A. Distance

Many operations on images such as linear filtering are neighborhood operations. For this reason, the distance between pixels plays an important role in image processing. It is possible to use different definitions of the distance between pixels \((i, j)\) and \((h, k)\) (metrics):

- Euclidean distance:
  \[
  D_e = \sqrt{(i - h)^2 + (j - k)^2}
  \]  
  (1)

- City block distance
  \[
  D_c = |i - h| + |j - k|
  \]  
  (2)

A comparison is shown in figure 1.

I. PIXEL ADJACENCY

- Any pixel \((i, j)\) has two vertical and two horizontal neighbors. Each one of them is at 1 unit distance away from \((i, j)\).
- Any pixel \((i, j)\) has four diagonal neighbors. Each one of them is at Euclidean distance of \(\sqrt{2}\) away from \((i, j)\).
- The direct horizontal, vertical and the diagonal neighbors of \((i, j)\) together form 8 neighbors called 8–adjacency.

A. Noise in images

Noise can occur during image capture, transmission, or processing. Usually because we do not have the exact knowledge of the characteristics of the noise, we use probabilistic models, i.e., probability density functions. Noise can be dependent or independent of the image signal. Clearly, it is much easier to deal with independent noise.

B. Gaussian and impulse noise

Gaussian and impulse models are among the most widely used models in image processing.

- Additive noise: \(\hat{f}(i, j) = f(i, j) + noise\)
- Multiplicative noise: \(f(i, j) = f(i, j) \times noise\)

Gaussian noise: The noise has a Gaussian distribution. This distribution is widely used for modeling natural processes. The PDF of the Gaussian noise is given by

\[
PDF = \frac{2}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]  
  (3)

where
- $z$: gray-level
- $\mu$: mean value of $z$
- $\sigma$: standard deviation

- Impulse noise: some pixels are very different from their neighbors. Impulse noise is usually caused by transmission errors, faulty CCD sites etc. Example: salt and pepper noise. The PDF of the impulse noise given by

$$
\delta = \begin{cases} 
P_a & \text{for } z = a \\
P_b & \text{for } z = b \\
0 & \text{otherwise}
\end{cases}
$$

(4)

- An illustration of the Gaussian and salt and pepper noise is shown in figures 4 and 5. Figure 3 shows the original image and figure 6 shows the histograms.

### C. Linear filtering

The image may be corrupted by noise. Image enhancement aims at eliminating undesirable characteristics such as noise and poor contrast. One way to enhance images is by using filters. Filters can be used for many operations:

- sharpen details
- remove noise
- image smoothing
- detect edges in an image, etc.

There are two different ways to perform linear filtering:

- in the time domain (actually space domain): the filtering operation is a convolution. Spatial techniques operate on the pixels of an image directly,
- in the frequency domain. A convolution in the space domain becomes a multiplication in the frequency domain.

Linear filtering is simply a convolution that involves operations on neighbors. It is a neighborhood operation.

Consider figure 7. When the input to the system is an impulse $\delta$, the output $h$ is the impulse response. A linear space invariant (LSI) system can be described by its impulse response $h$. In LSI systems, the input and output are related as follows

$$
g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l)$$

(5)

or

$$
g(i, j) = \sum_{k,l} f(k, l)h(i - k, j - l)$$

(6)

This operation is called convolution. The notation for convolution is:

$$
g = f * h$$

(7)

where $h$ is the impulse response function of the filter convolution mask, that is

$$
h * \delta = h$$

(8)

- Mathematically, a filter is characterized by its impulse response, $h$, which can be seen as the transfer function of the filter.
- Numerically, a filter is characterized by its mask, which is a simple matrix. The general form for a 3 by 3 convolution kernel looks like

$$
\begin{bmatrix}
h_{11} & h_{12} & h_{13} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & h_{33}
\end{bmatrix}
$$

(9)

or

$$
\frac{1}{\sum h(i,j)} \begin{bmatrix}
h_{11} & h_{12} & h_{13} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & h_{33}
\end{bmatrix}
$$

(10)

where the sum of the weights is normalized to one.

- Common masks are
  - $3 \times 3$
  - $1 \times 3$
  - $3 \times 1$
  - $1 \times 5$
  - $5 \times 1$
  - $2 \times 2$
  - $5 \times 5$

The size of the filter determines the amount of filtering

Large masks $\implies$ greater degree of filtering $\implies$ loss of image details

(11)

- Examples of kernels
Consider the filter given by
\[
I_d = \begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\] (12)
What is the effect of this Kernel on the image? The answer is that it leaves the image unchanged. You can look at it as an identity filter.

Consider the filter given by
\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{bmatrix}
\] (13)
This filter shifts the image horizontally.

### D. Numerical convolution

- Convolution is the treatment of a matrix by another one called “kernel”.

#### E. Example

We want to perform a convolution on the following 3 by 3 image
\[
f = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\] (14)
with mask
\[
h = \begin{bmatrix}
0 & 1 & 0 \\
1 & 3 & 1 \\
0 & 1 & 0 \\
\end{bmatrix}
\] (15)
If we write the result of the convolution operation as
\[
g = f * h = \begin{bmatrix}
b_1 & b_2 & b_3 \\
b_4 & b_5 & b_6 \\
b_7 & b_8 & b_9 \\
\end{bmatrix}
\] (16)
then, \(b_1, b_2, b_5\) are given by:
\[
b_1 = 3 \times 1 + 1 \times 2 + 1 \times 4 = 9 \\
b_5 = 3 \times 5 + 1 \times 4 + 1 \times 6 + 1 \times 8 = 35 \\
b_2 = 1 \times 1 + 2 \times 3 + 3 \times 1 + 3 \times 5 = 15
\] (17)
and the full matrix is given by:
\[
g = f * h = \begin{bmatrix}
0 & 1 & 2 & 3 & 0 \\
1 & 9 & 15 & 17 & 3 \\
4 & 25 & 35 & 35 & 6 \\
7 & 33 & 45 & 41 & 9 \\
0 & 7 & 8 & 9 & 0 \\
\end{bmatrix}
\] (18)

- Linear filtering is a convolution operation where the output is a weighted sum of the neighboring pixels.
- Any filter that is not a weighted sum is a nonlinear filter

#### F. How do frequencies appear in a image?

- Low frequencies in the image appear as slow variations of the intensities. Low frequencies correspond to long wavelengths.
- High frequencies appear as abrupt changes in image intensity such as edges and corners. High frequencies correspond to short wavelengths.
- Low pass filter: This filter passes low frequencies where the image intensity changes slowly, and blocks high frequencies where intensity changes quickly.
Fig. 8. The convolution operation

- High pass filter: This filter keeps high frequency components where intensity changes abruptly, and removes low frequency components where intensity changes slowly.
- What is the effect of a low pass filter? Blurring or smoothing of the image and the sharp details in the image are lost.
- What is the effect of a high pass filter? Sharpen the image, and therefore, edges are more visible.

G. Mean filter

A mean filter is a smoothing filter where each pixel is replaced by the average value of its neighbors, that is

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

where \( N \) is the neighborhood and \( M \) is the number of pixels in the neighborhood.

- **Example 1**
  For a 3 x 3 neighborhood we have
  \[
  g(i, j) = \frac{1}{9} \sum_{k+i=1}^{i+1} \sum_{l+j=1}^{j+1} f(k,l)
  \]

- **Example 2**
  Find the mask of a 3 by 3 mean filter.

\[
\begin{align*}
g(i, j) &= \frac{1}{9} \left[ f(i-1, j-1) + f(i-1, j) + f(i-1, j+1) + f(i, j-1) + f(i, j) + f(i, j+1) + f(i+1, j-1) + f(i+1, j) + f(i+1, j+1) \right] \\
\text{and therefore} \\
mask &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}
\end{align*}
\]

This means that a mean filter can be implemented as a convolution operation with equal weights in the convolution masks.

H. Smoothing filter

A smoothing filter is a low pass filter, i.e., removes high frequency components. Sharp details in the image are lost. A typical 3 x 3 smoothing filter is given by

\[
mask = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

Three important remarks about this filter
- The sum of all entries is 1.
- Symmetry in the horizontal and vertical directions.
- The filter has a single peak in the middle called “main lobe”. This means more importance is given to the center pixel.

I. Gaussian filter

Gaussian filters are a class of low pass linear filters. They can be used for removing Gaussian noise and smoothing of digital images. A two dimensional discrete Gaussian filter is given by

\[
h(i, j) = e^{-\frac{i^2+j^2}{2\sigma^2}}
\]

where \( \sigma \) is the standard deviation.

J. Some properties of the Gaussian filter

- In 2D, Gaussian filters are rotationally symmetric. This implies that the filtering operation is independent of the direction. Converting the Gaussian filter from rectangular to polar coordinates yields

\[
h(r, \theta) = e^{-\frac{r^2}{2\sigma^2}}
\]

with

\[
r^2 = i^2 + j^2
\]

Equation (25) does not depend on the angle, and thus filtering is independent of the direction. The disadvantage with this is that it is not possible to target a specific direction for filtering.

- Gaussian filter has a single main lobe at the center with weights decreasing with the distance from the central pixel. This means that more significance is given to the center pixel and its closest neighbors.

- The degree of smoothing (filtering) is determined by \( \sigma \). A larger \( \sigma \) implies a wider filter and thus greater smoothing.

- The horizontal and vertical directions in the Gaussian filter are separable. Thus it is possible to perform smoothing in one direction and then in the other one.

- Example
  Prove the separability property of the Gaussian filter.

K. Designing Gaussian filters

- Pascal’s triangle is a good way to approximate the coefficients. For example, from the 5th row of the Pascal’s
Machine vision FB

Fig. 9. Pascal Triangle coefficients

triangle shown in figure 9, it is possible to approximate a five point 1D Gaussian filter using

\[
\begin{bmatrix}
1 & 4 & 6 & 4 & 1
\end{bmatrix}
\]  \hspace{1cm} (27)

- It is also possible to compute the weights of the mask directly from the discrete Gaussian distribution. The obtained matrix is then normalized so that the sum is 1. The normalized filter can be written as

\[
h(i, j) = Ce^{-\frac{i^2+j^2}{2\sigma^2}} \hspace{1cm} (28)
\]

where \( C \) is the normalization coefficient.

- Example: With \( \sigma = 1 \), it is possible to obtain the following 3 by 3 Gaussian filter:

\[
\begin{bmatrix}
0.3679 & 0.6065 & 0.3679 \\
0.6065 & 1.0000 & 0.6065 \\
0.3679 & 0.6065 & 0.3679
\end{bmatrix}
\]  \hspace{1cm} (29)

which becomes after normalization:

\[
\begin{bmatrix}
0.0751 & 0.1238 & 0.0751 \\
0.1238 & 0.2042 & 0.1238 \\
0.0751 & 0.1238 & 0.0751
\end{bmatrix}
\]  \hspace{1cm} (30)

L. High pass filter

- There is a relationship between high pass and low pass filters

\[
\text{High pass filtered image} = \text{original image} - \text{low pass filtered image} \hspace{1cm} (31)
\]

This can be expressed as follows

\[
H_p * f = I_d * f - L_p * f = (I_d - L_p) * f \hspace{1cm} (32)
\]

with

- \( H_p \): high pass filter mask
- \( L_p \): low pass filter mask
- \( f \): original image

Note that \( I_d \) is not an identity matrix.

- Example

Find the mask of the high pass filter corresponding to the low pass filter given by (30).

We have

\[
H_p = I_d - \text{low pass} \hspace{1cm} (34)
\]

and thus

\[
H_p = \begin{bmatrix}
-0.0751 & -0.1238 & -0.0751 \\
-0.1238 & 0.7958 & -0.1238 \\
-0.0751 & -0.1238 & -0.0751
\end{bmatrix}
\]  \hspace{1cm} (35)

II. MEDIAN FILTER

The median filter replaces each pixel in the image by the median value of its neighbors. Median filters are very effective in removing impulse noise including salt and pepper noise. There are two steps to perform median filtering:

- Sort pixels in ascending order.
- Select the value of the middle pixel as the new value.

A. Example

The original image is given by:

\[
I_m = \begin{bmatrix}
156 & 131 & 125 \\
225 & 96 & 89 \\
7 & 199 & 202
\end{bmatrix}
\]  \hspace{1cm} (36)

After sorting the pixels, we get

\[
\begin{bmatrix}
7 & 89 & 96 & 125 & 131 & 156 & 199 & 202 & 225
\end{bmatrix}
\] \hspace{1cm} (37)

and the new image is given by

\[
I_m\text{New} = \begin{bmatrix}
156 & 131 & 125 \\
225 & 131 & 89 \\
7 & 199 & 202
\end{bmatrix}
\]  \hspace{1cm} (38)

B. Median filter and salt and pepper noise

Median filters are very effective in removing salt and pepper noise. Consider the following example where one pixel has a very different value:

\[
I_m = \begin{bmatrix}
1 & 2 & 3 \\
6 & 255 & 7 \\
7 & 9 & 4
\end{bmatrix}
\]  \hspace{1cm} (39)

The result of the median filter is

\[
I_m\text{New} = \begin{bmatrix}
1 & 2 & 3 \\
6 & 6 & 7 \\
7 & 9 & 4
\end{bmatrix}
\]  \hspace{1cm} (40)

and the salt noise pixel is removed.

Since the median filter uses a sorting algorithm, it is relatively expensive and complex to compute.

RANK FILTERS

If we consider a neighborhood of \( N \) pixels, a filter of rank \( k \) will arrange the pixels in the neighborhood in ascending order from smallest (\( M_0 \)) to largest (\( M_N \)) gray level value and assign the \( k^{th} \) value to the center point of the window. Rank filters \( k = 1 \) and \( k = N \) are called minimum and maximum filters, respectively. The median filter is a particular case with \( k = \text{floor}(N/2) + 1 \) where \( \text{floor}(X) \) rounds \( X \) to the nearest integer towards minus infinity.

III. SOME USEFUL FUNCTIONS

- \textbf{imfilter} can be used to perform filtering in the space domain. The syntax for this function is

\[
B = \text{imfilter}(A,H)
\]  \hspace{1cm} (41)

where \( A \) is the image and \( H \) is the filter.
• **fspecial** can be used to create a two-dimensional filter (mask). The general syntax for **fspecial** is

\[
H = \text{fspecial}(\text{type})
\]  

where \(\text{type}\) stands for the type of the filter, which includes
- ‘average’: averaging filter
- ‘disk’: circular averaging filter
- ‘gaussian’: Gaussian lowpass filter
- ‘unsharp’: unsharp contrast enhancement filter

**fspecial** may take additional parameters depending on the type of the filter. For example, for a Gaussian filter, the parameters are the size and the standard deviation.

• **imnoise** can be used to add noise to an image. Its syntax is

\[
J = \text{imnoise}(I, \text{type}, ...)
\]  

where \(\text{type}\) is a string that can have one of these values:
- ‘gaussian’: Gaussian white noise with constant mean and variance.
- ‘localvar’: zero-mean Gaussian white noise with an intensity-dependent variance
- ‘salt & pepper’: on and off pixels
- ‘speckle’: multiplicative noise

**imnoise** may take additional parameters related to the type of the noise.