MA3111: Mathematical Image Processing
Intensity Transformations and Spatial Filtering

Suh-Yuh Yang (楊肅煜)

Department of Mathematics, National Central University
Jhongli District, Taoyuan City 320317, Taiwan

First version: April 13, 2021/Last updated: September 3, 2024

Spatial domain and transform domain

The spatial domain approach and transform domain approach are two main categories in image processing:

- *Spatial domain:* refers to the image plane itself, and image processing methods in this category are based on direct manipulation of pixels in an image.
- *Transform domain:* involves first transforming an image into the transform domain, doing the processing there, and obtaining the inverse transform to bring the results back into spatial domain.

Outline of “intensity transformations & spatial filtering”

In this lecture, we will discuss a number of classical techniques for two principal categories of spatial domain processing:

- *Intensity transformations (强度變換):* operate on single pixels of an image for tasks such as contrast manipulation and image thresholding.
- *Spatial filtering (空間濾鏡):* performs operations on the neighborhood of every pixel in an image. Examples of spatial filtering include image smoothing and sharpening.

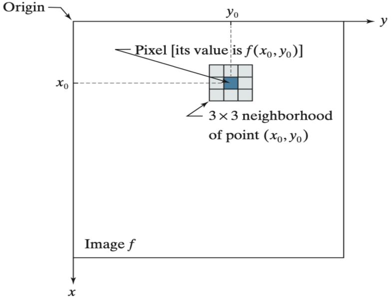
The material of this lecture is based on Chapter 3 in [GW2018].

Spatial domain process

The spatial domain process is generally posed in the form:

$$g(x,y) = T(f(x,y)),$$

where $f(x,y)$ is an input image, $g(x,y)$ is the output image, and T is an operator on f defined over a neighborhood (typically a rectangle) of point (x,y).



A 3×3 neighborhood about the point (x_0, y_0). The neighborhood is moved from pixel to pixel in the image to generate the output image.

Spatial filtering and intensity transformation

- **A smoothing spatial filter T:** Suppose that the neighborhood is a square of size 3×3 and that operator T is defined as *compute the average intensity of the pixels in the neighborhood. Then T is a smoothing filter (平滑濾波器).*

  Consider an arbitrary location in an image f, say $(100,150)$. Then

  $$g(100,150) = T\big(f(100,150)\big) = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1} f(100-i,150-j).$$

  (A neighborhood processing technique)

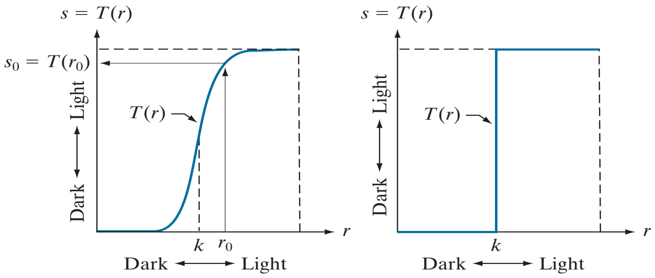
- **Intensity transformation:** The smallest possible neighborhood is of size 1×1. T becomes an intensity transformation of the form

  $$g(x,y) =: s = T(r) := T(f(x,y)).$$

  (A point processing technique)

Intensity transformations

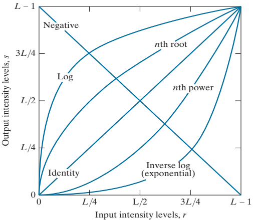
- **Contrast stretching function:** "left figure" produces an image of higher contrast than the original, by darkening the intensity levels below k and brightening the levels above k.
- **Thresholding function:** In the limiting case shown in "right figure," $T(r)$ produces a two level (binary) image.

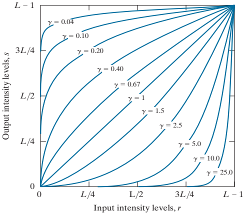


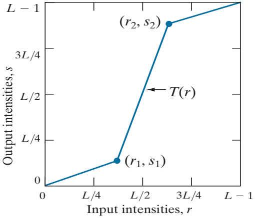
$$g(x,y) =: s = T(r) := T(f(x,y))$$

Some examples: $g(x,y) =: s = T(r) := T(f(x,y))$

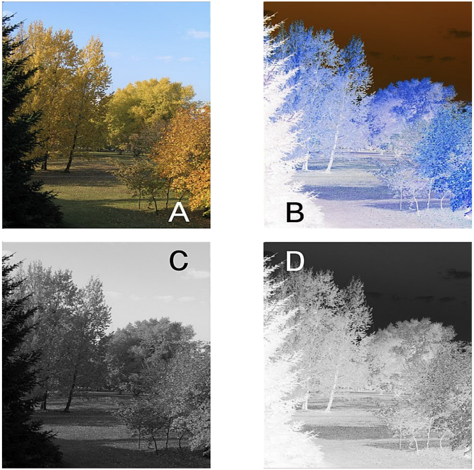
- **Negative transformation:** The negative of an image with intensity levels in the range $[0, L-1]$ is obtained by $s = L - 1 - r$.
- **Log transformation:** $s = c\log(1 + r)$, where $c > 0$ is a constant.
- **Power-law (gamma) transformation:** $s = cr^{\gamma}$, where c and γ are positive constants. *Note that inputs and outputs are typically normalized in the range $[0,1]$, i.e., $r \in [0,1]$.*
- **Piecewise linear transformation**





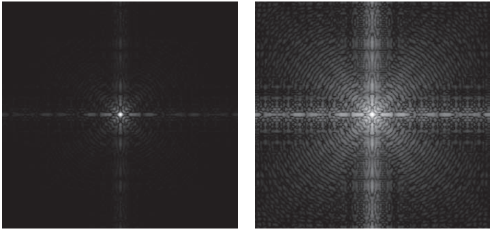


Negative images (負片)

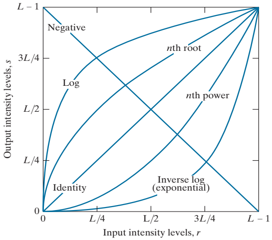


Color image: B is the negative image of positive image A;
Grayscale image: D is the negative image of positive image C.
(cited from Wikipedia)

Log image

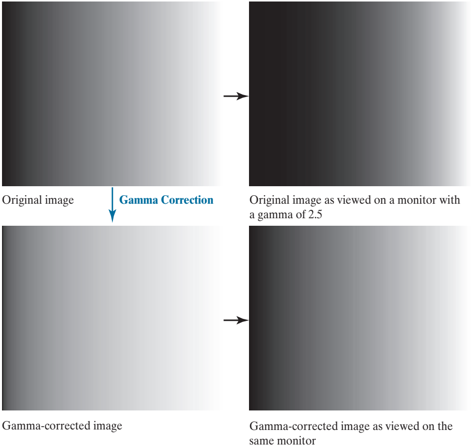


Log transformation of Fourier spectrum with $c = 1$



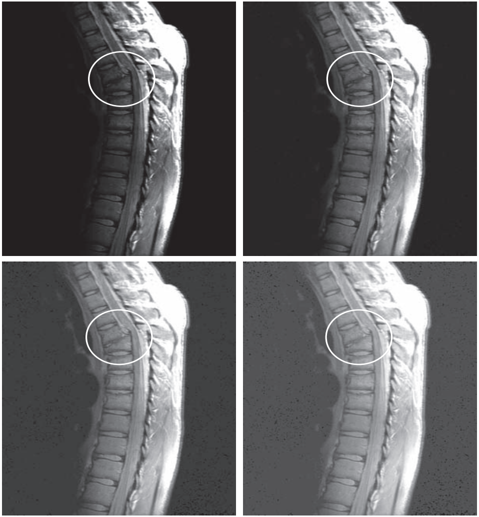
Images of gamma transformation

Many devices used for image capture, printing, and display obey a power law, e.g., cathode ray tube (CRT, 陰極射線管、映像管)



Intensity ramp images with $c = 1$, $\gamma = 2.5$ *and correction* $s = r^{1/(2.5)}$

Gamma transformation: MRI of a fractured human spine



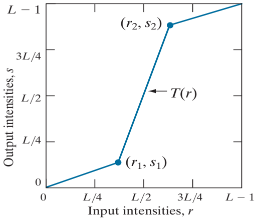
Region of the fracture is enclosed by the circle: $c = 1,\ \gamma = 0.6, 0.4, 0.3$

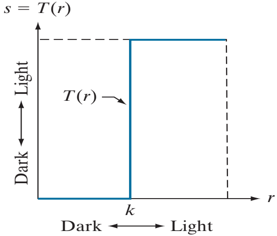
Gamma transformation: aerial images (空拍影像)

$c = 1, \gamma = 3.0, 4.0, 5.0$

Piecewise linear transformation: contrast stretching

A low-contrast electron microscope image of pollen (花粉); Result of contrast stretching; Result of thresholding

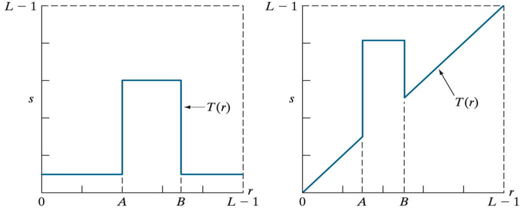




(Right) thresholding function: $r_1 = r_2 = k, s_1 = 0, s_2 = L - 1$

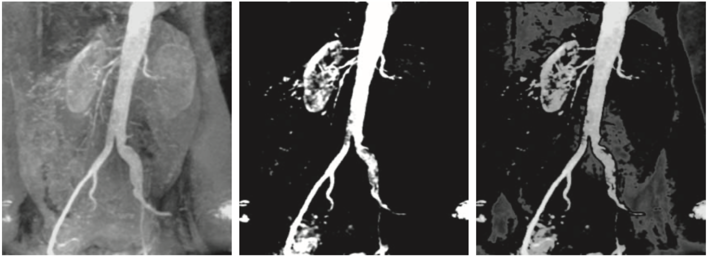
Intensity-level slicing (强度準位切片)

- *Intensity-level slicing is to highlight a specific range of intensities in an image,* e.g., enhancing features in satellite imagery such as masses of water, and enhancing flaws in X-ray images.
- One approach is to display in one value (say, white) all the values in the range of interest and in another (say, black) all other intensities, i.e., produces a binary image.
- The second approach brightens (or darkens) the desired range of intensities, but leaves all other intensity levels in the image unchanged.



(Left) first approach; (Right) second approach

Examples of the intensity-level slicing



(L) aortic angiogram (X-ray photograph); (M) first approach; (R) second approach, with the selected range set near black

Histogram (直方圖)

- Let r_k, for $k = 0, 1, \cdots, L-1$, denote the intensities of an L-level image $f(x, y)$. The unnormalized histogram of f is defined as

$$h(r_k) = n_k, \quad k = 0, 1, \cdots, L-1,$$

  where n_k is the number of pixels in f with intensity r_k.

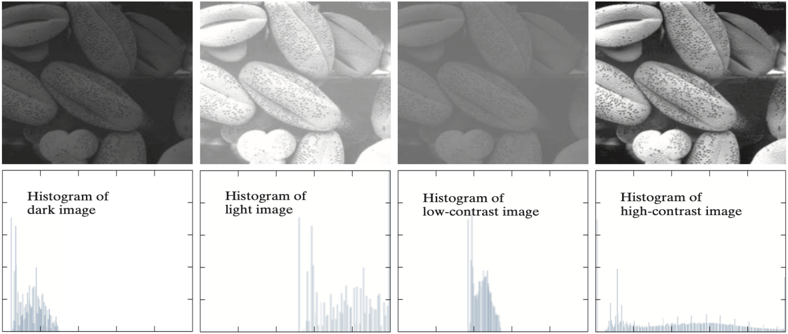
- The normalized histogram of f is defined as

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN},$$

  where f is an $M \times N$ image. That is, $p(r_k)$ is the *probability* of intensity level r_k occurring in an image. Then $\sum_{k=0}^{L-1} p(r_k) = 1$.

- Histograms are simple to compute and are also suitable for fast hardware implementations, thus making histogram-based techniques a popular tool for real-time image processing.

Four image types and their corresponding histograms



The horizontal axis of the histograms are values of r_k
and the vertical axis are values of $p(r_k)$

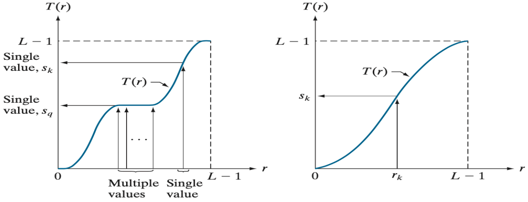
Intensity transformation

Let the variable r denote the intensities of an image to be processed. Assume that $r \in [0, L-1]$ with $r = 0$ representing black and $r = L-1$ representing white. We consider the intensity transformation

$$s = T(r), \quad 0 \le r \le L-1.$$

For a given intensity value r in the input image, T produces an output intensity value s. We assume that

- *$T(r)$ is a monotonic increasing function in the interval $[0, L-1]$.*
- *$T(r) \in [0, L-1]$ for all $r \in [0, L-1]$.*
- *If we need to use the inverse $r = T^{-1}(s)$, $s \in range(T)$, then we assume $T(r)$ is a strictly increasing function in $[0, L-1]$.*



Histogram equalization (HE): $g(x,y) =: s = T(r) := T(f(x,y))$

- We are given a grayscale image $f : \overline{\Omega} \to [0,1]$. The cumulative histogram *(cumulative distribution function)* T is defined by considering f as a random variable: for $\eta \in [0,1]$, we define

$$\begin{aligned} T(\eta) &:= Prob(f \le \eta) \\ &= \frac{1}{|\overline{\Omega}|} \Big| \{(x,y) \in \overline{\Omega} : f(x,y) \le \eta\} \Big|. \end{aligned}$$

  Then $T : [0,1] \to [0,1]$ is a monotonic increasing function.

- The histogram equalized image $g : \overline{\Omega} \to [0,1]$ is obtained by defining

$$g(x,y) := T(f(x,y)).$$

Histogram equalized image $g \sim \mathcal{U}(0,1)$ if T is invertible

If T is strictly increasing, then T is invertible and the *cumulative distribution function* of the histogram equalized image g is

$$\begin{aligned} Prob(g \le \eta) &= Prob(T(f) \le \eta) = Prob(f \le T^{-1}(\eta)) \\ &= T(T^{-1}(\eta)) = \eta. \end{aligned}$$

Hence, the *probability density function* of g is

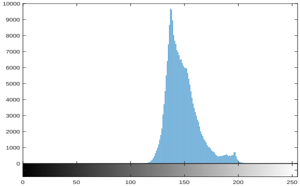
$$p(t) = \begin{cases} 1 & \text{for } 0 \le t \le 1, \\ 0 & \text{elsewhere.} \end{cases}$$

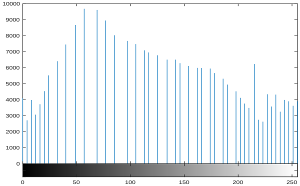
Therefore, g has a uniform distribution, i.e., $g \sim \mathcal{U}(0,1)$.

Remark: Let X be a random variable and $p(t)$ the probability density function (pdf) of X. The cumulative distribution function (cdf) of X is

$$F(\eta) := Prob(X \le \eta) = \int_{-\infty}^{\eta} p(t)\, dt.$$

Example of histogram equalized image

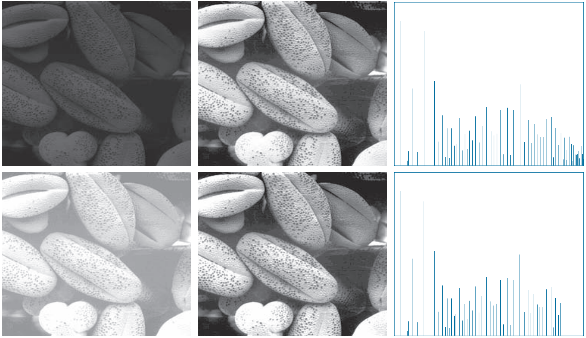




Histogram equalization of 400×600 *image:*
(top) before; (bottom) after; and the corresponding histograms

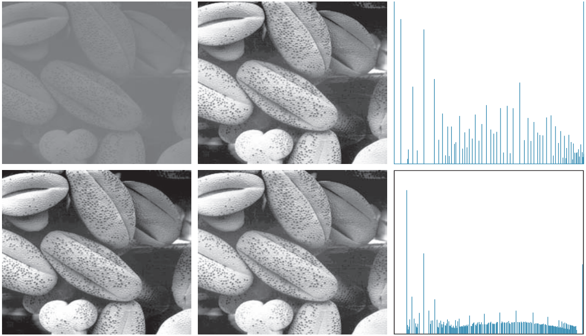
Matlab commands: `imhist(A)`, `histeq(A)`,
`histogram(A,'Normalization','probability')`

Histogram-equalized images



Histogram-equalized images and the corresponding normalized histograms

Histogram-equalized images (cont'd)

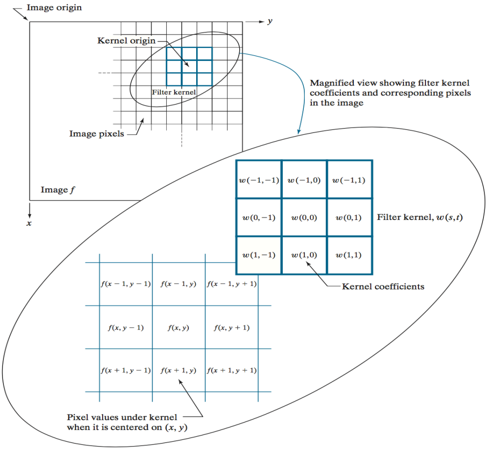


Histogram-equalized images and the corresponding normalized histograms

Spatial filter (空間濾波)

- *Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbors.* (discrete!)
- If the operation performed on the image pixels is linear, then the filter is called a linear spatial filter. Otherwise, the filter is a nonlinear spatial filter.
- *A linear spatial filter performs a sum-of-products operation between an image f and a filter kernel w.* The kernel is an array whose size defines the neighborhood of operation, and whose entries determine the nature of the filter.
- Other terms used to refer to a spatial filter kernel are *mask, template,* and *window.* We use the term *"filter kernel"* or simply *"kernel."*

Filter kernel of a linear spatial filter



Linear spatial filtering using a 3×3 *kernel*

Linear spatial filtering

1. **The spatial correlation (空間相關):**

   (1) 3×3 *kernel:* at any point (x, y) in the image f, the response $g(x, y)$ of the filter is the sum of products of the kernel entries and the image pixel values:

$$\begin{aligned} g(x,y) &= w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots \\ &\quad + \underbrace{w(0,0)f(x,y)} + \cdots + w(1,1)f(x+1,y+1). \end{aligned}$$

   As x and y are varied, the *center of the kernel* moves from pixel to pixel, generating the filtered image g.

   (2) $m \times n$ *kernel:* Assume that $m = 2p + 1$ and $n = 2q + 1$. Then

$$g(x,y) = \sum_{i=-p}^{p} \sum_{j=-q}^{q} w(i,j) f(x+i, y+j).$$

2. **The spatial convolution (空間卷積, $*$ or $\circledast$):** The mechanics are the same, except that the kernel is *rotated by 180° counterclockwise.*

Convolution of two functions: continuous cases

- **1-D case:** Let f and g be two integrable real-valued functions with compact supports in $\mathbb{R}$. Then the convolution of f and g is defined as a function in variable t,

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau, \quad \forall\, t \in \mathbb{R}.$$

  It can be shown on the next page that

$$(f * g)(t) = \int_{-\infty}^{\infty} g(\eta)f(t-\eta)\,d\eta = (g * f)(t), \quad \forall\, t \in \mathbb{R},$$

  and then the operation can be described as *a weighted average of the input f at t according to the weight function (or kernel) g.*

- **2-D case:** Let f and g be two integrable real-valued functions with compact supports in $\mathbb{R}^2$. Then the convolution of f and g is defined as a function in variable $\boldsymbol{x}$,

$$(f * g)(\boldsymbol{x}) := \int_{\mathbb{R}^2} f(\boldsymbol{y})g(\boldsymbol{x}-\boldsymbol{y})\,d\boldsymbol{y}, \quad \forall\, \boldsymbol{x} \in \mathbb{R}^2.$$

- $f * g = g * f$, $(f * g) * h = f * (g * h)$, $f * (g + h) = (f * g) + (f * h)$

Commutativity: $f * g = g * f$

$\because$ f and g are two integrable functions with compact supports in $\mathbb{R}$.

$\therefore$ $\exists\, L > 0$ such that $f(t) = 0 = g(t)$ for $t \notin [-L, L]$.

$$\therefore\ (f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau = \int_{-L}^{L} f(\tau) g(t-\tau)\, d\tau, \quad \forall\, t \in \mathbb{R}$$

Let $\eta = -(\tau - t)$. Then $\tau = t - \eta$ and $d\eta = -d\tau$, and we have

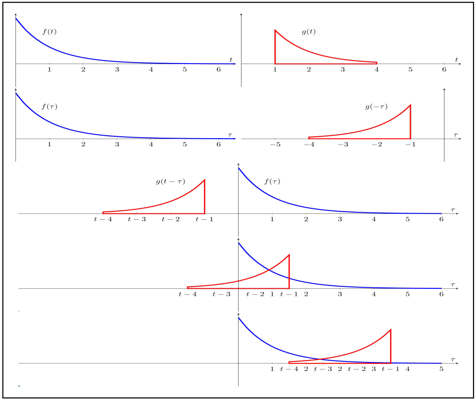
$$\int_{-L}^{L} f(\tau) g(t-\tau)\, d\tau = \int_{t+L}^{t-L} f(t-\eta) g(\eta)(-d\eta) = \int_{t-L}^{t+L} f(t-\eta) g(\eta)\, d\eta.$$

If $t \geq 0$, then $\displaystyle\int_{t-L}^{t+L} f(t-\eta) g(\eta)\, d\eta = \int_{-L}^{L} g(\eta) f(t-\eta)\, d\eta = (g * f)(t)$.

If $t < 0$, then $\displaystyle\int_{t-L}^{t+L} f(t-\eta) g(\eta)\, d\eta = \int_{-L}^{L} g(\eta) f(t-\eta)\, d\eta = (g * f)(t)$.

$\therefore$ $(f * g)(t) = (g * f)(t), \quad \forall\, t \in \mathbb{R}$

Convolution of two 1-D functions



$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau, \quad t \in \mathbb{R}$$

Wikipedia: https://en.wikipedia.org/wiki/Convolution

Discrete convolution: 1-D

The discrete convolution of input (signal) f and kernel g is defined by

$$(f * g)(t) := \sum_{\tau=-\infty, \tau \in \mathbb{Z}}^{\infty} f(\tau) g(t - \tau), \quad t \in \mathbb{Z}.$$

- When f and g have finite supports, a finite summation may be used.
- f and g can be viewed as piecewise constant functions in each unit integer interval.

Correlation vs. convolution: 1-D example

Correlation

Origin f w

0 0 0 **1** 0 0 0 0   **1 2 4 2 8**

0 0 0 **1** 0 0 0 0
1 2 4 2 8
Starting position alignment

Zero padding

0 0 0 0 0 **1** 0 0 0 0 0 0
1 2 4 2 8
Starting position

0 0 0 0 0 **1** 0 0 0 0 0 0
1 2 4 2 8
Position after 1 shift

0 0 0 0 0 **1** 0 0 0 0 0 0
1 2 4 2 8
Position after 3 shifts

0 0 0 0 0 **1** 0 0 0 0 0 0
1 2 4 2 8
Final position

Correlation result

0 8 2 4 2 1 0 0

Extended (full) correlation result

0 0 0 8 2 4 2 1 0 0 0 0

Convolution

Origin f w rotated 180°

0 0 0 **1** 0 0 0 0   **8 2 4 2 1**

0 0 0 **1** 0 0 0 0
8 2 4 2 1
Starting position alignment

Zero padding

0 0 0 0 0 **1** 0 0 0 0 0 0
8 2 4 2 1
Starting position

0 0 0 0 0 **1** 0 0 0 0 0 0
8 2 4 2 1
Position after 1 shift

0 0 0 0 0 **1** 0 0 0 0 0 0
8 2 4 2 1
Position after 3 shifts

0 0 0 0 0 **1** 0 0 0 0 0 0
8 2 4 2 1
Final position

Convolution result

0 1 2 4 2 8 0 0

Extended (full) convolution result

0 0 0 1 2 4 2 8 0 0 0 0

1-D discrete full convolution

Let $\boldsymbol{u} = [u_1, \cdots, u_n]^\top \in \mathbb{R}^n$ and $\boldsymbol{v} = [v_1, \cdots, v_m]^\top \in \mathbb{R}^m$. The convolution of $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\boldsymbol{u} * \boldsymbol{v} := \begin{bmatrix} u_1 v_1 \\ u_1 v_2 + u_2 v_1 \\ u_1 v_3 + u_2 v_2 + u_3 v_1 \\ \vdots \\ u_{n-2} v_m + u_{n-1} v_{m-1} + u_n v_{m-2} \\ u_{n-1} v_m + u_n v_{m-1} \\ u_n v_m \end{bmatrix} \in \mathbb{R}^{m+n-1}.$$

Convolution: 1-D example in MATLAB

```
u = [1 1 0 0 0 1 1];      % input signal
v = [1 1 1];              % filter kernel
```

```
w1 = conv(u,v)
```

% full convolution, `w1 = conv(v,u)` has the same result

```
w1 = 1 2 2 1 0 1 2 2 1
```

```
w2 = conv(u,v,'same')
```

% returns the central part of the convolution that is the same size as `u`

```
w2 = 2 2 1 0 1 2 2
```

```
w3 = conv(v,u,'same')
```

% returns the central part of the convolution that is the same size as `v`

```
w3 = 1 0 1
```

```
w4 = conv(u,v,'valid')
```

% returns those parts that are computed without the zero-padded

```
w4 = 2 1 0 1 2
```

Correlation vs. convolution: 2-D example

Origin f

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

w

1	2	3
4	5	6
7	8	9

(a)

Padded f

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

(b)

Initial position for w

1	2	3	0	0	0	0
4	5	6	0	0	0	0
7	8	9	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(c)

Correlation result

0	0	0	0	0
0	9	8	7	0
0	6	5	4	0
0	3	2	1	0
0	0	0	0	0

(d)

Full correlation result

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	9	8	7	0	0
0	0	6	5	4	0	0
0	0	3	2	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(e)

Rotated w

9	8	7	0	0	0	0
6	5	4	0	0	0	0
3	2	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(f)

Convolution result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

(g)

Full convolution result

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	2	3	0	0
0	0	4	5	6	0	0
0	0	7	8	9	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(h)

2-D discrete convolution: `conv2(f,K,'valid')`

$$
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0\\
0 & 0 & 1 & 1 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 1 & 1 & 0\\
0 & 0 & 0 & 1 & 1 & 0 & 0\\
0 & 0 & 1 & 1 & 0 & 0 & 0\\
0 & 1 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 0 & 1\\
0 & 1 & 0\\
1 & 0 & 1
\end{bmatrix}
=
\begin{bmatrix}
\boxed{1} & 4 & 3 & 4 & 1\\
1 & 2 & 4 & 3 & 3\\
1 & 2 & 3 & 4 & 1\\
1 & 3 & 3 & 1 & 1\\
3 & 3 & 1 & 1 & 0
\end{bmatrix}
$$

no padding, stride 1

In MATLAB: `conv2(f,K,'valid')`

Full convolution: `conv2(f,K)` $\Longrightarrow (7+3-1)\times(7+3-1)$ matrix!

Stride (步長)

$$\begin{bmatrix} 1 & 1 & 2 & 0 & 0 & 2 \\ 4 & 3 & 1 & 5 & 2 & 1 \\ 2 & 5 & 1 & 1 & 4 & 5 \\ 1 & 3 & 2 & 3 & 1 & 4 \\ 2 & 5 & 4 & 0 & 0 & 1 \\ 3 & 4 & 4 & 1 & 3 & 4 \end{bmatrix} \circledast \begin{bmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 21 & \\ & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 2 & 0 & 0 & 2 \\ 4 & 3 & 1 & 5 & 2 & 1 \\ 2 & 5 & 1 & 1 & 4 & 5 \\ 1 & 3 & 2 & 3 & 1 & 4 \\ 2 & 5 & 4 & 0 & 0 & 1 \\ 3 & 4 & 4 & 1 & 3 & 4 \end{bmatrix} \circledast \begin{bmatrix} 1 & 2 & 0 \\ 0 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 21 & 14 \\ & \end{bmatrix}$$

(correction: 34 34)

Convolution of a 6×6 matrix and a 3×3 filter kernel with stride 3, no padding $\Longrightarrow 2 \times 2$ matrix

Padding (填補)

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 2 & 0 & 0 & 2 & 0 \\
0 & 4 & 3 & 1 & 2 & 2 & 1 & 0 \\
0 & 2 & 0 & 1 & 1 & 4 & 0 & 0 \\
0 & 1 & 3 & 2 & 3 & 1 & 4 & 0 \\
0 & 2 & 0 & 4 & 0 & 0 & 1 & 0 \\
0 & 3 & 4 & 4 & 1 & 3 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\circledast
\begin{bmatrix}
1 & 0 & 2 \\
0 & 2 & 0 \\
1 & 1 & 0
\end{bmatrix}
=
\begin{bmatrix}
5 & 11 & 12 & 4 & 5 & 8 \\
10 & 14 & 5 & 10 & 8 & 12 \\
14 & 8 & 14 & 11 & 21 & 3 \\
4 & 15 & 6 & 19 & 7 & 8 \\
12 & 15 & 22 & 15 & 11 & 12 \\
8 & 12 & 12 & 2 & 7 & 9
\end{bmatrix}
$$

Convolution of a 6×6 *matrix with zero-padding* 1
and a 3×3 *filter kernel with stride* 1

In MATLAB: `conv2(f,K,'same')`

A Matlab file for convolution and correlation

```
clear all
clc
m=5;%image size
w=3;%window size of convolution
I=reshape(1:m^2,m,m)
K=reshape(1:w^2,w,w)
%convolution
conv2_output=conv2(I,K,'valid')
%manual implementation
C=zeros(m-w+1,m-w+1);
for i=1:m-w+1
  for j=1:m-w+1
    C(i,j)=sum(sum(I(i:i+w-1,j:j+w-1).*rot90(K,2)));
  end
end
C
```

A Matlab file for convolution and correlation (cont'd)

```
%correlation
corr_output=filter2(K,I,'valid')
% manual implementation
D=zeros(m-w+1,m-w+1);
for i=1:m-w+1
  for j=1:m-w+1
    D(i,j)=sum(sum(I(i:i+w-1,j:j+w-1).*K));
  end
end
D
```

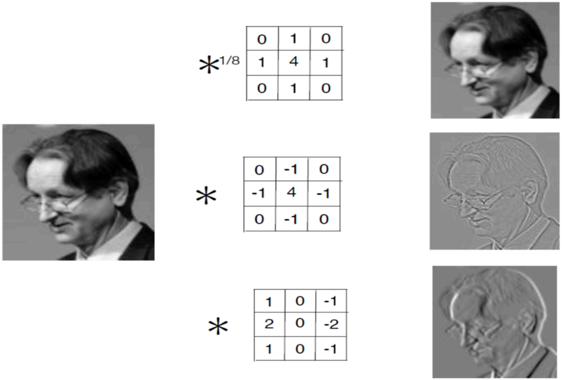
```
% function 'imfilter' is provided in the MATLAB toolbox
imfilter_conv_output=imfilter(I,K,'conv','same')
imfilter_corr_output=imfilter(I,K,'corr','same')
```

Results of the Matlab file

```
I =

     1     6    11    16    21
     2     7    12    17    22
     3     8    13    18    23
     4     9    14    19    24
     5    10    15    20    25


K =

     1     4     7
     2     5     8
     3     6     9


conv2_output =

   219   444   669
   264   489   714
   309   534   759


C =

   219   444   669
   264   489   714
   309   534   759
```

```
corr_output =

   411   636   861
   456   681   906
   501   726   951


D =

   411   636   861
   456   681   906
   501   726   951


imfilter_conv_output =

    32   114   249   384   440
    68   219   444   669   734
    89   264   489   714   773
   110   309   534   759   812
    96   252   417   582   600


imfilter_corr_output =

   128   276   441   606   320
   202   411   636   861   436
   241   456   681   906   457
   280   501   726   951   478
   184   318   453   588   280
```

Convolution operation = spatial filtering

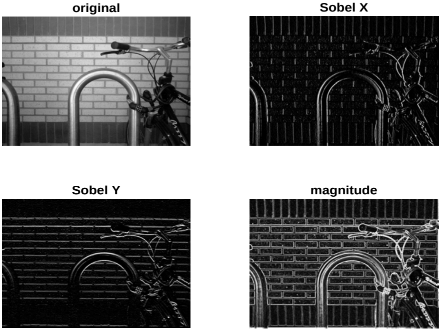


Different kernels reveal a different characteristics of the input image

Example of edge detection: Sobel operator

The Sobel operator is used for edge detection, which creates an image that emphasizes edges. Below are two kernels used in the operation:

$$\text{Sobel X} = f * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \text{Sobel Y} = f * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



$$magnitude(i,j) := \|(SobelX(i,j), SobelY(i,j))\|_2$$