Fast Matching of Image Targets based on ORB Feature Points

Xingxing Li[a], Chao Duan, Panpan Yin, Yan Zhi

Department of electronics and information engineering, Guangzhou College of Technology and Business, FoShan, 528138, China.

[a]lixingxing_Edooy@163.com

Keywords: ORB, FAST feature point, BRIEF feature descriptor.

Abstract: ORB is the abbreviation of Oriented Fast and Rotated Brief. It can be used to quickly create feature vectors for key points in the image. These feature vectors can be used to identify objects in the image. The ORB feature is a combination of the FAST feature point detection method and the BRIEF feature descriptor, and has been improved and optimized based on their original, which greatly reduces the matching time.

1. Introduction

ORB (Oriented FAST and Rotated BRIEF) [1] is a fast feature point extraction and description algorithm. The ORB algorithm is divided into two parts, namely feature point extraction and feature point description. Feature extraction is developed by the FAST (Features from Accelerated Segment Test) algorithm [2-3]. The feature point description is improved based on the BRIEF (Binary Robust Independent Elementary Features) feature description algorithm. The ORB feature is a combination of the FAST feature point detection method and the BRIEF feature descriptor, and is improved and optimized based on their original features. The biggest feature of the ORB algorithm is its fast calculation speed. This is primarily due to the use of FAST to detect feature points. FAST's detection speed is as famous as its name. Once again, the descriptor algorithm is used to calculate the descriptor. The descriptor's unique binary string representation not only saves storage space, but also greatly shortens the matching time [4].

Among them, Fast and Brief are feature detection algorithms and vector creation algorithms, respectively. The ORB first looks for special areas in the image, called keypoint [5]. Key points are small areas that stand out in the image, such as corner points. For example, they have the feature of sharply changing pixel values from light to dark. The ORB then calculates the corresponding feature vector for each keypoint. The feature vector created by the ORB algorithm contains only 1 and 0, which is called a binary feature vector. The order of 1 and 0 varies depending on the specific keypoint and the pixel area around it. This vector represents the intensity pattern around the keypoint, so multiple feature vectors can be used to identify larger areas and even specific objects in the image.

ORB is characterized by super fast speed, and is not affected by noise and image transformation to a certain extent, such as rotation and scaling transformation.

2. Algorithm

2.1 FAST algorithm

The first step of ORB feature detection is to find the key points in the image, and the key point detection algorithm uses the FAST algorithm. FAST is the abbreviation of Features from Accelerated Segments Test, which can quickly select key points. The algorithm steps are as follows. Given a pixel point *p*, FAST compares 16 pixels in the circle range of the target *p*. Each pixel is divided into three categories according to whether it is higher than *p*, less than *p*, or similar to *p*. As follow in Figure 1.

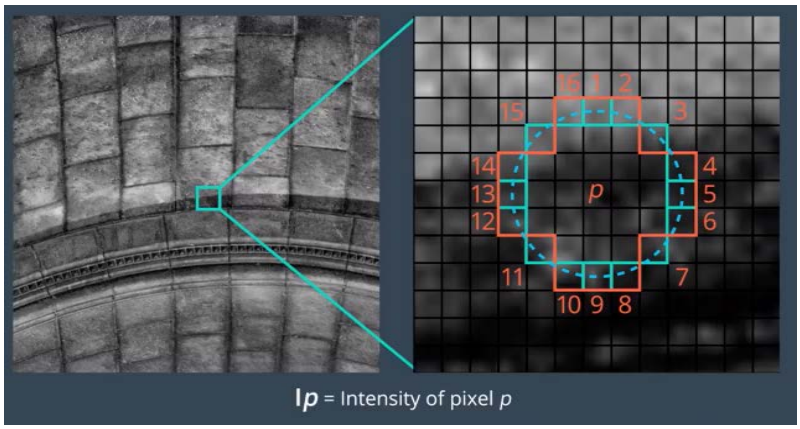

Figure 1. The relationship between the target and surrounding pixels in fast.

Note that the comparison here is with a threshold h. For a given threshold h, the brighter pixels will be those whose brightness exceeds I*p*+h, the darker pixels will be those whose brightness is lower than I*p*-h, and similar pixels will be those whose brightness is between these two values. After classifying the pixels, if there are more than 8 connected pixels on the circle, darker or brighter than p, the pixel p is selected as the key point.

The reason why FAST is so efficient is that it only compares p with 4 equidistant pixels in the circle. This method has proven to be the same as comparing 16 surrounding pixels. If the brightness of at least one pair of consecutive pixels is higher or lower than p, p is selected as the key point. This optimization cuts the search time for key points in the entire image by four times. As follow in Figure 2.

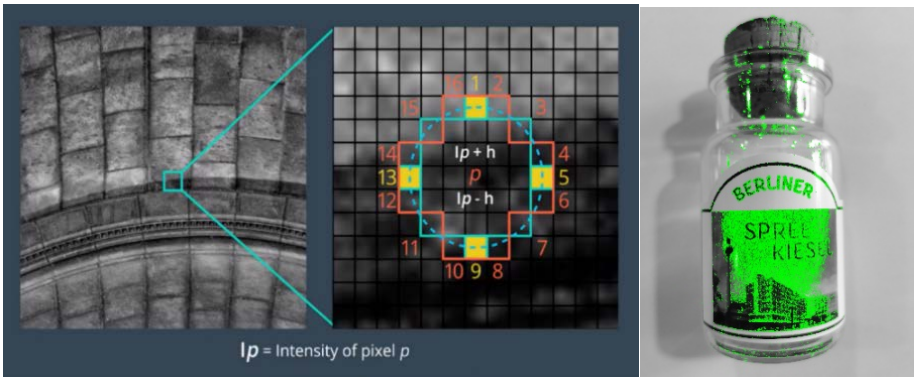

Figure 2. (left)Location of key points in FAST. (right) Image of key points marked by FAST.

It can be seen that the key points are located in areas where the brightness changes. Such areas usually determine some kind of edge, such as the edge of a human bottle. The edges define the boundaries of the bottle, as well as the boundaries of the bottle area, so these key points allow us to identify this bottle, not any other object or background in the image.

2.2 BRIEF algorithm

Already know how the ORB uses FAST to determine the key points in the image. Now understand how the OR uses the BRIEF algorithm and converts these key points into feature vectors.

The second step of the ORB algorithm is to turn the key points found by the first algorithm into feature vectors, which can represent an object together.

To create a feature vector, the ORB uses the BRIEF algorithm. Brief is short for Binary Robust Independent Elementary Features, and its role is to create a binary feature vector based on a set of key points. As you saw in the intro video, binary feature vectors, also called binary descriptors, are feature vectors containing only 1s and 0s. Each key point in the BRIEF is described by a binary feature vector, which is generally a 128-512-bit string, which contains only 1s and 0s. Note that "bit"

here is short for binary bit. 1 bit can only store one binary value, either 1 or 0. A bit string is a set of bits. Here is an example of a bit string:

1-bit string: 0

2-bit string: 01

3-bit string: 010

4-bit string: 0111

The first is a 1-bit string, so only 1 bit is stored. The second is a 2-bit string, so 2 binary bits can be stored. In this example it stores 0 and 1. Similarly, the third is a 3-bit string, so 3 bits can be stored, and so on. Computers run binary or machine code, so a big advantage of using binary feature vectors is that they can be stored in memory very efficiently and can be calculated quickly. Speed is critical for real-time applications. These features not only make BRIEF very fast, but also enable BRIEF to run on devices with limited computing resources, such as smartphones.

How does BRIEF create these binary descriptors for each point? The BRIEF algorithm first uses a Gaussian kernel to smooth a given image to prevent descriptors from being too sensitive to high-frequency noise. Then, for a given key point, such as this point on a bottle. As follow in Figure 3.

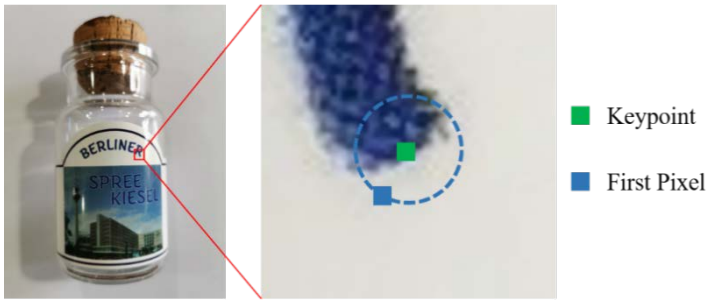

Figure 3. Bottle patch example diagram.

BRIEF randomly selects a pair of pixels from the defined neighborhood around the keypoint. The neighborhood around the keypoint is called a patch, which is a square with a specific pixel width and height.

As shown in Figure 4, the first pixel in the random pair shown here is a blue square, which is a pixel extracted from a Gaussian distribution centered at a key point, with a standard deviation or a dispersion trend of σ.

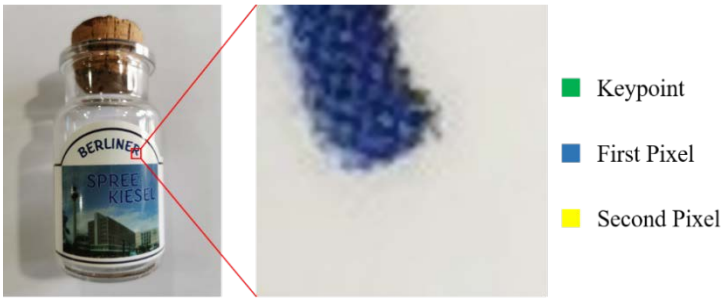

Figure 4. Show different pixels in a patch.

The pixel shown here as a yellow square is the second pixel in a random pair. It is a pixel extracted from a Gaussian distribution centered on the first pixel, and the standard deviation is σ / 2. Experience shows that this Gaussian selection improves the feature matching rate.

BRIEF then starts building a binary descriptor for the keypoint by comparing the brightness of the two pixels as shown below. If the first pixel is brighter than the second, a value of 1 is assigned to the corresponding bit in the descriptor, otherwise a value of 0 is assigned.

The second pixel is brighter than the first in this example, so assign a value of 0 to the first bit of the feature vector. The first bit of the feature vector corresponds to the first random point pair of this keypoint, then BRIEF will select a new random pixel pair for the same keypoint to compare their brightness and assign 1 or 0. As shown in Figure 5.

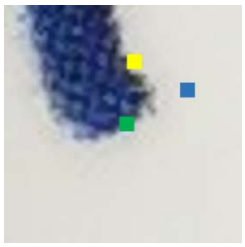

Figure 5. Pixels at different positions in the patch.

In the newly selected random pixels above, review that the first pixel is now brighter than the second, so the second bit in the feature vector is assigned a value of 1.

For 256-bit vectors, BRIEF repeats this process 256 times for the same keypoint, and then moves on to the next keypoint. Then the 256 pixel brightness comparison result is put into the binary feature vector of the key point. BRIEF creates a vector for each keypoint in the image like this.

2.3 Scale invariance and rotation invariance

ORB uses FAST to detect key points in the image, and through several additional steps to ensure that objects in the image can be detected regardless of the size or position of the object.

Given an image ORB algorithm, the image pyramid is first started [4]. An image pyramid is a multi-scale representation of a single image, consisting of a series of different resolution versions of the original image. Each level of the pyramid consists of a downsampled version of the image from the previous level. Downsampling means that the resolution of the image is reduced. As shown in Figure 6, for example, the image is downsampled by 1/2 scale. So the original 4x4 square area is now a 2x2 square. The downsampling of the image contains fewer pixels and is reduced in size by a factor of 1/2.

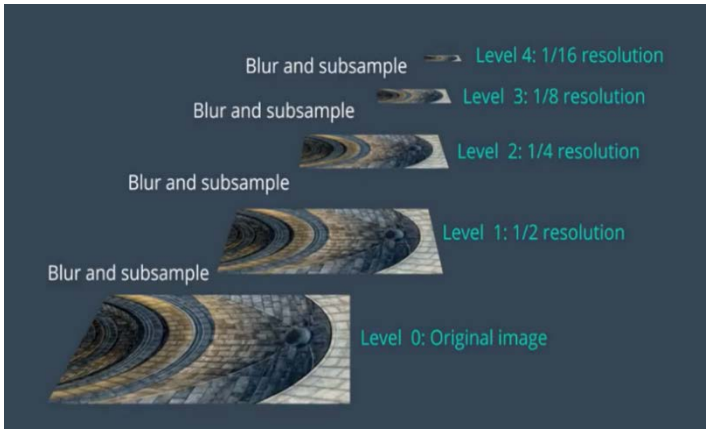

Figure 6. Graphical pyramid with 5 levels.

This is an example of a graphic pyramid with 5 levels, with each level of the image downsampled by 1/2. At the fourth level, the resolution is 1/16 of the original image. After the ORB has created the image pyramid, it will use the FAST algorithm to quickly find key points from images of different sizes at each level. Because each level of the pyramid is composed of a smaller version of the original image, any object in the original image is also reduced in size at each level of the pyramid.

By determining the keypoint of each level, the ORB can effectively find the keypoint of objects of different sizes, so that the ORB achieves partial scaling invariance. This is important because objects are unlikely to be exactly the same size in every image, especially objects like someone's bottle may be close to the camera at some point and far away from the camera at another point. As shown in Figure 7.

Figure 7. Level 5 Pyramid of Specific Images.

The ORB now has the key points associated with each level of this image pyramid. After finding key points in all levels of the pyramid, the ORB now assigns a direction to each key point, such as left or right, depending on how the intensity around that key point changes. The ORB first selects the image in the pyramid Level 0, for which the ORB will calculate the direction of the keypoint. The gray centroid method assumes that there is an offset between the gray level of the corner point and the centroid, and this vector can be used to represent a direction. For any feature point p, define the moment of the neighborhood pixels of p as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{1}$$

where $I(x, y)$ is the gray value at point (x, y). Then can get the centroid of the image as:

$$c = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2}$$

Then the angle between the feature point and the center of mass is defined as the direction of the FAST feature point:

$$\theta = arctan(m_{01}, m_{10}) \tag{3}$$

The ORB now has the key points associated with each level of this image pyramid. After finding key points in all levels of the pyramid, the ORB now assigns a direction to each key point, such as left or right, depending on how the intensity around that key point changes. As shown in Figure 8.

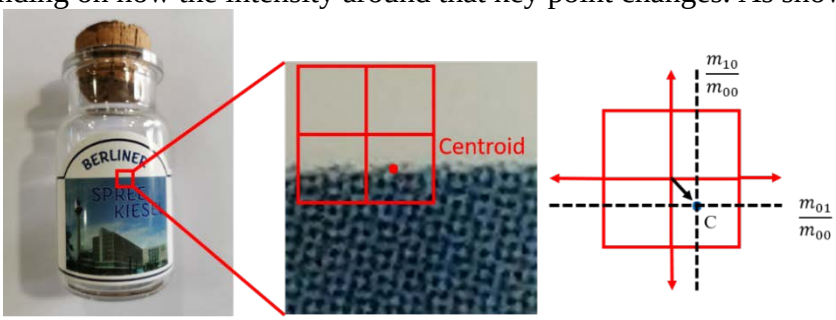

Figure 8. Calculation of key points in a level pyramid.

The method is to first calculate the intensity centroid in the box centered on the key point. The intensity centroid can be seen as the position of the average pixel intensity in a given patch. After calculating the intensity centroid, the direction of the keypoint is obtained by drawing a vector from the keypoint to the intensity centroid, as shown in the figure above. The direction of this key point is down and to the right, because the brightness of this area increases in this direction.

After assigning a direction to each keypoint in the pyramid level 0 image, ORB now repeats the same process for all other pyramid level images. It should be noted that the patch size is not reduced at each image pyramid level, so the image area covered by the same patch at each pyramid level will be larger, resulting in different key point sizes. Circles represent the size of each key point, and key points in higher pyramid levels are larger.

After finding the key points and assigning directions to them, ORB now uses the modified BRIEF version to create feature vectors. This modified BRIEF version is called rBRIEF, or Rotation-Aware BRIEF. Regardless of the orientation of the object, it can create the same vector for keypoint, making the ORB algorithm rotationally invariant, meaning that it can detect the same keypoint in an image rotated towards any angle. Like BRIEF, rBRIEF first randomly selects 256 pixel pairs in a defined patch around a given keypoint to construct a 256-bit vector. Then rotate these random pixel pairs according to the direction angle of the key point, so that the direction of the random point is consistent with the key point. Finally, rBRIEF compares the brightness of random pixel pairs and assigns 1 and 0 accordingly to create corresponding feature vectors. All feature vector sets created for all key points in the image are called ORB descriptors.

There are ORB algorithm steps. First construct the scale pyramid, there are n layers in the pyramid. Unlike SIFT, each layer has only one image. The scale of the s layer is $scale_s=Fator^s$, the initial scale of the Fator (the default is 1.2), and the original image is at the 0 layer. Layer s image size:

$$size_s = (H \times \frac{1}{scale}) \times (W \times \frac{1}{scale}) \tag{4}$$

Secondly, use Fast to detect feature points on different scales; calculate the number of feature points n to be extracted according to the formula on each layer, sort by the Fast corner response value on this layer, extract the first $2n$ feature points, and then according to the Harris angle Point response value ranking, take the first n feature points as the feature points of this layer. Then calculate the principal direction of each feature point (centroid method). Finally, rotate the patch of each feature point to the main direction, use the optimal 256 pairs of feature points selected in step 3 above to do a τ test to form a 256-dimensional descriptor, occupying 32 bytes. Use Hamming distance as feature points match.

$$\tau(p;x,y) \coloneqq \begin{cases} 1, & p(x) < p(y) \\ 0, & p(x) \geq p(y) \end{cases} \tag{5}$$

$$f_n(p) \coloneqq \sum_{1 \leq i \leq n} 2^{i-1} \tau(p;x_i,y_i) \, (\mathrm{n} = 256) \tag{6}$$

3. Experiment

Suppose to detect this bottle in other images. For example, in this group photo, the first image as the query image, and the second image, the image to be bottle searched. For searching images. As shown in Figure 9.

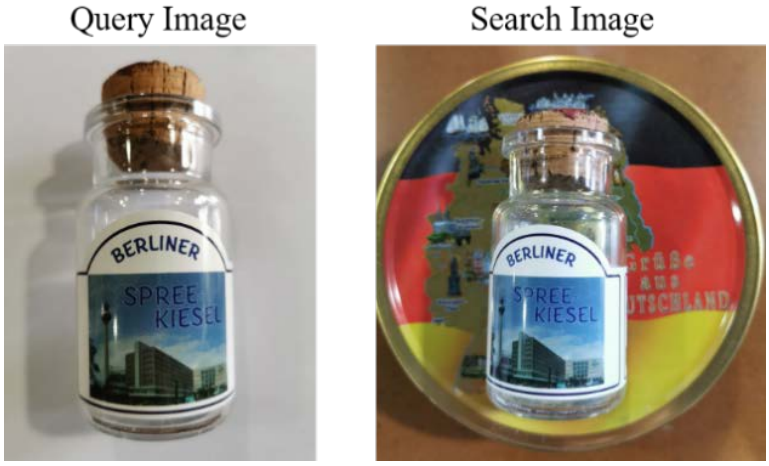

Figure 9. Query image and search image.

Given this query image, and find similar features in this search image. The first step is to calculate the ORB descriptor as figure 10 of the training image and store it in memory.

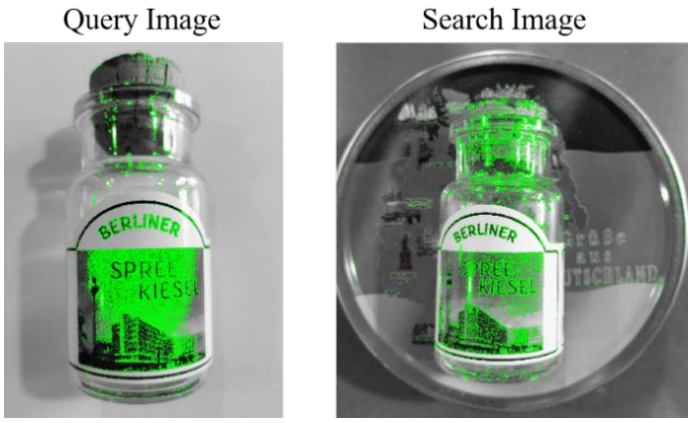

Figure 10. Mark key points for query and search images.

The ORB descriptor will contain a binary feature vector describing the key points in this training image. The second step is to calculate and save the ORB descriptor of the query image. After obtaining the descriptors of the training and query images, the last step is to use the corresponding descriptors to match the key points of the two images. This is usually done using a matching function.

The purpose of the matching function is to match the key points of two different images by comparing the descriptors of the two images to see if they are close enough to match. When the matching function compares two key points, it will obtain the matching quality according to some index, which indicates the similarity of the key point feature vectors. Think of this indicator as the similarity to the Euclidean distance between two key points. Some indicators directly detect whether the feature vector contains similarly ordered ones and zeros. It should be noted that different matching functions use different indicators to judge the quality of the match. For binary descriptors used by ORB, etc., Hamming distance is usually used because it performs very fast.

Hamming distance judges the quality of matching between two key points by calculating the number of different bits between binary descriptors. When comparing the keypoint of a query image and a search image, the keypoint pair with the least difference is considered the best match. After the matching function compares all the keypoint in the query image and the search image, it returns the most matching keypoint pair. There is result of experiment as figure 11.

Figure 11. Result of match.

The best matching point between our training image and query image is shown here, it can be clearly seen that the most matching point between the training image and the query image mainly corresponds to the bottle of the training image. There are one or two features that do not match too well, probably because the intensity patterns of the image regions are relatively similar. Because most points correspond to the bottle in the training image, it can be seen that the matching function can correctly identify the bottle in the query image.

4. Conclusion

The SIFT feature and the SURF feature are very common features. The SIFT feature is known for its accuracy, but the amount of calculation is huge, and it cannot be calculated in real time on the CPU. The SURF feature reduces the accuracy of SIFT, but improves performance. The ORB feature is currently the fastest computing feature and is very suitable for real-time.

Acknowledgments

This work was financially supported by fund project, that is, Guangzhou Institute of industry and commerce college level research project in 2019"Research and design of S bandradio frequency fron t-end" KA201937 and KA201939.

References

[1] Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF[C]. International conference on computer vision, 2011: 2564-2571.

[2] Peng L, Yanjiang W. Multi-target Tracking Algorithm Based on ORB Feature Points Matching [J]. Journal of Hunan University (Natural Sciences), 2017, 44(10):139-149.

[3] Wang G, Zhai Z, Xu B, et al. A parallel method for aerial image stitching using ORB feature points[C] 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). IEEE Computer Society, 2017.

[4] Lile He, Erwei Lu, Zhaoxing Li, et al. Fast stereo matching algorithm based on ORB operator [J]. Computer Engineering & Applications, 2017.

[5] Calonder M, Lepetit V, Fua P, et al. Keypoint Signatures for Fast Learning and Recognition[C]. European conference on computer vision, 2008: 58-71.