Computer Vision:
Image filtering

Siheng Chen 陈思衡

Image filtering

Intro to image filtering

More maths about filtering

Image filter design

Why we talk about image filtering

image filtering

feature extractor

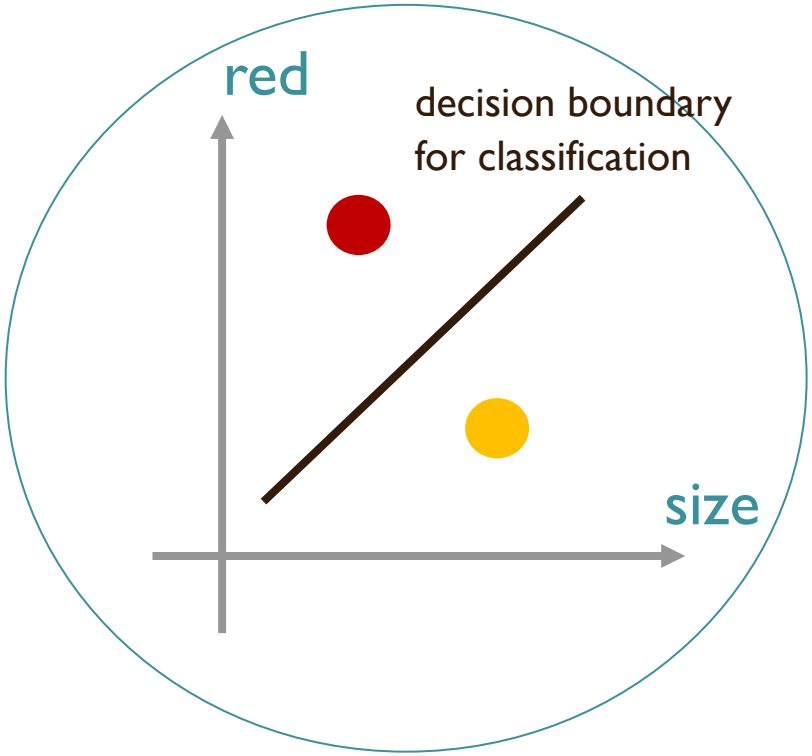


image space

feature space

vector space

Why we talk about image filtering?

Image editing

Why we talk about image filtering?

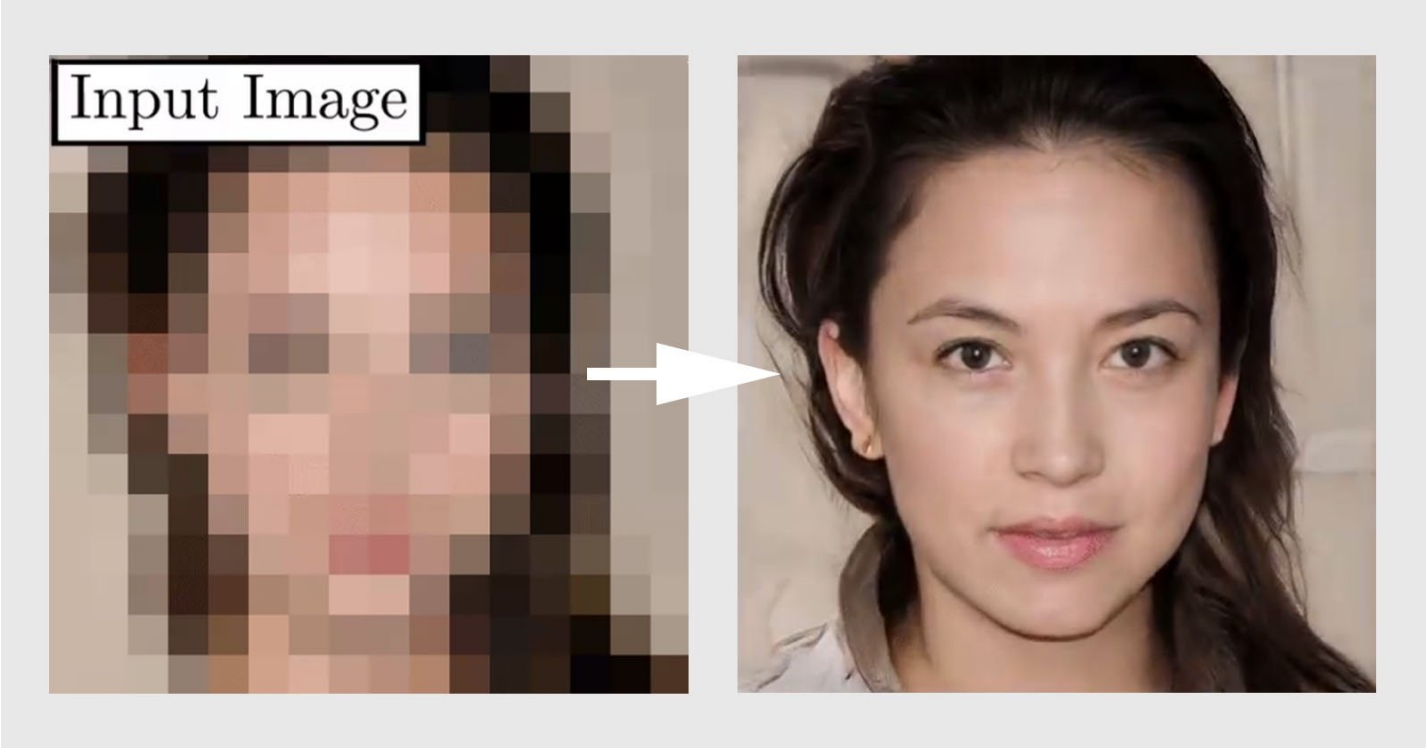
Image denoising

Why we talk about image filtering?

Image inpainting

Why we talk about image filtering?

Image super-resolution



What is an image?

A (color) image is a 3D tensor of numbers.

What is an image?

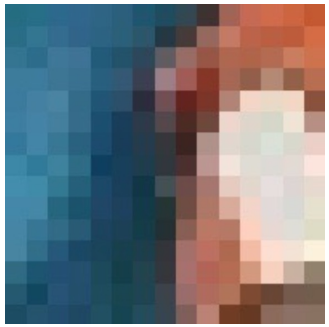
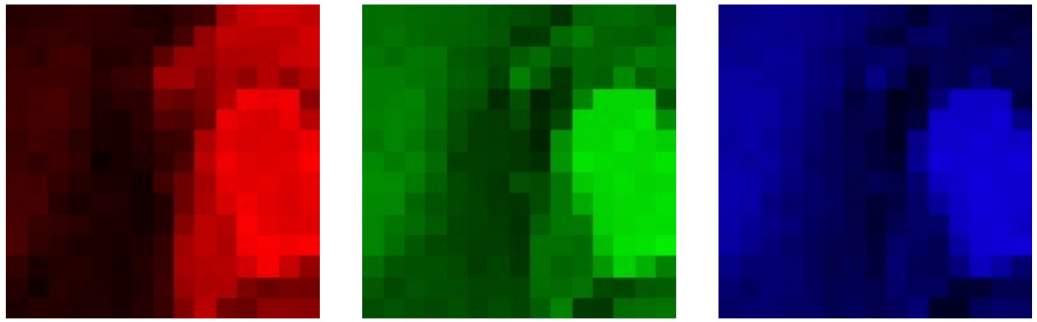
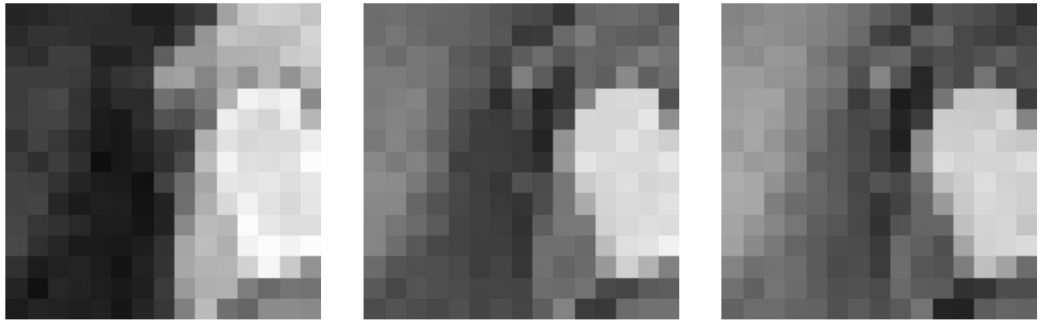


image patch

red green blue





Intensity value at each channel

Each channel is a 2D array of numbers

What is an image?

A (grayscale) image is a 2D function.

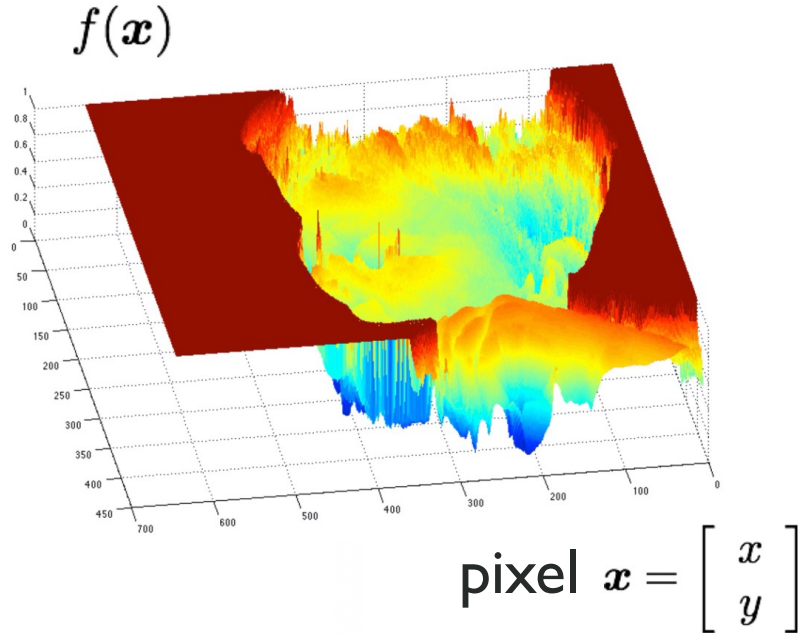
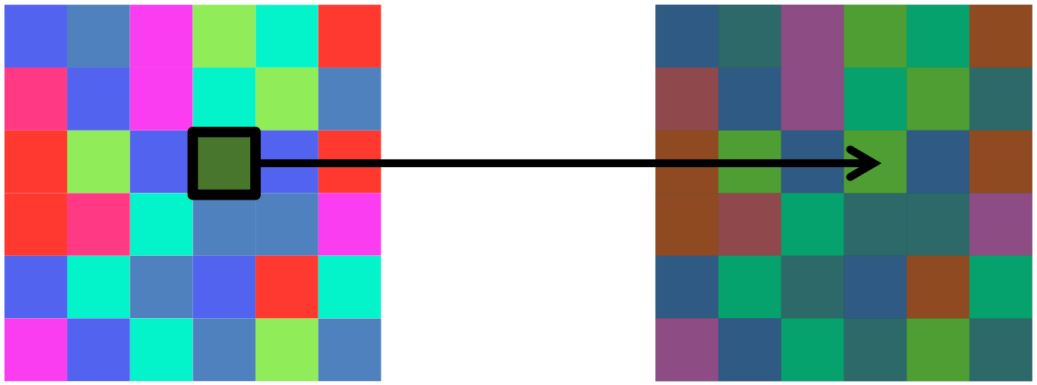


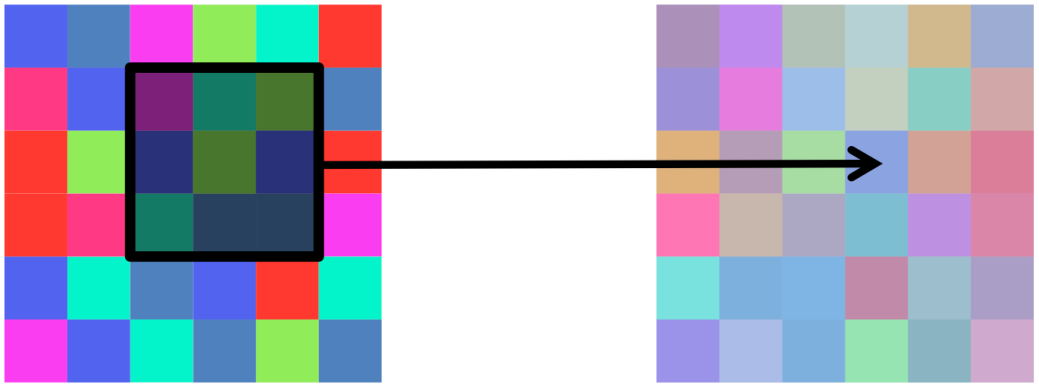
Image filtering is an operation to modify the 2D function

Image filtering

Point-wise operation



Neighbor-wise operation



Point-wise operation

original

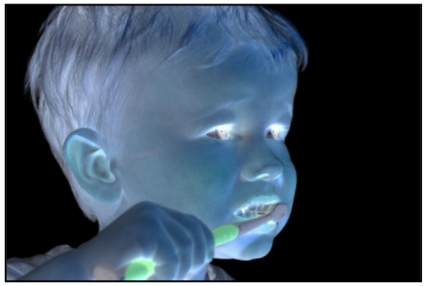
x

darken

lower contrast

non-linear lower contrast

invert



lighten

raise contrast

non-linear raise contrast

Neighbor-wise operation

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

output

	0								

Neighbor-wise operation

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

output

	0	10							

Neighbor-wise operation

input

filter

0	0	0
0	1	0
0	0	0

output

unchanged

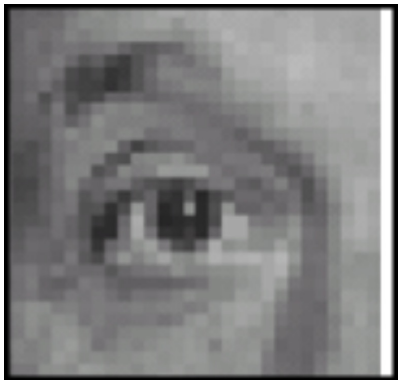
Neighbor-wise operation

input

filter

0	0	0
0	0	1
0	0	0

output



shift to left by one

Neighbor-wise operation

input

filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

output

sharpening

Image filtering

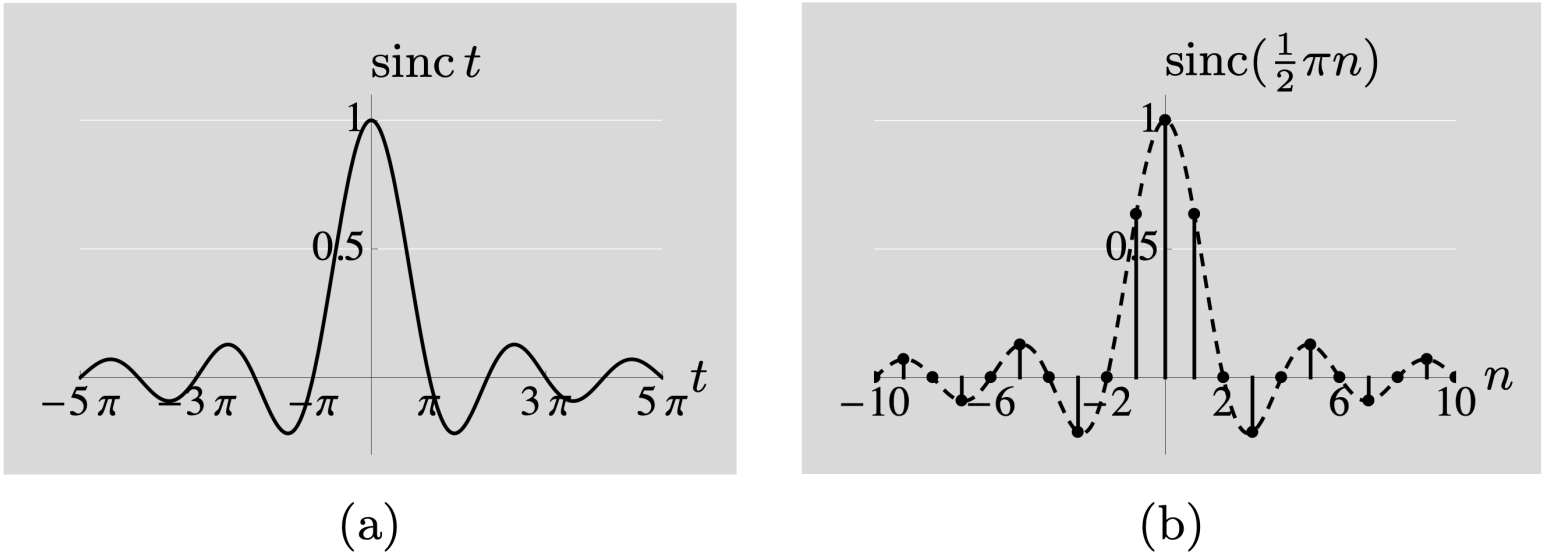
Intro to filtering

More maths about filtering

Image filter design

Filtering

Sequence



(a) The sinc function sinc t. (b) The sinc sequence sinc$(\frac{1}{2}\pi n)$.

Filtering

Sequence

- two-sided infinite

$$x = \begin{bmatrix} \dots & x_{-2} & x_{-1} & \boxed{x_0} & x_1 & x_2 & \dots \end{bmatrix}^\top$$

- a sequence of length N

$$x = \begin{bmatrix} \boxed{x_0} & x_1 & x_2 & \dots & x_{N-1} \end{bmatrix}^\top$$

Filtering

Sequence

- Autocorrelation

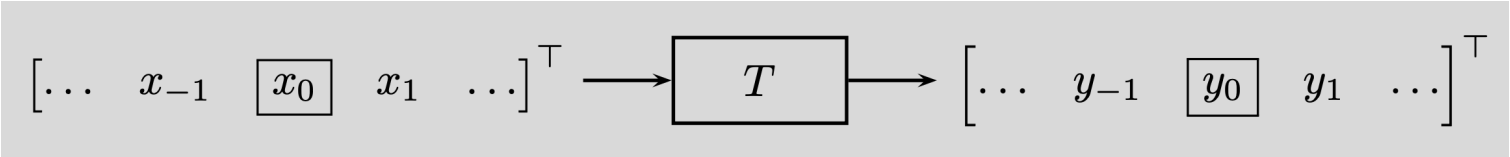
$$a_n = \sum_{k \in \mathbb{Z}} x_k x_{k-n}^* = \langle x_k, x_{k-n} \rangle_k$$

- Cross-correlation

$$c_n = \sum_{k \in \mathbb{Z}} x_k y_{k-n}^* = \langle x_k, y_{k-n} \rangle_k$$

Filtering

Systems



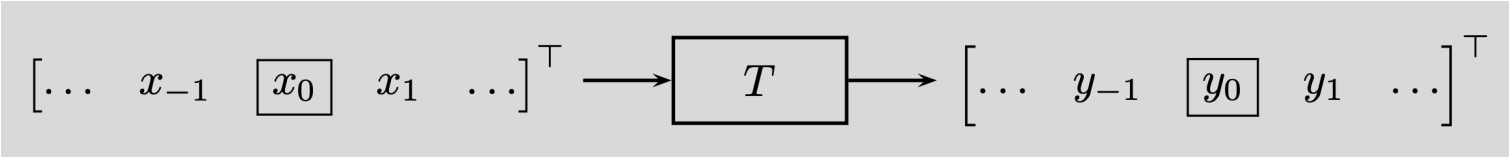
A discrete-time system

- Example
  - The shift-by-1 operator, or delay $y_n = x_{n-1}, \qquad n \in \mathbb{Z}$

$$y = \begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ x_{-2} \\ \boxed{x_{-1}} \\ x_0 \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & \boxed{0} & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ \vdots \end{bmatrix}$$

Filtering

Systems



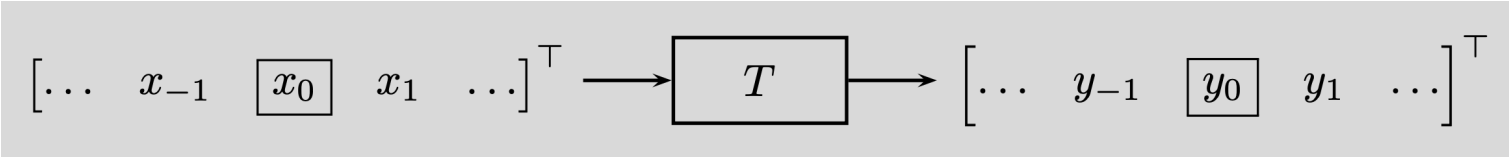
A discrete-time system

- Example
  - The advance-by-1 operator $y_n = x_{n+1}, \quad n \in \mathbb{Z}$

$$y = \begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ x_0 \\ \boxed{x_1} \\ x_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & \boxed{0} & 1 & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ \vdots \end{bmatrix}$$

Filtering

Systems



A discrete-time system

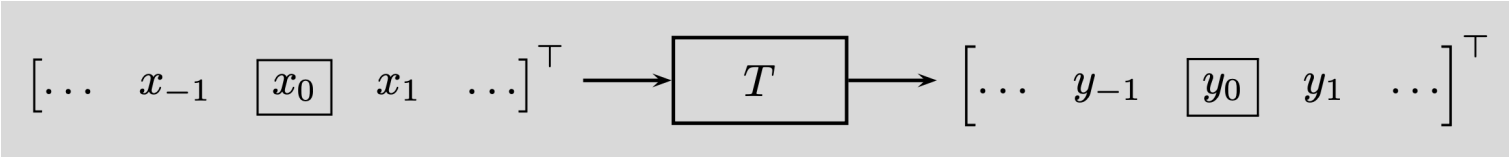
- Example
  - Modulator

$$y_n = \alpha_n x_n, \qquad n \in \mathbb{Z}$$

$$\begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \alpha_{-1}x_{-1} \\ \boxed{\alpha_0 x_0} \\ \alpha_1 x_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & \alpha_{-1} & 0 & 0 & \cdots \\ \cdots & 0 & \boxed{\alpha_0} & 0 & \cdots \\ \cdots & 0 & 0 & \alpha_1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ \vdots \end{bmatrix}$$

Filtering

Systems



A discrete-time system

- Example
  - Averaging operators

$$y_n = \frac{1}{3}(x_{n-1} + x_n + x_{n+1}), \qquad n \in \mathbb{Z}$$

$$\begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \frac{1}{3} \begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & 1 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & \boxed{1} & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

Filtering

Systems

DEFINITION 3.1 (LINEAR SYSTEM) A discrete-time system T is called *linear* when, for any inputs x and y and any $\alpha, \beta \in \mathbb{C}$,

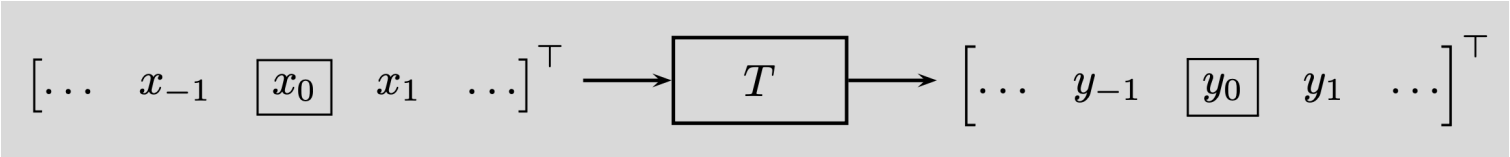
$$T(\alpha x + \beta y) \;=\; \alpha T(x) + \beta T(y). \tag{3.32}$$

DEFINITION 3.4 (SHIFT-INVARIANT SYSTEM) A discrete-time system T is called *shift-invariant* when, for any integer k and input x,

$$y \;=\; T(x) \quad \Rightarrow \quad y' \;=\; T(x'), \qquad \text{where } x'_n = x_{n-k} \text{ and } y'_n = y_{n-k}. \tag{3.37}$$

Filtering

Systems



A discrete-time system

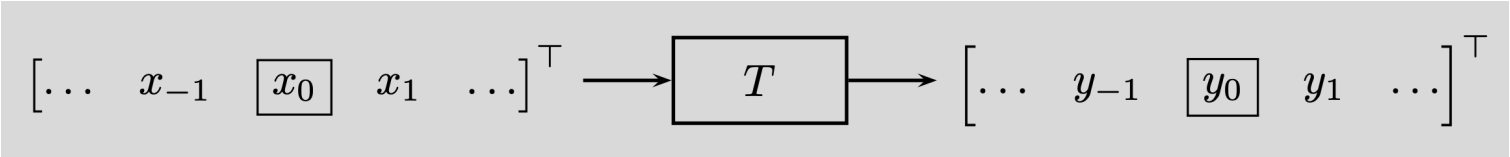
- Example
  - The shift-by-1 operator, or delay $\quad y_n = x_{n-1}, \qquad n \in \mathbb{Z}$

$$y = \begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ x_{-2} \\ \boxed{x_{-1}} \\ x_0 \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & \boxed{0} & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ \vdots \end{bmatrix}$$

LSI !

Filtering

Systems



A discrete-time system

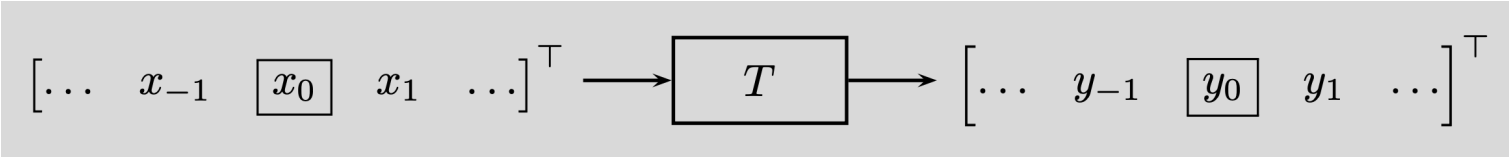
- Example
  - The advance-by-1 operator $y_n = x_{n+1}, \quad n \in \mathbb{Z}$

$$y = \begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ x_0 \\ \boxed{x_1} \\ x_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & \boxed{0} & 1 & \cdots \\ \cdots & 0 & 0 & 0 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ \vdots \end{bmatrix}$$

LSI !

Filtering

Systems



A discrete-time system

- Example
  - Modulator

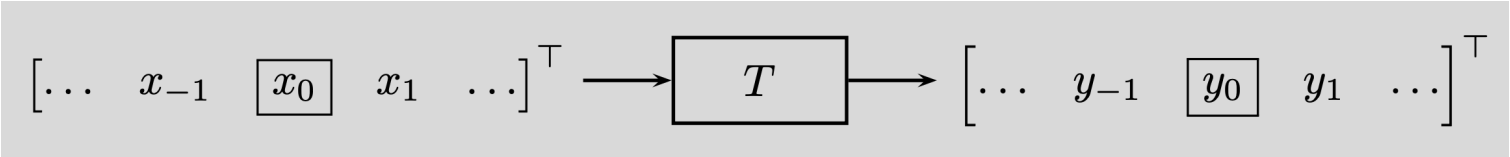
$$y_n = \alpha_n x_n, \qquad n \in \mathbb{Z}$$

$$\begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \alpha_{-1}x_{-1} \\ \boxed{\alpha_0 x_0} \\ \alpha_1 x_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & \alpha_{-1} & 0 & 0 & \cdots \\ \cdots & 0 & \boxed{\alpha_0} & 0 & \cdots \\ \cdots & 0 & 0 & \alpha_1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ \vdots \end{bmatrix}$$

No LSI !

Filtering

Systems



A discrete-time system

- Example
  - Averaging operators

$$y_n = \frac{1}{3}(x_{n-1} + x_n + x_{n+1}), \qquad n \in \mathbb{Z}$$

$$\begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \frac{1}{3} \begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & 1 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & \boxed{1} & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

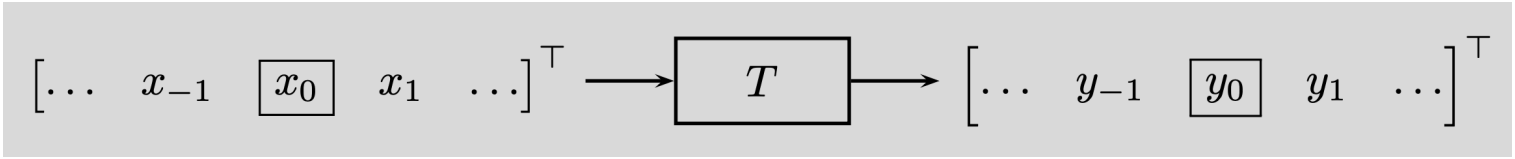
LSI !

Filtering

Linear shift-invariant systems

DEFINITION 3.6 (IMPULSE RESPONSE) A sequence h is called the *impulse response* of LSI discrete-time system T when input δ produces output h.

$$x = \delta$$



Filtering

Linear shift-invariant systems

DEFINITION 3.7 (CONVOLUTION) The *convolution* between sequences h and x is defined as

$$(Hx)_n = (h * x)_n = \sum_{k\in\mathbb{Z}} x_k h_{n-k} = \sum_{k\in\mathbb{Z}} x_{n-k} h_k, \tag{3.61}$$

where H is called the *convolution operator* associated with h.

$$y = Tx = T\sum_{k\in\mathbb{Z}} x_k \delta_{n-k} \overset{(a)}{=} \sum_{k\in\mathbb{Z}} x_k T\delta_{n-k} \overset{(b)}{=} \sum_{k\in\mathbb{Z}} x_k h_{n-k} = h * x$$

linearity — shift invariance — definition

Filtering

Linear shift-invariant systems

DEFINITION 3.7 (CONVOLUTION) The *convolution* between sequences h and x is defined as

$$(Hx)_n = (h * x)_n = \sum_{k\in\mathbb{Z}} x_k h_{n-k} = \sum_{k\in\mathbb{Z}} x_{n-k} h_k, \tag{3.61}$$

where H is called the *convolution operator* associated with h.

$$y = Tx = T\sum_{k\in\mathbb{Z}} x_k \delta_{n-k} \overset{(a)}{=} \sum_{k\in\mathbb{Z}} x_k T\delta_{n-k} \overset{(b)}{=} \sum_{k\in\mathbb{Z}} x_k h_{n-k} = h * x$$

filter

filtering

The impulse response of a system is called a **filter**

Convolution with the impulse response is called **filtering**

Convolution is a mathematical way to implement a LSI system

Filtering

Linear shift-invariant systems

- Convolution

$$y = \begin{bmatrix} \vdots \\ y_{-2} \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & h_0 & h_{-1} & h_{-2} & h_{-3} & h_{-4} & \cdots \\ \cdots & h_1 & h_0 & h_{-1} & h_{-2} & h_{-3} & \cdots \\ \cdots & h_2 & h_1 & \boxed{h_0} & h_{-1} & h_{-2} & \cdots \\ \cdots & h_3 & h_2 & h_1 & h_0 & h_{-1} & \cdots \\ \cdots & h_4 & h_3 & h_2 & h_1 & h_0 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}}_{H} \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} = Hx$$

Filtering

Linear shift-invariant systems

- Properties of convolution

(i) *Connection to the inner product:*

$$(h * x)_n \;=\; \sum_{k\in\mathbb{Z}} x_k h_{n-k} \;=\; \langle x_k,\, h^*_{n-k}\rangle_k.$$

(ii) *Commutativity:*

$$h * x \;=\; x * h.$$

(iii) *Associativity:*

$$g * (h * x) \;=\; g * h * x \;=\; (g * h) * x.$$

(iv) *Deterministic autocorrelation:*

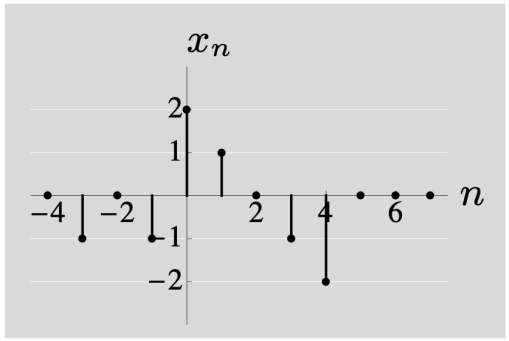
$$a_n \;=\; \sum_{k\in\mathbb{Z}} x_k x^*_{k-n} \;=\; x_n *_n x^*_{-n}.$$

(v) *Shifting:* For any $k \in \mathbb{Z}$,

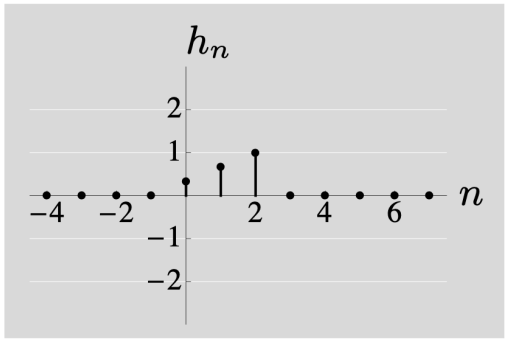
$$x_n *_n \delta_{n-k} \;=\; x_{n-k}.$$

Filtering

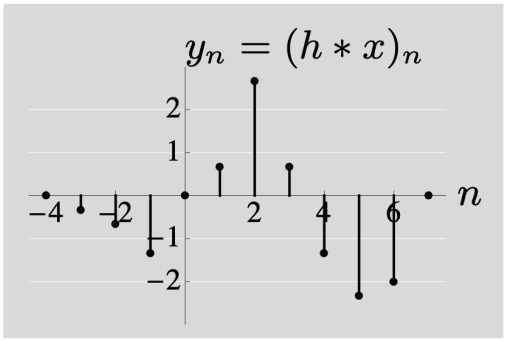
Linear shift-invariant systems



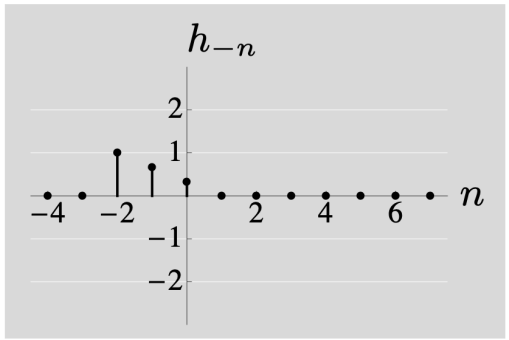
(a) Sequence.



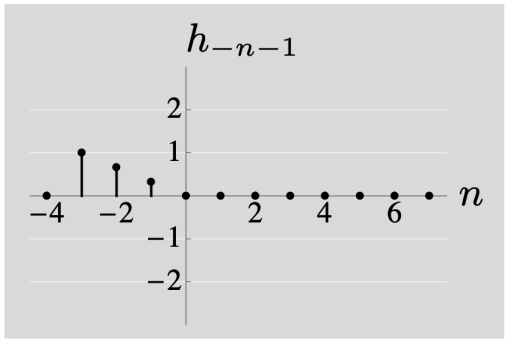
(b) Impulse response.



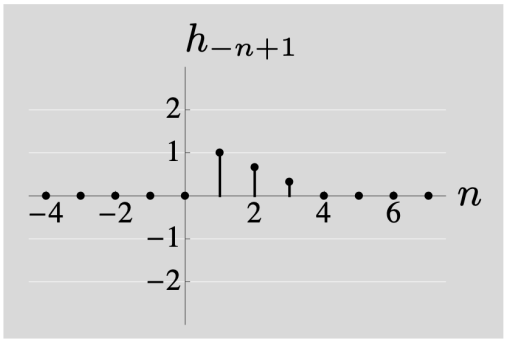
(c) Convolution.



(d)



(e)



(f)

Image filtering

Intro to filtering

More maths about filtering

Image filter design

Image filter design

DEFINITION 3.6 (IMPULSE RESPONSE) A sequence h is called the *impulse response* of LSI discrete-time system T when input δ produces output h.

$$x = \delta$$

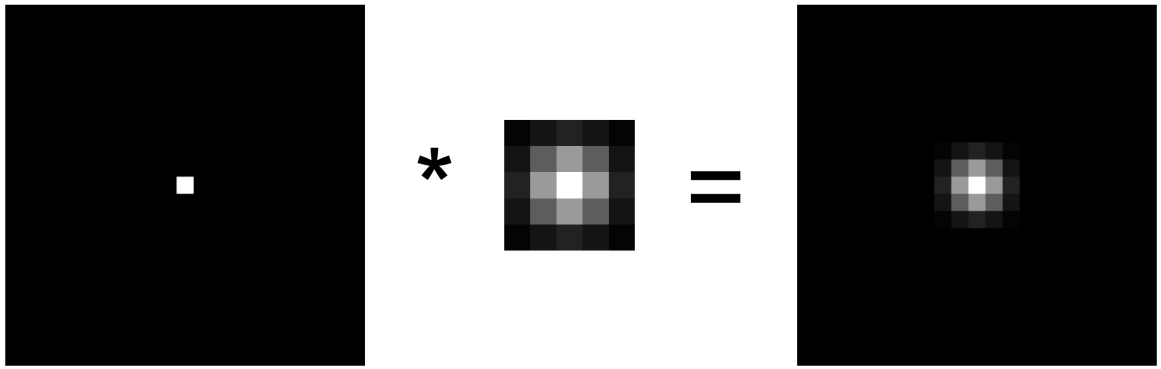
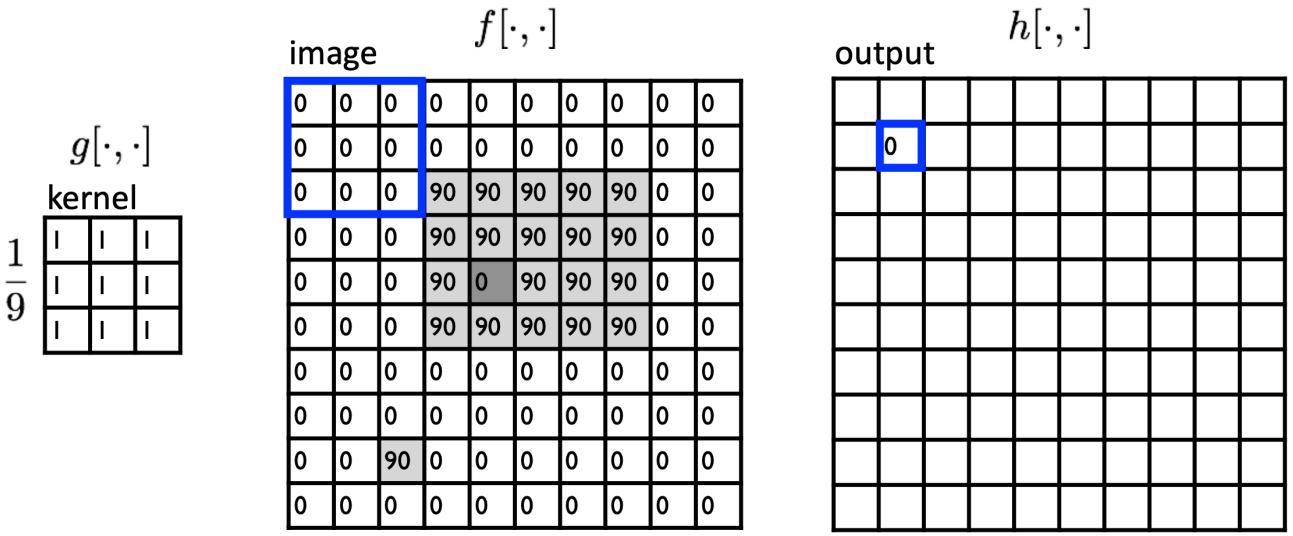


Image filter design

DEFINITION 3.7 (CONVOLUTION) The *convolution* between sequences h and x is defined as

$$(Hx)_n = (h * x)_n = \sum_{k\in\mathbb{Z}} x_k h_{n-k} = \sum_{k\in\mathbb{Z}} x_{n-k} h_k, \tag{3.61}$$

where H is called the *convolution operator* associated with h.

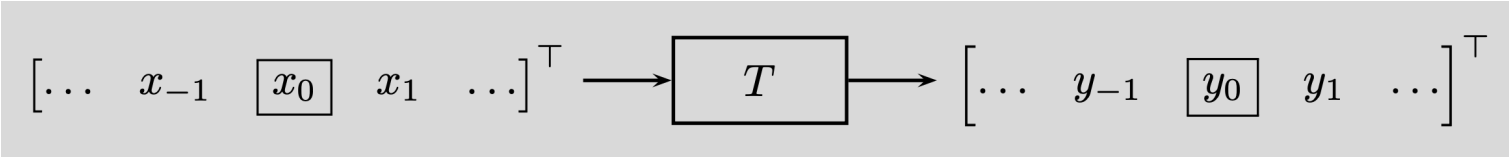


$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

output filter image (signal)

Filtering

Systems



A discrete-time system

- Example
  - Averaging operators

$$y_n = \frac{1}{3}(x_{n-1} + x_n + x_{n+1}), \qquad n \in \mathbb{Z}$$

$$\begin{bmatrix} \vdots \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ \vdots \end{bmatrix} = \frac{1}{3} \begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & 1 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & \boxed{1} & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

Image filter design

DEFINITION 3.7 (CONVOLUTION) The *convolution* between sequences h and x is defined as

$$(Hx)_n = (h * x)_n = \sum_{k\in\mathbb{Z}} x_k h_{n-k} = \sum_{k\in\mathbb{Z}} x_{n-k} h_k, \qquad (3.61)$$

where H is called the *convolution operator* associated with h.

image filter:
averaging operator

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$ × | 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$f[\cdot,\cdot]$

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

output

0									

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

output — filter — image (signal)

Image filter design

DEFINITION 3.7 (CONVOLUTION) The *convolution* between sequences h and x is defined as

$$(Hx)_n = (h * x)_n = \sum_{k\in\mathbb{Z}} x_k h_{n-k} = \sum_{k\in\mathbb{Z}} x_{n-k} h_k, \qquad (3.61)$$

where H is called the *convolution operator* associated with h.

image filter:
averaging operator

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

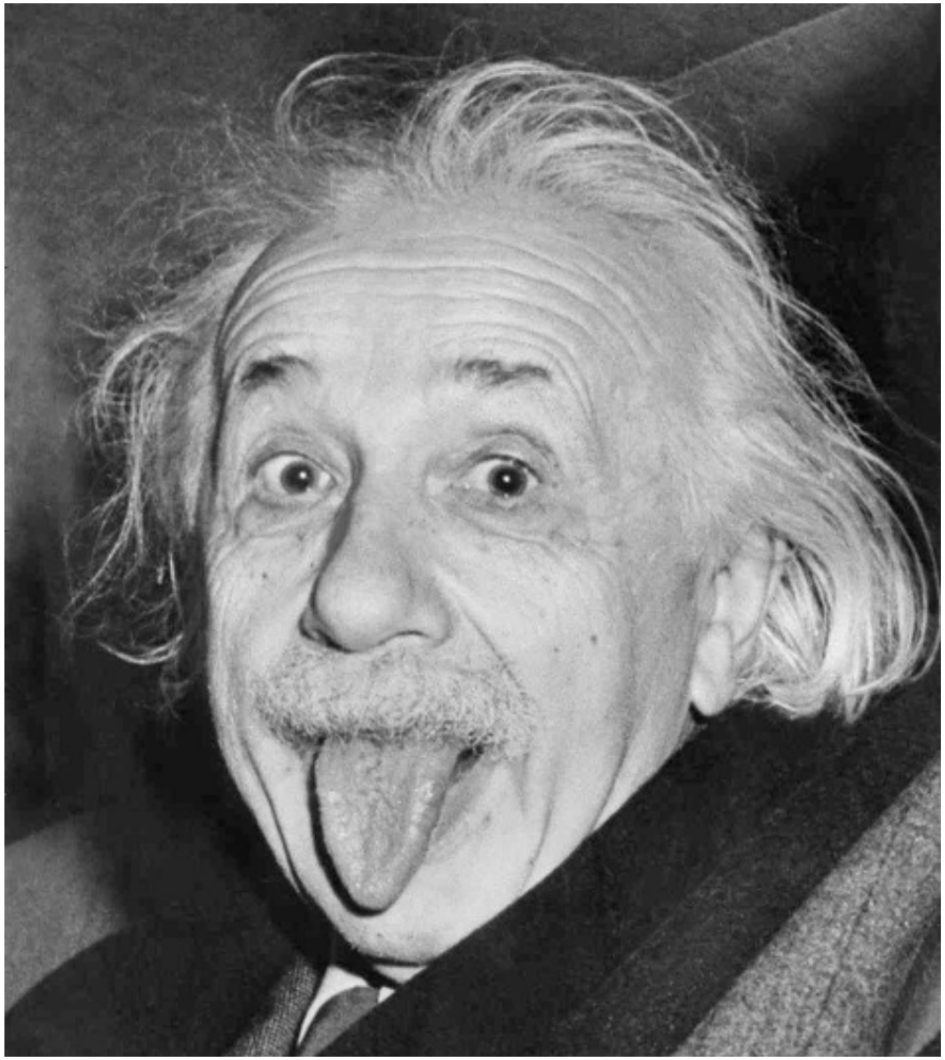
output

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

output — filter — image (signal)

Image filter design



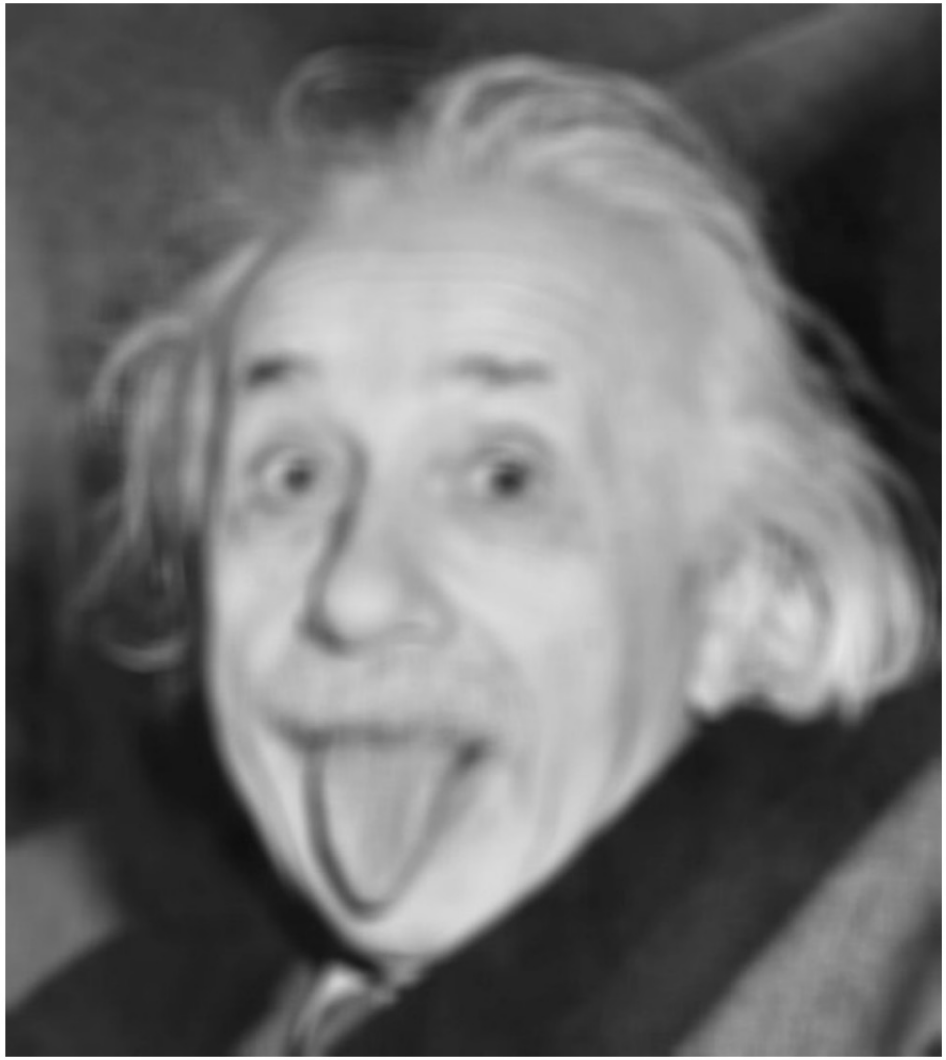


Image filter design

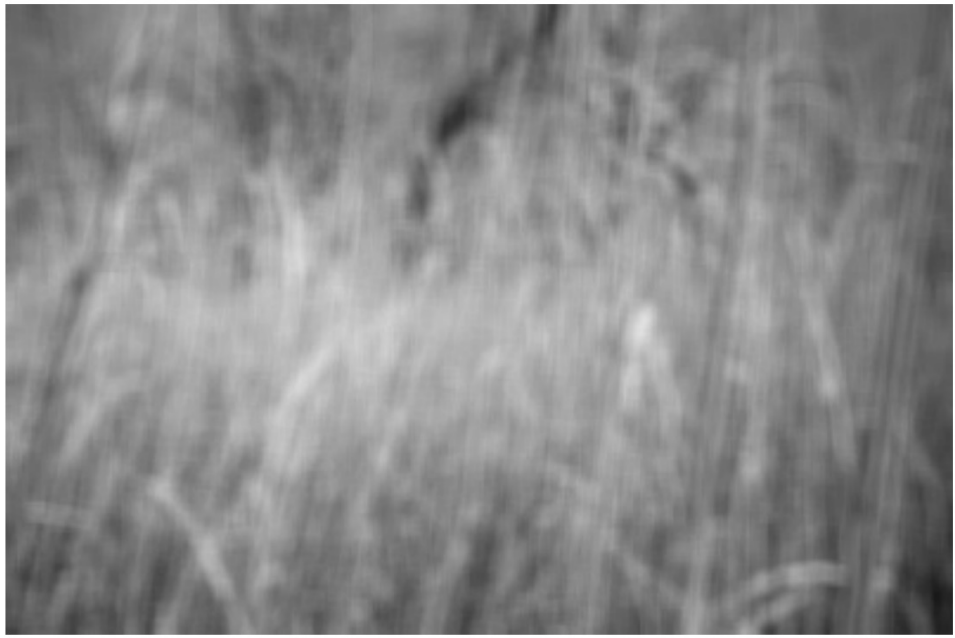
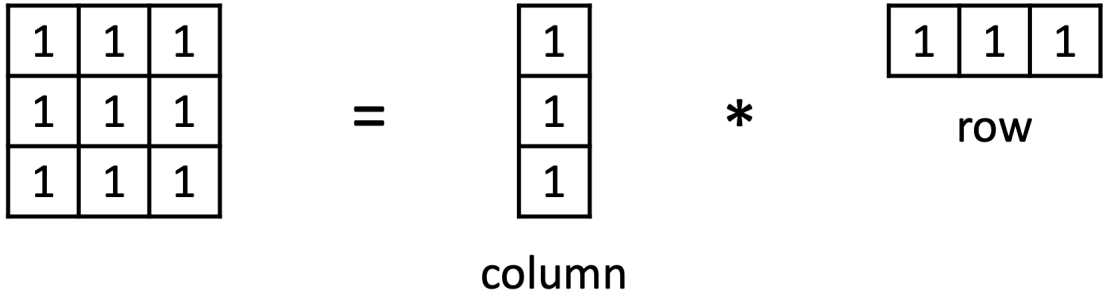


Image filter design

Image filter design

A 2D filter is separable if it can be written as the product of a "column" and a "row".



2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:

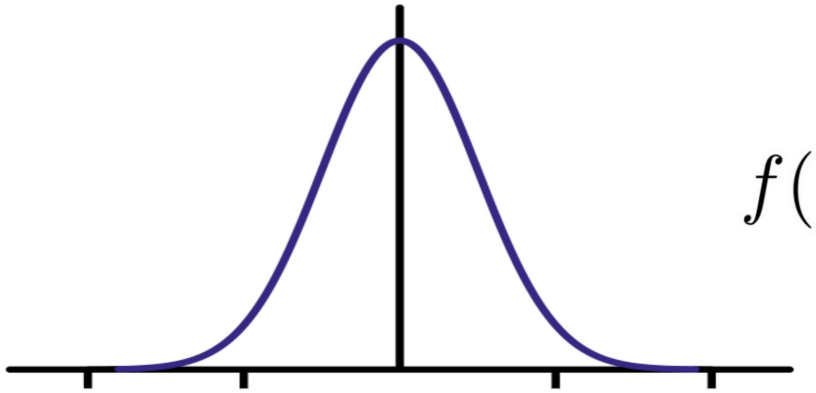
What is the cost of convolution with a non-separable filter? $M^2 \times N^2$

What is the cost of convolution with a separable filter? $2 \times N \times M^2$

Image filter design

Gaussian filter

- kernel values sampled from the 2D Gaussian function



$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

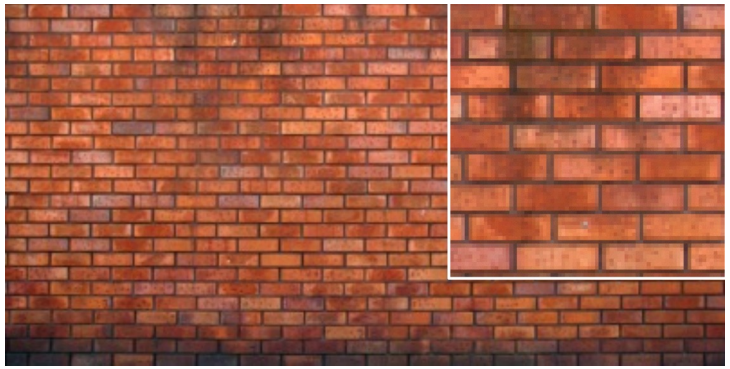
weight falls off with distance from center pixel

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

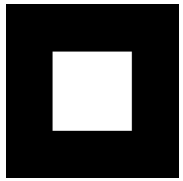
Is this a separable filter? YES !

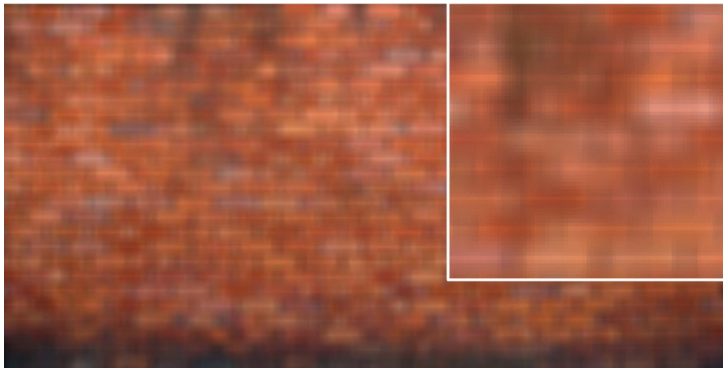
Image filter design

Gaussian filter

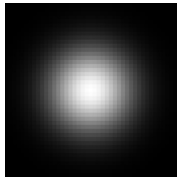


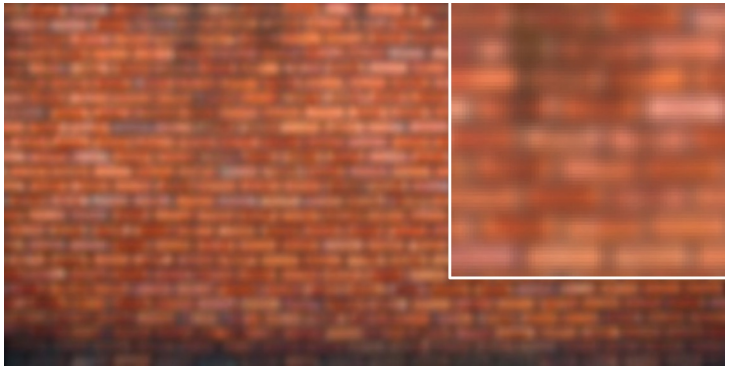
averaging operator





filtering





Gaussian filter

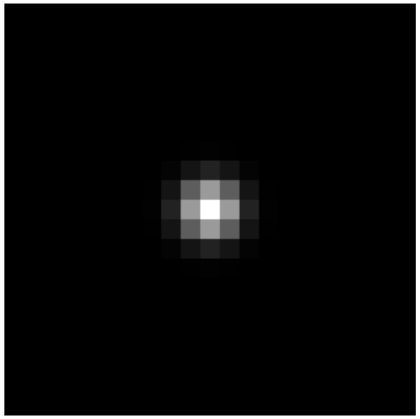
Smoothing!

Image filter design

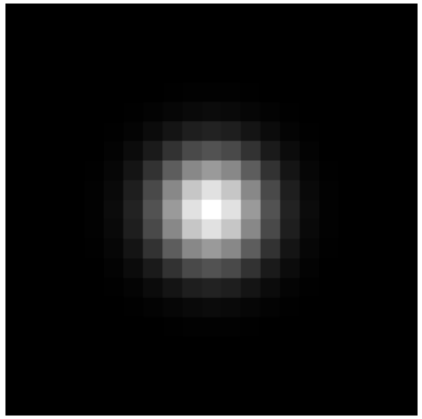
Gaussian filter

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

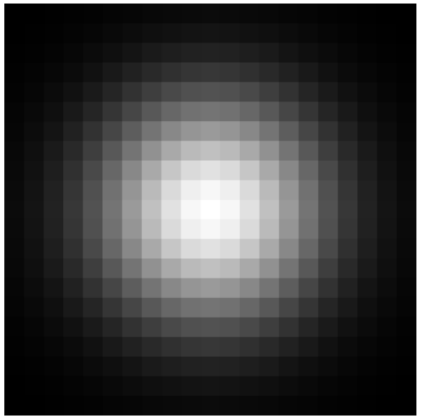
σ = 1
filter = 21x21



σ = 2
filter = 21x21



σ = 4
filter = 21x21



σ = 8
filter = 21x21

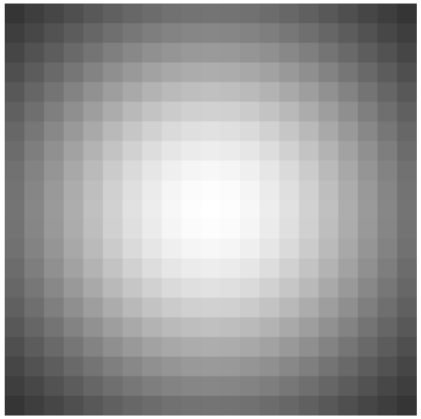
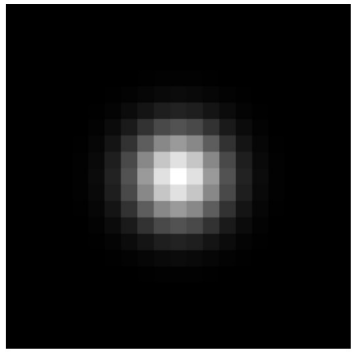


Image filter design

Gaussian filter

σ = 2



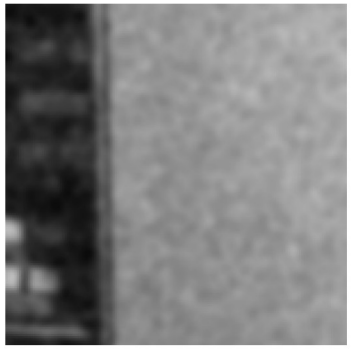


Image filter design

Gaussian filter

$\sigma = 4$

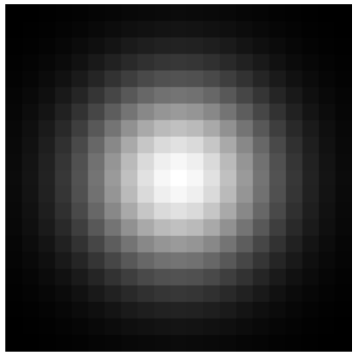


Image filter design

Gaussian filter

σ = 8

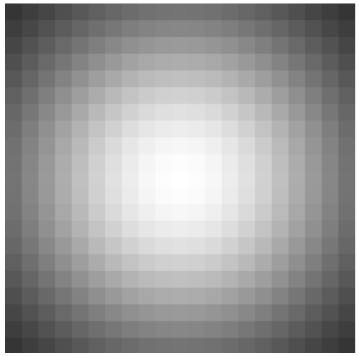


Image filter design

Gaussian filter

$\sigma = 8$

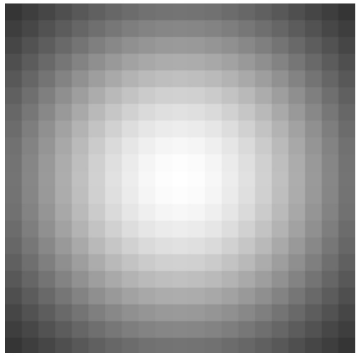
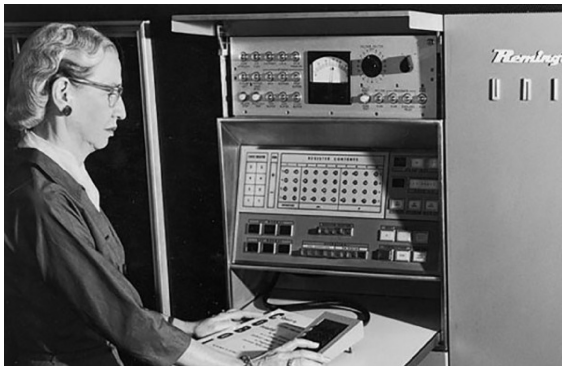


Image filter design

Sharpening (high-pass)

image



−

smoothed

=

details

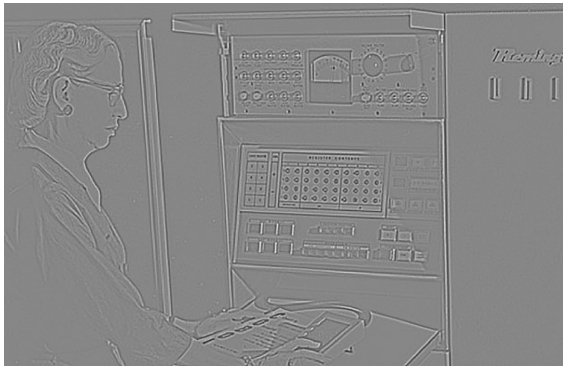
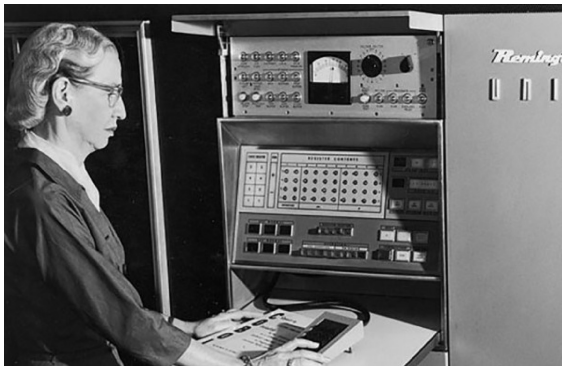


Image filter design

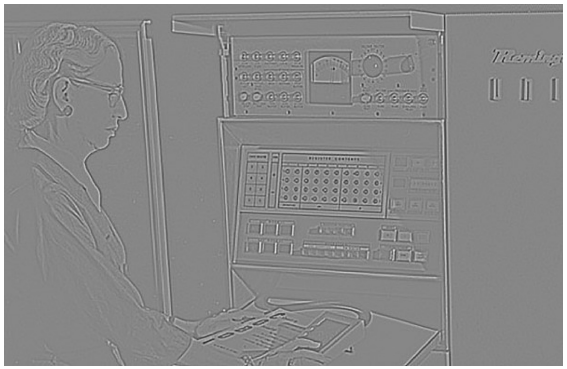
Sharpening (high-pass)

image



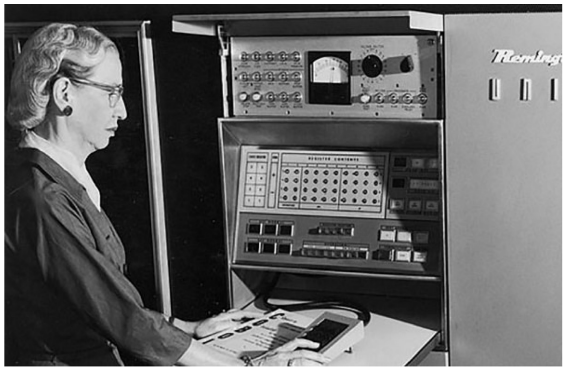
$+\alpha$

details



$=$

sharpened

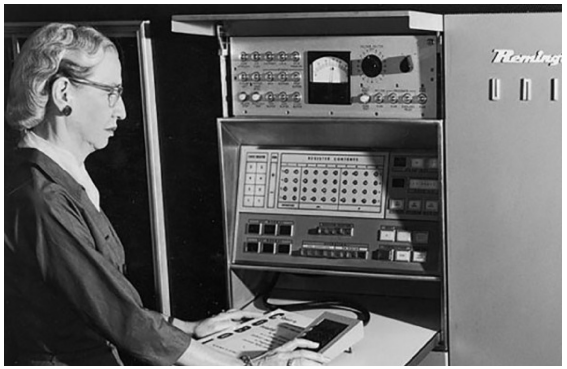


$\alpha = 1$

Image filter design

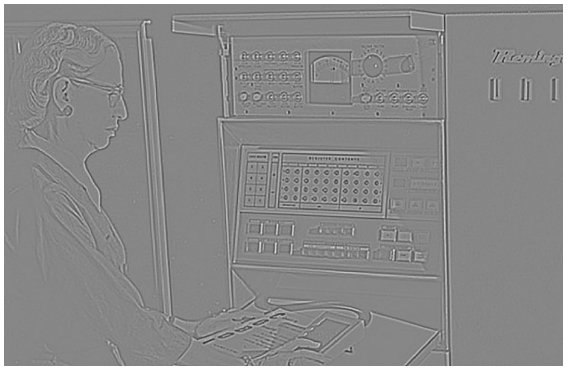
Sharpening (high-pass)

image



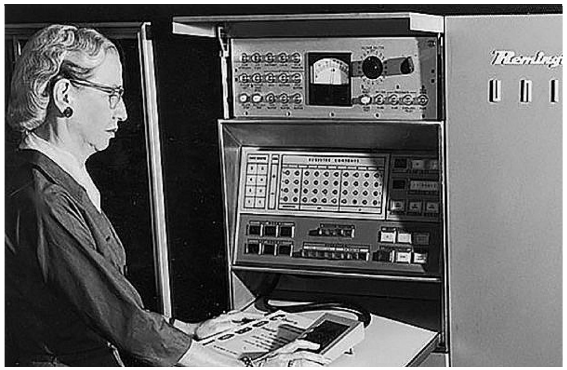
+α

details



=

sharpened



$\alpha = 10$

Image filter design

Sharpening (high-pass)

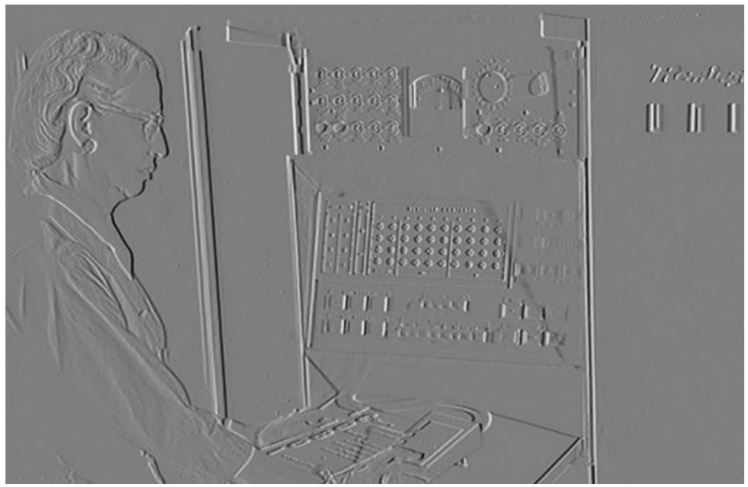
-1	0	1

Dx

-1	0	1

^{T}

Dy



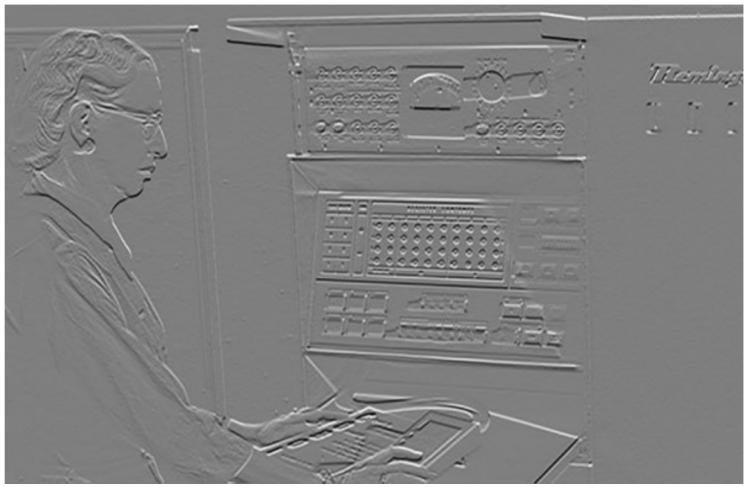


Image filter design

Gabor filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$.

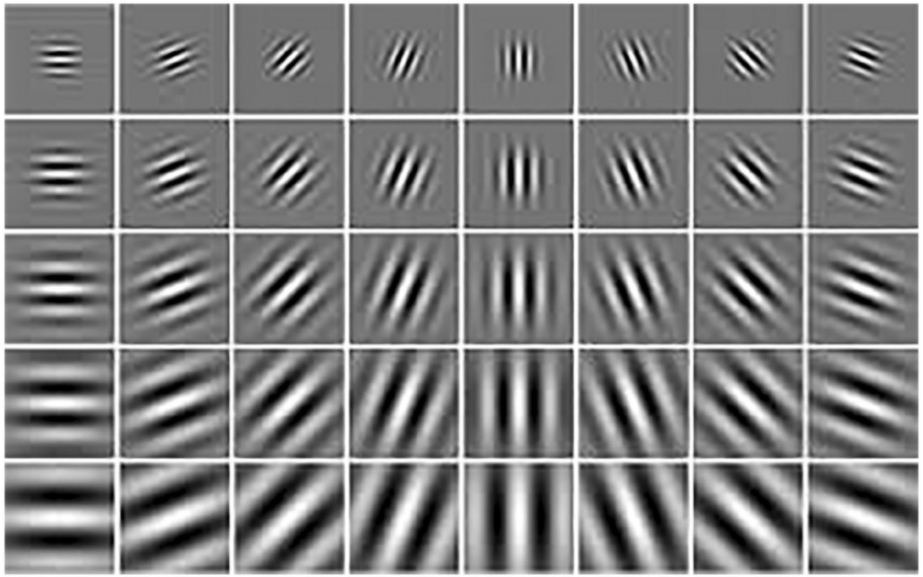


Image filter design

Convolutional neural network

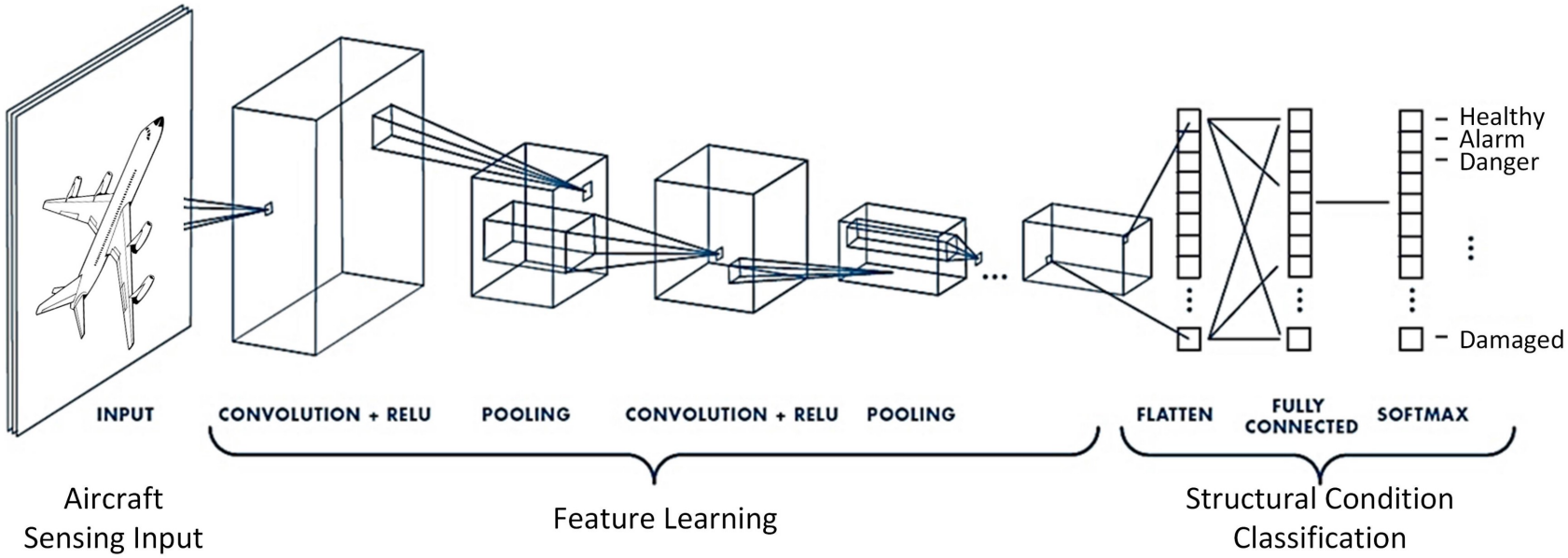
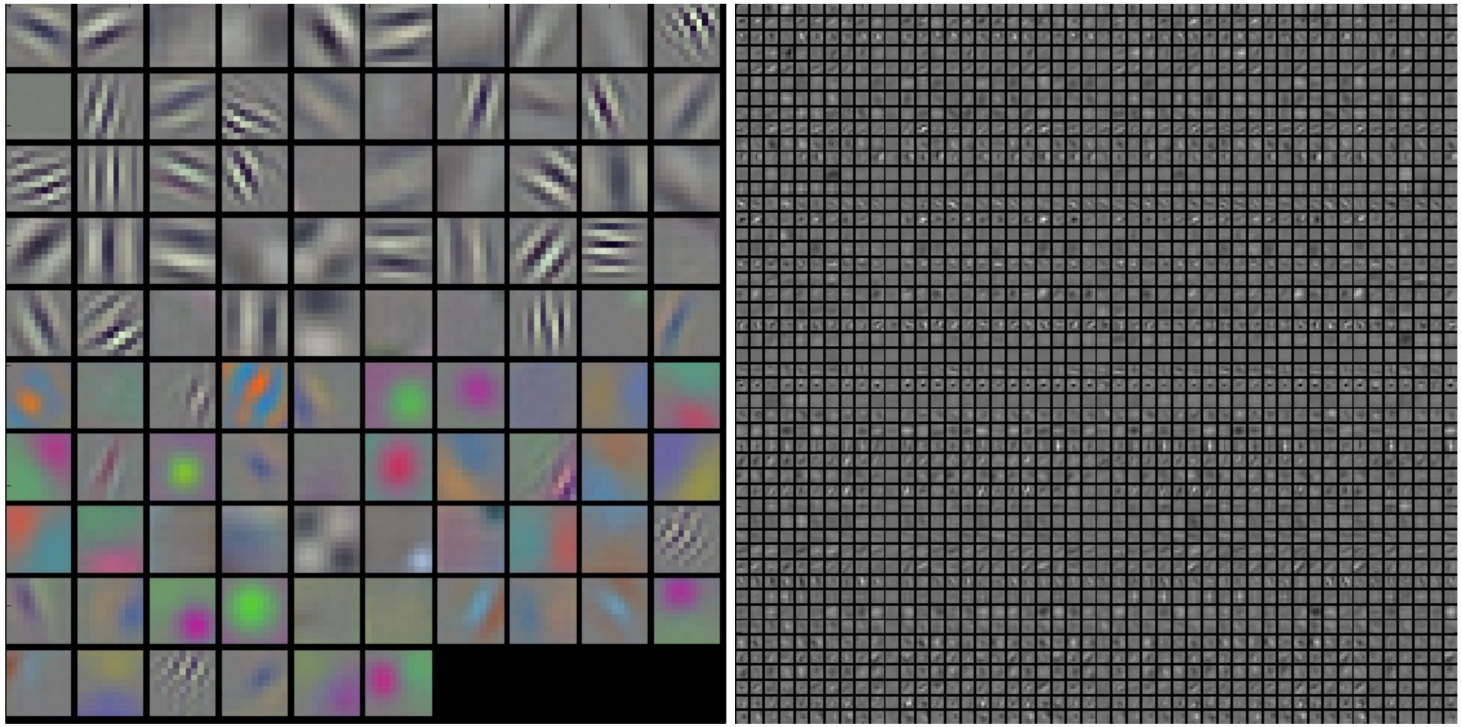


Image filter design

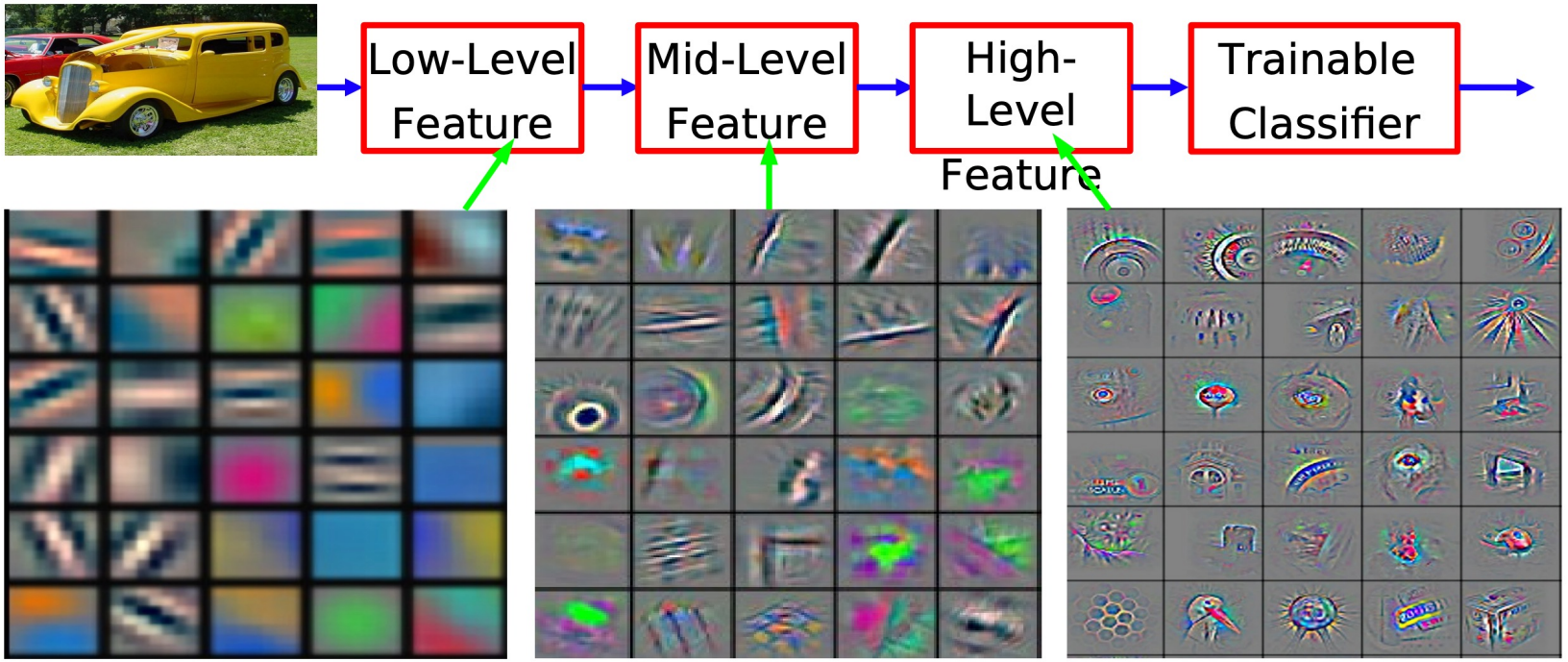
Convolutional neural network



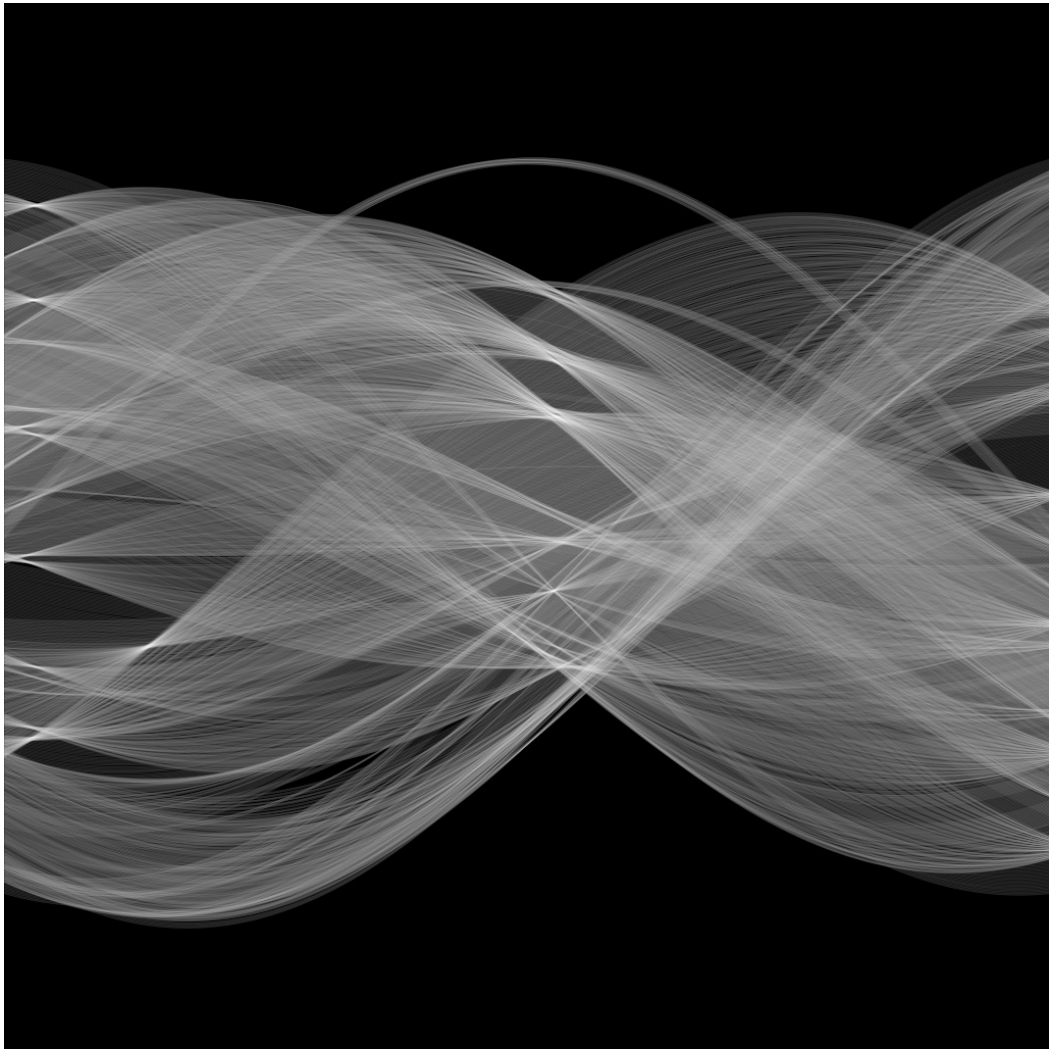
Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet.

Image filter design

Convolutional neural network



Next lecture: Line detection



https://jayrambhia.com/blog/hough-transform

Thank you very much!

sihengc@sjtu.edu.cn