# Tutorial for Image Keypoint Matching

# 影像特徵點匹配

**Author: You-Jia Wu**

**Edit by Jian-Jiun Ding**

吳悠嘉 著

丁建均老師編輯

2021.12

# Abstract

Keypoint matching is often applied in image processing. By matching the features of two different pictures, we can perform object detection, image stitching or even motion tracking. There are a variety of applications for keypoint matching. Researchers have been focusing on finding features that are robust to environment change, scaling, rotation or other transforms. Furthermore, effective and efficient generation of features are also important in computer vision.

In this tutorial, we will introduce different ways to detect features and construct their descriptors. These methods and some stimulation results of feature matching will also be presented.

We try to make the tutorial as clear as possible to make the readers can realize the knowledge about keypoint matching.

# Content

# Chapter 1 Introduction

Keypoints, also known as interest points, are the points that contain some distinctive properties within an image. For example, the corners can be view as a type of the keypoints within an image (but may not be the best one). Matching the keypoints of two different images may help us to realize which parts of an image are similar to those of the other image. The number of keypoints is much less than the number of pixels. (For example, a typical image may have more than 1M pixels but have only about 100~1,000 corners). Therefore, using keypoint for template matching is much more efficient than using the original image directly. Thus, in object detection, image stitching, and motion tracking, keypoint matching plays a significant role.

The process of keypoint matching can be briefly described in three steps (*Fig. 1.1*): (i) keypoint detection, (ii) keypoint description, and (iii) keypoint matching.


***Fig. 1.1**: Main steps of keypoint matching.*

In keypoint detection, we localize where the keypoints are. In keypoint description, we want to construct the descriptors of keypoints. As for the final matching, we see whether the features from different images are alike or not. The term "descriptor" means a vector that can describe the properties surrounding the keypoints. With descriptors, keypoint matching can be performed by measuring the similarities of the surrounding properties (e.g., the similarities of colors, intensities, and the distributions of gradients).

In this tutorial, we will first introduce some basic keypoints. Then, the content will be extended to keypoints with non-binary and binary descriptors. Finally, how to perform keypoint matching will be discussed. *Figure 1.2* shows the features that we will introduce in detail in the following chapters.

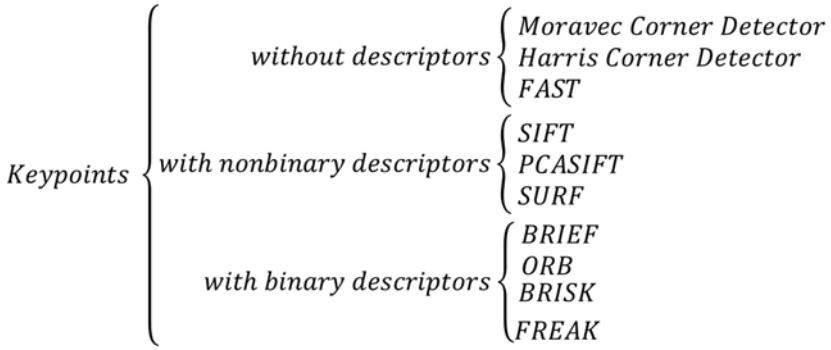


***Fig. 1.2**: Summary of the features that will be introduced in the following chapters.*

# Chapter 2 Fundamental Corner Detectors

In this chapter, three fundamental corner detectors will be introduced. These corner detectors do not compute descriptors for robust matching. However, they can easily extract some distinctive points in pictures. Some of the feature detection methods presented after this chapter are the extensions these three detectors.

We will introduce the corner detectors of (i) Moravec [12], which is one of the earliest corner detectors; (ii) Harris [1], which is the improvement of Moravec; (iii) FAST [2], which is a completely different detector.

## 2.1 Moravec Corner Detector

Moravec introduced a corner detection algorithm in 1980 [12], which is known as one of the earliest methods. He firstly defined a corner as a low self-similarity point and we can easily recognize the point by looking through a small "window".

Shifting the window should give different kinds of change of intensity when it is centered at a flat area, an edge, or a corner. The illustration is shown in *figure 2.1*.

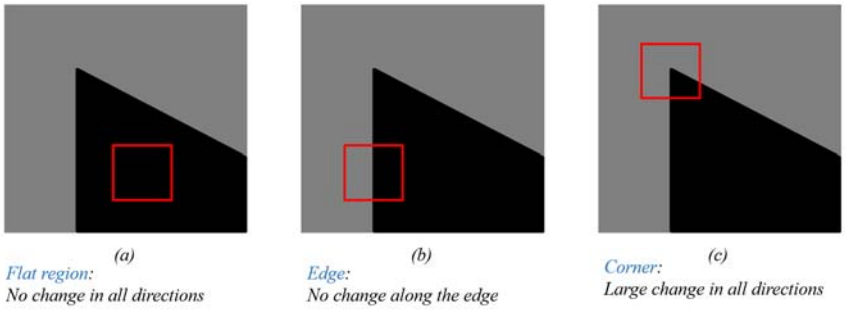


***Fig. 2.1**: Illustration of the windows through a flat region, an edge, and a corner. Look at the black and gray distribution in the red circle, it shows that for different regions, the distributions of intensity are also different.*

The variation of intensity corresponding to a shift $(u, v)$ is defined as:

$$E_{m,n}[u, v] = \sum_{x,y} w(m - x, n - y)[I(x + u, y + v) - I(x, y)]^2, \quad [2\text{-}1]$$

where $w[x, y]$ is a square window function

$$w[x, y] = \begin{cases} \frac{1}{(2L+1)^2} & for - L \le m, n \le L \\ 0 & otherwise \end{cases}, \quad [2\text{-}2]$$

$I[x, y]$ is the intensity and $I[x + u, y + v]$ is the intensity shifted by $[u, v]$ which is considered in four directions: $[u, v] = (1,0), (1,1), (0,1), or\ (-1,1)$.

Then, we determine $min(E_{m,n})$ from

$$min(E_{m,n}) = min(E_{m,n}[1,0], E_{m,n}[1,1], E_{m,n}[0,1], E_{m,n}[-1,1]) \quad [2\text{-}3]$$

The corner pixel will have a larger value of $min(E_{m,n})$. Therefore, by looking for local maxima of $min(E_{m,n})$, we can find the corners. However, Moravec's detector exists some problems:

1. Noisy response due to the binary $w[m,n]$.
2. Only a set of shifts at every 45 degree is considered.
3. Only minimum of E is taken into account which results in too many edges misjudged as corners.

The Harris corner detector is an improvement of Moravec's detector and can solve the problems above.

## 2.2 Harris Corner Detector

In 1988, Chris Harris and Mike Stephens [1] introduced an improvement of Moravec's detector. It is the most popular to extract corners.

First, in order to eliminate the noisy response due to a binary window function, Harris used a Gaussian function as follows instead of a rectangular window.

$$\mathrm{w(x,y)} = e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad [2\text{-}4]$$

Next, consider all possible small shifts by Taylor's expansion instead of shifting the window only at every $45^{\circ}$. $E_{m,n}[u,v]$ is changed as follows:

$$E_{m,n}[u,v] = \sum_{x,y} w(m-x,n-y)[I(x+u,y+v) - I(x,y)]^2$$

$$= \sum_{x,y} w(m-x,n-y)\left[I_x u + I_y v + O(u^2,v^2)\right]^2, \quad [2\text{-}5]$$

where

$$E_{m,n}[u,v] = Au^2 + 2Cuv + Bv^2, \quad [2\text{-}6]$$

$$A = \sum_{x,y} w(x,y) I_x^2(x,y), \quad [2\text{-}7]$$

$$B = \sum_{x,y} w(x,y) I_y^2(x,y), \quad [2\text{-}8]$$

$$C = \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \quad [2\text{-}9]$$

At last, to reduce the misjudgment between an edge and a corner, consider a new corner measurement by investigating the shape of the error function. For small shifts $(u, v)$, apply a bilinear approximation. $E$ can be rewritten as follow:

$$E_{m,n}[u,v] \cong [u\ v]\boldsymbol{M}\begin{bmatrix} u \\ v \end{bmatrix},\ where\ \boldsymbol{M} = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \quad [2\text{-}10]$$

M is a hessian matrix performed on an image patch. Analyze the two eigenvalues of M and the relation between the eigenvalues, $\lambda_1$ and $\lambda_2$, and the kinds of points is shown in *Figure 2.2 (a) & (b)*.

Thus, measurement of corner response *R* is calculated as follow:

$$R = \det \boldsymbol{M} - k(trace\boldsymbol{M})^2, \quad [2\text{-}11]$$

where

$$\det\mathbf{M} = \lambda_1\lambda_2 = AB - C^2, \quad [2\text{-}12]$$

$$\text{trace}\mathbf{M} = \lambda_1 + \lambda_2 = A + B, \quad [2\text{-}13]$$

and *k* is a constant threshold, usually chosen between 0.04~0.06.

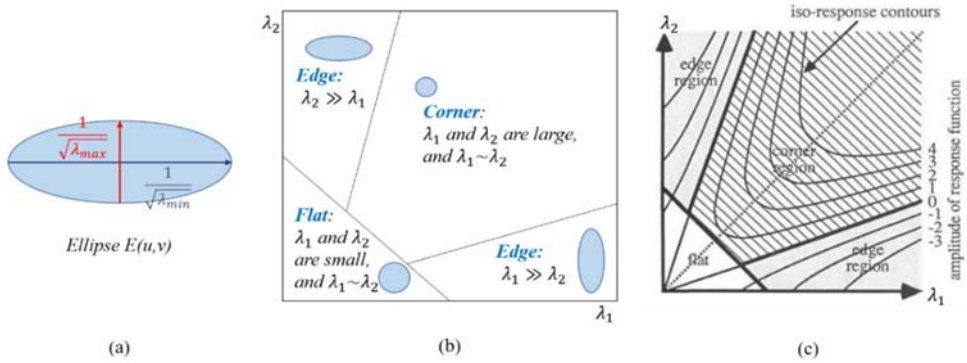


***Fig. 2.2:*** *How eigenvalues represent the type of region. (a) shows the geometrical meaning of two eigenvalues. Then, extend (a) to (b), showing the shape of $E(u,v)$ and the corresponding region. (c) add the contour of the amplitude of response function to (c) [1].*

In *Eqs. (2-6)~(2-9)*, *A*, *B*, *C*, $\lambda_1$, $\lambda_2$, and *R* are all functions of $[m, n]$ (i.e., their values vary for different pixels). The response *R* is illustrated in *Figure 2.2 (c)*. That is:

(i) If both $\lambda_1$ and $\lambda_2$ are small, then it means that $I[m,n]$ varies slowly along all directions. Therefore, $[m, n]$ should be at a flat region.

(ii) If $\lambda_1$ is large but $\lambda_2$ is small (or $\lambda_1$ is small but $\lambda_2$ is large), then it means that $I[m,n]$ varies fast along a direction but varies slowly along another direction. Therefore, $[m, n]$ should be on an edge.

(iii) If both $\lambda_1$ and $\lambda_2$ are large, then it means that $I[m, n]$ varies fast along all directions. Therefore, $[m, n]$ should be at a corner.

We apply the value of $R$ in *Eq. (2-9)* to check whether both $\lambda_1$ and $\lambda_2$ are large. For a pixel $[m, n]$, if

(i) $R[m,n] > R[m+a,n+b] \ \ where \ -1 \leq a,b \leq 1, \ a,b \neq [0,0]$ [2-14]

(ii) $R[m,n] > threshold$ [2-15]

then $[m, n]$ is identified as a corner. The threshold can be set as a fixed constant. From our experiment, it would be better to set it according to

$$threshold = Max(R[m,n])/100 \quad [2\text{-}16]$$

Although the Harris corner detector does not take scale and rotation invariant into consideration, it still establishes a foundation of feature extracting.

## 2.3 FAST: Features from Accelerated Segment Test

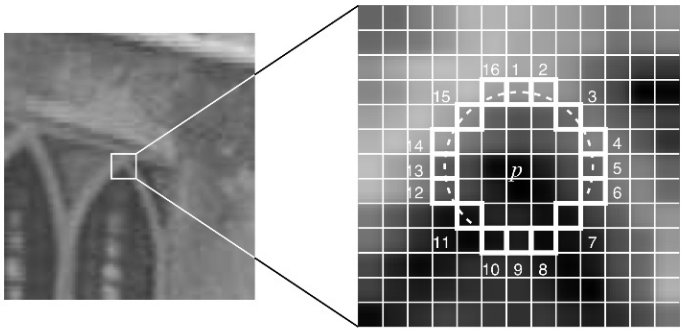


***Fig. 2.3:*** *An example of FAST detector with radius 3. The center point p is the point we want to classify. The pixels with number 1, 5, 9 and 13 is firstly checked for high speed test. [2]*

FAST [2], features from accelerated segment test, is also a simple way to detect corner features, but much different from the two methods above. It operates by considering a circle of sixteen pixels around the corner candidate $p$, which is illustrated in *figure 2.3*. It shows a circle with radius 3 and 16 pixels on the circle.

Suppose that $p$ is the interested point and its intensity is $I_p$. Then, $p$ is identified as a corner if there exists a set of $n$ contiguous pixels in the circle which are all brighter than $I_p + t$ or all darker than $I_p - t$ where $t$ is some threshold. Denote the number of pixels on the circle as $P$. We can denote a FAST detector as FAST $n$-$P$. The detailed algorithm of FAST 12-16 is shown below:

1. Choose a pixel $p$ and a proper value of threshold $t$ and consider a circle of radius 3 centered at $p$.
2. For a high speed test, we first check the intensity of pixels 1, 5, 9 and 13 (corresponding *figure 2.3*). Examine whether $I_1, I_5, I_9$ and $I_{13}$ are all brighter than $I_p + t$ or all darker than $I_p - t$. If yes, repeat this examination for the rest of the pixels. If no, $p$ will not be taken as a corner.
3. Calculate how many pixels are contiguous in the set of pixels passing the test above. If there exists more than $n = 12$ pixels fall in the criterion, p is considered a corner.

FAST simply uses comparison of the intensity of the interest point and its surrounding points. It might be the easiest understanding method. Although, it does not holds image transformation robustness, it is commonly applied as the first step to detect features.

# Chapter 3 Non-binary Descriptor

To be robust to a variety of possible transformations of images, such as scaling, rotating or even change in illuminance, the keypoints with descriptors should be applied. In this chapter, keypoints with non-binary descriptors will be adopted. The scaled invariant feature transform (SIFT) [3][4], considered a mostly used keypoint, will be presented first. Then, two improving methods of it, the principle component analysis SIFT (PCA-SIFT) [5] and the speeded up robust feature (SURF) [6], will be discussed.

## 3.1 SIFT: Scale Invariant Feature Transform

The Scale Invariant Feature Transform (SIFT) was first presented in 1999 [3] and finally completed in 2004 [4] by D. Lowe. It aims to transform an image into a large set of local feature vectors, which are invariant to image translation, scaling, and rotation, partially invariant to illumination changes, and robust to noise. It is considered one of the most important algorithms of feature detection and has inspired many template-matching and object detection applications that will be introduced later.

### 3.1.1 Keypoint Localization

The SIFT algorithm for keypoints detection can be mainly divided into three steps. The flow chart is shown in *figure 3.1* and the algorithm will be introduced step by step.


***Fig. 3.1**: Flow chart of keypoint localization of SIFT.*

- ***Step 1: Construct Scale Space***

  The scale space is divided into octaves. Each octave holds different scale of images. The next octave is the downsampling result of the previous octave in order to construct an image pyramid. The illustration is shown in *figure 3.2*.

  Furthermore, it has been shown that under some rather general assumptions on scale invariance, the Gaussian kernel and its derivatives are the only possible smoothing kernels for scale space analysis. Thus, in [3], the Gaussian kernel was used to create the scale space.

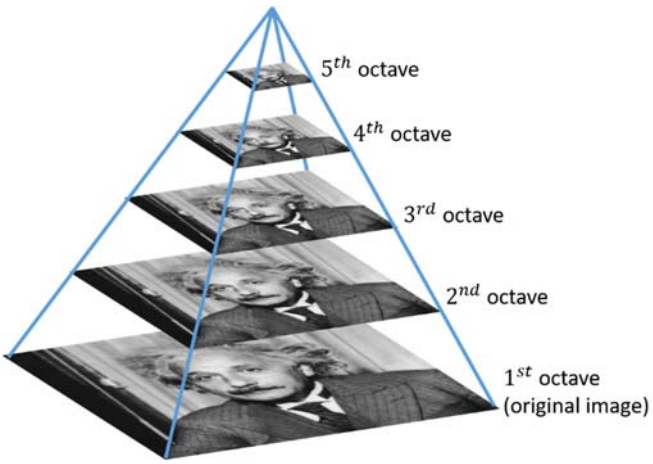


***Fig. 3.2:*** *Illustration of an image pyramid. Each image is the downsampling result of the previous one. The higher the octave is, the smaller the image will be. The photo of Einstein is from [16].*

In each octave, the first image $I(x, y)$ is filtered with a Gaussian kernel to get the second blurred image $L(x, y, \sigma)$. The function is denoted as

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad [3\text{-}1]$$

where

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2} \quad [3\text{-}2]$$

Then, for the next blurred image multiply a constant k to the scale of the previous. That is, the third blurred image can be denote as follow

$$L(x, y, \mathrm{k}\sigma) = G(x, y, \mathrm{k}\sigma) * I(x, y) \quad [3\text{-}3]$$

Thus, the scale of fourth image is $k^2\sigma$ and so on. k is usually $\sqrt{2}$ and σ doubles for the next octave. The scale space is constructed and an example is given in *figure 3.3*.

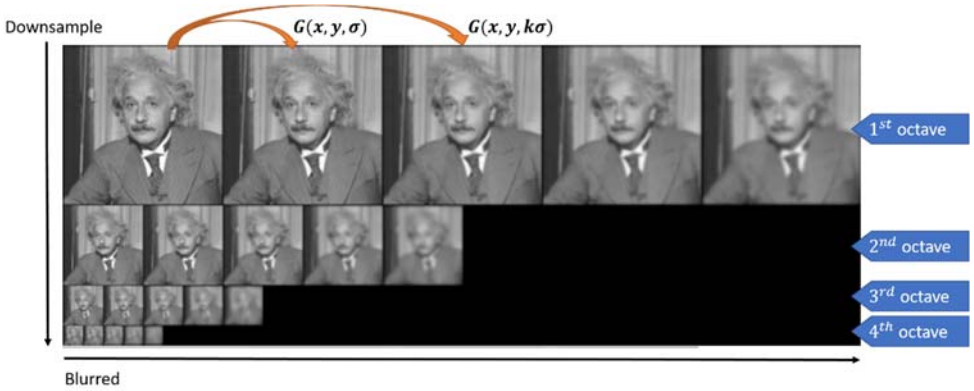


***Fig. 3.3:*** *This is an example of scaling space. In each octave, images are blurred from the previous one. As it has mentioned, as octaves goes higher, images become smaller. (Modified from the picture from [13]).*

- ***Step 2: Locate Potential Keypoints***

To find the location of keypoints, first, the Difference-of-Gaussian (DoG) filter defined as follows is applied to the scale space.

$$G(x, y, k\sigma) - G(x, y, \sigma) \quad [3\text{-}4]$$

Then, convolution the scale space with a DoG filter:

$$\begin{aligned} D(x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \quad [3\text{-}5]$$

It shows that simply subtracting the next image with the previous image in the same octave will get the result. *Figure 3.4 (a)* shows the illustration of the scale space applied with a DoG filter. The SIFT uses this because it is efficient and stable.

Next, locate the extrema of the DoG outputs. For each pixel, scan their neighboring points (as the green points in *figure 3.4 (b)*). One pixel in an image is compared with its 8 neighbors as well as 9 pixels in the next scale and 9 pixels in the previous scales. The pixel *x* is selected if it is the maxima or the minimum among these comparing points.

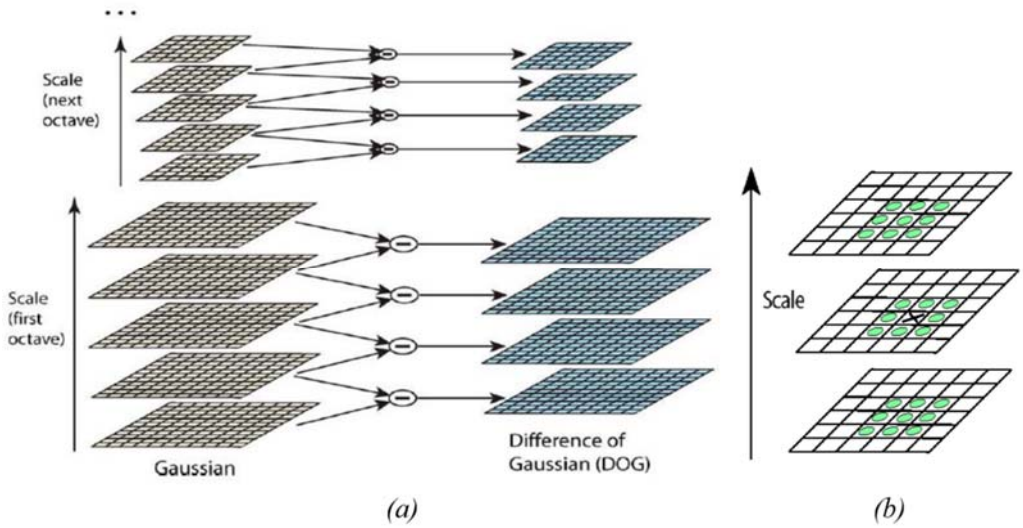


***Fig. 3.4:*** *Illustration of the DoG pyramid. In (a), after applying Gaussian smoothing filters on each level in each octave, the difference of Gaussian is applied to perform keypoint localization. In (b), the point with an x on is the interest point and the green ones are neighbors that will be compared to it in order to confirm if x is an extremum. [4]*

- ***Step 3: Sub-pixel Location***

The extrema mentioned in step2 are search in discrete space, so these value isn't the real extreme values. *Fig. 3.5* illustrates the reason. Thus, using the known discrete space to obtain extrema in continuous space is called sub-pixel location [18]. In this way, we can find the accurate location of keypoints.

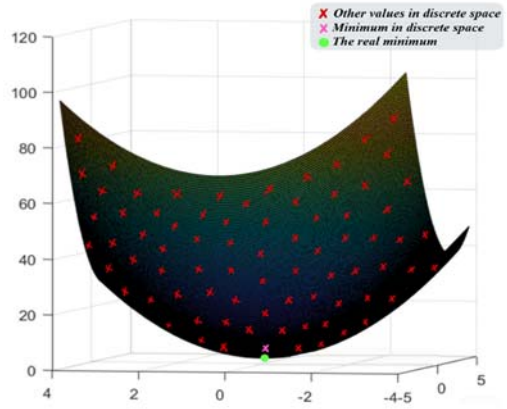


***Fig. 3.5:*** *This figure is an illustration of the different between minimum in discrete and continuous space. Our goal is to find the real minimum (the green point).*

First, apply Taylor expansion on a DoG image D

$$D(\boldsymbol{x}) = D + \frac{\partial D}{\partial \boldsymbol{x}}^T \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T \frac{\partial^2 D}{\partial \boldsymbol{x}^2}\boldsymbol{x} \qquad [3\text{-}6]$$

**x** is the location of a point on D. Then, to find the extreme value find a location $\hat{\boldsymbol{x}}$ to make *Eq.(3-6)* have zero value.

$$\hat{\boldsymbol{x}} = -\frac{\partial^2 D}{\partial \boldsymbol{x}^2}^{-1} \frac{\partial D}{\partial \boldsymbol{x}} \qquad [3\text{-}7]$$

Next, the extrema $D(\hat{\boldsymbol{x}})$ can be derived as follow

$$D(\hat{\boldsymbol{x}}) = D + \frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \hat{\boldsymbol{x}} + \frac{1}{2}\hat{\boldsymbol{x}}^T \frac{\partial^2 D}{\partial \boldsymbol{x}^2}\hat{\boldsymbol{x}} \qquad [3\text{-}8.1]$$

$$= D + \frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \hat{\boldsymbol{x}} + \frac{1}{2}\left(-\frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1} \frac{\partial D}{\partial \boldsymbol{x}}\right)^T \frac{\partial^2 D}{\partial \boldsymbol{x}^2}\left(-\frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1} \frac{\partial D}{\partial \boldsymbol{x}}\right) \qquad [3\text{-}8.2]$$

$$= D + \frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \hat{\boldsymbol{x}} + \frac{1}{2}\frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-T} \frac{\partial^2 D}{\partial \boldsymbol{x}^2}\frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1} \frac{\partial D}{\partial \boldsymbol{x}} \qquad [3\text{-}8.3]$$

$$= D + \frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \hat{\boldsymbol{x}} + \frac{1}{2}\frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1} \frac{\partial D}{\partial \boldsymbol{x}} \qquad [3\text{-}8.4]$$

$$= D + \frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T \hat{\boldsymbol{x}} + \frac{1}{2}\frac{\partial \mathrm{D}}{\partial \boldsymbol{x}}^T (-\hat{\boldsymbol{x}}) \qquad [3\text{-}8.5]$$

Substitute $\hat{\boldsymbol{x}}$ in *Eq. (3-8.1)* with *Eq. (3-7)* to get *Eq. (3-8.2)*. Then, in *Eq. (3-8.3)*, due to $\frac{\partial^2 D}{\partial \boldsymbol{x}^2}\frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1} = 0$ and $D$ is a 2D matrix, i.e. $\frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-T} = \frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1}$, we can obtain *Eq. (3-8.3)*. Finally, changing $\frac{\partial^2 \mathrm{D}}{\partial \boldsymbol{x}^2}^{-1}\frac{\partial D}{\partial \boldsymbol{x}}$ back to $-\hat{\boldsymbol{x}}$, we can get the accurate DoG extreme value $D(\hat{\boldsymbol{x}})$ on location $\hat{\mathbf{x}}$.

- ***Step 4: Filter Edge and Low Contrast Responses***

Finally, edges and low contrast points should be removed from the potential keypoints. To remove low contrast points, we apply a low contrast points filter which is equal to the real extrema of $D$ in *Eq. (3-8.5)*. Then, remove the points with the following constraint

$$|D(\hat{\boldsymbol{x}})| < 0.03 \qquad [3\text{-}9]$$

To remove points on edges, consider a Hessian matrix of a potential keypoint

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \qquad [3\text{-}10]$$

From the knowledge mentioned in chapter 2.1, the absolute value of the disparity between the two eigenvalues of $\boldsymbol{H}$, $\lambda_1$ and $\lambda_2$, should be large. Thus, consider the trace and the determinant of $\boldsymbol{H}$:

$$tr(\boldsymbol{H}) = \lambda_1 + \lambda_2 \qquad [3\text{-}11]$$
$$det(\boldsymbol{H}) = \lambda_1\lambda_2 \qquad [3\text{-}12]$$

Let $\lambda_1 = r\lambda_2$, we have

$$\frac{tr(\boldsymbol{H})^2}{det(\boldsymbol{H})} = \frac{(r+1)^2}{r} \qquad [3\text{-}13]$$

Note that

$$\frac{(r+1)^2}{r} \to \infty \ when\ r \to 0\ or\ r \to \infty \qquad [3\text{-}14]$$

and $\frac{(r+1)^2}{r}$ is minimum at $r = 1$. Since SIFT points are usually corner-like and for a corner-like pixel $\lambda_1$ and $\lambda_2$ should not have a very large difference, to identify whether a pixel is a SIFT point, we can identify a threshold $T$:

$$T = \frac{(r_0+1)^2}{r_0} \quad [3\text{-}15]$$

where $r_0$ is some adjustable constant, which corresponds to the maximal allowed ratio of $\lambda_2$ to $\lambda_1$ for a SIFT point. If

$$\frac{tr(\boldsymbol{H})^2}{det(\boldsymbol{H})} \leq \mathrm{T}, \quad [3\text{-}16]$$

then the potential keypoint is treated as a SIFT point.

### 3.1.2 Keypoint Description

For rotation invariance, assigning an orientation to each keypoint is necessary. After assigning the orientation, the descriptors for each keypoint will be determined. The descriptors are helpful for keypoint matching, i.e., determining whether the keypoints in different images correspond to the same part of an object.

- ***Orientation Assignment***

  Consider the Gaussian-smoothed image $L$. The gradient magnitude $m(x, y)$ and the orientation $\theta(x, y)$ at the keypoint $(x, y)$ are defined as follows:

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2} \quad [3\text{-}17]$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right) \quad [3\text{-}18]$$

  For the region around the keypoint, we create a histogram with 36 bins for orientation, i.e., each bin has the range of 10 degrees. As the process shown in *figure 3.6*, we weight each point with Gaussian window of $1.5\sigma$. Assign the orientation value that is the highest peak in the histogram. Then, consider all peaks with the histogram value larger than 0.8×Max(histogram). Create keypoints with same location and scale and assign the orientation value that has the peaks with value bigger than 0.8×Max(histogram). It means that keypoints with same location and scale, but different directions, are created in order to contributes the stability of matching. The maximal number of multiple peaks at the same location is limited to 2.

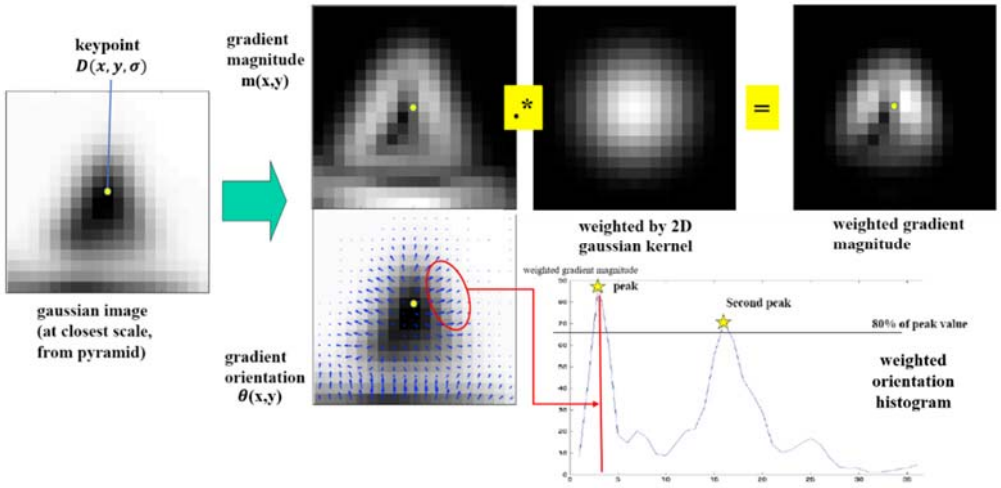


***Fig. 3.6:*** *Illustration of the process of SIFT orientation assignment. The chart on the lower righter corner is the orientation histogram. In this case, we can see that there are 2 peaks above 0.8×Max(histogram). Thus, 2 keypoints with same locations and scales but different orientation is created. (Modified from [15])*

- ***Local Image Descriptor***

Now, each keypoint has its location, scale, and orientation. The final step is to construct a descriptor for each keypoint.

First, find the blurred image according to the scale of the keypoint and consider a $16 \times 16$ window center at the point. Then, rotate the gradients and coordinates by the previous assigned orientation. Divide the window to $4 \times 4$ and create a histogram for each sub-region with 8 bins, i.e., the histograms of the gradients along 8 different orientations (see *figure 3.7*). $4 \times 4 \times 8$ directions give a 128 bins. For illumination independence, normalized the intensity, any number (of the 128) greater than 0.2 is changed to 0.2, and then renormalized it. Finally, a descriptor is presented in a dimension of 128 vector which is quite large for computation and this is indeed a problem of the SIFT.

Two matching examples of SIFT are shown in *fig. 3.8*. The red points are the detected SIFT keypoints and the lines are the matching results. We can see that even though the image is transformed (scaling in (a) and shearing in (b)), the relative locations of red points remain unchanged. This represents that SIFT is robust to some affine transformations. Although the SIFT descriptor is accurate and has invariant properties, it is really time-consuming, so it may not be the best choice for real-time application. Also, it is not robust enough to luminance changes. Thus, many other algorithms were developed to improve the performance and reduce the complexity of the SIFT.

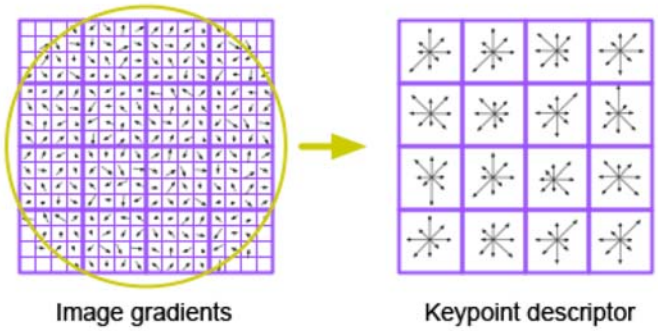


***Fig. 3.7:*** *The SIFT considers an* $16 \times 16$ *area around the keypoint and each pixel has its own orientation(the left part). Then divide this area to* $4 \times 4$ *sub-block and get the histogram of each sub-block (the right part).[14]*

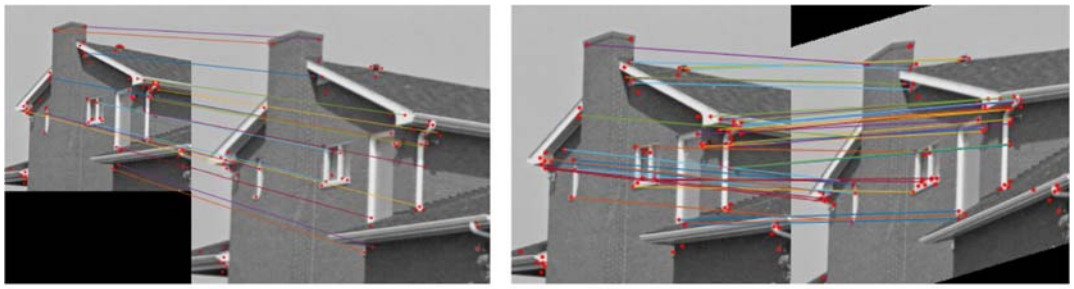


***Fig. 3.8:*** *This figure shows two example of SIFT keypoint locations and matching. ((a) for scaling and (b) for shearing.) Those red points and lines are SIFT keypoints and matching results respectively. We can see that the relative locations of keypoints remain unchanged, e.g. there's a SIFT keypoint on the right top of the chimney in original picture and there's also one in each of the transformed image.*

## 3.2 PCA-SIFT

The principle component analysis SIFT (PCA-SIFT) [5] is the simplest way to accelerate the SIFT. Principle component analysis (PCA) is used to reduce the dimension of SIFT descriptors. Its algorithm in keypoints localization and orientation assignment is the same as that of the SIFT. It only changes the way to get final descriptors.

First, the PCA-SIFT create $41 \times 41$ patches centered at the keypoint and pre-compute an eigenspace to express the gradient images for both horizontal and vertical directions of the patch. It only chooses the center $39 \times 39$ area of the patch for the computation of next step. Thus, here we get $2 \times 39 \times 39 = 3042$ elements (the 2 is for the two direction: horizontal and vertical). Next, PCA is applied to the covariance matrix of the 3042-element vectors. Find the eigenvalues and sort the eigenvectors from the largest to the smallest eigenvalue.

Finally, preserve only the top 20 of the eigenvectors. The original SIFT descriptor has a dimension of 128 and the gradients of each sub-block has to be computed individually. The PCA-SIFT reduces the dimension to 20 and the gradient is simply presented by the eigenspace corresponding to these 20 eigenvectors. It efficiently reduces the computation time of the SIFT.

## 3.3 SURF: Speed Up Robust Features

The PCA-SIFT decreases the matching complexity only by dimensionality reduction of descriptors. The speed up robust features (SURF) [6] not only reduces the dimension of descriptors, but also apply box filters and the integral image calculation to reduce the computation time of keypoint extraction. Meanwhile, the SURF applies new ways to decide the locations and orientations of keypoints.

### 3.3.1 Keypoints Extraction

For computational efficiency, integral images and hessian approximation with box filter are applied. In the followings, these two tools will be introduced first. Thus, how to construct the SURF's scale space and localize keypoints will then be presented.

- ***Integral Images***

  The SURF applies the idea of integral images that was first introduced by Viola and Jones for fast computation of box type convolution filters, which can be used in Hessian matrix approximation introduced in the next step.

  The entry of an integral image $I_\Sigma(\boldsymbol{x})$ at a location $\boldsymbol{x} = (x, y)^T$ represents the sum of all pixels in the input image I within a rectangular region formed by the origin and $\boldsymbol{x}$.

$$I_\Sigma(\boldsymbol{x}) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j) \qquad [3\text{-}19]$$

  In *figure 3.9*, by using the idea of integral image, the sum of intensity in the area $\Sigma$ can be easily found out by simply calculating $I_A - I_B - I_C + I_D$. It can much reduce the computation time.

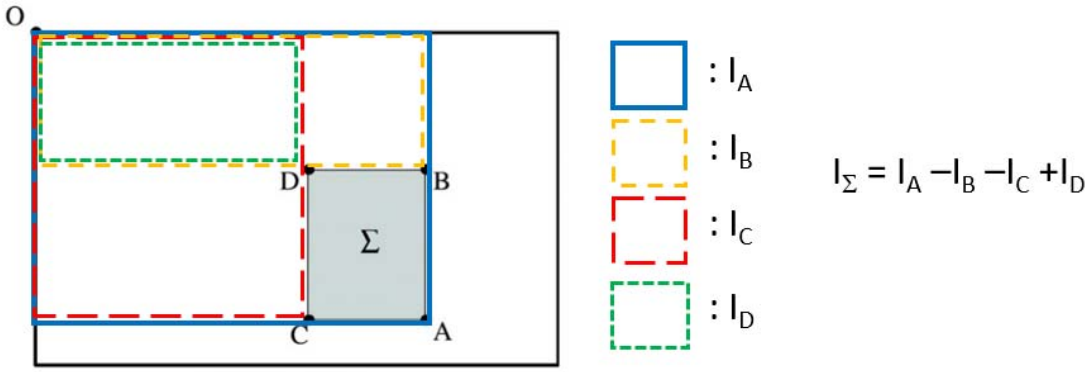


***Fig. 3.9:*** *Illustration of the use of integral image. To obtain the intensity in ∑, only three additions are taken (modified from [6]).*

- ***Hessian Matrix Approximation***

The SUFF applies Hessian matrix approximation which is different from the DoG image in the SIFT because of its computation efficiency and good performance in accuracy.

Given a point $\boldsymbol{x} = (\mathrm{x}, \mathrm{y})$ in an image I and the Gaussian filter is used first to make keypoints scale-invariant. The Hessian matrix $\mathcal{H}(\boldsymbol{x}, \sigma)$ in $\boldsymbol{x}$ at scale $\sigma$ is defined as

$$\mathcal{H}(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix} \quad [3\text{-}20]$$

where $L_{xx}(\boldsymbol{x}, \sigma)$ is the convolution of the Gaussian second order derivatives $\frac{\partial^2}{\partial x^2} g(\sigma)$ with $I(\boldsymbol{x})$ and similarly for $L_{xy}(\boldsymbol{x}, \sigma)$ and $L_{yy}(\boldsymbol{x}, \sigma)$.

Furthermore, the SURF pushes the approximation for the Hessian matrix with the box filter, shown in *figure 3.10*, which combine the 2 processes, (i) Gaussian smoothing and (ii) second derivative, to only one step. This approximation can use integral image for calculation to reach low computational cost. The Hessian matrix is changed as

$$\mathcal{H}_{approx}(\boldsymbol{x}, \sigma) = \begin{bmatrix} D_{xx}(\boldsymbol{x}, \sigma) & D_{xy}(\boldsymbol{x}, \sigma) \\ D_{xy}(\boldsymbol{x}, \sigma) & D_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix} \quad [3\text{-}21]$$

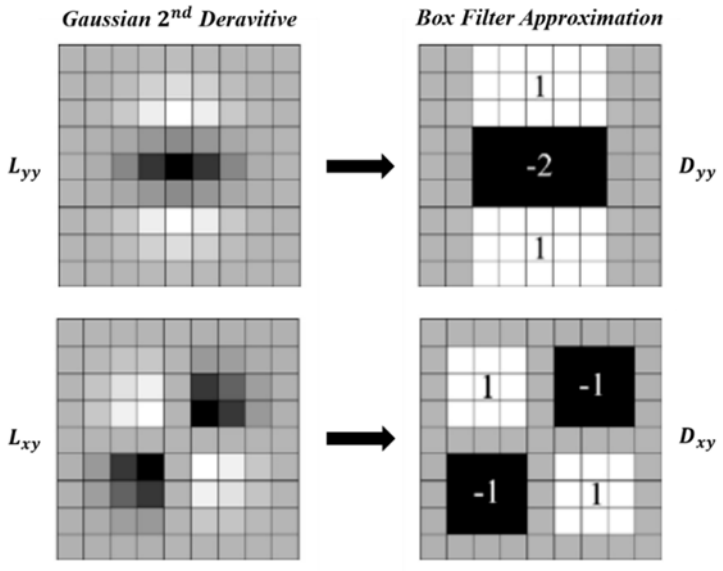


***Fig. 3.10:*** *The SURF uses box filters instead of the original Gaussian kernels. The top part shows the approximation of Gaussian second order partial derivatives in y-direction. The lower part shows the one in xy-direction.[6]*

Finally, similar to the DoG images of the SIFT, the transformed images for keypoint localization is needed. The SURF then uses the approximation of $det(\mathcal{H}_{approx})$:

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad [3\text{-}22]$$

It can be viewed as a simplification of *Eqs. (2-9)* and *(3-11)*. This operation is applied to every $\boldsymbol{x}$ over different scales and the local maxima of $det(\mathcal{H}_{approx})$ are considered as potential keypoints.

- ***Scale Space and Keypoints Localization***

The scale space of the SURF is much similar to the one in the SIFT. However, instead of downsampling the image to get the next octave image (see *fig. 3.11(a)*), for the SURF the input image is convolved with a set of filters of increasing size (see *fig. 3.11(b)*). With the help of the integral image and the box filter, it can filter the image with a variety of filter sizes in an efficient way.

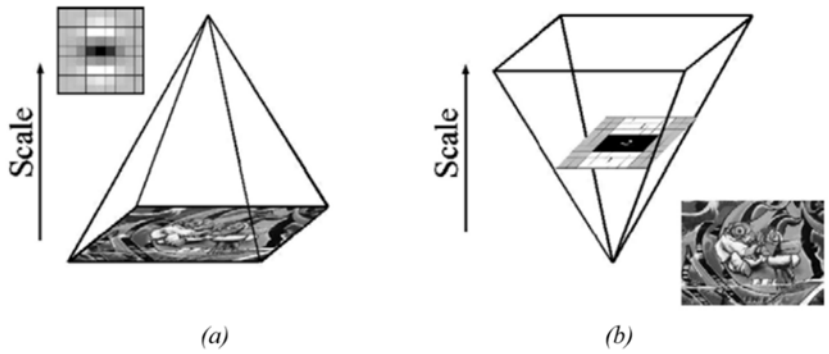


***Fig. 3.11:*** *The comparison of SIFT and SURF scale spaces. In (a), the image is getting smaller along the scale axis. However, in (b), the image size remains the same and the filter sizes keep changing instead. [6]*

Denote the order of the octave as $n$, i.e the first octave is $n = 1$, The increasing rule of the filter size is defined as follows (We can compare the rule with that of the SIFT in *figure 3.2* and *figure 3.3*):

- Construct the scaled space starts with the $9 \times 9$ filter, which is for the image of the smallest scale.
- For the first octave, the following filter size is increased by 6 pixels, as in *figure 3.10*. The first filter is with sizes $9 \times 9$ and then $15 \times 15$, $21 \times 21$, and $27 \times 27$.
- For other octaves, denote the smallest side length of the filter of the $n^{th}$ octave as $f(n)$, then

$$f(n+1) = f(n) + 6(n-1),\ f(1) = 9 \qquad [3\text{-}23]$$

- For each octave, the increasing size of the following filter is doubled. That is, for the second octave, the filter size is increased by 12 pixels, and for the third octave, the filter size is increased by 24 pixels.

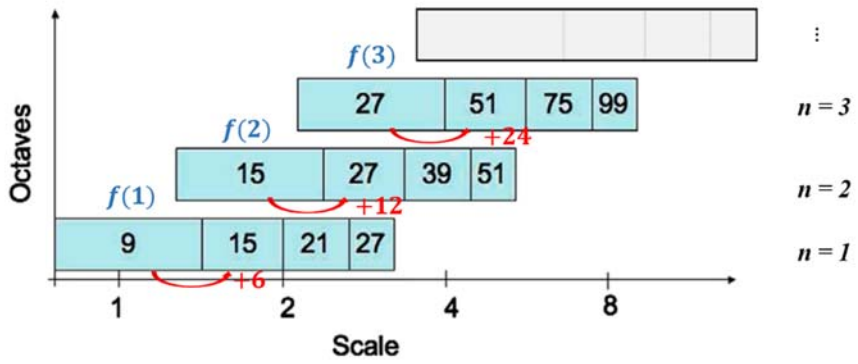


***Fig. 3.12:*** *The graphical representation of the filter size lengths for the first three octaves. The original image is filtered with a* $9 \times 9$ *filter and the filter size increasing rule is as mentioned. (Modified from [6])*

Now, the scale space is finally constructed. A graphical representation of the filter side lengths is shown in *figure 3.12*. The keypoint localization is much similar to that of the SIFT. However, the SURF finds the maximum of $det(\mathcal{H}_{approx})$ instead of finding the extrema of the DoG.

### 3.3.2 Keypoint Description

Quite different from the SIFT, the SUFT applies the response of the Haar wavelet to keypoint description rather than the gradient. The first step consists of fixing a reproducible orientation based on the information from a circular region around a keypoint for rotation invariant. The next step is to construct the descriptor.

- ***Orientation Assignment***

  First, calculate the Haar wavelet responses in both x and y direction within a circular neighborhood of radius $6s$ centered at the keypoint. ($s$ denotes the scale of the keypoint.) Then, filter the responses with a Gaussian ($\sigma = 2s$). For fast filtering, the filter shown in *figure 3.13* is used and integral image calculation can be applied.

  Next, sum up the horizontal and vertical responses, $d_x$ and $d_y$, within a sliding orientation window which is a circular sector with $\theta = {}^{\pi}/_{3}$ (see *figure 3.14*). The window that has the largest sum of horizontal and vertical responses is then the orientation of the keypoint.

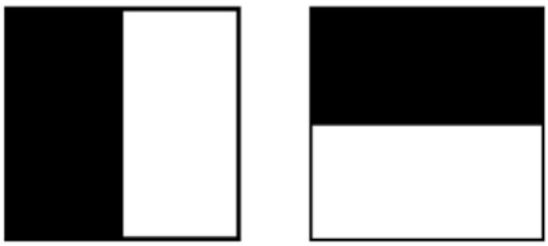

***Fig. 3.13:*** *From the left to right, it shows the Haar wavelet filter to compute the responses in x and y-direction. The dark parts have the weight -1 and the light parts 1. [6]*

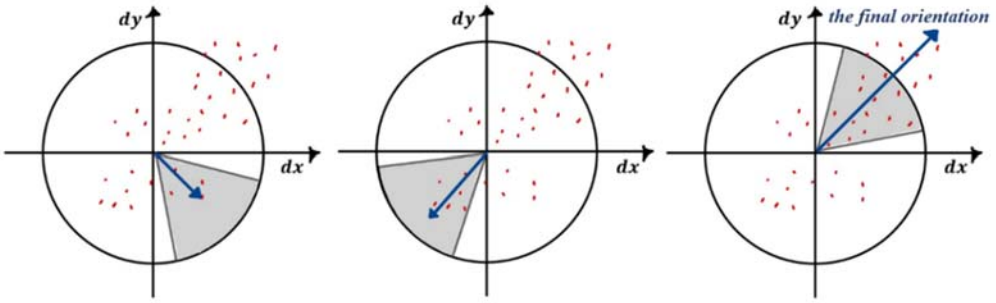


***Fig. 3.14:*** *The gray area in this image is the sliding orientation window. It scans the image patch and calculate the sum of Haar wavelet response in x and y-directions. The right most part shows the final orientation and the density of the distribution of the responses is indeed the highest.*

- ***Local Image Descriptor***

To construct the descriptor, a square window of size $20s \times 20s$ is first created ($s$ denotes the scale of the keypoint.) The window is centered at the keypoint and rotated to the orientation of the keypoint. Then, the window is divided into $4 \times 4$ square sub-regions. Each sub-region has the size of $5s \times 5s$. For each sub-region, Haar wavelet responses in both horizontal and vertical direction are calculated and filtered with a Gaussian function ($\sigma = 3.3s$).

Just like the orientation assignment mentioned above, the wavelet response, $d_x$ and $d_y$, is summed up. In order to bring information about the polarity of the intensity changes, the sums of $|d_x|$ and $|d_y|$ are also considered. *Figure 3.15* gives an example for the different value of $d_x$, $d_y$, $|d_x|$ and $|d_y|$ in different image-intensity patterns. In this figure, *x* is the horizontal axis and *y* is the vertical axis.

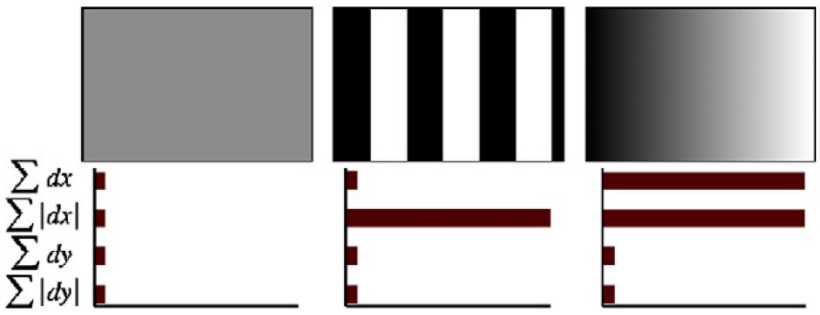


***Fig. 3.15:*** *The 4 descriptor entries perform differently in different image patterns. In a homogeneous region (left), each values are low. When frequencies impulsively vary in x-direction (middle),* $\sum|d_x|$ *is high. When frequencies equally vary in x-direction (right), then both* $\sum|d_x|$ *and* $|d_x|$ *are high. [6]*

Finally, for each sub-region, a descriptor $\boldsymbol{v}$ with dimension of 4 is created as follows.

$$\boldsymbol{v} = (\sum d_x, \sum d_y, \sum|d_x|, \sum|d_y|) \qquad [3\text{-}24]$$

Concatenating all the $\boldsymbol{v}$ for all sub-regions results in a 64D descriptor, which is a half of the 128D SIFT descriptor. The illustration of a descriptor is shown in *figure 3.16*.

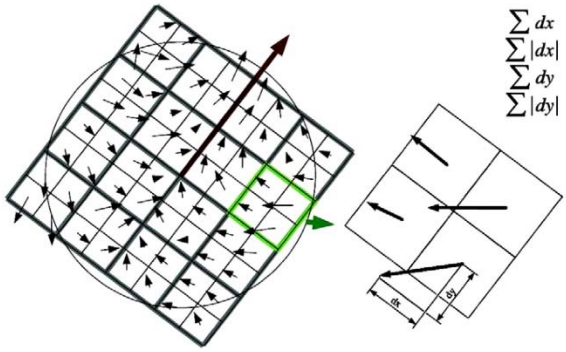


***Fig. 3.16:*** *Illustration of the construction of descriptors. In each sub-block, four values are calculated and describe the orientation properties of each sub-block. [6]*

# Chapter 4 Binary Descriptor

In this chapter, 4 kinds of keypoints using binary descriptors are introduced. Different from chapter 3, these keypoints use binary strings as efficient keypoint descriptors. They may apply the Hamming distance rather than the $L_2$ norm when matching. The details about matching will be discussed in the next chapter. We will introduce the BRIEF [7] and the ORB [8], which use random pairs when constructing descriptors. Other two keypoints, the BRISK [9] and the FREAK [10], apply circular patterns.

## 4.1 BRIEF: Binary Robust Independent Elementary Features

The binary robust independent elementary feature (BRIEF) [7] is one of the most fundamental binary descriptors. Most of the binary descriptors before the BRIEF first compute the full descriptors then reduce them to binary strings. This process cost lots of time. However, the BRIEF directly computing binary strings from image patches which makes it fast to build.

A patch is defined as a square centered at some keypoint. First, we define a test function $\tau$ on an image patch $\boldsymbol{p}$ of size $S \times S$ as

$$\tau(\boldsymbol{p}; \boldsymbol{x}; \boldsymbol{y}) := \begin{cases} 1 \; if \; \boldsymbol{p}(\boldsymbol{x}) < \boldsymbol{p}(\boldsymbol{y}) \\ 0 \; otherwise \end{cases} \quad [4\text{-}1]$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the chosen two points in a patch (will be discuss next) and $\boldsymbol{p}(\boldsymbol{x})$ is the pixel intensity of $\boldsymbol{p}$ at $\boldsymbol{x} = (u, v)^T$ after smoothing with a Gaussian kernel with $\sigma = 2$.

Next, choose a set of $n_d$ $(\boldsymbol{x}, \boldsymbol{y})$-location pairs in the image patch. The selection is randomly sampled as follows:

$$(\boldsymbol{X}, \boldsymbol{Y}) \sim i.i.d. Gaussian(0, \tfrac{1}{25} S^2) \quad [4\text{-}2]$$

where i.i.d. means independent and identically distributed. The chosen pairs are sampled from an isotropic Gaussian distribution with zero mean and $\sigma = \frac{1}{25} S^2$. This test gives the best result. Examples for chosing pairs in different ways also tested in [7] is shown in *figure 4.1*. $GII$ is the distribution of selected pairs, which BRIEF applies and is considered the best result. The black lines in *figure 4.1* are the chosen pairs, i.e. the two endpoints of each line are the chosen points to form a pair.

Finally, the BRIEF descriptor is taken to be $n_d$-dimensional bitstring $f_{n_d}(\boldsymbol{p})$

$$f_{n_d}(\boldsymbol{p}) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(\boldsymbol{p}; \boldsymbol{x_i}; \boldsymbol{y_i}) \quad [4\text{-}3]$$

where $n_d$ can be 128, 256 and 512. The BRIEF descriptor with different length is referred as

BRIEF-k, where $k = n_d/8$ is the number of bytes required to store the descriptor.

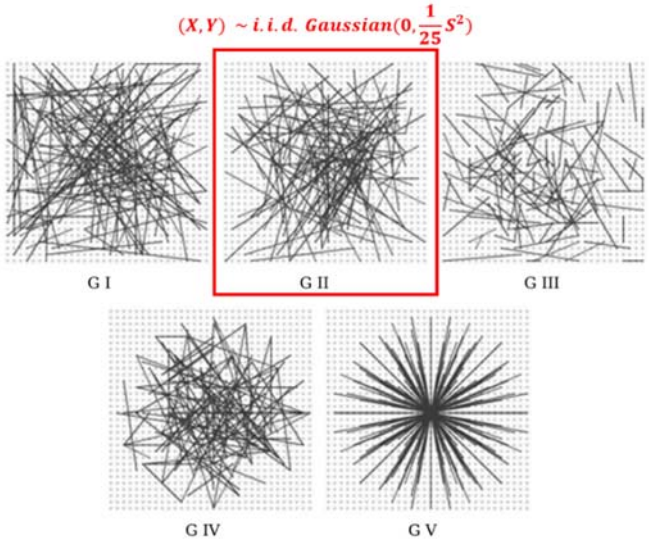


***Fig. 4.1:*** *This shows some different distribution of chosen pairs in one patch. BRIEF applies the distribution in GII. The two endpoints of each dark line represent the two points which form a sampling pair. [7]*

BRIEF is fast to build and match, but it actually performs poorly with rotation because it considers only the intensity. The next method, the ORB, improves it to the rotation-aware BRIEF and makes it more robust to rotation.

## 4.2 ORB: Oriented FAST and Rotated BRIEF

The oriented FAST and rotated BRIEF (ORB) [8] adds orientation assignment to the FAST feature detector (see Section 2.3) and the BRIEF descriptor. In this section, two processes, (i) the oFAST and (ii) the rBRIEF, will be introduced separately.

- ***oFAST: FAST Keypoint Orientation***

  The ORB starts by detecting FAST features in the image. It applies the FAST-9 detector for the best performance. To reduce large responses along edges, a Harris corner measure is used to sort the FAST keypoints and pick the top *N* of them. For scale invariance, the images are put in a scale pyramid and the keypoints are localized at each scale.

  Orientation of keypoints is measured by the intensity centroid. It is assumed that the intensity of the corner is offset from its center. The centroid is defined as

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \qquad [4\text{-}4]$$

where $m_{10}$, $m_{01}$ and $m_{00}$ are the moments of a patch.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y). \quad [4\text{-}5]$$

The patch is chosen to be a circular region of radius *r*, so $x, y \in [-r, r]$.

Then, construct a vector $\overrightarrow{OC}$ from the corner center $O$ to the centroid *C*. Finally, the orientation of the patch is

$$\theta = atan2(m_{01}, m_{10}) \quad [4\text{-}6]$$

where *atan*2 is the quadrant-aware version of arctan, i.e.,

$$\theta = \arctan\left(\frac{m_{01}}{m_{10}}\right) = \arcsin\left(\frac{m_{01}}{\sqrt{{m_{01}}^2+{m_{10}}^2}}\right) = \arcsin\left(\frac{m_{10}}{\sqrt{{m_{01}}^2+{m_{10}}^2}}\right) \quad [4\text{-}7]$$

An illustration of orientation assignment is shown in *figure 4.2*.

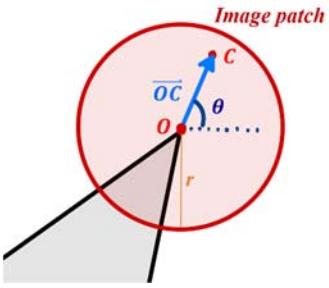


***Fig. 4.2:*** *An illustration of centroid orientation. The center $O$ of the circular image patch is the detected keypoint and $\theta$ is its orientation. C is the intensity centroid.*

- ***rBRIEF: Rotation-Aware BRIEF***

The rotation-aware BRIEF (rBRIEF), as its name implies, improves the original BRIEF descriptor by including orientation description. From the knowledge mentioned in Section 4.1, a BRIEF descriptor with length $n_d$ is taken to be

$$f_{n_d}(\boldsymbol{p}) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(\boldsymbol{p}; \boldsymbol{x}_i; \boldsymbol{y}_i) \quad [4\text{-}8]$$

where the test $\tau(\mathbf{p}; \boldsymbol{x}_i; \boldsymbol{y}_i)$ is defined as

$$\tau(\boldsymbol{p}; \boldsymbol{x}_i; \boldsymbol{y}_i) := \begin{cases} 1 \; if \; \boldsymbol{p}(\boldsymbol{x}_i) < \boldsymbol{p}(\boldsymbol{y}_i) \\ 0 \; otherwise \end{cases} \quad [4\text{-}9]$$

The random selection of $(\boldsymbol{x}, \boldsymbol{y})$-location pairs are also i.i.d. $Gaussian(0, \frac{1}{25}S^2)$.

For the ORB, usually the descriptors of the BRIEF-32 are applied (i.e. $n_d = 256$. See the definition in chapter 4.1). Furthermore, to allow the BRIEF to be robust to in-plane rotation, it is steered according to the orientation of keypoints. For any set of $n_d$ binary tests at location $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, an $2 \times n_d$ matrix $\boldsymbol{S}$ is defined:

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{x}_i, \dots, \boldsymbol{x}_n \\ \boldsymbol{y}_i, \dots, \boldsymbol{y}_n \end{pmatrix} \quad [4\text{-}10]$$

Use the patch orientation, $\theta$, calculated in the oFAST, and rotate the patch to $\theta$ with a rotation matrix $\boldsymbol{\mathcal{R}}_\theta$. Then, the steered version of $\boldsymbol{S}$ is constructed:

$$\boldsymbol{S}_\theta = \boldsymbol{\mathcal{R}}_\theta \boldsymbol{S} \quad [4\text{-}11]$$

Finally, an rBRIEF descriptor $g_{n_d}$ which is the rotated version of $f_{n_d}$ can be created. It means that under the condition that $(\mathbf{x}_i, \mathbf{y}_i)$ in $\mathrm{f_{n_d}}(\mathbf{p})$ belongs to $\boldsymbol{S}_\theta$, $g_{n_d}$ can be defined:

$$g_{n_d}(\boldsymbol{p}, \theta) \coloneqq f_{n_d}(\boldsymbol{p}) | (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \boldsymbol{S}_\theta \quad [4\text{-}12]$$

In conclusion, the ORB combines the advantage of the computation efficiency of FAST and BRIEF, and makes some correction to have robustness to rotated cases. It requires less computation time than the SIFT and the SURF, meanwhile, remaining a good performance in distorted images.

## 4.3 BRISK: Binary Robust Invariant Scalable Keypoints

The binary robust invariant scalable keypoints (BRISK) [9] combines a FAST-based detector and bit-string descriptors. Unlike the BRIEF and the ORB, it applies circular image patches and the sampling points are equally distributed. The two steps, (i) keypoints localization and (ii) description, will be introduced separately.

### 4.3.1 Keypoint Localization

To localize keypoints, the BRISK mainly applies FAST 9-16 detectors (see Section 2-3) to find possible features. Its process can be summarized as the flowchart shown in *figure 4.3*. These steps are applied in the scaling space, so its scaling space will be introduced first.


***Fig. 4.3:*** *Flow chart of the BRISK.*

- ***The Scaling Space***

In the BRISK, the scaling space consists of $n$ octaves (denoted by $c_i$, $i$ = 1, 2, …, $n$) and $n$ intra-octaves (denoted by $d_i$, $i$ = 1, 2, …, $n$) and usually $n = 4$. Each octave or intra-octave is the downsampling result of the previous one. The downsampling rules are as follows:

- $c_{i+1}$ is downsampled from $c_i$ by a factor of 2.
- $d_{i+1}$ is downsampled from $d_i$ by a factor of 2.
- $d_i$ is downsampled from $c_i$ by a factor 1.5.

The illustration of the above method is shown in the left part of *figure 4.4*.

- ***FAST 9-16 Detection and Non-maxima Suppression***

A FAST 9-16 detector, which needs 9 consecutive pixels in the 16-pixel circle to satisfy the criterion of FAST, is applied on each octave and intra-octave with the same threshold. Before using FAST, the image should be smoothed with Gaussian kernels as the others mentioned before. Integral images and the box filter, i.e. the skills in the SURF, can be used here for computation efficiency.

The non-maxima suppression is quite similar to the SIFT. The BRISK estimates the FAST score $s$ of a potential keypoint and its $3^3 - 1 = 26$ neighbors. (8 in the same layer and total 18 in the lower and upper layer). An keypoint must be a local maximum in this score. The FAST score $s$ is defined as:

$$s = \sum_{i=1}^{16} |I_p - I_{p_i}| \qquad [4\text{-}13]$$

where $I_p$ is the intensity of the interest point and $I_{p_i}$ is the intensity of the $i^{\text{th}}$ point on the circle in Fig. 2.3.

Moreover, in the case where $c_0$ which is the original input image, the FAST 5-8 detector is applied. By this way, keypoints can be well localized in $c_0$.

Each potential point is compared with those in upper and lower layers.

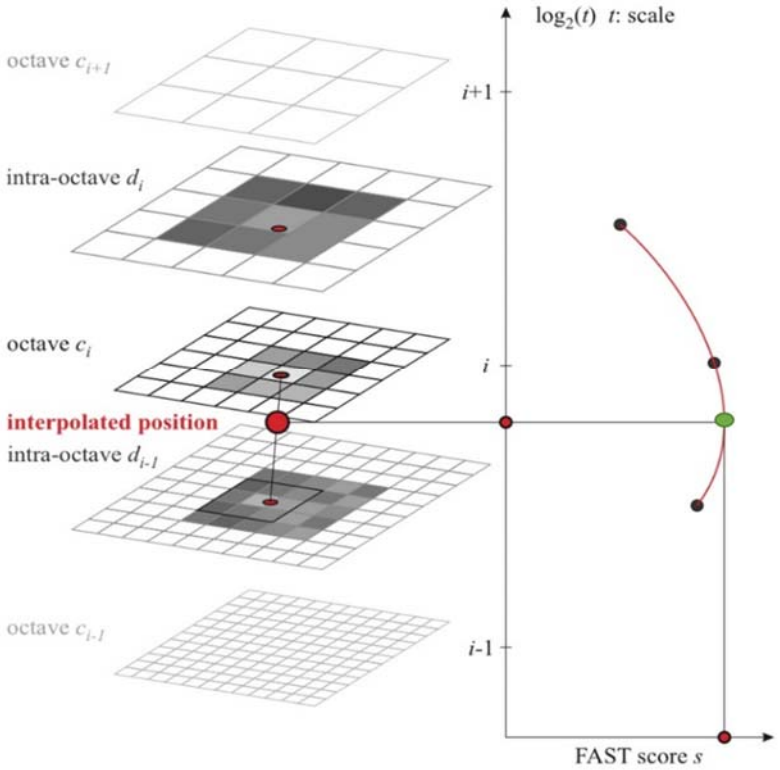


***Fig. 4.4:*** *Illustration of the BRISK scale space. In the left part, we can see that the scales of octaves and intra-octaves follow the rules above. The red points are the keypoint and its corresponding points on the neighboring octaves. In the left part, a fitted parabola of the scale and score of these three red points on the octaves is shown. The maximum in score-axis is the green point. With the help of its scale, we can re-interpolate a position between the octaves to be the final keypoint location (the biggest red point on the right part). [9]*

- ***Location Interpolation***

To improve the performance in the blurred image, the BRISK aims to estimate the true scale of each keypoint. Thus, interpolation is applied to get a more precise location and scale. To obtain the location, apply sub-pixel location (similar to *Eq. (3.6)* and *(3.7)* in SIFT) on the FAST-scored-patch of the keypoint and the corresponding ones in upper and lower octaves. Thus, we can obtain a more accurate location in each of these three octaves, which are the 3 red points on $d_{i-1}$, $c_i$ and $d_i$ respectively in the left part of *figure 4.4*.

To avoid resampling, reconsider a $3 \times 3$ patch on each of these three octaves and calculate the new scores using *Eq. (4-13)*. Next, fit a 1D parabola along the refined scores and the scale of the 3 points mentioned above. This step is illustrated in the right part of *figure 4.4*. The scale of the point with maximum score value on the fitted parabola (the green point on *figure 4.4)* is considered the real scale of the keypoint.

Eventually, after interpolating a new position with the scale of the previous 3 points and the final real scale, a keypoint with accurate scale and location is created.

### 4.3.2 Keypoint Description

After keypoints localization, the keypoint descriptors are determined. Before discussing its orientation, the sampling pattern for keypoint description will first be presented.

- ***The Sampling Patterns***

  The sampling pattern is shown in *figure 4.5*. We can see that the samples (denoted by blue dots) are equally distributed in the image patch and form concentric circles with the keypoint, which is quite different from the random sampling in the BRIEF and the ORB. The points should be Gaussian smoothed before sampling the image intensity. To avoid aliasing effects, $\sigma$ of Gaussian kernels is proportional to the distance between the samples and the keypoint. We can denote the intensity of $(x, y)$ in the receptive field filtered with Gaussian kernel with $\sigma$ as $\mathrm{I}(\mathrm{x}, \mathrm{y}, \sigma)$.

  Similar to the BRIEF and the ORB, the selection of sampling-point pairs is required. Here, every point is paired with the other points in the image patch. Thus, we have

$$C_2^N = N(N-1)/2 \qquad [4\text{-}14]$$

  pairs in total in each image patch.

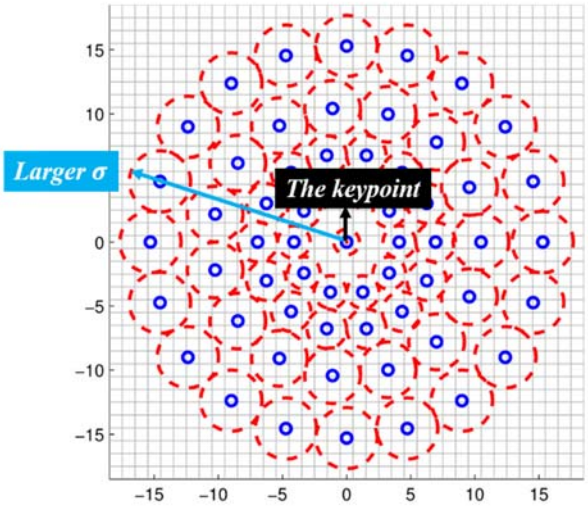


***Fig. 4.5:*** *Illustration of the BRISK sampling pattern with 60 points. The center is the detected keypoint and the other blue points are the sampling points. The red circles with different radius represent different Gaussian kernels. The bigger the circles are, the bigger the sizes of the kernels are. [9]*

- ***Orientation Assignment***

  In the BRISK, local gradients are considered to be the orientations of the keypoints. The local gradients $g(\boldsymbol{p}_i, \boldsymbol{p}_j)$ of a sampling-point pair $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ is defined as

$$g(\boldsymbol{p}_i, \boldsymbol{p}_j) = (\boldsymbol{p}_i - \boldsymbol{p}_j) \cdot \frac{I(\boldsymbol{p}_j, \sigma_j) - I(\boldsymbol{p}_i, \sigma_i)}{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2}, \quad [4\text{-}15]$$

where $I(\boldsymbol{p}_i, \sigma_i)$ and $I(\boldsymbol{p}_j, \sigma_j)$ are the smoothed intensity of the points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ $\sigma_i$ and $\sigma_j$ are smoothing scales of their receptive field respectively.

For the overall keypoint orientation, the set $\mathcal{A}$ of all sampling points is considered:

$$\mathcal{A} = \{(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid i < N \wedge j < i \wedge i, j \in \mathbb{N}\} \quad [4\text{-}16]$$

To choose the most valuable pairs, the BRISK then consider another two subsets constrained in the distance of the pair. It can be classified to short-distance pairings, $\mathcal{S}$, and long-distance pairings, $\mathcal{L}$.

$$\mathcal{S} = \{(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \mathcal{A} \mid \|\boldsymbol{p}_i - \boldsymbol{p}_j\| < \delta_{max} = 13.67t\} \subseteq \mathcal{A} \quad [4\text{-}17]$$

$$\mathcal{L} = \{(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \mathcal{A} \mid \|\boldsymbol{p}_i - \boldsymbol{p}_j\| > \delta_{min} = 9.75t\} \subseteq \mathcal{A} \quad [4\text{-}18]$$

where $t$ is the scale of the keypoints. Thus, the pairs in $\mathcal{L}$ is considered valuable and is the overall gradient, i.e. all the long-distance pairs are used for the keypoint orientation. The orientation is defined as $\mathcal{g}$

$$\mathcal{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \sum_{(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \mathcal{L}} g(\boldsymbol{p}_i, \boldsymbol{p}_j) \quad [4\text{-}19]$$

- ***Building the Descriptor***

  Applying the orientation assignment above, rotate the sampling pattern by

$$\alpha = \arctan2(g_y, g_x) = \arctan(\frac{g_y}{g_x}) \quad [4\text{-}20]$$

around the keypoint. Then, the descriptor is obtained by the intensity comparisons of point pairs in the short-distance set $\mathcal{S}$. The descriptor b is defined as

$$b = \begin{cases} 1, & I(\boldsymbol{p}_j^{\alpha}, \sigma_j) > I(\boldsymbol{p}_i^{\alpha}, \sigma_i) \\ 0, & otherwise \end{cases}, \quad \forall (\boldsymbol{p}_i^{\alpha}, \boldsymbol{p}_j^{\alpha}) \in \mathcal{S} \quad [4\text{-}21]$$

$(\mathbf{p}_i^{\alpha}, \mathbf{p}_j^{\alpha})$ is a short-distance sampling pairs rotated by the derived orientation angle α. Using pairs in $\mathcal{S}$ only is because the intensity of two close points won't differ a lot. Thus, the descriptor of BRISK can be robust to brightness change and it's usually a 512 bit-string which can performs a fast matching.

## 4.4 FREAK: Fast Retina Keypoints

The fast retina keypoint (FREAK) [10] is usually used together with the FAST detector. The FREAK applies circular image patches which are similar to those of the BRISK, but the sampling patterns are close to the distribution of ganglion cells in human's retina. We will briefly introduce human's retina first and then extends it to the FREAK descriptor.

### 4.4.1 Human Retina

The retina is the innermost and light-sensitive layer tissue of an eye. Ganglion cells are the communicating directly with the brain. In *figure 4.6*, the right image shows the four areas in retina: foveola, fovea, parafoveal and perifoveal. The density of ganglion cells reduces exponentially along the radius from perifoveal to foveola. The size of the receptive field, where light influences the response of a ganglion cell, increases with radial distance from the center. The sampling pattern in the FREAK is just like the distribution mentioned above.

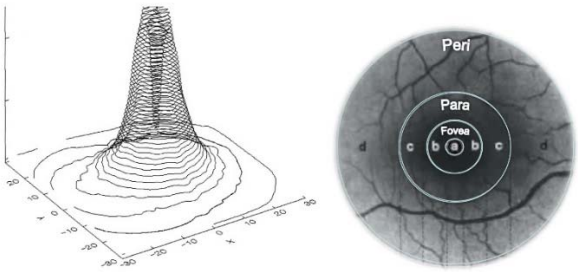


***Fig. 4.6:*** *Illustration of human's retina. The left part shows the density of ganglion cells and the right part shows the four areas in retina: foveola (the center part), fovea, parafoveal and perifoveal. [10]*

### 4.4.2 FREAK Descriptor

- ***Retinal Sampling Pattern***

  To stimulate human's retina, the FREAK uses circular patterns with equally spaced samples on concentric circles. It separates the image patch to 4 parts just like retina. The most inner part is the detected keypoint. As the distance to the keypoint decreases, the density of sampling points goes higher and the smoothing kernel size gets smaller.

  Until now, the FREAK and the BRISK are quite alike. To make it closer to the retina model, the Gaussian kernel size exponentially increases with the radius and it makes the smoothed area (circles in *figure 4.7(b)*) overlapped to capture more information. In FREAK, we called the smoothed areas "receptive fields" because of their similarity in size distribution (see chapter 4.4.1 again for introduction of receptive fields). Comparing (b) to (a), we can see that FREAK's sampling pattern is similar to retina cells.

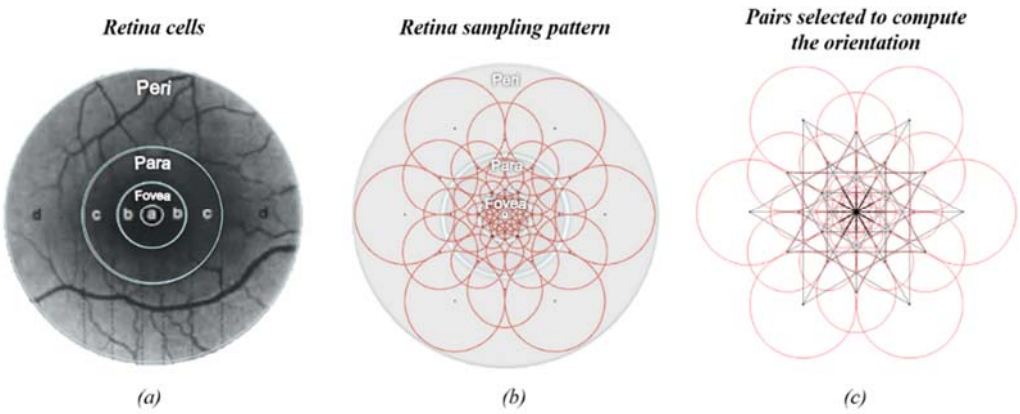


***Fig. 4.7:*** *From left to right are the human retina cells, FREAK sampling pattern and the pairs selection. We can see that (b) and (a) are similar. Each circles in (b) and (c) represents an area, called receptive filed here, smoothed with a Gaussian kernel. The bigger the circle is, the larger the kernel is. The black lines in (c) means sampling pairs [10]*

- ***Orientation Assignment***

For the purpose of rotation invariance, orientation assignment is needed. The FREAK puts sampling points to pairs and sums up their gradients. Unlike the BRIEF and the ORB, the FREAK selects sampling point pairs with symmetric receptive fields with respect to the center. In *figure 4.7(c)*, each black lines represents a sampling pairs and we can see that the lines form a symmetric pattern centered at the keypoint.

Let $\mathcal{G}$ be the set of all the pairs used to compute the local gradients with *M* pairs. The orientation of the keypoint is

$$O = \frac{1}{M}\sum_{P_o \in \mathcal{G}} \left(I\left(P_a^{r_1}\right) - I\left(P_a^{r_2}\right)\right) \frac{P_o^{r_1} - P_o^{r_2}}{\left\|P_o^{r_1} - P_o^{r_2}\right\|} \qquad [4\text{-}22]$$

where $P_o^{r_1}$ is the 2D vector of the spatial coordinates of the center of the first receptive field and $P_o^{r_2}$ is of the second. We can think of the receptive fields as the different scaled images in one octave of SIFT because they are both images (or patches) smoothed with Gaussian in different size. Then, just like the process in SIFT (*Eq. (3-5)*), difference of Gaussian (DoG) is applied on the receptive fields. Then, the following binary test can be performed on the DoG of receptive fields.

- ***The Binary Test***

Here, the descriptor $\mathcal{F}$ is formed by a sequence of the one-bit DoG.

$$\mathcal{F} = \sum_{0 \le a < N} 2^a T(P_a) \qquad [4\text{-}23]$$

where $P_a$ is a pair of receptive fields, *N* is the desired size of the descriptor. *T* is a binary test defined as

$$T(P_a) = \begin{cases} 1 \; if \; I(P_a^{r_1}) - I(P_a^{r_2}) > 0 \\ 0 \; otherwise \end{cases} \quad [4\text{-}24]$$

$I(P_a^{r_1})$ is the smoothed intensity of the first receptive field of the pair $P_a$ and $I(P_a^{r_2})$ denotes that of the second area. Finally, the binary descriptor T is constructed, but there's still too much elements, so non-maxima suppression is needed.

- ***Non-maxima Suppression***

  Descriptors are now constructed by a binary test. However, some pairs might not be useful to efficiently describe an image. Thus, the following algorithm is helpful to get the best pairs:

  1. *Create a matrix D with each row representing the descriptor for each keypoint.*
  2. *Compute the mean of each column and order the columns with respect to the highest variance.*
  3. *A mean of 0.5 results in the highest variance of a binary distribution. Keep the column whose mean > 0.5 and iteratively add remaining columns having low correlation with the selected columns.*

  In average, the first 512 pairs are the most relevant and hold the best performance. Finally, the FREAK descriptor is completed.

Now, the introduction of different keypoints comes to an end. Let's take a brief review of the content covered in chapter 2 to 4. In chapter 2, we introduce some basic corner detection method. Moravec's detector determine the corner by measuring the variation of intensity corresponding to shifts in an image patch. Then Harris corner detector improves Moravec's method and detect a corner by measurement of corner response *R*. FAST aims to present a really fast and easy way. It compares the intensity of the points which are on a circles center at a potential keypoint.

The methods above are simple, but aren't invariant when some transformations occur. Thus, chapter 3 and 4 have both presented some keypoints approaching to be robust to scale, rotation, illuminance and any other transformations happening in real life. Furthermore, they all construct descriptors to enhance the accuracy in matching. *Table 4.1* gives an overview of these keypoints. Those in upper region are with non-binary descriptors (chapter 3) and those in lower one are with binary descriptors (chapter 4). BRIEF isn't on the table because it isn't considered a kind of keypoints and is actually a fundamental binary descriptor. PCA-sift isn't there either because its process is almost same as SIFT. It performs PCA to reduce the dimension of the outputs of SIFT's descriptors.

***Table. 4.1:*** *It is an overview of keypoints introduced in chapter 3 and 4. The upper region shows those with non-binary descriptors and the lower, binary.*

| | SIFT | SURF |
|---|---|---|
| Scale space | Scale pyramid<br>Each octave holds different image size and blurred images. | Scale pyramid<br>Image sizes remain the same but box filter's sizes change. |
| Keypoint Detection | Sub-pixel localization on extrema of DoG | Sub-pixel localization on maxima of $det(\mathcal{H}_{approx})$ |
| Orientation Assignment | The peak of gradient of gradient histogram | Sum of Harr wavelet responses in both x & y direction |
| Descriptor Dimension | A 128D vector | A 64D vector |
| Image Patch for Descriptor | Divide a $16 \times 16$ window centered at a keypoint to 16 $4 \times 4$ sub blocks | Divide a $20s \times 20s$ window centered at a keypoint (s: scale of the keypoint) 16 $5s \times 5s$ sub blocks |
| Descriptor Construction | Orientation of each sub block | Sum of Harr wavelet responses in both x & y direction and their absolute value of each sub block |

| | ORB | BRISK | FREAK |
|---|---|---|---|
| Scale space | Same as SIFT | - | - |
| Sampling pattern | Points on a circle center at a keypoint | Uniform distributed points form concentric circles | Similar as BRISK, but more alike to human's retina. |
| Keypoint Detection | oFAST<br>(FAST + Orientation) | FAST 9-16 | Usually use FAST |
| Orientation Assignment | Intensity Centroid | Intensity gradient of long-distance sampling pairs | Intensity gradient of sampling pairs |
| Descriptor Dimension | A 256 bit string | Usually a 512 bit string | Usually a 256 or 512 bit string |
| Sampling pair for Descriptor | Random pairs<br>$\sim$ i. i. d. Gaussian$(0, \frac{1}{25}S^2)$ | Every point is paired with the other points of the sampling pattern. | Symmetrically chosen with respect to the center |
| Descriptor Construction | rBRIEF<br>(Rotation-Aware BRIEF) | Intensity binary test on sampling pairs with sampling pattern rotated to its orientation | Bit strings formed by a sequence of one-bit DOG |

# Chapter 5 Keypoint Matching

In the previous two chapters, we focus on different methods to extract features and obtain descriptors of keypoints. We can see that a descriptor consists of the information of the neighboring points. Thus, the comparison between descriptors can lead to a better matching result than comparing only the intensities of keypoints. In other words, features introduced in chapters 3 and 4 are more often applied to template matching than those in chapter 2.

In this chapter, matching ways of non-binary and binary descriptors are discussed. Last but not least, RANSAC, which is considered a helpful tool, is also introduced.

## 5.1 Matching for Non-binary and Binary Descriptors

To match two keypoints, the most direct way is to calculate the distances of their descriptors. The pairs with the minimal distances are considered the matched keypoints.

For non-binary descriptors, Euclidean distance is used.

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad [5\text{-}1]$$

As for binary descriptors, the Hamming distance is applied. The Hamming distance represents the number of bits different in the two bit-strings which can measure how similar the two descriptors are.

In order to reduce the mismatching probability, a threshold can be set to constrain the minimal distance. Consider two different images, A and B, we want to match the features from A to B. If the minimal distance of a keypoint on A and all the keypoints on B is larger than the threshold, then there is no matching point to this feature.

## 5.2 The Application of RANSAC

The random sample consensus (RANSAC) [11] is an iterative algorithm for finding a mathematical model from a set of data. A simple example shown in figure 5.1 is to fit a set of 2D data by a line. The red line is the resulting fitting line. The blue points which are near the line are called inliers and other points in black are considered outliers.

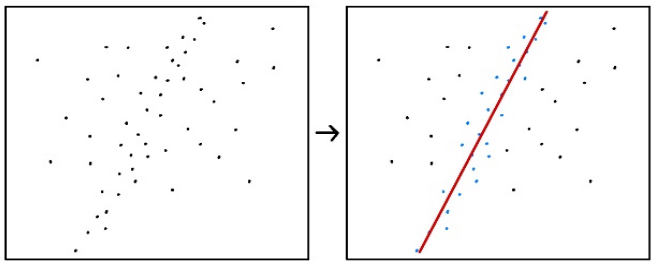

***Fig. 5.1:*** *It's the simplest example of RANSAC. The red line is the output fitting line.*

Consider $N$ data points. The algorithm of RANSAC is shown as follows:

1. Choose $n$ samples in the data randomly.
2. Fit a mathematical model with these $n$ samples.
3. For each of other $N - n$ points, compute their distances to the fitting model and count the number of inlier points, $c$.
4. Run the above steps for $k$ times and output the model with the largest $c$.

For the application in matching keypoints, the RANSAC algorithm is performed on the differences of $x$-coordinate and $y$-coordinate of the potential matching points. Here, the objective model is the fitting line. Finally, we preserve the inliers of the output model to be the final matching pairs. Figure 5.2 shows a comparison of SIFT feature matching without the RANSAC and with the RANSAC, respectively. We can see that applying RANSAC leads to a better result.

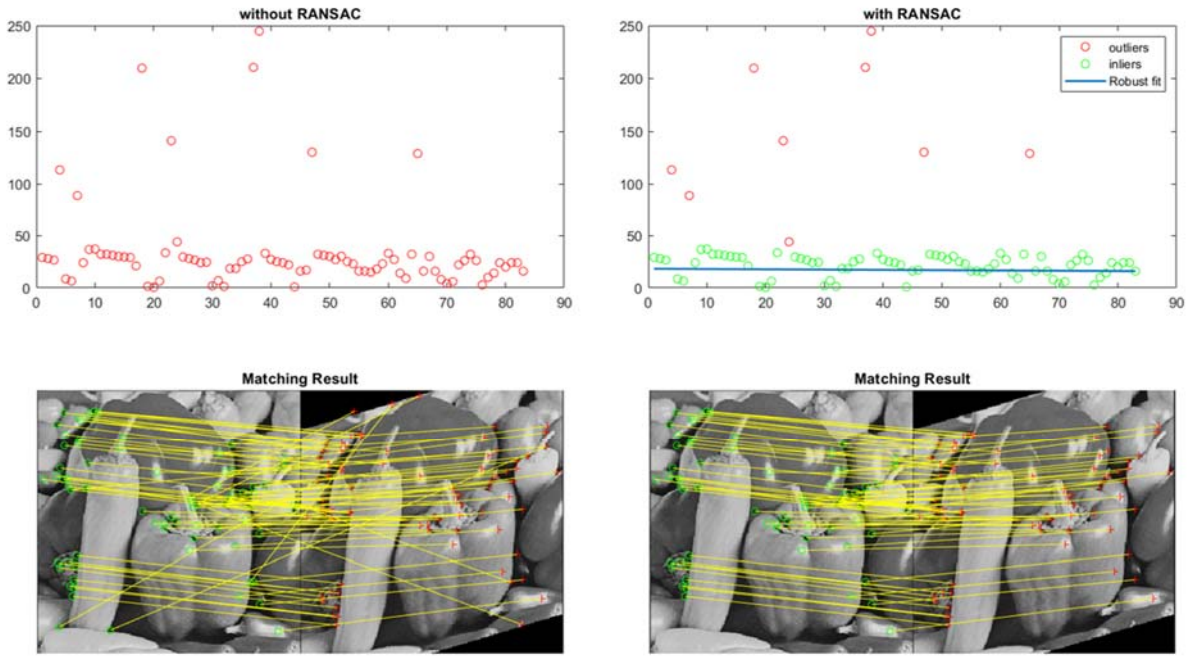


***Fig. 5.2:*** *Comparison of SIFT feature matching without RANSAC and with RANSAC. The red points are outliers and the green ones are inliers. We can see that after using RANSAC, the results become better.*

# Chapter 6 Stimulation Results

In this chapter, we take advantage of *Computer Vision Toolbox* in *MATLAB* to stimulate the matching results of the SURF, the ORB, the BRISK and the FREAK descriptor with the FAST detector. A variety of datasets are applied. In the first section, the results of some simply affine transformed images are shown, while in the latter sections, some more complicated cases will be discussed. Results of viewpoint changed, zoomed and rotated, brightness changed, JPEG compressed and blurred images will then be presented. To make the resulting matching lines clearer, the number of matching points is controlled to be less than 30 if possible. The datasets in the second to the last sections are all from [17].

## 6.1 Affine Transform

***Fig 6.1 Lena:*** *The leftmost one is the original image. Then, from the left to right are images: rotated by* $30^{\circ}$ *clockwise, scale to* $1.5M \times 1.6N$ *(M, N are the original length and width) and sheared along y-axis.*

*Figure 6.1* are the test images in this section, the left most image is the original image. We will match its keypoints to some affined transformed images. *Figure 6.2 and 6.3 are the results.*

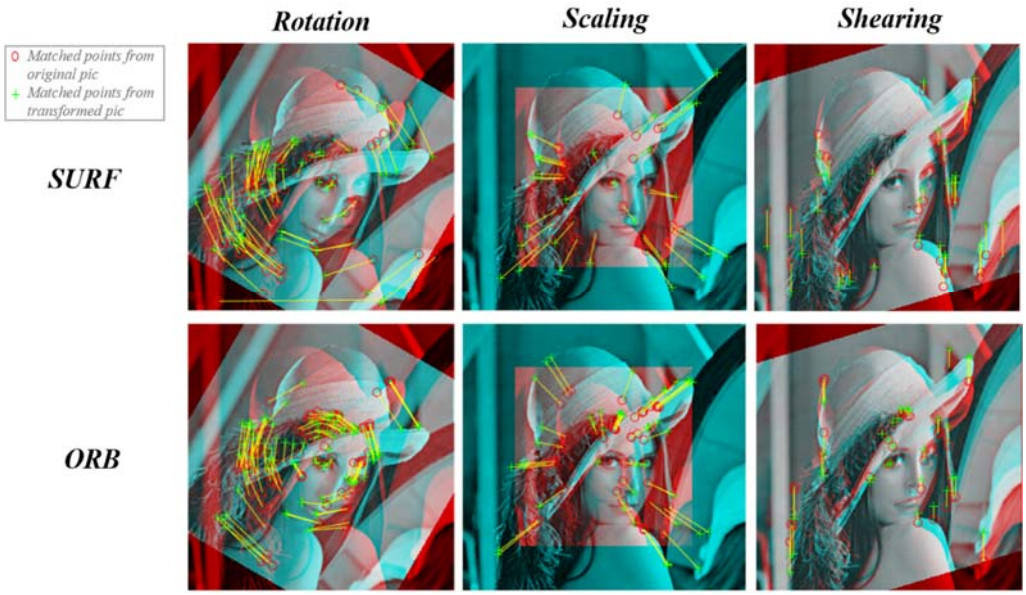


***Fig 6.2:*** *The results of SURF and ORB. Red images in the background of each results are the original images and the blue ones are the transformed. The yellows lines point to which points are matched with.*

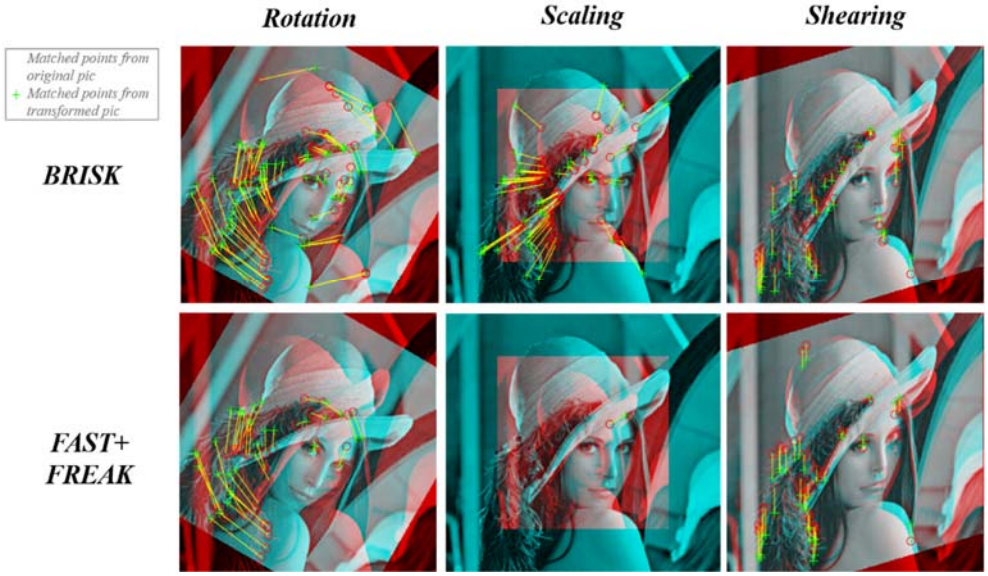


***Fig 6.3:*** *The results of BRISK and FAST+FREAK. Red images in the background of each results are the original images and the blue ones are the transformed. The yellows lines point to which points are matched with.*

## 6.2 Viewpoint Change

***Fig 6.4 Graffiti:*** *The one on the upper-left corner is the original image and the followings are the same graffiti but taken from different point of view.*

After the stimulation of simple transforms, we start to perform experiments on the more complicated cases. In this section, the goal is to see whether the features can be matched correctly even the pictures are taken in different viewing angle. *Figure 6.4* shows our original data and *figure 6.5* presents the stimulation results.

In these 6 test images, a latter one can be considered a more difficult case. See *figure 6.4*, no matter which method is chosen, as the viewing angle becomes steeper, the matching lines in the result get messier. Starting from the fourth experiment, most of the keypoints cannot be matched properly. However, in such a severe challenge, the ORB performs relatively well. We can see that in the top four matching experiment, it can match properly. Even in the fourth one that other methods have already failed, it still has a high correctness.

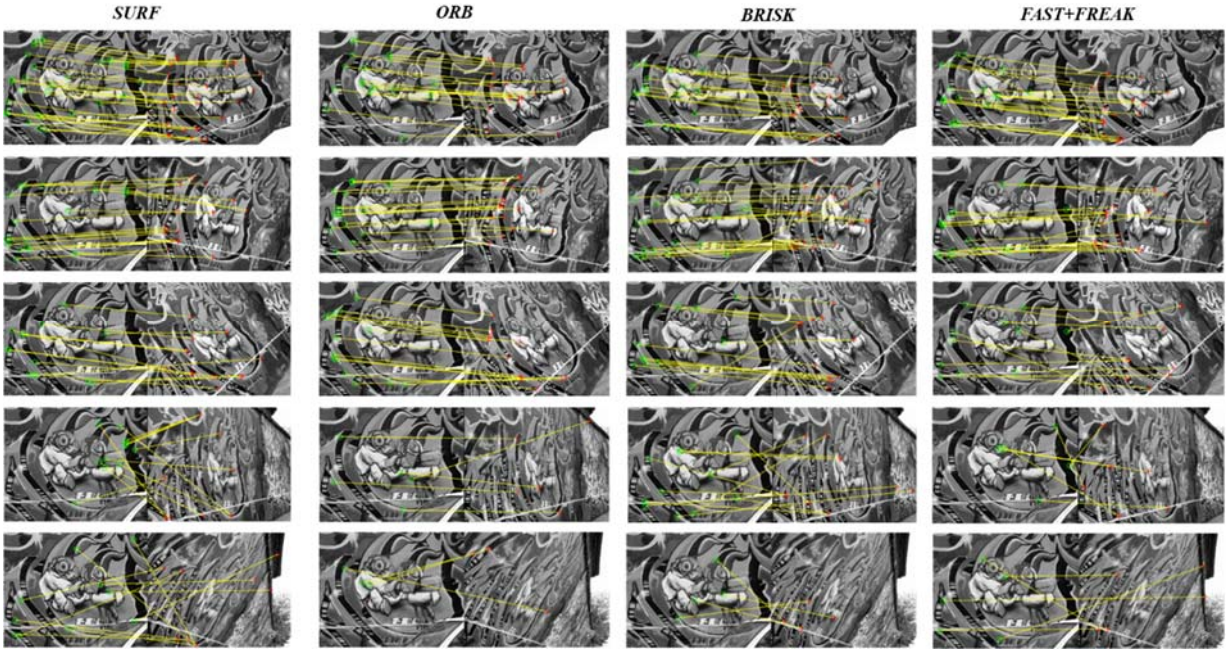


***Fig 6.5:*** *The matching results of Graffiti. In this case, the ORB performs the best.*

## 6.3 Zoom and Rotation

***Fig 6.6 Boat:*** *The one on the upper-left corner is the original image and the following ones are the same landscape of boat but are rotated along with zoom.*

In this section, we make the rotation case a bit harder. Each transformed image is not only rotated, but is also zoomed in. It is the most irregular transform among our stimulations. The image in the upper left corner of *figure 6.6* is the original image and the followings are images with different rotation degrees and different scales of zooming. See *figure 6.7*, the ORB also has the best performance here. In these five experiments, it matches almost perfectly. Furthermore, the SURF matches well when the rotation angle is small, but when the image is rotated by a large degree, e.g. the third and the fifth results, the performance becomes worse.

The BRISK and FAST+FREAK have the similar problem. However, for the fourth experiment, which only performs zooming, the SURF, the ORB, and the BRISK keypoints perform well.

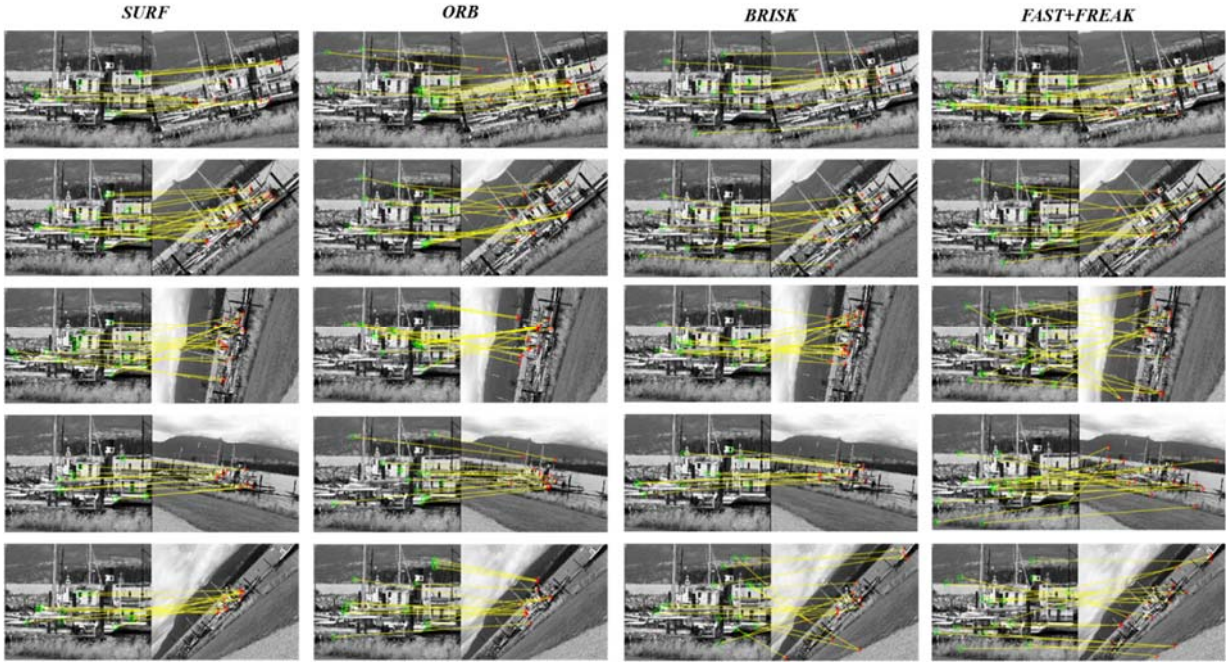


***Fig 6.7:*** *The matching results of image Boat. In this case, the ORB performs the best.*

## 6.4 Brightness Change

***Fig 6.8 Leuven:*** *The one on the upper-left corner is the original image and the followings are the darken ones.*

In this section, we investigate whether the brightness change affects the matching result. *Figure 6.8* shows our data here and we can see that the images go darker and darker. As for the matching results in *figure 6.9*, it seems that most of the features are perfectly matched except the SURF. It shows that the SURF is less robust to the brightness change.

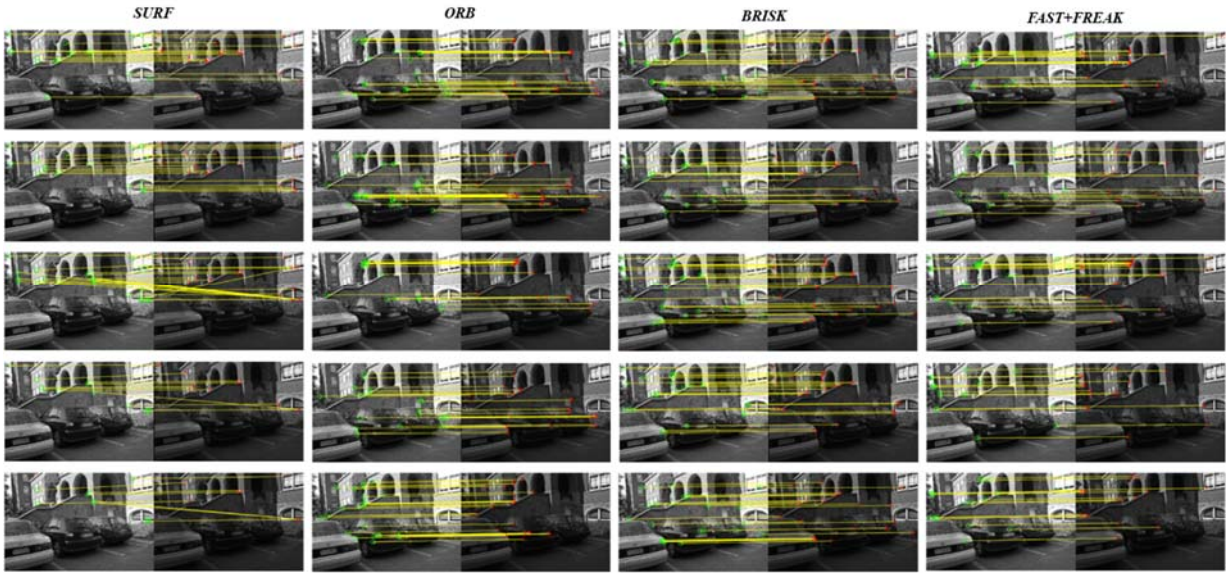


***Fig 6.9:*** *The matching results of the Leuven image. In this case, most of the methods perform well.*

## 6.5 JPEG Compression

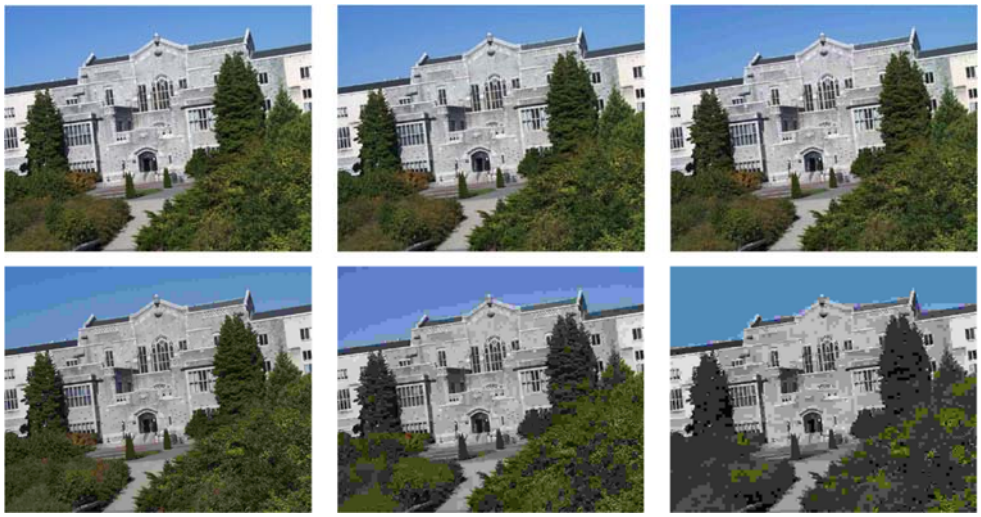

***Fig 6.10 Ubc:*** *The sub-image on the upper-left corner is the original image and the followings are the ones applied JPEC Compression with different quantization levels.*

In this section, we aim to see the matching results of different quantization levels of JPEC compression. The image on lower right corner in *figure 6.10* misses most of the details of the original image. In this case, all the features of the corresponding methods are matched perfectly (see *figure 6.11*).

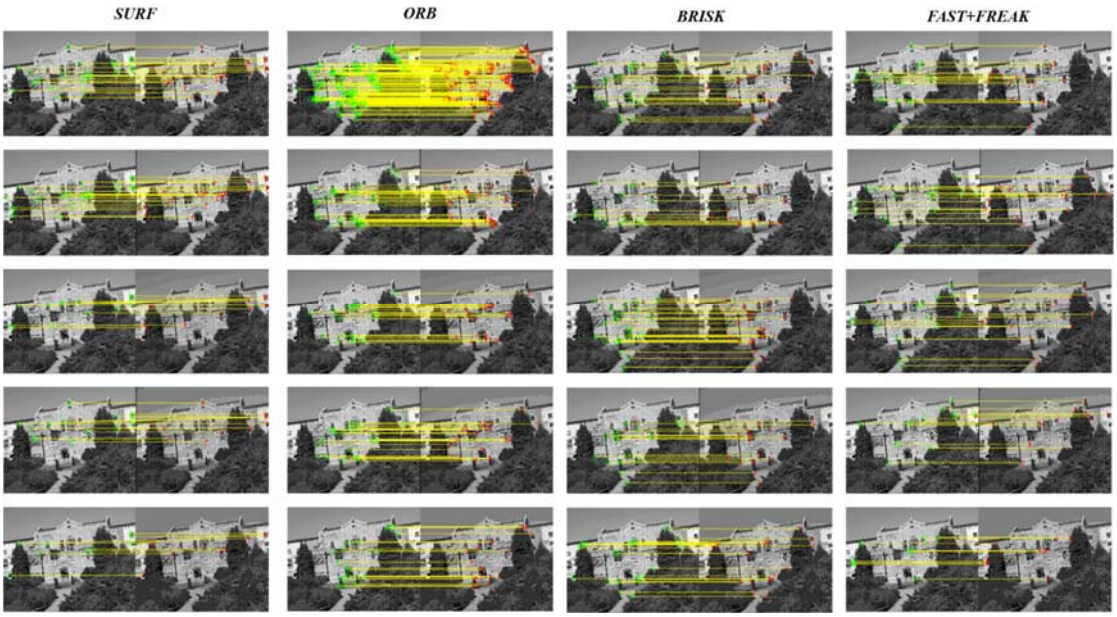


***Fig 6.11:*** *It's the matching results of the Ubc image. In this case, most of the methods perform well.*

## 6.6 Blurring

***Fig. 6.12 Bikes:*** *The one on the upper-left corner is the original image and the followings are its blurred results.*

In the final section of this chapter, we perform some experiments on the blurred images. From upper left to lower right, the images are getting blurring in each stimulation. The test images and the matching results are all shown in *figures 6.12 and 6.13*, respectively. In this case, the ORB and the BRISK keypoints can be matched perfectly.

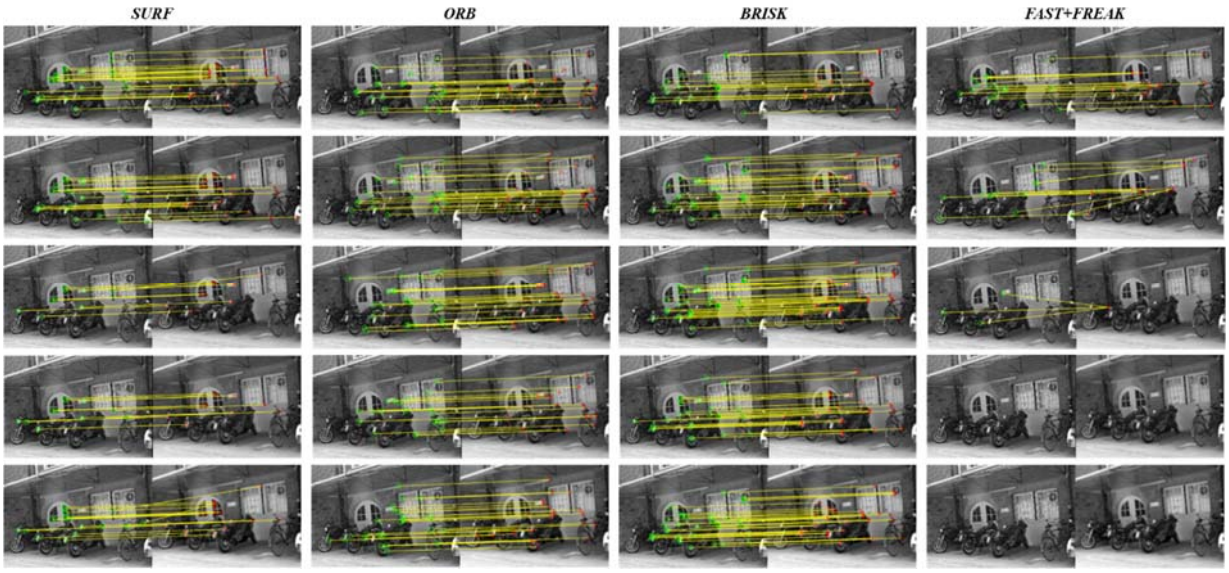


***Fig. 6.13:*** *The matching results of image Bikes. In this case, FAST+FREAK is the least sensitive in blurring. The ORB and the BRISK are seem to be perfectly matched.*

According to our stimulations, we can conclude that rotation and viewpoint change could affect the performance of keypoint matching. By contrast, most of existing keypoint matching methods are robust to scaling, shearing, brightness change, JPEG compression, and blurring. Among the four kinds of features that we have compared, the ORB seems to have a more stable performance in each case.

# Chapter 7 Conclusion

In early times, keypoints can only be corners or some points on edges. With the development of the SIFT, the SURF, the ORB, the BRISK, and the FREAK, keypoints can definitely describe some properties of small parts of an image. In this tutorial, we can see that there are a variety of ways to extract keypoints and to construct descriptors. Some methods aim to reduce the complexity of previous research, while some approach is more robust to rotation, noise, illuminance, etc. From our experimental results, we can see that each kind of features has its own pros and cons. We can choose the most appropriate features depending on which kind of data set it is. As mentioned, there are lots of applications of feature matching. In the future, extracting features with efficient ways meanwhile remaining its performance on matching will still be an interesting issue.

# Reference


[1] C. Harris, M. Stephens, A combined corner and edge detector, in: *Proceedings of the Alvey Vision Conference, 1988, pp. 147-151*

[2] E. Rosten, T. Drummond, Machine Learning for high speed corner detection, in: *In European Conference on Computer Vision, volume 1, 2006*.

[3] D. Lowe, Object recognition from local scale-invariant feature, in: *ICCV, 1999*.

[4] D. Lowe, Distinctive image features from scale-invariant keypoints, in: *IJCV 60 (2) (2004) 91-110*.

[5] Y. Ke, R. Sukthankar, PCA-SIFT: a more distinctive representation for local image descriptors, in: *CVPR, issue 2, 2004, pp. 506-513.*

[6] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, SURF: Speed-Up Robust Features, *Computer Vision and Image Understanding 10, 346-359, 2008*.

[7] M. Calonder, V. Lepetit, C. Strecha, P. Fua, BRIEF: Binary robust independent elementary features, in: *In European Conference on Computer Vision, 2010*.

[8] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, in: *IEEE International Conference on Computer Vision, 2011*.

[9] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints, in: *IEEE International Conference on Computer Vision, 2011*.

[10] Alahi, A., Ortiz, R., & Vandergheynst, P. (2012, June). Freak: Fast Retina Keypoint. in: *IEEE conference on computer vision and pattern recognition (pp. 510-517), 2012.*

[11] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. in: *Communications of the ACM, 24(6):381–395, 1981*.

[12] H. P. Moravec, Obstacle Avoidance and Navigation in the Real World by Seeing Robot Rover, *Doctoral dissertation, Stanford University, 1980*.

[13] H. Kamel, Y. Chahir. SIFT Detectors for Matching Aerial Images in Reduced Space. in: *4th International Conference on Computer Integrated Manufacturing CIP, 2007*

[14] N. Hatami, Y. Gavet, J. Debay. Bag of Recurrence Patterns Representation for Time-series classification. in: *Pattern Analysis and Applications. 22. 10.1007/s10044-018-0703-6, 2018*

[15] *https://www.eecs.tu-berlin.de/fileadmin/fg144/Courses/10WS/pdci/talks/sift-feature_extraction.pdf*

[16] *https://www.gettyimages.hk/%E5%9C%96%E7%89%87/albert-einstein*

[17] *http://kahlan.eps.surrey.ac.uk/featurespace/web/data.htm*

[18] Brown, Matthew A. and David G. Lowe. Invariant Features from Interest Point Groups. in: *BMVC, 2002*.