Report Template for EQ2425 Analysis and Search of Visual Data

EQ2425, Project 1

Yuzhou Liu yuzhoul@kth.se

Yage Hao yage@kth.se

September 17, 2021

Summary

In this project, our first task is to apply SIFT and SURF keypoint detectors on image "obj1_5.JPG" and test the detectors' robustness against some simple transformation such as rotation and scaling.

Our second task is to do image feature matching between the database image "obj1_5.JPG" and the test one "obj1_t5.JPG" using three different algorithms: "fixed threshold" matching algorithm, "nearest neighbor" matching algorithm and "nearest neighbor distance ratio" matching algorithm.

1 Introduction

1.1 Robustness of Keypoint Detector

The local image feature can be regarded as a combination of a robust interest point detector and a feature descriptor. In order to test the robustness of the keypoint detector and descriptor, we need to modify the original image in a variety of ways and evaluate the "repeatability" metric. The repeatability measure is defined as the number of matching features between the original image and the modified image divided by the number of features detected in the original image.[1]

1.2 Image Feature Matching

The key to match a modified image to our database image is to compare the distance between keypoints in the descriptor space. Intuitively, we choose to compute **Euclidean distance** which in Euclidean plane is defined as:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{1}$$

where point x has coordinate (x_1, x_2) and point y has coordinate (y_1, y_2).

To compare images using the obtained distance, we applied three different strategies:

- **Fixed Threshold**: Set a threshold. For distance less than the threshold, we record the keypoint pair as matching.
- **Nearest Neighbor**: For each keypoint in the reference image, find the corresponding keypoint which has shortest distance in the target image and mark such pairs as matching points.
- **Nearest Neighbor Distance Ratio**: We set a threshold. And for each keypoint in the reference image, we compute the value of distance ratio as:

$$distance\_ratio = \frac{distance\_to\_nearest\_neighbor}{distance\_to\_second\_nearest\_neighbor} \tag{2}$$

  Then we filter and get the matching pairs whose distance ratio is less then the ratio threshold.

2 Problem Description

2.1 Robustness of Keypoint Detector

(a) We use SIFT and SURF detectors to find keypoints, with the application of the image obj1_5.jpg to obtain hundreds of keypoints. At the beginning, we give some default values. When peak threshold = 0 and edge threshold = 10, for SIFT detection, approximately 17,000 key points can be found. When the strongest threshold = 1000, for SURF detection, about six thousand key points can be found. Among these key points, most of them are unimportant information, so we hope to eliminate them and only keep a few hundred key points. We decided to use 500 key points. These key points can identify some important features, such as KTH signs, corners of windows, roofs and chimneys.

To get the matching results with the above problem, run *P1_2A.m*.

(b) After finding approximately 500 keypoints, the next is to compare the repeatability with rotation angle. Rotate the image with 15 degree step by step, detecting modified keypoints. Then, rotate the keypoints from the original image, to find if the keypoints of modified image is already matched. Counting our mateches number. At last, compute repeatability by using the matches number.

To get the matching results with the above problem, run *P1_2B.m* with function *Kps_rotate_fun.m*.

(c) Then, we shold focus on scaling factor, compared with repeatability. Scale the images to get 8 different modified images. Find the keypoints of modified image. And scale the coordinates of the original image, to find if the keypoints of modified image is already matched. Counting our mateches number. At last, compute repeatability by using the matches number.

To get the matching results with the above problem, run *P1_2C.m* with function *Kps_scale_fun.m*.

2.2 Image Feature Matching

Apart from the original image's SIFT and SURF features, we need to extract the SIFT and SURF features of the test image. In MatLab, SURF features detection requires the Computer Vision Toolbox, which can be installed in the software. And SIFT features detection requires an open-source VLFeat library. To setup VLFeat for temporary use, run *setup.m* first.

Then, to extract SIFT features for both reference image and test image, run *q3_sift.m*. This will generate 128 dimensional keypoint descriptor, and intuitively we plot images with the keypoints marked.

Now, we can do matching. With background knowledge provided in part1.2, we can implement three matching algorithms based on the following pseudo code:

- **Pseudo code for Fixed Threshold Algorithm**

```
DEFINE threshold
FOR point x in reference image:
    FOR point y in test image:
        compute d(x,y)
        IF d(x,y) < threshold:
            match x,y
```

- **Pseudo code for Nearest Neighbor Algorithm**

```
FOR point x in reference image:
    For point y in test image:
        compute d(x,y)
    get minimum d(x,y)
    match x,y
```

- **Pseudo code for Nearest Neighbor Distance Ratio Algorithm**

```
DEFINE threshold
FOR point x in reference image:
    For point y in test image:
        compute d(x,y)
    get distance_ratio
    IF distance_ratio < threshold:
        match x,y
```

To get the matching results with the above three algorithms applied, run *q3_fixedthreshold.m*, *q3_nn.m* or *q3_nndr.m* respectively.

In addition, to get the matching result on SURF features with Nearest Neighbor Distance Ratio Matching, run *q3_surf_nndr.m*.

3 Results

3.1 Robustness of Keypoint Detector

3.1.1 (a) Finding the thresholds of SIFT and SURF

This problem is dividing by two parts, SIFT and SURF.

For SIFT, the peak threshold is set to 10 and the edge threshold is set to 3.28. Increasing the peak threshold, which is keeping the peak of the DoG scale space larger in absolute value, and lowering the edge threshold, which is keeping only the peak of the DoG scale space with larger curvature. We can find a fewer key points, ensuring that we cover the visual is the most important key point. Here, it is 499 key points, approximating to 500.

For SURF, we slightly increased the value of the strongest threshold until 5730, and found about 500 important key points.

The result is as following, where Figure 1 is SIFT, Figure 2 is SURF:

Figure 1: SIFT with approximately 500 keypoints

Figure 2: SURF with approximately 500 keypoints

3.1.2 (b)Repeatability vs Rotation Angle

From what we have done above, we continue to find the relationship between rotation angle and repeatability. The following Figure 3 plots the relationship between the repeatability of the two key point detection methods and the rotation angle, the increment is 15 degrees, from 0 to 360. We can find that since the rotation has a period of a fixed number of degrees, the two images are repetitive, and the repeatability for a 360-degree rotation is equal to 1, because it compares the image with itself. SURF achieves overall better performance, even if there is no higher margin. In particular, the angle where SURF has a valley, which means performance is poor, is different from the angle where SIFT has a valley. On the other hand, the peaks are reached at the same angle (90°, 180°, and 270°) and perform better in these SURF, with a repeatability of about 1.

3.1.3 (c)Repeatability vs Scaling factor

From what we have done above, we continue to find the relationship between scale and repeatability. In the following Figure 4, the repeatability and scaling factor ($m^0, m^1, \cdots m^8$, where m = 1.2) of the two key point detectors are plotted respectively.

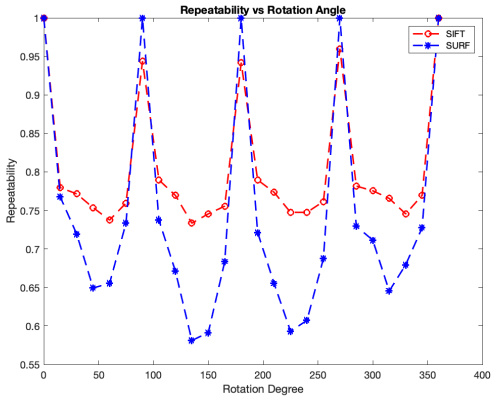

Figure 3: Repeatability vs Rotation Angle

If we compare SIFT and SURF in terms of repeatability in scaling, SIFT performs far better than SURF. Due to the trade-off between performance and speed, SURF is faster than SIFT in finding key points in the zoomed image, while being less efficient and robust.

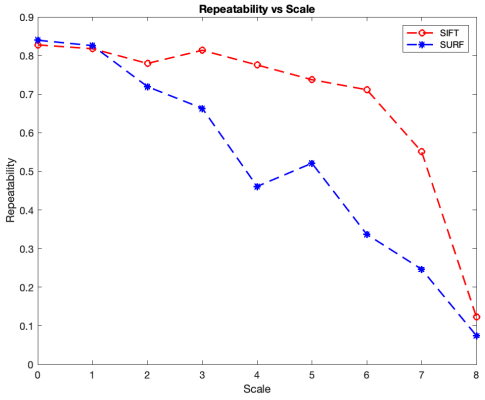

Figure 4: Repeatability vs Rotation Angle

3.2 Image Feature Matching

3.2.1 (a) SIFT Features

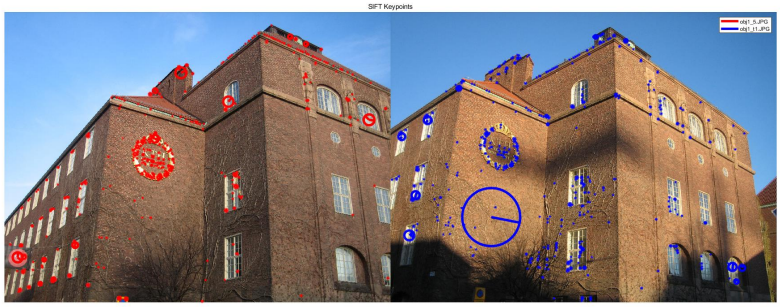

Figure 5: SIFT feature keypoints on reference image (left) and test image (right) with $peak\_threshold = 10$, $edge\_threshold = 3.28$

By adjusting the peak threshold and the edge threshold, we obtained 499 keypoints on reference image and 518 keypoints on test image as shown in Figure 5. For the following parts, we applied matching based on these SIFT feature keypoints.

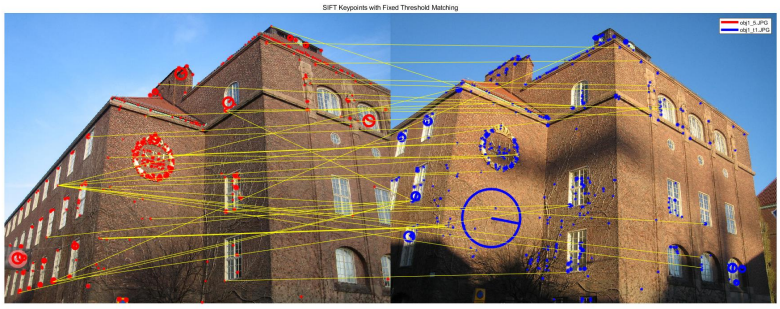

Figure 6: Fixed Threshold Matching on SIFT feature keypoints with $threshold = 60$

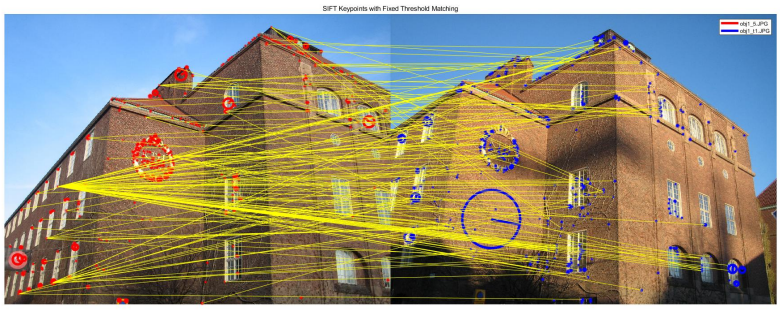

Figure 7: Fixed Threshold Matching on SIFT feature keypoints with $threshold = 65$

3.2.2 (b) Fixed Threshold Matching

By adjusting the threshold, we noticed that as threshold increasing, the number of matching pairs increases. As we set threshold to 60 (Figure 6), 43 matches are formed, while there are 235 matches with threshold setting to 178 (Figure 7). The larger threshold will provide more matches, but probably include more wrong matches with one-to-many relationships.

3.2.3 (c) Nearest Neighbor Matching

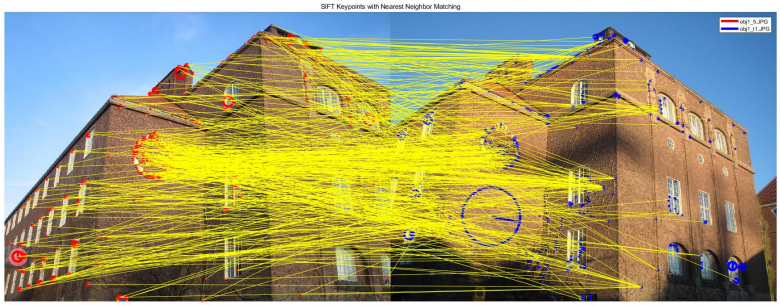

Figure 8: Nearest Neighbor Matching on SIFT feature keypoints

As shown in Figure 8, every point of the reference's 499 keypoints is matched to its nearest keypoint on test image. For those points that appear in both images, the Nearest Neighbor matching provides a reliable result. However, for those that only appear once, the Nearest Neighbor algorithm gives a mismatch.

3.2.4 (d) Nearest Neighbor Distance Ratio Matching

For the Nearest Neighbor Distance Ratio Matching, we first tried the empirical ratio threshold of 0.8 and only obtained 33 matched pairs (Figure 9), though there are only a few wrong matches. We then tried to increase the ratio threshold to 0.85, and the visualized result (Figure 10) is also satisfying since more feature keypoints, like the ones at the edge of the building and on the KTH logo, are paired.

3.2.5 (e) SURF Features with Nearest Neighbor Distance Ratio Matching

Similar to part 3.2.4, we applied Nearest Neighbor Distance Ratio Matching on SURF feature keypoints, set ratio threshold to 0.8 and obtained Figure 11. To make sure the comparison are on a similar level, we took the 500 strongest keypoints

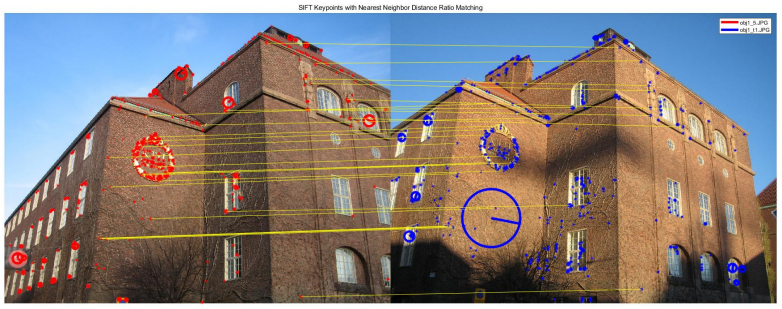

Figure 9: Nearest Neighbor Distance Ratio Matching on SIFT feature keypoints with $threshold = 0.8$

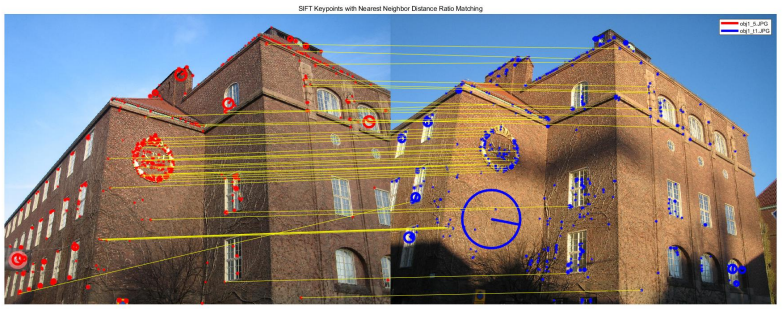

Figure 10: Nearest Neighbor Distance Ratio Matching on SIFT feature keypoints with $threshold = 0.85$

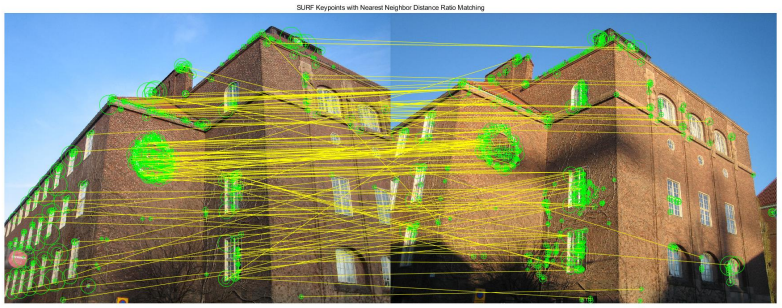

Figure 11: Nearest Neighbor Distance Ratio Matching on SURF feature keypoints with $threshold = 0.8$

from SURF features, which is close to the number of SIFT feature keypoints in test image. Figure 11 shows 140 matches, which is higher than 33 matches in Figure 9. Through the visual examination, we noticed the Nearest Neighbor Distance Matching on SURF keypoints performed well both on correctness and efficiency. And we also find that the wrong matches are mostly happened on those keypoints that lie on the edge of windows.

4 Conclusions

In conclusion, for key point detection, SIFT performs better in terms of scaling. On the other hand, SURF is better when the target image rotates at a faster speed. For key point matching, in the three SIFT key point matching strategies, the nearest neighbor ratio finds a more correct match, followed by the nearest neighbor and a fixed threshold. In a wide range of situations, if we compare the nearest neighbor ratios of SIFT and SURF, SURF performs better than SIFT, and the number of accurate matches is larger. Some improvements can be made to the matching, such as adjusting the peak and edge thresholds to obtain more key points. At the same time, we guess that if different distance measurement metrics are used to search for the nearest neighbors in the descriptor space, the results may be improved.

Appendix

Who Did What

Yage Hao is responsible for the "Image Feature Matching". In detail, she implemented question 3's code and wrote the report's 'Summary', '1.2', '2.2' and '3.2' parts.

Yuzhou Liu is responsible for the "Robustness of Keypoint Detector". In detail, he implemented question 2's code and wrote the report's '1.1', '2.1', '3.1' and 'Conclusions' parts.

References

[1] Markus Flierl, *EQ2425 Analysis and Search of Visual Data*, Lecture Slides, 2021

[2] Wikipedia, *Euclidean distance*, Available:
https://en.wikipedia.org/wiki/Euclidean_distance, Last edited: August 15, 2021