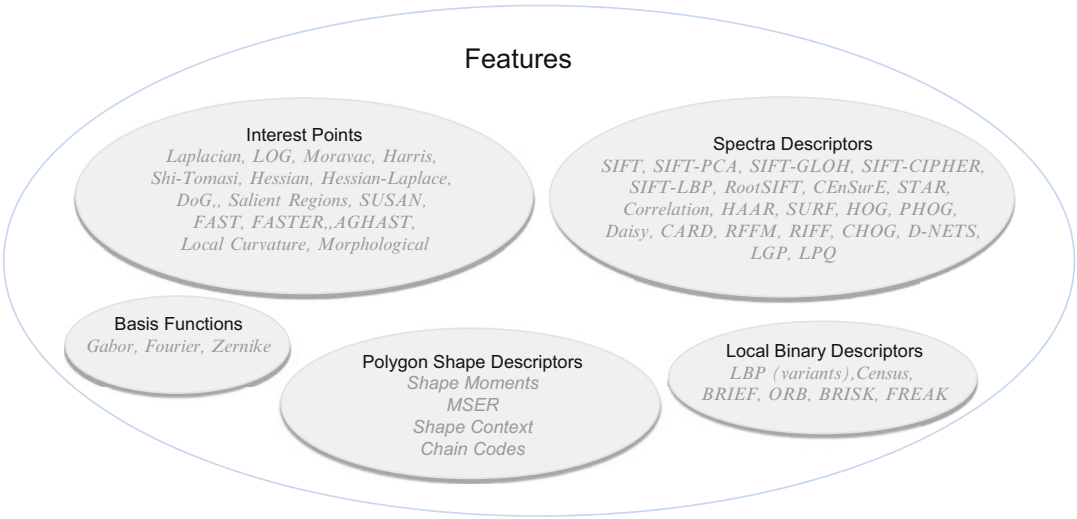
Interest Point Detector and Feature Descriptor Survey

6

"Who makes all these?"

—*Jack Sparrow,* Pirates of the Caribbean

Many algorithms for computer vision rely on locating interest points, or keypoints in each image, and calculating a feature description from the pixel region surrounding the interest point. This is in contrast to methods such as correlation, where a larger rectangular pattern is stepped over the image at pixel intervals and the correlation is measured at each location. The interest point is the *anchor point*, and often provides the scale, rotational, and illumination invariance attributes for the descriptor; the descriptor adds more detail and more invariance attributes. Groups of interest points and descriptors together describe the actual objects.



However, there are many methods and variations in feature description. Some methods use features that are not anchored at interest points, such as polygon shape descriptors, computed over larger segmented polygon-shaped structures or regions in an image. Other methods use interest points only, without using feature descriptors at all. And some methods use feature descriptors only, computed across a regular grid on the image, with no interest points at all.

Terminology varies across the literature. In some discussions, interest points may be referred to as *keypoints*. The algorithms used to find the interest points maybe referred to as *detectors*, and the

S. Krig, *Computer Vision Metrics*, DOI 10.1007/978-3-319-33762-3_6

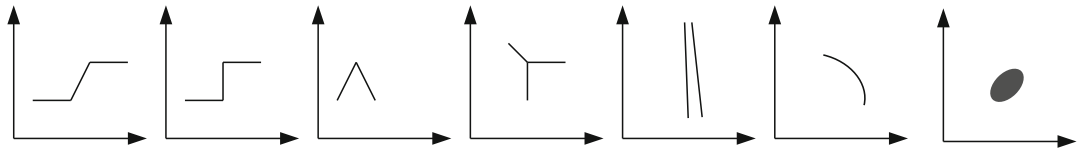


Figure 6.1 Types of keypoints, including corners and interest points. (*Left* to *right*) Step, roof, corner, line or edge, ridge or contour, maxima region

algorithms used to describe the features may be called *descriptors*. We use the terminology interchangeably in this work. Keypoints may be considered a set composed of (1) interest points, (2) corners, (3) edges or contours, and (4) larger features or regions such as blobs; see Fig. 6.1. This chapter surveys the various methods for designing local interest point detectors and feature descriptors.

Interest Point Tuning

What is a good keypoint for a given application? Which ones are most useful? Which ones should be ignored? Tuning the detectors is not simple. Each detector has different parameters to tune for best results on a given image, and each image presents different challenges regarding lighting, contrast, and image preprocessing. Additionally, each detector is designed to be useful for a different class of interest points, and must be tuned accordingly *to* filter the results down to a useful set of good candidates for a specific feature descriptor. Each feature detector will work best with certain descriptors, see Appendix A.

So the keypoints are further filtered to be useful for the chosen feature descriptor. In some cases, a keypoint is not suitable for producing a useful feature descriptor, even if the keypoint has a high score and high response. If the feature descriptor computed at the keypoint produces a descriptor score that is too weak, for example, the keypoint and corresponding descriptor should both be rejected. OpenCV provides several novel methods for working with detectors, enabling the user to try different detectors and descriptors in a common framework, and automatically adjust the parameters for tuning and culling as follows:

- **DynamicAdaptedFeatureDetector.** This class will tune supported detectors using an *adjusterAdapter()* to only keep a limited number of features, and iterate the detector parameters several times and redetect features in an attempt to find the best parameters, keeping only the requested number of best features. Several OpenCV detectors have an *adjusterAdapter()* provided, some do not; the API allows for adjusters to be created.
- **AdjusterAdapter.** This class implements the criteria for culling and keeping interest points. Criteria may include KNN nearest neighbor matching, detector response or strength, radius distance to nearest other detected points, number of keypoints within a local region, and other measures that can be included for culling keypoints for which a good descriptor cannot be computed.
- **PyramidAdaptedFeatureDetector.** This class can be used to adapt detectors that do not use a scale-space pyramid, and the adapter will create a Gaussian pyramid and detect features over the pyramid.
- **GridAdaptedFeatureDetector.** This class divides an image into grids and adapts the detector to find the best features within each grid cell.

Interest Point Concepts

An interest point may be composed of various types of corner, edge, and maxima shapes, as shown in Fig. 6.1. In general, a good interest point must be easy to find and ideally fast to compute; it is hoped that the interest point is at a good location to compute a feature descriptor. The interest point is thus the qualifier or *keypoint* around which a feature may be described.

There are various concepts behind the interest point methods currently in use, as this is an active area of research. One of the best analyses of interest point detectors is found in Mikolajczyk et al. [145], with a comparison framework and taxonomy for affine covariant interest point detectors, where *covariant* refers to the elliptical shape of the interest region, which is an affine deformable representation. Scale invariant detectors are represented well in a circular region. Maxima region and blob detectors can take irregular shapes. See the response of several detectors against synthetic interest point and corner alphabets in Appendix A.

Figure 6.2 Candidate edge interest point filters. (*Left* to *right*) Laplacian, derivative filter, and gradient filter

Commonly, detectors use *maxima and minima points*, such as gradient peaks and corners; however, edges, ridges, and contours are also used as keypoints, as shown in Fig. 6.2. There is no superior method for interest point detection for all applications. A simple taxonomy provided by Tuytelaars and Van Gool [511] lists edge-based region methods (EBR), maxima or intensity-based region methods (IBR), and segmentation methods to find shape-based regions (SBR) that may be blobs or features with high entropy.

Corners are often preferred over edges or isolated maxima points, since the corner is a structure and can be used to compute an angular orientation for the feature. Interest points are computed over color components as well as gray scale luminance. Many of the interest point methods will first apply some sort of Gaussian filter across the image and then perform a gradient operator. The idea of using the Gaussian filter first is to reduce noise in the image, which is otherwise amplified by gradient operators.

Each detector locates features with different degrees of invariance to attributes such as rotation, scale, perspective, occlusion, and illumination. For evaluations of the quality and performance of interest point detection methods measured against various robustness and invariance criteria on standardized datasets, see Mikolajczyk and Schmidt [136] and Gauglitz et al. [137]. One of the key challenges for interest point detection is scale invariance, since interest points change dramatically in some cases over scale. Lindberg [204] has extensively studied the area of scale independent interest point methods.

Affine invariant interest points have been studied in detail by Mikolajcyk and Schmid [99, 133, 136, 145, 298, 303]. In addition, Mikolajcyk and Schmid [501] developed an affine-invariant version of the Harris detector. As shown in [523], it is often useful to combine several interest point detection methods to form a hybrid, for example, using the Harris or Hessian to locate suitable maxima regions,

and then using the Laplacian to select the best scale attributes. Variations are common, Harris-based and Hessian-based detectors may use scale-space methods, while local binary detector methods do not use scale space.

A few fundamental concepts behind many interest point methods come from the field of linear algebra, where the local region of pixels is treated as a matrix. (*Refer to a good linear algebra textbook as background for this section.)* Additional concepts come from other areas of mathematical analysis. Some of the key math useful for locating interest points are illustrated below, however note that in practice various forms of equations and algorithms are used which deviate from those shown here, see the references for more details.

- **A Matrix.** We start with a 2d rectangular pixel region, or matrix, of some dimension *x,y*:

$$M_{x,y} = \begin{bmatrix} 0,0 & \dots & x,0 \\ \dots & \dots & \dots \\ 0,y & \dots & x,y \end{bmatrix}$$

- **Gradient Magnitude.** This is the first derivative of the pixels in the local interest region, and assumes a direction. This is an unsigned positive number, and is also a Laplacian operator.

$$\left(\frac{\partial M_{x,y}}{\partial x}\right)^2 + \left(\frac{\partial M_{x,y}}{\partial y}\right)^2$$

- **Gradient Direction.** This is the angle or direction of the largest gradient angle from pixels in the local region in the range +π to -π.

$$tan^{-1}\left(\frac{\partial M_{x,y}}{\partial x}\right)^2 / \left(\frac{\partial M_{x,y}}{\partial y}\right)^2$$

- **Laplacian.** This is the second derivative and can be computed selectively using any of three terms:

$$\frac{\partial^2 f\mathbf{M}_{x,y}}{\partial x^2}$$
$$\frac{\partial^2 \mathbf{M}_{x,y}}{\partial y^2}$$
$$\frac{\partial^2 \mathbf{M}_{x,y}}{\partial x \partial x}$$

  However, the Laplacian operator does not use the third form above, and computes a signed value of average orientation with respect to *x and y* partials only, see the Gradient Magnitude operator above.
- **Hessian Matrix or Hessian.** A square matrix containing second order partial derivatives of each pixel within the matrix region, describing surface curvature at each pixel. The Hessian has several interesting properties useful for interest point detection methods discussed in this section, which we can express in ***L*** notation as follows:

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

- **Largest Hessian.** This is based on the second derivative, as is the Laplacian, but the Hessian uses all three terms of the second derivative to compute the direction along which the second derivative is maximum as a signed value.

- **Smallest Hessian.** This is based on the second derivative, is computed as a signed number, and may be a useful metric as a ratio between largest and smallest Hessian.
- **Hessian Orientation, largest and smallest values.** This is the orientation of the largest second derivative in the range $+\pi$ to $-\pi$, which is a signed value, and it corresponds to an orientation without direction. The smallest orientation can be computed by adding or subtracting $\pi/2$ from the largest value.
- **Determinant of Hessian, Trace of Hessian, Laplacian of Gaussian.** All three names are used to describe the trace characteristic of a matrix, which can reveal geometric scale information by the absolute value, and orientation by the sign of the value. See SURF [166] for an application, which we can express in $\boldsymbol{L}$ notation as follows.

$$trace\ \mathcal{H}_{norm}\boldsymbol{L} = t^{\gamma}\nabla^2\boldsymbol{L} = t^{\gamma}\left(\boldsymbol{L}_{xx} + \boldsymbol{L}_{yy}\right)$$

$$det\ \mathcal{H}_{norm}\boldsymbol{L} = t^{2\gamma}\left(\boldsymbol{L}_{xx}\,\boldsymbol{L}_{yy} - \boldsymbol{L}^2{}_{xy}\right)$$

- **Eigenvalues, Eigenvectors, Eigenspaces.** Eigen properties are important to understanding vector direction in local pixel region matrices. When a matrix acts on a vector, and the vector orientation is preserved, and when the sign or direction is simply reversed, the vector is considered to be an eigenvector, and the matrix factor is considered to be the eigenvalue. An eigenspace is therefore all eigenvectors within the space with the same eigenvalue. Eigen properties are valuable for interest point detection, orientation, and feature detection. For example, Turk and Petland [150] use eigenvectors reduced into a smaller set of vectors via PCA for face recognition, in a method they call Eigenfaces.

Interest Point Method Survey

We will now look briefly at algorithms and computational methods for some common interest point detector methods including:

- Laplacian of Gaussian (LOG)
- Moravac corner detector
- Harris and Stephens corner detection
- Shi and Tomasi corner detector (improvement on Harris method)
- Difference of Gaussians (DoG; an approximation of LOG)
- Harris methods, Harris–/Hessian–Laplace, Harris/Hessian Affine
- Determinant of Hessian (DoH)
- Salient regions
- SUSAN
- FAST, FASTER, AGAST
- Local curvature
- Morphological interest points
- MSER (discussed in the section on polygon shape descriptors)
- *NOTE: many feature descriptors, such as SIFT, SURF, BRISK, and others, provide their own detector method along with the descriptor method, see Appendix A.

Laplacian and Laplacian of Gaussian

The Laplacian operator, as used in image processing, is a method of finding the derivative or maximum rate of change in a pixel area. Commonly, the Laplacian is approximated using standard convolution kernels that add up to zero, such as:

$$\mathrm{L1} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

$$\mathrm{L2} = \begin{pmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{pmatrix}$$

The Laplacian of Gaussian (LOG) is simply the Laplacian performed over a region that has been processed using a Gaussian smoothing kernel to focus edge energy; see Gun [147].

Moravac Corner Detector

The Moravic corner detection algorithm is an early method of corner detection whereby each pixel in the image is tested by correlating overlapping patches surrounding each neighboring pixel. The strength of the correlation in any direction reveals information about the point: a corner is found when there is change in all directions, and an edge is found when there is no change along the edge direction. A flat region yields no change in any direction. The correlation difference is calculated using the SSD between the two overlapping patches. Similarity is measured by the near-zero difference in the SSD. This method is compute intensive; see Moravac [322].

Harris Methods, Harris–Stephens, Shi–Tomasi, and Hessian Type Detectors

The Harris or Harris–Stephens corner detector family [148, 357] provides improvements over the Moravic method. The goal of the Harris method is to find the direction of fastest and lowest change for feature orientation, using a covariance matrix of local directional derivatives. The directional derivative values are compared with a scoring factor to identify which features are corners, which are edges, and which are likely noise. Depending on the formulation of the algorithm, the Harris method can provide high rotational invariance, limited intensity invariance, and in some of the formulations of the algorithm, scale invariance is provided such as the Harris–Laplace method using scale space [204, 501]. Many Harris family algorithms can be implemented in a compute-efficient manner.

Note that corners have an ill-defined gradient, since two edges converge at the corner, but near the corner the gradient can be detected with two different values with respect to x and y—this is a basic idea behind the Harris corner detector.

Variations on the Harris method include:

- The Shi, Tomasi, and Kanade corner detector [149] is an optimization on the Harris method, using only the minimum eigenvalues for discrimination, thus streamlining the computation considerably.

- The Hessian (Hessian affine) corner detector [145] is designed to be affine invariant, and it uses the basic Harris corner detection method but combines interest points from several scales in a pyramid, with some iterative selection criteria and a Hessian matrix.
- Many other variations on the basic Harris operator exist, such as the Harris–Hessian–Laplace [323], which provides improved scale invariance using a scale selection method, and the Harris/Hessian affine method [145, 298].

Hessian Matrix Detector and Hessian–Laplace

The Hessian Matrix method, also referred to as Determinant of Hessian (DoH) method, is used in the popular SURF algorithm [152]. It detects interest objects from a multi-scale image set where the determinant of the Hessian matrix is at a maxima and the Hessian matrix operator is calculated using the convolution of the second-order partial derivative of the Gaussian to yield a gradient maxima.

The DoH method uses integral images to calculate the Gaussian partial derivatives very quickly. Performance for calculating the Hessian Matrix is therefore very good, and accuracy is better than many methods. The related Hessian–Laplace method [298, 323] also operates on local extrema, using the determinant of the Hessian at multiple scales for spatial localization, and the Laplacian at multiple scales for scale localization.

Difference of Gaussians

The Difference of Gaussians (DoG) is an approximation of the Laplacian of Gaussians, but computed in a simpler and faster manner using the difference of two smoothed or Gaussian filtered images to detect local extrema features. The idea with Gaussian smoothing is to remove noise artifacts that are not relevant at the given scale, which would otherwise be amplified and result in false DoG features. The DoG features are used in the popular SIFT method [153], and as shown in Fig. 6.15, the simple difference of Gaussian filtered images is taken to identify maxima regions.

Salient Regions

Salient regions [154, 155] are based on the notion that interest points over a range of scales should exhibit local attributes or entropy that are "unpredictable" or "surprising" compared to the surrounding region. The method proceeds as follows:

1. The Shannon entropy E of pixel attributes such as intensity or color are computed over a scale space, where Shannon entropy is used the measure of unpredictability.
2. The entropy values are located over the scale space with maxima or peak values M. At this stage, the optimal scales are determined as well.
3. The probability density function (PDF) is computed for magnitude deltas at each peak within each scale, where the PDF is computed using a histogram of pixel values taken from a circular window of desired radius from the peak.
4. Saliency is the product of E and M at each peak, and is also related to scale. So the final detector is salient and robust to scale.

SUSAN, and Trajkovic and Hedly

The SUSAN method [156, 157] is dependent on segmenting image features based on local areas of similar brightness, which yields a bimodal valued feature. No noise filtering and no gradients are used. As shown in Fig. 6.3, the method works by using a center nucleus pixel value as a comparison reference against which neighbor pixels within a given radius region are compared, yielding a set of pixels with similar brightness, called a Univalue Segment Assimilating Nucleus (USAN).

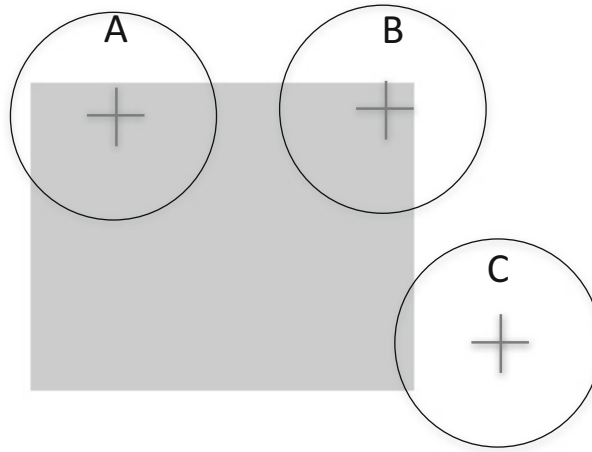


Figure 6.3 SUSAN method of computing interest points. The *dark region* of the image is a *rectangle* intersecting USANs A, B, and C. USAN A will be labeled as an edge, USAN B will be labeled as a corner, and USAN C will be labeled as neither an edge nor a corner

Each USAN contains structural information about the image in the local region, and the size, centroid, and second-order moments of each USAN can be computed. The SUSAN method can be used for both edge and corner detection. Corners are determined by the ratio of pixels similar to the center pixel in the circular region: a low ratio around 25 % indicates a corner, and a higher ratio around 50 % indicates an edge. SUSAN is very robust to noise.

The Trajkovic and Hedly method [206] is similar to SUSAN, and discriminates among points in USAN regions, edge points, and corner points.

SUSAN is also useful for noise suppression, and the bilateral filter [294], discussed in Chap. 2, is closely related to SUSAN. SUSAN uses fairly large circular windows; several implementations use 37 pixel radius windows. The FAST [130] detector is also similar to SUSAN, but uses a smaller 7×7 or 9×9 window and only some of the pixels in the region instead of all of them; FAST yields a local binary descriptor.

Fast, Faster, AGHAST

The FAST methods [130] are derived from SUSAN with respect to a bimodal segmentation goal. However, FAST relies on a connected set of pixels in a circular pattern to determine a corner. The connected region size is commonly 9 or 10 out of a possible 16; either number may be chosen, referred to as FAST9 and FAST10. FAST is known to be efficient to compute and fast to match; accuracy is also quite good. FAST can be considered a relative of the local binary pattern LBP.

FAST is not a scale-space detector, and therefore it may produce many more edge detections at the given scale than a scale-space method such as used in SIFT.

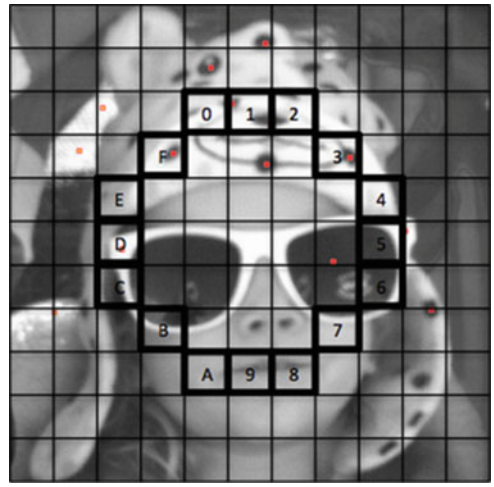


Figure 6.4 The FAST detector with a 16-element circular sampling pattern grid. Note that each pixel in the grid is compared against the center pixel to yield a binary value, and each binary value is stored in a bit vector

As shown in Fig. 6.4, FAST uses binary comparison with each pixel in a circular pattern against the center pixel using a threshold to determine if a pixel is less than or greater than the center pixel. The resulting descriptor is stored as a contiguous bit vector in order from 0 to 15. Also, due to the circular nature of the pixel compare pattern, it is possible to retrofit FAST and store the bit vector in a rotational-invariant representation, as demonstrated by the RILBP descriptor discussed later in this chapter; see Fig. 6.11.

Local Curvature Methods

Local curvature methods [200–204] are among the early means of detecting corners, and some local curvature methods are the first known to be reliable and accurate in tracking corners over scale variations [202]. Local curvature detects points where the gradient magnitude and the local surface curvature are both high. One approach taken is a differential method, computing the product of the gradient magnitude and the level curve curvature together over scale space, and then selecting the maxima and minima absolute values in scale and space. One formulation of the method is shown here.

$$\widetilde{\alpha}\left\{x, y; t\right\} = L_x^2 L_{yy} + L_y^2 L_{xx} - 2L_x L_y L_{xy}$$

Various formulations of the basic algorithm can be taken depending on the curvature equation used. To improve scale invariance and noise sensitivity, the method can be modified using a normalized formulation of the equation over scale space, as follows:

$$\widetilde{\alpha}_{\text{norm}}\left\{x, y; t\right\} = t^{2\gamma}\left(L_x^2 L_{yy} + L_y^2 L_{xx} - 2L_x L_y L_{xy}\right)$$

where

$$\gamma = 0.875$$

At larger scales, corners can be detected with less sharp and more rounded features, while at lower scales or at unity scale sharper corners over smaller areas are detected. The Wang and Brady method [205] also computes interest points using local curvature on the 2D surface, looking for inflection points where the surface curvature changes rapidly.

Morphological Interest Regions

Interest points can be determined from a pipeline of morphological operations, such as thresholding followed by combinations or erosion and dilation to smooth, thin, grown, and shrink pixel groups. If done correctly for a given application, such morphological features can be scale and rotation invariant. Note that the simple morphological operations alone are not enough; for example, erode left unconstrained will shrink regions until they disappear. So intelligence must be added to the morphology pipeline to control the final region size and shape. For polygon shape descriptors, morphological interest points define the feature, and various image moments are computed over the feature, as described in Chap. 3 and also in the section on polygon shape descriptors later in this chapter.

Morphological operations can be used to create interest regions on binary, gray scale, or color channel images. To prepare gray scale or color channel images for morphology, typically some sort of preprocessing is used, such as pixel remapping, LUT transforms, or histogram equalization. (These methods were discussed in Chap. 2.) For binary images and binary morphology approaches, binary thresholding is a key preprocessing step. Many binary thresholding methods have been devised, ranging from simple global thresholds to statistical and structural kernel-based local methods.

Note that the morphological interest region approach is similar to the maximally stable extrema region (MSER) feature descriptor method discussed later in the section on polygon shape descriptors, since both methods look for connected groups of pixels at maxima or minima. However, MSER does not use morphology operators.

A few examples of morphological and related operation sequences for interest region detection are shown in Fig. 6.5, and many more can be devised.

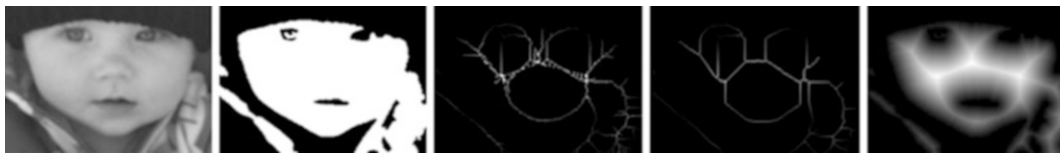


Figure 6.5 Morphological methods to find interest regions. (*Left* to *right*) Original image, binary thresholded and segmented image using Chan Vese method, skeleton transform, pruned skeleton transform, and distance transform image. Note that binary thresholding requires quite a bit of work to set parameters correctly for a given application

Feature Descriptor Survey

This section provides a survey and observations about a few representative feature descriptor methods, with no intention to directly compare descriptors to each other. For more detailed information on analytical methods for comparing feature descriptors, see also Lempitsky [844], and Huang [889]. In practice, the feature descriptor methods are often modified and customized, and often several descriptors are used together as a multivariate descriptor to increase confidence, see Varma

[770], Vedaldi [887], and Gehler [792] for more details about multivariate descriptors, and applying boosting to weight the descriptors in the classifier (i.e. a multi-stage classifier). The goal of this survey is to examine a range of feature descriptor approaches from each feature descriptor family from the taxonomy that was presented in Chap. 5:

- Local binary descriptors
- Spectra descriptors
- Basis space descriptors
- Polygon shape descriptors
- 3D, 4D, and volumetric descriptors

For key feature descriptor methods, we provide here a summary analysis:

- **General Vision Taxonomy and FME**: covering feature attributes including spectra, shape, and pattern, single or multivariate, compute complexity criteria, data types, memory criteria, matching method, robustness attributes, and accuracy.
- **General Robustness Attributes**: covering invariance attributes such as illumination, scale, perspective, and many others.

No direct comparisons are made between feature descriptors here, but ample references are provided to the literature for detailed comparisons and performance information on each method. See Table 8.2 for a comparison of the memory footprints for various feature descriptor methods in this survey, which is useful for performance analysis.

Local Binary Descriptors

This family of descriptors represents features as binary bit vectors. To compute the features, image pixel point-pairs are compared and the results are stored as binary values in a vector. Local binary descriptors are efficient to compute, efficient to store, and efficient to match using Hamming distance. In general, local binary pattern methods achieve very good accuracy and robustness compared to other methods.

A variety of local sampling patterns are used with local binary descriptors to set the pairwise point comparisons; see the section in Chap. 4 on local binary descriptor point-pair patterns for a discussion on local binary sampling patterns. We start this section on local binary descriptors by analyzing the local binary pattern (LBP) and some LBP variants, since the LBP is a powerful metric all by itself and is well known.

Local Binary Patterns

Local binary patterns (LBP) were developed in 1994 by Ojala et al. [165] as a novel method of encoding both pattern and contrast to define texture [161–165]. LBPs can be used as an image processing operator. The LBP creates a descriptor or texture model using a set of histograms of the local texture neighborhood surrounding each pixel. In this case, local texture is the feature descriptor.

The LBP metric is simple yet powerful; see Fig. 6.6. We cover some level of detail on LBPs, since there are so many applications for this powerful texture metric as a feature descriptor as well. Also, hundreds of researchers have added to the LBP literature [165] in the areas of theoretical foundations, generalizations into 2D and 3D, applied as a descriptor for face detection, and also applied to spatiotemporal applications such as motion analysis. LBP research remains quite active at this

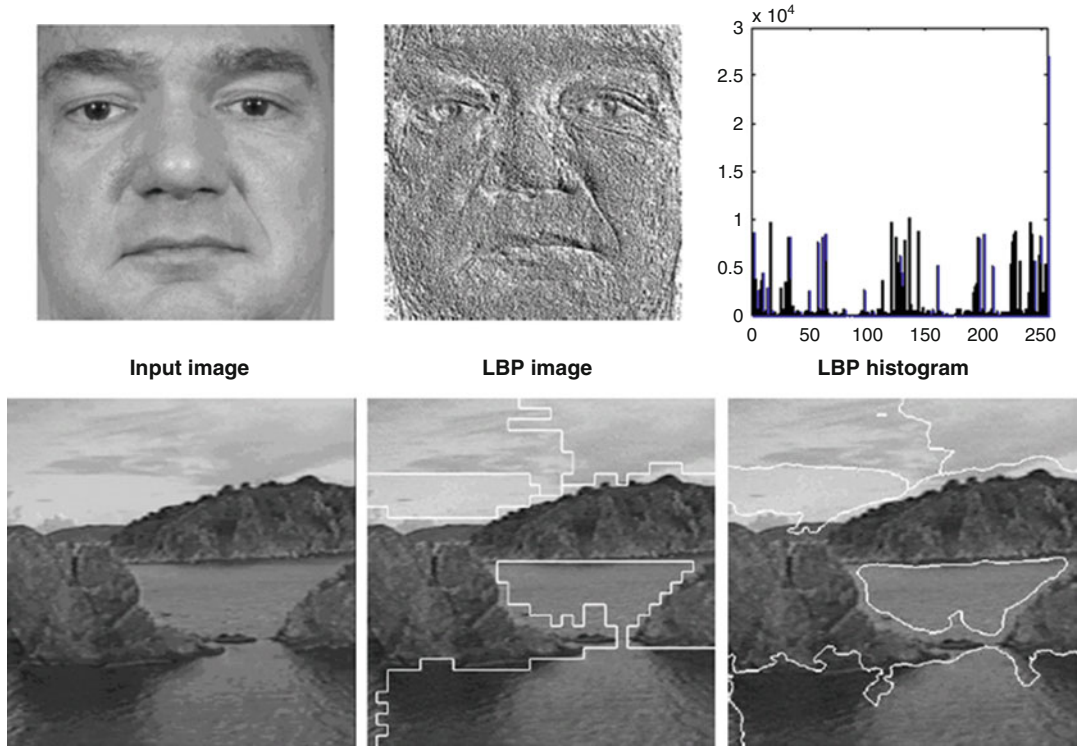


Figure 6.6 (*Above*) A local binary pattern representation of an image where the LBP is used as an image processing operator, and the corresponding histogram of cumulative LBP features. (*Bottom*) Segmentation results using LBP texture metrics. (Images courtesy and © Springer Press, from Computer Vision Using Local Binary Patterns, by Matti Pietikäinen and Janne Heikkilä [165])

time. In addition, the LBP is used as an image processing operator, and has been used as a feature descriptor retrofit in SIFT with excellent results, described in this chapter.

In its simplest embodiment, LBP has the goal of creating a binary coded neighborhood descriptor for a pixel. It does this by comparing each pixel against its neighbors using the $>$ operator and encoding the compare results (*1,0*) into a binary number, as shown in Fig. 6.8. LPB histograms from larger image regions can even be used as signals and passed into a 1D FFT to create a feature descriptor. The Fourier spectrum of the LBP histogram is rotational invariant; see Fig. 6.6. The FFT spectrum can then be concatenated onto the LBP histogram to form a multivariate descriptor, see Varma [770], Vedaldi [887], and Gehler [792] for more details about multivariate descriptors, and applying boosting to weight the features.

As shown in Fig. 6.6, the LBP is used as an image processing operator, region segmentation method, and histogram feature descriptor. The LBP has many applications. An LBP may be calculated over various sizes and shapes using various sizes of forming kernels. A simple 3×3 neighborhood provides basic coverage for local features, while wider areas and kernel shapes are used as well.

Assuming a 3×3 LBP kernel pattern is chosen, this means that there will be 8 pixel compares and up to 2^8 combinations of results for a 256-bin histogram possible. However, it has been shown [18] that reducing the 8-bit 256-bin histogram to use only 58 LBP bins based on *uniform patterns* is the optimal number. The 58 bins or uniform patterns are chosen to represent only two contiguous LBP patterns around the circle, which consists of two connected contiguous segments rather than all

256 possible pattern combinations [15, 165]. The same uniform pattern logic applies to LBPs of dimension larger than 8 bits. So uniform patterns provide both histogram space savings and feature compare-space optimization, since fewer features need be matched (58 instead of all 256).

LPB feature recognition may follow the steps shown in Fig. 6.7.

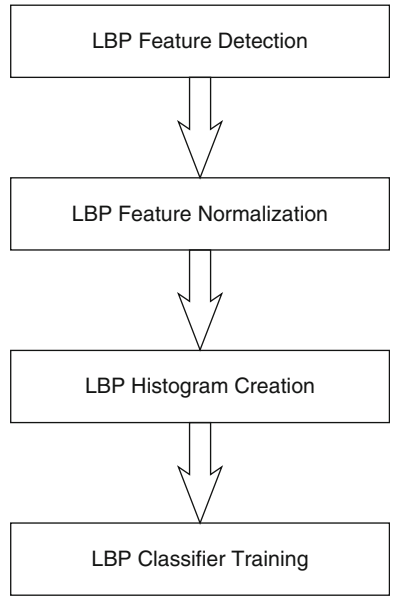


Figure 6.7 LBP feature flow for feature detection. (Image used by permission, © Intel Press, from Building Intelligent Systems)

The LBP is calculated by assigning a binary weighting value to each pixel in the local neighborhood and summing up the pixel compare results as binary values to create a composite LBP value. The LBP contains region information encoded in a compact binary pattern, as shown in Fig. 6.8, so the LBP is thus a binary coded neighborhood texture descriptor.

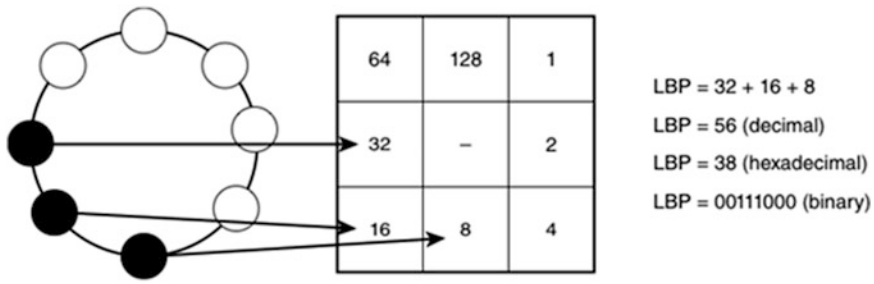


Figure 6.8 Assigned LBP weighting values. (Image used by permission, © Intel Press, from Building Intelligent Systems)

Assuming a 3 × 3 neighborhood is used to describe the LBP patterns, one may compare the 3 × 3 rectangular region to a circular region, suggesting 360° directionality at 45° increments, as shown in Fig. 6.9.

The steps involved in calculating a 3 × 3 LBP are illustrated in Fig. 6.10.

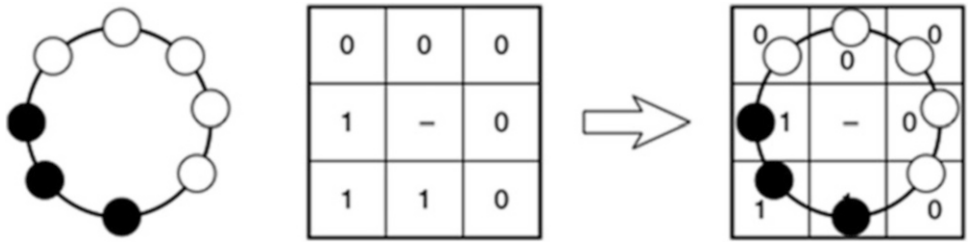
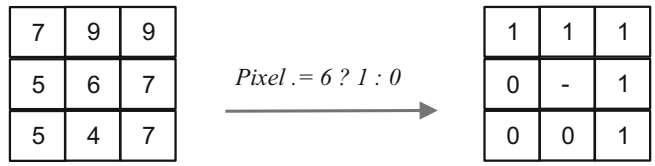


Figure 6.9 The concept of LBP directionality. (Image used by permission, © Intel Press, from Building Intelligent Systems)



Pixel[0,0](7) >= 6 ? 1 : 0 = 00000001
Pixel[1,0](9) >= 6 ? 1 : 0 = 00000010
Pixel[2,0](9) >= 6 ? 1 : 0 = 00000100
Pixel[2,1](7) >= 6 ? 1 : 0 = 00001000
Pixel[2,2](7) >= 6 ? 1 : 0 = 00010000
Pixel[1,2](4) >= 6 ? 1 : 0 = 00000000
Pixel[0,2](5) >= 6 ? 1 : 0 = 00000000
Pixel[0,1](5) >= 6 ? 1 : 0 = 00000000
LBP ***00011111***

Figure 6.10 LBP neighborhood comparison

Neighborhood Comparison

Each pixel in the 3 $\times$ 3 region is compared to the center pixel. If the pixel $\geq$ the center pixel, then the LBP records a bit value of 1 for that position, and a bit value of 0 otherwise. See Fig. 6.10

Histogram Composition

Each LBP descriptor over an image region is recorded in a histogram to describe the cumulative texture feature. Uniform LBP histograms would have 56 bins, since only single-connected regions are histogrammed.

Optionally Normalization

The final histogram can be reduced to a smaller number of bins using binary decimation for powers of two or some similar algorithm, such as 256 $\rightarrow$ 32. In addition, the histograms can be reduced in size by thresholding the range of contiguous bins used for the histogram—for example, by ignoring bins 1–64 if little or no information is binned in them.

Descriptor Concatenation

Multiple LBPs taken over overlapping regions may be concatenated together into a larger histogram feature descriptor to provide better discrimination.

LBP SUMMARY TAXONOMY

Spectra:	*Local binary*
Feature shape:	*Square*
Feature pattern:	*Pixel region compares with center pixel*
Feature density:	*Local 3 × 3 at each pixel*
Search method:	*Sliding window*
Distance function:	*Hamming distance*
Robustness:	*3 (brightness, contrast, *rotation for RILBP)*

Rotation Invariant LBP (RILBP)

To achieve rotational invariance, the rotation invariant LBP (RILBP) [165] is calculated by circular bitwise rotation of the local LBP to find the minimum binary value. The minimum value LBP is used as a rotation invariant signature and is recorded in the histogram bins. The RILBP is computationally very efficient.

To illustrate the method, Fig. 6.11 shows a pattern of three consecutive LBP bits; in order to make this descriptor rotation invariant, the value is *left-shifted* until a minimum value is reached.

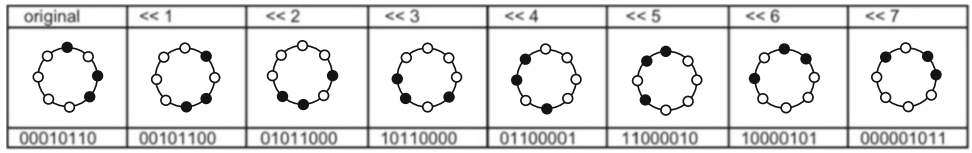


Figure 6.11 Method of calculating the minimum LBP by using circular bit shifting of the binary value to find the minimum value. The LBP descriptor is then rotation invariant

Note that many researchers [163, 164] are extending the methods used for LBP calculation to use refinements such as local derivatives, local median or mean values, trinary or quinary compare functions, and many other methods, rather than the simple binary compare function, as originally proposed.

Dynamic Texture Metric Using 3D LBPs

Dynamic textures are visual features that morph and change as they move from frame to frame; examples include waves, clouds, wind, smoke, foliage, and ripples. Two extensions of the basic LBP used for tracking such dynamic textures are discussed here: VLBP and LBP-TOP.

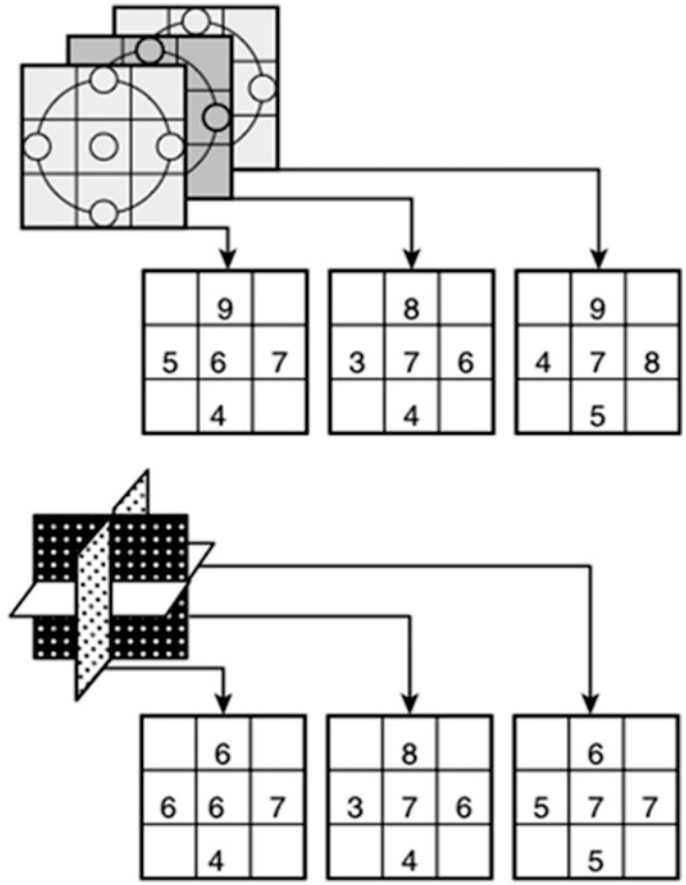


Figure 6.12 (*Top*) VLBP method [167] of calculating LBPs from parallel planes. (*Bottom*) LBP-TOP method [168] of calculating LBPs from orthogonal planes. (Image used by permission, © Intel Press, from Building Intelligent Systems)

Volume LBP (VLBP)

To create the VLBP [167] descriptor, first an image volume is created by stacking together at least three consecutive video frames into a volume 3D dataset. Next, three LBPs are taken centered on the selected interest point, one LBP from each parallel plane in the volume, into a summary volume LBP or VLBP, and the histogram of each orthogonal LBP is concatenated into a single dynamic descriptor vector, the VLBP. The VLPB can then be tracked from frame to frame and recalculated to account for dynamic changes in the texture from frame to frame. See Fig. 6.12.

LPB-TOP

The LBP-TOP [168] is created like the VLBP, except that instead of calculating the three individual LBPs from parallel planes, they are calculated from orthogonal planes in the volume (x,y,z) intersecting the interest point, as shown in Fig. 6.12. The 3D composite descriptor is the same size as the VLBP and contains three planes' worth of data. The histograms for each LBP plane are also concatenated for the LBP-TOP like the VLBP.

Other LBP Variants

As shown in Table 6.1, there are many variants of the LBP [165]. Note that the LBP has been successfully used as a replacement for SIFT, SURF, and also as a texture metric.

Table 6.1 LBP variants (from reference [165])

ULBP (Uniform LBP) Uses only 56 uniform bins instead of the full 256 bins possible with 8-bit pixels to create the histogram. The uniform patterns consist of contiguous segments of connected TRUE values.

RLBP (ROBUST LBP) Adds + scale factor to eliminate transitions due to noise (p1 - p2 + SCALE)

CS-LBP Circle-symmetric, half as many vectors an LBP, comparison of opposite pixel pairs vs. w/center pixel, useful to reduce LBP bin counts

LBP-HF Fourier spectrum descriptor + LBP

MLBP Median LBP Uses area median value instead of center pixel value for comparison

M-LBP Multiscale LBP combining multiple radii LBPs concatenated

MB-LBP Multiscale Block LBP; compare average pixel values in small blocks

SEMB-LBP: Statistically Effective MB-LBP (SEMB-LBP) uses the percentage in distributions, instead of the number of 0-1 and 1-0 transitions in the LBP and redefines the uniform patterns in the standard LBP. Used effectively in face recognition using GENTLE ADA-BOOSTing [531]

VLBP Volume LBP over adjacent video frames OR within a volume - concatenate histograms together to form a longer vector

LGBP (Local Gabor Binary Pattern) 40 or so Gabor filters are computed over a feature, LBPs are extracted and concatenated to form a long feature vector that is invariant over more scales and orientations

LEP Local Edge Patterns: Edge enhancement (Sobel) prior to standard LBP

EBP Elliptic Binary Pattern Standard LBP but over elliptical area instead of circular

EQP Elliptical Quinary Patterns - LBP extended from binary (2) level resolution to quinary (5) level resolution (-2,-1, 0,-1,2)

LTP - LBP extended over Ternary range to deal with near constant areas (-1, 0, 1)

LLBP Local line Binary Pattern - calculates LBP over line patterns (cross shape) and then calculates a magnitude metrics using SQRT of SQUARES of each X/Y dimension

TPLBP- [x5]three LBPs are calculated together: the basic LBP for the center pixel, plus two others around adjacent pixels so the total descriptor is a set of overlapping LBP's,

FPLBP- [x5]four LBPs are calculated together: the basic LBP for the center pixel, plus two others around adjacent pixels so the total descriptor is a set of overlapping LBP's, XPLBP –

**NOTE: The TPLBP and FPLBP method can be extended to 3,4,n dimensions in feature space. LARGE VECTORS.*

TBP - Ternary (3) Binary pattern, like LBP, but uses three levels of encoding (1,0,-1) to effectively deal with areas of equal or near equal intensity, uses twobinary patterns (one for + and one for -) concatenated together

ETLP - Elongated Ternary Local Patterns (elliptical + ternary[3] levels

FLBP - Fuzzy LBP where each pixel contributes to more than one bin

PLBP - Probabilistic LBP computes magnitude of difference between each pixel & center pixel (more compute, more storage)

SILTP - Scale invariant LBP using a 3 part piece-wise comparison function to compensate and support intensity scale invariance to deal with image noise

tLBP - Transition Coded LBP, where the encoding is clockwise between adjacent pixels in the LBP

(continued)

Table 6.1 (continued)

dLBP - Direction Coded LBP - similar to CSLBP, but stores both maxima and comparison info (is this pixel greater, less than, or maxima)

CBP - Centralized Binary pattern - center pixel compared to average of all nine kernel neighbors

S-LBP Semantic LBP done in a colorimetric-accurate space (like CIE LAB etc.) over uniform connected LBP circular patterns to find principal direction + arc length used to form a 2D histogram as the descriptor.

F-LBP - Fourier Spectrum of color distance from center pixel to adjacent pixels

LDP - Local Derivate Patterns (higher order derivatives) - basic

LBP is the first order directional derivative, which is combined with additional nth order directional derivatives concatenated into a histogram, more sensitive to noise of course

BLBP - Baysian LBP - combination of LBP and LTP together using Baysian methods to optimize towards a more robust pattern

FLS - Filtering, Labeling and Statistical Framework for LBP comparison, translates LBP's or any type of histogram descriptor into vector space allowing efficient comparison "A Bayesian Local Binary Pattern Texture Descriptor"

MB-LBP Multiscale Block LBP - compare average pixel values in small blocks instead of individual pixels, thus a 3x3 pixel PBL will become a 9x9 block LBP where each block is a 3x3 region. The histogram is calculated by scaling the image and creating a rendering at each scale and creating a histogram of each scaled image and concatenating the histograms together.

PM-LBP Pyramid Based MultiStructured LBP - used 5 templates to extract different structural info at varying levels 1) Gaussian filters, 4 anisotrophic filters to detect gradient directions

MSLBF - Multiscale Selected Local Binary Features

RILBP - Rotation Invariant LBP rotates the bins (binary LBP value) until minimum value is achieved, the max value is considered rotational invariant. This is the most widely used method for LBP rotational invariance.

ALBP - Adaptive LBP for rotational invariance, instead of shifting to a maximal value as in the standard LBP method, find the dominant vector orientation and shift the vector to the dominant vector orientation

LBPV - Local binary pattern variance - uses local area variance to weight pixel contribution to the LBP, align features to principal orientations, determine non-dominant patterns and reduce their contribution.

OCLBP - Opponent Color LBP - describes color and texture together - each color channel LBP is converted, then opposing color channel LBP's are converted by using one color as the center pixel and another color as the neighborhood, so a total of 9 RGB combination LBP patterns are considered.

SDMCLBP - SDM (co -LBP images for each color are used as the basis for generating occurrence matrices, and then Haralick features are extracted from the images to form a multi dimensional feature space.

MSCLBP - Multi Scale Color Local Binary Patterns (concatenate 6 histograms together)- USES COLOR SPACE COMPONENTS

HUE-LBP OPPONENT-LBP (ALL 3 CHANNELS) nOPPONENT-LBP (COMPUTED OVER 2 CHANNELS), light intensity change, intensity shift, intensity change+shift, color-change color-shift, DEFINE SIX NEW OPERATORS: transformed color LBP (RGB)[subtract mean, divide by STD DEV], opponent LBP, nOpponent LBP, Hue LBP, RGB-LBP, nRGB-LBP [x8] "Multi-scale Color Local Binary Patterns for Visual Object Classes Recognition", Chao ZHU, Charles-Edmond BICHOT, Liming CHEN

3D histograms - 3DRGBLBP [best performance, high memory footprint] - 3D histogram computed over RGB-LBP color image space using uniform pattern minimization to yield 10 levels or patterns per color yielding a large descriptor: 10 x 10 x 10 = 1000 descriptors.

LATCH - *LATCH: Learned Arrangements of Three Patch Codes* [871]

Census

The Census transform [169] is basically an LBP, and like a population census, it uses simple greater-than and less-than queries to count and compare results. Census records pixel comparison results made between the center pixel in the kernel and the other pixels in the kernel region. It employs comparisons and possibly a threshold, and stores the results in a binary vector. The Census transform also uses a feature called the *rank value scalar*, which is the number of pixel values less than the center pixel. The Census descriptor thus uses both a bit vector and a rank scalar.

CENSUS SUMMARY VISION TAXONOMY

Spectra:	*Local binary + scalar ranking*
Feature shape:	*Square*
Feature pattern:	*Pixel region compares with center pixel*
Feature density:	*Local 3 × 3 at each pixel*
Search method:	*Sliding window*
Distance function:	*Hamming distance*
Robustness:	*2 (brightness, contrast)*

Modified Census Transform

The Modified Census trasform (MCT) [197] seeks to improve the local binary pattern robustness of the original Census transform. The method uses an ordered comparison of each pixel in the 3×3 neighborhood against the mean intensity of all the pixels of the 3×3 neighborhood, generating a binary descriptor bit vector with bit values set to an intensity lower than the mean intensity of all the pixels. The bit vector can be used to create an MCT image using the MCT value for each pixel. See Fig. 6.13.

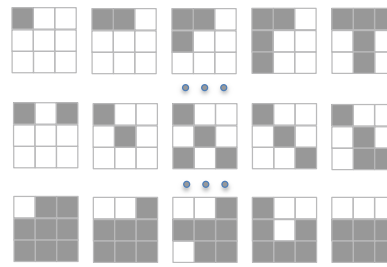


Figure 6.13 Abbreviated set of 15 out of a possible 511 possible binary patterns for a 3×3 MCT. The structure kernels in the pattern set are the basis set of the MCT feature space comparison. The structure kernels form a pattern basis set which can represent lines, edges, corners, saddle points, semicircles, and other patterns

As shown in Fig. 6.13, the MCT relies on the full set of possible 3×3 binary patterns ($2^9 - 1$ or 511 variations) and uses these as a kernel index into the binary patterns as the MCT output, since each binary pattern is a unique signature by itself and highly discriminative. The end result of the MCT is analogous to a nonlinear filter that assigns the output to any of the $2^9 - 1$ patterns in the kernel index. Results show that the MCT results are better than the basic CT for some types of object recognition [197].

BRIEF

As described in Chap. 4, in the section on local binary descriptor point-pair patterns, and illustrated in Fig. 4.11, the BRIEF [124, 125] descriptor uses a random distribution pattern of 256 point-pairs in a local 31 $\times$ 31 region for the binary comparison to create the descriptor. One key idea with BRIEF is to select random pairs of points within the local region for comparison.

BRIEF is a local binary descriptor and has achieved very good accuracy and performance in robotics applications [195]. BRIEF and ORB are closely related; ORB is an oriented version of BRIEF, and the ORB descriptor point-pair pattern is also built differently than BRIEF. BRIEF is known to be not very tolerant of rotation.

BRIEF SUMMARY TAXONOMY

Spectra:	*Local binary*
Feature shape:	*Square centered at interest point*
Feature pattern:	*Random local pixel point-pair compares*
Feature density:	*Local 31 $\times$ 31 at interest points*
Search method:	*Sliding window*
Distance function:	*Hamming distance*
Robustness:	*2 (brightness, contrast)*

ORB

ORB [126] is an acronym for Oriented BRIEF, and as the name suggests, ORB is based on BRIEF and adds rotational invariance to BRIEF by determining corner orientation using FAST9, followed by a Harris corner metric to sort the keypoints; the corner orientation is refined by intensity centroids using Rosin's method [53]. The FAST, Harris, and Rosin processing are done at each level of an image pyramid scaled with a factor of 1.4, rather than the common octave pyramid scale methods. ORB is discussed in some detail in Chap. 4, in the section on local binary descriptor point-pair patterns, and is illustrated in Fig. 4.11.

It should be noted that ORB is a highly optimized and very well engineered descriptor, since the ORB authors were keenly interested in compute speed, memory footprint, and accuracy. Many of the descriptors surveyed in this section are primarily research projects, with less priority given to practical issues, but ORB focuses on optimizing and practical issues.

Compared to BRIEF, ORB provides an improved training method for creating the local binary patterns for pairwise pixel point sampling. While BRIEF uses random point pairs in a 31 $\times$ 31 window, ORB goes through a training step to find uncorrelated point pairs in the window with high variance and means 0.5, which is demonstrated to work better. For details on visualizing the ORB patterns, see Fig. 4.11.

For correspondence search, ORB uses multi-probe locally sensitive hashing (MP-LSH), which searches for matches in neighboring buckets when a match fails, rather than renavigating the hash tree. The authors report that MP-LSH requires fewer hash tables, resulting in a lower memory footprint. MP-LSH also produces more uniform hash bucket sizes than BRIEF. Since ORB is a binary descriptor based on point-pair comparisons, Hamming distance is used for correspondence.

ORB is reported to be an order of magnitude faster than SURF, and two orders of magnitude faster than SIFT, with comparable accuracy. The authors provide impressive performance results in a test of over 24 NTSC resolution images on the Pascal dataset [126].

ORB*	SURF	SIFT
15.3ms	217.3ms	5228.7ms

**Results reported as measured in reference* [126].

ORB SUMMARY TAXONOMY

Spectra:	*Local binary + orientation vector*
Feature shape:	*Square*
Feature pattern:	*Trained local pixel point-pair compares*
Feature density:	*Local 31 × 31 at interest points*
Search method:	*Sliding window*
Distance function:	*Hamming distance*
Robustness:	*3 (brightness, contrast, rotation, limited scale)*

BRISK

BRISK [123, 135] is a local binary method using a circular-symmetric pattern region shape and a total of 60 point-pairs as line segments arranged in four concentric rings, as shown in Fig. 4.10 and described in detail in Chap. 4. The method uses point-pairs of both short segments and long segments, and this provides a measure of scale invariance, since short segments may map better for fine resolution and long segments may map better at coarse resolution.

The brisk algorithm is unique, using a novel FAST detector adapted to use scale space, reportedly achieving an order of magnitude performance increase over SURF with comparable accuracy. Here are the main computational steps in the algorithm:

- Detects keypoints using FAST or AGHAST based selection in scale space.
- Performs Gaussian smoothing at each pixel sample point to get the point value.
- Makes three sets of pairs: long pairs, short pairs, and unused pairs (the unused pairs are not in the long pair or the short pair set; see Fig. 4.10).
- Computes gradient between long pairs, sums gradients to determine orientation.
- Uses gradient orientation to adjust and rotate short pairs.
- Creates binary descriptor from short pair point-wise comparisons.

BRISK SUMMARY TAXONOMY

Spectra:	*Local binary + orientation vector*
Feature shape:	*Square*
Feature pattern:	*Trained local pixel point-pair compares*
Feature density:	*Local 31 × 31 at FAST interest points*
Search method:	*Sliding window*
Distance function:	*Hamming distance*
Robustness:	*4 (brightness, contrast, rotation, scale)*

FREAK

FREAK [122] uses a novel foveal-inspired multiresolution pixel pair sampling shape with trained pixel pairs to mimic the design of the human eye as a coarse-to-fine descriptor, with resolution highest in the center and decreasing further into the periphery, as shown in Fig. 4.9. In the opinion of this author, FREAK demonstrates many of the better design approaches to feature description; it combines performance, accuracy, and robustness. Note that FREAK is fast to compute, has good discrimination compared to other local binary descriptors such as LBP, Census, BRISK, BRIEF, and ORB, and compares favorably with SIFT.

The FREAK feature training process involves determining the point-pairs for the binary comparisons based on the training data, as shown in Fig. 4.9. The training method allows for a range of descriptor sampling patterns and shapes to be built by weighting and choosing sample points with high variance and low correlation. Each sampling point is taken from the overlapping circular regions, where the value of each sampling point is the Gaussian average of the values in each region. The circular regions are designed in concentric circles of 6 regions in each circle, with small regions in the center, and larger regions towards the edge, similar to the biological retinal distribution of receptor cells with some overlap to adjacent regions, which improves accuracy.

The feature descriptor is thus designed in a coarse-to-fine cascade of four groups of 16 byte coarse-to-fine descriptors containing pixel-pair binary comparisons stored in a vector. The first 16 bytes, the coarse resolution set in the cascade, is normally sufficient to find 90 % of the matching features and to discard nonmatching features. FREAK uses 45 point pairs for the descriptor from a 31 $\times$ 31 pixel patch sampling region.

By storing the point-pair comparisons in four cascades of decreasing resolution pattern vectors, the matching process proceeds from coarse to fine, mimicking the human visual system's saccadic search mechanism, allowing for accelerated matching performance when there is early success or rejection in the matching phase. In summary, the FREAK approach works very well.

FREAK SUMMARY TAXONOMY

Spectra:	*Local binary coarse-to-fine + orientation vector*
Feature shape:	*Square*
Feature pattern:	*31 $\times$ 31 region pixel point-pair compares*
Feature density:	*Sparse local at AGAST interest points*
Search method:	*Sliding window over scale space*
Distance function:	*Hamming distance*
Robustness:	*6 (brightness, contrast, rotation, scale, viewpoint, blur)*

Spectra Descriptors

Compared to the local binary descriptor group, the spectra group of descriptors typically involves more intense computations and algorithms, often requiring floating point calculations, and may consume considerable memory. In this taxonomy and discussion, *spectra* is simply a quantity that can be measured or computed, such as light intensity, color, local area gradients, local area statistical features and moments, surface normals, and sorted data such 2D or 3D histograms of any spectral

type, such as histograms of local gradient direction. Many of the methods discussed in this section use local gradient information.

Local binary descriptors, as discussed in the previous section, are an attempt to move away from more costly spectral methods to reduce power and increase performance. Local binary descriptors in many cases offer similar accuracy and robustness to the more compute-intensive spectra methods.

SIFT

The Scale Invariant Feature Transform (SIFT) developed by Lowe [153, 170] is the most well-known method for finding interest points and feature descriptors, providing invariance to scale, rotation, illumination, affine distortion, perspective and similarity transforms, and noise. Lowe demonstrates that by using several SIFT descriptors together to describe an object, there is additional invariance to occlusion and clutter, since if a few descriptors are occluded, others will be found [153]. We provide some detail here on SIFT since it is well designed and well known.

SIFT is commonly used as a benchmark against which other vision methods are compared. The original SIFT research paper by author David Lowe was initially rejected several times for publication by the major computer vision journals, and as a result Lowe filed for a patent and took a different direction. According to Lowe, "By then I had decided the computer vision community was not interested, so I applied for a patent and intended to promote it just for industrial applications."[1] Eventually, the SIFT paper was published and went on to become the most widely cited article in computer vision history!

SIFT is a complete algorithm and processing pipeline, including both an interest point and a feature descriptor method. SIFT includes stages for selecting center-surrounding circular weighted Difference of Gaussian (DoG) maxima interest points in scale space to create scale-invariant keypoints (a major innovation), as illustrated in Fig. 6.14. Feature descriptors are computed surrounding the scale-invariant keypoints. The feature extraction step involves calculating a binned Histogram Of Gradients (HOG) structure from local gradient magnitudes into Cartesian rectangular bins, or into log polar bins using the GLOH variation, at selected locations centered around the maximal response interest points derived over several scales.

The descriptors are fed into a matching pipeline to find the nearest distance ratio metric between closest match and second closest match, which considers a primary match and a secondary match together and rejects both matches if they are too similar, assuming that one or the other may be a false match. The local gradient magnitudes are weighted by a strength value proportional to the pyramid scale level, and then binned into the local histograms. In summary, SIFT is a very well thought out and carefully designed multi-scale localized feature descriptor.

A variation of SIFT for color images is known as CSIFT [171].

Here is the basic SIFT descriptor processing flow (note: the matching stage is omitted since this chapter is concerned with feature descriptors and related metrics):

1. Create a Scale Space Pyramid

[1] http://yann.lecun.com/ex/pamphlets/publishing-models.html.

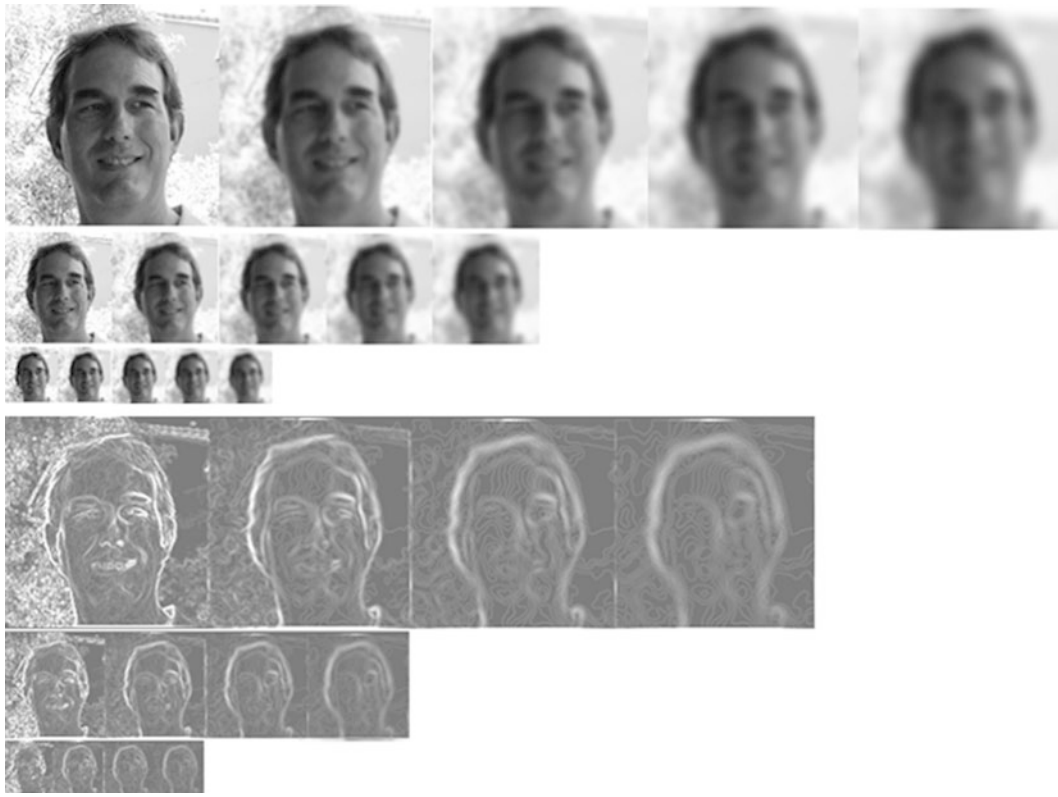


Figure 6.14 (*Top*) Set of Gaussian Images obtained by convolution with a Gaussian kernel and the corresponding set of DoG images. (*Bottom*) In octave sets. The DOG function approximates a LOG gradient, or tunable bypass filter. Matching features against the various images in the scaled octave sets yields scale invariant features

An octave scale ***n/2*** image pyramid is used with Gaussian filtered images in a scale space. The amount of Gaussian blur is proportional to the scale, and then the Difference of Gaussians (DoG) method is used to capture the interest point extrema maxima and minima in adjacent images in the pyramid. The image pyramid contains five levels. SIFT also uses a double-scale first pyramid level using pixels at two times the original magnification to help preserve fine details. This technique increases the number of stable keypoints by about four times, which is quite significant. Otherwise, computing the Gaussian blur across the original image would have the effect of throwing away the high-frequency details. See Figs. 6.15 and 6.16.

2. Identify Scale-Invariant Interest Points

As shown in Fig. 6.16, the candidate interest points are chosen from local maxima or minima as compared between the 26 adjacent pixels in the DOG images from the three adjacent octaves in the pyramid. In other words, the interest points are scale invariant.

The selected interest points are further qualified to achieve invariance by analyzing local contrast, local noise, and local edge presence within the local 26 pixel neighborhood. Various methods may be used beyond those in the original method, and several techniques are used together to select the best interest points, including local curvature interpolation over small regions, and balancing edge responses to include primary and secondary edges. The keypoints are localized to sub-pixel precision over scale and space. The complete interest points are thus invariant to scale.

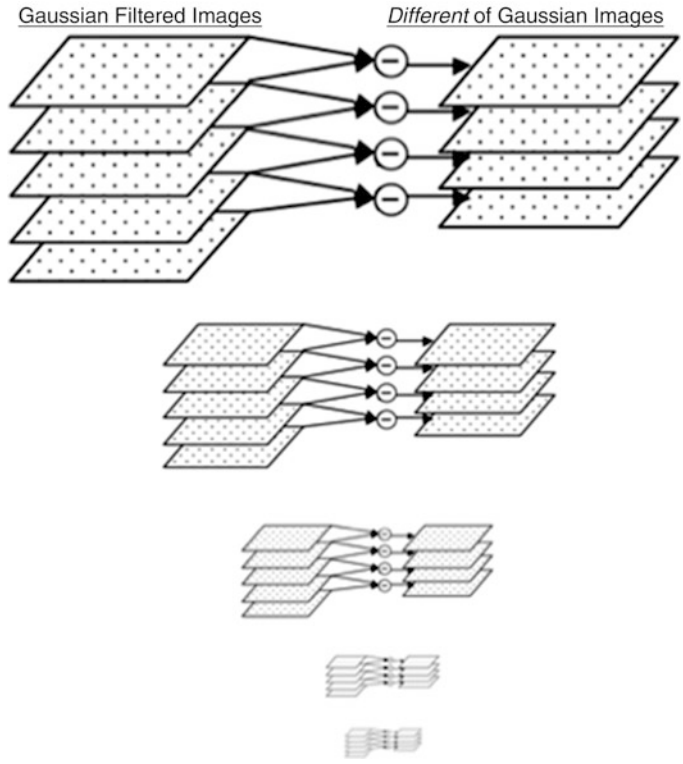


Figure 6.15 SIFT DoG as the simple arithmetic difference between the Gaussian filtered images in the pyramid scale

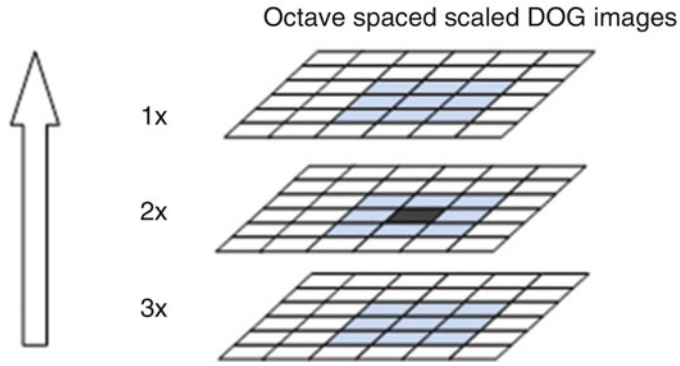


Figure 6.16 SIFT interest point or keypoint detection using scale invariant extrema detection, where the dark pixel in the middle octave is compared within a $3 \times 3 \times 3$ area against its 26 neighbors in adjacent DOG octaves, which includes the eight neighbors at the local scale plus the nine neighbors at adjacent octave scales (up or down)

3. Create Feature Descriptors

A local region or patch of size 16×16 pixels surrounding the chosen interest points is the basis of the feature vector. The magnitude of the local gradients in the 16×16 patch and the gradient orientations are calculated and stored in a HOG (Histogram of Gradients) feature vector, which is

weighted in a circularly symmetric fashion to downweight points farther away from the center interest point around which the HOG is calculated using a Gaussian weighting function.

As shown in Fig. 6.17, the 4 × 4 gradient binning method allows for gradients to move around in the descriptor and be combined together, thus contributing invariance to various geometric distortions that may change the position of local gradients, similar to the human visual system treatment of the 3D position of gradients across the retina [240]. The SIFT HOG is reasonably invariant to scale, contrast, and rotation. The histogram bins are populated with gradient information using trilinear interpolation, and normalized to provide illumination and contrast invariance.

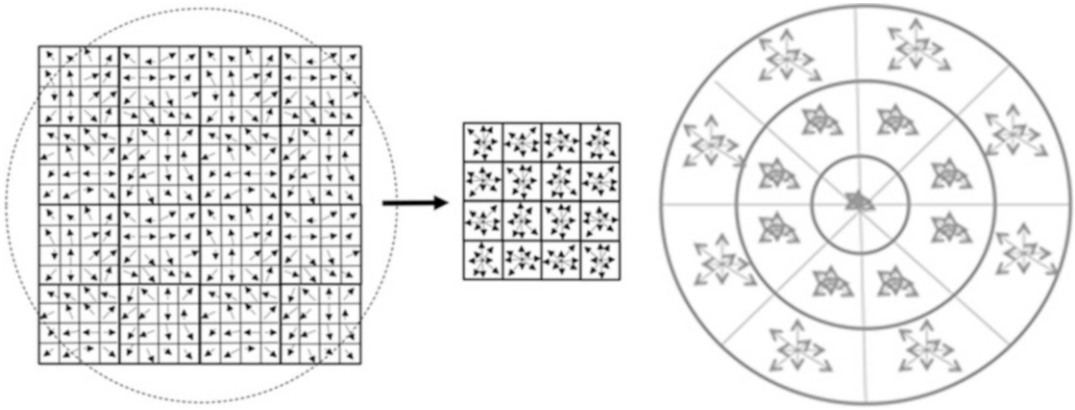


Figure 6.17 (*Left* and *center*) Gradient magnitude and direction binned into histograms for the SIFT HOG, note the circle over the bin region on the left image suggests how SIFT weights bins farther from center less than bins closer to the center, (*Right*) GLOH descriptors

SIFT can also be performed using a variant of the HOG descriptor called the Gradient Location and Orientation Histogram (GLOH), which uses a log polar histogram format instead of the Cartesian HOG format; see Fig. 6.17. The calculations for the GLOH log polar histogram are straightforward, as shown below from the Cartesian coordinates used for the Cartesian HOG histogram, where the vector magnitude is the hypotenuse and the angle is the arctangent.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - (x,y-1))^2}$$
$$\theta(x,y) = \text{TAN}^{-1}(L(x,y+1) - L(x,y-1))/(L(x+1,y) - (x-1,y))$$

As shown in Fig. 6.17, SIFT HOG and GLOH are essentially 3D histograms, and in this case the histogram bin values are gradient magnitude and direction. The descriptor vector size is thus 4 × 4 × 8 = 128 bytes. The 4 × 4 descriptor (center image) is a set of histograms of the combined eight-way gradient direction and magnitude of each 4 × 4 group in the left image, in Cartesian coordinates, while the GLOH gradient magnitude and direction are binned in polar coordinate spaced into 17 bins over a greater binning region. SIFT-HOG (left image) also uses a weighting factor to smoothly reduce the contribution of gradient information in a circularly symmetric fashion with increasing distance from the center.

Table 6.2 SIFT compute complexity (from Vinukonda [172])

SIFT Pipeline Step	Complexity	Number of Operations
Gaussian blurring pyramid	$\ominus N^2\ U^2\ \ s$	$4N^2\ W^2\ s$
Difference of Gaussian pyramid	$\ominus sN^2$	$4N^2\ s$
Scale-space extrema detection	$\ominus sN^2$	$104sN^2$
Keypoint detection	$\ominus \alpha sN^2$	$100s\alpha N^2$
Orientation assignment	$\ominus sN^2(1-\alpha\beta)$	$48sN^2$
Descriptor generation	$\ominus(x^2\ N^2(\alpha\beta+\gamma))$	$\ominus 1520x^2\ N^2(\alpha\beta+\gamma)\ N^2$

Overall compute complexity for SIFT is high [172], as shown in Table 6.2. Note that feature description is most compute-intensive owing to all the local area gradient calculations for orientation assignment and descriptor generation including histogram binning with trilinear interpolation. The gradient orientation histogram developed in SIFT is a key innovation that provides substantial robustness.

The resulting feature vector for SIFT is 128 bytes. However, methods exist to reduce the dimensionality and vary the descriptor, which are discussed next.

SIFT SUMMARY TAXONOMY

Spectra: *Local gradient magnitude + orientation*
Feature shape: *Square, with circular weighting*
Feature pattern: *Square with circular-symmetric weighting*
Feature density: *Sparse at local 16 × 16 DoG interest points*
Search method: *Sliding window over scale space*
Distance function: *Euclidean distance (*or Hellinger distance with RootSIFT retrofit)*
Robustness: *6 (brightness, contrast, rotation, scale, affine transforms, noise)*

SIFT-PCA

The SIFT-PCA method developed by Ke and Suthankar [175] uses an alternative feature vector derived using principal component analysis (PCA), based on the normalized gradient patches rather than the weighted and smoothed histograms of gradients, as used in SIFT. In addition, SIFT-PCA reduces the dimensionality of the SIFT descriptor to a smaller set of elements. SIFT originally was reported using 128 vectors, but using SIFT-PCA the vector is reduced to a smaller number such as 20 or 36.

The basic steps for SIFT-PCA are as follows:

1. Construct an eigenspace based on the gradients from the local 41 × 41 image patches resulting in a 3042 element vector; this vector is the result of the normal SIFT pipeline.
2. Compute local image gradients for the patches.
3. Create the reduced-size feature vector from the eigenspace using PCA on the covariance matrix of each feature vector.

SIFT-PCA is shown to provide some improvements over SIFT in the area of robustness to image warping, and the smaller size of the feature vector results in faster matching speed. The authors note that while PCA in general is not optimal as applied to image patch features, the method works well for the SIFT style gradient patches that are oriented and localized in scale space [175].

SIFT-GLOH

The Gradient Location and Orientation Histogram (GLOH) [136] method uses polar coordinates and radially distributed bins rather than the Cartesian coordinate style histogram binning method used by SIFT. It is reported to provide greater accuracy and robustness over SIFT and other descriptors for some ground truth datasets [136]. As shown in Fig. 6.17, GLOH uses a set of 17 radially distributed bins to sum the gradient information in polar coordinates, yielding a 272-bin histogram. The center bin is not direction oriented. The size of the descriptor is reduced using PCA. GLOH has been used to retrofit SIFT.

SIFT-SIFER Retrofit

The Scale Invariant Feature Detector with Error Resilience (SIFER) [216] method provides alternatives to the standard SIFT pipeline, yielding measurable accuracy improvements reported to be as high as 20 % for some criteria. However, the accuracy comes at a cost, since the performance is about twice as slow as SIFT. The major contributions of SIFER include improved scale-space treatment using a higher granularity image pyramid representation, and better scale-tuned filtering using a cosine modulated Gaussian filter.

The major steps in the method are shown in Table 6.3. The scale-space pyramid is blurred using a cosine modulated Gaussian (CMG) filter, which allows each scale of the octave to be subdivided into six scales, so the result is better scale accuracy.

Table 6.3 Comparison of SIFT, SURF, and SIFER pipelines (adapted from [216])

	SIFT	SURF	SIFER
Scale Space Filtering	Gaussian 2nd derivative	Gaussian 2nd derivative	Cosine Modulated Gaussian
Detector	LoG	Hessian	Wavelet Modulus Maxima
Filter approximation level	OK accuracy	OK accuracy	Good accuracy
Optimizations	DoG for gradient	Integral images, constant time	Convolution, constant time
Image up-sampling	2x	2x	Not used
Sub-sampling	Yes	Yes	Not used

Since the performance of the CMG is not good, SIFER provides a fast approximation method that provides reasonable accuracy. Special care is given to the image scale and the filter scale to increase accuracy of detection, thus the cosine is used as a bandpass filter for the Gaussian filter to match the scale as well as possible, tuning the filter in a filter bank over scale space with well-matched filters for each of the six scales per octave. The CMG provides more error resilience than the SIFT Gaussian second derivative method.

SIFT CS-LBP Retrofit

The SIFT-CSLBP retrofit method [165, 194] combines the best attributes of SIFT and the center symmetric LBP (CS-LBP) by replacing the SIFT gradient calculations with much more compute-

efficient LBP operators, and by creating similar histogram-binned orientation feature vectors. LBP is computationally simpler both to create and to match than the SIFT descriptor.

The CS-LBP descriptor begins by applying an adaptive noise-removal filter (a Weiner filter is the variety used in this work) to the local patch for adaptive noise removal, which preserves local contrast. Rather than computing all 256 possible 8-bit local binary patterns, the CS-LBP only computes 16 center symmetric patterns for reduced dimensionality, as shown in Fig. 6.18.

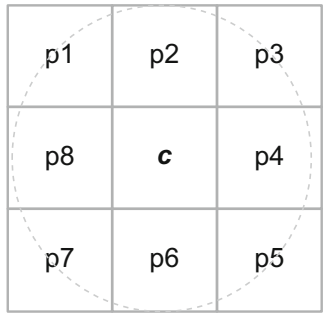


Figure 6.18 CS-LBP sampling pattern for reduced dimensionality

Instead of weighting the histogram bins using the SIFT circular weighting function, no weighting is used, which reduces compute. Like SIFT, the CS-LBP binning method uses a 3×3 region Cartesian grid; simpler bilinear interpolation for binning is used, rather than trilinear, as in SIFT. Overall, the CS-LCP retrofit method simplifies the SIFT compute pipeline and increases performance with comparable accuracy; greater accuracy is reported for some datasets. See Table 6.4.

Table 6.4 SIFT and CSLBP retrofit performance (as per reference [194])

	Feature extraction	Descriptor construction	Descriptor normalization	Total ms time
CS-LBP 256	0.1609	0.0961	0.007	0.264
CS-LBP 128	0.1148	0.0749	0.0022	0.1919
SIFT 128	0.4387	0.1654	0.0025	0.6066

RootSIFT Retrofit

The RootSift method [166] provides a set of simple, key enhancements to the SIFT pipeline, resulting in better compute performance and slight improvements in accuracy, as follows:

- **Hellinger distance**: RootSIFT uses a simple performance optimization of the SIFT object retrieval pipeline using Hellinger distance instead of Euclidean distance for correspondence. All other portions of the SIFT pipeline remain the same; K-means is still employed to build the feature vector set, and other approximate nearest neighbor methods may still be used as well for larger feature vector sets. The authors claim a simple modification to SIFT code to perform the Hellinger distance optimization instead of Euclidean distance can be a simple set of one-line changes to the code. Other enhancements in RootSIFT are optional, discussed next.

- **Feature augmentation**: This method increases total recall. Developed by Turcot and Lowe [324], it is applied to the features. Feature vectors or visual words from similar views of the same object in the database are associated into a graph used for finding correspondence among similar features, instead of just relying on a single feature.
- **Discriminative query expansion (DQE):** This method increases query expansion during training. Feature vectors within a region of proximity are associated by averaging into a new feature vector useful for requeries into the database, using both positive and negative training data in a linear SVM; better correspondence is reported in reference [166].

By combining the three innovations described above into the SIFT pipeline, performance, accuracy, and robustness are shown to be significantly improved.

Table 6.5 Major differences between CenSurE and SIFT and SURF (adapted from reference [177])

	CenSurE	SIFT	SURF
Resolution	Every pixel	Pyramid sub-sampled	Pyramid sub-sampled
Edge filter method	Harris	Hessian	Hessian
Scale space extrema method	Laplace, Center Surround	Laplace, DOG	Hessian, DOB
Rotational invariance	Approximate	yes	no
Spatial resolution in scale	Full	subsampled	Subsampled

CenSurE and STAR

The Center Surround Extrema or CenSurE [177] method provides a true multi-scale descriptor, creating a feature vector using full spatial resolution at all scales in the pyramid, in contrast to SIFT and SURF, which find extrema at subsampled pixels that compromises accuracy at larger scales. CenSurE is similar to SIFT and SURF, but some key differences are summarized in Table 6.5. Modifications have been made to the original CenSurE algorithm in OpenCV, which goes by the name of STAR descriptor.

The authors have paid careful attention to creating methods which are computationally efficient, memory efficient, with high performance and accuracy [177]. CenSurE defines an optimized approach to find extrema by first using the Laplacian at all scales, followed by a filtering step using the Harris method to discard corners with weak responses.

The major innovations of CenSurE over SIFT and SURF are as follows:

1. Use of bilevel center-surround filters, as shown in Fig. 6.19, including Difference of Boxes (DoB), Difference of Octagons (DoO) and Difference of Hexagons (DoH) filters, octagons and hexagons are more rotationally invariant than boxes. DoB is computationally simple and may be computed with integral images vs. the Gaussian scale space method of SIFT. The DoO and DoH filters are also computed quickly using a modified integral image method. Circle is the desired shape, but more computationally expensive.
2. To find the extrema, the DoB filter is computed using a seven-level scale space of filters at each pixel, using a $3 \times 3 \times 3$ neighborhood. The scale space search is composed using center-surround Haar-like features on non-octave boundaries with filter block sizes [1,2,3,4,5,6,7] covering 2.5 octaves between [1 and 7] yielding five filters. This scale arrangement provides more

discrimination than an octave scale. A threshold is applied to eliminate weak filter responses at each level, since the weak responses are likely not to be repeated at other scales.
3. Nonrectangular filter shapes, such as octagons and hexagons, are computed quickly using combinations of overlapping integral image regions; note that octagons and hexagons avoid artifacts caused by rectangular regions and increase rotational invariance; see Fig. 6.19.
4. CenSurE filters are applied using a fast, modified version of the SURF method called Modified Upright SURF (MU-SURF) [180, 181], discussed later with other SURF variants, which pays special attention to boundary effects of boxes in the descriptor by using an expanded set of overlapping subregions for the HAAR responses.

Figure 6.19 CenSurE bilevel center surround filter shape approximations to the Laplacian using binary kernel values of 1 and −1, which can be efficiently implemented using signed addition rather than multiplication. Note that the circular shape is the desired shape, but the other shapes are easier to compute using integral images, especially the rectangular method

CENSURE SUMMARY TAXONOMY

Spectra:	*Center-surround shaped bi-level filters*
Feature shape:	*Octagons, circles, boxes, hexagons*
Feature pattern:	*Filter shape masks, 24 × 24 largest region*
Feature density:	*Sparse at Local interest points*
Search method:	*Dense sliding window over scale space*
Distance function:	*Euclidean distance*
Robustness:	*5 (brightness, contrast, rotation, scale, affine transforms)*

Correlation Templates

One of the most well known and obvious methods for feature description and detection, as used as the primary feature in basic deep learning architectures discussed in Chaps. 9 and 10, takes an image of the complete feature and searches for it by direct pixel comparison—this is known as *correlation*. Correlation involves stepping a sliding window containing a first pixel region template across a second image region template and performing a simple pixel-by-pixel region comparison using a method such as sum of differences (SAD); the resulting score is the correlation.

Since image illumination may vary, typically the correlation template and the target image are first intensity normalized, typically by subtracting the mean and dividing by the standard deviation; however, contrast leveling and LUT transform may also be used. Correlation is commonly implemented in the spatial domain on rectangular windows, but can be used with frequency domain methods as well [4, 9].

Correlation is used in video-based target tracking applications where translation as orthogonal motion from frame-to-frame over small adjacent regions predominates. For example, video motion encoders find the displacement of regions or blocks within the image using correlation, since usually

small block motion in video is orthogonal to the Cartesian axis and maps well to simple displacements found using correlation. Correlation can provide sub-pixel accuracy between 1/4 and 1/20 of a pixel, depending on the images and methods used; see reference [143]. For video encoding applications, correlation allows for the motion vector displacements of corresponding blocks to be efficiently encoded and accurately computed. Correlation is amenable to fixed function hardware acceleration.

Variations on correlation include cross-correlation (sliding dot product), normalized cross-correlation (NCC), zero-mean normalized cross-correlation (ZNCC), and texture auto correlation (TAC).

In general, correlation is a good detector for orthogonal motion of a constant-sized mono-space pattern region. It provides sub-pixel accuracy, has limited robustness and accuracy over illumination, but little to no robustness over rotation or scale. However, to overcome these robustness problems, it is possible to accelerate correlation over a scale space, as well as various geometric translations, using multiple texture samplers in a graphics processor in parallel to rapidly scale and rotate the correlation templates. Then, the correlation matching can be done either via SIMD SAD instructions or else using the fast fixed function correlators in the video encoding engines.

Correlation is illustrated in Fig. 6.20.

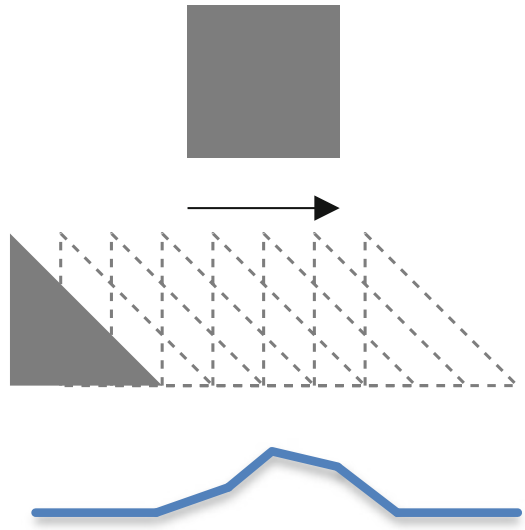


Figure 6.20 Simplified model of digital correlation using a triangular template region swept past a rectangular region. The best correlation is shown at the location of the highest point

CORRELATION SUMMARY TAXONOMY

Spectra:	*Correlation*
Feature shape:	*Square, rectangle*
Feature pattern:	*Dense*
Feature density:	*Variable sized kernels*
Search method:	*Dense sliding window*
Distance function:	*SSD typical, others possible*
Robustness:	*1 (illumination, sub-pixel accuracy)*

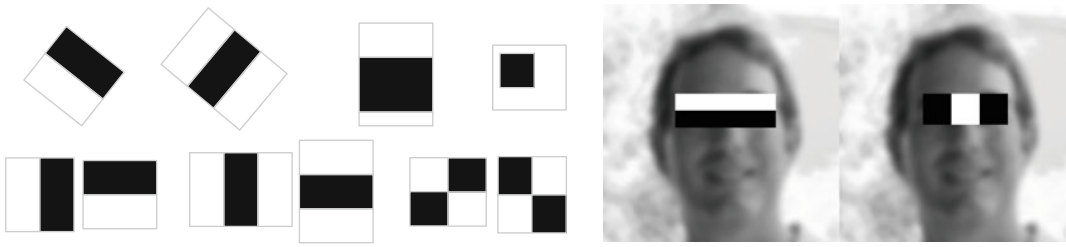


Figure 6.21 Example HAAR-like features

HAAR Features

HAAR-like features [4, 9] were popularized in the field of computer vision by the Viola–Jones [178] algorithm. HAAR features are based on specific sets of rectangle patterns, as shown in Fig. 6.21, which approximate the basic HAAR wavelets, where each HAAR feature is composed of the average pixel value of pixels within the rectangle. This is efficiently computed using integral images.

By using the average pixel value in the rectangular feature, the intent is to find a set of small patterns in adjacent areas where brighter or darker region adjacency may reveal a feature—for example, a bright cheek next to a darker eye socket. However, HAAR features have drawbacks, since rectangles by nature are not rotation invariant much beyond 15°. Also, the integration of pixel values within the rectangle destroys fine detail.

Depending on the type of feature to be detected, such as eyes, a specific set of HAAR feature is chosen to reveal eye/cheek details and eye/nose details. For example, HAAR patterns with two rectangles are useful for detecting edges, while patterns with three rectangles can be used for lines, and patterns with an inset rectangle or four rectangles can be used for single-object features. Note that HAAR features may be a rotated set.

Of course, the scale of the HAAR patterns is an issue, and since a given HAAR feature only works with an image of appropriate scale. Image pyramids are used for HAAR feature detection, along with other techniques for stepping the search window across the image in optimal grid sizes for a given application. Another method to address feature scale is to use a wider set of scaled HAAR features to perform the pyramiding in the feature space rather than the image space. One method to address HAAR feature granularity and rectangular shape is to use overlapping HAAR features to approximate octagons and hexagons; see the CenSurE and STAR methods in Fig. 6.19.

HAAR features are closely related to wavelets [219, 326]. Wavelets can be considered as an extension of the earlier concept of Gabor functions [179, 325]. We provide only a short discussion of wavelets and Gabor functions here; more discussion was provided in Chap. 2. Wavelets are an *orthonormal* set of small duration functions. Each set of wavelets is designed to meet various goals to locate short-term signal phenomenon. There is no single wavelet function; rather, when designing wavelets, a mother wavelet is first designed as the basis of the wavelet family, and then daughter wavelets are derived using translation and compression of the mother wavelet into a basis set. Wavelets are used as a set of nonlinear basis functions, where each basis function can be designed as needed to optimally match a desired feature in the input function. So, unlike transforms which use a uniform set of basis functions like the Fourier transform, composed of SIN and COS functions, wavelets use a dynamic set of basis functions that are complex and nonuniform in nature. Wavelets can be used to describe very complex short-term features, and this may be an advantage in some feature detection applications.

However, compared to integral images and HAAR features, wavelets are computationally expensive, since they represent complex functions in a complex domain. HAAR 2D basis functions are commonly used owing to the simple rectangular shape and computational simplicity, especially when HAAR features are derived from integral images.

HAAR SUMMARY TAXONOMY

Spectra:	*Integral box filter*
Feature shape:	*Square, rectangle*
Feature pattern:	*Dense*
Feature density:	*Variable-sized kernels*
Search method:	*Grid search typical*
Distance function:	*Simple difference*
Robustness:	*1 (illumination)*

Viola–Jones with HAAR-Like Features

The Viola–Jones method [178] is a feature detection pipeline framework based on HAAR-like features using a perceptron learning algorithm to train a detector matching network that consists of three major parts:

1. Integral images used to rapidly compute HAAR-like features.
2. The ADA-BOOST learning algorithm to create a strong pattern matching and classifier network by combining strong classifiers with good matching performance with weak classifiers that have been "boosted" by adjusting weighting factors during the training process.
3. Combining classifiers into a detector cascade or funnel to quickly discard unwanted features at early stages in the cascade.

Since thousands of HAAR pattern matches may be found in a single image, the feature calculations must be done quickly. To make the HAAR pattern match calculation rapidly, the entire image is first processed into an integral image. Each region of the image is searched for known HAAR features using a sliding window method stepped at some chosen interval, such as every n pixels, and the detected features are fed into a classification funnel known as a *HAAR* Cascade Classifier. The top of the funnel consists of feature sets which yield low false positives and false negatives, so the first-order results of the cascade contain high-probability regions of the image for further analysis. The HAAR features become more complex progressing deeper into the funnel of the cascade. With this arrangement, images regions are rejected as soon as possible if the desired HAAR features are not found, minimizing processing overhead.

A complete HAAR feature detector may combine hundreds or thousands of HAAR features together into a final classifier, where not only the feature itself may be important but also the spatial arrangements of features—for example, the distance and angular relationships between features could be used in the classifier.

SURF

The Speeded-up Robust Features Method (SURF) [152] operates in a scale space and uses a fast Hessian detector based on the determinant maxima points of the Hessian matrix. SURF uses a scale space over a $3 \times 3 \times 3$ neighborhood to localize bloblike interest point features. To find feature orientation, a set of HAAR-like feature responses are computed in the local region surrounding each interest point within a circular radius, computed at the matching pyramid scale for the interest point.

The dominant orientation assignment for the local set of HAAR features is found, as shown in Fig. 6.22, using a sliding sector window of size $\frac{\pi}{3}$. This sliding sector window is rotated around the interest point at intervals. Within the sliding sector region, all HAAR features are summed. This includes both the horizontal and vertical responses, which yield a set of orientation vectors; the largest vector is chosen to represent dominant feature orientation. By way of comparison, SURF integrates gradients to find the dominant direction, while SIFT uses a histogram of gradient directions to record orientation.

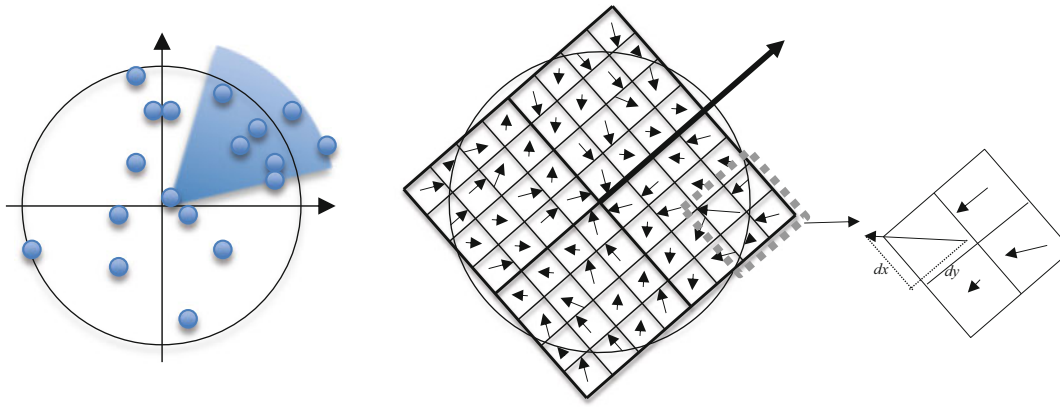


Figure 6.22 (*Left*) The sliding sector window used in SURF to compute the dominant orientation of the HAAR features to add rotational invariance to the SURF features. (*Right*) The feature vector construction process, showing a grid containing a 4×4 region subdivided into 4×4 subregions and 2×2 subdivisions

To create the SURF descriptor vector, a rectangular grid of 4×4 regions is established surrounding the interest point, similar to SIFT, and each region of this grid is split into 4×4 subregions. Within each subregion, the HAAR wavelet response is computed over 5×5 sample points. Each HAAR response is weighted using a circularly symmetric Gaussian weighting factor, where the weighting factor decreases with distance from the center interest point, which is similar to SIFT. Each feature vector contains four parts:

$$v = \left(\sum d_x , \sum d_y , \sum |d_x|, \sum |d_y| \right)$$

The wavelet responses d_x and d_y for each subregion are summed, and the absolute value of the responses $|d_x|$ and $|d_y|$ provide polarity of the change in intensity. The final descriptor vector is $4 \times 4 \times 4$: 4×4 regions with four parts per region, for a total vector length of 64. Of course, other vector lengths can be devised by modifying the basic method.

As shown in Fig. 6.22, the SURF gradient grid is rotated according to the dominant orientation, computed during the sliding sector window process, and then the wavelet response is computed in

each square region relative to orientation for binning into the feature vector. Each of the wavelet directional sums d_x, d_y, $|d_x|$, $|d_y|$ is recorded in the feature vector.

The SURF and SIFT pipeline methods are generally comparable in implementation steps and final accuracy, but SURF is one order of magnitude faster to compute than SIFT, as compared in an ORB benchmarking test [126]. However, the local binary descriptors, such as ORB, are another order of magnitude faster than SURF, with comparable accuracy for many applications [126]. For more information, see the section earlier in this chapter on local binary descriptors.

SURF SUMMARY TAXONOMY

Spectra:	*Integral box filter + orientation vector*
Feature shape:	*HAAR rectangles*
Feature pattern:	*Dense*
Feature density:	*Sparse at Hessian interest points*
Search method:	*Dense sliding window over scale space*
Distance function:	*Mahalanobis or Euclidean*
Robustness:	*4 (scale, rotation, illumination, noise)*

Variations on SURF

A few variations on the SURF descriptor [180, 181] are worth discussing, as shown in Table 6.6. Of particular interest are the G-SURF methods [180], which use a differential geometry concept [182] of a local region gauge coordinate system to compute the features. Since gauge coordinates are not global but, rather, local to the image feature, gauge space features carry advantages for geometrical accuracy.

Table 6.6 SURF variants (as discussed in Alcantarilla et al. [180])

SURF	Circular Symmetric Gaussian Weighting Scheme, 20x20 grid
U-SURF [181]	Faster version of SURF, only upright features are used; no orientation. Like M-SURF except calculated upright "U" with no rotation of the grid, uses a 20x20 grid, no overlapping HAAR features, modified Gaussian weighting scheme, bilinear interpolation between histogram bins.
M-SURF MU-SURF [181]	Circular symmetric Gaussian weighting scheme computed in two steps instead of one as for normal SURF, 24x24 grid using overlapping HAAR features, rotation orientation left out in MU-SURF version.
G-SURF, GU-SURF [180]	Instead of HAAR features, substitutes 2^{nd} order gauge derivatives in Gauge coordinate space, no Gaussian weighting, 20x20 grid. Gauge derivatives are rotation and translation invariant, while the HAAR features are simple rectangles, and rectangles have poor rotational invariance, maybe +/-15 degrees at best.
MG-SURF [180]	Same as M-SURF, but uses gauge derivatives.
NG-SURF [180]	N = No Gaussian weighting as in SURF; same as SURF but no Gaussian weighting applied, allows for comparison between gauge derivate features and HAAR features.

Histogram of Gradients (HOG) and Variants

The Histogram of Gradients (HOG) method [98] is intended for image classification, and relies on computing local region gradients over a dense grid of overlapping blocks, rather than at interest points. HOG is appropriate for some applications, such as person detection, where the feature in the image is quite large.

HOG operates on raw data; while many methods rely on Gaussian smoothing and other filtering methods to prepare the data, HOG is designed specifically to use all the raw data without introducing filtering artifacts that remove fine details. The authors show clear benefits using this approach. It is a trade-off: *filtering artifacts* such as smoothing vs. *image artifacts* such as fine details. The HOG method shows preferential results for the raw data. See Fig. 4.12, showing a visualization of a HOG descriptor.

Major aspects in the HOG method are as follows:

- Raw RGB image is used with no color correction or noise filtering, using other color spaces and color gamma adjustment provided little advantage for the added cost.
- Prefers a 64 × 128 sliding detector window; 56 × 120 and 48 × 112 sized windows were also tested. Within this detector window, a total of 8 × 16 8 × 8 pixel block regions are defined for computation of gradients. Block sizes are tunable.
- For each 8 × 8 pixel block, a total of 64 local gradient magnitudes are computed. The preferred method is simple line and column derivatives $[-1,0,1]$ in x/y; other gradient filter methods are tried, but larger filters with or without Gaussian filtering degrade accuracy and performance. Separate gradients are calculated for each color channel.
- Local gradient magnitudes are binned into a 9-bin histogram of edge orientations, quantizing dimensionality from 64 to 9, using bilinear interpolation; <9 bins produce poorer accuracy, >9 bins does not seem to matter. Note that either rectangular R-HOG or circular log polar C-HOG binning regions can be used.
- Normalization of gradient magnitude histogram values to unit length to provide illumination invariance. Normalization is performed in groups, rather than on single histograms. Overlapping 2 × 2 blocks of histograms are used within the detector window; the block overlapping method reduces sharp artifacts, and the 2 × 2 region size seems to work best.
- For the 64 × 128 pixel detector window method, a total of 128 8 × 8 pixel blocks are defined. Each 8 × 8 block has four cells for computing separate 9-bin histograms. The total descriptor size is then $8 \times 16 \times 4 \times 9 = 4608$.

Note that various formulations of the sliding window and block sizes are used for dealing with specific application domains. See Fig. 4.12, showing a visualization of HOG descriptor computed using 7 × 15 8 × 8 pixel cells. Key findings from the HOG [98] design approach include:

- The abrupt edges at fine scales in the raw data are required for accuracy in the gradient calculations, and post-processing and normalizing the gradient bins later works well.
- L2 style block normalization of local contrast is preferred and provides better accuracy over global normalization; note that the local region blocks are overlapped to assist in the normalization.
- Dropping the L2 block normalization stage during histogram binning reduces accuracy by 27 %.
- HOG features perform much better than HAAR-style detectors, and this makes sense when we consider that a HAAR wavelet is an integrated directionless value, while gradient magnitude and direction over the local HOG region provides a richer spectra.

HOG SUMMARY TAXONOMY

Spectra:	*Local region gradient histograms*
Feature shape:	*Rectangle or circle*
Feature pattern:	*Dense 64 × 128 typical rectangle*
Feature density:	*Dense overlapping blocks*
Search method:	*Grid over scale space*
Distance function:	*Euclidean*
Robustness:	*4 (illumination, viewpoint, scale, noise)*

PHOG and Related Methods

The Pyramid Histogram of Oriented Gradients (PHOG) [183] method is designed for global or regional image classification, rather than local feature detection. PHOG combines regional HOG features with whole image area features using spatial relationships between features spread across the entire image in an octave grid region subdivision; see Fig. 6.23.

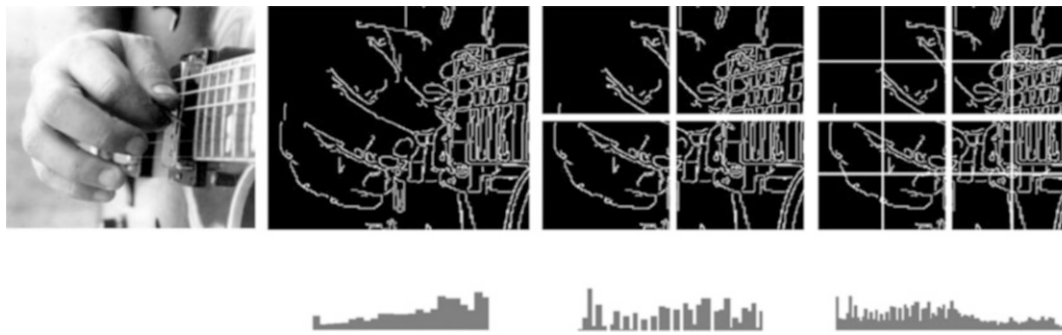


Figure 6.23 Set of PHOG descriptors computed over the whole image, using octave grid cells to bound the edge information. (*Center Left*) A single histogram. (*Center right*) Four histograms shown concatenated together. (*Right*) Sixteen histograms shown concatenated

PHOG is similar to related work using a coarse-to-fine grid of region histograms called Spatial Pyramid Matching by Lazebni, Schmid, and Ponce [516], using histograms of oriented edges and SIFT features to provide multi-class classification. It is also similar to earlier work on pyramids of concatenated histogram features taken over a progressively finer grid, called Pyramid Match Kernel and developed by Grauman and Darrell [517], which computes correspondence using weighted, multi-resolution histogram intersection. Other related earlier work using multi-resolution histograms for texture classification are described in reference [47].

The PHOG descriptor captures several feature variables, including:

- **Shape features**, derived from local distribution of edges based on gradient features inspired by the HOG method [98].
- **Spatial relationships**, across the entire image by computing histogram features over a set of octave grid cells with blocks of increasingly finer size over the image.

- **Appearance features**, using a dense set of SIFT descriptors calculated across a regularly spaced dense grid. PHOG is demonstrated to compute SIFT vectors for color images; results are provided in [183] for the HSV color space.

A set of training images is used to generate a set of PHOG descriptor variables for a class of images, such as cars or people. This training set of PHOG features is reduced using K-means clustering to a set of several hundred visual words to use for feature matching and image classification.

Some key concepts of the PHOG are illustrated in Fig. 6.23. For the feature shape, the edges are computed using the Canny edge detector, and the gradient orientation is computed using the Sobel operator. The gradient orientation binning is linearly interpolated across adjacent histogram bins by gradient orientation (HOG), each bin represents the angle of the edge. A HOG vector is computed for each size of grid cell across the entire image. The final PHOG descriptor is composed of a weighted concatenation of all the individual HOG histograms from each grid level. There is no scale-space smoothing between the octave grid cell regions to reduce fine detail.

As shown in Fig. 6.23, the final PHOG contains all the HOGs concatenated. Note that for the center left image, the full grid size cell produces 1 HOG, for the center right, the half octave grid produces 4 HOGs, and for the right image, the fine grid produces 16 HOG vectors. The final PHOG is normalized to unity to reduce biasing due to concentration of edges or texture.

PHOG SUMMARY TAXONOMY

Spectra:	*Global and regional gradient orientation histograms*
Feature shape:	*Rectangle*
Feature pattern:	*Dense grid of tiles*
Feature density:	*Dense tiles*
Search method:	*Grid regions, no searching*
Distance function:	*l2 norm*
Robustness:	*3 (image classification under some invariance to illumination, viewpoint, noise)*

Daisy and O-Daisy

The Daisy Descriptor [206, 308] is inspired by SIFT and GLOH-like descriptors, and is devised for dense-matching applications such as stereo mapping and tracking, reported to be about 40 % faster than SIFT. See Fig. 6.24. Daisy relies on a set of radially distributed and increasing size Gaussian convolution kernels that overlap and resemble a flower-like shape (Daisy).

Daisy does not need local interest points, and instead computes a descriptor densely at each pixel, since the intended application is stereo mapping and tracking. Rather than using gradient magnitude and direction calculations like SIFT and GLOH, Daisy computes a set of convolved orientation maps based on a set of oriented derivatives of Gaussian filters to create eight orientation maps spaced at equal angles.

As shown in Fig. 6.24, the size of each filter region and the amount of blur in each Gaussian filter increase with distance away from the center, mimicking the human visual system by maintaining a sharpness and focus in the center of the field of view and decreasing focus and resolution farther away from the center. Like SIFT, Daisy also uses histogram binning of the local orientation to form the descriptor.

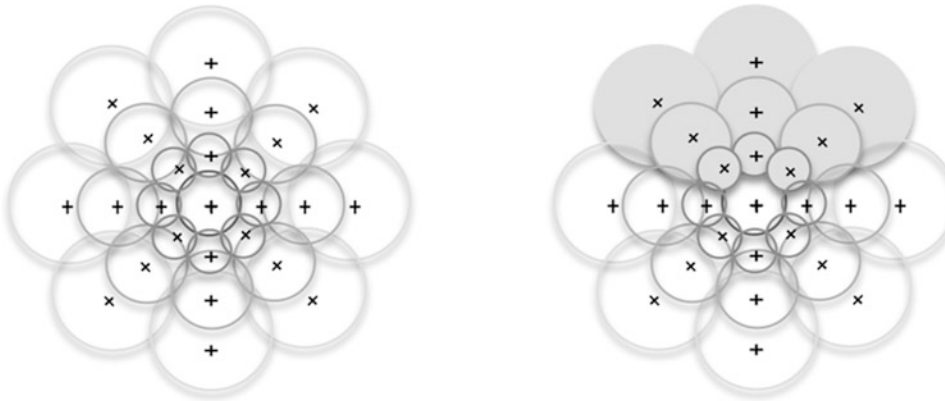


Figure 6.24 (*Left*) Daisy pattern region, which is composed of four sets of eight overlapping concentric circles, with increasing Gaussian blur in the outer circles, where the radius of each circle is proportional to the Gaussian kernel region standard deviation. The overlapping circular regions provide a degree of filtering against adjacent region transition artifacts. (*Right*) A hypothetical binary occlusion mask; *darker regions* indicate points that may be occluded and "turned off" in the descriptor during matching

Daisy is designed with optimizations in mind. The convolution orientation map approach consumes fewer compute cycles than the gradient magnitude and direction approach of SIFT and GLOH, yet yields similar results. The Daisy method also includes optimizations for computing larger Gaussian kernels by using a sequential set of smaller kernels, and also by computing certain convolution kernels recursively. Another optimization is gained using a circular grid pattern instead of the rectangular grid used in SIFT, which allows Daisy to vary the rotation by rotating the sampling grid rather than recomputing the convolution maps.

As shown in Fig. 6.24 (right image), Daisy also uses binary occlusion masks to identify portions of the descriptor pattern to use or ignore in the feature matching distance functions. This is a novel feature and provides for invariance to occlusion.

An FPGA optimized version of Daisy, called O-Daisy [209], provides enhancements for increased rotational invariance.

DAISY SUMMARY TAXONOMY

Spectra:	*Gaussian convolution values*
Feature shape:	*Circular*
Feature pattern:	*Overlapping concentric circular*
Feature density:	*Dense at each pixel*
Search method:	*Dense sliding window*
Distance function:	Euclidean
Robustness:	*3 (illumination, occlusion, noise)*

CARD

The Compact and Realtime Descriptor (CARD) method [210] is designed with performance optimizations in mind, using learning-based sparse hashing to convert descriptors into binary codes

supporting fast Hamming distance matching. A novel concept from CARD is the lookup-table descriptor extraction of histograms of oriented gradients from local pixel patches, as well as the lookup-table binning into Cartesian or log polar bins. CARD is reported to achieve significantly better rotation and scale robustness compared to SIFT and SURF, with performance at least ten times better than SIFT and slightly better than SURF.

CARD follows the method of RIFF [211, 214] for feature detection, using FAST features located over octave levels in the image pyramid. The complete CARD pyramid includes intermediate levels between octaves for increased resolution. The pyramid levels are computed at intervals of $1/\sqrt{2}$, with level 0 being the full image. Keypoints are found using a Shi–Tomasi [149] optimized Harris corner detector.

Like SIFT, CARD computes the gradient at each pixel, and can use either Cartesian coordinate binning, or log polar coordinate binning like GLOH; see Fig. 6.17. To avoid the costly bilinear interpolation of gradient information into the histogram bins, CARD instead optimizes this step by rotating the binning pattern before binning, as shown in Fig. 6.25. Note that the binning is further optimized using lookup tables, which contain function values based on principal orientations of the gradients in the patch.

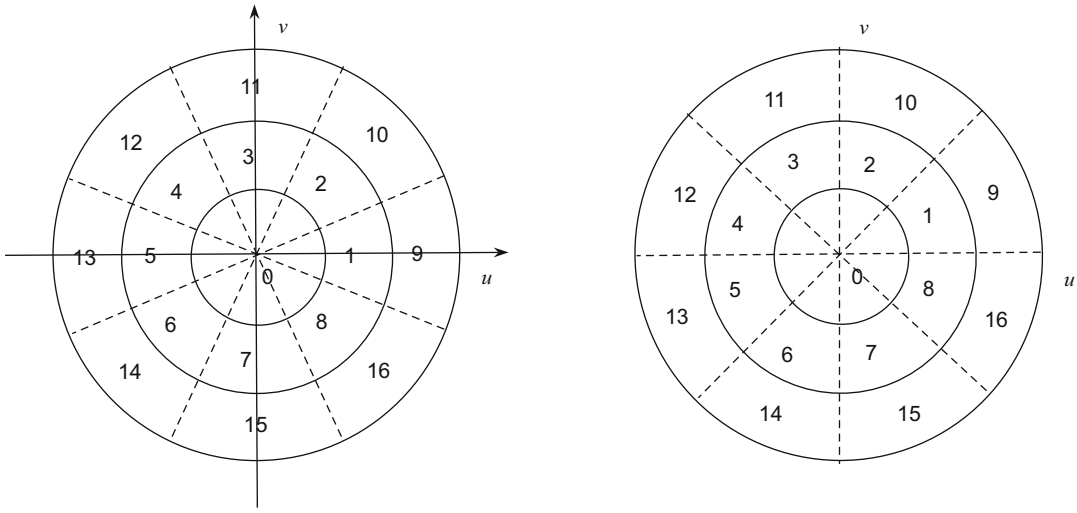


Figure 6.25 CARD patch pattern containing 17 log polar coordinate bins, with image on *left* rotated to optimize binning

As shown in Fig. 6.25, to speed up binning, instead of rotating the patch based on the estimated gradient direction to extract and bin a rotationally invariant descriptor, as done in SIFT and other methods, CARD rotates the binning pattern over the patch based on the gradient direction and then performs binning, which is much faster. Figure 6.25 shows the binning pattern unrotated on the right, and rotated by $\pi/8$ on the left. All binned values are concatenated and normalized to form the descriptor, which is 128 bits long in the most accurate form reported [210].

CARD SUMMARY TAXONOMY
Spectra: Gradient magnitude and direction
Feature shape: *Circular, variable sized based on pyramid scale and principal orientation*
Feature pattern: *Dense*
Feature density: *Sparse at FAST interest points over image pyramid*
Search method: *Sliding window*
Distance function: *Hamming*
Robustness: *3 (illumination, scale, rotation)*

Robust Fast Feature Matching

Robust Feature Matching in 2.3us developed by Taylor, Rosten and Drummond [212] (RFM2.3) (this *acronym is coined here by the author*) is a novel, fast method of feature description and matching, optimized for both compute speed and memory footprint. RFM2.3 stands alone among the feature descriptors surveyed here with regard to the combination of methods and optimizations employed, including sparse region histograms and binary feature codes. One of the key ideas developed in RFM2.3 is to compute a descriptor for multiple views of the same patch by creating a set of scaled, rotated, and affine warped views of the original feature, which provides invariance under affine transforms such as rotation and scaling, as well as perspective.

In addition to warping, some noise and blurring is added to the warped patch set to provide robustness to the descriptor. RFM2.3 is one of few methods in the class of deformable descriptors [336–338]. FAST keypoints in a scale space pyramid are used to locate candidate features, and the warped patch set is computed for each keypoint. After the warped patch set has been computed, FAST corners are again generated over each new patch in the set to determine which patches are most distinct and detectable, and the best patches are selected and quantized into binary feature descriptors and saved in the pattern database.

As shown in Fig. 6.26, RFM2.3 uses a sparse 8×8 sampling pattern within a 16×16 region to capture the patch. A sparse set of 13 pixels in the 8×8 sampling pattern is chosen to form the index into the pattern database for the sparse pattern. The index is formed as a 13-bit integer, where each bit

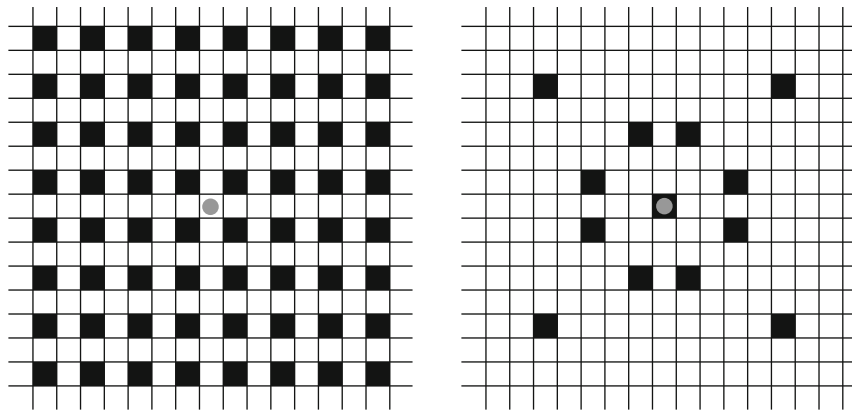


Figure 6.26 RFM2.3 (*Left*) Descriptor sparse sampling pattern. (*Right*) Sparse descriptor using 13 samples used to build the feature index into the database

is set to 1 if the pixel value is greater than the patch mean value, limiting the index to 2^{13} or 8192 entries, so several features in the database may share the same index. However, feature differences can be computed very quickly using Hamming distance, so the index serves mostly as a database key for organizing like-patches. A training phase determines the optimal set of index values to include in the feature database, and the optimal patterns to save, since some patterns are more distinct than others. Initially, features are captured at full resolution, but if few good features are found at full resolution, additional features are extracted at the next level of the image pyramid.

The descriptor is modeled during training as a 64-value normalized intensity distribution function, which is reduced in size to compute the final descriptor vector in two passes: first, the 64 values are reduced to a five-bin histogram of pixel intensity distribution; second, when training is complete, each histogram bin is binary encoded with a 1 bit if the bin is used, and a 0 bit if the bin is rarely used. The resulting descriptor is a compressed, binary encoded bit vector suitable for Hamming distance.

RFM2.3 SUMMARY TAXONOMY

Spectra:	*Normalized histogram patch intensity encoded into binary patch index code*
Feature shape:	*Rectangular, multiple viewpoints*
Feature pattern:	*Sparse patterns in 15 × 15 pixel patch*
Feature density:	*Sparse at FAST9 interest points*
Search method:	*Sliding window over image pyramid*
Distance function:	*Hamming*
Robustness:	*4 (illumination, scale, rotation, viewpoint)*

RIFF, CHOG

The Rotation Invariant Fast Features (RIFF) [211, 214] method is motivated by tracking and mapping applications in mobile augmented reality. The basis of the RIFF method includes the development of a radial gradient transform (RGT), which expresses gradient orientation and magnitude in a compute-efficient and rotationally invariant fashion. Another contribution of RIFF is a tracking method, which is reported to be more accurate than KLT with 26× better performance. RIFF is reported to be 15× faster than SURF.

RIFF uses a HOG descriptor computed at FAST interest points located in scale space, and generally follows the method of the author's previous work in CHOG [215] (compressed HOG) for reduced dimensionality, low bitrate binning. Prior to binning the HOG gradients, a radial gradient transform (RGT) is used to create a rotationally invariant gradient format. As shown in Fig. 6.27 (left image), the RGT uses two orthogonal basis vectors (r,t) to form the radial coordinate system that surrounds the patch center point c, and the HOG gradient g is projected onto (r,t) to express as the rotationally invariant vector $(g^{\mathrm{T}} r, g^{\mathrm{T}} t)$. A vector quantizer and a scalar quantizer are both suggested and used for binning, illustrated in Fig. 6.27.

As shown in Fig. 6.27 (right image) the basis vectors can be optimized by using gradient direction approximations in the approximated radial gradient transform (ARGT), which is optimized to be easily computed using a simple differences between adjacent, normalized pixels along the same gradient line, and simple 45° quantization. Also note in Fig. 6.27 (center left image), that the histogramming is optimized by sampling every other pixel within the annuli regions, and four annuli regions are used for practical reasons as a trade-off between discrimination and performance. To meet real-time system performance goals for quantizing the gradient histogram bins, RIFF uses a 5 × 5 scalar quantizer rather than a vector quantizer.

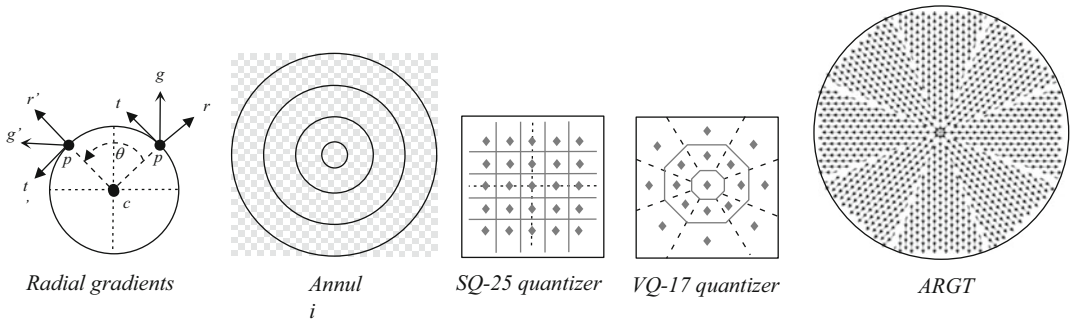


Figure 6.27 Concepts behind the RIFF descriptor [211, 214], based partially on CHOG [215]

In Fig. 6.27 (left image), the gradient projection of $\boldsymbol{g}$ at point $\boldsymbol{c}$ onto a radial coordinate system ($\boldsymbol{r,t}$) is used for a rotationally invariant gradient expression, and the descriptor patch is centered at $\boldsymbol{c}$. The center left image (Annuli) illustrates the method of binning, using four annuli rings, which reduces dimensionality, and sampling only the gray pixels provides a 2× speedup. The center and center right images illustrate the bin centering mechanism for histogram quantization: (1) the more flexible scalar quantizer SQ-25 and (2) the faster vector quantizer VQ-17. And the right image illustrates the radial coordinate system basis vectors for gradient orientation radiating from the center outwards, showing the more compute efficient ARGT, or approximated radial gradient transform (RGT), which does not use floating point math (RGT not shown, see [214]).

RIFF SUMMARY TAXONOMY

Spectra:	*Local region histogram of approximated radial gradients*
Feature shape:	*Circular*
Feature pattern:	*Sparse every other pixel*
Feature density:	*Sparse at FAST interest points over image pyramid*
Search method:	*Sliding window*
Distance function:	*Symmetric KL-divergence*
Robustness:	*4 (illumination, scale, rotation, viewpoint)*

Chain Code Histograms

A Chain Code Histogram (CCH) [198] descriptor records the shape of the perimeter as a histogram by binning the direction of the connected components—connected perimeter pixels in this case. As the perimeter is traversed pixel by pixel, the direction of the traversal is recorded as a number, as shown in Fig. 6.28, and recorded in a histogram feature. To match the CCH features, SSD or SAD distance metrics can be used.

Chain code histograms are covered by U.S. Patent US4783828. CCH was invented in 1961 [198] and is also known as the Freeman chain code. A variant of the CCH is the Vertex chain code [199], which allows for descriptor size reduction and is reported to have better accuracy.

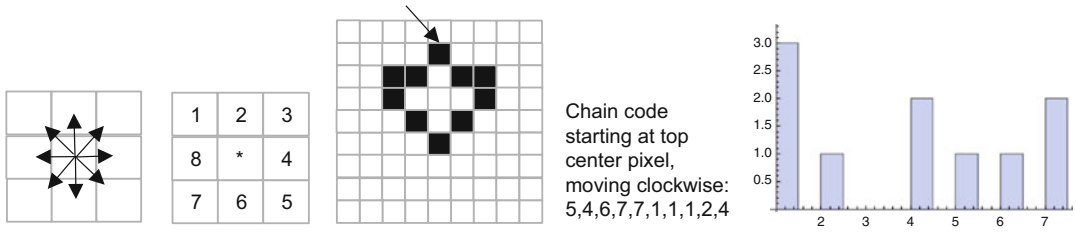


Figure 6.28 Chain code process for making a histogram. (*Left* to *right*) *1*. The eight possible directions that the connected perimeter may change. *2*. Chain code values for each connected perimeter direction change; direction for determining the chain code value is starting from the *center* pixel. *3*. An object with a connected perimeter highlighted by *black* pixels. *4*. Chain code for the object following the connected perimeter starting at the *top* pixel. *5*. Histogram of all the chain code values

D-NETS

The D-NETS (Descriptor-NETS) [127] approach developed by Hundelshausen and Sukthankar abandons patch or rectangular descriptor regions in favor of a set of strips connected at endpoints. D-NETS allows for a family of strip patterns composed of directed graphs between a set of endpoints; it does not specifically limit the types of endpoints or strip patterns that may be used. The D-NETS paper provides a discussion of results from three types of patterns:

- **Clique D-NETS:** A fully connected network of strips linking all the interest points. While the type of interest point used may vary within the method, the initial work reports results using SIFT keypoints.
- **Iterative D-NETS**: Dynamically creates the network using a subset of the interest points, increasing the connectivity using a stopping criterion to optimize the connection density for obtaining desired matching performance and accuracy.
- **Densely sampled D-NETS:** This variant does not use interest points, and instead densely samples the nets over a regularly spaced grid, a 10-pixel grid being empirically chosen and preferred, with some hysteresis or noise added to the grid positions to reduce pathological sampling artifacts. The dense method is suitable for highly parallel implementations for increased performance.

For an illustration of the three D-NETS patterns and some discussion, see Fig. 4.8.

Each strip is an array of raw pixel values sampled between two points. The descriptor itself is referred to as a *d-token*, and various methods for computing the d-token are suggested, such as binary comparisons among pixel values in the strip similar to FERNS or ORB, as well as comparing the 1D Fourier transforms of strip arrays, or using wavelets. The best results reported are a type of empirically engineered d-token, created as follows:

- **Strip vector sampling**, where each pixel strip vector is sampled at equally spaced locations between 10 and 80 % of the length of the pixel strip vector; this sampling arrangement was determined empirically to ignore pixels near the endpoints.
- **Quantize** the pixel strip vector by integrating the values into a set of uniform chunks, $\boldsymbol{s}$, to reduce noise.
- **Normalize** the strip vector for scaling and translation.
- **Discretize** the vector values into a limited bit range, $\boldsymbol{b}$.
- **Concatenate** all uniform chunks into the d-token, which is a bit string of length $\boldsymbol{s} \times \boldsymbol{b}$.

Descriptor matching makes use of an efficient and novel hashing and hypothesis correspondence voting method. D-NETS results are reported to be higher in precision and recall than ORB or SIFT.

D-NETS SUMMARY TAXONOMY

Spectra:	*Normalized, averaged linear pixel intensity chunks*
Feature shape:	*Line segment connected networks*
Feature pattern:	*Sparse line segments between chosen points*
Feature density:	*Sparse along lines*
Search method:	*Sliding window*
Distance function:	*Hashing and voting*
Robustness:	*5 (illumination, scale, rotation, viewpoint, occlusion)*

Local Gradient Pattern

A variation of the LBP approach, the local gradient pattern (LGP) [196] uses local region gradients instead of local image intensity pair comparison to form the binary descriptor. The 3×3 gradient of each pixel in the local region is computed, then each gradient magnitude is compared to the mean value of all the local region gradients, and the binary bit value of 1 is assigned if the value is greater, and 0 otherwise. The authors claim accuracy and discrimination improvements over the basic LBP in face-recognition algorithms, including a reduction in false positives. However, the compute requirements are greatly increased due to the local region gradient computations.

LGP SUMMARY TAXONOMY

Spectra:	*Local region gradient comparisons between center pixel and local region gradients*
Feature shape:	*Square*
Feature pattern:	*Every pixel 3×3 kernel region*
Feature density:	*Dense in 3×3 region*
Search method:	*Sliding window*
Distance function:	*Hamming*
Robustness:	*3 (illumination, scale, rotation)*

Local Phase Quantization

The local phase quantization (LPQ) descriptor [158–160] was designed to be robust to image blur, and it leverages the blur insensitive property of Fourier phase information. Since the Fourier transform is required to compute phase, there is some compute overhead; however, integer DFT methods can be used for acceleration. LPQ is reported to provide robustness for uniform blur, as well as uniform illumination changes. LPQ is reported to provide equal or slightly better accuracy on nonblurred images than LBP and Gabor filter bank methods. While mainly used for texture description, LPQ can also be used for local feature description to add blur invariance by combining LPQ with another descriptor method such as SIFT.

To compute, first a DFT is computed at each pixel over small regions of the image, such as 8×8 blocks. The low four frequency components from the phase spectrum are used in the descriptor. The authors note that the kernel size affects the blur invariance, so a larger kernel block may provide more invariance at the price of increased compute overhead.

Before quantization, the coefficients are de-correlated using a whitening transform, resulting in a uniform phase shift and 8-degree rotation, which preserves blur invariance. De-correlating the coefficients helps to create samples that are statistically independent for better quantization.

For each pixel, the resulting vectors are quantized into an 8-dimensional space, using an 8-bit binary encoded bit vector like the LBP and a simple scalar quantizer to yield 1 and 0 values. Binning into the feature vector is performed using 256 hypercubes derived from the 8-dimensional space. The resulting feature vector is a 256-dimensional 8-bit code.

LPQ SUMMARY TAXONOMY

Spectra:	*Local region whitened phase using DFT → an 8-bit binary code*
Feature shape:	*Square*
Feature pattern:	*8×8 kernel region*
Feature density:	*Dense every pixel*
Search method:	*Sliding window*
Distance function:	*Hamming*
Robustness:	*3 (contrast, brightness, blur)*

Basis Space Descriptors

This section covers the use of basis spaces to describe image features for computer vision applications. A *basis space* is composed of a set of functions, the *basis functions*, which are composed together as a set, such as a series like the Fourier series (discussed in Chap. 3). A complex signal can be decomposed into a chosen basis space as a descriptor.

Basis functions can be designed and used to describe, reconstruct, or synthesize a signal. They require a forward transform to project values into the basis set, and an inverse transform to move data back to the original values. A simple example is transforming numbers between the base 2 number system and the base 10 number system; each basis had advantages.

Sometimes it is useful to transform a dataset from one basis space to another to gain insight into the data, or to process and filter the data. For example, images captured in the time domain as sets of pixels in a Cartesian coordinate system can be transformed into other basis spaces, such as the Fourier basis space in the frequency domain, for processing and statistical analysis. A good basis space for computer vision applications will provide forward and inverse transforms. Again, the Fourier transform meets these criteria, as well as several other basis spaces.

Basis spaces are similar to coordinate systems, since both have invertible transforms to related spaces. In some cases, simply transforming a feature spectra into another coordinate system makes analysis and representation simpler and more efficient. (Chapter 4 discusses coordinates systems used for feature representation.) Several of the descriptors surveyed in this chapter use non-Cartesian coordinate systems, including GLOH, which uses polar coordinate binning, and RIFF, which uses radial coordinate descriptors.

Fourier Descriptors

Fourier descriptors [219] represent feature data as sine and cosine terms, which can be observed in a Fourier Power Spectrum. The Fourier series, Fourier transform, and Fast Fourier transform are used for a wide range of signal analysis, including 1D, 2D, and 3D problems. No discussion of image processing or computer vision is complete without Fourier methods, so we will explore Fourier methods here with applications to feature description.

Instead of developing the mathematics and theory behind the Fourier series and Fourier transform, which has been done very well in the standard text by Bracewell [219], we discuss applications of the Fourier Power Spectrum to feature description and provide minimal treatment of the fundamentals here to frame the discussion; see also Chap. 3. The basic idea behind the Fourier series is to define a series of sine and cosine basis functions in terms of magnitude and phase, which can be summed to approximate any complex periodic signal. Conversely, the Fourier transform is used to decompose a complex periodic signal into the Fourier series set of sine and cosine basis terms. The Fourier series components of a signal, such as a line or 2D image area, are used as a Fourier descriptor of the region.

For this discussion, a *Fourier descriptor* is the selected components from a Fourier Power Spectrum—typically, we select the lower-frequency components, which carry most of the power. Here are a few examples using Fourier descriptors; note that either or both the Fourier magnitude and phase may be used.

- **Fourier Spectrum of LBP Histograms.** As shown in Fig. 3.10, an LBP histogram set can be represented as a Fourier Spectrum magnitude, which makes the histogram descriptor invariant to rotation.
- **Fourier Descriptor of Shape Perimeter.** As shown in Fig. 6.29, the shape of a polygon object can be described by Fourier methods using an array of perimeter to centroid line segments taken at intervals, such as 10°. The array is fed into an FFT to produce a shape descriptor, which is scale and rotation invariant.
- **Fourier Descriptor of Gradient Histograms.** Many descriptors use gradients to represent features, and use gradient magnitude or direction histograms to bin the results. Fourier Spectrum magnitudes may be used to create a descriptor from gradient information to add invariance.
- **Fourier Spectrum of Radial Line Samples.** As used in the RFAN descriptor [128], radial line samples of pixel values from local regions can be represented as a Fourier descriptor of Fourier magnitudes.

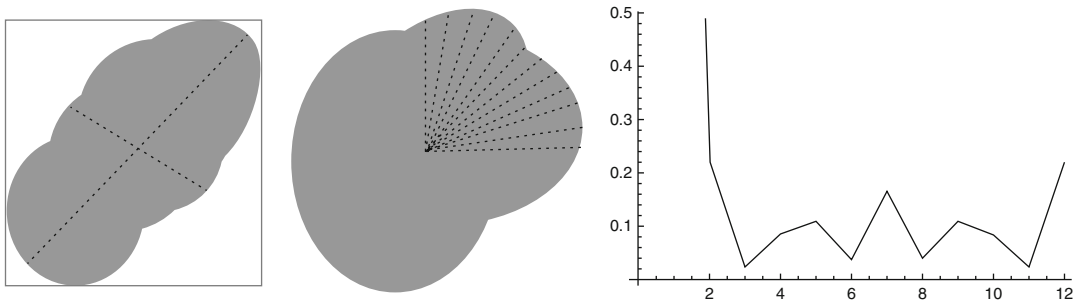


Figure 6.29 (*Left*) Polygon shape major and minor axis and bounding box. (*Center*) Object with radial sample length taken from the centroid to the perimeter, each sample length saved in an array, normalized. (*Right*) Image fed into the Fourier Spectrum to yield a Fourier descriptor

- **Fourier Spectrum Phase.** The LPQ descriptor, described in this chapter, makes use of the Fourier Spectrum phase information in the descriptor, and the LPQ is reported to be insensitive to blur owing to the phase information.

Other Basis Functions for Descriptor Building

Besides the Fourier basis series, other function series and basis sets are used for descriptor building, pattern recognition, and image coding. However, such methods are usually applied over a global or regional area. See Chap. 3 for details on several other methods.

Sparse Coding Methods

Any of the local feature descriptor methods discussed in this chapter may be used as the basis for a sparse codebook, which is a collection of descriptors boiled down to a representative set. Sparse coding and related methods are discussed in more detail in Chap. 10. Interesting examples are found in the work by Aharon, Alad, and Bruckstein [518] as well as Fei-Fei, Fergus, and Torralba [519]. See Fig. 6.30.

Figure 6.30 One method of feature learning using sparse coding, showing how Histograms of Sparse Codes (HSC) are constructed from a set of learned sparse codes. The HSC method [117] is reported to outperform HOG in many cases

Polygon Shape Descriptors

Polygon shape descriptors compute a set of *shape features* for an arbitrary polygon or blob, and the shape is described using statistical moments or image moments (as discussed in Chap. 3). These shape features are based on the perimeter of the polygon shape. The methods used to delineate image perimeters to highlight shapes prior to measurement and description are often complex, empirically tuned pipelines of image preprocessing operations, like thresholding, segmentation, and morphology (as discussed in Chap. 2). Once the polygon shapes are delineated, the shape descriptors are computed; see Fig. 6.31. Typically, polygon shape methods are applicable to larger region-size features. In the literature, this topic may also be discussed as *image moments*. For a deep dive into the topic of image moments, see Flusser et al. [500].

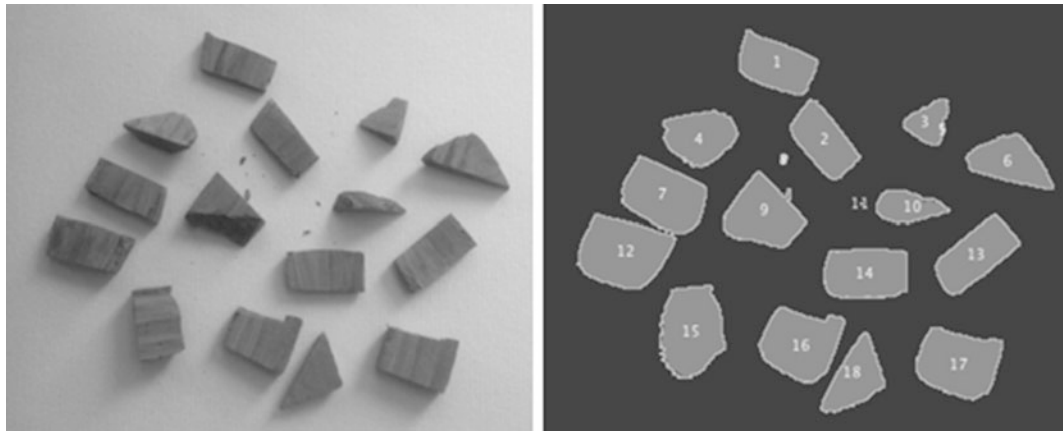


Figure 6.31 Polygon shape descriptors. (*Left*) Malachite pieces. (*Right*) Polygon shapes defined and labeled after binary thresholding, perimeter tracing, and feature labeling. (Image processing and particle analysis performed using ImageJ Fiji)

Polygon shape methods are commonly used in medical and industrial applications, such as automated microscopy for cell biology, and also for industrial inspection; see Fig. 6.31. Commercial software libraries are available for polygon shape description, commonly referred to as *particle analysis* or *blob analysis*. See Appendix C.

MSER Method

The Maximally Stable Extremal Regions (MSER) method [186] is usually discussed in the literature as an interest region detector, and in fact it is. However, we include MSER in the shape descriptor section because MSER regions can be much larger than other interest point methods, such as HARRIS or FAST.

The MSER detector was developed for solving disparity correspondence in a wide baseline stereo system. Stereo systems create a warped and complex geometric depth field, and depending on the baseline between cameras and the distance of the subject to the camera, various geometric effects must be compensated for. In a wide baseline stereo system, features nearer the camera are more distorted under affine transforms, making it harder to find exact matches between the left/right image pair. The MSER approach attempts to overcome this problem by matching on blob-like features. MSER regions are similar to morphological blobs and are fairly robust to skewing and lighting. MSER is essentially an efficient variant of the watershed algorithm, except that the goal of MSER is to find a range of thresholds that leave the watershed basin unchanged in size.

The MSER method involves sorting pixels into a set of regions based on binary intensity thresholding; regions with similar pixel value over a range of threshold values in a connected component pattern are considered maximally stable. To compute a MSER, pixels are sorted in a binary intensity thresholding loop, which sweeps the intensity value from min to max. First, the binary threshold is set to a low value such as zero on a single image channel—luminance, for example. Pixels $<$ the threshold value are black, pixels $>=$ are white. At each threshold level, a list of connected components or pixels is kept. The intensity threshold value is incremented from 0 to the max pixel value. Regions that do not grow or shrink or change as the intensity varies are

considered maximally stable, and the MSER descriptor records the position of the maximal regions and the corresponding thresholds.

In stereo applications, smaller MSER regions are preferred and correlation is used for the final correspondence, and similarity is measured inside a set of circular MSER regions at chosen rotation intervals. Some interesting advantages of the MSER include:

- Multi-scale features and multi-scale detection. Since the MSER features do not require any image smoothing or scale space, both coarse features and fine-edge features can be detected.
- Variable-size features computed globally across an entire region, not limited to patch size or search window size.
- Affine transform invariance, which is a specific goal.
- General invariance to shape change, and stability of detection, since the extremal regions tend to be detected across a wide range of image transformations.

The MSER can also be considered as the basis for a shape descriptor, and as an alternative to morphological methods of segmentation. Each MSER region can be analyzed and described using shape metrics, as discussed later in this chapter.

Object Shape Metrics for Blobs and Polygons

Object shape metrics are powerful and yield many degrees of freedom with respect to invariance and robustness. Object shape metrics are not like local feature metrics, since object shape metrics can describe much larger features. This is advantageous for tracking from frame to frame. For example, a large object described by just a few simple object shape metrics such as area, perimeter, and centroid can be tracked from frame to frame under a wide range of conditions and invariance. For more information, see references [120, 121] for a survey of 2D shape description methods.

Shape can be described by several methods, including:

- **Object shape moments and metrics:** the focus of this section.
- **Image moments:** see Chap. 3 under "Image Moments."
- **Fourier descriptors:** discussed in this chapter and Chap. 3.
- **Shape Context feature descriptor:** discussed in this section.
- **Chain code descriptor for perimeter description:** discussed in this section.

Object shape is closely related to the field of morphology, and computer methods for morphological processing are discussed in detail in Chap. 2. Also see the discussion about morphological interest points earlier in this chapter.

In many areas of computer vision research, local features seem to be favored over object shape-based features. The lack of popularity of shape analysis methods may be a reaction to the effort involved in creating preprocessing pipelines of filtering, morphology, and segmentation to prepare the image for shape analysis. If the image is not preprocessed and prepared correctly, shape analysis is not possible. (See Chap. 8 for a discussion of a hypothetical shape analysis preprocessing pipeline.)

Polygon shape metrics can be used for virtually any scene analysis application to find common objects and take accurate measurements of their size and shape; typical applications include biology and manufacturing. In general, most of the polygon shape metrics are rotational and scale invariant. Table 6.7 provides a sampling of some of the common metrics that can be derived from region shapes, both binary shapes and gray scale shapes.

Table 6.7 Various common object shape and blob object metrics

Object Binary Shape Metrics	Description
Perimeter	Length of all points around the edge of the object, including the sum of diagonal lengths ~= 1.4 and adjacent lengths = 1
Area	Total area of object in pixels
Convex hull	Polygon shape or set of line segments enclosing all perimeter points
Centroid	Center of object mass, average value of all pixel coordinates or average value of all perimeter coordinates
Fourier descriptor	Fourier spectrum result from an array containing the length of a set of radial line segments passing from centroid to perimeter at regular angles used to model a 1D signal function, the 1D signal function is fed into a 1D FFT and the set of FFT magnitude data is used as a metric for a chosen set of octave frequencies
Major/minor axis	Longest and shortest line segments passing through centroid contained withinand touching the perimeter
Feret	Largest caliper diameter of object
Breadth	Shortest caliper diameter
Aspect ratio	Feret/Breadth
Circularity	4 X Pi X Area/Perimeter^2
Roundness	4 X Area/(Pi X Feret^2) (Can also be calculated from the Fourier descriptors)
Area equivalent diameter	sqrt((4/Pi) X Area)
Perimeter equivalent diameter	Area/Pi
Equivalent ellipse	(Pi X Feret X Breadth)/4
Compactness	sqrt((4/Pi) X Area)/Feret
Solidity	Area/Convex_Area
Concavity	Convex_Area-Area
Convexity	Convex_Hull/Perimeter
Shape	Perimeter^2/Area
Modification ratio	(2 X MinR)/Feret
Shape matrix	A 2D matrix representation or plot of a polygon shape (may use Cartesian or polar coordinates; see Figure 6-32)
Grayscale Object Shape Metrics	
SDM plots	*See Chapter 3, “Texture Metrics” section.
Scatter plots	*See Chapter 3, “Texture Metrics” section.
Statistical moments of grayscale pixel values	Minimum Maximum Median Average Average deviation Standard deviation Variance Skewness Kurtosis Entropy

*Note: some of binary object metrics also apply to grayscale objects.

Shape is considered to be binary; however, shape can be computed around intensity channel objects as well, using gray scale morphology. Perimeter is considered as a set of connected components. The shape is defined by a single pixel wide perimeter at a binary threshold or within an intensity band, and pixels are either on, inside, or outside of the perimeter. The perimeter edge may be computed by scanning the image, pixel by pixel, and examining the adjacent touching pixel neighbors for connectivity. Or, the perimeter may be computed from the shape matrix [327] or chain code discussed earlier in this chapter. Perimeter length is computed for each segment (pixel), where segment length $= 1$ for horizontal and vertical neighbors, and $\sqrt{2}$ otherwise for diagonal neighbors.

The perimeter may be used as a mask, and gray scale or color channel statistical metrics may be computed within the region. The object area is the count of all the pixels inside the perimeter. The centroid may be computed either from the average of all (x,y) coordinates of all points contained within the perimeter area, or from the average of all perimeter (x,y) coordinates.

Shape metrics are powerful. For example, shape metrics may be used to remove or excluding objects from a scene prior to measurement. For example, objects can be removed from the scene when the area is smaller than a given size, or if the centroid coordinates are outside a given range.

As shown in Fig. 6.29 and Table 2, the Fourier descriptor provides a rotation and scale invariant shape metric, with some occlusion invariance also. The method for determining the Fourier descriptor is to take a set of equally angular-spaced radius measurements, such as every 10°, from the centroid out to points on the perimeter, and then to assemble the radius measurements into a 1D array that is run through a 1D FFT to yield the Fourier moments of the object. Or radial pixel spokes can be used as a descriptor.

Other examples of useful shape metrics, shown in Fig. 6.29, include the bounding box with major and minor axis, which has longest and shortest diameter segments passing through the centroid to the perimeter; this can be used to determine rotational orientation of an object.

The SNAKES method [522] uses a spline model to fit a collection of interest points, such as selected perimeter points, into a region contour. The interest points are the spline points. The SNAKE can be used to track contoured features from frame to frame, deforming around the interest point locations.

In general, the 2D object shape methods can be extended to 3D data; however, we do not explore 3D object shape metrics here, see reference [192, 193] for a survey of 3D shape descriptors.

Shape Context

The shape context method developed by Belongie, Malik, and Puzicha [231–233], describes local feature shape using a reference point on the perimeter as the Cartesian axis origin, and binning selected perimeter point coordinates relative to the reference point origin. The relative coordinates of each point are binned into a log polar histogram. Shape context is related to the earlier shape matrix descriptor [327] developed in 1985 as shown in Fig. 6.32, which describes the perimeter of an object using log polar coordinates also. The shape context method provides for variations, described in several papers by the authors [231–233]. Here, we look at a few key concepts.

To begin, the perimeter edge of the object is sparsely sampled at uniform intervals, typically keeping about 100 edge sample points for coarse binning. Sparse perimeter edge points are typically distinct from interest points, and found using perimeter tracing. Next, a reference point is chosen on the perimeter of the object as the origin of a Cartesian space, and the vector angle and magnitude (r, θ) from the origin point to each other perimeter point are computed. The magnitude or distance is normalized to fit the histogram. Each sparse perimeter edge point is used to compute a tangent with

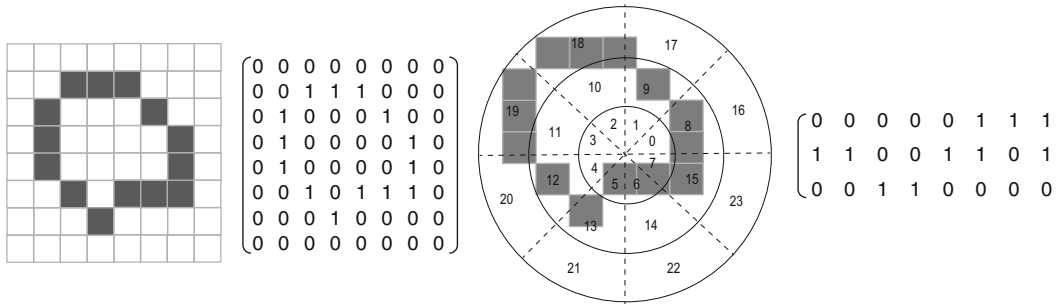


Figure 6.32 A shape matrix descriptor [327] for the perimeter of an object. (*Left* two images) Cartesian coordinate shape matrix. (*Right* two images) polar coordinate shape matrix using three rows of eight numbered bin regions, *gray boxes* represent pixels to be binned. Note that multiple shape matrices can be used together. Values in matrix are set if the pixel fills at least half of the bin region, no interpolation is used

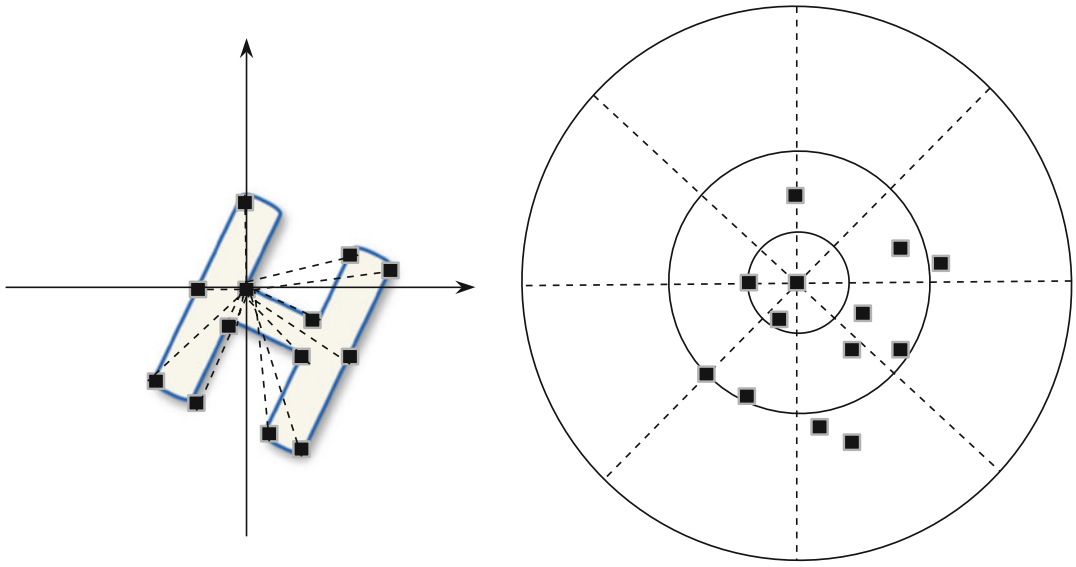


Figure 6.33 Shape context method. (*Left*) Perimeter points are measured as a shape vector, both angle and distance, with respect to a chosen perimeter point as the reference Cartesian origin. (*Right*) Shape vectors are binned into a log polar histogram feature descriptor

the origin. Finally, each normalized vector is binned using (r, θ) into a log polar histogram, which is called the *shape context*.

An alignment transform is generated between descriptor pairs during matching, which yields the difference between targets and chosen patterns, and could be used for reconstruction. The alignment transform can be chosen as desired from affine, Euclidean, spline-based, and other methods. Correspondence uses the Hungarian method, which includes histogram similarity, and is weighted by the alignment transform strength using the tangent angle dissimilarity. Matching may also employ a local appearance similarity measure, such as normalized correlation between patches or color histograms.

The shape context method provides a measure of invariance over scale, translation, rotation, occlusion, and noise. See Fig. 6.33.

3D, 4D, Volumetric and Multimodal Descriptors

With the advent of more and more 3D sensors, such as stereo cameras and other depth-sensing methods, as well as the ubiquitous accelerometers and other sensors built into inexpensive mobile devices, the realm of 3D feature description and multimodal feature description is beginning to blossom.

Many 3D descriptors are associated with robotics research and 3D localization. Since the field of 3D feature description is early in the development cycle, it is not yet clear which methods will be widely adopted, so we present only a small sampling of 3D descriptor methods here. These include 3D HOG [188], 3D SIFT [187], and HON 4D [190], which are based on familiar 2D methods. We refer the interested reader to references [192, 193, 208] for a survey of 3D shape descriptors. Several interesting 3D descriptor metrics are available as open source in the Point Cloud Library,[2] including Radius-Based Surface Descriptors (RSD) [521], Principal Curvature Descriptors (PCD), Signatures of Histogram Orientations (SHOT) [523], Viewpoint Feature Histogram (VFH) [381], and Spin Images [520].

Some noteworthy 3D descriptors we do not survey include 3D Shapenets by Wu [863], 3D voxel patterns [864], triangular surface patches [865], 3D surface patch features [865], see also [866–870]. Applications driving the research into 3D descriptors include robotics and activity recognition, where features are tracked frame to frame as they morph and deform. The goals are to localize position and recognize human actions, such as walking, waving a hand, turning around, or jumping. See also the LBP variants for 3D: V-LBP and LBP-TOP, which are surveyed earlier in this chapter as illustrated in Fig. 6.12, which are also used for activity recognition. Since the 2D features are moving during activity recognition, time is the third dimension incorporated into the descriptors. We survey some notable 3D activity-recognition research here.

One of the key concepts in the action-recognition work is to extend familiar 2D features into a 3D space that is spatiotemporal, where the 3D space is composed of 2D x,y video image sequences over time t into a volumetric representation with the form $v(x,y,t)$. In addition, the 3D surface normal, 3D gradient magnitude, and 3D gradient direction are used in many of the action-recognition descriptor methods.

Development of 3D descriptors is continuing, which is beyond the scope of this brief introduction. However, for the interested reader, we mention recent work in the areas of volumetric shape descriptors, depth image surface shape descriptors, and 3D reconstruction using depth-based landmark detectors, which can be found in the references [863–870].

3D HOG

The 3D HOG [188] is partially based on some earlier work in volumetric features [191]. The general idea is to employ the familiar HOG descriptor [98] in a 3D HOG descriptor formulation, using a stack of sequential 2D video frames or slices as a 3D volume, and to compute spatiotemporal gradient orientation on adjacent frames within the volume. For efficiency, a novel integral video approach is developed as an alternative to image pyramids based on the same line of thinking as the integral image approach use in the Viola–Jones method.

A similar approach using the integral video concept was also developed in [191] using a subsampled space of 64 $\times$ 64 over 4–40 video frames in the volume, using pixel intensity instead

[2] http://pointclouds.org.

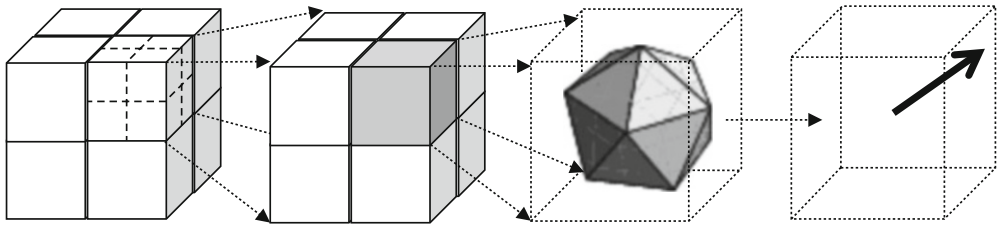


Figure 6.34 HOG 3D descriptor computation. (*Left*) $2 \times 2 \times 2$ descriptor cell block. (*Left center*) Gradient orientation histogram computed over $2 \times 2 \times 2$ cell sub-blocks. (*Right center*) Gradient orientations quantized by projecting the vector intersection to the faces of a 20-faceted icosahedron. (*Right*) Mean gradient orientation computed over integral video blocks (volume vector integral)

of the gradient direction. The integral video method, which can also be considered an *integral volume* method, allows for arbitrary cuboid regions from stacked sequential video frames to be integrated together to compute the local gradient orientation over arbitrary scales. This is space efficient and time efficient compared to using precomputed image pyramids. In fact, this integral video integration method is a novel contribution of the work, and may be applied to other spectra such as intensity, color, and gradient magnitude in either 2D or 3D to eliminate the need for image pyramids—providing more choices in terms of image scale besides just octaves.

The 3D HOG descriptor computations are illustrated in Fig. 6.34. To find feature keypoints to anchor the descriptors, a space-time extension of the Harris operator [189] is used, then a histogram descriptor is computed from the mean of the oriented gradients in a cubic region at the keypoint. Since gradient magnitude is sensitive to illumination changes, gradient orientation is used instead to provide invariance to illumination, and it is computed over 3D cuboid regions using simple x, y, z derivatives. The mean gradient orientation of any 3D cuboid is computed quickly using the integral video method. Gradient orientations are quantized into histogram bins via projection of each vector onto the faces of a regular icosahedron 20-sided shape to combine all vectors, as shown in Fig. 6.34. The 20 icosahedron faces act as the histogram bins. The sparse set of spatiotemporal features is combined into a bag of features or bag of words in a visual vocabulary.

HON 4D

A similar approach to the 3D HOG is called HON 4D [190], which computes descriptors as Histogram of Oriented 4D Normals, where the 3D surface normal + time add up to four dimensions (4D). HON 4D uses sequences of depth images or 3D depth maps as the basis for computing the descriptor, rather than 2D image frames, as in the 3D HOG method. So a depth camera is needed. In this respect, HON 4D is similar to some volume rendering methods which compute 3D surface normals, and may be accelerated using similar methods [434–436].

In the HON 4D method, the surface normals capture the surface shape cues of each object, and changes in normal orientation over time can be used to determine motion and pose. Only the orientation of the surface normal is significant in this method, so the normal lengths are all normalized to unity length. As a result, the binning into histograms acts differently from the HOG style binning, so that the fourth dimension of time encodes differences in the gradient from frame to frame. The HON 4D descriptor is binned and quantized using 4D projector functions, which quantize local surface normal orientation into a 600-cell polychron, which is a geometric extension of a 2D polygon into 4-space.

Consider the discrimination of the HON 4D method using gradient orientation vs. the HOG method using gradient magnitude. If two surfaces are the same or similar with respect to gradient magnitude, the HOG style descriptor cannot differentiate; however, the HON 4D style descriptor can differentiate owing to the orientation of the surface normal used in the descriptor. Of course, computing 3D normals is compute-intensive without special optimizations considering the noncontiguous memory access patterns required to access each component of the volume.

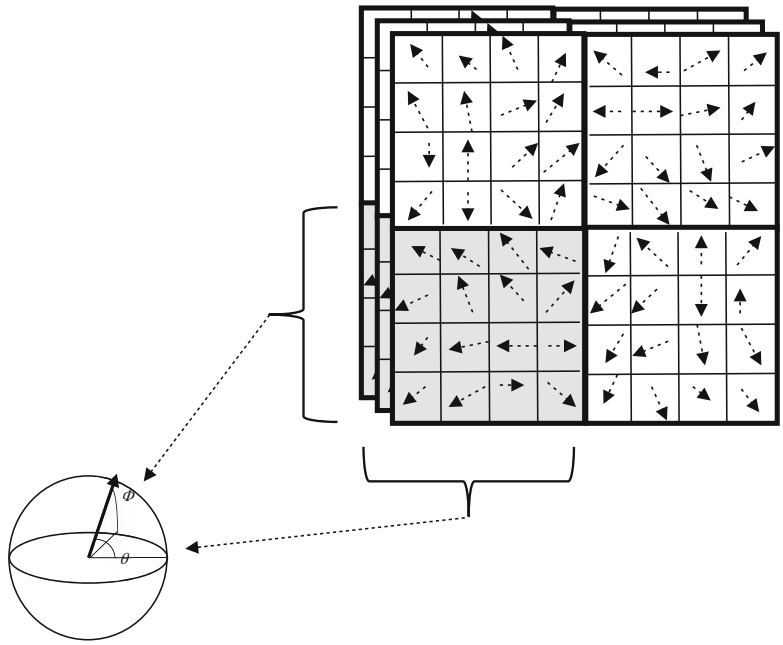


Figure 6.35 Computation of the 3D SIFT [187] vector histogram bins as a combination of the combined gradient orientation of the sub-volumes in a volume space or 3D spatiotemporal region of three consecutive 2D image frames

3D SIFT

The 3D SIFT method [187] starts with the 2D SIFT feature method and reformulates the feature binning to use a volumetric spatiotemporal area $v(x,y,t)$, as shown in Fig. 6.35.

The 3D orientation of the gradient pair orientation is computed as follows:

$$m3D(x,y,t) = \sqrt{L_x^2 + L_y^2 + L_t^2}$$

$$\theta(x,y,t) = \tan^{-1}\left(\frac{L_y}{L_x}\right)$$

$$\phi(x,y,t) = \tan^{-1}\left(\frac{L_{yt}}{\sqrt{L_x^2 + L_y^2}}\right)$$

This method provides a unique two-valued (ϕ, θ) representation for each angle of the gradient orientation in 3-space at each keypoint. The binning stage is handled differently from SIFT, and instead uses orthogonal bins defined by meridians and parallels in a spherical coordinate space. This is simpler to compute, but requires normalization of each value to account for the spherical difference in the apparent size ranging from the poles to the equator.

To compute the SIFT descriptor, the 3D gradient orientation of each sub-histogram is used to guide rotation of the 3D region at the descriptor keypoint to point to 0, which provides a measure of rotational invariance to the descriptor. Each point will be represented as a single gradient magnitude and two orientation vectors (ϕ, θ) instead of one, as in 2D SIFT. The descriptor binning is computed over three dimensions into adjacent cubes instead of over two dimensions in the 2D SIFT descriptor.

Once the feature vectors are binned, the feature vector set is clustered into groups of like features, or words, using hierarchical K-means clustering into a spatiotemporal word vocabulary. Another step beyond the clustering could be to reduce the feature set using sparse coding methods [107–109], but the sparse coding step is not attempted.

Results using 3D SIFT for action recognition are reported to be quite good compared to other similar methods; see reference [187].

Summary

In this chapter we survey a wide range of local interest point detectors and feature descriptor methods to learn "what" practitioners are doing, including both 2D and 3D methods. The vision taxonomy from Chap. 5 is used to divide the feature descriptor survey along the lines of descriptor families, such as local binary methods, spectra methods, and polygon shape methods. There is some overlap between local and regional descriptors; however, this chapter tries to focus on local descriptor methods, leaving regional methods to Chap. 3. Local interest point detectors are discussed in a simple taxonomy including intensity-based regions methods, edge-based region methods, and shape-based region methods, including background on key concepts and mathematics used by many interest point detector methods. Some of the difficulties in choosing an appropriate interest point detector are discussed and several detector methods are surveyed.

This chapter also highlights retrofits to common descriptor methods. For example, many descriptors are retrofitted by changing the descriptor spectra used, such as LBP vs. gradient methods, or by swapping out the interest point detector for a different method. Summary information is provided for feature descriptors following the taxonomy attributes developed in Chap. 5 to enable limited comparisons, using concepts from the analysis of local feature description design concepts presented in Chap. 4.

Chapter 6: Learning Assignments

1. Interest points, or Keypoints, are located in images at locations such as maxima and minima. Describe the types of maxima and minima features found in images.
2. Interest point detectors must be selected and parametrically tuned to give best results. Describe various approaches to select and tune interest point detectors for a range of different types of images.
3. Describe your favorite interest point detector, discuss the advantages compared to other detectors, and describe the basic algorithm.
4. Describe and summarize the names of as many interest point detectors as you can remember, and describe the basic concepts and goals of each algorithm.
5. An interest point adapter function can be devised to help tune interest point parameters to automatically find better interest points. Select an interest point detector of your choice, describe how the interest point detector algorithm works using pseudo code, describe each parameter to the interest point function, describe the image search pattern the adapter could use, and describe parameters to control region size and iterations. (See also assignment 6 below).
6. Write an interest point adapter function using your favorite interest point detector in your favorite programming language, and provide test results.
7. Describe how the local binary pattern (LBP) algorithm works using pseudo code.
8. List a few applications for the local binary pattern.
9. Describe how the local binary pattern can be stored in a rotationally invariant format.
10. Compare local binary pattern algorithms including Brief, Brisk, Orb, and Freak, and highlight the differences in the pixel region sampling patterns.
11. List the distance function most applicable to local binary descriptors, and how it can be optimized.
12. Describe the basic SIFT algorithm, highlighting the scales over which the pixel regions are sampled, the algorithm for detecting interest points, the algorithm for computing the feature descriptor, and the summary information stored in the descriptor.
13. Describe at least one enhancement to the basic SIFT algorithm, such as SIFT-PCA and SIFT-GLOH, SIFT-SIFER, or RootSIFT, and highlight the major improvements provided by the enhancement.
14. Describe the pixel patch region shape and sizes used in the SIFT algorithm, and describe how the pixel samples are weighted within the region.
15. Discuss the SURF feature descriptor algorithm.
16. Compare the local binary feature descriptors ORB, FREAK, and BRISK.
17. Describe how HAAR-like features are used in feature description, draw or describe a few example HAAR-like features, and discuss how HAAR features are related to Wavelets.
18. Describe integral images, how they are built, and discuss why integral images can be used to optimize working with HAAR filters.
19. Describe the Viola–Jones feature classification funnel and pipeline.
20. Design an algorithm to compute *gradient histograms* from a local region, describe how to create a useful feature descriptor from the *gradient histograms*, and select a specific distance function that could be applied to measure correspondence between gradient histograms, and discuss the strengths and weaknesses of your algorithm.
21. Describe the algorithm for your favorite feature descriptor, discuss the advantages, and provide simple comparisons to a few other feature descriptors.

22. Describe how a chain code histogram is computed.
23. Describe how a polygon feature shape can be refined (for example using morphology operations, thresholding operations), and then describe the types of feature metrics that can be computed over polygon shapes.
24. List at least five polygon shape feature metrics and describe how they are computed, including perimeter and centroid.