

# Assignment2

April 13, 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.

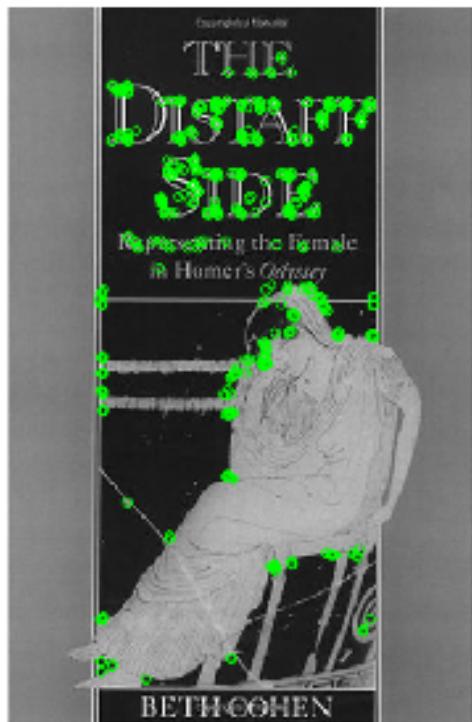
***When converting to PDF, include the outputs and analysis only, not your code.*** You can do this from the command line using the nbconvert command (installed as part of Jupyter) as follows:

```
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 macthes by using orb method

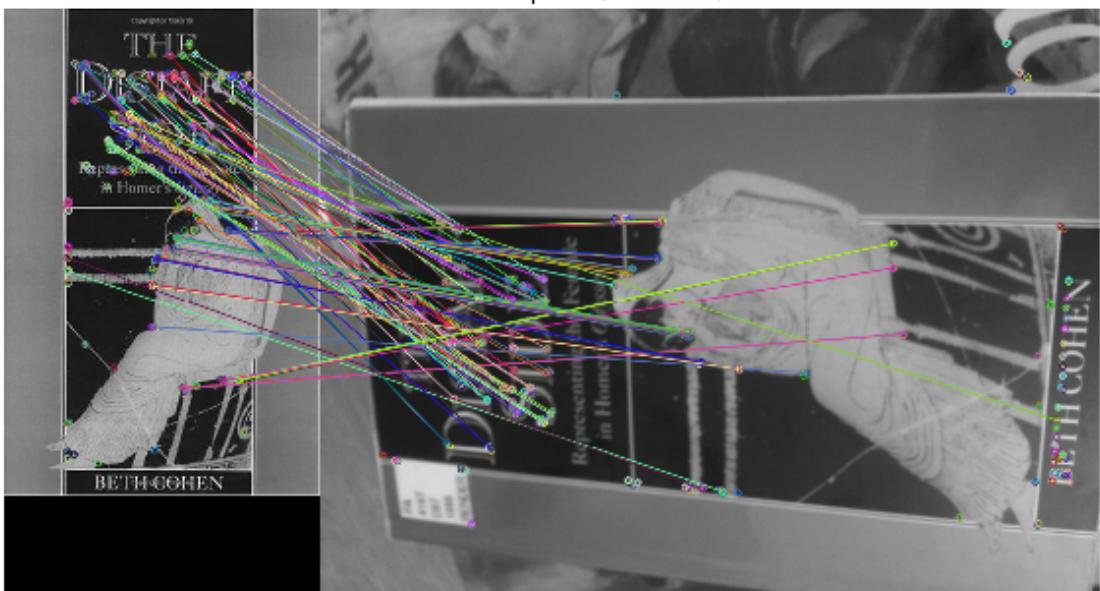
keypoints of reference image (orb)



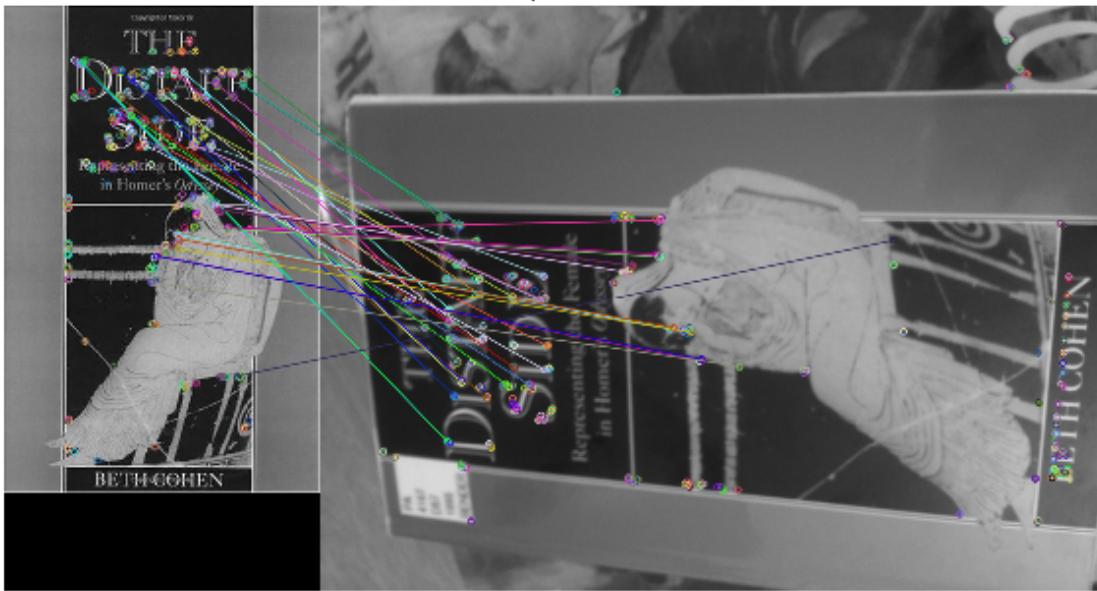
keypoints of query image (orb)



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.

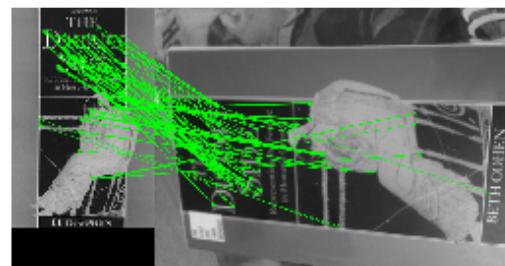
---

### 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

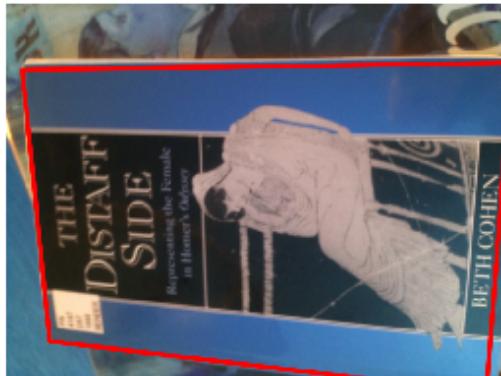
## *Summary*

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

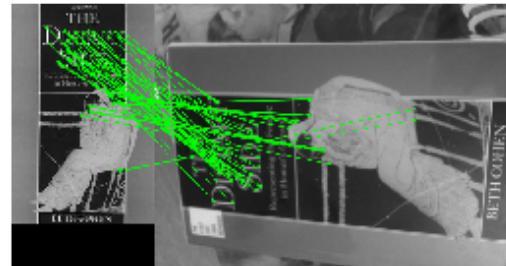
---

### 2.0.3 Draw outline and inliers by using RANSAC

draw outline (using RANSAC)



draw inliers



number of detected inliers: 100

number of detected outliers: 31.0

## *Summary*

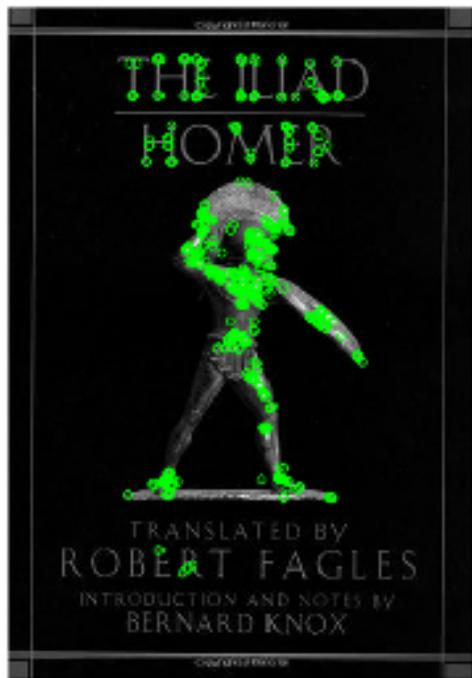
Compare to the RANSAC and the regular method, I notice the number of detected inliers is different. The ***total number of detected points are 131***, and when I use ***RANSAC*** method, the number of detected inliers are ***only 100***, however, when I use ***regular method***, the number of detected 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)

keypoints of reference image (orb)



keypoints of query image (orb)



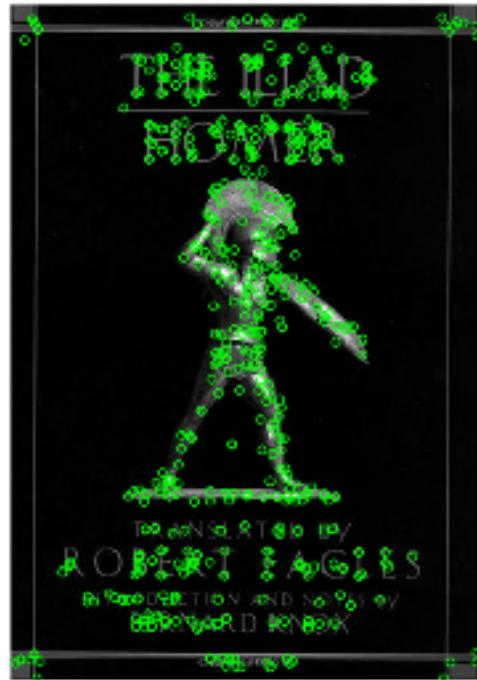
keypoints of reference image (brisk)



keypoints of query image (brisk)



keypoints of reference image (sift)



keypoints of query image (sift)



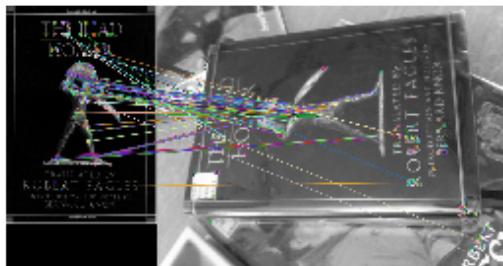
### 2.1.2 summary of finding key ponits part

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

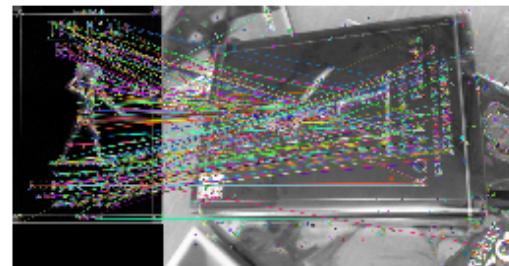
---

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

matches (orb)



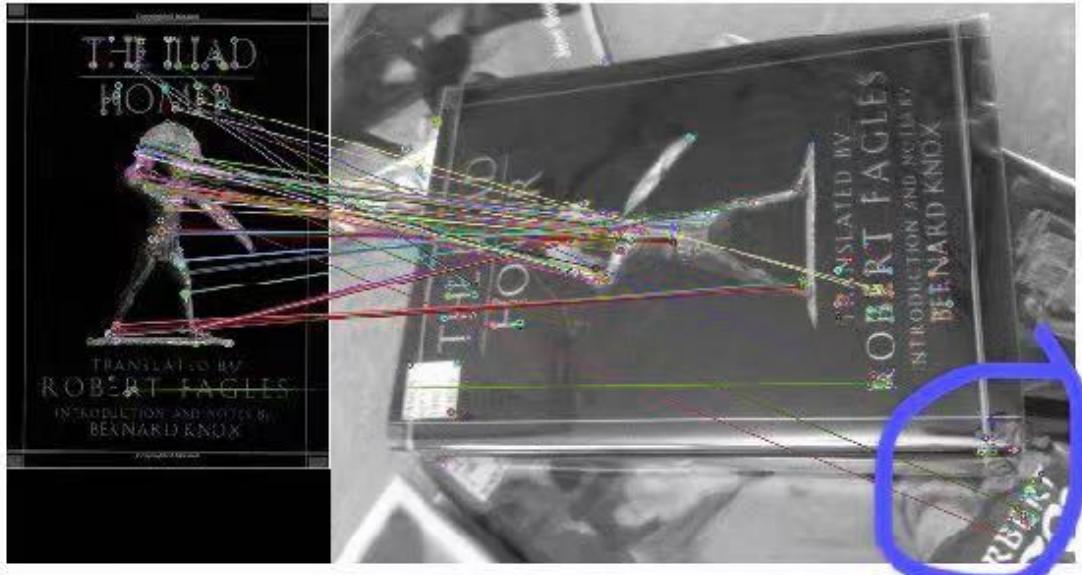
matches (sift)



#### 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)

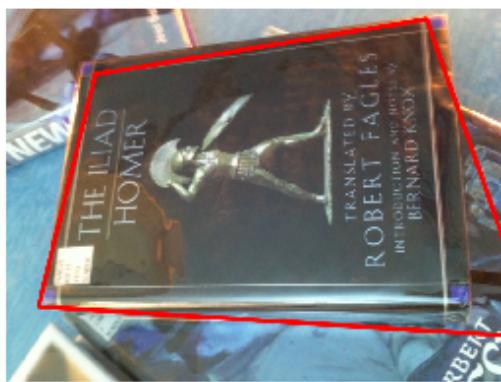
picture: wrong matches (orb)



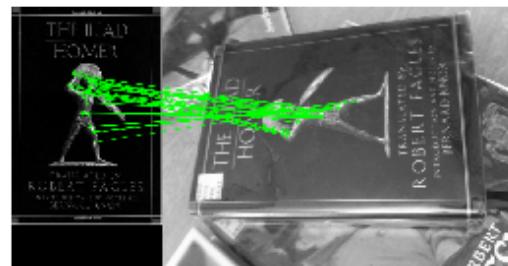
---

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

draw outline (orb)



draw inliers (orb)

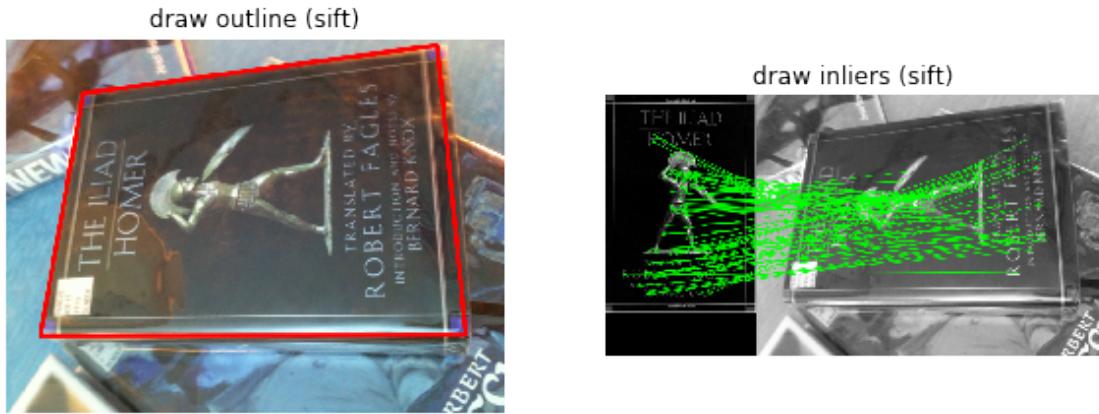


number of detected inliers: 57

number of detected outliers: 36.0

Ratio: 0.6129032258064516

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



```
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 keypoints which outside the book cover may interfere with drawing outlines, this is why we can not get the correct area when we draw outlines.

picture\_01

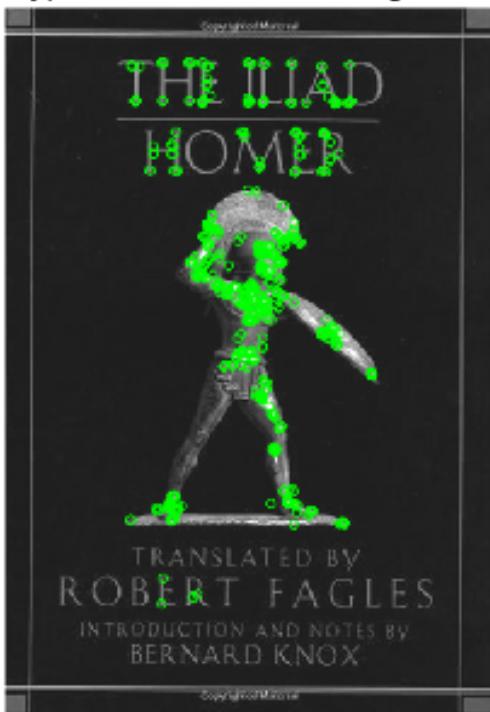


---

### 2.1.8 Little improvement

*However, changing the brightness of the image can solve this problem (wrong outline with orb).*

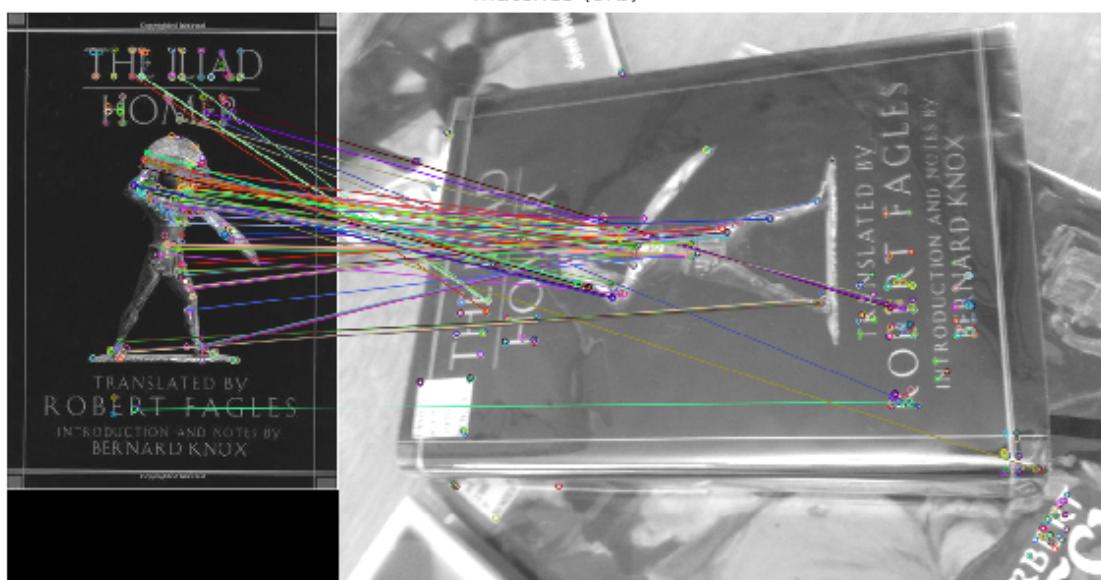
keypoints of reference image (orb)



keypoints of query image (orb)



matches (orb)





```

number of detected inliers: 52
number of detected outliers: 34.0
Ratio: 0.6046511627906976

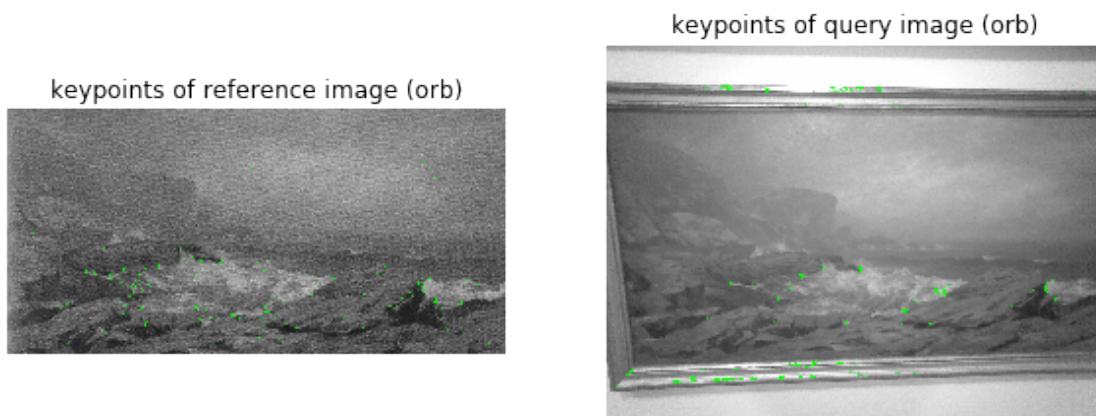
```

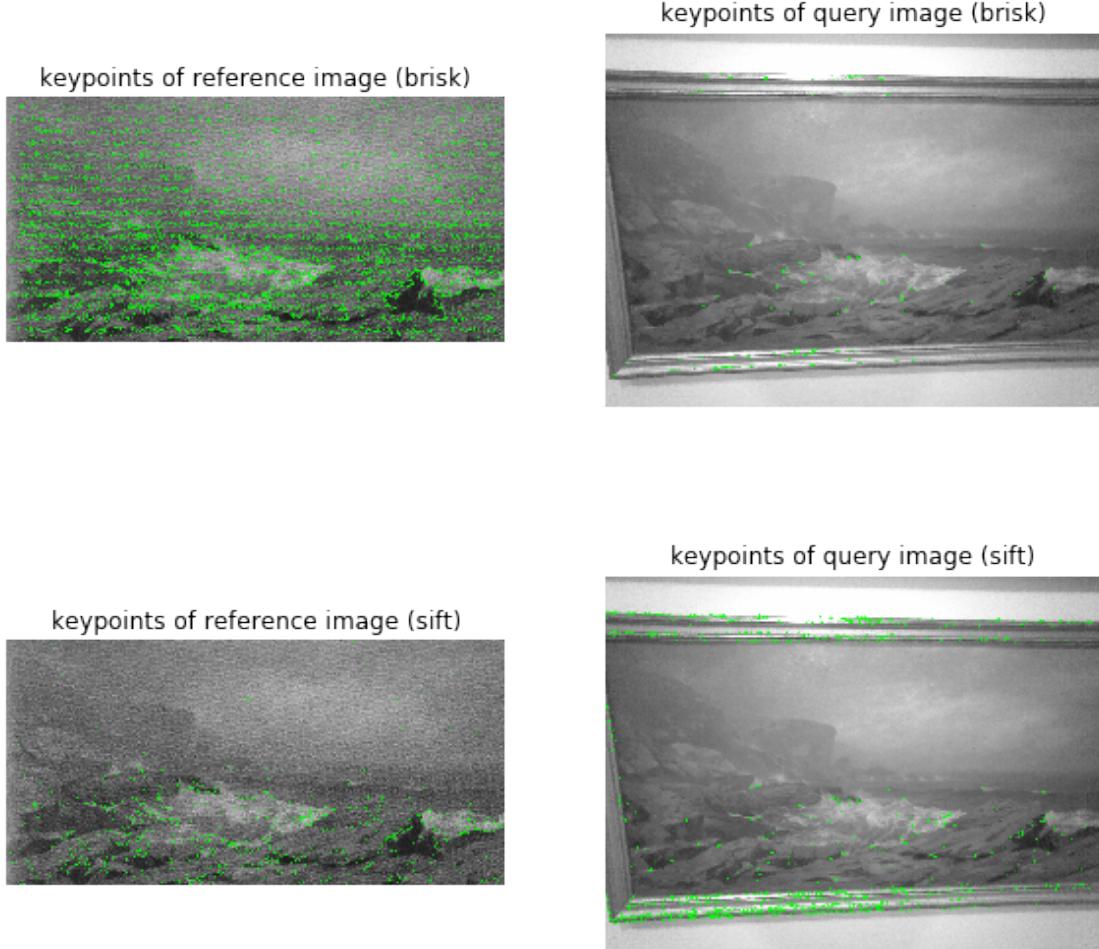
We can see that when we change the brightness (**using cv2.add(image, 30) and set the value to 30**), it makes orb method be more accurate and get a correct outline.

---

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

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



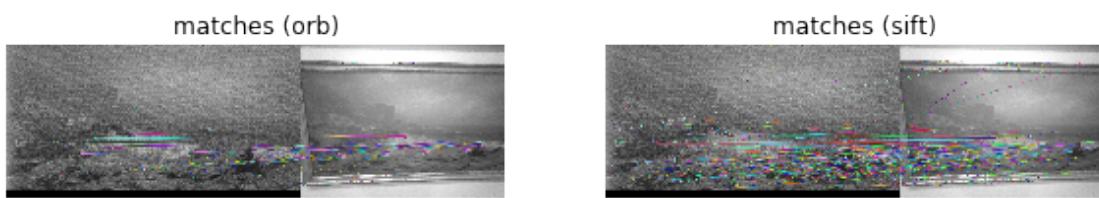


### 2.2.2 summary of finding key ponits part

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

---

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



#### 2.2.4 Summary of drawing match part

In this part, two pictures are draw all mactches correctly. However, sift makes more matches than the orb.

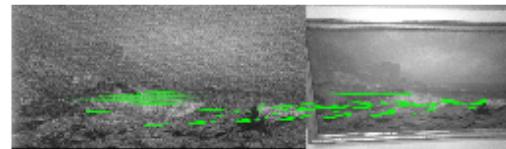
---

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

draw outline (orb)



draw inliers (orb)



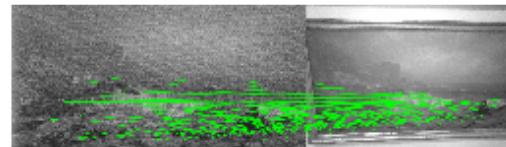
```
number of detected inliers: 52  
number of detected outliers: 9.0  
Ratio: 0.8524590163934426
```

#### 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).

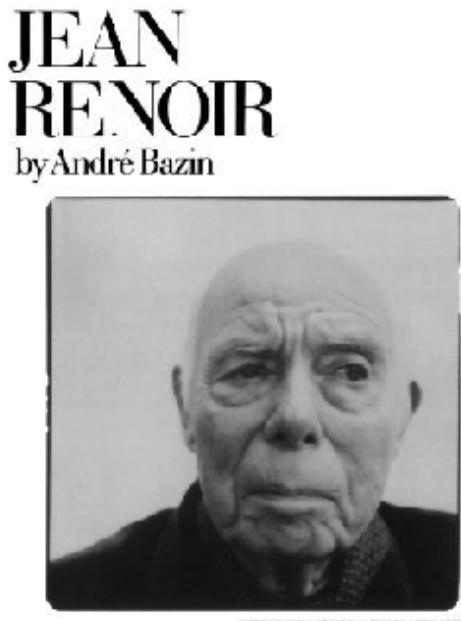
---

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

### 3.1 Using orb

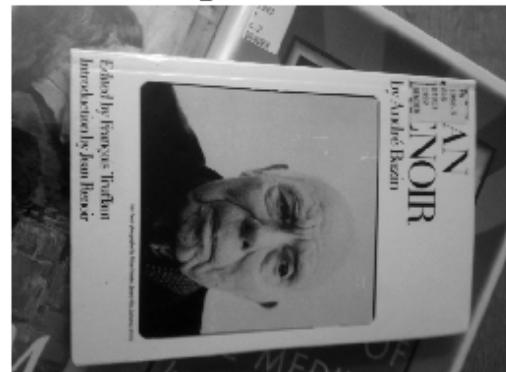
*Example -> this case is one of a suucessful cases (book\_covers 12) :*

book\_reference (success)



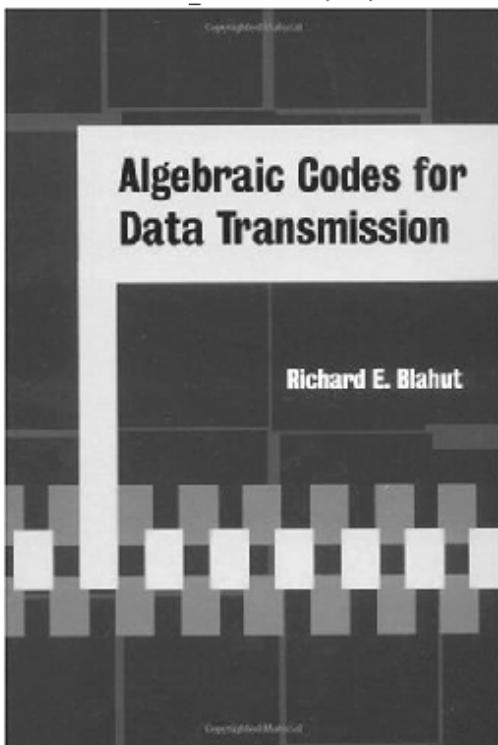
Edited by François Truffaut  
Introduction by Jean Renoir

book\_test (success)



*Example -> this case is one of a failed cases (book\_covers 59) :*

book\_reference (fail)



book\_test (fail)



### 3.1.1 Result – use orb some examples)

*test case : 01, 12, 13, 23, 59, 73, 56, 89, 46, 81, 86 (random select)*

The details are :

** case *****	score	***** result **
image 01	100	success
image 12	56	success
image 13	54	success
image 23	48	success
image 59	11	fail
image 73	13	fail
image 56	12	fail
image 89	9	fail

image 46	9	fail
image 81	23	success
image 86	20	success

### 3.1.2 Summary

numbers of pairs (orb): 66  
accuracy (orb): 65.34653465346535 %

numbers of pairs (orb + top-5): 70  
accuracy (orb + top-5): 69.3069306930693 %

method we used :

- image size : **original**
- feature detector : **orb**
- matching strategy : **KNN**

**Total Accuracy : 65.3465% -> (66/101)**

**Total Accuracy (top-5): 69.3069% -> (70/101)**

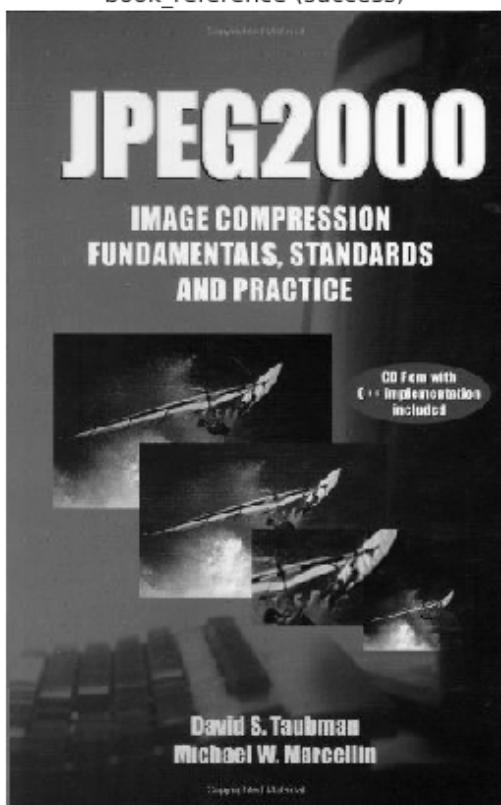
Through apply the top-k accuracy method, we get a higher accuracy, it increase almost 4%! (from 65.3% to 69.3%)

---

## 3.2 Using sift

*Example -> this case is one of a sucessful cases (book\_covers 89)*

book\_reference (success)



book\_test (success)



### 3.2.1 Result – use sift (some examples)

*test case : 01, 12, 13, 23, 59, 73, 56, 89, 46, 81, 86 (randomly select)*

The details are (some cases) :

```
** case ***** score ***** result **
```

image 01	127	success
image 12	163	success
image 13	411	success
image 23	30	fail
image 59	138	success
image 73	160	success
image 56	320	success

image 89	52	success
image 46	154	success
image 81	155	success
image 86	120	success

### 3.2.2 Summary

numbers of pairs (sift): 93  
accuracy (sift): 92.07920792079209 %

numbers of pairs (sift + top-5): 98  
accuracy (sift + top-5): 97.02970297029702 %

what we used :

- image size : **original**
- feature detector : **sift**
- matching strategy : **KNN**

**Total Accuracy : 92.0792% -> (93/101)**

**Total Accuracy (top-5): 97.0297% -> (98/101)**

When we use top-5 accuracy to measure, we can get a high accuracy with sift method. It increases **4.95%**. (from 92.08% to 97.03%)

I change the method of **feature detector**, from orb method to sift method. It works good and the accuracy increases almost **27.7%** (from 69.3% to 97%).

**Small thing :**

- However, I discovered that the image (book\_covers 23) failed in sift method (which has a high accuracy), succeeded in the orb method (which has a low accuracy).

keypoints of reference image (orb)



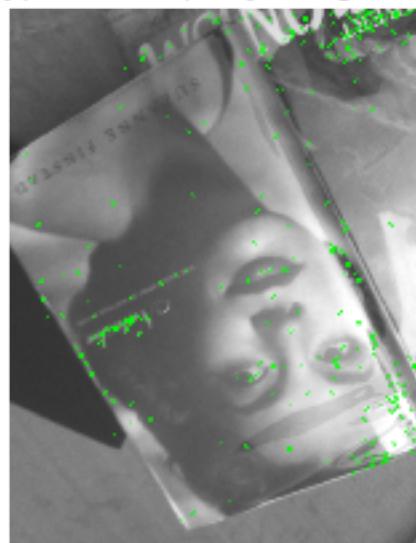
keypoints of query image (orb)

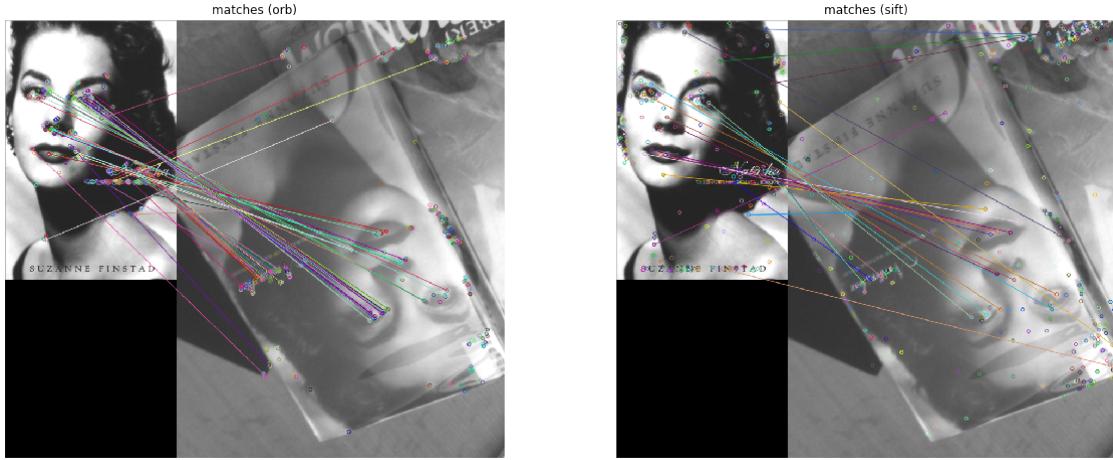


keypoints of reference image (sift)



keypoints of query image (sift)





From the images (macthes (orb), matches (sift)), we can see orb method works better than the sift method in finding keypoints in objects, but in finding keypoints in text area, sift method works better.

### 3.2.3 Summary

After searching the internet, I discover that the difference between ORB and SIFT method.

**ORB:** It computes the intensity weighted centroid of the patch with located corner at center.

In case(book\_covers 023), we can see the reference image shows a strong intensity contrast. So I think this is a reason that orb works better.

So I get an idea that if we resize the image or change the brightness then use orb method, would the accuracy increase?

## 3.3 Changing the size (using orb)

*Example -> this case is one of a sucessful cases (book\_covers 12)*

book\_reference (success)

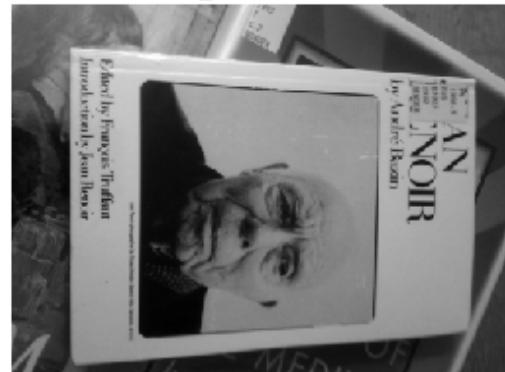
# JEAN RENOIR

by André Bazin



Edited by François Truffaut  
Introduction by Jean Renoir

book\_test (success)



### 3.3.1 Result – change size with factor 0.5 (some examples)

*test case : 01, 12, 13, 23, 59, 73, 56, 89, 46, 81, 86 (random select)*

The details are :

```
** case ***** score ***** result **

image 01      142      success
image 12      53       success
image 13      71       success
image 23      54       success
image 59      44       success
image 73      47       success
image 56      68       success
image 89      26       success
```

image 46	19	success
image 81	115	success
image 86	48	success

### 3.3.2 Summary

numbers of successful pairs (resize - 0.3) : 94  
accuracy : 93.06930693069307 %

numbers of successful pairs (resize - 0.5) : 96  
accuracy : 95.04950495049505 %

numbers of successful pairs (resize - 0.8) : 90  
accuracy : 89.10891089108911 %

what we used :

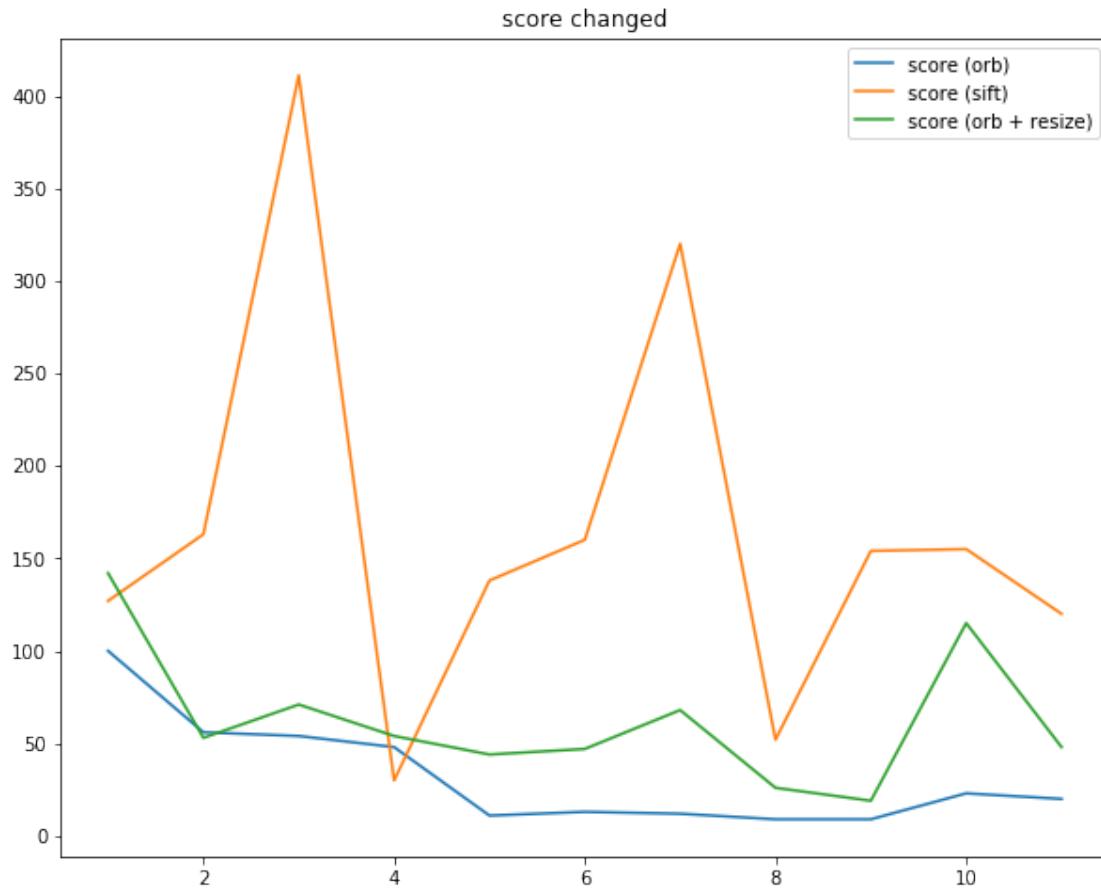
- image size : **0.3, 0.5, 0.8**
- feature detector : **orb**
- matching strategy : **KNN**

**Total Accuracy(with factor 0.3 + top-5): 93.0693% -> (94/101)**

**Total Accuracy(with factor 0.5 + top-5): 95.0495% -> (96/101)**

**Total Accuracy(with factor 0.8 + top-5): 89.1089% -> (90/101)**

When we resize the book covers with factor 0.5, the accuracy is become 95.0693%. Compare to use the orb method without resziing, the accuracy increase **35.74%** (from 69.31% to 95.05%) after resize the image size. Although the total accuracy is lower than the sift method,



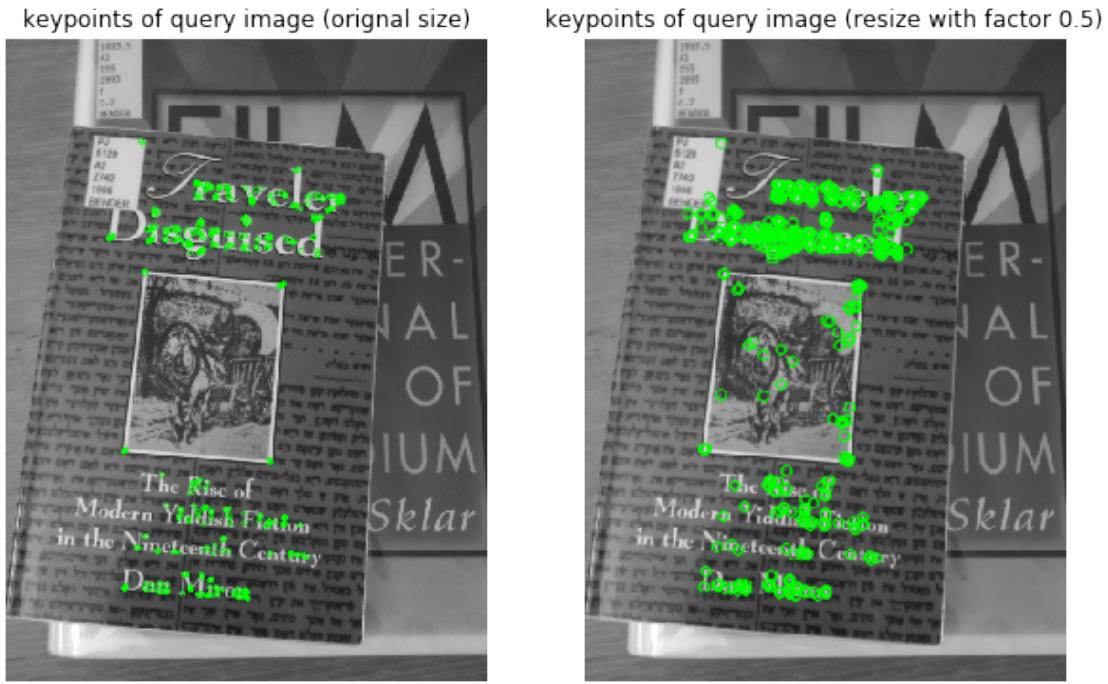
From the image, we can clearly see that sift method works perfect (expect some cases). However, when we use orb method, resize the image with factor 0.5 can increase the score. It helps a lot.

### 3.3.3 Discuss

Here we discuss a case which fail in the original size but success after resizing.

Let's see what differnece between resize and the original size.

**Here is a case book\_covers 046 -> which fail in the original size but success after resizing**



Compare these two images, we can see the key points **increase** a lot when after we resize the image, especially in text area. It is the main reason we can success.

---

## 4 Not in the data set

### 4.1 Using orb

**Result – using orb**

*test case : 01, 02, 03, 04, 05, 06, 07, 08, 09*

**Accuracy : 77.78% -> 7/9**

**The details are :**

```
** case ***** score ***** result **

image 01      N      not in the set
image 02      N      not in the set
image 03      N      not in the set
image 04      N      not in the set
image 05      25     success
```

image 06	N	not in the set
image 07	28	success
image 08	N	not in the set
image 09	N	not in the set

#### 4.1.1 Summary

What we used :

- image size : **original**
- image brightness : **original**
- feature detector : **orb**
- matching strategy : **KNN**

I set a value (13) to the threshold, and the accuracy is **77.78%**.

If the threshold is smaller than the threshold, I identify it is not in the data set. If the threshold is larger than the threshold, I identify it is in the data set.

---

## 4.2 Using sift

**Result – using sift**

*test case : 01, 02, 03, 04, 05, 06, 07, 08, 09*

**Accuracy : 77.78% -> (7/9)**

The details are :

```
** case ***** score ***** result **

image 01      N      not in the set
image 02      69     success
image 03      157    success
image 04      N      not in the set
image 05      N      not in the set
image 06      N      not in the set
image 07      N      not in the set
```

image 08	N	not in the set
image 09	N	not in the set

#### 4.2.1 Summary

What we used :

- image size : **original**
- image brightness : **original**
- feature detector : **sift**
- matching strategy : **KNN**

I set a value (52) to the threshold, and the accuracy is still **77.78%**.

If the threshold is smaller than the threshold, I identify it is not in the data set. If the threshold is larger than the threshold, I identify it is in the data set.

---

### 4.3 Change the brightness

**Result – change the brightness**

*test case : 01, 02, 03, 04, 05, 06, 07, 08, 09*

**Accuracy : 100.00% -> (9/9)**

The details are :

** case ***** score ***** result **		
image 01	N	not in the set
image 02	N	not in the set
image 03	N	not in the set
image 04	N	not in the set
image 05	N	not in the set
image 06	N	not in the set
image 07	N	not in the set
image 08	N	not in the set
image 09	N	not in the set

**Summary** What we used :

- image size : **original**
- image brightness : **add 20**
- feature detector : **orb**
- matching strategy : **KNN**

I change the **image brightness**. And I set a value (13) to the threshold, and the accuracy is **100%**!

If the threshold is smaller than the threshold, I identify it is not in the data set. If the threshold is larger than the threshold, I identify it is in the data set.

---

## 5 Landmarks

### 5.1 In data set

#### 5.1.1 Using orb method

numbers of pairs (orb): 28  
accuracy (orb): 27.722772277227726 %

*Total Accuracy : 27.72% -> (28/101)*

---

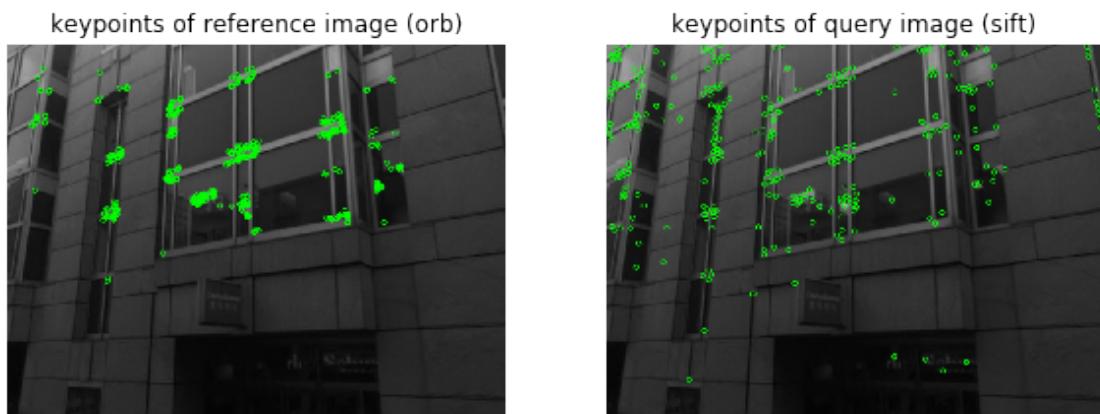
#### 5.1.2 Using sift method

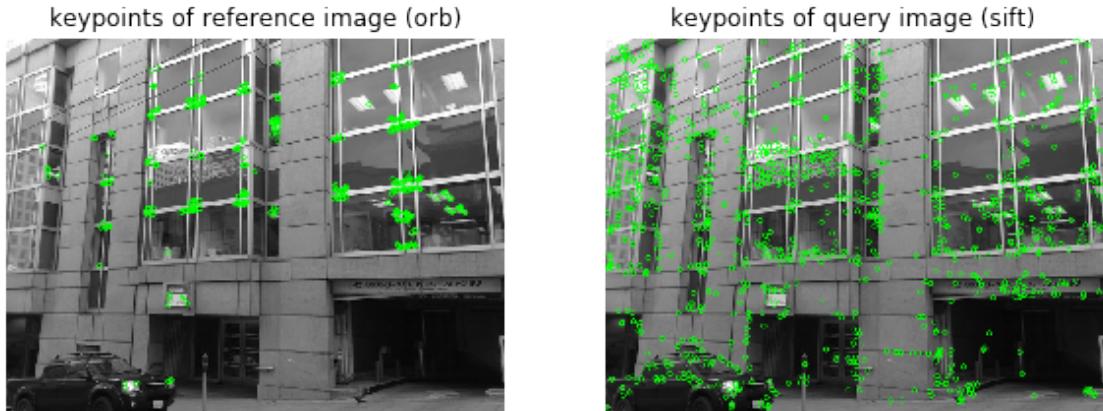
numbers of pairs (sift): 31  
accuracy (sift): 30.693069306930692 %

*Total Accuracy : 30.69% -> (31/101)*

---

#### 5.1.3 Discuss -> landmarks\_018





In this case, we can see if we use orb, it fails. However, if we use sift, we can get the correct area. In detecting key points part, we can see orb method can not detect many key points below the windows (including the wall and the signage). So we can not get the correct area when we use orb method because of the poor quantity of key points.

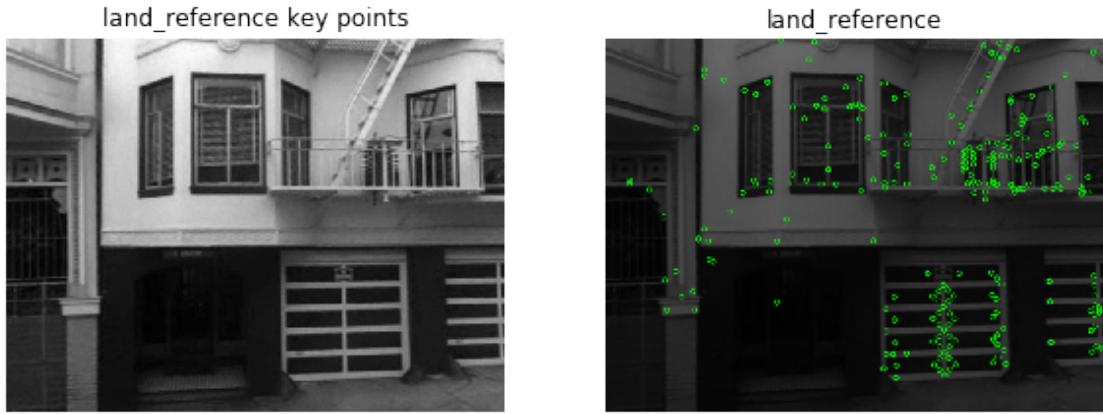
## 5.2 Not in data set

### 5.2.1 Using orb

```
min numbers of successful cases' key points: 11
numbers which not in data set: 1
numbers which not in data set (top-5): 5
```

### 5.2.2 Case -> image 2

```
not in the set  
index: 23  
max_value: 8
```



### Result – using orb

*test case : 1, 2, 3, 4, 5, 6, 7*

The details are :

** case ***** score ***** result **		
image 01	11	success
image 02	N	not in the set
image 03	14	success
image 04	13	success
image 05	18	success
image 06	11	success
image 07	11	success

Total Accuracy: 14.3% -> (1/7)

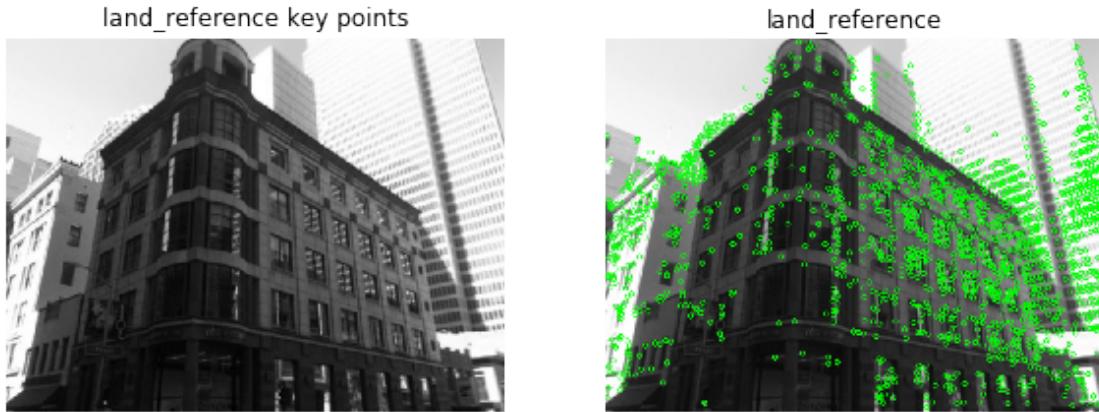
Total Accuracy (top-5): 71.4% -> (5/7)

### 5.2.3 Using sift

```
min numbers of successful cases' key points: 22  
numbers which not in data set: 1  
numbers which not in data set (top-5): 3
```

### 5.2.4 Case – image 7

```
not in the set  
index: 24  
max_value: 19
```



### Result – using sift

*test case : 1, 2, 3, 4, 5, 6, 7*

The details are :

```
** case ***** score ***** result **  
  
image 01      27      success  
  
image 02      141     success  
  
image 03      28      success  
  
image 04      27      success  
  
image 05      51      success  
  
image 06      31      success  
  
image 07      N       not in the set
```

Total Accuracy: 14.3% -> (1/7)

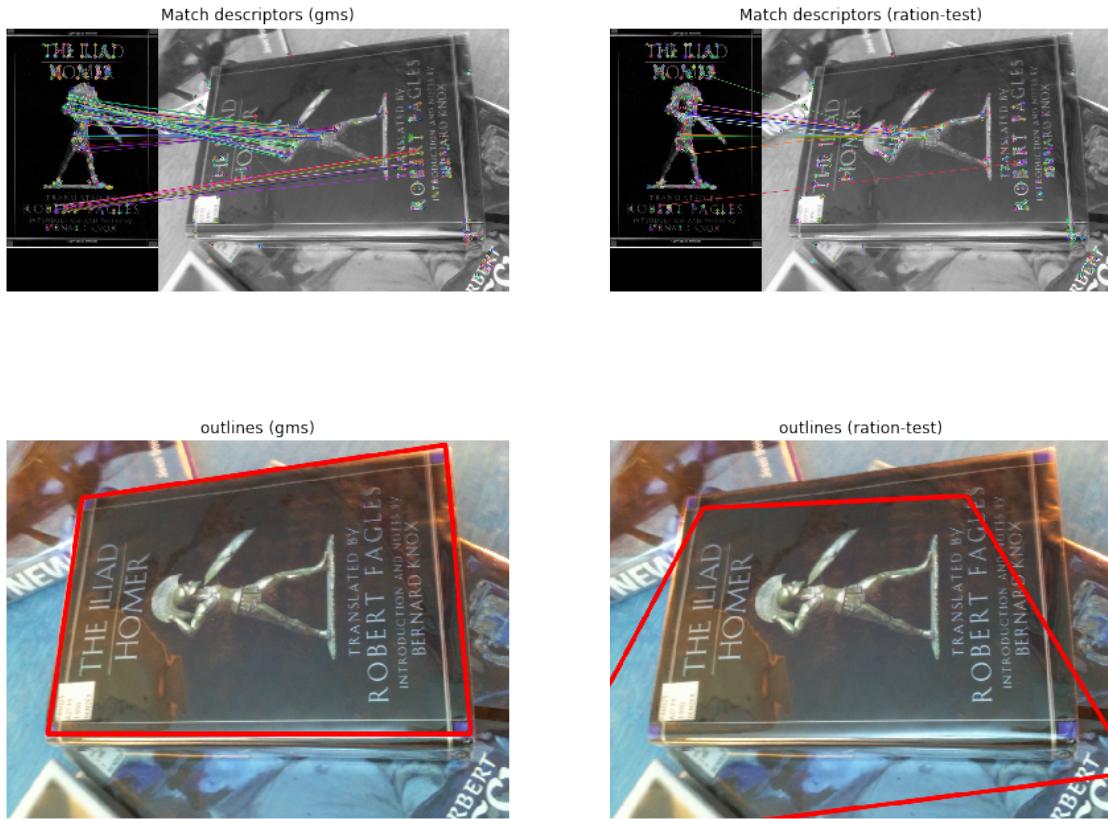
Total Accuracy (top-5): 42.9% -> (3/7)

---

## 6 GMS and ratio\_test (10%)

### 6.1 Book covers

case: book\_covers 025



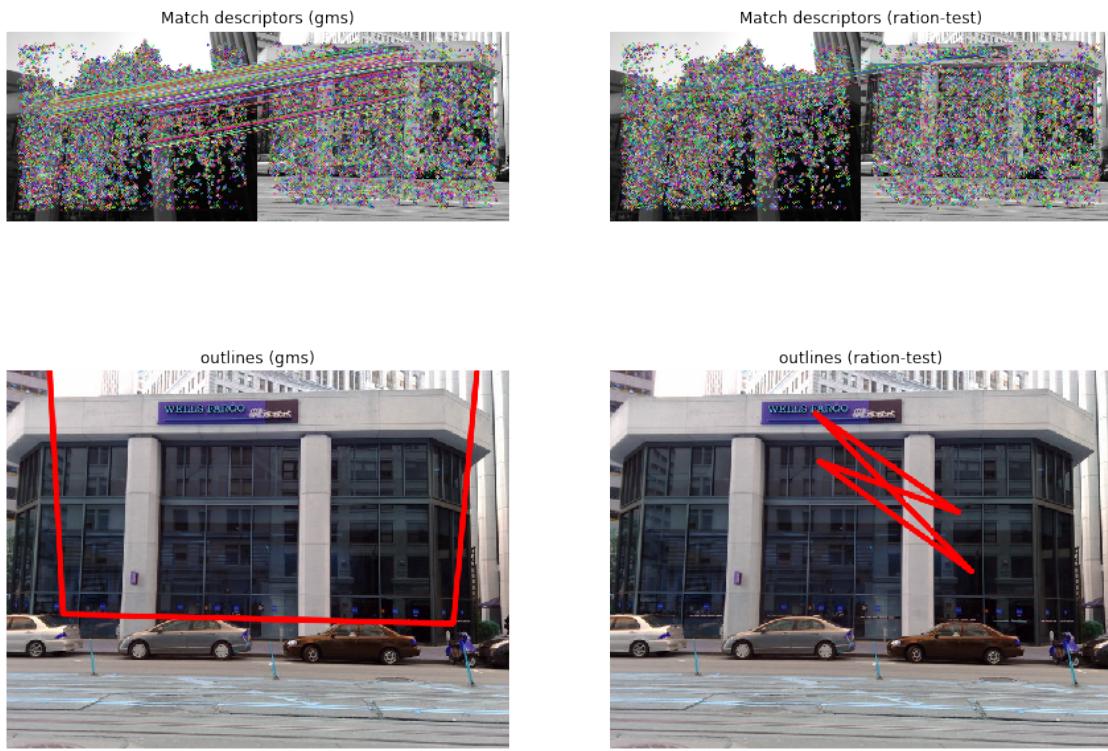
#### 6.1.1 Discovery

At first time, I set numbers of features be **10,000**, GMS and ratio-test method all returns correct outlines. However, when I set number of features be **1000**, GMS do better than the ratio-test method.

---

## 6.2 Landmarks

case: landmarks 0000



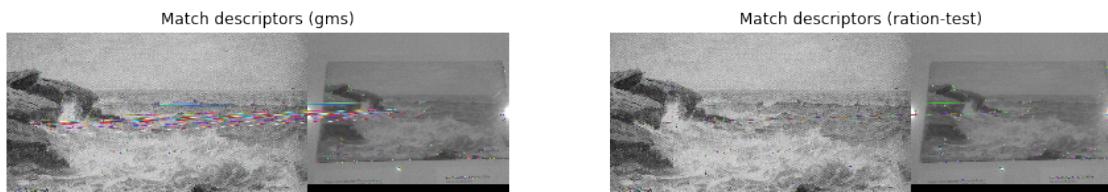
### 6.2.1 Discovery

At first time, I set numbers of features be **20,000**, we can easily see that GMS works better than the ration-test.

---

## 6.3 Museum paintings

case: museum\_paintings 055





### 6.3.1 Discovery

At first time, I set numbers of features be **10,000**, GMS and ratio-test method all returns correct outlines. However, when I set number of features be **1,300**, GMS do better than the ratio-test method.

### 6.4 Summary

All in all, compare to the GMS and ratio-test, GMS works better if two pictures return same numbers of keypoints. Sometimes, when the numbers of keypoints are limited, GMS works better than the ratio-test because GMS has a better matching strategy.

## 7 Reflection Questions (5%)

- Describe the hardest situation you faced during the first two assignments. And how you overcome it? (3%)
  - The hardest part of assignment 1, I think it is implement the convolution part. The algorithm is quite complex, so I need to learn more about the convolution, I searched a lot in the internet and I get lots of useful information to help me to implement the convolution function. resize function also very tricky, I got mix the width and height.
  - The haradest part of assignment 2, I think it has to explain many reasons and how you get improved. I have to write them step by step with the truth (real data), and test them again and again until I find the correct way to improve. By the way, I have to do lots of research in this part to find some inspirations which can help me.
  - However, I think this process is pretty good because through these two assignments, I can study deeper and make many boring knowledge visualization.
- How do you plan to finish the assignment to meet tight deadline? (2%)
  - I start the assignment when the assignment release, although it takes me a lot of time, but still not feel tight.