Chapter 2
Classical Local Descriptors

Abstract Classical local descriptors refer to those were proposed many years ago but have a profound influence on the development of local image description as well as related applications. Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Feature (SURF) are the two widely used descriptors in computer vision. Especially for SIFT, it is an extremely popular solution to various applications, ranging from object recognition, image retrieval, to structure from motion, etc. While for SURF, it is a first and predominant choice for those applications requiring fast or near real-time image matching until the very recent flourish of binary descriptors. Another classical local feature is Local Binary Pattern (LBP) proposed in the 1990s. Along with many variants, LBP has been ubiquitous in texture classification and many face-related tasks, e.g., face recognition, face detection, and facial expression recognition. Because of their popularity, we choose to introduce them in detail in this chapter.

Keywords SIFT · SURF · LBP · Scale space representation · Float type descriptor

2.1 Scale-Invariant Feature Transform (SIFT)

Scale-Invariant Feature Transform (SIFT) [9] is one of the most successful local descriptors, and it has been widely used in various vision tasks, such as image classification, image retrieval, image registration, pose estimation, to name a few. Basically, SIFT includes both feature detection and feature description. According to the context and application, sometimes SIFT only refers to its method for feature description, i.e., SIFT feature descriptor. For clarity, the term of "SIFT keypoint" is usually used to indicate its detected feature.

Generally speaking, SIFT is a scale-invariant method, and at the same time keeps robustness to many other image transformations, including in-plane rotation, small-to-medium viewpoint changes (caused by out-plane rotation of camera), image blur, illumination, and noise. The scale invariance and its robustness to image rotation are achieved by SIFT feature detector, which can detect keypoints with their scales and reference orientations. While the robustness to other transformations is obtained by the way of constructing a SIFT descriptor.

B. Fan et al., *Local Image Descriptor: Modern Approaches*,
SpringerBriefs in Computer Science, DOI 10.1007/978-3-662-49173-7_2

2.1.1 Scale Space Representation in SIFT

In order to achieve scale invariance, SIFT operates on a Gaussian scale space to detect keypoints and construct their local descriptors. Gaussian scale space of an input image is defined as a series of images obtained by convolving the input image with gradually increased Gaussian variances. Given an input image $I(x, y)$, its Gaussian convolved image can be represented by:

$$L(x, y, \sigma) = I(x, y) * G(\sigma) \tag{2.1}$$

where $G(\sigma)$ denotes a Gaussian filter with the standard variance of σ. Thus, a set of convolved images $\{L(x, y, \sigma_i), i = 1, 2, \ldots, n\}$ is used to denote the Gaussian scale space of the image I. In the implementation of SIFT, σ is increased by a constant factor k, i.e., $\sigma_{i+1} = k\sigma_i$.

In order to detect keypoints which are robust to scale changes and with a high repeatability, SIFT takes the extrema in a DoG (Difference of Gaussian) scale space as the initial keypoints, and then with some refinement and adjustment to determine the final keypoints (will be described in the next section). For this purpose, a DoG scale space is constructed on the basis of the Gaussian scale space, by subtracting adjacent image scales:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.2}$$

The computation time of Gaussian convolution is positively related to the size of image as well as the size of used Gaussian kernel. Therefore, for efficiency, $L(\sigma_{i+1})$ is computed by convolving $L(\sigma_i)$ with $\sigma'_{i+1} = \sqrt{\sigma^2_{i+1} - \sigma^2_i}$, instead of directly convolving I with σ_{i+1}. On the other hand, SIFT also uses an octave-based structure for scale space representation. More specifically, images in a scale space are divided into octaves, each of which doubles σ, i.e., scale. In each octave, it further divides the scale space into s layers, so, $k = 2^{1/s}$ as $k^s = 2$. As we will describe later, SIFT searches for extrema in the DoG scale space along three dimensions for keypoint detection. Therefore, it needs $s + 2$ DoG images to cover a complete octave. As shown in Fig. 2.1, the two additional layers are used for keypoint detection in the first and the last layer. Accordingly, $s + 3$ Gaussian blurred images are required to generate these $s + 2$ DoG images. After one octave has been processed, the $(s + 1)$th image in this octave is downsampled by a factor of 2 to generate the first image in the next octave. In this way, the whole octaves will cover a set of scales increased by a constant factor, as can be seen in Fig. 2.1.

According to several experiments, Lowe suggested to set $\sigma = 1.6$ [9]. Meanwhile, in order to get more stable keypoints, he also proposed to double the original image by a factor of 2 and use it to construct the first octave of the scale space.

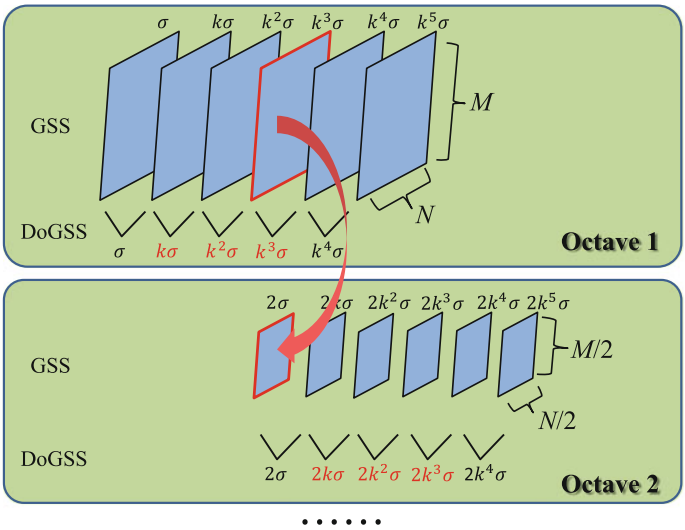

Fig. 2.1 The scale space representation implemented in SIFT when $s = 3$. DoG scale space (*DoGSS*) is generated by subtracting adjacent images in the Gaussian scale space (*GSS*). The *red* values indicate the scales that will be used for keypoint detection by 3D extrema search. To make these scales consistently differ by a factor of k, it has to generate $s + 2$ images in each octave of DoGSS and $s + 3$ images in GSS. See the text for details

2.1.2 Keypoint Detection

To detect scale-invariant keypoints, SIFT first searches for the local maxima and minima of $D(x, y, \sigma)$ by comparing each pixel to its 8 neighbors in the current layer and the 18 neighbors in the above and below layers. Then, non-maximum suppression is used to filter out those close enough but unstable local extrema, obtaining a set of points in discrete positions and some levels of predefined scales. The next step is to estimate the accurate positions (up to subpixels) and scales. Meanwhile, it is important to remove some keypoints with low contrasts and those poorly localized along edges, because both of these cases will result in unstable keypoints.

The accurate localization of a keypoint $X_0 = (x, y, \sigma)$ is obtained by fitting a 3D quadratic function around the local area of X_0 and taking the interpolated position of its extremum. To this end, we first have to shift the origin of the scale response function, i.e., DoG response, to X_0. Then, the Taylor expansion is applied to the shifted $D(x, y, \sigma)$:

$$D(X) = D(X_0) + X^T \frac{\partial D}{\partial X}(X_0) + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2}(X_0) X \tag{2.3}$$

To get the extremum of $D(X)$, we can compute the derivative of $D(X)$ and set it to zero. In this way, we obtain the following equation:

$$\frac{\partial^2 D}{\partial X^2}(X_0)\Delta X = -\frac{\partial D}{\partial X}(X_0) \tag{2.4}$$

ΔX is the offset of the extreme point to the original point X_0, so the refined keypoint is localized at $X = X_0 + \Delta X$. It is worth to note that if the offset ΔX is larger than 0.5 in any dimension, it implies that the extremum lies closer to a neighboring point of X_0. In this case, X_0 is changed to this neighboring point and the above procedure of getting offset is repeated, until all the dimensions in the offset are less than 0.5.

Substituting $X = X_0 + \Delta X$ into Eq. (2.3), we can obtain its DoG value:

$$D(X) = D(X_0) + \frac{1}{2}\Delta X^T \frac{\partial D}{\partial X}(X_0) \tag{2.5}$$

The extremum with a value of $|D(X)|$ less than 0.03 (the pixel value is assumed in the range of [0,1]) is discarded due to its low contrast (sensitive to noise).

Since the DoG function will have a strong response along edges, this kind of extremum is unstable to noise and usually has a large localization error. Meanwhile, points localized along the edges usually have less distinctive local appearance, which will make extracting discriminative descriptors for them a difficult task. For these reasons, they have to be discarded.

Except for the extreme response in the DoG scale space, these kind of points also have a large principal curvature along an edge and with a small one along the prependicular direction. These two curvatures are proportional to the eigenvalues of a Hessian matrix H, which is computed at the scale and location of the keypoint as:

$$H = \begin{bmatrix} Dxx & Dxy \\ Dxy & Dyy \end{bmatrix} \tag{2.6}$$

where the derivatives Dxx, Dxy, Dyy are computed by taking the differences of neighboring sample points in the scale space.

Directly computing the eigenvalues of H is a bit slow, fortunately, here we only concern the ratio of its two eigenvalues, but not the concrete eigenvalues. Suppose the two eigenvalues are both positives and as λ and $r\lambda$ with $r \geq 1$, they satisfy:

$$tr(H) = \lambda + r\lambda = (1 + r)\lambda \tag{2.7}$$

$$Det(H) = \lambda \times r\lambda = r\lambda^2 \tag{2.8}$$

We can further obtain:

$$\frac{tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r} \tag{2.9}$$

Since $r \geq 1$, $\frac{tr(H)^2}{Det(H)}$ is monotonic increased along with r. As a result, we just need to set a threshold on $\frac{tr(H)^2}{Det(H)}$ to discard the point with large r, which corresponds to

a large principal curvature and a small one. In the implementation of SIFT, r is set to be 10. In case that $Det(H)$ is negative (it will have a negative eigenvalue), the keypoint is discarded as not being an extremum, but this rarely happens.

The final step for keypoint detection is to assign an orientation to each keypoint. Note that this step is optional. If the task does not have to deal with images with rotations, this step can be skipped and obtain a better performance both in efficiency and accuracy. Otherwise, this step is useful to achieve rotation invariance. The orientation of a keypoint is computed from a circular neighborhood depending on its scale. Specifically, given a keypoint (x, y, σ), we first take out the Gaussian blurred image closest to σ from the scale space. All the following operations are conducted in this image so as to be scale invariant. Then, a histogram of gradient orientations (360 degree is quantized to 36 bins) is computed from a local circular region around the keypoint. The radius of this circular region is set to be 4.5σ. Besides the gradient magnitude of each sample point in this region, it is also weighted by a Gaussian function with 1.5σ as standard variance when adding it to the histogram. Finally, the orientation corresponding to the highest peak in this histogram is taken as the orientation of this keypoint. For other peaks that are within 80 % of the highest peak, their corresponding orientations are also taken as the orientations of the keypoint. Therefore, for one keypoint, it may split into several keypoints, which differ only by their orientations. It is worth to point out that when computing the orientation corresponding to a peak, a parabola is fitted to the 3 histogram bins centered at the peak to interpolate the peak position for better accuracy.

2.1.3 Feature Description

In the keypoint detection, each keypoint is detected along with its position, scale, and orientation. These parameters can be used to construct a local descriptor to describe this keypoint in a scale and rotation-invariant manner. Briefly speaking, as shown in Fig. 2.2, SIFT descriptor is constructed by dividing the local region defined by these parameters into 4×4 grids, then computing histograms of gradient orientations in these grids, and finally concatenating these histograms together which is further normalized to an unit vector as the descriptor.

First, similar to the computation of keypoint orientation, we have to take out the Gaussian blurred image in the scale space according to the scale of the keypoint. SIFT descriptor is constructed in this image to assure the scale invariance. Then, a local region with 16×16 sample points around the keypoint location is taken out from the image according to the keypoint scale σ. For each sample point, its gradient magnitude and orientation are computed. Note that to achieve rotation invariance, the coordinates and gradient orientations of these sample points are rotated relatively to the keypoint orientation. A Gaussian weighting function with standard variance as $1.5W$ (W is the width of the local region) is used to weight the magnitude of each sample point in order to put more emphasize on those sample points near to the keypoint.

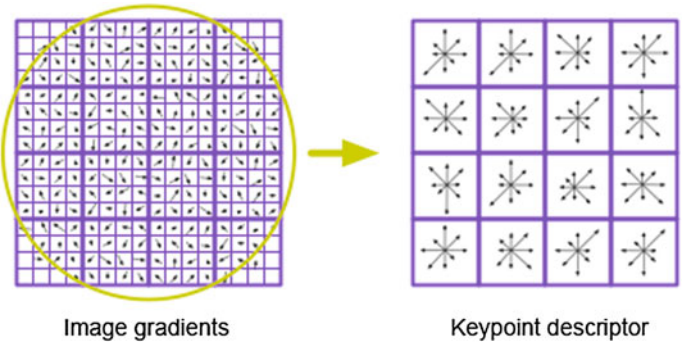

Fig. 2.2 SIFT descriptor is computed by accumulating and concatenating gradient orientation distributions in 4×4 spatial grids. Gradient orientation is weighted by its magnitude and a Gaussian weighting function centered at the keypoint. Reprinted from Lowe [9], with kind permission from Springer Science+Business Media

Meanwhile, in order to be robust to shift in these sample points as well as encoding some spatial information to improve the distinctiveness, the 16×16 local region is further divided into 4×4 grids. In each grid, a histogram of gradient orientations is computed on the basis of gradient magnitudes and Gaussian weights. All histograms are concatenated together to form a 128-dimensional descriptor. To handle sudden illumination changes, a two-step normalization is used. The 128 descriptor is first normalized to unit length. Then any element larger than 0.2 is clipped to 0.2. The clipped vector is renormalized to unit length as the final SIFT descriptor.

From the above procedure, we can see that SIFT actually accumulates a 3D histogram of spatial locations and gradient orientations, weighted by a Gaussian function and gradient magnitude. To improve its robustness and avoid boundary effects, soft assignment is used. In this way, a sample point contributes to 8 neighboring bins in the 128-dimensional SIFT descriptor according to its distance to the center of each bin.

It is worth to point out that the famous Histograms of Oriented Gradients (HoG) proposed by Dalal and Triggs [4] for pedestrian detection shares a similar basic idea to SIFT descriptor. Both of them resort to accumulate oriented gradient responses, which are weighted by gradient magnitude, on different spatial areas in the described region. The difference is that HoG uses much more overlapping cells to collect histograms of oriented gradients, which are normalized according to the 'energy' in a larger region (block). The HoG detection window/described region is divided into many blocks to obtain a discriminative representation. Nowadays, HoG has been a standard technique of feature extraction for object detection, not merely to pedestrian detection.

2.2 Speeded Up Robust Feature (SURF)

Although SIFT is discriminative and robust to many geometric and photometric transformations, it is slow to compute, which limits its application to some extent. To alleviate this problem, Bay et al. [2] proposed a fast alternative to SIFT by modifying SIFT to make it adaptive to integral image, which can be computed very fast. Their method is called Speeded Up Robust Feature (SURF). Its algorithmic pipeline is similar to that of SIFT, which contains mainly three steps: (1) scale space representation of the input image; (2) detect SURF keypoints (interest points) in the scale space; (3) assign orientations to keypoints and construct their SURF descriptors. We will elaborate each of these steps in the following sections.

2.2.1 Integral Image

Before formally introducing these algorithmic steps of SURF, we start with a brief introduction of the integral image technique, which is at the core of SURF. By definition, the entry of an integral image $I_{\sum}$ at a location (x, y) is the sum of all pixels in the input image I within a rectangular region formed by the origin (left top of an image) and (x, y). Therefore, given an image I of size $m \times n$, its integral image $I_{\sum}$ can be computed as:

$$I_{\sum}(x, y) = \sum_{u=0}^{x} \sum_{v=0}^{y} I(u, v)$$

The advantage of integral image lies in the fast computation of the sum of the intensities over any upright, rectangular region. It can be shown that once the integral image has been computed, for any upright, rectangular region, it only requires three additions to obtain its summed intensities, no matter what the size of the region is. This is extremely useful when one has to frequently obtain intensity sums over regions and has to operate on large regions, which are the cases that SURF faces.

2.2.2 Scale Space Representation in SURF

In order to detect interest points along with different scales, one has to resort to a scale space representation of the input image. Usually, scale space is implemented as an image pyramid as SIFT does [9], in which the input image is repeatedly smoothed with a Gaussian kernel and then subsampled to obtain a higher level of the pyramid. Different from this usual strategy by iteratively reducing the image size, SURF proposes to obtain a scale space representation of the input image by repeatedly increasing the filter size, as shown in Fig. 2.3.

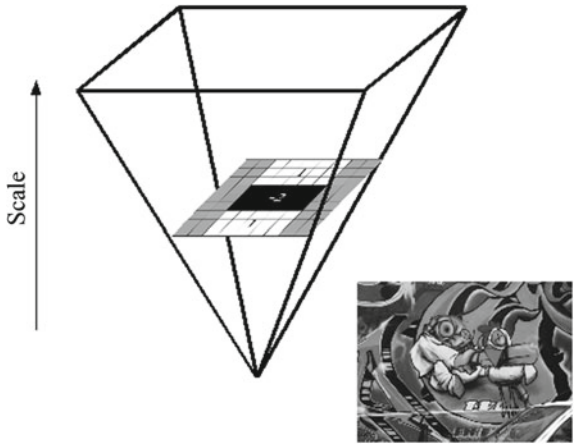

Fig. 2.3 Scale space representation of an input image is achieved by convolving with filters with gradually increasing sizes, while keeping the size of image. Reprinted from Bay et al. [2], with permission from Elsevier

Besides the scale normalized Laplacian of Gaussian (LoG) approximated by the Difference of Gaussian (DoG) used in SIFT, the determinant of Hessian matrix is another popular measure for detecting scale-invariant interest points. SURF's interest point detector is based on the computation of such determinant, but with some approximation so as to be computed very fast. Mathematically, the Hessian matrix of a point x in image I at scale σ can be computed by:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma)\ L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma)\ L_{yy}(x,\sigma) \end{bmatrix}$$

where $L_{xx}(x,\sigma)$ is the convolution of the second-order Gaussian derivative with image I in x, and σ is the standard variance of the Gaussian. Similarly for $L_{xy}(x,\sigma)$ and $L_{yy}(x,\sigma)$.

In order to use the integral image to efficiently compute the Hessian responses, SURF uses filters consisted by several box filters to approximate the second-order Gaussian derivatives as shown in Fig. 2.4. As the convolution of these box filters to image can be computed with only several additions based on the integral image, computing the approximated Hessian matrix is extremely fast. Formally, the approximated Hessian matrix $\hat{H}(x,\sigma)$ is defined by:

$$\hat{H}(x,\sigma) = \begin{bmatrix} D_{xx}(x,\sigma)\ D_{xy}(x,\sigma) \\ D_{xy}(x,\sigma)\ D_{yy}(x,\sigma) \end{bmatrix}$$

where $D_{xx}(x,\sigma)$, $D_{yy}(x,\sigma)$ and $D_{xy}(x,\sigma)$ are the convolutions of the approximated filters with image I respectively.

The determinant of Hessian matrix can be served as the response of a blob point, that is, $R(x,\sigma) = L_{xx}L_{yy} - L_{xy}^2 \approx D_{xx}D_{yy} - (0.9D_{xy}^2)$. In order to guarantee a constant Frobenius norm of any filter size, $R(x)$ is normalized with the corresponding filter size. This is important for scale space analysis as it prevents the reduction of filter

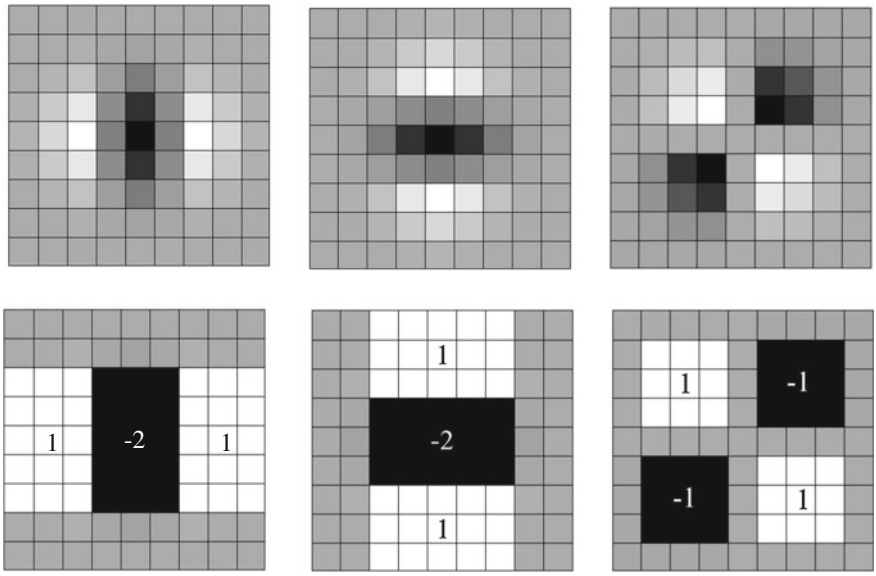

Fig. 2.4 Approximation of Gaussian second-order partial derivatives by *box filters*. From *left* to *right* are *Lxx*, *Lyy*, and *Lxy*. The approximated *box filters* are listed below their original ones. Reprinted from Bay et al. [3], with kind permission from Springer Science+Business Media

response at larger scales. Based on this measurement, interest points are detected across scales by non-maximum suppression which will be described in the next section. As can be seen, only $D_{xx}(x, \sigma)$, $D_{yy}(x, \sigma)$, and $D_{xy}(x, \sigma)$ are related to the interest point detection, thus three scale spaces are actually generated. One is for $D_{xx}(x, \sigma)$, and the other two for $D_{yy}(x, \sigma)$ and $D_{xy}(x, \sigma)$.

Similar to the octave structure utilized in SIFT, SURF also adapts this structure to represent the scale space. Specifically, each octave includes a scaling factor of 2, and contains several levels, each of which corresponds to an convolved image by a predefined filter size related to the scale of this level. The filter sizes of neighboring scales differ in a constant factor within an octave, and this size difference is doubled for neighboring octaves. As explained before, SURF uses box filters to approximate the second-order Gaussian derivatives. 9×9 is the minimal box filter size for this approximation (corresponding to a Gaussian with $\sigma = 1.2$), so it serves as the minimal scale in the scale space. As shown in Fig. 2.4, there are three parts in D_{xx} and D_{yy}. In order to assure that the total filter size is uneven, the minimal size that can be increased by each part is 2 for the two successive scales. As a result, it is 6 for the whole filter. Therefore, in the first octave, the filter sizes are 9, 15, 21, ... In the following step of finding extreme points in the scale space, it can be seen that it requires two additional neighboring levels to detect interest points in a certain level. Therefore, the first level and the last level in each octave are only for auxiliary purpose. Meanwhile, due to the interpolation in scale space for refining an interest

point's scale, the smallest scale in the first octave is $\frac{(9+15)}{2} \times \frac{1.2}{9} = 1.6$. Since an octave includes a scaling factor of 2, therefore, the largest scale in the first octave is 3.2, which corresponds to a filter size of 24, so the last useful level for interest point detection is 21 (because $\frac{(21+27)}{2} = 24$). Similarly, the filter sizes in the second octave are 15, 27, 39, 51; in the third octave are 27, 51, 75, 99, until the input image size is no longer larger than the filter size. As can be seen, the octaves in this scale space representation are overlapped to cover all the possible scales seamlessly.

From the above structure of the scale space, it is obvious that the sampling of scales is quite crude. For example, in the first two levels of the first octave, the scale change is $15/9 = 1.67$. Therefore, SURF also supplies a strategy for obtaining a scale space representation with finer sampling scales at the cost of a little more computational time. This strategy starts by doubling the input image using linear interpolation. Then, in the first octave, its first level is 15×15, instead of 9×9 in the original version. The filter sizes are 21, 27, 33, 39 in the remaining levels of the first octave. The remaining things are similar to those in the original version. The second octave starts with 27 and increased by a factor of 12, and so on for the third octave, the fourth octave, etc. In this implementation of the scale space, it can be found that the scale change is $21/15 = 1.4$, smaller than 1.67. Meanwhile, the finest scale that can be obtained by quadratic interpolation is $\frac{(15+21)}{2} \times \frac{1.2}{9} \times \frac{1}{2} = 1.2$.

2.2.3 Scale-Invariant Interest Point Detection

Interest points are localized by applying a non-maximum suppression of $R(x, \sigma)$ in $3 \times 3 \times 3$ neighborhood of both spatial and scale spaces. The fast non-maximum suppression technique described in [10] is used in SURF. The basic idea is the two local maxima are separated by at least r pixels in each dimension, where r is the neighborhood radius used for non-maximum suppression. Therefore, there is no need to check all the pixels.

As the position defined by the spatial grid and the crude sampling of scales is unstable, it is necessary to use an additional procedure to refine the location. This is achieved by interpolation through fitting a paraboloid over the space and scale that is identical to SIFT. Please refer to Sect. 2.1 for the details. After this procedure, maxima with subpixel/subscale accuracy are obtained, and are taken as the detected interest points.

2.2.4 Orientation Assignment and Descriptor Construction

In order to make the SURF descriptor invariant to image rotation, a reproducible orientation is first computed based on the information from a circular region around the interest point. Then, a square region aligned to the computed orientation is constructed. Finally, SURF descriptor is extracted from this square region.

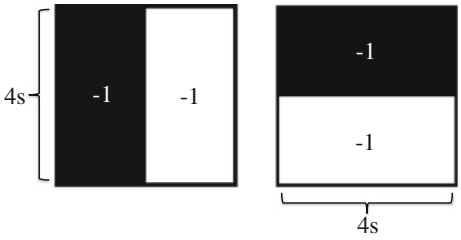

Fig. 2.5 The Haar wavelet filters used for computing responses in x (*left*) and y (*right*) directions at scale s. Reprinted from Bay et al. [2], with permission from Elsevier

To compute a reproducible orientation for each interest point, a circular neighborhood of radius $6s$ around the interest point is used, where s is the scale of the interest point. For each sampling point (rounded to be integer in order to use integral images) in this region, the Haar wavelet responses in x and y directions are calculated, which are denoted by dx and dy respectively. In order to keep scale invariance, the sampling step is set to be s. Meanwhile, the size of the Haar wavelet is also scale dependent and set to be $4s$, as illustrated in Fig. 2.5. By using integral images, only six operations are needed for computing the response in x/y direction at any scale. To give a more emphasis on the central sampling points, a Gaussian ($\sigma = 2s$) weighting function is applied to these Haar wavelet responses. Then, all these Gaussian weighted responses are mapped to a 2D space with x direction response as abscissa and y direction response as ordinate. A sliding orientation window of size $\pi/3$ is used to sum all the points to get a local orientation vector (cf. Fig. 2.6). Finally, the longest orientation vector across all sliding windows is taken as the orientation of the interest point.

In order to extract the SURF descriptor for an interest point, it has to construct a square region centered in the interest point and aligned to its orientation. The size of this square region is set to be $20s$ with a sampling step of s so as to obtain scale invariance. The square region is then regularly splitted into 4×4 subregions to keep

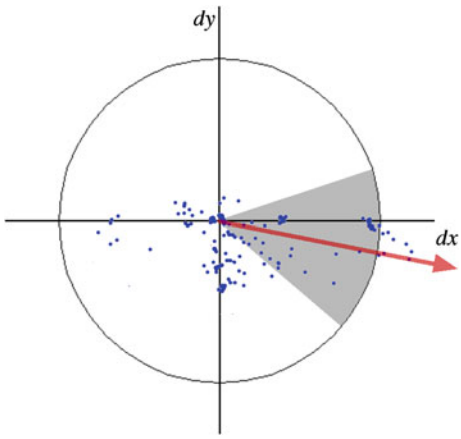

Fig. 2.6 A sliding orientation window is used for accumulating the weighted responses in x and y directions. The longest accumulated vector across all sliding windows is taken as the orientation of the interest point. Reprinted from Bay et al. [2], with permission from Elsevier

the important spatial information for distinctiveness. For each sample point in this squared region, its Haar wavelet responses in horizontal and vertical directions (dx and dy) are computed. Note that the 'horizontal' and 'vertical' are defined according to the orientation of the interest point. Different from the Haar wavelets used in orientation assignment, here the filter size is set to be $2s$. For efficiency, instead of rotating image and computing these Haar wavelet responses, they are first computed in the unrotated image and then rotated according to the interest point's orientation. They are then mapped to a 4-dimensional vector to encode information about the polarity of the intensity changes, i.e., $(dx, |dx|, dy, |dy|)$. Finally, these 4-dimensional vectors are summed up over each subregion, weighted by a Gaussian function ($\sigma = 3.3s$) centered at the interest point. All the accumulated vectors in all the 16 subregions are concatenated together to form the SURF descriptor (64 dimension), which is normalized to be unit length to achieve invariance to contrast changes. Using Gaussian function to weight the responses of sample points is to increase the robustness toward localization error and geometric deformation.

To further improve the descriptor's discriminative ability, an alternative version of SURF descriptor is to map the Haar wavelet responses to a 8-dimensional vector: $((dx, |dx|)_{dy>0}, (dx, |dx|)_{dy\leq 0}, (dy, |dy|)_{dx>0}, (dy, |dy|)_{dx\leq 0}) \in R^8$, where $(dy, |dy|)_{dx>0}$ means it is only valid when $dx > 0$, otherwise, it is set to $(0, 0)$. Similarly for other terms. This alternative version has a dimension of 128. It is more distinctive and only a bit slower to compute, but slower to match and requires two times of storage due to its high dimensionality.

2.3 Local Binary Pattern and Its Variants

Another very popular and well-known local feature is the family of Local Binary Pattern (LBP). Since the first basic LBP has been introduced in 1990s, LBP methodology has developed a lot in the past two decades, ranging from extensions, related theories, to various new applications.

In the beginning, LBP [11] was proposed to describe texture, which could be characterized by a nonuniform distribution of intensities or colors, with applications to texture classification and segmentation. Due to its good performance and computational simplicity, LBP rapidly gained its popularity in the computer vision community and has become a widely used operator for image processing in real world applications, not limited to texture-related applications.

By applying LBP operator to an image, each pixel is denoted by an integer label (e.g., 256 different labels in the original LBP with 3×3 neighborhood configuration) which is robust to monotonic illumination change. Each of these labels is called a LBP pattern. In the original version of LBP [11], the LBP pattern for a pixel is computed by comparing each of its neighboring pixel in a 3×3 area to the central pixel about their intensities, as shown in Fig. 2.7. Such a comparison will result in

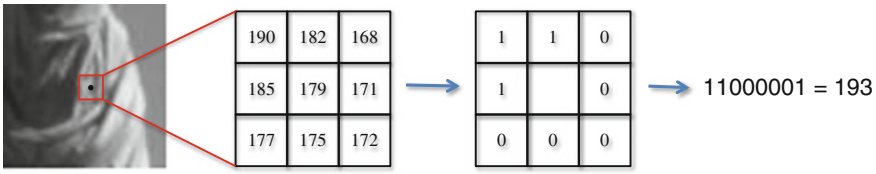

Fig. 2.7 Illustration of LBP computation for a pixel

a bit string with 8 elements, therefore, $2^8 = 256$ different possible labels in total. Mathematically, a more *generic definition of LBP* is given in the following [12]:

$$LBP_{R,N}(x) = \sum_{i=0}^{N-1} sign(I(x_i) - I(x))2^i \tag{2.10}$$

where $sign(x)$ is the sign function that outputs 1 when input is larger than 0, otherwise outputs 0. R defines a circular neighborhood of the pixel x with radius R, and N denotes the number of regularly sampling points on this circle used for LBP computation. In any case that a sampling point is not located in a discrete pixel, the bilinear interpolation is used to get its intensity.

Based on experimental results, Ojala et al. [12] proposed uniform patterns of LBP. Based on the definition of LBP pattern, they further defined an uniformity measure of a pattern as the number of bitwise transitions from 0 to 1 or from 1 to 0 in its binary representation. Note that in this definition, a pattern is considered circularly. The uniformity measure can be formulated as:

$$U(LBP_{R,N}) = \sum_{i=1}^{N-1} |b_{i+1} - b_i| + |b_1 - b_N| \tag{2.11}$$

in which b_i is the ith bit of the LBP bit string.

According to this uniformity measure, Ojala et al. defined the LBP patterns whose uniformity measures are no larger than 2 as the *uniform patterns*: $LBP^u_{R,N} = \{LBP_{R,N} | U(LBP_{R,N}) \leq 2\}$. For N bits LBP pattern, there are $N(N-1)+2$ uniform patterns. Usually, for the remaining nonuniform patterns, they are considered together as a single label (pattern). Therefore, with the uniform patterns, histogram of LBP can be reduced from 2^N bins to $N(N-1)+3$ bins, largely reducing the dimension used for feature description. Meanwhile, uniform patterns are found to be more stable and dominant in natural images. For example, nearly 90 % of all patterns are uniform patterns when using $(R = 1, N = 8)$ LBP neighborhood according to Ojala et al. experiments with texture images. Similar statistical result is also observed in facial images [1].

To achieve a rotation-invariant representation, a LBP pattern is circularly shifted into its minimum integer value [14], as following:

$$LBP^{ri}_{R,N} = \min_{i\in[1,N]} shift(LBP_{R,N}, i) \tag{2.12}$$

where $shift(LBP_{R,N}, i)$ indicates the right-shifted operation (in a circular way) to $LBP_{R,N}$ by i bits. The pattern $LBP^{ri}_{R,N}$ corresponding to this minimum integer value is defined as the *rotation-invariant LBP*. In case of $N = 8$, there are 36 rotation-invariant LBPs. We can see that rotation-invariant LBP actually maps a set of LBP patterns into a single one, thus the discriminative power may be reduced. Due to the definition of $LBP_{R,N}$ and $LBP^{ri}_{R,N}$, theoretically, such a rotation invariance is only available by angles $2\pi \times i/N, i = 1, \ldots, N-1$. However, in practice, it is very robust by any angle.

In the set of uniform patterns, besides the patterns with all 1 s and all 0s, it contains $N-1$ groups of patterns. Each group contains N patterns with i $(1 \leq i < N)$ 1s, all of which only differ by rotation. Therefore, each group actually corresponds to one pattern if we do not consider rotation. As a result, there are totally $N+1$ different such patterns, known as *rotation-invariant uniform patterns*. They can be formulated by:

$$LBP^{riu}_{R,N} = \left\{ \sum_{i=1}^{N} b_i | U(LBP_{R,N}) \leq 2 \right\} \tag{2.13}$$

Similar to the case of uniform pattern, all the other patterns that do not belong to any of the rotation-invariant uniform patterns are considered as a single pattern.

Many extensions of LBP have been proposed to either improve its robustness and discriminative power or adapt it to new computer vision tasks. These variants include improvements on many parts of LBP, for example, incorporating other complementary feature into LBP, changing its method for thresholding and intensity comparison, extending with multiscale analysis, making it capable of dealing with color information, extending it to spatiotemporal area, etc. In this section, we will briefly describe some of these variants which we think are important in the development of LBP. For a comprehensive review and introduction of LBP variants, we refer the interested readers to [13].

Local Gabor Binary Patterns: Local Gabor Binary Patterns (LGBP) [17] is a very good local feature popularly used in face recognition. It effectively combines local and regional information by using Gabor filtering and LBP operator together. Specifically, a facial image is first filtered by a set of Gabor filters with different scales and orientations, in order to obtain rich appearance information over a board range of scales and orientations. Then, LBP operator is applied to each of these Gabor filtered images (only the magnitudes are used), and a set of histograms are obtained for spatial subregions divided from the input facial image for each Gabor filtered image. Finally, all the histograms are concatenated together as the LGBP feature. Its pipeline is depicted in Fig. 2.8. However, the disadvantage of this method is the high dimensionality of this kind of representation, for example, with 5 scales and 8 orientations, it results in a face representation that have 40 times larger dimensions than that of using the original LBP.

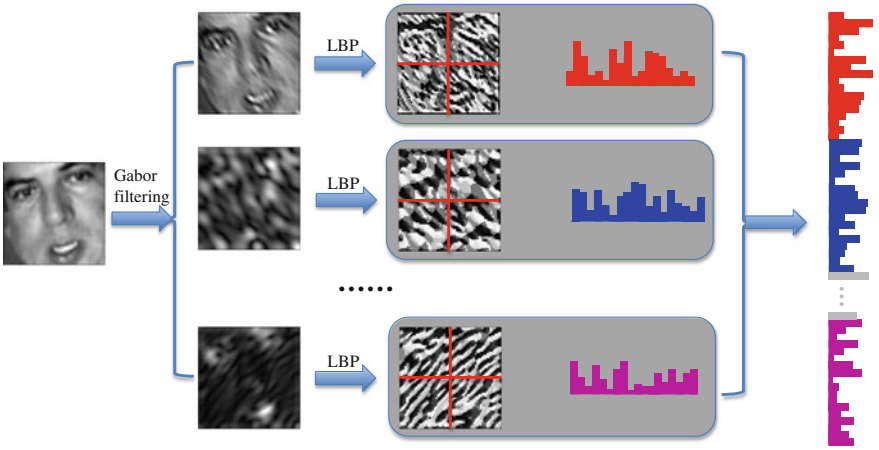

Fig. 2.8 The flowchart of computing LGBP

Multiscale Block Local Binary Pattern: Another widely used LBP variant in face recognition is Multiscale Block Local Binary Pattern (MB-LBP) [8]. It extends the original LBP to encode information from larger areas, so as to cover both micro- and macrostructures in facial images. Thus higher discriminative power is expected. The basic ideas of MB-LBP are to use the average intensity over a $s \times s$ subregion to replace the intensity of the central pixel/neighboring pixel in the original LBP. The size of s corresponds to the scale of MB-LBP. Usually, several scales of MB-LBP are used together, followed by the Adaboost algorithm [5] to select most discriminative features for face recognition. Figure 2.9 gives an example of using 3×3 subregions.

Another modification of LBP that MB-LBP proposed is to use a statistically effective set of uniform patterns obtained from training data. More specifically, a histogram of MB-LBP patterns is first obtained for each scale. Then, for each scale, the MB-LBP patterns correspond to the top N bins are obtained as the N uniform MB-LBP patterns and receive labels from 0 to $N - 1$. All the remaining patterns share a single label N. Since different scales are processed separately, the obtained

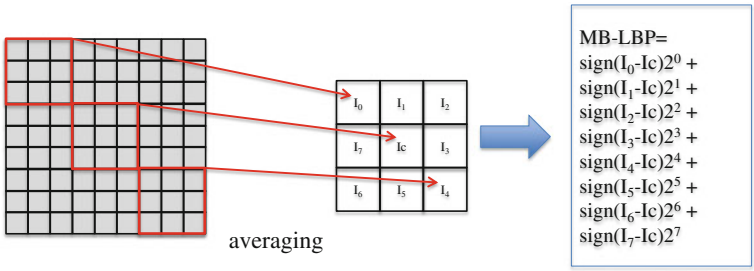

Fig. 2.9 Illustration of MB-LBP computation with 3×3 blocks

set of uniform MB-LBP patterns will have large redundance. The final set used for face recognition is selected by the Adaboost algorithm.

Local Ternary Pattern: To make LBP more robust to small changes/noise in intensity, which usually occurs in flat or near uniform image regions and shadow areas, Tan and Triggs [15] modified the binary pattern in LBP to a ternary pattern, obtaining the so-called Local Ternary Pattern (LTP). In LBP, it computes a two-value bit code (0 or 1) for each sampling point in the neighborhood by thresholding its intensity on that of the central pixel. While in LTP, it computes a three-value code (-1, 0, 1) for each sampling point by thresholding on two intensities as follows:

$$s(x_i) = \begin{cases} 1, & I(x_i) \geq I(x) + t \\ 0, & I(x) - t \leq I(x_i) \leq I(x) + t \\ -1, & I(x_i) < I(x) - t \end{cases} \tag{2.14}$$

where $I(x)$ is the intensity of the central pixel, x_i is the ith sampling point in the neighborhood of x, and t is a predefined offset to generate two thresholds, usually set to be 5. All values of N sampling points are concatenated together to serve as the LTP code of the central pixel.

Intuitively, N sampling points could form a total of 3^N possible LTP codes. Such a large range will further make the histogram representation very sparse, which may degrade its discriminative power and make it sensitive to noise. For these reasons, the N three value codes $s(x_i)$, $i = 1, \ldots, N$ are divided into two sets according to their signs, obtaining two local binary patterns, as shown in Fig. 2.10. These two parts are used separately to obtain representations of the described region, e.g., histograms. These representations are finally concatenated as a single feature vector.

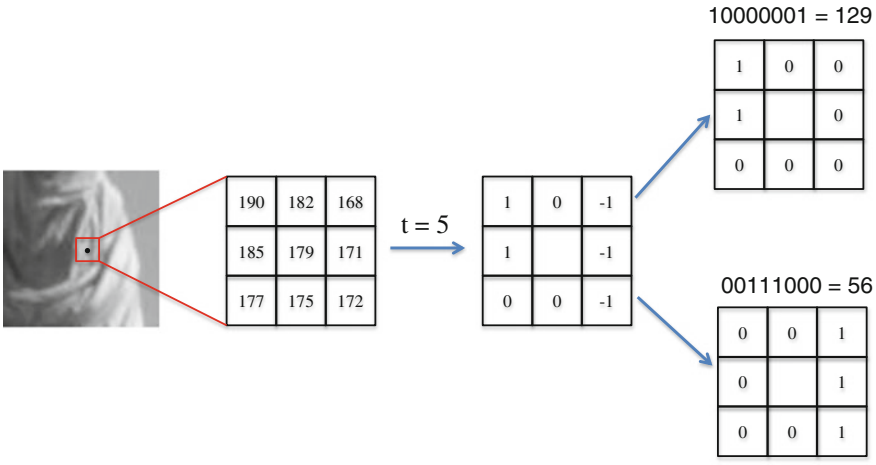

Fig. 2.10 Illustration of LTP computation for a pixel when $t = 5$

A drawback of LTP is that using a constant offset t to generate two thresholds makes the computed LTP code not invariant under scale change of intensities. Liao et al. [7] noticed this problem and proposed a Scale-Invariant LTP (SILTP) to address it. The basic idea of SILTP is very simple, i.e., replacing the offset t in LTP with a value proportional to the intensity of the central pixel. Meanwhile, to avoid splitting a ternary code into two binary ones, they used two bits to represent the obtained three values. Concatenating all the binary codes of sampling points then forms the SILTP representation for a pixel. Mathematically, the SILTP operator can be formulated as:

$$s_i(x_i) = \begin{cases} 01, & I(x_i) > (1+\tau)I(x) \\ 00, & (1-\tau)I(x) \leq I(x_i) \leq (1+\tau)I(x) \\ 10, & I(x_i) < (1-\tau)I(x) \end{cases} \tag{2.15}$$

where τ is a predefined scale factor, indicating how much of the central pixel's intensity can be tolerated. Apparently, SILTP operator is scale-invariant about intensities.

Center-Symmetric Local Binary Pattern: Although LBP has high discriminative power and good robustness to illumination change, its high dimensionality makes it not suitable for describing an interest region around a keypoint, for which a low-dimensional descriptor is usually required, for example, the 128-dimensional SIFT descriptor. Even the histogram of uniform LBP is still too long to be used in an interest region descriptor. One may argue that we can take the histogram of LBP in the entire interest region as the descriptor whose dimension is 256 if the original LBP is used. However, such method does not encode any spatial information in the descriptor which is very important on distinguishing different interest regions. Therefore, its discriminative ability is significantly reduced, making it useless in matching interest regions across different images.

In spite of the difficulties, the good performance of LBP against illumination changes makes it attractive to researchers working on the area of interest region description. Motivated by this, Heikkila et al. [6] modified LBP to have only 16 possible patterns in total when 8 neighborhood is used. In this case, it can be used in a feature description algorithm similar to SIFT and leads to a high-performance descriptor. The core idea of this modification is to compare intensities of the center-symmetric sampling points instead of compare intensities of the sampling point and the central pixel. Therefore, for a N neighborhood samplings, there are only $N/2$ bits generated, which can be further encoded into a $2^{N/2}$ dimensional histogram. The modified pattern is called Center-Symmetric Local Binary Pattern (CS-LBP). Moreover, since there may exist large portion of flat area in an interest region, it is necessary to adapt LBP to be robust to small intensity changes as this often occurs in flat area with noise. For this purpose, CS-LBP operator introduces a tolerant threshold on intensity comparison. To sum up, the CS-LBP operator of pixel x can be formulated as:

$$CS-LBP_{R,N}(x) = \sum_{i=0}^{N/2-1} sign(I(x_i) - I(x_{i+N/2}) - t)2^i \tag{2.16}$$

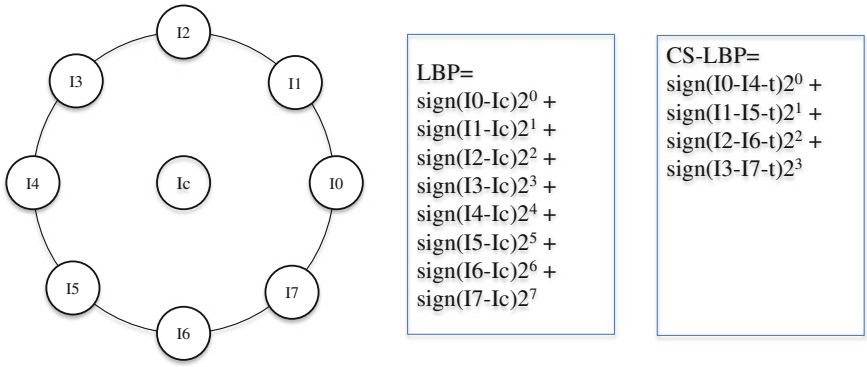

Fig. 2.11 A comparison of LBP operator and CS-LBP operator when 8 neighborhood samplings are used. Reprinted from Heikkila et al. [6], with kind permission from Springer Science+Business Media

The meanings of R, N and $sign(x)$ are identical to those in Eq. (2.10). t is a threshold balancing the tolerance to noise and discriminative power of CS-LBP. The bigger value the t is, the bigger turbulence to intensity it can resist. However, a small value of t is required to make CS-LBP discriminative. A typical setting of t is 3 in most cases. $I(x_i)$ and $I(x_{i+N/2})$ are intensities of two center-symmetric sampling points. Figure 2.11 depicts the difference of LBP and CS-LBP when $N = 8$.

The low-dimensional property of CS-LBP makes it suitable for interest region description by concatenating histograms in divided subregions like SIFT does. This is indeed the way that Heikkila et al. [6] did to construct a descriptor so as to combine strengths of SIFT and LBP. The resulting descriptor is called CS-LBP as well. The pipeline for constructing a CS-LBP descriptor is similar to that of SIFT, except for the following two differences:

(1) It does not have the feature weighting step, which assigns each pixel in the interest region a weight according to its gradient magnitude and a Gaussian function in SIFT. Omitting this step is supported by their experimental results reported in [6].
(2) When accumulating the 3D histogram of CS-LBP values and locations, a pixel is contributed to its 4 neighboring bins in spatial by using bilinear interpolation. While in SIFT, a pixel is contributed to its 8 neighboring bins by interpolation in both spatial and gradient orientation. Since CS-LBP is quantized by nature, it is not necessary to interpolate in this dimension when constructing CS-LBP descriptor.

Local Intensity Order Pattern: To fully explore the intensity relationship among neighboring sampling points of a pixel, Wang et al. [16] proposed the Local Intensity Order Pattern (LIOP) for interest region description. Although both proposed for interest region description, the advantage of LIOP over CS-LBP is that it not only

compares intensities between center-symmetric sampling points, but also compares intensities among nearby sampling points. Due to this intensity comparing strategy, LIOP is more discriminative than CS-LBP. Moreover, LIOP only uses 4 neighboring points instead of 8 neighboring points while achieving a better performance.

Given 4 neighboring points x_0, x_1, x_2, x_3 of a pixel x, their intensities are denoted by $I(x_0), I(x_1), I(x_2), I(x_3)$. As there are 24 possible permutations for 4 intensities, each permutation can be defined as a local intensity order pattern. A local intensity order pattern can be further represented by a 24-dimensional vector, with one element being 1 and the others being 0. Therefore, by sorting intensities of the 4 neighboring points of x, we can first obtain a permutation and then have its corresponding 24-dimensional vector as the local intensity order pattern of x. Please see Sect. 3.3 for more details about how to use it to construct a robust and distinctive descriptor for interest region matching.

References

1. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. IEEE Trans. Pattern Anal. Mach. Intell. **28**(12), 2037–2041 (2006)
2. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: SURF: speeded up robust features. Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
3. Bay, H., Tuytelaars, T., Gool, L.V.: SURF: Speeded up robust features. In: European Conference on Computer Vision, pp. 404–417 (2006)
4. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 886–893 (2005)
5. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**(2), 337–407 (2000)
6. Heikkila, M., Pietikainen, M., Schmid, C.: Description of interest regions with center-symmetric local binary patterns. In: 5th Indian Conference on Computer Vision, Graphics and Image Processing, pp. 58–69 (2006)
7. Liao, S., Zhao, G., Kellokumpu, V., Pietikainen, M., Li, S.: Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1301–1306 (2010)
8. Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.: Learning multi-scale block local binary patterns for face recognition. In: International Conference on Biometrics, pp. 828–837 (2007)
9. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
10. Neubeck, A., Van Gool, L.: Efficient non-maximum suppression. Int. Conf. Pattern Recogn. **3**, 850–855 (2006)
11. Ojala, T., Pietikainen, M., Harwood, D.: A comparative study of texture measures with classification based on feature distributions. Pattern Recogn. **29**, 51–59 (1996)
12. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
13. Pietikainen, M., Hadid, A., Zhao, G., Ahonen, T.: Computer Vision Using Local Binary Patterns. Springer (2011)
14. Pietikainen, M., Ojala, T., Xu, Z.: Rotation-invariant texture classification using feature distributions. Pattern Recogn. **33**, 43–52 (2000)
15. Tan, X., Triggs, B.: Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE Trans. Image Process. **19**(6), 1635–1650 (2010)

16. Wang, Z., Fan, B., Wu, F.: Local intensity order pattern for feature description. In: International Conference on Computer Vision, pp. 603–610 (2011)
17. Zhang, W., Shan, S., Gao, W., Chen, X., Zhang, H.: Local gabor binary pattern histogram sequence (LGBPHS): A novel non-statistical model for face representation and recognition. In: International Conference on Computer Vision, pp. 786–791 (2005)

http://www.springer.com/978-3-662-49171-3

Local Image Descriptor: Modern Approaches

Fan, B.; Wang, Z.; Wu, F.

2015, XII, 99 p. 33 illus., 7 illus. in color., Softcover
ISBN: 978-3-662-49171-3