EQ2425 Analysis and Search of Visual Data

EQ2425, Project 1

Wenrui Zhao	Jingxi Huang	Letao Feng
wenruiz@kth.se	Jingxi@kth.se	letao@kth.se

September 16, 2024

Summary

The project is mainly about image features and matching. First, we test and compare the robustness of SIFT and SURF keypoint detectors against rotation and scale changes. Our second task is to do image feature matching between a query image and a database image. We implement three feature matching algorithms and compare their performance.

1 Introduction

In this project, we mainly focus on Image Features and Matching.

In the first part, we study two detectors: SIFT and SURF. SIFT features and SURF features are both excellent scale-invariant features and are commonly used for object recognition and image matching. SIFT (Scale-Invariant Feature Transform) feature is a feature extraction algorithm for computer vision. The main steps including Scale-space extrema detection, Keypoint localization, Orientation assignment, and Keypoint descriptor. SURF (Speeded Up Robust Features) is also a robust image recognition and description algorithm. It is an efficient variant of SIFT. SIFT is usually more accurate in searching for correct features and of course more time consuming, whereas SURF's keypoint descriptors are mostly based on the difference of intensities, which makes the computation faster.

In the second part, we pay more attention to three feature matching algorithms. The fixed threshold algorithm matches features if the descriptor distance is below a set threshold, which is sensitive to the threshold value. The nearest neighbor algorithm finds the closest match by selecting the descriptor with the smallest distance, but it can result in false matches. The nearest neighbor distance ratio improves accuracy by comparing the distance of the best match to the second-best match, ensuring the nearest match is significantly better, which reduces incorrect matches.

2 Problem Description

2.1 Image Features

In this part, we mainly study the robustness of keypoint detectors and descriptors.

In problem(a), we firstly apply the SIFT detector on the original image. In Python, we can implement it using the OpenCV library. To control the number of detected keypoints in a few hundred, we manually set the parameters. The contrastThreshold controls the contrast threshold of the keypoints.The larger the contrastThreshold, the fewer keypoints are returned. And the edgeThreshold controls the threshold along the edge of a keypoint. The larger the edge threshold, the more keypoints are returned (the fewer are filtered out).

Similarly, we also apply the SURF detector on the original image. Here we use this code to create surf detector: surf = cv2.xfeatures2d.SURF_create(hessianThreshold=5000). The hessianThreshold is the threshold for keypoint detection, the higher it is, the fewer points are detected.

In the next step, to test the robustness of keypoint detectors and descriptors, we need to modify the original image, such as rotating and scaling, and then apply the detectors on the modified image. Finally, in order to quantify the robustness, we here introduce the "repeatability" measure.

$$\text{Repeatability} = \frac{\text{Number of matching features between the original image and the modified image}}{\text{Number of detected features in the original image}} \tag{1}$$

In problem(b), the image needs to be rotated. The function first calculates the image's center, then generates a 2D rotation matrix using that center and the angle. Finally, it applies the rotation to the image and returns the rotated result.

Then we apply the detectors on the rotated image and calculate the repeatability. Here we use SIFT detector as an example.

```python
def compute_repeatability(detector, img1, img2, angle):  2用法
    # Detect keypoints and descriptors in original and rotated images
    kp1, des1 = detector.detectAndCompute(img1, None) #original image
    kp2, des2 = detector.detectAndCompute(img2, None) #rotated image
    # Predict the position of each key point in the rotated image
    predicted_kp1 = []
    (h, w) = img1.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle,  scale: 1.0)
    for kp in kp1:
        pt = np.array([kp.pt[0], kp.pt[1], 1.0]) #to array
        predicted_pt = np.dot(M, pt) #where it should be after rotation
        predicted_kp1.append((predicted_pt[0], predicted_pt[1]))
    # Check if the keypoint is in the neighborhood of the predicted position
    repeatable_count = 0
    for (x1, y1) in predicted_kp1:
        for kp in kp2:
            x2, y2 = kp.pt
            if abs(x2 - x1) <= 2 and abs(y2 - y1) <= 2:
                repeatable_count += 1
                break
    # compute repeatability
    repeatability = repeatable_count / len(kp1) # if len(kp1) > 0 else 0
    return repeatability
```

Figure 1: compute repeatability

The function detects keypoints in both images, predicts where the original keypoints should appear in the rotated image using a rotation matrix, and checks if the detected keypoints in the rotated image are near the predicted positions. Finally, it calculates the repeatability.

In the end, by setting different angles and calling the function, we can get the image of "Repeatability vs Rotation Angle".

In problem(c), the image needs to be scaled. We almost do the same thing as in problem(b). Finally, we set scale_factors and use the cv2.resize function to scale the image. After calling the above function, we can get the image of "Repeatability vs Scaling Factor".

2.2 Image Feature Matching

In this section, we explore feature extraction using the SIFT detector and implement three different feature matching algorithms. Our goal is to analyze the performance of these matching techniques and investigate the robustness of the SIFT features under various conditions.

In problem (a), we start by extracting keypoints from the original images using the SIFT detector. This is implemented in Python using the OpenCV library. To control the number of detected keypoints, we adjust the parameters of the SIFT detector. Once the keypoints are extracted, they are superimposed onto the original images (obj1_5.JPG and obj1_t5.JPG) to visualize the detected features. This step provides an initial look at the keypoint distribution across the images, as shown in Figure 5.

The next phase involves matching these keypoints between images using three different algorithms: fixed threshold, nearest neighbor, and nearest neighbor distance ratio. Each method has its own strengths and weaknesses in terms of matching accuracy and robustness.

In problem (b), we implement the fixed threshold matching algorithm. This approach compares the descriptors of keypoints from two images and retains matches with a distance below a specified threshold. Adjusting the distance threshold allows us to filter out weaker matches. The objective is to find a balance where we obtain sufficient matches without including too many incorrect pairs. This algorithm is implemented by computing the Euclidean distance between the descriptors of keypoints in the query and database images, and retaining matches where the distance is below the manually defined threshold. Then we experiment with different thresholds to find the best and a suboptimal match.

In problem (c), we implement the nearest neighbor matching algorithm. For each keypoint in the query image, the NN algorithm finds the keypoint in the database image with the smallest descriptor distance. For each keypoint in the first image, this algorithm finds the keypoint in the second image whose descriptor has the smallest distance, and retains only the closest match for each keypoint.

In problem (d), we implement the nearest neighbor distance ratio matching algorithm. This approach introduces an additional constraint: it considers a match valid only if the ratio of the distances of the nearest

neighbor to the second nearest neighbor is below a specified ratio threshold. This technique helps filter out ambiguous matches. For each keypoint, it find the two closest descriptors in the database image, and compute the ratio of the distance to the nearest neighbor and the second nearest neighbor. Then retain the match if this ratio is below a manually defined threshold. We experiment with different ratio thresholds to observe the impact on matching accuracy.

In problem (e), we extract SURF features from the test image and implement the nearest neighbor distance ratio matching algorithm. We then compare the result with those obtained from the SIFT feature matching process in problem(d), to evaluate the differences in feature detection and matching performance between SURF and SIFT.

3 Results

3.1 Image Features

The result of problem(a) is shown below:

Figure 2: Keypoints detected by SIFT (left) and SURF (right)

We set the parameters as follows: For SIFT, contrastThreshold=0.17, edgeThreshold=10. For SURF, hessianThreshold=5000. Thus the SIFT detector can detect 563 keypoints, and the SURF detector can detect 740 keypoints.

Based on the results, we can find many key points on the logo and the roof. For the SIFT detector, it is easier to detect plants on the wall. For the SURF detector, it is more sensitive to the left part of the window.

For problem(b), we have:

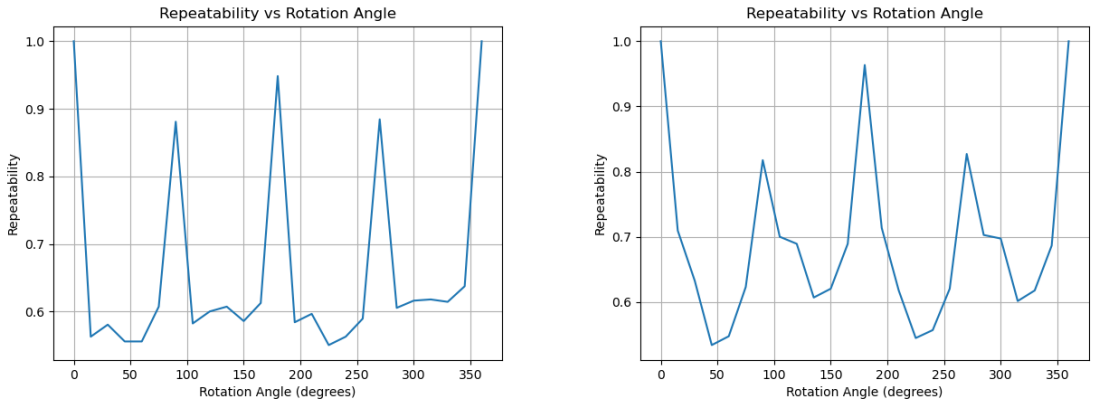


Figure 3: Repeatability vs Rotation Angle [SIFT (left), SURF (right)]

From the result, we can find the repeatability is high when the image is rotated by a multiple of 90 degrees. In this application, the SIFT detector generally performs better than the SURF detector. We considered whether it had something to do with the number of keypoints, as there was a difference of two hundred keypoints between the two detectors. So we went back and adjusted the parameters of the detectors, but the results showed no significant change.

For problem(c), we have:

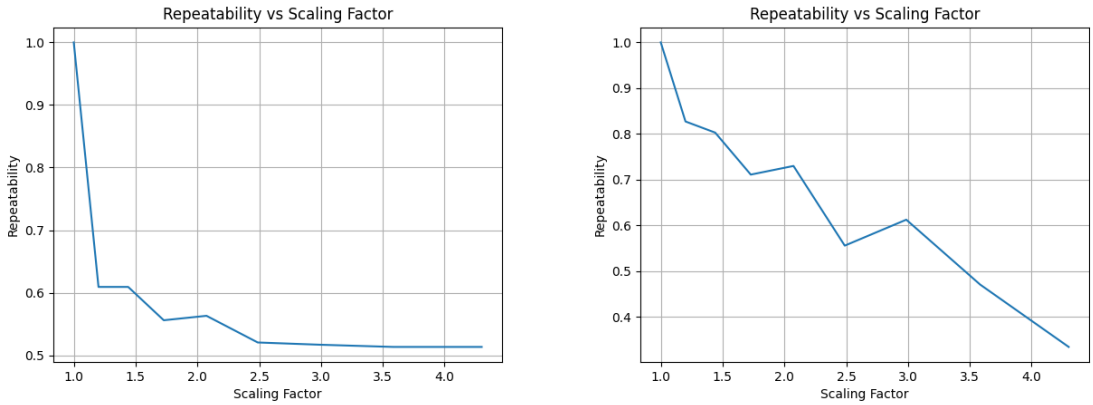


Figure 4: Repeatability vs Scaling Factor [SIFT (left), SURF (right)] with Different Vertical Coordinate Ranges

As can be seen in the figure, the curve shows a decreasing trend. This is relatively easy to understand because the performance of the detector decreases as the image is scaled.

From the result, we can find that the SIFT detector performs better than SURF. Note that the ranges of the vertical coordinates are different in these two graphs! The SIFT detector consistently has higher repeatability across different scaling factors. However, the SURF detector shows lower repeatability, with more significant drops.

3.2 Image Feature Matching

Firstly, in problem (a) We used the OpenCV library to extract SIFT features. The algorithm detects keypoints in both images and computes their descriptors. Figure 5 shows the extracted keypoints superimposed on the images.

(a) SIFT Features on ‘obj1_5.JPG‘

(b) SIFT Features on ‘obj1_t5.JPG‘

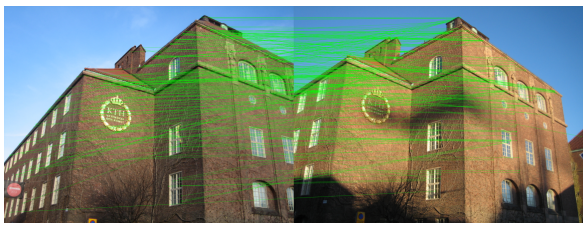
Figure 5: SIFT Features Extracted from the Images

Next, we implement and evaluate three different feature matching algorithms: fixed threshold, nearest neighbor, and nearest neighbor distance ratio.

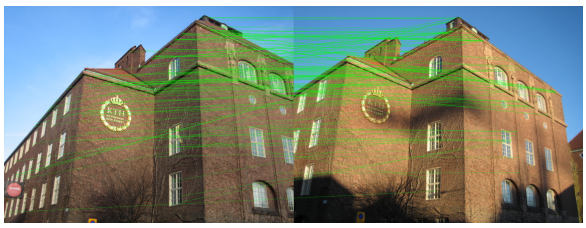
The fixed threshold matching algorithm calculates the distance between descriptors of keypoints from two images and matches pairs that fall below a specified distance threshold. We adjusted the distance threshold until we obtained a satisfactory result.

Figure 6 shows two matching results: the optimal result and a suboptimal result using a different distance threshold.

From the results, we can see that most matches are correct despite a few mistakes. However, the fixed threshold algorithm's performance heavily depends on the chosen distance threshold. A lower threshold results in fewer but more reliable matches, while a higher threshold includes more matches, some of which may be incorrect. Therefore the result might show good matches if the threshold is well-tuned, but could miss or falsely include matches if the threshold is not optimal.



(a) Best Matching Result (Threshold = 75)



(b) Suboptimal Matching Result (Threshold = 70)

Figure 6: Fixed Threshold Matching Results

The nearest neighbor matching algorithm selects the keypoint in the second image whose descriptor has the minimum distance to a keypoint in the first image. This approach finds the closest match for each keypoint.

Figure 7 shows the matching result obtained using the nearest neighbor algorithm.

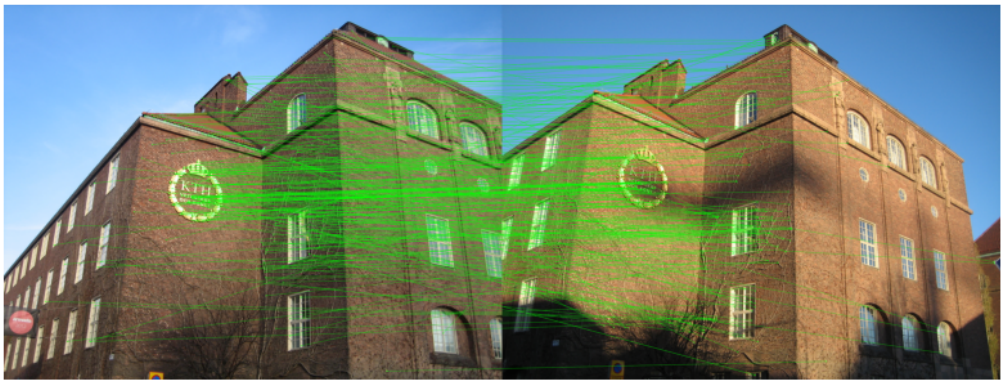


Figure 7: Nearest Neighbor Matching Results

As the results shown, the nearest neighbor matching algorithm finds more matches compared to the fixed threshold method. However, it may include some incorrect matches due to relying only on the nearest descriptor, which could result from noise or similar-looking features.

For problem(d), we implement nearest neighbor distance ratio matching algorithm and the result is shown in figure 8. First we adjust the ratio threshold to 0.75, and from the figure we can observe 63 matches with only 2 mismatches. For the suboptimal result we set the ratio threshold to 0.9 and few mismatches appear within all 183 matches. Compare to fixed threshold algorithm and nearest neighbor algorithm, it is clear that neighbor distance ratio matching algorithm performs the best.

Then we extract SURF features from the images and implement the nearest neighbor distance ratio matching algorithm. The ratio threshold is set to 0.75 and the result in shown in figure 9. It can be observed that the nearest neighbor distance ratio matching algorithm works well on both SIFT and SURF features. However, SURF performs slightly inferior to SIFT, since there are more mismatches on the edge of the windows. The reason for this is that the use of integral images, approximate Hessian matrices, and less detailed descriptors which leads to a reduction in the uniqueness of keypoints and descriptors.

4 Conclusions

In conclusion, SIFT performs better than SURF both in robustness and image features matching. Its use of scale-space theory, accurate keypoint localization, robust orientation assignment, and detailed descriptors enables it to capture distinctive features that are resistant to changes and ensures reliable feature matching. While in contrast, SURF simplify certain steps to improve speed, often at the cost of the robustness and accuracy that SIFT provides.

(a) Optimal Matching (Ratio Threshold=0.75)

(b) Suboptimal Matching (Ratio Threshold=0.9)

Figure 8: Nearest Neighbor Distance Ratio Matching Results

Figure 9: Nearest Neighbor Distance Ratio Matching on SURF features(Threshold=0.75)

Of the three algorithms, the nearest neighbor distance ratio matching algorithm performs the best by showing least mismatches, since it combines the simplicity of nearest neighbor matching with an added measure of confidence that significantly improves its performance.

Appendix

Who Did What

	contribution
Letao Feng	33.3...%
Jingxi Huang	33.3...%
Wenrui Zhao	33.3...%

References

[1] Markus Flierl, EQ2425 Analysis and Search of Visual Data, *Lecture Slides*, 2024