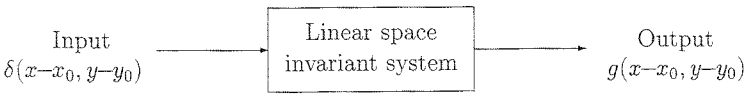
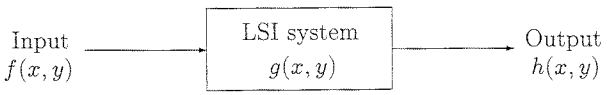
For a linear system, when the input to the system is an impulse $\delta(x, y)$ centered at the origin, the output $g(x, y)$ is the system's *impulse response.* Furthermore, a system whose response remains the same irrespective of the position of the input pulse is called a *space invariant* system:



A linear space invariant (LSI) system can be completely described by its impulse response $g(x, y)$ as follows:



where $f(x, y)$ and $h(x, y)$ are the input and output images, respectively. The above system must satisfy the following relationship:

$$a \cdot f_1(x, y) + b \cdot f_2(x, y) \Longrightarrow a \cdot h_1(x, y) + b \cdot h_2(x, y)$$

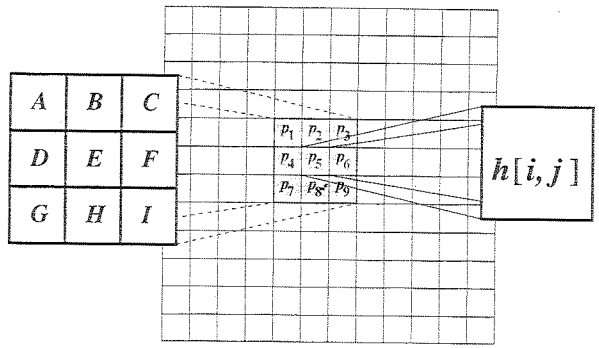
where $f_1(x, y)$ and $f_2(x, y)$ are the input images, $h_1(x, y)$ and $h_2(x, y)$ are the output images corresponding to f_1 and f_2, and a and b are constant scaling factors.

For such a system, the output $h(x, y)$ is the *convolution* of $f(x, y)$ with the impulse response $g(x, y)$ and is defined as:

$$\begin{aligned} h(x, y) &= f(x, y) \star g(x, y) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x', y')\, g(x - x', y - y')\, dx'\, dy'. \end{aligned} \tag{4.5}$$

For discrete functions, this becomes:

$$\begin{aligned} h[i, j] &= f[i, j] \star g[i, j] \\ &= \sum_{k=1}^{n}\sum_{l=1}^{m} f[k, l]\, g[i - k, j - l]. \end{aligned} \tag{4.6}$$



$$h[i, j] = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

Figure 4.4: An example of a 3×3 convolution mask. The origin of the convolution mask corresponds to location E and the weights $A, B, \ldots I$ are the values of $g[-k, -l]$, $k, l = -1, 0, +1$.

If f and h are images, convolution becomes the computation of weighted sums of the image pixels. The impulse response, $g[i, j]$, is referred to as a *convolution mask.* For each pixel $[i, j]$ in the image, the value $h[i, j]$ is calculated by translating the convolution mask to pixel $[i, j]$ in the image, and then taking the weighted sum of the pixels in the neighborhood about $[i, j]$ where the individual weights are the corresponding values in the convolution mask. This process is illustrated in Figure 4.4 using a 3×3 mask.

Convolution is a linear operation, since

$$g[i, j] \star \{a_1 h_1[i, j] + a_2 h_2[i, j]\} = a_1\{g[i, j] \star h_1[i, j]\} + a_2\{g[i, j] \star h_2[i, j]\}$$

for any constants a_1 and a_2. In other words, the convolution of a sum is the sum of the convolutions, and the convolution of a scaled image is the scaled convolution. Convolution is a spatially invariant operation, since the same filter weights are used throughout the image. However, a spatially varying filter requires different filter weights in different parts of the image.

Fourier Transform

An $n \times m$ image can be represented by its frequency components as follows:

$$f[k,l] = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} F(u,v)\, e^{jku}\, e^{jlv}\, du\, dv \tag{4.7}$$

where $F(u,v)$ is the *Fourier transform* of the image. The Fourier transform encodes the amplitude and phase of each frequency component and is defined as

$$\begin{aligned} F(u,v) &= \mathcal{F}\{f[k,l]\} \\ &= \sum_{k=1}^{n} \sum_{l=1}^{m} f[k,l]\, e^{-jku}\, e^{-jlv} \end{aligned} \tag{4.8}$$

where $\mathcal{F}$ denotes the Fourier transform operation. The values near the origin of the (u,v) plane are the called the low-frequency components of the Fourier transform, and those distant from the origin are the high-frequency components. Note that $F(u,v)$ is a continuous function.

Convolution in the image domain corresponds to multiplication in the spatial frequency domain. Therefore, convolution with large filters, which would normally be an expensive process in the image domain, can be implemented efficiently using the fast Fourier transform. This is an important technique in many image processing applications. In machine vision, however, most algorithms are nonlinear or spatially varying, so the fast Fourier transform cannot be used. In most cases where the vision algorithm can be modeled as a linear, spatially invariant system, the filter sizes are so small that implementing convolution with the fast Fourier transform provides little or no benefit; hence, linear filters, such as the smoothing filters discussed in the following sections, are usually implemented through convolution in the image domain.

4.3 Linear Filters

As mentioned earlier, images are often corrupted by random variations in intensity values, called *noise*. Some common types of noise are *salt and pepper* noise, *impulse* noise, and *Gaussian* noise. Salt and pepper noise contains random occurrences of both black and white intensity values. However, impulse

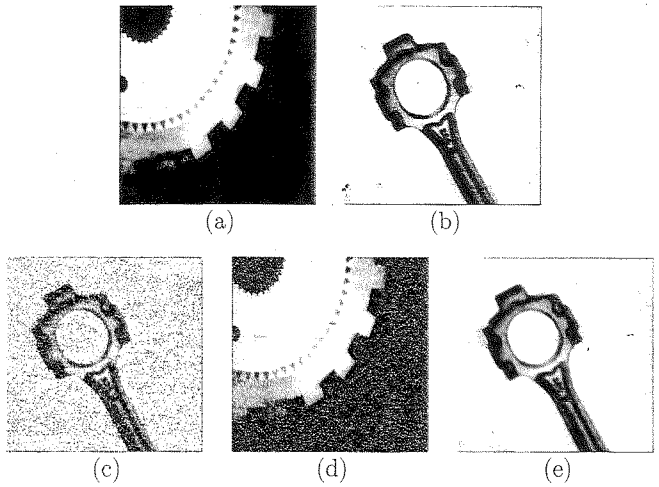


Figure 4.5: Examples of images corrupted by salt and pepper, impulse, and Gaussian noise. (a) & (b) Original images. (c) Salt and pepper noise. (d) Impulse noise. (e) Gaussian noise.

noise contains only random occurrences of white intensity values. Unlike these, Gaussian noise contains variations in intensity that are drawn from a Gaussian or normal distribution and is a very good model for many kinds of sensor noise, such as the noise due to camera electronics (see Figure 4.5).

Linear smoothing filters are good filters for removing Gaussian noise and, in most cases, the other types of noise as well. A linear filter is implemented using the weighted sum of the pixels in successive windows. Typically, the same pattern of weights is used in each window, which means that the linear filter is spatially invariant and can be implemented using a convolution mask. If different filter weights are used for different parts of the image, but the filter is still implemented as a weighted sum, then the linear filter is spatially varying. Any filter that is not a weighted sum of pixels is a nonlinear filter. Nonlinear filters can be spatially invariant, meaning that the same calculation is performed regardless of the position in the image, or spatially varying. The median filter, presented in Section 4.4, is a spatially invariant, nonlinear filter.

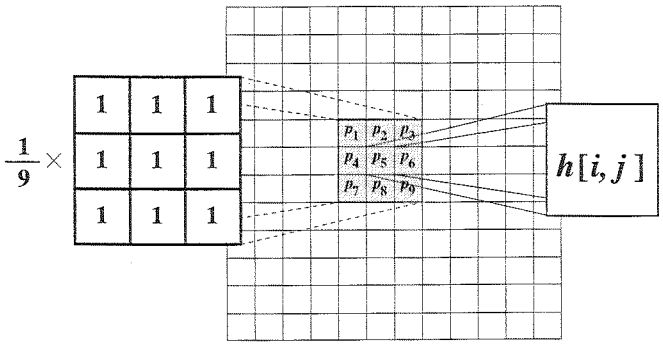


Figure 4.6: An example illustrating the mean filter using a 3×3 neighborhood.

Mean Filter

One of the simplest linear filters is implemented by a local averaging operation where the value of each pixel is replaced by the average of all the values in the local neighborhood:

$$h[i,j] = \frac{1}{M} \sum_{(k,l) \in N} f[k,l] \tag{4.9}$$

where M is the total number of pixels in the neighborhood N. For example, taking a 3×3 neighborhood about $[i,j]$ yields:

$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{k=j-1}^{j+1} f[k,l]. \tag{4.10}$$

Compare this with Equation 4.6. Now if $g[i,j] = 1/9$ for every $[i,j]$ in the convolution mask, the convolution operation in Equation 4.6 reduces to the local averaging operation shown above. This result shows that a mean filter can be implemented as a convolution operation with equal weights in the convolution mask (see Figure 4.6). In fact, we will see later that many image processing operations can be implemented using convolution.

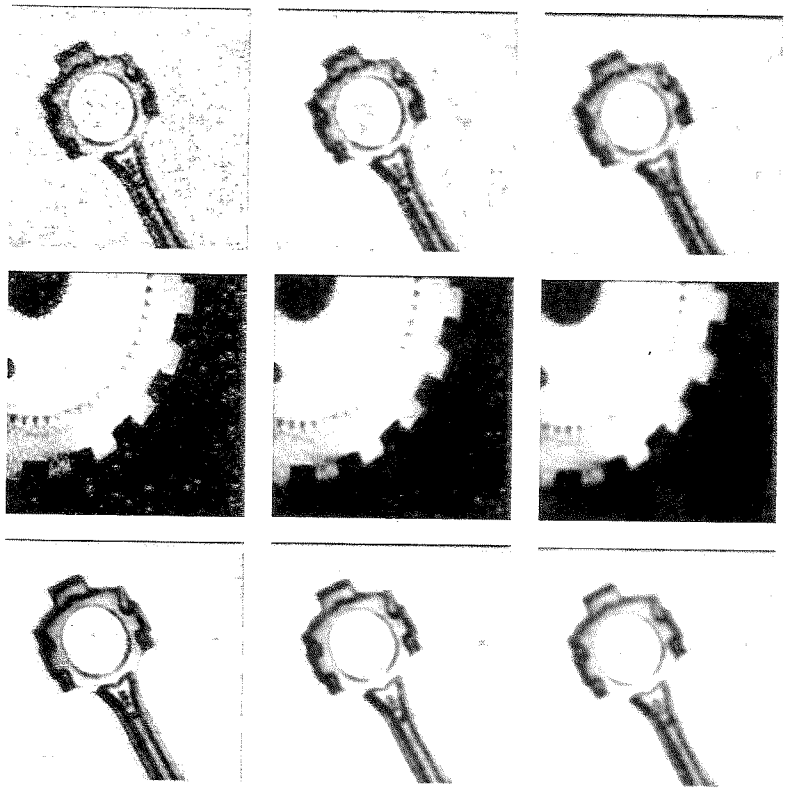


Figure 4.7: The results of a 3×3, 5×5, and 7×7 mean filter on the noisy images from Figure 4.5.

The size of the neighborhood N controls the amount of filtering. A larger neighborhood, corresponding to a larger convolution mask, will result in a greater degree of filtering. As a trade-off for greater amounts of noise reduction, larger filters also result in a loss of image detail. The results of mean filters of various sizes are shown in Figure 4.7.

When designing linear smoothing filters, the filter weights should be chosen so that the filter has a single peak, called the main lobe, and symmetry in the vertical and horizontal directions. A typical pattern of weights for a 3×3 smoothing filter is

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

Linear smoothing filters remove high-frequency components, and the sharp detail in the image is lost. For example, step changes will be blurred into gradual changes, and the ability to accurately localize a change will be sacrificed. A spatially varying filter can adjust the weights so that more smoothing is done in a relatively uniform area of the image, and little smoothing is done across sharp changes in the image. The results of a linear smoothing filter using the mask shown above are shown in Figure 4.8.

4.4 Median Filter

The main problem with local averaging operations is that they tend to blur sharp discontinuities in intensity values in an image. An alternative approach is to replace each pixel value with the median of the gray values in the local neighborhood. Filters using this technique are called *median filters.*

Median filters are very effective in removing salt and pepper and impulse noise while retaining image details because they do not depend on values which are significantly different from typical values in the neighborhood. Median filters work in successive image windows in a fashion similar to linear filters. However, the process is no longer a weighted sum. For example, take a 3×3 window and compute the median of the pixels in each window centered around $[i, j]$:

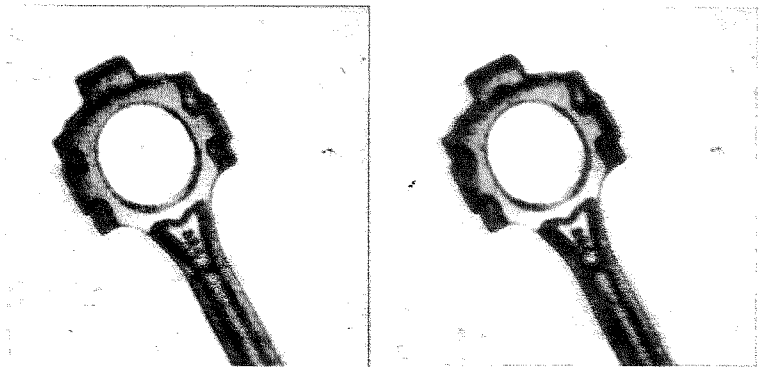


Figure 4.8: The results of a linear smoothing filter on an image corrupted by Gaussian noise. *Left:* Noisy image. *Right:* Smoothed image.

1. Sort the pixels into ascending order by gray level.
2. Select the value of the middle pixel as the new value for pixel $[i, j]$.

This process is illustrated in Figure 4.9. In general, an odd-size neighborhood is used for calculating the median. However, if the number of pixels is even, the median is taken as the average of the middle two pixels after sorting. The results of various sizes of median filters are shown in Figure 4.10.

4.5 Gaussian Smoothing

Gaussian filters are a class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function. The Gaussian smoothing filter is a very good filter for removing noise drawn from a normal distribution.[1] The zero-mean Gaussian function in one dimension is

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}, \tag{4.11}$$

[1]The fact that the filter weights are chosen from a Gaussian distribution and that the noise is also distributed as a Gaussian is merely a coincidence.

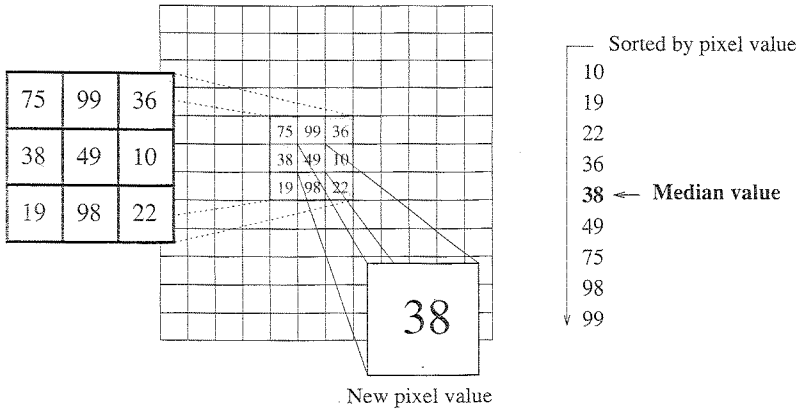


Figure 4.9: An example illustrating the median filter using a 3×3 neighborhood.

where the Gaussian spread parameter σ determines the width of the Gaussian. For image processing, the two-dimensional zero-mean discrete Gaussian function,

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}, \tag{4.12}$$

is used as a smoothing filter. A plot of this function is shown in Figure 4.11.

Gaussian functions have five properties that make them particularly useful in early vision processing. These properties indicate that the Gaussian smoothing filters are effective low-pass filters from the perspective of both the spatial and frequency domains, are efficient to implement, and can be used effectively by engineers in practical vision applications. The five properties are summarized below. Further explanation of the properties is provided later in this section.

1. In two dimensions, Gaussian functions are rotationally symmetric. This means that the amount of smoothing performed by the filter will be the same in all directions. In general, the edges in an image will not be oriented in some particular direction that is known in advance; consequently, there is no reason a priori to smooth more in one direction

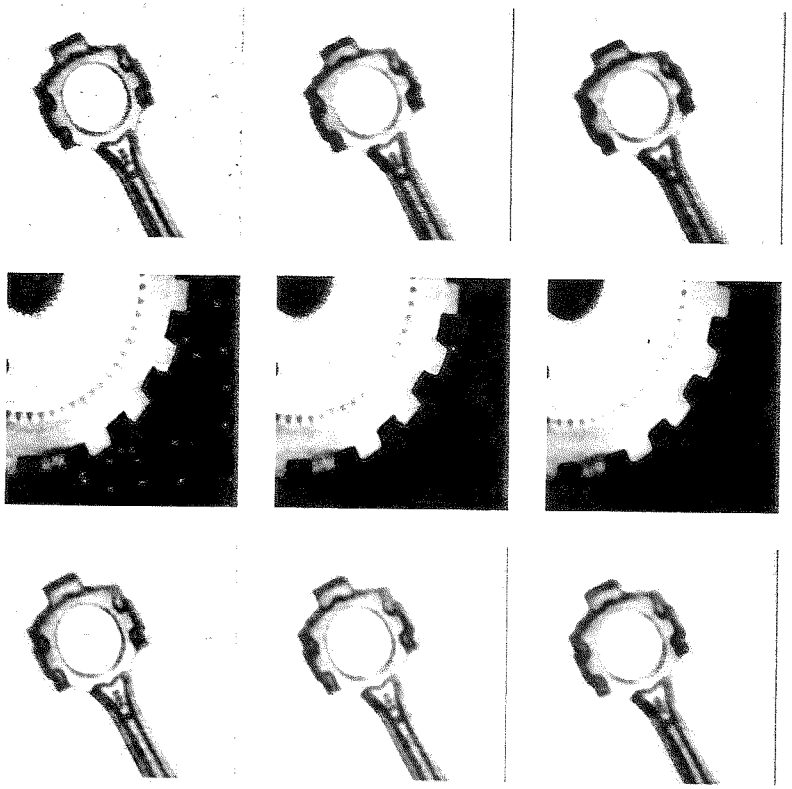


Figure 4.10: The results of a 3×3, 5×5, and 7×7 median filter on the noisy images from Figure 4.5.

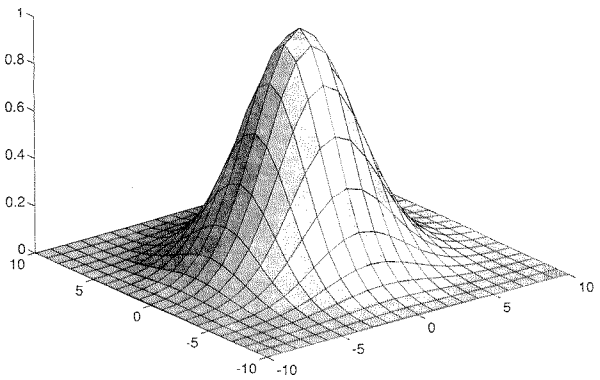


Figure 4.11: The two-dimensional Gaussian function with zero mean.

than in another. The property of rotational symmetry implies that a Gaussian smoothing filter will not bias subsequent edge detection in any particular direction.

2. The Gaussian function has a single lobe. This means that a Gaussian filter smooths by replacing each image pixel with a weighted average of the neighboring pixels such that the weight given to a neighbor decreases monotonically with distance from the central pixel. This property is important since an edge is a local feature in an image, and a smoothing operation that gives more significance to pixels farther away will distort the features.

3. The Fourier transform of a Gaussian has a single lobe in the frequency spectrum. This property is a straightforward corollary of the fact that the Fourier transform of a Gaussian is itself a Gaussian, as will be shown below. Images are often corrupted by undesirable high-frequency signals (noise and fine texture). The desirable image features, such as edges, will have components at *both* low and high frequencies. The single lobe in the Fourier transform of a Gaussian means that the smoothed image will not be corrupted by contributions from unwanted high-frequency signals, while most of the desirable signals will be re-

4. The width, and hence the degree of smoothing, of a Gaussian filter is parameterized by σ, and the relationship between σ and the degree of smoothing is very simple. A larger σ implies a wider Gaussian filter and greater smoothing. Engineers can adjust the degree of smoothing to achieve a compromise between excessive blur of the desired image features (too much smoothing) and excessive undesired variation in the smoothed image due to noise and fine texture (too little smoothing).

5. Large Gaussian filters can be implemented very efficiently because Gaussian functions are separable. Two-dimensional Gaussian convolution can be performed by convolving the image with a one-dimensional Gaussian and then convolving the result with the same one-dimensional filter oriented orthogonal to the Gaussian used in the first stage. Thus, the amount of computation required for a 2-D Gaussian filter grows linearly in the width of the filter mask instead of growing quadratically.

4.5.1 Rotational Symmetry

The rotational symmetry of the Gaussian function can be shown by converting the function from rectangular to polar coordinates. Remember the two-dimensional Gaussian function

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}. \tag{4.13}$$

Since the radius in polar coordinates is given by $r^2 = i^2 + j^2$, it is easy to see that the Gaussian function in polar coordinates,

$$g(r,\theta) = e^{-\frac{r^2}{2\sigma^2}}, \tag{4.14}$$

does not depend on the angle θ and consequently is rotationally symmetric. It is also possible to construct rotationally nonsymmetric Gaussian functions if they are required for an application where it is known in advance that more smoothing must be done in some specified direction. Formulas for rotationally nonsymmetric Gaussian functions are provided by Wozencraft and Jacobs [257, pp. 148–171], where they are used in the probabilistic analysis of communications channels.

4.5.2 Fourier Transform Property

The Gaussian function has the interesting property that its Fourier transform is also a Gaussian function. Since the Fourier transform of a Gaussian is a real function, the Fourier transform is its own magnitude. The Fourier transform of a Gaussian is computed by

$$\mathcal{F}\{g(x)\} = \int_{-\infty}^{\infty} g(x)\, e^{-j\omega x}\, dx \tag{4.15}$$

$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}}\, e^{-j\omega x}\, dx \tag{4.16}$$

$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} (\cos\omega x + j \sin\omega x)\, dx \tag{4.17}$$

$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \cos\omega x\, dx + j \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \sin\omega x\, dx. \tag{4.18}$$

The Gaussian is a symmetric function and the sine function is antisymmetric, so the integrand in the second integral is antisymmetric. Therefore, the integral must be zero, and the Fourier transform simplifies to:

$$\mathcal{F}\{g(x)\} = \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \cos\omega x\, dx \tag{4.19}$$

$$= \sqrt{2\pi}\sigma e^{-\frac{\omega^2}{2\nu^2}}, \qquad \nu^2 = \frac{1}{\sigma^2}. \tag{4.20}$$

The spatial frequency parameter is ω, and the spread of the Gaussian in the frequency domain is controlled by ν, which is the reciprocal of the spread parameter σ in the spatial domain. This means that a narrower Gaussian function in the spatial domain has a wider spectrum, and a wider Gaussian function in the spatial domain has a narrower spectrum. This property relates to the noise suppression ability of a Gaussian filter. A narrow-spatial-domain Gaussian does less smoothing, and in the frequency domain its spectrum has more bandwidth and passes more of the high-frequency noise and texture. As the width of a Gaussian in the spatial domain is increased, the amount of smoothing that the Gaussian performs is increased, and in the frequency domain the Gaussian becomes narrower and passes less high-frequency noise and texture. This simple relationship between spatial-domain Gaussian width and frequency-domain spectral width enhances the ease of use of the Gaussian filter in practical design situations. The Fourier

transform duality of Gaussian functions also explains why the single-lobe property in the spatial domain carries over into the frequency domain.

4.5.3 Gaussian Separability

The separability of Gaussian filters is easy to demonstrate:

$$g[i,j] \star f[i,j] = \sum_{k=1}^{m}\sum_{l=1}^{n} g[k,l]\, f[i-k, j-l] \tag{4.21}$$

$$= \sum_{k=1}^{m}\sum_{l=1}^{n} e^{-\frac{(k^2+l^2)}{2\sigma^2}} f[i-k, j-l] \tag{4.22}$$

$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \left\{ \sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f[i-k, j-l] \right\}. \tag{4.23}$$

The summation in brackets is the convolution of the input image $f[i,j]$ with a vertical one-dimensional Gaussian function. The result of this summation is a two-dimensional image, blurred in the vertical dimension, that is then used as the input to a second convolution with a horizontal one-dimensional Gaussian that blurs the image in the horizontal dimension (see Figure 4.12). Since convolution is associative and commutative, the order of the convolutions can be reversed so that the horizontal convolution is performed first and the vertical convolution is performed on the result of the horizontal convolution.

This method can be implemented using the composition of two horizontal convolutions and a single horizontal convolution mask. The input $f[i,j]$ is first convolved with a horizontal Gaussian, and the result is placed in a temporary array in its transposed position. The temporary array is then used as input to the same convolution code so that the vertical convolution is performed by horizontal convolution. The output data from the second convolution is again transposed as the convolution is performed so that the data is restored to its proper (original) orientation. Results of this separable convolution are shown in Figure 4.13.

4.5.4 Cascading Gaussians

A related property of Gaussian filters is that the convolution of a Gaussian with itself yields a scaled Gaussian with larger σ. This is easily shown for

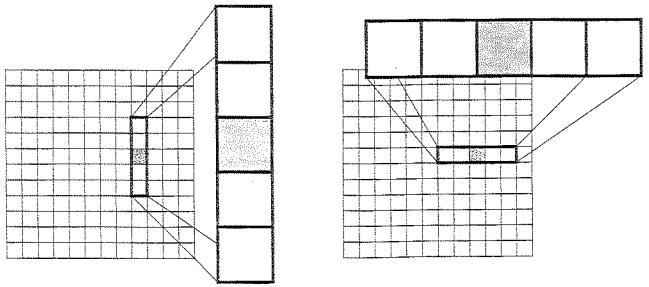


Figure 4.12: An example of the separability of Gaussian convolution. *Left:* Convolution with the vertical mask. *Right:* Convolution with the horizontal mask. Note that the origin of each mask is shaded.

the one-dimensional case:

$$\begin{aligned} g(x) \star g(x) &= \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{2\sigma^2}} e^{-\frac{(x-\xi)^2}{2\sigma^2}} \, d\xi \\ &= \int_{-\infty}^{\infty} e^{-\frac{(\frac{x}{2}+\xi)^2}{2\sigma^2}} e^{-\frac{(\frac{x}{2}-\xi)^2}{2\sigma^2}} \, d\xi, \quad \xi \to \xi + \frac{x}{2} \\ &= \int_{-\infty}^{\infty} e^{-\frac{(2\xi^2+\frac{x^2}{2})}{2\sigma^2}} \, d\xi \\ &= e^{-\frac{x^2}{4\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{\sigma^2}} \, d\xi \\ &= \sqrt{\pi}\sigma e^{-\frac{x^2}{2(\sqrt{2}\sigma)^2}}. \end{aligned} \tag{4.24}$$

The product of the convolution of two Gaussian functions with spread σ is a Gaussian function with spread $\sqrt{2}\sigma$ scaled by the area of the Gaussian filter. The result holds in two dimensions as well. This means that if an image has been filtered with a Gaussian at a certain spread σ and if the same image must be filtered with a larger Gaussian with spread $\sqrt{2}\sigma$, then instead of filtering the image with the larger Gaussian, the previous result can just be refiltered with the same Gaussian filter of spread σ used to obtain the desired filtered image. This implies a significant reduction in computation in situations where multiple smoothed versions of images must be computed. Similar savings can be obtained when cascading Gaussian filters with different values of σ.

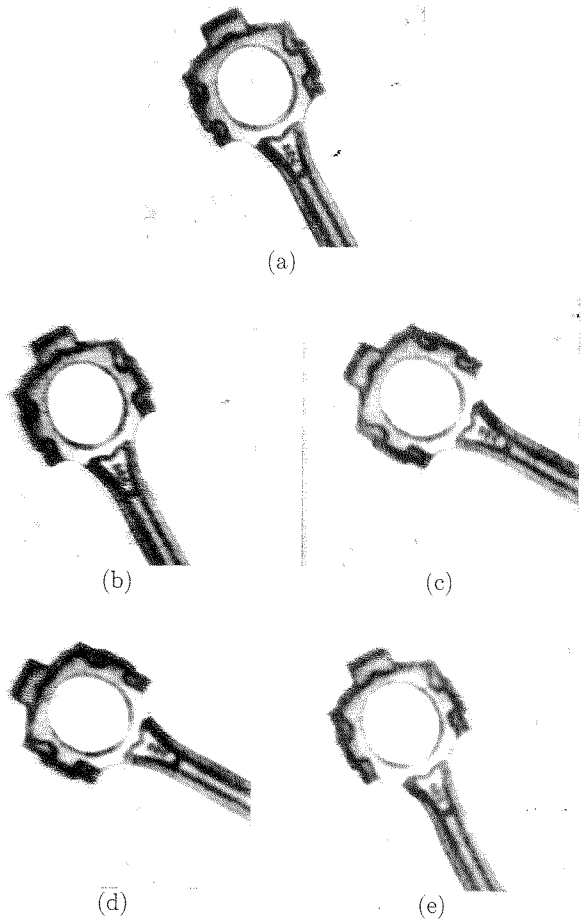


Figure 4.13: The results of separable Gaussian convolution using a single horizontal convolution mask. (a) Original noisy image. (b) Results of convolution with horizontal Gaussian mask. (c) The transposition of (b). (d) The convolution of (c) with the horizontal mask. (e) The transposition of (d). This is the final smoothed image.

4.5.5 Designing Gaussian Filters

An excellent approximation to a Gaussian is provided by the coefficients of the binomial expansion:

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n. \qquad (4.25)$$

In other words, use row n of Pascal's triangle as a one-dimensional, n-point approximation to a Gaussian filter. For example, the five-point approximation is:

1	4	6	4	1

corresponding to the fifth row of Pascal's triangle. This mask is used to smooth an image in the horizontal direction. Remember from Section 4.5.3 that a two-dimensional Gaussian filter can be implemented as the successive convolutions of two one-dimensional Gaussians, one in the horizontal direction and the other in the vertical direction. Also remember that this can be implemented using only the single one-dimensional mask by transposing the image between convolutions and after the final convolution. The results of Gaussian filtering using this approximation are shown in Figure 4.14.

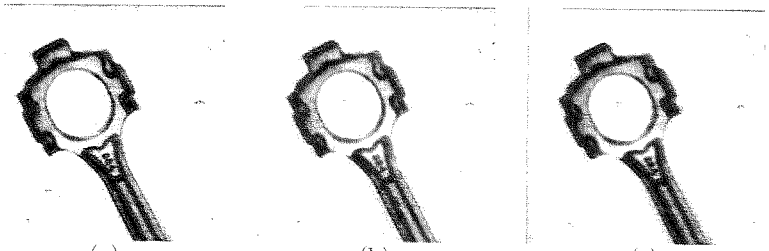
This technique works well for filter sizes up to around $n = 10$. For larger filters, the coefficients in the binomial expansion are too large for most computers; however, arbitrarily large Gaussian filters can be implemented by repeatedly applying a smaller Gaussian filter. The σ of the binomial approximation to a Gaussian filter can be computed by using least-squares to fit a Gaussian function to the binomial coefficients.

Another approach in designing Gaussian filters is to compute the mask weights directly from the discrete Gaussian distribution [146]:

$$g[i,j] = c\,e^{-\frac{(i^2+j^2)}{2\sigma^2}} \qquad (4.26)$$

where c is a normalizing constant. By rewriting this as

$$\frac{g[i,j]}{c} = e^{-\frac{(i^2+j^2)}{2\sigma^2}} \qquad (4.27)$$



(a) (b) (c)

Figure 4.14: The approximation of a Gaussian filter using the fifth row of Pascal's triangle. (a) Original noisy image. (b) Result after smoothing in the horizontal direction. (c) Final result after smoothing in the vertical direction.

and choosing a value for σ^2, we can evaluate it over an $n \times n$ window to obtain a kernel, or mask, for which the value at $[0,0]$ equals 1. For example, choosing $\sigma^2 = 2$ and $n = 7$, the above expression yields the array:

$[i,j]$	-3	-2	-1	0	1	2	3
-3	.011	.039	.082	.105	.082	.039	.011
-2	.039	.135	.287	.368	.287	.135	.039
-1	.082	.287	.606	.779	.606	.287	.082
0	.105	.368	.779	1.000	.779	.368	.105
1	.082	.287	.606	.779	.606	.287	.082
2	.039	.135	.287	.368	.287	.135	.039
3	.011	.039	.082	.105	.082	.039	.011

However, we desire the filter weights to be integer values for ease in computations. Therefore, we take the value at one of the corners in the array, and choose k such that this value becomes 1. Using the above example, we get

$$\frac{g[3,3]}{k} = e^{-\frac{(3^2+3^2)}{2(2)^2}} = 0.011 \implies k = \frac{g[3,3]}{0.011} = \frac{1.0}{0.011} = 91.$$

Now, by multiplying the rest of the weights by k, we obtain

$[i,j]$	-3	-2	-1	0	1	2	3
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

This is the resulting convolution mask for the Gaussian filter (also shown in Figure 4.15). However, the weights of the mask do *not* sum to 1. Therefore, when performing the convolution, the output pixel values must be normalized by the sum of the mask weights to ensure that regions of uniform intensity are not affected. From the above example,

$$\sum_{i=-3}^{3}\sum_{j=-3}^{3} g[i,j] = 1115.$$

Therefore,

$$h[i,j] = \frac{1}{1115}(f[i,j] \star g[i,j])$$

where the weights of $g[i,j]$ are all integer values. The results of Gaussian smoothing using the above mask are given in Figure 4.16. Other common Gaussian filter masks are given in Figure 4.17.

4.5.6 Discrete Gaussian Filters

The samples of a Gaussian filter, or the coefficients obtained from the binomial expansion, form a discrete Gaussian filter. When discrete Gaussian filters are convolved, the result is a larger discrete Gaussian filter. If an image is smoothed with an $n \times n$ discrete Gaussian filter, and this intermediate result is smoothed by an $m \times m$ discrete Gaussian filter, then the result is exactly the same as if the original image had been smoothed by an $(n+m-1) \times (n+m-1)$ discrete Gaussian filter. In other words, convolving row n in Pascal's triangle with row m yields row $n+m-1$ in Pascal's triangle.

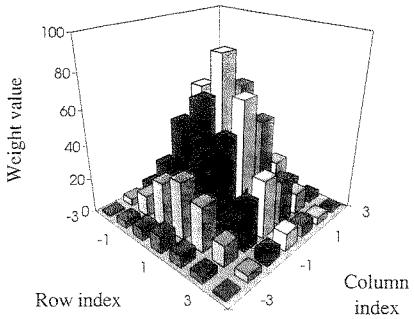
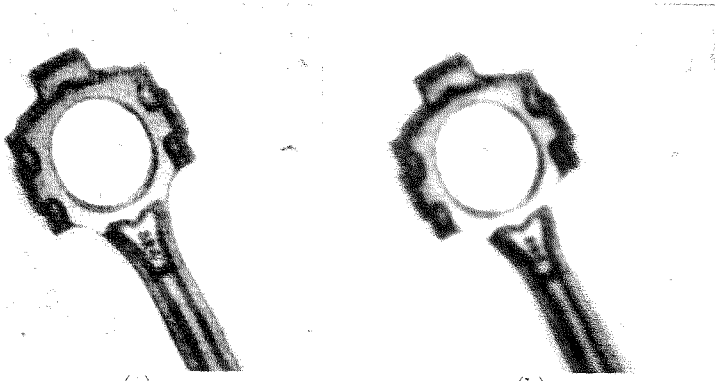


Figure 4.15: A 3-D plot of the 7×7 Gaussian mask.



(a) (b)

Figure 4.16: The results of smoothing using the 7×7 Gaussian mask. (a) Original image corrupted by Gaussian noise. (b) Smoothed image.

7×7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

15×15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

Figure 4.17: Other commonly used Gaussian masks [146].

Further Reading

Rosenfeld and Kak [205] provide a brief discussion of histogram modification. Books by Pratt [195] and Gonzalez and Woods [90] on image processing include material on histogram methods, median filters, and linear filters.

There is a discussion of linear systems theory, both continuous and discrete, and linear systems for image processing in the book by Horn [109]. Digital signal processing on two dimensions is covered by Oppenheim and Shafer [190] and by Rabiner and Gold [199]. The relationship between digital filters and numerical methods is discussed by Hamming [96]. More detailed explanations of this property and its use are provided by Crowley and Stern [65] and Burt [52].

Exercises

4.1 A mean filter is a linear filter, but a median filter is not. Why?

4.2 Compare the characteristics of median and mean filters and identify the situations where you will use them.

4.3 Gaussian filtering is usually a preferred averaging method. Why?

4.4 What is the separability property of Gaussian filtering? Why would you want a filtering scheme to be separable?

4.5 In many applications, an image is smoothed by applying Gaussian filters of several sizes. Why would one want to smooth an image using different parameters of the Gaussian?

4.6 What is the cascading property of Gaussian filters? How is it useful in machine vision?

4.7 An image contains a thin vertical line one pixel thick. It has a gray level of 50 and lies on a background of salt and pepper noise having gray values of 0 and 100, where

$$\text{Probability}(\text{gray level} = 0) = 0.4$$
$$\text{Probability}(\text{gray level} = 100) = 0.6.$$