25 Scale-Invariant Feature Transform (SIFT)

Many real applications require the localization of reference positions in one or more images, for example, for image alignment, removing distortions, object tracking, 3D reconstruction, etc. We have seen that corner points[1] can be located quite reliably and independent of orientation. However, typical corner detectors only provide the position and strength of each candidate point, they do not provide any information about its characteristic or "identity" that could be used for matching. Another limitation is that most corner detectors only operate at a particular scale or resolution, since they are based on a rigid set of filters.

This chapter describes the *Scale-Invariant Feature Transform* (SIFT) technique for local feature detection, which was originally proposed by D. Lowe [152] and has since become a "workhorse" method in the imaging industry. Its goal is to locate image features that can be identified robustly to facilitate matching in multiple images and image sequences as well as object recognition under different viewing conditions. SIFT employs the concept of "scale space" [151] to capture features at *multiple* scale levels or image resolutions, which not only increases the number of available features but also makes the method highly tolerant to scale changes. This makes it possible, for example, to track features on objects that move towards the camera and thereby change their scale continuously or to stitch together images taken with widely different zoom settings.

Accelerated variants of the SIFT algorithm have been implemented by streamlining the scale space calculation and feature detection or the use of GPU hardware [20, 90, 218].

In principle, SIFT works like a multi-scale corner detector with sub-pixel positioning accuracy and a rotation-invariant feature descriptor attached to each candidate point. This (typically 128-dimensional) feature descriptor summarizes the distribution of the gradient directions in a spatial neighborhood around the corresponding feature point and can thus be used like a "fingerprint". The main steps involved in the calculation of SIFT features are as follows:

[1] See Chapter 7.

© Springer-Verlag London 2016
W. Burger, M.J. Burge, *Digital Image Processing*, Texts in Computer Science,
DOI 10.1007/978-1-4471-6684-9_25

1. Extrema detection in a Laplacian-of-Gaussian (LoG) scale space to locate potential interest points.
2. Key point refinement by fitting a continuous model to determine precise location and scale.
3. Orientation assignment by the dominant orientation of the feature point from the directions of the surrounding image gradients.
4. Formation of the feature descriptor by normalizing the local gradient histogram.

These steps are all described in the remaining parts of this chapter. There are several reasons why we explain the SIFT technique here at such great detail. For one, it is by far the most complex algorithm that we have looked at so far, its individual steps are carefully designed and delicately interdependent, with numerous parameters that need to be considered. A good understanding of the inner workings and limitations is thus important for successful use as well as for analyzing problems if the results are not as expected.

25.1 Interest Points at Multiple Scales

The first step in detecting interest points is to find locations with stable features that can be localized under a wide range of viewing conditions and different scales. In the SIFT approach, interest point detection is based on Laplacian-of-Gaussian (LoG) filters, which respond primarily to distinct bright blobs surrounded by darker regions, or vice versa. Unlike the filters used in popular corner detectors,[2] LoG filters are *isotropic*, i.e., insensitive to orientation. To locate interest points over multiple scales, a scale space representation of the input image is constructed by recursively smoothing the image with a sequence of small Gaussian filters. The difference between the images in adjacent scale layers is used to approximate the LoG filter at each scale. Interest points are finally selected by finding the local maxima in the 3D LoG scale space.

25.1.1 The LoG Filter

In this section, we first outline LoG filters and the basic construction of a Gaussian scale space, followed by a detailed description of the actual implementation and the parameters used in the SIFT approach.

The LoG is a so-called *center-surround* operator, which most strongly responds to isolated local intensity peaks, edge, and corner-like image structures. The corresponding filter kernel is based on the second derivative of the Gaussian function, as illustrated in Fig. 25.1 for the 1D case. The 1D Gaussian function of width σ is defined as

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{x^2}{2\sigma^2}} \tag{25.1}$$

and its *first* derivative is

[2] See Chapter 7.

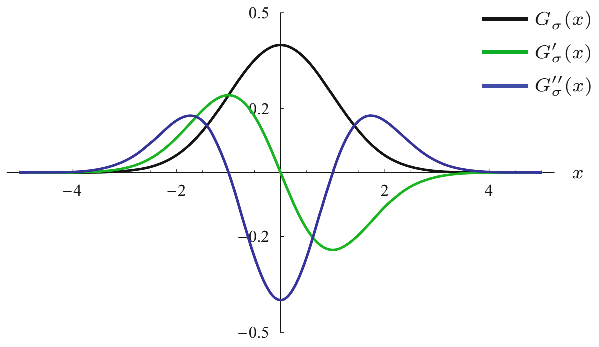


Fig. 25.1
1D Gaussian function $G_\sigma(x)$ with $\sigma = 1$ (black), its first derivative $G'_\sigma(x)$ (green) and second derivative $G''_\sigma(x)$ (blue).

$$G'_\sigma(x) = \frac{\mathrm{d}G_\sigma}{\mathrm{d}\,x}(x) = -\frac{x}{\sqrt{2\pi}\cdot\sigma^3}\cdot e^{-\frac{x^2}{2\sigma^2}}. \tag{25.2}$$

Analogously, the *second* derivative of the 1D Gaussian is

$$G''_\sigma(x) = \frac{\mathrm{d}^2G_\sigma}{\mathrm{d}\,x^2}(x) = \frac{x^2-\sigma^2}{\sqrt{2\pi}\cdot\sigma^5}\cdot e^{-\frac{x^2}{2\sigma^2}}. \tag{25.3}$$

The *Laplacian* (denoted ∇^2) of a continuous, 2D function $f(x,y)$ is defined as the sum of the second partial derivatives for the x- and y-directions, traditionally written as

$$\left(\nabla^2 f\right)(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y)\,. \tag{25.4}$$

Note that, unlike the *gradient*[3] of a 2D function, the result of the Laplacian is not a vector but a *scalar* quantity. Its value is invariant against rotations of the coordinate system, that is, the Laplacian operator has the important property of being *isotropic*.

By applying the *Laplacian* operator to a rotationally symmetric 2D Gaussian,

$$G_\sigma(x,y) = \frac{1}{2\pi\cdot\sigma^2}\cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{25.5}$$

with identical widths $\sigma = \sigma_x = \sigma_y$ in the x/y directions (see Fig. 25.2(a)), we obtain the LoG function

$$\begin{aligned} L_\sigma(x,y) = \left(\nabla^2 G_\sigma\right)(x,y) &= \frac{\partial^2 G_\sigma}{\partial x^2}(x,y) + \frac{\partial^2 G_\sigma}{\partial y^2}(x,y) \\ &= \frac{(x^2-\sigma^2)}{2\pi\cdot\sigma^6}\cdot e^{-\frac{x^2+y^2}{2\cdot\sigma^2}} + \frac{(y^2-\sigma^2)}{2\pi\cdot\sigma^6}\cdot e^{-\frac{x^2+y^2}{2\cdot\sigma^2}} \\ &= \frac{1}{\pi\cdot\sigma^4}\cdot\left(\frac{x^2+y^2-2\sigma^2}{2\cdot\sigma^2}\right)\cdot e^{-\frac{x^2+y^2}{2\cdot\sigma^2}}, \end{aligned} \tag{25.6}$$

as shown in Fig. 25.2(b). The continuous LoG function in Eqn. (25.6) has the absolute value integral

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |L_\sigma(x,y)|\,\mathrm{d}x\,\mathrm{d}y \;=\; \frac{4}{\sigma^2 e}, \tag{25.7}$$

[3] See Chapter 6, Sec. 6.2.1.

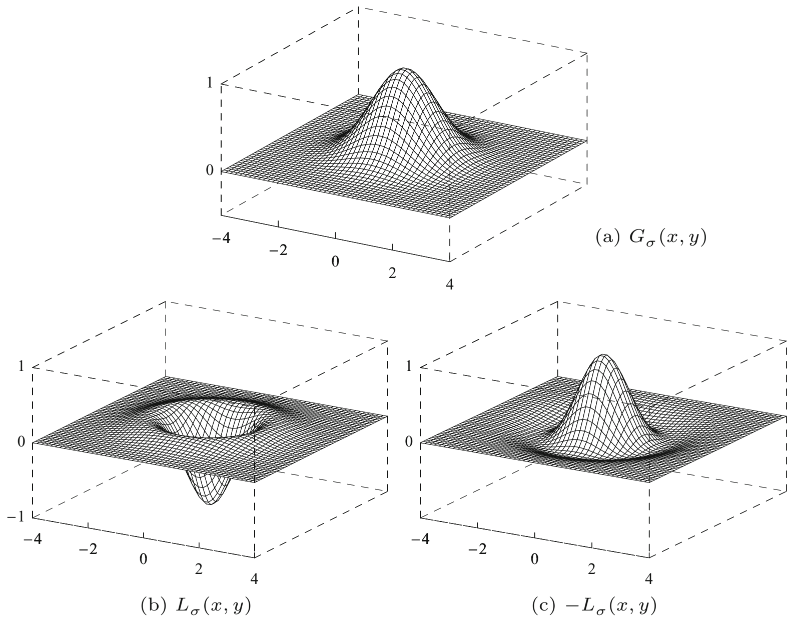


Fig. 25.2
2D Gaussian and LoG. Gaussian function $G_\sigma(x, y)$ with $\sigma = 1$ (a); the corresponding LoG function $L_\sigma(x, y)$ in (b), and the inverted function ("Mexican hat" or "Sombrero" kernel) $-L_\sigma(x, y)$ in (c). For illustration, all three functions are normalized to an absolute value of 1 at the origin.

and zero average, that is,

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} L_\sigma(x, y)\, \mathrm{d}x\, \mathrm{d}y \;=\; 0\,. \tag{25.8}$$

When used as the kernel of a linear filter,[4] the LoG responds maximally to circular spots that are *darker* than the surrounding background and have a radius of approximately σ.[5] Blobs that are *brighter* than the surrounding background are enhanced by filtering with the negative LoG kernel, that is, $-L_\sigma$, which is often referred to as the "Mexican hat" or "Sombrero" filter (see Fig. 25.2). Both types of blobs can be detected simultaneously by simply taking the absolute value of the filter response (see Fig. 25.3).

Since the LoG function is based on derivatives, its magnitude strongly depends on the steepness of the Gaussian slope, which is controlled by σ. To obtain responses of comparable magnitude over multiple scales, a *scale normalized* LoG kernel can be defined in the form [151]

$$\hat{L}_\sigma(x, y) = \sigma^2 \cdot \left(\nabla^2 G_\sigma\right)(x, y) = \sigma^2 \cdot L_\sigma(x, y) \tag{25.9}$$

$$= \frac{1}{\pi\sigma^2} \cdot \left(\frac{x^2 + y^2 - 2\sigma^2}{2\sigma^2}\right) \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}\,. \tag{25.10}$$

[4] To produce a sufficiently accurate discrete LoG filter kernel, the support radius should be set to at least 4σ (kernel diameter $\geq 8\sigma$).

[5] The LoG is often used as a model for early processes in biological vision systems [161], particularly to describe the center-surround response of receptive fields. In this model, an "on-center" cell is *stimulated* when the center of its receptive field is exposed to light, and is *inhibited* when light falls on its surround. Conversely, an "off-center" cell is stimulated by light falling on its surround. Thus filtering with the original LoG L_σ (Eqn. (25.6)) corresponds to the behavior of off-center cells, while the response to the negative LoG kernel $-L_\sigma$ is that of an on-center cell.

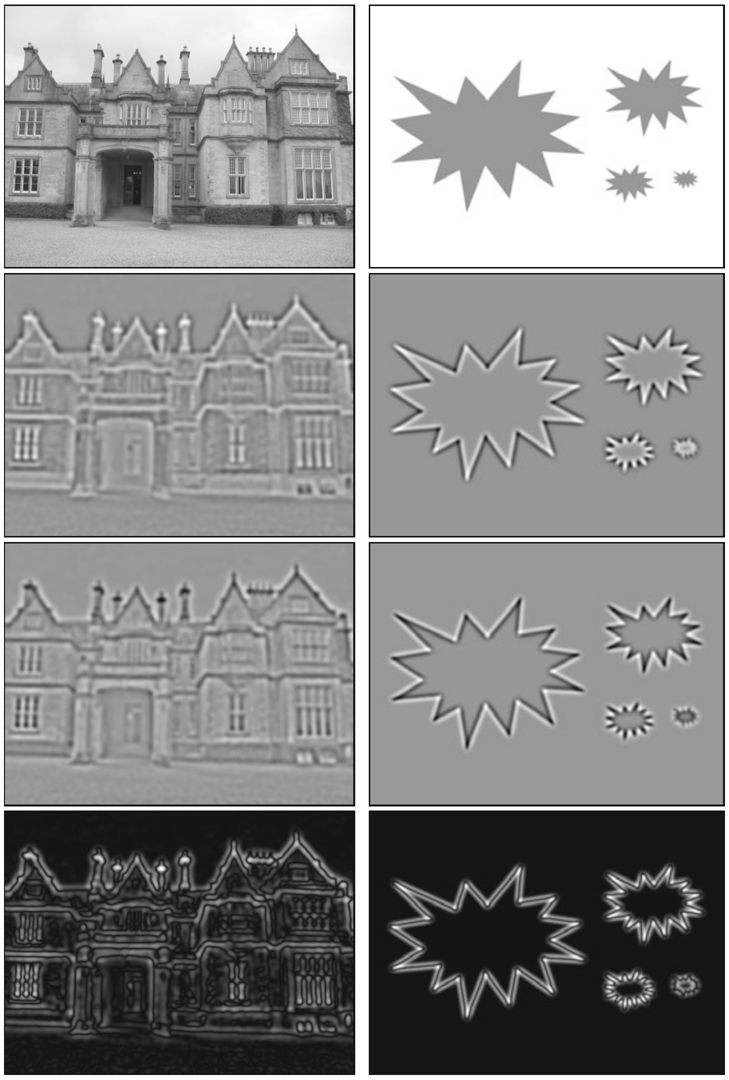
(a)
(b)
(c)
(d)


Fig. 25.3
Filtering with the LoG kernel (with $\sigma = 3$). Original images (a). A linear filter with the LoG kernel $L_\sigma(x, y)$ responds strongest to dark spots in a bright surround (b), while the inverted kernel $-L_\sigma(x, y)$ responds strongest to bright spots in a dark surround (c). In (b, c), zero values are shown as medium gray, negative values are dark, positive values are bright. The absolute value of (b) or (c) combines the responses from both dark and bright spots (d).

Note that the integral of this function,

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \big|\hat{L}_\sigma(x, y)\big| \, \mathrm{d}x \, \mathrm{d}y \;=\; \frac{4}{e}, \tag{25.11}$$

is constant and thus (unlike Eqn. (25.7)) independent of the scale parameter σ (see Fig. 25.4).

Approximating the LoG by the difference of two Gaussians (DoG)

Although the LoG is "quasi-separable" [113, 243] and can thus be calculated efficiently, the most common method for implementing the LoG filter is to approximate it by the *difference of two Gaussians* (DoG) of widths σ and $\kappa\sigma$, respectively, that is,

$$L_\sigma(x, y) \approx \lambda \cdot \big[\underbrace{G_{\kappa\sigma}(x, y) - G_\sigma(x, y)}_{= D_{\sigma,\kappa}(x,y)}\big] \tag{25.12}$$

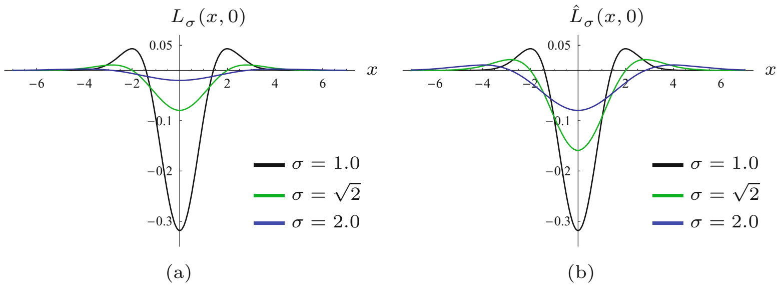


Fig. 25.4
Normalization of the LoG function. Cross section of LoG function $L_\sigma(x, y)$ as defined in Eqn. (25.6) (a); scale-normalized LoG (b) as defined in Eqn. (25.10). $\sigma = 1.0$ (black), $\sigma = \sqrt{2}$ (green), $\sigma = 2.0$ (blue). All three functions in (b) have the same absolute value integral that is independent of σ (see Eqn. (25.11)).

with the parameter $\kappa > 1$ specifying the relative width of the two Gaussians (defined in Eqn. (25.5)). Properly scaled (by some factor λ, see Eqn. (25.13)), the DOG function $D_{\sigma,\kappa}(x, y)$ approximates the LoG function $L_\sigma(x, y)$ in Eqn. (25.6) with arbitrary precision, as κ approaches 1 ($\kappa = 1$ being excluded, of course). In practice, values of κ in the range $1.1, \ldots, 1.3$ yield sufficiently accurate results. As an example, Fig. 25.5 shows the cross-section of the 2D DoG function for $\kappa = 2^{1/3} \approx 1.25992$.[6]

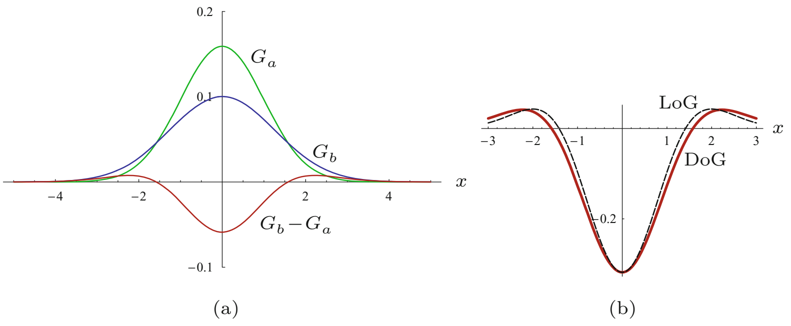


Fig. 25.5
Approximating the LoG by the DoG. The two original Gaussians, $G_a(x)$ with $\sigma_a = 1.0$ and $G_b(x)$ with $\sigma_b = \sigma_a \cdot \kappa = \kappa = 2^{1/3}$, shown by the green and blue curves, respectively (a). The red curve in (a) shows the DoG function $D_{\sigma,\kappa}(x, y) = G_b(x, y) - G_a(x, y)$ for $y = 0$. In (b), the dashed line shows the reference LoG function in comparison to the DoG (red). The DoG is scaled to match the magnitude of the LoG function.

The factor $\lambda \in \mathbb{R}$ in Eqn. (25.12) controls the magnitude of the DoG function; it depends on both the ratio κ and the scale parameter σ. To match the magnitude of the original LoG (Eqn. (25.6)) at the origin, it must be set to

$$\lambda = \frac{2\kappa^2}{\sigma^2 \cdot (\kappa^2 - 1)}. \tag{25.13}$$

Similarly, the *scale-normalized* LoG $\hat{L}_\sigma$ (Eqn. (25.10)) can be approximated by the DoG function $D_{\sigma,\kappa}$ (Eqn. (25.12)) as

$$\begin{aligned}\hat{L}_\sigma(x, y) &= \sigma^2 L_\sigma(x, y) \\ &\approx \underbrace{\sigma^2 \cdot \lambda}_{\hat{\lambda}} \cdot D_{\sigma,\kappa}(x, y) = \frac{2\kappa^2}{\kappa^2 - 1} \cdot D_{\sigma,\kappa}(x, y),\end{aligned} \tag{25.14}$$

[6] The factor $\kappa = 2^{1/3}$ originates from splitting the scale interval 2 (i.e., one scale octave) into 3 equal intervals, as described later on. Another factor mentioned frequently in the literature is 1.6, which, however, does not yield a satisfactory approximation. Possibly that value refers to the ratio of the *variances* σ_2^2/σ_1^2 and not the ratio of the *standard deviations* σ_2/σ_1.

$\mathcal{G}(x, y, \sigma)$	continuous Gaussian scale space
$\mathsf{G} = (\mathsf{G}_0, \ldots, \mathsf{G}_{K-1})$	discrete Gaussian scale space with K levels
G_k	single level in a discrete Gaussian scale space
$\mathsf{L} = (\mathsf{L}_0, \ldots, \mathsf{L}_{K-1})$	discrete LoG scale space with K levels
L_k	single level in a LoG scale space
$\mathsf{D} = (\mathsf{D}_0, \ldots, \mathsf{D}_{K-1})$	discrete DoG scale space with K levels
D_k	single level in a DoG scale space
$\mathbf{G} = (\mathbf{G}_0, \ldots, \mathbf{G}_{P-1})$	hierarchical Gaussian scale space with P octaves
$\mathbf{G}_p = (\mathbf{G}_{p,0}, \ldots, \mathbf{G}_{p,Q-1})$	octave in a hier. Gaussian scale space with Q levels
$\mathbf{G}_{p,q}$	single level in a hierarchical Gaussian scale space
$\mathbf{D} = (\mathbf{D}_0, \ldots, \mathbf{D}_{P-1})$	hierarchical DoG scale space with P octaves
$\mathbf{D}_p = (\mathbf{D}_{p,0}, \ldots, \mathbf{D}_{p,Q-1})$	octave in a hierarchical DoG scale space with Q levels
$\mathbf{D}_{p,q}$	single level in a hierarchical DoG scale space
$\mathsf{N}_{\mathbf{c}}(i, j, k)$	$3\times3\times3$ neigborhood in DoG scale space
$\boldsymbol{k} = (p, q, u, v)$	discrete key point position in hierarchical scale space ($p, q, u, v \in \mathbb{Z}$)
$\boldsymbol{k}' = (p, q, x, y)$	continuous (refined) key point position ($x, y \in \mathbb{R}$)

Table 25.1
Scale space-related symbols used in this chapter.

with the factor $\hat{\lambda} = \sigma^2 \cdot \lambda = 2\kappa^2/(\kappa^2 - 1)$ being constant and therefore independent of the scale σ. Thus, as pointed out in [153], with a fixed scale increment κ, the DoG already approximates the scale-normalized LoG up to a constant factor, and thus no additional scaling is required to compare the magnitudes of the DoG responses obtained at different scales.[7]

In the SIFT approach, the DoG is used as an approximation of the (scale-normalized) LoG filter at multiple scales, based on a Gaussian scale space representation of the input image that is described next.[8]

25.1.2 Gaussian Scale Space

The concept of scale space [150] is motivated by the observation that real-world scenes exhibit relevant image features over a large range of sizes and, depending on the particular viewing situation, at various different scales. To relate image structures at different and unknown sizes, it is useful to represent the images simultaneously at different scale levels. The scale space representation of an image adds *scale* as a third coordinate (in addition to the two image coordinates). Thus the scale space is a 3D structure, which can be navigated not only along the x/y positions but also across different scale levels.

Continuous Gaussian scale space

The scale-space representation of an image at a particular scale level is obtained by filtering the image with a kernel that is parameterized to the desired scale. Because of its unique properties [11, 71], the most common type of scale space is based on successive filtering with Gaussian kernels. Conceptually, given a continuous, 2D function $F(x, y)$, its Gaussian scale space representation is a 3D function

[7] See Sec. E.4 in the Appendix for additional details.
[8] See Table 25.1 for a summary of the most important scale space-related symbols used in this chapter.

$$\mathcal{G}(x,y,\sigma) = (F * H^{\mathrm{G},\sigma})(x,y), \tag{25.15}$$

where $H^{\mathrm{G},\sigma} \equiv G_\sigma(x,y)$ is a 2D Gaussian kernel (see Eqn. (25.5)) with unit integral, and $*$ denotes the linear convolution over x, y. Note that $\sigma \geq 0$ serves as both the continuous scale parameter and the width of the corresponding Gaussian filter kernel.

A fully continuous Gaussian scale space $\mathcal{G}(x,y,\sigma)$ covers a 3D volume and represents the original function $F(x,y)$ at varying scales σ. For $\sigma = 0$, the Gaussian kernel $H^{\mathrm{G},0}$ has zero width, which makes it equivalent to an impulse or Dirac function $\delta(x,y)$.[9] This is the neutral element of linear convolution, that is,

$$\mathcal{G}(x,y,0) = (F * H^{\mathrm{G},0})(x,y) = (F * \delta)(x,y) = F(x,y). \tag{25.16}$$

Thus the base level $\mathcal{G}(x,y,0)$ of the Gaussian scale space is identical to the input function $F(x,y)$. In general (with $\sigma > 0$), the Gaussian kernel $H^{\mathrm{G},\sigma}$ acts as a low-pass filter with a cutoff frequency proportional to $1/\sigma$ (see Sec. E.3 in the Appendix), the maximum frequency (or bandwidth) of the original "signal" $F(x,y)$ being potentially unlimited.

Discrete Gaussian scale space

This is different for a *discrete* input function $I(u,v)$, whose bandwidth is implicitly limited to half the sampling frequency, as mandated by the sampling theorem to avoid aliasing.[10] Thus, in the discrete case, the lowest level $\mathcal{G}(x,y,0)$ of the Gaussian scale space is not accessible! To model the implicit bandwidth limitations of the sampling process, the discrete input image $I(u,v)$ is assumed to be pre-filtered (with respect to the underlying continuous signal) with a Gaussian kernel of width $\sigma_\mathrm{s} \geq 0.5$ [153], that is,

$$\mathcal{G}(u,v,\sigma_\mathrm{s}) \;\equiv\; I(u,v). \tag{25.17}$$

Thus the discrete input image $I(u,v)$ is implicitly placed at some initial level σ_s of the Gaussian scale space, and the lower levels with $\sigma < \sigma_\mathrm{s}$ are not available.

Any higher level $\sigma_\mathrm{h} > \sigma_\mathrm{s}$ of the Gaussian scale space can be derived from the original image $I(u,v)$ by filtering with Gaussian kernel $H^{\mathrm{G},\bar{\sigma}}$, that is,

$$\mathcal{G}(u,v,\sigma_\mathrm{h}) = (I * H^{\mathrm{G},\bar{\sigma}})(u,v), \quad \text{with } \bar{\sigma} = \sqrt{\sigma_\mathrm{h}^2 - \sigma_\mathrm{s}^2}\,. \tag{25.18}$$

This is due to the fact that applying two Gaussian filters of widths σ_1 and σ_2, one after the other, is equivalent to a single convolution with a Gaussian kernel of width $\sigma_{1,2}$, that is,[11]

$$\bigl(I * H^{\mathrm{G},\sigma_1}\bigr) * H^{\mathrm{G},\sigma_2} \equiv I * H^{\mathrm{G},\sigma_{1,2}}, \tag{25.19}$$

[9] See Chapter 5, Sec. 5.3.4.

[10] See Chapter 18, Sec. 18.2.1.

[11] See Sec. E.1 in the Appendix for additional details on combining Gaussian filters.

with $\sigma_{1,2} = (\sigma_1^2 + \sigma_2^2)^{1/2}$. We define the *discrete Gaussian scale space* representation of an image I as a vector of M images, one for each scale level m:

$$\mathsf{G} = (\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{M-1}). \tag{25.20}$$

Associated with each level G_m is its absolute scale $\sigma_m > 0$, and each level G_m represents a blurred version of the original image, that is, $\mathsf{G}_m(u,v) \equiv \mathcal{G}(u,v,\sigma_m)$ in the notation introduced in Eqn. (25.15). The scale ratio between adjacent scale levels,

$$\varDelta_\sigma = \frac{\sigma_{m+1}}{\sigma_m}, \tag{25.21}$$

is pre-defined and constant. Usually, $\varDelta_\sigma$ is specified such that the absolute scale σ_m doubles with a given number of levels Q, called an *octave*. In this case, the resulting scale increment is $\varDelta_\sigma = 2^{1/Q}$ with (typically) $Q = 3, \ldots, 6$.

In addition, a *base scale* $\sigma_0 > \sigma_\mathrm{s}$ is specified for the initial level G_0, with σ_s denoting the smoothing of the discrete image implied by the sampling process, as discussed already. Based on empirical results, a base scale of $\sigma_0 = 1.6$ is recommended in [153] to achieve reliable interest point detection. Given Q and the base scale σ_0, the absolute scale at an arbitrary scale space level G_m is

$$\sigma_m = \sigma_0 \cdot \varDelta_\sigma^m = \sigma_0 \cdot 2^{m/Q}, \tag{25.22}$$

for $m = 0, \ldots, M-1$.

As follows from Eqn. (25.18), each scale level G_m can be obtained directly from the discrete input image I by a filter operation

$$\mathsf{G}_m = I * H^{\mathrm{G},\bar{\sigma}_m}, \tag{25.23}$$

with a Gaussian kernel $H^{\mathrm{G},\bar{\sigma}_m}$ of width

$$\bar{\sigma}_m = \sqrt{\sigma_m^2 - \sigma_\mathrm{s}^2} = \sqrt{\sigma_0^2 \cdot 2^{2m/Q} - \sigma_\mathrm{s}^2}. \tag{25.24}$$

In particular, the initial scale space level G_0, (with the specified base scale σ_0) is obtained from the discrete input image I by linear filtering using a Gaussian kernel of width

$$\bar{\sigma}_0 = \sqrt{\sigma_0^2 - \sigma_\mathrm{s}^2}. \tag{25.25}$$

Alternatively, using the relation $\sigma_m = \sigma_{m-1} \cdot \varDelta_\sigma$ (from Eqn. (25.21)), the scale levels $\mathsf{G}_1, \ldots, \mathsf{G}_{M-1}$ could be calculated recursively from the base level G_0 in the form

$$\mathsf{G}_m = \mathsf{G}_{m-1} * H^{\mathrm{G},\sigma'_m}, \tag{25.26}$$

for $m > 0$, with a sequence of Gaussian kernels H^{G,σ'_m} of width

$$\sigma'_m = \sqrt{\sigma_m^2 - \sigma_{m-1}^2} = \sigma_0 \cdot 2^{m/Q} \cdot \sqrt{1 - 1/\varDelta_\sigma^2}\,. \tag{25.27}$$

Table 25.2 lists the resulting kernel widths for $Q = 3$ levels per octave and base scale $\sigma_0 = 1.6$ over a scale range of 6 octaves. The value $\bar{\sigma}_m$ denotes the size of the Gaussian kernel required to compute the image at scale m from the discrete input image I (assumed to be sampled with $\sigma_s = 0.5$). σ'_m is the width of the Gaussian kernel to compute level m recursively from the previous level $m-1$. Apparently (though perhaps unexpectedly), the kernel size required for recursive filtering (σ'_m) grows at the same (exponential) rate as the absolute kernel size $\bar{\sigma}_m$.[12]

Table 25.2
Filter sizes required for calculating Gaussian scale levels G_m for the first 6 octaves. Each octave consists of $Q = 3$ levels, placed at increments of Δ_σ along the scale coordinate. The discrete input image I is assumed to be pre-filtered with σ_s. Column σ_m denotes the absolute scale at level m, starting with the specified base offset scale σ_0. $\bar{\sigma}_m$ is the width of the Gaussian filter required to calculate level G_m directly from the input image I. Values σ'_m are the widths of the Gaussian kernels required to calculate level G_m from the previous level G_{m-1}. Note that the width of the Gaussian kernels needed for recursive filtering (σ'_m) grows at the same exponential rate as the size of the direct filter ($\bar{\sigma}_m$).

m	σ_m	$\bar{\sigma}_m$	σ'_m
18	102.4000	102.3988	62.2908
17	81.2749	81.2734	49.4402
16	64.5080	64.5060	39.2408
15	51.2000	51.1976	31.1454
14	40.6375	40.6344	24.7201
13	32.2540	32.2501	19.6204
12	25.6000	25.5951	15.5727
11	20.3187	20.3126	12.3601
10	16.1270	16.1192	9.8102
9	12.8000	12.7902	7.7864
8	10.1594	10.1471	6.1800
7	8.0635	8.0480	4.9051
6	6.4000	6.3804	3.8932
5	5.0797	5.0550	3.0900
4	4.0317	4.0006	2.4525
3	3.2000	3.1607	1.9466
2	2.5398	2.4901	1.5450
1	2.0159	1.9529	1.2263
0	1.6000	1.5199	—

m	. . . linear scale index
σ_m	. . . absolute scale at level m (Eqn. (25.22))
$\bar{\sigma}_m$	. . . relative scale at level m w.r.t. the original image (Eqn. (25.24))
σ'_m	. . . relative scale at level m w.r.t. the previous level $m-1$ (Eqn. (25.27))

$\sigma_s = 0.5$ (sampling scale)
$\sigma_0 = 1.6$ (base scale)
$Q = 3$ (levels per octave)
$\Delta_\sigma = 2^{1/Q} \approx 1.256$

At scale level $m = 16$ and absolute scale $\sigma_{16} = 1.6 \cdot 2^{16/3} \approx 64.5$, for example, the Gaussian filters required to compute G_{16} directly from the input image I has the width $\bar{\sigma}_{16} = (\sigma_{16}^2 - \sigma_s^2)^{1/2} = (64.5080^2 - 0.5^2)^{1/2} \approx 64.5$, while the filter to blur incrementally from the previous scale level has the width $\sigma'_{16} = (\sigma_{16}^2 - \sigma_{15}^2)^{1/2} = (64.5080^2 - 51.1976^2)^{1/2} \approx 39.2$. Since recursive filtering also tends to accrue numerical inaccuracies, this approach does not offer a significant advantage in general. Fortunately, the growth of the Gaussian kernels can be kept small by spatially sub-sampling after each octave, as will be described in Sec. 25.1.4.

The process of constructing a discrete Gaussian scale space using the same parameters as in Table 25.2 is illustrated in Fig. 25.6. Again the input image I is assumed to be pre-filtered at $\sigma_s = 0.5$ due to sampling and the absolute scale of the first level G_0 is set to $\sigma_0 = 1.6$. The scale ratio between successive levels is fixed at $\Delta_\sigma = 2^{1/3} \approx 1.25992$, that is, each octave spans three discrete scale levels. As shown in this figure, each scale level G_m can be calculated either directly from the input image I by filtering with a Gaussian of width $\bar{\sigma}_m$, or recursively from the previous level by filtering with σ'_m.

[12] The ratio of the kernel sizes $\bar{\sigma}_m/\sigma'_m$ converges to $\sqrt{1 - 1/\Delta_\sigma^2}$ (≈ 1.64 for $Q = 3$) and is thus practically constant for larger values of m.

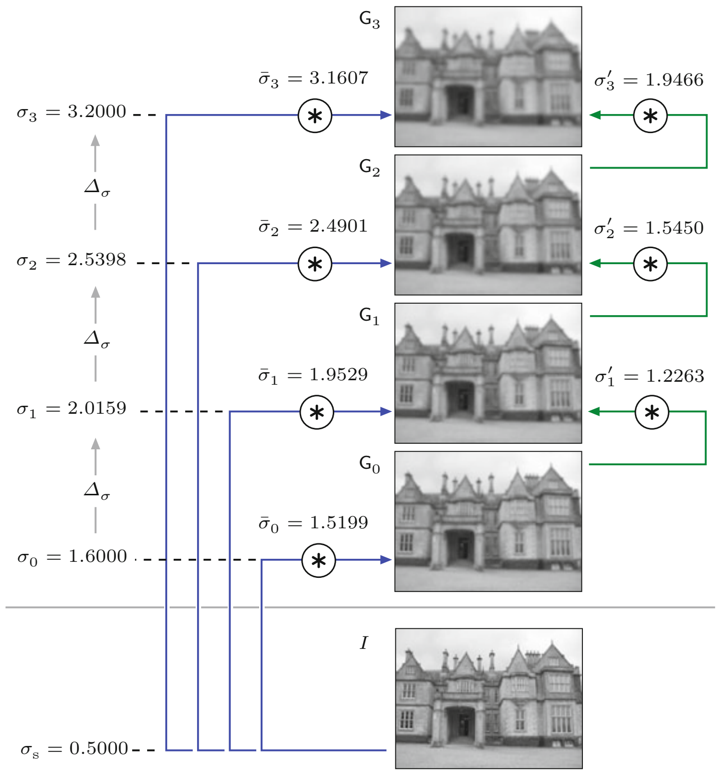


Fig. 25.6
Gaussian scale space construction (first four levels). Parameters are the same as in Table 25.2. The discrete input image I is assumed to be pre-filtered with a Gaussian of width $\sigma_s = 0.5$; the scale of the initial level (base scale offset) is set to $\sigma_0 = 1.6$. The discrete scale space levels $\mathsf{G}_0, \mathsf{G}_1, \ldots$ (at absolute scales $\sigma_0, \sigma_1, \ldots$) are slices through the continuous scale space. Scale levels can either be calculated by filtering directly from the discrete image I with Gaussian kernels of width $\bar{\sigma}_0, \bar{\sigma}_1, \ldots$ (blue arrows) or, alternatively, by recursively filtering with $\sigma'_1, \sigma'_2, \ldots$ (green arrows).

25.1.3 LoG/DoG Scale Space

Interest point detection in the SIFT approach is based on finding local maxima in the output of LoG filters over multiple scales. Analogous to the discrete Gaussian scale space described in Sec. 25.1.2, a LoG scale space representation of an image I can be defined as

$$\mathsf{L} = (\mathsf{L}_0, \mathsf{L}_1, \ldots, \mathsf{L}_{M-1}), \tag{25.28}$$

with levels $\mathsf{L}_m = I * H^{\mathrm{L},\sigma_m}$, where $H^{\mathrm{L},\sigma_m}(x,y) \equiv \hat{L}_{\sigma_m}(x,y)$ is a scale-normalized LoG kernel of width σ_m (see Eqn. (25.10)).

As demonstrated in Eqn. (25.12), the LoG kernel can be approximated by the the difference of two Gaussians whose widths differ by a certain ratio κ. Since pairs of adjacent scale layers in the Gaussian scale space are also separated by a fixed scale ratio, it is straightforward to construct a multi-scale DoG representation,

$$\mathsf{D} = (\mathsf{D}_0, \mathsf{D}_1, \ldots, \mathsf{D}_{M-2}) \tag{25.29}$$

from an existing Gaussian scale space $\mathsf{G} = (\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{M-1})$. The individual levels in the DoG scale space are defined as

$$\mathsf{D}_m = \hat{\lambda} \cdot (\mathsf{G}_{m+1} - \mathsf{G}_m) \approx \mathsf{L}_m, \tag{25.30}$$

for $m = 0, \ldots, M-2$. The constant factor $\hat{\lambda}$ (defined in Eqn. (25.14)) can be omitted in the aforementioned expression, as the relative width of the involved Gaussians,

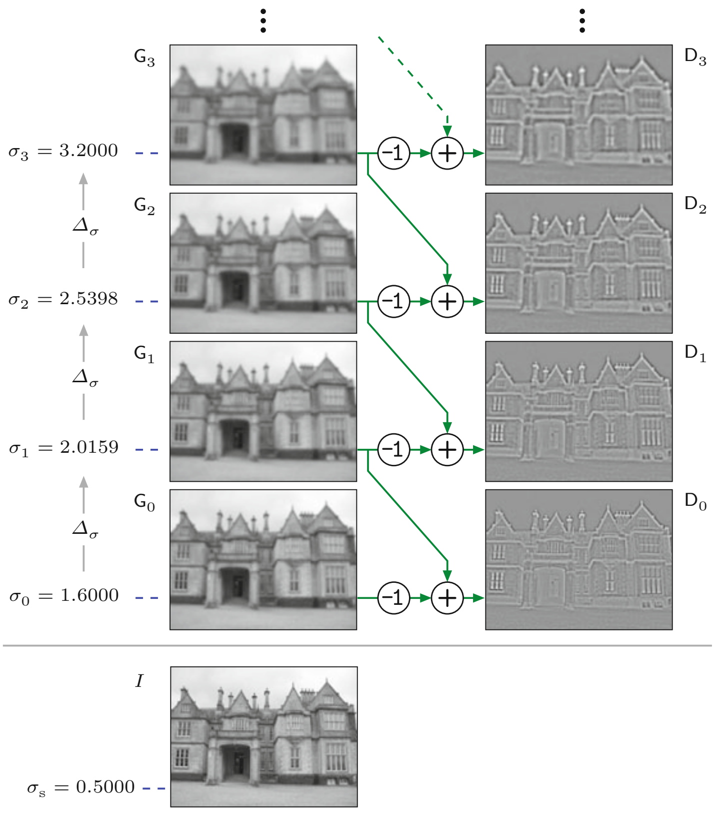


Fig. 25.7
DoG scale-space construction. The differences of successive levels $G_0, G_1, \ldots$ of the Gaussian scale space (see Fig. 25.6) are used to approximate a LoG scale space. Each DoG-level D_m is calculated as the point-wise difference $G_{m+1} - G_m$ between Gaussian levels G_{m+1} and G_m. The values in $D_0, \ldots, D_3$ are scale-normalized (see Eqn. (25.14)) and mapped to a uniform intensity range for viewing.

$$\kappa = \Delta_\sigma = \frac{\sigma_{m+1}}{\sigma_m} = 2^{1/Q}, \qquad (25.31)$$

is simply the fixed scale ratio Δ_σ between successive scale space levels. Note that the DoG approximation does not require any additional normalization to approximate a scale-normalized LoG representation (see Eqns. 25.10 and 25.14). The process of calculating a DoG scale space from a discrete Gaussian scale space is illustrated in Fig. 25.7, using the same parameters as in Table 25.2 and Fig. 25.6.

25.1.4 Hierarchical Scale Space

Despite the fact that 2D Gaussian filter kernels are separable into 1D kernels,[13] the size of the required filter grows quickly with increasing scale, regardless if a direct or recursive approach is used (as shown in Table 25.2). However, each Gaussian filter operation reduces the bandwidth of the signal inversely proportional to the width of the kernel (see Sec. E.3 in the Appendix). If the image size is kept constant over all scales, the images become increasingly oversampled at higher scale levels. In other words, the sampling rate in a Gaussian scale space can be reduced with increasing scale without losing relevant signal information.

[13] See also Chapter 5, Sec. 5.3.3.

Octaves and sub-sampling (decimation)

In particular, doubling the scale cuts the bandwidth by half, that is, the signal at scale level 2σ has only half the bandwidth of the signal at level σ. An image signal at scale level 2σ of a Gaussian scale space thus shows only half the bandwidth of the same image at scale level σ. In a Gaussian scale space representation it is thus safe to down-sample the image to half the sample rate after each octave without any loss of information. This suggests a very efficient, "pyramid-like" approach for constructing a DoG scale space, as illustrated in Fig. 25.8.[14]

At the start (bottom) of each octave, the image is down-sampled to half the resolution, that is, each pixel in the new octave covers twice the distance of the pixels in the previous octave in every spatial direction. Within each octave, the same small Gaussian kernels can be used for successive filtering, since their relative widths (with respect to the original sampling lattice) also implicitly double at each octave. To describe these relations formally, we use

$$\mathbf{G} = (\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_{P-1}) \tag{25.32}$$

to denote a *hierarchical Gaussian scale space* consisting of P octaves. Each octave

$$\mathbf{G}_p = \left(\mathbf{G}_{p,0}, \mathbf{G}_{p,1}, \ldots, \mathbf{G}_{p,Q}\right), \tag{25.33}$$

consists of $Q+1$ scale levels $\mathbf{G}_{p,q}$, where $p \in [0, P-1]$ is the octave index and $q \in [0, Q]$ is the level index within the containing octave $\mathbf{G}_p$. With respect to *absolute* scale, a level $\mathbf{G}_{p,q} = \mathbf{G}_p(q)$ in the hierarchical Gaussian scale space corresponds to the level G_m in the non-hierarchical Gaussian scale space (see Eqn. (25.20)) with index

$$m = Q \cdot p + q. \tag{25.34}$$

As follows from Eqn. (25.22), the *absolute scale* at level $\mathbf{G}_{p,q}$ then is

$$\begin{aligned}\sigma_{p,q} &= \sigma_m = \sigma_0 \cdot \Delta_\sigma^m = \sigma_0 \cdot 2^{m/Q} \\ &= \sigma_0 \cdot 2^{(Qp+q)/Q} = \sigma_0 \cdot 2^{p+q/Q},\end{aligned} \tag{25.35}$$

where $\sigma_0 = \sigma_{0,0}$ denotes the predefined base scale offset (e.g., $\sigma_0 = 1.6$ in Table 25.2). In particular, the absolute scale of the base level $\mathbf{G}_{p,0}$ of *any* octave $\mathbf{G}_p$ is

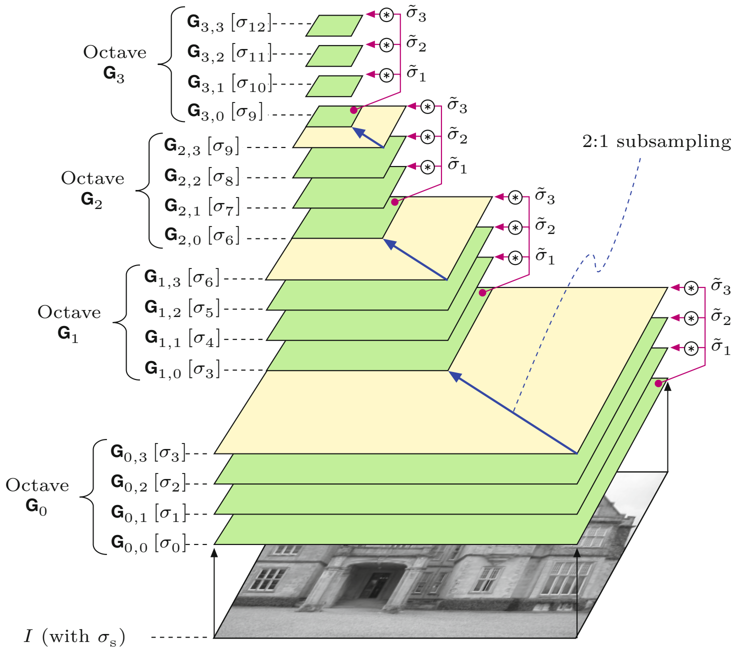
$$\sigma_{p,0} = \sigma_0 \cdot 2^p. \tag{25.36}$$

The **decimated scale** $\dot{\sigma}_{p,q}$ is the absolute scale $\sigma_{p,q}$ (Eqn. (25.35)) expressed in the coordinate units of octave $\mathbf{G}_p$, that is,

$$\dot{\sigma}_{p,q} = \dot{\sigma}_q = \sigma_{p,q} \cdot 2^{-p} = \sigma_0 \cdot 2^{p+q/Q} \cdot 2^{-p} = \sigma_0 \cdot 2^{q/Q}. \tag{25.37}$$

Note that the decimated scale $\dot{\sigma}_{p,q}$ is independent of the octave index p and therefore $\dot{\sigma}_{p,q} \equiv \dot{\sigma}_q$, for any level index q.

[14] Successive reduction of image resolution by sub-sampling is the core concept of "image pyramid" methods [41].



Octave $\mathbf{G}_3$
$\mathbf{G}_{3,3}$ $[\sigma_{12}]$
$\mathbf{G}_{3,2}$ $[\sigma_{11}]$
$\mathbf{G}_{3,1}$ $[\sigma_{10}]$
$\mathbf{G}_{3,0}$ $[\sigma_9]$
Octave $\mathbf{G}_2$
$\mathbf{G}_{2,3}$ $[\sigma_9]$
$\mathbf{G}_{2,2}$ $[\sigma_8]$
$\mathbf{G}_{2,1}$ $[\sigma_7]$
$\mathbf{G}_{2,0}$ $[\sigma_6]$
Octave $\mathbf{G}_1$
$\mathbf{G}_{1,3}$ $[\sigma_6]$
$\mathbf{G}_{1,2}$ $[\sigma_5]$
$\mathbf{G}_{1,1}$ $[\sigma_4]$
$\mathbf{G}_{1,0}$ $[\sigma_3]$
Octave $\mathbf{G}_0$
$\mathbf{G}_{0,3}$ $[\sigma_3]$
$\mathbf{G}_{0,2}$ $[\sigma_2]$
$\mathbf{G}_{0,1}$ $[\sigma_1]$
$\mathbf{G}_{0,0}$ $[\sigma_0]$
I (with σ_s)
2:1 subsampling
$\tilde{\sigma}_3$
$\tilde{\sigma}_2$
$\tilde{\sigma}_1$

Fig. 25.8
Hierarchical Gaussian scale space. Each octave extends over $Q = 3$ scale steps. The base level $\mathbf{G}_{p,0}$ of each octave $p > 0$ is obtained by 2:1 sub-sampling of the top level $\mathbf{G}_{p-1,3}$ of the next-lower octave. At the transition between octaves, the resolution (image size) is cut in half in the x- and y-direction. The absolute scale at octave level $\mathbf{G}_{p,q}$ is σ_m, with $m = Qp + q$. Within each octave, the same set of Gaussian kernels ($\tilde{\sigma}_1$, $\tilde{\sigma}_2$, $\tilde{\sigma}_3$) is used to calculate the following levels from the octave's base level $\mathbf{G}_{p,0}$.

From the octave's base level $\mathbf{G}_{p,0}$, the subsequent levels in the same octave can be calculated by filtering with relatively small Gaussian kernels. The size of the kernel needed to calculate scale-level $\mathbf{G}_{p,q}$ from the octave's base level $\mathbf{G}_{p,0}$ is obtained from the corresponding decimated scales (Eqn. (25.37)) as

$$\tilde{\sigma}_{p,q} = \sqrt{\dot{\sigma}_{p,q}^2 - \dot{\sigma}_{p,0}^2} = \sqrt{(\sigma_0 \cdot 2^{q/Q})^2 - \sigma_0^2} = \sigma_0 \cdot \sqrt{2^{2q/Q} - 1}\,, \tag{25.38}$$

for $q \geq 0$. Note that $\tilde{\sigma}_q$ is independent of the octave index p and thus the *same* filter kernels can be used at each octave. For example, with $Q = 3$ and $\sigma_0 = 1.6$ (as used in Table 25.2) the resulting kernel widths are

$$\tilde{\sigma}_1 = 1.2263, \qquad \tilde{\sigma}_2 = 1.9725, \qquad \tilde{\sigma}_3 = 2.7713. \tag{25.39}$$

Also note that, instead of filtering all scale levels $\mathbf{G}_{p,q}$ in an octave from the corresponding base level $\mathbf{G}_{p,0}$, we could calculate them recursively from the next-lower level $\mathbf{G}_{p,q-1}$. While this approach requires even smaller Gaussian kernels (and is thus more efficient), recursive filtering tends to accrue numerical inaccuracies. Nevertheless, the method is used frequently in scale-space implementations.

Decimation between successive octaves

With $M \times N$ being the size of the original image I, every sub-sampling step between octaves cuts the size of the image by half, that is,

$$M_{p+1} \times N_{p+1} = \left\lfloor \frac{M_p}{2} \right\rfloor \times \left\lfloor \frac{N_p}{2} \right\rfloor, \tag{25.40}$$

for octaves with index $p \geq 0$. The resulting image size at octave $\mathbf{G}_p$ is thus

$$M_p \times N_p = \left\lfloor \frac{M_0}{2^p} \right\rfloor \times \left\lfloor \frac{N_0}{2^p} \right\rfloor. \tag{25.41}$$

The base level $\mathbf{G}_{p,0}$ of each octave $\mathbf{G}_p$ (with $p > 0$) is obtained by sub-sampling the top level $\mathbf{G}_{p-1,Q}$ of the next-lower octave $\mathbf{G}_{p-1}$ as

$$\mathbf{G}_{p,0} = \mathsf{Decimate}(\mathbf{G}_{p-1,Q}), \tag{25.42}$$

where $\mathsf{Decimate}(G)$ denotes the 2:1 sub-sampling operation, that is,

$$\mathbf{G}_{p,0}(u,v) \leftarrow \mathbf{G}_{p-1,Q}(2u, 2v), \tag{25.43}$$

for each sample position $(u,v) \in [0, M_p - 1] \times [0, N_p - 1]$. Additional low-pass filtering is not required prior to sub-sampling since the Gaussian smoothing performed in each octave also cuts the bandwidth by half.

The main steps involved in constructing a hierarchical Gaussian scale space are summarized in Alg. 25.1. In summary, the input image I is first blurred to scale σ_0 by filtering with a Gaussian kernel of width $\bar{\sigma}_0$. Within each octave $\mathbf{G}_p$, the scale levels $\mathbf{G}_{p,q}$ are calculated from the base level $\mathbf{G}_{p,0}$ by filtering with a set of Gaussian filters of width $\tilde{\sigma}_q$ $(q = 1, \ldots, Q)$. Note that the values $\tilde{\sigma}_q$ and the corresponding Gaussian kernels $H^{\mathrm{G},\tilde{\sigma}_q}$ can be pre-calculated once since they are independent of the octave index p (Alg. 25.1, lines 13–14). The base level $\mathbf{G}_{p,0}$ of each higher octave $\mathbf{G}_p$ is obtained by decimating the top level $\mathbf{G}_{p-1,Q}$ of the previous octave $\mathbf{G}_{p-1}$. Typical parameter values are $\sigma_\mathrm{s} = 0.5$, $\sigma_0 = 1.6$, $Q = 3$, $P = 4$.

Spatial positions in the hierarchical scale space

To properly associate the spatial positions of features detected in different octaves of the hierarchical scale space we define the function

$$\boldsymbol{x}_0 \leftarrow \mathsf{AbsPos}(\boldsymbol{x}_p, p),$$

that maps the continuous position $\boldsymbol{x}_p = (x_p, y_p)$ in the local coordinate system of octave p to the corresponding position $\boldsymbol{x} = (x, y)$ in the coordinate system of the original full-resolution image I (octave $p = 0$). The function AbsPos can be defined recursively by relating the positions in successive octaves as

$$\mathsf{AbsPos}(\boldsymbol{x}_p, p) = \begin{cases} \boldsymbol{x}_p & \text{for } p = 0, \\ \mathsf{AbsPos}(2 \cdot \boldsymbol{x}_p, p-1) & \text{for } p > 0, \end{cases} \tag{25.44}$$

which gives $\boldsymbol{x}_0 = \mathsf{AbsPos}(2^p \cdot \boldsymbol{x}_p, 0)$ and thus

$$\mathsf{AbsPos}(\boldsymbol{x}_p, p) = 2^p \cdot \boldsymbol{x}_p. \tag{25.45}$$

Hierarchical LoG/DoG scale space

Analogous to the scheme shown in Fig. 25.7, a *hierarchical* DoG scale space representation is obtained by calculating the difference of adjacent scale levels within each octave of the hierarchical Gaussian scale space, that is,

Alg. 25.1
Building a hierarchical Gaussian scale space. The input image I is first blurred to scale σ_0 by filtering with a Gaussian kernel of width $\bar{\sigma}_0$ (line 3). In each octave $\mathbf{G}_p$, the scale levels $\mathbf{G}_{p,q}$ are calculated from the base level $\mathbf{G}_{p,0}$ by filtering with a set of Gaussian filters of width $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_Q$ (line 13–14). The base level $\mathbf{G}_{p,0}$ of each higher octave is obtained by subsampling the top level $\mathbf{G}_{p-1,Q}$ of the previous octave (line 6).

```
1:  BuildGaussianScaleSpace($I, \sigma_s, \sigma_0, P, Q$)
        Input: $I$, source image; $\sigma_s$, sampling scale; $\sigma_0$, reference scale of the first octave; $P$, number of octaves. $Q$, number of scale steps per octave. Returns a hierarchical Gaussian scale space representation $\mathbf{G}$ of the image $I$.
2:      $\bar{\sigma}_0 \leftarrow (\sigma_0^2 - \sigma_s^2)^{1/2}$          ▷ scale to base of 1st octave, Eq. 25.25
3:      $\mathsf{G}_{\mathrm{init}} \leftarrow I * H^{\mathrm{G},\bar{\sigma}_0}$          ▷ apply 2D Gaussian filter of width $\bar{\sigma}_0$
4:      $\mathbf{G}_0 \leftarrow$ MakeGaussianOctave($\mathsf{G}_{\mathrm{init}}, 0, Q, \sigma_0$)          ▷ create octave $\mathbf{G}_0$
5:      for $p \leftarrow 1, \ldots, P-1$ do          ▷ octave index $p$
6:          $\mathsf{G}_{\mathrm{next}} \leftarrow$ Decimate($\mathbf{G}_{p-1,Q}$)          ▷ dec. top level of octave $p-1$
7:          $\mathbf{G}_p \leftarrow$ MakeGaussianOctave($\mathsf{G}_{\mathrm{next}}, p, Q, \sigma_0$)          ▷ create octave $\mathbf{G}_p$
8:      $\mathbf{G} \leftarrow (\mathbf{G}_0, \ldots, \mathbf{G}_{P-1})$
9:      return $\mathbf{G}$          ▷ hierarchical Gaussian scale space $\mathbf{G}$

10: MakeGaussianOctave($\mathsf{G}_{\mathrm{base}}, p, Q, \sigma_0$)
        Input: $\mathsf{G}_{\mathrm{base}}$, octave base level; $p$, octave index; $Q$, number of levels per octave; $\sigma_0$, reference scale.
11:     $\mathbf{G}_{p,0} \leftarrow \mathsf{G}_{\mathrm{base}}$
12:     for $q \leftarrow 1, \ldots, Q$ do          ▷ level index $q$
13:         $\tilde{\sigma}_q \leftarrow \sigma_0 \cdot \sqrt{2^{2q/Q} - 1}$          ▷ see Eq. 25.38
14:         $\mathbf{G}_{p,q} \leftarrow \mathsf{G}_{\mathrm{base}} * H^{\mathrm{G},\tilde{\sigma}_q}$          ▷ apply 2D Gaussian filter of width $\tilde{\sigma}_q$
15:     $\mathbf{G}_p \leftarrow (\mathbf{G}_{p,0}, \ldots, \mathbf{G}_{p,Q})$
16:     return $\mathbf{G}_p$          ▷ scale space octave $\mathbf{G}_p$

17: Decimate($\mathsf{G}_{\mathrm{in}}$)
        Input: $\mathsf{G}_{\mathrm{in}}$, Gaussian scale space level.
18:     $(M, N) \leftarrow$ Size($\mathsf{G}_{\mathrm{in}}$)
19:     $M' \leftarrow \lfloor \frac{M}{2} \rfloor, \quad N' \leftarrow \lfloor \frac{N}{2} \rfloor$          ▷ decimated size
20:     Create map $\mathsf{G}_{\mathrm{out}} : M' \times N' \mapsto \mathbb{R}$
21:     for all $(u, v) \in M' \times N'$ do
22:         $\mathsf{G}_{\mathrm{out}}(u, v) \leftarrow \mathsf{G}_{\mathrm{in}}(2u, 2v)$          ▷ 2:1 subsampling
23:     return $\mathsf{G}_{\mathrm{out}}$          ▷ decimated scale level $\mathsf{G}_{\mathrm{out}}$
```

$$\mathbf{D}_{p,q} = \mathbf{G}_{p,q+1} - \mathbf{G}_{p,q} \qquad (25.46)$$

for level numbers $q \in [0, Q-1]$. Figure 25.9 shows the corresponding Gaussian and DoG scale levels for the previous example over a range of three octaves. To demonstrate the effects of sub-sampling, the same information is shown in Fig. 25.10 and 25.11, with all level images scaled to the same size. Figure 25.11 also shows the absolute values of the DoG response, which are effectively used for detecting interest points at different scale levels. Note how blob-like features stand out and disappear again as the scale varies from fine to coarse. Analogous results obtained from a different image are shown in Figs. 25.12 and 25.13.

25.1.5 Scale Space Structure in SIFT

In the SIFT approach, the absolute value of the DoG response is used to localize interest points at different scales. For this purpose, local maxima are detected in the 3D space spanned by the spatial x/y-positions and the scale coordinate. To determine local maxima along the scale dimension over a full octave, two additional DoG levels,

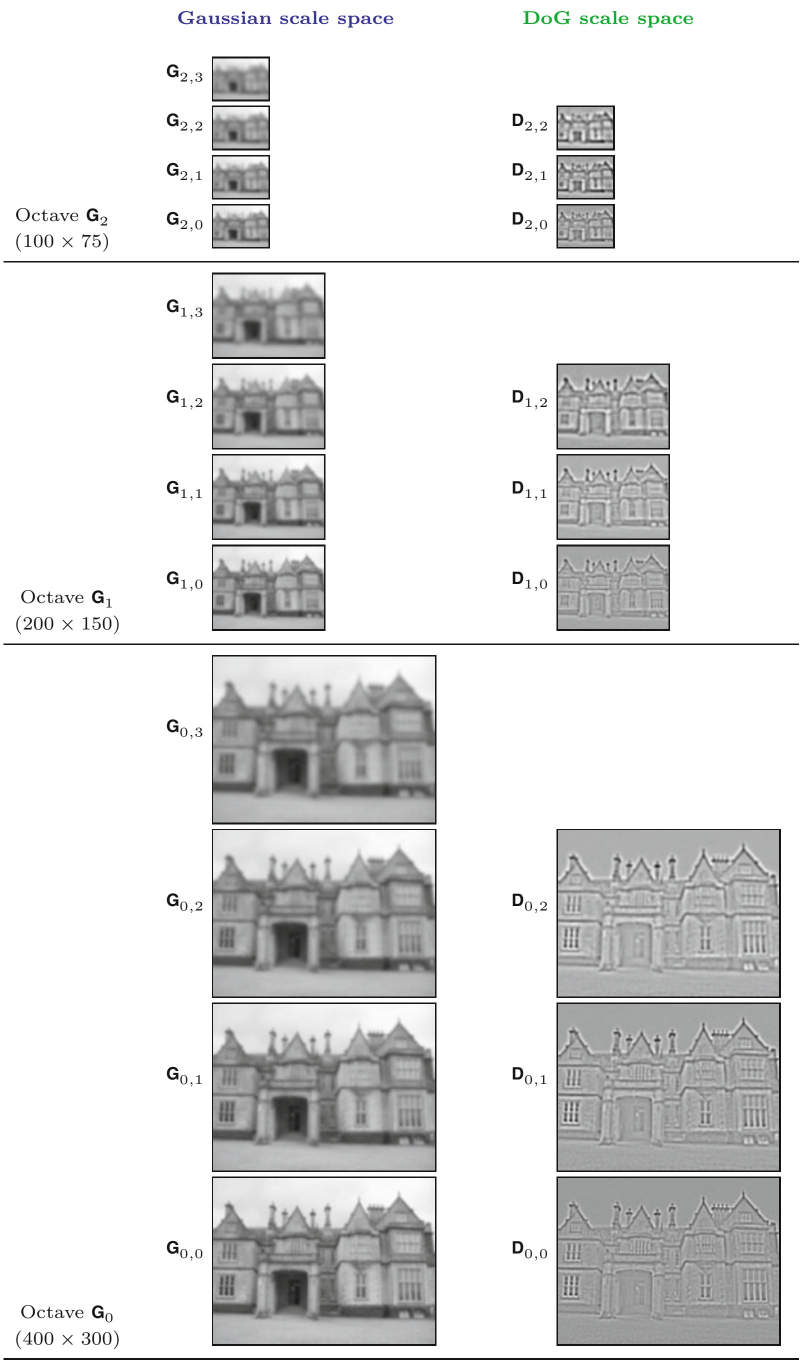


Fig. 25.9
Hierarchical Gaussian and DoG scale space example, with $P = Q = 3$. Gaussian scale space levels $\mathbf{G}_{p,q}$ are shown in the left column, DoG levels $\mathbf{D}_{p,q}$ in the right column. All images are shown at their real scale.

$\mathbf{D}_{p,-1}$ and $\mathbf{D}_{p,Q}$, and two additional Gaussian scale levels, $\mathbf{G}_{p,-1}$ and $\mathbf{G}_{p,Q+1}$, are required in each octave.

In total, each octave $\mathbf{G}_p$ then consists of $Q+3$ Gaussian scale levels $\mathbf{G}_{p,q}$ $(q = -1, \ldots, Q+1)$ and $Q+2$ DoG levels $\mathbf{D}_{p,q}$ $(q = -1, \ldots, Q)$, as shown in Fig. 25.14. For the base level $\mathbf{G}_{0,-1}$, the scale index is $m = -1$ and its absolute scale (see Eqns. (25.22) and (25.35)) is

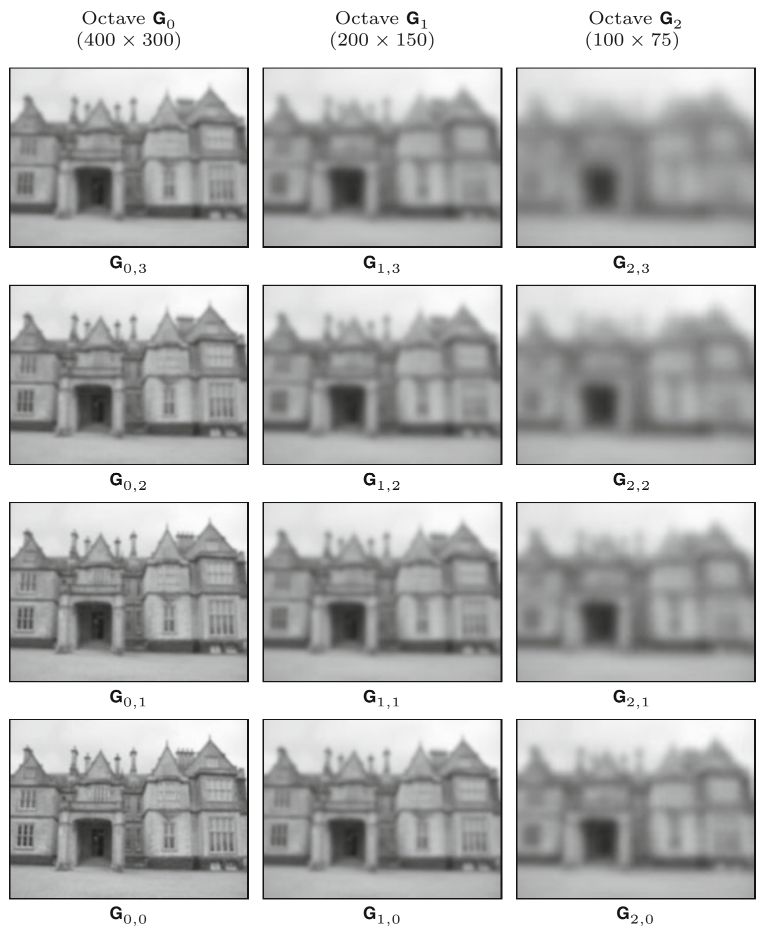


Fig. 25.10
Hierarchical Gaussian scale space example (`castle` image). All images are scaled to the same size. Note that $\mathbf{G}_{1,0}$ is merely a sub-sampled copy of $\mathbf{G}_{0,3}$; analogously, $\mathbf{G}_{2,0}$ is sub-sampled from $\mathbf{G}_{1,3}$.

$$\sigma_{0,-1} = \sigma_0 \cdot 2^{-1/Q} = \sigma_0 \cdot \frac{1}{\Delta_\sigma}. \tag{25.47}$$

Thus, with the usual settings ($\sigma_0 = 1.6$ and $Q = 3$), the *absolute* scale values for the six levels of the first octave are

$$\begin{aligned} \sigma_{0,-1} &= 1.2699, & \sigma_{0,0} &= 1.6000, & \sigma_{0,1} &= 2.0159, \\ \sigma_{0,2} &= 2.5398, & \sigma_{0,3} &= 3.2000, & \sigma_{0,4} &= 4.0317. \end{aligned} \tag{25.48}$$

The complete set of scale values for a SIFT scale space with four octaves ($p = 0, \ldots, 3$) is listed in Table 25.3.

To construct the Gaussian part of the first scale space octave $\mathbf{G}_0$, the initial level $\mathbf{G}_{0,-1}$ is obtained by filtering the input image I with a Gaussian kernel of width

$$\bar{\sigma}_{0,-1} = \sqrt{\sigma_{0,-1}^2 - \sigma_s^2} = \sqrt{1.2699^2 - 0.5^2} \approx 1.1673 \tag{25.49}$$

For the higher octaves ($p > 0$), the initial level ($q = -1$) is obtained by sub-sampling (decimating) level $Q-1$ of the next-lower octave $\mathbf{G}_{p-1}$, that is,

$$\mathbf{G}_{p,-1} \leftarrow \mathsf{Decimate}(\mathbf{G}_{p-1,Q-1}), \tag{25.50}$$

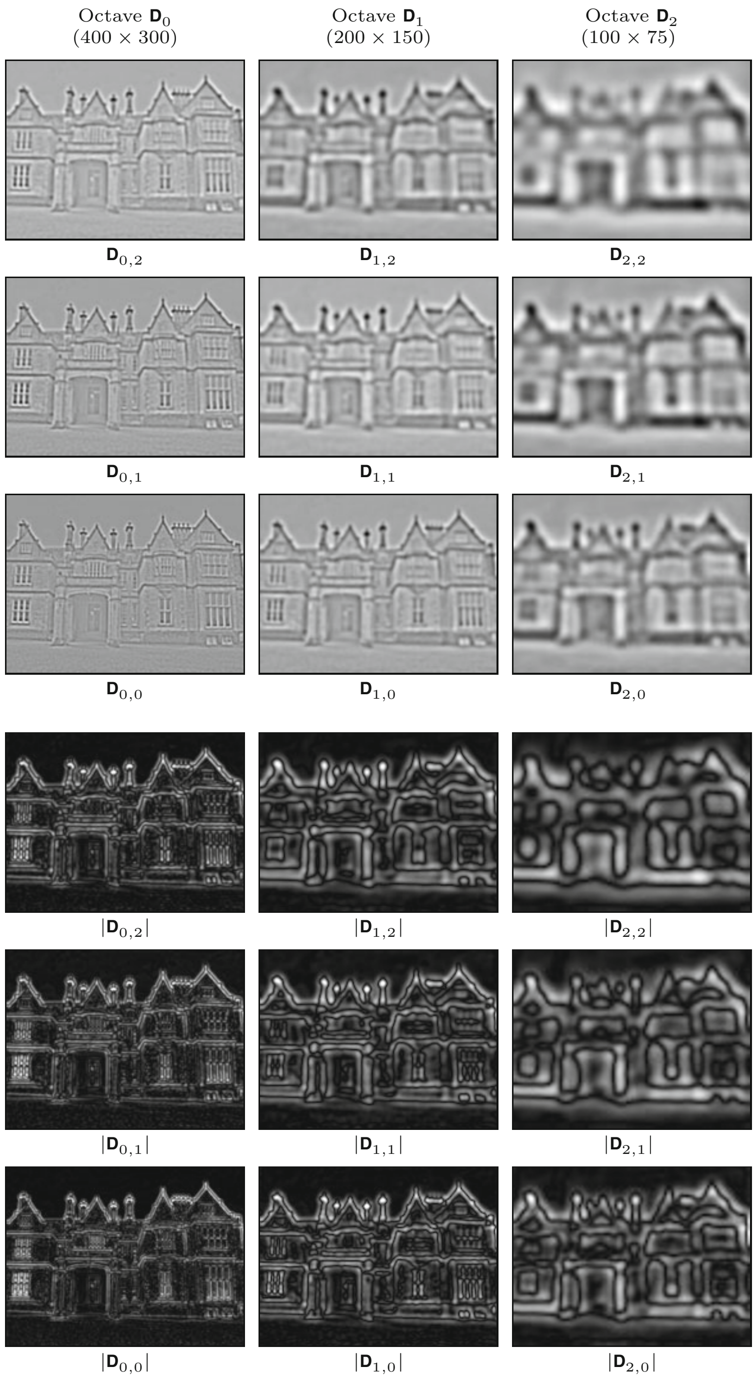


Fig. 25.11
Hierarchical DoG scale space example (`castle` image). The three top rows show the positive and negative DoG values (zero is mapped to intermediate gray). The three bottom rows show the absolute values of the DoG results (zero is mapped to black, maximum values to white). All images are scaled to the size of the original image.

analogous to Eqn. (25.42). The remaining levels $\mathbf{G}_{p,0}, \ldots, \mathbf{G}_{p,Q+1}$ of the octave are either calculated by incremental filtering (as described in Fig. 25.6) or by filtering from the octave's initial level $\mathbf{G}_{p,-1}$ with a Gaussian of width $\tilde{\sigma}_{p,q}$ (see Eqn. (25.38)). The advantage of the direct approach is that numerical errors do not accrue across the scale space; the disadvantage is that the kernels are up to 50 % larger than those needed for the incremental approach ($\tilde{\sigma}_{0,4} = 3.8265$ vs.

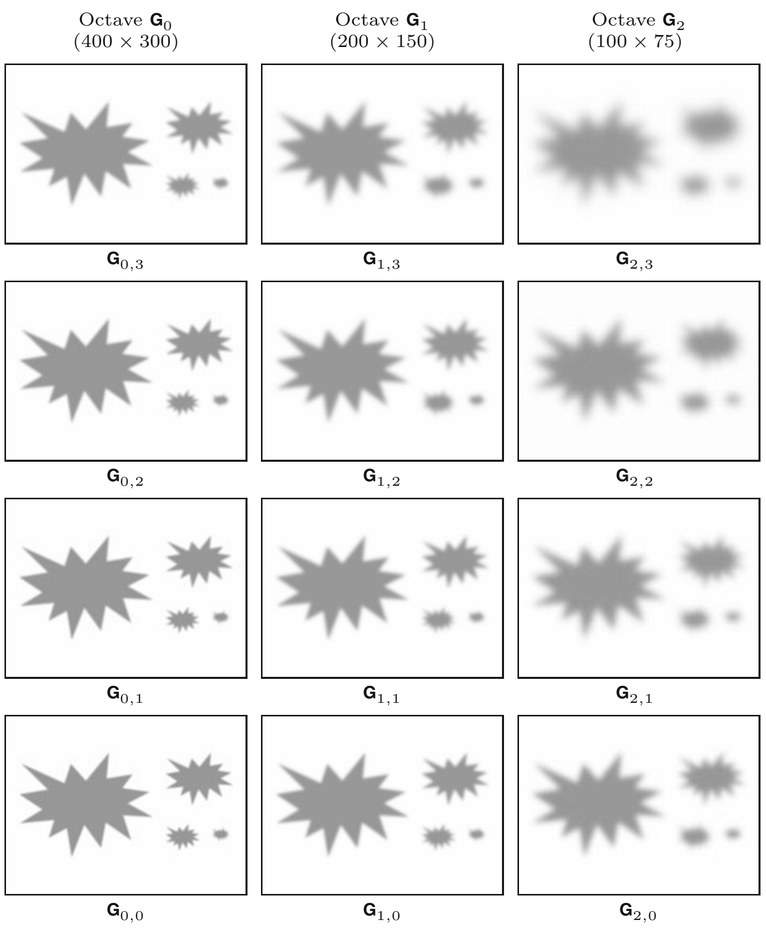


Fig. 25.12
Hierarchical Gaussian scale space example (`stars` image).

Table 25.3
Absolute and relative scale values for a SIFT scale space with four octaves. Each octave with index $p = 0, \ldots, 3$ consists of 6 Gaussian scale layers $\mathbf{G}_{p,q}$, with $q = -1, \ldots, 4$. For each scale layer, m is the scale index and $\sigma_{p,q}$ is the corresponding *absolute* scale. Within each octave p, $\tilde{\sigma}_{p,q}$ denotes the *relative* scale with respect to the octave's base layer $\mathbf{G}_{p,-1}$. Each base layer $\mathbf{G}_{p,-1}$ is obtained by sub-sampling (decimating) layer $q = Q-1 = 2$ in the previous octave, i.e., $\mathbf{G}_{p,-1} = \mathsf{Decimate}(\mathbf{G}_{p-1,Q-1})$, for $p > 0$. The base layer $\mathbf{G}_{0,-1}$ in the bottom octave is derived by Gaussian smoothing of the original image. Note that the relative scale values $\tilde{\sigma}_{p,q} = \tilde{\sigma}_q$ are the same inside every octave (independent of p) and thus the same Gaussian filter kernels can be used for calculating all octaves.

p	q	m	d	$\sigma_{p,q}$	$\dot{\sigma}_q$	$\tilde{\sigma}_q$
3	4	13	8	32.2540	4.0317	3.8265
3	3	12	8	25.6000	3.2000	2.9372
3	2	11	8	20.3187	2.5398	2.1996
3	1	10	8	16.1270	2.0159	1.5656
3	0	9	8	12.8000	1.6000	0.9733
3	−1	8	8	10.1594	1.2699	0.0000
2	4	10	4	16.1270	4.0317	3.8265
2	3	9	4	12.8000	3.2000	2.9372
2	2	8	4	10.1594	2.5398	2.1996
2	1	7	4	8.0635	2.0159	1.5656
2	0	6	4	6.4000	1.6000	0.9733
2	−1	5	4	5.0797	1.2699	0.0000
1	4	7	2	8.0635	4.0317	3.8265
1	3	6	2	6.4000	3.2000	2.9372
1	2	5	2	5.0797	2.5398	2.1996
1	1	4	2	4.0317	2.0159	1.5656
1	0	3	2	3.2000	1.6000	0.9733
1	−1	2	2	2.5398	1.2699	0.0000
0	4	4	1	4.0317	4.0317	3.8265
0	3	3	1	3.2000	3.2000	2.9372
0	2	2	1	2.5398	2.5398	2.1996
0	1	1	1	2.0159	2.0159	1.5656
0	0	0	1	1.6000	1.6000	0.9733
0	−1	−1	1	1.2699	1.2699	0.0000

p …	octave index
q …	level index
m …	linear scale index ($m = Qp + q$)
d …	decimation factor ($d = 2^p$)
$\sigma_{p,q}$ …	absolute scale (Eqn. (25.35))
$\dot{\sigma}_q$ …	decimated scale (Eqn. (25.37))
$\tilde{\sigma}_q$ …	relative decimated scale w.r.t. octave's base level $\mathbf{G}_{p,-1}$ (Eqn. (25.38))

$P = 3$ (number of octaves)

$Q = 3$ (levels per octave)

$\sigma_0 = 1.6$ (base scale)

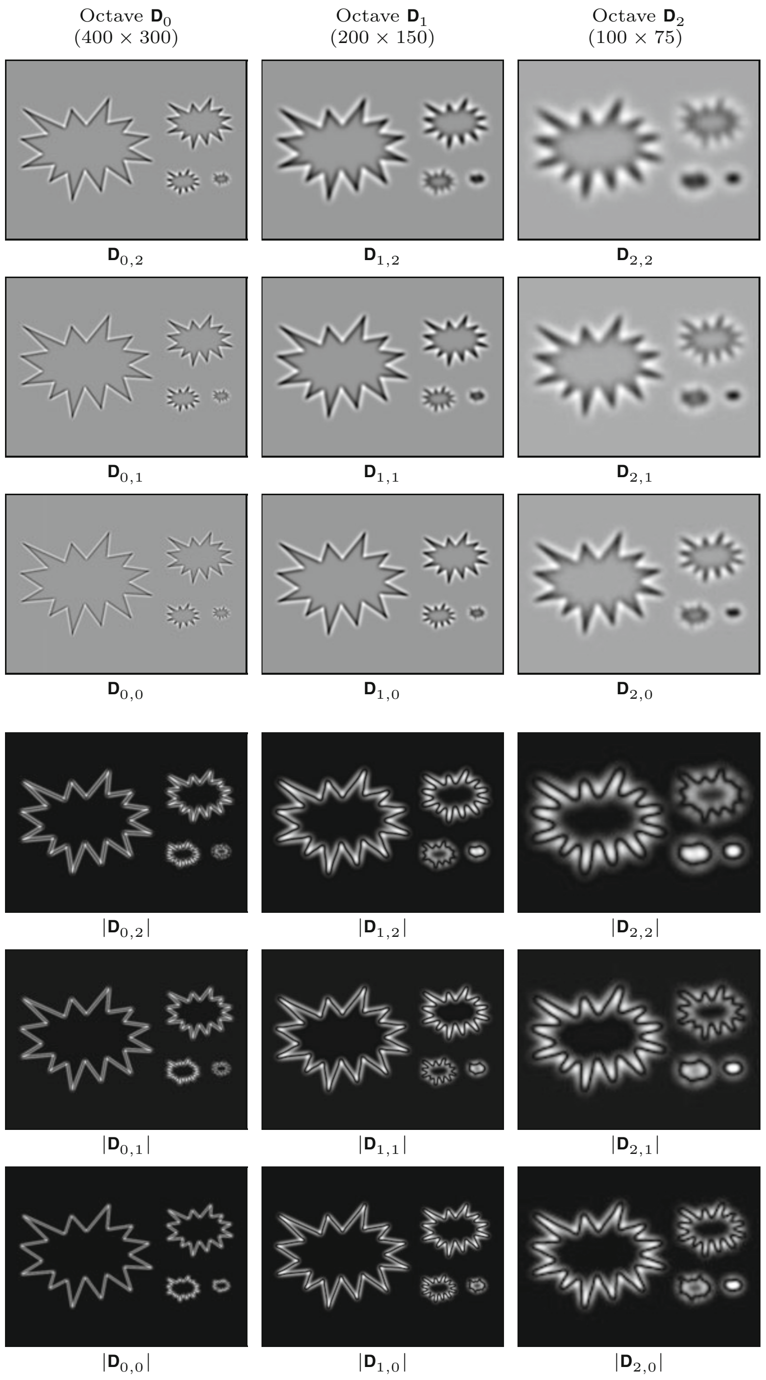
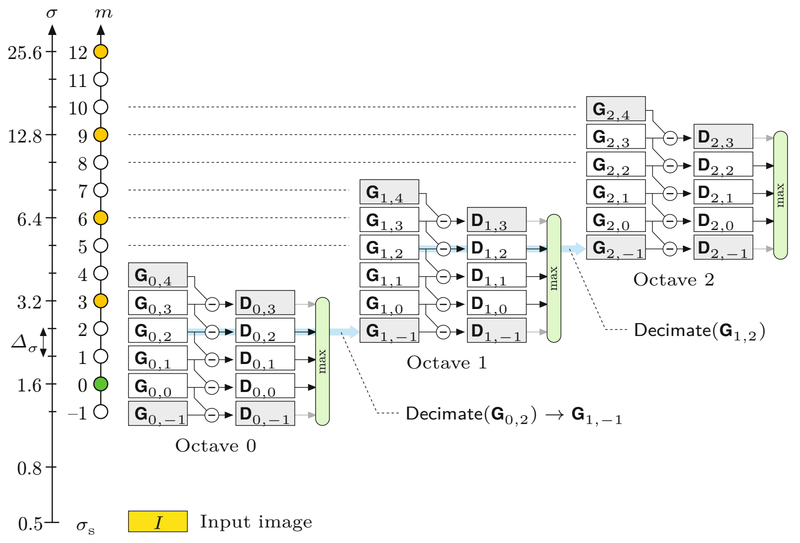


Fig. 25.13
Hierarchical DoG scale space example (`stars` image). The three top rows show the positive and negative DoG values (zero is mapped to intermediate gray). The three bottom rows show the absolute values of the DoG results (zero is mapped to black, maximum values to white). All images are scaled to the size of the original image.

$\sigma'_{0,4} = 2.4525$). Note that the inner levels $\mathbf{G}_{p,q}$ of all higher octaves (i.e., $p > 0, q \geq 0$) are calculated from the base level $\mathbf{G}_{p,-1}$, using the *same* set of kernels as for the first octave, as listed in Table 25.3. The complete process of building a SIFT scale space is summarized in Alg. 25.2.



σ
m
25.6
12.8
6.4
3.2
1.6
0.8
0.5
Δ_σ
σ_s
12 11 10 9 8 7 6 5 4 3 2 1 0 −1
$G_{0,4}$ $G_{0,3}$ $G_{0,2}$ $G_{0,1}$ $G_{0,0}$ $G_{0,-1}$
$D_{0,3}$ $D_{0,2}$ $D_{0,1}$ $D_{0,0}$ $D_{0,-1}$
Octave 0
$G_{1,4}$ $G_{1,3}$ $G_{1,2}$ $G_{1,1}$ $G_{1,0}$ $G_{1,-1}$
$D_{1,3}$ $D_{1,2}$ $D_{1,1}$ $D_{1,0}$ $D_{1,-1}$
Octave 1
$G_{2,4}$ $G_{2,3}$ $G_{2,2}$ $G_{2,1}$ $G_{2,0}$ $G_{2,-1}$
$D_{2,3}$ $D_{2,2}$ $D_{2,1}$ $D_{2,0}$ $D_{2,-1}$
Octave 2
max
Decimate($G_{0,2}$) → $G_{1,-1}$
Decimate($G_{1,2}$)
I
Input image

Fig. 25.14 Scale space structure for SIFT with $P = 3$ octaves and $Q = 3$ levels per octave. To perform local maximum detection ("max") over the full octave, $Q+2$ DoG scale space levels $(\mathsf{D}_{p,-1}, \ldots, \mathsf{D}_{p,Q})$ are required. The blue arrows indicate the decimation steps between successive Gaussian octaves. Since the DoG levels are obtained by subtracting pairs of Gaussian scale space levels, $Q+3$ such levels $(\mathsf{G}_{p,-1}, \ldots, \mathsf{G}_{p,Q+1})$ are needed in each octave G_p. The two vertical axes on the left show the absolute scale (σ) and the discrete scale index (m), respectively. Note that the values along the scale axis are logarithmic with constant multiplicative scale increments $\Delta_\sigma = 2^{1/Q}$. The absolute scale of the input image (I) is assumed as $\sigma_s = 0.5$.

25.2 Key Point Selection and Refinement

Key points are identified in three steps: (1) detection of extremal points in the DOG scale space, (2) position refinement by local interpolation, and (3) elimination of edge responses. These steps are detailed in the following and summarized in Algs. 25.3–25.6.

25.2.1 Local Extrema Detection

In the first step, candidate interest points are detected as local extrema in the 3D DoG scale space that we described in the previous section. Extrema detection is performed independently within each octave p. For the sake of convenience we define the 3D scale space coordinate $\boldsymbol{c} = (u, v, q)$, composed of the spatial position (u, v) and the level index q, as well as the function

$$D(\boldsymbol{c}) := \mathsf{D}_{p,q+k}(u, v) \tag{25.51}$$

as a short notation for selecting DoG values from a given octave p. Also, for collecting the DoG values in the 3D neighborhood around a scale space position $\boldsymbol{c}$, we define the map

$$\mathsf{N}_{\boldsymbol{c}}(i, j, k) := D\big(\boldsymbol{c} + i \cdot \mathbf{e}_\mathrm{i} + j \cdot \mathbf{e}_\mathrm{j} + k \cdot \mathbf{e}_\mathrm{k}\big), \tag{25.52}$$

with $i, j, k \in \{-1, 0, 1\}$ and the 3D unit vectors

$$\mathbf{e}_\mathrm{i} = (1,0,0)^\mathsf{T}, \qquad \mathbf{e}_\mathrm{j} = (0,1,0)^\mathsf{T}, \qquad \mathbf{e}_\mathrm{k} = (0,0,1)^\mathsf{T}. \tag{25.53}$$

The neighborhood $\mathsf{N}_{\boldsymbol{c}}$ includes the center value $D(\boldsymbol{c})$ and the 26 values of its immediate neighbors (see Fig. 25.15(a)). These values are used to estimate the 3D gradient vector and the Hessian matrix for the 3D scale space position $\boldsymbol{c}$, as will be described.

A DoG scale space position $\boldsymbol{c}$ is accepted as a local extremum (minimum or maximum) if the associated value $D(\boldsymbol{c}) = \mathsf{N}_{\boldsymbol{c}}(0, 0, 0)$

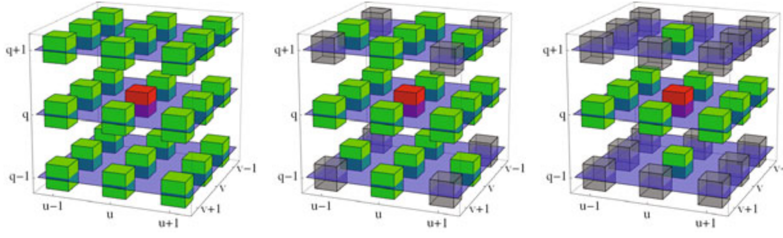
```
1:  BuildSiftScaleSpace($I, \sigma_s, \sigma_0, P, Q$)
        Input: $I$, source image; $\sigma_s$, sampling scale; $\sigma_0$, reference scale of
        the first octave; $P$, number of octaves; $Q$, number of scale steps
        per octave. Returns a SIFT scale space representation $\langle \mathbf{G}, \mathbf{D} \rangle$ of
        the image $I$.
2:      $\sigma_{\mathrm{init}} \leftarrow \sigma_0 \cdot 2^{-1/Q}$            ▷ abs. scale at level $(0,-1)$, Eq. 25.47
3:      $\bar{\sigma}_{\mathrm{init}} \leftarrow \sqrt{\sigma_{\mathrm{init}}^2 - \sigma_s^2}$         ▷ relative scale w.r.t. $\sigma_s$, Eq. 25.49
4:      $\mathsf{G}_{\mathrm{init}} \leftarrow I * H^{\mathrm{G},\bar{\sigma}_{\mathrm{init}}}$          ▷ 2D Gaussian filter with $\bar{\sigma}_{\mathrm{init}}$
5:      $\mathbf{G}_0 \leftarrow \mathsf{MakeGaussianOctave}(\mathsf{G}_{\mathrm{init}}, 0, Q, \sigma_0)$       ▷ Gauss. octave 0
6:      for $p \leftarrow 1, \ldots, P-1$ do                      ▷ for octaves $1, \ldots, P-1$
7:          $\mathsf{G}_{\mathrm{next}} \leftarrow \mathsf{Decimate}(\mathbf{G}_{p-1,Q-1})$               ▷ see Alg. 25.1
8:          $\mathbf{G}_p \leftarrow \mathsf{MakeGaussianOctave}(\mathsf{G}_{\mathrm{next}}, p, Q, \sigma_0)$       ▷ octave $p$
9:      $\mathbf{G} \leftarrow (\mathbf{G}_0, \ldots, \mathbf{G}_{P-1})$          ▷ assemble the Gaussian scale space $\mathbf{G}$
10:     for $p \leftarrow 0, \ldots, P-1$ do
11:         $\mathbf{D}_p \leftarrow \mathsf{MakeDogOctave}(\mathbf{G}_p, p, Q)$
12:     $\mathbf{D} \leftarrow (\mathbf{D}_0, \ldots, \mathbf{D}_{P-1})$              ▷ assemble the DoG scale space $\mathbf{D}$
13:     return $\langle \mathbf{G}, \mathbf{D} \rangle$

14: MakeGaussianOctave($\mathsf{G}_{\mathrm{base}}, p, Q, \sigma_0$)
        Input: $\mathsf{G}_{\mathrm{base}}$, Gaussian base level; $p$, octave index; $Q$, scale steps
        per octave, $\sigma_0$, reference scale. Returns a new Gaussian octave
        $\mathbf{G}_p$ with $Q+3$ levels levels.
15:     $\mathbf{G}_{p,-1} \leftarrow \mathsf{G}_{\mathrm{base}}$                                   ▷ level $q = -1$
16:     for $q \leftarrow 0, \ldots, Q+1$ do                    ▷ levels $q = -1, \ldots, Q+1$
17:         $\tilde{\sigma}_q \leftarrow \sigma_0 \cdot \sqrt{2^{2q/Q} - 2^{-2/Q}}$    ▷ rel. scale w.r.t base level $\mathsf{G}_{\mathrm{base}}$
18:         $\mathbf{G}_{p,q} \leftarrow \mathsf{G}_{\mathrm{base}} * H^{\mathrm{G},\tilde{\sigma}_q}$              ▷ 2D Gaussian filter with $\tilde{\sigma}_q$
19:     $\mathbf{G}_p \leftarrow (\mathbf{G}_{p,-1}, \ldots, \mathbf{G}_{p,Q+1})$
20:     return $\mathbf{G}_p$

21: MakeDogOctave($\mathbf{G}_p, p, Q$)
        Input: $\mathbf{G}_p$, Gaussian octave; $p$, octave index; $Q$, scale steps per
        octave. Returns a new DoG octave $\mathbf{D}_p$ with $Q+2$ levels.
22:     for $q \leftarrow -1, \ldots, Q$ do
23:         $\mathbf{D}_{p,q} \leftarrow \mathbf{G}_{p,q+1} - \mathbf{G}_{p,q}$              ▷ diff. of Gaussians, Eq. 25.30
24:     $\mathbf{D}_p \leftarrow (\mathbf{D}_{p,-1}, \mathbf{D}_{p,0}, \ldots, \mathbf{D}_{p,Q})$             ▷ levels $q = -1, \ldots, Q$
25:     return $\mathbf{D}_p$
```

Alg. 25.2
Building a SIFT scale space. This procedure is an extension of Alg. 25.1 and takes the same parameters. The SIFT scale space (see Fig. 25.14) consists of two components: a hierarchical Gaussian scale space $\mathbf{G} = (\mathbf{G}_0, \ldots, \mathbf{G}_{P-1})$ with P octaves and a (derived) hierarchical DoG scale space $\mathbf{D} = (\mathbf{D}_0, \ldots, \mathbf{D}_{P-1})$. Each Gaussian octave $\mathbf{G}_p$ holds $Q+3$ levels $(\mathbf{G}_{p,-1}, \ldots, \mathbf{G}_{p,Q+1})$. At each Gaussian octave, the lowest level $\mathbf{G}_{p,-1}$ is obtained by decimating level $Q-1$ of the previous octave $\mathbf{G}_{p-1}$ (line 7). Every DoG octave $\mathbf{D}_p$ contains $Q+2$ levels $(\mathbf{D}_{p,-1}, \ldots, \mathbf{D}_{p,Q})$. A DoG level $\mathbf{D}_{p,q}$ is calculated as the pointwise difference of two adjacent Gaussian levels $\mathbf{G}_{p,q+1}$ and $\mathbf{G}_{p,q}$ (line 23). Typical parameter settings are $\sigma_s = 0.5$, $\sigma_0 = 1.6$, $Q = 3$, $P = 4$.

is either *negative* and also *smaller* or *positive* and *greater* than all neighboring values. In addition, a minimum difference $\mathrm{t}_{\mathrm{extrm}} \geq 0$ can be specified, indicating how much the center value must at least deviate from the surrounding values. The decision whether a given neighborhood $\mathsf{N}_{\boldsymbol{c}}$ contains a local minimum or maximum can thus be expressed as

$$\begin{aligned}\mathsf{IsLocalMin}(\mathsf{N}_{\boldsymbol{c}}) := \;& \mathsf{N}_{\boldsymbol{c}}(0,0,0) < 0 \;\wedge \\ & \mathsf{N}_{\boldsymbol{c}}(0,0,0) + \mathrm{t}_{\mathrm{extrm}} < \min_{\substack{(i,j,k)\neq\\(0,0,0)}} \mathsf{N}_{\boldsymbol{c}}(i,j,k), \end{aligned} \tag{25.54}$$

$$\begin{aligned}\mathsf{IsLocalMax}(\mathsf{N}_{\boldsymbol{c}}) := \;& \mathsf{N}_{\boldsymbol{c}}(0,0,0) > 0 \;\wedge \\ & \mathsf{N}_{\boldsymbol{c}}(0,0,0) - \mathrm{t}_{\mathrm{extrm}} < \max_{\substack{(i,j,k)\neq\\(0,0,0)}} \mathsf{N}_{\boldsymbol{c}}(i,j,k) \end{aligned} \tag{25.55}$$



(a) 26-neighborhood (b) 18-neighborhood (c) 10-neighborhood

Fig. 25.15 Different 3D neighborhoods for detecting local extrema in the DoG scale space. The red cube represents the DoG value at the reference coordinate $\boldsymbol{c} = (u, v, q)$ at the spatial position (u, v) at scale level q (within some octave p). Full $3 \times 3 \times 3$ neighborhood with 26 elements (a); other types of neighborhoods with 18 (b) or 10 (c) elements, respectively, are also commonly used. A local maximum/minimum is detected if the DoG value at the center is greater/smaller than all neighboring values (green cubes).

(see procedure $\mathsf{IsExtremum}(\mathsf{N}_{\boldsymbol{c}})$ in Alg. 25.5). As illustrated in Fig. 25.15(b–c), alternative 3D neighborhoods with 18 or 10 cells may be specified for extrema detection.

25.2.2 Position Refinement

Once a local extremum is detected in the DoG scale space, only its *discrete* 3D coordinates $\boldsymbol{c} = (u, v, q)$ are known, consisting of the spatial grid position (u, v) and the index (q) of the associated scale level. In the second step, a more accurate, *continuous* position for each candidate key point is estimated by fitting a quadratic function to the local neighborhood, as proposed in [37]. This is particularly important at the higher octaves of the scale space, where the spatial resolution becomes increasingly coarse due to successive decimation. Position refinement is based on a local second-order Taylor expansion of the discrete DoG function, which yields a continuous approximation function whose maximum or minimum can be found analytically. Additional details and illustrative examples are provided in Sec. C.3.2 of the Appendix.

At any extremal position $\boldsymbol{c} = (u, v, q)$ in octave p of the hierarchical DoG scale space $\mathbf{D}$, the corresponding 3×3×3 neighborhood $\mathcal{N}_D(\boldsymbol{c})$ is used to estimate the elements of the continuous 3D gradient, that is,

$$\nabla_D(\boldsymbol{c}) = \begin{pmatrix} d_{\mathrm{x}} \\ d_{\mathrm{y}} \\ d_{\sigma} \end{pmatrix} \approx \frac{1}{2} \cdot \begin{pmatrix} D(\boldsymbol{c}+\mathbf{e}_{\mathrm{i}}) - D(\boldsymbol{c}-\mathbf{e}_{\mathrm{i}}) \\ D(\boldsymbol{c}+\mathbf{e}_{\mathrm{j}}) - D(\boldsymbol{c}-\mathbf{e}_{\mathrm{j}}) \\ D(\boldsymbol{c}+\mathbf{e}_{\mathrm{k}}) - D(\boldsymbol{c}-\mathbf{e}_{\mathrm{k}}) \end{pmatrix}, \tag{25.56}$$

with $D(\,)$ as defined in Eqn. (25.51). Similarly, the 3×3 Hessian matrix for position $\boldsymbol{c}$ is obtained as

$$\mathbf{H}_D(\boldsymbol{c}) = \begin{pmatrix} d_{xx} & d_{xy} & d_{x\sigma} \\ d_{xy} & d_{yy} & d_{y\sigma} \\ d_{x\sigma} & d_{y\sigma} & d_{\sigma\sigma} \end{pmatrix}, \tag{25.57}$$

with the required second order derivatives estimated as

$$\begin{aligned}
d_{xx} &= D(\boldsymbol{c}-\mathbf{e}_{\mathrm{i}}) - 2 \cdot D(\boldsymbol{c}) + D(\boldsymbol{c}+\mathbf{e}_{\mathrm{i}}), \\
d_{yy} &= D(\boldsymbol{c}-\mathbf{e}_{\mathrm{j}}) - 2 \cdot D(\boldsymbol{c}) + D(\boldsymbol{c}+\mathbf{e}_{\mathrm{j}}), \\
d_{\sigma\sigma} &= D(\boldsymbol{c}-\mathbf{e}_{\mathrm{k}}) - 2 \cdot D(\boldsymbol{c}) + D(\boldsymbol{c}+\mathbf{e}_{\mathrm{k}}), \\
d_{xy} &= \tfrac{D(\boldsymbol{c}+\mathbf{e}_{\mathrm{i}}+\mathbf{e}_{\mathrm{j}}) - D(\boldsymbol{c}-\mathbf{e}_{\mathrm{i}}+\mathbf{e}_{\mathrm{j}}) - D(\boldsymbol{c}+\mathbf{e}_{\mathrm{i}}-\mathbf{e}_{\mathrm{j}}) + D(\boldsymbol{c}-\mathbf{e}_{\mathrm{i}}-\mathbf{e}_{\mathrm{j}})}{4}, \\
d_{x\sigma} &= \tfrac{D(\boldsymbol{c}+\mathbf{e}_{\mathrm{i}}+\mathbf{e}_{\mathrm{k}}) - D(\boldsymbol{c}-\mathbf{e}_{\mathrm{i}}+\mathbf{e}_{\mathrm{k}}) - D(\boldsymbol{c}+\mathbf{e}_{\mathrm{i}}-\mathbf{e}_{\mathrm{k}}) + D(\boldsymbol{c}-\mathbf{e}_{\mathrm{i}}-\mathbf{e}_{\mathrm{k}})}{4}, \\
d_{y\sigma} &= \tfrac{D(\boldsymbol{c}+\mathbf{e}_{\mathrm{j}}+\mathbf{e}_{\mathrm{k}}) - D(\boldsymbol{c}-\mathbf{e}_{\mathrm{j}}+\mathbf{e}_{\mathrm{k}}) - D(\boldsymbol{c}+\mathbf{e}_{\mathrm{j}}-\mathbf{e}_{\mathrm{k}}) + D(\boldsymbol{c}-\mathbf{e}_{\mathrm{j}}-\mathbf{e}_{\mathrm{k}})}{4}.
\end{aligned} \tag{25.58}$$

See the procedures $\mathsf{Gradient}(\mathsf{N}_{\boldsymbol{c}})$ and $\mathsf{Hessian}(\mathsf{N}_{\boldsymbol{c}})$ in Alg. 25.5 (p. 651) for additional details. From the gradient vector $\nabla_D(\boldsymbol{c})$ and the Hessian matrix $\mathbf{H}_D(\boldsymbol{c})$, the second order Taylor expansion around point $\boldsymbol{c}$ is

$$\tilde{D}_{\boldsymbol{c}}(\boldsymbol{x}) = D(\boldsymbol{c}) + \nabla_D^{\mathsf{T}}(\boldsymbol{c})\cdot(\boldsymbol{x}-\boldsymbol{c}) + \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{c})^{\mathsf{T}}\cdot\mathbf{H}_D(\boldsymbol{c})\cdot(\boldsymbol{x}-\boldsymbol{c}), \tag{25.59}$$

for the continuous position $\boldsymbol{x} = (x, y, \sigma)^{\mathsf{T}}$. The scalar-valued function $\tilde{D}_{\boldsymbol{c}}(\boldsymbol{x}) \in \mathbb{R}$, with $\boldsymbol{c} = (u, v, q)^{\mathsf{T}}$ and $\boldsymbol{x} = (x, y, \sigma)^{\mathsf{T}}$, is a local, *continuous* approximation of the discrete DoG function $\mathbf{D}_{p,q}(u, v)$ at octave p, scale level q, and spatial position u, v. This is a quadratic function with an extremum (maximum or minimum) at position

$$\breve{\boldsymbol{x}} = \begin{pmatrix} \breve{x} \\ \breve{y} \\ \breve{\sigma} \end{pmatrix} = \boldsymbol{c} + \boldsymbol{d} = \boldsymbol{c} \underbrace{- \mathbf{H}_D^{-1}(\boldsymbol{c}) \cdot \nabla_D(\boldsymbol{c})}_{\boldsymbol{d}=\breve{\boldsymbol{x}}-\boldsymbol{c}} \tag{25.60}$$

with $\boldsymbol{d} = (x', y', \sigma')^{\mathsf{T}} = \breve{\boldsymbol{x}} - \boldsymbol{c}$, under the assumption that the inverse of the Hessian matrix $\mathbf{H}_D$ exists. By inserting the extremal position $\breve{\boldsymbol{x}}$ into Eqn. (25.59), the peak (minimum or maximum) *value* of the continuous approximation function $\tilde{D}$ is found as[15]

$$\begin{aligned} D_{\text{peak}}(\boldsymbol{c}) = \tilde{D}_{\boldsymbol{c}}(\breve{\boldsymbol{x}}) &= D(\boldsymbol{c}) + \tfrac{1}{2} \cdot \nabla_D^{\mathsf{T}}(\boldsymbol{c}) \cdot (\breve{\boldsymbol{x}} - \boldsymbol{c}) \\ &= D(\boldsymbol{c}) + \tfrac{1}{2} \cdot \nabla_D^{\mathsf{T}}(\boldsymbol{c}) \cdot \boldsymbol{d}, \end{aligned} \tag{25.61}$$

where $\boldsymbol{d} = \breve{\boldsymbol{x}} - \boldsymbol{c}$ (cf. Eqn. (25.60)) denotes the 3D vector between the neighborhood's discrete center position $\boldsymbol{c}$ and the continuous extremal position $\breve{\boldsymbol{x}}$.

A scale space location $\boldsymbol{c}$ is only retained as a candidate interest point if the estimated magnitude of the DoG exceeds a given threshold t_{peak}, that is, if

$$|D_{\text{peak}}(\boldsymbol{c})| > \mathrm{t}_{\text{peak}}. \tag{25.62}$$

If the distance $\boldsymbol{d} = (x', y', \sigma')^{\mathsf{T}}$ from $\boldsymbol{c}$ to the estimated (continuous) peak position $\breve{\boldsymbol{x}}$ in Eqn. (25.60) is greater than a predefined limit (typically 0.5) in any spatial direction, the center point $\boldsymbol{c} = (u, v, q)^{\mathsf{T}}$ is moved to one of the neighboring DoG cells by maximally ± 1 unit steps along the u, v axes, that is,

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \begin{pmatrix} \min(1, \max(-1, \mathrm{round}(x'))) \\ \min(1, \max(-1, \mathrm{round}(y'))) \\ 0 \end{pmatrix}. \tag{25.63}$$

The q component of $\boldsymbol{c}$ is not modified in this version, that is, the search continues at the original scale level.[16] Based on the surrounding 3D neighborhood of this new point, a Taylor expansion (Eqn. (25.60)) is again performed to estimate a new peak location. This is repeated until either the peak location is inside the current DoG cell or the allowed number of repositioning steps $\mathrm{n}_{\text{refine}}$ is reached

[15] See Eqn. (C.64) in Sec. C.3.3 in the Appendix for details.

[16] This is handled differently in other SIFT implementations.

(typically $\mathrm{n_{refine}}$ is set to 4 or 5). If successful, the result of this step is a candidate feature point

$$\breve{\boldsymbol{c}} = (\breve{x}, \breve{y}, \breve{q})^{\mathsf{T}} = \boldsymbol{c} + (x', y', 0)^{\mathsf{T}}. \tag{25.64}$$

Notice that (in this implementation) the scale level q remains unchanged even if the 3D Taylor expansion indicates that the estimated peak is located at another scale level. See procedure RefineKeyPosition() in Alg. 25.4 (p. 650) for a concise summary of these steps.

It should be mentioned that the original publication [153] is not particularly explicit about the aforementioned position refinement process and thus slightly different approaches are used in various open-source SIFT implementations. For example, the implementation in *VLFeat*[17] [241] moves to one of the direct neighbors at the same scale level as described earlier, as long as $|x'|$ or $|y'|$ is greater than 0.6. *AutoPano-SIFT*[18] by S. Nowozin calculates the length of the spatial displacement $d = \|(x', y')\|$ and discards the current point if $d > 2$. Otherwise it moves by $\Delta_u = \mathrm{round}(x')$, $\Delta_v = \mathrm{round}(y')$ without limiting the displacement to ± 1. The *Open-Source SIFT Library*[19] [106] used in *OpenCV* also makes full moves in the spatial directions and, in addition, potentially also changes the scale level by $\Delta_q = \mathrm{round}(\sigma')$ in each iteration.

25.2.3 Suppressing Responses to Edge-Like Structures

In the previous step, candidate interest points were selected as those locations in the DoG scale space where the Taylor approximation had a local maximum and the extrapolated DoG value was above a given threshold ($\mathrm{t_{peak}}$). However, the DoG filter also responds strongly to edge-like structures. At such positions, interest points cannot be located with sufficient stability and repeatability. To eliminate the responses near edges, Lowe suggests the use of the principal curvatures of the 2D DoG result along the spatial x, y axes, using the fact that the principal curvatures of a function are proportional to the eigenvalues of the function's Hessian matrix at a given point.

For a particular lattice point $\boldsymbol{c} = (u, v, q)$ in DoG scale space, with neighborhood N_D (see Eqn. (25.52)), the 2×2 Hessian matrix for the spatial coordinates is

$$\mathbf{H}_{xy}(\boldsymbol{c}) = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{xy} & d_{yy} \end{pmatrix}, \tag{25.65}$$

with d_{xx}, d_{xy}, d_{yy} as defined in Eqn. (25.58), that is, these values can be extracted from the corresponding 3×3 Hessian matrix $\mathbf{H}_D(\boldsymbol{c})$ (see Eqn. (25.57)).

The matrix $\mathbf{H}_{xy}(\boldsymbol{c})$ has two eigenvalues λ_1, λ_2, which we define as being ordered, such that λ_1 has the greater magnitude ($|\lambda_1| \geq |\lambda_2|$). If both eigenvalues for a point $\boldsymbol{c}$ are of similar magnitude, the function exhibits a high curvature along two orthogonal directions and in this

[17] http://www.vlfeat.org/overview/sift.html.

[18] http://sourceforge.net/projects/hugin/files/autopano-sift-C/.

[19] http://robwhess.github.io/opensift/.

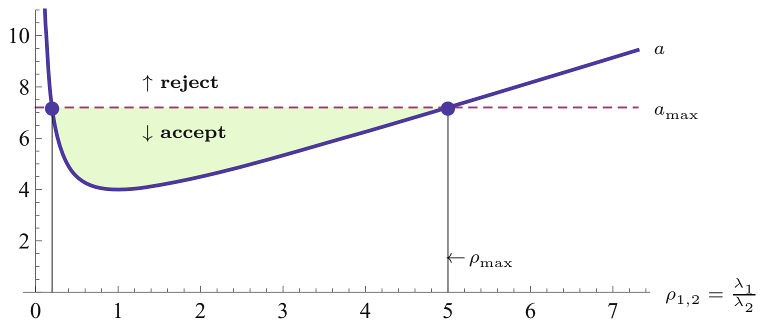


Fig. 25.16
Limiting the ratio of principal curvatures (edge ratio) $\rho_{1,2}$ by specifying $a_{\max}$. The quantity a (blue line) has a minimum when the eigenvalue ratio $\rho_{1,2} = \frac{\lambda_1}{\lambda_2}$ is one, that is, when the two eigenvalues λ_1, λ_2 are equal, indicating a corner-like event. Typically only one of the eigenvalues is dominant in the vicinity of image lines, such that $\rho_{1,2}$ and a values are significantly increased. In this example, the principal curvature ratio $\rho_{1,2}$ is limited to $\rho_{\max} = 5.0$ by setting $a_{\max} = (5+1)^2/5 = 7.2$ (red line).

case $\boldsymbol{c}$ is likely to be a good reference point that can be located reliably. In the optimal situation (e.g., near a corner), the ratio of the eigenvalues $\rho = \lambda_1/\lambda_2$ is close to 1. Alternatively, if the ratio ρ is high it can be concluded that a single orientation dominates at this position, as is typically the case in the neighborhood of edges.

To estimate the ratio ρ it is not necessary to calculate the eigenvalues themselves. Following the description in [153], the sum and product of the eigenvalues λ_1, λ_2 can be found as

$$\lambda_1 + \lambda_2 = \text{trace}(\mathbf{H}_{xy}(\boldsymbol{c})) = d_{xx} + d_{yy}, \tag{25.66}$$

$$\lambda_1 \cdot \lambda_2 = \det(\mathbf{H}_{xy}(\boldsymbol{c})) = d_{xx} \cdot d_{yy} - d_{xy}^2. \tag{25.67}$$

If the determinant $\det(\mathbf{H}_{xy})$ is *negative*, the principal curvatures of the underlying 2D function have opposite signs and thus point $\boldsymbol{c}$ can be discarded as not being an extremum. Otherwise, if the signs of both eigenvalues λ_1, λ_2 are the *same*, then the ratio

$$\rho_{1,2} = \frac{\lambda_1}{\lambda_2} \tag{25.68}$$

is positive (with $\lambda_1 = \rho_{1,2} \cdot \lambda_2$), and thus the expession

$$a = \frac{[\text{trace}(\mathbf{H}_{xy}(\boldsymbol{c}))]^2}{\det(\mathbf{H}_{xy}(\boldsymbol{c}))} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \cdot \lambda_2} \tag{25.69}$$

$$= \frac{(\rho_{1,2} \cdot \lambda_2 + \lambda_2)^2}{\rho_{1,2} \cdot \lambda_2^2} = \frac{\lambda_2^2 \cdot (\rho_{1,2} + 1)^2}{\rho_{1,2} \cdot \lambda_2^2} = \frac{(\rho_{1,2} + 1)^2}{\rho_{1,2}} \tag{25.70}$$

depends only on the ratio $\rho_{1,2}$. If the determinant of $\mathbf{H}_{xy}$ is positive, the quantity a has a minimum (4.0) at $\rho_{1,2} = 1$, if the two eigenvalues are equal (see Fig. 25.16). Note that the ratio a is the same for $\rho_{1,2} = \lambda_1/\lambda_2$ or $\rho_{1,2} = \lambda_2/\lambda_1$, since

$$a = \frac{(\rho_{1,2} + 1)^2}{\rho_{1,2}} = \frac{(\frac{1}{\rho_{1,2}} + 1)^2}{\frac{1}{\rho_{1,2}}} . \tag{25.71}$$

To verify that the eigenvalue ratio $\rho_{1,2}$ at a given position $\boldsymbol{c}$ is *below* a specified limit $\rho_{\max}$ (making $\boldsymbol{c}$ a good candidate), it is thus sufficient to check the condition

$$a \leq a_{\max}, \qquad \text{with} \qquad a_{\max} = \frac{(\rho_{\max} + 1)^2}{\rho_{\max}}, \tag{25.72}$$

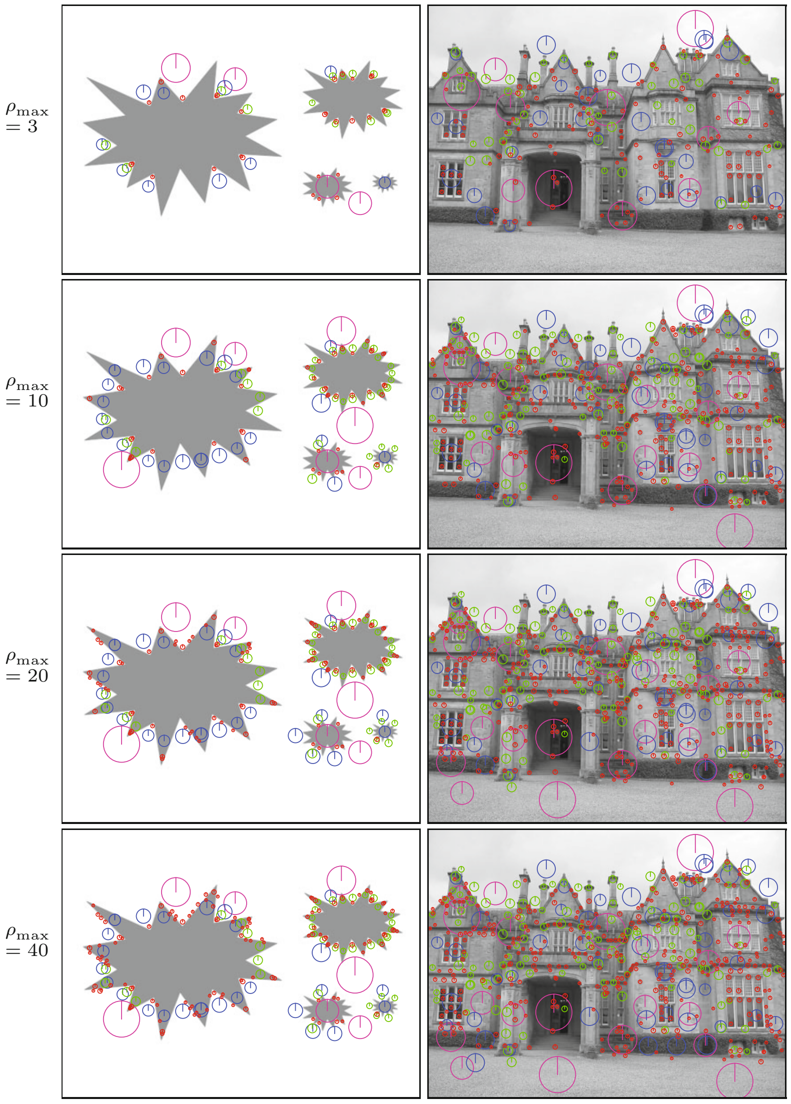


Fig. 25.17
Rejection of edge-like features by controlling the max. curvature ratio $\rho_{\max}$. The size of the circles is proportional to the scale level at which the corresponding key point was detected, the color indicating the containing octave (0 = red, 1 = green, 2 = blue, 3 = magenta).

without the need to actually calculate the individual eigenvalues λ_1 and λ_2.[20] $\rho_{\max}$ should be greater than 1 and is typically chosen to be in the range $3, \ldots, 10$ ($\rho_{\max} = 10$ is suggested in [153]). The resulting value of $a_{\max}$ in Eqn. (25.72) is constant and needs only be calculated once (see Alg. 25.3, line 2). Detection examples for varying values of $\rho_{\max}$ are shown in Fig. 25.17. Note that considerably more candidates appear near edges as $\rho_{\max}$ is raised from 3 to 40.

25.3 Creating Local Descriptors

For each local maximum detected in the hierarchical DoG scale space, a candidate key point is created, which is subsequently refined to

[20] A similar trick is used in the *Harris* corner detection algorithm (see Chapter 7).

a continuous position following the steps we have just described (see Eqns. (25.56)–(25.64)). Then, for each refined key point $\boldsymbol{k}' = (p, q, x, y)$, one or more (up to four) local descriptors are calculated. Multiple (up to four) descriptors may be created for a position if the local orientation is not unique. This process involves the following steps:

1. Find the *dominant* orientation(s) of the key point $\boldsymbol{k}'$ from the distribution of the gradients at the corresponding Gaussian scale space level.
2. For each dominant orientation, create a separate SIFT *descriptor* at the key point $\boldsymbol{k}'$.

25.3.1 Finding Dominant Orientations

Local orientation from Gaussian scale space

Orientation vectors are obtained by sampling the *gradient* values of the hierarchical Gaussian scale space $\mathbf{G}_{p,q}(u, v)$ (see Eqn. (25.32)). For any lattice position (u, v) at octave p and scale level q, the local gradient is calculated as

$$\nabla_{p,q}(u,v) = \begin{pmatrix} d_\mathrm{x} \\ d_\mathrm{y} \end{pmatrix} = 0.5 \cdot \begin{pmatrix} \mathbf{G}_{p,q}(u{+}1,v) - \mathbf{G}_{p,q}(u{-}1,v) \\ \mathbf{G}_{p,q}(u,v{+}1) - \mathbf{G}_{p,q}(u,v{-}1) \end{pmatrix}. \qquad (25.73)$$

From these gradient vectors, the gradient *magnitude* and *orientation* (i.e., polar coordinates) are found as[21]

$$E_{p,q}(u,v) = \left\| \nabla_{p,q}(u,v) \right\| = \sqrt{d_\mathrm{x}^2 + d_\mathrm{y}^2}\,, \qquad (25.74)$$

$$\phi_{p,q}(u,v) = \angle \nabla_{p,q}(u,v) \;= \tan^{-1}(d_\mathrm{y}/d_\mathrm{x}). \qquad (25.75)$$

These scalar fields $E_{p,q}$ and $\phi_{p,q}$ are typically pre-calculated for all relevant octaves/levels p, q of the Gaussian scale space $\mathbf{G}$.

Orientation histograms

To find the dominant orientations for a given key point, a histogram h_ϕ of the orientation angles is calculated for the gradient vectors collected from a square window around the key point center. Typically the histogram has $\mathrm{n}_\mathrm{orient} = 36$ bins, that is, the angular resolution is 10°. The orientation histogram is collected from a square region using an isotropic Gaussian weighting function whose width σ_w is proportional to the *decimated scale* $\dot{\sigma}_q$ (see Eqn. (25.37)) of the key point's scale level q. Typically a Gaussian weighting function "with a σ that is 1.5 times that of the scale of the key point" [153] is used, that is,

$$\sigma_\mathrm{w} = 1.5 \cdot \dot{\sigma}_q = 1.5 \cdot \sigma_0 \cdot 2^{q/Q}. \qquad (25.76)$$

Note that σ_w is independent of the octave index p and thus the same weighting functions are used in each octave. To calculate the *orientation histogram*, the Gaussian gradients around the given key point are collected from a square region of size $2r_\mathrm{w} \times 2r_\mathrm{w}$, with

[21] See also Chapter 16, Sec. 16.1.

$$r_{\mathrm{w}} = \lceil 2.5 \cdot \sigma_{\mathrm{w}} \rceil \tag{25.77}$$

amply dimensioned to avoid numerical truncation effects. For the parameters listed in Table 25.3 ($\sigma_0 = 1.6$, $Q = 3$), the values for σ_{w} (expressed in the octave's coordinate units) are

q	0	1	2	3
σ_{w}	1.6000	2.0159	2.5398	3.2000
r_{w}	4	5	6	7

(25.78)

In Alg. 25.7, σ_{w} and r_{w} of the Gaussian weighting function are calculated in lines 7 and 8, respectively. At each lattice point (u, v), the gradient vector $\nabla_{p,q}(u, v)$ is calculated in octave p and level q of the Gaussian scale space $\mathbf{G}$ (Alg. 25.7, line 16). From this, the gradient magnitude $E_{p,q}(u, v)$ and orientation $\phi_{p,q}(u, v)$ are obtained (lines 29–30). The corresponding Gaussian weight is calculated (in line 18) from the spatial distance between the grid point (u, v) and the interest point (x, y) as

$$w_{\mathrm{G}}(u, v) = \exp\left(-\frac{(u-x)^2+(v-y)^2}{2\cdot\sigma_{\mathrm{w}}^2}\right). \tag{25.79}$$

For the grid point (u, v), the quantity to be accumulated into the orientation histogram is

$$z = E_{p,q}(u, v) \cdot w_{\mathrm{G}}(u, v), \tag{25.80}$$

that is, the local gradient magnitude weighted by the Gaussian window function (Alg. 25.7, line 19).

The orientation histogram h_ϕ consists of $\mathrm{n_{orient}}$ bins and thus the *continuous* bin number for the angle $\phi(u, v)$ is

$$\kappa_\phi = \frac{\mathrm{n_{orient}}}{2\pi} \cdot \phi(u, v) \tag{25.81}$$

(see Alg. 25.7, line 20). To collect the *continuous* orientations into a histogram with discrete bins, quantization must be performed. The simplest approach is to select the "nearest" bin (by rounding) and to add the associated quantity (denoted z) entirely to the selected bin. Alternatively, to reduce quantization effects, a common technique is to *split* the quantity z onto the two closest bins. Given the continuous bin value κ_ϕ, the indexes of the two closest discrete bins are

$$k_0 = \lfloor \kappa_\phi \rfloor \bmod \mathrm{n_{orient}} \quad \text{and} \quad k_1 = (\lfloor \kappa_\phi \rfloor + 1) \bmod \mathrm{n_{orient}}, \tag{25.82}$$

respectively. The quantity z (Eqn. (25.80)) is then partitioned and accumulated into the neighboring bins k_0, k_1 of the orientation histogram h_ϕ in the form

$$\begin{aligned} \mathsf{h}_\phi(k_0) &\leftarrow \mathsf{h}_\phi(k_0) + (1-\alpha)\cdot z, \\ \mathsf{h}_\phi(k_1) &\leftarrow \mathsf{h}_\phi(k_1) + \alpha \cdot z, \end{aligned} \tag{25.83}$$

with $\alpha = \kappa_\phi - \lfloor \kappa_\phi \rfloor$. This process is illustrated by the example in Fig. 25.18 (see also Alg. 25.7, lines 21–25).

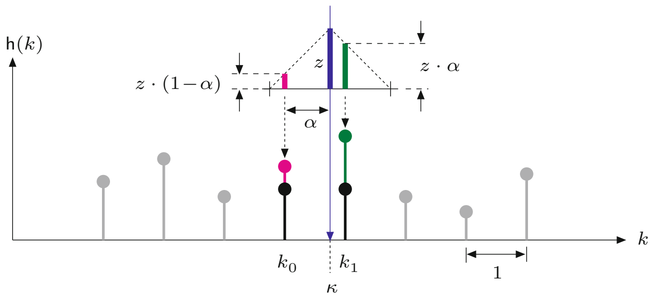


Fig. 25.18
Accumulating into multiple histogram bins by linear interpolation. Assume that some quantity z (blue bar) is to be added to the *discrete* histogram h_ϕ at the *continuous* position κ_ϕ. The histogram bins adjacent to κ_ϕ are $k_0 = \lfloor \kappa_\phi \rfloor$ and $k_1 = \lfloor \kappa_\phi \rfloor + 1$. The fraction of z accumulated into bin k_1 is $z_1 = z \cdot \alpha$ (red bar), with $\alpha = \kappa_\phi - k_0$. Analogously, the quantity added to bin k_0 is $z_0 = z \cdot (1-\alpha)$ (green bar).

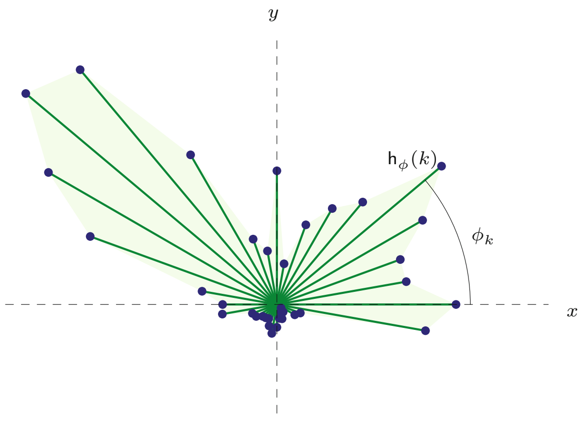


Fig. 25.19
Orientation histogram example. Each of the 36 radial bars corresponds to one entry in the orientation histogram h_ϕ. The *length* (radius) of each radial bar with index k is proportional to the accumulated value in the corresponding bin $\mathsf{h}_\phi(k)$ and its orientation is ϕ_k.

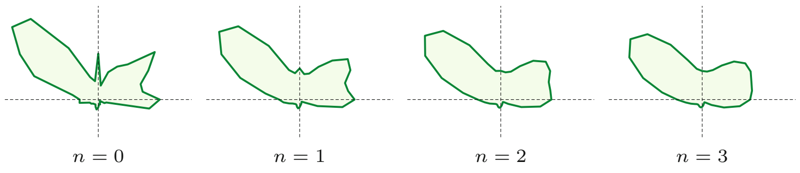


Fig. 25.20
Smoothing the orientation histogram (from Fig. 25.19) by repeatedly applying a circular low-pass filter with the 1D kernel $H = \frac{1}{4} \cdot (1, \mathbf{2}, 1)$.

Orientation histogram smoothing

Figure 25.19 shows a geometric rendering of the orientation histogram that explains the relevance of the cell indexes (discrete angles ϕ_k) and the accumulated quantities (z). Before calculating the dominant orientations, the raw orientation histogram h_ϕ is usually smoothed by applying a (circular) low-pass filter, typically a simple 3-tap Gaussian or box-type filter (see procedure SmoothCircular() in Alg. 25.7, lines 6–16).[22] Stronger smoothing is achieved by applying the filter multiple times, as illustrated in Fig. 25.20. In practice, two to three smoothing iterations appear to be sufficient.

Locating and interpolating orientation peaks

After smoothing the orientation histogram, the next step is to detect the peak entries in h_ϕ. A bin k is considered a significant orientation peak if $\mathsf{h}_\phi(k)$ is a local maximum and its value is not less than a certain fraction of the maximum histogram entry, that is, only if

[22] Histogram smoothing is not mentioned in the original SIFT publication [153] but used in most implementations.

$$\begin{aligned} &\mathsf{h}_\phi(k) > \mathsf{h}_\phi((k-1) \bmod \mathrm{n}_{\mathrm{orient}}) \wedge \\ &\mathsf{h}_\phi(k) > \mathsf{h}_\phi((k+1) \bmod \mathrm{n}_{\mathrm{orient}}) \wedge \\ &\mathsf{h}_\phi(k) > \mathrm{t}_{\mathrm{domor}} \cdot \max_i \mathsf{h}_\phi(i)\,, \end{aligned} \tag{25.84}$$

with $\mathrm{t}_{\mathrm{domor}} = 0.8$ as a typical limit.

To achieve a finer angular resolution than provided by the orientation histogram bins (typically spaced at 10° steps) alone, a continuous peak orientation is calculated by quadratic interpolation of the neighboring histogram values. Given a discrete peak index k, the interpolated (continuous) peak position $\breve{k}$ is obtained by fitting a quadratic function to the three successive histogram values $\mathsf{h}_\phi(k-1)$, $\mathsf{h}_\phi(k)$, $\mathsf{h}_\phi(k+1)$ as[23]

$$\breve{k} = k + \frac{\mathsf{h}_\phi(k-1) - \mathsf{h}_\phi(k+1)}{2 \cdot \left[\, \mathsf{h}_\phi(k-1) - 2\,\mathsf{h}_\phi(k) + \mathsf{h}_\phi(k+1) \,\right]}\,, \tag{25.85}$$

with all indexes taken modulo $\mathrm{n}_{\mathrm{orient}}$. From Eqn. (25.81), the (continuous) dominant orientation angle $\theta \in [0, 2\pi)$ is then obtained as

$$\theta = (\breve{k} \bmod \mathrm{n}_{\mathrm{orient}}) \cdot \frac{2\pi}{\mathrm{n}_{\mathrm{orient}}}\,, \tag{25.86}$$

mit $\theta \in [0, 2\pi)$. In this way, the dominant orientation can be estimated with accuracy much beyond the coarse resolution of the orientation histogram. Note that, in some cases, multiple histogram peaks are obtained for a given key point (see procedure FindPeakOrientations() in Alg. 25.6, lines 18–31). In this event, individual SIFT descriptors are created for each dominant orientation at the same key point position (see Alg. 25.3, line 8).

Figure 25.21 shows the orientation histograms for a set of detected key points in two different images after applying a varying number of smoothing steps. It also shows the interpolated dominant orientations θ calculated from the orientation histograms (Eqn. (25.86)) by the corresponding vectors.

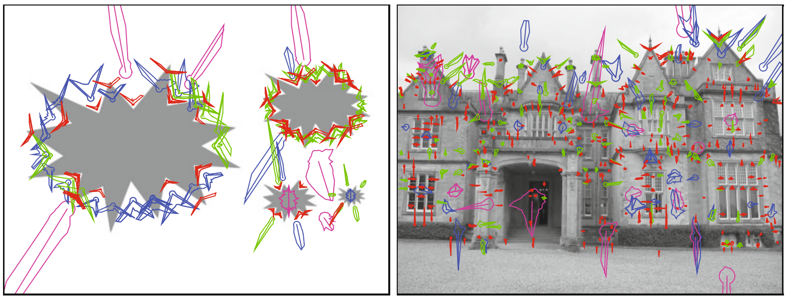
25.3.2 SIFT Descriptor Construction

For each key point $\boldsymbol{k}' = (p, q, x, y)$ and each dominant orientation θ, a corresponding SIFT descriptor is obtained by sampling the surrounding gradients at octave p and level q of the Gaussian scale space $\mathbf{G}$.

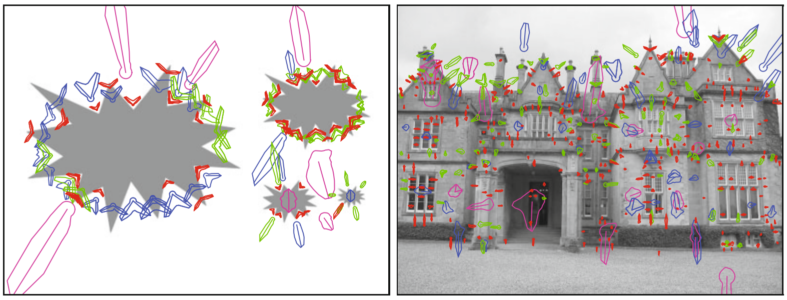
Descriptor geometry

The geometry underlying the calculation of SIFT descriptors is illustrated in Fig. 25.22. The descriptor combines the gradient orientation and magnitude from a square region of size $w_\mathrm{d} \times w_\mathrm{d}$, which is centered at the (continuous) position (x, y) of the associated feature point and aligned with its dominant orientation θ. The side length of the descriptor is set to $w_\mathrm{d} = 10 \cdot \dot{\sigma}_q$, where $\dot{\sigma}_q$ denotes the key point's decimated scale (radius of the inner circle). It depends on the key point's scale level q (see Table 25.4).

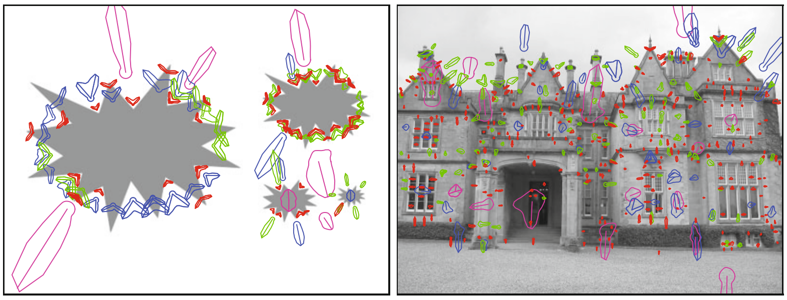
[23] See Sec. C.1.2 in the Appendix for details.



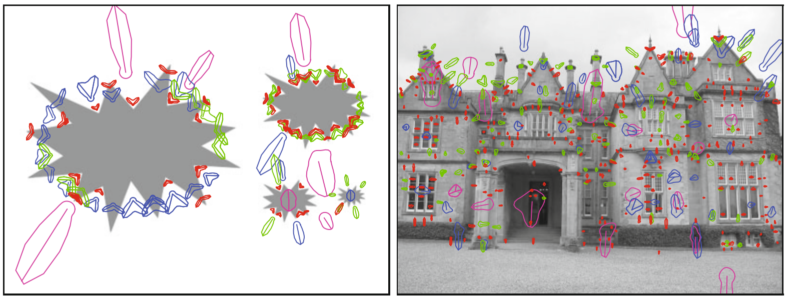
(a) $n = 0$



(b) $n = 1$



(c) $n = 2$



(d) $n = 3$

Fig. 25.21
Orientation histograms and dominant orientations (examples). $n = 0, \ldots, 3$ smoothing iterations were applied to the orientation histograms. The (interpolated) dominant orientations are shown as radial lines that emanate from each feature's center point. The size of the histogram graphs is proportional to the absolute scale ($\sigma_{p,q}$, see Table 25.3) at which the corresponding key point was detected. The colors indicate the index of the containing scale space octave p (red = 0, green = 1, blue = 2, magenta = 3).

The region is partitioned into $\mathrm{n_{spat}} \times \mathrm{n_{spat}}$ sub-squares of identical size; typically $\mathrm{n_{spat}} = 4$ (see Table 25.5). The contribution of each gradient sample is attenuated by a circular Gaussian function of width $\sigma_{\mathrm{d}} = 0.25 \cdot w_{\mathrm{d}}$ (blue circle). The weights drop off radially and are practically zero at $r_{\mathrm{d}} = 2.5 \cdot \sigma_{\mathrm{d}}$ (green circle in Fig. 25.22). Thus only samples outside this zone need to be included for calculating the descriptor statistics.

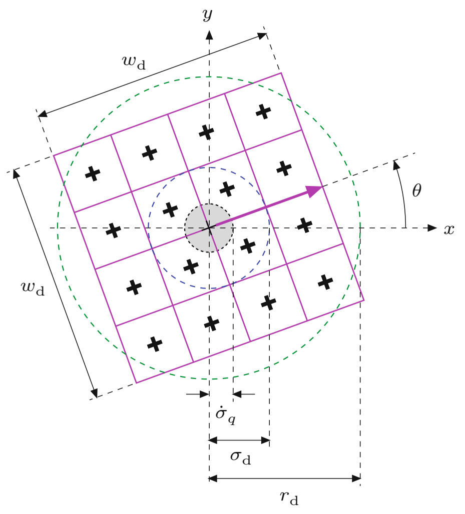


Fig. 25.22
Geometry of a SIFT descriptor. The descriptor is calculated from a square support region that is centered at the key point's position (x, y), aligned to the key point's dominant orientation θ, and partitioned into $n_{spat} \times n_{spat}$ (4×4) subsquares. The radius of the inner (gray) circle corresponds to the feature point's decimated scale value $(\dot{\sigma}_q)$. The blue circle displays the width (σ_d) of the Gaussian weighting function applied to the gradients; its value is practically zero outside the green circle (r_d).

To achieve rotation invariance, the descriptor region is aligned to the key point's dominant orientation, as determined in the previous steps. To make the descriptor invariant to scale changes, its size w_d (expressed in the grid coordinate units of octave p) is set proportional to the key point's *decimated scale* $\dot{\sigma}_q$ (see Eqn. (25.37)), that is,

$$w_d = \mathsf{s}_d \cdot \dot{\sigma}_q = \mathsf{s}_d \cdot \sigma_0 \cdot 2^{q/Q}, \tag{25.87}$$

where s_d is a constant size factor. For $\mathsf{s}_d = 10$ (see Table 25.5), the descriptor size w_d ranges from 16.0 (at level 0) to 25.4 (at level 2), as listed in Table 25.4. Note that the descriptor size w_d only depends on the scale level index q and is independent of the octave index p. Thus the same descriptor geometry applies to all octaves of the scale space.

Table 25.4
SIFT descriptor dimensions for different scale levels q (for size factor $\mathsf{s}_d = 10$ and $Q = 3$ levels per octave). $\dot{\sigma}_q$ is the key point's decimated scale, w_d is the descriptor size, σ_d is the width of the Gaussian weighting function, and r_d is the radius of the descriptor's support region. For $Q = 3$, only scale levels $q = 0, 1, 2$ are relevant. All lengths are expressed in the octave's (i.e., decimated) coordinate units.

q	$\dot{\sigma}_q$	$w_d = \mathsf{s}_d \cdot \dot{\sigma}_q$	$\sigma_d = 0.25 \cdot w_d$	$r_d = 2.5 \cdot \sigma_d$
3	3.2000	32.000	8.0000	20.0000
2	2.5398	25.398	6.3495	15.8738
1	2.0159	20.159	5.0398	12.5994
0	1.6000	16.000	4.0000	10.0000
−1	1.2699	12.699	3.1748	7.9369

The descriptor's *spatial resolution* is specified by the parameter n_{spat}. Typically $n_{spat} = 4$ (as shown in Fig. 25.22) and thus the total number of spatial bins is $n_{spat} \times n_{spat} = 16$ (in this case). Each spatial descriptor bin relates to an area of size $(w_d/n_{spat}) \times (w_d/n_{spat})$. For example, at scale level $q = 0$ of any octave, $\dot{\sigma}_0 = 1.6$ and the corresponding descriptor size is $w_d = \mathsf{s}_d \cdot \dot{\sigma}_0 = 10 \cdot 1.6 = 16.0$ (see Table 25.4). In this case (illustrated in Fig. 25.23), the descriptor covers 16×16 gradient samples, as suggested in [153]. Figure 25.24 shows an example with M-shaped feature point markers aligned to the dominant orientation and scaled to the descriptor region width w_d of the associated scale level.

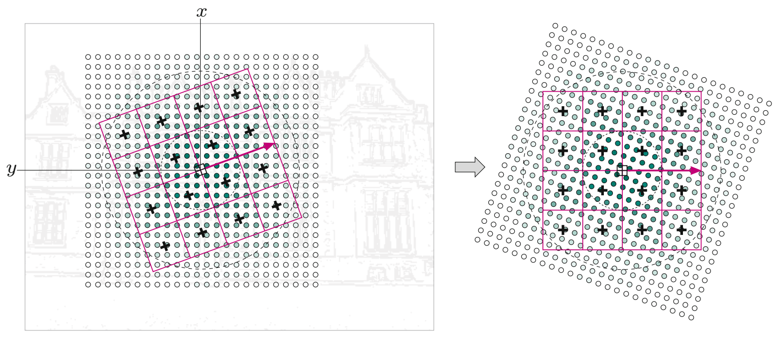


Fig. 25.23
Geometry of the SIFT descriptor in relation to the discrete sample grid of the associated octave (level $q = 0$, parameter $\mathsf{s}_\mathrm{d} = 10$). In this case, the decimated scale is $\dot{\sigma}_0 = 1.6$ and the width of the descriptor is $w_\mathrm{d} = \mathsf{s}_\mathrm{d} \cdot \dot{\sigma}_0 = 10 \cdot 1.6 = 16.0$.

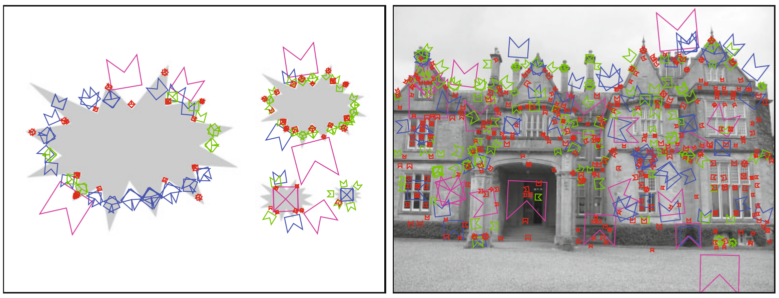


Fig. 25.24
Marked key points aligned to their dominant orientation. Note that multiple feature instances are inserted at key point positions with more than one dominant orientation. The size of the markers is proportional to the absolute scale ($\sigma_{p,q}$, see Table 25.3) at which the corresponding key point was detected. The colors indicate the index of the scale space containing octave p (red = 0, green = 1, blue = 2, magenta = 3).

Gradient features

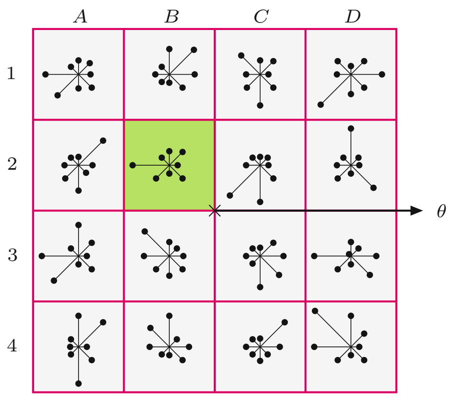
The actual SIFT descriptor is a feature vector obtained by histogramming the gradient orientations of the Gaussian scale level within the descriptors spatial support region. This requires a 3D histogram $\mathsf{h}_\nabla(i, j, k)$, with two spatial dimensions (i, j) for the $\mathrm{n_{spat}} \times \mathrm{n_{spat}}$ sub-regions and one additional dimension (k) for $\mathrm{n_{angl}}$ gradient orientations. This histogram thus contains $\mathrm{n_{spat}} \times \mathrm{n_{spat}} \times \mathrm{n_{angl}}$ bins.

Figure 25.25 illustrates this structure for the typical setup, with $\mathrm{n_{spat}} = 4$ and $\mathrm{n_{angl}} = 8$ (see Table 25.5). In this arrangement, eight orientation bins $k = 0, \ldots, 7$ are attached to each of the 16 spatial position bins $(A1, \ldots, D4)$, which makes a total of 128 histogram bins.

For a given key point $\boldsymbol{k}' = (p, q, x, y)$, the histogram h_∇ accumulates the orientations (angles) of the gradients at the Gaussian scale space level $\mathbf{G}_{p,q}$ within the support region around the (continous) center coordinate (x, y). At each grid point (u, v) inside this region, the gradient vector ∇_G is estimated (as described in Eqn. (25.73)), from which the gradient magnitude $E(u, v)$ and orientation $\phi(u, v)$ are calculated (see Eqns. (25.74)–(25.75) and lines 27–31 in Alg. 25.7). For efficiency reasons, $E(u, v)$ and $\phi(u, v)$ are typically pre-calculated for all relevant scale levels.

Each gradient sample contributes to the gradient histogram h_∇ a particular quantity z that depends on the gradient magnitude E and the distance of the sample point (u, v) from the key point's center (x, y). Again a Gaussian weighting function (of width σ_d) is used to attenuate samples with increasing spatial distance; thus the resulting accumulated quantity is

(a)



(b)

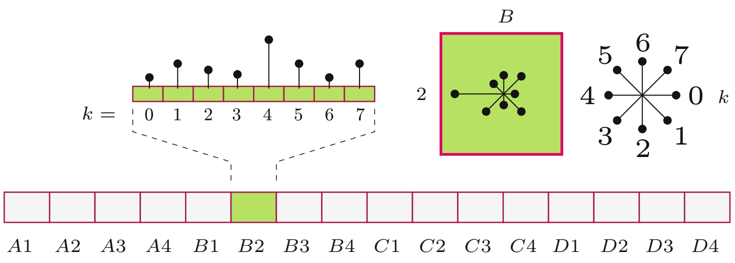


Fig. 25.25
SIFT descriptor structure for $n_{spat} = 4$ and $n_{angl} = 8$. Eight orientation bins $k = 0, \ldots, 7$ are provided for each of the 16 spatial bins $ij = A1, \ldots, D4$. Thus the gradient histogram h_{∇} holds 128 cells that are arranged to a 1D feature vector $(A1_0, A1_2 \ldots, D4_6, D4_7)$ as shown in (b).

$$z(u, v) = R(u, v) \cdot w_G = R(u, v) \cdot \exp\left(-\tfrac{(u-x)^2+(v-y)^2}{2\sigma_d^2}\right). \tag{25.88}$$

The width σ_d of the Gaussian function $w_G(\,)$ is proportional to the side length of the descriptor region, with

$$\sigma_d = 0.25 \cdot w_d = 0.25 \cdot s_d \cdot \dot{\sigma}_q. \tag{25.89}$$

The weighting function drops off radially from the center and is practically zero at distance $r_d = 2.5 \cdot \sigma_d$. Therefore, only gradient samples that are closer to the key point's center than r_d (green circle in Fig. 25.22) need to be considered in the gradient histogram calculation (see Alg. 25.8, lines 7 and 17). For a given key point $\boldsymbol{k}' = (p, q, x, y)$, sampling of the Gaussian gradients can thus be confined to the grid points (u, v) inside the square region bounded by $x \pm r_d$ and $y \pm r_d$ (see Alg. 25.8, lines 8–10 and 15–16). Each sample point (u, v) is then subjected to the affine transformation

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \frac{1}{w_d} \cdot \begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix} \cdot \begin{pmatrix} u-x \\ v-y \end{pmatrix}, \tag{25.90}$$

which performs a rotation by the dominant orientation θ and maps the original (rotated) square of size $w_d \times w_d$ to the unit square with coordinates $u', v' \in [-0.5, +0.5]$ (see Fig. 25.23).

To make feature vectors rotation invariant, the individual gradient orientations $\phi(u, v)$ are rotated by the dominant orientation, that is,

$$\phi'(u, v) = (\phi(u, v) - \theta) \bmod 2\pi, \tag{25.91}$$

with $\phi'(u, v) \in [0, 2\pi)$, such that the relative orientation is preserved.

For each gradient sample, with the continuous coordinates (u', v', ϕ'), the corresponding quantity $z(u, v)$ (Eqn. (25.88)) is accumulated into the 3D gradient histogram h_∇. For a complete description of this step see procedure UpdateGradientHistogram() in Alg. 25.9. It first maps the coordinates (u', v', ϕ') (see Eqn. (25.90)) to the continuous histogram position (i', j', k') by

$$\begin{aligned} i' &= \mathrm{n_{spat}} \cdot u' + 0.5 \cdot (\mathrm{n_{spat}} - 1), \\ j' &= \mathrm{n_{spat}} \cdot v' + 0.5 \cdot (\mathrm{n_{spat}} - 1), \\ k' &= \phi' \cdot \frac{\mathrm{n_{angl}}}{2\pi}, \end{aligned} \tag{25.92}$$

such that $i', j' \in [-0.5, \mathrm{n_{spat}} - 0.5]$ and $k' \in [0, \mathrm{n_{angl}})$.

Analogous to inserting into a continuous position of a 1D histogram by linear interpolation over *two* bins (see Fig. 25.18), the quantity z is distributed over *eight* neighboring histogram bins by *tri-linear* interpolation. The quantiles of z contributing to the individual histogram bins are determined by the distances of the coordinates (i', j', k') from the discrete indexes (i, j, k) of the affected histogram bins. The indexes (i, j, k) are found as the set of possible combinations $\{i_0, i_1\} \times \{j_0, j_1\} \times \{k_0, k_1\}$, with

$$\begin{aligned} i_0 &= \lfloor i' \rfloor, & i_1 &= (i_0 + 1), \\ j_0 &= \lfloor j' \rfloor, & j_1 &= (j_0 + 1), \\ k_0 &= \lfloor k' \rfloor \bmod \mathrm{n_{angl}}, & k_1 &= (k_0 + 1) \bmod \mathrm{n_{angl}}, \end{aligned} \tag{25.93}$$

and the corresponding quantiles (weights) are

$$\begin{aligned} \alpha_0 &= \lfloor i' \rfloor + 1 - i' = i_1 - i', & \alpha_1 &= 1 - \alpha_0, \\ \beta_0 &= \lfloor j' \rfloor + 1 - j' = j_1 - j', & \beta_1 &= 1 - \beta_0, \\ \gamma_0 &= \lfloor k' \rfloor + 1 - k', & \gamma_1 &= 1 - \gamma_0, \end{aligned} \tag{25.94}$$

and the (eight) affected bins of the gradient histogram are finally updated as

$$\begin{aligned} \mathsf{h}_\nabla(i_0, j_0, k_0) &\overset{+}{\leftarrow} z \cdot \alpha_0 \cdot \beta_0 \cdot \gamma_0, \\ \mathsf{h}_\nabla(i_1, j_0, k_0) &\overset{+}{\leftarrow} z \cdot \alpha_1 \cdot \beta_0 \cdot \gamma_0, \\ \mathsf{h}_\nabla(i_0, j_1, k_0) &\overset{+}{\leftarrow} z \cdot \alpha_0 \cdot \beta_1 \cdot \gamma_0, \\ &\vdots \qquad \vdots \\ \mathsf{h}_\nabla(i_1, j_1, k_1) &\overset{+}{\leftarrow} z \cdot \alpha_1 \cdot \beta_1 \cdot \gamma_1. \end{aligned} \tag{25.95}$$

Attention must be paid to the fact that the coordinate k represents an orientation and must therefore be treated in a *circular* manner, as illustrated in Fig. 25.26 (also see Alg. 25.9, lines 11–12).

For each histogram bin, the range of contributing gradient samples covers half of each neighboring bin, that is, the support regions of neighboring bins overlap, as illustrated in Fig. 25.27.

Normalizing SIFT descriptors

The elements of the gradient histogram h_∇ are the raw material for the SIFT feature vectors $\boldsymbol{f}_{\mathrm{sift}}$. The process of calculating the feature vectors from the gradient histogram is described in Alg. 25.10.

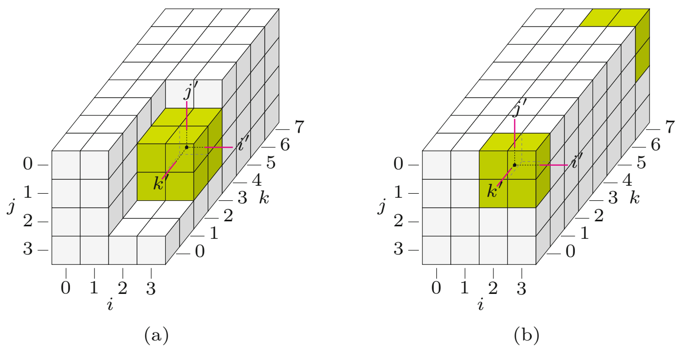


Fig. 25.26
3D structure of the gradient histogram, with $\mathrm{n_{spat}} \times \mathrm{n_{spat}} = 4 \times 4$ bins for the spatial dimensions (i, j) and $\mathrm{n_{angl}} = 8$ bins along the orientation axis (k). For the histogram to accumulate a quantity z into some continuous position (i', j', k'), eight adjacent bins receive different quantiles of z that are determined by tri-linear interpolation (a). Note that the bins along the orientation axis ϕ are treated circularly; for example, bins at $k = 0$ are also considered adjacent to the bins at $k = 7$ (b).

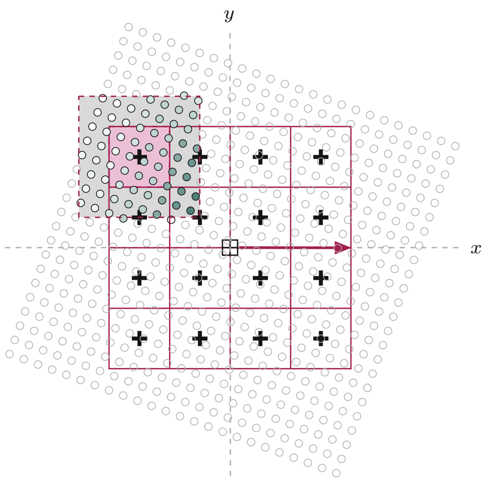


Fig. 25.27
Overlapping support regions in the gradient field. Due to the tri-linear interpolation used in the histogram calculation, the spatial regions associated with the cells of the orientation histogram h_∇ *overlap*. The shading of the circles indicates the weight w_G assigned to each sample by the Gaussian weighting function, whose value depends on the distance of each sample from the key point's center (see Eqn. (25.88)).

Initially, the 3D gradient histogram h_∇ (which contains continuous values) of size $\mathrm{n_{spat}} \times \mathrm{n_{spat}} \times \mathrm{n_{angl}}$ is flattened to a 1D vector $\boldsymbol{f}$ of length $\mathrm{n_{spat}^2} \cdot \mathrm{n_{angl}}$ (typ. 128), with

$$\boldsymbol{f}\big((i \cdot \mathrm{n_{spat}} + j) \cdot \mathrm{n_{angl}} + k\big) \;\leftarrow\; \mathsf{h}_\nabla(i, j, k), \tag{25.96}$$

for $i, j = 0, \ldots, \mathrm{n_{spat}}-1$ and $k = 0, \ldots, \mathrm{n_{angl}}-1$. The elements in $\boldsymbol{f}$ are thus arranged in the same order as shown in Fig. 25.25, with the orientation index k being the fastest moving and the spatial index i being the slowest (see Alg. 25.10, lines 3–8).[24]

Changes in image contrast have a linear impact upon the gradient magnitude and thus also upon the values of the feature vector $\boldsymbol{f}$. To eliminate these effects, the vector $\boldsymbol{f}$ is subsequently normalized to

$$\boldsymbol{f}(m) \;\leftarrow\; \frac{1}{\|\boldsymbol{f}\|} \cdot \boldsymbol{f}(m), \tag{25.97}$$

for all m, such that $\boldsymbol{f}$ has unit norm (see Alg. 25.10, line 9). Since the gradient is calculated from local pixel differences, changes in absolute

[24] Note that different ordering schemes for arranging the elements of the feature vector are used in various SIFT implementations. For successful matching, the ordering of the elements must be identical, of course.

brightness do not affect the gradient magnitude, unless saturation occurs. Such nonlinear illumination changes tend to produce peak gradient values, which are compensated for by clipping the values of $\boldsymbol{f}$ to a predefined maximum $\mathrm{t_{fclip}}$, that is,

$$\boldsymbol{f}(m) \leftarrow \min(\boldsymbol{f}(m), \mathrm{t_{fclip}}), \tag{25.98}$$

with typically $\mathrm{t_{fclip}} = 0.2$, as suggested in [153] (see Alg. 25.10, line 10). After this step, $\boldsymbol{f}$ is normalized once again, as in Eqn. (25.97). Finally, the real-valued feature vector $\boldsymbol{f}$ is converted to an integer vector by

$$\boldsymbol{f}_{\mathrm{sift}}(m) \leftarrow \min\bigl(\mathrm{round}(\mathsf{s}_{\mathrm{fscale}} \cdot \boldsymbol{f}(m)), 255)\bigr), \tag{25.99}$$

with $\mathsf{s}_{\mathrm{fscale}}$ being a predefined constant (typ. $\mathsf{s}_{\mathrm{fscale}} = 512$). The elements of $\boldsymbol{f}_{\mathrm{sift}}$ are in the range $[0, 255]$ to be conveniently encoded and stored as a byte sequence (see Alg. 25.10, line 12).

The final SIFT descriptor for a given key point $\boldsymbol{k}' = (p, q, x, y)$ is a tuple

$$\boldsymbol{s} = \langle x', y', \sigma, \theta, \boldsymbol{f}_{\mathrm{sift}} \rangle, \tag{25.100}$$

which contains the key point's interpolated position x', y' (in original image coordinates), the absolute scale σ, its dominant orientation θ, and the corresponding integer-valued gradient feature vector $\boldsymbol{f}_{\mathrm{sift}}$ (see Alg. 25.8, line 27). Remember that multiple SIFT descriptors may be produced for different dominant orientations located at the same key point position. These will have the same position and scale values but different θ and $\boldsymbol{f}_{\mathrm{sift}}$ data.

25.4 SIFT Algorithm Summary

This section contains a collection of algorithms that summarizes the SIFT feature extraction process described in the previous sections of this chapter.

Algorithm 25.3 shows the top-level procedure GetSiftFeatures(I), which returns a sequence of SIFT feature descriptors for the given image I. The remaining parts of Alg. 25.3 describe the key point detection as extrema of the DOG scale space. The refinement of key point positions is covered in Alg. 25.4. Algorithm 25.5 contains the procedures used for neighborhood operations, detecting local extrema, and the calculation of the gradient and Hessian matrix in 3D. Algorithm 25.6 covers the operations related to finding the dominant orientations at a given key point location, based on the orientation histogram that is calculated in Alg. 25.7. The final formation of the SIFT descriptors is described in Alg. 25.8, which is based on the procedures defined in Algs. 25.9 and 25.10. The global constants used throughout these algorithms are listed in Table 25.5, together with the corresponding Java identifiers in the associated source code (see Sec. 25.7).

Table 25.5
Predefined constants used in the SIFT algorithms (Algs. 25.3–25.11).

Scale space parameters

Symbol	*Java id.*	*Value*	*Description*
Q	`Q`	3	scale steps (levels) per octave
P	`P`	4	number of scale space octaves
σ_s	`sigma_s`	0.5	sampling scale (nominal smoothing of the input image)
σ_0	`sigma_0`	1.6	base scale of level 0 (base smoothing)

Key-point detection

Symbol	*Java id.*	*Value*	*Description*
$\mathrm{n_{orient}}$	`n_Orient`	36	number of orientation bins (angular resolution) used for calculating the dominant key point orientation
$\mathrm{n_{refine}}$	`n_Refine`	5	max. number of iterations for repositioning a key point
$\mathrm{n_{smooth}}$	`n_Smooth`	2	number of smoothing iterations applied to the orientation histogram
$\rho_{\max}$	`rho_Max`	10.0	max. ratio of principal curvatures $(3,\ldots,10)$
$\mathrm{t_{domor}}$	`t_DomOr`	0.8	min. value in orientation histogram for selecting dominant orientations (rel. to max. entry)
$\mathrm{t_{extrm}}$	`t_Extrm`	0.0	min. difference w.r.t. any neighbor for extrema detection
$\mathrm{t_{mag}}$	`t_Mag`	0.01	min. DoG magnitude for initial key point candidates
$\mathrm{t_{peak}}$	`t_Peak`	0.01	min. DoG magnitude at interpolated peaks

Feature descriptor

Symbol	*Java id.*	*Value*	*Description*
$\mathrm{n_{spat}}$	`n_Spat`	4	number of spatial descriptor bins along each x/y axis
$\mathrm{n_{angl}}$	`n_Angl`	16	number of angular descriptor bins
$\mathrm{s_d}$	`s_Desc`	10.0	spatial size factor of descriptor (relative to feature scale)
$\mathrm{s_{fscale}}$	`s_Fscale`	512.0	scale factor for converting normalized feature values to byte values in $[0, 255]$
$\mathrm{t_{fclip}}$	`t_Fclip`	0.2	max. value for clipping elements of normalized feature vectors

Feature matching

Symbol	*Java id.*	*Value*	*Description*
$\rho_{\max}$	`rho_ax`	0.8	max. ratio of best and second-best matching feature distance

25.5 Matching SIFT Features

Most applications of SIFT features aim at locating corresponding interest points in two or more images of the same scene, for example, for matching stereo pairs, panorama stitching, or feature tracking. Other applications like self-localization or object recognition might use a large database of model descriptors and the task is to match these to the SIFT features detected in a new image or video sequence. All these applications require possibly large numbers of pairs of SIFT features to be compared reliably and efficiently.

25.5.1 Feature Distance and Match Quality

In a typical situation, two sequences of SIFT features $S^{(a)}$ and $S^{(b)}$ are extracted independently from a pair of input images I_a, I_b, that is,

$$S^{(a)} = (\boldsymbol{s}_1^{(a)}, \boldsymbol{s}_2^{(a)}, \ldots, \boldsymbol{s}_{N_a}^{(a)}) \quad \text{and} \quad S^{(b)} = (\boldsymbol{s}_1^{(b)}, \boldsymbol{s}_2^{(b)}, \ldots, \boldsymbol{s}_{N_b}^{(b)}).$$

The goal is to find matching descriptors in the two feature sets. The similarity between a given pair of descriptors, $\boldsymbol{s}_i = \langle x_i, y_i, \sigma_i, \theta_i, \boldsymbol{f}_i\rangle$ and $\boldsymbol{s}_j = \langle x_j, y_j, \sigma_j, \theta_j, \boldsymbol{f}_j\rangle$, is measured by the *distance* between the corresponding feature vectors $\boldsymbol{f}_i, \boldsymbol{f}_j$, that is,

```
1:  GetSiftFeatures($I$)
        Input: $I$, the source image (scalar-valued).
        Returns a sequence of SIFT feature descriptors detected in $I$.
2:      $\langle \mathbf{G}, \mathbf{D} \rangle \leftarrow$ BuildSiftScaleSpace($I, \sigma_s, \sigma_0, P, Q$)      ▷ Alg. 25.2
3:      $C \leftarrow$ GetKeyPoints($\mathbf{D}$)
4:      $S \leftarrow ()$                                   ▷ empty list of SIFT descriptors
5:      for all $\boldsymbol{k}' \in C$ do                    ▷ $\boldsymbol{k}' = (p, q, x, y)$
6:          $A \leftarrow$ GetDominantOrientations($\mathbf{G}, \boldsymbol{k}'$)      ▷ Alg. 25.6
7:          for all $\theta \in A$ do
8:              $\boldsymbol{s} \leftarrow$ MakeSiftDescriptor($\mathbf{G}, \boldsymbol{k}', \theta$)      ▷ Alg. 25.8
9:              $S \leftarrow S \smile (\boldsymbol{s})$
10:     return $S$

11: GetKeypoints($\mathbf{D}$)
        $\mathbf{D}$: DoG scale space (with $P$ octaves, each containing $Q$ levels).
        Returns a set of key points located in $\mathbf{D}$.
12:     $C \leftarrow ()$                                   ▷ empty list of key points
13:     for $p \leftarrow 0, \ldots, P-1$ do                 ▷ for all octaves $p$
14:         for $q \leftarrow 0, \ldots, Q-1$ do             ▷ for all scale levels $q$
15:             $E \leftarrow$ FindExtrema($\mathbf{D}, p, q$)
16:             for all $\boldsymbol{k} \in E$ do            ▷ $\boldsymbol{k} = (p, q, u, v)$
17:                 $\boldsymbol{k}' \leftarrow$ RefineKeyPosition($\mathbf{D}, \boldsymbol{k}$)      ▷ Alg. 25.4
18:                 if $\boldsymbol{k}' \neq$ nil then        ▷ $\boldsymbol{k}' = (p, q, x, y)$
19:                     $C \leftarrow C \smile (\boldsymbol{k}')$       ▷ add refined key point $\boldsymbol{k}'$
20:     return $C$

21: FindExtrema($\mathbf{D}, p, q$)
22:     $\mathbf{D}_{p,q} \leftarrow$ GetScaleLevel($\mathbf{D}, p, q$)
23:     $(M, N) \leftarrow$ Size($\mathbf{D}_{p,q}$)
24:     $E \leftarrow ()$                                   ▷ empty list of extrema
25:     for $u \leftarrow 1, \ldots, M-2$ do
26:         for $v \leftarrow 1, \ldots, N-2$ do
27:             if $\left|\mathbf{D}_{p,q}(u, v)\right| > \mathrm{t}_{\mathrm{mag}}$ then
28:                 $\boldsymbol{k} \leftarrow (p, q, u, v)$
29:                 $\mathsf{N}_c \leftarrow$ GetNeighborhood($\mathbf{D}, \boldsymbol{k}$)      ▷ Alg. 25.5
30:                 if IsExtremum($\mathsf{N}_c$) then                ▷ Alg. 25.5
31:                     $E \leftarrow E \smile (\boldsymbol{k})$          ▷ add $\boldsymbol{k}$ to $E$
32:     return $E$
```

Alg. 25.3
SIFT feature extraction (part 1). Top-level SIFT procedure. Global parameters: σ_s, σ_0, $\mathrm{t}_{\mathrm{mag}}$, Q, P (see Table 25.5).

$$\mathrm{dist}(\boldsymbol{s}_i, \boldsymbol{s}_j) := \left\|\boldsymbol{f}_i - \boldsymbol{f}_j\right\|, \tag{25.101}$$

where $\|\cdots\|$ denotes an appropriate norm (typically Euclidean, alternatives will be discussed further).[25]

Note that this distance is measured between individual points distributed in a high-dimensional (typically 128-dimensional) vector space that is only sparsely populated. Since there is *always* a best-matching counterpart for a given descriptor, matches may occur between unrelated features even if the correct feature is not contained in the target set. This is particularly critical if feature matching is used to determine whether two images show any correspondence at all.

Obviously, significant matches should exhibit small feature distances but setting a *fixed limit* on the acceptable feature distance

[25] See also Sec. B.1.2 in the Appendix.

Alg. 25.4
SIFT feature extraction (part 2). Position refinement. Global parameters: $\mathrm{n}_{\mathrm{refine}}$, $\mathrm{t}_{\mathrm{peak}}$, $\rho_{\max}$ (see Table 25.5).

```
1:  RefineKeyPosition($\mathsf{D}, \boldsymbol{k}$)
        Input: $\mathsf{D}$, hierarchical DoG scale space; $\boldsymbol{k} = (p, q, u, v)$, candidate
        (extremal) position.
        Returns a refined key point $\boldsymbol{k}'$ or nil if no proper key point could
        be localized at or near the extremal position $\boldsymbol{k}$.
2:      $a_{\max} \leftarrow \frac{(\rho_{\max}+1)^2}{\rho_{\max}}$                          ▷ see Eq. 25.72
3:      $\boldsymbol{k}' \leftarrow$ nil                                    ▷ refined key point
4:      $done \leftarrow$ false
5:      $n \leftarrow 1$                              ▷ number of repositioning steps
6:      while $\neg done \wedge n \leq \mathrm{n}_{\mathrm{refine}} \wedge$ IsInside($\mathsf{D}, \boldsymbol{k}$) do
7:          $\mathsf{N}_{\boldsymbol{c}} \leftarrow$ GetNeighborhood($\mathsf{D}, \boldsymbol{k}$)                        ▷ Alg. 25.5
8:          $\nabla = \begin{pmatrix} d_x \\ d_x \\ d_\sigma \end{pmatrix} \leftarrow$ Gradient($\mathsf{N}_{\boldsymbol{c}}$)                    ▷ Alg. 25.5
9:          $\mathbf{H}_{\mathrm{D}} = \begin{pmatrix} d_{xx} & d_{xy} & d_{x\sigma} \\ d_{xy} & d_{yy} & d_{y\sigma} \\ d_{x\sigma} & d_{y\sigma} & d_{\sigma\sigma} \end{pmatrix} \leftarrow$ Hessian($\mathsf{N}_{\boldsymbol{c}}$)         ▷ Alg. 25.5
10:         if $\det(\mathbf{H}_{\mathrm{D}}) = 0$ then                        ▷ $\mathbf{H}_{\mathrm{D}}$ is not invertible
11:             $done \leftarrow$ true                    ▷ ignore this point and finish
12:         else
13:             $\boldsymbol{d} = \begin{pmatrix} x' \\ y' \\ \sigma' \end{pmatrix} \leftarrow -\mathbf{H}_{\mathrm{D}}^{-1} \cdot \nabla$                          ▷ Eq. 25.60
14:             if $|x'| < 0.5 \wedge |y'| < 0.5$ then ▷ stay in the same DoG cell
15:                 $done \leftarrow$ true
16:                 $D_{\mathrm{peak}} \leftarrow \mathsf{N}_{\boldsymbol{c}}(0,0,0) + \frac{1}{2} \cdot \nabla^{\mathsf{T}} \cdot \boldsymbol{d}$                ▷ Eq. 25.61
17:                 $\mathbf{H}_{\mathrm{xy}} \leftarrow \begin{pmatrix} d_{xx} & d_{xy} \\ d_{xy} & d_{yy} \end{pmatrix}$       ▷ extract 2D Hessian from $\mathbf{H}_{\mathrm{D}}$
18:                 if $|D_{\mathrm{peak}}| > \mathrm{t}_{\mathrm{peak}} \wedge \det(\mathbf{H}_{\mathrm{xy}}) > 0$ then
19:                     $a \leftarrow \frac{[\mathrm{trace}(\mathbf{H}_{\mathrm{xy}})]^2}{\det(\mathbf{H}_{\mathrm{xy}})}$                         ▷ Eq. 25.69
20:                     if $a < a_{\max}$ then       ▷ suppress edges, Eq. 25.72
21:                         $\boldsymbol{k}' \leftarrow \boldsymbol{k} + (0, 0, x', y')^{\mathsf{T}}$        ▷ refined key point
22:             else
                    Move to a neighboring DoG position at same level $p, q$:
23:                 $u' \leftarrow \min(1, \max(-1, \mathrm{round}(x')))$ ▷ move by max. $\pm 1$
24:                 $v' \leftarrow \min(1, \max(-1, \mathrm{round}(y')))$ ▷ move by max. $\pm 1$
25:                 $\boldsymbol{k} \leftarrow \boldsymbol{k} + (0, 0, u', v')^{\mathsf{T}}$
26:         $n \leftarrow n + 1$
27:     return $\boldsymbol{k}'$          ▷ $\boldsymbol{k}'$ is either a refined key point position or nil
```

turns out to be inappropriate in practice, since some descriptors are more discriminative than others. The solution proposed in [153] is to compare the distance obtained for the *best* feature match to that of the *second-best* match. For a given reference descriptor $\boldsymbol{s}_{\mathrm{r}} \in S^{(a)}$, the best match is defined as the descriptor $\boldsymbol{s}_1 \in S^{(b)}$ which has the smallest distance from $\boldsymbol{s}_{\mathrm{r}}$ in the multi-dimensional feature space, that is,

$$\boldsymbol{s}_1 = \operatorname*{argmin}_{\boldsymbol{s}_j \in S^{(b)}} \mathrm{dist}(\boldsymbol{s}_{\mathrm{r}}, \boldsymbol{s}_j), \qquad (25.102)$$

```
1:  IsInside($\mathsf{D}, \boldsymbol{k}$)
        Checks if coordinate $\boldsymbol{k} = (p,q,u,v)$ is inside the DoG scale space $\mathsf{D}$.
2:      $(p,q,u,v) \leftarrow \boldsymbol{k}$
3:      $(M,N) \leftarrow \mathsf{Size}(\mathsf{GetScaleLevel}(\mathsf{D},p,q))$
4:      return $(0 < u < M-1) \wedge (0 < v < N-1) \wedge (0 \leq q < Q)$

5:  GetNeighborhood($\mathsf{D}, \boldsymbol{k}$)                    ▷ $\boldsymbol{k} = (p,q,u,v)$
        Collects and returns the $3 \times 3 \times 3$ neighborhood values around
        position $\boldsymbol{k}$ in the hierarchical DoG scale space $\mathsf{D}$.
6:      Create map $\mathsf{N}_c : \{-1,0,1\}^3 \mapsto \mathbb{R}$
7:      for all $(i,j,k) \in \{-1,0,1\}^3$ do    ▷ collect $3{\times}3{\times}3$ neighborhood
8:          $\mathsf{N}_c(i,j,k) \leftarrow \mathsf{D}_{p,q+k}(u{+}i, v{+}j)$
9:      return $\mathsf{N}_c$

10: IsExtremum($\mathsf{N}_c$)                                ▷ $\mathsf{N}_c$ is a $3{\times}3{\times}3$ map
        Determines if the center of the 3D neighborhood $\mathsf{N}_c$ is either a
        local minimum or maximum by the threshold $\mathrm{t_{extrm}} \geq 0$. Returns
        a boolean value (i.e., true or false).
11:     $c \leftarrow \mathsf{N}_c(0,0,0)$                          ▷ center DoG value
12:     $isMin \leftarrow c < 0 \wedge (c + \mathrm{t_{extrm}}) < \min_{\substack{(i,j,k)\neq\\(0,0,0)}} \mathsf{N}_c(i,j,k)$      ▷ s. Eq. 25.54
13:     $isMax \leftarrow c > 0 \wedge (c - \mathrm{t_{extrm}}) > \max_{\substack{(i,j,k)\neq\\(0,0,0)}} \mathsf{N}_c(i,j,k)$      ▷ s. Eq. 25.55
14:     return $isMin \vee isMax$

15: Gradient($\mathsf{N}_c$)                                  ▷ $\mathsf{N}_c$ is a $3{\times}3{\times}3$ map
        Returns the estim. gradient vector ($\nabla$) for the 3D neighborhood
        $\mathsf{N}_c$.
16:     $d_{\mathrm{x}} \leftarrow 0.5 \cdot (\mathsf{N}_c(1,2,1) - \mathsf{N}_c(1,0,1))$
17:     $d_{\mathrm{y}} \leftarrow 0.5 \cdot (\mathsf{N}_c(1,1,2) - \mathsf{N}_c(1,1,0))$          ▷ see Eq. 25.56
18:     $d_{\sigma} \leftarrow 0.5 \cdot (\mathsf{N}_c(2,1,1) - \mathsf{N}_c(0,1,1))$
19:     $\nabla \leftarrow (d_{\mathrm{x}}, d_{\mathrm{y}}, d_{\sigma})^{\mathsf{T}}$
20:     return $\nabla$

21: Hessian($\mathsf{N}_c$)                                   ▷ $\mathsf{N}_c$ is a $3{\times}3{\times}3$ map
        Returns the estim. Hessian matrix ($\mathbf{H}$) for the neighborhood $\mathsf{N}_c$.
22:     $d_{xx} \leftarrow \mathsf{N}_c(-1,0,0) - 2{\cdot}\mathsf{N}_c(0,0,0) + \mathsf{N}_c(1,0,0)$    ▷ see Eq. 25.58
23:     $d_{yy} \leftarrow \mathsf{N}_c(0,-1,0) - 2{\cdot}\mathsf{N}_c(0,0,0) + \mathsf{N}_c(0,1,0)$
24:     $d_{\sigma\sigma} \leftarrow \mathsf{N}_c(0,0,-1) - 2{\cdot}\mathsf{N}_c(0,0,0) + \mathsf{N}_c(0,0,1)$
25:     $d_{xy} \leftarrow [\,\mathsf{N}_c(1,1,0) - \mathsf{N}_c(-1,1,0) - \mathsf{N}_c(1,-1,0) + \mathsf{N}_c(-1,-1,0)\,]/4$
26:     $d_{x\sigma} \leftarrow [\,\mathsf{N}_c(1,0,1) - \mathsf{N}_c(-1,0,1) - \mathsf{N}_c(1,0,-1) + \mathsf{N}_c(-1,0,-1)\,]/4$
27:     $d_{y\sigma} \leftarrow [\,\mathsf{N}_c(0,1,1) - \mathsf{N}_c(0,-1,1) - \mathsf{N}_c(0,1,-1) + \mathsf{N}_c(0,-1,-1)\,]/4$
28:     $\mathbf{H} \leftarrow \begin{pmatrix} d_{xx} & d_{xy} & d_{x\sigma} \\ d_{xy} & d_{yy} & d_{y\sigma} \\ d_{x\sigma} & d_{y\sigma} & d_{\sigma\sigma} \end{pmatrix}$
29:     return $\mathbf{H}$
```

Alg. 25.5
SIFT feature extraction (part 3): Neighborhood operations. Global parameters: Q, $\mathrm{t_{extrm}}$ (see Table 25.5).

and the primary distance is $d_{\mathrm{r},1} = \mathrm{dist}(\boldsymbol{s}_{\mathrm{r}}, \boldsymbol{s}_1)$. Analogously, the second-best matching descriptor is

$$\boldsymbol{s}_2 = \operatorname*{argmin}_{\substack{\boldsymbol{s}_j \in S^{(b)},\\ \boldsymbol{s}_j \neq \boldsymbol{s}_1}} \mathrm{dist}(\boldsymbol{s}_{\mathrm{r}}, \boldsymbol{s}_j), \tag{25.103}$$

and the corresponding distance is $d_{\mathrm{r},2} = \mathrm{dist}(\boldsymbol{s}_{\mathrm{r}}, \boldsymbol{s}_2)$, with $d_{\mathrm{r},1} \leq d_{\mathrm{r},2}$. Reliable matches are expected to have a distance to the primary

Alg. 25.6
SIFT feature extraction (part 4): Key point orientation assignment. Global parameters: $\mathrm{n_{smooth}}$, $\mathrm{t_{domor}}$ (see Table 25.5).

```
1: GetDominantOrientations($\mathsf{G}, \boldsymbol{k}'$)
       Input: $\mathsf{G}$, hierarchical Gaussian scale space; $\boldsymbol{k}' = (p, q, x, y)$, refined key point at octave $p$, scale level $q$ and spatial position $x, y$ (in octave's coordinates).
       Returns a list of dominant orientations for the key point $\boldsymbol{k}'$.
2:     $\mathsf{h}_\phi \leftarrow$ GetOrientationHistogram($\mathsf{G}, \boldsymbol{k}'$)          ▷ Alg. 25.7
3:     SmoothCircular($\mathsf{h}_\phi, \mathrm{n_{smooth}}$)
4:     $A \leftarrow$ FindPeakOrientations($\mathsf{h}_\phi$)
5:     return $A$

6: SmoothCircular($\boldsymbol{x}, n_{\mathrm{iter}}$)
       Smooths the real-valued vector $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ circularly using the 3-element kernel $H = (h_0, h_1, h_2)$, with $h_1$ as the hot-spot. The filter operation is applied $n_{\mathrm{iter}}$ times and "in place", i.e., the vector $\boldsymbol{x}$ is modified.
7:     $(h_0, h_1, h_2) \leftarrow \frac{1}{4} \cdot (1, \mathbf{2}, 1)$          ▷ 1D filter kernel
8:     $n \leftarrow$ Size($\boldsymbol{x}$)
9:     for $i \leftarrow 1, \ldots, n_{\mathrm{iter}}$ do
10:        $s \leftarrow \boldsymbol{x}(0)$
11:        $p \leftarrow \boldsymbol{x}(n-1)$
12:        for $j \leftarrow 0, \ldots, n-2$ do
13:            $c \leftarrow \boldsymbol{x}(j)$
14:            $\boldsymbol{x}(j) \leftarrow h_0 \cdot p + h_1 \cdot \boldsymbol{x}(j) + h_2 \cdot \boldsymbol{x}(j+1)$
15:            $p \leftarrow c$
16:        $\boldsymbol{x}(n-1) \leftarrow h_0 \cdot p + h_1 \cdot \boldsymbol{x}(n-1) + h_2 \cdot s$
17:    return

18: FindPeakOrientations($\mathsf{h}_\phi$)
       Returns a (possibly empty) sequence of dominant directions (angles) obtained from the orientation histogram $\mathsf{h}_\phi$.
19:    $n \leftarrow$ Size($\mathsf{h}_\phi$)
20:    $A \leftarrow (\,)$
21:    $h_{\max} \leftarrow \max_{0 \le i < n} \mathsf{h}_\phi(i)$
22:    for $k \leftarrow 0, \ldots, n-1$ do
23:        $h_\mathrm{c} \leftarrow \mathsf{h}(k)$
24:        if $h_\mathrm{c} > \mathrm{t_{domor}} \cdot h_{\max}$ then          ▷ only accept dominant peaks
25:            $h_\mathrm{p} \leftarrow \mathsf{h}_\phi((k-1) \bmod n)$
26:            $h_\mathrm{n} \leftarrow \mathsf{h}_\phi((k+1) \bmod n)$
27:            if $(h_\mathrm{c} > h_\mathrm{p}) \wedge (h_\mathrm{c} > h_\mathrm{n})$ then          ▷ local max. at index $k$
28:                $\breve{k} \leftarrow k + \frac{h_\mathrm{p} - h_\mathrm{n}}{2 \cdot (h_\mathrm{p} - 2 \cdot h_\mathrm{c} + h_\mathrm{n})}$          ▷ quadr. interpol., Eq. 25.85
29:                $\theta \leftarrow \left(\breve{k} \cdot \frac{2\pi}{n}\right) \bmod 2\pi$          ▷ domin. orientation, Eq. 25.86
30:                $A \leftarrow A \smile (\theta)$
31:    return $A$
```

feature $\boldsymbol{s}_1$ that is considerably smaller than the distance to any other feature in the target set. In the case of a weak or ambiguous match, on the other hand, it is likely that other matches exist at a distance similar to $d_{\mathrm{r},1}$, including the second-best match $\boldsymbol{s}_2$. Comparing the best and the second-best distances thus provides information about the likelihood of a false match. For this purpose, we define the *feature distance ratio*

$$\rho_{\mathrm{match}}(\boldsymbol{s}_\mathrm{r}, \boldsymbol{s}_1, \boldsymbol{s}_2) := \frac{d_{\mathrm{r},1}}{d_{\mathrm{r},2}} = \frac{\mathrm{dist}(\boldsymbol{s}_\mathrm{r}, \boldsymbol{s}_1)}{\mathrm{dist}(\boldsymbol{s}_\mathrm{r}, \boldsymbol{s}_2)}, \tag{25.104}$$

Alg. 25.7
SIFT feature extraction (part 5): Calculation of the orientation histogram and gradients from Gaussian scale levels. Global parameters: $\mathrm{n}_{\mathrm{orient}}$ (see Table 25.5).

```
1:  GetOrientationHistogram($\mathbf{G}, \boldsymbol{k}'$)
        Input: $\mathbf{G}$, hierarchical Gaussian scale space; $\boldsymbol{k}' = (p, q, x, y)$, refined key point at octave $p$, scale level $q$ and relative position $x, y$.
        Returns the gradient orientation histogram for key point $\boldsymbol{k}'$.
2:      $\mathbf{G}_{p,q} \leftarrow$ GetScaleLevel($\mathbf{G}, p, q$)
3:      $(M, N) \leftarrow$ Size($\mathbf{G}_{p,q}$)
4:      Create a new map $\mathsf{h}_\phi : [0, \mathrm{n}_{\mathrm{orient}}-1] \mapsto \mathbb{R}$.   ▷ new histogram $\mathsf{h}_\phi$
5:      for $i \leftarrow 0, \ldots, \mathrm{n}_{\mathrm{orient}}-1$ do                    ▷ initialize $\mathsf{h}_\phi$ to zero
6:          $\mathsf{h}_\phi(i) \leftarrow 0$
7:      $\sigma_{\mathrm{w}} \leftarrow 1.5 \cdot \sigma_0 \cdot 2^{q/Q}$      ▷ $\sigma$ of Gaussian weight fun., see Eq. 25.76
8:      $r_{\mathrm{w}} \leftarrow \max(1, 2.5 \cdot \sigma_{\mathrm{w}})$          ▷ rad. of weight fun., see Eq. 25.77
9:      $u_{\min} \leftarrow \max(\lfloor x - r_{\mathrm{w}} \rfloor, 1)$
10:     $u_{\max} \leftarrow \min(\lceil x + r_{\mathrm{w}} \rceil, M-2)$
11:     $v_{\min} \leftarrow \max(\lfloor y - r_{\mathrm{w}} \rfloor, 1)$
12:     $v_{\max} \leftarrow \min(\lceil y + r_{\mathrm{w}} \rceil, N-2)$
13:     for $u \leftarrow u_{\min}, \ldots, u_{\max}$ do
14:         for $v \leftarrow v_{\min}, \ldots, v_{\max}$ do
15:             $r^2 \leftarrow (u-x)^2 + (v-y)^2$
16:             if $r^2 < r_{\mathrm{w}}^2$ then
17:                 $(E, \phi) \leftarrow$ GetGradientPolar($\mathbf{G}_{p,q}, u, v$)      ▷ see below
18:                 $w_{\mathrm{G}} \leftarrow \exp\left(-\frac{(u-x)^2+(v-y)^2}{2\sigma_{\mathrm{w}}^2}\right)$       ▷ Gaussian weight
19:                 $z \leftarrow E \cdot w_{\mathrm{G}}$                        ▷ quantity to accumulate
20:                 $\kappa_\phi \leftarrow \frac{\mathrm{n}_{\mathrm{orient}}}{2\pi} \cdot \phi$             ▷ $\kappa_\phi \in [-\frac{\mathrm{n}_{\mathrm{orient}}}{2}, +\frac{\mathrm{n}_{\mathrm{orient}}}{2}]$
21:                 $\alpha \leftarrow \kappa_\phi - \lfloor \kappa_\phi \rfloor$                            ▷ $\alpha \in [0, 1]$
22:                 $k_0 \leftarrow \lfloor \kappa_\phi \rfloor \bmod \mathrm{n}_{\mathrm{orient}}$                 ▷ lower bin index
23:                 $k_1 \leftarrow (k_0 + 1) \bmod \mathrm{n}_{\mathrm{orient}}$              ▷ upper bin index
24:                 $\mathsf{h}_\phi(k_0) \overset{+}{\leftarrow} (1-\alpha) \cdot z$                    ▷ update bin $k_0$
25:                 $\mathsf{h}_\phi(k_1) \overset{+}{\leftarrow} \alpha \cdot z$                          ▷ update bin $k_1$
26:     return $\mathsf{h}_\phi$

27: GetGradientPolar($\mathbf{G}_{p,q}, u, v$)
        Returns the gradient magnitude ($E$) and orientation ($\phi$) at position $(u, v)$ of the Gaussian scale level $\mathbf{G}_{p,q}$.
28:     $\begin{pmatrix} d_{\mathrm{x}} \\ d_{\mathrm{y}} \end{pmatrix} \leftarrow 0.5 \cdot \begin{pmatrix} \mathbf{G}_{p,q}(u+1, v) - \mathbf{G}_{p,q}(u-1, v) \\ \mathbf{G}_{p,q}(u, v+1) - \mathbf{G}_{p,q}(u, v-1) \end{pmatrix}$   ▷ gradient at $u, v$
29:     $E \leftarrow \left(d_{\mathrm{x}}^2 + d_{\mathrm{y}}^2\right)^{1/2}$                              ▷ gradient magnitude
30:     $\phi \leftarrow$ ArcTan($d_{\mathrm{x}}, d_{\mathrm{y}}$)        ▷ gradient orientation ($-\pi \leq \phi \leq \pi$)
31:     return $(E, \phi)$
```

such that $\rho_{\mathrm{match}} \in [0, 1]$. If the distance $d_{\mathrm{r},1}$ between $\boldsymbol{s}_{\mathrm{r}}$ and the primary feature $\boldsymbol{s}_1$ is small compared to the secondary distance $d_{\mathrm{r},2}$, then the value of ρ_{match} is small as well. Thus, large values of ρ_{match} indicate that the corresponding match (between $\boldsymbol{s}_{\mathrm{r}}$ and $\boldsymbol{s}_1$) is likely to be weak or ambiguous. Matches are only accepted if they are sufficiently distinctive, for example, by enforcing the condition

$$\rho_{\mathrm{match}}(\boldsymbol{s}_{\mathrm{r}}, \boldsymbol{s}_1, \boldsymbol{s}_2) \leq \rho_{\max}\,, \tag{25.105}$$

where $\rho_{\max} \in [0, 1]$ is a predefined constant (see Table 25.5). The complete matching process, using the Euclidean distance norm and sequential search, is summarized in Alg. 25.11. Other common options for distance measurement are the L_1 and L_∞ norms.

Alg. 25.8
SIFT feature extraction (part 6): Calculation of SIFT descriptors. Global parameters: Q, σ_0, s_d, n_spat, n_angl (see Table 25.5).

```
1:  MakeSiftDescriptor($\mathbf{G}, \boldsymbol{k}', \theta$)
        Input: $\mathbf{G}$, hierarchical Gaussian scale space; $\boldsymbol{k}' = (p, q, x, y)$, refined key point; $\theta$, dominant orientation.
        Returns a new SIFT descriptor for the key point $\boldsymbol{k}'$.
2:      $\mathbf{G}_{p,q} \leftarrow$ GetScaleLevel($\mathbf{G}, p, q$)
3:      $(M, N) \leftarrow$ Size($\mathbf{G}_{p,q}$)
4:      $\dot{\sigma}_q \leftarrow \sigma_0 \cdot 2^{q/Q}$                ▷ decimated scale at level $q$
5:      $w_\mathrm{d} \leftarrow \mathsf{s}_\mathrm{d} \cdot \dot{\sigma}_q$      ▷ descriptor size is prop. to key point scale
6:      $\sigma_\mathrm{d} \leftarrow 0.25 \cdot w_\mathrm{d}$            ▷ width of Gaussian weighting function
7:      $r_\mathrm{d} \leftarrow 2.5 \cdot \sigma_\mathrm{d}$             ▷ cutoff radius of weighting function
8:      $u_\mathrm{min} \leftarrow \max(\lfloor x - r_\mathrm{d} \rfloor, 1)$
9:      $u_\mathrm{max} \leftarrow \min(\lceil x + r_\mathrm{d} \rceil, M-2)$
10:     $v_\mathrm{min} \leftarrow \max(\lfloor y - r_\mathrm{d} \rfloor, 1)$
11:     $v_\mathrm{max} \leftarrow \min(\lceil y + r_\mathrm{d} \rceil, N-2)$
12:     Create map $\mathsf{h}_\nabla : \mathrm{n}_\mathrm{spat} \times \mathrm{n}_\mathrm{spat} \times \mathrm{n}_\mathrm{angl} \mapsto \mathbb{R}$     ▷ gradient histogram $\mathsf{h}_\nabla$
13:     for all $(i, j, k) \in \mathrm{n}_\mathrm{spat} \times \mathrm{n}_\mathrm{spat} \times \mathrm{n}_\mathrm{angl}$ do
14:         $\mathsf{h}_\nabla(i, j, k) \leftarrow 0$                 ▷ initialize $\mathsf{h}_\nabla$ to zero
15:     for $u \leftarrow u_\mathrm{min}, \ldots, u_\mathrm{max}$ do
16:         for $v \leftarrow v_\mathrm{min}, \ldots, v_\mathrm{max}$ do
17:             $r^2 \leftarrow (u-x)^2 + (v-y)^2$
18:             if $r^2 < r_\mathrm{d}^2$ then
                    Map to canonical coord. frame, with $u', v' \in [-\frac{1}{2}, +\frac{1}{2}]$:
19:                 $\begin{pmatrix} u' \\ v' \end{pmatrix} \leftarrow \frac{1}{w_\mathrm{d}} \cdot \begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix} \cdot \begin{pmatrix} u-x \\ v-y \end{pmatrix}$
20:                 $(E, \phi) \leftarrow$ GetGradientPolar($\mathbf{G}_{p,q}, u, v$)       ▷ Alg. 25.7
21:                 $\phi' \leftarrow (\phi - \theta) \bmod 2\pi$        ▷ normalize gradient angle
22:                 $w_\mathrm{G} \leftarrow \exp\left(-\frac{r^2}{2\sigma_\mathrm{d}^2}\right)$              ▷ Gaussian weight
23:                 $z \leftarrow E \cdot w_\mathrm{G}$                  ▷ quantity to accumulate
24:                 UpdateGradientHistogram($\mathsf{h}_\nabla, u', v', \phi', z$)   ▷ Alg. 25.9
25:     $\boldsymbol{f}_\mathrm{sift} \leftarrow$ MakeFeatureVector($\mathsf{h}_\nabla$)          ▷ see Alg. 25.10
26:     $\sigma \leftarrow \sigma_0 \cdot 2^{p+q/Q}$              ▷ absolute scale, Eq. 25.35
27:     $\begin{pmatrix} x' \\ y' \end{pmatrix} \leftarrow 2^p \cdot \begin{pmatrix} x \\ y \end{pmatrix}$             ▷ real position, Eq. 25.45
28:     $\boldsymbol{s} \leftarrow \langle x', y', \sigma, \theta, \boldsymbol{f}_\mathrm{sift} \rangle$    ▷ create a new SIFT descriptor
29:     return $\boldsymbol{s}$
```

25.5.2 Examples

The following examples were calculated on pairs of stereographic images taken at the beginning of the 20th century.[26] From each of the two frames of a stereo picture, a sequence of (ca. 1000) SIFT descriptors (marked by blue rectangles) was extracted with identical parameter settings. Matching was done by enumerating all possible descriptor pairs from the left and the right image, calculating their (Euclidean) distance, and showing the 25 closest matches obtained from ca. 1000 detected key points in each frame. Only the

[26] The images used in Figs. 25.28–25.31 are historic stereographs made publicly available by the *Library of Congress* (www.loc.gov).

Alg. 25.9
SIFT feature extraction (part 7): Updating the gradient descriptor histogram. The quantity z pertaining to the continuous position (u', v', ϕ') is to be accumulated into the 3D histogram h_∇ (u', v' are normalized spatial coordinates, ϕ' is the orientation). The quantity z is distributed over up to eight neighboring histogram bins (see Fig. 25.26) by tri-linear interpolation. Note that the orientation coordinate ϕ' receives special treatment because it is circular. Global parameters: $\mathrm{n_{spat}}$, $\mathrm{n_{angl}}$ (see Table 25.5).

```
1:  UpdateGradientHistogram($\mathsf{h}_\nabla, u', v', \phi', z$)
        Input: $\mathsf{h}_\nabla$, gradient histogram of size $\mathrm{n_{spat}} \times \mathrm{n_{spat}} \times \mathrm{n_{angl}}$, with
        $\mathsf{h}_\nabla(i,j,k) \in \mathbb{R}$; $u', v' \in [-0.5, 0.5]$, normalized spatial position;
        $\phi' \in [0, 2\pi)$, normalized gradient orientation; $z \in \mathbb{R}$, quantity to
        be accumulated into $\mathsf{h}_\nabla$.
        Returns nothing but modifies the histogram $\mathsf{h}_\nabla$.
2:      $i' \leftarrow \mathrm{n_{spat}} \cdot u' + 0.5 \cdot (\mathrm{n_{spat}} - 1)$                ▷ see Eq. 25.92
3:      $j' \leftarrow \mathrm{n_{spat}} \cdot v' + 0.5 \cdot (\mathrm{n_{spat}} - 1)$       ▷ $-0.5 \leq i', j' \leq \mathrm{n_{spat}} - 0.5$
4:      $k' \leftarrow \mathrm{n_{angl}} \cdot \frac{\phi'}{2\pi}$                         ▷ $-\frac{\mathrm{n_{angl}}}{2} \leq k' \leq \frac{\mathrm{n_{angl}}}{2}$
5:      $i_0 \leftarrow \lfloor i' \rfloor$
6:      $i_1 \leftarrow i_0 + 1$
7:      $\mathbf{i} \leftarrow (i_0, i_1)$                    ▷ see Eq. 25.93; $\mathbf{i}(0) = i_0$, $\mathbf{i}(1) = i_1$
8:      $j_0 \leftarrow \lfloor j' \rfloor$
9:      $j_1 \leftarrow j_0 + 1$
10:     $\mathbf{j} \leftarrow (j_0, j_1)$                              ▷ $\mathbf{j}(0) = j_0$, $\mathbf{j}(1) = j_1$
11:     $k_0 \leftarrow \lfloor k' \rfloor \bmod \mathrm{n_{angl}}$
12:     $k_1 \leftarrow (k_0 + 1) \bmod \mathrm{n_{angl}}$
13:     $\mathbf{k} \leftarrow (k_0, k_1)$                              ▷ $\mathbf{k}(0) = k_0$, $\mathbf{k}(1) = k_1$
14:     $\alpha_0 \leftarrow i_1 - i'$                                   ▷ see Eq. 25.94
15:     $\alpha_1 \leftarrow 1 - \alpha_0$
16:     $A \leftarrow (\alpha_0, \alpha_1)$                            ▷ $A(0) = \alpha_0$, $A(1) = \alpha_1$
17:     $\beta_0 \leftarrow j_1 - j'$
18:     $\beta_1 \leftarrow 1 - \beta_0$
19:     $B \leftarrow (\beta_0, \beta_1)$                              ▷ $B(0) = \beta_0$, $B(1) = \beta_1$
20:     $\gamma_0 \leftarrow 1 - (k' - \lfloor k' \rfloor)$
21:     $\gamma_1 \leftarrow 1 - \gamma_0$
22:     $C \leftarrow (\gamma_0, \gamma_1)$                            ▷ $C(0) = \gamma_0$, $C(1) = \gamma_1$
        Distribute quantity $z$ among (up to) 8 adjacent histogram bins:
23:     for all $a \in \{0, 1\}$ do
24:         $i \leftarrow \mathbf{i}(a)$
25:         if $(0 \leq i < \mathrm{n_{spat}})$ then
26:             $w_a \leftarrow A(a)$
27:             for all $b \in \{0, 1\}$ do
28:                 $j \leftarrow \mathbf{j}(b)$
29:                 if $(0 \leq j < \mathrm{n_{spat}})$ then
30:                     $w_b \leftarrow B(b)$
31:                     for all $c \in \{0, 1\}$ do
32:                         $k \leftarrow \mathbf{k}(c)$
33:                         $w_c \leftarrow C(c)$
34:                         $\mathsf{h}_\nabla(i, j, k) \overset{+}{\leftarrow} z \cdot w_a \cdot w_b \cdot w_c$      ▷ see Eq. 25.95
35:     return
```

best 25 matches are shown in the examples. Feature matches are numbered according to their goodness, that is, label "1" denotes the best-matching descriptor pair (with the smallest feature distance). Selected details from these results are shown in Fig. 25.29. Unless otherwise noted, all SIFT parameters are set to their default values (see Table 25.5).

Although the use of the Euclidean (L_2) norm for measuring the distances between feature vectors in Eqn. (25.101) is suggested in [153], other norms have been considered [130, 181, 227] to improve

Alg. 25.10
SIFT feature extraction (part 8): Converting the orientation histogram to a SIFT feature vector. Global parameters: n_{spat}, n_{angl}, t_{fclip}, s_{fscale} (see Table 25.5).

```
1:  MakeSiftFeatureVector($h_\nabla$)
        Input: $h_\nabla$, gradient histogram of size $n_{spat} \times n_{spat} \times n_{angl}$.
        Returns a 1D integer (unsigned byte) vector obtained from $h_\nabla$.
2:      Create map $\boldsymbol{f} : [0, n_{spat}^2 \cdot n_{angl} - 1] \mapsto \mathbb{R}$      ▷ new 1D vector $\boldsymbol{f}$
3:      $m \leftarrow 0$
4:      for $i \leftarrow 0, \ldots, n_{spat} - 1$ do                ▷ flatten $h_\nabla$ into $\boldsymbol{f}$
5:          for $j \leftarrow 0, \ldots, n_{spat} - 1$ do
6:              for $k \leftarrow 0, \ldots, n_{angl} - 1$ do
7:                  $\boldsymbol{f}(m) \leftarrow h_\nabla(i, j, k)$
8:                  $m \leftarrow m + 1$
9:      Normalize($\boldsymbol{f}$)
10:     ClipPeaks($\boldsymbol{f}, t_{fclip}$)
11:     Normalize($\boldsymbol{f}$)
12:     $\boldsymbol{f}_{sift} \leftarrow$ MapToBytes($\boldsymbol{f}, s_{fscale}$)
13:     return $\boldsymbol{f}_{sift}$

14: Normalize($\boldsymbol{x}$)
        Scales vector $\boldsymbol{x}$ to unit norm. Returns nothing, but $\boldsymbol{x}$ is modified.
15:     $n \leftarrow$ Size($\boldsymbol{x}$)
16:     $s \leftarrow \sum_{i=0}^{n-1} \boldsymbol{x}(i)$
17:     for $i \leftarrow 0, \ldots, n-1$ do
18:         $\boldsymbol{x}(i) \leftarrow \frac{1}{s} \cdot \boldsymbol{x}(i)$
19:     return

20: ClipPeaks($\boldsymbol{x}, x_{max}$)
        Limits the elements of $\boldsymbol{x}$ to $x_{max}$. Returns nothing, but $\boldsymbol{x}$ is
        modified.
21:     $n \leftarrow$ Size($\boldsymbol{x}$)
22:     for $i \leftarrow 0, \ldots, n-1$ do
23:         $\boldsymbol{x}(i) \leftarrow \min\big(\boldsymbol{x}(i), x_{max}\big)$
24:     return

25: MapToBytes($\boldsymbol{x}, s$)
        Converts the real-valued vector $\boldsymbol{x}$ to an integer (unsigned byte)
        valued vector with elements in $[0, 255]$, using the scale factor
        $s > 0$.
26:     $n \leftarrow$ Size($\boldsymbol{x}$)
27:     Create a new map $\boldsymbol{x}_{int} : [0, n-1] \mapsto [0, 255]$      ▷ new byte vector
28:     for $i \leftarrow 0, \ldots, n-1$ do
29:         $a \leftarrow$ round$(s \cdot \boldsymbol{x}(i))$                       ▷ $a \in \mathbb{N}_0$
30:         $\boldsymbol{x}_{int}(i) \leftarrow \min\big(a, 255\big)$              ▷ $\boldsymbol{x}_{int}(i) \in [0, 255]$
31:     return $\boldsymbol{x}_{int}$
```

the statistical robustness and noise resistance. In Fig. 25.30, matching results are shown using the L_1, L_2, and L_∞ norms, respectively. Note that the resulting sets of top-ranking matches are almost the same with different distance norms, but the ordering of the strongest matches does change.

Figure 25.31 demonstrates the effectiveness of selecting feature matches based on the ratio between the distances to the best and the second-best match (see Eqns. (25.102)–(25.103)). Again the figure shows the 25 top-ranking matches based on the minimum (L_2) feature distance. With the maximum distance ratio ρ_{max} set to 1.0, rejection is practically turned off with the result that several false or ambiguous matches are among the top-ranking feature matches (Fig. 25.31(a)).

```
1: MatchDescriptors($S^{(a)}, S^{(b)}, \rho_{\max}$)
     Input: $S^{(a)}$, $S^{(b)}$, two sets of SIFT descriptors; $\rho_{\max}$, max. ratio
     of best and second-best matching distance (s. Eq. 25.105).
     Returns a sorted list of matches $\boldsymbol{m}_{ij} = \langle \boldsymbol{s}_a, \boldsymbol{s}_b, d_{ij} \rangle$, with $\boldsymbol{s}_a \in S^{(a)}$, $\boldsymbol{s}_b \in S^{(b)}$
     and $d_{ij}$ being the distance between $\boldsymbol{s}_a, \boldsymbol{s}_b$ in feature space.
2:   M ← ( )                                        ▷ empty sequence of matches
3:   for all $\boldsymbol{s}_a \in S^{(a)}$ do
4:     $\boldsymbol{s}_1 \leftarrow$ nil,   $d_{\mathrm{r},1} \leftarrow \infty$          ▷ best nearest neighbor
5:     $\boldsymbol{s}_2 \leftarrow$ nil,   $d_{\mathrm{r},2} \leftarrow \infty$          ▷ second-best nearest neighbor
6:     for all $\boldsymbol{s}_b \in S^{(b)}$ do
7:       $d \leftarrow$ Dist($\boldsymbol{s}_a, \boldsymbol{s}_b$)
8:       if $d < d_{\mathrm{r},1}$ then                        ▷ $d$ is a new 'best' distance
9:         $\boldsymbol{s}_2 \leftarrow \boldsymbol{s}_1$,   $d_{\mathrm{r},2} \leftarrow d_{\mathrm{r},1}$
10:        $\boldsymbol{s}_1 \leftarrow \boldsymbol{s}_b$,   $d_{\mathrm{r},1} \leftarrow d$
11:      else
12:        if $d < d_{\mathrm{r},2}$ then      ▷ $d$ is a new 'second-best' distance
13:          $\boldsymbol{s}_2 \leftarrow \boldsymbol{s}_b$,   $d_{\mathrm{r},2} \leftarrow d$
14:    if $(\boldsymbol{s}_2 \neq \mathrm{nil}) \wedge (\frac{d_{\mathrm{r},1}}{d_{\mathrm{r},2}} \leq \rho_{\max})$ then     ▷ Eqns. (25.104–25.105)
15:      $\boldsymbol{m} \leftarrow \langle \boldsymbol{s}_a, \boldsymbol{s}_1, d_{\mathrm{r},1} \rangle$                      ▷ add a new match
16:      M ⌣ ($\boldsymbol{m}$)
17:  Sort(M)                              ▷ sort M to ascending distance $d_{\mathrm{r},1}$
18:  return M

19: Dist($\boldsymbol{s}_a, \boldsymbol{s}_b$)
     Input: descriptors $\boldsymbol{s}_a = \langle x_a, y_a, \sigma_a, \theta_a, \boldsymbol{f}_a \rangle$, $\boldsymbol{s}_b = \langle x_b, y_b, \sigma_b, \theta_b, \boldsymbol{f}_b \rangle$.
     Returns the Euclidean distance between feature vectors $\boldsymbol{f}_a$ and $\boldsymbol{f}_b$.
20:  $d \leftarrow \|\boldsymbol{f}_a - \boldsymbol{f}_b\|$
21:  return $d$
```

Alg. 25.11
SIFT feature matching using Euclidean feature distance and linear search. The returned sequence of SIFT matches is sorted to ascending distance between corresponding feature pairs. Function Dist($\boldsymbol{s}_a, \boldsymbol{s}_b$) demonstrates the calculation of the Euclidean (L_2) feature distance, other options are the L_1 and L_∞ norms.

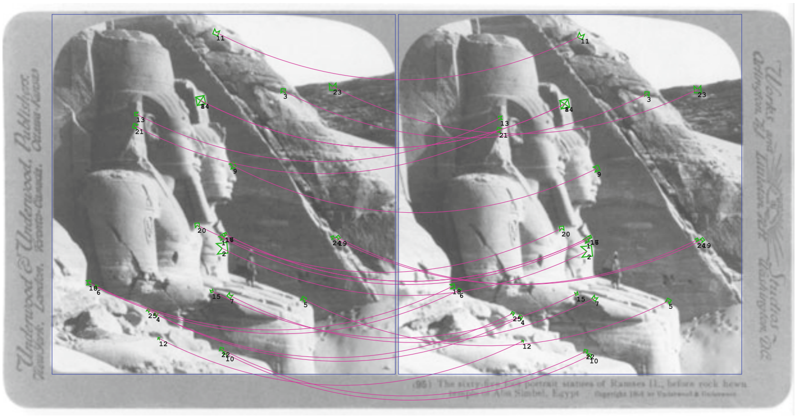
With $\rho_{\max}$ set to 0.8 and finally 0.5, the number of false matches is effectively reduced (Fig. 25.31(b, c)).[27]

25.6 Efficient Feature Matching

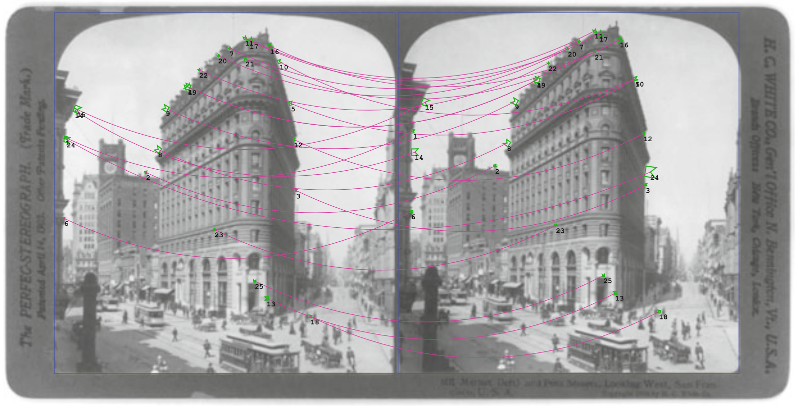
The task of finding the best match based on the minimum distance in feature space is called "nearest-neighbor" search. If performed exhaustively, evaluating all possible matches between two descriptor sets $S^{(a)}$ and $S^{(b)}$ of size N_a and N_b, respectively, requires $N_a \cdot N_b$ feature distance calculations and comparisons. While this may be acceptable for small feature sets (with maybe up to 1000 descriptors each), this linear (brute-force) approach becomes prohibitively expensive for large feature sets with possibly millions of candidates, as required, for example, in the context of image database indexing or robot self-localization. Although efficient methods for exact nearest-neighbor search based on tree structures exist, such as the k-d tree method [80], it has been shown that these methods lose their effectiveness with increasing dimensionality of the search space.

[27] $\rho_{\max} = 0.8$ is recommended in [153].

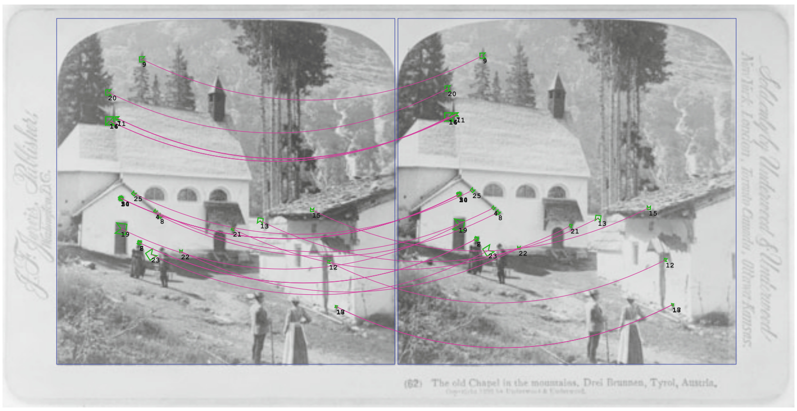
Fig. 25.28
SIFT feature matching examples on pairs of stereo images. Shown are the 25 best matches obtained with the L_2 feature distance and $\rho_{\max} = 0.8$.



(a)



(b)



(c)

In fact, no algorithms are known that significantly outperform exhaustive (linear) nearest neighbor search in feature spaces that are more than about 10-dimensional [153]. SIFT feature vectors are 128-dimensional and therefore exact nearest-neighbor search is not a viable option for efficient matching between large descriptor sets.

The approach taken in [21, 153] abandons exact nearest-neighbor search in favor of finding an *approximate* solution with substantially reduced effort, based on ideas described in [9]. This so-called

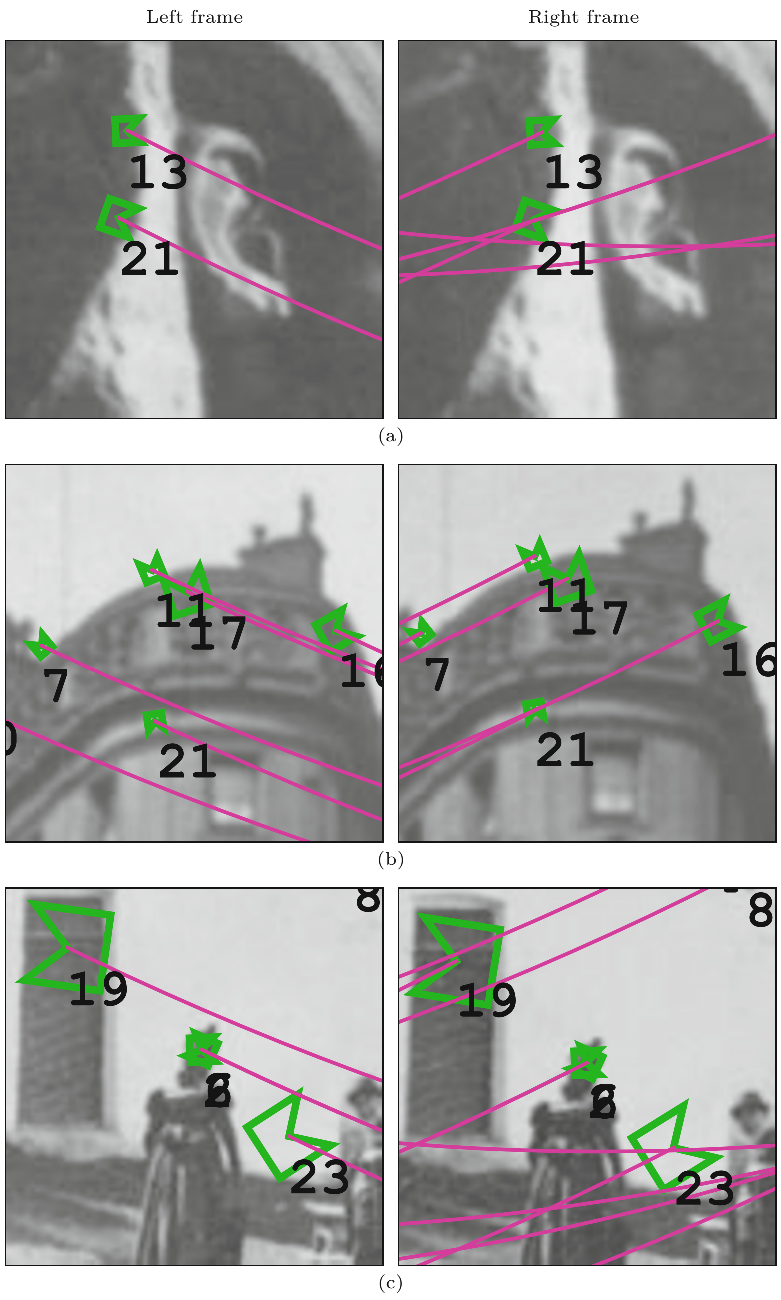
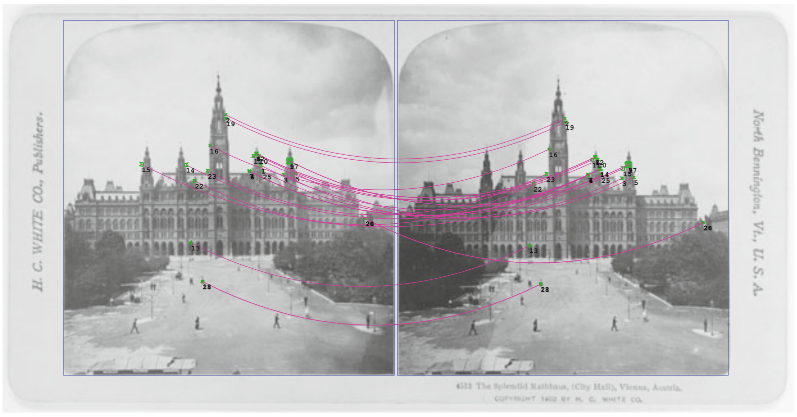


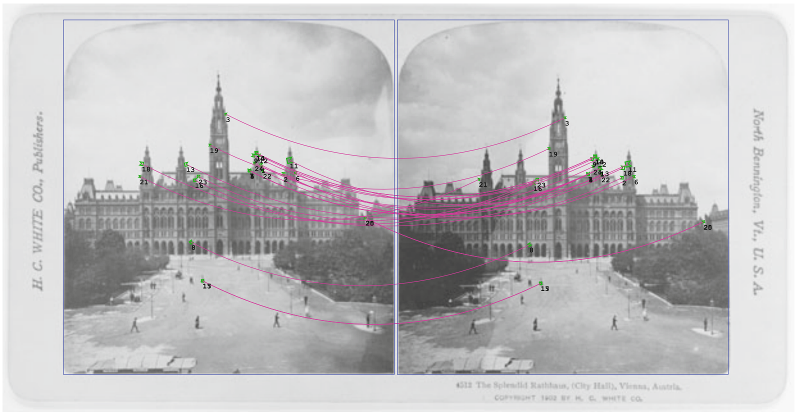
Fig. 25.29
Stereo matching examples (enlarged details from Fig. 25.28).

"best-bin-first" method uses a modified k-d algorithm, which searches neighboring feature space partitions in the order of their closest distance from the given feature vector. To limit the exploration to a small fraction of the feature space, the search is cut off after checking the first 200 candidates, which results in a substantial speedup without compromising the search results, particularly when combined with feature selection based on the ratio of primary and secondary distances (see Eqns. (25.104)–(25.105)). Additional details can be found in [21].

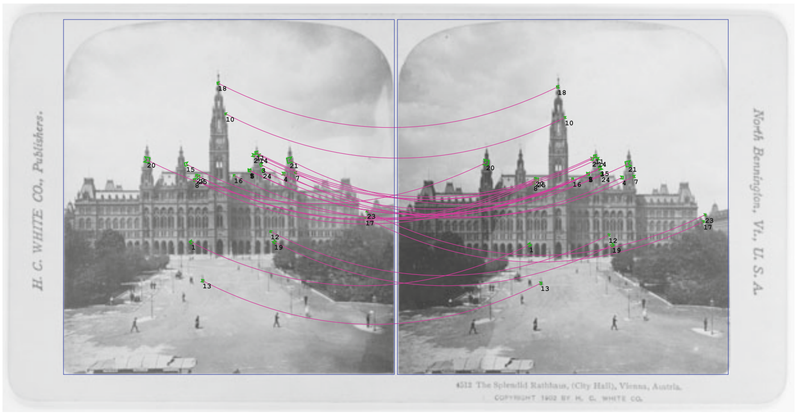
Fig. 25.30
Using different distance norms for feature matching. L_1 (a), L_2 (b), and L_∞ norm (c). All other parameters are set to their default values (see Table 25.5).


H. C. WHITE CO., Publishers.
North Bennington, Vt., U. S. A.
4512 The Splendid Rathhaus, (City Hall), Vienna, Austria.
COPYRIGHT 1902 BY H. C. WHITE CO.

(a) L_1-norm

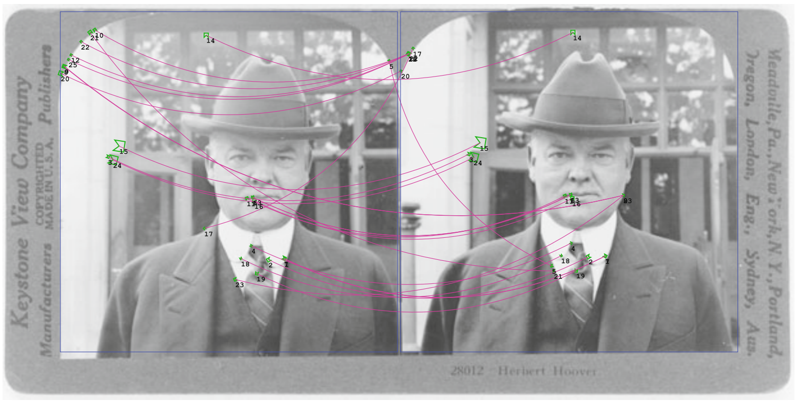


(b) L_2-norm

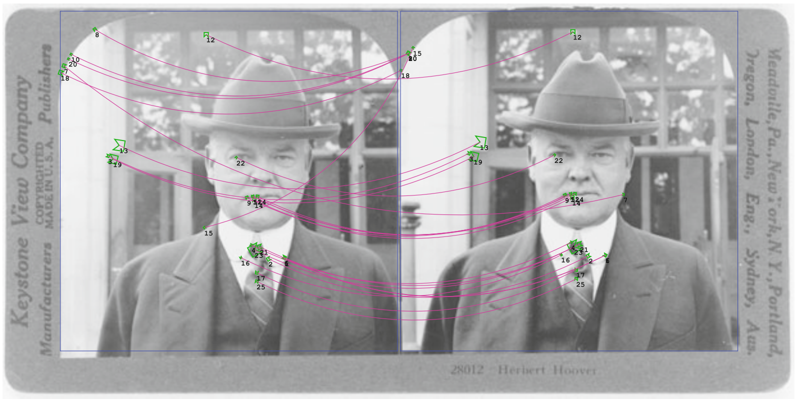


(c) L_∞-norm

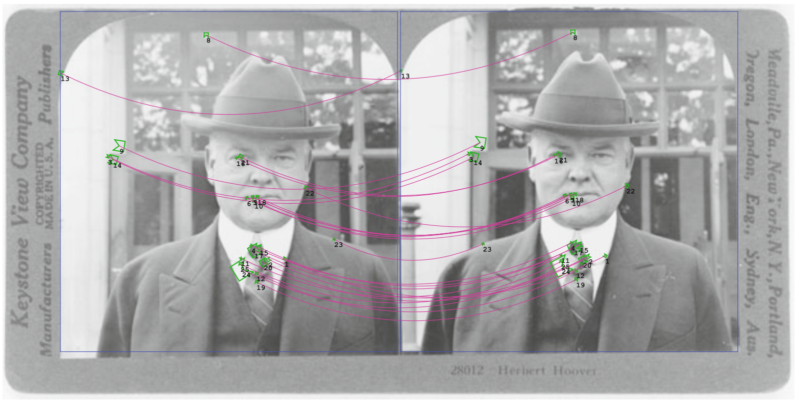
Approximate nearest-neighbor search in high-dimensional spaces is not only essential for practical SIFT matching in real time, but is a general problem with numerous applications in various disciplines and continued research. Open-source implementations of several different methods are available as software libraries.



(a) $\rho_{\max} = 1.0$



(b) $\rho_{\max} = 0.8$



(c) $\rho_{\max} = 0.5$

Fig. 25.31
Rejection of weak or ambiguous matches by limiting the ratio of primary and secondary match distance $\rho_{\max}$ (see Eqns. (25.104)–(25.105)).

25.7 Java Implementation

A new and complete Java implementation of the SIFT method has been written from ground up to complement the algorithms described in this chapter. Space limitations do not permit a full listing here, but the entire implementation and additional examples can be found in the source code section of this book's website. Most Java methods are named and structured identically to the procedures listed in the algorithms for easy identification. Note, however, that this implementation is again written for instructional clarity and readability. The code is neither tuned for efficiency nor is it intended to be used in a production environment.

25.7.1 SIFT Feature Extraction

The key class in this Java library is `SiftDetector`, which implements a SIFT detector for a given floating-point image. The following example illustrates its basic use for a given `ImageProcessor` object `ip`:

```
...
FloatProcessor I = ip.convertToFloatProcessor();
SiftDetector sd = new SiftDetector(I);
List<SiftDescriptor> S = sd.getSiftFeatures();
... // process descriptor set S
```

The initial work of setting up the required Gaussian and DoG scale space structures for the given image `I` is accomplished by the constructor in `new SiftDetector(I)`.

The method `getSiftFeatures()` then performs the actual feature detection process and returns a sequence of `SiftDescriptor` objects (`S`) for the image `I`. Each extracted `SiftDescriptor` in `S` holds information about its image position (`x`, `y`), its absolute scale σ (`scale`) and its dominant orientation θ (`orientation`). It also contains an invariant, 128-element, `int`-type feature vector $\boldsymbol{f}_{\mathrm{sift}}$ (see Alg. 25.8).

The SIFT detector uses a large set of parameters that are set to their default values (see Table 25.5) if the simple constructor `new SiftDetector(I)` is used, as in the previous example. All parameters can be adjusted individually by passing a parameter object (of type `SiftDetector.Parameters`) to its constructor, as in the following example, which shows feature extraction from two images `A`, `B` using identical parameters:

```
  ...
  FloatProcessor Ia = A.convertToFloatProcessor();
  FloatProcessor Ib = B.convertToFloatProcessor();
  ...
  SiftDetector.Parameters params =
      new SiftDetector.Parameters();
  params.sigma_s = 0.5; // modify individual parameters
  params.sigma_0 = 1.6;
  ...
  SiftDetector sdA = new SiftDetector(Ia, params);
  SiftDetector sdB = new SiftDetector(Ib, params);
  List<SiftDescriptor> SA = sda.getSiftFeatures();
  List<SiftDescriptor> SB = sdb.getSiftFeatures();
  ...
  // process descriptor sets SA and SB
```

25.7.2 SIFT Feature Matching

Finding matching descriptors from a pair of SIFT descriptor sets Sa, Sb is accomplished by the class SiftMatcher.[28] One descriptor set (Sa) is considered the "reference" or "model" set and used to initialize a new SiftMatcher object, as shown in the following example. The actual matches are then calculated by invoking the method matchDescriptors(), which implements the procedure MatchDescriptors() outlined in Alg. 25.11. It takes the second descriptor set (Sb) as the only argument. The following code segment continues from the previous example:

```java
...
SiftMatcher.Parameters params =
              new SiftMatcher.Parameters();
// set matcher parameters here (see below)
SiftMatcher matcher = new SiftMatcher(SA, params);
List<SiftMatch> matches = matcher.matchDescriptors(SB);
...
// process matches
```

As noted, certain parameters of class SiftMatcher can be set individually, for example,

```java
params.norm = FeatureDistanceNorm.L1; // L1, L2, or Linf
params.rmMax = 0.8; // ρmax, max. ratio of best and second-best match
params.sort = true; // set to true if sorting of matches is desired
```

The method matchDescriptors() in this prototypical implementation performs an exhaustive search over all possible descriptor pairs in the two sets Sa and Sb. To implement efficient approximate nearest-neighbor search (see Sec. 25.6), one would pre-calculate the required search tree structures for the model descriptor set (Sa) once inside SiftMatcher's constructor method. The same matcher object could then be reused to match against multiple descriptor sets without the need to recalculate the search tree structure over and over again. This is particularly effective when the given model set is large.

25.8 Exercises

Exercise 25.1. As claimed in Eqn. (25.12), the 2D LoG function $L_\sigma(x,y)$ can be approximated by the DoG in the form $L_\sigma(x,y) \approx \lambda \cdot (G_{\kappa\sigma}(x,y) - G_\sigma(x,y))$. Create a combined plot, similar to the one in Fig. 25.5(b), showing the 1D cross sections of the LoG and DoG functions (with $\sigma = 1.0$ and $y = 0$). Compare both functions by varying the values of $\kappa = 2.00, 1.25, 1.10, 1.05$, and 1.01. How does the approximation change as κ approaches 1, and what happens if κ becomes exactly 1?

Exercise 25.2. Test the performance of the SIFT feature detection and matching on pairs of related images under (a) changes of image brightness and contrast, (b) image rotation, (c) scale changes,

[28] File imagingbook.sift.SiftMatcher.java.

(d) adding (synthetic) noise. Choose (or shoot) your own test images, show the results in a suitable way and document the parameters used.

Exercise 25.3. Evaluate the SIFT mechanism for tracking features in video sequences. Search for a suitable video sequence with good features to track and process the images frame-by-frame.[29] Then match the SIFT features detected in pairs of successive frames by connecting the best-matching features, as long as the "match quality" is above a predefined threshold. Visualize the resulting feature trajectories. Could other properties of the SIFT descriptors (such as position, scale, and dominant orientation) be used to improve tracking stability?

[29] In ImageJ, choose an AVI video short enough to fit into main memory and open it as an image stack.