Microscopy Network Basel

Image processing course

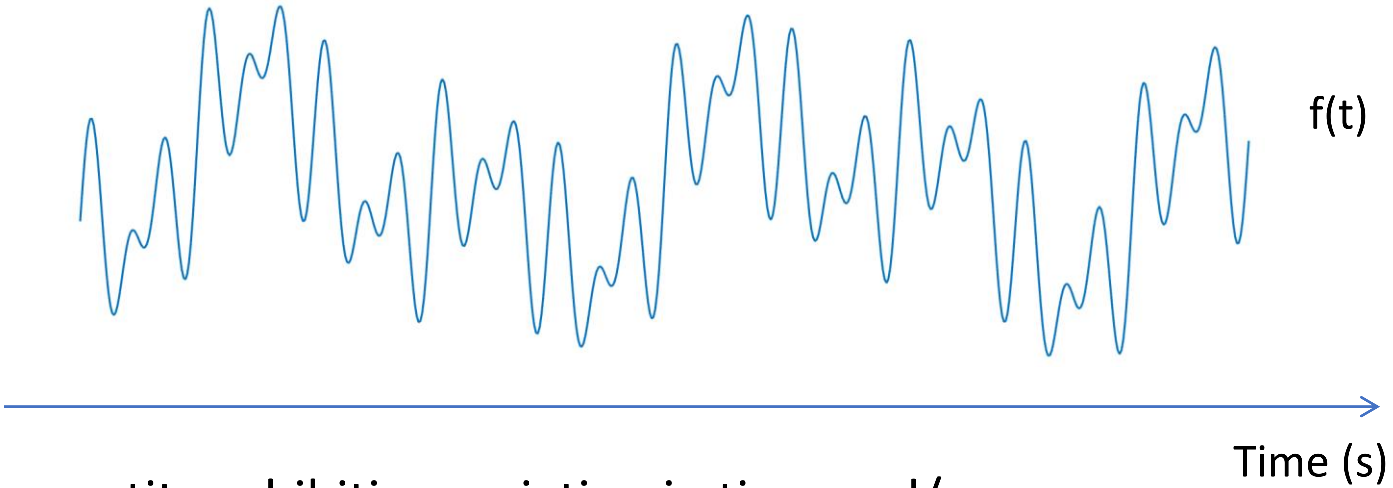
Linear shift-invariant systems

Aaron Ponti

Signals

Signals

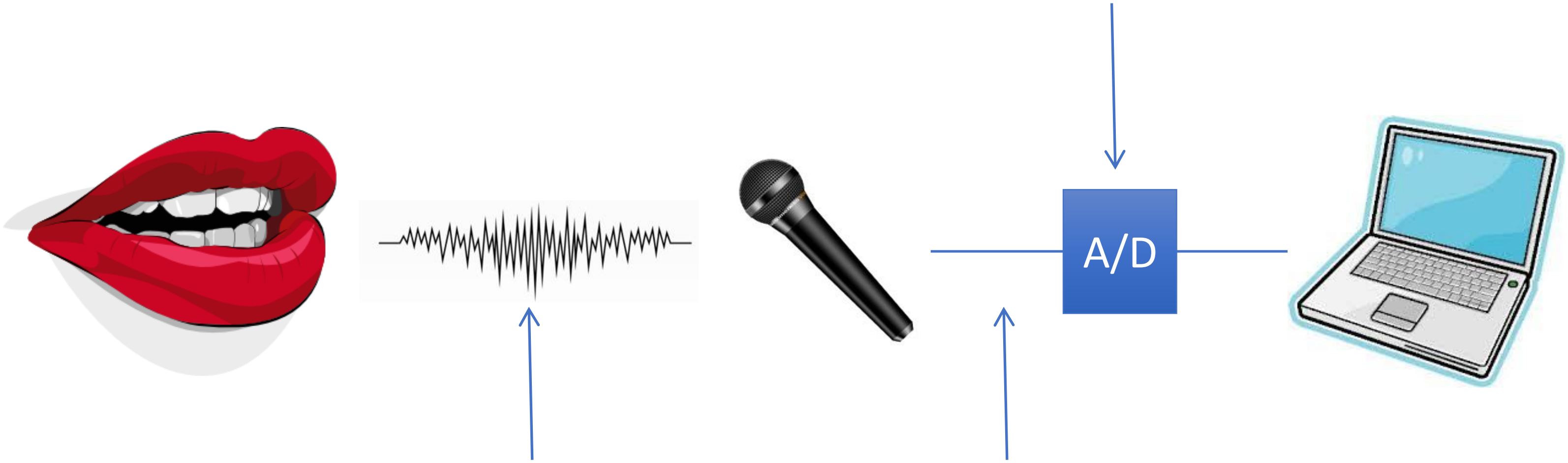
- A function containing information about some phenomenon of interest.



- A quantity exhibiting variation in time and/or space.

Analog and digital signals (1D)

Analog-to-digital converter



A/D

Acoustic signal
(continuous)

Electric signal
(continuous)

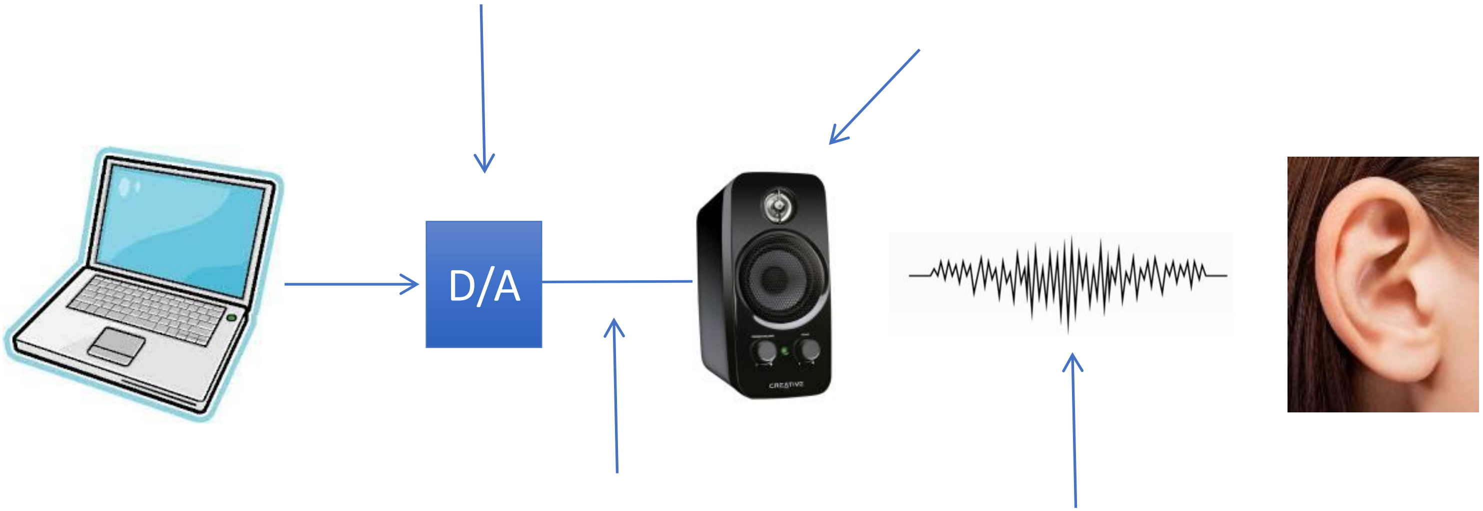
…0110101011…
(discrete)

1D

Analog and digital signals (1D)

Digital-to-analog converter

Transducer



...0110101011...
(discrete)

Electric signal
(continuous)

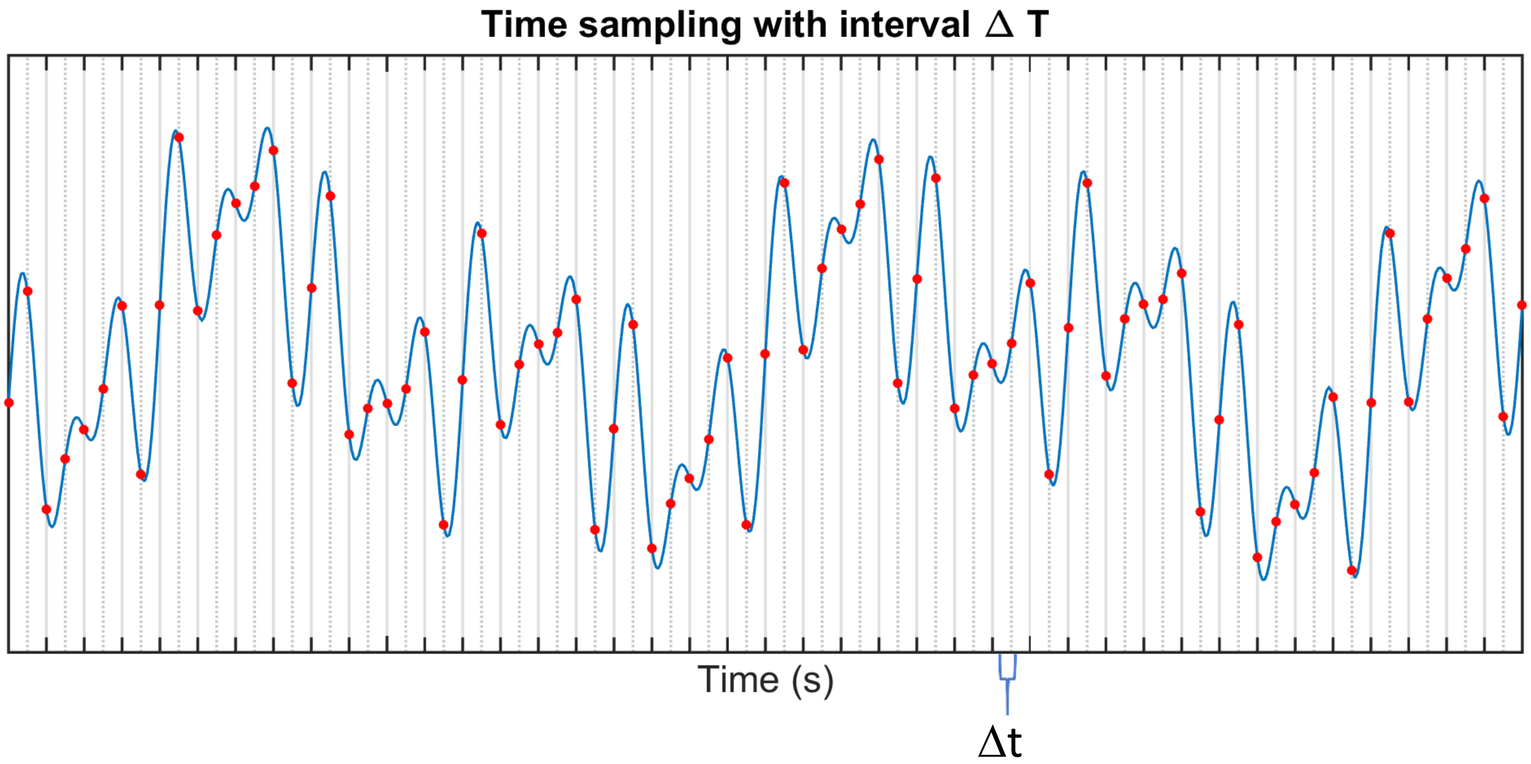
Acoustic signal
(continuous)

1D

A/D conversion

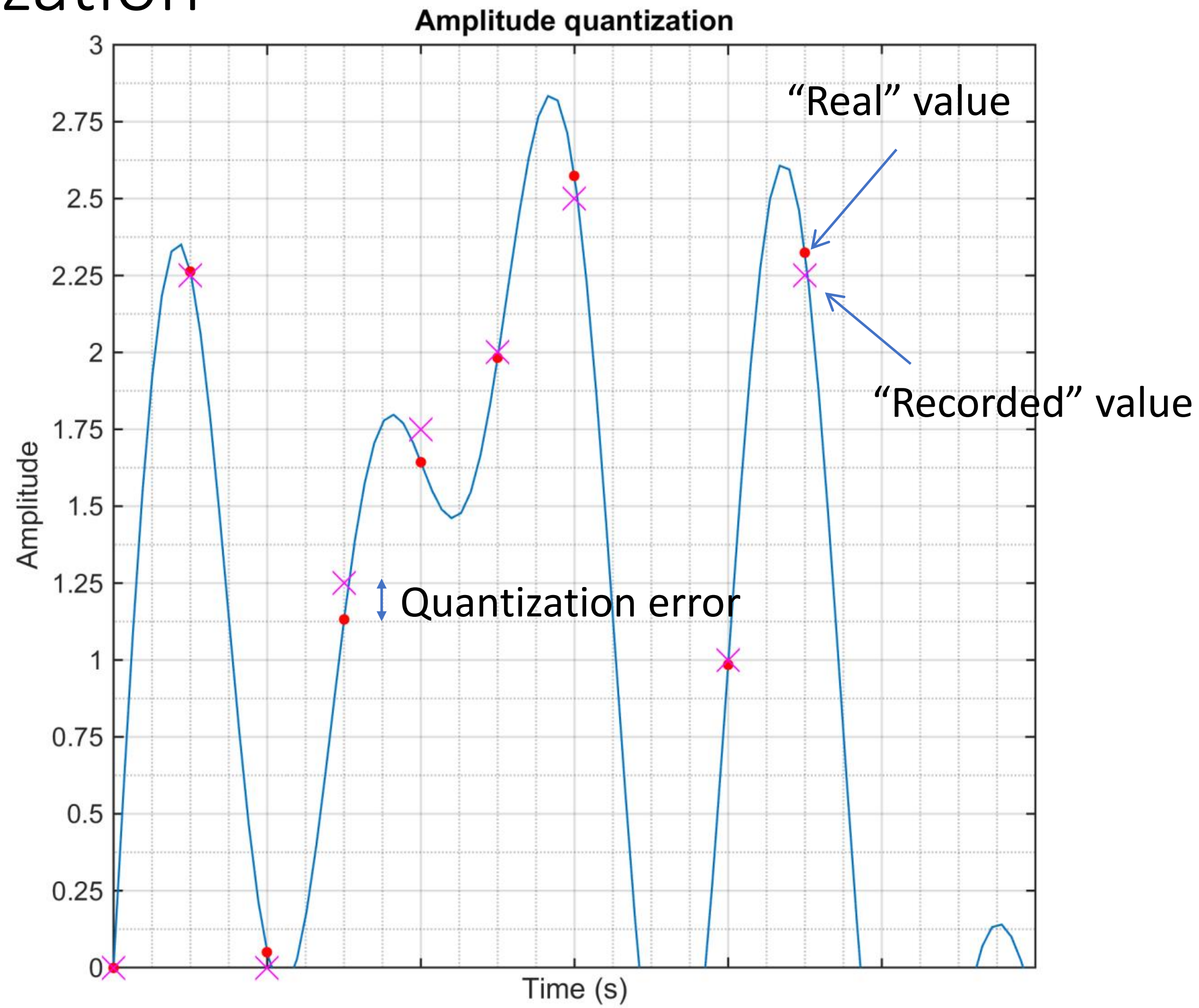
- Analog-to-digital conversion is a 2-step process:
  - **Sampling**: converts a **continuous** signal into a **discrete** one
  - **Quantization**: discretizes the **amplitude** of the signal.

Sampling



The continuous signal (blue) is measured at discrete time intervals (red dots).

Quantization



The continuous signal (blue) at discrete time intervals is approximated to a fixed number of discrete values (magenta crosses).

Example: CD sound quality
44.1 kHz, 16bit, stereo

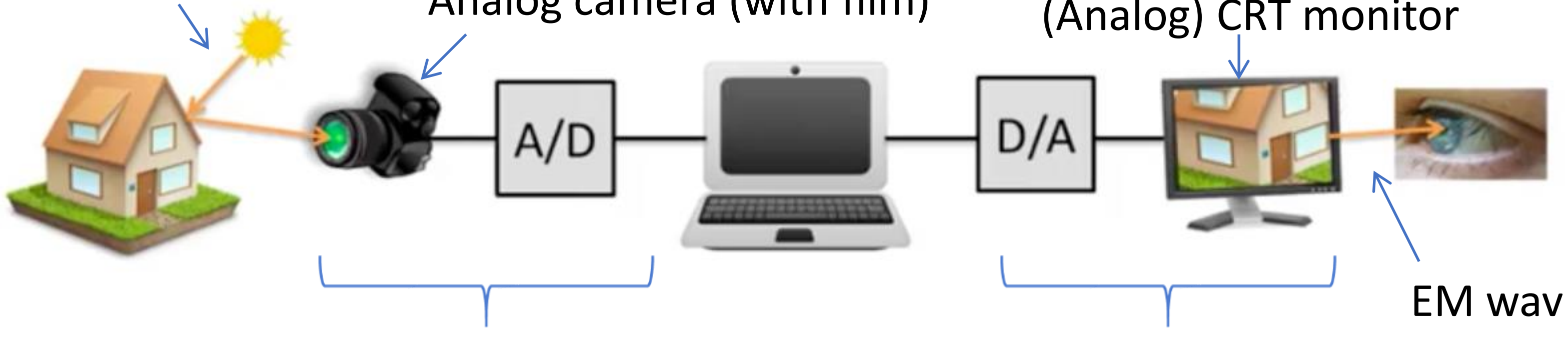
Compact disc

2 channels, each with:

Sampling rate
44.1 kHz → $\Delta T = 1/\text{rate} \approx 2.3 \cdot 10^{-5}$ s

Quantization levels
16 bit → $2^{16} = 65536$ levels

Analog and digital signals (2D)



Electromagnetic waves (continuous)

Analog camera (with film)

(Analog) CRT monitor

EM waves (continuous)

Digital camera

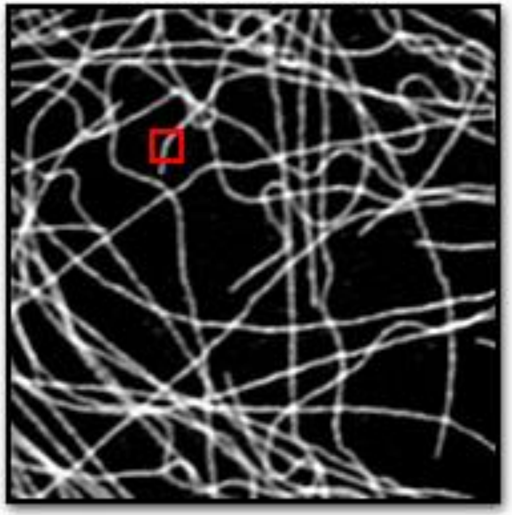
Digital monitor

2D

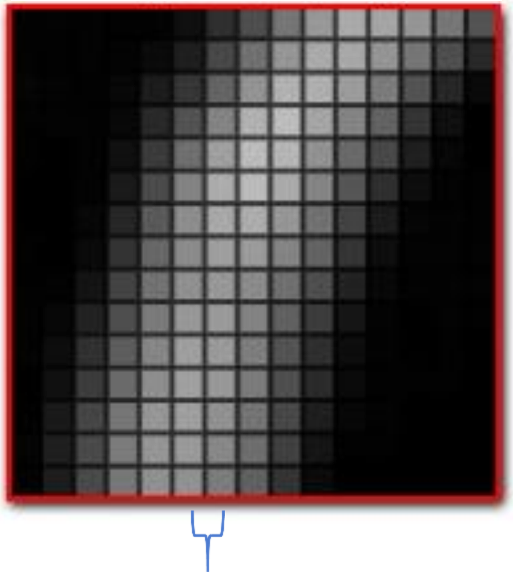
Sampling and quantization (2D)

Digital image

Analog signal

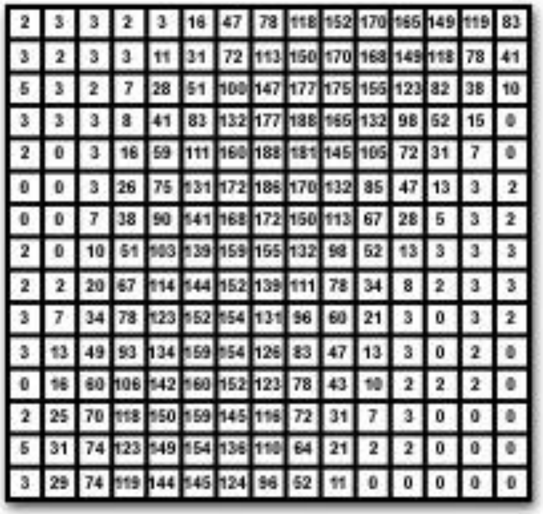


Sampling



Δx

Quantization



2	3	3	2	3	16	47	78	118	152	170	165	149	119	83
3	2	3	3	11	31	72	113	150	170	168	149	118	78	41
5	3	2	7	28	51	100	147	177	175	156	123	82	38	10
3	3	3	8	41	83	132	177	188	165	132	98	52	15	0
2	0	3	16	59	111	160	188	181	145	105	72	31	7	0
0	0	3	26	75	131	172	186	170	132	85	47	13	3	2
0	0	7	38	90	141	168	172	150	113	67	28	5	3	2
2	0	10	51	103	139	159	155	132	98	52	13	3	3	3
2	2	20	67	114	144	152	139	111	78	34	8	2	3	3
3	7	34	78	123	152	154	131	96	60	21	3	0	3	2
3	13	49	93	134	159	154	126	83	47	13	3	0	2	0
0	16	60	106	142	160	152	123	78	43	10	2	2	2	0
2	25	70	118	150	159	145	116	72	31	7	3	0	0	0
5	31	74	123	149	154	136	110	64	21	2	2	0	0	0
3	29	74	119	144	145	124	96	52	11	0	0	0	0	0

In this example, signal intensity is approximated by **256 discrete values (8 bits)**.

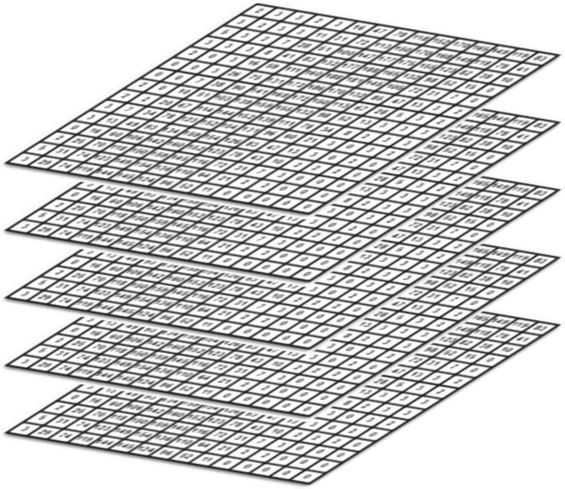
$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \ldots f(0,N-1) \\ f(1,0) & f(1,1) & \ldots f(1,N-1) \\ \ldots & \ldots & \ldots \\ f(N-1,0) & f(N-1,1) & \ldots f(M-1,N-1) \end{bmatrix}$$

2D and 3D images

- In 2D images, each grid element, or ***pixel*** (picture element), is defined as a location and a value representing the characteristic of the signal at that location.

I(x,y)

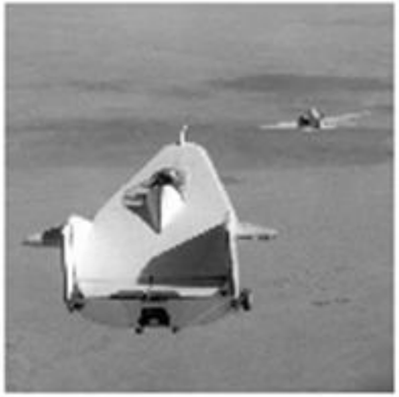
2	3	3	2	3	16	47	78	118	152	170	165	149	119	83
3	2	3	3	11	31	72	113	150	170	168	149	118	78	41
5	3	2	7	28	51	100	147	177	175	155	123	82	38	10
3	3	3	8	41	83	132	177	188	165	132	98	52	15	0
2	0	3	16	59	111	160	188	181	145	105	72	31	7	0
0	0	3	26	75	131	172	186	170	132	85	47	13	3	2
0	0	7	38	90	141	168	172	150	113	67	28	5	3	2
2	0	10	51	103	139	159	155	132	98	52	13	3	3	3
2	2	20	67	114	144	152	139	111	78	34	8	2	3	3
3	7	34	78	123	152	154	131	96	60	21	3	0	3	2
3	13	49	93	134	159	154	126	83	47	13	3	0	2	0
0	16	60	106	142	160	152	123	78	43	10	2	2	2	0
2	25	70	118	150	159	145	116	72	31	7	3	0	0	0
5	31	74	123	149	154	136	110	64	21	2	2	0	0	0
3	29	74	119	144	145	124	96	52	11	0	0	0	0	0



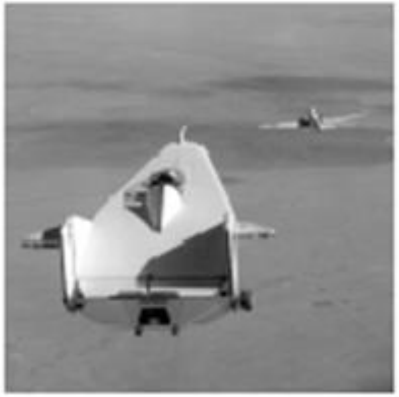
I(x,y,z)

- In 3D images, the pixel is called ***voxel*** (volume element).
- Common in the field of **biomedical imaging**.

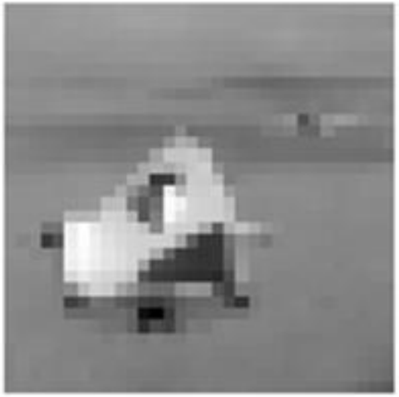
Sampling → spatial resolution



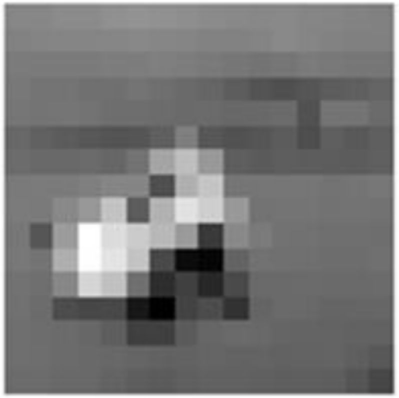
256 x 256



128 x 128

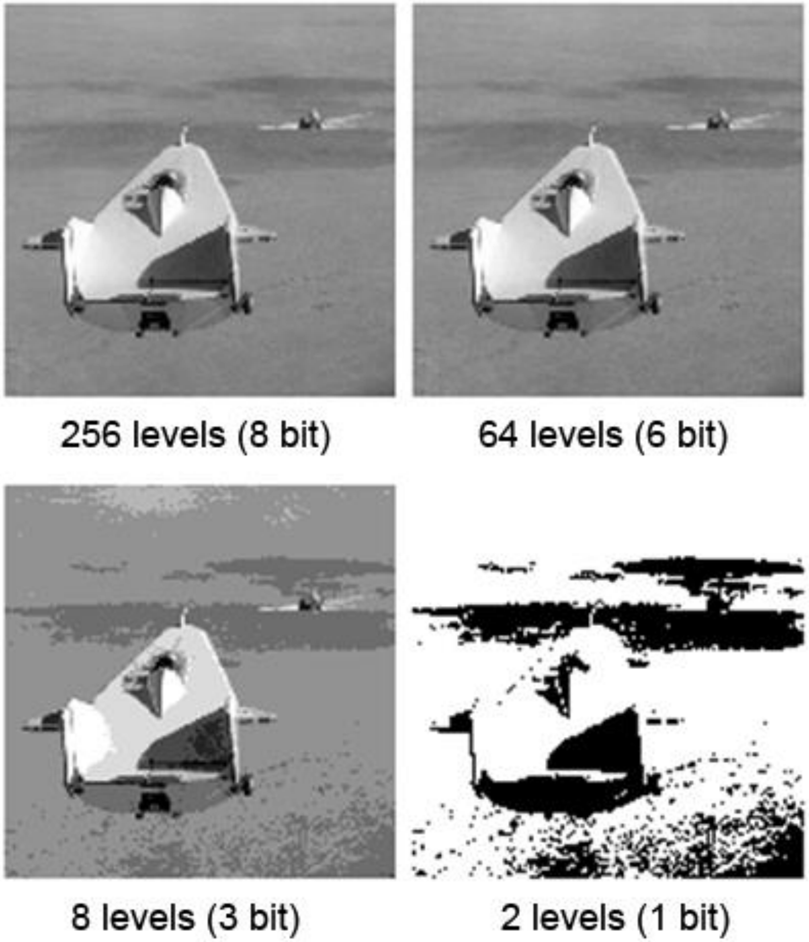


32 x 32



16 x 16

Quantization ➔ grayscale resolution



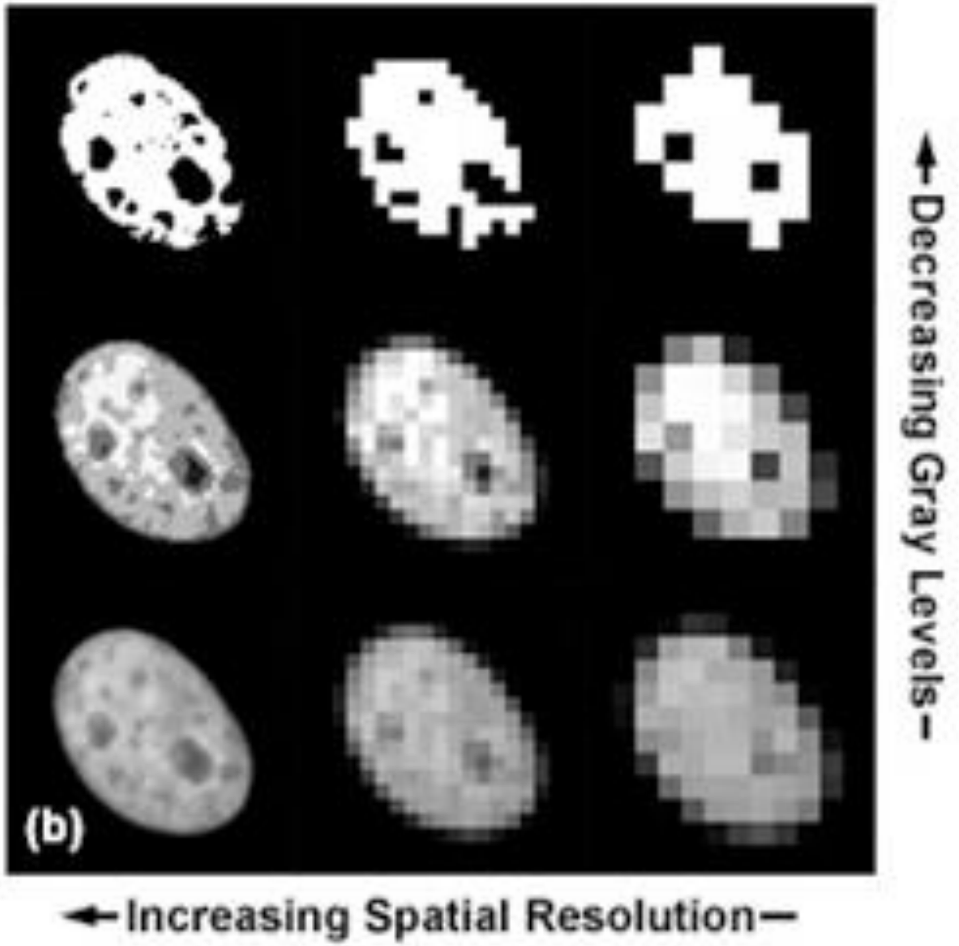
256 levels (8 bit)

64 levels (6 bit)

8 levels (3 bit)

2 levels (1 bit)

Resolution summary

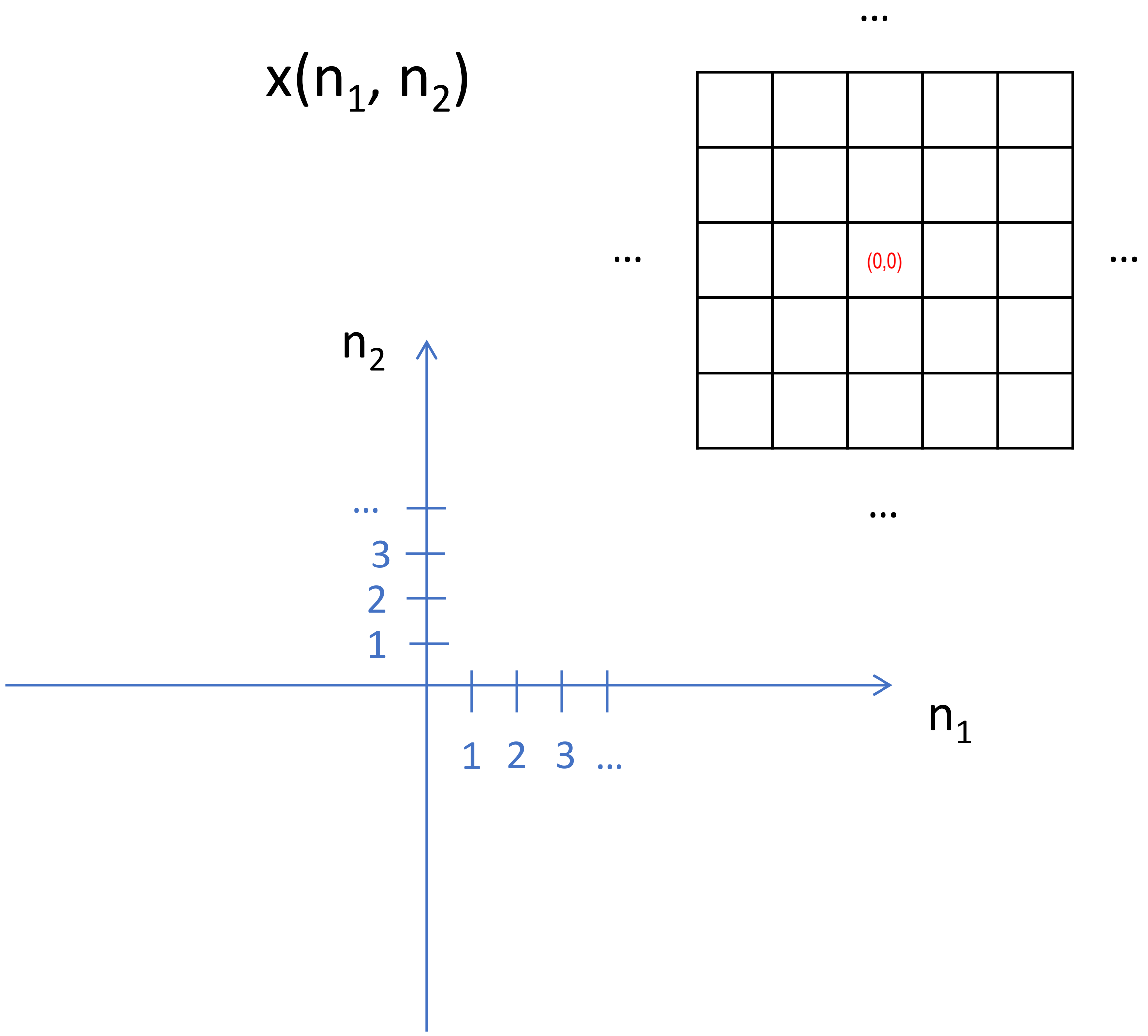


Digital Camera System

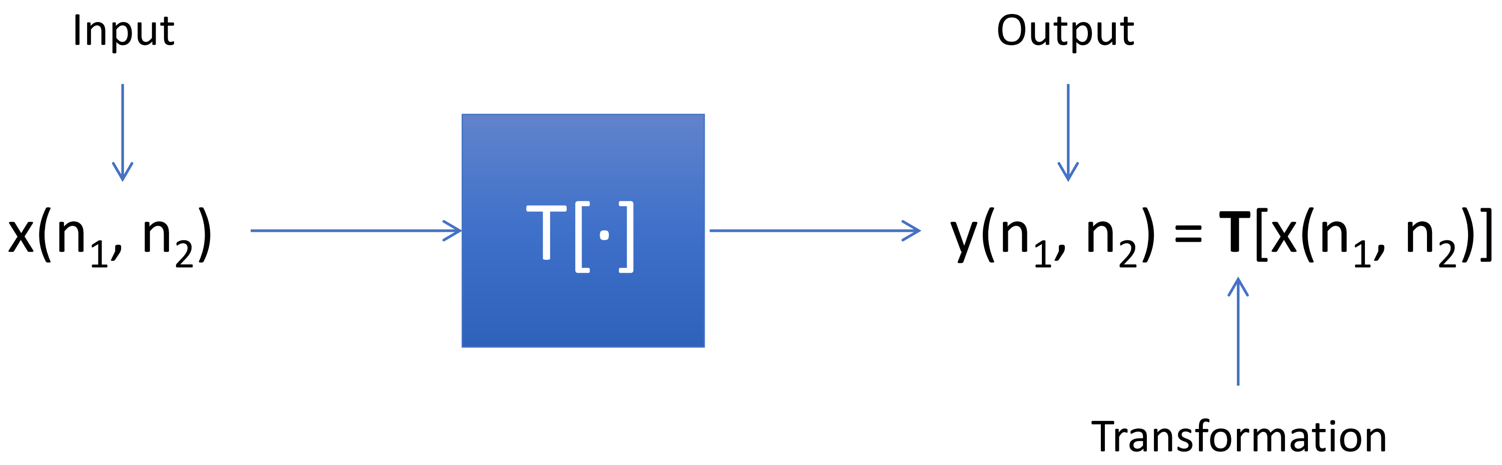
Systems

Discrete signal (notation)

$x(n_1, n_2)$



Systems



Input

$x(n_1, n_2) \longrightarrow \boxed{T[\cdot]} \longrightarrow y(n_1, n_2) = \mathbf{T}[x(n_1, n_2)]$

Output

Transformation

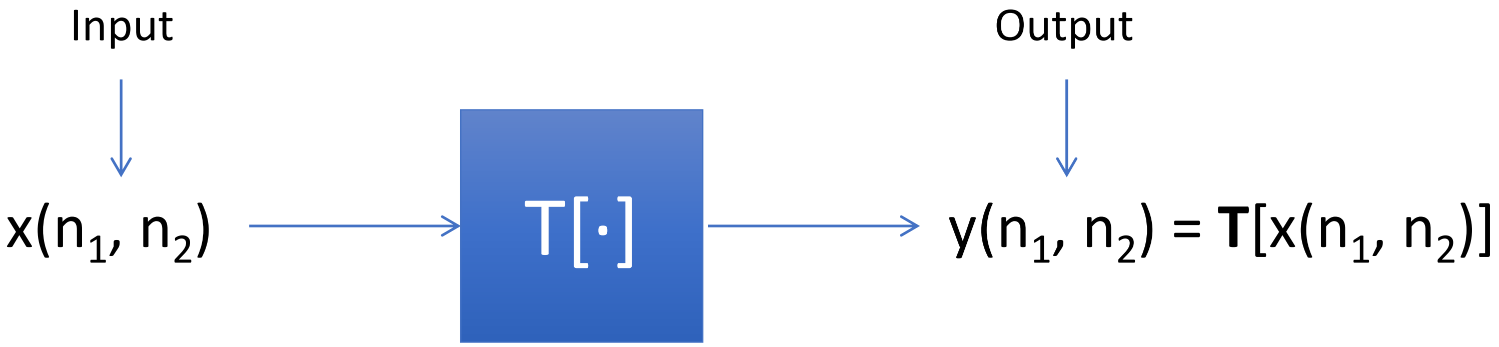
<u>Examples</u>:

$y(n_1, n_2) = 255 - x(n_1, n_2)$

$y(n_1, n_2) = \text{median}(\mathbf{N}(x(n_1, n_2)))$

A *neighborhood* of a given position (pixel) $x(n_1, n_2)$

Systems



Input

Output

$x(n_1, n_2)$

$T[\cdot]$

$y(n_1, n_2) = \mathbf{T}[x(n_1, n_2)]$

T[] can be any sort of transformation (system) of the input signal $x(n_1, n_2)$.

We will now consider a family of systems with following properties:

- Linearity
- Spatial (shift) invariance

Linear systems

If

$$\mathbf{T}[a_1x_1(n_1, n_2) + a_2x_2(n_1, n_2)] = a_1\mathbf{T}[x_1(n_1, n_2)] + a_2\mathbf{T}[x_2(n_1, n_2)]$$

then **T**[] is linear.

The transformed version of a weighted sum of signals is the same as the weighted sum of the signals transformed individually.

(Alternatively, a linear system can be decomposed into constituents that are processed independently, and the result combined in the end.)

Shift-invariant systems

Given:

$$\mathbf{T}[x(n_1, n_2)] = y(n_1, n_2)$$

If

$$\mathbf{T}[x(n_1 - k_1, n_2 - k_2)] = y(n_1 - k_1, n_2 - k_2)$$

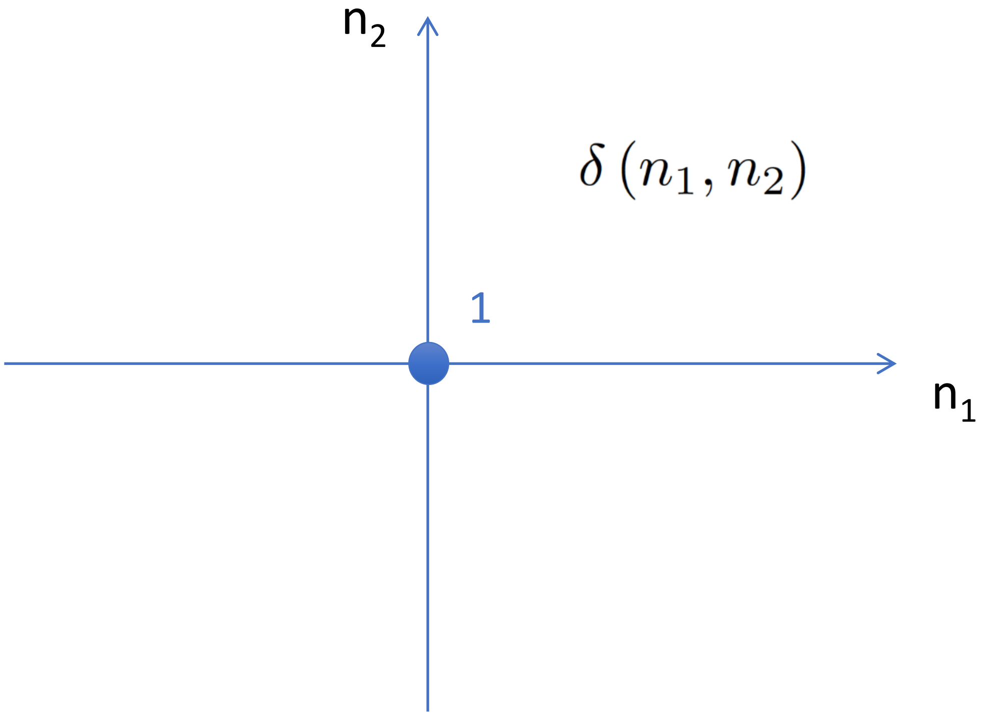
then **T**[] is shift-invariant.

If the input is shifted by a given amount, the output will be shifted by the same amount.

(Or, the location of the origin of the coordinate system is irrelevant.)

Discrete Unit Impulse

$$\delta(n_1, n_2) = \begin{cases} 1, & for\ n_1 = n_2 = 0 \\ 0, & otherwise \end{cases}$$



…

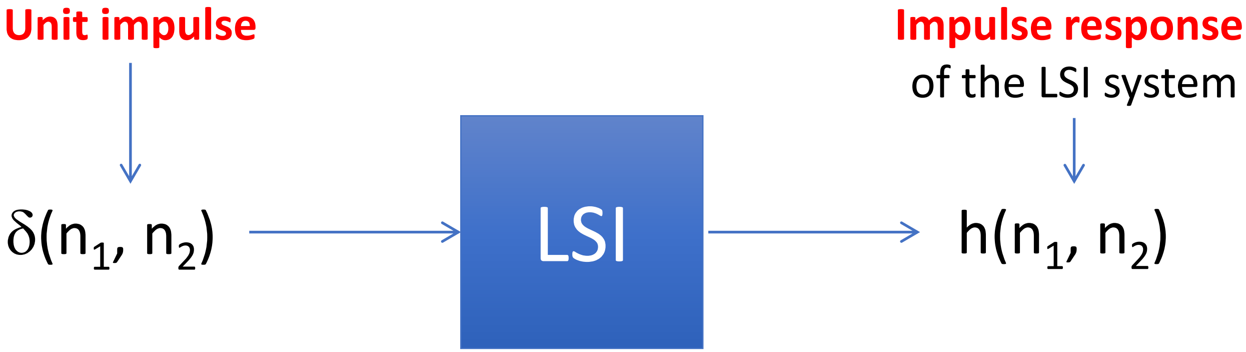
…

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

…

…

Linear Shift-Invariant systems



Unit impulse

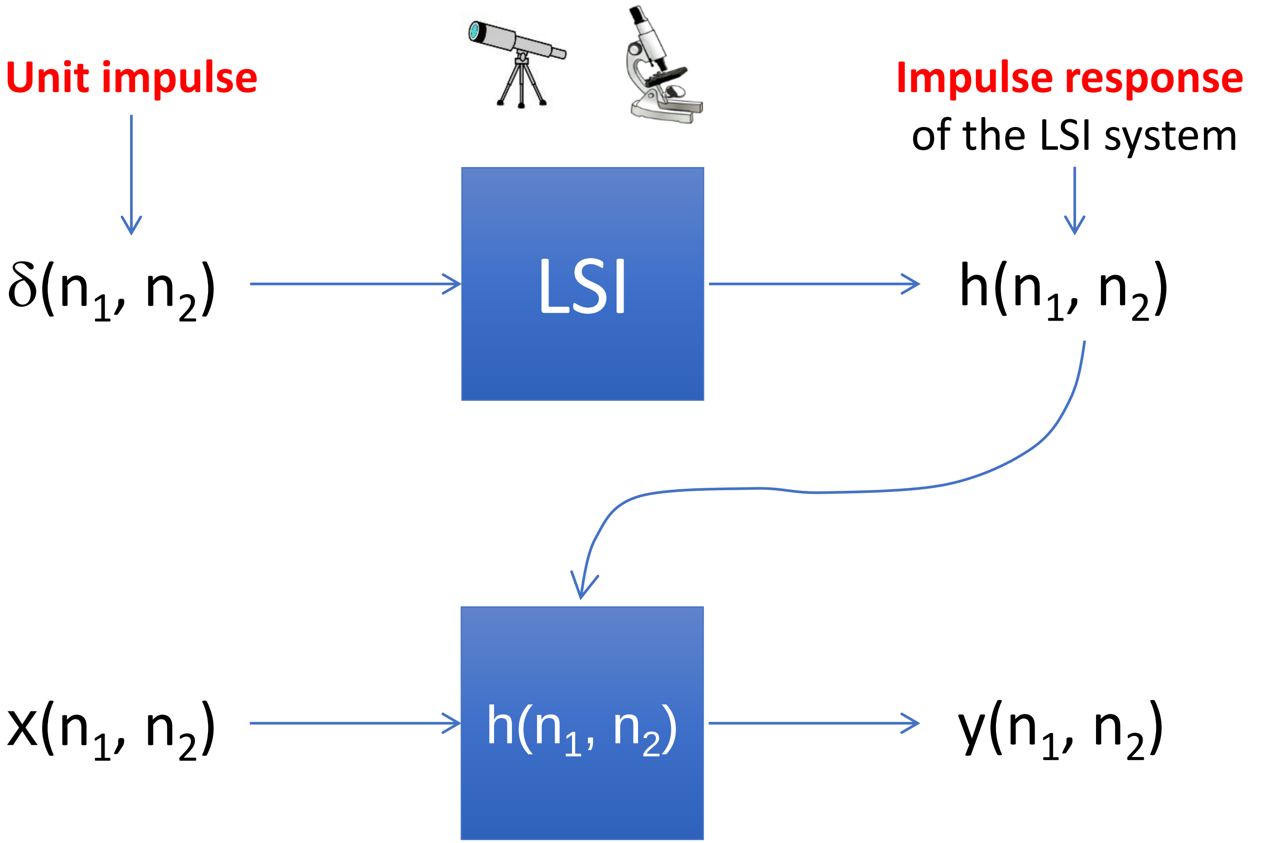
$\delta(n_1, n_2)$

LSI

Impulse response of the LSI system

$h(n_1, n_2)$

Linear Shift-Invariant systems



Unit impulse

$\delta(n_1, n_2)$

LSI

Impulse response of the LSI system

$h(n_1, n_2)$

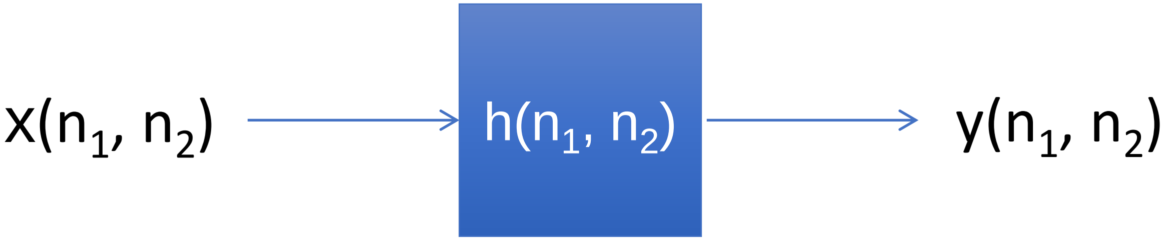
$x(n_1, n_2)$

$h(n_1, n_2)$

$y(n_1, n_2)$

The system response to the unit impulse is all we need to fully describe the LSI system.

Convolution



x(n_1, n_2)

h(n_1, n_2)

y(n_1, n_2)

LSI systems can be described and efficiently implemented by the mathematical operation of **convolution**.

$$\boxed{y(n_1, n_2) = x(n_1, n_2) \circledast h(n_1, n_2)}$$

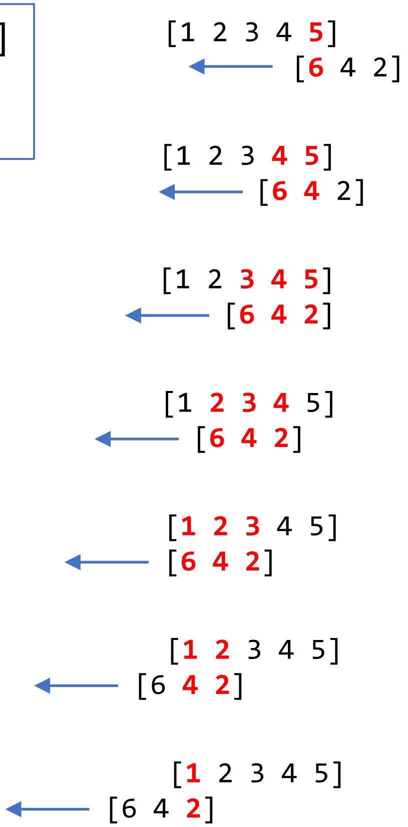
$$y\left(n_1, n_2\right) = x\left(n_1, n_2\right) \circledast h\left(n_1, n_2\right) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x\left(k_1, k_2\right) h\left(n_1 - k_1, n_2 - k_2\right)$$

Convolution (1D example)

$$y(n) = x(n) \circledast h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n \overset{\downarrow}{-} k)$$

x = [1 2 3 4 5]

h = [2 4 6]



[1 2 3 4 **5**]
[**6** 4 2]

5 * 6 = [**30**]

[1 2 3 **4** **5**]
[**6** **4** 2]

6 * 4 + 5 * 4 = [**44** 30]

[1 2 **3** **4** **5**]
[**6** **4** **2**]

3 * 6 + 4 * 4 + 5 * 2 = [**44** 44 30]

[1 **2** **3** **4** 5]
[**6** **4** **2**]

2 * 6 + 3 * 4 + 4 * 2 = [**32** 44 44 30]

[**1** **2** **3** 4 5]
[**6** **4** **2**]

1 * 6 + 2 * 4 + 3 * 2 = [**20** 32 44 44 30]

[**1** **2** 3 4 5]
[6 **4** **2**]

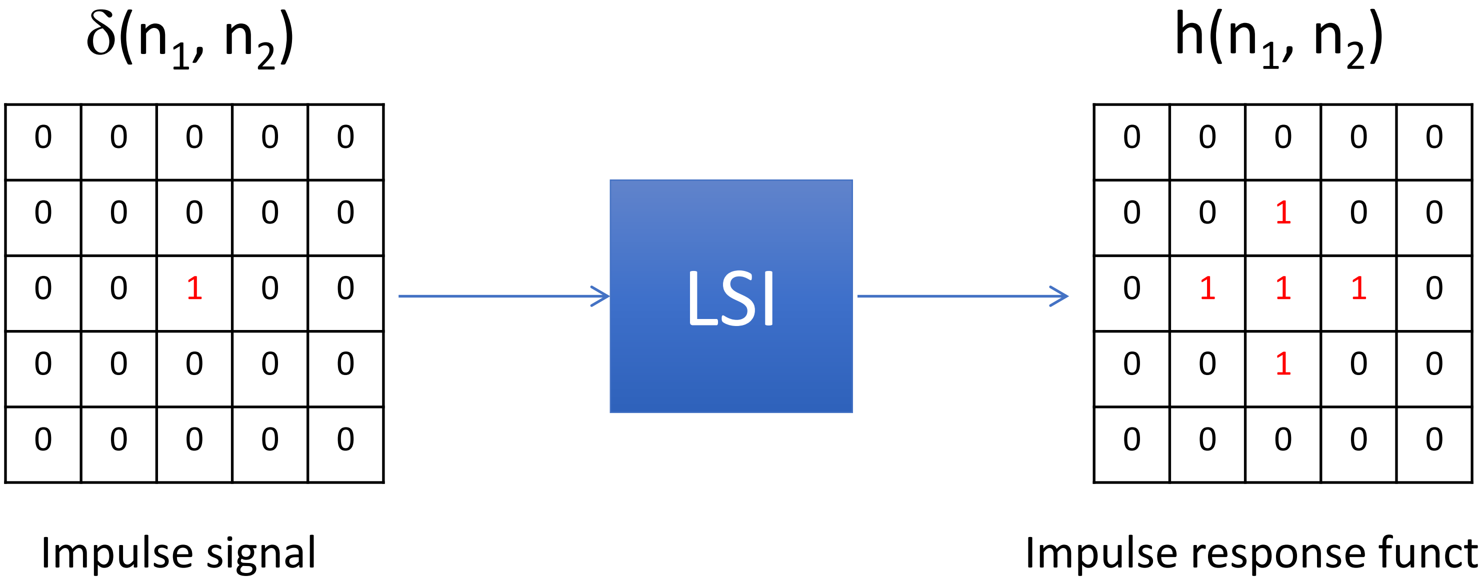
1 * 4 + 2 * 2 = [**8** 20 32 44 44 30]

[**1** 2 3 4 5]
[6 4 **2**]

1 * 2 = [**2** 8 20 32 44 44 30]

Full convolution: y(n) = [2 8 20 32 44 44 30]

Linear Shift-Invariant systems



$\delta(n_1, n_2)$

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

Impulse signal

$h(n_1, n_2)$

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

Impulse response function

Linear Shift-Invariant systems

$x(n_1, n_2)$

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

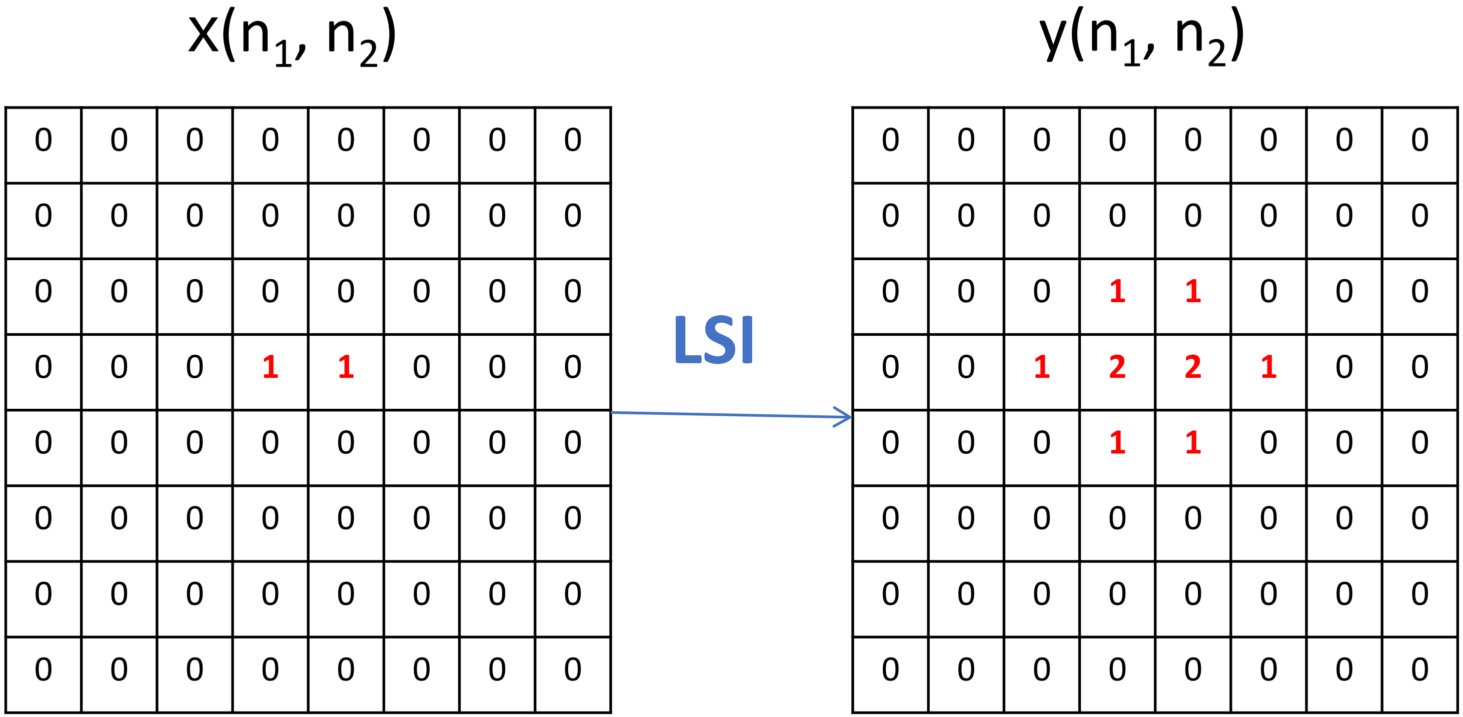
$$\xrightarrow{\text{LSI}}$$

$y(n_1, n_2)$

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

$$y(n_1, n_2) = x(n_1, n_2) \circledast h(n_1, n_2)$$

Linear Shift-Invariant systems



$x(n_1, n_2)$

0	0	0	0	0	0	0	0
0	0	0	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**	**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	0	0	0
0	0	0	0	0	0	0	0

$y(n_1, n_2)$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**1**	**1**	0	0	0
0	0	**1**	**2**	**2**	**1**	0	0
0	0	0	**1**	**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	0	0	0

The output signal is formed as a linear combination (i.e. weighted sum) of spatially-shifted[1] impulse response functions.

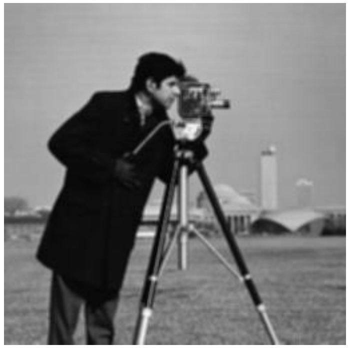
[1] Or time-shifted in 1D.

Spatial filtering through convolution

Examples of convolution filters

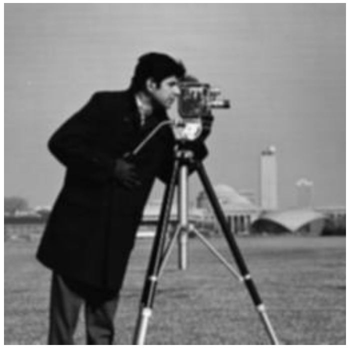
$x(n_1, n_2)$

3x3 average filter
(poor) Low-pass filter



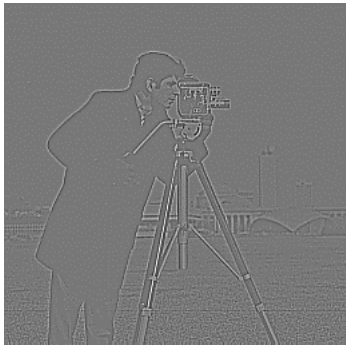
$$h = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

3x3 Gaussian filter
(better) Low-pass filter



$$h = \begin{bmatrix} 0.0751 & 0.1238 & 0.0751 \\ 0.1238 & 0.2042 & 0.1238 \\ 0.0751 & 0.1238 & 0.0751 \end{bmatrix}$$

High-pass filter



$$h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

“kernels” h

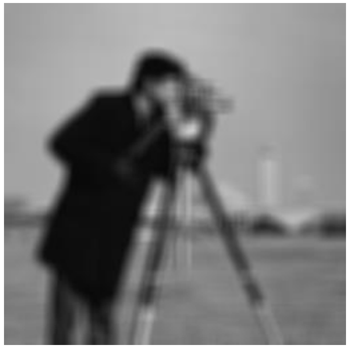
$$y(n_1, n_2) = x(n_1, n_2) \circledast h(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2)\, h(n_1 - k_1, n_2 - k_2)$$

Spatial filtering through convolution

Examples of convolution filters

$x(n_1, n_2)$

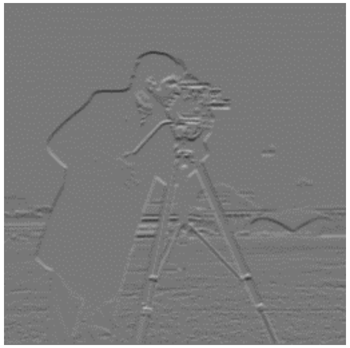
11x11 average filter



$$h = \begin{bmatrix} 1/121 & 1/121 & 1/121 & \dots & 1/121 \\ 1/121 & 1/121 & 1/121 & \dots & 1/121 \\ 1/121 & 1/121 & 1/121 & \dots & 1/121 \\ \dots & \dots & \dots & \dots & \dots \\ 1/121 & 1/121 & 1/121 & \dots & 1/121 \end{bmatrix}$$

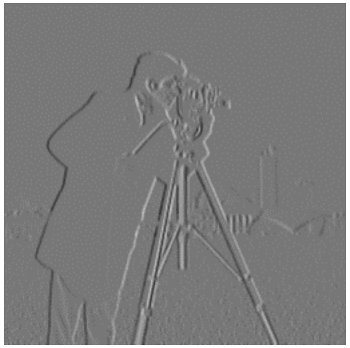
Larger “support” of the filter

Prewitt
Vertical edge



$$h = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt
Horizontal edge

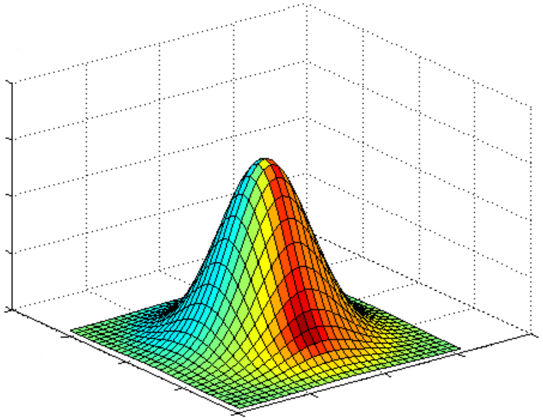


$$h = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

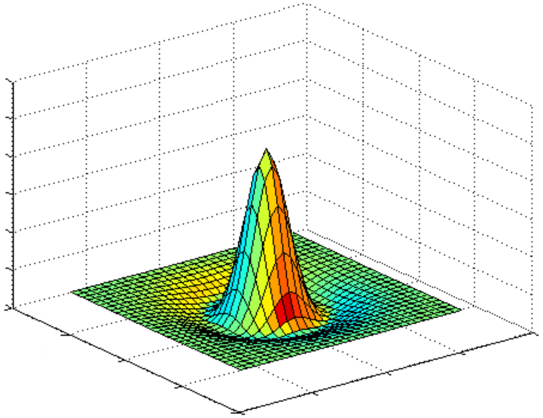
“kernels” h

$$y(n_1, n_2) = x(n_1, n_2) \circledast h(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2)\, h(n_1 - k_1, n_2 - k_2)$$

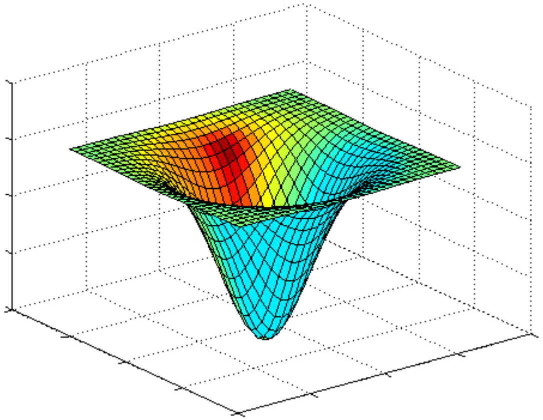
Spatial filtering



Low-pass filter
Gaussian kernel



Band-pass filter
Difference of Gaussians kernel



High-pass filter
Laplacian kernel

The convolution of a signal in the spatial domain has very specific effects on the frequency content of the signal.

Exponential sequences

$$x\left(n_1, n_2\right) = e^{jw'_1 n_1} e^{jw'_2 n_2}$$

Euler's formula

Periodic with period 2π.

$$e^{jw'_1 n_1} e^{jw'_2 n_2} = \cos\left(w'_1 n_1 + w'_2 n_2\right) + j \sin\left(w'_1 n_1 + w'_2 n_2\right)$$

Polar representation

Cartesian representation

What happens if we pass exponential sequences through an LSI system?

$$\overset{e^{jw'_1 n_1} e^{jw'_2 n_2}}{x(n_1, n_2)} \xrightarrow[h(n_1, n_2)]{\text{LSI}} y(n_1, n_2)\ ?$$

Frequency response of a system

$$x(n_1, n_2) \xrightarrow[h(n_1, n_2)]{\text{LSI}} y(n_1, n_2)\ ?$$

$$\boxed{\text{We calculate the convolution of } x(n_1, n_2) \text{ with the impulse response } h(n_1, n_2)\text{:}}$$

$$y(n_1, n_2) = x(n_1, n_2) \otimes h(n_1, n_2)$$

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} e^{jw'_1(n_1-k_1)} e^{jw'_2(n_2-k_2)} h(k_1, k_2)$$

$$y(n_1, n_2) = \underbrace{e^{jw'_1 n_1} e^{jw'_2 n_2}}_{x(n_1, n_2)} \underbrace{\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(k_1, k_2)\, e^{-jw'_1 \overset{\downarrow}{k_1}} e^{-jw'_2 \overset{\downarrow}{k_2}}}_{H(\omega'_1, \omega'_2)}$$

The signal goes through untouched!

Frequency response of a system

$$y\left(n_1, n_2\right) = \underbrace{e^{jw'_1 n_1} e^{jw'_2 n_2}}_{x\left(n_1, n_2\right)} \underbrace{\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h\left(k_1, k_2\right) e^{-jw'_1 k_1} e^{-jw'_2 k_2}}_{H\left(\omega'_1, \omega'_2\right)}$$

$H\left(\omega'_1, \omega'_2\right)$:

- is the **frequency response** of the system
- tells us how the LSI system reacted to the input frequencies
- is the **Fourier transform** of the impulse response $h(n_1,n_2)$
- has a magnitude and a phase

Exponential sequences

(Joseph Fourier, 1768 – 1830)

Exponential sequences are **building blocks** of <u>any signal</u> and so called **eigen-functions** of LSI systems.

Frequencies are left untouched

$$\underline{e^{jw_1n_1}e^{jw_2n_2}} \xrightarrow{\text{LSI}} A \cdot \underline{e^{jw_1n_1}e^{jw_2n_2}} \cdot e^{j\phi}$$

Amplitude change

Linear phase shift

… and any signal

LSI systems cannot produce frequencies that are not in the input.

Continuous Fourier Transform

- We consider the **continuous** Fourier transform of a **discrete signal**.
- The Fourier transform maps a signal to its frequency representation.

Fourier transform

Continuous variables

Discrete signal

$$X\left(\omega_1, \omega_2\right) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x\left(n_1, n_2\right) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

- The Fourier transform is periodic with period 2π in the ω_1 and ω_2 directions (since the exponential sequences have the same periodicity).

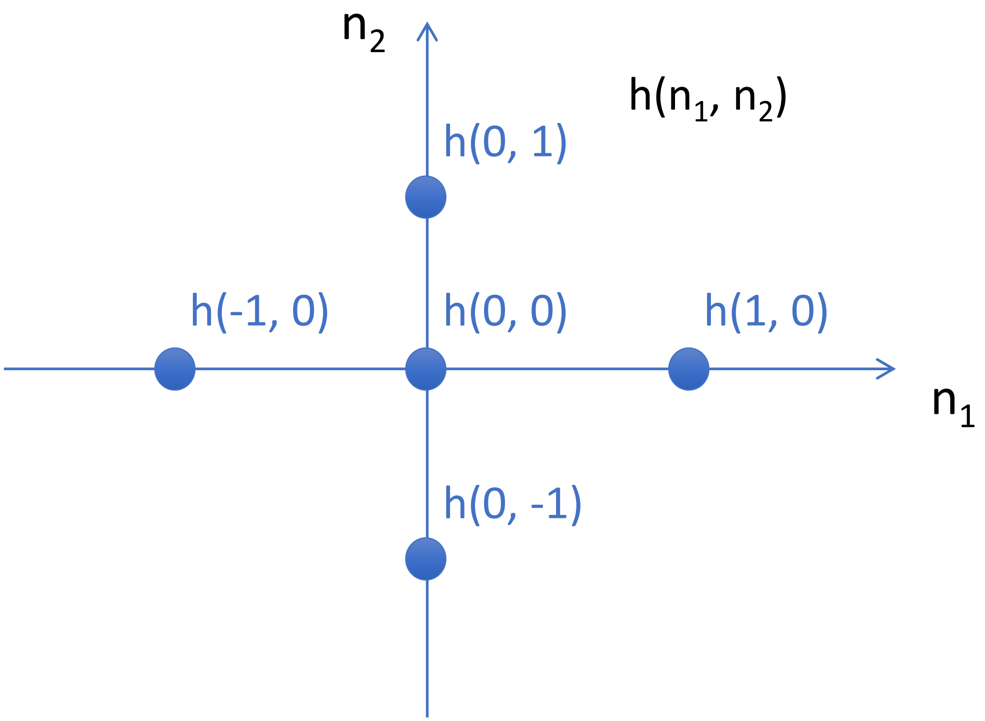
Inverse Fourier transform

$$x\left(n_1, n_2\right) = \frac{1}{4\pi^2} \underbrace{\int_{-\pi}^{\pi} \int_{-\pi}^{\pi}}_{\text{One period}} X\left(\omega_1, \omega_2\right) e^{j\omega_1 n_1} e^{j\omega_2 n_2} d\omega_1 d\omega_2$$

← Important implications when **sampling** the signal!

Frequency response: example

Given an LSI system and its impulse response $h(n_1, n_2)$, we want to calculate its (continuous) frequency response $H(\omega_1, \omega_2)$, i.e. the Fourier Transform of $h(n_1, n_2)$.



	1/3	
1/3	1/6	1/3
	1/3	

$h(n_1, n_2)$

$$H\left(\omega_1,\omega_2\right)=\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}h\left(n_1,n_2\right)e^{-j\omega_1 n_1}e^{-j\omega_2 n_2}$$

Frequency response: example

$$H\left(\omega_1,\omega_2\right)=\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}h\left(n_1,n_2\right)e^{-j\omega_1 n_1}e^{-j\omega_2 n_2}$$

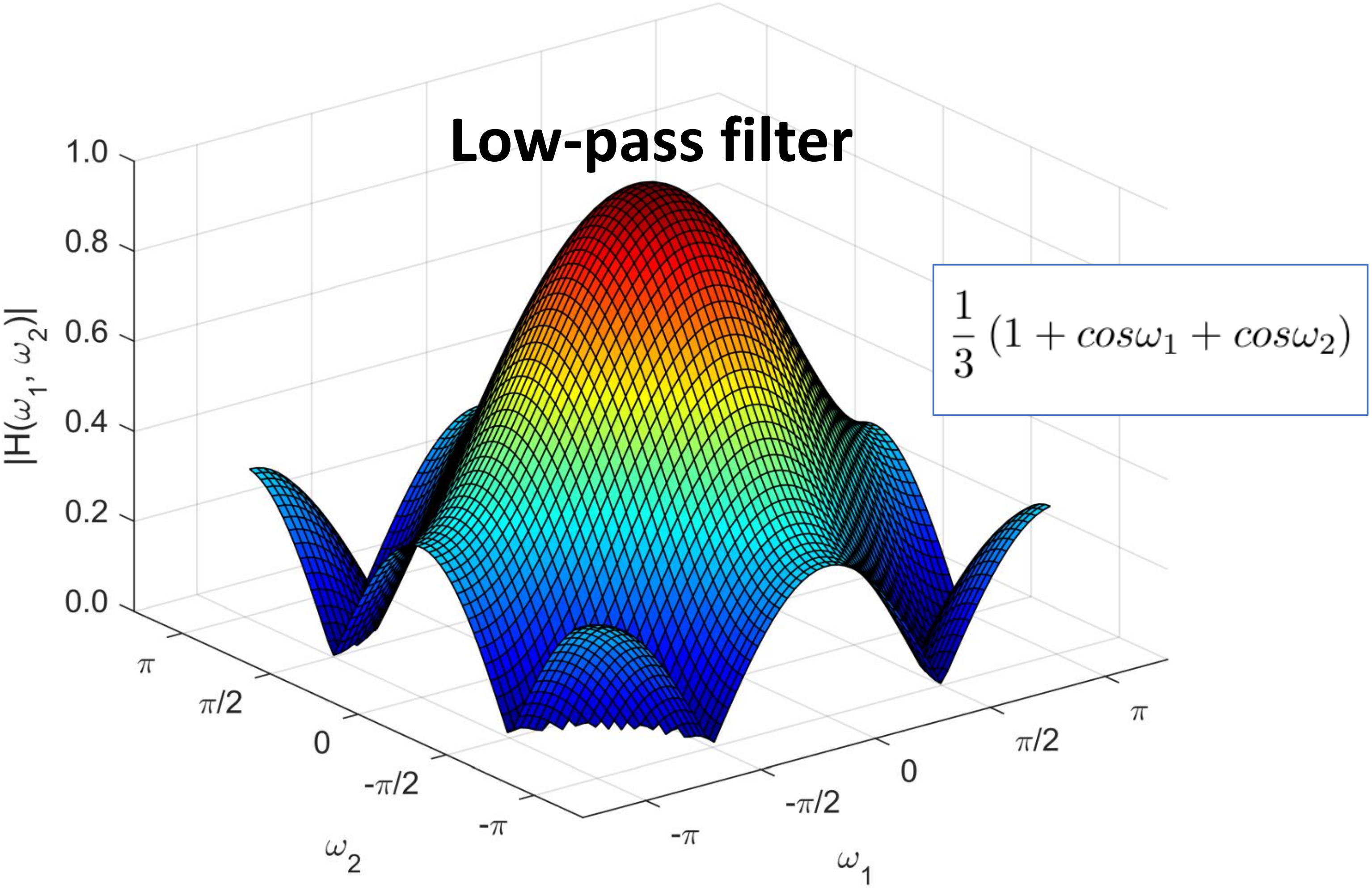
$$h\left(0,0\right)+h\left(-1,0\right)e^{j\omega_1}+h\left(1,0\right)e^{-j\omega_1}+h\left(0,-1\right)e^{j\omega_2}+h\left(0,1\right)e^{-j\omega_2}=$$

$$\frac{1}{3}+\frac{1}{6}e^{j\omega_1}+\frac{1}{6}e^{-j\omega_1}+\frac{1}{6}e^{j\omega_2}+\frac{1}{6}e^{-j\omega_2}=$$

$$\frac{1}{3}+\frac{1}{6}\cdot 2cos\omega_1+\frac{1}{6}\cdot 2cos\omega_2=\frac{1}{3}\left(1+\underset{\uparrow}{cos\omega_1}+\underset{\uparrow}{cos\omega_2}\right)$$

Continuous and periodic.

Frequency response: example



Low-pass filter

$$\frac{1}{3}\left(1 + cos\omega_1 + cos\omega_2\right)$$

Magnitude of the frequency response
of $h(n_1, n_2)$ over one period ($-\pi : \pi$)

$H(0, 0) = 1$

$H(-\pi, \pi/2) = 0$

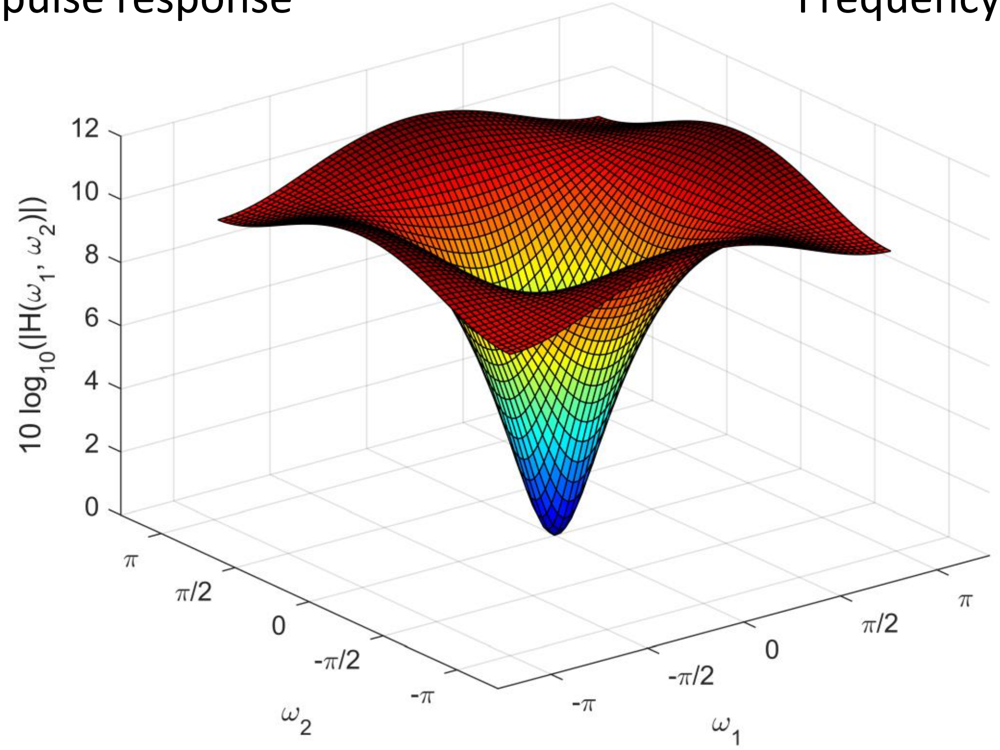
$H(\pi, \pi/2) = 0$

Frequency response: example 2

$$h(n_1, n_2) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \longrightarrow H(\omega_1, \omega_2) = 9 - 2 \cdot cos\omega_1 - 2 \cdot cos\omega_2 - 2 \cdot cos(\omega_1 + \omega_2) - 2 \cdot cos(\omega_1 - \omega_2)$$

Impulse response

Frequency response



High-pass filter

Convolution theorem

The convolution theorem describes the input – output relationships of an LSI system in the **frequency** domain.

$$\begin{tikzcd}
y(n_1,n_2) \arrow[d, shift right, "FT"'] & = & x(n_1,n_2) \arrow[d, shift right, "FT"'] & \circledast & h(n_1,n_2) \arrow[d, shift right, "FT"'] \\
Y(\omega_1,\omega_2) \arrow[u, shift right, "IFT"'] & = & X(\omega_1,\omega_2) \arrow[u, shift right, "IFT"'] & \cdot & H(\omega_1,\omega_2) \arrow[u, shift right, "IFT"']
\end{tikzcd}$$

Multiplication in the frequency domain

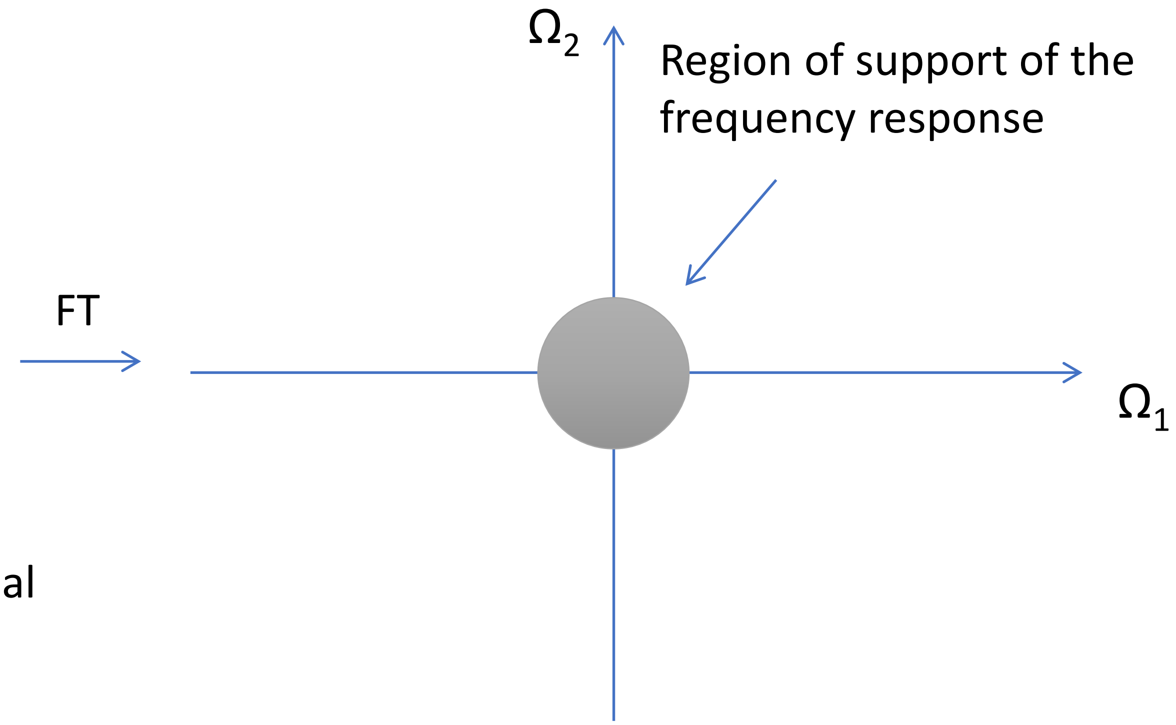
Reverse is also true.

Sampling

Under which conditions can we reconstruct a continuous, band-limited signal from its discrete representation with no loss of information?

Analog, continuous signal
(band-limited)

$X_a(\Omega_1, \Omega_2)$



Region of support of the frequency response

An alternative view: sampling can be modelled in direct space with a **multiplication** of the signal with a train of delta functions. The convolution theorem tells us that this corresponds to a **convolution** of the Fourier Transform of the signal with the Fourier Transform of the impulse train.

Sampling

Spectrum of the analog signal

$$X\left(\overset{\omega_1}{\Omega_1 T_1}, \overset{\omega_2}{\Omega_2 T_2}\right) = \frac{1}{T_1 T_2} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} X_a\left(\Omega_1 - \frac{2\pi}{T_1}k_1, \Omega_2 - \frac{2\pi}{T_2}k_2\right)$$

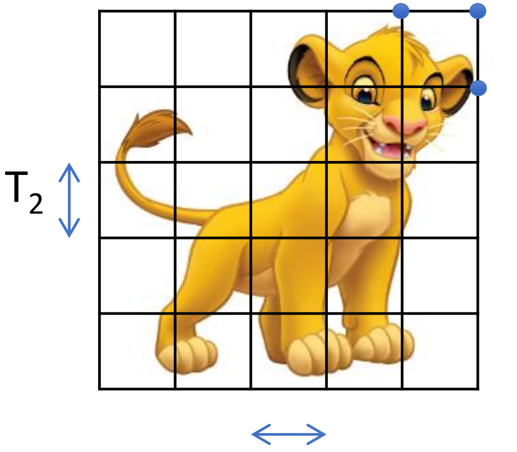
Periodic expansion

$X_S(\Omega_1 T_1, \Omega_2 T_2)$

We only store the values at the intersection points of the grid.

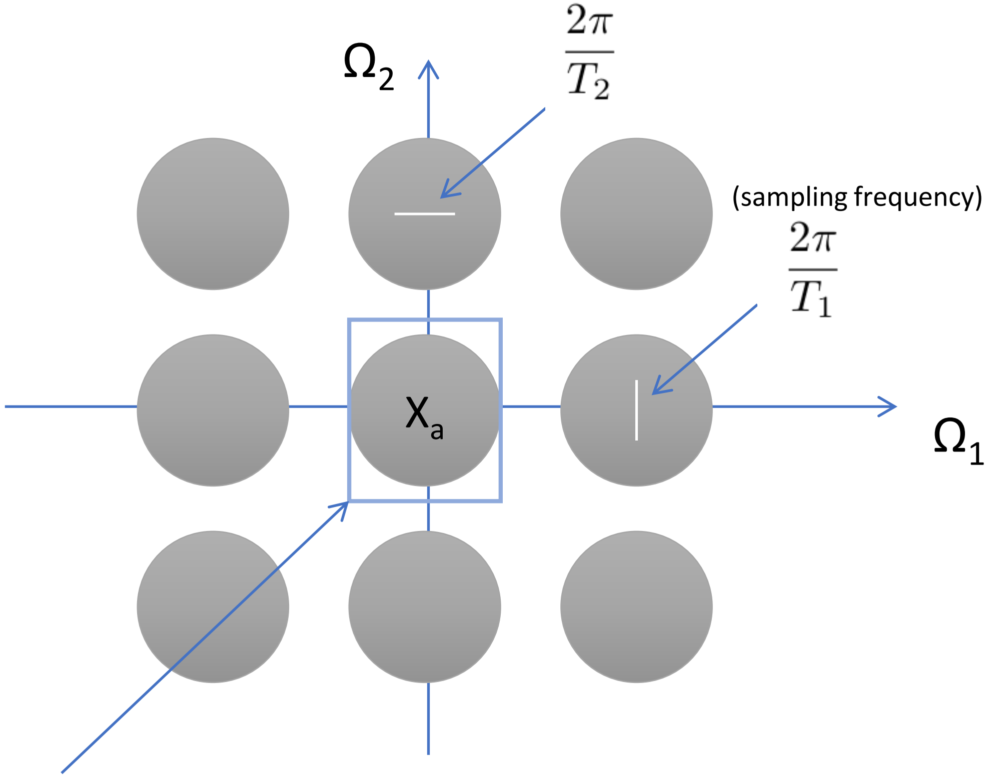
$\omega_i = \Omega_i T_i$

Sampling on a regular grid



T_1 (sampling period)

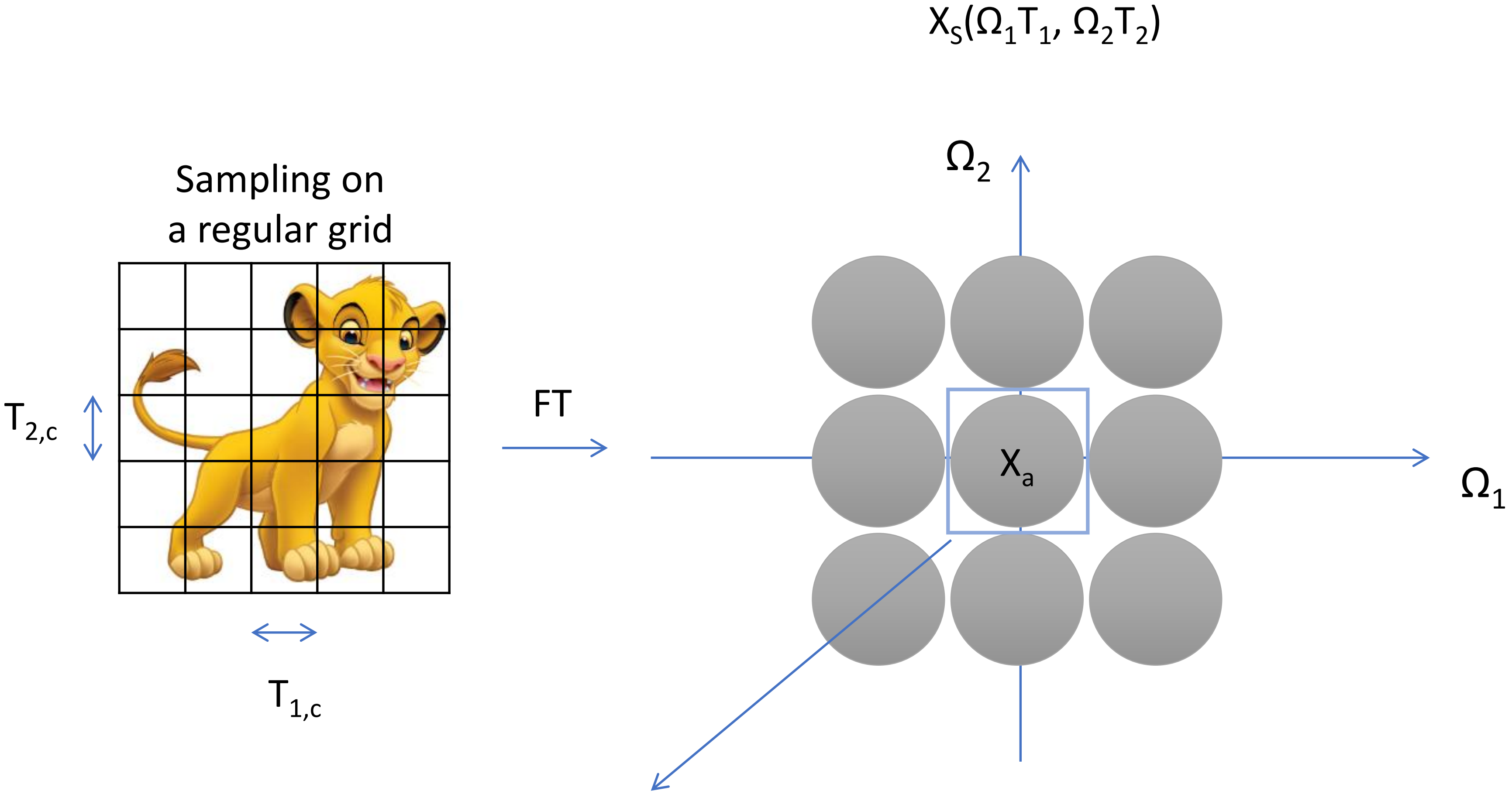
FT



(sampling frequency)

The spectrum of the analog signal X_a is periodically extended with periods $2\pi/T_i$. T_1 and T_2 define how far apart the replica of the spectrum will be.

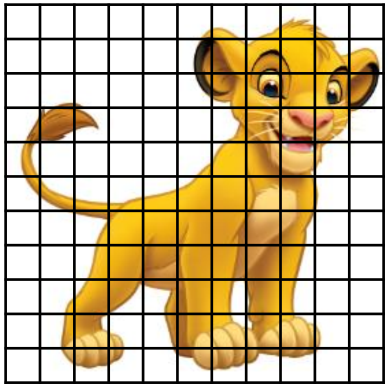
Critical sampling



This (base) band contains the spectrum of the analog signal with no loss of information.

Oversampling

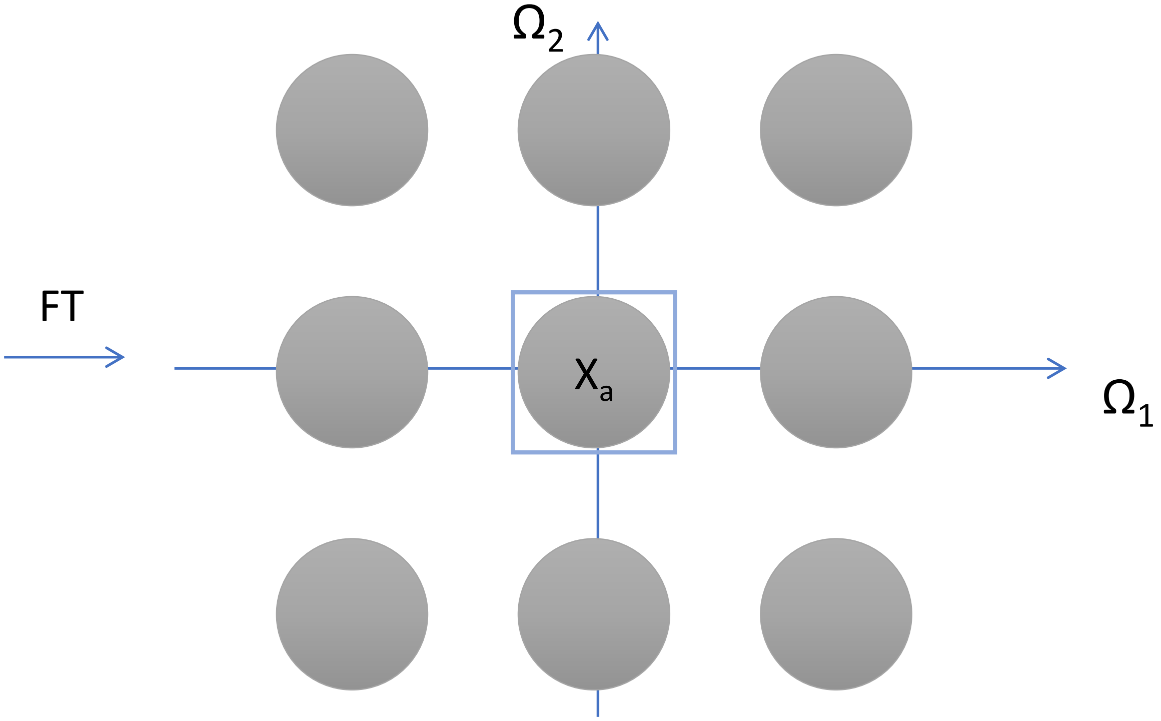
Sampling on a regular grid



$T_1 < T_{1,c}$

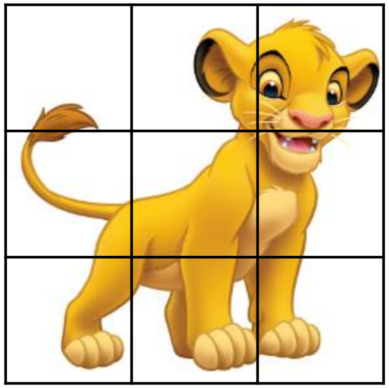
$T_2 < T_{2,c}$

$X_S(\Omega_1 T_1, \Omega_2 T_2)$



Undersampling

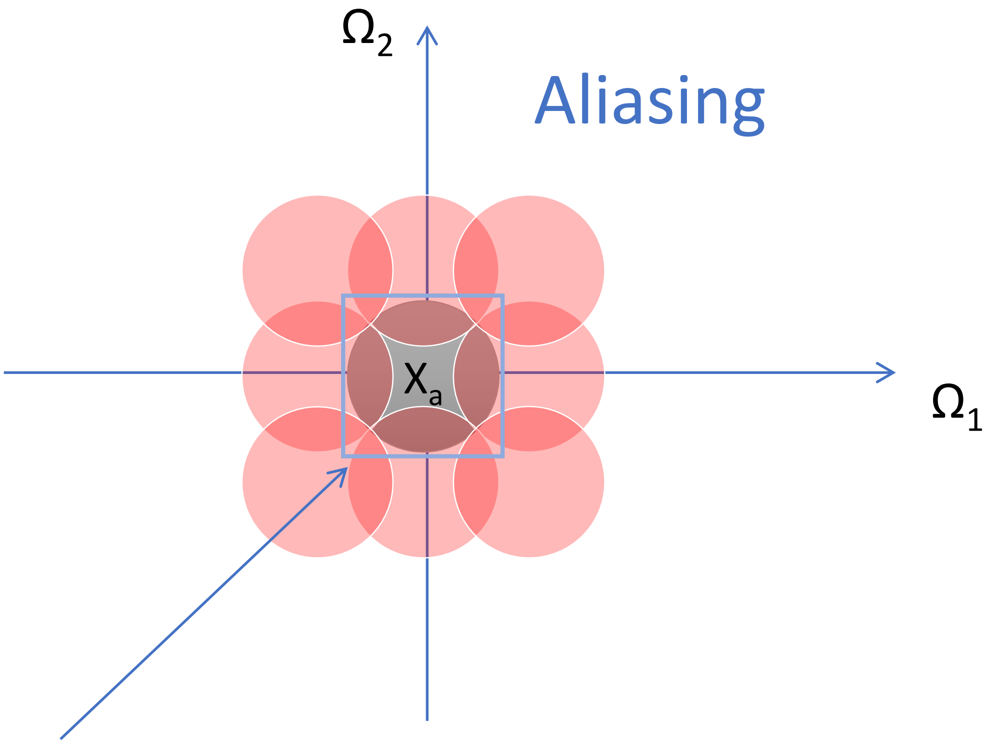
Sampling on
a regular grid



$T_1 > T_{1,c}$
$T_2 > T_{2,c}$

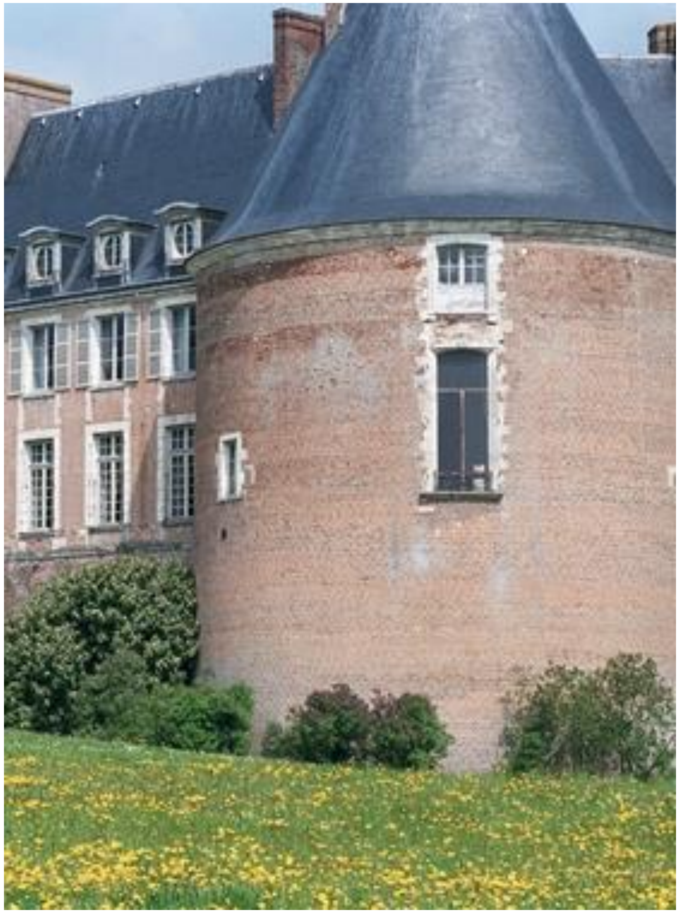
$\xrightarrow{FT}$

$X_S(\Omega_1 T_1, \Omega_2 T_2)$

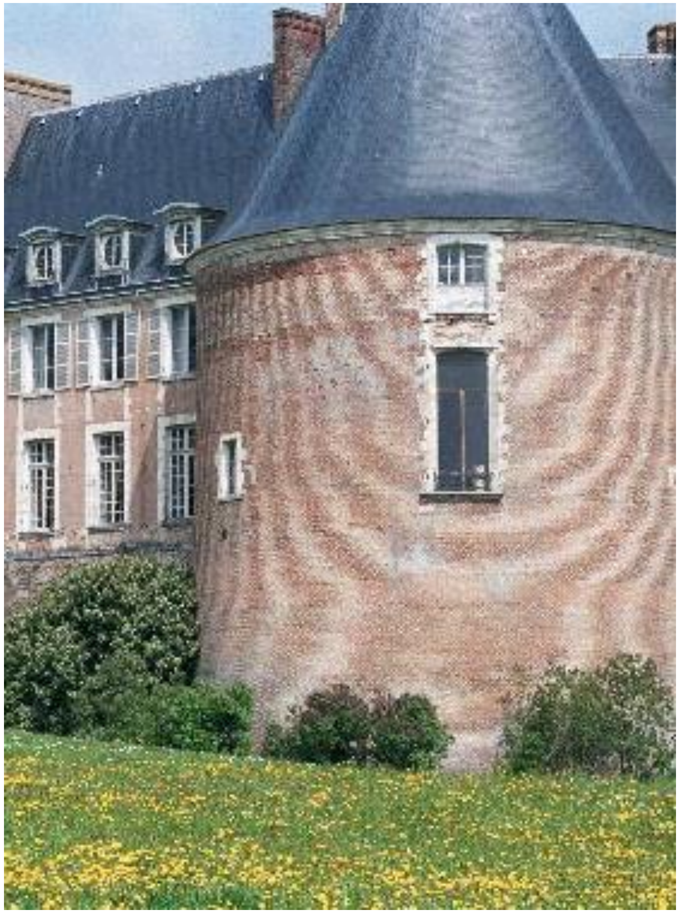


Aliased spectrum of the analog signal

Aliasing: example



Properly sampled



Undersampled and aliased

Nyquist sampling theorem

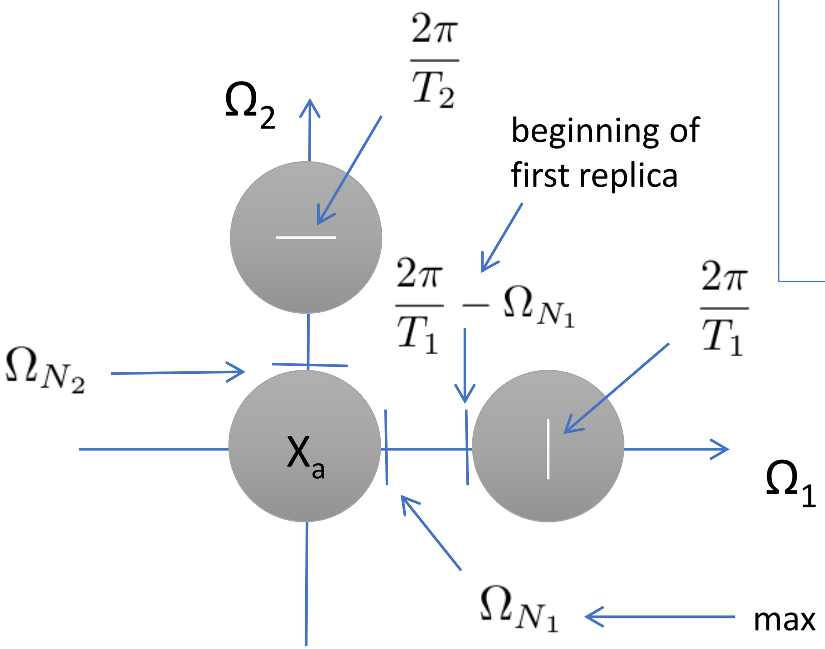
Sampling frequency

$$\frac{2\pi}{T_1} - \Omega_{N_1} \geq \Omega_{N_1} \Rightarrow \frac{2\pi}{T_1} \geq 2\Omega_{N_1}$$

$$\frac{2\pi}{T_2} - \Omega_{N_2} \geq \Omega_{N_2} \Rightarrow \frac{2\pi}{T_2} \geq 2\Omega_{N_2}$$

Nyquist critical sampling

$X_A(\Omega_1, \Omega_2)$



max frequency in n_1 direction

What should the optimal sampling period T_i be?

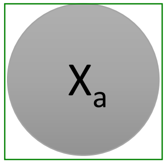
To reconstruct the analog image, extract (multiply) the digital spectrum with a low-pass filter:

$$F(\Omega_1, \Omega_2) = \begin{cases} T_1 T_2, & |\,\Omega_1\,| < \pi/T_1, |\,\Omega_2\,| < \pi/T_2 \\ 0 & otherwise \end{cases}$$

A 2D sync function in spatial domain.

Constant (with gain T_1T_2) on the support area.

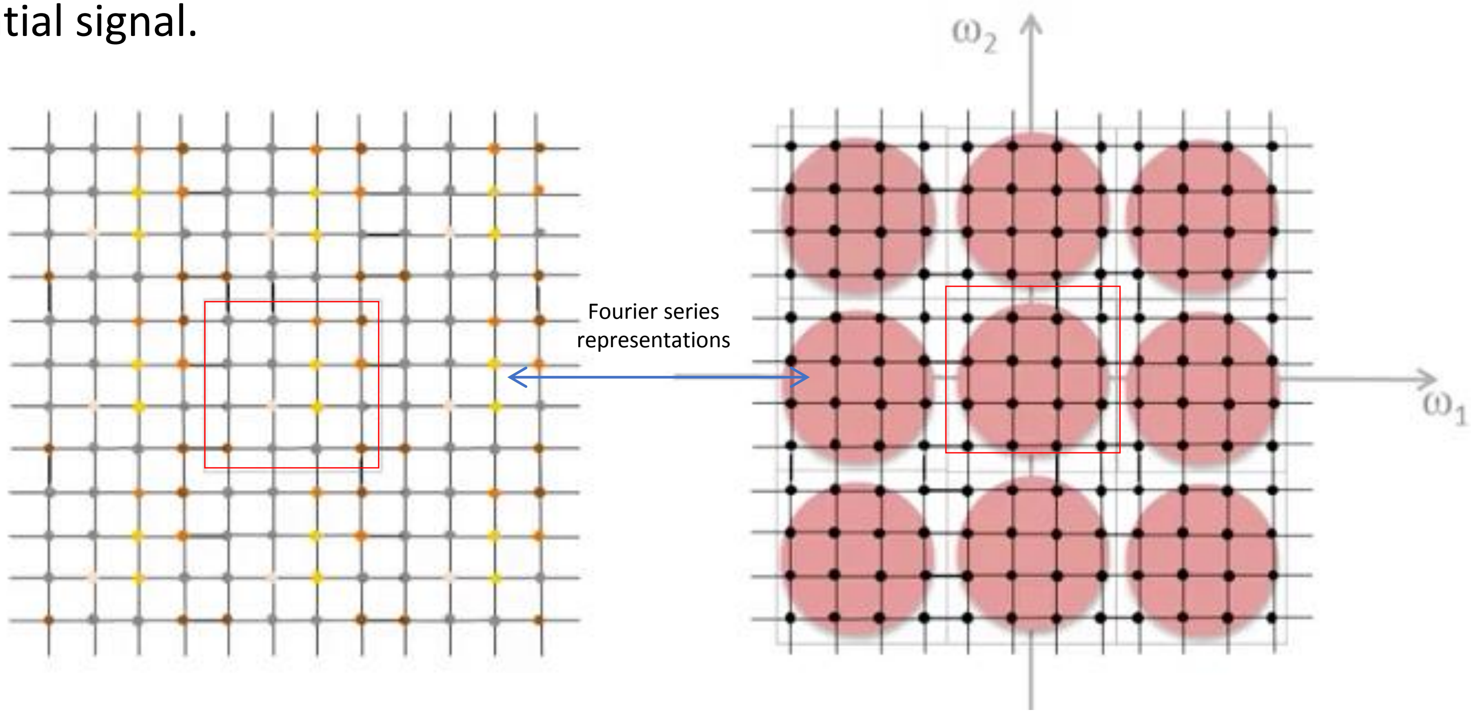
$F(\Omega_1, \Omega_2)$



Discrete Fourier Transform (DFT)

- The continuous Fourier Transform of a discrete signal is not computable
  - continuous (i.e. infinitely many) frequencies ω_1 and ω_2.

- Sampling in the frequency domain results in periodic extension of the sampled spatial signal.



Fourier series representations

- One period in the frequency domain corresponds to one period of the spatial domain: this mapping is the **Discrete Fourier Transform (DFT)**.

- A sampled version of one period of the continuous Fourier transform is all is needed to reconstruct the analog signal.

Discrete Fourier Transform (DFT)

Continuous

$$X\left(\omega_1, \omega_2\right) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x\left(n_1, n_2\right) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

Sampling in frequency space. We keep only one period.

$$X\left(k_1, k_2\right) = X\left(\omega_1, \omega_2\right) \Big|_{\omega_1 = \frac{2\pi}{N_1} k_1, \omega_2 = \frac{2\pi}{N_2} k_2} \qquad \begin{cases} k_1 = 0, \ldots, N_1 - 1 \\ k_2 = 0, \ldots, N_2 - 1 \end{cases}$$

N_1 samples $\quad$ N_2 samples

DFT pair

$$X\left(k_1, k_2\right) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x\left(n_1, n_2\right) e^{-j\frac{2\pi}{N_1} n_1 k_1} e^{-j\frac{2\pi}{N_2} n_2 k_2} \qquad \begin{cases} k_1 = 0, \ldots, N_1 - 1 \\ k_2 = 0, \ldots, N_2 - 1 \end{cases}$$

$$n\left(n_1, n_2\right) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X\left(k_1, k_2\right) e^{j\frac{2\pi}{N_1} n_1 k_1} e^{j\frac{2\pi}{N_2} n_2 k_2} \qquad \begin{cases} n_1 = 0, \ldots, N_1 - 1 \\ n_2 = 0, \ldots, N_2 - 1 \end{cases}$$

Most properties of the continuous FT apply to the DFT with one exception: **linear shifts** become **circular shifts** (“wrap around”).

Fast Fourier Transforms (FFTs)

DFT: $$X\left(k_1, k_2\right) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x\left(n_1, n_2\right) e^{-j\frac{2\pi}{N_1}n_1k_1} e^{-j\frac{2\pi}{N_2}n_2k_2}$$

For each (k_1, k_2): $N_1 * N_2$ multiplications; $$\begin{cases} k_1 = 0, \ldots, N_1 - 1 \\ k_2 = 0, \ldots, N_2 - 1 \end{cases}$$

For $N_1 = N_2 = N$, a full DFT requires $\mathbf{N^4}$ multiplications.

Fast Fourier Transforms (**FFTs**) are a family of algorithms that impressively speed up calculation of the DFT.

Best runtime is in the order:

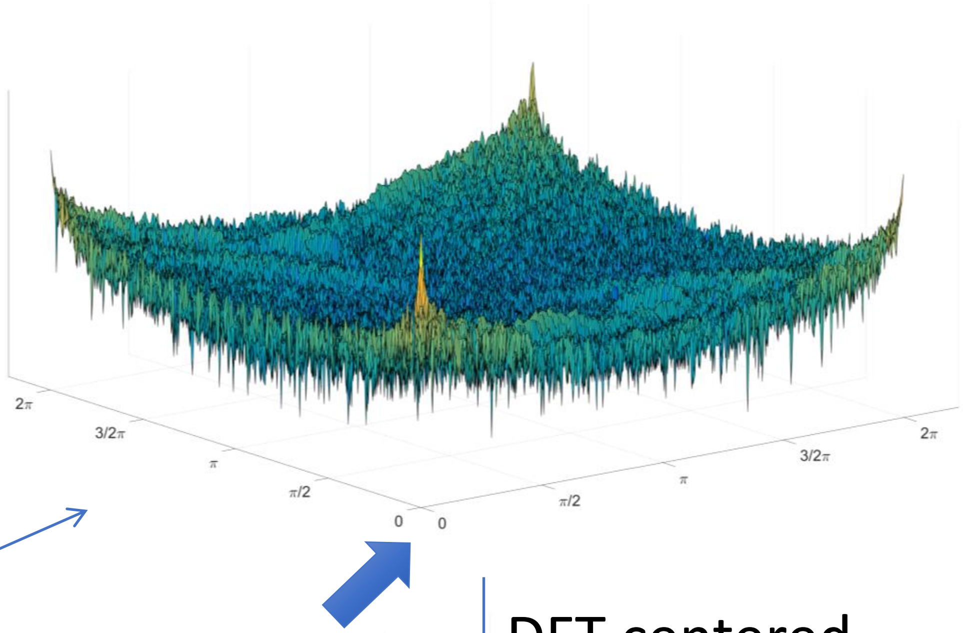
$$aN^2\log_2 N$$

with a < 1.

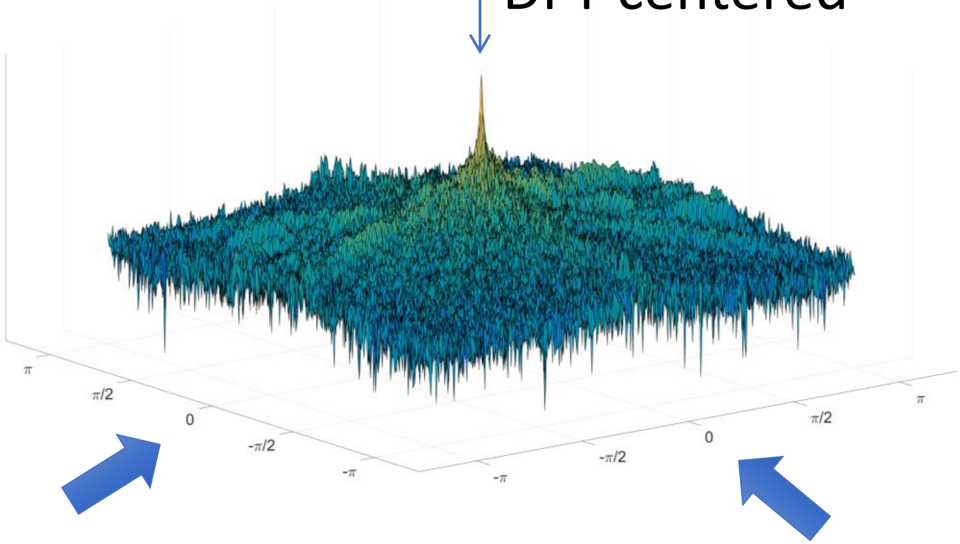
For a 1024 x 1024 image, the FFT is approximately 10^5 times faster that the DFT.

DFT centered

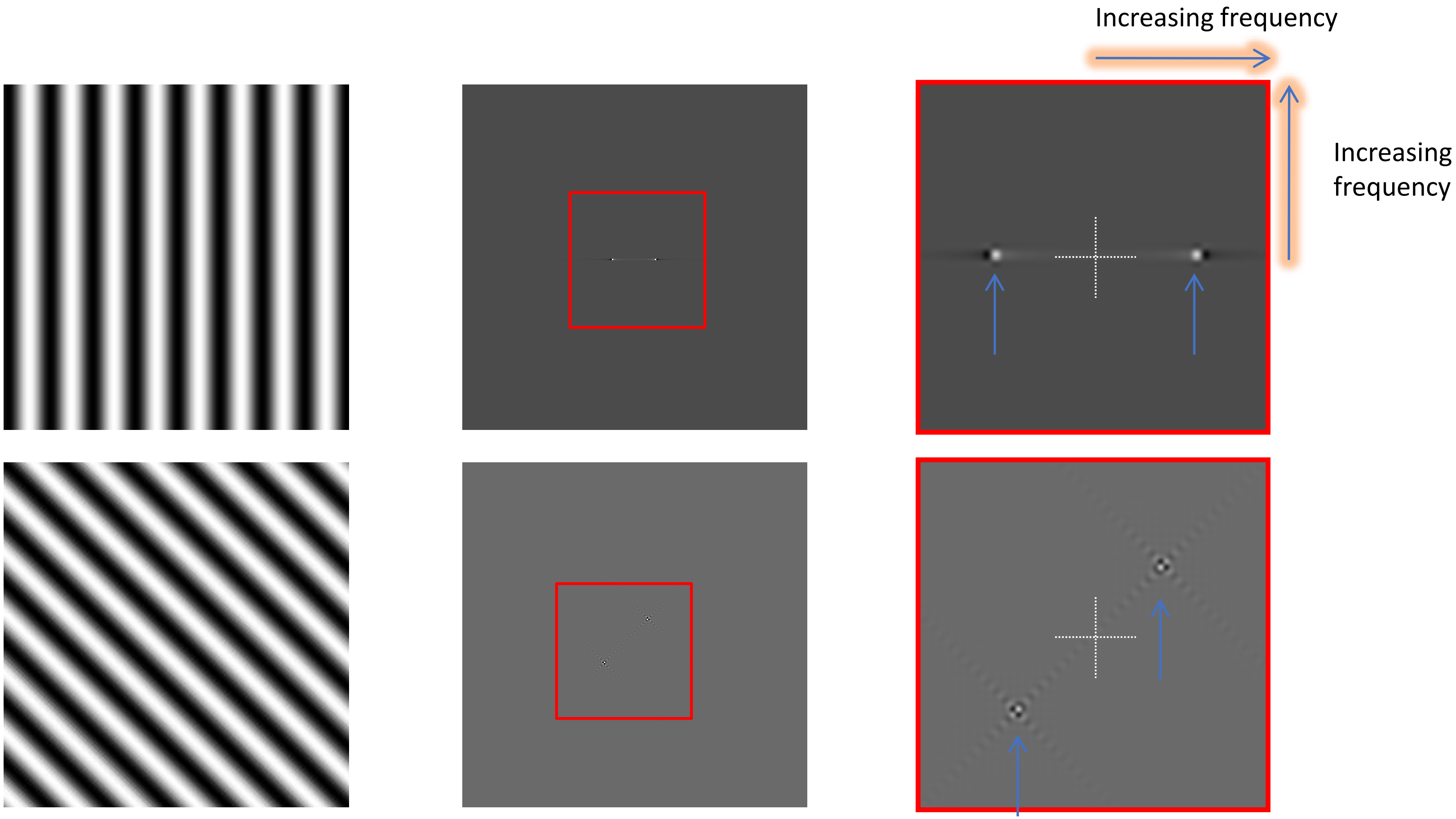
DFT



DFT centered



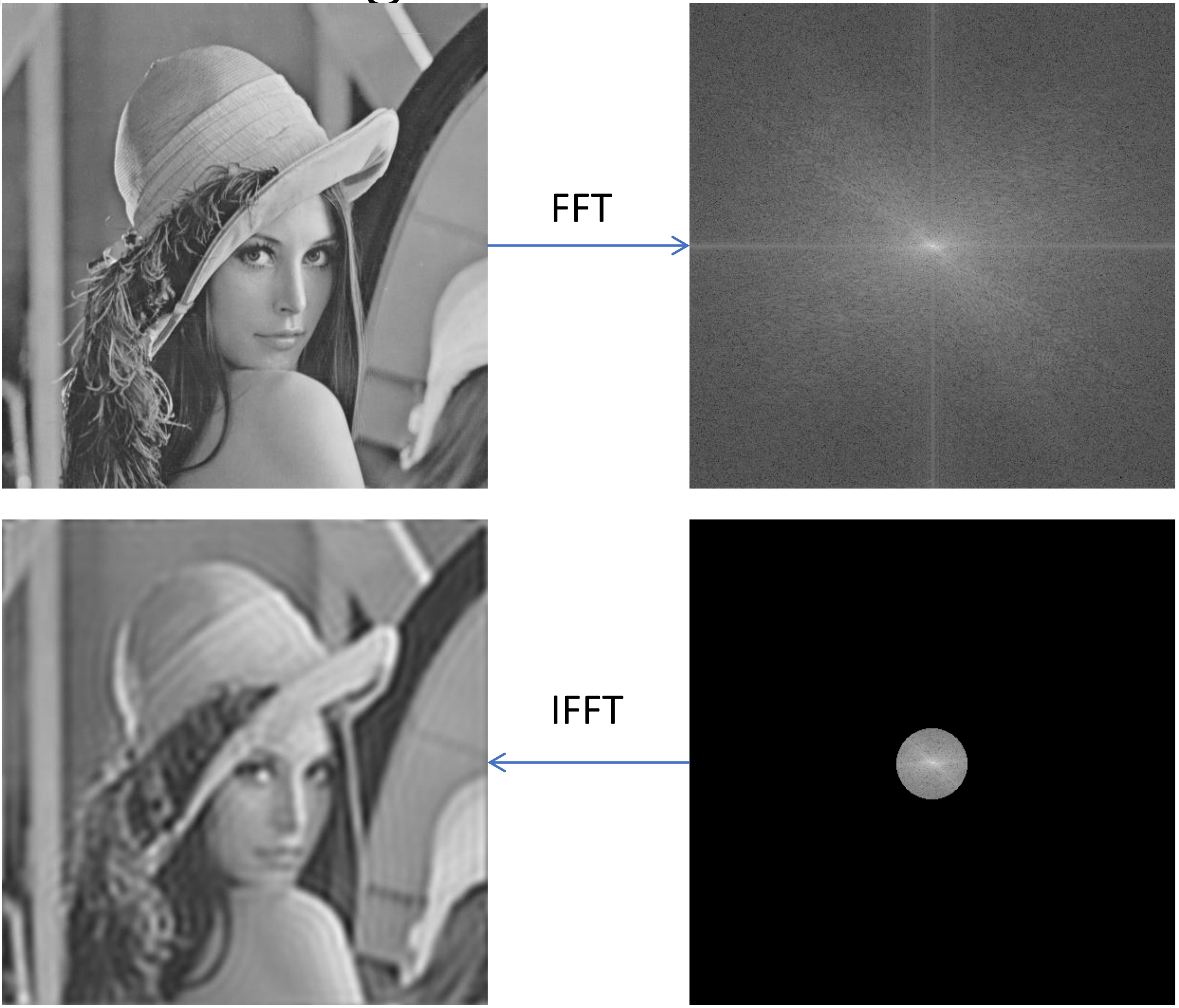
DFT examples



Increasing frequency

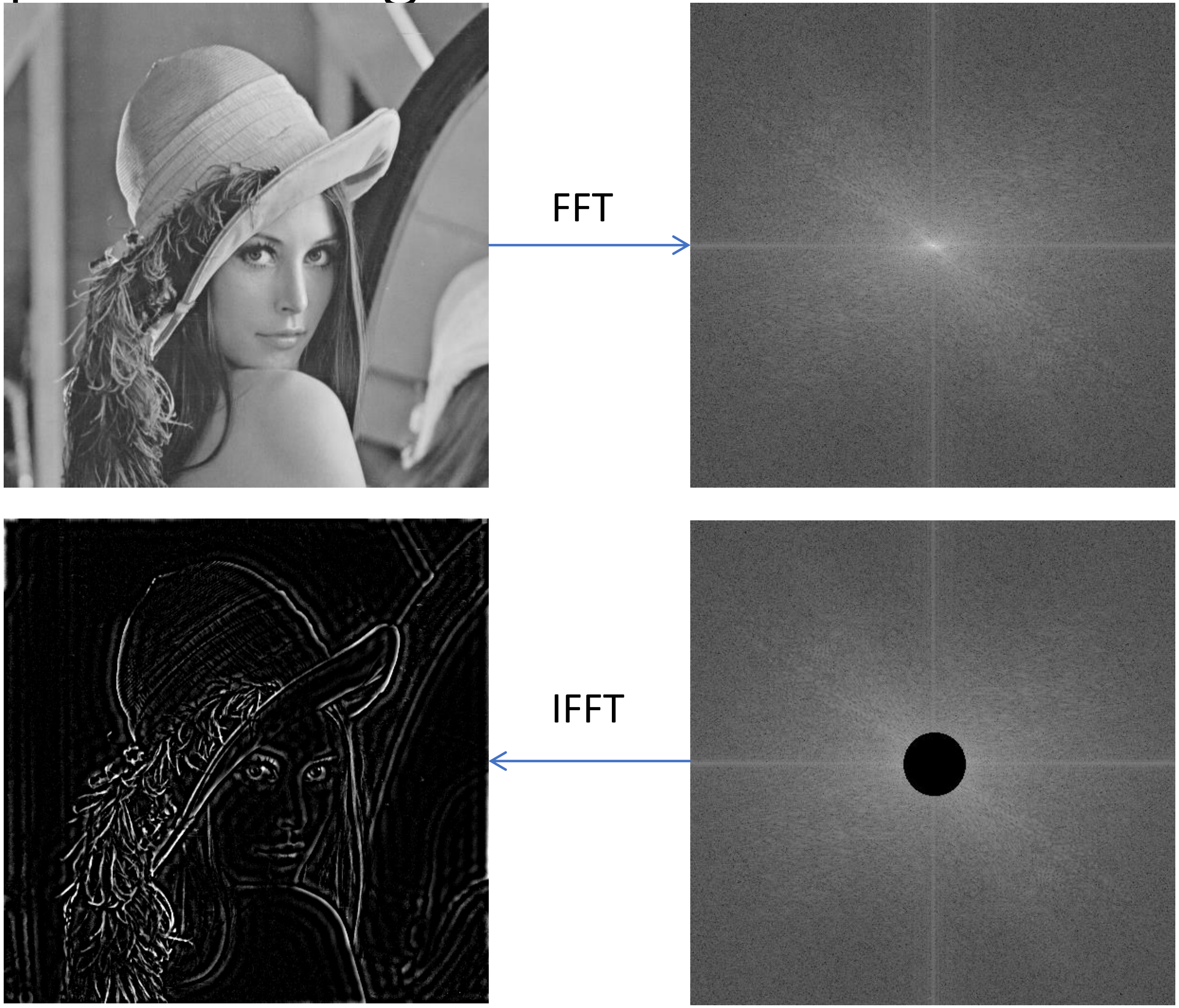
Increasing frequency

Low-pass filtering



In practice, one does not create sharp cut-offs in frequency domain, since this creates **ringing artifacts** that appear as spurious signals near sharp transitions in a signal, i.e. they appear as "rings" near edges.

High-pass filtering

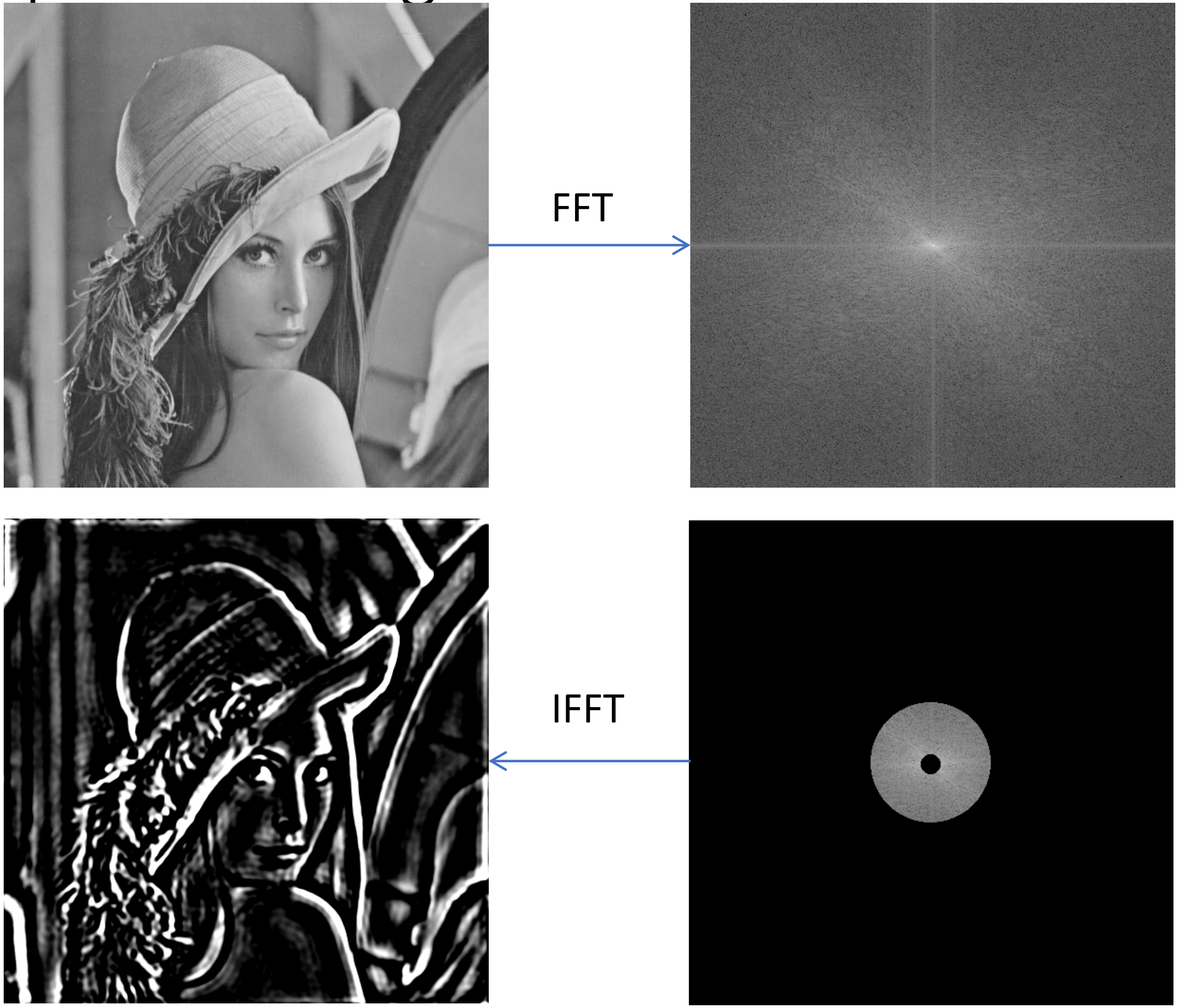


FFT

IFFT

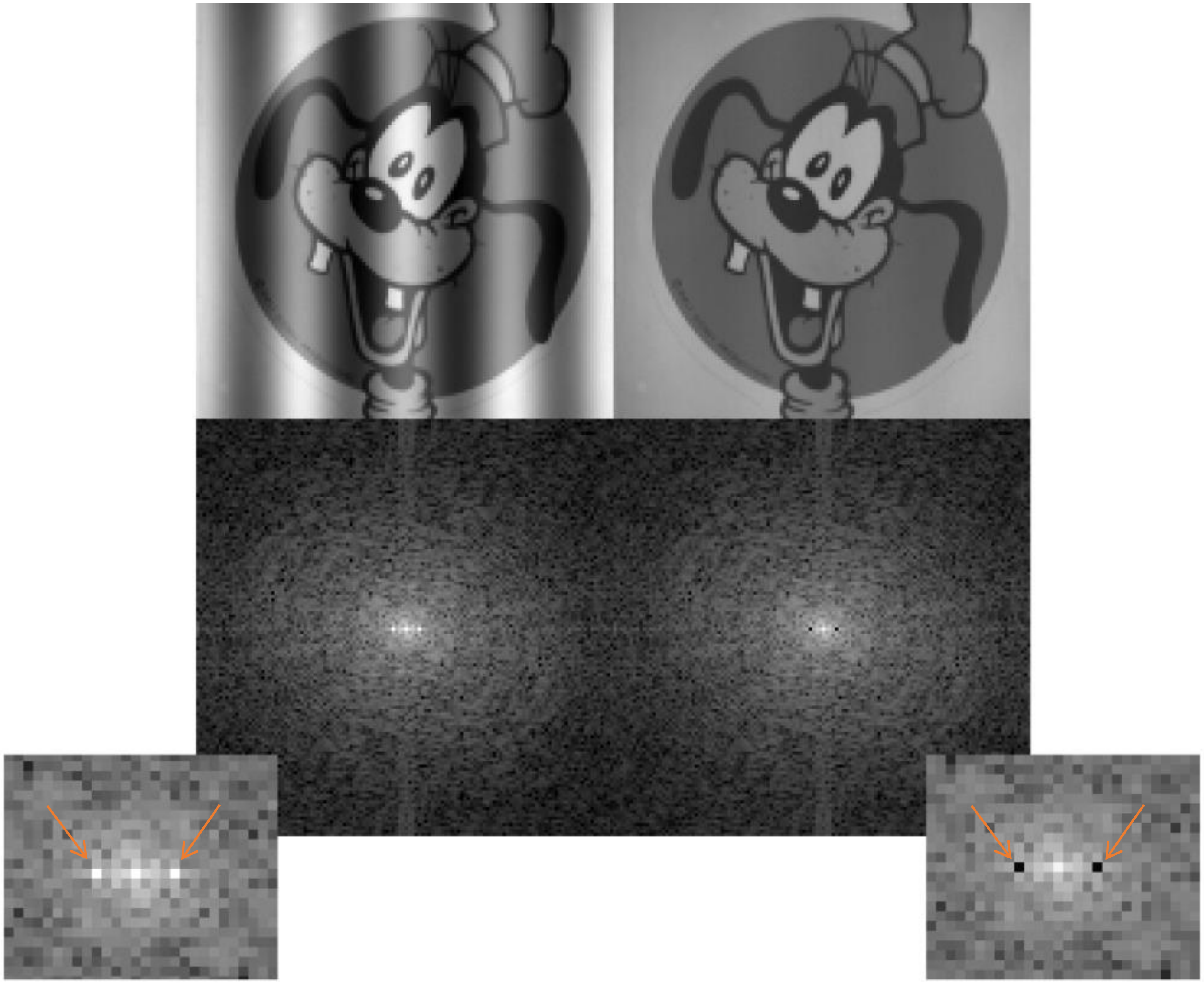
In practice, one does not create sharp cut-offs in frequency domain, since this creates **ringing artifacts** that appear as spurious signals near sharp transitions in a signal, i.e. they appear as "rings" near edges.

Band-pass filtering



In practice, one does not create sharp cut-offs in frequency domain, since this creates **ringing artifacts** that appear as spurious signals near sharp transitions in a signal, i.e. they appear as "rings" near edges.

Removing unwanted frequencies



Circular convolution

linear convolution

$$y\left(n_1, n_2\right) = x\left(n_1, n_2\right) \circledast h\left(n_1, n_2\right)$$

DFT

DFT

$$Y\left(k_1, k_2\right) = X\left(k_1, k_2\right) \cdot H\left(k_1, k_2\right)$$

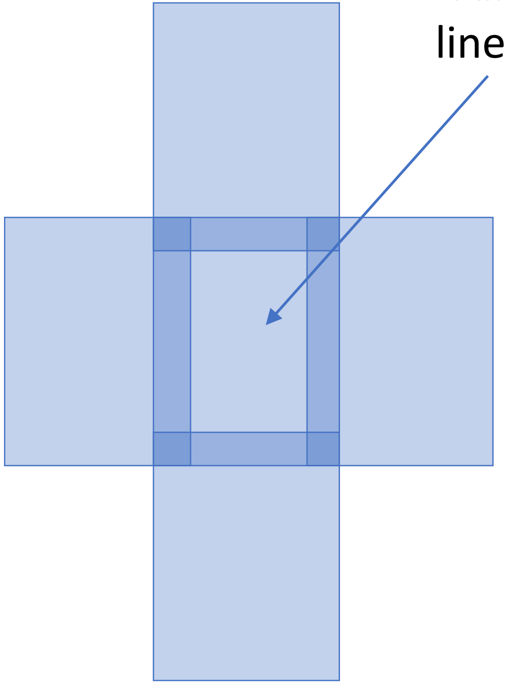
IDFT

circular convolution

$$y\left(n_1, n_2\right)$$

The circular convolution is infinite-length and periodic whereas the linear convolution is finite length, therefore a trick is needed to calculate linear convolution in frequency domain.

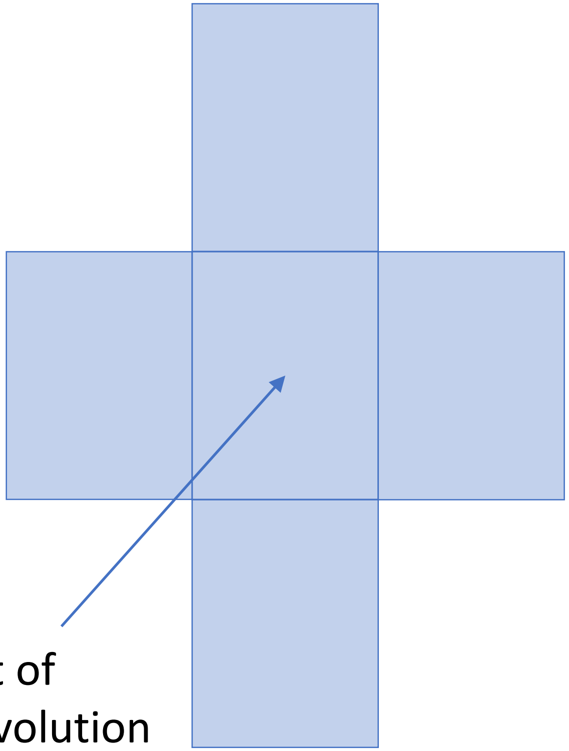
Circular convolution



Aliased result of linear convolution

Inappropriate support → aliasing

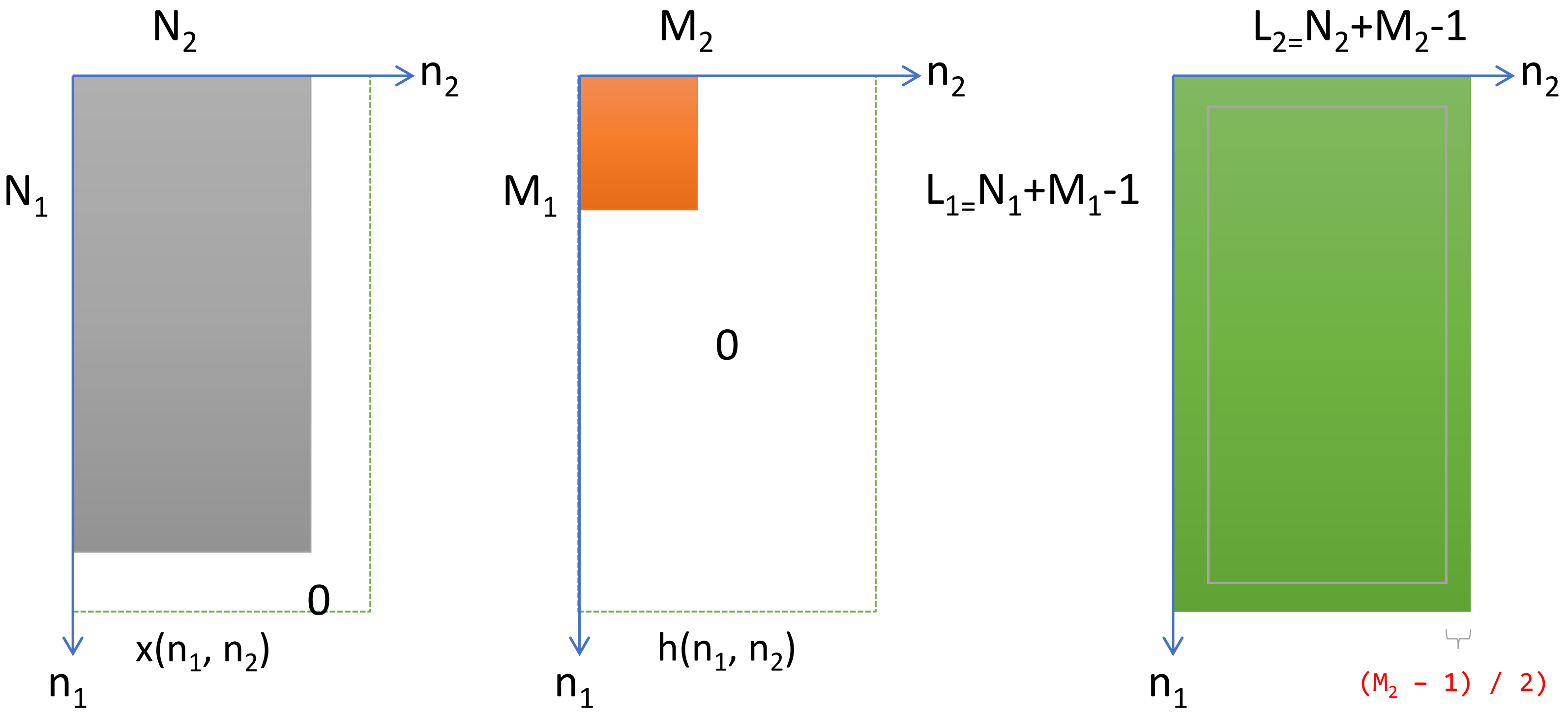
vs.



Result of linear convolution

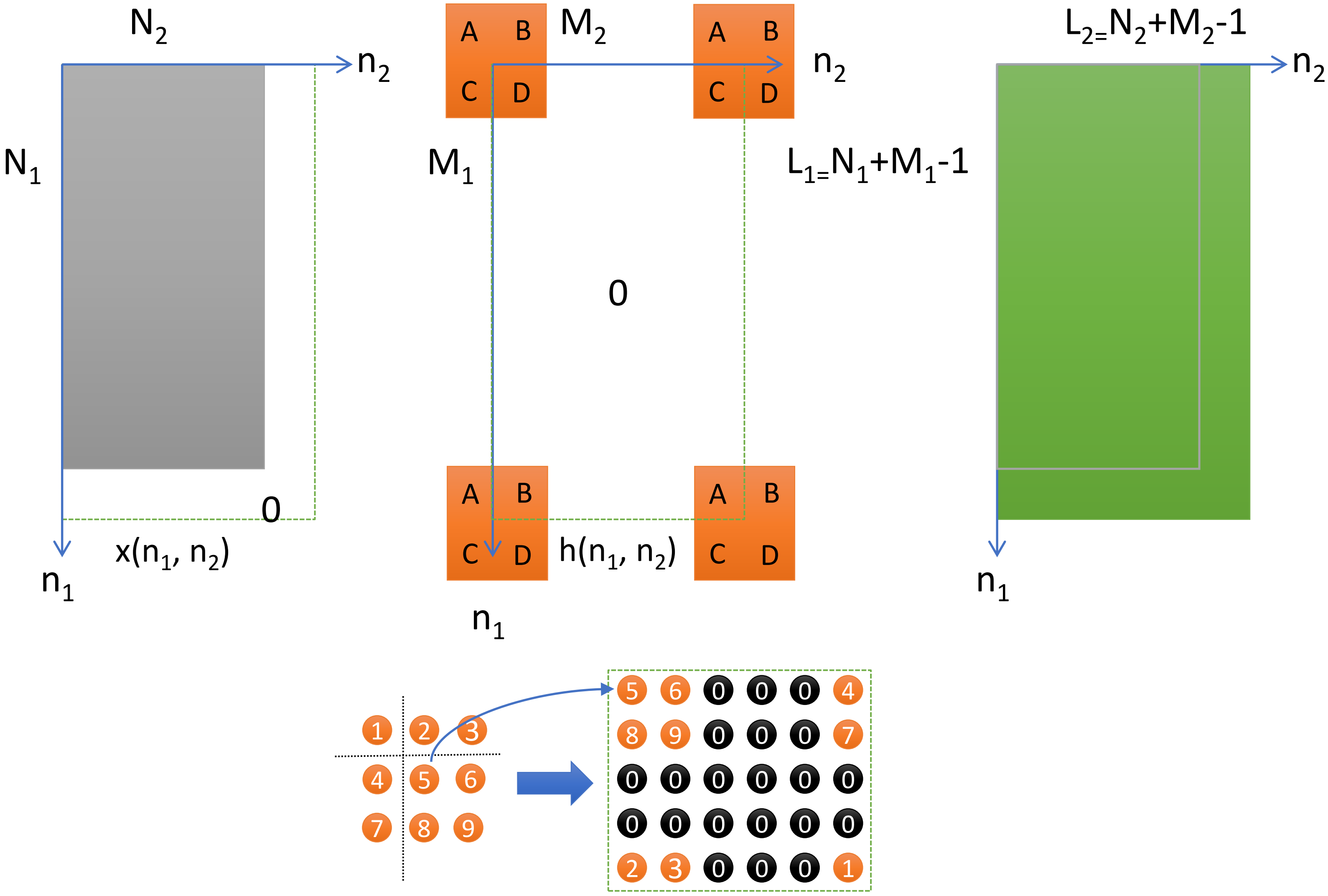
Appropriate support

Linear convolution in frequency domain (how to)



- Pad $x(n_1, n_2)$ and $h(n_1, n_2)$ with zeros to size (N_1+M_1-1 x N_2+M_2-1)
- Calculate the FFT (DFT) of both
- Multiply the transforms together
- Calculate the inverse FFT of the result → same result as linear convolution
- Carve out the result from the center of the result

Linear convolution in frequency domain (how to)



1	2	3
4	5	6
7	8	9

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

Linear convolution in frequency domain (how to)

Convolution in spatial domain

```
>> x = [1 2 3 4 5 4 3 2 1];    % N = 9
>> h = [1 2 3 2 1];            % M = 5
>> y = conv(x, h)
y =
     1     4    10    18    27    34    37    34    27    18    10     4     1   % N + M – 1 = 13

>> y = conv(x, h, 'same')
y =
    10    18    27    34    37    34    27    18    10   % N (the borders have size = (M – 1) / 2)
```

Convolution in Frequency domain with inappropriate support → aliasing

```
>> ifft(fft([ 1 2 3 4 5 4 3 2 1]) .* fft([1 2 3 2 1 0 0 0 0])) % They need to have at least the same size
y =
    19    14    14    19    27    34    37    34    27
```

Convolution in Frequency domain with appropriate support

```
% y = ifft(fft(x, numel(x) + numel(h) - 1) .* fft(h, numel(x) + numel(h) - 1))
% y = cconv(x, h)

>> ifft(fft([ 1 2 3 4 5 4 3 2 1 0 0 0 0]) .* fft([1 2 3 2 1 0 0 0 0 0 0 0 0]))
y =
    1.00   4.00  10.00  18.00  27.00  34.00  37.00  34.00  27.00  18.00  10.00   4.00   1.00

>> ifft(fft([ 1 2 3 4 5 4 3 2 1 0 0 0 0]) .* fft([3 2 1 0 0 0 0 0 0 0 0 1 2]))  3 4 5 0 0 0 1 2
y =
   10.00  18.00  27.00  34.00  37.00  34.00  27.00  18.00  10.00   4.00   1.00 1.00   4.00
```

The “border” is all on one side.