# Implementation and Analysis of Various Corner Detection Algorithms

Kenichi Sakamoto[#1], Hyunhee Kwak[*2], Ziyad Kassar[#3]

***Abstract*— Our study displays a comparison of various corner detection algorithms where we used a set of real world images and complexities such as noise and scale variation in order to find out which algorithm is the most efficient for feature detection and matching. The algorithms that we used were Scale Invariant Feature Transform (SIFT), Oriented FAST and Rotated BRIEF (ORB), and KAZE. Each algorithm's performance was evaluated using a set of images that featured different archeological structures such as Mt. Rushmore and Notre Dame under different conditions. Our analysis made use of the OpenCV library for the algorithm implementation in order to understand how each technique affects feature detection and matching results. We discussed the computational intensity of each of the algorithms and their applicability to real world uses in different scenarios.**

***Keywords*— Computer Vision, SIFT, ORB, KAZE, Feature Matching, Algorithms**

## I. Introduction

In this project, we revisited homework assignment 4 with two main goals. Initially, in assignment 4, we used the Harris detection algorithm for finding interest points and the SIFT descriptor for feature matching. However, there are numerous different detection and descriptor algorithms that exist, such as ORB, KAZE, AKAZE, BRISK, and SIFT's own keypoint detection algorithm. Thus, we explored various different algorithms in order to understand their differences and find out which are the most efficient under different scenarios, considering that each of these detection algorithms has their own approach for finding keypoints. In this project, we opted to use those different algorithms rather than using the Harris detector in order to explore the different results for feature matching.

## II. Description of the Algorithms

### *A. SIFT*

In Computer Vision, corner detection is one of the most significant keys of object detection and Harris corner detection is a popular method for identifying corner points in images. Although its strong points such as robustness to noise and invariance to image rotation, it has its weakness on not being scale invariant.

In 2004, D.Lowe, University of British Columbia, came up with Scale Invariant Feature Transform (SIFT) in his paper, Distinctive Image Features from Scale-Invariant Keypoints, which extract keypoints and compute its descriptors. SIFT is a robust and widely used method in computer vision for detecting and describing distinctive features in images. SIFT addresses the challenge of identifying and matching image features across different scales, rotations, and viewpoints.

One of the key strengths of SIFT is to generate feature descriptors that are highly distinctive and invariant to various transformations. These descriptors encode information about the local image region surrounding each key point, capturing details such as gradient orientations and magnitudes.

SIFT employs a multi-scale approach to handle objects of different sizes within an image. By constructing a scale space pyramid and applying Gaussian blurring at different levels, SIFT can detect features at multiple scales, enabling robustness against changes in object size and viewpoint.

One drawback of SIFT is its computational complexity, especially during the keypoint detection and descriptor extraction phases. SIFT involves intensive operations such as convolutions with Gaussian kernels, gradient calculation, and histogram generation.

### *B. ORB*

ORB(Oriented FAST and Rotated BRIEF) is a combination of FAST(features from accelerated segment test) and BRIEF(binary robust independent elementary features) descriptor algorithms with adding more features to improve the performance. It

was designed to provide a fast and efficient alternative to the SIFT.

ORB first uses the FAST algorithm for identifying keypoints in an image. Then, it applies a multi-scale image pyramid so that makes the detection algorithm scale-invariant. Then, it uses the BRIEF descriptor to find similarities. One interesting characteristic of the BRIEF descriptor is that it converts keypoints to a binary feature vector. However, this algorithm is not rotation-invariant, so ORB uses rBRIEF(rotation-aware BRIEF) to achieve rotation-invariance.

### C. KAZE

KAZE algorithm is a feature detector and descriptor algorithm that uses non linear scale spaces which makes it more robust to real world image factors such as scale and noise. KAZE is different from other algorithms such as Gaussian smoothing since it does not apply linear filtering, which in return allows it to capture fine details and complex textures in the images that it is processing.

The KAZE algorithm first extracts keypoints using a determinant of the Hessian matrix method and then describes them with floating-point vectors that encode gradient magnitudes and orientations. However, because KAZE is superior to other feature descriptors in terms of robustness and fine detail capturing, it is consequently computationally intensive, so you would be sacrificing time in order to achieve much more quality feature detection.

## III. Challenges

Our initial challenge that we faced was that the pkl file from assignment 4 wasn't able to be unpacked, which prevented us from accessing the ground truth locations. In order to get around this issue, we had to adapt our approach by applying the ratio test method instead. Another significant challenge that we faced was that there was not much information about the KAZE algorithm on the internet, which slowed down our progress on understanding and implementing this algorithm as we did not have access to many examples of its usage. Our final challenge was the time it took to carefully understand each of the algorithms and selecting the appropriate OpenCV libraries. We had to dive deeper into the foundations of each algorithm in order to classify it as appropriate to be used for our study.

## IV. Experiments

### A. Dataset

The images we used in this project are the same as homework assignment 4. The dataset folder contains three image sets, including Episcopal Gaudi, Mount Rushmore, and Notre Dame. Each image set contains two images, A and B. They are the same picture of a building or a monument, but they have different scales or are taken from different angles.

### B. Methods

We used the opencv library for implementation of the algorithms. The basic implementations we used in all three algorithms are cv2.__.detect() and cv2.__.compute. cv2.__.detect() function detects interest points in an image, where __ is the name of the algorithms, as well as cv2.__.compute function computes the descriptors of keypoints in an image. For the feature matching algorithm, we used the Brute-Force matcher in the opencv library. By default, it applies euclidean distance when computing matching between two descriptors. However, for binary descriptors such as ORB, the hamming distance is often used. Thus, we implemented the cv2.NORM_HAMMING method for the ORB algorithm. For computing features across two different images, we used Lowe's ratio test. This test checks if the distance to best match is significantly smaller than the distance to the second best match by applying $\frac{distance\ to\ best\ match}{distance\ to\ second\ best\ match} < ratio\ threshold.$ If this statement is true, then we store the results in a list "good_matches". Thus, we can simply score the overall ratio by calculating $\frac{number\ of\ elements\ in\ good\ matches}{number\ of\ elements\ in\ matches}$. We set the default threshold value to be 0.75 in our ratio test. Different implementations in each algorithm are described below.

*1) SIFT*

After the preparation process of the input images, such as loading, converting to grayscale using the OpenCV library, SIFT key points are detected using the 'cv2.SIFT_create()' function. Key points are filtered based on a threshold value to retain only the significant keypoints. To see how the accuracy is affected by this threshold, we changed the "threshold" in the range from 0 to 0.05 by 0.01 gap (0, 0.01, 0.02, 0.03, 0.04, 0.05). The threshold value beyond 0.05 is not considered because the "good match" number of one of the images reached 0m. Among the values, we picked a threshold value that shows significantly high accuracy in any of the images, and more explored keypoint numbers and accuracy at nearby numbers (0.025 and 0.018) and of the picked threshold value.

*2) ORB*

By default, the .create() function for ORB sets the maximum number of keypoints to be 500. We also set this number to 3000 to observe how those number of features affect the matching results. For computing the matching, brute force matching algorithm was used.

*3) KAZE*

Unlike the ORB algorithm, our .create() function did not set a maximum number of keypoints for KAZE, allowing it to observe a non-limited number of keypoints. The keypoints are then filtered based on a response threshold in order to eliminate weaker detections. We also used a brute force matching algorithm for the matching.

*C. Results*

The tables below show the different results we obtained in each algorithm with different threshold values.

*1) SIFT*

| Threshold value | Number of key points filtered (*ratio from previous threshold) | | | | | | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ND | | | EG | | | MR | | |
| | ND_A | ND_B | EG_A | EG_B | MR_A | MR_B | Good match | Match | Ratio | Good match | Match | Ratio | Good match | Match | Ratio |
| 0 | 24023 | 20431 | 73048 | 8948 | 30318 | 30318 | 4005 | 6637 | 60.34 | 649 | 4591 | 14.14 | 243 | 4612 | 5.27 |
| 0.01 | 24023 (1) | 20431 (1) | 73048 (1) | 8948 (1) | 30318 (1) | 30318 (1) | 4005 | 6637 | 60.34 | 649 | 4591 | 14.14 | 243 | 4612 | 5.27 |
| 0.02 | 15336 (.638) | 14856 (.727) | 59780 (.818) | 4910 (.549) | 19409 (.64) | 19409 (.64) | 3841 | 4622 | 83.10 | 422 | 2749 | 15.35 | 8 | 2969 | 0.27 |
| 0.03 | 7064 (.461) | 7845 (.528) | 41131 (.688) | 1807 (.368) | 9425 (.486) | 9425 (.486) | 443 | 2315 | 19.14 | 269 | 1125 | 23.01 | 8 | 1501 | 0.53 |
| 0.04 | 3007 (.426) | 3658 (.466) | 26378 (.641) | 669 (.37) | 4348 (.461) | 4348 (.461) | 180 | 1036 | 17.37 | 78 | 476 | 16.39 | 1 | 713 | 0.14 |
| 0.05 | 1076 (.358) | 1733 (.474) | 15503 (.588) | 220 (.329) | 1702 (.391) | 4348 (.391) | 32 | 445 | 7.19 | 17 | 178 | 9.55 | 0 | 282 | 0.0 |

| Threshold value | Number of key points filtered (*ratio from previous threshold) | | | | | | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ND | | | EG | | | MR | | |
| | ND_A | ND_B | EG_A | EG_B | MR_A | MR_B | Good match | Match | Ratio | Good match | Match | Ratio | Good match | Match | Ratio |
| 0.025 | 10566 | 11021 | 49990 | 3011 | 13588 | 13588 | 1145 | 3321 | 34.48 | 261 | 1763 | 14.8 | 19 | 2082 | 0.912 |
| 0.018 | 17628 | 16507 | 63709 | 5979 | 22256 | 22256 | 3017 | 5210 | 57.91 | 493 | 3238 | 15.23 | 26 | 3377 | 0.77 |

Fig. 1. result table for SIFT

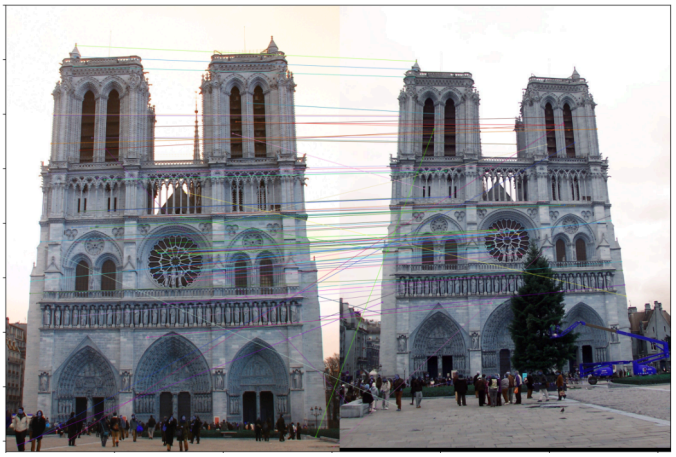

Fig. 2. result image of Notre Dame with threshold = 0.02

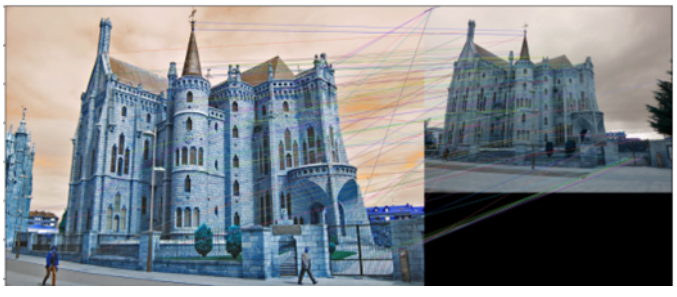

Fig. 3. result image of Episcopal Gaudi with threshold = 0.03

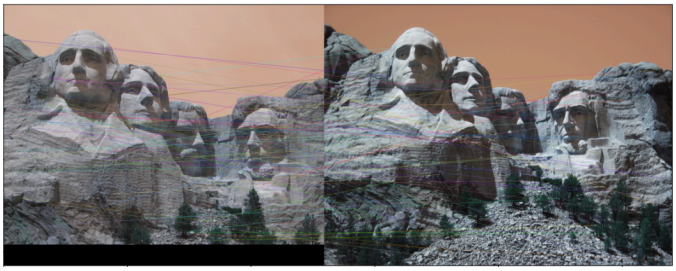

Fig. 4. result image of Mount Rushmore with threshold = 0.

*2) ORB*

number of features = 500

| Threshold value | Number of key points filtered | | | | | | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ND | | | EG | | | MR | | |
| | ND_A | ND_B | EG_A | EG_B | MR_A | MR_B | good match | match | ratio | good match | match | ratio | good match | match | Ratio |
| 0 | 500 | 500 | 500 | 500 | 500 | 500 | 6 | 143 | 4.20 | 35 | 113 | 30.98 | 0 | 112 | 0 |
| 0.001 | 446 | 482 | 500 | 171 | 236 | 236 | 6 | 137 | 4.38 | 12 | 62 | 19.35 | 0 | 46 | 0 |
| 0.0001 | 500 | 500 | 500 | 500 | 500 | 500 | 6 | 143 | 4.20 | 35 | 113 | 30.98 | 0 | 112 | 0 |
| 0.0005 | 500 | 500 | 500 | 407 | 500 | 500 | 6 | 143 | 4.20 | 30 | 101 | 29.7 | 0 | 112 | 0 |

Fig. 5. result table for ORB with 500 features

Number of features = 3000

| Threshold value | Number of key points filtered | | | | | | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ND | | | EG | | | MR | | |
| | ND_A | ND_B | EG_A | EG_B | MR_A | MR_B | good match | match | ratio | good match | match | ratio | good match | match | Ratio |
| 0 | 3000 | 3000 | 3000 | 3000 | 3000 | 3000 | 342 | 928 | 36.86 | 217 | 718 | 30.22 | 34 | 513 | 6.628 |
| 0.001 | 1312 | 2095 | 2992 | 239 | 245 | 377 | 97 | 479 | 20.64 | 20 | 125 | 16.0 | 2 | 84 | 2.38 |
| 0.0001 | 3000 | 3000 | 3000 | 2726 | 3000 | 3000 | 342 | 928 | 36.86 | 289 | 675 | 42.81 | 34 | 513 | 6.628 |
| 0.0005 | 2525 | 2810 | 3000 | 696 | 2030 | 2030 | 268 | 817 | 32.8 | 131 | 279 | 47.0 | 4 | 375 | 1.07 |

Fig. 6. result table for ORB with 3000 features

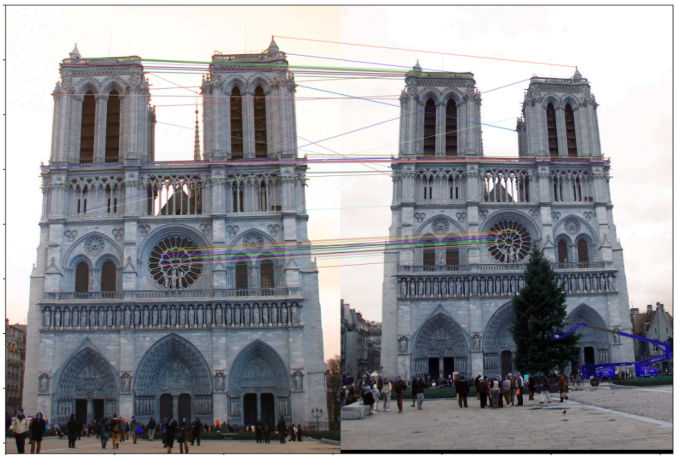

Fig. 7. result image of Notre Dame with threshold value = 0 in Fig. 6.

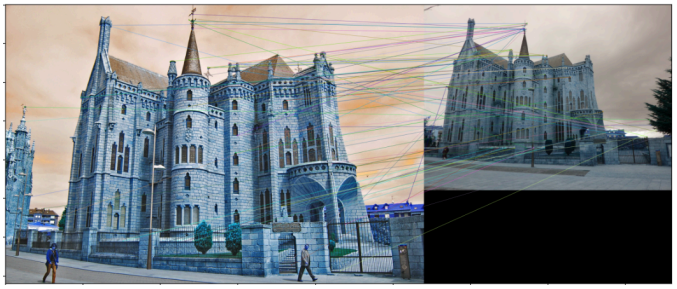

Fig. 8. result image of Episcopal Gaudi with threshold value = 0.0005 in Fig. 6.

Fig. 9. result image of Mount Rushmore with threshold value = 0 in Fig. 6.

*3) KAZE*

| Threshold value | Number of key points filtered | | | | | | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ND | | | EG | | | MR | | |
| | ND_A | ND_B | EG_A | EG_B | MR_A | MR_B | good match | match | ratio | good match | match | ratio | good match | match | Ratio |
| 0.002 | 4623 | 5372 | 23216 | 1825 | 14360 | 14360 | 157 | 3163 | 4.96 | 436 | 1620 | 26.9 | 0 | 0 | 0 |
| 0.005 | 1274 | 1691 | 7636 | 467 | 4484 | 4484 | 219 | 434 | 50.46 | 23 | 233 | 12.44 | 0 | 0 | 0 |
| 0.009 | 424 | 640 | 2818 | 153 | 1439 | 1439 | 5 | 157 | 3.18 | 43 | 81 | 53 | 33 | 117 | 28 |
| 0.01 | 338 | 499 | 2236 | 116 | 1119 | 1119 | 59 | 117 | 50.4 | 13 | 60 | 21.6 | 26 | 88 | 29.5 |

Fig. 10. result table for KAZE

Fig. 11. result image of Notre Dame with threshold = 0.005

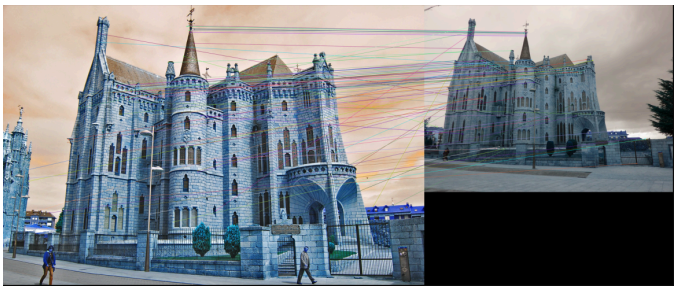

Fig. 12. result image of Episcopal Gaudi with threshold value = 0.002

Fig. 13. result image of Mount Rushmore with threshold value = 0.01

*D. Analysis*

*1) SIFT*

We did not have a restricted number of features for SIFT. At the threshold of 0.03, SIFT performed best for the Episcopal Gaudi image which suggests that a moderate amount of detail was necessary in order to capture the unique details of the building's design. At the threshold of 0.02, SIFT performed best for the Notre Dame image which indicated that this threshold was optimal for detecting significant features along with the architectural details. At the very low threshold of 0/0.01, SIFT performed best for the Mount Rushmore image which was very interesting to us and helped us conclude that this result was likely due to the less intricate and more uniform surface of the mountain in comparison to Notre Dame and the Episcopal Gaudi.

While the threshold increased by 0.01, the number of keypoints filtered decreased by a number in the range from 0.329 to 0.818 of its previous filtered keypoint sizes.

However, besides the reduction of the number of keypoints, we did not see any clear pattern in changes of the number of keypoints or in accuracy along the threshold changes. Also, the intensities of changes are diverse among images.

Although we looked into the accuracy change with nearby numbers (0.018 and 0.025) that are close to the threshold (0.02) that yielded maximum accuracy, we could not find any valid relation between threshold and the accuracy.

*2) ORB*

First of all, we concluded that the default number of features being 500 is too small to find the matching results. It seems that Episcopal Gaudi worked the best with this algorithm, especially with threshold value 0.0005 we got almost 50% accuracy. This proves that ORB is indeed scale-invariant as the second image of Episcopal Gaudi being much smaller than the first image. This result was interesting because the number of filtered keypoints for the second image of Episcopal Gaudi was only 696. This indicates that the majority of keypoints in there were accurate. The results for Notre Dame were acceptable but we were not able to get a single good score on Mount Rushmore. In addition, one thing we noticed

was that ORB runs noticeably faster than the other two algorithms. Despite the number of features being set 10,000 it still ran very fast.

Google colab link:
SIFT.ipynb
ORB.ipynb
KAZE.ipynb

3) *KAZE*

Similarly to SIFT, we did not have a restricted number of features for KAZE, but the results were still quite interesting to us. At the threshold of 0.002, KAZE performed best for the Episcopal Gaudi image which suggests that the finer details were important in this scenario. At the threshold of 0.005, KAZE performed best for Notre Dame which indicated a balance between detail and noise reduction. At the threshold of 0.01, KAZE performed best for Mount Rushmore which shows KAZE's robustness against the larger scale variations of this particular image.

## V. CONCLUSION

Overall, each algorithm worked the best on different image sets. SIFT with threshold value 0.02 scored above 80% accuracy on Notre Dame. This suggests that the SIFT corner detection algorithm works better than the Harris algorithm on this image set since none of us passed 60% accuracy in homework 4 with the combination of the Harris and SIFT. For Episcopal Gaudi, we conclude that both ORB and KAZE were suitable algorithms for corner detection and matching. Those algorithms both score about 50% accuracy. KAZE also worked better on Mount Rushmore and it is the only algorithm among those algorithms that passed the accuracy above 20%. Neither SIFT nor ORB passed 10% while KAZE got about 30% with threshold value 0.01. However, since each algorithm has different strengths and weaknesses and we applied those algorithms over three different image sets, we can not simply conclude which algorithm is the best.

## REFERENCES

[1] "Introduction to SIFT (Scale-Invariant Feature Transform)," https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html (accessed May. 3, 2024)