

# Assignment2

April 9, 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.

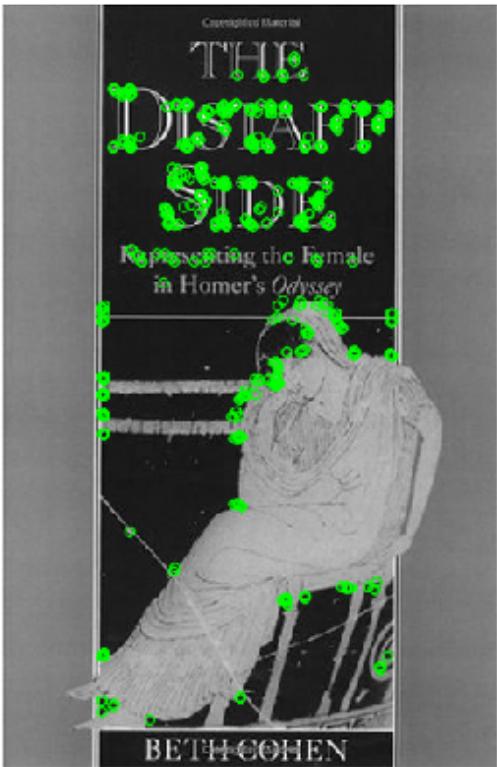
***When converting to PDF, include the outputs and analysis only, not your code.*** You can do this from the command line using the nbconvert command (installed as part of Jupyter) as follows:

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

## 2 Matching an object in a pair of images (45%)

### 2.0.1 draw key points and matches by using orb method

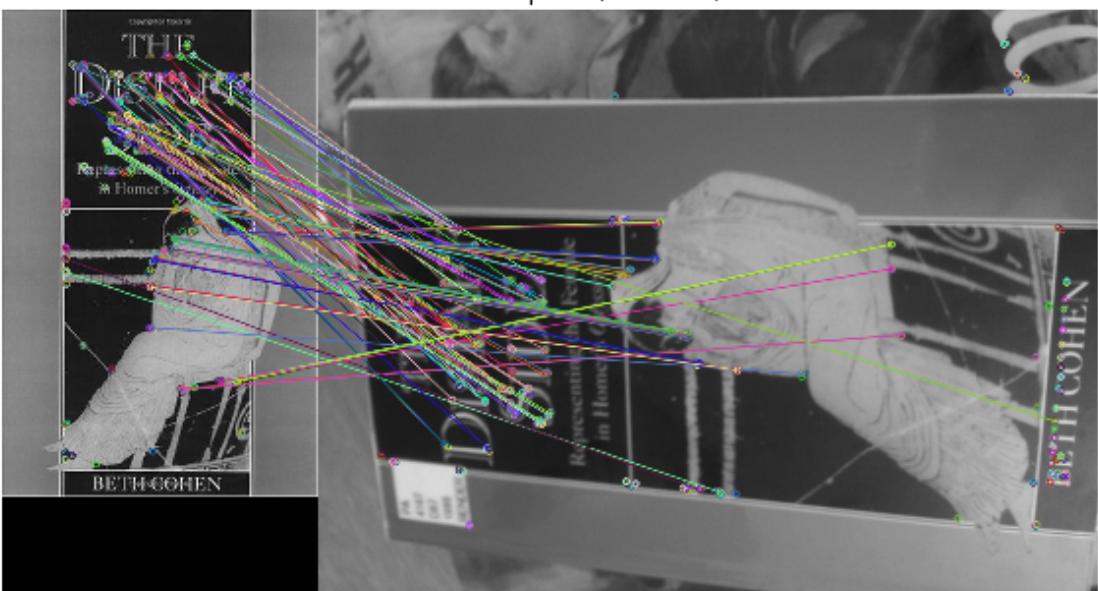
keypoints of reference image (orb)



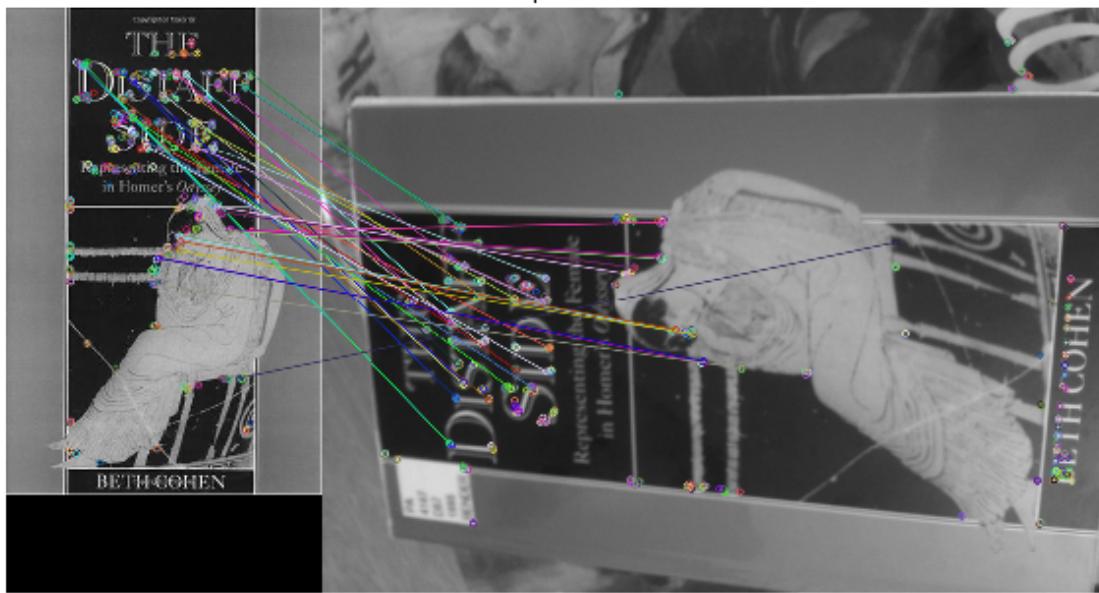
keypoints of query image (orb)



Match descriptors (ratio: 0.8)



Match descriptors (ratio: 0.6)



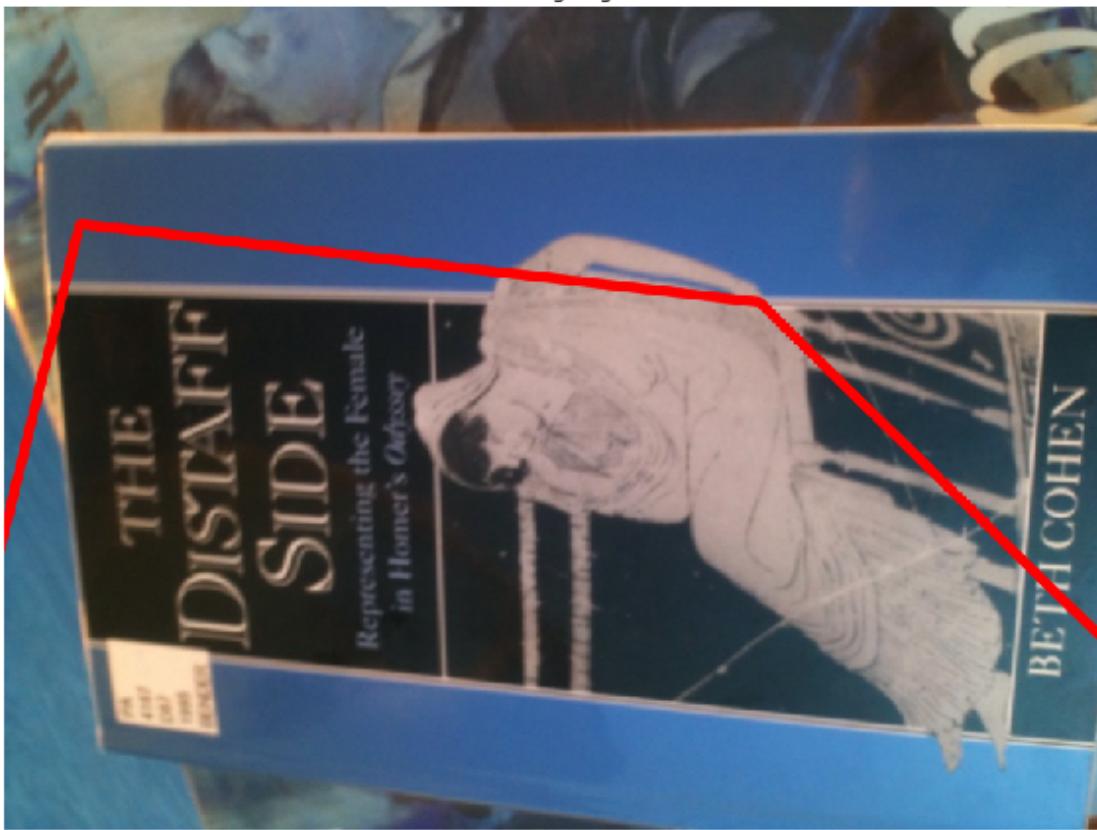
### Change the ratio

By compare the ration 0.6 and the rtatio 0.8, we can get:

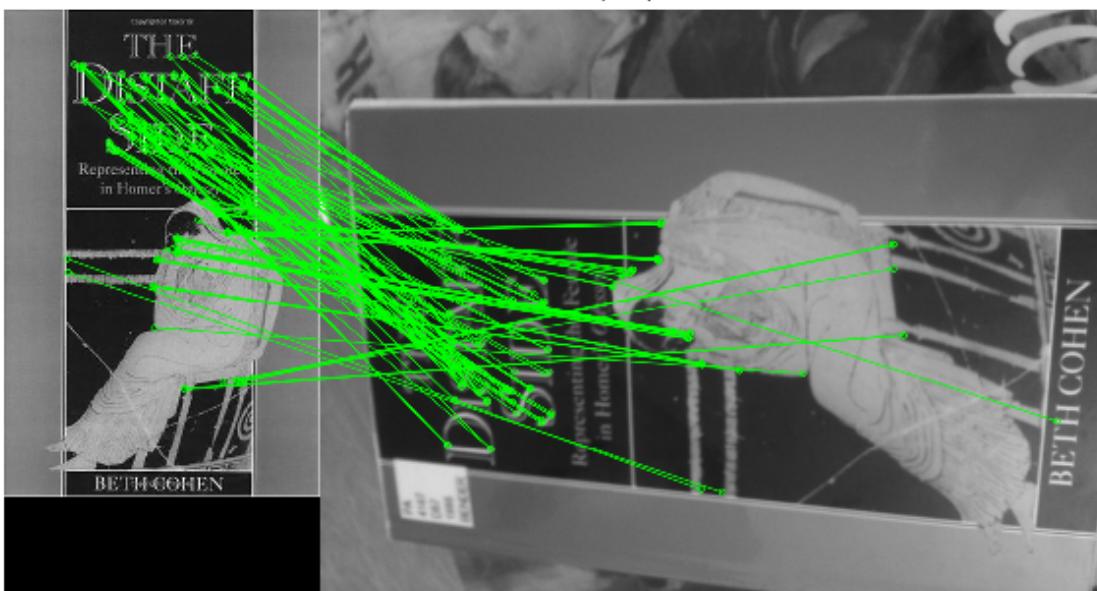
- when the ratio become small, the keypoints and match numbers become less.
  - when the ratio become large, the keypoints and match numbers become more.
-

## 2.0.2 draw outline and inliers by using regular method

draw outline (using regular method)



draw inliers (orb)



number of detected inliers: 131  
number of detected outliers: 0.0

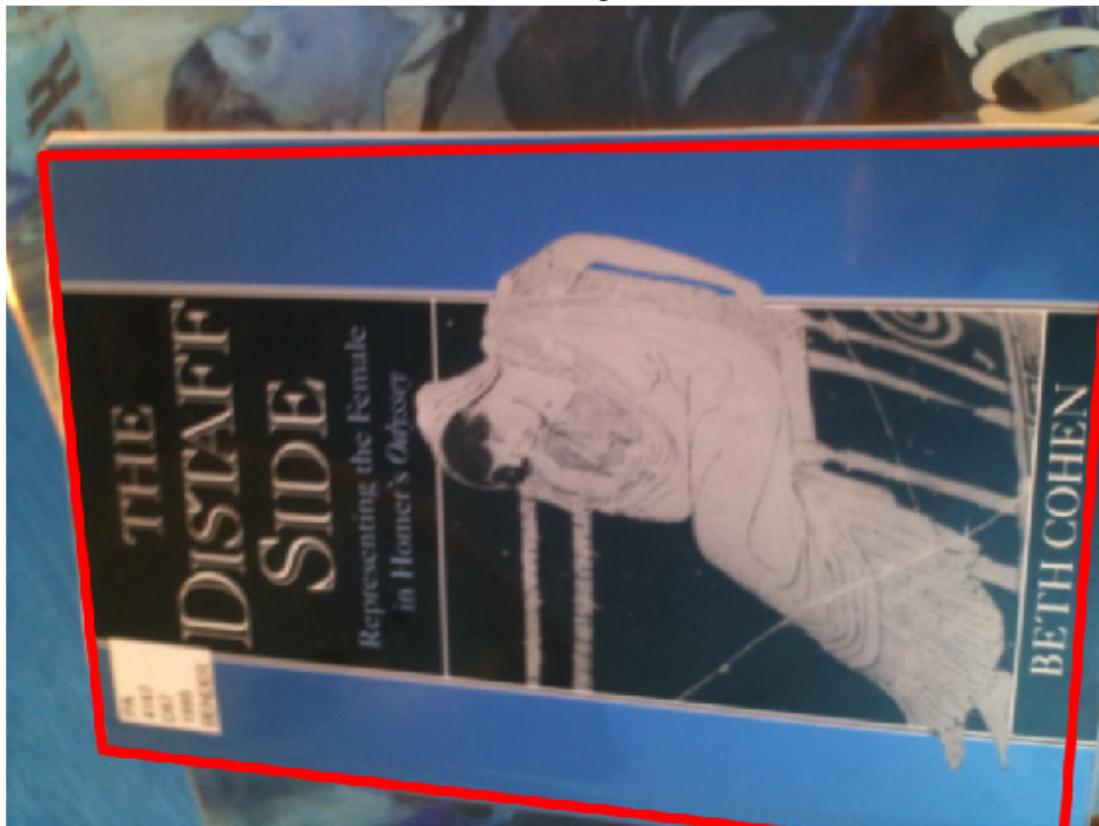
### *Summary*

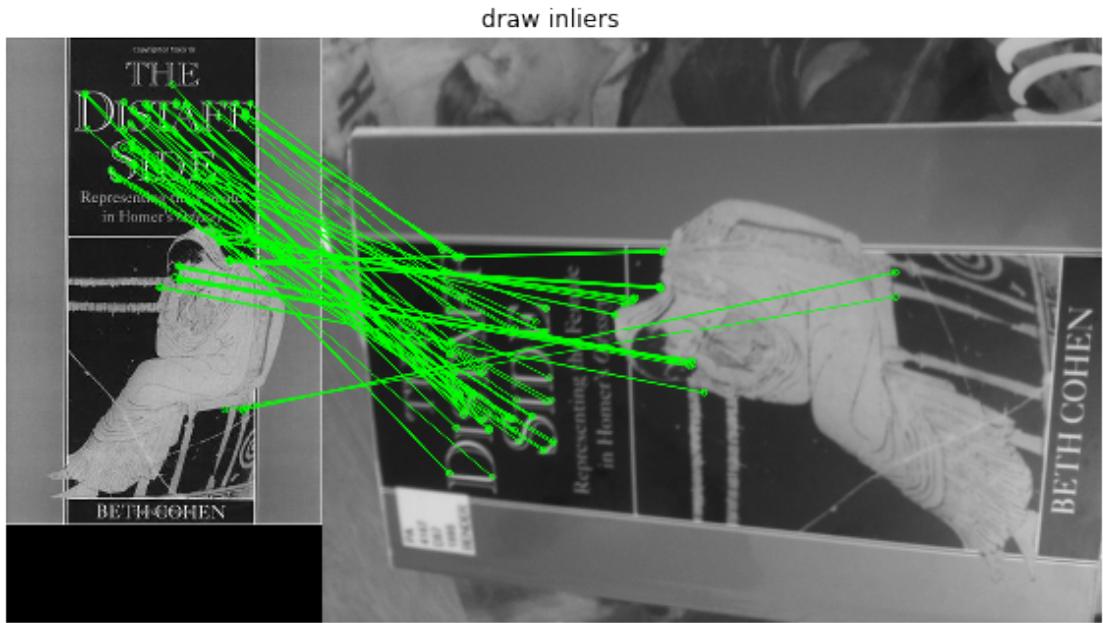
When I try to use regular method, I notice draw oulines does not work correctly.

---

#### 2.0.3 Draw outline and inliers by using RANSAC

draw outline (using RANSAC)





```
number of detected inliers: 100
number of detected outliers: 31.0
```

### *Summary*

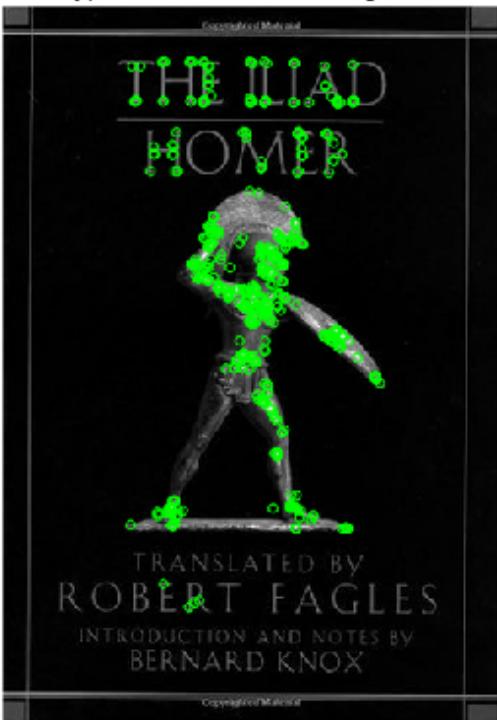
Compare to the RANSAC and the regular method, I notice the number of detected inliers is different. The ***total number of detected points are 131***, and when I use ***RANSAC*** method, the number of detected inliers are ***only 100***, however, when I use ***regular method***, the number of detected inliers are ***131***. There is ***a difference of 31 points***. However, it may influence when draw outlines because these 31 points should not be calculated in the inliers.

## 2.1 Book cover (example book\_cover\_025)

### 2.1.1 draw keypoints of reference image and query image (using orb, brisk and sift)

```
[ WARN:0@2.599] global /Users/runner/work/opencv-python/opencv-
python/opencv_contrib/modules/xfeatures2d/misc/python/shadow_sift.hpp (15)
SIFT_create DEPRECATED: cv.xfeatures2d.SIFT_create() is deprecated due SIFT
tranfer to the main repository. https://github.com/opencv/opencv/issues/16736
```

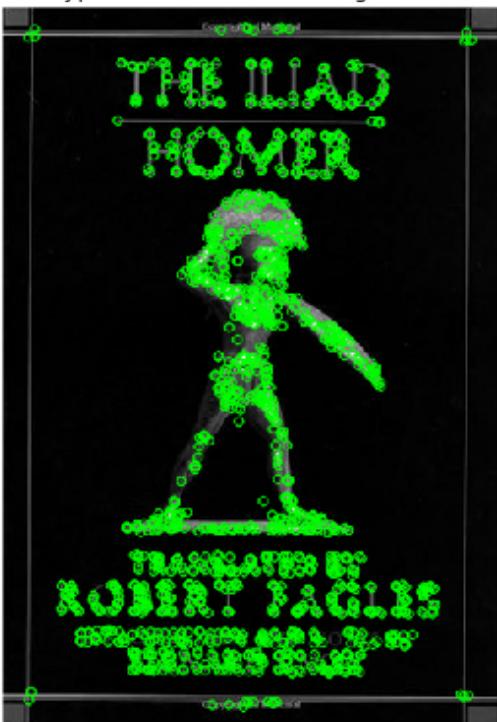
keypoints of reference image (orb)



keypoints of query image (orb)



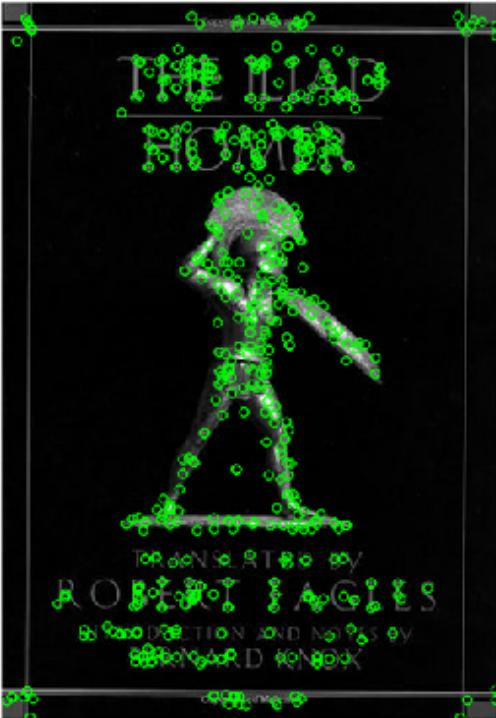
keypoints of reference image (brisk)



keypoints of query image (brisk)



keypoints of reference image (sift)



keypoints of query image (sift)



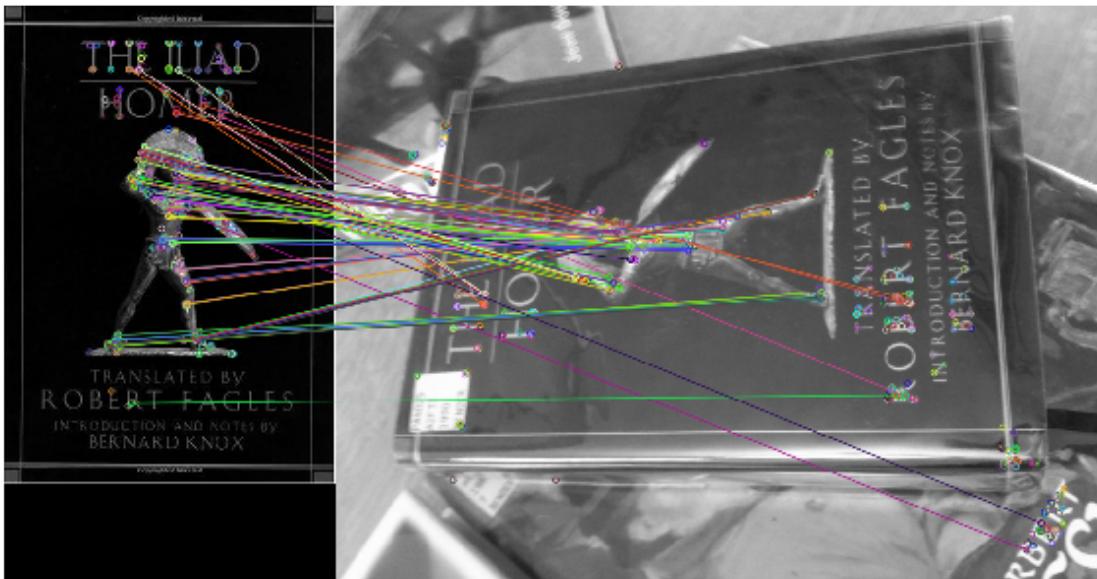
### 2.1.2 summary of finding key ponits part

In *getting keypoints*, the shift works better than orb and the brisk, it returns more key points in the query image and the reference image.

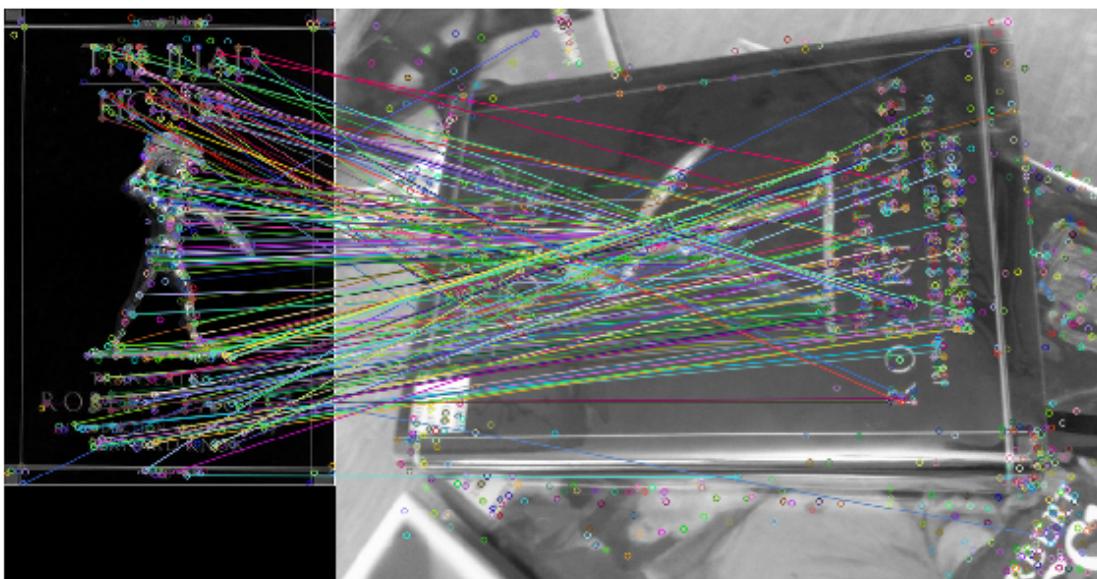
---

### 2.1.3 Draw match part (using orb and sift)

matches (orb)



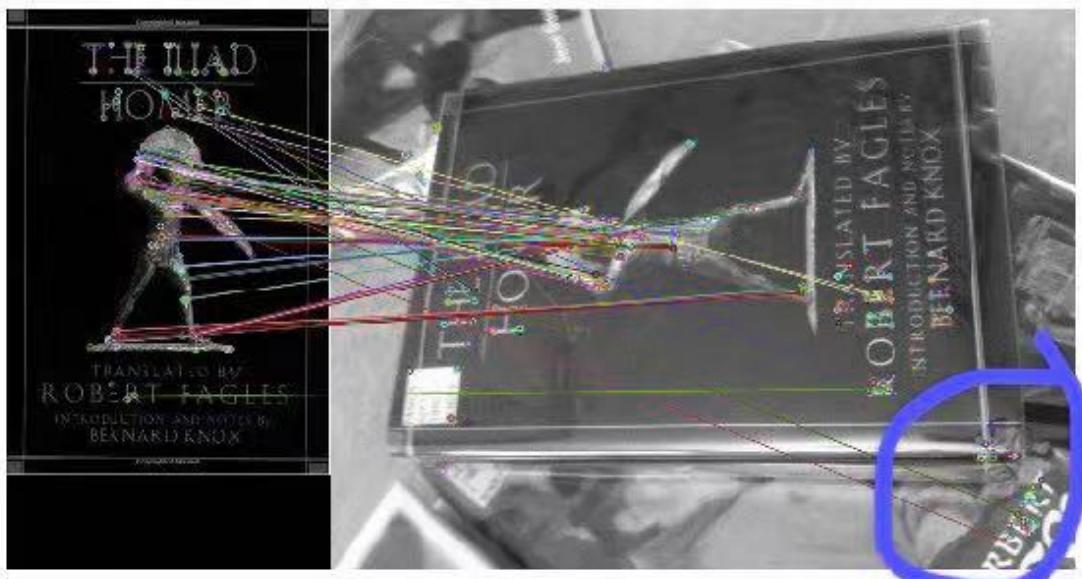
matches (sift)



### 2.1.4 Summary of drawing match part

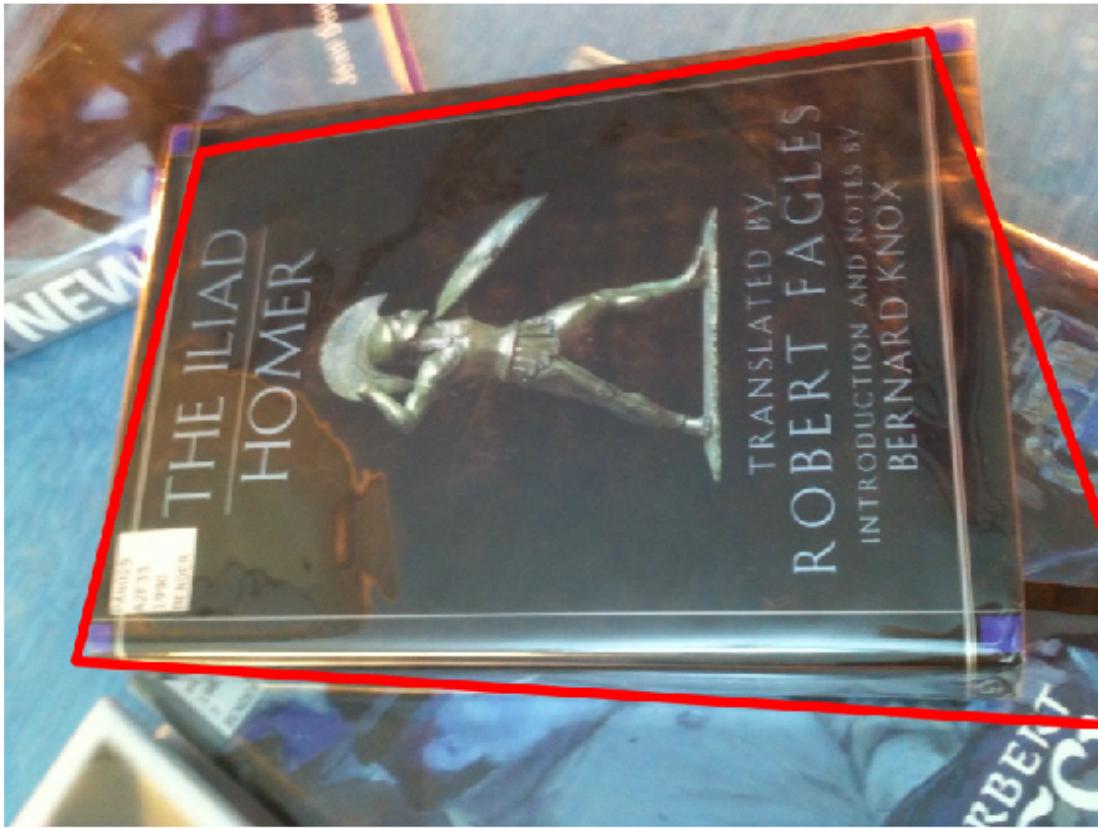
In this part, I notice some matches in ***orb method*** matches the keypoints ***outside the reference book***, which are ***wrong matches***. (shown in the picture: wrong matches)

picture: wrong matches

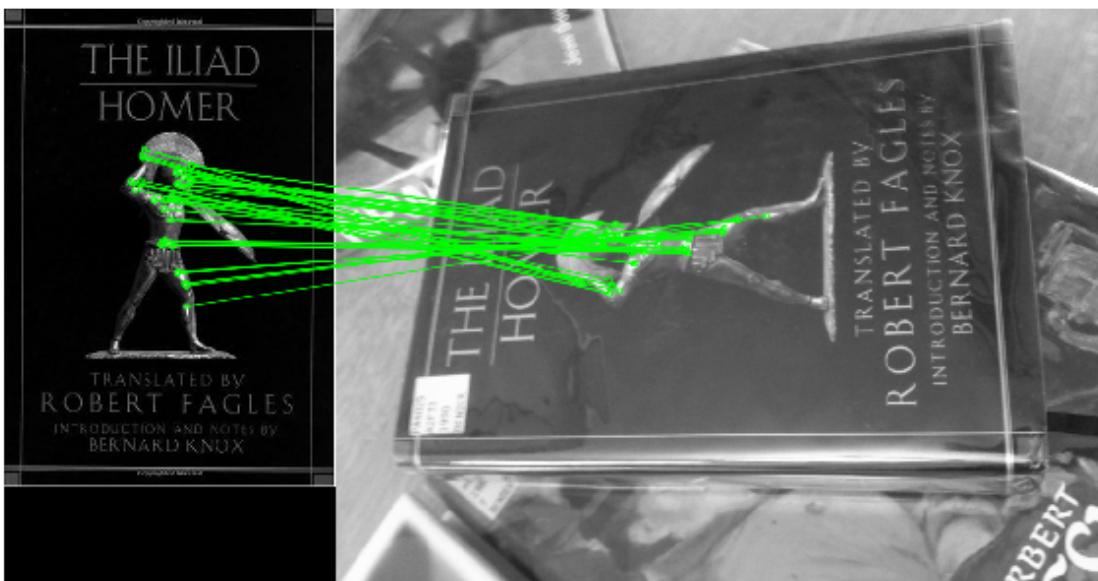


### 2.1.5 Draw outlines and inliers (using orb)

draw outline (orb)



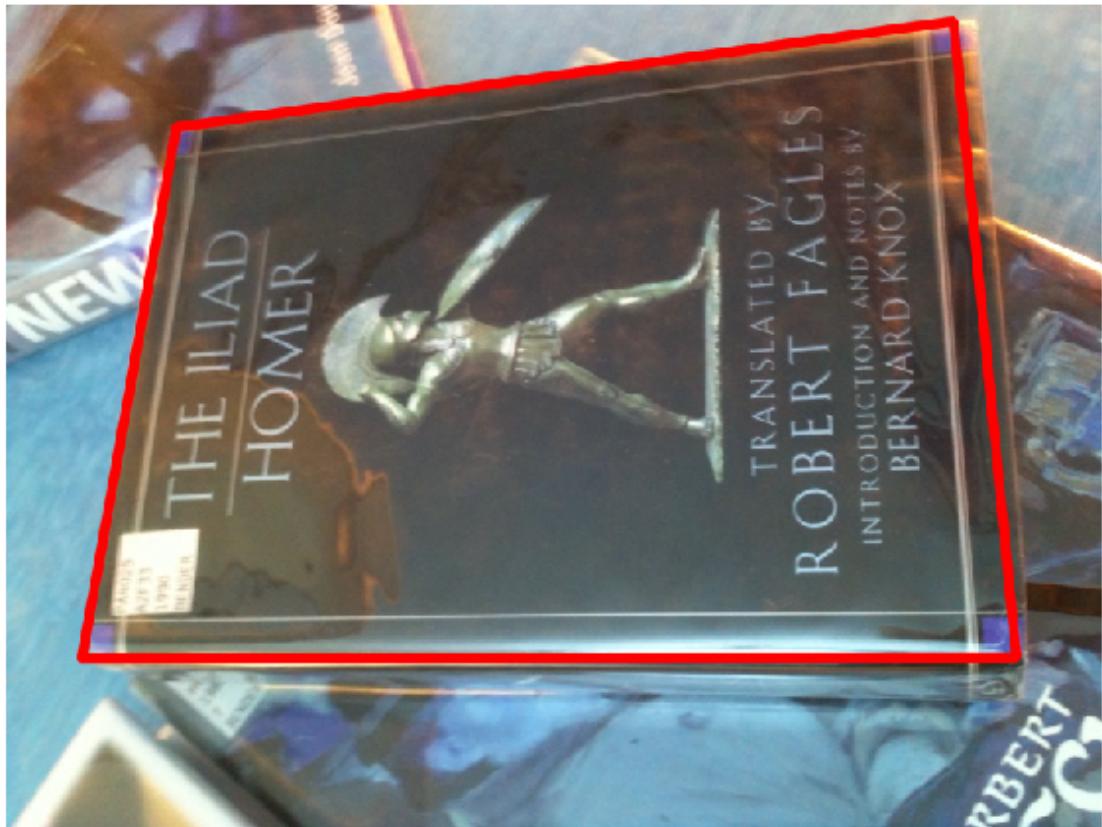
draw inliers (orb)

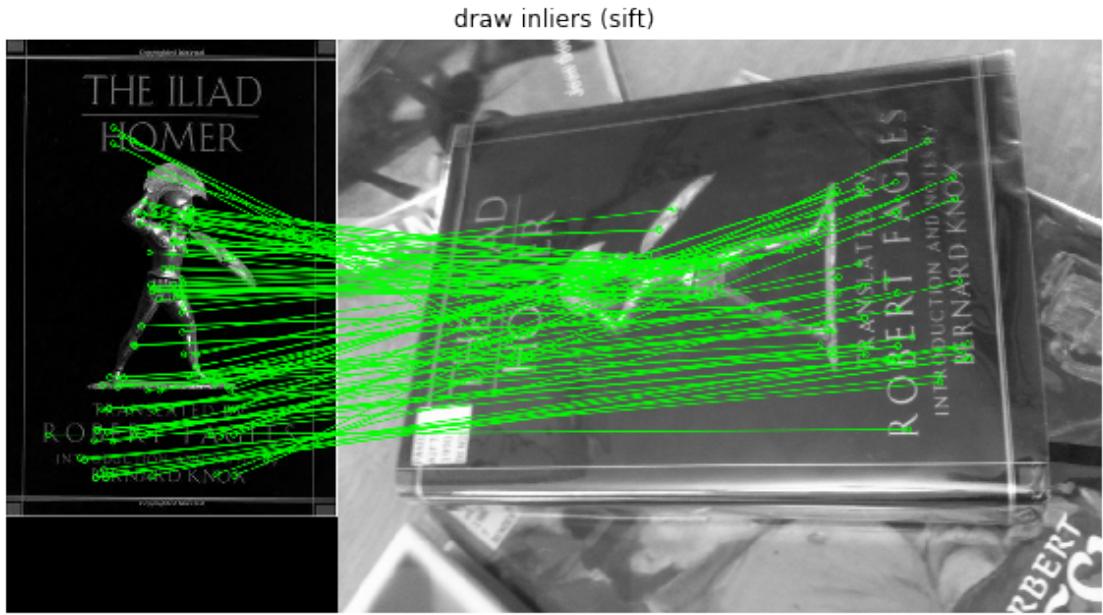


number of detected inliers: 57  
number of detected outliers: 36.0  
Ratio: 0.6129032258064516

### 2.1.6 Draw outlines and inliers (using sift)

draw outline (sift)





```

number of detected inliers: 106
number of detected outliers: 70.0
Ratio: 0.6022727272727273

```

### 2.1.7 Summary

In image 25 from book\_covers. we can see when we use orb method to get the outline, it is not very precise, however, sift method works good.

**Reason (why orb method will fail):** when I use orb method, the keypoints of query image do not detect many keypoints in this area (draw in the picture\_01), the key points which are detected are focus on the person (middle in the book cover). So when do matches, some keyponits will match the keypoints which detect ouside the book cover, this is why we can not get the correct area when we draw outlines.

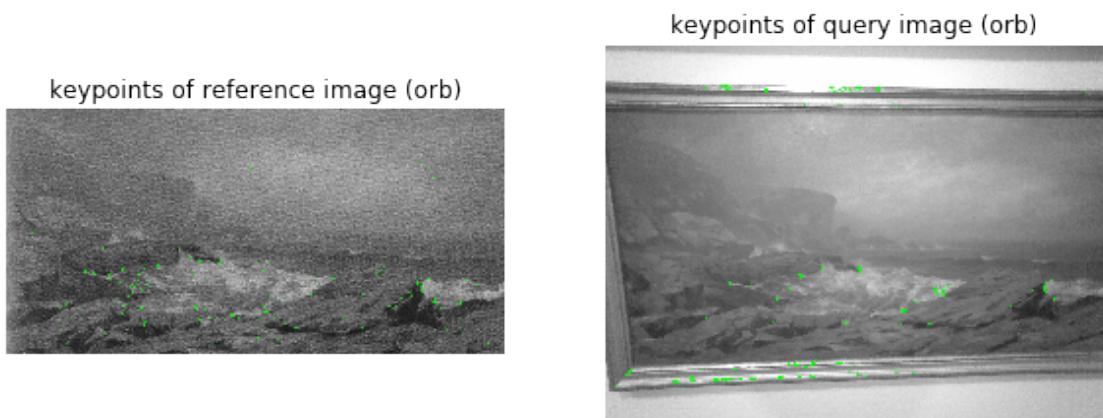
picture\_01

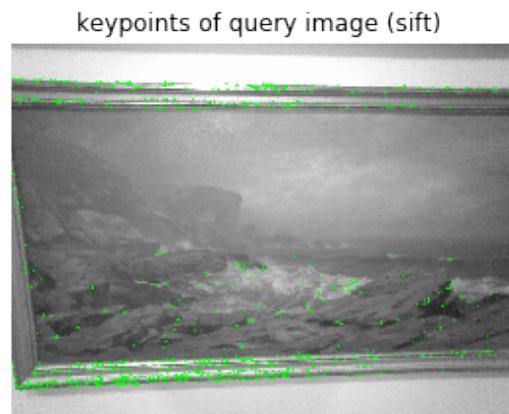
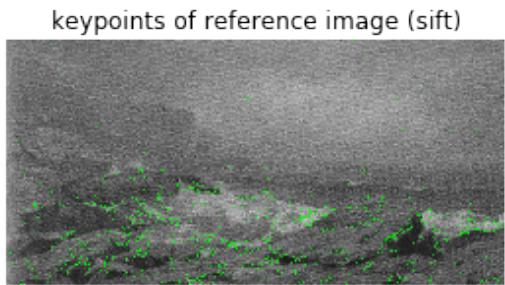
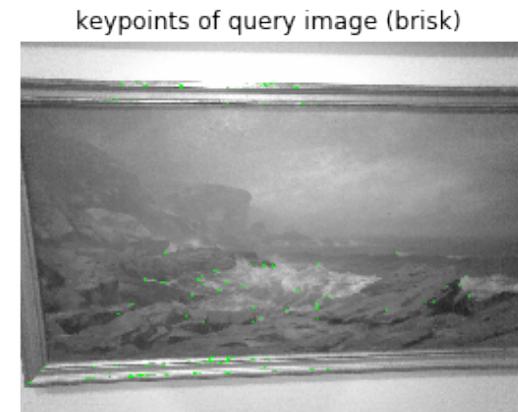
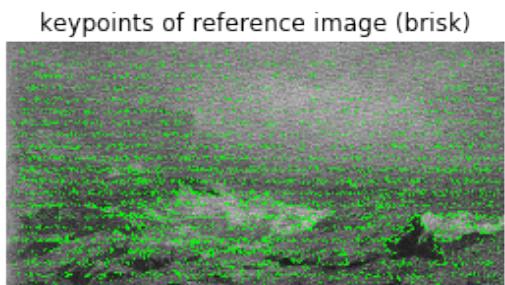


---

## 2.2 museum paintings (example museum\_paintings\_064)

### 2.2.1 draw keypoints of reference image and query image (using orb, brisk and sift)





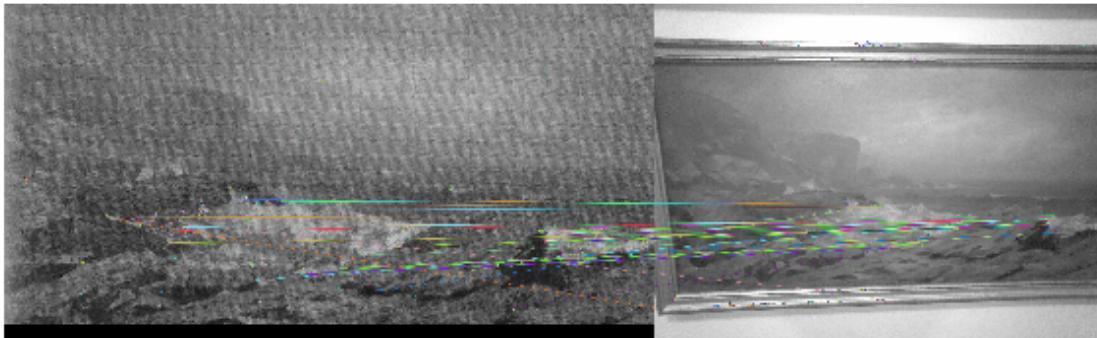
### 2.2.2 summary of finding key ponits part

In getting keypoints, the shift works better than orb and the brisk, it returns more key points in the query image and the reference image.

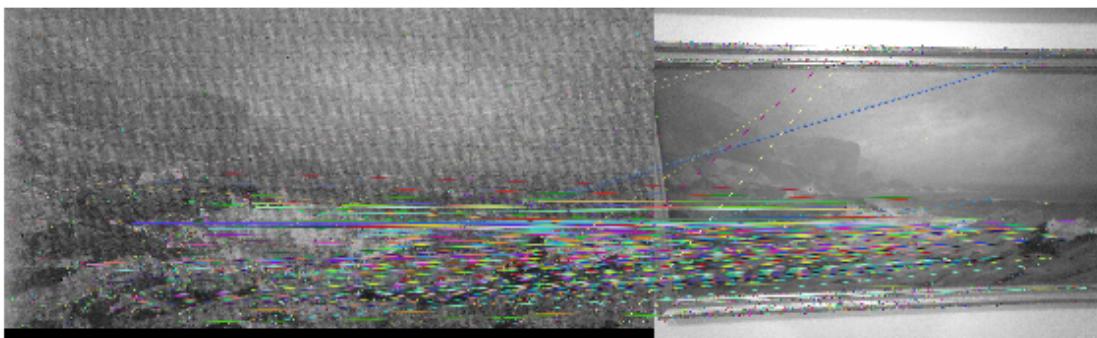
---

### 2.2.3 Draw match part (using orb and sift)

matches (orb)



matches (sift)



### 2.2.4 Summary of drawing match part

In this part, two pictures are draw all mactches correctly.

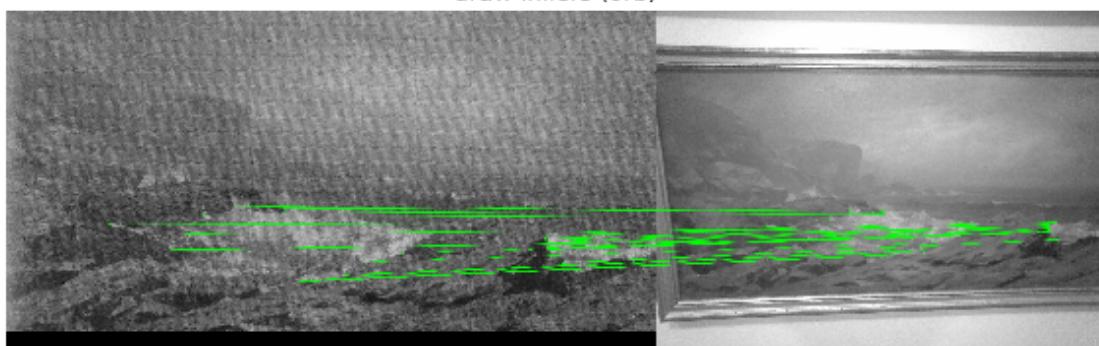
---

## 2.2.5 Draw outlines and inliers (using orb)

draw outline (orb)



draw inliers (orb)



number of detected inliers: 52

number of detected outliers: 9.0

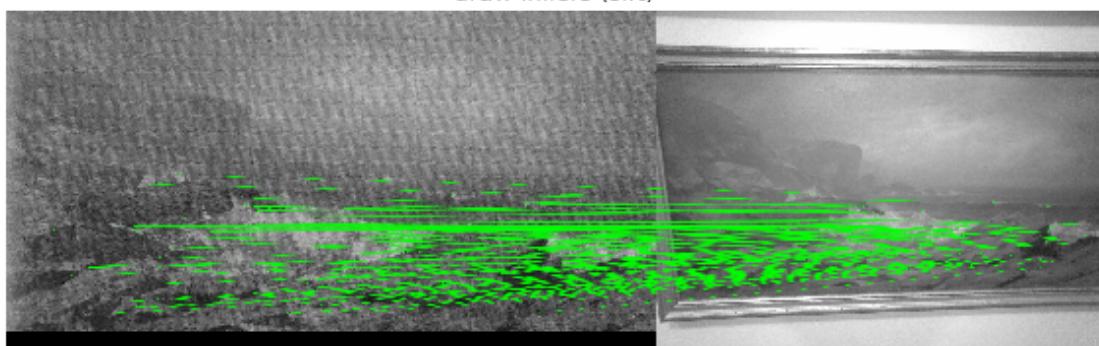
Ratio: 0.8524590163934426

## 2.2.6 Draw outlines and inliers (using sift)

draw outline (sift)



draw inliers (sift)



number of detected inliers: 142

number of detected outliers: 48.0

Ratio: 0.7473684210526316

## 2.2.7 Summary

**Ratio = number of inlier/number of matches**

I compare the ratio of two method (orb and sift), when the ratio become smaller (should bigger than 0), the ouline will become more accurrate. By compare **picture draw outline (orb)** and **picture draw outline (sift)**, we can see the **picture draw outline (sift)** works better. By the way, the ratio is 0.747 (sift method) which is smaller than 0.852 (orb method).

---

## 3 What am I looking at? (40%)

### 3.0.1 At first, I try use orb method to get matches

*this case is one of a suucessful cases (book\_covers 12) :*

We get the number 18 image is the best, it has 56 points matches.

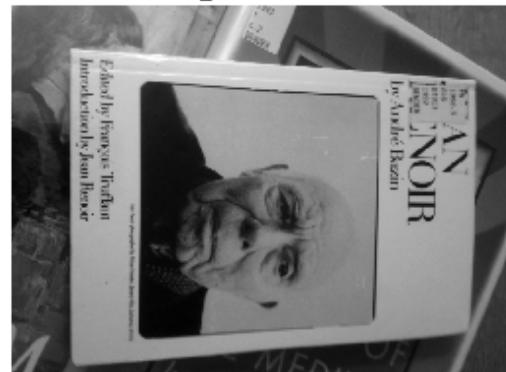
book\_reference (success)

JEAN  
RENOIR  
by André Bazin



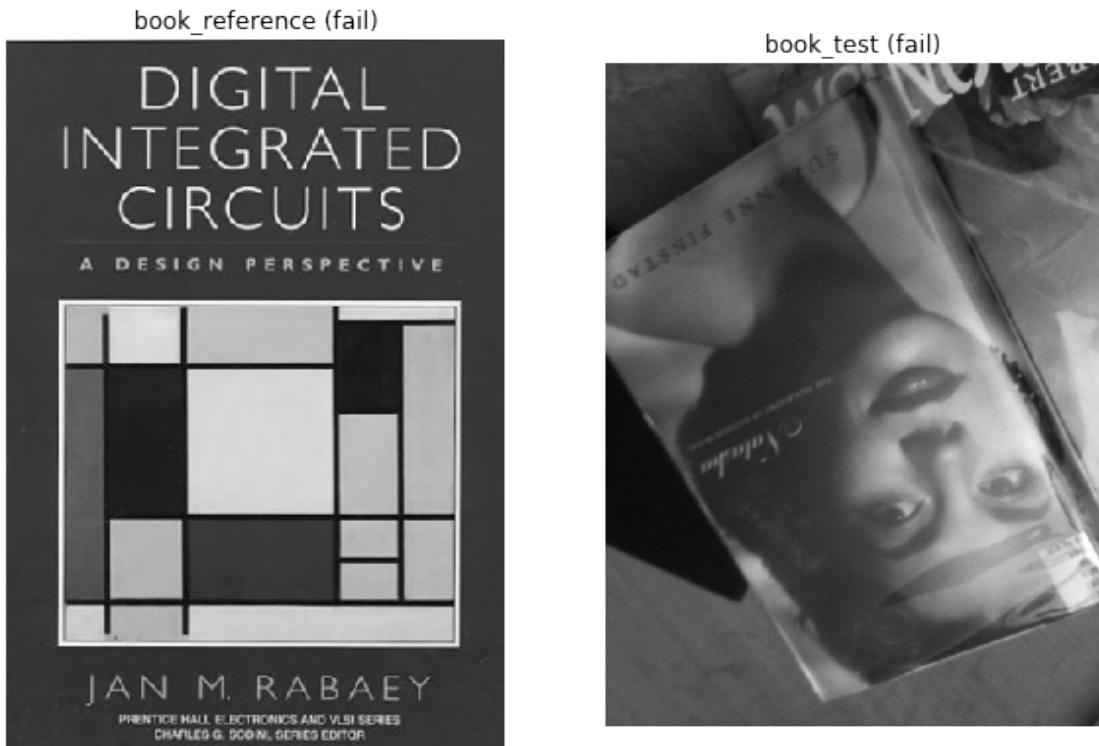
Edited by François Truffaut  
Introduction by Jean Renoir

book\_test (success)



*this case is one of a failed cases (book\_covers 23) :*

We get the number 72 image is the best, it has 48 points matches.



### 3.0.2 Result (use orb)

test case: 10, 12, 13, 67, 56, 89, 46, 23, 59, 81

successful case: 12, 13

failed case: 10, 23, 59, 67, 56, 89, 46, 81

accuracy: **20%**

### 3.0.3 The reason (why will fail):

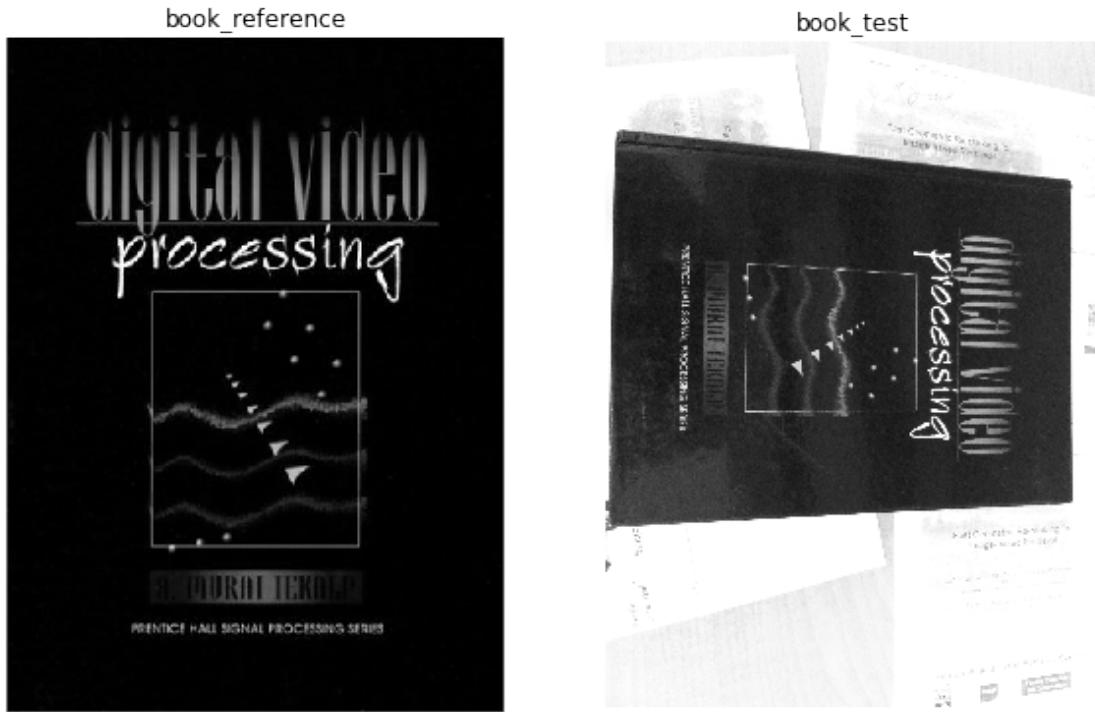
- The main reason I think when we use orb method, it can not detect many keypoints of each images, so it is hard to get correct matches.

I find the accuracy is pretty low, so I try to change the method of detecting keypoints.

### 3.0.4 Using sift method to get matches

*this case is one of a sucessful cases (book\_covers 81)*

We get the number 62 image is the best, it has 155 points matches.



### 3.0.5 Result (use sift)

test case: 10, 12, 13, 67, 56, 89, 46, 23, 59, 81

successful case: 10, 12, 13, 67, 56, 89, 46, 81

failed case: 23, 59

accuracy: 80%

### 3.0.6 Summary

I change the moethod of detecting keypoints, from orb method to sift method. It works good and the accuracy increase **60%** (from 20% to 80%).

I also notice when I do pairs, some piuctures do not have more than 4 points pairs, so the code will report error (in cv2.findHomography step). I add a condition [if len(good\_without\_list) > 4] to avoid this situation. However, this is not perfect, becasue when images do not have four points' pairs, it will skip these images, so the index will be confusion. This is the main reason will cause case failed.

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.

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

## 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 recommended. You need to implement it and save results of 3 image pairs (you can select any image pairs from the dataset), where your 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 you 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>”.

*Your results here*

## 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%)