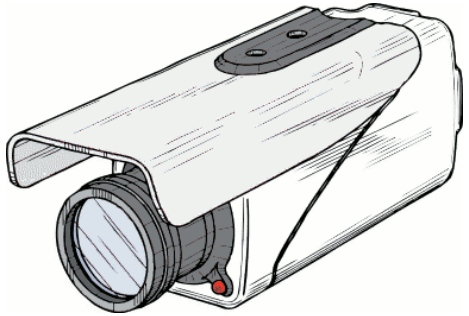
Univerza v Ljubljani

ViCOS Visual Cognitive Systems Lab

Machine perception
Image processing 2

Matej Kristan



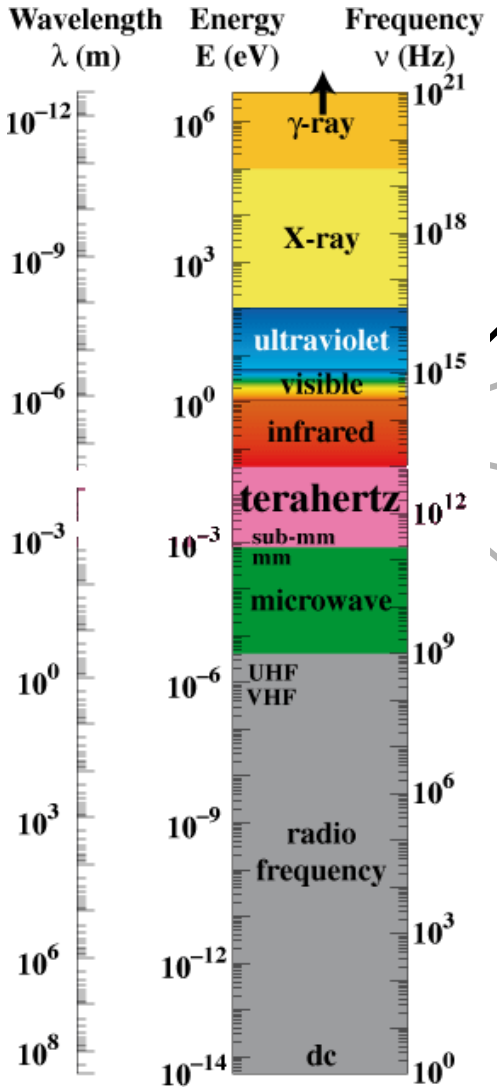
Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

Machine perception

COLOR

Sensor: Camera

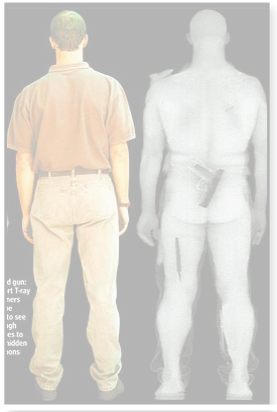
Electromagnetic spectrum



Visible light

Near-infrared light

Far-infrared light



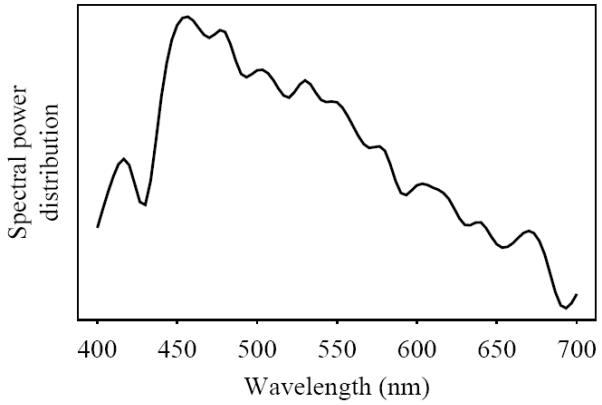
Terahertz light

http://userweb.elec.gla.ac.uk/d/dpaul/terahertz.html

Light

- *Light is an electromagnetic radiation composed of several frequencies*
- Properties are described by its spectrum (i.e., how much of each frequency is present)
- E.g., laser light contains only a narrow band of wavelengths (frequencies)

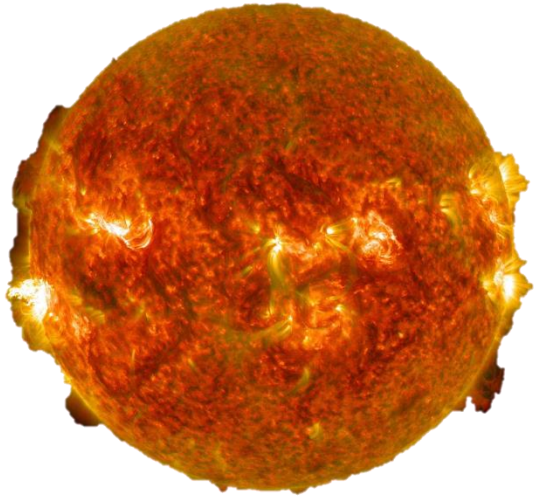
- Visible light contains radiation with wavelength of 400-700nm

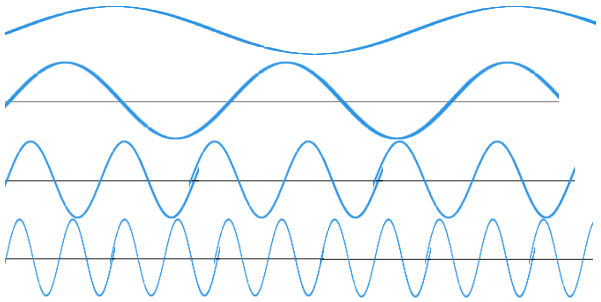


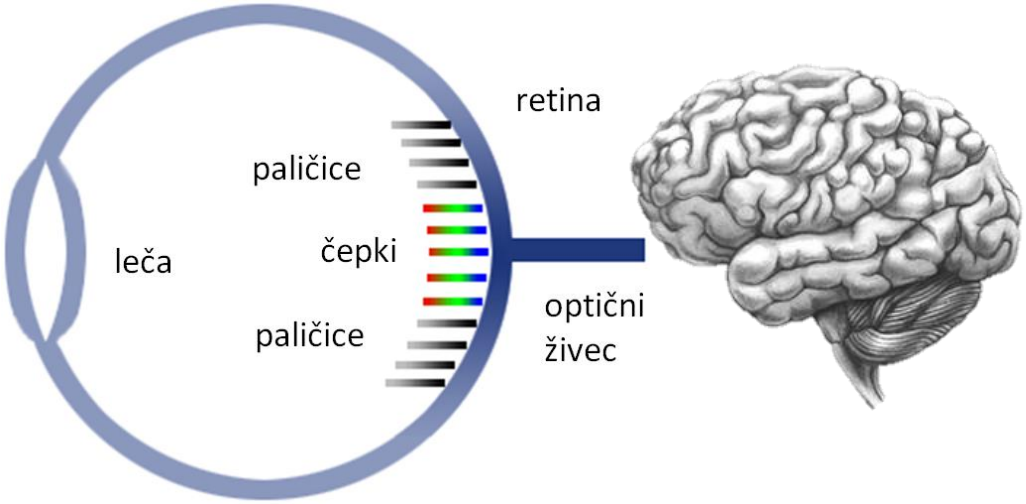
Spectrum: *Energy radiated in a unit of time w.r.t. wavelength.*

Human color perception

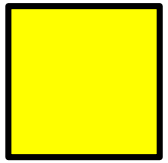
- Human eye (retina) contains specialized cells that react to different wavelengths differently.
- Three types of cells called “cones”: R, G, B
- A type of cells called “rods”: intensity only





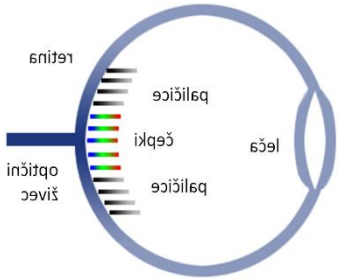


Yellow

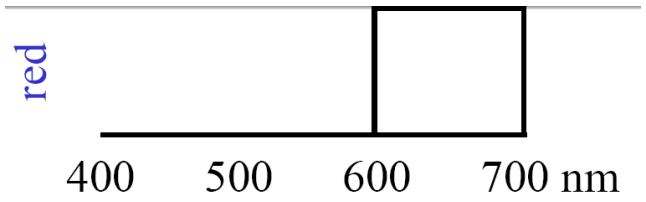


Additive mixture model

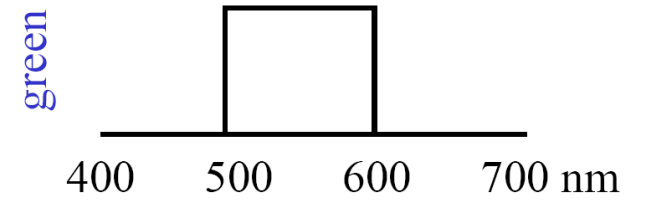
- What color do we get if we shine a red and green light?



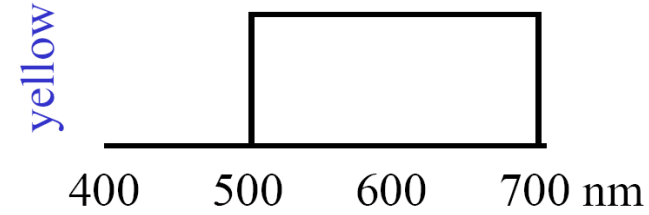
perceive



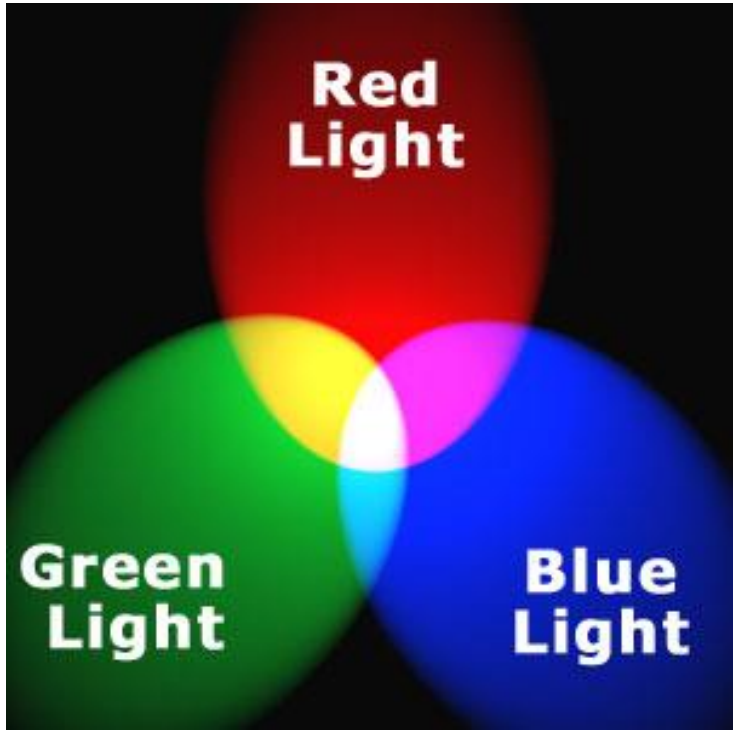
perceive



perceive



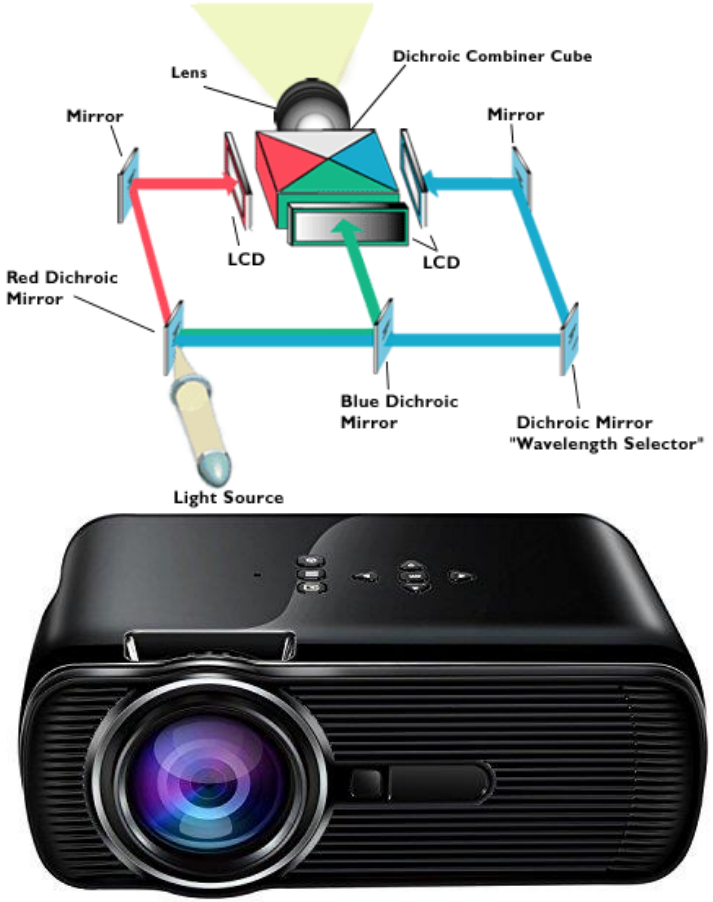
colors mix by summation of their spectra.



colors *added to black*.

Systems using the additive model

Monitors

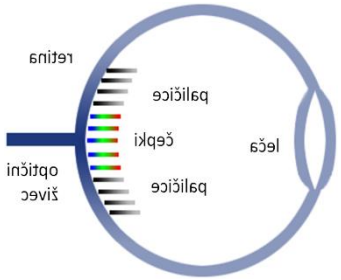


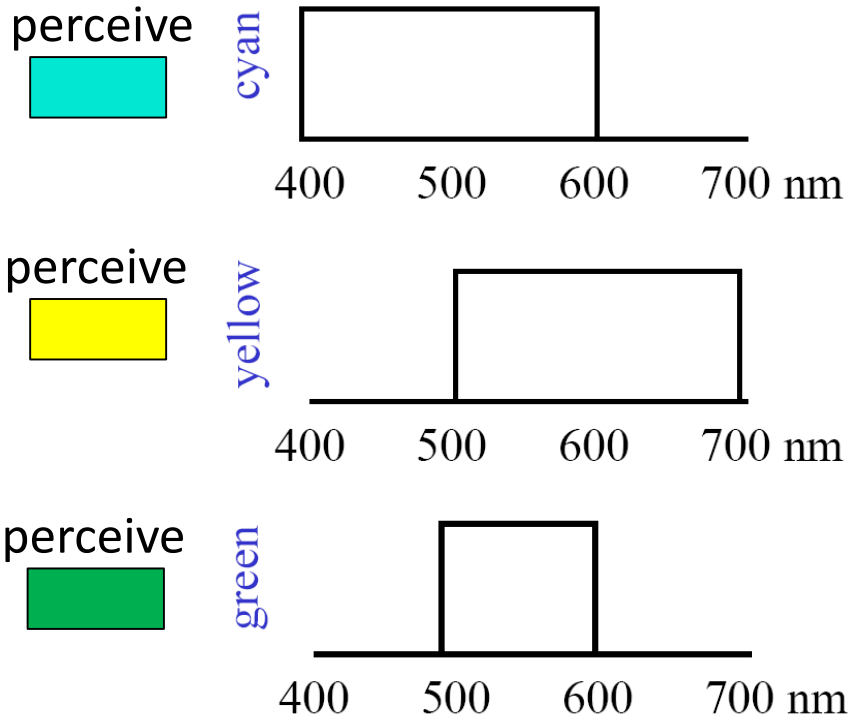
LCD projector

http://www.tech-faq.com/how-lcd-projectors-work.html

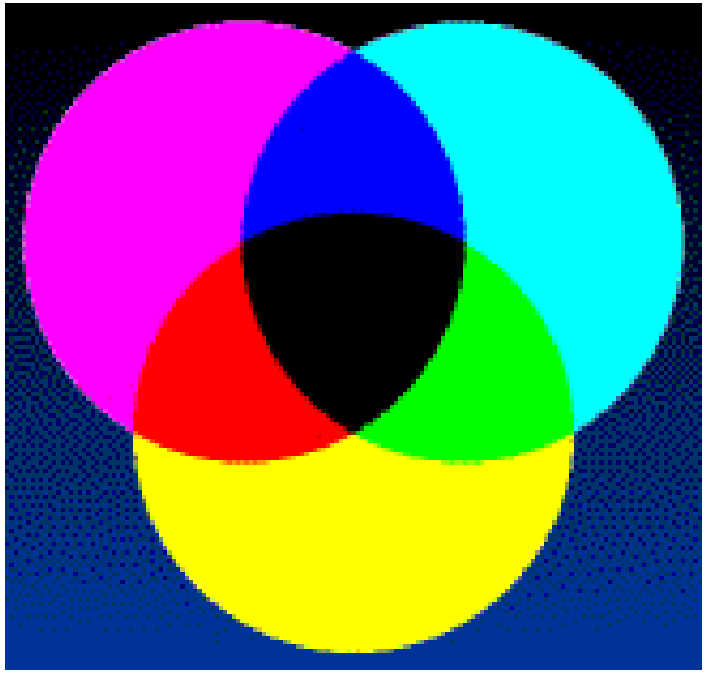
Subtractive models

- What color do we get if applying
  cyan and yellow pigment to white paper?





colors mix by *spectra multiplication*.



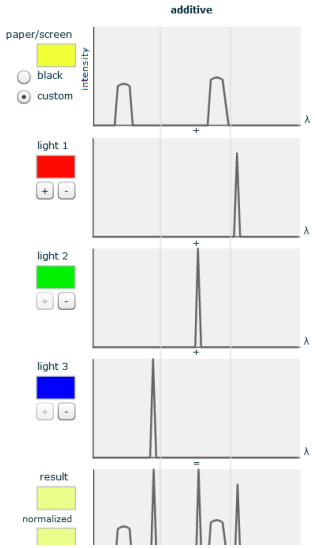
Pigments *remove* the color from the incident white light.

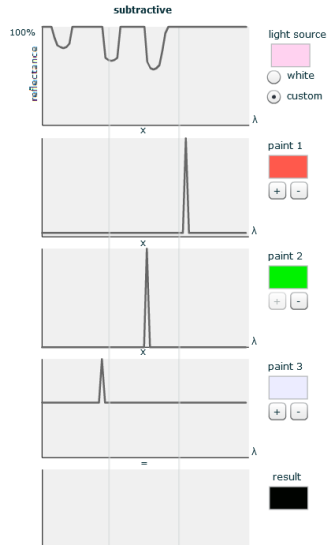
Systems using a subtractive model

- Printing on paper
- Crayons
- Photographic film

- See this nice app and play with setups:

  https://graphics.stanford.edu/courses/cs178/applets/colormixing.html



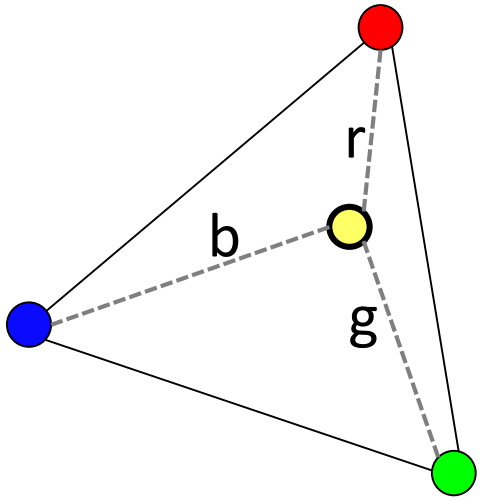


Color spaces

- Role of colors pace: Unique color specification (e.g., for reproduction)
  - *Specifying a color in a color space allows accurate color reproduction on various media like photo, print and monitor.*
- Defined by the choice of primary colors *(primaries)*

  Recall: The human eye is equipped with sensory cells for the perception of the three primary colors (RGB)
- New color is a weighted sum of primaries

By mixing the colours, we get any colour that lies within the triangle of primaries.



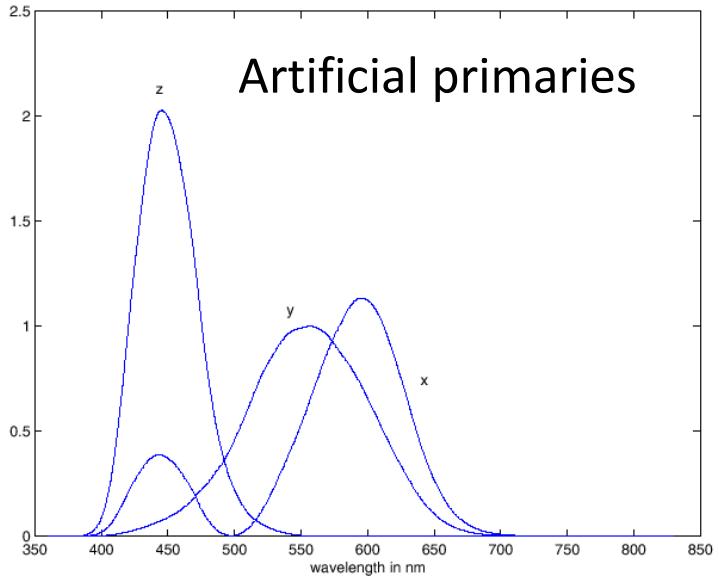
- Mixing weights *r,g,b* to get any color were estimated on human subjects

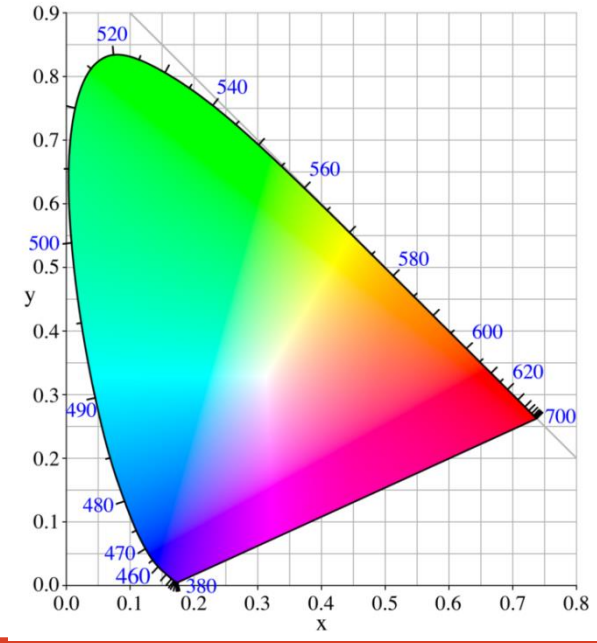
Linear color space example: CIE XYZ

- Commission international d'eclairage (CIE), 1931
- Y is intensity
- Representation by xy colorspace:

$$x = \frac{X}{X+Y+Z}, y = \frac{Y}{X+Y+Z}, z = \frac{Z}{X+Y+Z}$$

$$x + y + z = 1$$

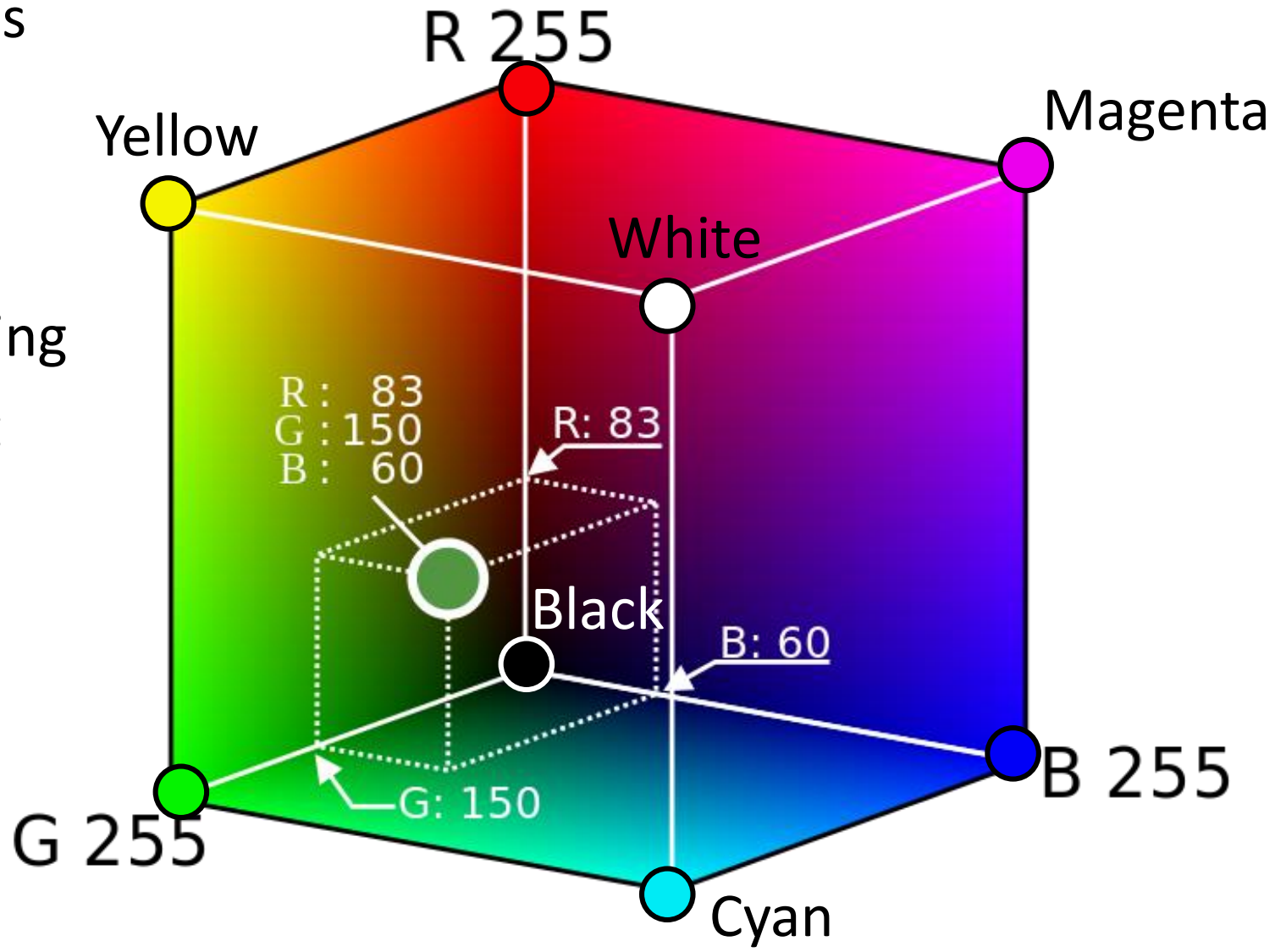




Linear color space example: RGB

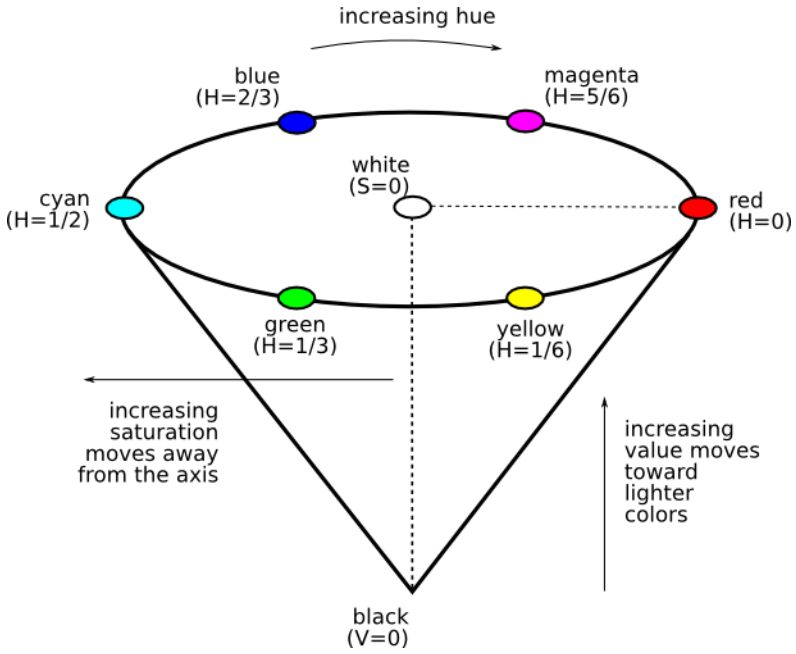
- Single wave-length primaries

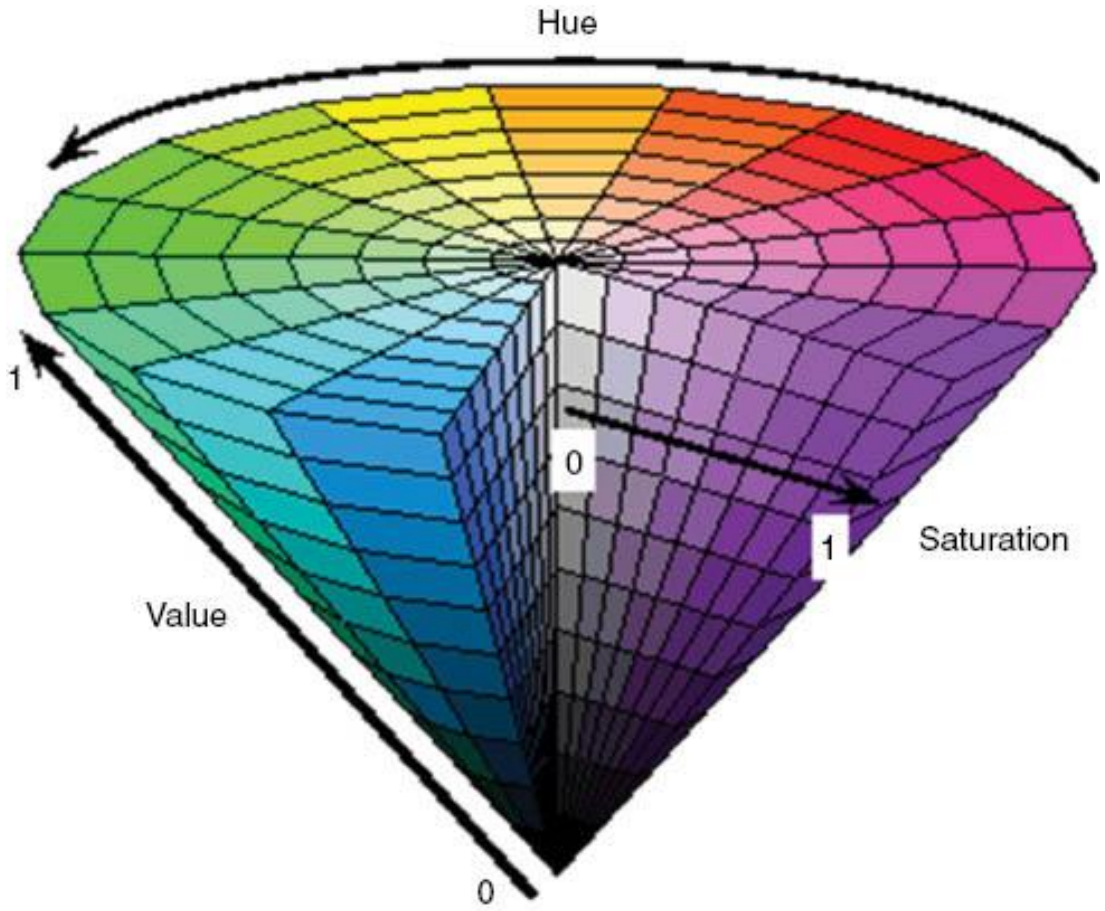
- Appropriate for use in imaging devices (e.g., monitors), but not for human perception



HSV colorspace

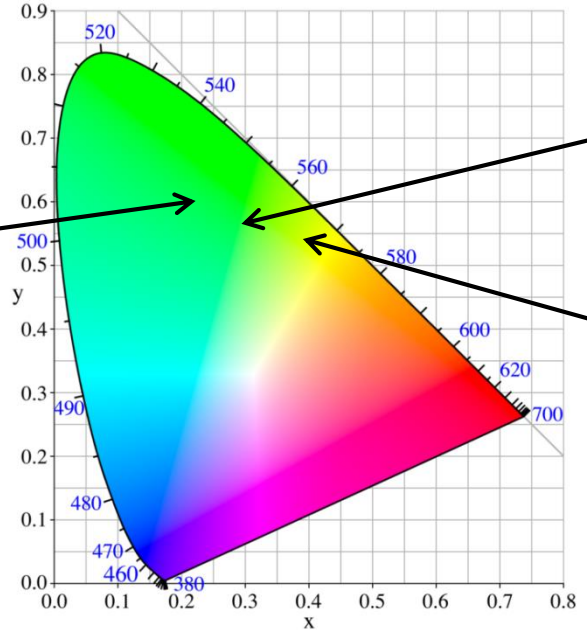
- Hue (barvnost), Saturation (nasičenje), Value (intenziteta)
- Nonlinear – hue coded by angle
- Matlab: hsv2rgb, rgb2hsv.





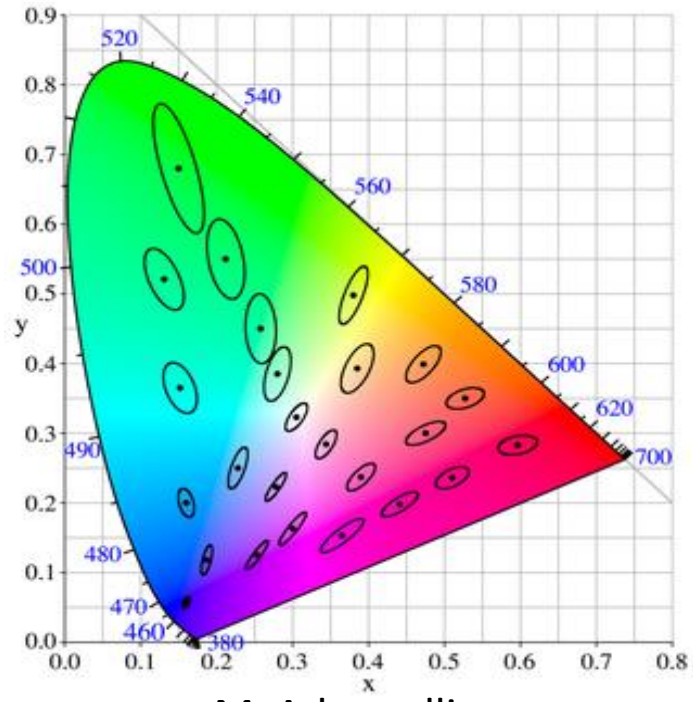
Distances in colourspaces

- Do distances between points in the colourspace make sense perceptually?



Distances in colorspaces

- Not necessarily: CIE XYZ is nonuniform colorspace – *Euclidean distance between coordinates of colors in colorspace is not a good indicator of color similarity (in terms of human perception).*



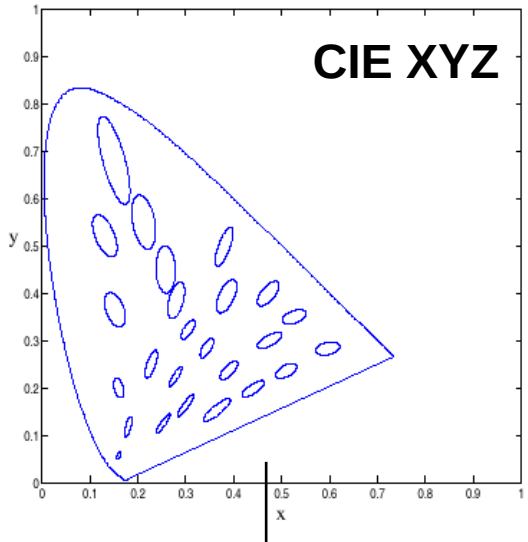
McAdam ellipses:
Just (human) noticeable differences in color

Uniform colour spaces

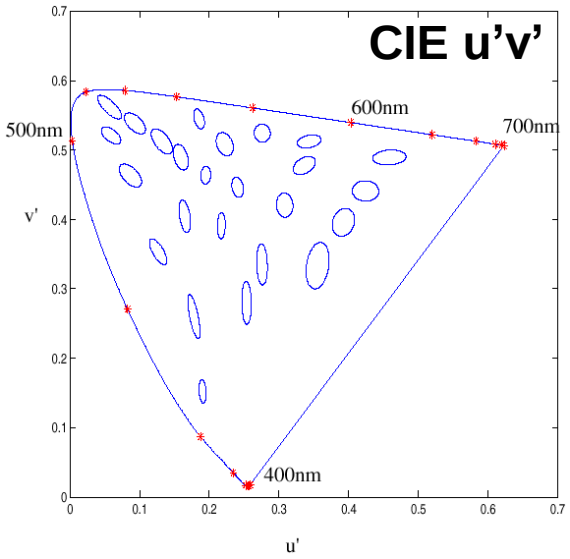
- Transforms such that ellipses are mapped into circles

  → *distances better replicate the human perception of color similarity.*

- Examples of uniform colour spaces:
  - CIE u‘v‘
  - CIE Lab



Nonuniform colour space



Uniform colour space

Machine perception

COLOR SIMILARITY USING HISTOGRAMS

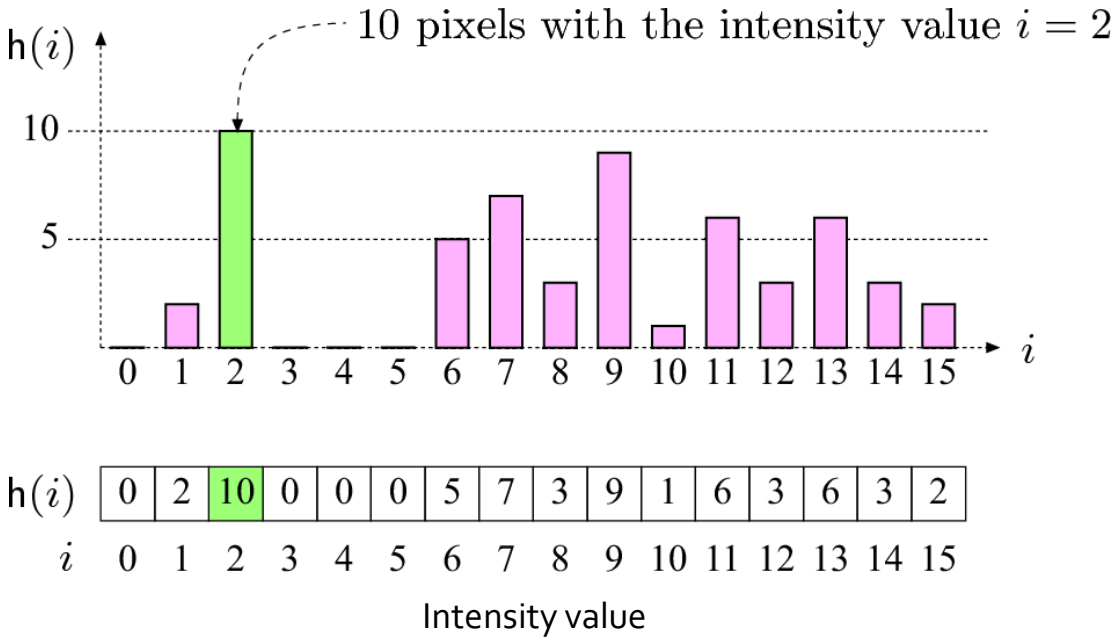
What is a histogram?

- Image histogram records the frequency of intensity levels

$$\mathrm{h}(i) = \text{ the } \textit{number} \text{ of pixels in } I \text{ with the intensity value } i$$

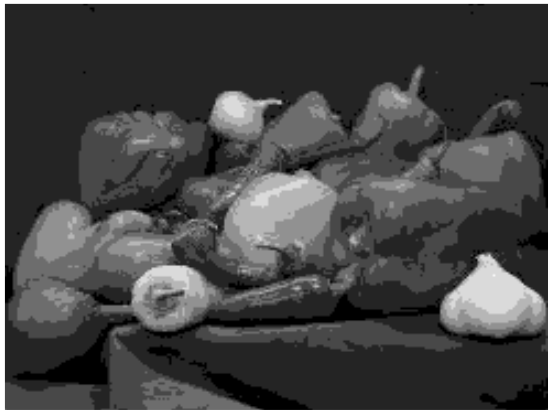
$$\mathrm{h}(i) = \mathrm{card}\{(u,v) \mid I(u,v) = i\}$$

- Example:



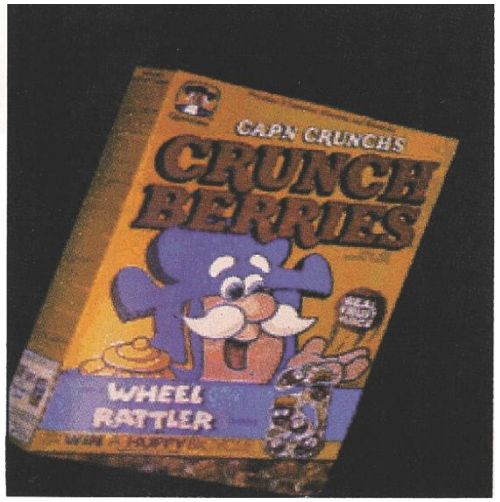
256 intensity levels

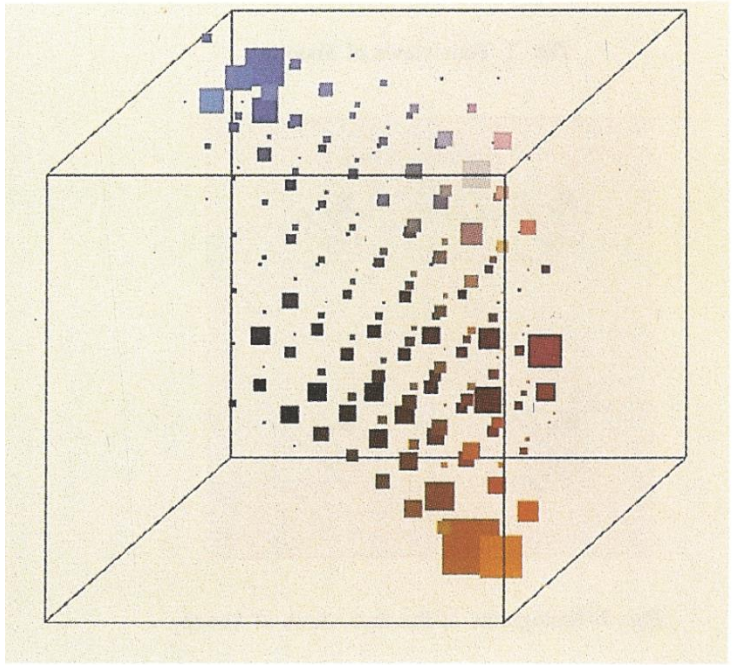
16 intensity levels



Color histogram

- Color statistic
  - Example of a 3D RGB histogram $H(R, G, B)$ visualization
  - Each pixel color is a point in 3D space (RGB)
  - Calculate the 3D color histogram
    - H(R,G,B) = number of pixels with color [R,G,B]

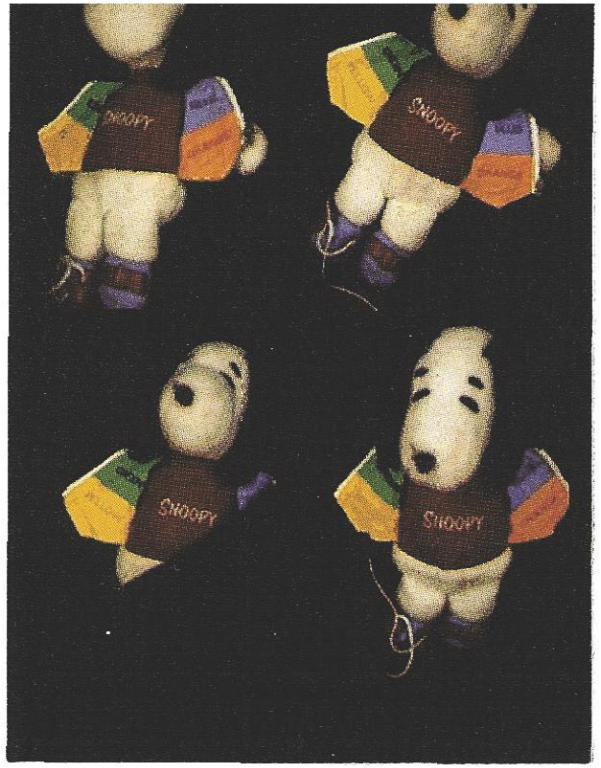


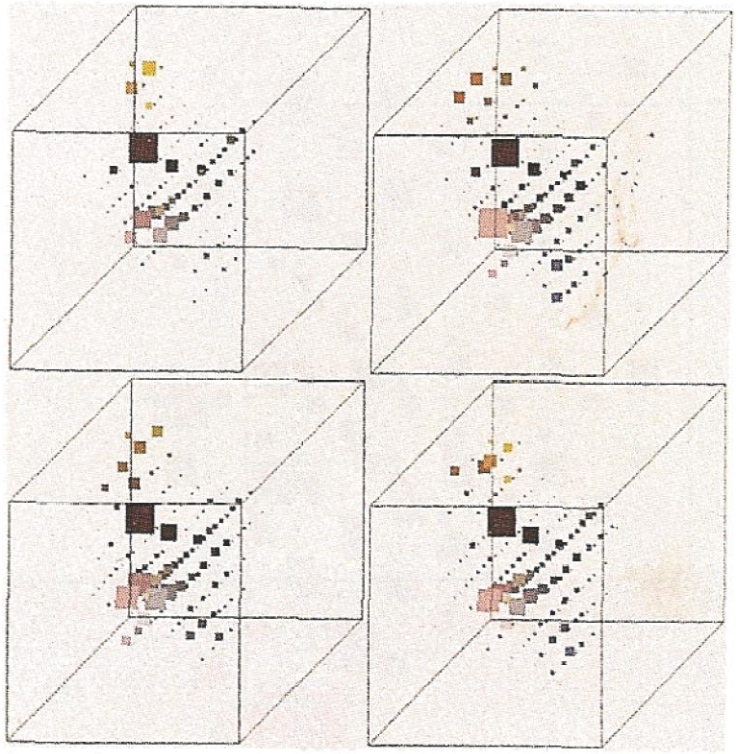


[Swain & Ballard, 1991]

Color histogram

- Robust representation of images
  - Translation, scale, partial occlusion





[Swain & Ballard, 1991]

Intensity normalization

- Intensity is contained in each color channel
  - Multiplying a color by a scalar changes the intensity but not really the true „color“ (hue).
  - This means that we can normalize a color by its intensity.
    - Intensity is defined as: $I = R + G + B$:
  - Chromatic representation:

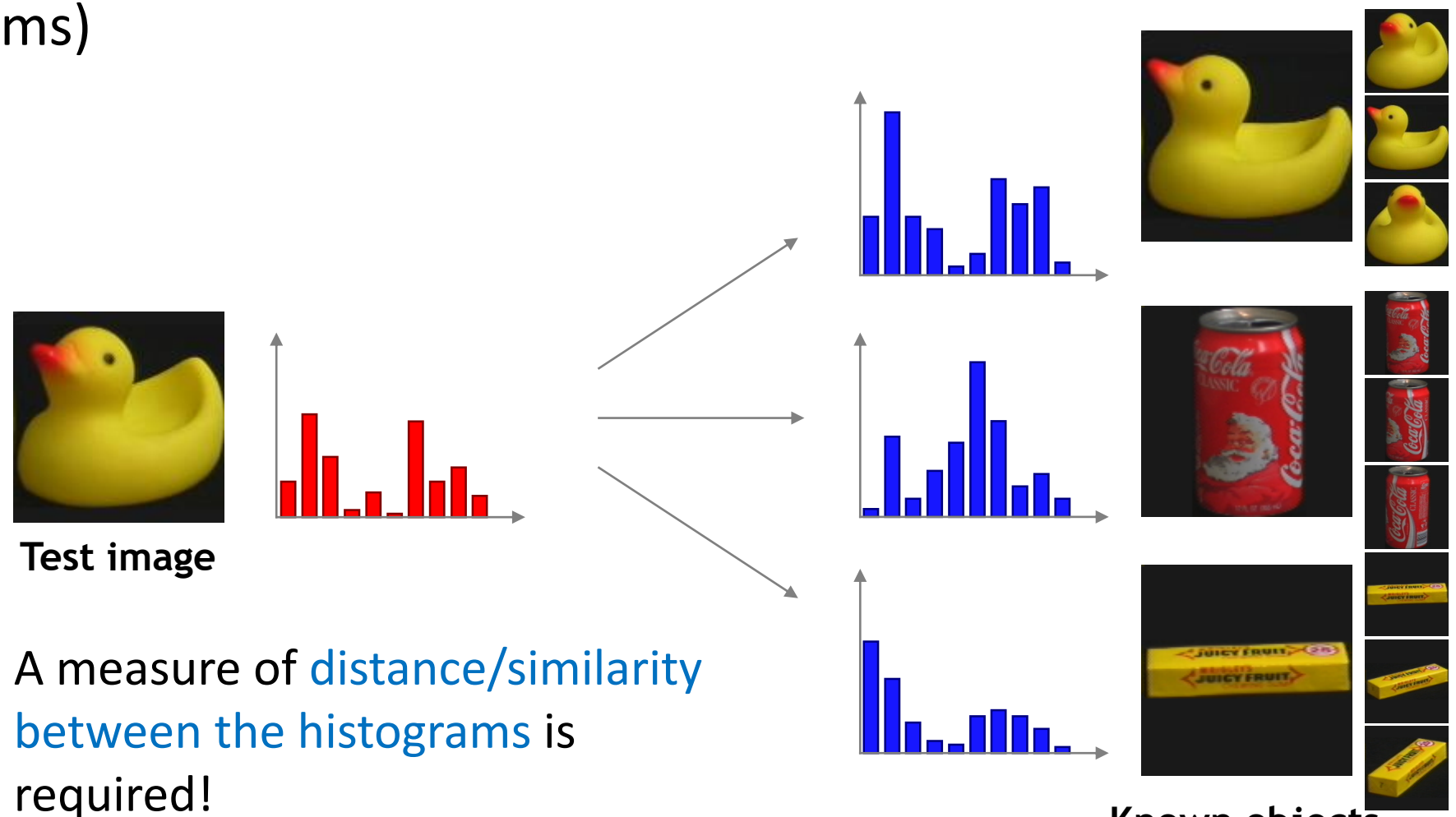
$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B}$$

  - We can now use only a 2D space (rg), since it holds that

$$r + g + b = 1$$

Color comparison via histograms

- Compare images indirectly – compare only their descriptors (histograms)

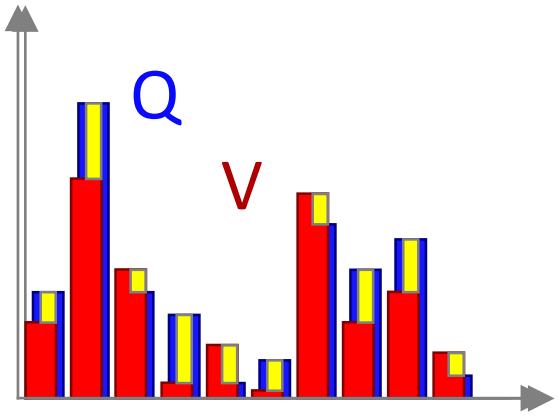


A measure of distance/similarity between the histograms is required!

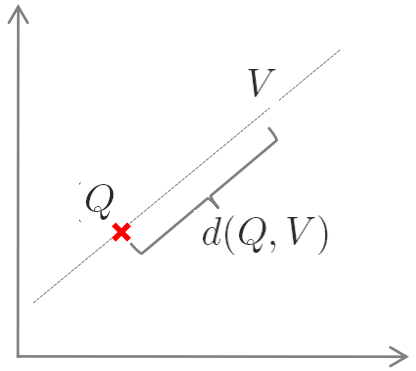
Popular distances: Euclidean distance

- Definition (=L_2 norm)

$$d(Q, V) = \sum_i (q_i - v_i)^2$$



- Explanation
  - Looks for differences in histogram cells.
  - Interpretation: Distance in feature space.
  - Range of output values: [0,1]
  - All cells receive equal weight.
  - Susceptible to noise!

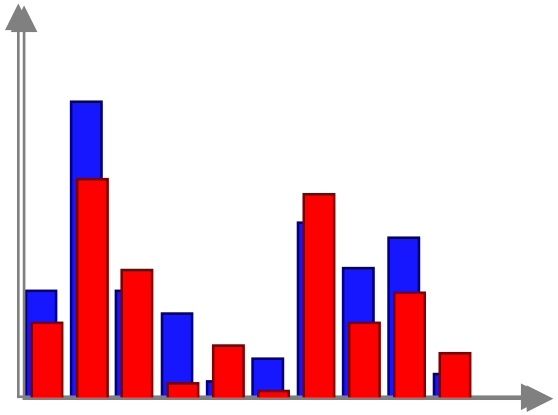


Popular distances: pdf similarity

- Similarity between two probability density functions
- Chi-squared (slo., hi-kvadrat):

$$\chi^2(Q,V) = \sum_i \frac{(q_i - v_i)^2}{q_i + v_i}$$

WATCH OUT FOR qi=vi=0!!



- Kullback-Leibler divergence:

$$KL(Q,V) = \sum_i q_i \log \frac{q_i}{v_i}$$

Not a proper metric (not symmetric)

Symmetric version (Jeferey's divergence):

$$JD(Q,V) = KL(Q,V) + KL(V,Q)$$

- Hellinger distance:

$$d_{\mathrm{Hell}}(Q,V) = \sqrt{1 - \sum_i \sqrt{q_i v_i}}$$

Proper metric, constrained to interval [0,1]

Machine perception

FILTERING

Can be applied for…

- Noise reduction and image restoration

noisy

Filtering

Less noisy

- Structure extraction/enhancement (later in the course)

Filtering

Types of image noise

- Salt and pepper (*sol in poper)*
  - Random black and white dots.
- Impulse noise (Impulzni šum)
  - Random occurrence of white dots.
- Gaussian noise (Gausov šum)
  - The intensity variation sampled from a Gaussian (Normal) distribution.

Original

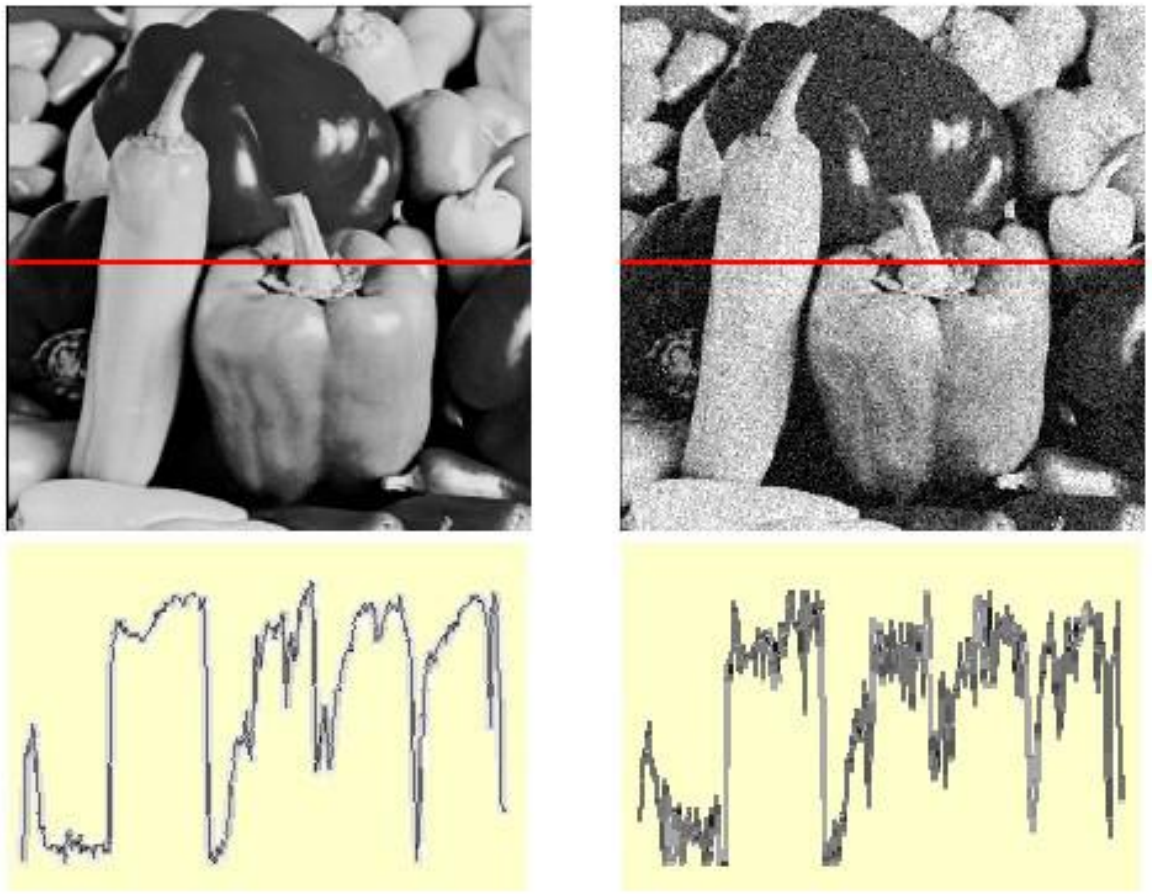
Salt and pepper noise

Impulse noise

Gaussian noise

Vir: Steve Seitz

Gaussian noise



$$f(x,y) = \overbrace{\bar{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

```
Matlab:
>> im = imread("peppers.jpg");
>> noise = randn(size(im)).*5;
>> output = im + noise;
```

Let’s try to remove the noise...

- Assumption:
  - Pixels are similar to their neighboring pixels
  - The noise is independent among pixels
    (“i.i.d. = independent, identically distributed”)

- So let’s compute an improved estimate of pixel’s intensity by replacing it with an average of pixel intensities in its immediate neighborhood...

A moving average 2D

$F[x, y]$

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

$G[x, y]$

	0	10							

Source: S. Seitz

A moving average 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Source: S. Seitz

A moving average 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

A moving average 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Source: S. Seitz

A moving average 2D

$F[x, y]$

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

$G[x, y]$

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

Source: S. Seitz

A moving average 2D

- Assume the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Equal weights for all pixels.}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\text{A loop over all pixels within the neighbourhood of F[i,j].}}$$

- Now let's generalize this by making a weight depend on relative position from the central element.

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Nonuniform weights}} F[i+u, j+v]$$

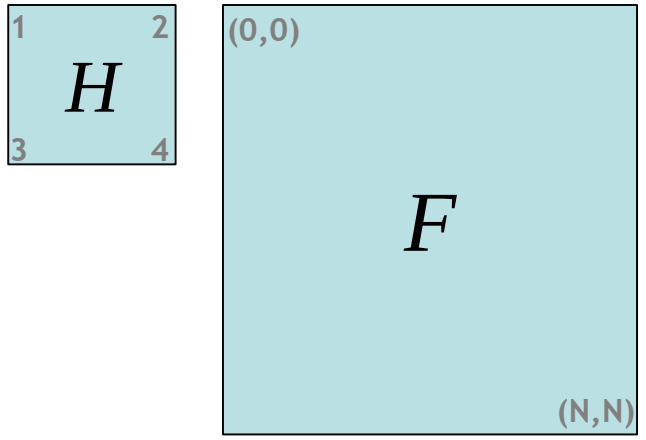
Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

- This is called cross-correlation and abbreviate as:

$$G = H \otimes F$$

- Image filtering
  - Replace image intensity with a weighted sum of a window centered at that pixel.
  - The weights in the linear combination are prescribed by the filter's *kernel*.



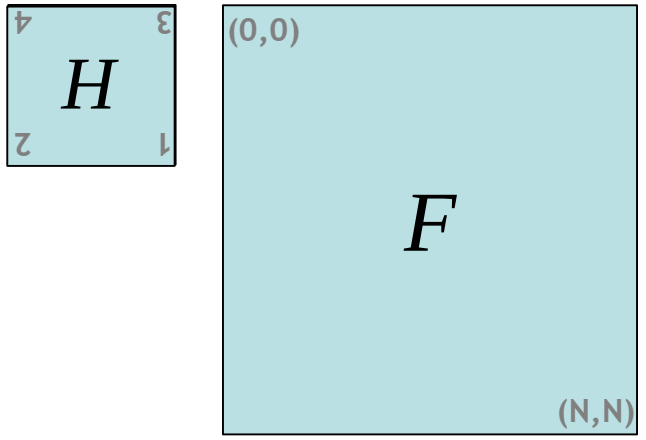
Convolution as correlation

- Compute convolution by cross-correlation:
  - Flip the filter in both dimensions (horizontal + vertical)
  - Apply cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

convolution operator



Convolution vs. Correlation

- Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

Matlab:
`filter2`
`imfilter`

Notice the difference?

- Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

Matlab:
`conv2`

(we will also use "*" to denote convolution, i.e., $G = H * F$.)

- Comment:
  - For a symmetric filter, $H[-u,-v] = H[u,v]$, correlation ≡ convolution.

Linear shift-invariant system

- Shift-invariant:
  - Equal behavior irrespective of position, i.e., the output only depends on the local pattern, but not image coordinates.

- Linearity:
  - Superposition: $h * (f_1 + f_2) = (h * f_1) + (h * f_2)$
  - Scaling: $h * (kf) = k(h * f)$ for a scalar k

Properties of convolution

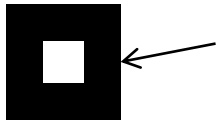
- Linear, shift-invariant
- Commutative: $f * g = g * f$
- Associative: $(f * g) * h = f * (g * h)$
  - As result, application of multiple filters is equal to application of a single filter :

$$\left((f * b_1) * b_2\right) * b_3 = f * (b_1 * b_2 * b_3)$$

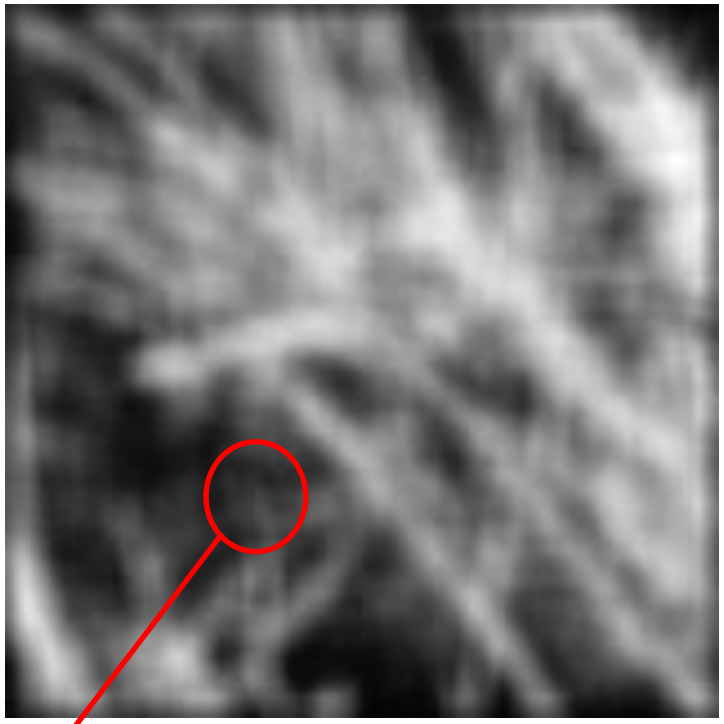
- Identity: $f * e = f$
  - A unit impulse: $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$.
- Derivative:

$$\frac{\partial}{\partial x}(f * g) = \left(\frac{\partial}{\partial x} g\right) * f = \left(\frac{\partial}{\partial x} f\right) * g$$

Recall smoothing by averaging



This is a box filter:
white = large weight,
black = low weight

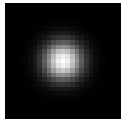


Original

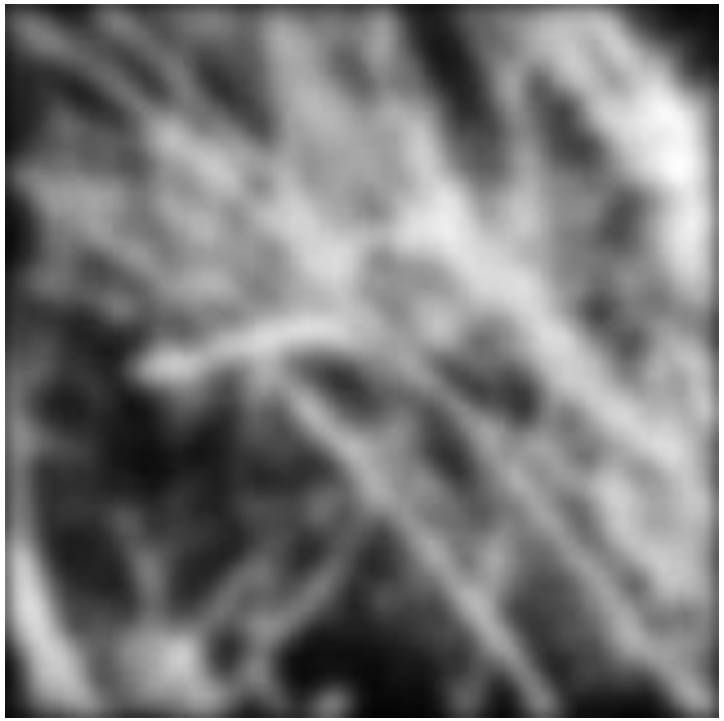
Filtered

Side lobe artefacts!

Smoothing by a Gaussian



Original

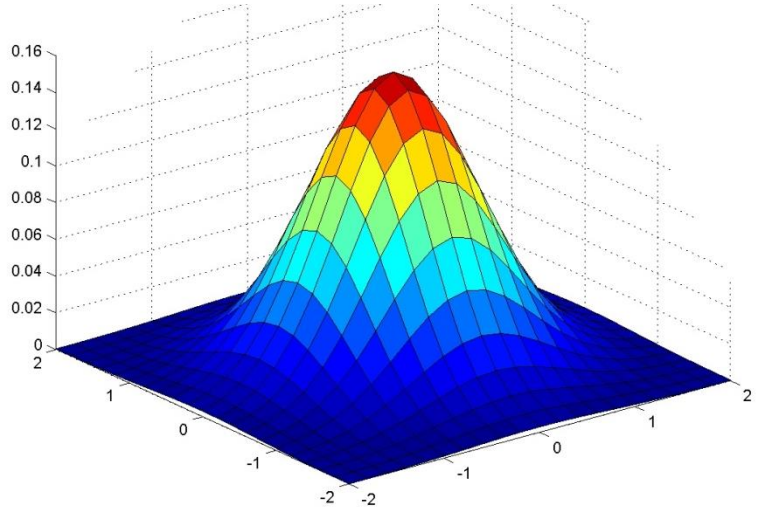


Filtered

Gaussian smoothing

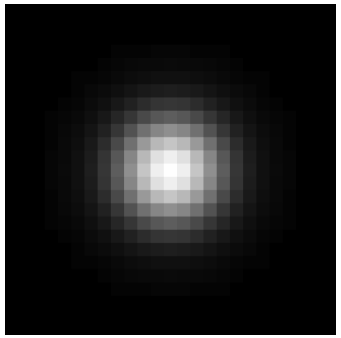
- Gaussian kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



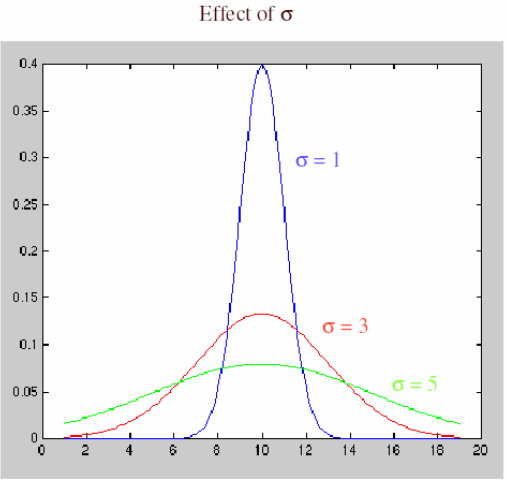
- Rotation symmetric

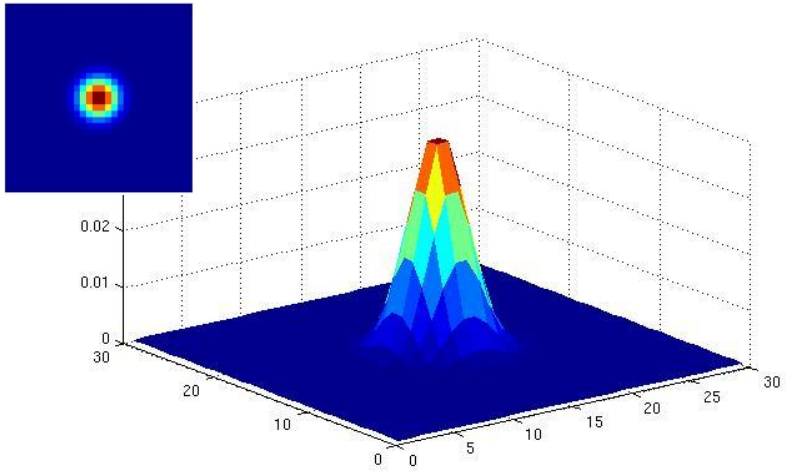
- Pixels closer to center get higher weight
  - Makes sense for a probabilistic inference of a signal content.



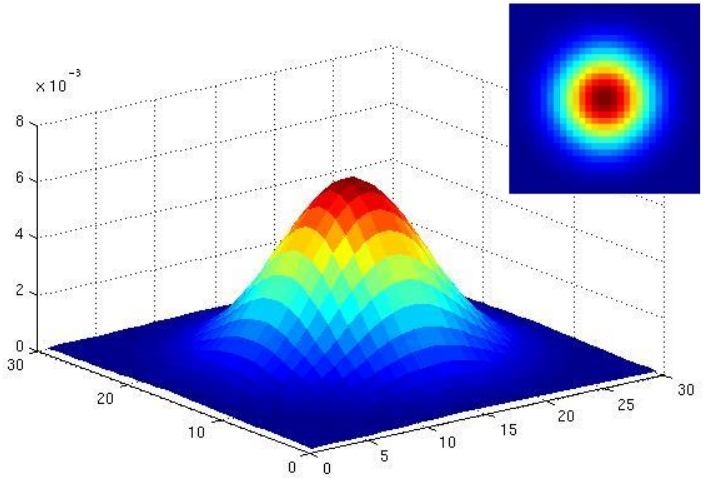
Gaussian smoothing

- How about parameters?
- *Variance* σ^2 determines the extent of smoothing...





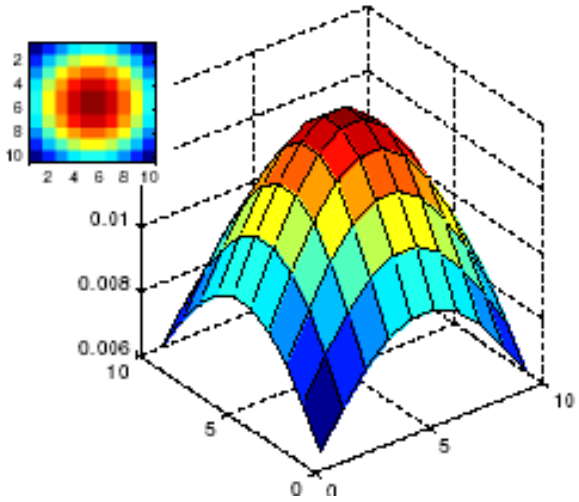
$\sigma = 2$ by kernel 30×30



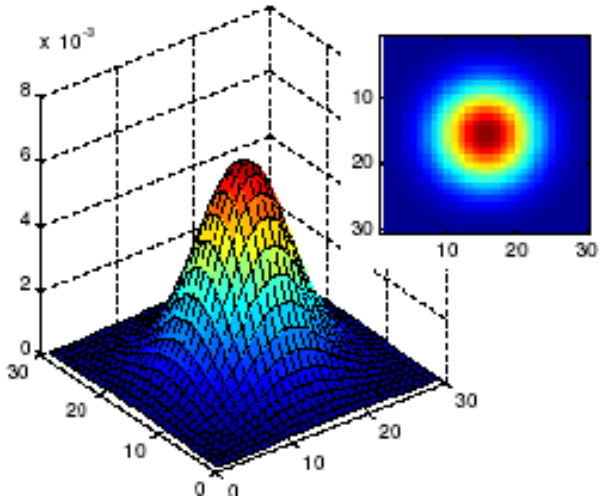
$\sigma = 5$ by kernel 30×30

Gaussian smoothing

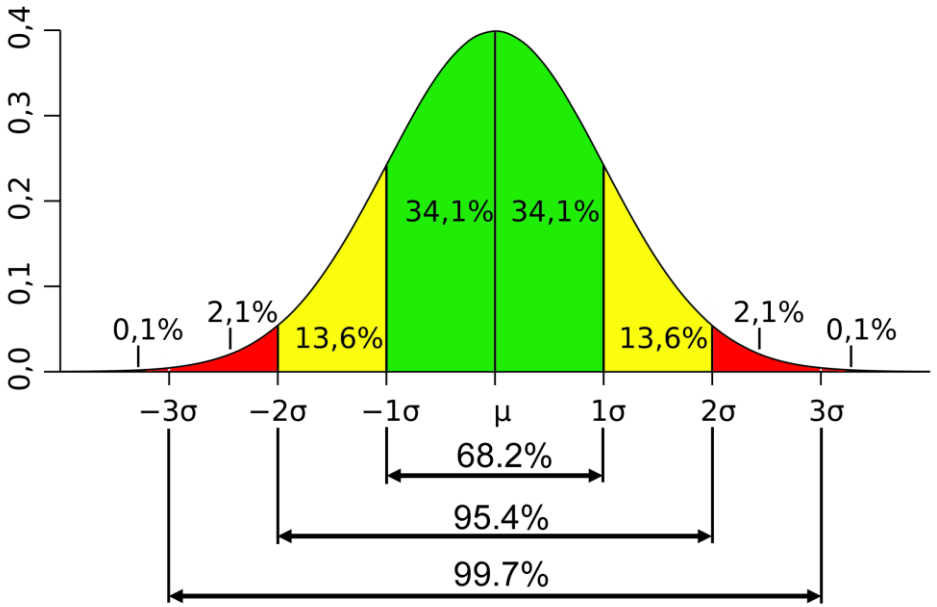
- How about parameters?
- *Kernel size*!
  - Infinite support, but discretization makes it finite.



$\sigma = 5$ with 10×10 kernel



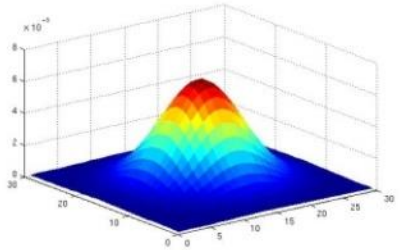
$\sigma = 5$ with 30×30 kernel

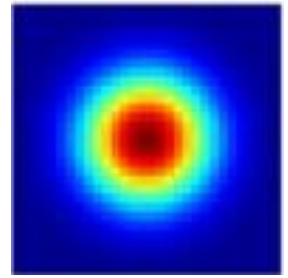


- Rule of thumb: set half size of the kernel to 3σ

Gaussian filtering in Matlab

```
>> sigma = 5;
>> hsize = 2*sigma*3+1;
>> h = fspecial('gaussian', hsize, sigma);
```



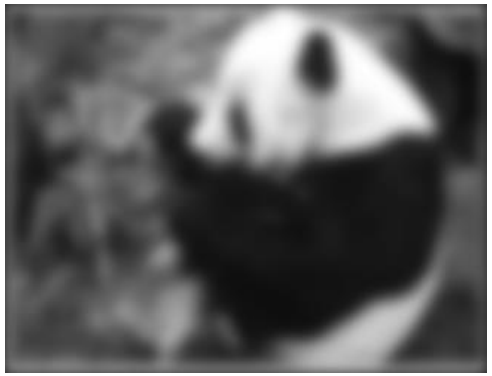


```
>> figure(1); surf(h);

>> figure(2); imagesc(h); axis equal ;

>> outim = imfilter(im, h);
>> figure(3); imshow(outim);
```

im

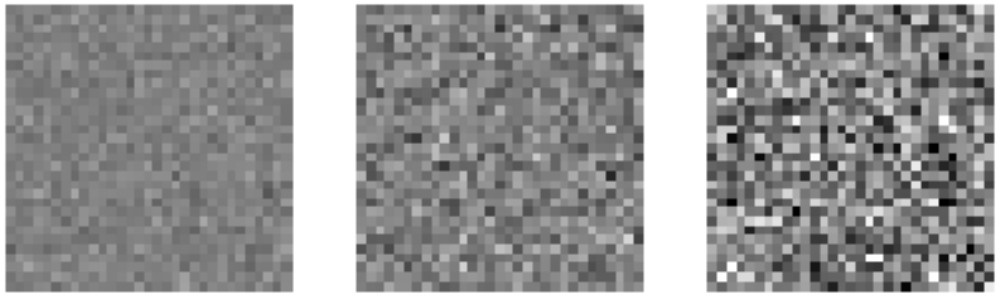


outim

Effects of smoothing

Increasing the noise extent →

σ=0.05 σ=0.1 σ=0.2



no smoothing

Increasing the kernel size →

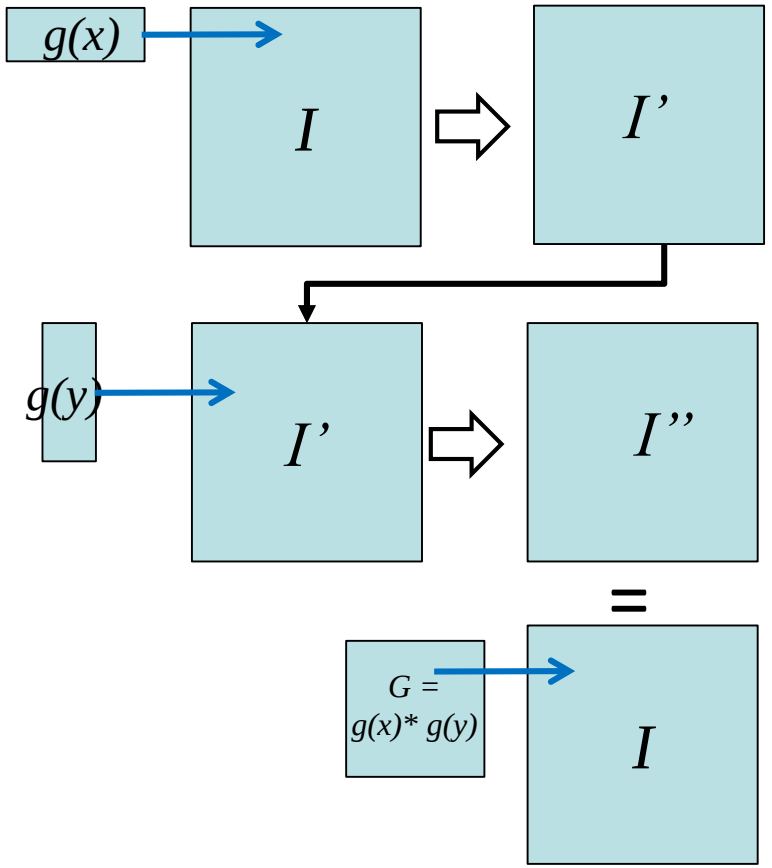
Efficient implementation

- Both, Uniform as well as Gaussian kernels are separable:
  - Apply convolution at each row separately using a 1D kernel:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-x^2/(2\sigma^2))$$

  - Next apply a 1D convolution at each column:

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-y^2/(2\sigma^2))$$

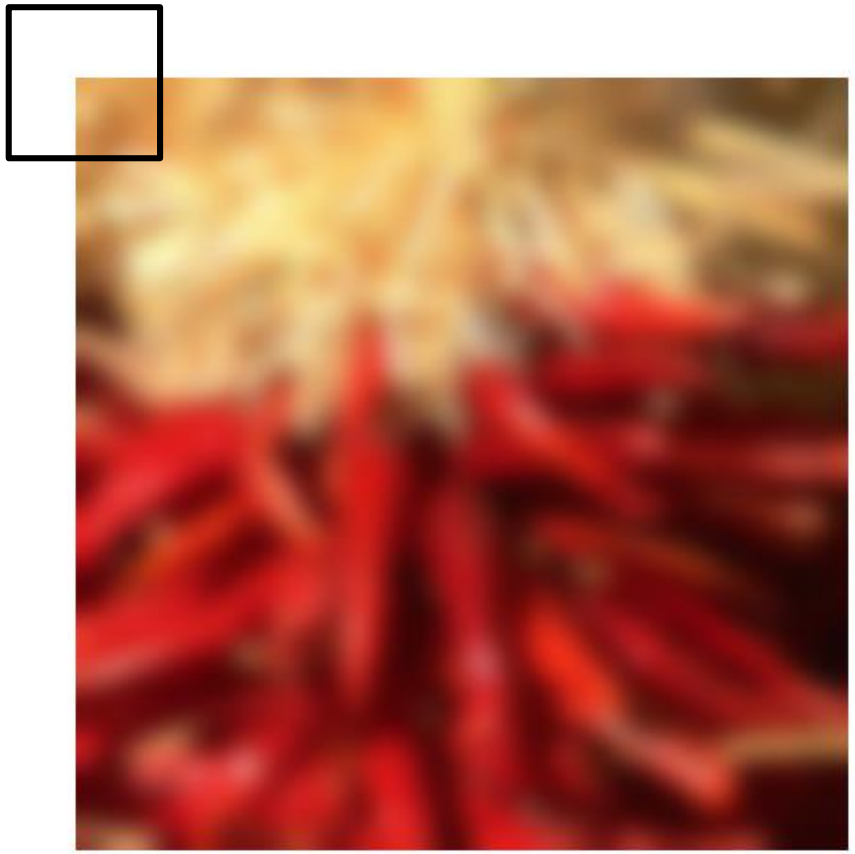


- Why is this separation possible?
  - Convolution is linear, associative + commutative

$$g_x * (g_y * I) = (g_x * g_y) * I$$

Filtering: Boundary conditions

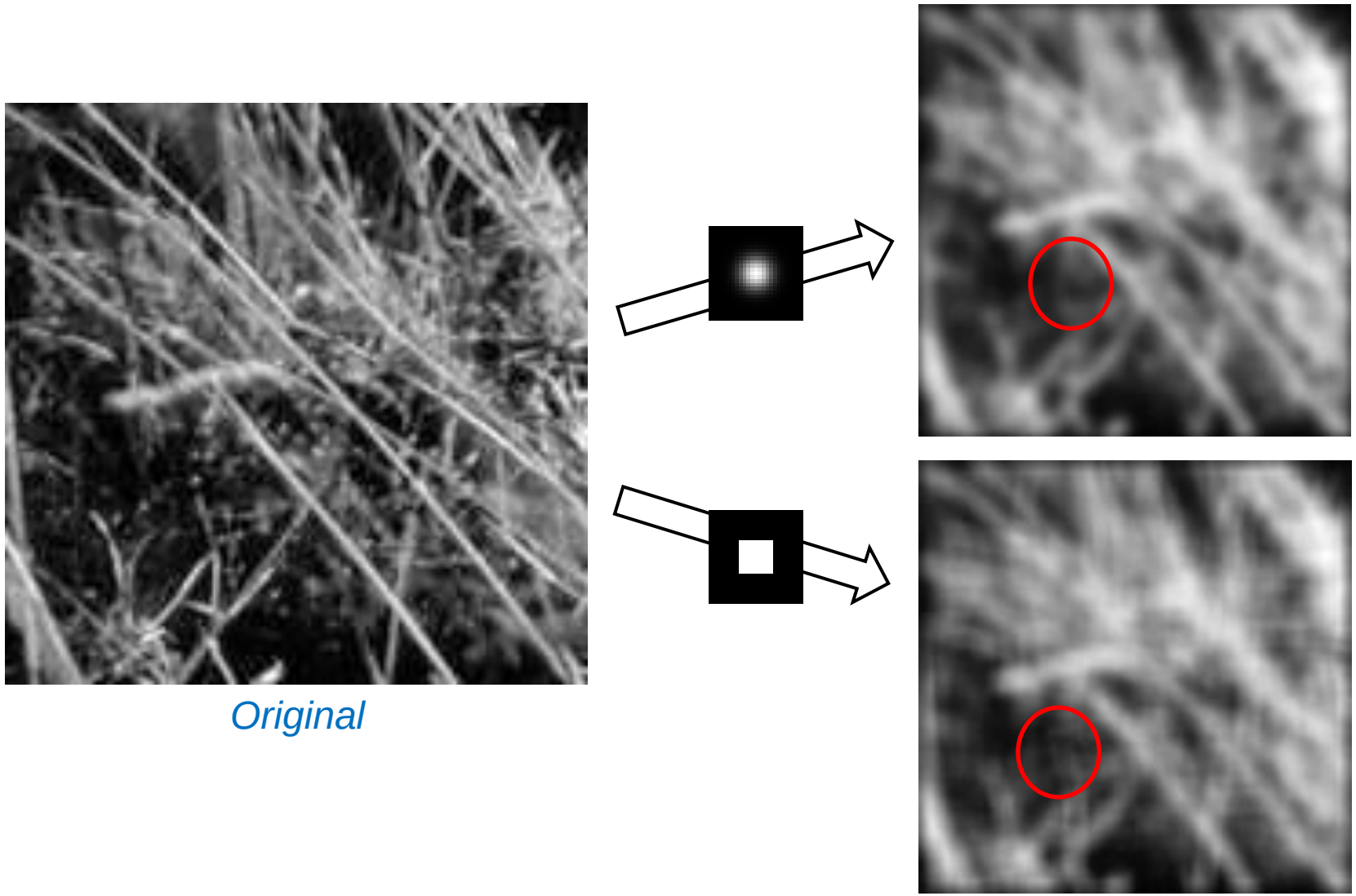
- What to do at the image boundaries?
  - The kernel exceeds image boundaries at the edge
  - Need for extrapolation
  - Methods:
    - Crop (black)
    - Bend image around
    - Replicate edges
    - Mirror image



Filtering: Boundary conditions

- What to do at the image boundaries?
  - The kernel exceeds image boundaries at the edge
  - Need for extrapolation
  - Methods (Matlab):
    - Crop (black): `imfilter(f,g,0)`
    - Bend image: `imfilter(f,g,'circular')`
    - Replicate edges: `imfilter(f,g,'replicate')`
    - Mirror image: `imfilter(f,g,'symmetric')`

Strange artefacts in convolution results…



Original

Filtered

Convolution and spectrum

- Convolution of two functions in image space is equivalent to the product of their corresponding Fourier transforms (spectra).

$$\mathcal{F}(f * g) = \mathcal{F}(f) \odot \mathcal{F}(g)$$

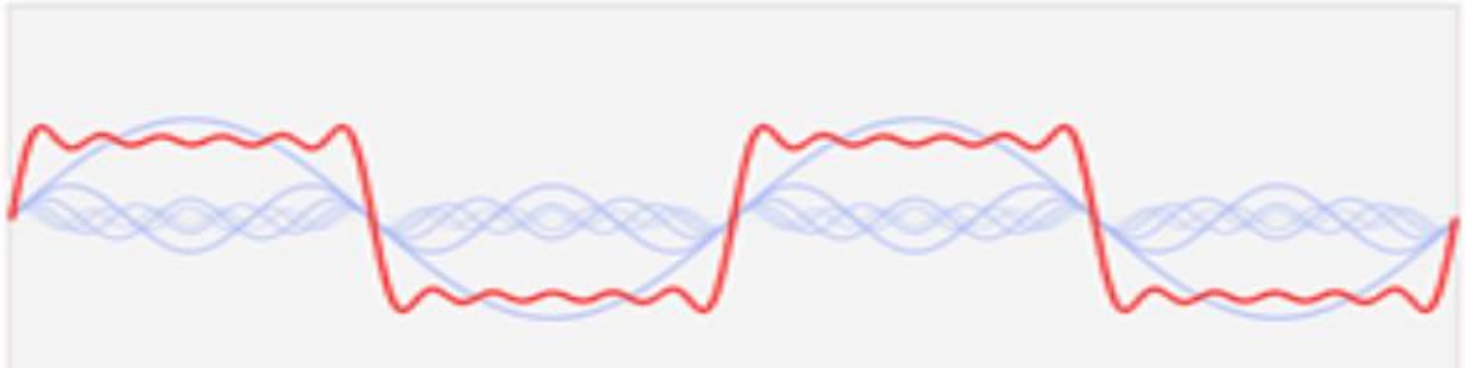
Image f and filter g

Fourier transforms of f and g.

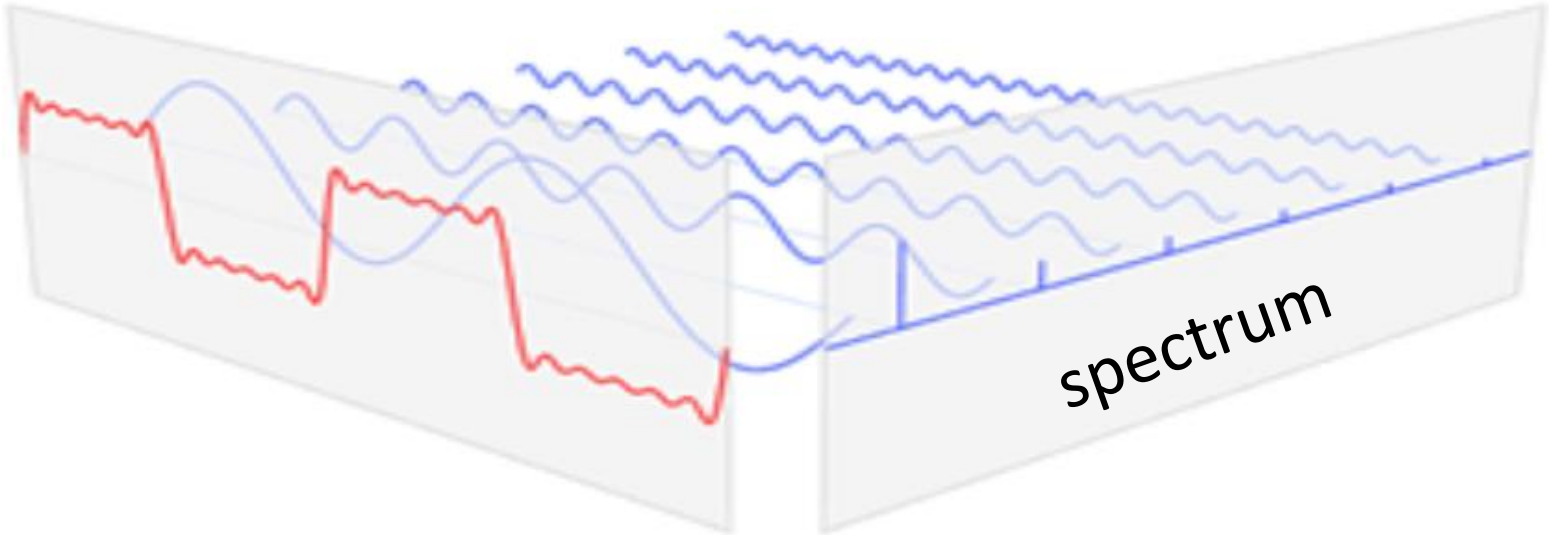
- Convolution manipulates the image spectrum
  - Enhancing/suppressing frequency bands in image.

Recall the Fourier transform

A signal is represented as a sum of sines/cosines of various frequencies



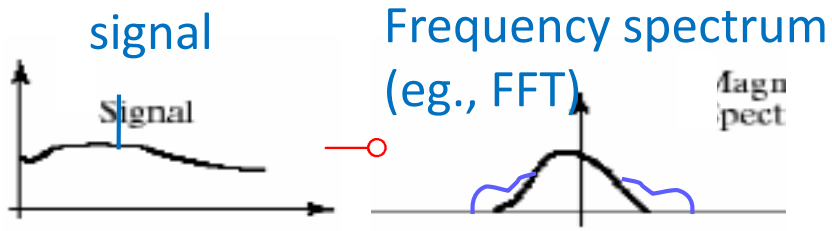
$$f(x) = \sum_n a_n \cos(nx) + b_n \sin(nx)$$



Images from: https://en.wikipedia.org/wiki/Fourier_transform

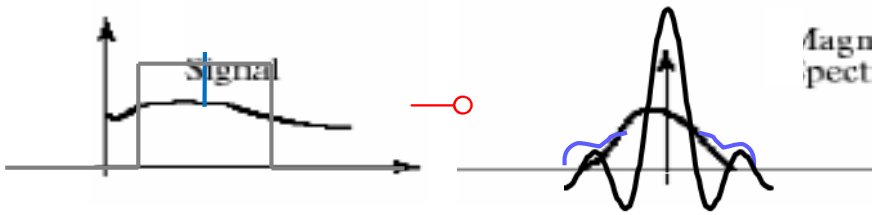
Convolution: removing noise

- Noise corresponds to adding high frequencies. To remove these, we apply a *low-band pass* filter.

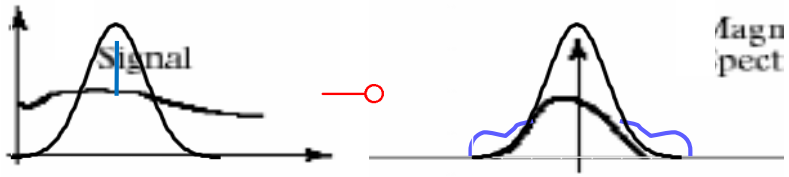


- The spatial box filter also transforms to a sinc in frequency space, causing artefacts (side lobes).

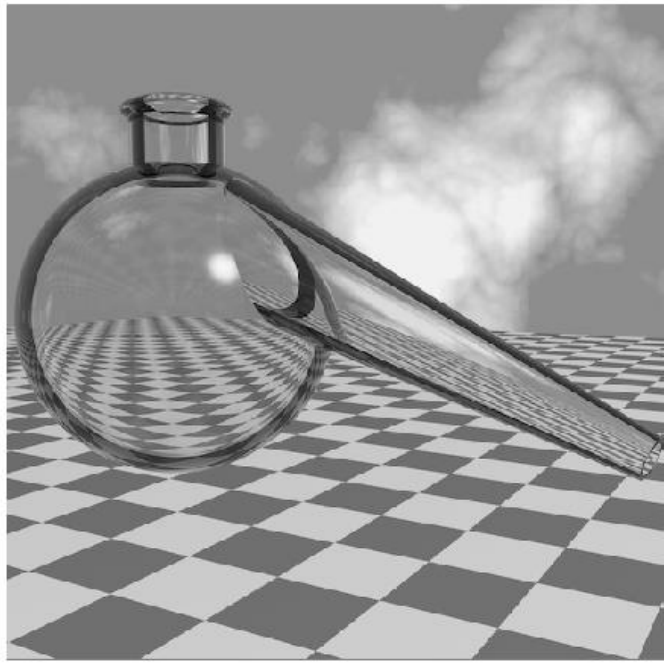
$$\mathcal{F}(f * g) = \mathcal{F}(f) \odot \mathcal{F}(g)$$



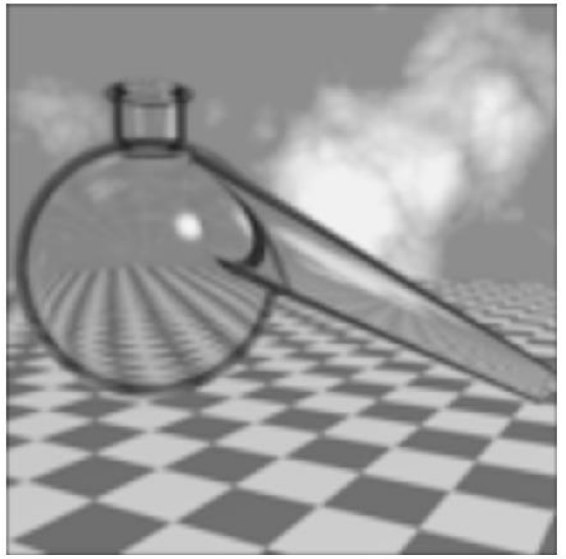
- A Gaussian maintains a compact support in both image and frequency space. Hence, it's more appropriate as a low-band-pass filter.



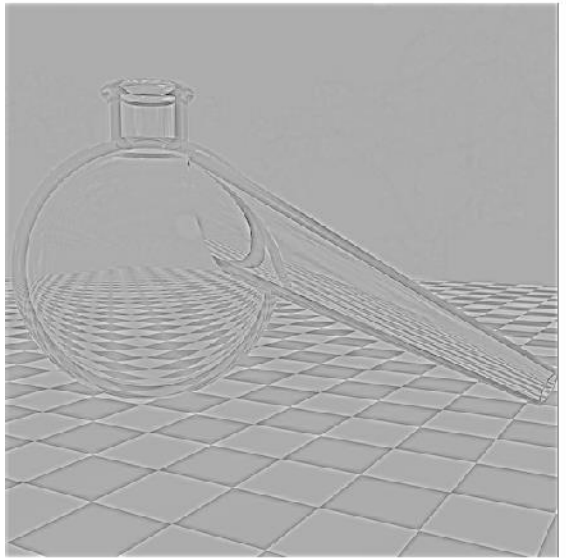
Low-band vs High-band pass filters



Original image

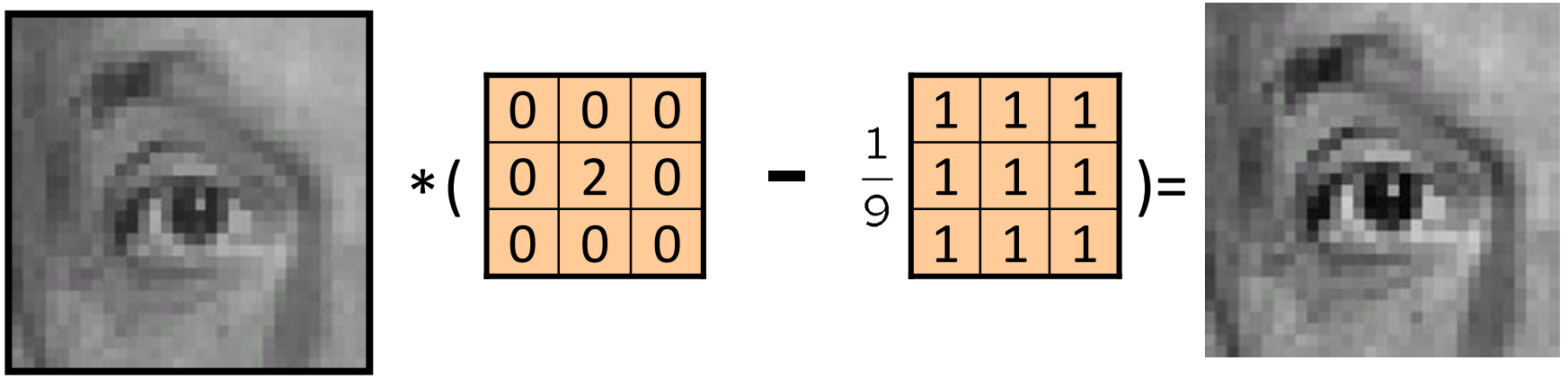


Filtered by a low-band-pass filter



Filtered by a high-band-pass filter

Linear filters in practice



Original

Sharpening filter:

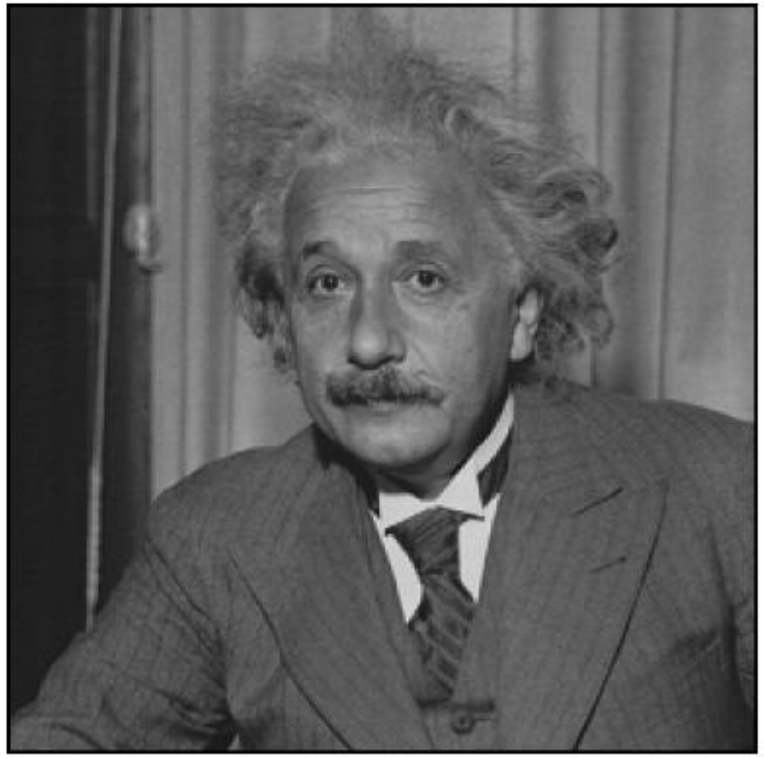
Enhances differences by local averaging.

To explain this, think about what happens in frequency domain.

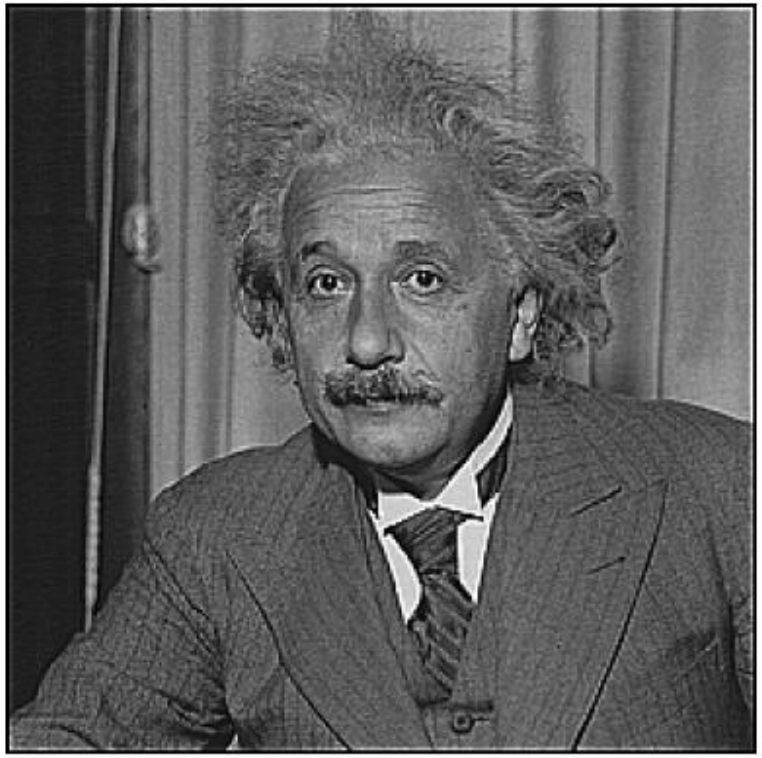
Source: D. Lowe

Sharpening filter

before



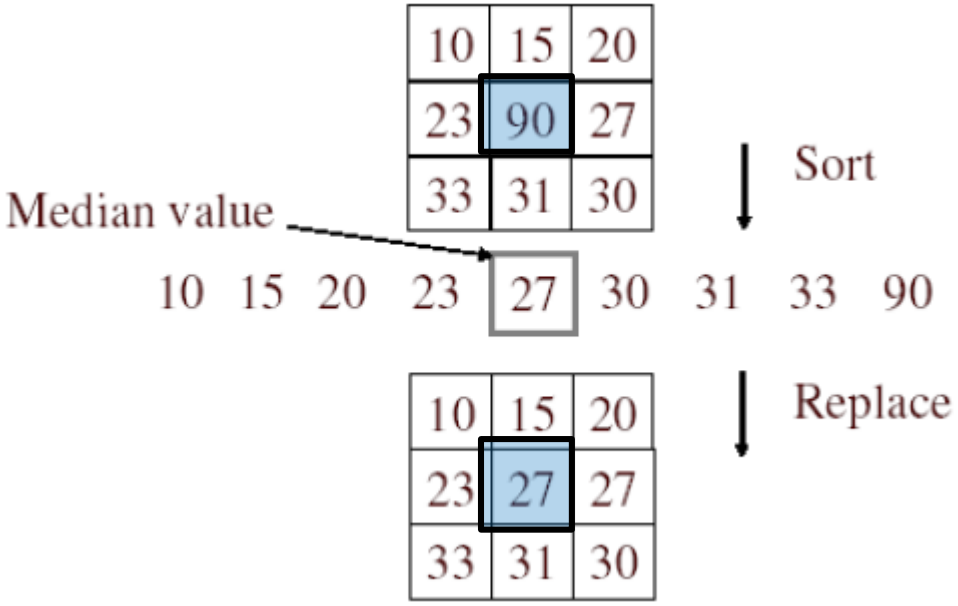
after



To explain this, think about what happens in frequency domain.

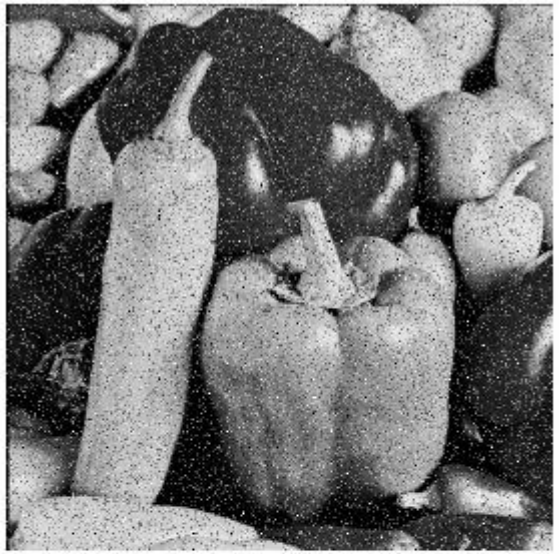
Nonlinear filters: Median filter

- Basic idea
  - Replace the pixel intensity by a median of intensities within a small patch.
- Properties
  - Does not add new gray-levels into the image.
  - Removes impulse noise: appropriate for impulse noise and salt&pepper noise removal.

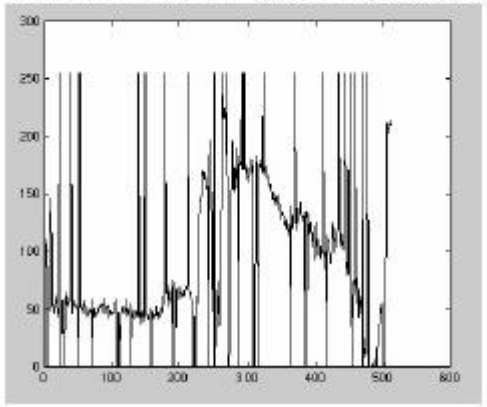


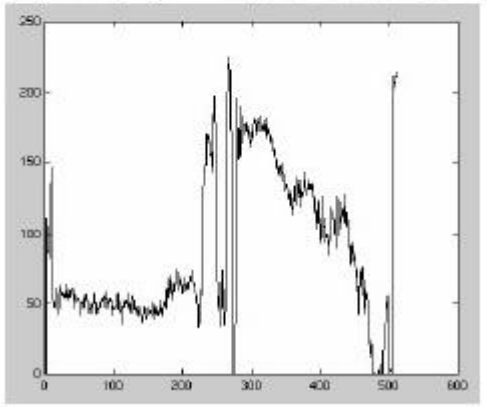
The Median filter

Salt&pepper noise



After median filtering





Plot of a line in the image

Median vs. Gaussian

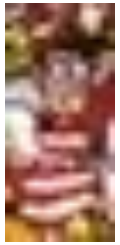
Slide credit: Svetlana Lazebnik

Machine perception

FILTERING AS TEMPLATE MATCHING

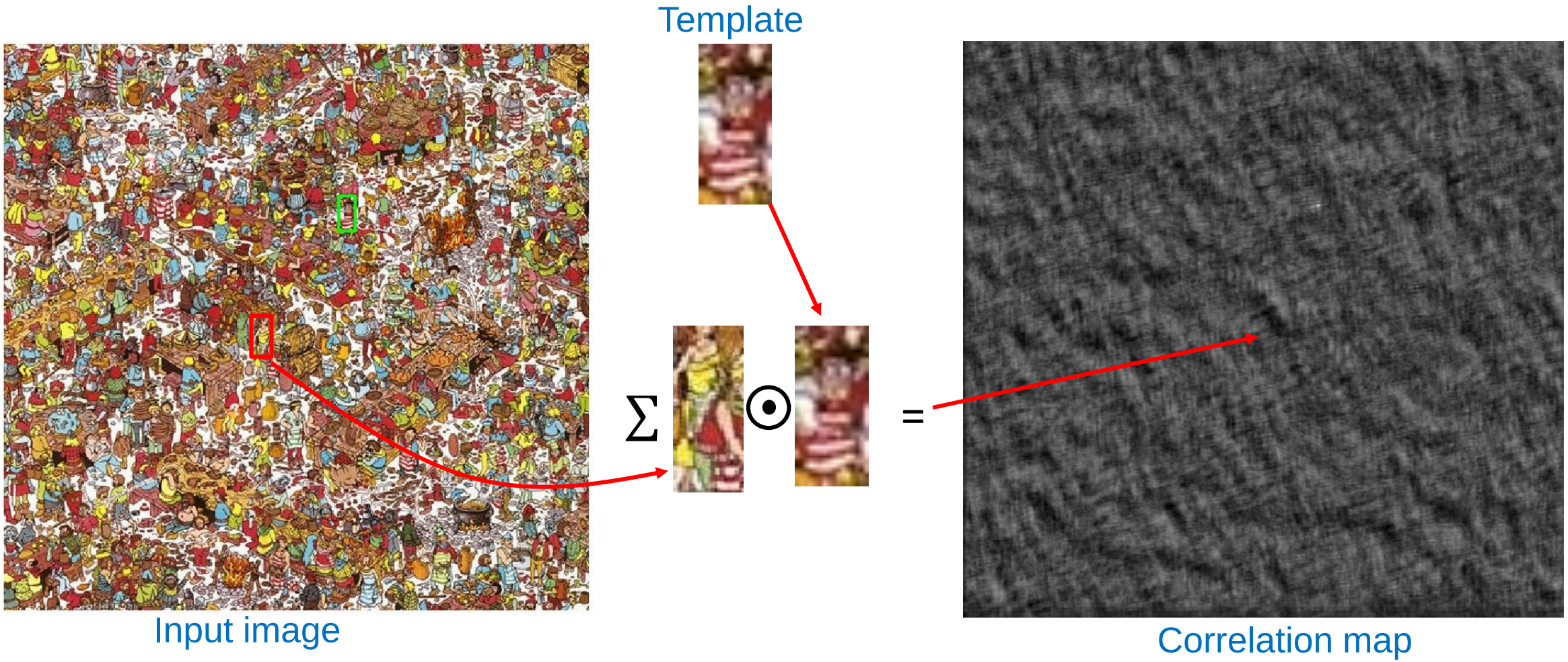
Filtering as template matching

Where's waldo?



Template

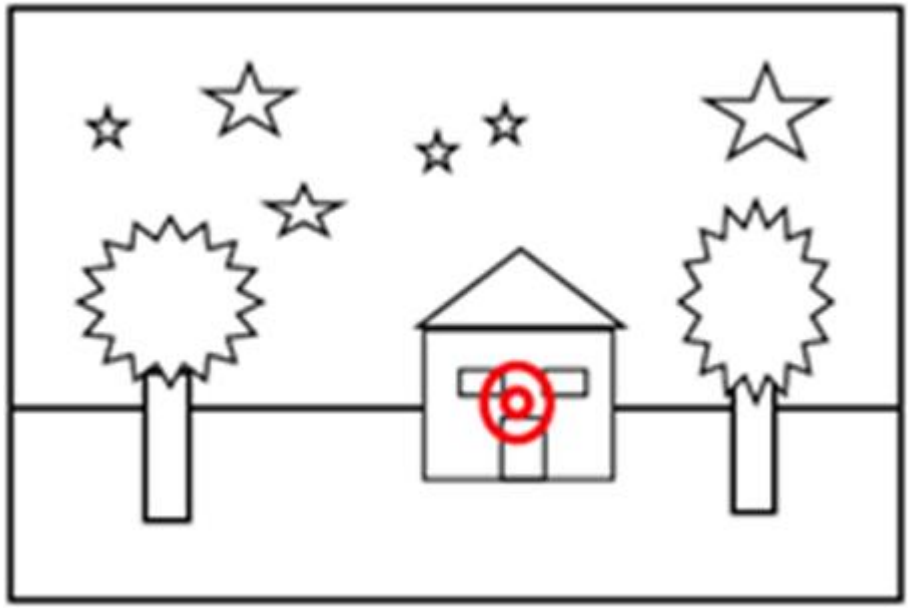
Apply correlation with template



- By the way, this can be done *really* fast by Fast Fourier transform (FFT).

Issues with template matching over scales

- But the object may be bigger/smaller in the image!
- Well, we could carry out correlation for different scales of the template...



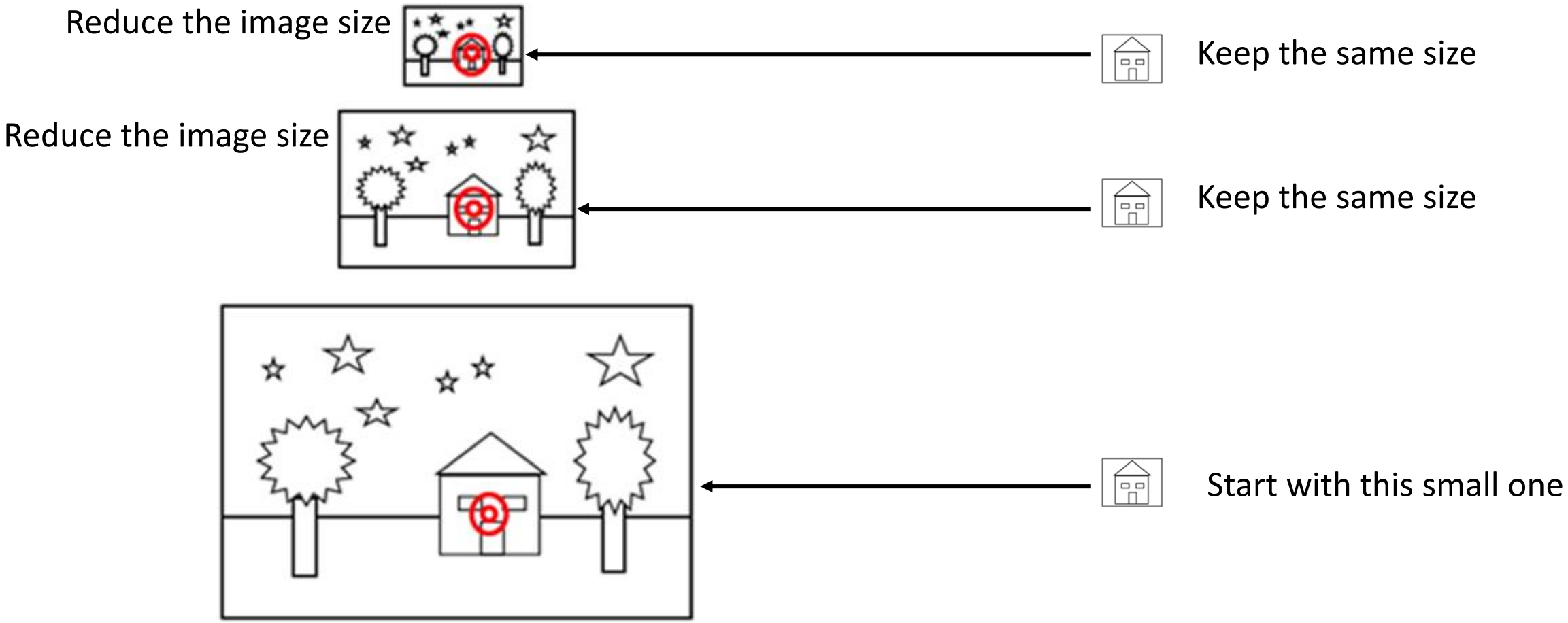
Start with this small one

Then with this one

etc. …

Template matching in scale space

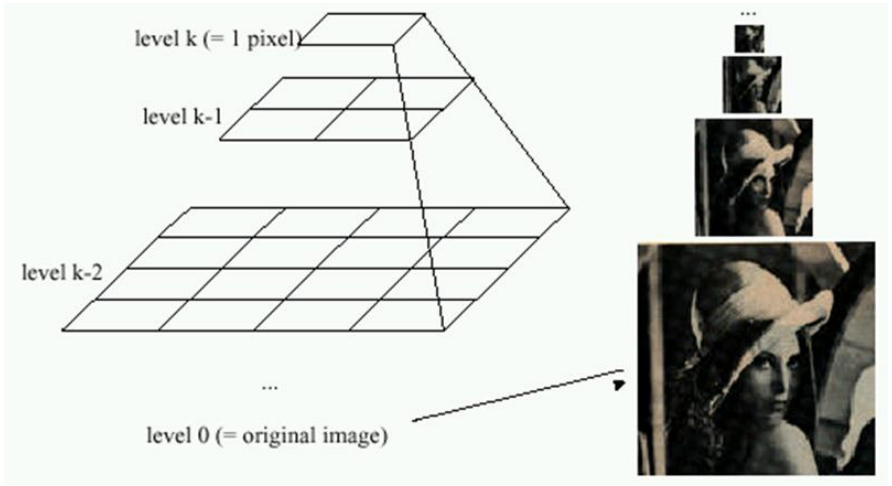
- But rather than template, we scale the input image



Efficient resizing: Image pyramids

Low resolution

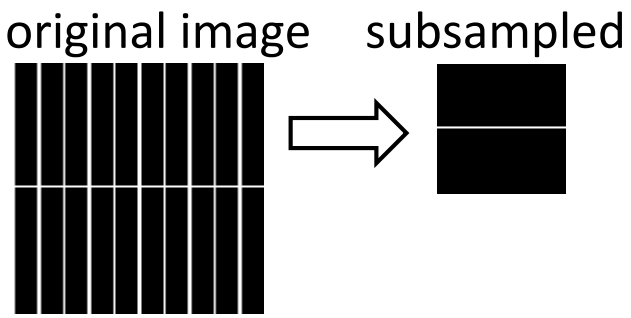
High resolution



Reduce (resample) the image!

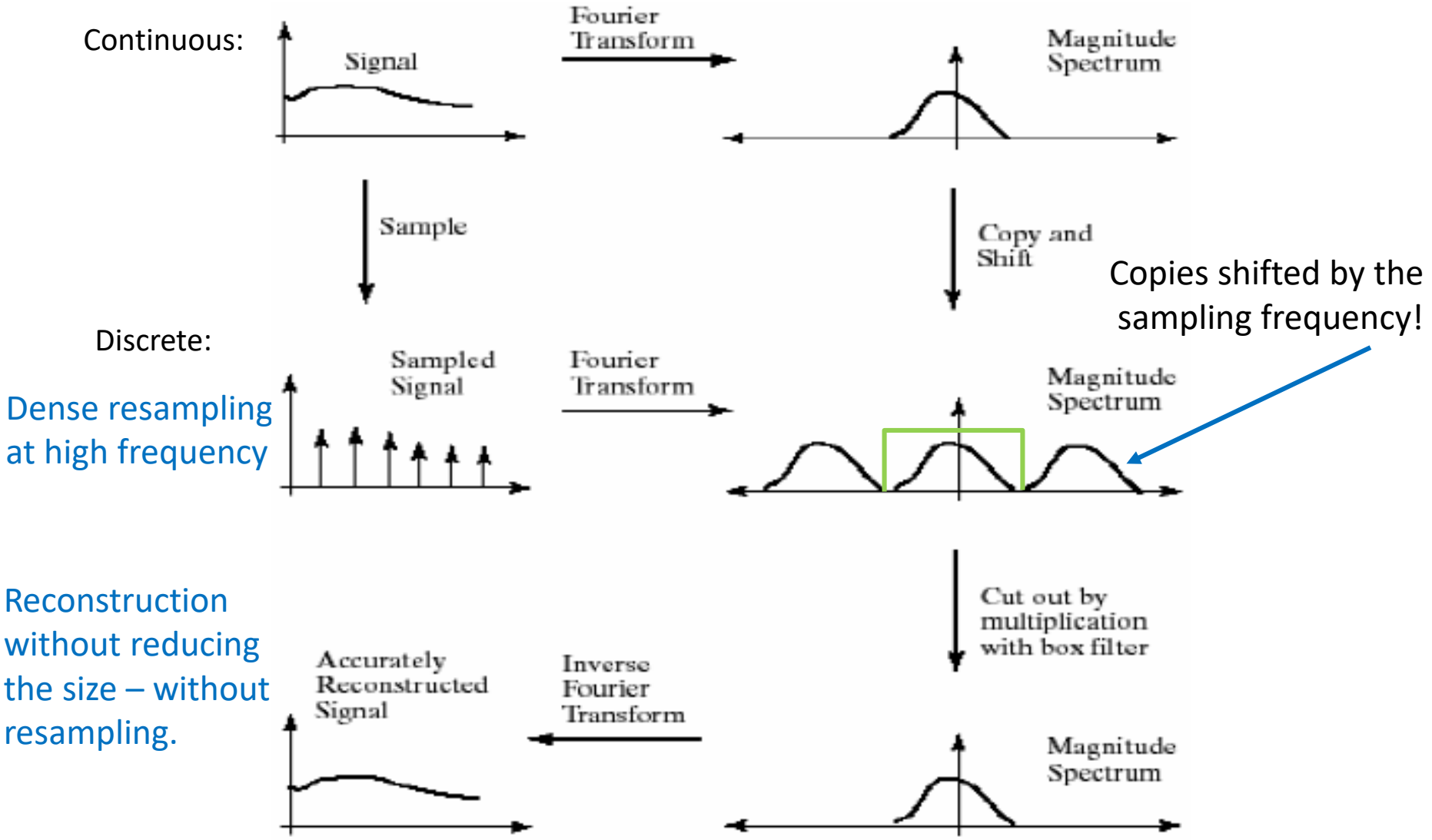
How do we reduce an image?

- Naive:
  - Remove every second pixel...



  - Problem: *the structures in image change!*
- This effect is called *Aliasing.*
- Let's look into frequency domain to explain this…

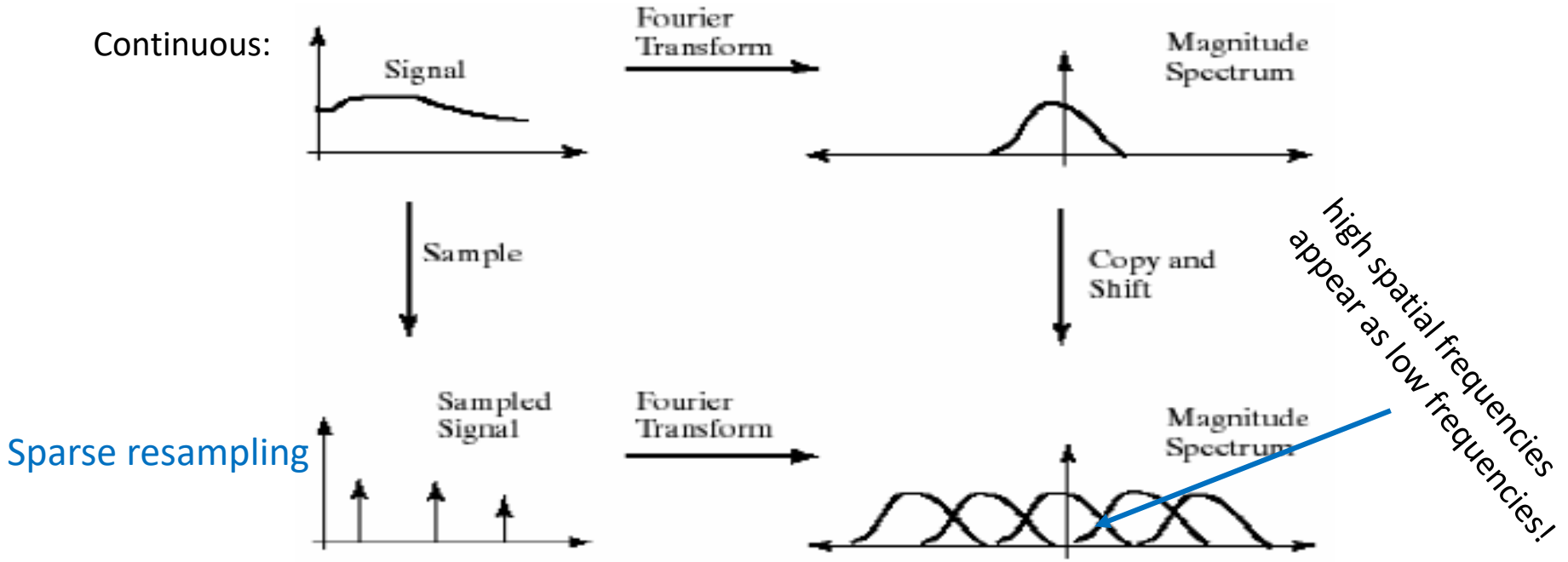
Sampling and aliasing (slo., prekrivanje)



See Forsyth, Ponce: Computer vision, a modern approach.

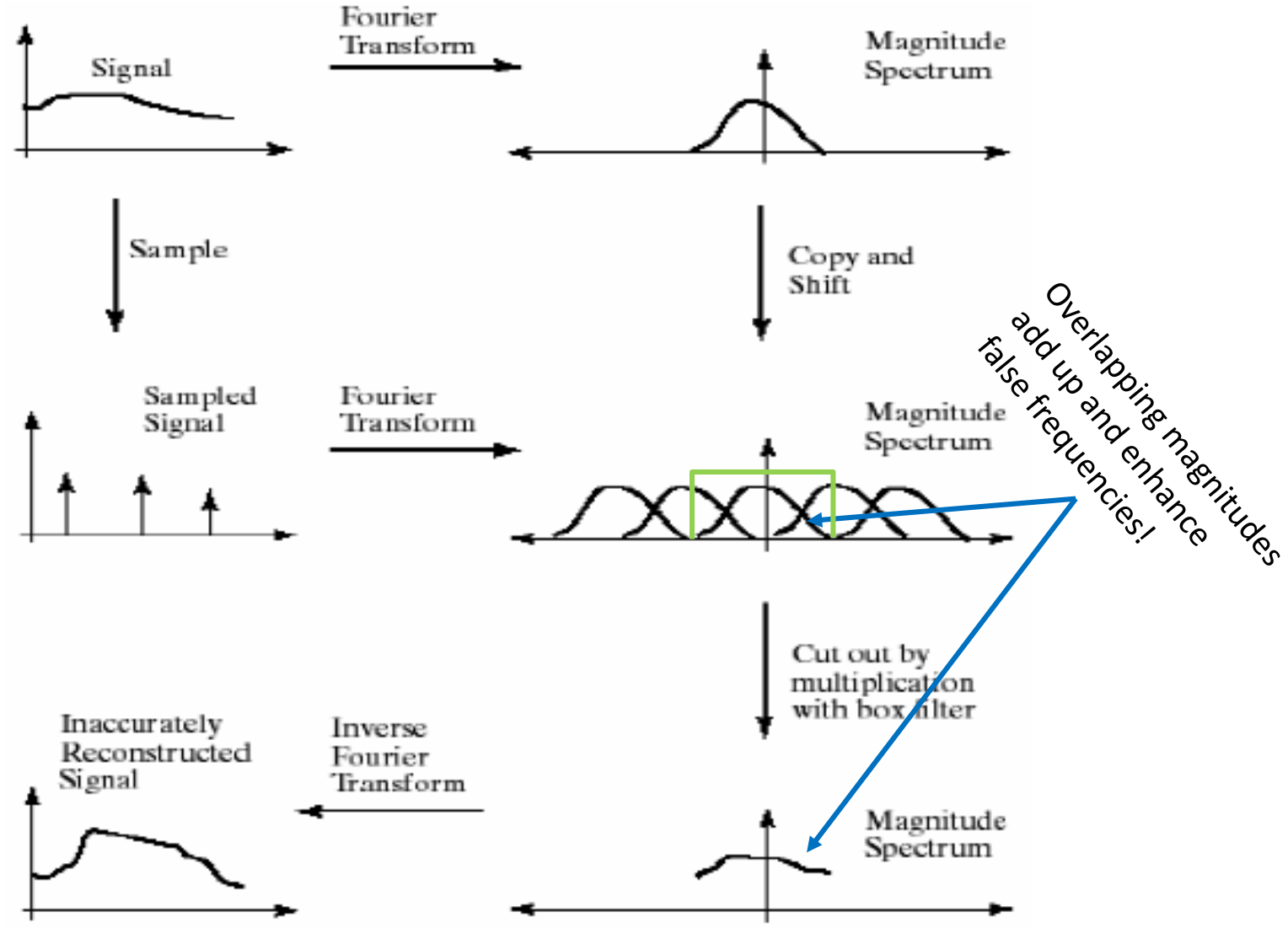
Images from: Forsyth & Ponce

Sampling and aliasing (slo., prekrivanje)



- Nyquist theorem:
  - If we want to reconstruct all frequencies up to *f*, we have to sample the signal by at least a frequency equal to *2f*.
  - This corresponds to a point after which repeated frequency spectra start to overlap (aliasing)!

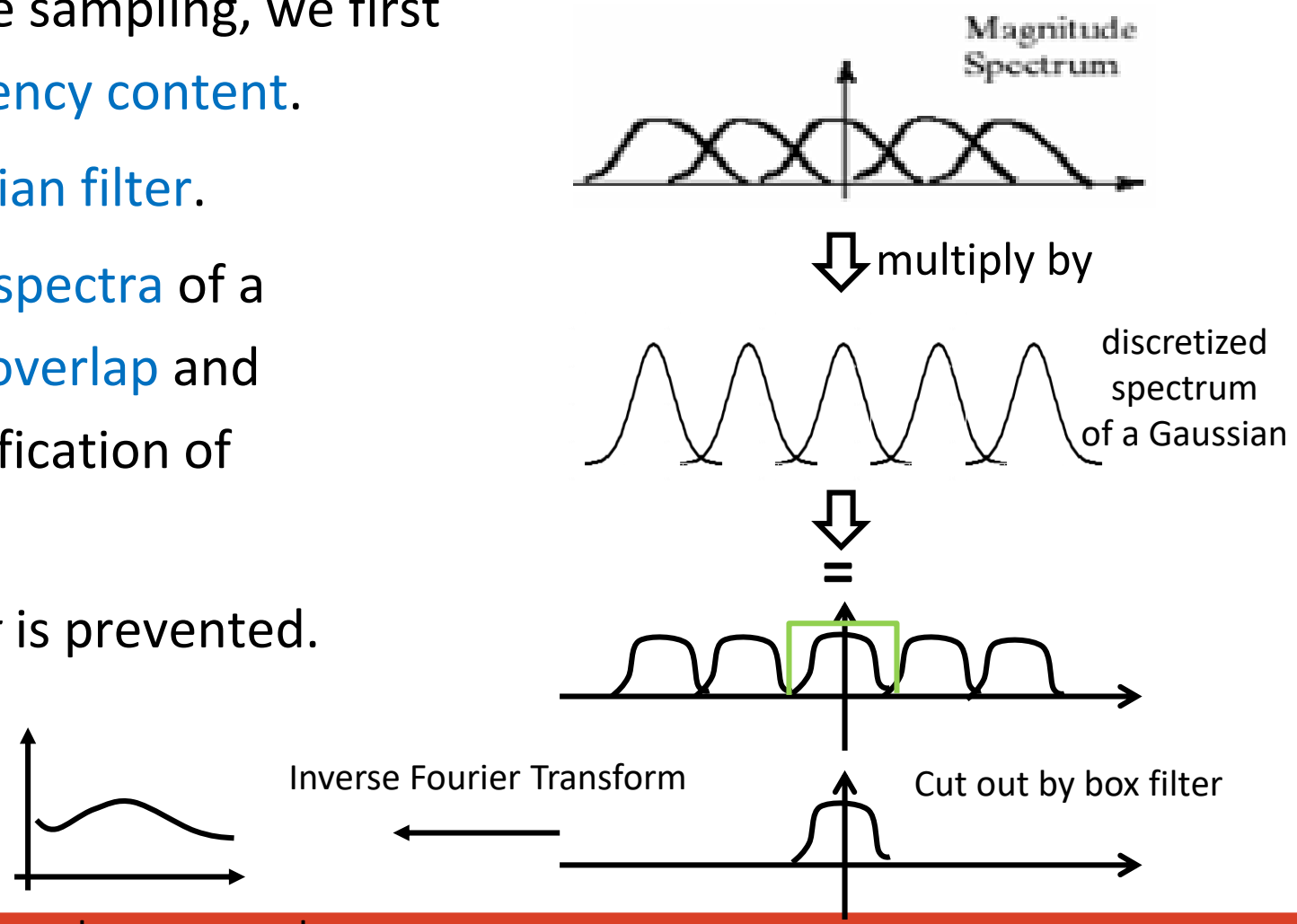
Sampling and aliasing (slo., prekrivanje)



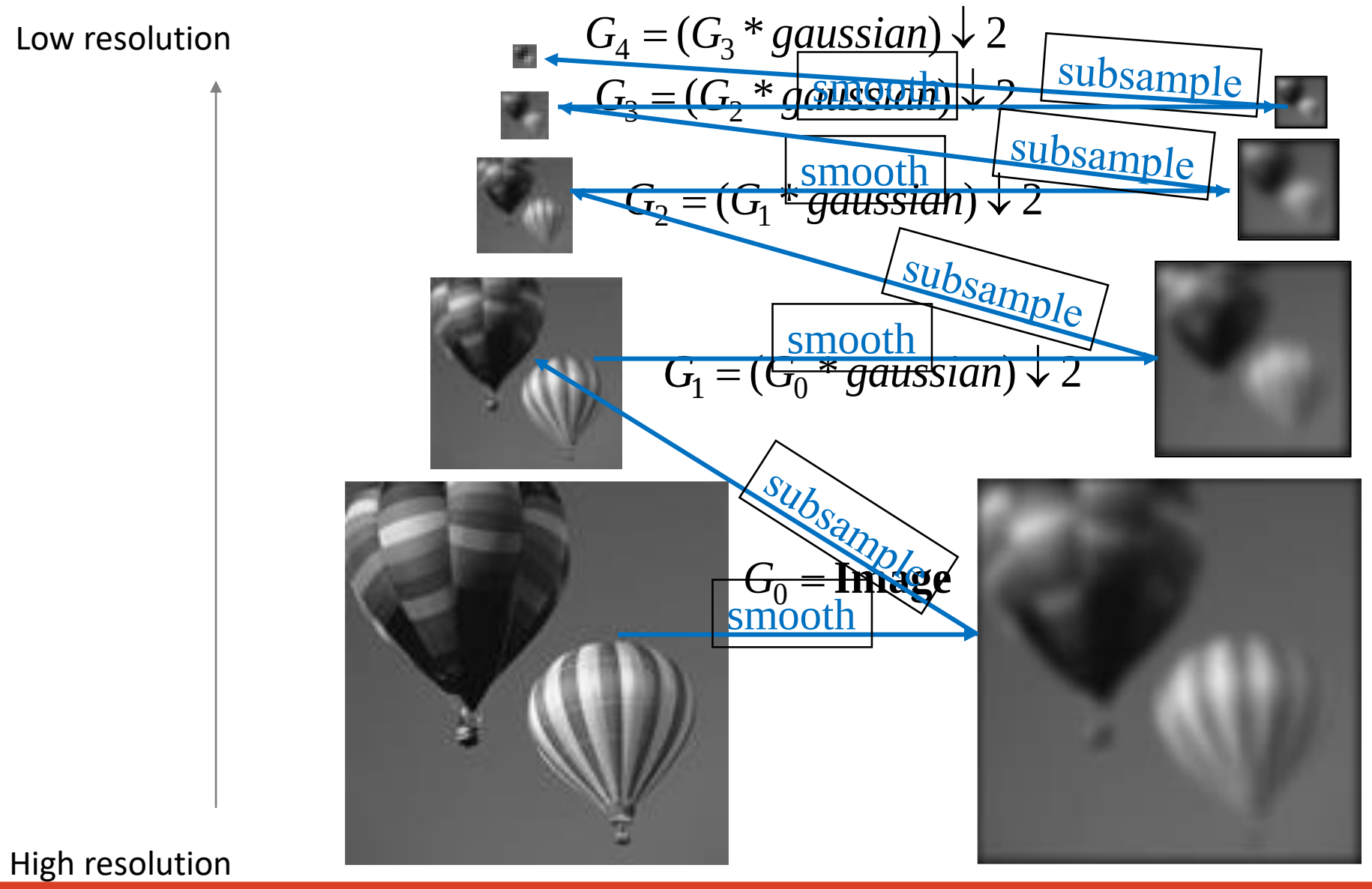
Sampling and aliasing (slo., prekrivanje)

Proper resampling:

- When applying a sparse sampling, we first remove the high frequency content.
- This is done by a Gaussian filter.
- This way the repeated spectra of a sampled signal do not overlap and do not introduce amplification of false frequencies.
- Thus the *aliasing effect* is prevented.



Gaussian pyramid



Source: Irani & Basri

Summary: Gaussian pyramid

- Construction: get a new level directly from the previous
  - Smooth by a small filter and resample
- Reasons for Gaussian smoothing...
  - Convolution Gauss*Gauss = new Gauss
  - $G(\sigma_1^2) * G(\sigma_2^2) = G(\sigma_1^2 + \sigma_2^2)$
- Reason for size reduction...
  - Gaussian is a low-band-pass filter, so we get a redundant representation of a smoothed image.

$\Rightarrow$ No need to store a smoothed image in full resolution.

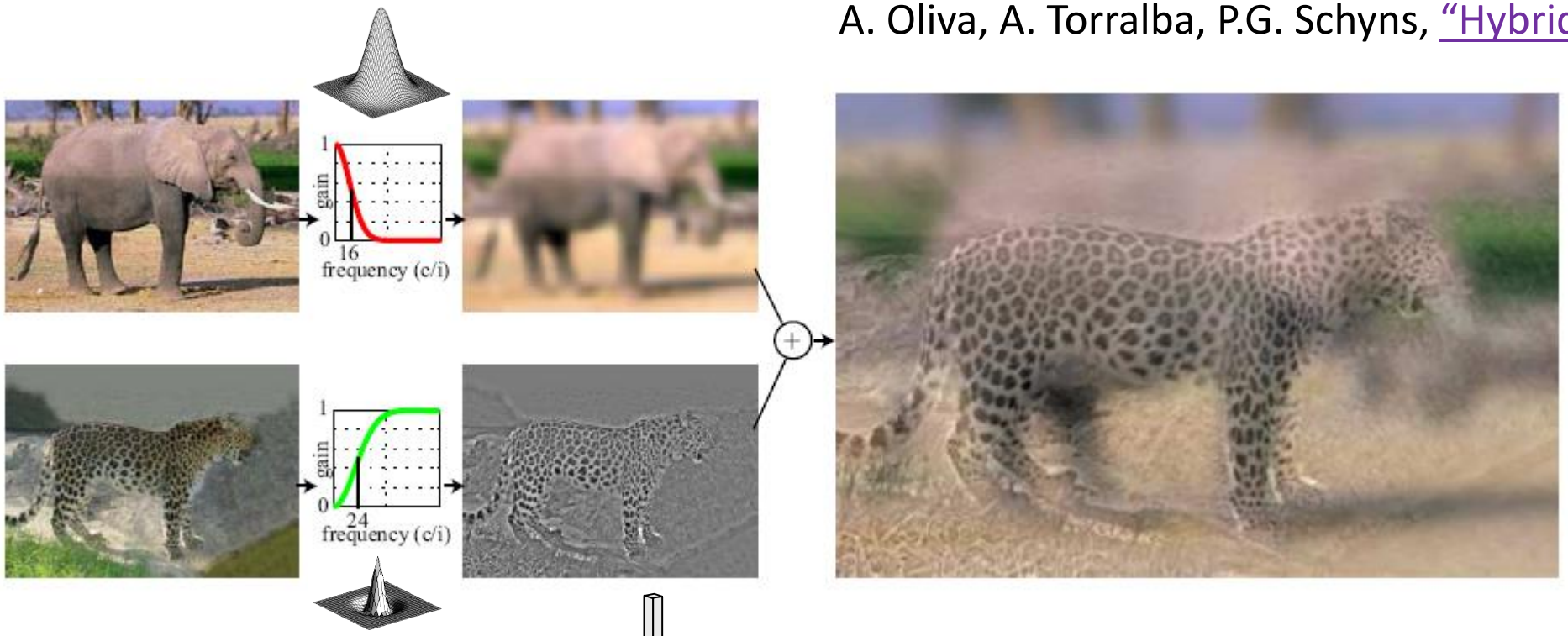
Why a pyramid?

- Enables efficient implementation of many detection methods
- Multi-scale object detection ...
- Multi-scale edge detection ...
- Multi-scale feature point detection ...
- Manipulation of selected frequency bands ...
- Old stuff: Scale space

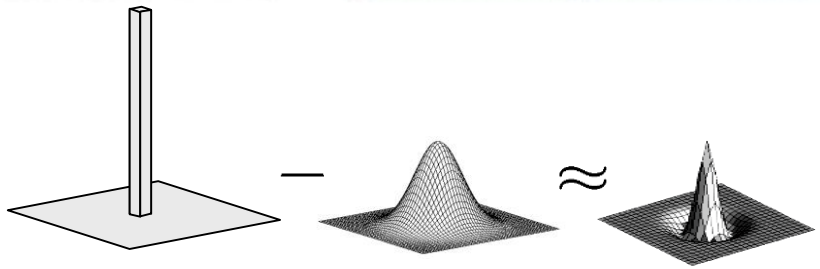
Fun with hybrid images...

Gaussian filter

A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006



Laplacian filter:



Unit impulse — Gauss ≈ Laplacian of Gaussian

References

- David A. Forsyth, Jean Ponce, Computer Vision: A Modern Approach (2nd Edition), (prva izdaja dostopna na spletu)
  (Ozadje linearnih filtrov in povezavo s Fourierjevim transformom najdete v Poglavjih 7 in 8)
- R. Szeliski,Computer Vision: Algorithms and Applications, 2010
- Kristen Grauman, „Computer Vision“, lectures