ORB Feature Detection and Matching

CSCE 748: Computational Photography Final Project

Team Members:

1. Aditya Thagarthi Arun (731009189)
2. Ganeshprasad Rajashekar Biradar (931002164)
3. Vishruth Raghuraj Gollahalli (730007888)

Introduction

For this project, we have implemented the paper, ORB: An Efficient alternative to SIFT or SURF, by Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski[1]. As with any traditional feature detection/extraction methods, ORB uses a feature detection component called FAST and a descriptor generator component called BRIEF. As proposed in the original paper, we have incorporated an orientation component to FAST (Features from Accelerated Segment Test) feature extraction to incorporate rotational invariance for image matching. We have also implemented our own version of the OpenCV BRIEF (Binary Robust Independent Elementary Features) descriptor generator algorithm. In the subsequent section, we will discuss our methodology of implementing the ORB feature matching and description generation.

Methodology

Our feature matching algorithm mainly consists of the following parts:

- FAST key points generator
- BRIEF descriptor generator
- Matching features

a. FAST key points generator

The FAST algorithm is used to detect keypoints from which descriptors can be generated. Here keypoints means “interesting pixels”, which basically are pixels that can be easily identified across different images due to their nature (for e.g they may describe a corner, which is usually more identifiable compared to a constant color region).

The main steps involved in this algorithm are:

1. For the pixel under consideration, select neighboring pixels which lie in a bresenham’s circle (figure 1).
2. For a circle of radius 3, we will get 16 neighboring pixels as shown in the figure.
3. For the current pixel to be considered as a keypoint, it must be either greater or lesser (in intensity, i.e the pixel value) than 12 contiguous pixels in the circle. We usually

add/subtract a threshold value to/from the intensity of the current pixel to make this process more robust.

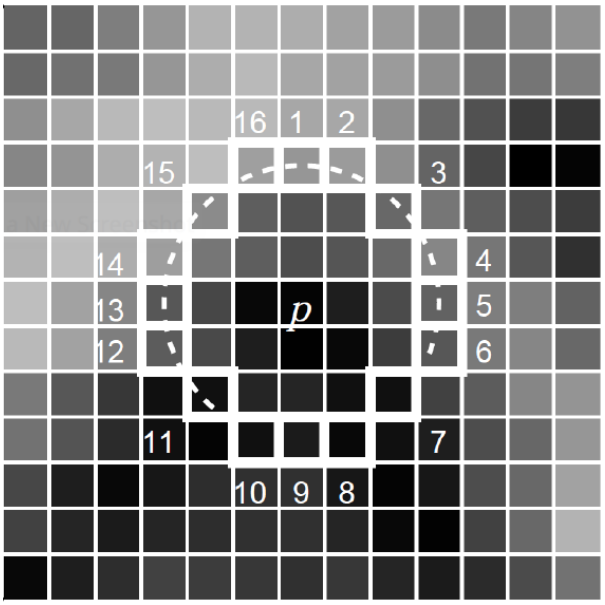


Figure 1: Bresenham's circle[2]

The parameters such as circle radius and required contiguous pixels are modifiable. We have used the ones which worked best for our images.

To reduce the number of keypoints and get only good keypoints, we have used non-maximal suppression (NMS). It works in the following way:

1. Take a parameter NMS window (say a).
2. We filter the key points such that in a square window of size axa around any keypoint, only the keypoint with maximum score is kept and all others are discarded. The "score" is calculated by:
    a. The sum of absolute values of difference of current pixel intensity and neighboring pixel intensities:

$$V = \max \begin{cases} \sum(\text{pixel values} - p) \text{ if } (\text{value} - p) > t \\ \sum(p - \text{pixel values}) \text{ if } (p - \text{value}) > t \end{cases}$$

Equation 1: p is the current pixel, pixel values are neighbors[2].

b. BRIEF descriptor generator[3]

A BREIF descriptor is a string of 0's and 1's which tries to describe a keypoint properly. This can then be used to match keypoints across different images. This string is generated for each keypoint in the following way:

1. Select a square patch around the keypoint
2. Initialize an empty string for this keypoint
3. Select 2 random pixels from this patch:
    a. This can be done in different ways for e.g using a gaussian or uniform distribution. We have used uniform distribution.
4. Apply rotation to the selected pixels to find new pixels corresponding to rotation (more about rotation below)
5. Append a 1 to the string if intensity of first pixel < intensity of second pixel otherwise append a 0.
6. Repeat steps 2-5 required number of times (for e.g 256 times to construct a string of length 256)

In implementation this is not followed exactly as described to make it more efficient (for e.g using a boolean matrix instead of building the strings). But it follows the same logic as described above.

i. Incorporating Scale Invariance

To make keypoint matching robust to scale changes, we construct the keypoints and their descriptors across a gaussian pyramid of the image.

ii. Incorporating Rotational Invariance

The regular FAST key points generation is not inherently invariant to rotation of images. To incorporate this, we have followed the method proposed by the authors in the paper and utilized the orientation by centroid intensity method.

This method is used to estimate the orientation or direction of an object or feature within an image based on the intensity values of its pixels and the centroid of the object or feature. The centroid is the center of mass or the average location of the pixels in the object or feature.

To calculate the orientations, we had to first calculate the moments of each patch in the image. This was done using Rosin's moment of a patch using the equation:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

Equation 2: m is the moment, I is the intensity of the pixel at (x,y)[1].

Using this, we calculated m_{10} (moment along x axis) and m_{01} (moment along y axis). The moments were calculated for values of x and y within a patch of radius r. For this, we created a patch of a radius R as a mask and computed moments only for (x,y) values within the mask.

The orientation of patch, theta was then finally calculated using the equation:

$$\theta = \arctan2(m_{01}, m_{10})$$

Equation 3: arctan2 is the quadrant-aware version of the arctan function[1].

c. Matching features

We now find the best matches between two sets of descriptors from the two images respectively. We first calculate the Hamming distance(from descriptor 1 to descriptor 2) between feature descriptors. The pair of descriptors that have the least distance between are deemed best preliminary matches. Additionally we used two methods to improve the accuracy of the matches, these methodologies are discussed below.

i. Cross Check[4]:

We now compute the reverse Hamming distances (from descriptor 2 to descriptor 1), and then find the reverse matches by matching a pair of descriptors that have the least distance between them. We consider a match to be valid only when the same pair of descriptors is obtained in both directions, meaning that each descriptor is the best match for the other. If a match does not satisfy this criterion, it is discarded.

ii. Distance Ratio:

We compute the distance ratios (taught in class) between the best and second-best matches for each descriptor in the first descriptor, and filter out any matches whose ratio exceeds a given distance ratio (default set to 0.7). This allows us to eliminate any ambiguous matches.

Results

We now present our findings after running the script on a set of images.

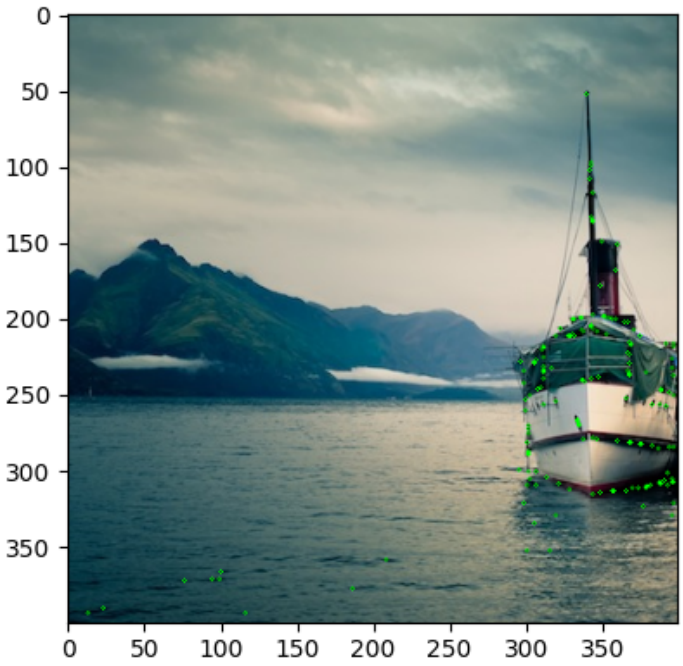
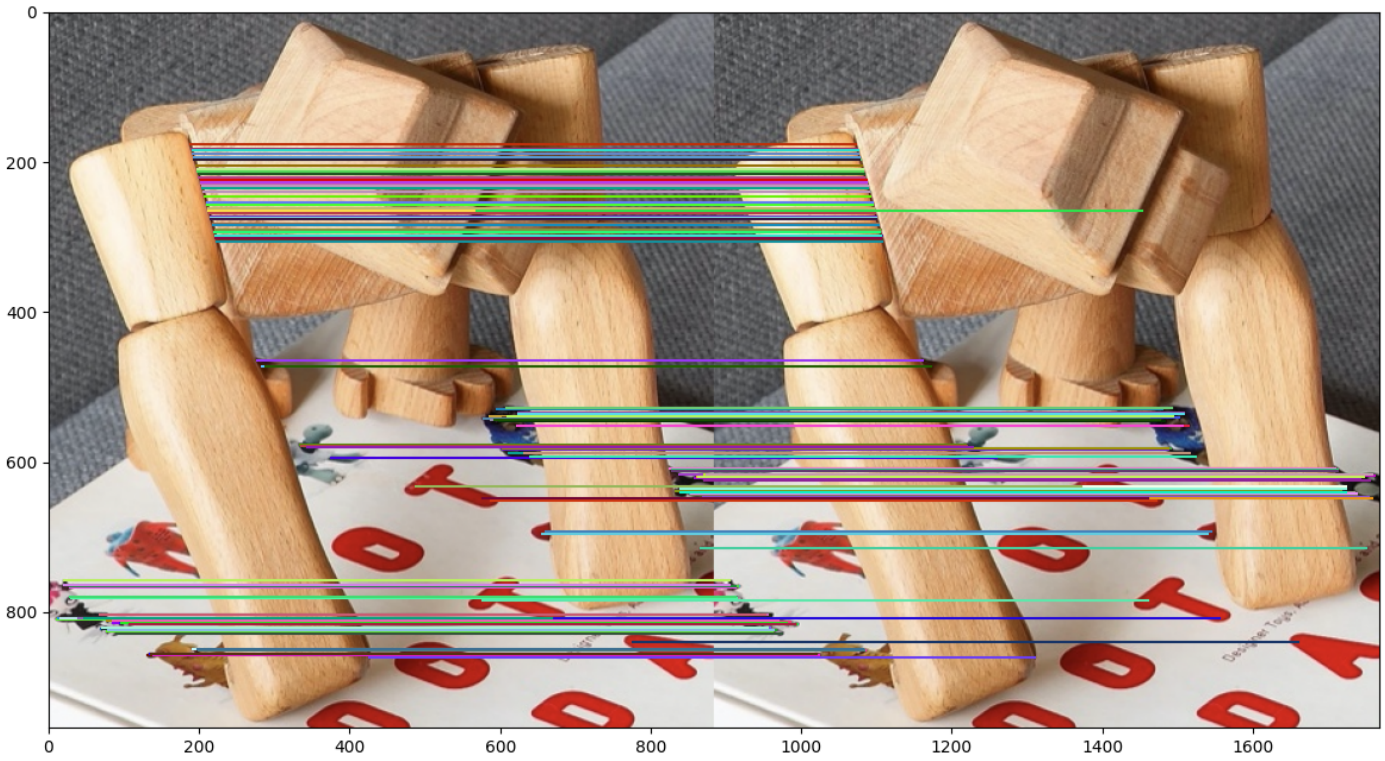


Figure 2: Detecting keypoints

In figure 2, we can see that the detected keypoints circled in green. The next example showcases the feature matching for identical images.



.Figure 3: Feature detection for identical images

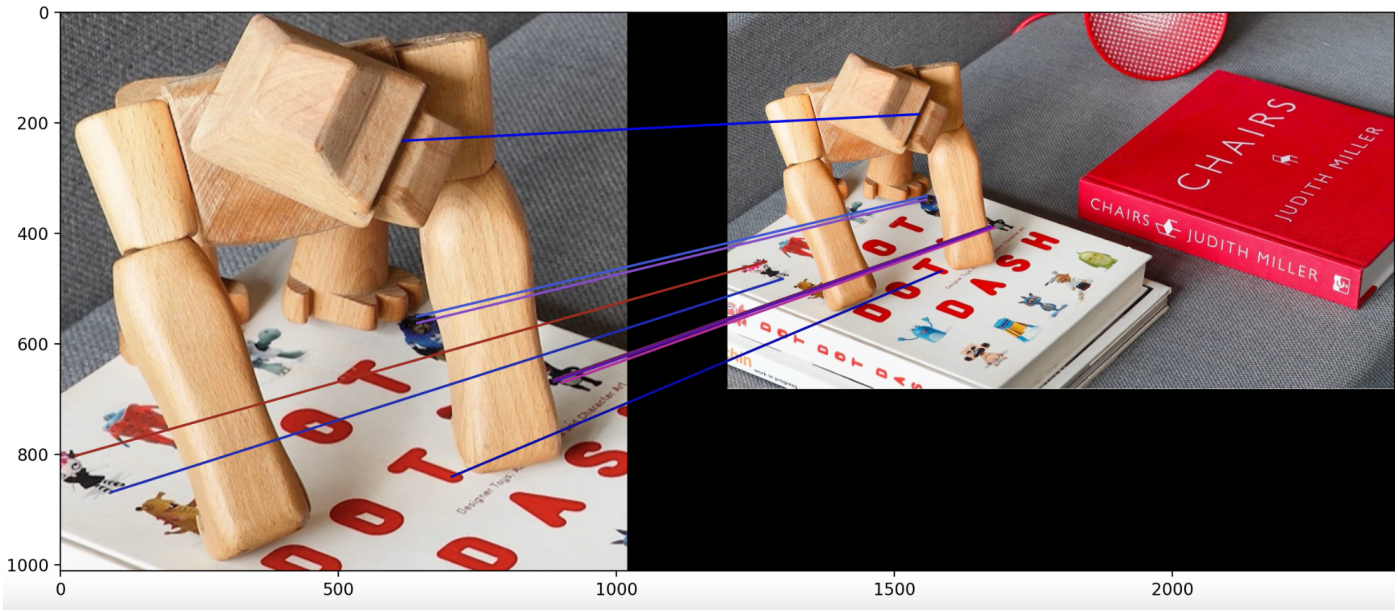


Figure 4: Feature detection and matching with scale invariance

In figure 4, we can see that there are multiple points being matched from the object on the right to the object on the smaller-scaled image on the right. To remove multiple ambiguous matches, the distance ratio used for this example was 0.4.

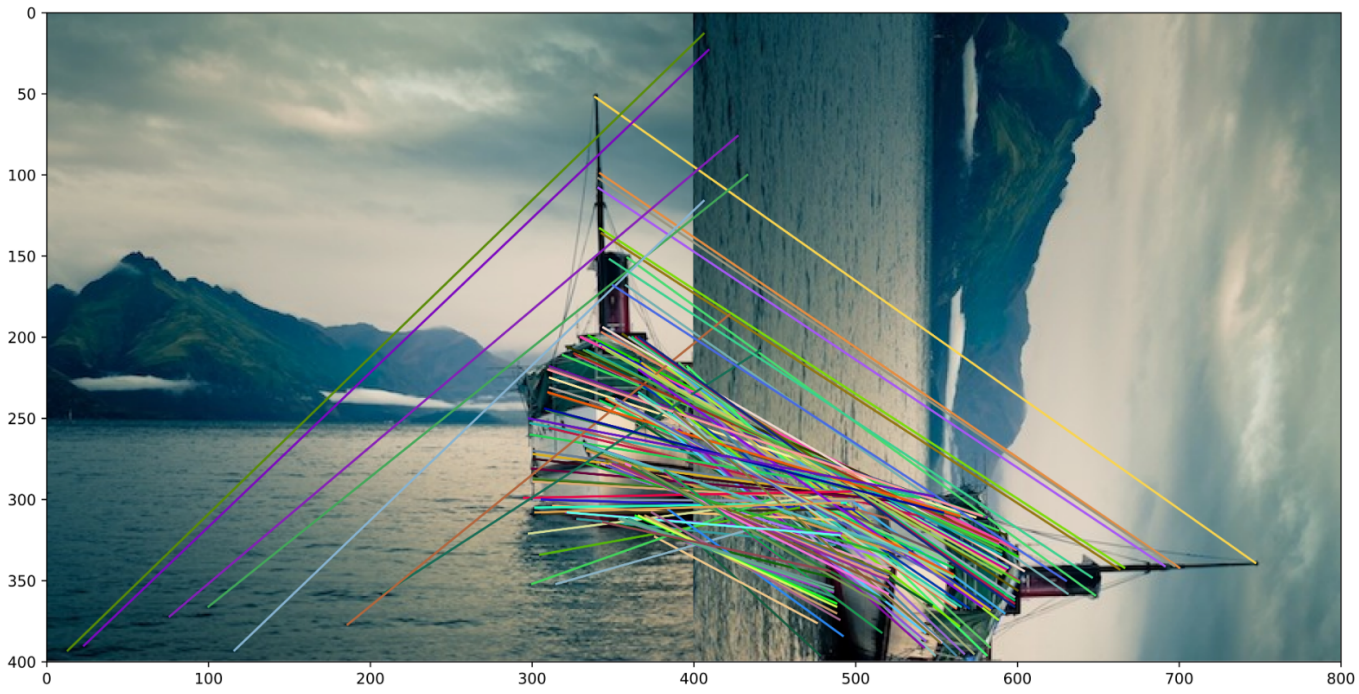


Figure 5: Feature detection and matching with rotation invariance

Figure 6: Feature detection and matching with rotation invariance(with second image cropped)

In figure 5, we can see that the algorithm is able to match features even though the second image is rotated. Figure 6 is also an example of the rotation invariant feature but the first image is matched with a cropped version that is rotated. Figure 7 seen below, showcases the robustness of the algorithm. The first image is matched with a scaled down (0.5X) and rotated version of itself.

Conclusion

From the above results, we observe that ORB feature detection and image matching algorithm does a pretty good job on identifying features, descriptors and matching. Unlike the SIFT and SURF algorithms which are patented algorithms, ORB, FAST and BRIEF are open source algorithms that are computationally more efficient than the former group of algorithms owing to fewer filtering necessary for FAST, BRIEF and ORB. Although FAST and BRIEF are not size and rotation invariant, we have shown that with some modifications to the algorithm, size and scale invariance can be built into these algorithms.

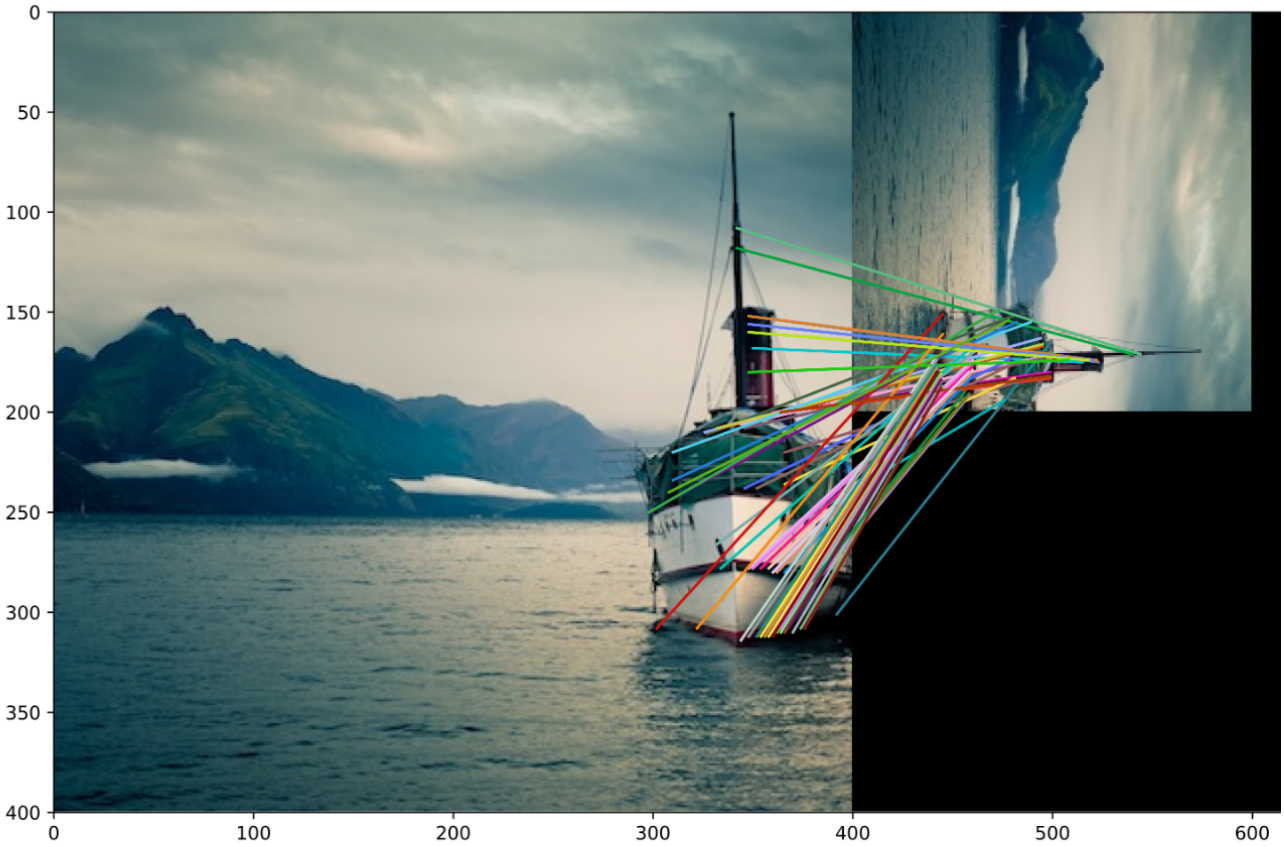


Figure 7: Feature detection and matching with scale and rotation invariance

Future works

As future work, this project would require outlier detection to be incorporated into the algorithm. At the moment, outlier points are not detected. Further, there might be some more information that may be gained about the performance of the ORB algorithm if a measure of accuracy is developed. Having this would greatly help evaluate the efficiency of not only the matching but also identifying the performance of the FAST feature detection as well as the BRIEF descriptor generator algorithms too.

References

1. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 2564-2571.
2. Viswanathan, Deepak. “Features from Accelerated Segment Test (FAST).” (2011).
3. Calonder, M., Lepetit, V., Strecha, C., Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15561-1_56
4. Ong, D. S. (2018). OpenCV 4 computer vision with Python recipes: master advanced techniques in OpenCV with Python. O'Reilly Media, Inc. Retrieved from https://www.oreilly.com/library/view/opencv-4-computer/9781789340723/04b5045a-c8c3-4d51-a265-9d4307f2b11c.xhtml