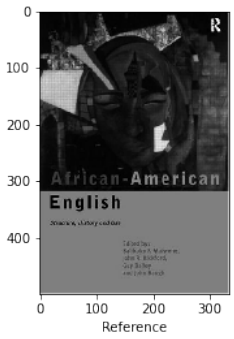
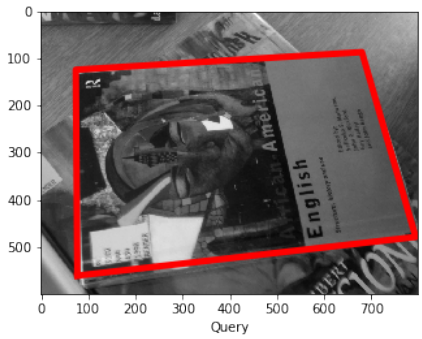
Assignment2

April 22, 2024

1 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.

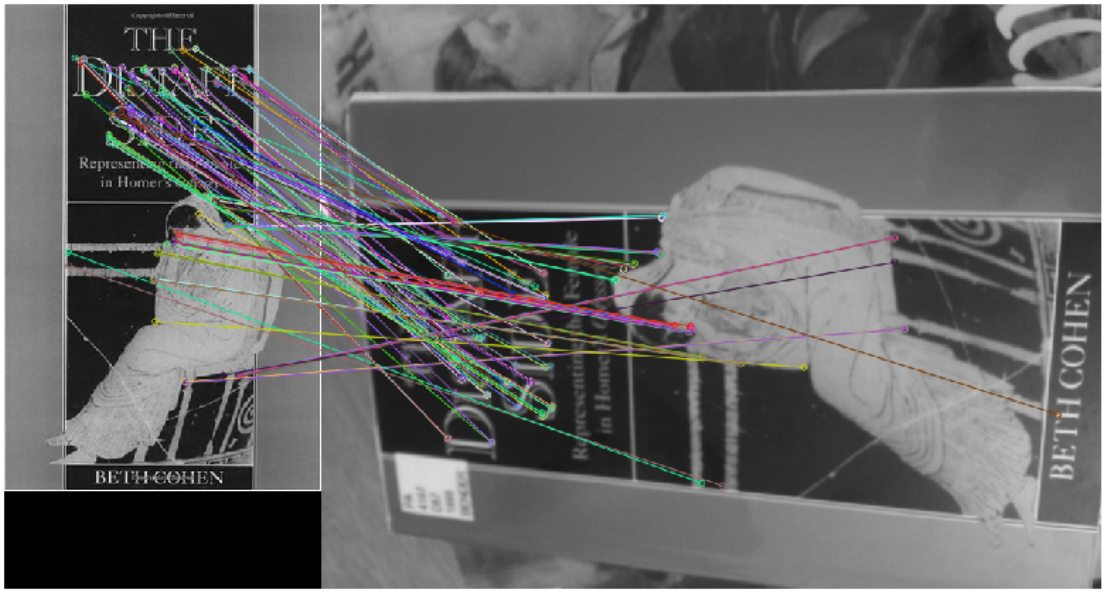
1. [Load images] Load the first (reference, query) image pair from the "book_covers" category using opencv (e.g. `img=cv2.imread()`). Check the parameter option in " cv2.imread()" to ensure that you read the gray scale image, since it is necessary for computing ORB features.

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`.

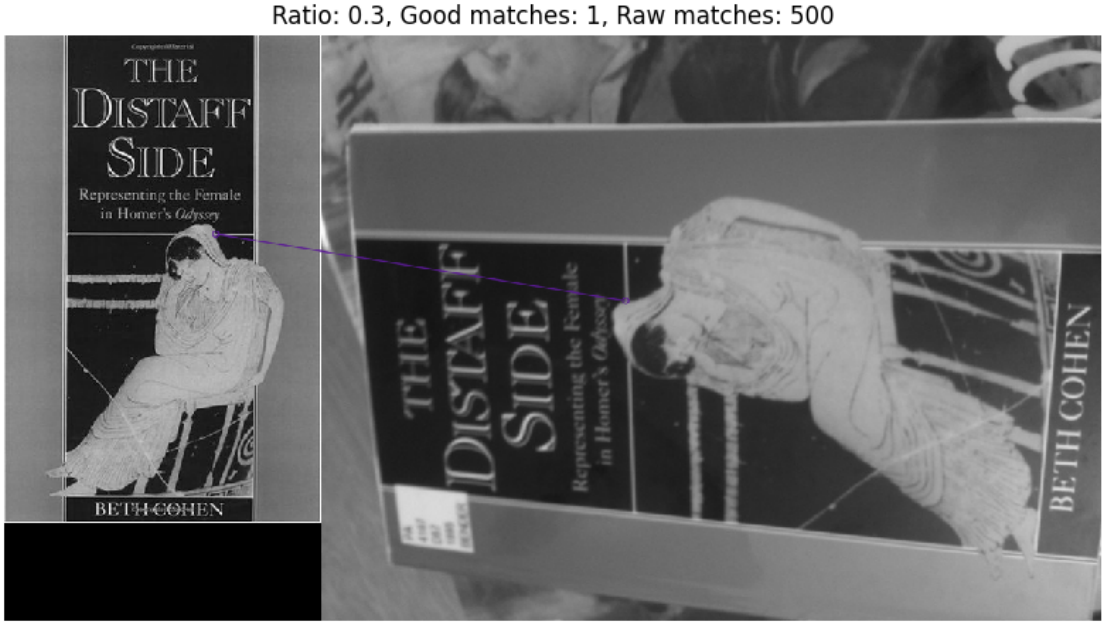
4. [Plot and analyze] You need to visualize the matches by using the `cv2.drawMatches()` function. Also you can change the ratio values, parameters in `cv2.ORB_create()`, and distance functions in `cv2.BFMatcher()`. Please discuss how these changes influence the match numbers.

1.1 Inition image base on the above steps

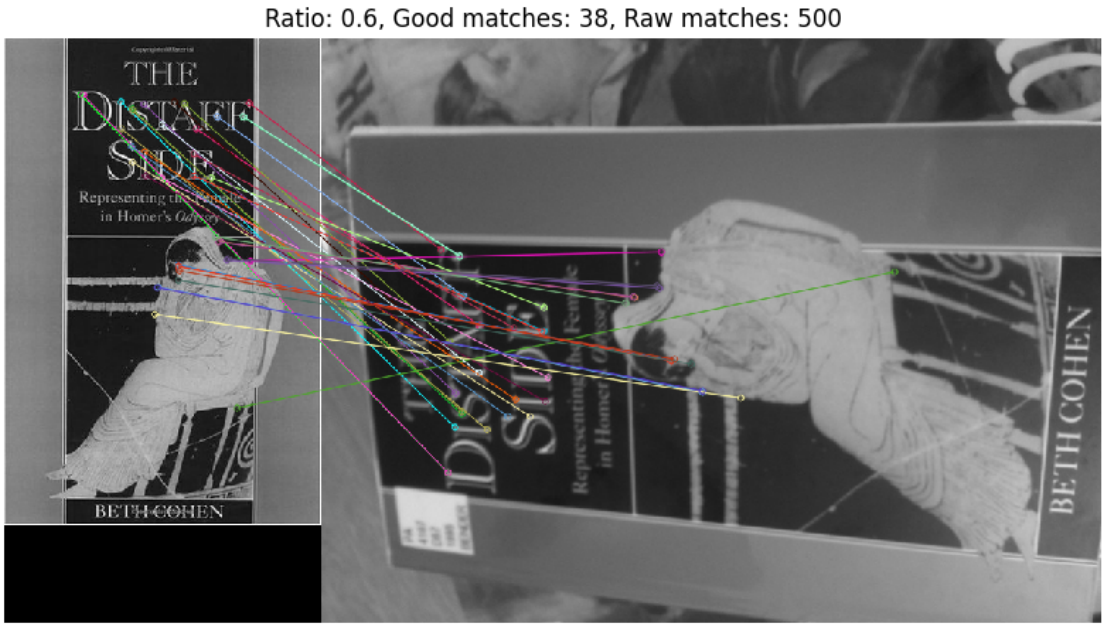


1.2 Variations in the Ratio

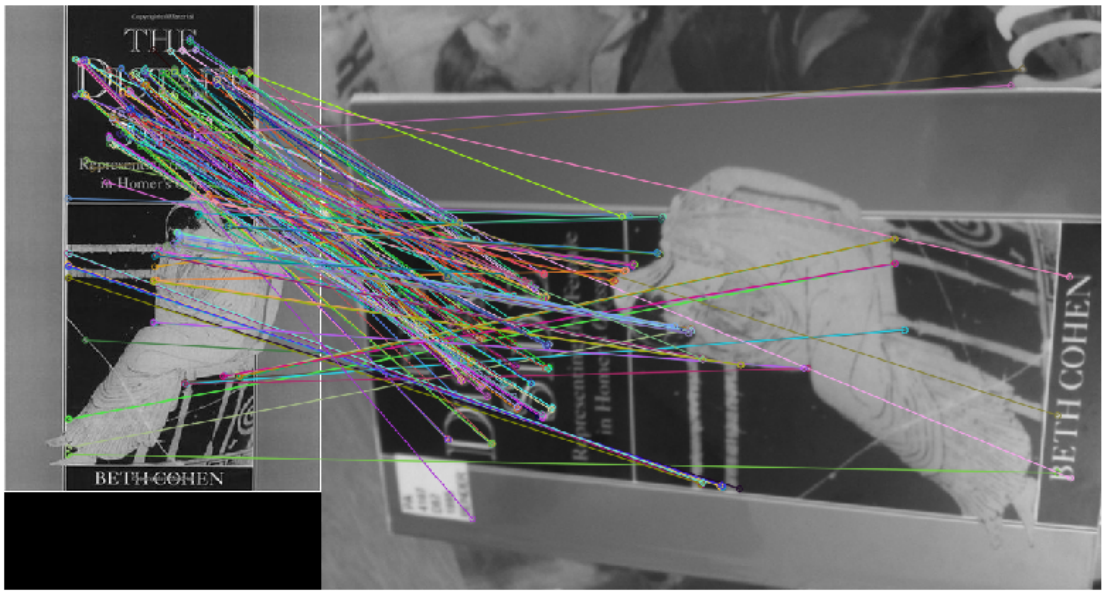
```
RAW Matches (knn):  500
Good matches:  1
Ratio:  0.3
```



```
RAW Matches (knn):  500
Good matches:  38
Ratio:  0.6
```



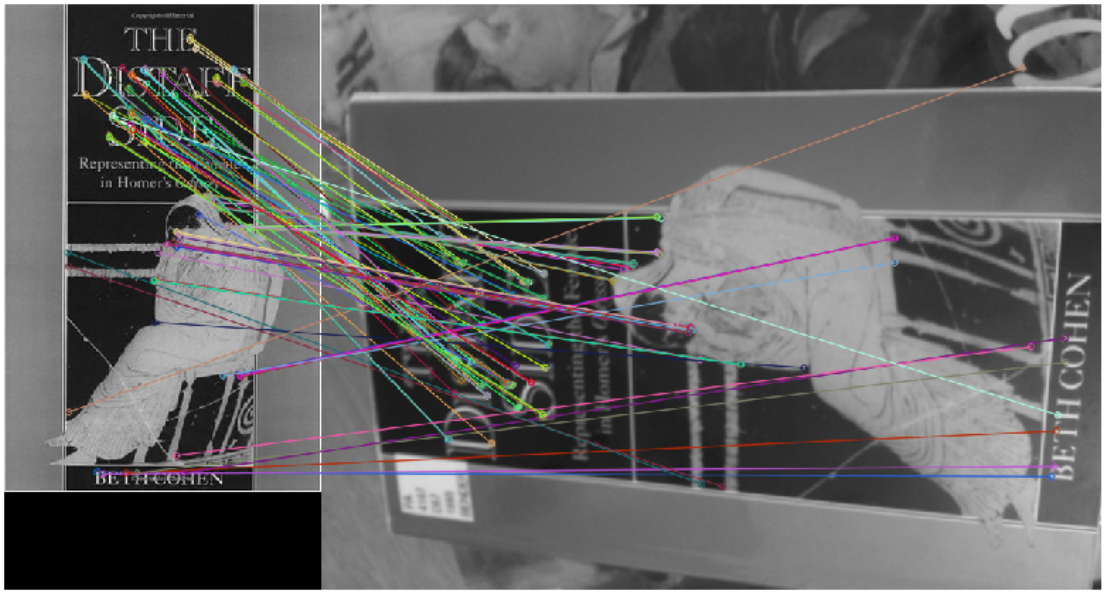

```
RAW Matches (knn):  500
Good matches:  242
Ratio:  0.9
```



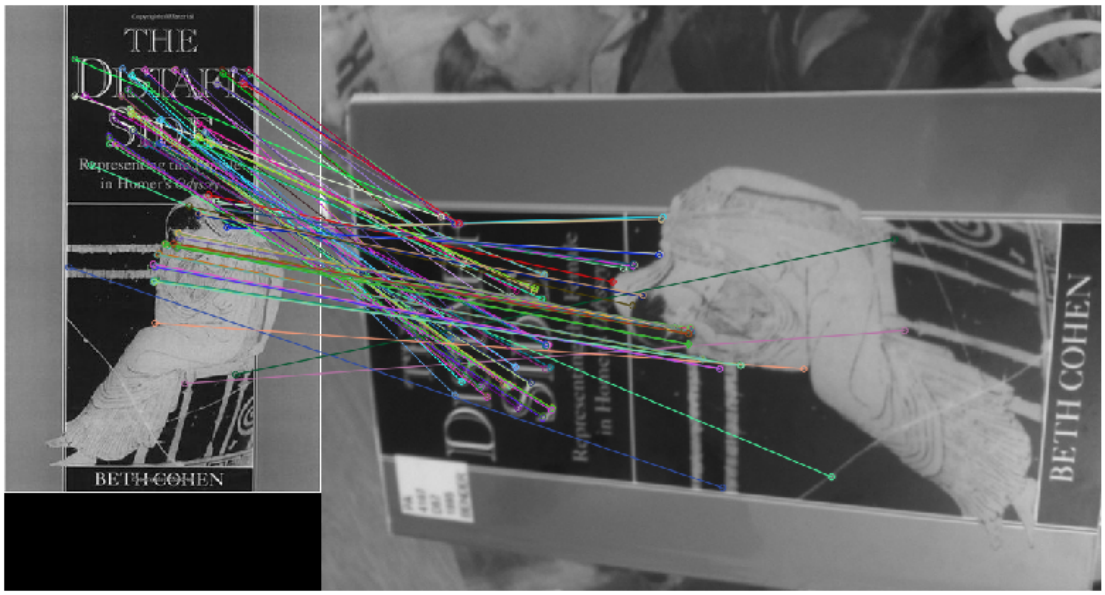
1.3 Changing the input parameters or ORB_create()

Revertin back to the inital steps this time only changing input to `ORB_create()` function

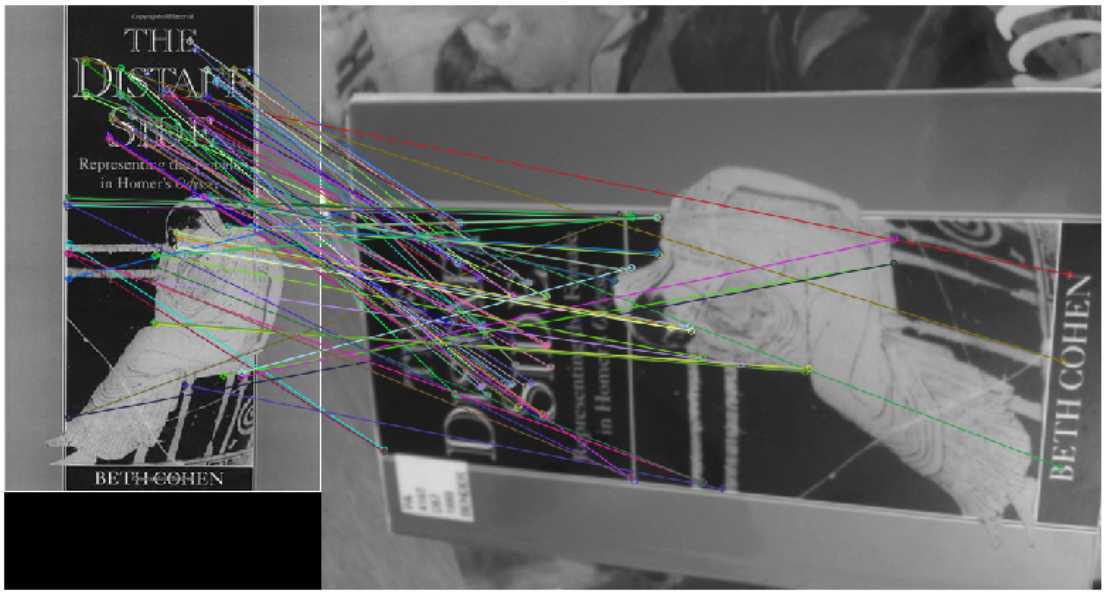
`Good matches edgeThreshold=15:  112`



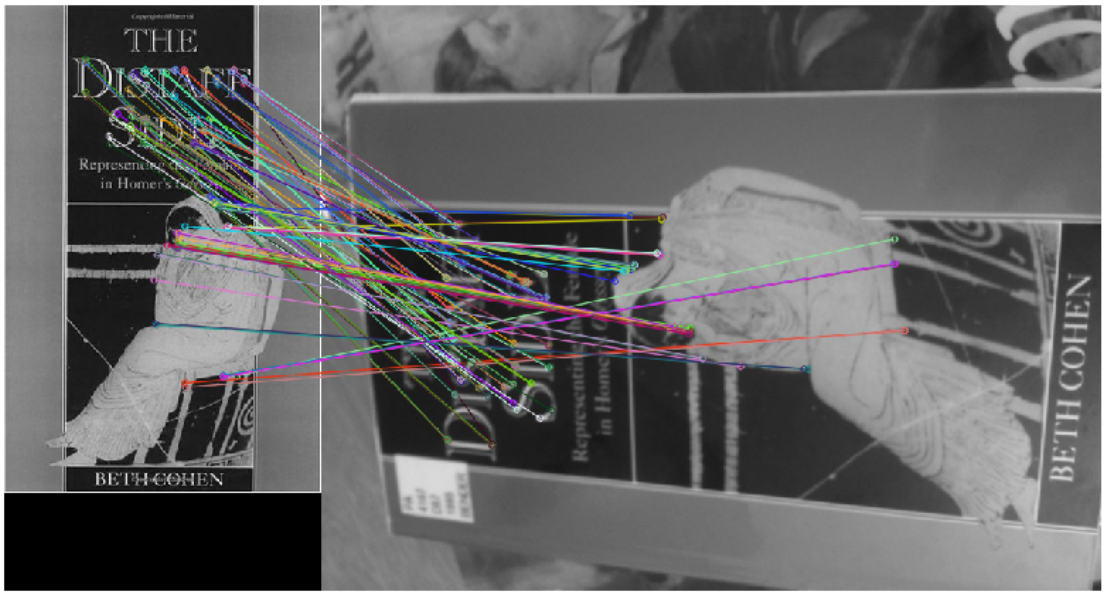
`Good matches edgeThreshold=50:  110`



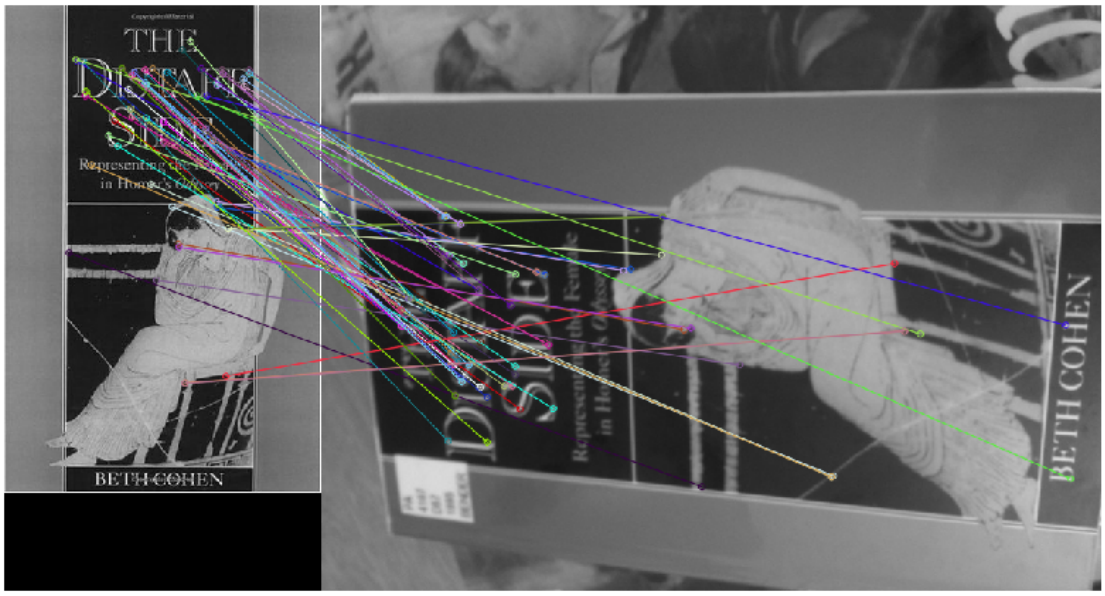
Good matches patchSize=10: 102



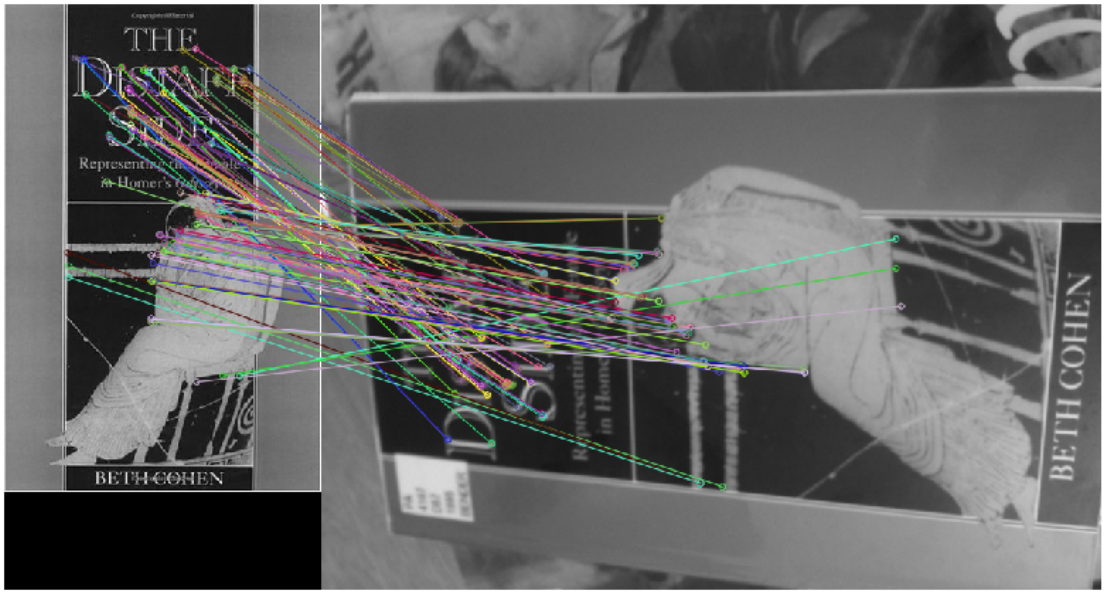
Good matches patchSize=50: 117



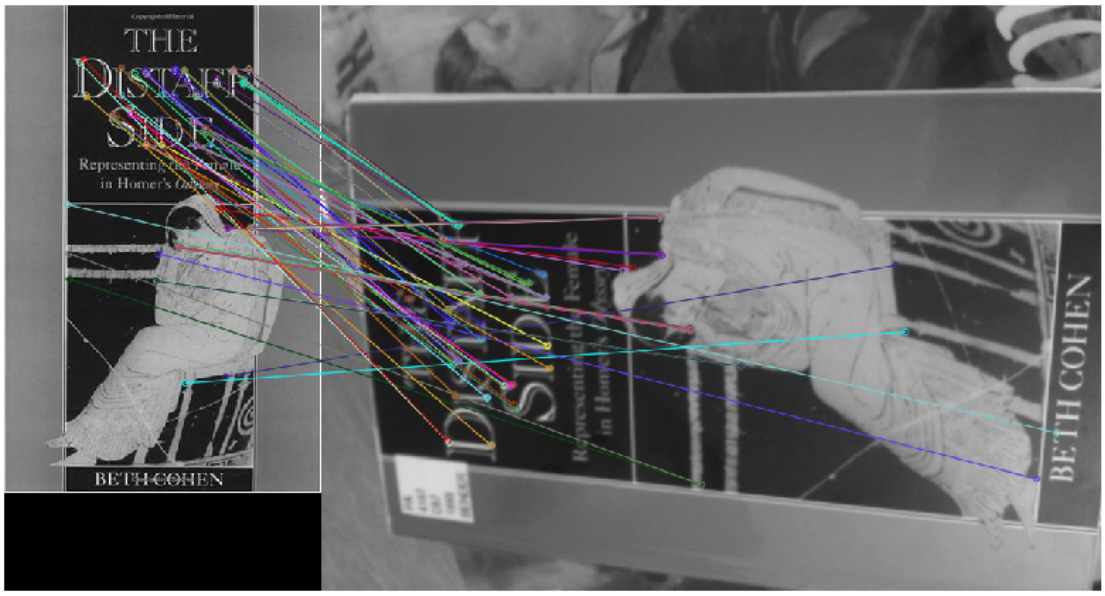
Good matches nlevels=3: 66



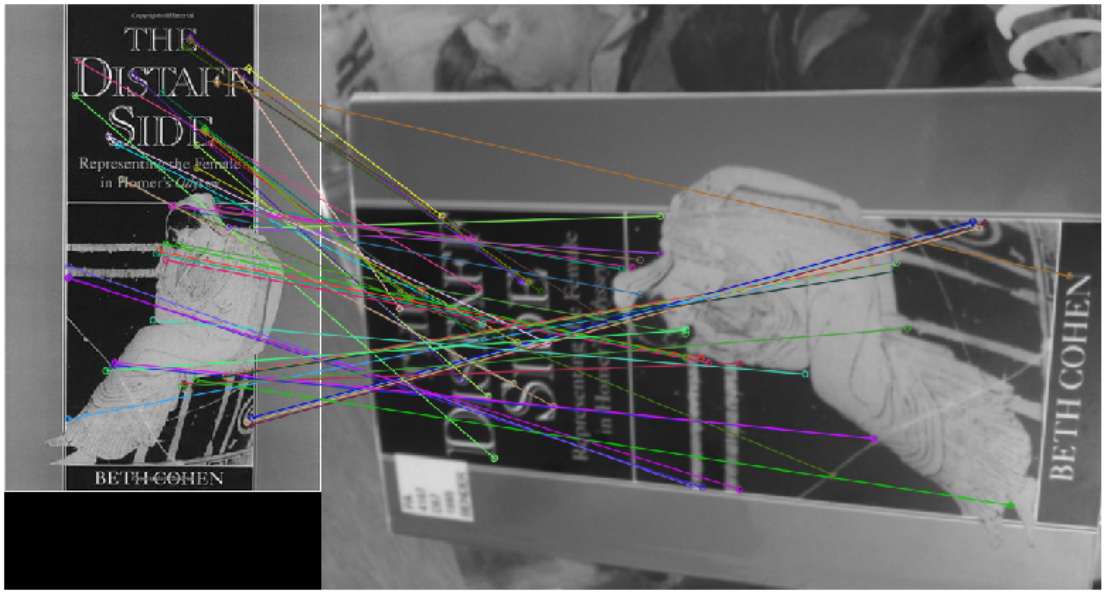
Good matches nlevels=15: 130



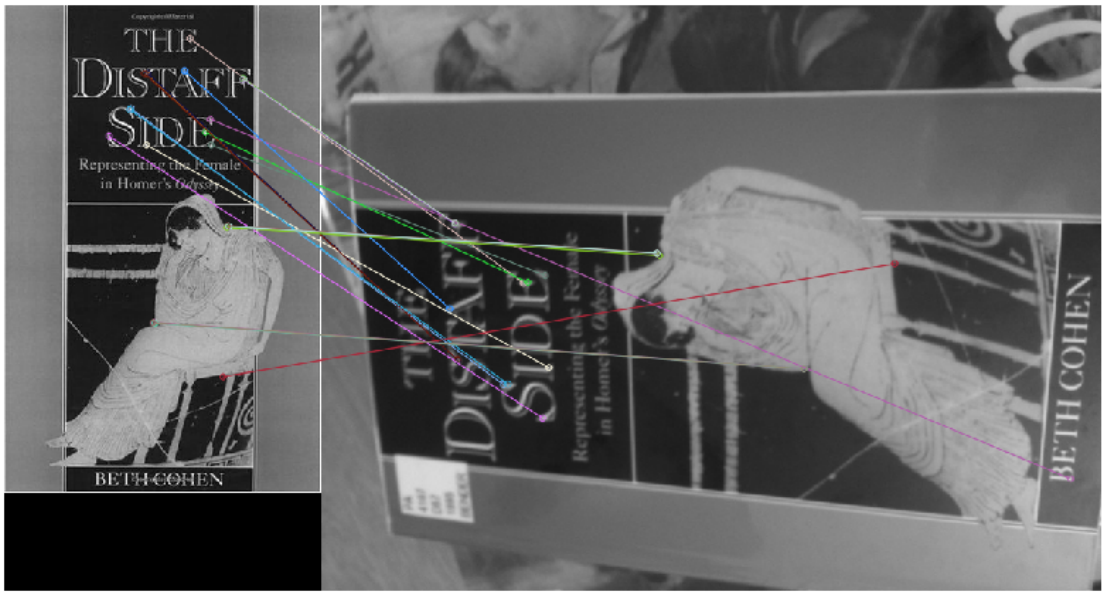
Good matches scaleFactor=1.1: 95



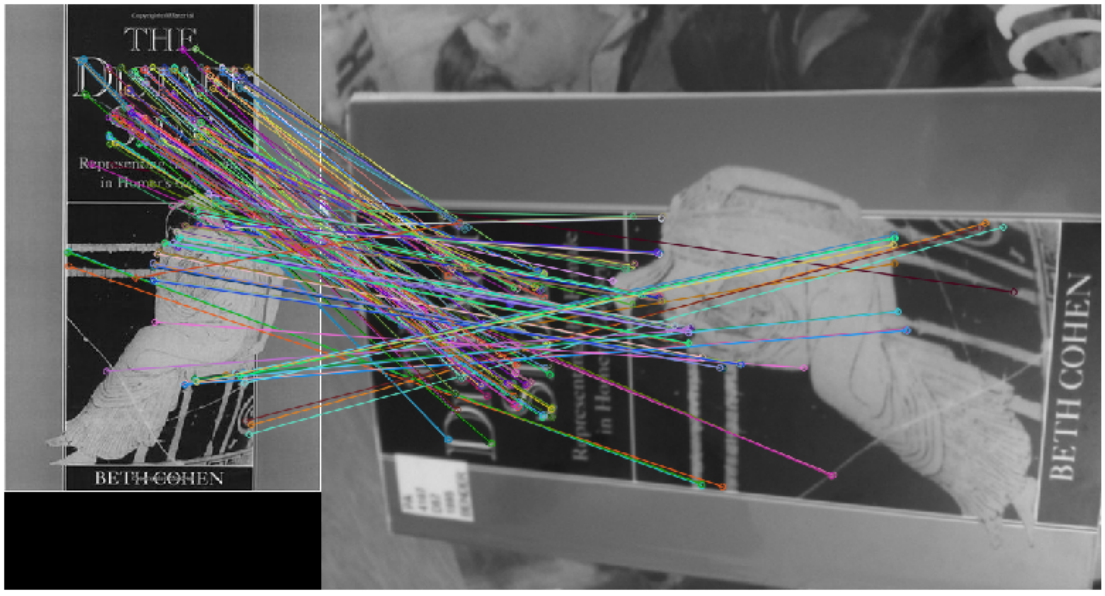
Good matches scaleFactor=1.1: 51



Good matches nfeatures=100: 23




```
Good matches nfeatures=1000:  221
```

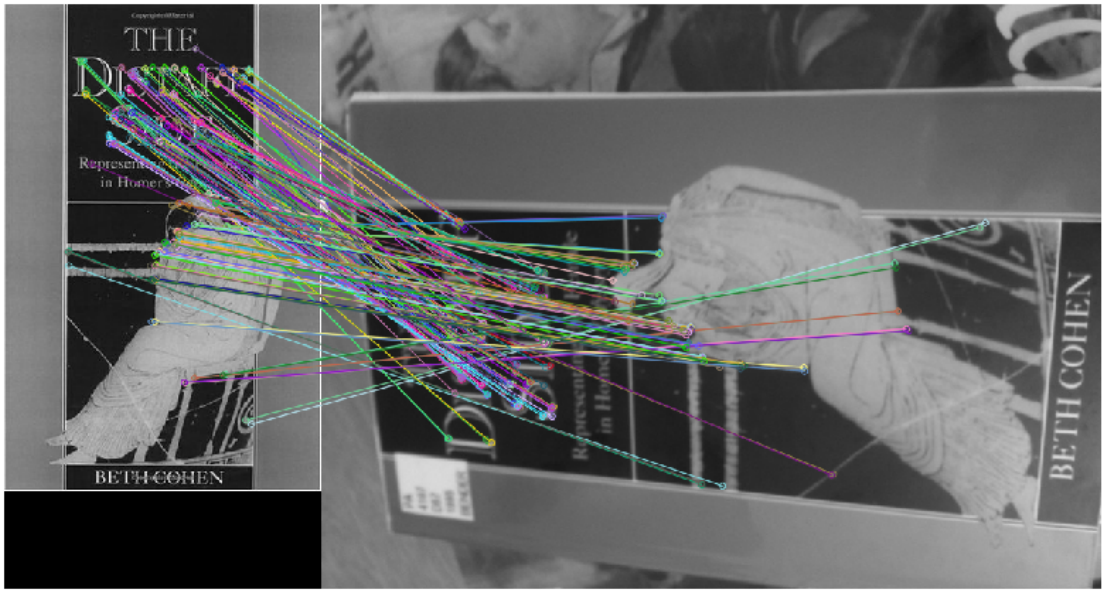


1.4 Picking the best values from the image above

Adding the best good feature inputs in the `ORB_create()` function

```
matches (knn):  769
Good matches:  202
```

```
Ratio:  0.8
nfeatures:  1000
nlevels:  15
edgeThreshold:  40
```



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

1.4.1 Effect of Changing Ratio

1.4.2 Changing the input parameters or ORB_create()

1.4.3 Effect of Edge Threshold

1.4.4 Effect of Patch Size

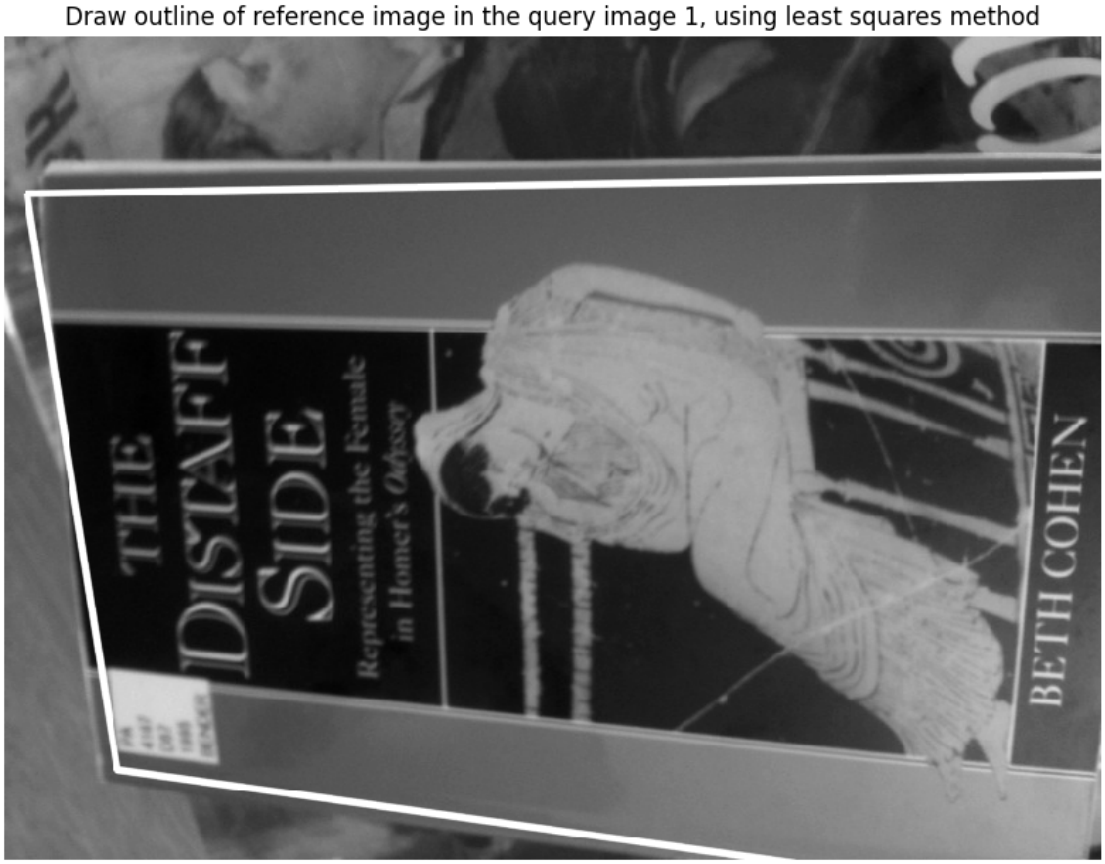
1.4.5 Effect of Scale Factor

1.4.6 Effect of Number of Levels

1.4.7 Effect of Number of Features

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.

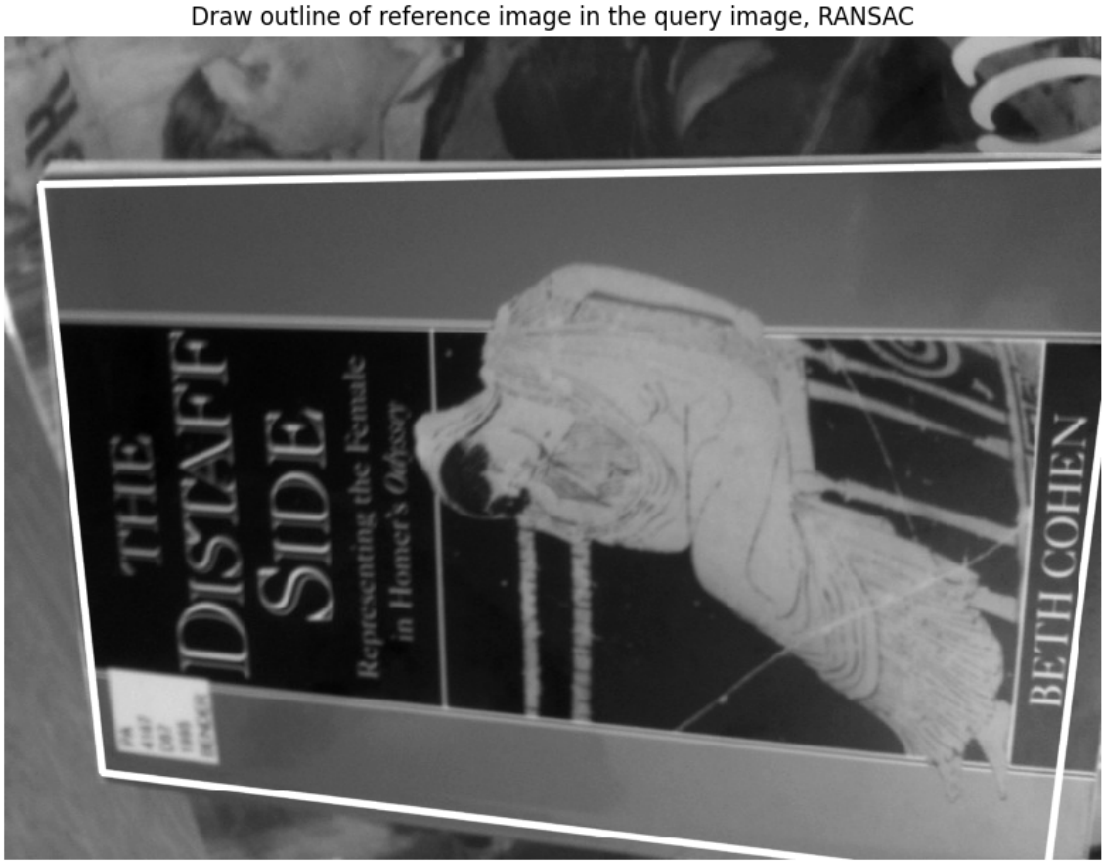
    - We provide a function `draw_outline()` to help with the display, but you may need to edit it for your needs.
    - Try the 'least squre method' option to compute homography, and visualize the inliers by using `cv2.drawMatches()`. Explain your results.
    - Again, you don't need to compare results numerically at this stage. Comment on what you observe visually.



Your explanation of results here

The above image is an example of the homography using the least squares method. As can been seen in the image above. the outlout that is genearted is around around the features that was picket up in the image the feature matching (this can be seen in the previous part). Though, this is not selecting the extire book it is alest deteching a large potion of the book.

Try the RANSAC option to compute homography. Change the RANSAC parameters, and explain your results. Print and analyze the inlier numbers.

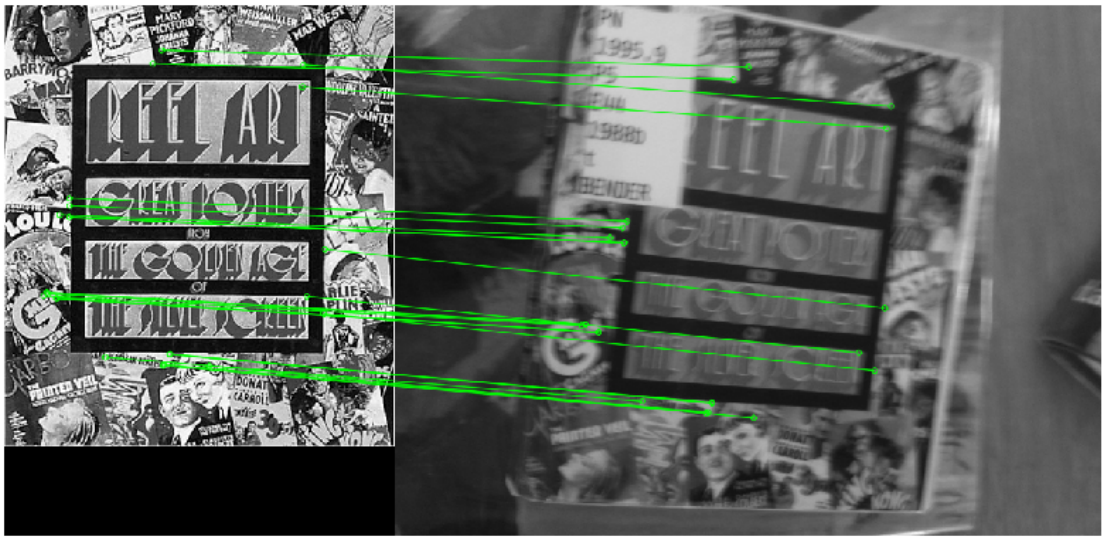



```
Number of inliers:  141
```

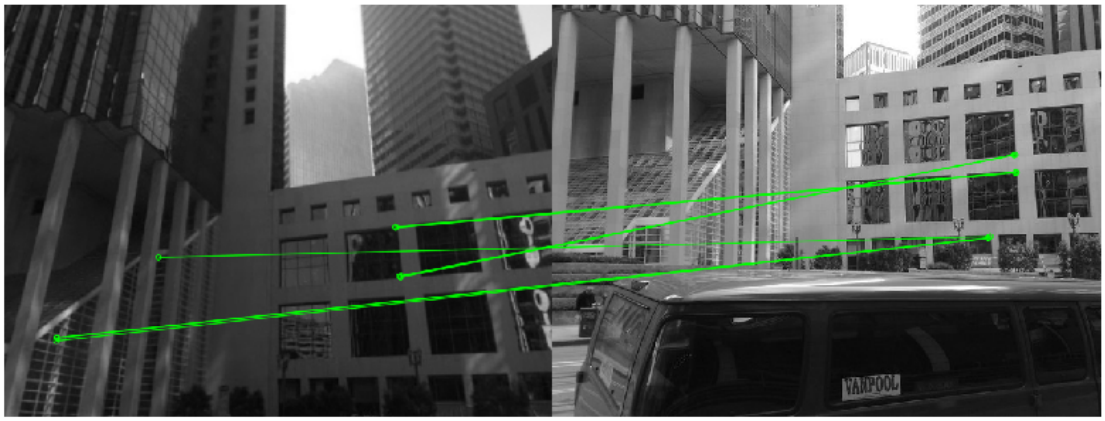
Your explanation of what you have tried, and results here

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.

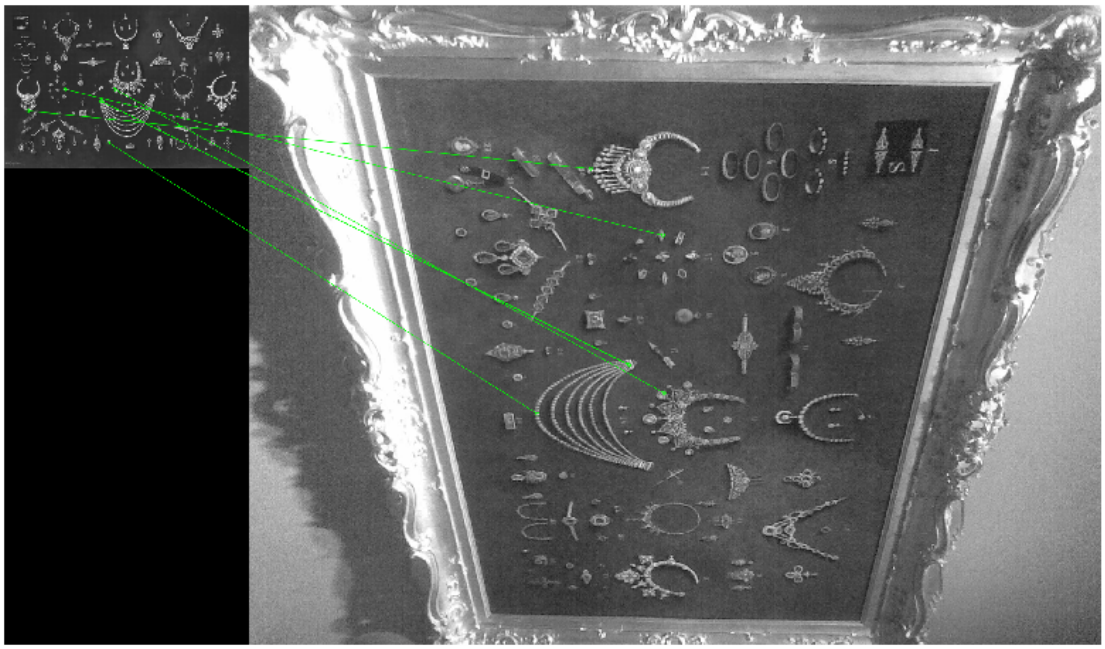
```
Processing image 50
```



Processing Landscape image 2



Processing Museum Painting image 2



Your explanation of results here

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

2.1 Caculating the Matches for a small subset.

```
Query image 1
    reference image 1 with 141 inliers
    reference image 2 with 6 inliers
    reference image 3 with 5 inliers
    reference image 4 with 4 inliers
    reference image 5 with 5 inliers
    reference image 6 with 4 inliers
    reference image 7 with 4 inliers
    reference image 8 with 5 inliers
    reference image 9 with 4 inliers
Best match would be refference image 1
```

2.2 Caculating the Matches for all the books

```
Accuracy: 0.7070707070707071
    70 correct guesses out of 99

Top-3 Accuracy: 0.9090909090909091
    90 correct guesses out of 99
```

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

One way that I would be able to increate the accuracy of the image is being able to calcul

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.

2.3 Adding images that are not in the dataset

```
Accuracy: 0.3763440860215054
Top-3 Accuracy: 0.4838709677419355
```

Your explanation of results and any changes made here

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.

```
Accuracy: 0.2604166666666667
Top-3 Accuracy: 0.2604166666666667
```

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