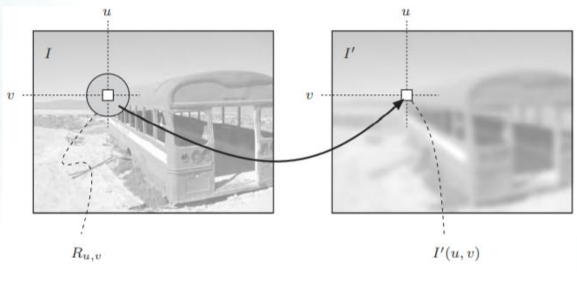
Filters

3

The essential property of ***point operations*** is that each new pixel value only depends on the original pixel at the same position. Point operations cannot accomplish the task of *sharpening* or *smoothing* an image. This is what filters can do.

The main difference between filters and point operations is that filters generally use more than one pixel from the source image for computing each new pixel value.



Filters …

Smoothing an image could be to simply replace every pixel by the average of its neighbouring pixels.

$$\begin{aligned}I'(u,v) \leftarrow \tfrac{1}{9}\cdot [\,I(u-1,v-1) &+ I(u,v-1) + I(u+1,v-1) \;+\\ I(u-1,v) &+ I(u,v) \quad\;\; + I(u+1,v) \quad\;\; +\\ I(u-1,v+1) &+ I(u,v+1) + I(u+1,v+1)\,]\,.\end{aligned}$$

Written more compactly, this is equivalent to

$$I'(u,v) \leftarrow \frac{1}{9}\cdot\sum_{j=-1}^{1}\sum_{i=-1}^{1} I(u+i,v+j).$$

The ***size*** of the filter region is an important parameter of the filter because it specifies how many original pixels contribute to each resulting pixel value and thus determines the spatial extent of the filter. The ***shape*** of the filter region is not necessarily quadratic or even rectangular

Linear Filters

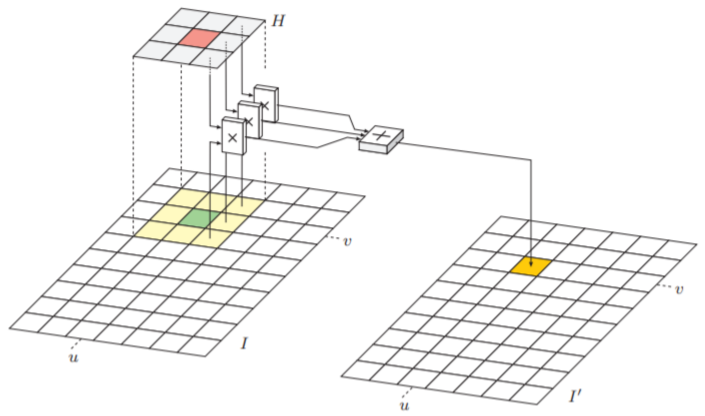
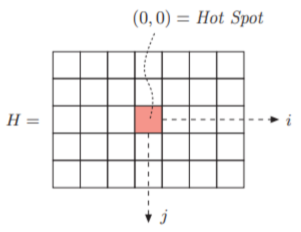


Figure Linear filter. The filter matrix H is placed with its origin at position (u, v) on the image I. Each filter coefficient $H(i, j)$ is multiplied with the corresponding image pixel $I(u+i, v+j)$, the results are added, and the final sum is inserted as the new pixel value $I'(u, v)$.



Linear Filters …

For a linear filter, the result is unambiguously and completely specified by the coefficients of the filter matrix. Applying the filter to an image is a simple process that is illustrated in Fig.

The following steps are performed at each image position (u, v):

1. The filter matrix H is moved over the original image I such that its origin $H(0, 0)$ coincides with the current image position (u, v).
2. All filter coefficients $H(i, j)$ are multiplied with the corresponding image element $I(u+i, v+j)$, and the results are added.
3. Finally, the resulting sum is stored at the current position in the new image $I'(u, v)$.

Described formally, pixels in the new image $I'(u, v)$ are computed by the operation

$$I'(u,v) \leftarrow \sum_{(i,j)\in R_H} I(u+i, v+j) \cdot H(i,j),$$

$$I'(u,v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u+i, v+j) \cdot H(i,j$$

Computing the Filter Operation

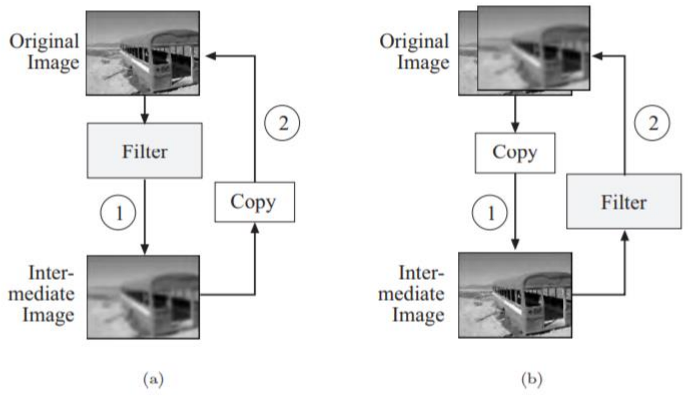


Figure Practical implementation of filter operations. **Version A** (a): The result of the filter is first stored in an intermediate image and subsequently copied back to the original image. **Version B** (b): The original image is first copied to an intermediate image that serves as the source for the actual filter operation. The result replaces the pixels in the original image.

Filters of Arbitrary Size

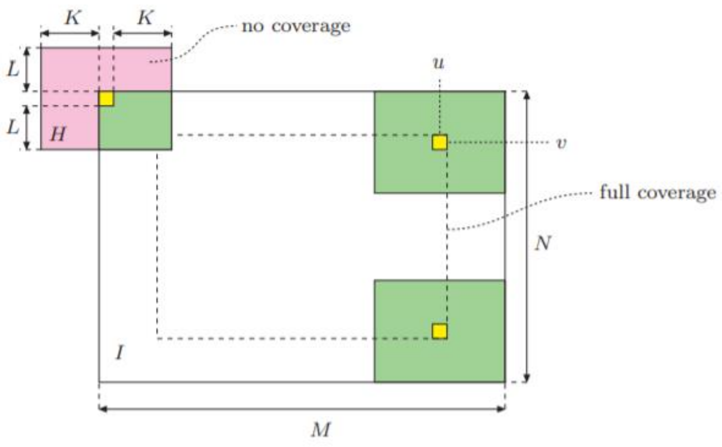
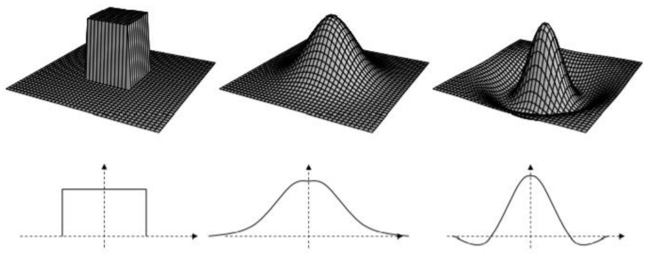


Figure Border geometry. The filter can be applied only at locations (u, v) where the filter matrix H of size $(2K+1) \times (2L+1)$ is fully contained in the image (inner rectangle).

Types of Linear Filters



(a)

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

(b)

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

(c)

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Figure Typical examples of linear filters, illustrated as 3D plots (top), profiles (center), and approximations by discrete filter matrices (bottom). The "box" filter (a) and the Gauss filter (b) are both *smoothing filters* with all-positive coefficients. The "Laplace" or "Mexican hat" filter (c) is a *difference filter*. It computes the weighted difference between the center pixel and the surrounding pixels and thus reacts most strongly to local intensity peaks.

Types of Linear Filters …

Box filter

- This simplest of all smoothing filters, whose 3D shape resembles a box
- Described in frequency terms, smoothing corresponds to ***low-pass filtering***
- Produces strong "ringing" in frequency space and is therefore ***not*** considered a high-quality smoothing filter

Gaussian filter

- This smoothing filter corresponds to a two-dimensional Gaussian function

$$G_\sigma(x,y) = e^{-\frac{r^2}{2\sigma^2}} = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where *σ* denotes the width (standard deviation) of the bell-shaped function and *r* is the distance (radius) from the center.

Types of Linear Filters …

- As a lowpass filter, the Gaussian is "well-behaved" in frequency space and thus clearly superior to the box filter.
- The two-dimensional Gaussian filter is separable into a pair of one-dimensional filters, which facilitates its efficient implementation.

Laplace filter

Local intensity variations are smoothed by averaging, thus we can expect the exact contrary to happen when differences are taken: local intensity changes are enhanced. Important applications of difference filters thus include edge detection and image sharpening.

Properties of Linear Convolution

1. **Commutativity** $I * H = H * I$
2. **Linearity**

$$(s \cdot I) * H = I * (s \cdot H) = s \cdot (I * H)$$

$$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$$

$$(b + I) * H \neq b + (I * H)$$

3. **Associativity** $A * (B * C) = (A * B) * C.$

Separability of Linear Filters

If a convolution kernel *H* can be expressed as the convolution of multiple kernels itself,

$$H = H_1 * H_2 * \ldots * H_n,$$

then the filter operation I ∗ H may be performed as a sequence of convolutions with the constituting kernels,

$$\begin{aligned} I * H &= I * (H_1 * H_2 * \ldots * H_n) \\ &= \big(\ldots((I * H_1) * H_2) * \ldots * H_n\big). \end{aligned}$$

Separability of Linear Filters …

x/y-separability

$$I' \;\leftarrow\; (I * H_x) * H_y \;=\; I * \underbrace{(H_x * H_y)}_{H_{xy}}$$

$$H_{x,y}(i,j) = (H_x \otimes H_y)\,(i,j) = H_x(i) \cdot H_y(j)$$

Example:

$$G_\sigma(x,y) \;=\; e^{-\frac{x^2+y^2}{2\sigma^2}} \;=\; e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}} \;=\; g_\sigma(x) \cdot g_\sigma(y)$$

Nonlinear Filters

Linear filters have an important disadvantage when used for smoothing or removing noise: all image structures, including points, edges, and lines, are also blurred, and the quality of the whole image is evenly reduced.

This effect cannot be avoided, and thus the use of linear filters for these kinds of tasks (noise removal in particular) is limited.

Nonlinear Filters …

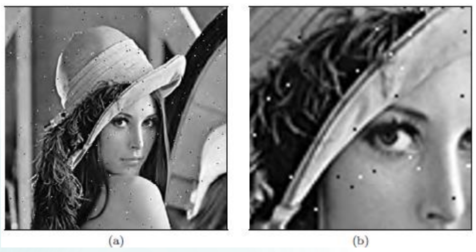
Like all other filters, nonlinear filters compute the result at some image position (*u*, *v*) from the pixels inside the moving region *Ru,v* of the original image. The filters are called "nonlinear" because the source pixel values are combined by some nonlinear function Filters.

Nonlinear Filters …

Minimum and Maximum Filters

$$I'(u,v) \leftarrow \min\{I(u+i,v+j) \mid (i,j) \in R\},$$
$$I'(u,v) \leftarrow \max\{I(u+i,v+j) \mid (i,j) \in R\},$$

Nonlinear Filters …



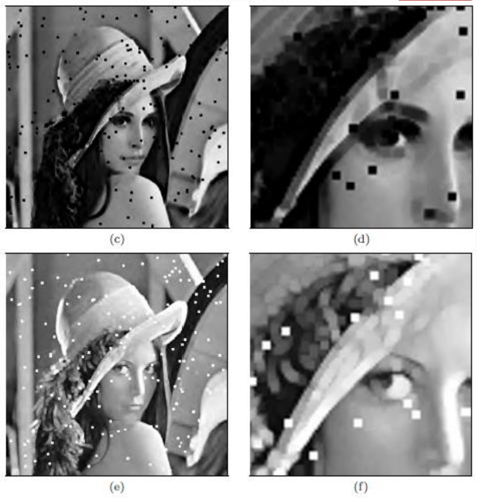


Figure Minimum and maximum filters applied to a grayscale image. The original image (a, b) is corrupted with "salt-and-pepper" noise. The 3 × 3 pixel *minimum* filter eliminates the bright dots and widens all dark image structures (c, d). The *maximum* filter shows the exact opposite effects (e, f).

Nonlinear Filters …

Median Filter $I'(u,v) \leftarrow \text{median}\,\{I(u+i,v+j) \mid (i,j) \in R\}$

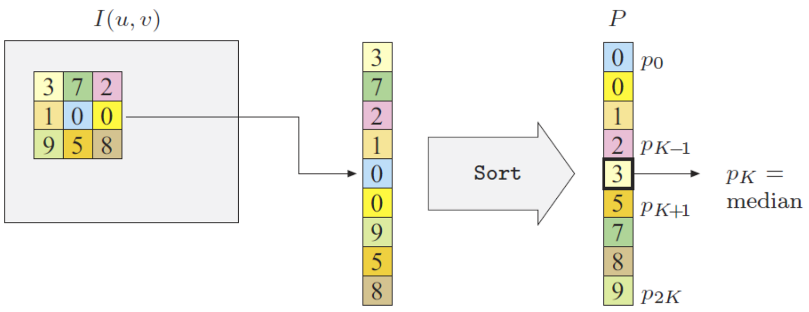


Figure Computation of a 3 × 3 pixel median filter. The nine pixel values extracted from the 3 × 3 image region are arranged as a vector (P) that is sorted, and the resulting center value is taken as the median.

Nonlinear Filters …

Weighted Median Filter

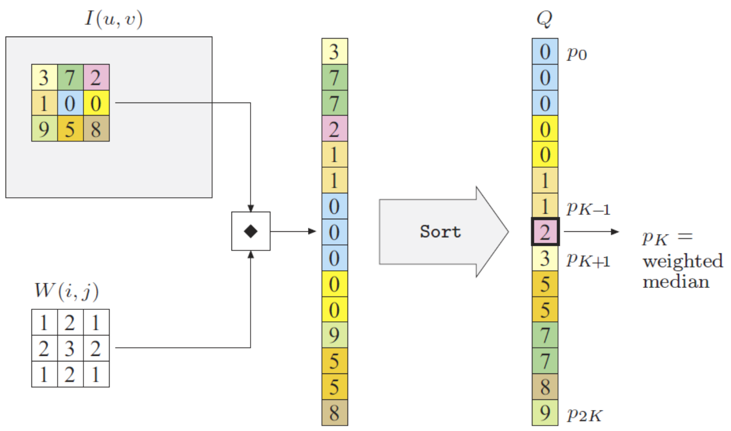


Figure Weighted median example. Each pixel value is inserted into the extended pixel vector multiple times, as specified by the weight matrix W. For example, the value 0 from the center pixel is inserted three times (since $W(0,0) = 3$) and the pixel value 7 twice. The pixel vector is sorted and the center value (2) is taken as the median.

Handling Image Borders

No filter results can be computed at positions where the filter matrix is not fully contained in the image array. Thus any filter operation would reduce the size of the resulting image, which is not acceptable in most applications. While no formally correct remedy exists, there are several more or less practical methods for handling the remaining border regions:

Method 1: Set the unprocessed pixels at the borders to some constant value (e.g., “black”).

Method 2: Set the unprocessed pixels to the original (unfiltered) image values.

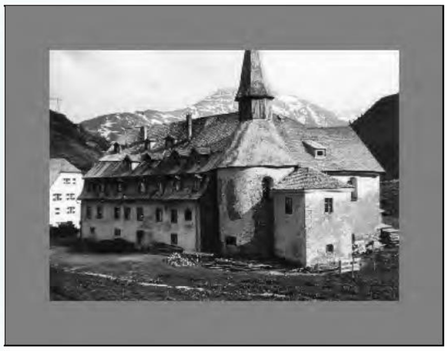
Handling Image Borders …

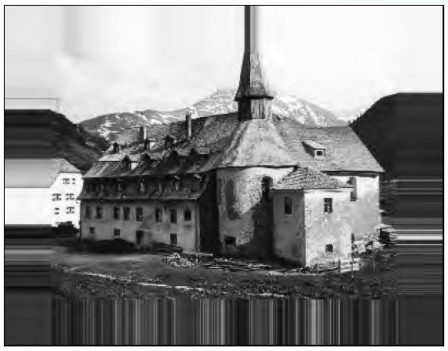
Method 3: Extend the image by “padding” additional pixels around it and filter the border regions, too, assuming that:

- The pixels outside the image have a *constant value*. This may produce strong artifacts at the image borders, particularly when large filters are used.
- The *border pixels extend* beyond the image boundaries. Only minor artifacts can be expected at the borders. The method is also simple to compute and is thus often considered the method of choice.
- The *image is mirrored* at each of its four boundaries. The results will be similar to those of the previous method unless very large filters are used.
- The *image repeats periodically* in the horizontal and vertical directions.

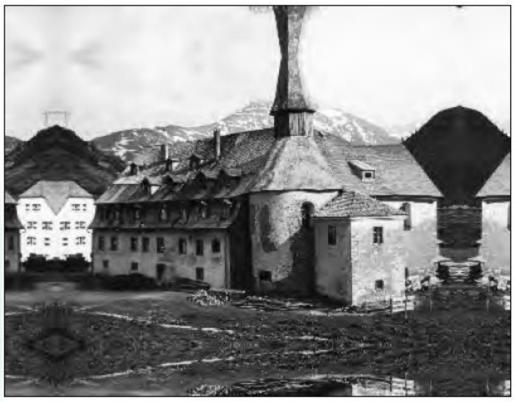
Handling Image Borders …

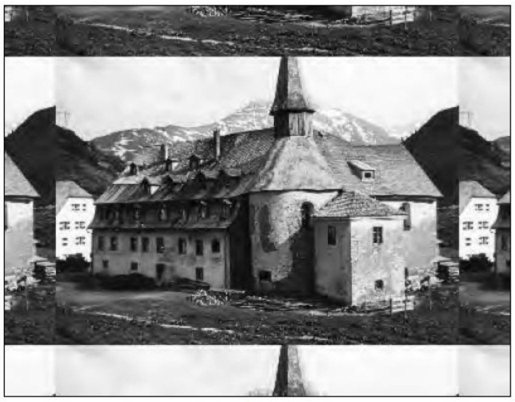
None of these methods is perfect and, as usual, the right choice depends upon the type of image and the filter applied.


(a)


(b)

Handling Image Borders …


(c)


(d)

Methods for padding the image to facilitate filtering along the borders. The assumption is that the pixels outside the original image are either set to some constant value (a), take on the value of the closest border pixel (b), are mirrored at the image boundaries (c), or repeat periodically along the coordinate axes (d).

Edges and Contours

- Edges and contours play a dominant role in human vision and probably in many other biological vision systems as well.
- Edges can roughly be described as image positions where the local intensity changes distinctly along a particular orientation. The stronger the local intensity change, the higher is the evidence for an edge at that position.
- In mathematics, the amount of change with respect to spatial distance is known as the first derivative of a function.

Gradient-Based Edge Detection

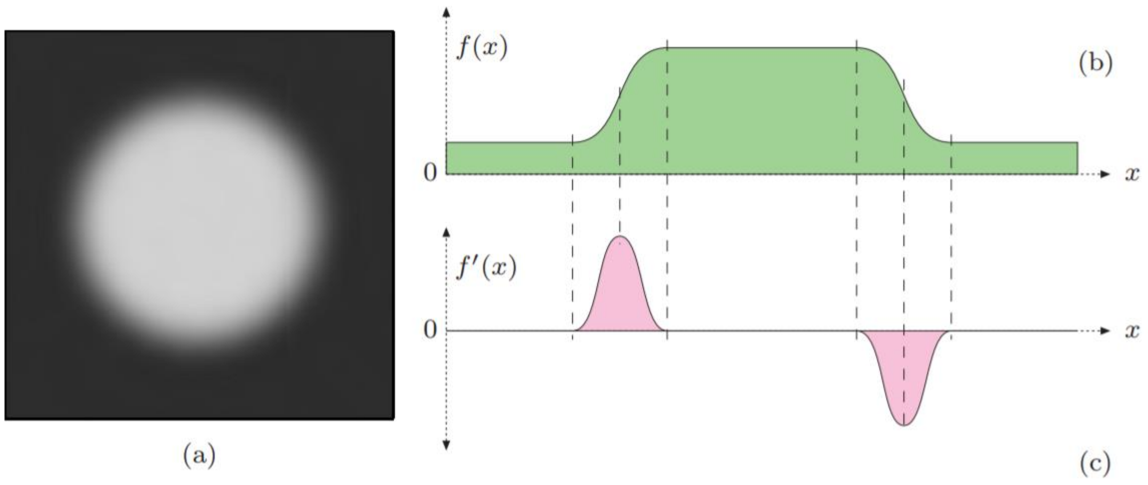


Figure Sample image and first derivative in one dimension: original image (a), horizontal intensity profile $f(x)$ along the center image line (b), and first derivative $f'(x)$ (c).

Gradient-Based Edge Detection …

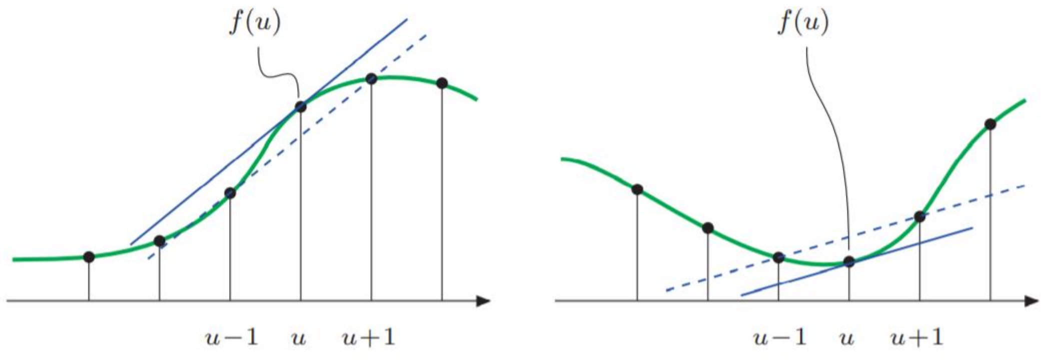


Figure Estimating the first derivative of a discrete function. The slope of the straight (dashed) line between the neighboring function values $f(u-1)$ and $f(u+1)$ is taken as the estimate for the slope of the tangent (i. e., the first derivative) at $f(u)$.

Partial Derivatives and the Gradient

A derivative of a multidimensional function taken along one of its coordinate axes is called a partial derivative; for example,

$$\frac{\partial I}{\partial u}(u,v) \qquad \text{and} \qquad \frac{\partial I}{\partial v}(u,v)$$

are the partial derivatives of the image function I(u, v) along the u and v axes, respectively. The function

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

is called the gradient vector (or "gradient" for short) of the function I at position (u, v).

Partial Derivatives and the Gradient …

The magnitude of the gradient,

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2},$$

is invariant under image rotation and thus independent of the orientation of the underlying image structures.

Partial Derivatives and the Gradient …

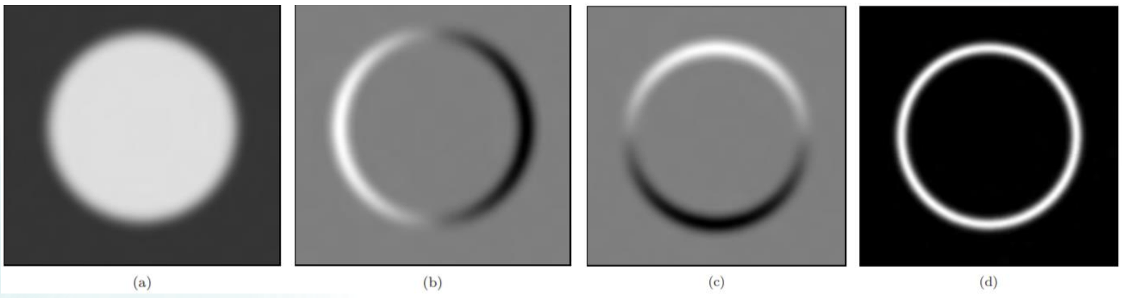


Figure Partial derivatives of a two-dimensional function: synthetic image function I (a); approximate first derivatives in the horizontal direction $\partial I/\partial u$ (b) and the vertical direction $\partial I/\partial v$ (c); magnitude of the resulting gradient $|\nabla I|$ (d). In (b) and (c), the lowest (negative) values are shown black, the maximum (positive) values are white, and zero values are gray.

Edge Operators

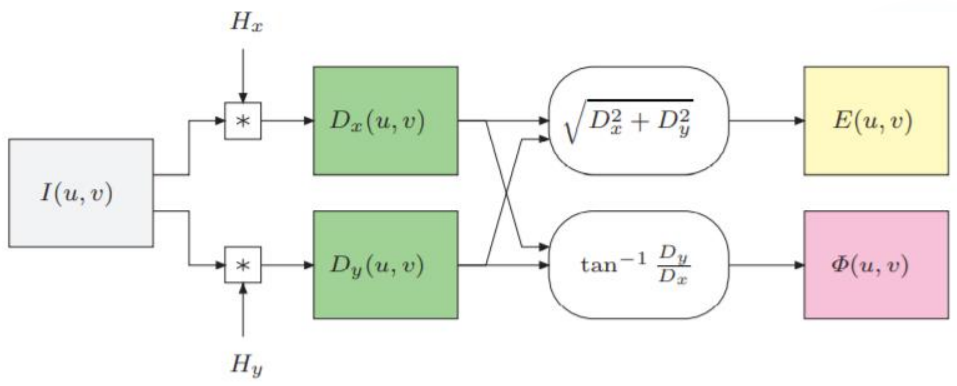


Figure Typical process of gradient-based edge extraction. The two linear gradient filters H_x and H_y produce two gradient images, D_x and D_y, respectively. They are used to compute the edge strength E and orientation Φ for each image position (u, v).

Edge Operators …

Prewitt and Sobel Operators

The edge operators by Prewitt and Sobel are two classic methods that differ only marginally in the filters they use. The Prewitt and Sobel operators use linear filters that extend over three adjacent lines and columns, respectively, to counteract the noise sensitivity of the simple gradient operators.

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Edge Operators …

Edge strength and orientation

$$D_x = H_x * I \qquad \text{and} \qquad D_y = H_y * I.$$

The local edge strength E(u, v) is defined as the gradient magnitude

$$E(u,v) = \sqrt{\big(D_x(u,v)\big)^2 + \big(D_y(u,v)\big)^2},$$

and the local edge orientation angle Φ(u, v) is

$$\Phi(u,v) = \tan^{-1}\Big(\frac{D_y(u,v)}{D_x(u,v)}\Big) = \text{Arctan}\big(D_y(u,v), D_x(u,v)\big).$$

Edge Operators …

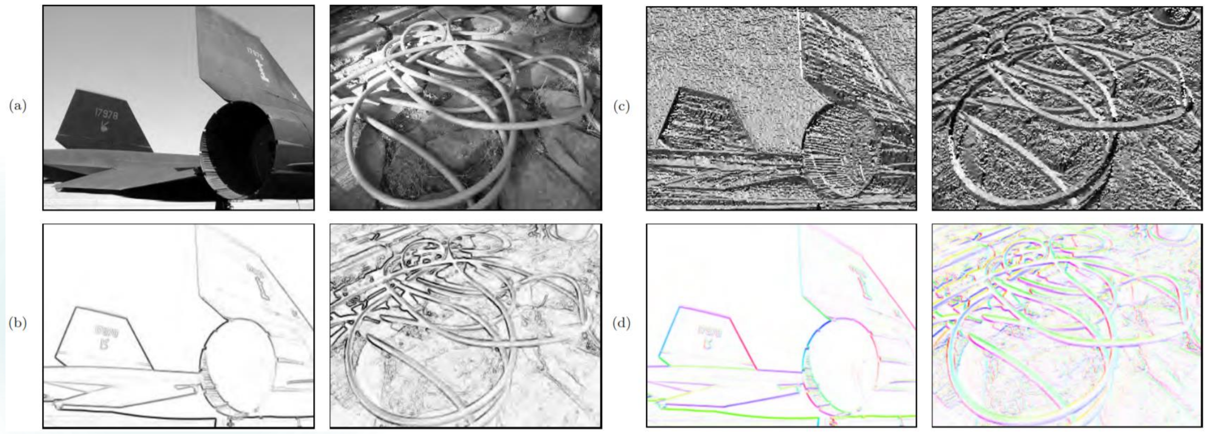


Figure Edge strength and orientation obtained with a Sobel operator. Original images (a), the edge strength $E(u,v)$ (b), and the local edge orientation $\Phi(u,v)$ (c). The images in (d) show the orientation angles coded as color hues, with the edge strength controlling the color saturation.

Edge Operators …

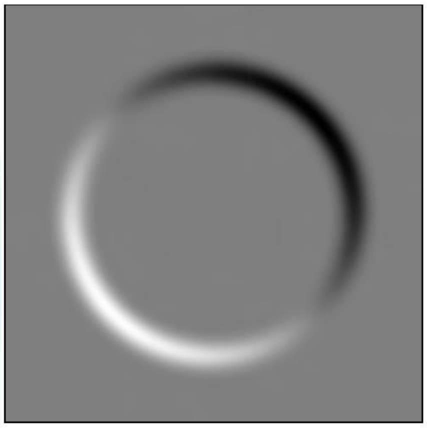
Robert's Operator

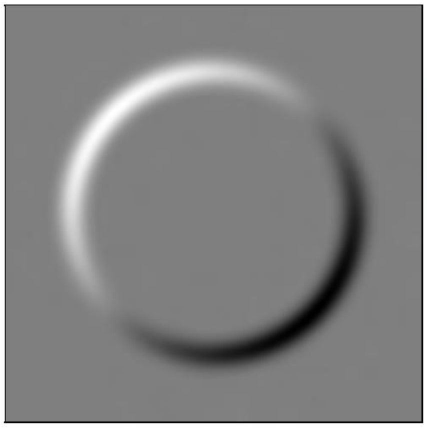
- It employs two extremely small filters of size 2 × 2 for estimating the directional gradient along the image diagonals:

$$H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$

- These filters naturally respond to diagonal edges but are not highly selective to orientation

Edge Operators …


$D_1 = I * H_1^R$


$D_2 = I * H_2^R$

Edge Operators …

Compass Operators

The design of linear edge filters involves a trade-off: the stronger a filter responds to edge-like structures, the more sensitive it is to orientation. In other words, filters that are orientation-insensitive tend to respond to non-edge structures, while the most discriminating edge filters only respond to edges in a narrow range of orientations.

One solution is to use not only a single pair of relatively "wide" filters for two directions but a larger set of filters with narrowly spaced orientations.

A classic example is the extended Sobel operator, which employs the following eight filters with orientations spaced at 45°:

Edge Operators …

$$H_0^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_4^S = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix},$$

$$H_1^S = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad H_5^S = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix},$$

$$H_2^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_6^S = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$

$$H_3^S = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \qquad H_7^S = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}.$$

Edge Operators …

Only the results of four of the eight filters above must actually be computed since the remaining four are identical except for the reversed sign.

$$I * H_4^S = I * -H_0^S = -(I * H_0^S);$$

The directional outputs for the eight Sobel filters can thus be computed as follows:

$$\begin{array}{llll} D_0 \leftarrow I * H_0^S & D_1 \leftarrow I * H_1^S & D_2 \leftarrow I * H_2^S & D_3 \leftarrow I * H_3^S \\ D_4 \leftarrow -D_0 & D_5 \leftarrow -D_1 & D_6 \leftarrow -D_2 & D_7 \leftarrow -D_3. \end{array}$$

Edge Operators …

The edge strength E^S at position (u, v) is defined as the maximum of the eight filter outputs; i. e.,

$$\begin{aligned} E^S(u,v) &\triangleq \max\big(D_0(u,v), D_1(u,v), D_2(u,v), D_3(u,v), D_4(u,v), \ldots D_7(u,v)\big) \\ &= \max\big(|D_0(u,v)|, |D_1(u,v)|, |D_2(u,v)|, |D_3(u,v)|\big), \end{aligned}$$

and the strongest-responding filter also determines the local edge orientation as

$$\Phi^S(u,v) \triangleq \frac{\pi}{4} j, \qquad \text{with } j = \operatorname*{argmax}_{0 \le i \le 7} D_i(u,v).$$

Edge Detection Based on Second Derivatives

The second derivative of a function measures its local curvature. The idea is that edges can be found at zero positions or—even better—at the zero crossings of the second derivatives of the image function.

A popular example is the "Laplacian-of-Gaussian" (LoG) operator, which combines Gaussian smoothing and computing the second derivatives into a single linear filter.

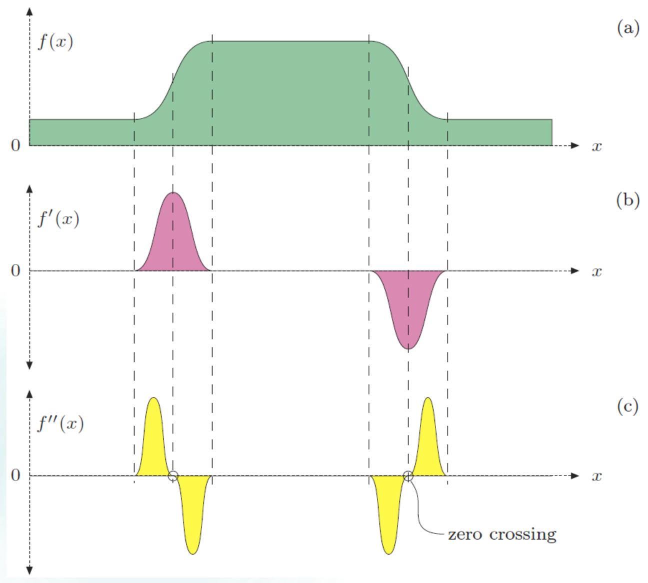


Figure Principle of edge detection with the second derivative: original function (a), first derivative (b), and second derivative (c). Edge points are located where the second derivative crosses through zero and the first derivative has a high magnitude.

Edges at Different Scales

Unfortunately, the results of the simple edge operators we have discussed so far often deviate from what we as humans perceive as important edges. The two main reasons for this are:

– First, edge operators only respond to local intensity differences, while our visual system is able to extend edges across areas of minimal or vanishing contrast.

– Second, edges exist not at a single fixed resolution or at a certain scale but over a whole range of different scales.

Edges at Different Scales …

To recognize edge-like events over a greater horizon, we would either need larger edge operators (with correspondingly large filters) or use the original (small) operators on reduced (i. e., scaled) images. This is the principal idea of "multiresolution" techniques (also referred to as "hierarchical" or "pyramid" techniques), which have traditionally been used in many image-processing applications.

In the context of edge detection, this typically amounts to detecting edges at various scale levels first and then deciding which edge (if any) at which scale level is dominant at each image position.

Canny Operator

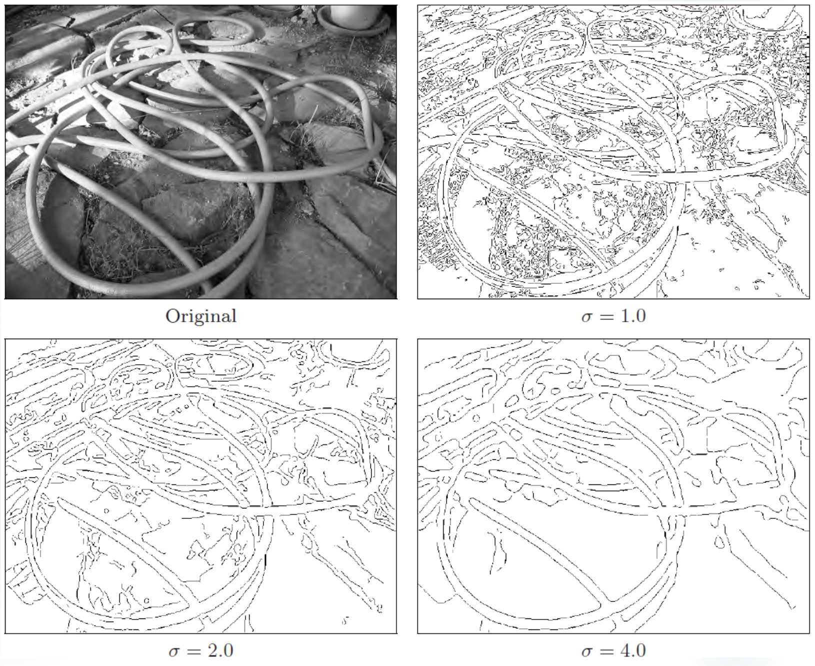
A popular example for such a method is the edge operator by Canny, which employs a set of relatively large, oriented filters at multiple image resolutions and merges the individual results into a common edge map. The method tries to reach three main goals:

(a) to minimize the number of false edge points,

(b) achieve good localization of edges, and

(c) deliver only a single mark on each edge.

At its core, the Canny "filter" is a gradient method, but it uses the zero crossings of second derivatives for precise edge localization. Canny operator is often preferred over other edge detection methods.



Edge Sharpening

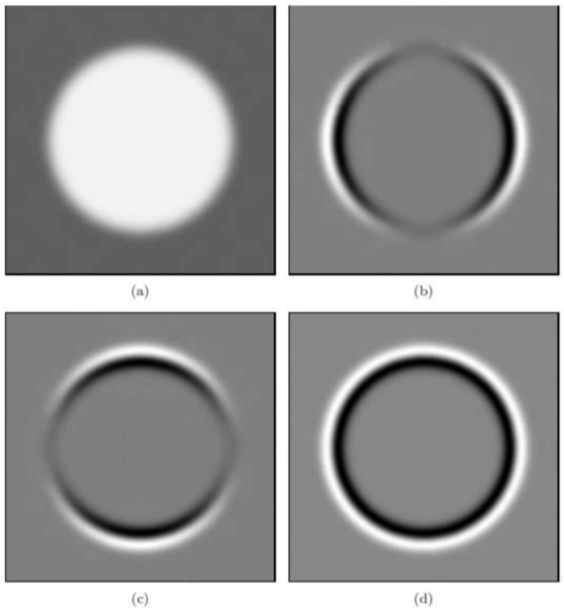


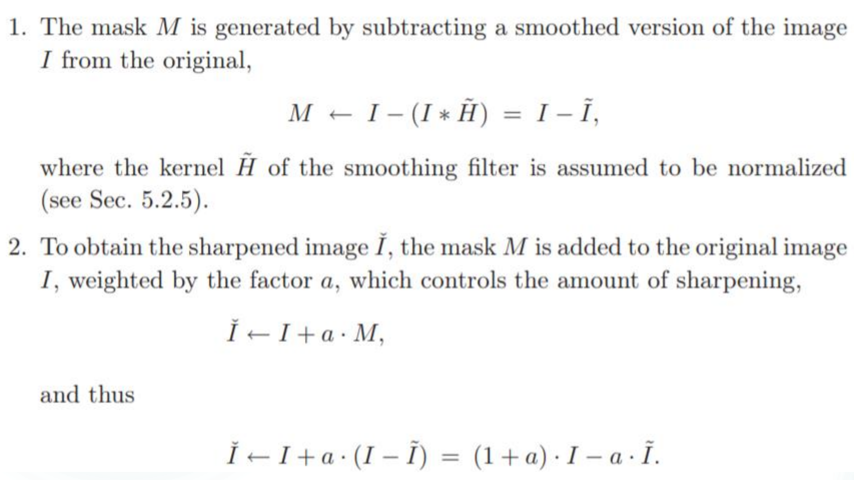
Figure 6.13 Results of Laplace filter H^L: synthetic test image I (a), second partial derivative $\partial^2 I/\partial^2 u$ in the horizontal direction (b), second partial derivative $\partial^2 I/\partial^2 v$ in the vertical direction (c), and Laplace filter $\nabla^2 I(u, v)$ (d). Intensities in (b–d) are scaled such that maximally negative and positive values are shown as black and white, respectively, and zero values are gray.

Unsharp Masking

“Unsharp masking” (USM) is a technique for edge sharpening that is particularly popular in astronomy, digital printing, and many other areas of image processing.

The first step in the USM filter is to subtract a smoothed version of the image from the original, which enhances the edges. The result is called the “mask”. In analog photography, the required smoothing was achieved by simply defocusing the lens. Subsequently, the mask is again added to the original, such that the edges in the image are sharpened.

Unsharp Masking …



1. The mask M is generated by subtracting a smoothed version of the image I from the original,

$$M \leftarrow I - (I * \tilde{H}) = I - \tilde{I},$$

where the kernel $\tilde{H}$ of the smoothing filter is assumed to be normalized (see Sec. 5.2.5).

2. To obtain the sharpened image $\check{I}$, the mask M is added to the original image I, weighted by the factor a, which controls the amount of sharpening,

$$\check{I} \leftarrow I + a \cdot M,$$

and thus

$$\check{I} \leftarrow I + a \cdot (I - \tilde{I}) = (1 + a) \cdot I - a \cdot \tilde{I}.$$