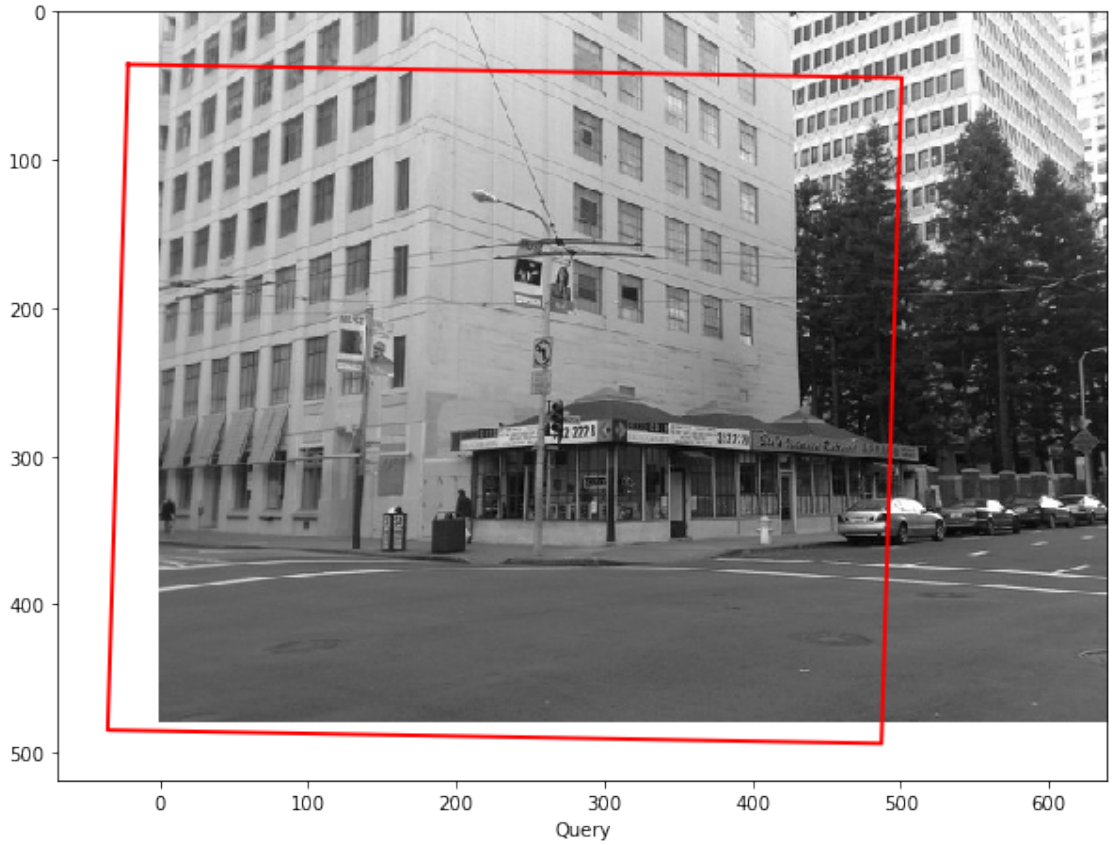


This feature matching case is using the 99th image in the landmarks categories. The reason why I choose this image is there is no significant content difference between the query and the reference. The failure isn't caused by the corner detector, in fact, the keypoints (black circles) on the image are all good quality corners, however, it is the match_descriptor step caused the failure. From the matches plot, it is observed that the there are several corners on the front of the building matched to the tree. Most of the keypoints on the front of the building are mismatched to the tree in reference building, which is mainly because there are various gaps between the leafs and are intrepreted as corners.In this case, the max_distance parameter should be added and specified in the match_descriptor() function to constrain the maximum distance between two possible matches, and in this case it's set to be 0.4.

```
Query image shape:  (480, 640)
Reference image shape:  (480, 640)
TP (inlier number): 7 FP (outlier number): 4 True Positive Rate:  0.64
```


Query vs Reference
0
100
200
300
400
0
200
400
600
800
1000
1200


0
100
200
300
400
500
0
100
200
300
400
500
600
Query

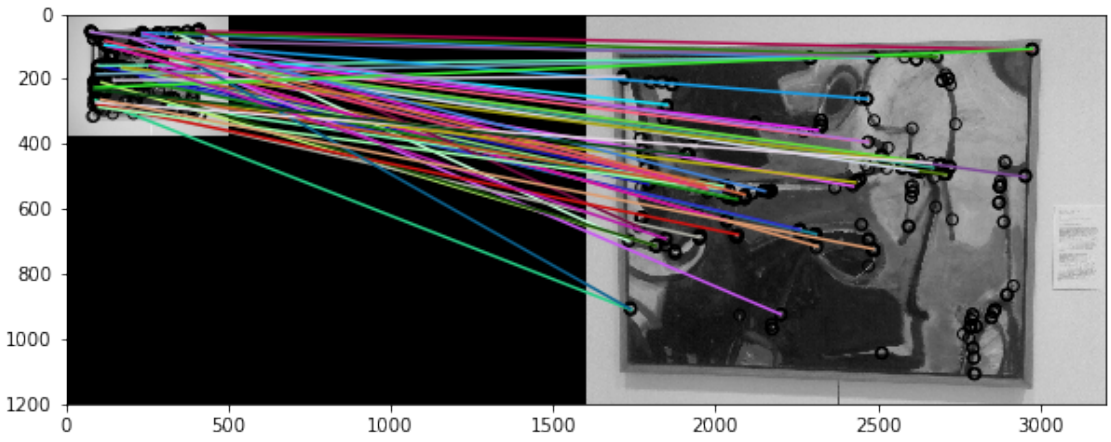


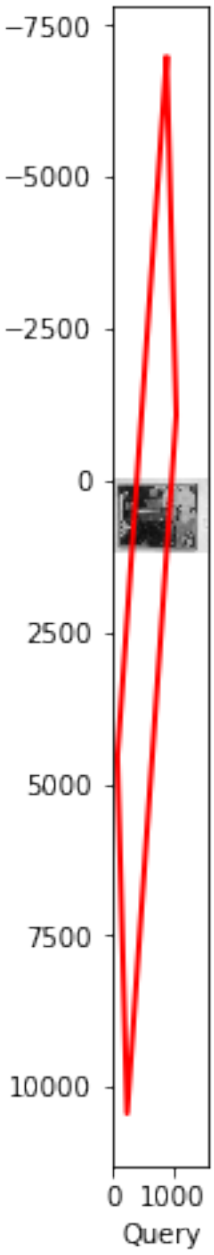
The query and the reference are in the same size of (480,640), so in this case, I didn't change the scale of the query image, however, from observation, it is shown that the brightness and the contrast between the query and the refernece are different. In the lecture, we learned that the descriptors are brightness and the contrast invariant, and some of them are scale invariant. For testing and experimenting, I modified the query image's brightness and contrast to make it close the reference image. However, from the match plot, there is no significant improvement in matching accuracy, and changing the brightness and the contrast couldn't lead to improvement. In the image there are also similar patterns like all the similar windows, so in the improvement, I reduced the max_ratio by 0.05 to deal with the ambiguious matches. However, almost of the keypoints are not correctly matched. The case is not very successful, but compared with the previous settings, this is an improvement in image matching performance.

```
Query image shape:  (1200, 1600)
Reference image shape:  (375, 500)
```

```
C:\Users\wang\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3334:
RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
C:\Users\wang\anaconda3\lib\site-packages\numpy\core\_methods.py:153:
RuntimeWarning: invalid value encountered in true_divide
  ret = um.true_divide(
C:\Users\wang\anaconda3\lib\site-packages\skimage\transform\_geometric.py:53:
```

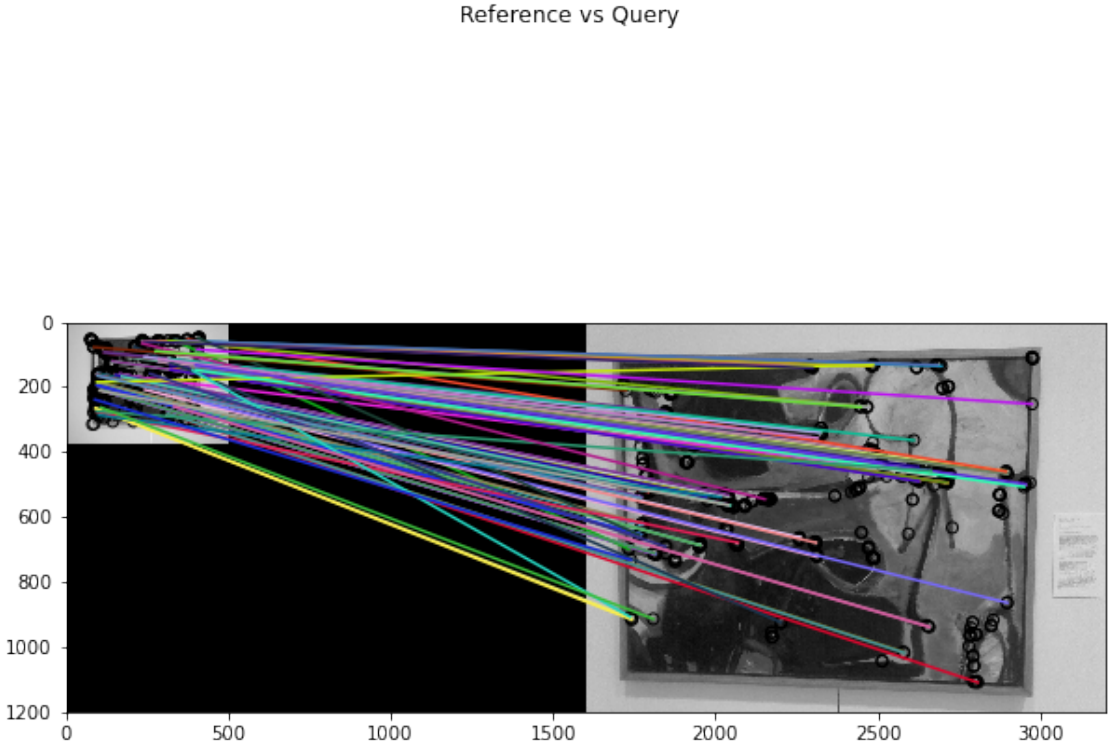
```
RuntimeWarning: invalid value encountered in double_scalars
  rms = math.sqrt(np.sum((points - centroid) ** 2) / points.shape[0])

TP (inlier number): 0 FP (outlier number): 52 True Positive Rate:  0.00
```

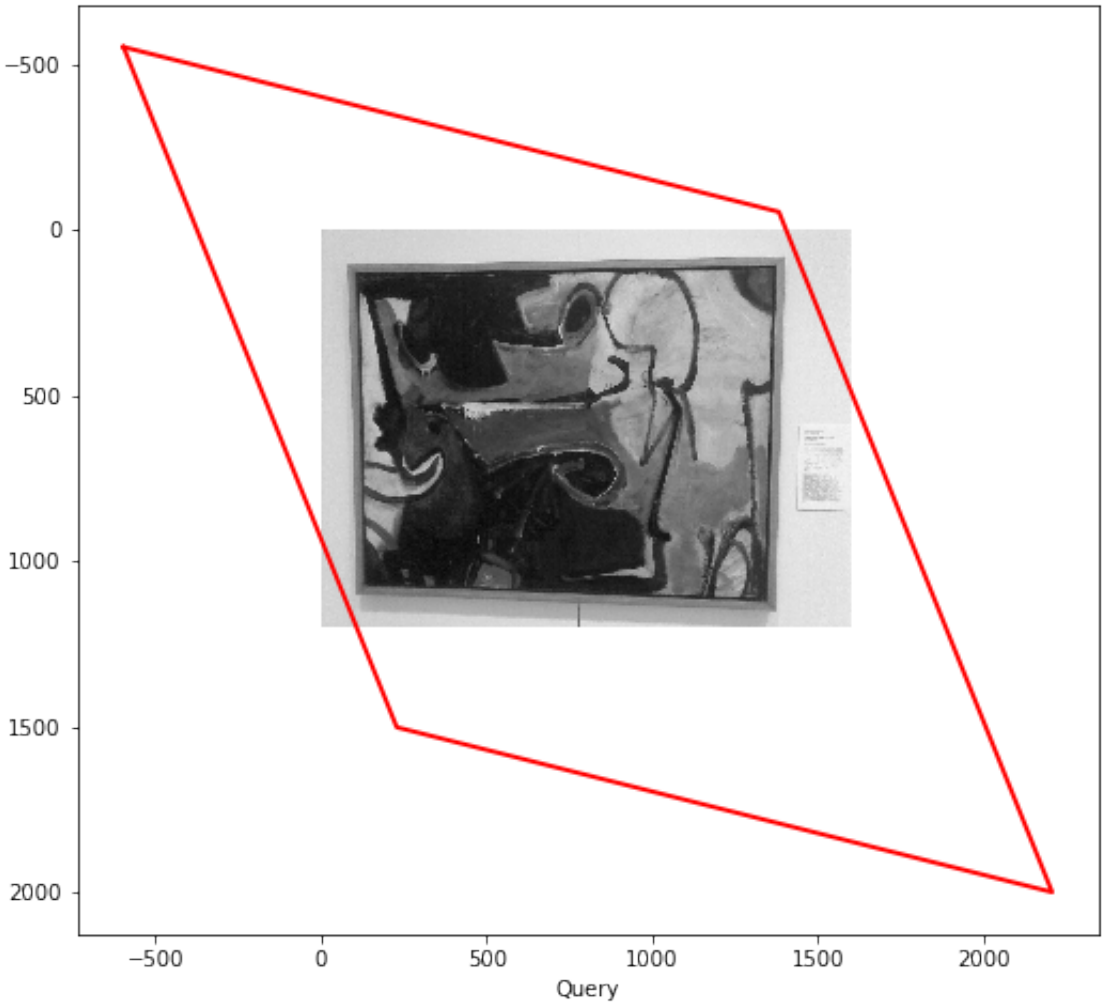



−7500
−5000
−2500
0
2500
5000
7500
10000
0
1000
Query

This is a failure because both the outline in linear least square and RANSAC couldn't oulined the reference image, with 0 TP, 52 FP and 0 TP rate. The match plot couldn't indicate whether they are good matches, because the reference image is too small, they have significant scale difference. The previous ORB settings doesn't work in this image pairs. To be specific, the n_scale should be tuned up or the query image should be scaled down, so that the pyramid can be constructed properly.

```
Query image shape:  (1200, 1600)
Reference image shape:  (375, 500)
TP (inlier number): 5 FP (outlier number): 92 True Positive Rate:  0.05
```


Reference vs Query
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500
3000

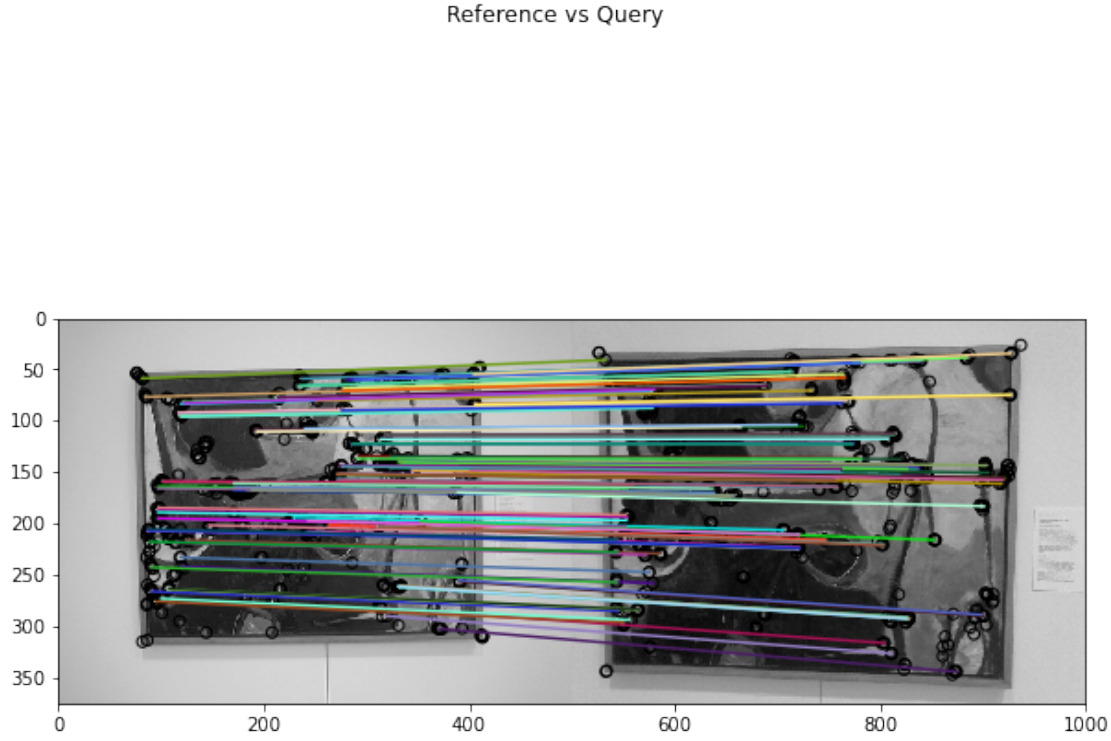

−500
0
500
1000
1500
2000
−500
0
500
1000
1500
2000
Query

In this time, the n_scale parameter is changed from 8 to 12, and the improvement shown in both outline plot and the TP rate. Althrough the linear least square transform estimation didn't output a proper estimation, RANSAC performed quite well, but both of these method have a better output than the previous setting.Next we will preprocess the query image using scaling.

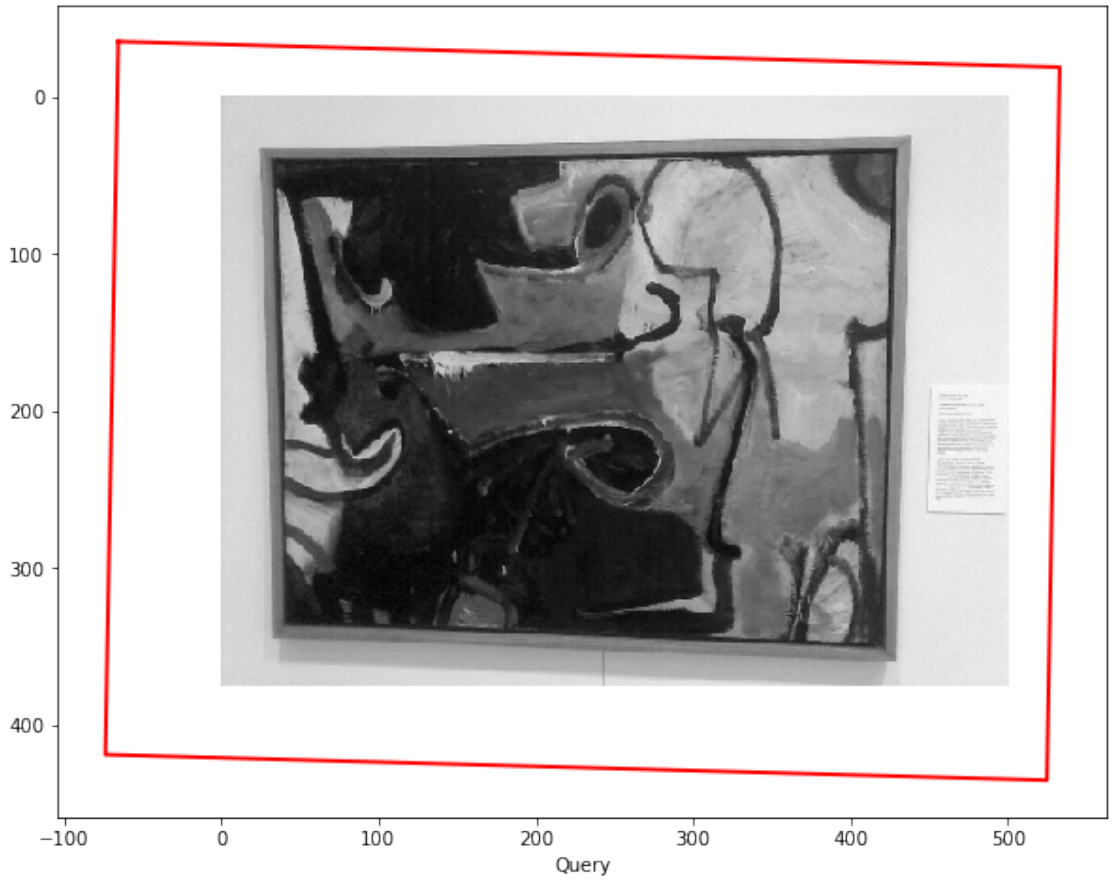
```
Query image shape:  (375, 500)
Reference image shape:  (375, 500)
```

museum painting query 011


```
TP (inlier number): 30 FP (outlier number): 94 True Positive Rate:  0.24
```


Reference vs Query
0
50
100
150
200
250
300
350
0
200
400
600
800
1000


0
100
200
300
400
0
100
200
300
400
500
Query



From the outputs image above, it is shown that after scaling to the similar size as the reference, the image matching performance is really improved, through the exact outline of the painting isn't outlined.This is due to the original reference isn't a picture of the painting only. In fact, the reference image is also a painting on the wall. Due to the extra wall information in the reference image, the image matching outline couldn't and shouldn't outline the exaxt painting out, and the outline in the query does indicate the image in the reference. Therefore, this image matching case can be defined as a success after the improvement. Most of parameters are maintained, except the max_ratio is reduced from 0,9 to 0.8, because the matches are sufficient and it's good to improve the quality of the matches by reducing the ratio and eliminating the ambiguious matches. Besides, due to the preprocess of scaling down, I changed the n_scales back to 8. After scaling, more keypoints are detected and more matches are correctly matched, as shown in the matches plot.With these data, transformation estimation can estimate an accurate transform and the outline is a success.

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. Hint 2: `skimage.io.ImageCollection()` may be useful here.
    3. 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?

```
The reference with highest score is:  54 ,which matches the 20 th in the
reference list
```

The Query image chosen is the 20th book cover. Using the above image matching method, the highest score is 54, which occurred in the 20th reference image, indicating an accurate match.

```
[27, 2, 1, 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 1, 1, 3,
1, 1, 1, 2]
q_pos:  0 highest_pois:  0
correct number:  1
[1, 24, 2, 2, 1, 2, 1, 1, 2, 1, 1, 2, 2, 1, 2, 3, 2, 1, 1, 1, 2, 2, 2, 2, 3, 2,
1, 1, 2, 2]
q_pos:  1 highest_pois:  1
correct number:  2
[2, 1, 34, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2,
1, 2, 2, 1]
q_pos:  2 highest_pois:  2
correct number:  3
[1, 1, 2, 13, 1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 2, 3, 2, 1, 1, 1, 2, 2, 1, 1, 1, 2,
2, 1, 2, 1]
q_pos:  3 highest_pois:  3
correct number:  4
[1, 2, 1, 1, 26, 1, 1, 1, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1, 2, 2, 3, 1,
1, 2, 2, 1]
q_pos:  4 highest_pois:  4
correct number:  5
[1, 2, 1, 2, 1, 38, 1, 1, 1, 1, 2, 3, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 3, 2, 1,
```

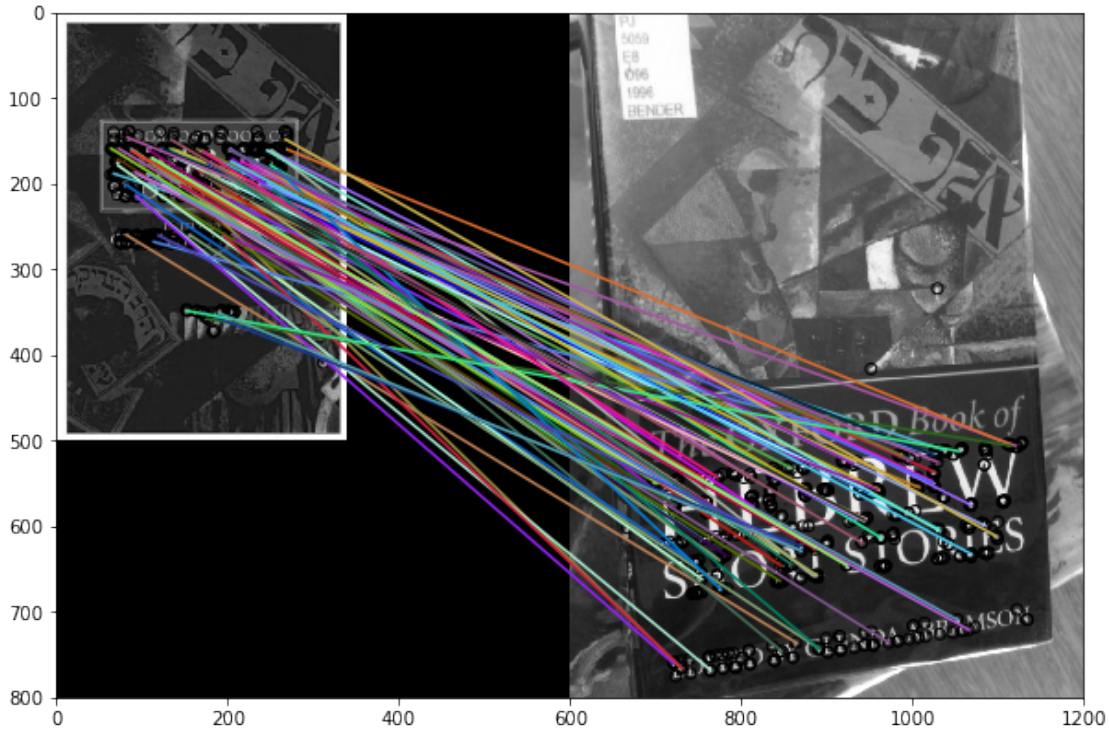
```
2, 1, 1, 2]
q_pos:  5 highest_pois:  5
correct number:  6
[1, 2, 2, 2, 1, 2, 35, 2, 1, 2, 1, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 1, 1]
q_pos:  6 highest_pois:  6
correct number:  7
[1, 3, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2,
2, 1, 2, 1]
q_pos:  7 highest_pois:  1
Not in the data set.
[1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 1,
1, 3, 2, 2]
q_pos:  8 highest_pois:  27
Not in the data set.
[2, 2, 2, 2, 2, 2, 2, 2, 3, 41, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 3, 2,
2, 2, 2, 2]
q_pos:  9 highest_pois:  9
correct number:  8
[1, 2, 1, 2, 2, 1, 2, 1, 1, 2, 48, 1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1,
1, 2, 2, 2]
q_pos:  10 highest_pois:  10
correct number:  9
[1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 8, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2,
1, 2, 2, 1]
q_pos:  11 highest_pois:  11
Not in the data set.
[1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 39, 2, 1, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 2,
1, 2, 2, 2]
q_pos:  12 highest_pois:  12
correct number:  10
[1, 2, 3, 2, 2, 1, 2, 2, 2, 2, 3, 2, 2, 30, 2, 2, 2, 1, 1, 2, 2, 3, 1, 2, 2, 1,
1, 2, 1, 2]
q_pos:  13 highest_pois:  13
correct number:  11
[2, 2, 2, 3, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 20, 2, 2, 2, 1, 2, 2, 3, 3, 1, 3, 1,
2, 3, 2, 2]
q_pos:  14 highest_pois:  14
correct number:  12
[2, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 5, 2, 2, 2, 2, 3, 2, 3, 2, 1, 3,
1, 1, 2, 1]
q_pos:  15 highest_pois:  15
Not in the data set.
[2, 2, 2, 2, 1, 1, 2, 3, 1, 1, 2, 3, 2, 2, 2, 3, 38, 2, 3, 2, 2, 2, 2, 2, 2, 2,
1, 2, 3, 2]
q_pos:  16 highest_pois:  16
correct number:  13
[1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 1, 1, 2, 2,
```

```
2, 2, 2, 3]
q_pos:  17 highest_pois:  29
Not in the data set.
[2, 2, 1, 1, 2, 2, 3, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 30, 1, 2, 2, 2, 2, 1, 2,
3, 2, 2, 2]
q_pos:  18 highest_pois:  18
correct number:  14
[1, 2, 1, 2, 1, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 1, 2, 2, 2, 60, 2, 2, 2, 2, 2, 2,
1, 3, 2, 2]
q_pos:  19 highest_pois:  19
correct number:  15
[1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 27, 2, 1, 2, 1, 2,
1, 2, 2, 2]
q_pos:  20 highest_pois:  20
correct number:  16
[1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 1, 2, 2, 2, 1, 1, 1, 1, 9, 2, 1, 2, 2,
2, 2, 2, 1]
q_pos:  21 highest_pois:  21
Not in the data set.
[1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 2, 2, 1, 3, 2, 2, 1, 1, 2, 4, 2, 2, 1,
3, 1, 2, 2]
q_pos:  22 highest_pois:  22
Not in the data set.
[1, 2, 2, 2, 1, 2, 3, 1, 1, 1, 1, 2, 2, 2, 2, 3, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2,
1, 2, 1, 1]
q_pos:  23 highest_pois:  6
Not in the data set.
[1, 2, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 3, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 28, 2,
2, 1, 2, 2]
q_pos:  24 highest_pois:  24
correct number:  17
[1, 1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 2, 3, 2, 2, 3, 1, 2, 2, 2, 2, 1, 3, 1, 2,
2, 2, 3, 2]
q_pos:  25 highest_pois:  13
Not in the data set.
[1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 4, 2,
2, 1, 2, 1]
q_pos:  26 highest_pois:  24
Not in the data set.
[1, 2, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 2, 2, 3, 1, 1, 1, 1, 2, 2, 3, 2, 2, 1,
2, 44, 2, 2]
q_pos:  27 highest_pois:  27
correct number:  18
[2, 1, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 4, 2]
q_pos:  28 highest_pois:  28
Not in the data set.
[1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 3, 2, 2,
```

```
3, 2, 2, 26]
q_pos:  29 highest_pois:  29
correct number:  19
accuracy percentage:  0.6333333333333333
```

The parameters are maintained from the past section, and the TP number are printed out when a query image has matched to every reference image in the set. The current query image location and index with the highest TP score is also returned, which indicates the index of the reference that gives the highest TP number. Therefore, if the index with the highest score is equal to the current query image index, then it indicates a correct match. The score threshold is set to be 10, that is if the TP number is less than 10, then this is not considered as an image match. If a top_k ranking method is usd, this will increase the accuracy rate significantly. For example, image 012, 016, 022, 023, 029 all have rank 1 for the correct image, but the TP number didn't reach the threshold. However, instead of using the threshold method, the top k ranking, the cases mentions will be matched correctly. For other failure cases,such as 008,009,018,024,027 are ranked around 15. For these cases, setting a 'k'(top-k) to be 15 is not appropriate, because there are only 30 images in total.Even if they are in the top 15 and considered to be a correct match, this function wouldn't be useful, because considering half of the set to be a correct matching image is not helpful in the real life. Therefore, for these images, we need to improve our feature detector and matcher, instead of changing the image pairing system. Overall, using a top-k method can improve the accuracy, but the 'k' value should be chosen reasonably. For example, k=5%*set_size i.e. chosing the top 5% of rank as correct image matches. Usually, a small k is preferred, but the actual value chosen should depend on the real applications, and it should be tuned according to the situation,like other parameters in ORB and match_descriptors(), to output an ideal result. In this test, there are 19 out of 30 images are correctly matched, with an accuracy of 63%. The cell below are some specific failure cases for deeper analysis.

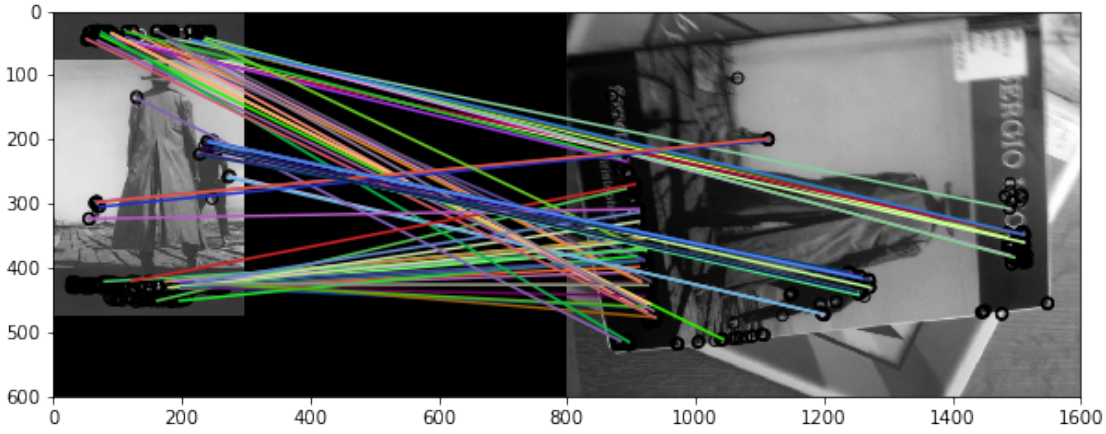
```
Query image shape:  (800, 600) Reference image shape:  (500, 340)
TP:  2 TPR:  0.025974025974025976
```

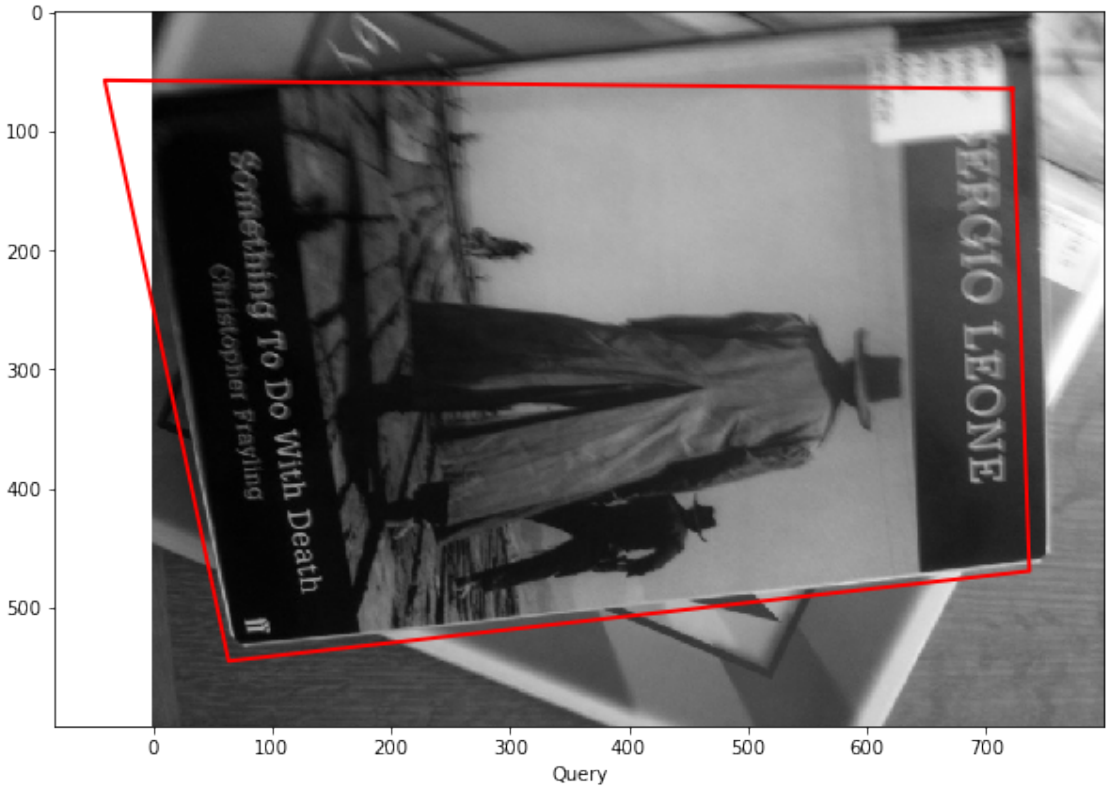

Reference vs Query
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200

Above is the case of book cover 008 (q_pos:7). The main issue here is that the book cover in reference image and the query image are not the same, but only with small layout differences. Firstly, the scale difference is big (800, 600) vs (500, 340), without tuning the n_scalse parameter,this can affect the keypoints matching process. Besides, the repeating letters could also misleading the keypoints matching. From the match plot, it is shown that there are mismatched matches associated with the repeating letters.

```
TP:  11 TPR:  0.1774193548387097
```

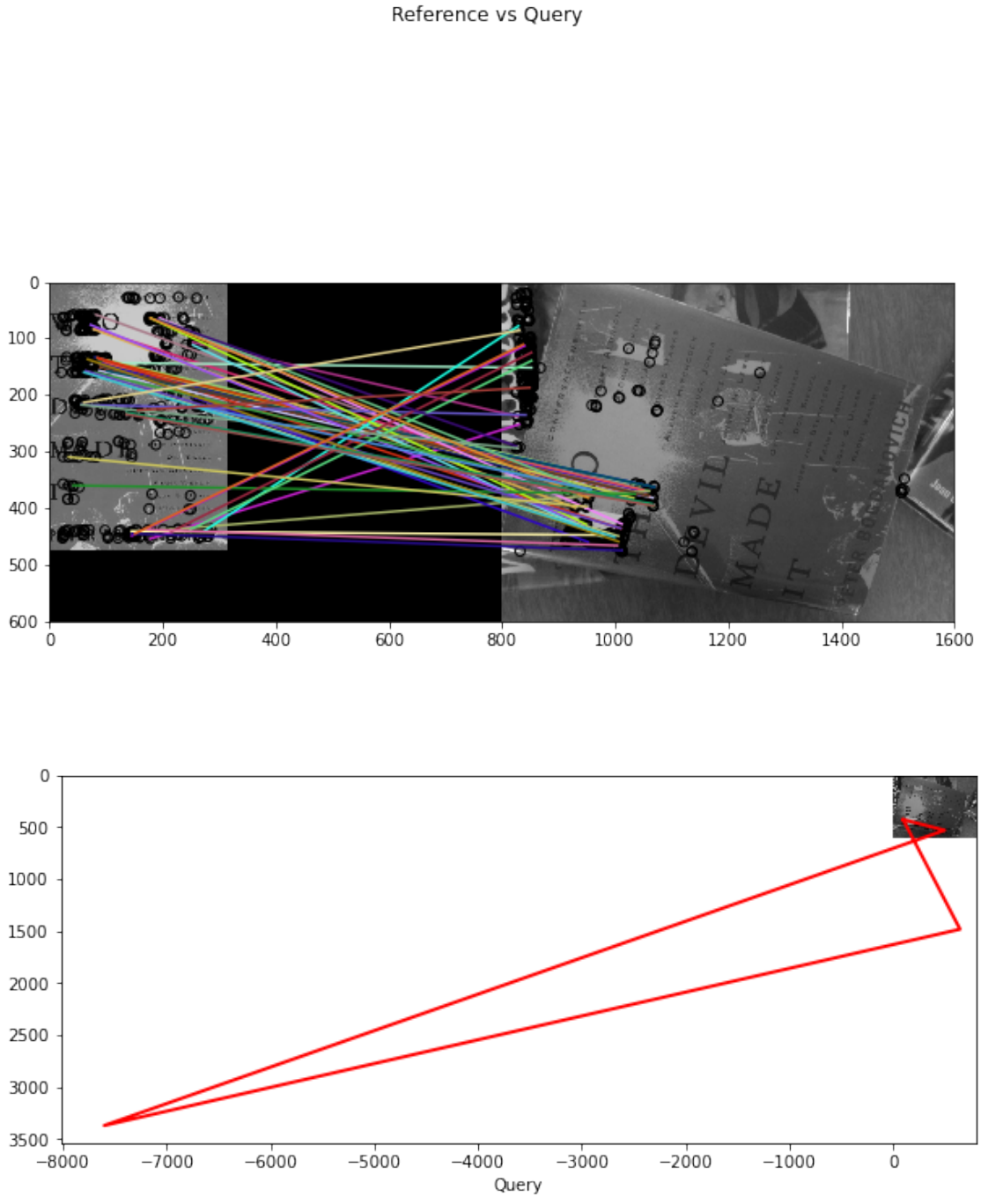
Reference vs Query





From the match plot,img 022's failure is mainly about the top title of the book mismatched to the bottom letters, these are a majority of the outliers.

```
TP:  7 TPR:  0.12962962962962962
```



From the match plot, apart from the mismatch of the letters, it is also shown that many keypoints

in the reference book cover are matched to the corners detected outside the book cover. The book under the book cover is the main distraction in the feature matching process.

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.

```
[2, 2, 1, 2, 1, 2, 1, 1, 1, 2, 47, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 2, 3, 2, 2,
1, 2, 2, 2]
q_pos:  0 highest_pos:  10
correct number:  1
[1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 5, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 2,
2, 2, 2, 1]
q_pos:  1 highest_pos:  11
Not in the data set.
[1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 35, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 1, 2, 2,
1, 1, 2, 1]
q_pos:  2 highest_pos:  12
correct number:  2
[1, 1, 2, 3, 2, 1, 2, 2, 3, 1, 2, 2, 3, 33, 1, 4, 2, 1, 1, 1, 2, 3, 1, 2, 1, 1,
2, 2, 2, 2]
q_pos:  3 highest_pos:  13
correct number:  3
[2, 2, 1, 2, 1, 1, 2, 3, 2, 1, 1, 2, 2, 1, 24, 2, 2, 1, 1, 2, 2, 2, 2, 2, 3, 2,
2, 3, 2, 2]
q_pos:  4 highest_pos:  14
correct number:  4
[1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 3, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 3, 2, 1, 2,
1, 2, 1, 1]
q_pos:  5 highest_pos:  11
Not in the data set.
[1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1, 1, 2, 3, 37, 1, 2, 1, 2, 2, 2, 2, 3, 2,
1, 1, 2, 1]
q_pos:  6 highest_pos:  16
correct number:  5
[1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 3, 2, 3, 1, 1, 2, 2, 1, 2, 2, 2,
1, 2, 2, 3]
q_pos:  7 highest_pos:  15
Not in the data set.
[1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 3, 24, 1, 2, 2, 1, 4, 2, 2,
1, 1, 2, 2]
q_pos:  8 highest_pos:  18
correct number:  6
[1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 3, 1, 2, 1, 2, 58, 2, 3, 1, 2, 3, 2,
2, 2, 2, 1]
q_pos:  9 highest_pos:  19
correct number:  7
```

```
[1, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 3, 1, 2, 1, 1, 2, 1, 2, 29, 2, 2, 1, 2, 2,
1, 1, 2, 1]
q_pos:  10 highest_pos:  20
correct number:  8
[1, 2, 1, 1, 1, 1, 2, 3, 1, 1, 2, 1, 2, 1, 2, 2, 4, 1, 1, 2, 1, 12, 1, 1, 1, 2,
2, 2, 1, 2]
q_pos:  11 highest_pos:  21
correct number:  9
[1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 3, 2, 1, 2, 1, 2, 4, 2, 2, 1,
1, 1, 1, 1]
q_pos:  12 highest_pos:  22
Not in the data set.
[1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 3, 2, 1,
1, 2, 1, 1]
q_pos:  13 highest_pos:  23
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 2, 2, 3, 2, 2, 2, 1, 2, 1, 1, 1, 2, 1, 2, 2, 2, 1, 24, 2,
2, 2, 1, 2]
q_pos:  14 highest_pos:  24
correct number:  10
[1, 2, 2, 2, 1, 1, 1, 1, 3, 1, 2, 2, 2, 3, 2, 1, 2, 1, 2, 1, 3, 2, 2, 3, 1, 13,
3, 1, 3, 2]
q_pos:  15 highest_pos:  25
correct number:  11
[1, 1, 1, 2, 0, 1, 2, 1, 1, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2,
1, 1, 2, 2]
q_pos:  16 highest_pos:  3
Not in the data set.
[3, 2, 1, 2, 1, 3, 1, 2, 1, 1, 2, 1, 1, 2, 2, 3, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2,
1, 40, 2, 2]
q_pos:  17 highest_pos:  27
correct number:  12
[3, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,
1, 2, 4, 1]
q_pos:  18 highest_pos:  28
Not in the data set.
[1, 3, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 1, 2, 5, 1, 1, 2, 1, 2, 3, 2, 2, 1,
3, 2, 1, 25]
q_pos:  19 highest_pos:  29
correct number:  13
[2, 3, 1, 3, 2, 1, 2, 2, 2, 3, 1, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1,
2, 2, 1, 2]
q_pos:  20 highest_pos:  1
Not in the data set.
[1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 1, 3, 1, 1, 2, 2, 2, 2, 1, 1, 2, 3, 1, 2, 2, 1,
2, 2, 2, 2]
q_pos:  21 highest_pos:  11
Not in the data set.
```

```
[2, 2, 1, 1, 1, 1, 3, 3, 1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 2,
2, 1, 1, 2]
q_pos:  22 highest_pos:  6
Not in the data set.
[1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 2, 3, 1, 1, 2, 3, 1, 1, 2, 2, 2, 3, 1, 2, 2, 2,
2, 2, 2, 2]
q_pos:  23 highest_pos:  11
Not in the data set.
[2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2, 1, 1, 2, 1,
1, 2, 2, 2]
q_pos:  24 highest_pos:  0
Not in the data set.
[2, 2, 1, 2, 1, 2, 1, 2, 3, 1, 2, 2, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1, 2,
1, 2, 2, 2]
q_pos:  25 highest_pos:  8
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1,
1, 2, 1, 1]
q_pos:  26 highest_pos:  7
Not in the data set.
[2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 3, 1, 2, 1, 1, 1, 2, 2, 2, 3, 2, 2, 1,
2, 1, 1, 2]
q_pos:  27 highest_pos:  13
Not in the data set.
[2, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2,
2, 3, 2, 1]
q_pos:  28 highest_pos:  27
Not in the data set.
[2, 2, 4, 1, 1, 1, 2, 2, 2, 2, 3, 2, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1,
2, 2, 2, 1]
q_pos:  29 highest_pos:  2
Not in the data set.
accuracy percentage with 10 extra images:  0.65
```

The cell above match two image set with 10 different images. In this task, I modified the query set, and make the query set input the query images from the 11th image to the 40 th image, and the true match has 10 index difference. To calculate the accuracy, the demoninator should be changed to 20, because there are only 20 true matched images maximum. In this run, the accuracy is found to be 65%, which is higher than the previous 63%, but close. Even if the setting are the same,the increase of accuracy is reasonable, because the RANSAC is a random sampling and consensus method. Therefore, even if all the parameters of the RANSAC are the same, the outputs at each run could be different due to the randomness in RANSAC.However,the difference won't be too large, because its accuracy and robustness. Furthermore, in this task, there is no case of mismatch of image, that is there is no case of a image computed to be matched to a wrong image. If it couldn't find the correct one, it will be 'not in the data set'.

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.

```
[6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2,
2, 1, 1, 2]
q_pos:  0 highest_pois:  0
Not in the data set.
```

```
C:\Users\wang\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3334:
RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
C:\Users\wang\anaconda3\lib\site-packages\numpy\core\_methods.py:153:
RuntimeWarning: invalid value encountered in true_divide
  ret = um.true_divide(
C:\Users\wang\anaconda3\lib\site-packages\skimage\transform\_geometric.py:53:
RuntimeWarning: invalid value encountered in double_scalars
  rms = math.sqrt(np.sum((points - centroid) ** 2) / points.shape[0])
```

```
[1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 1, 1]
q_pos:  1 highest_pois:  4
Not in the data set.
[1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 2, 1]
q_pos:  2 highest_pois:  2
Not in the data set.
[2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 2, 2, 2]
q_pos:  3 highest_pois:  0
Not in the data set.
[1, 1, 1, 2, 1, 1, 2, 1, 1, 0, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1]
q_pos:  4 highest_pois:  3
Not in the data set.
[1, 2, 1, 1, 2, 9, 2, 2, 1, 1, 1, 2, 2, 1, 3, 2, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1,
2, 1, 3, 2]
q_pos:  5 highest_pois:  5
Not in the data set.
[1, 1, 2, 2, 1, 2, 9, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2,
2, 1, 1, 1]
q_pos:  6 highest_pois:  6
Not in the data set.
[1, 2, 2, 3, 2, 2, 2, 29, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 2, 3, 1,
1, 1, 2, 2]
q_pos:  7 highest_pois:  7
correct number:  1
[2, 1, 1, 2, 1, 2, 1, 3, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 2, 2,
1, 1, 2, 2]
q_pos:  8 highest_pois:  7
```

```
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 2, 1, 12, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1,
1, 1, 2, 1]
q_pos:  9 highest_pois:  9
correct number:  2
[1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
2, 1, 2, 1]
q_pos:  10 highest_pois:  12
Not in the data set.
[1, 3, 1, 2, 1, 1, 1, 1, 1, 1, 1, 21, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 1, 1, 1,
2, 1, 2, 2]
q_pos:  11 highest_pois:  11
correct number:  3
[3, 2, 1, 2, 3, 2, 2, 2, 1, 3, 1, 2, 4, 1, 1, 1, 1, 2, 1, 2, 1, 1, 3, 2, 1, 2,
2, 3, 2, 1]
q_pos:  12 highest_pois:  12
Not in the data set.
[1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
1, 1, 2, 1]
q_pos:  13 highest_pois:  13
Not in the data set.
[1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1,
1, 1, 2, 2]
q_pos:  14 highest_pois:  3
Not in the data set.
[1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
2, 1, 2, 2]
q_pos:  15 highest_pois:  1
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 45, 1, 1, 1, 1, 1, 1, 2, 1, 1,
2, 1, 2, 1]
q_pos:  16 highest_pois:  16
correct number:  4
[1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 23, 0, 1, 1, 1, 1, 1, 2, 2,
2, 1, 1, 2]
q_pos:  17 highest_pois:  17
correct number:  5
[1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 2, 2]
q_pos:  18 highest_pois:  7
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1,
1, 2, 1, 1]
q_pos:  19 highest_pois:  12
Not in the data set.
[1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 1, 2, 2, 1, 1, 7, 1, 2, 2, 2, 2,
2, 1, 3, 2]
q_pos:  20 highest_pois:  20
```

```
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 0, 1, 1]
q_pos:  21 highest_pois:  8
Not in the data set.
[1, 1, 1, 2, 2, 3, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 3, 2, 1, 3, 2,
2, 2, 2, 2]
q_pos:  22 highest_pois:  5
Not in the data set.
[1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 1, 4, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 2, 1,
2, 1, 1, 2]
q_pos:  23 highest_pois:  11
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 2, 3,
2, 0, 2, 2]
q_pos:  24 highest_pois:  11
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 3, 1,
2, 1, 1, 2]
q_pos:  25 highest_pois:  24
Not in the data set.
[1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1, 1, 2, 1]
q_pos:  26 highest_pois:  28
Not in the data set.
[1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 0, 1, 2, 1,
1, 2, 1, 2]
q_pos:  27 highest_pois:  3
Not in the data set.
[1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2,
2, 2, 45, 1]
q_pos:  28 highest_pois:  28
correct number:  6
[1, 1, 1, 1, 3, 1, 1, 2, 1, 1, 2, 2, 1, 0, 1, 1, 1, 1, 0, 1, 2, 0, 1, 1, 1, 1,
1, 1, 2, 1]
q_pos:  29 highest_pois:  4
Not in the data set.
accuracy percentage:  0.2
```

In the museum paintings set, we can find that the matching accuracy is way lower than the book covers, which only have 20% accuracy. From the printed TP arrays, we can see that for (q_pos = img number -1) q_pos = 2,5,6,12,13,20, the index with the highest socre equals to its query image location, therefore, it is the threshold value setting issue that lowers the accuracy. In the improvement section, the threshold will be changed from 10 to 3. The ORB and match_descriptor parameter settings will be kept the same, because this setting is used previously and it is not possible to have a setting that's suitable for every image. As long as the setting is suitable for a reasonable majority of the image, then it's good.

```
C:\Users\wang\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3334:
```

```
RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
C:\Users\wang\anaconda3\lib\site-packages\numpy\core\_methods.py:153:
RuntimeWarning: invalid value encountered in true_divide
  ret = um.true_divide(
C:\Users\wang\anaconda3\lib\site-packages\skimage\transform\_geometric.py:53:
RuntimeWarning: invalid value encountered in double_scalars
  rms = math.sqrt(np.sum((points - centroid) ** 2) / points.shape[0])
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 2,
2, 1, 1, 1]
q_pos:  0 highest_pos:  11
Not in the data set.
[1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 21, 1, 0, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1,
2, 1, 2, 2]
q_pos:  1 highest_pos:  11
correct number:  1
[1, 1, 1, 2, 2, 2, 3, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1,
2, 3, 2, 3]
q_pos:  2 highest_pos:  6
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 1, 8, 1, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 1,
2, 1, 2, 1]
q_pos:  3 highest_pos:  13
Not in the data set.
[1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
2, 2, 2, 2]
q_pos:  4 highest_pos:  3
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 2, 1, 1]
q_pos:  5 highest_pos:  7
Not in the data set.
[1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 45, 1, 1, 1, 1, 1, 1, 1, 2, 1,
2, 1, 2, 1]
q_pos:  6 highest_pos:  16
correct number:  2
[1, 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 22, 1, 1, 1, 1, 1, 2, 2, 1,
3, 1, 1, 2]
q_pos:  7 highest_pos:  17
correct number:  3
[1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2]
q_pos:  8 highest_pos:  2
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 2, 3, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
1, 1, 2, 2]
q_pos:  9 highest_pos:  12
```

```
Not in the data set.
[2, 1, 2, 2, 2, 2, 2, 2, 3, 1, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2,
3, 2, 2, 3]
q_pos:  10 highest_pos:  8
Not in the data set.
[1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2,
1, 1, 2, 1]
q_pos:  11 highest_pos:  2
Not in the data set.
[1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 3, 2, 2, 2,
2, 2, 2, 2]
q_pos:  12 highest_pos:  22
Not in the data set.
[1, 2, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 3, 2, 1,
2, 1, 1, 1]
q_pos:  13 highest_pos:  23
Not in the data set.
[1, 0, 1, 2, 1, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 2, 1,
1, 0, 2, 1]
q_pos:  14 highest_pos:  3
Not in the data set.
[1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1,
1, 1, 2, 2]
q_pos:  15 highest_pos:  4
Not in the data set.
[1, 0, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 2, 1]
q_pos:  16 highest_pos:  8
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
2, 2, 2, 2]
q_pos:  17 highest_pos:  7
Not in the data set.
[1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 1, 41, 1]
q_pos:  18 highest_pos:  28
correct number:  4
[1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 3, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 2, 1]
q_pos:  19 highest_pos:  12
Not in the data set.
[1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1, 0, 0, 2, 1, 1, 1, 1, 1, 0, 1, 2,
1, 1, 1, 2]
q_pos:  20 highest_pos:  1
Not in the data set.
[1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
2, 2, 2, 1]
q_pos:  21 highest_pos:  3
```

```
Not in the data set.
[1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
2, 1, 2, 2]
q_pos:  22 highest_pos:  5
Not in the data set.
[1, 2, 1, 3, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
3, 1, 1, 1]
q_pos:  23 highest_pos:  3
Not in the data set.
[1, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 2, 2, 1, 1, 0, 2, 1, 2, 0, 1, 1, 1, 1, 2, 1,
3, 0, 2, 2]
q_pos:  24 highest_pos:  26
Not in the data set.
[2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1,
3, 4, 2, 2]
q_pos:  25 highest_pos:  27
Not in the data set.
[1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 1, 1, 1]
q_pos:  26 highest_pos:  5
Not in the data set.
[1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 1, 1, 2, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 3]
q_pos:  27 highest_pos:  29
Not in the data set.
[1, 1, 1, 1, 2, 3, 2, 2, 1, 1, 1, 2, 1, 2, 0, 1, 1, 1, 1, 1, 0, 1, 2, 1, 1, 1,
1, 1, 2, 1]
q_pos:  28 highest_pos:  5
Not in the data set.
[1, 1, 1, 1, 2, 2, 1, 1, 4, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 2,
1, 1, 3, 2]
q_pos:  29 highest_pos:  8
Not in the data set.
accuracy percentage with 10 extra images:  0.2
```

The accuracy is 20%, which is the same as the previous task. 4 image pairs out of 20 are correctly matched. As previous case, there are several image pairs are detected, but with a low TP value and failed to pass the threshold, for example image004,013,014 and 018. If the threshold is reduced, the accuracy could be increased to 40%. Furthermore, like the task in the book cover set, the museum paintings also don't have image mismatching issue. there is no case of mismatch of image, that is there is no case of a image computed to be matched to a wrong image. If it couldn't find the correct one, it will be 'not in the data set'.

```
TP:  7 TPR:  0.11475409836065574
```

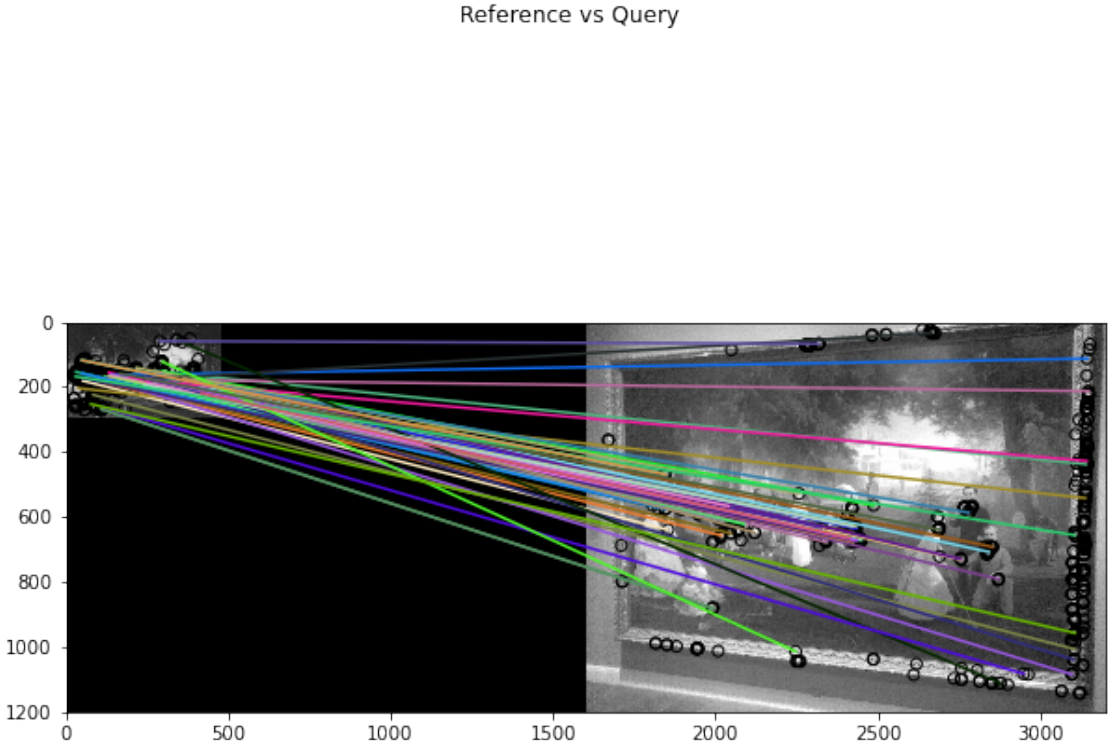
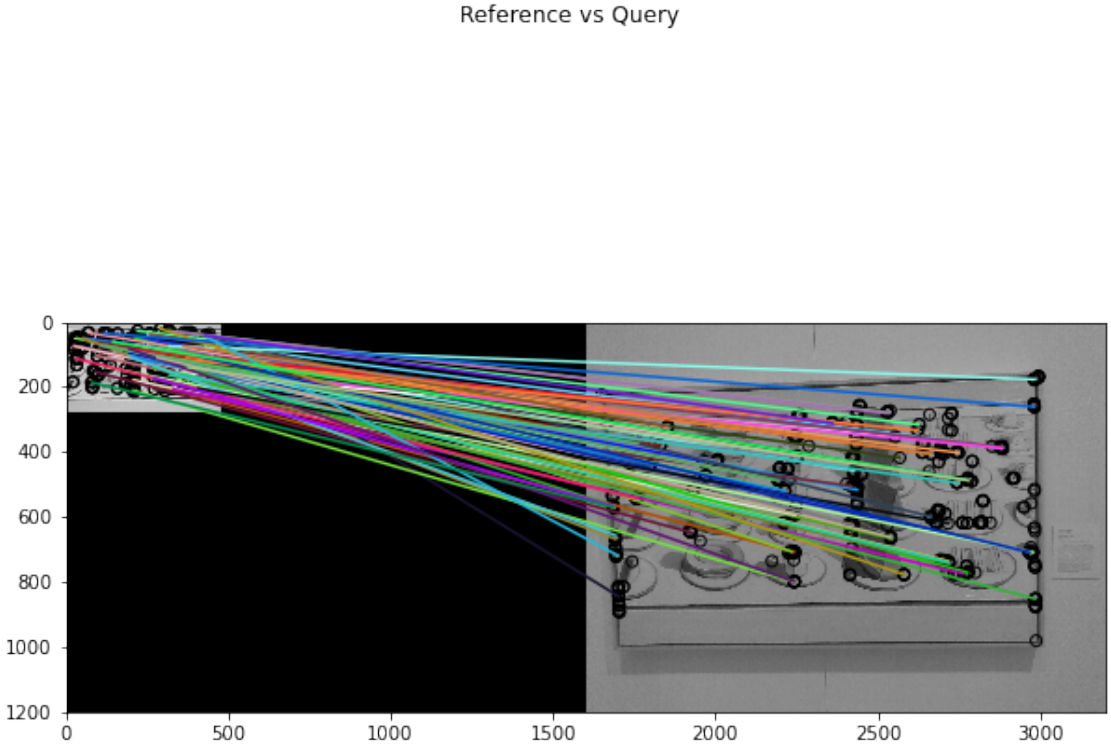

Reference vs Query
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500
3000

Image 001 is ranked highest in the list, but its TP number couldn't reach the threshold of 10. From the outline, it is observed that the top right tree isn't outlined inside, and the match plot also shows that there is a lack of keypoints and matches. Therefore, the detection in that region is poor performed.Furthermore, it is also shown that the scale difference between the query and the reference is very large, therefore, for this image pair, the n_scale should be tuned up to achieve a better performance.

```
TP:  1 TPR:  0.0136986301369863
```


Reference vs Query
0
200
400
600
800
1000
1200
0
500
1000
1500
2000
2500
3000

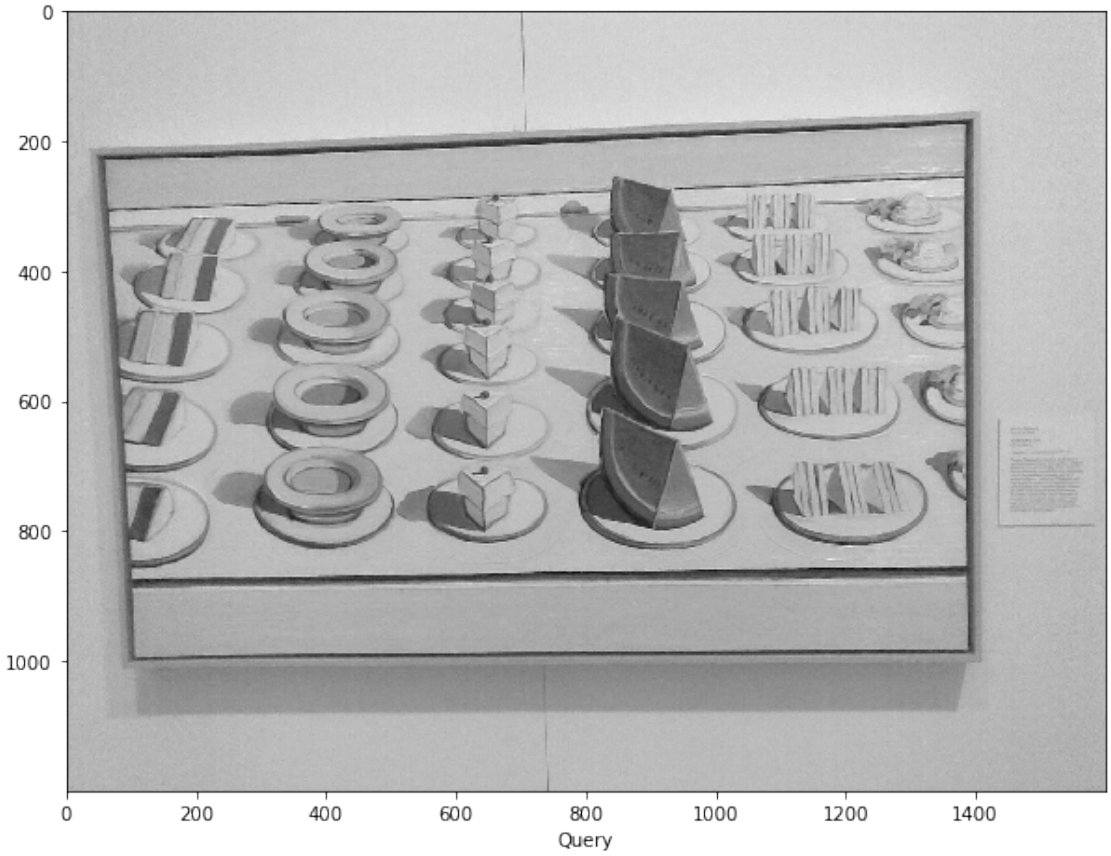


Image 19 is a painting with many repeating patterns, and these ambiguious corners can be easily mismatches to other corners. In the match plot, the similar corners are mismatched and it's difficult for RANSAC to estimate the transformation with a large number of wrong matches. For this image, it's better to tune down the max_ratio parameter in match_descriptors() function to have a proper transform estimation. However, this is an extreme case in the set, and in the improvement section, I won't tune up this parameter just for this specific case. Becasue, if max_ratio is small,other images may not have sufficient matches for RANSAC.

```
[5, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 2,
2, 1, 2, 2]
q_pos:  0 highest_pois:  0
correct number:  1
[1, 0, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2]
q_pos:  1 highest_pois:  3
Not in the data set.
[1, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 1, 2]
q_pos:  2 highest_pois:  2
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
1, 2, 2, 1]
q_pos:  3 highest_pois:  10
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 2]
q_pos:  4 highest_pois:  12
Not in the data set.
[3, 1, 1, 1, 2, 3, 2, 2, 1, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 3,
2, 2, 2, 2]
q_pos:  5 highest_pois:  0
[1, 1, 1, 2, 1, 1, 8, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2,
3, 2, 2, 1]
q_pos:  6 highest_pois:  6
correct number:  2
[1, 1, 1, 2, 2, 2, 1, 30, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2,
2, 1, 1, 2]
q_pos:  7 highest_pois:  7
correct number:  3
[1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 0, 1, 1, 1, 1, 1, 2, 2, 1, 2, 3, 1, 1, 2, 1, 2,
2, 1, 2, 2]
q_pos:  8 highest_pois:  20
[1, 1, 1, 2, 1, 1, 1, 1, 1, 14, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 2, 1]
q_pos:  9 highest_pois:  9
correct number:  4
[1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 2, 1]
q_pos:  10 highest_pois:  5
Not in the data set.
[1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 19, 1, 0, 1, 0, 1, 2, 0, 1, 2, 1, 1, 1, 1, 1,
2, 1, 2, 2]
q_pos:  11 highest_pois:  11
correct number:  5
[1, 1, 3, 2, 2, 2, 2, 2, 1, 3, 1, 2, 3, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 3, 2, 2,
1, 3, 3, 2]
q_pos:  12 highest_pois:  2
[1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 5, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1,
1, 1, 2, 1]
q_pos:  13 highest_pois:  13
correct number:  6
[2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1,
1, 1, 2, 2]
q_pos:  14 highest_pois:  0
Not in the data set.
[1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, 1, 3, 1]
q_pos:  15 highest_pois:  4
[1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 2, 49, 1, 1, 1, 1, 1, 2, 1, 2, 1,
```

```
2, 1, 2, 2]
q_pos:  16 highest_pois:  16
correct number:  7
[2, 2, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 26, 1, 1, 1, 1, 1, 1, 1, 2,
3, 1, 1, 1]
q_pos:  17 highest_pois:  17
correct number:  8
[1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 2, 2]
q_pos:  18 highest_pois:  2
Not in the data set.
[1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
1, 1, 1, 2]
q_pos:  19 highest_pois:  12
[2, 3, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2,
2, 2, 2, 3]
q_pos:  20 highest_pois:  1
[1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 2, 1,
2, 1, 1, 1]
q_pos:  21 highest_pois:  1
Not in the data set.
[2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2,
2, 3, 2, 2]
q_pos:  22 highest_pois:  27
[1, 1, 1, 2, 4, 1, 2, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1,
1, 1, 1, 1]
q_pos:  23 highest_pois:  4
[1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 2, 1, 2, 2,
1, 0, 2, 1]
q_pos:  24 highest_pois:  4
Not in the data set.
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 0, 1, 0, 1, 1, 1, 2, 1, 1, 1, 2, 1,
1, 1, 2, 2]
q_pos:  25 highest_pois:  11
Not in the data set.
[1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1,
1, 1, 2, 1]
q_pos:  26 highest_pois:  22
Not in the data set.
[1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
2, 2, 1, 2]
q_pos:  27 highest_pois:  1
Not in the data set.
[2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
3, 2, 44, 1]
q_pos:  28 highest_pois:  28
correct number:  9
[1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1,
```

```
3, 1, 2, 1]
q_pos:  29 highest_pois:  26
accuracy percentage:  0.3
```

With a lower threshold of 3 in the system, the accuracy increased to 30%. With a decrease in the threshold, the case like q_pos = 5,the index with the highest score equals to its query image location, they are all considered to be correct pairs i.e. all the q_pos = max_score_pos case are considered as correct image matches. Therefore, lowering the threshold increased the image matching accuracy. However, the features keypoints in museum paintings are more difficult to be detected and matched than the book covers. For a better corner detection and descriptor matching, a better descriptor is needed.