Spatial filtering

Outline

Spatial filtering

- Filtering idea
- *Linear* vs *non-linear* filters
- *Correlation* and *convolution*
- *Examples*: blurring, sharpening, denoising

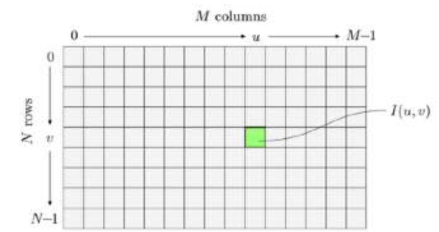
Filtering in Fourier space

- *Spatial* domain and *frequency* domain
- *Image details* and frequencies
- Why it's a good idea
- *Examples*: low- and high-pass filters

Recall…

- Limitations of **point operations**
  - They <u>know</u> only the *current pixel intensity*
  - They <u>don't know</u> anything about their *neighbors*
  - They <u>don't know</u> *where they are* in an image

$$J(u,v) \mapsto f\left[I(u,v)\right]$$



- Most image features (e.g. edges) require information from a **spatial neighborhood** of pixels

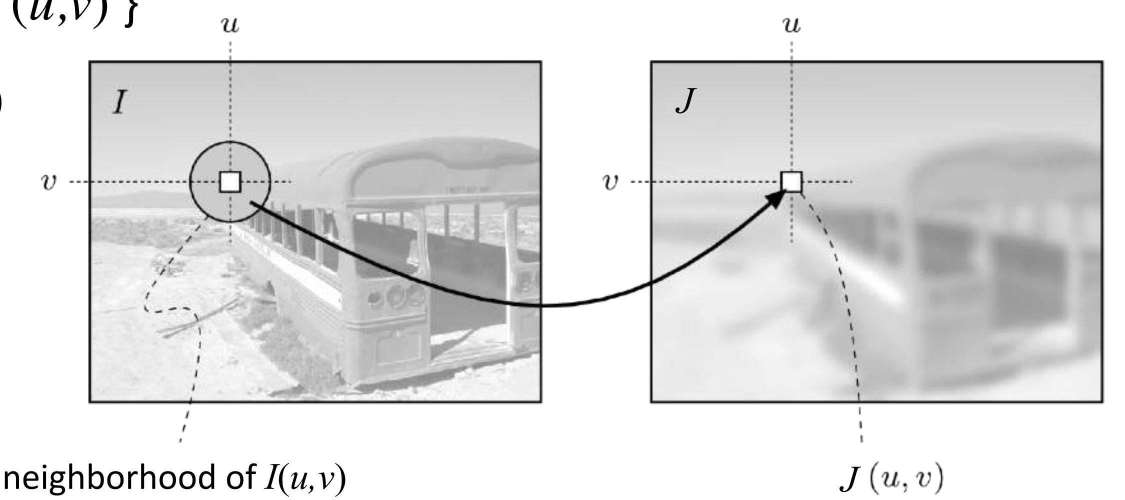
Requires derivatives (i.e. spatial information)

- To enhance such features, need to **go beyond point operations**

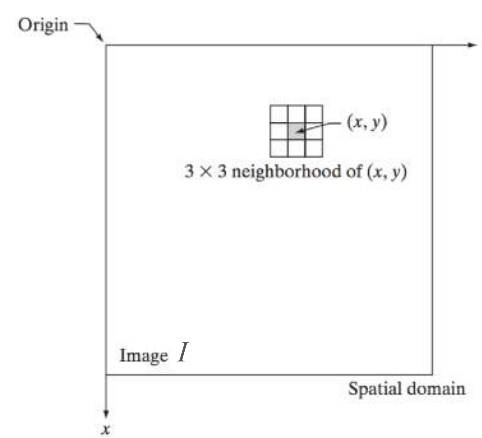
- **Spatial filtering** is an image operation where a pixel $I(u, v)$ is changed by a function of the *pixels in a neighborhood* of (u, v)

$$J(u,v) \mapsto f\left[\mathcal{N}\Big(I(u,v)\Big)\right]$$

  - $\mathcal{N}(\ I(u,v)\ )$ = { pixels “close” to pixel (u,v) }
  - f <u>depends on all pixels in</u> $\mathcal{N}(\ I(u,v)\ )$

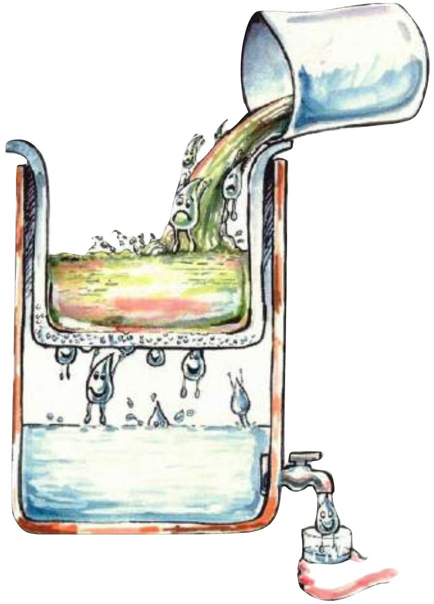


- **Neighborhood** of pixel (u,v)
  - Also called *filter*, *mask*, *kernel*, *template*, *window*
  - Usually defined as *squares* 3×3, 5×5, 7×7…
  - **Q**: what is a 1×1 neighborhood?



Definition of "spatial filtering" (2/3)

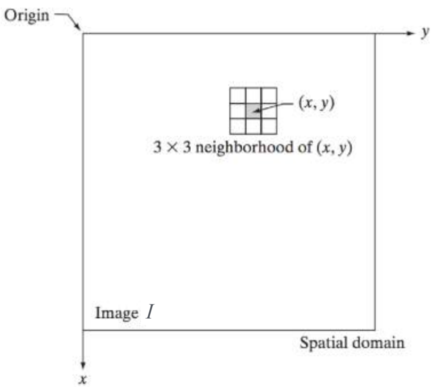
- Note 1: point operations are a **subset of spatial filters**
  - *Point operations* ➜ use only the intensity of pixel (u,v)
  - *Spatial filtering* ➜ use information from all pixels in the neighborhood of (u,v)
  - NB: called **neighborhood processing** or **spatial convolution** to be more explicit
    - Soon we will see what "*convolution*" means exactly
- Note 2: "*spatial*" refers to the **spatial domain**, i.e. image plane itself
  - Spatial filtering means "**direct manipulation of pixel intensities of an image**"
    - Soon we will see what "*direct*" means exactly
- Note 3: "*filtering*" refers to the **analogy** with classic filters
  - A classic *filter* can remove impurities from water
  - An *image processing filters* **remove undesired features** from an image (e.g. noise)



Definition of "spatial filtering" (3/3)

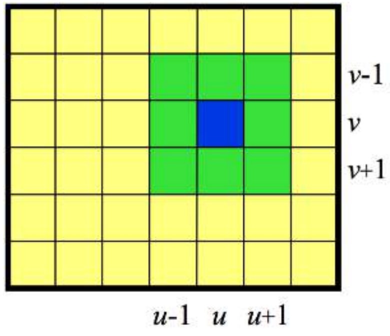
Pseudocode

- **Inputs**: *image* $I(u, v)$ defined on $[0 \ldots M\text{-}1] \times [0 \ldots N\text{-}1]$
  neighborhood $\mathcal{N}$ defined on $[0 \ldots K\text{-}1] \times [0 \ldots K\text{-}1]$, where $K \in \{3,5,7 \ldots\}$
- **Output**: new image $J(u, v)$
- *for* $v = 0 \ldots N\text{-}1$
  for $u = 0 \ldots M\text{-}1$
  set $J(u, v) = f[\, \mathcal{N}(\, I(u, v)\,)\,]$



Example: computing the mean in a 3 × 3 neighborhood

- The **blue pixel** is the one to be modified
- The **green pixels** represent the *neighborhood* $\mathcal{N}$
  - NB: a *point operation* could use information only from the blue pixel
- The function f is defined as: $\frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1} I(u+i, v+j)$
- The operation is *repeated* for all **yellow pixels**



Linear vs non-linear filters

- Two categories depending on how *function* f is implemented
  - If f can be implemented as a *linear operation of pixels* in the neighborhood
    ➜ **linear spatial filtering**
  - Otherwise, if the computation involves non-linear operations
    ➜ **non-linear spatial filtering**
- **Examples** of linear and non-linear functions
  - *Multiplication* and *sum* of voxels in the neighborhood ➜ *linear*
  - *Mean* of voxels in the neighborhood ➜ *linear*
  - *Median* of voxels in the neighborhood ➜ *non-linear*
  - *Max* of voxels in the neighborhood ➜ *non-linear*
- Linear filters have a **number of advantages**
  - We will see soon which and why
  - In this course, <u>we will focus on linear filters</u>

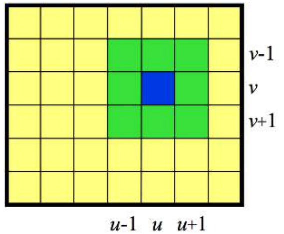
General (linear) filter equation

Filtering as **correlation**

$$I'(u,v) = \sum_{(i,j)\in\mathcal{N}} I(u+i, v+j) \cdot H(i,j)$$

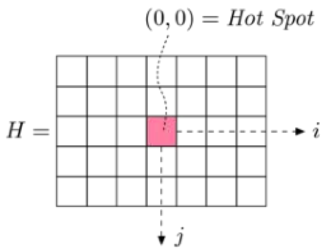
This operation is known as **correlation** of I and H
$(I \otimes H)$

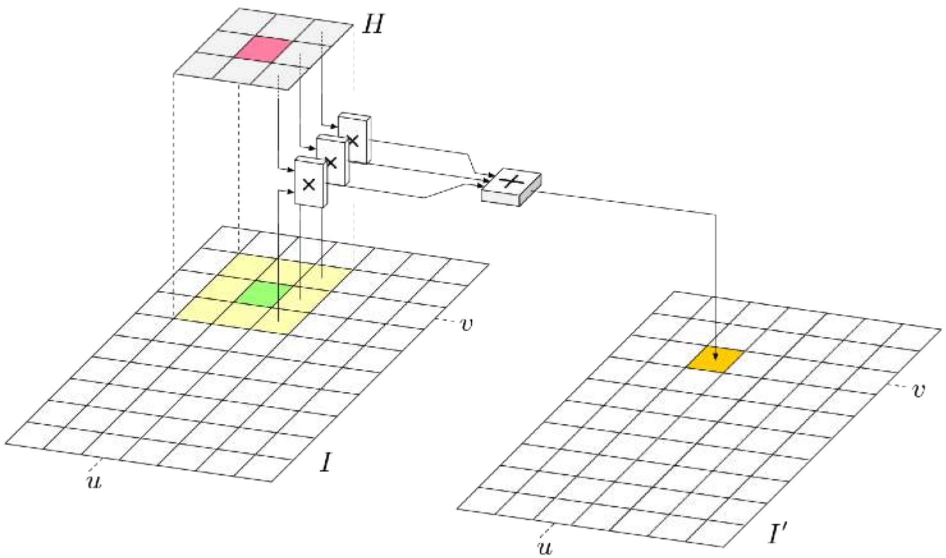
- I and I' are, respectively, the input and output images
- $\mathcal{N}$ describes the *spatial neighborhood* of a voxel
- $H(i, j) : \mathcal{N} \rightarrow [0, K-1]$ is the filter (a.k.a. *kernel* or *mask*)



In other words

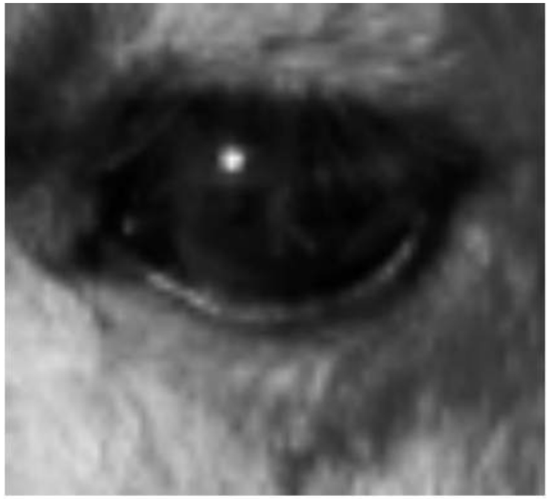
- A **small 2D matrix** moves across the image affecting *only one pixel at a time*
- The *coefficients in H* **determine the effect** on the output image



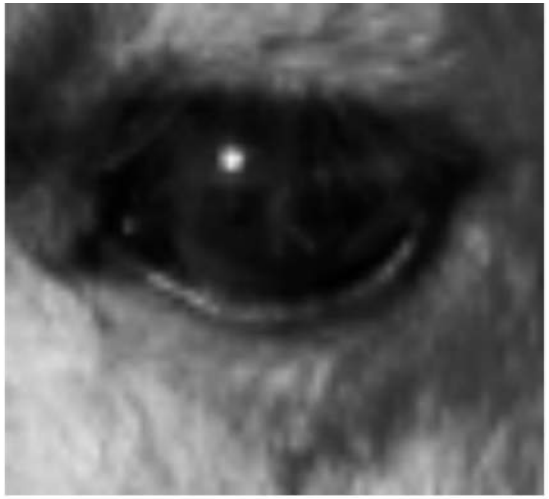


What do these filters do? (1/2)

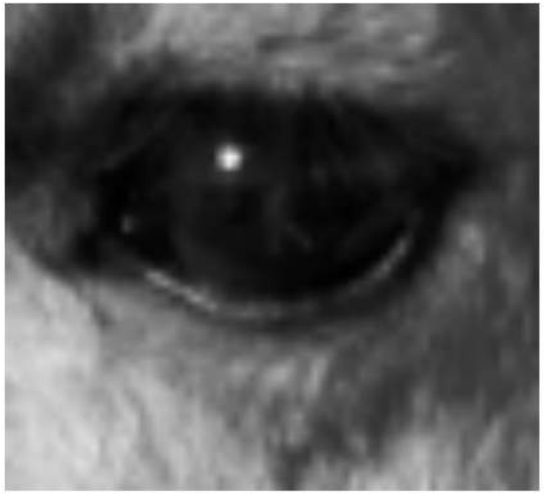
- Filter 1 ➜ **nothing!**



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



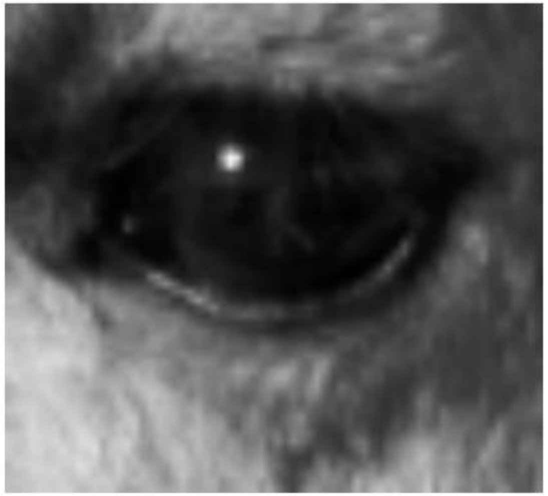
- Filter 2 ➜ **blurring** (mean)



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

What do these filters do? (2/2)

- Filter 3 ➜ **sharpening** (opposite to mean filter)



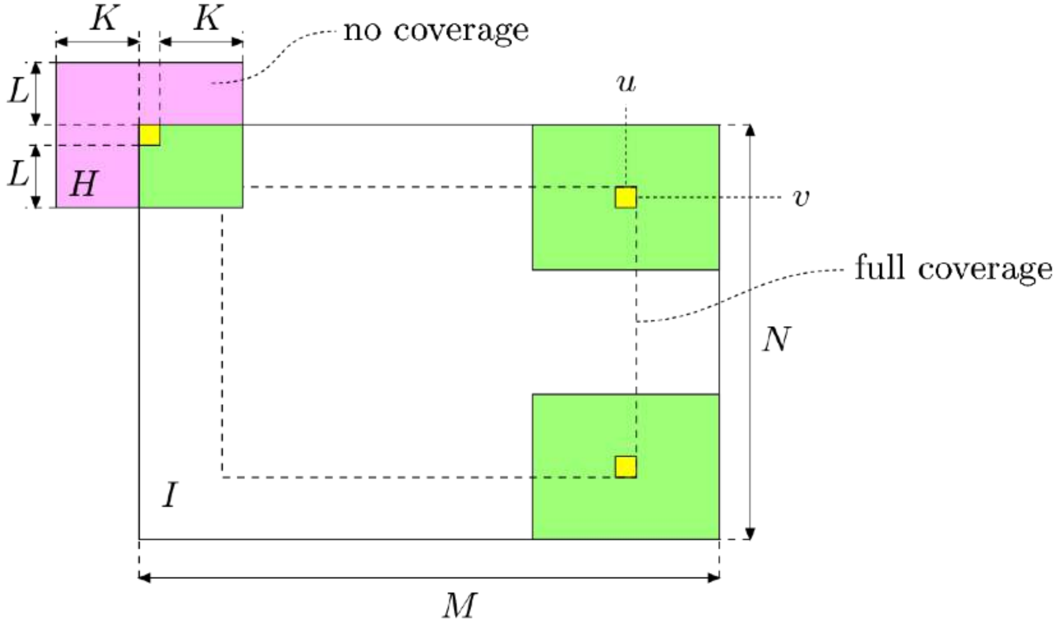
$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- Filter 4 ➜ **shift left by one pixel**

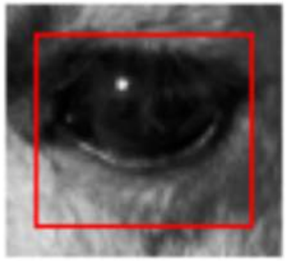
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Notes: what to do at the boundary?

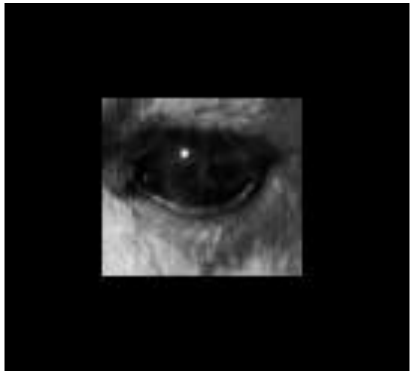
- The **neighborhood is not available** at the boundary



- Some options



crop



pad

extend

wrap

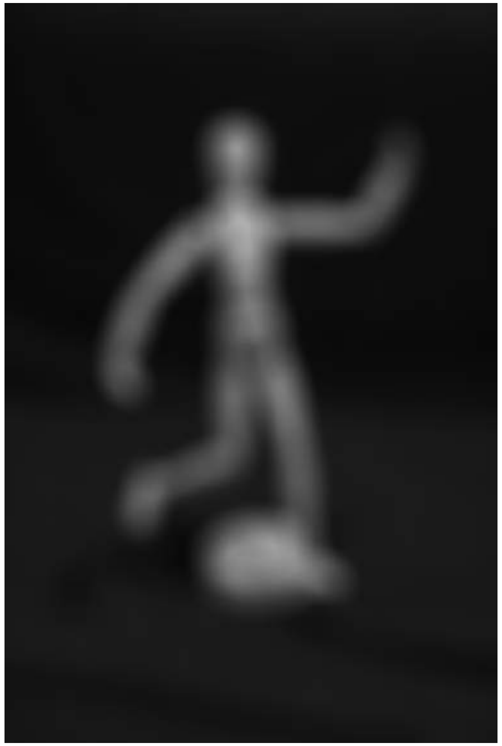
Notes: effect of filter size

- **Example**: mean filters
  (similar effects with any filter)

original

7 × 7

15 × 15



41 × 41

Notes: filter normalization

- Multiplying all entries in H by a constant would cause the **image** to be **multiplied** by that constant

$$\begin{aligned} I'(u,v) &= \sum_{i,j} I(u+i,v+j)\cdot(cH(i,j)) \\ &= c\sum_{i,j} I(u+i,v+j)\cdot H(i,j) \end{aligned}$$

- Hence, to **keep the overall brightness constant** we need H to sum up to one

- **Note**: all *previous filter examples* indeed sum up to one!

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} \quad \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \quad \frac{1}{9}\begin{bmatrix}-1&-1&-1\\-1&17&-1\\-1&-1&-1\end{bmatrix} \quad \begin{bmatrix}0&0&0\\0&0&1\\0&0&0\end{bmatrix}$$

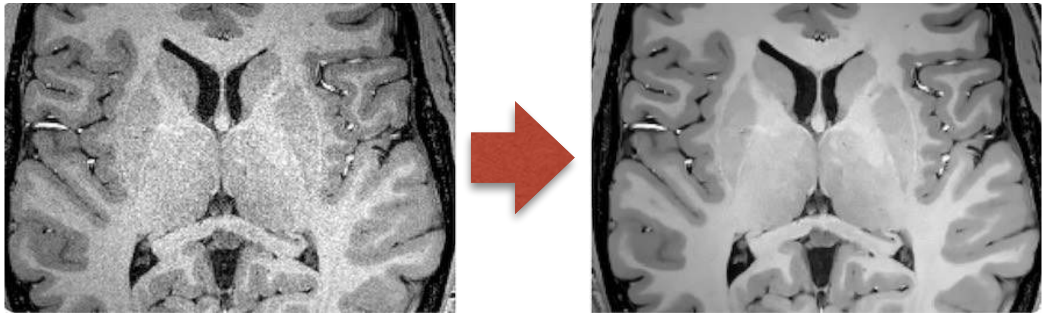
Application: noise reduction (1/2)

- Image pixels are the result of a **signal intensity measurement**
  - All recording devices are susceptible to *noise*
  - Noise causes *fluctuations* in actual signal intensities
  - Noise properties depend on acquisition equipment

- "Additive" noise model

$$\hat{s} = s + \epsilon$$

  - $\hat{s}$ is observed signal intensity
  - s is the true value
  - ϵ is the noise value



- Noise reduction is the process of **removing noise from a signal** (i.e. all pixels in the image)

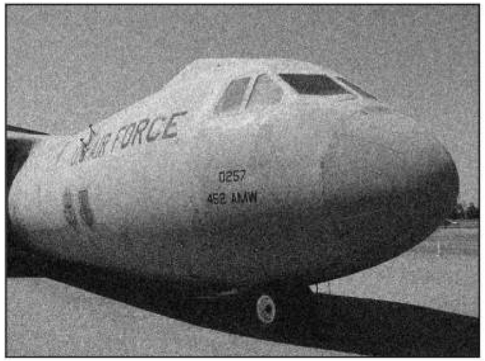
Application: noise reduction (2/2)

- **Typical noise**: Additive White Gaussian Noise (AWGN)
  - *Many random processes* that occur in nature follow this model
  - NB: white noise has *zero mean*

- **IDEA**: if we can average several pixels in a neighborhood with the same signal, the noise will "average out"

$$E[\hat{s}] = E[s + \epsilon] = E[s] + E[\epsilon] = s$$

- Any filter that *averages in a neighborhood* will reduce noise



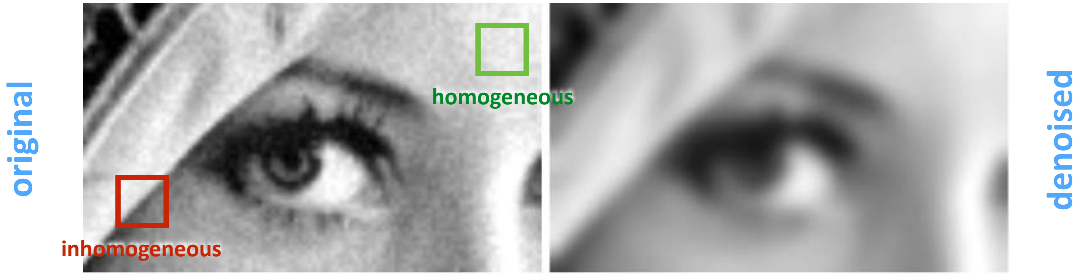
$$\otimes \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad =$$

  - These filters are also called *average filters*

Non-local means filter

Simple averaging may **remove important details**

- **Example**: pixels close to *edges/borders* are not constant in a neighborhood



- Pixels belong to distinct classes with **different intensities**
- Replaced by their **average value** ➜ edge/border is lost

More **advanced filters** exist

- Use *prior information* about image, e.g. image is piecewise smooth
- Here we'll see one basic example: **N**on-**L**ocal **M**eans (NLM)

<u>NLM idea</u>: exploit **redundancy** in images

- Squares with *same color* are on **similar areas** of the image
- Why not using all this information to infer values of pixels?

Non-local means filter (2/3)

Algorithm (naive implementation)

- Loop over each pixel (u,v)
- **Compare the neighborhood** of (u,v)
  to the neighborhood of all other pixels (i,j)
- Compute **similarity** between each pair of neighborhoods

$$d_{\mathcal{N}} = \exp\left(-\frac{\left|\left|\mathcal{N}\Big(I(u,v)\Big) - \mathcal{N}\Big(I(i,j)\Big)\right|\right|^2}{h^2}\right)$$

Gaussian weighting function
(h controls the degree of similarity)

- Set the new value of pixel (u,v) as a **weighted average** of all other pixels

$$I'(u,v) = \frac{1}{Z}\sum_{(i,j)\in I} d_{\mathcal{N}}\Big(I(u,v), I(i,j)\Big)\cdot I(i,j)$$

Z is a normalization constant
i.e. Z = sum of all $d_{\mathcal{N}}$

Notes

- $d_{\mathcal{N}}$ determines **how closely related** the image in (u,v) is to the image in (i,j)
- Averaging is now performed using "**very similar pixels**"
  i.e. pixels with very similar neighborhoods

- Comparison

original

noise added

average filter

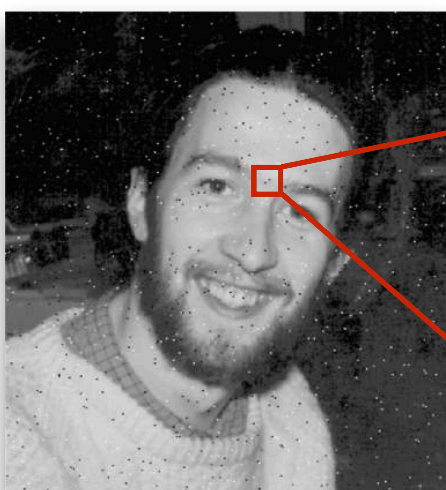
non-local means

Median filter (1/3)

- Particular noise sources require **specific denoising algorithms**
  - Example: ***Salt-and-Pepper*** noise
  - Caused by *sharp and sudden alterations* of the image signal
  - Manifests as sparsely occurring *white and black pixels*
  - Also known as *impulse noise*

- **Mean filter does not work** well because the *outliers* are included in the mean

original



240	245	0
247	0	244
251	246	250

mean([240,245,0,247,0,244,251,246,250]) = **191**

mean filter

Median filter (2/3)

- Taking the **median over the neighborhood** is more appropriate

240	245	0
247	0	244
251	246	250

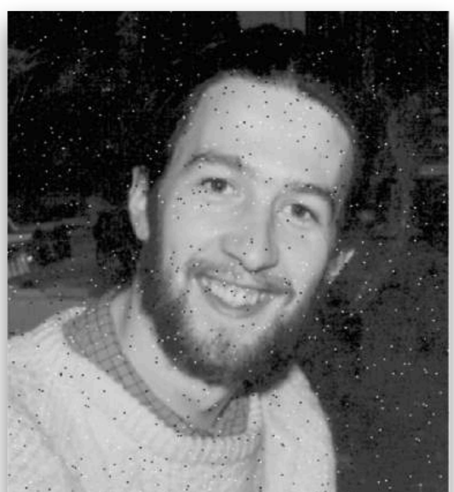
mean([240,245,0,247,0,244,251,246,250]) = **191**

sort the values

0	0	240	244	245	246	247	250	251

median([0,0,240,244,245,246,247,250,251]) = **245**

original



median filter

- NB: median is a **non-linear filter**

Median filter (3/3)

- Comparison

3×3 5×5 7×7

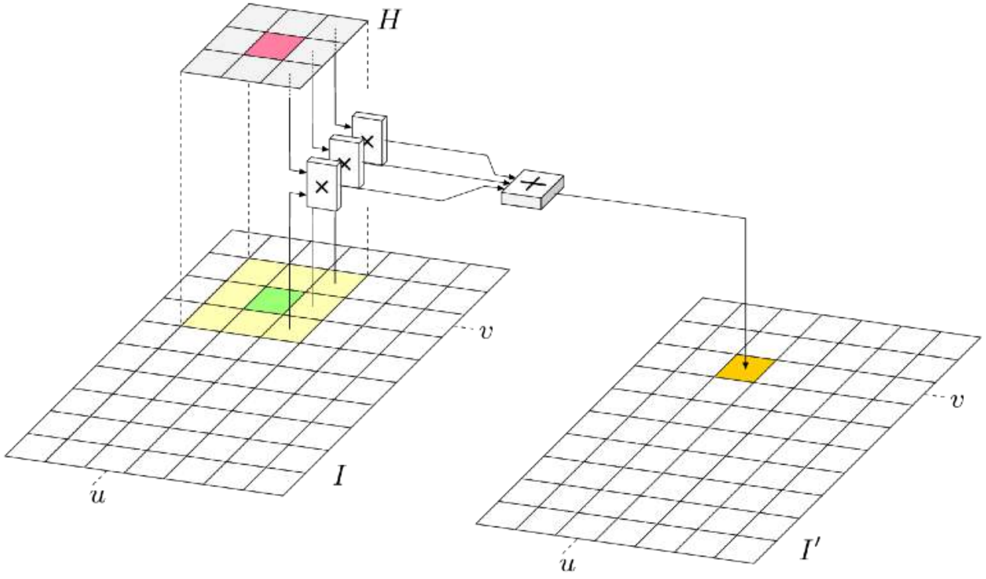
average filter

median filter

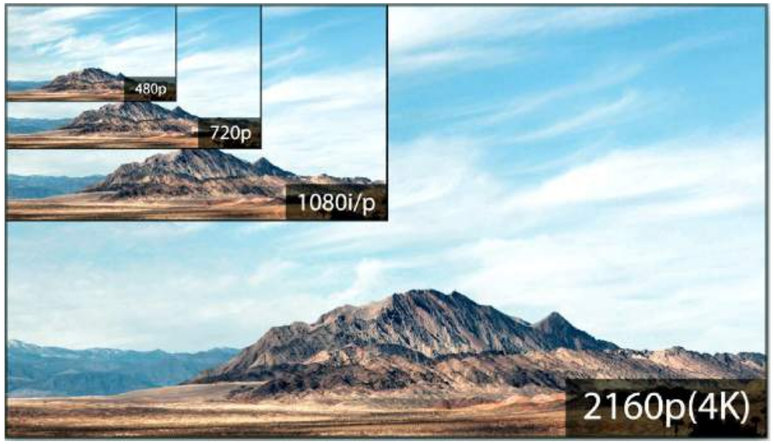
Filtering via correlation is slow!

Recall the procedure

- To filter a given pixel (u,v), **align the center of the filter** at (u,v)
- **Multiply** element-wise pixels in $\mathcal{N}$ with the corresponding pixels of H
- New pixel intensity = **sum** all these values
- **Repeat** for all pixels of the image



Imagine to process a 4k image
(i.e. 3840×2160, 8.3 megapixels)

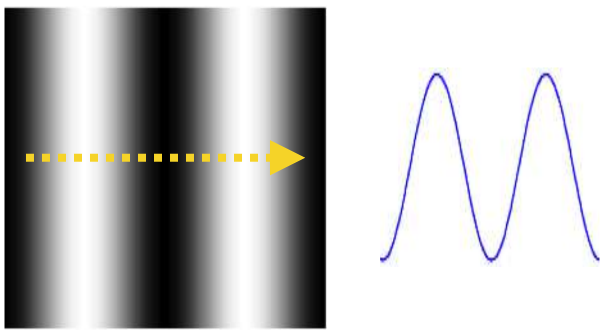


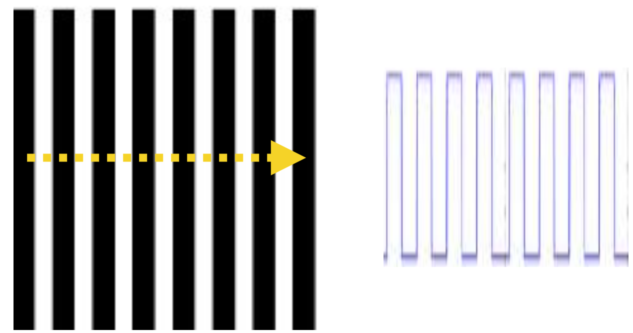
Luckily, there is a "trick"…thanks to **Fourier**!!!

Filtering in Fourier space

Image details and frequencies

- With pictures, the term "**frequency**" means the rate of change of intensity per pixel
  - Take a **line across an image** and plot the intensity values

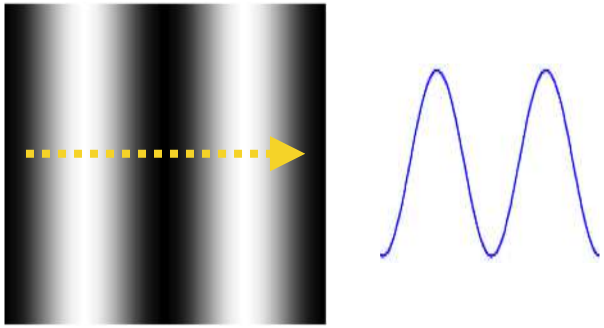


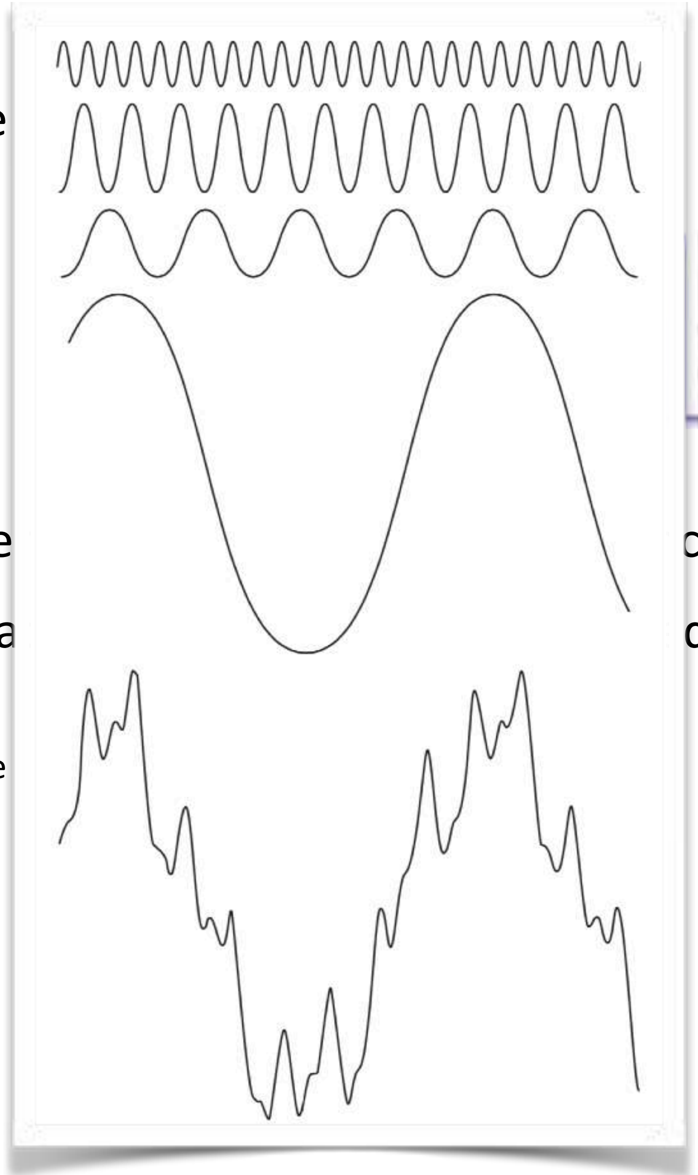


  - If it takes many pixels to undergo an intensity variation ➜ low frequency
  - The fewer pixels it takes to represent that change ➜ the higher the frequency
    - **edges** represent pretty high frequency features
    - **noise** is also a high-frequency component of the image

Image details and frequencies

- With pictures, the term "**frequency**" means the rate of change of intensity per pixel
  - Take a **line across an image** and plot the

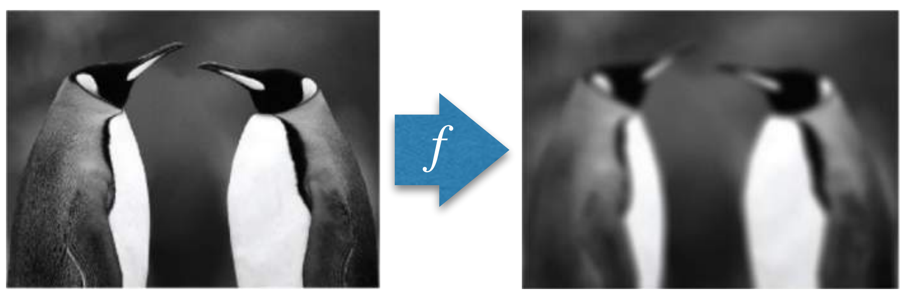




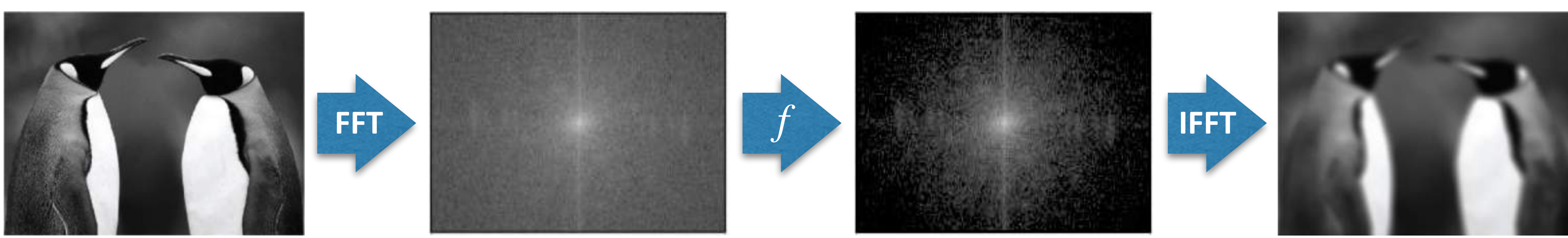
  - If it takes many pixels to undergo an inte… …cy
  - The fewer pixels it takes to represent tha… …quency
    - **edges** represent pretty high frequency features
    - **noise** is also a high-frequency component of the image
- In short
  - **Low frequencies** show you **structure**
  - **High frequencies** give you **detail**

Filtering in frequency domain

- **Spatial domain** refers to the *image plane* itself
- **Frequency domain** refers to the frequency components of the *Fourier transform* of the image
- **Filtering in the spatial domain**: changes <u>directly</u> the pixels



- **Filtering in the frequency domain**: changes <u>indirectly</u> the pixels by manipulating the Fourier transform of the image



- **Convolution** of an *image* I by a *kernel* H is given by

$$I'(u,v) = \sum_{(i,j)\in\mathcal{N}} I(u-i, v-j) \cdot H(i,j)$$

Convolution of I and H
$(I * H)$

  - $I(u, v)$ and $I'(u, v)$ are, respectively, the *input and output images*
  - $\mathcal{N}$ describes the *spatial neighborhood* of a voxel
  - $H(i, j) : \mathcal{N} \rightarrow [0, K-1]$ is the *filter*

- Notes
  - Recall the definition of **correlation**:

$$I'(u,v) = \sum_{(i,j)\in\mathcal{N}} I(u+i, v+j) \cdot H(i,j)$$

Correlation of I and H
$(I \otimes H)$

  - Similar to correlation, but with negative signs on the I indices
  - Equivalent to **vertical and horizontal flipping** of H

$$I'(u,v) = \sum_{(-i,-j)\in\mathcal{N}} I(u+i, v+j) \cdot H(-i,-j)$$

Convolution (2/3)

Why then another definition?

- **Correlation** was easier to explain for introducing spatial filters
- **Convolution** has more useful properties

Basic properties of convolution

- *Linear* $(a \cdot I) * H = a \cdot (I * H)$ $(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$
- *Commutative* $I * H = H * I$
- *Associative* $(I * H_1) * H_2 = I * (H_1 * H_2)$

Convolution theorem

- Let f, g be two functions with *convolution* $f*g$ and *Fourier transforms* $F\{f\}$ and $F\{g\}$. Then:

$$F\{f * g\} = F\{f\} \cdot F\{g\}$$

$$F\{f \cdot g\} = F\{f\} * F\{g\}$$

- *In other words*, convolution in a domain equals point-wise multiplication in the other

Convolution (3/3)

Filtering is **simpler in Fourier space!**

- In <u>spatial domain</u>, $I*H$ is performed by sliding H on the image I (*very slow*)
- In <u>frequency domain</u>, the operation is *replaced by a simple multiplication*

$$\mathrm{F}\{ I * H \} = \mathrm{F}\{ I \} \cdot \mathrm{F}\{ H \} \quad \Rightarrow \quad I * H = \mathrm{F}^{-1}\{ \mathrm{F}\{ I \} \cdot \mathrm{F}\{ H \} \}$$

- **NB:** *FFT* is efficient, *multiplication* is efficient ➜ **filtering in Fourier space is fast**

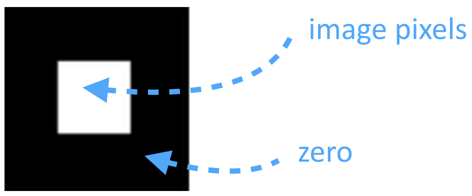
To make the theory work out, we **need a mathematical trick**

- Let's define our *image* and *kernel* **domains to be infinite**: $\Omega = \mathbb{Z} \times \mathbb{Z}$
- Now convolution is an **infinite sum**:

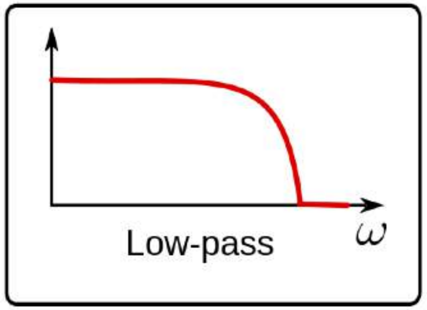
$$I'(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i, v-j) \cdot H(i,j)$$

New definition of $I*H$

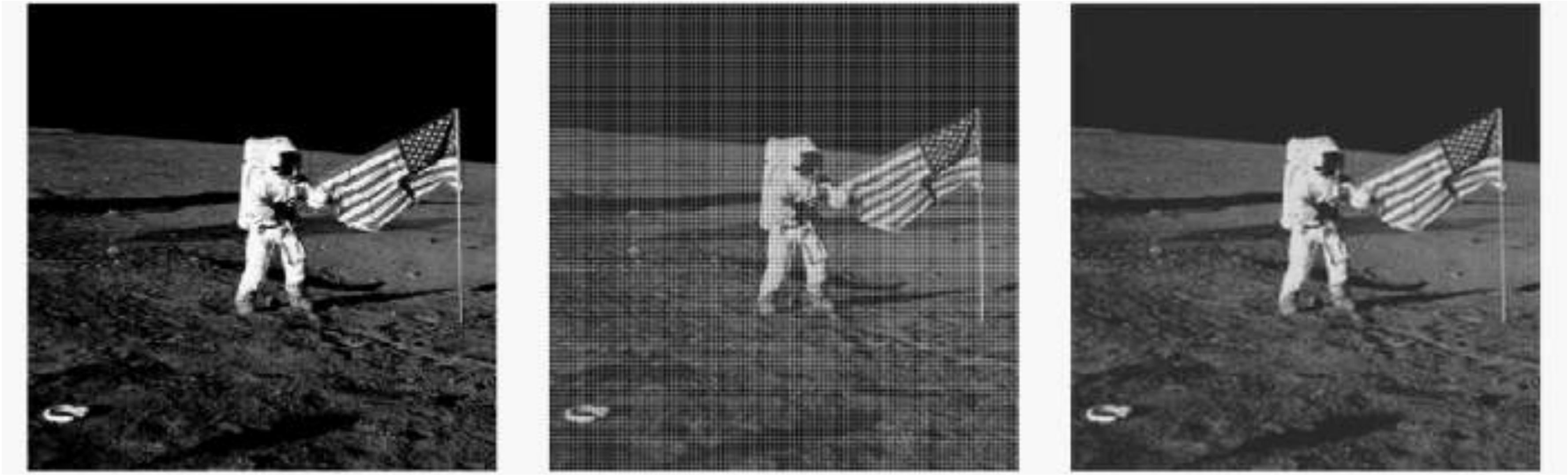
- We can still imagine that the image is **defined on a finite domain** $[0, M\text{-}1] \times [0, N\text{-}1]$ but is **set to zero outside this range**



- Behavior
  - *Low frequencies* of the image (from the Fourier transform) are kept
  - *High frequencies* are blocked (containing all fine details)



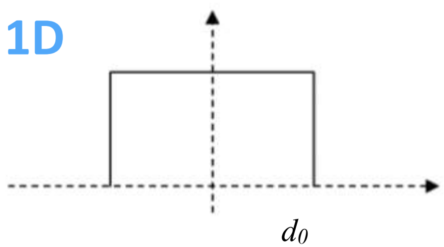
- Used to **smooth the image** or **reduce noise**



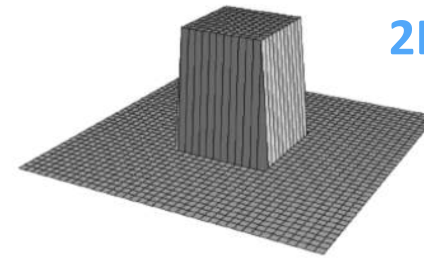
Low-pass filters

- **"Ideal"** filter

1D



$$H(x) = \begin{cases} 1, & \text{if } |x| \leq d_0 \\ 0, & \text{if otherwise} \end{cases}$$



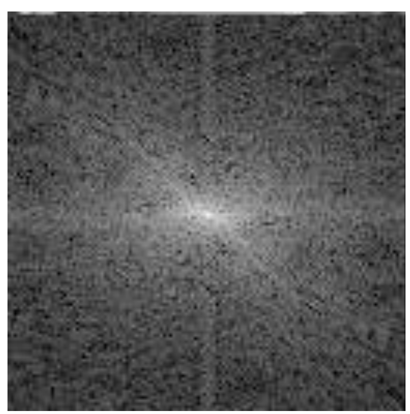
2D

- Do you remember the **mean filter**?

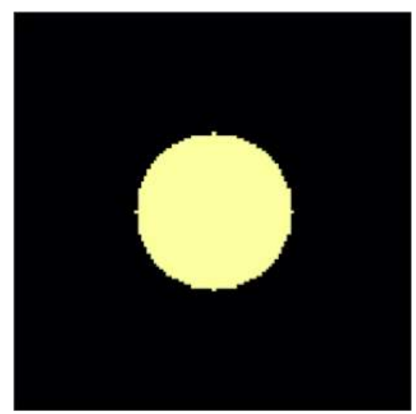
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

- Example

image I



FFT{ I }



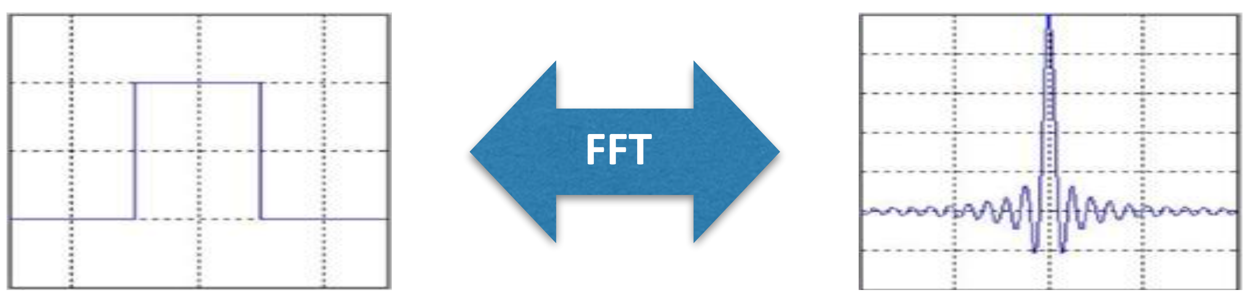
filter H

$I' = I * H$

Low-pass filters

- Note: the mean filter gives **blocky blurring/ringing**

- Why does this happen?
  - FFT pair

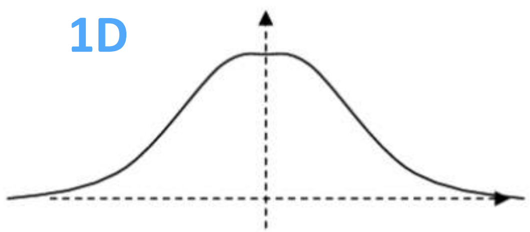


box sinc

  - What are we **convolving** with in Fourier space?

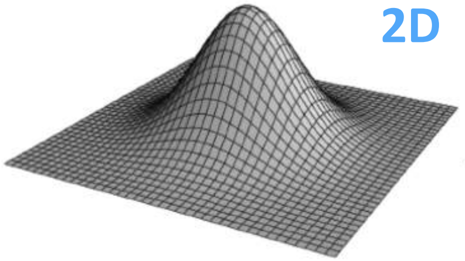
- **Gaussian** filter

1D

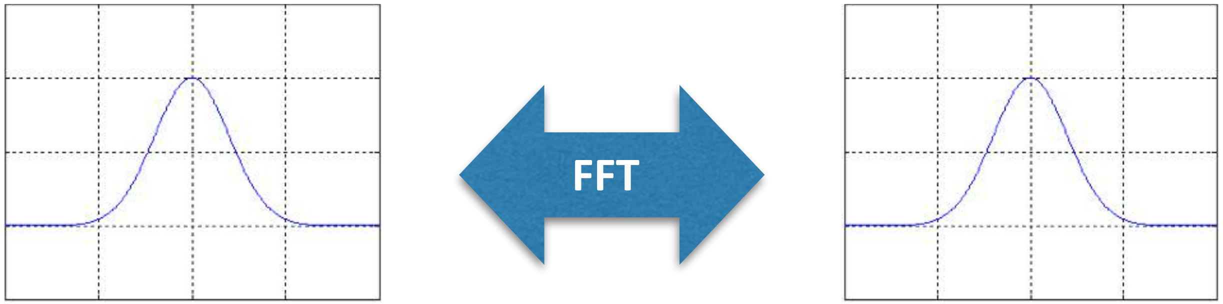


$$H(x) = e^{-\frac{x^2}{2\sigma^2}}$$

2D



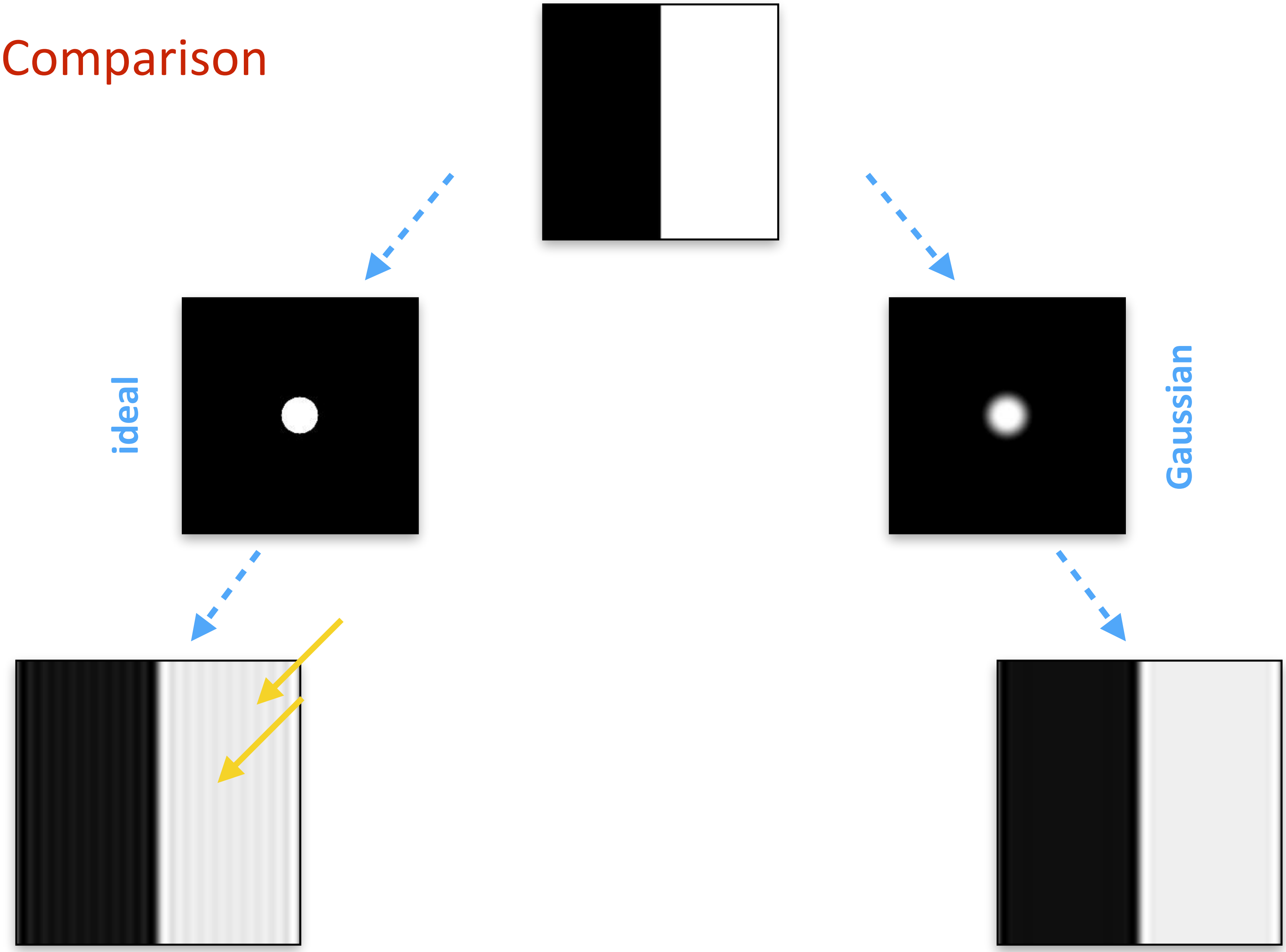
- Why is this better?
  - FFT pair



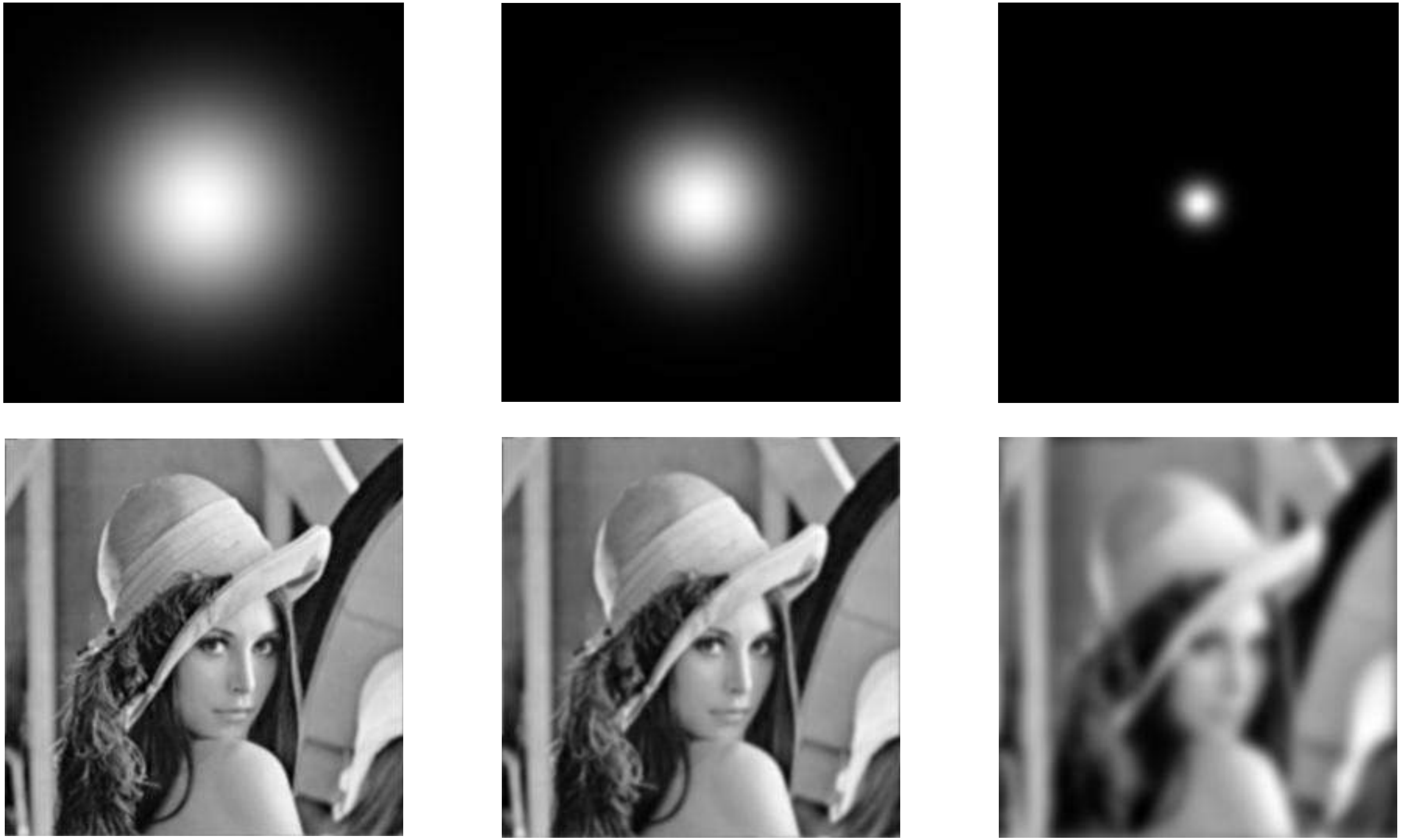
Gaussian | Gaussian

- What are we **convolving** with in Fourier space?
  - Do we expect again any ringing?

- Comparison

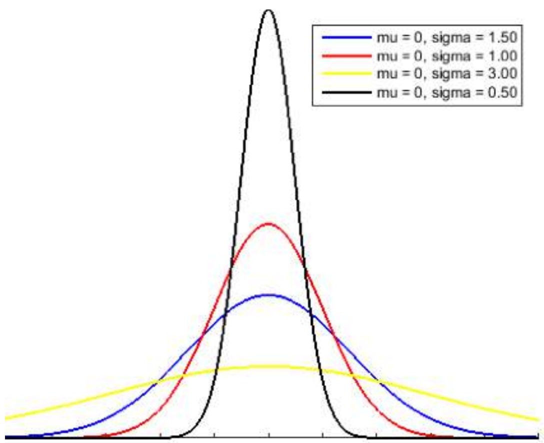


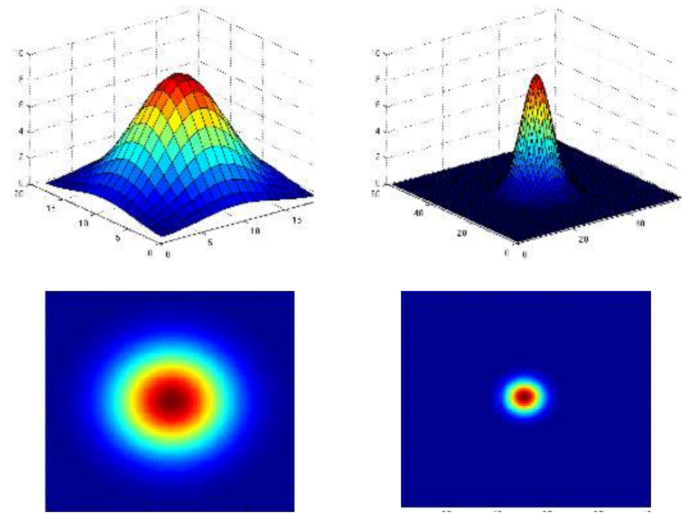
Comparison



- <u>Note 1</u>: the parameter σ controls **width** of Gaussian kernel
  - i.e. the amount of frequencies to keep/cut

1D

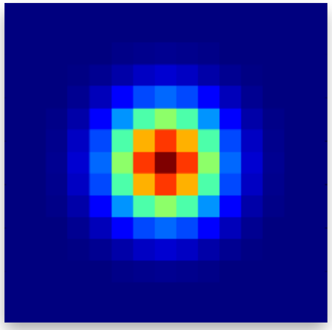


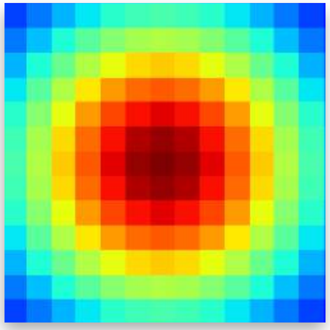


2D

- <u>Note 2</u>: theoretically, filters have **infinite support** but **discrete filters** use finite kernels
  - Usually assumed equal to **zero outside**
  - Which shape if σ **too large**?

small σ

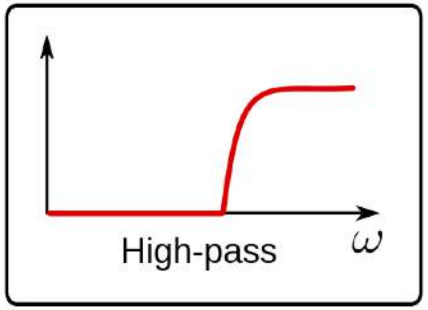




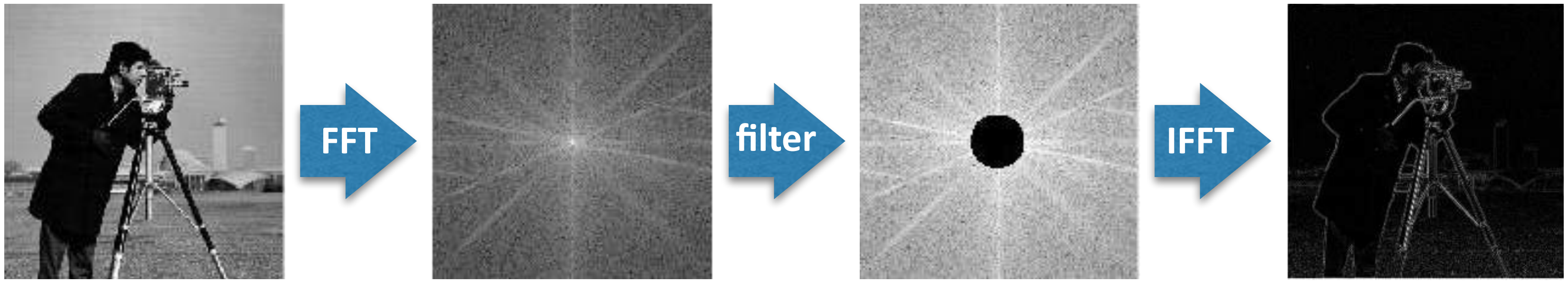
large σ

High-pass filters

- Behavior
  - *High frequencies* are kept, *low frequencies* are blocked
  - *Inverse* of low-pass filters



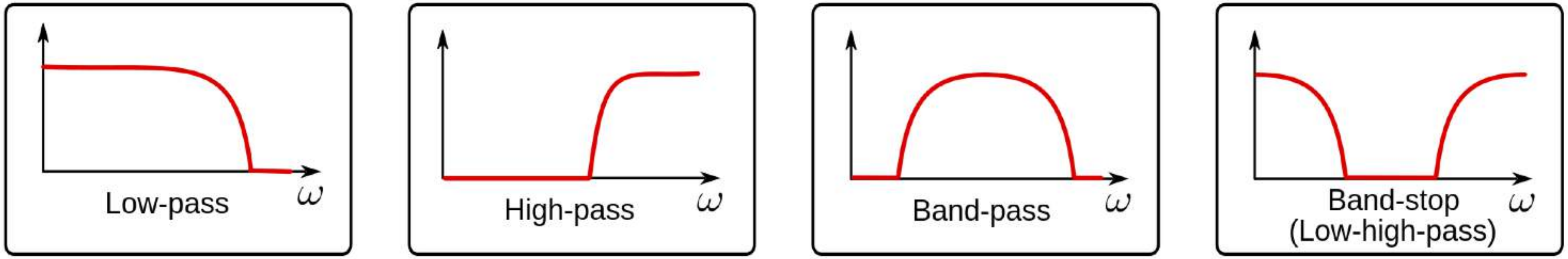
- Used to **sharpen the edges** of the image



- **Ideal** vs **Gaussian** filters
  - Same pros/cons of low-pass filters

Other frequency domain filters

- **Any filtering** can be performed in frequency domain



  - You can selectively **enhance** or **attenuate** any frequency component of the image
  - **NB**: always remember the *possible effects* in the spatial domain!
  - There is a *whole field of research* on this ➜ filter design

- Take-home messages
  - The process to move to the frequency domain usually is *more efficient*
  - It also provides valuable insight into the **nature of the image and filter**

- **NB**: soon, we will see other **uses of spatial filters**