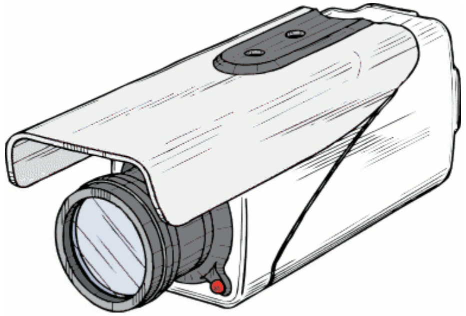
Univerza v Ljubljani

Machine perception
Image processing 2

Matej Kristan



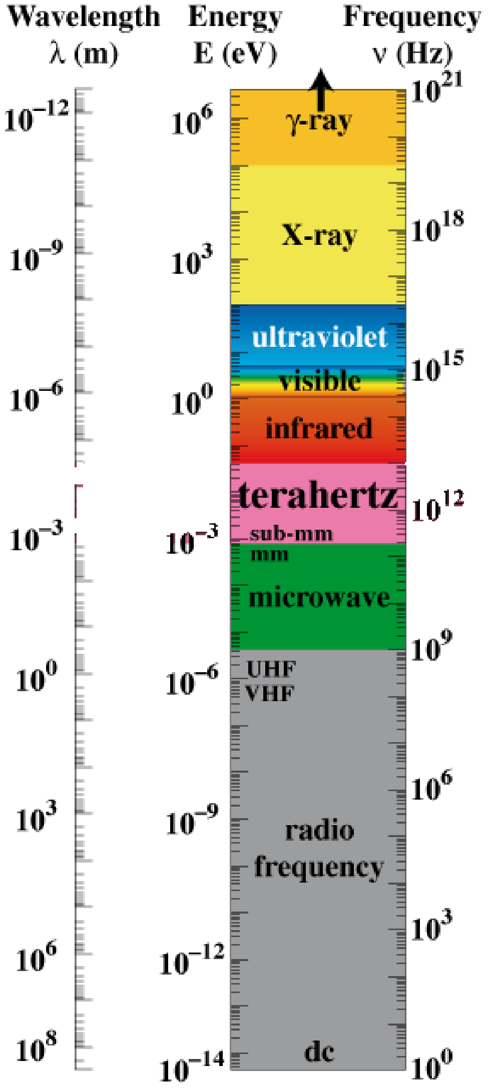
Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

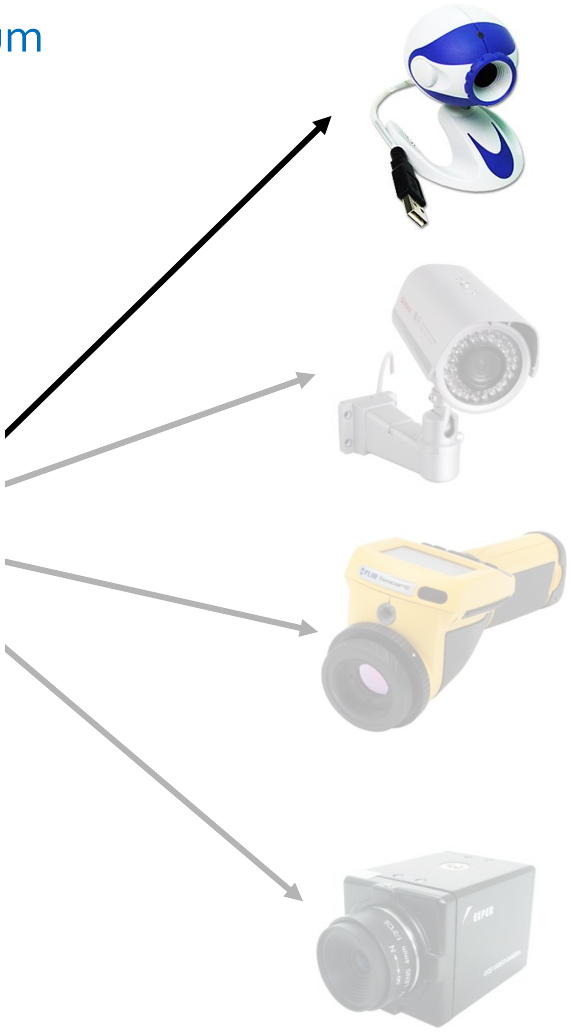
Machine perception

COLOR

Sensor: Camera

Electromagnetic spectrum

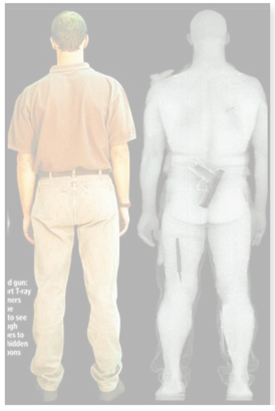




Visible light

Near-infrared light

Far-infrared light



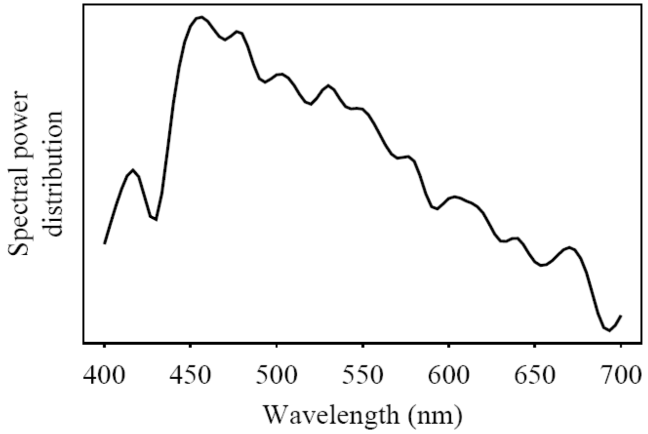
Terahertz light

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)

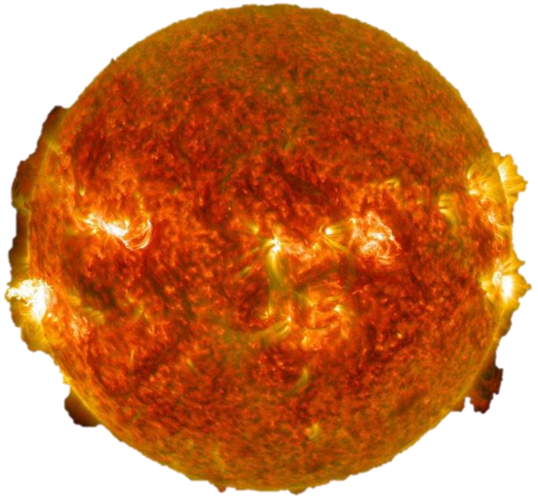
  Energy radiated in a unit of time w.r.t. wavelength
- E.g., laser light contains only a narrow band of wavelengths (frequencies)

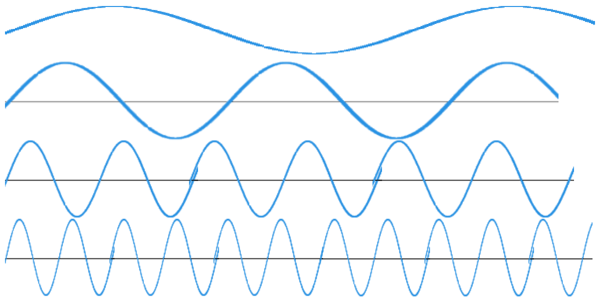
- Visible light contains radiation with wavelengths in interval 400-700nm

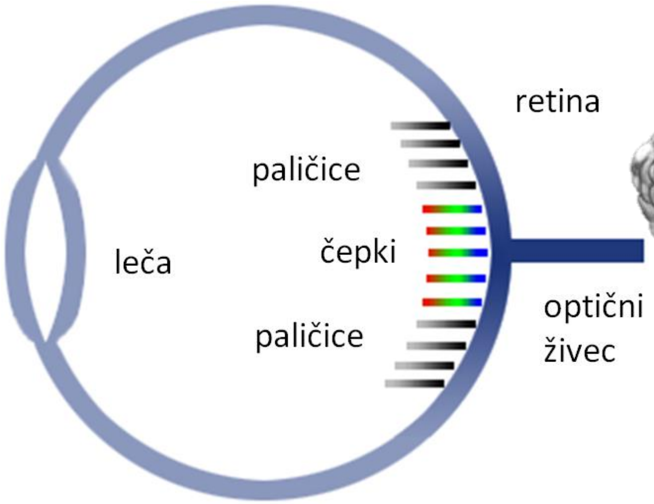


Human color perception

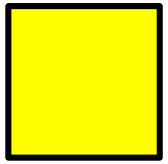
- Human eye (retina) contains specialized cells that react to different wavelengths differently.
- Three types of cells called “cones”: R, G, B
- A type of cells called “rods”: intensity only





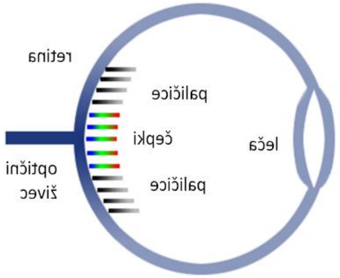


Yellow



Additive mixture model

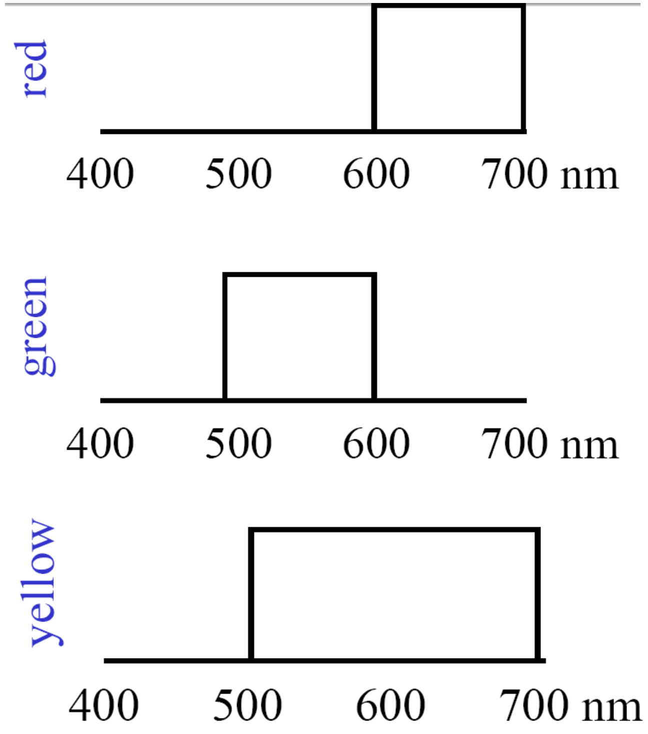
- What color do we get if we shine a red and green light?



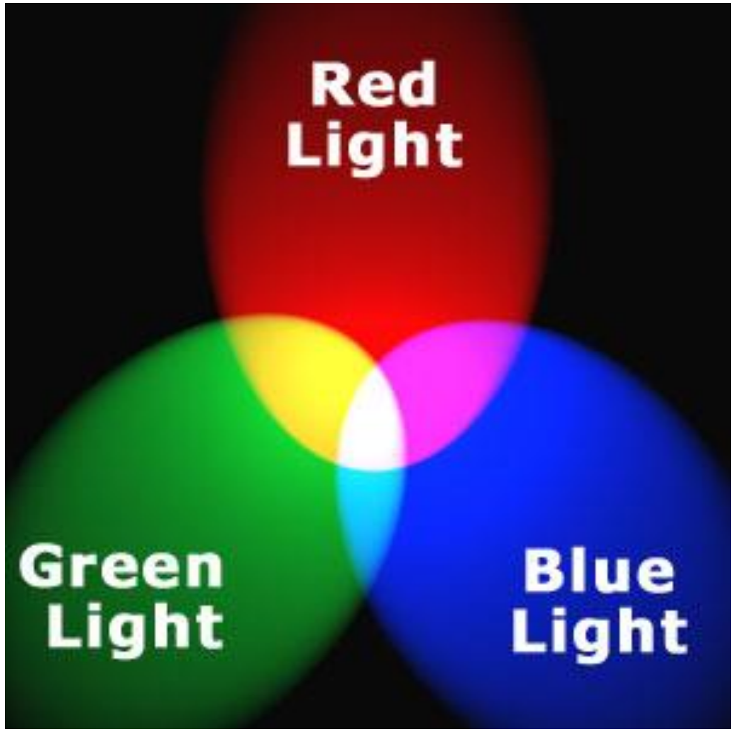
perceive

perceive

perceive



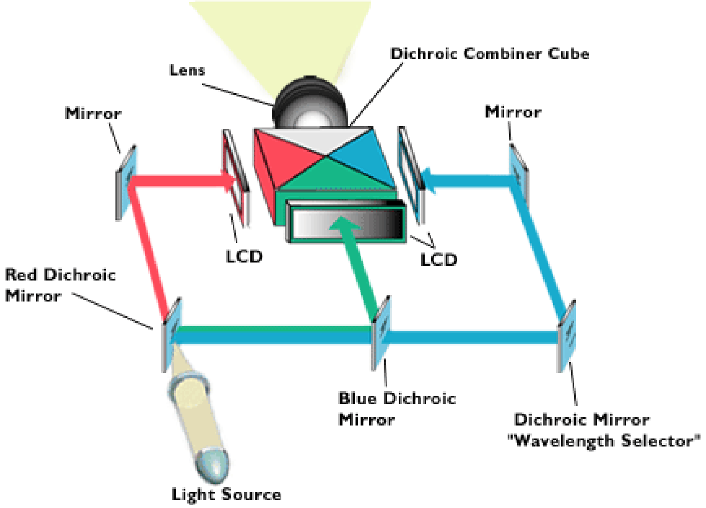
colors mix by summation of their spectra.



colors *added to black*.

Systems using the additive model

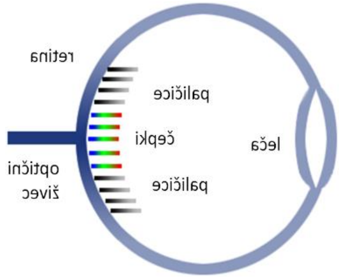
Monitors

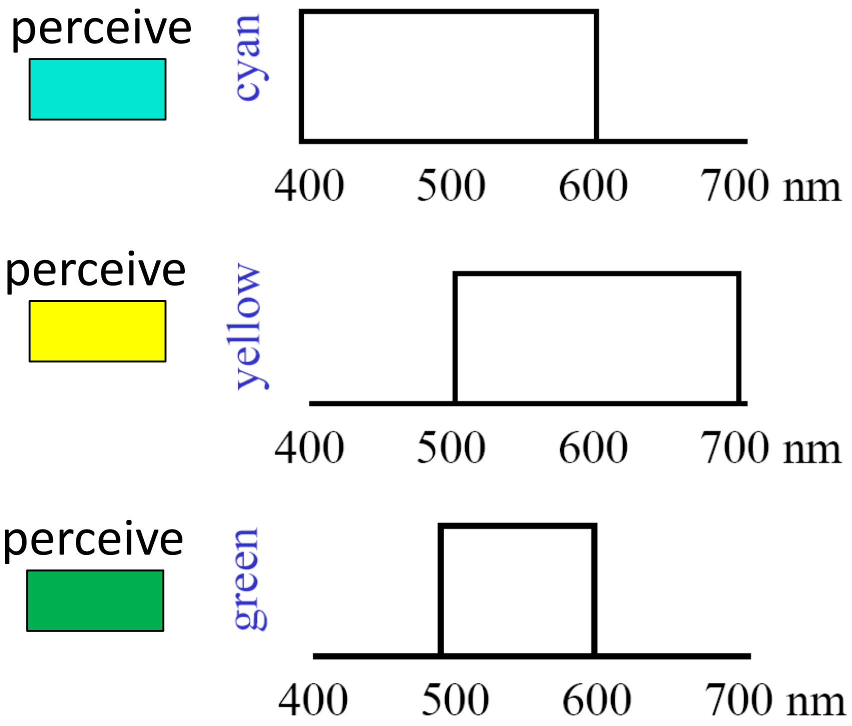


LCD projector

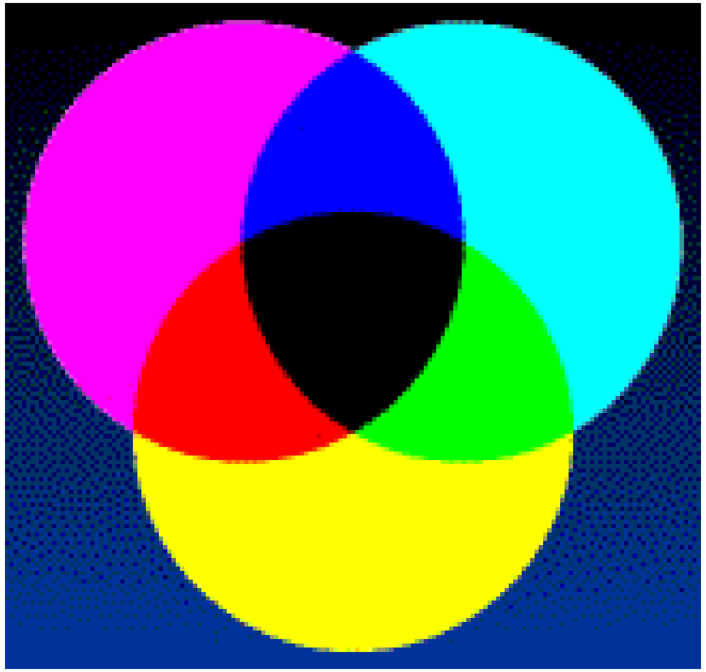
Subtractive models

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





colors mix by *spectra multiplication*.



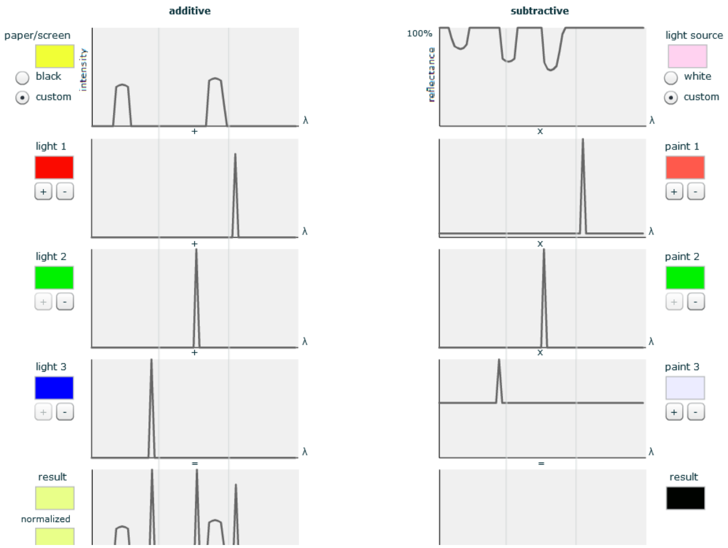
Pigments *remove* the color from the incident white light.

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

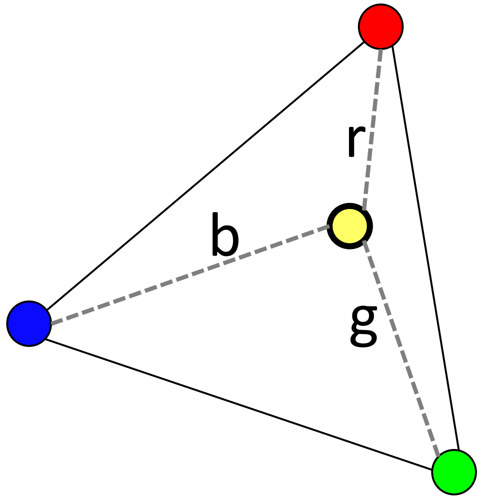


Color spaces

- 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)

- A new color is a weighted sum of primaries



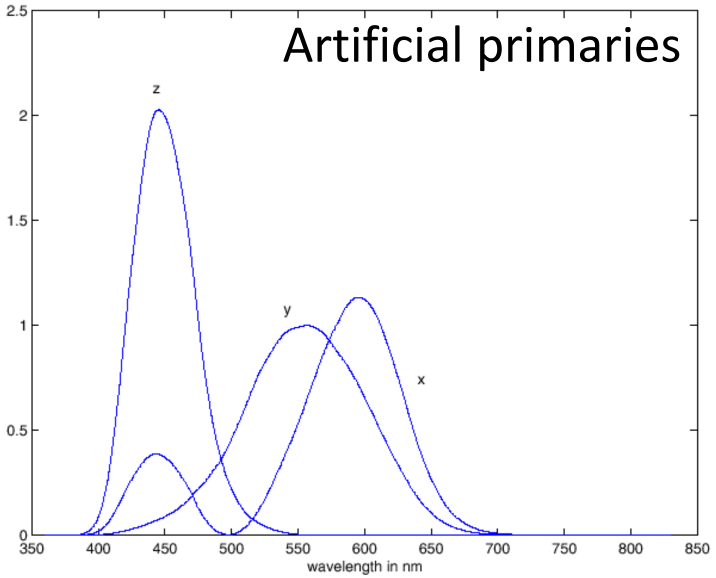
By mixing the colours, we get any colour that lies within the triangle of primaries.

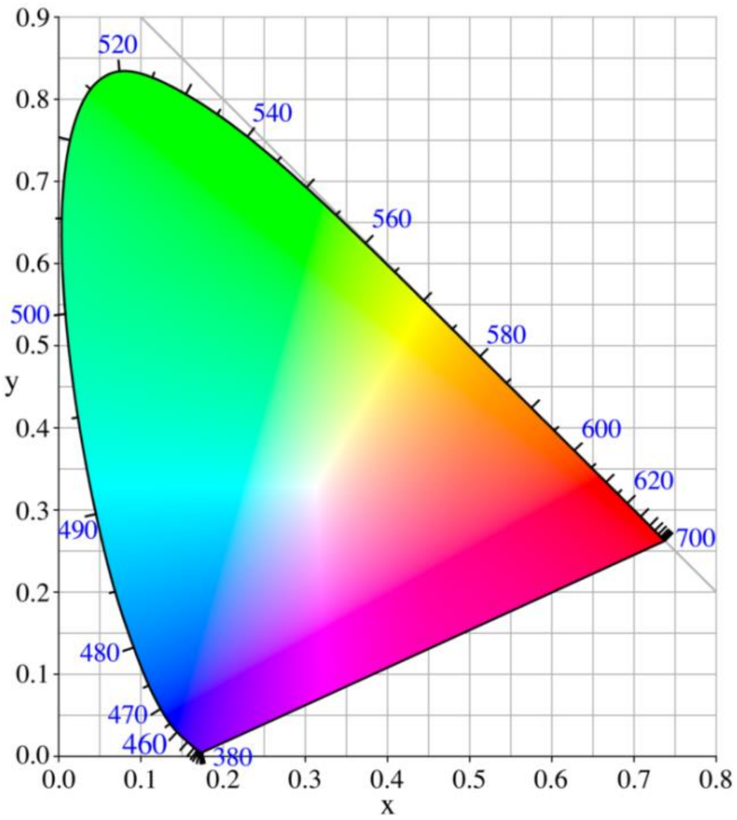
- Mixing weights *r,g,b* to get any color were estimated on human subjects

Linear color space example: CIE XYZ

- International Commission on Illumination (Commission international d'eclairage -- CIE), 1931

A nice overview available: https://www.sciencedirect.com/topics/engineering/color-matching-function





- Representation by chromaticity only [x,y]:

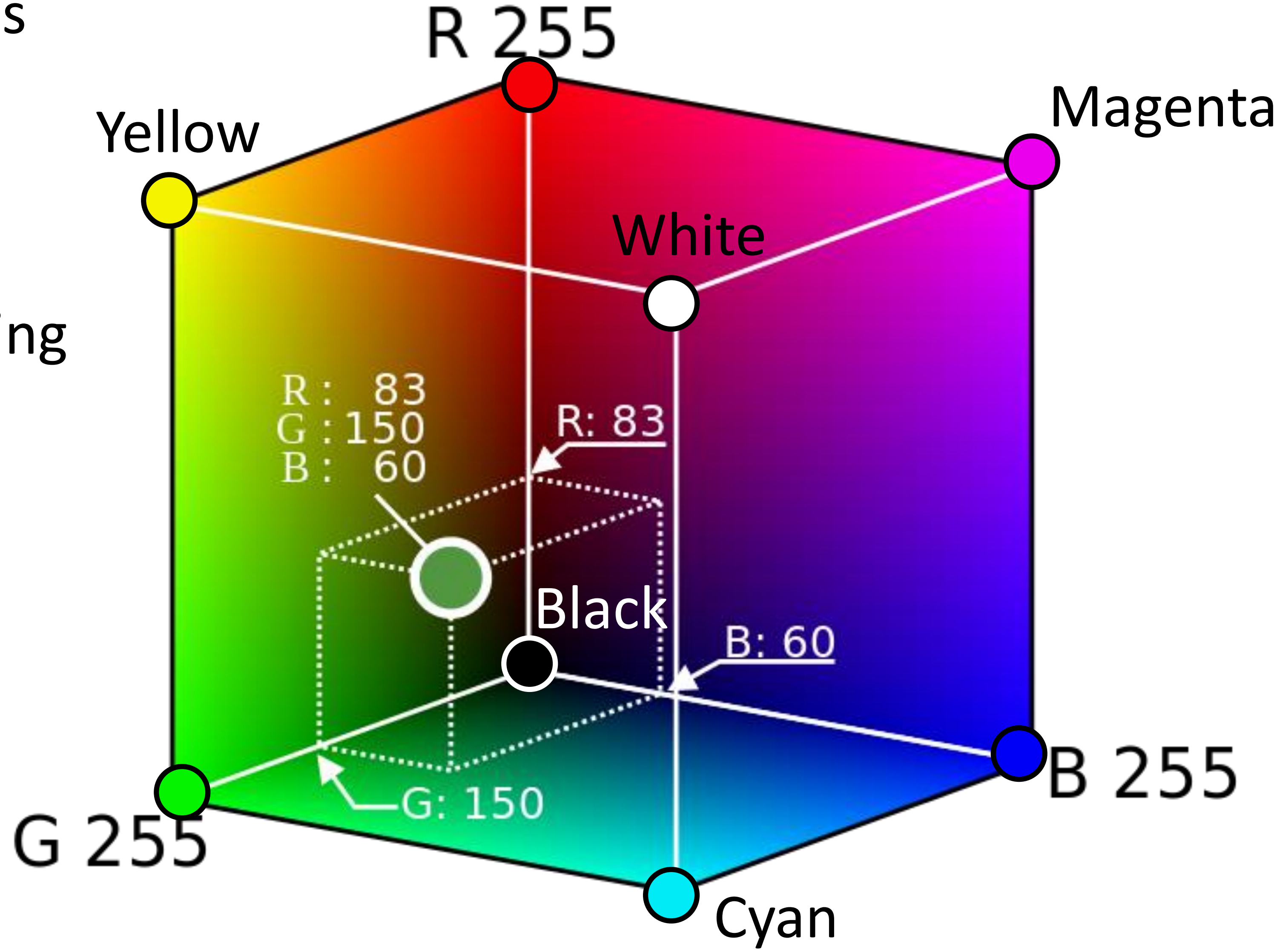
$$x = \frac{X}{X+Y+Z},\ y = \frac{Y}{X+Y+Z},\ z = \frac{Z}{X+Y+Z}$$

$$x + y + z = 1$$

Linear color space example: RGB

- Single wave-length primaries

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

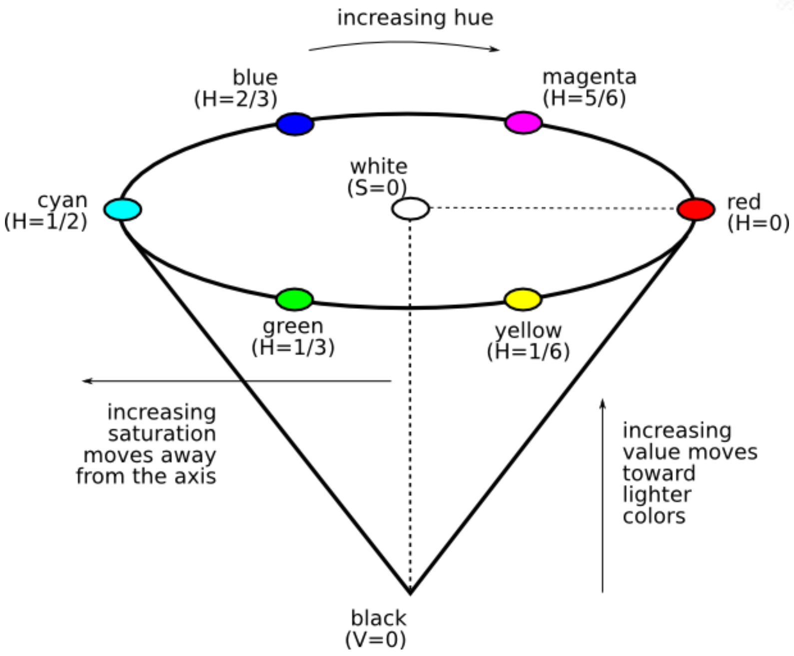


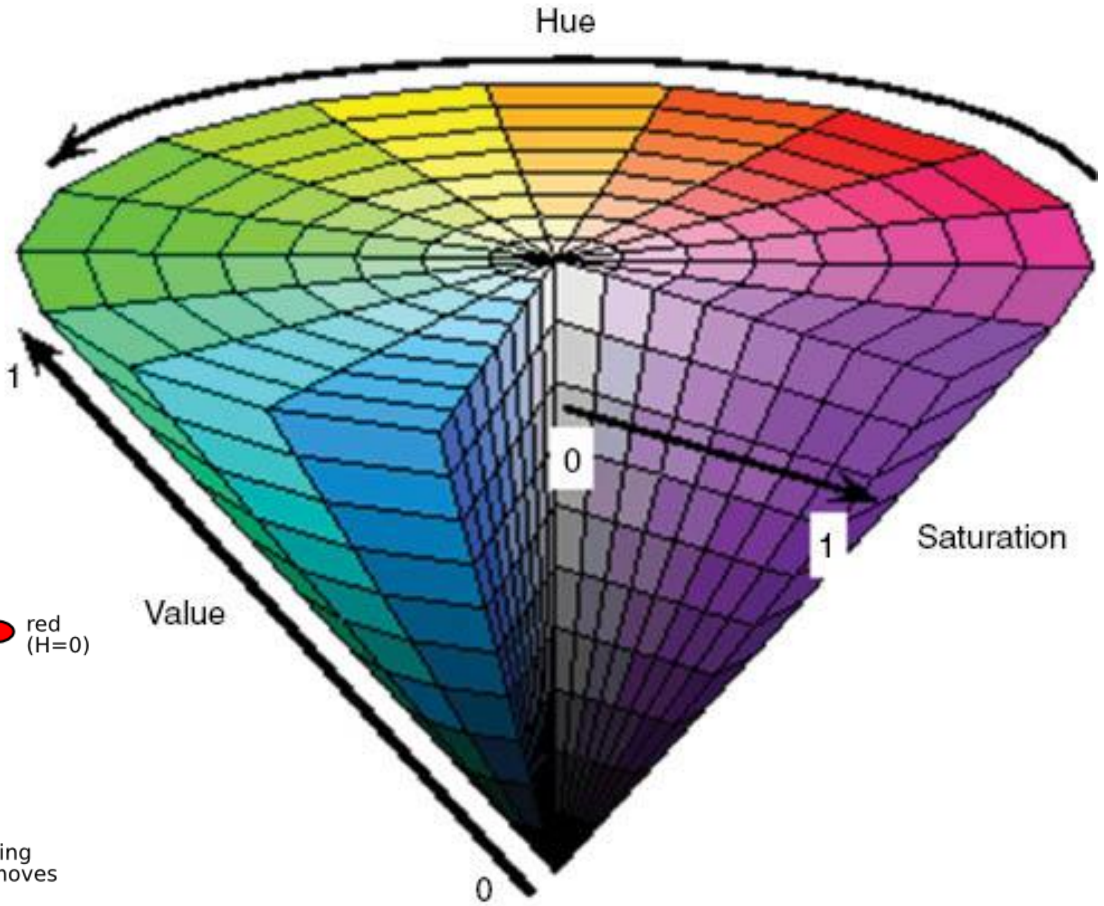
HSV colorspace

- Hue (barvnost), Saturation (nasičenje), Value (intenziteta)
- Nonlinear – hue coded by angle
- Python: cv2.cvtColor(*I*, *Type*)

  Type:

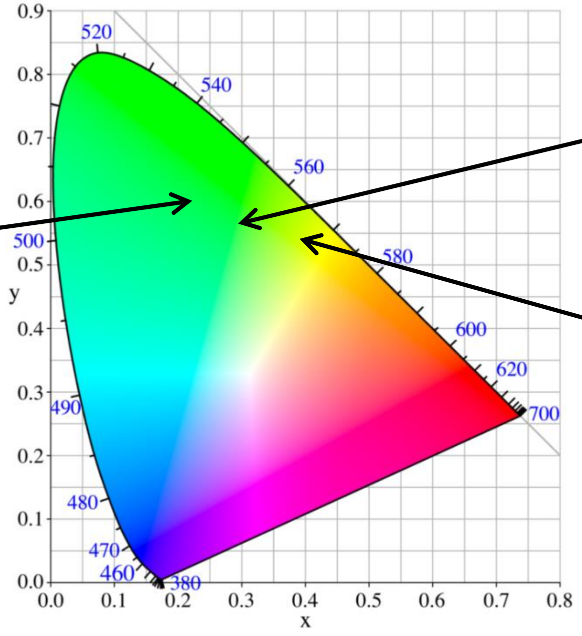
  - cv2.COLOR_RGB2HSV
  - cv2.COLOR_HSV2RGB





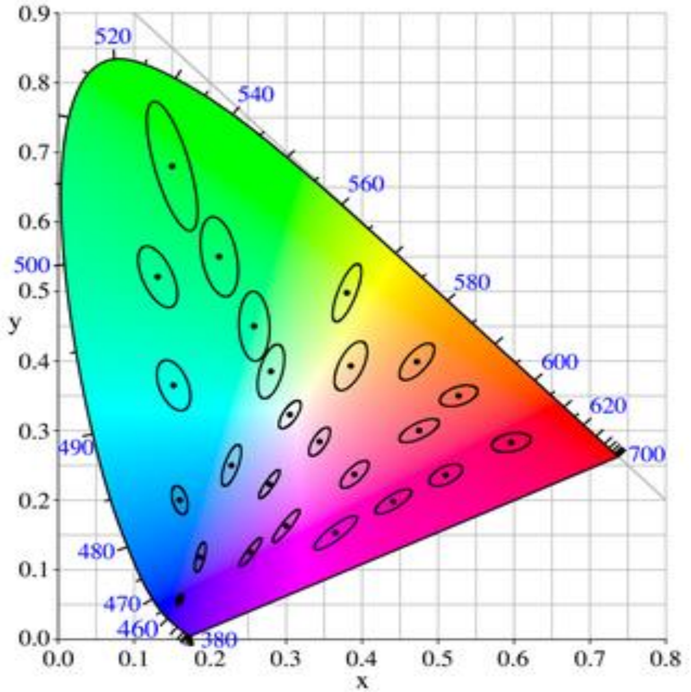
Distances in colourspaces

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



Distances in color spaces

- 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).*



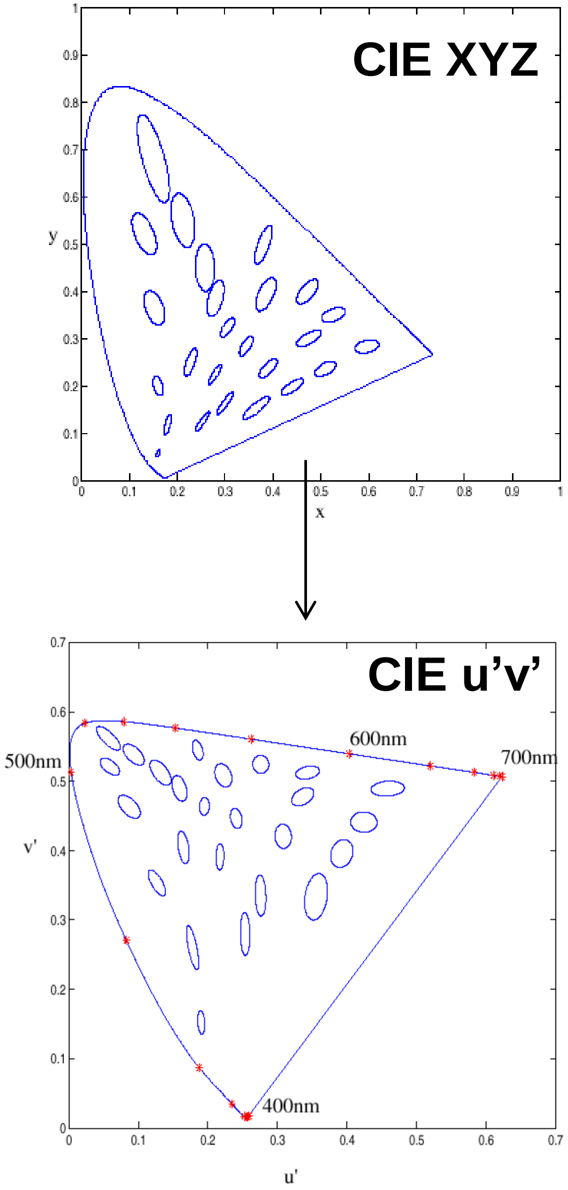
McAdam ellipses:
Just (human) noticeable differences in color

Uniform color 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 (1976)

Nonuniform colour space



Uniform colour space

Computing color similarity between objects

- How to summarize the color?
- Idea1: just compute the average (r,g,b)

$\xrightarrow{\mu_{rgb}}$

$$\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

$$\mathbf{x}_i = (r_i, g_i, b_i)^T$$

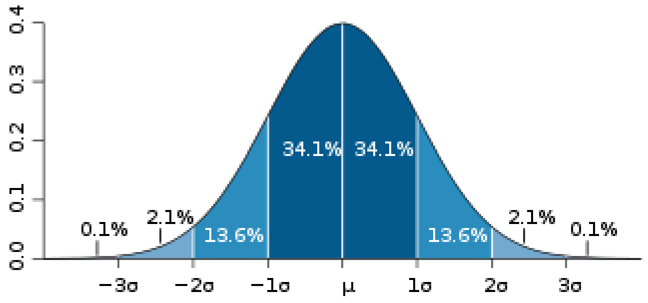
Color of the (r,g,b) at *i*-th pixel

$\xrightarrow{\mu_{rgb}}$

Issue: a single value does not sufficiently capture the color *distribution*

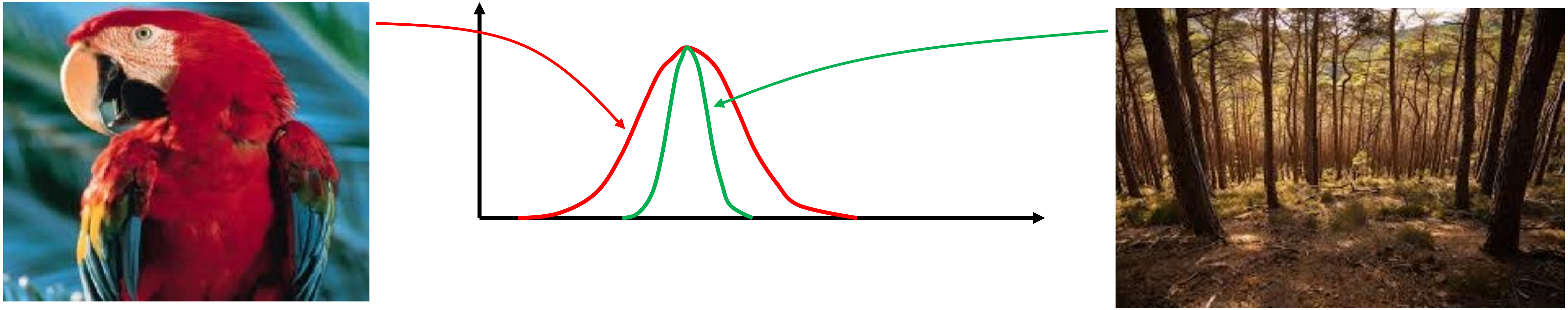
Describe the color by a Gaussian

- Summarize the color by parameters of a Gaussian distribution



$$\mu = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

$$\mathbf{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^T$$



But often a more flexible model of color distribution is required!

Machine perception

COLOR DESCRIPTION BY USING HISTOGRAMS

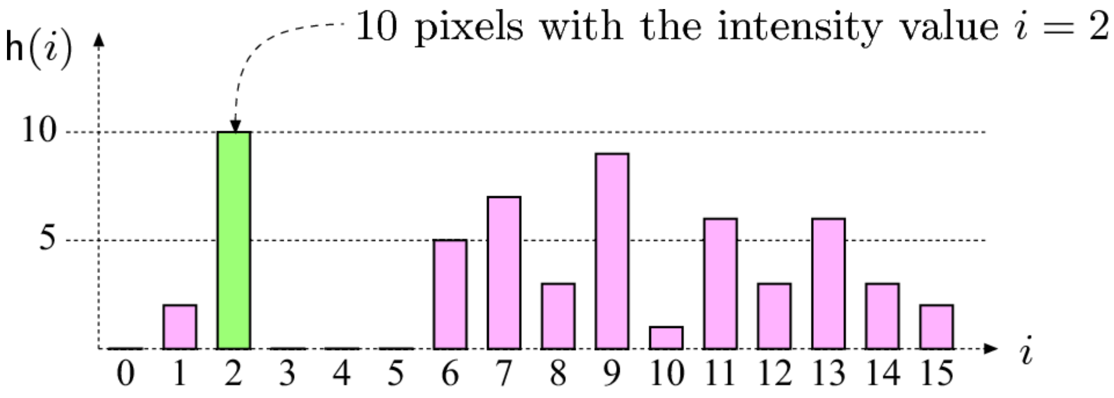
What is a histogram?

- Image histogram records the frequency of intensity levels

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

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

- Example:



$\mathsf{h}(i)$	0	2	10	0	0	0	5	7	3	9	1	6	3	6	3	2
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

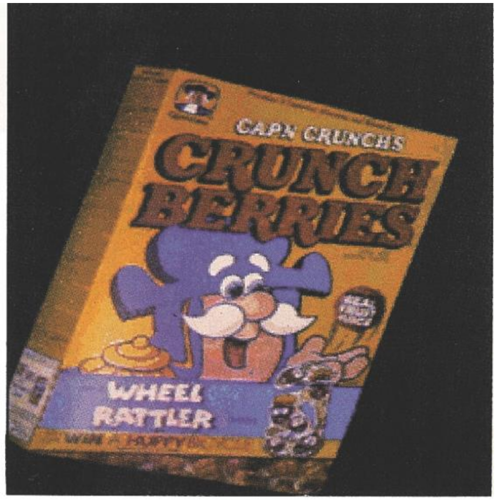
Intensity value

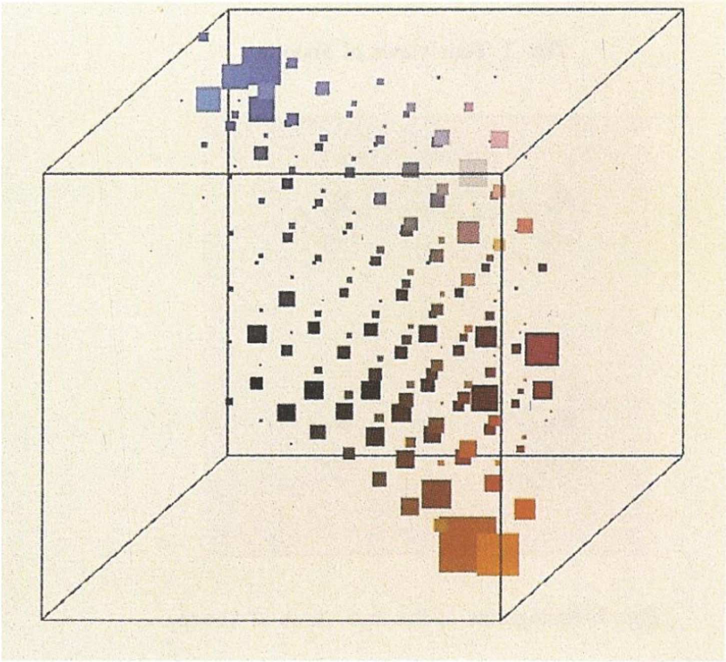
256 intensity levels

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]

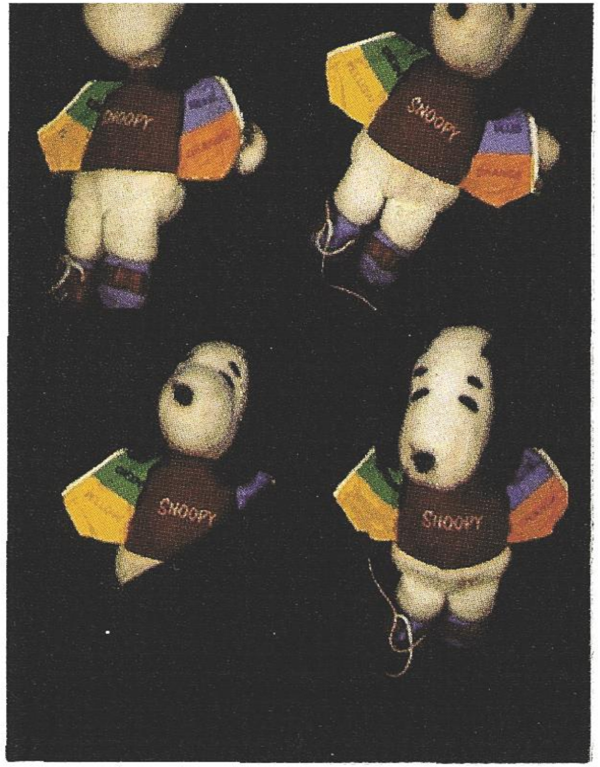


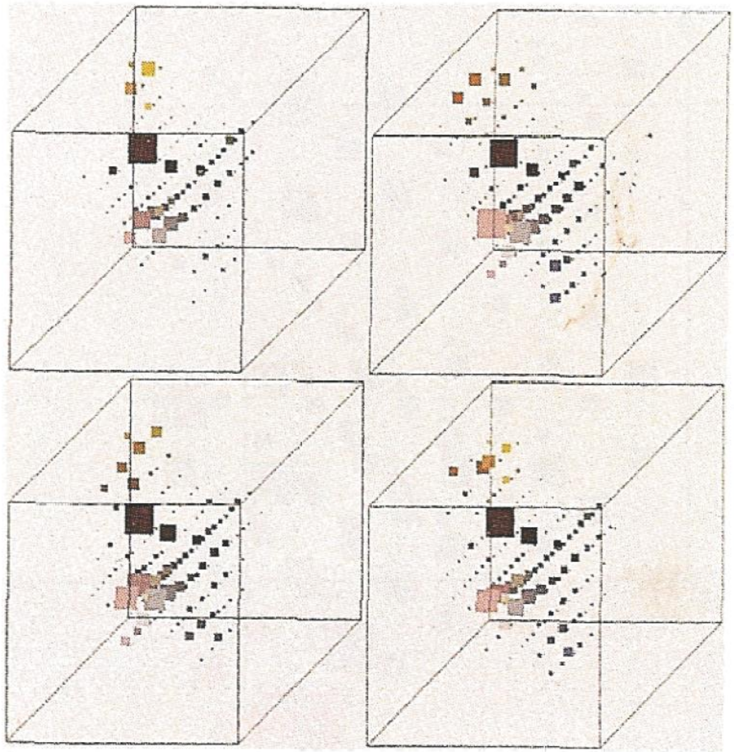


[Swain & Ballard, 1991]

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 the hue („true“ color).
  - This means that we can normalize a color by its intensity.
    - Intensity is defined as: $I = R + G + B$:
  - Chromatic representation:

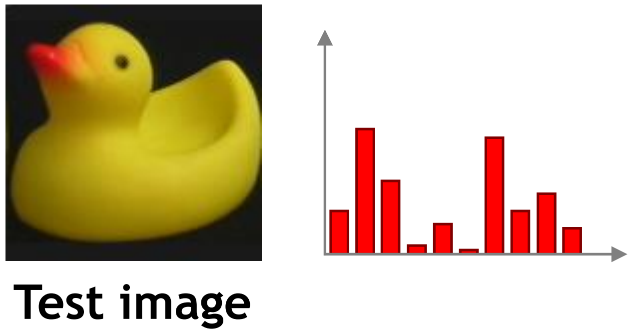
$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B}$$

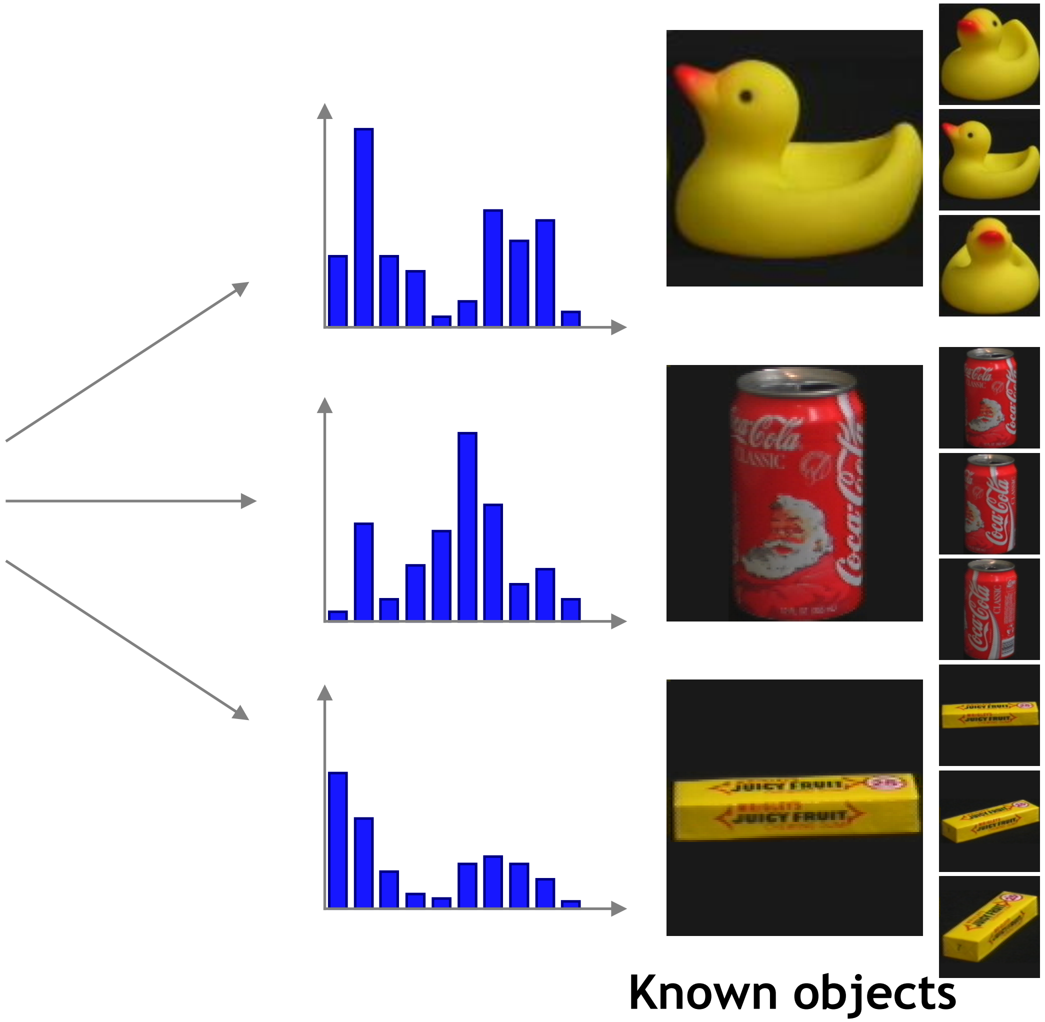
  - We can now use only a 2D space (rg), since it holds that

$$r + g + b = 1$$

Color comparison via histograms

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



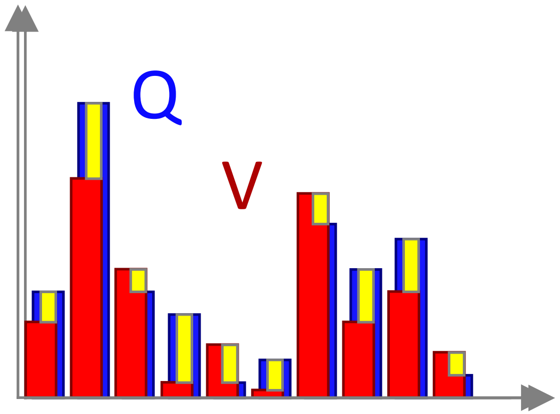


A measure of distance/similarity between the histograms is required!

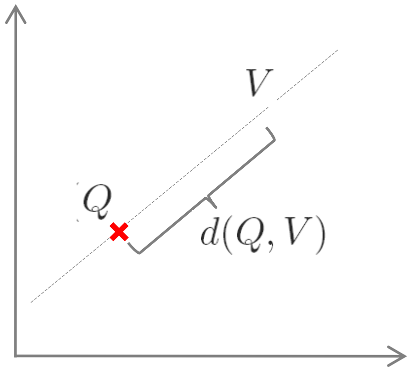
Popular distances: Euclidean distance

- Definition (=L_2 norm)

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



- 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!

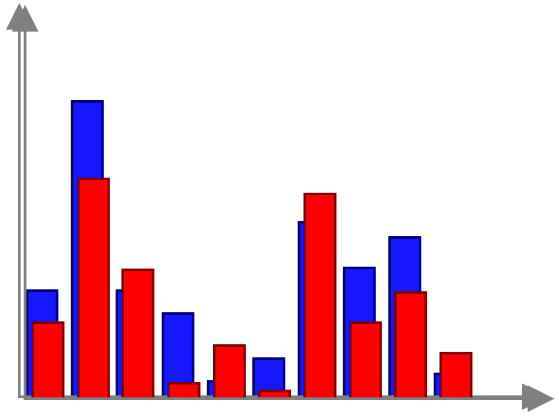


Popular distances: pdf similarity

- 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):

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

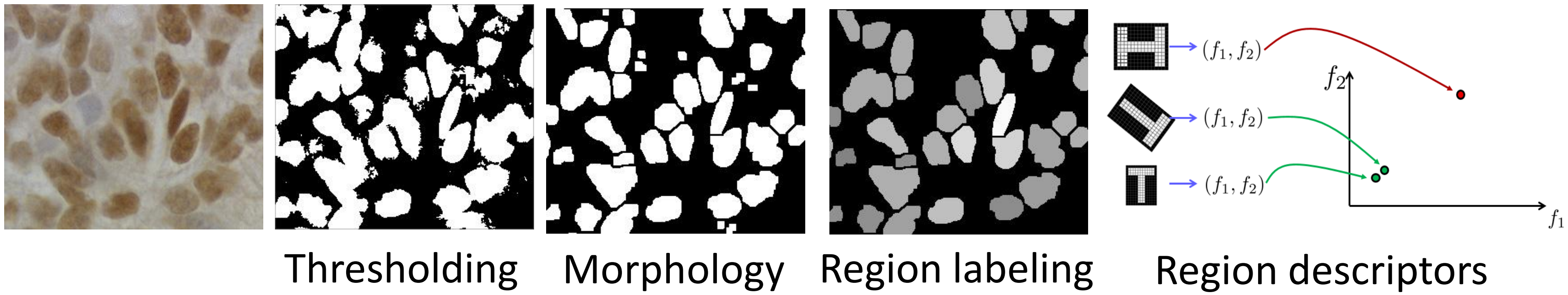
- Hellinger distance:

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

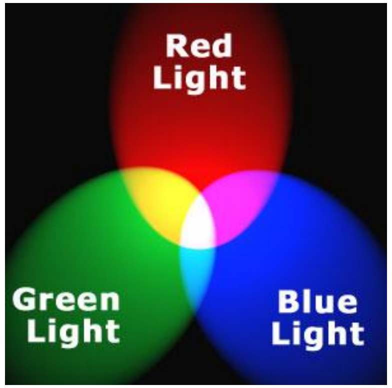
Proper metric, constrained to interval [0,1]

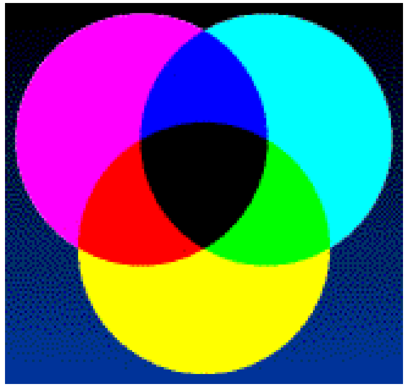
Previously at MP…

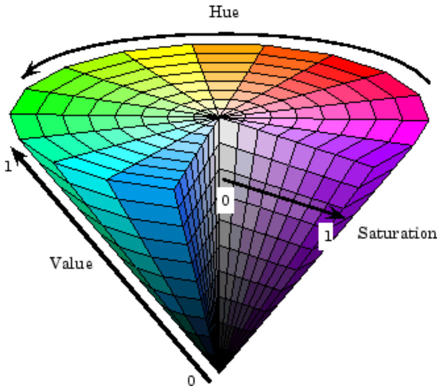
- Basic image processing techniques

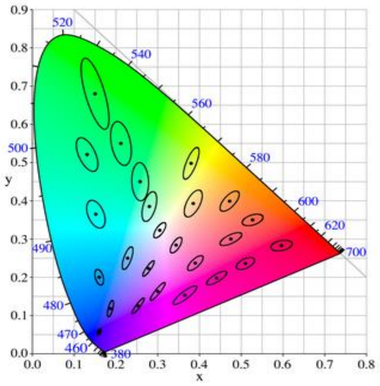


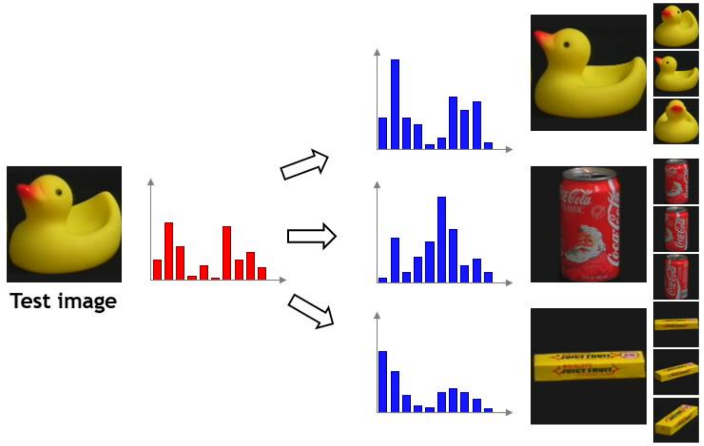
- Color – perception, color spaces and color histograms as image descriptors











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

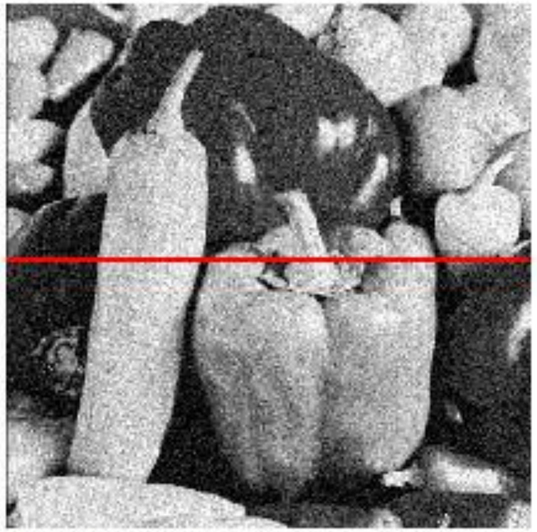

Salt and pepper noise


Impulse noise


Gaussian noise

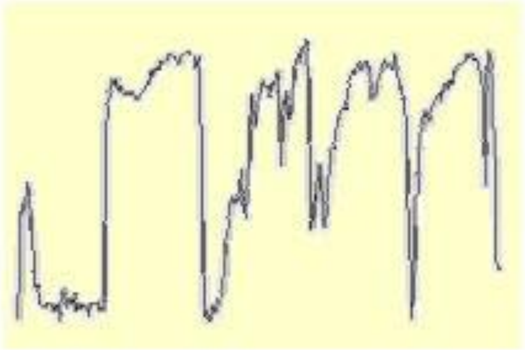
Vir: Steve Seitz

Gaussian noise

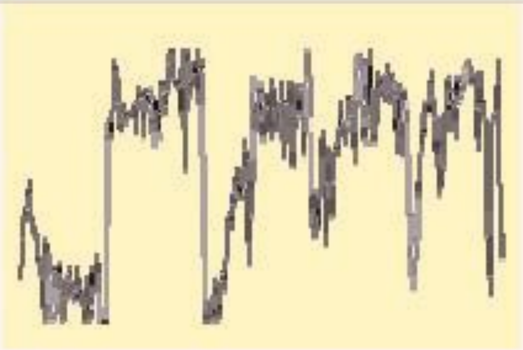


Matlab:

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



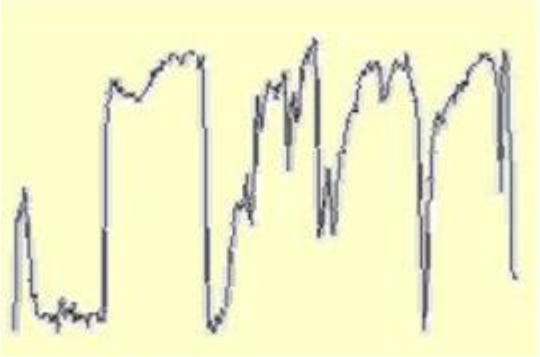
$$\overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}}$$



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

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

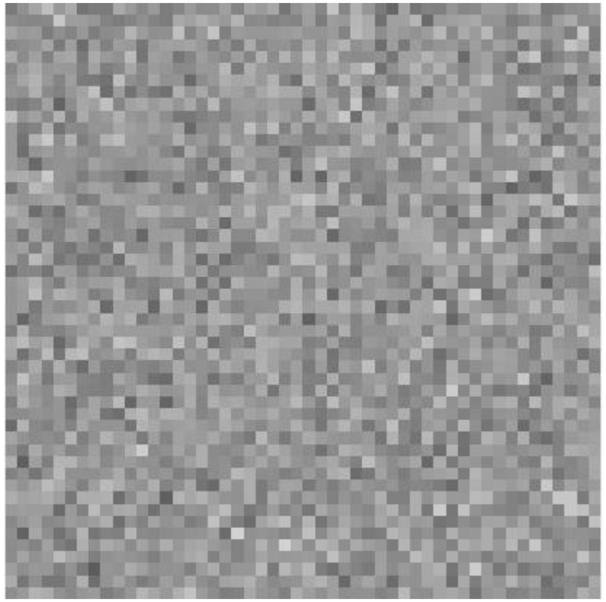
?



$$\overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}}$$

How to remove a Gaussian noise?

A = ones(100,100)*150



B = A + randn(size(A))*20

C = B*0 + mean(B(:))

- $\langle B \rangle = \langle A \rangle + \langle G_{noise} \rangle = \langle A \rangle = 150$
- Solution: just compute the average value!
- Might it really be this simple?

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					

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				

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	

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.

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

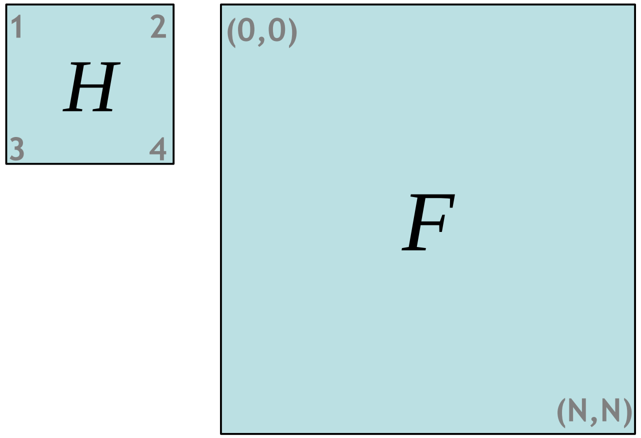
Correlation filtering

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

- 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*.



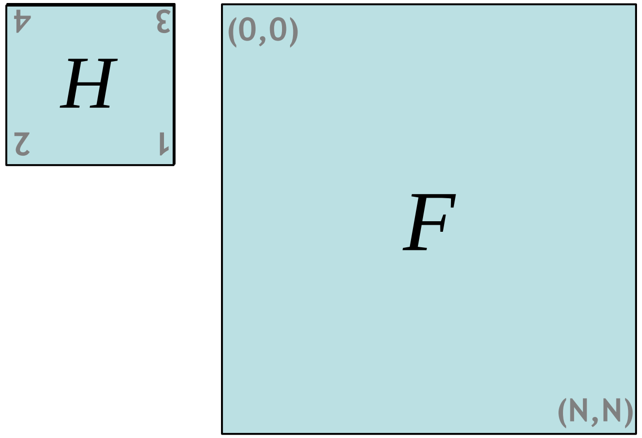
Convolution as correlation

- 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

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

$$G = H \otimes F$$

Notice the difference?

- Convolution

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

$$G = H \star F$$

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

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

Properties of convolution

- Shift-invariant:

  The filter weights remain the same, regardless the position.

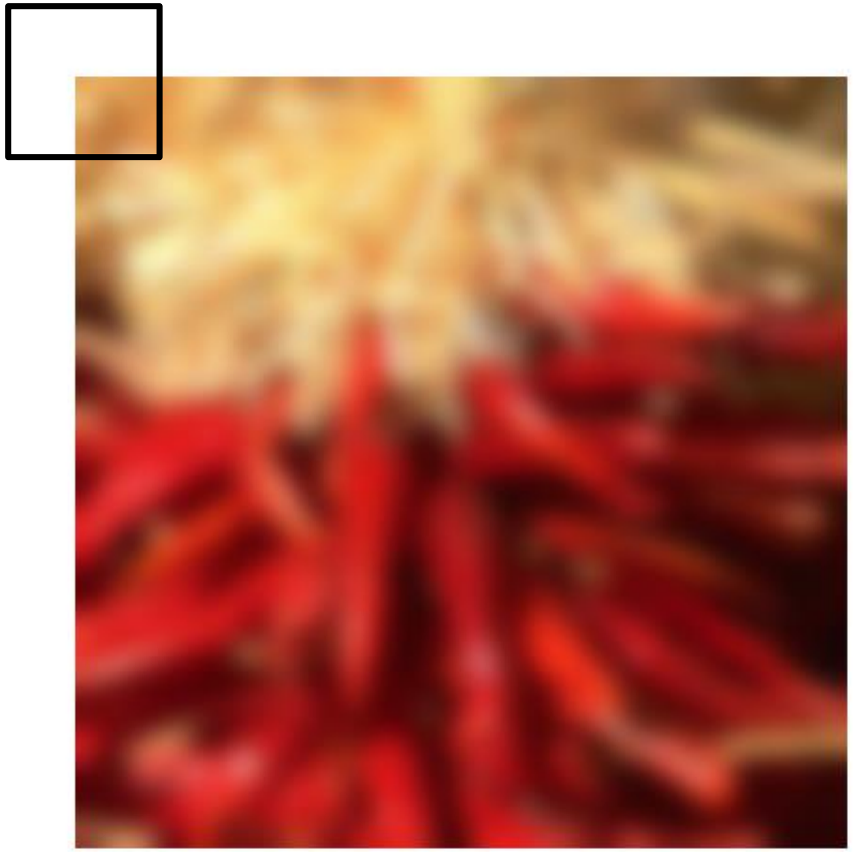
- Linear (superposition & scaling): $h * (\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 (h * f_1) + \alpha_2 (h * f_2)$
- Commutative: $f * g = g * f$
- Associative: $(f * g) * h = f * (g * h)$
  - As result, application of multiple filters is equal to application of a single filter :

    $$\big((f * b_1) * b_2\big) * b_3 = f * (b_1 * b_2 * b_3)$$

- Identity: $f * e = f$, where *e*= [*…, 0, 0, 1, 0, 0, …*] a unit impulse.
- Derivative: $\frac{\partial}{\partial x}(f * g) = \left(\frac{\partial}{\partial x} g\right) * f = \left(\frac{\partial}{\partial x} f\right) * g$

Filtering: Boundary conditions

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



Filtering: Boundary conditions

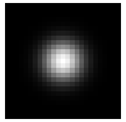
- What to do at the image boundaries?
  - The kernel exceeds image boundaries at the edge
  - Need for extrapolation
  - Methods (Python): `cv2.filter2D( … BorderTypes= )`

Enumerator	
BORDER_CONSTANT Python: cv.BORDER_CONSTANT	`iiiiii|abcdefgh|iiiiiii` with some specified `i`
BORDER_REPLICATE Python: cv.BORDER_REPLICATE	`aaaaaa|abcdefgh|hhhhhhh`
BORDER_REFLECT Python: cv.BORDER_REFLECT	`fedcba|abcdefgh|hgfedcb`
BORDER_WRAP Python: cv.BORDER_WRAP	`cdefgh|abcdefgh|abcdefg`
BORDER_REFLECT_101 Python: cv.BORDER_REFLECT_101	`gfedcb|abcdefgh|gfedcba`
BORDER_TRANSPARENT Python: cv.BORDER_TRANSPARENT	`uvwxyz|abcdefgh|ijklmno`
BORDER_REFLECT101 Python: cv.BORDER_REFLECT101	same as BORDER_REFLECT_101
BORDER_DEFAULT Python: cv.BORDER_DEFAULT	same as BORDER_REFLECT_101
BORDER_ISOLATED Python: cv.BORDER_ISOLATED	do not look outside of ROI

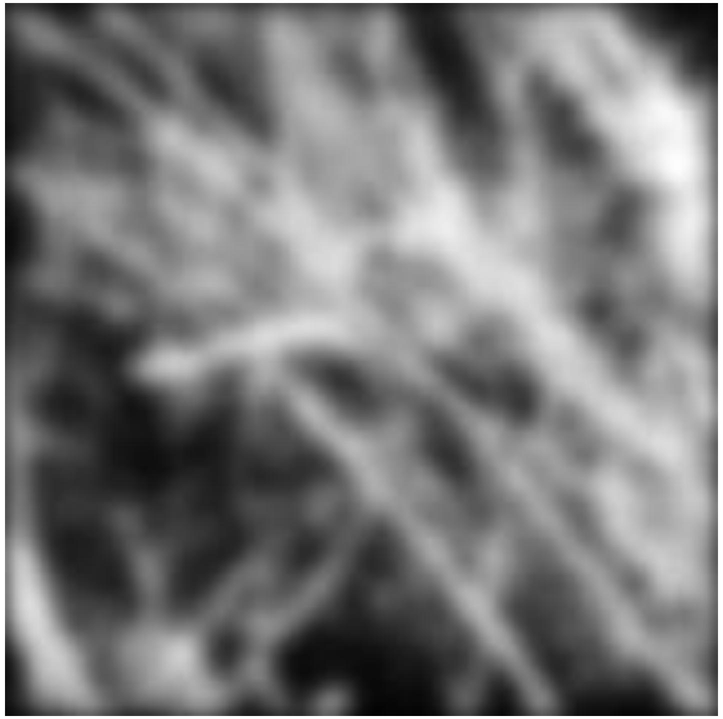
https://docs.opencv.org/master/d2/de8/group__core__array.html#ga209f2f4869e304c82d07739337eae7c5

Caution: the method performs *correlation*, not *convolution*

Smoothing by a Gaussian



Original

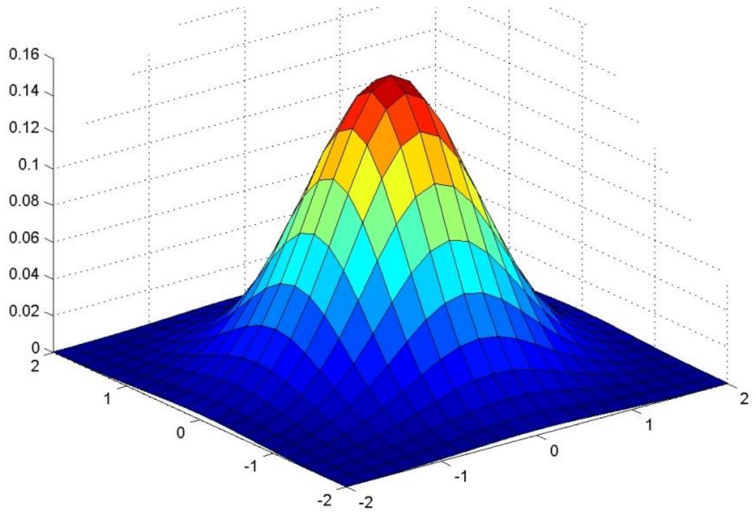


Filtered

Gaussian smoothing

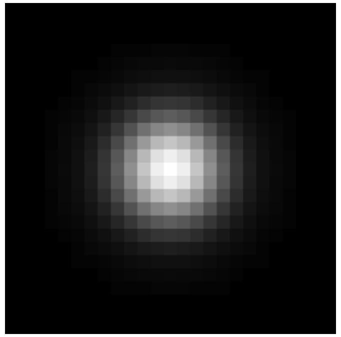
- Gaussian kernel

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

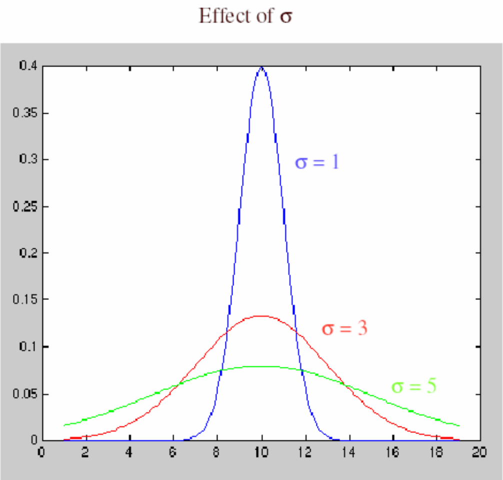


- Rotation symmetric

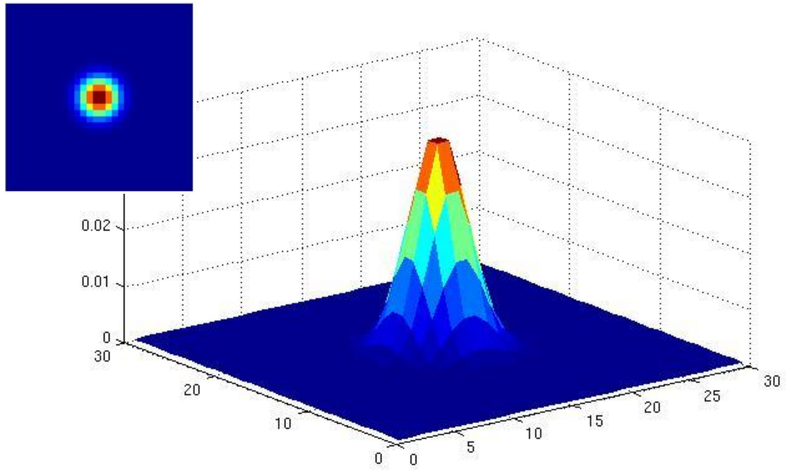
- Pixels closer to center get higher weight
  - Makes sense for a probabilistic inference of a signal content.



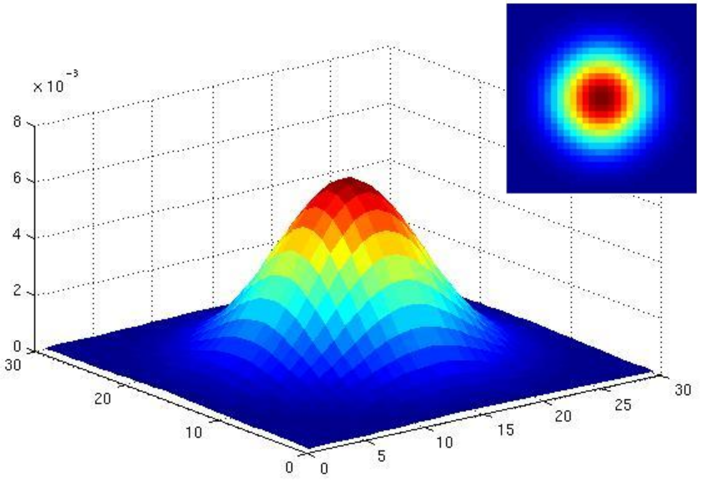
Gaussian smoothing



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



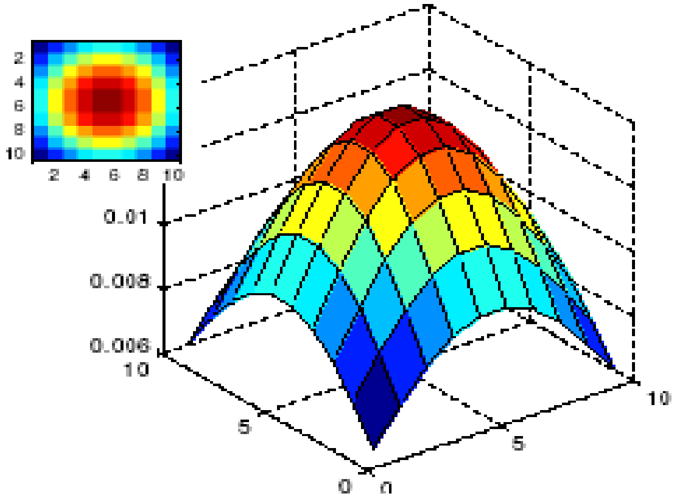
$\sigma = 2$ by kernel
30×30



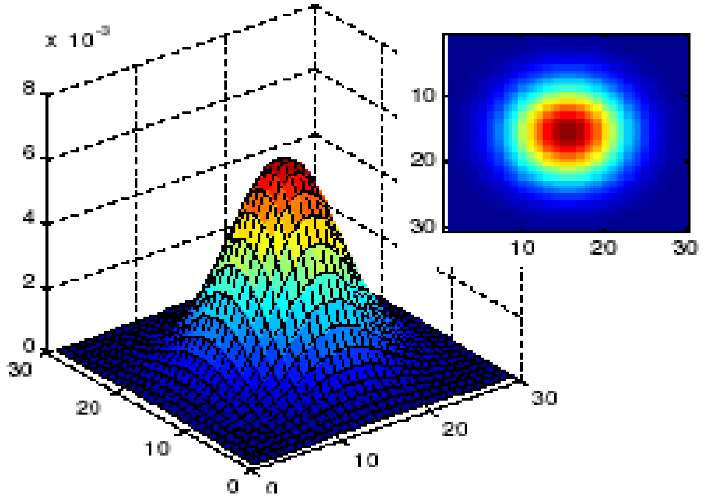
$\sigma = 5$ by kernel
30×30

Gaussian smoothing

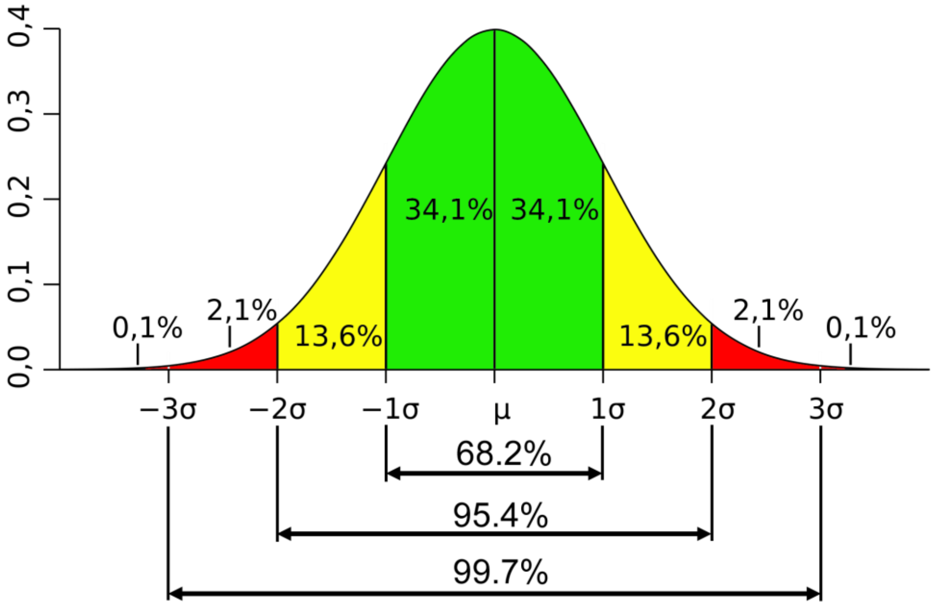
- How about parameters?
- *Kernel size*!
  - Infinite support, but discretization makes it finite.



$\sigma = 5$ with $10{\times}10$ kernel



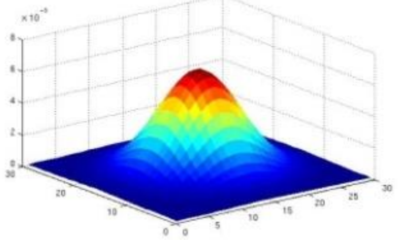
$\sigma = 5$ with $30{\times}30$ kernel



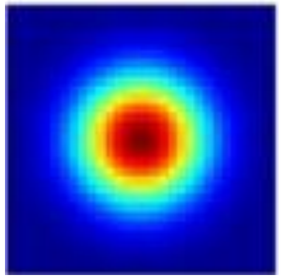
  - Rule of thumb: set half size of the kernel to 3σ

Gaussian filtering in Matlab

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



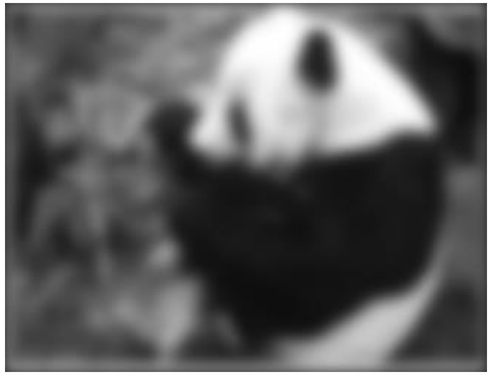
```
>> figure(1); surf(h);
```



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

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

im



outim

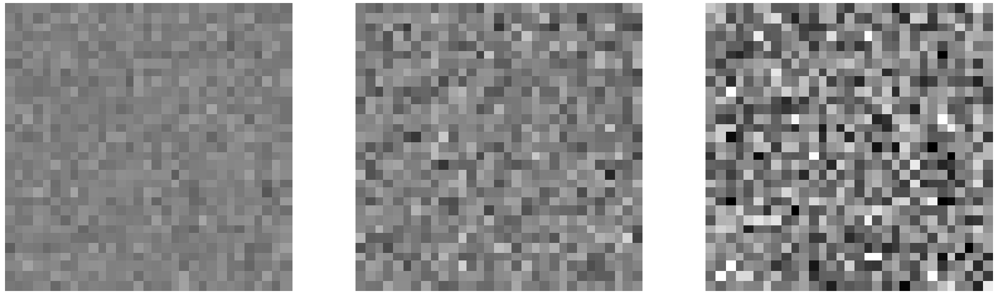
Python:

outim = cv2.GaussianBlur(im,(31,31),0)

Effects of smoothing

Increasing the noise extent→

σ=0.05 σ=0.1 σ=0.2



no smoothing

Increasing the kernel size →

Slide credit: Kristen Grauman

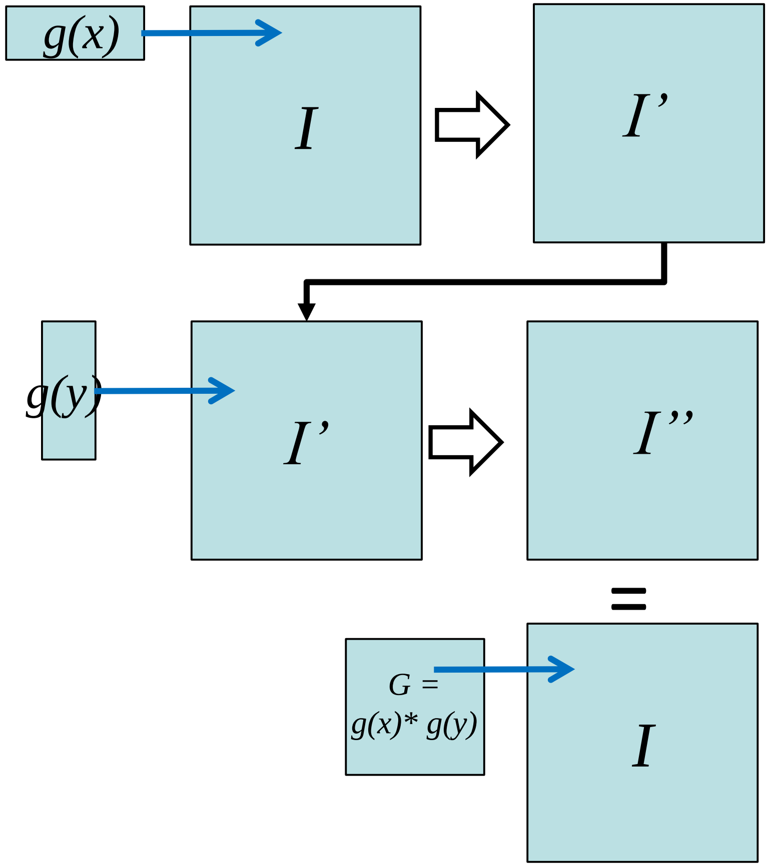
Efficient implementation

- 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:

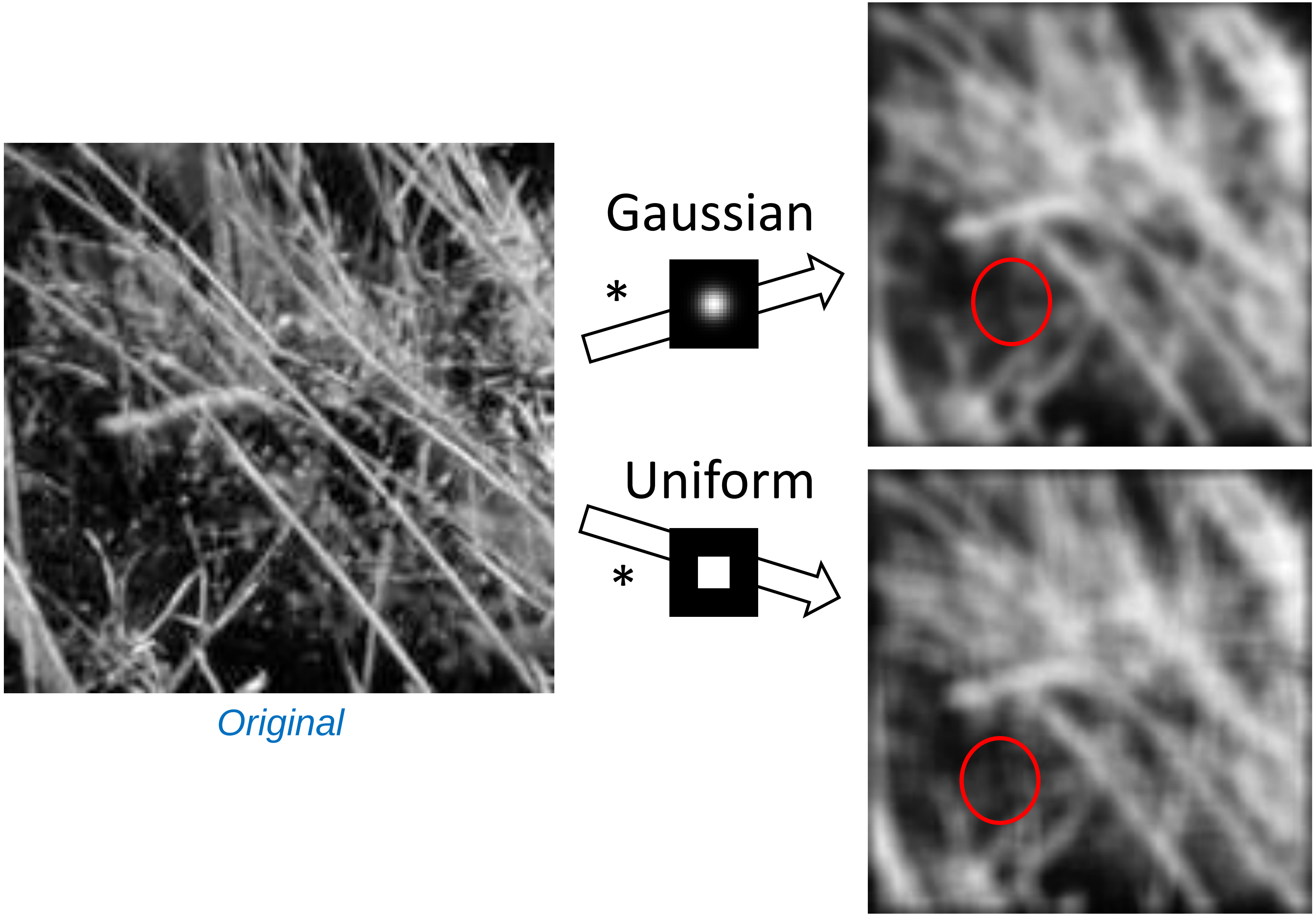
$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-y^2 / (2\sigma^2))$$



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

$$g_x * \left(g_y * I\right) = \left(g_x * g_y\right) * I$$

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).

$$\underbrace{\mathcal{F}(f * g)}_{\text{Image } f \text{ and filter } g} = \mathcal{F}(f) \odot \mathcal{F}(g)$$

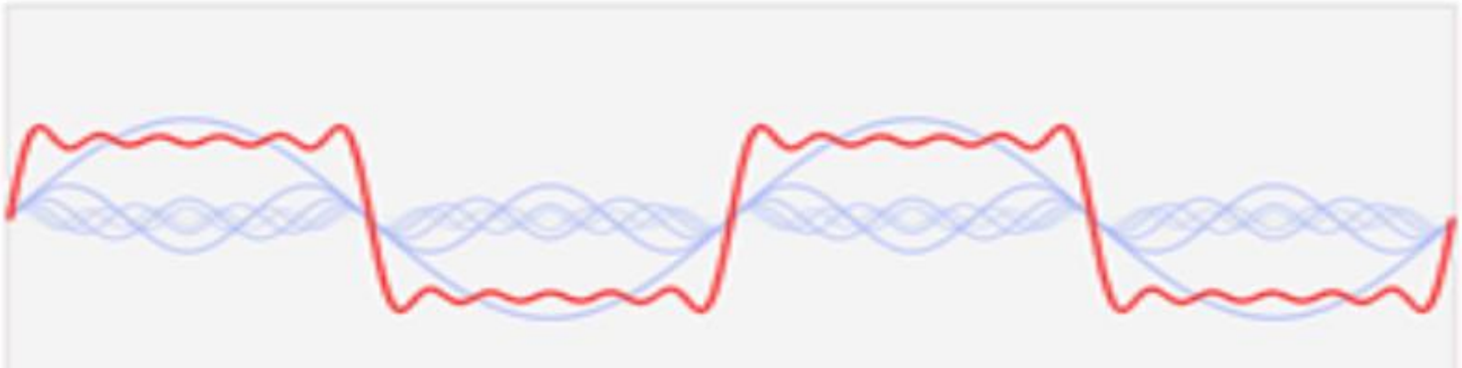
Image f
and filter g

Fourier transforms
of f and g.

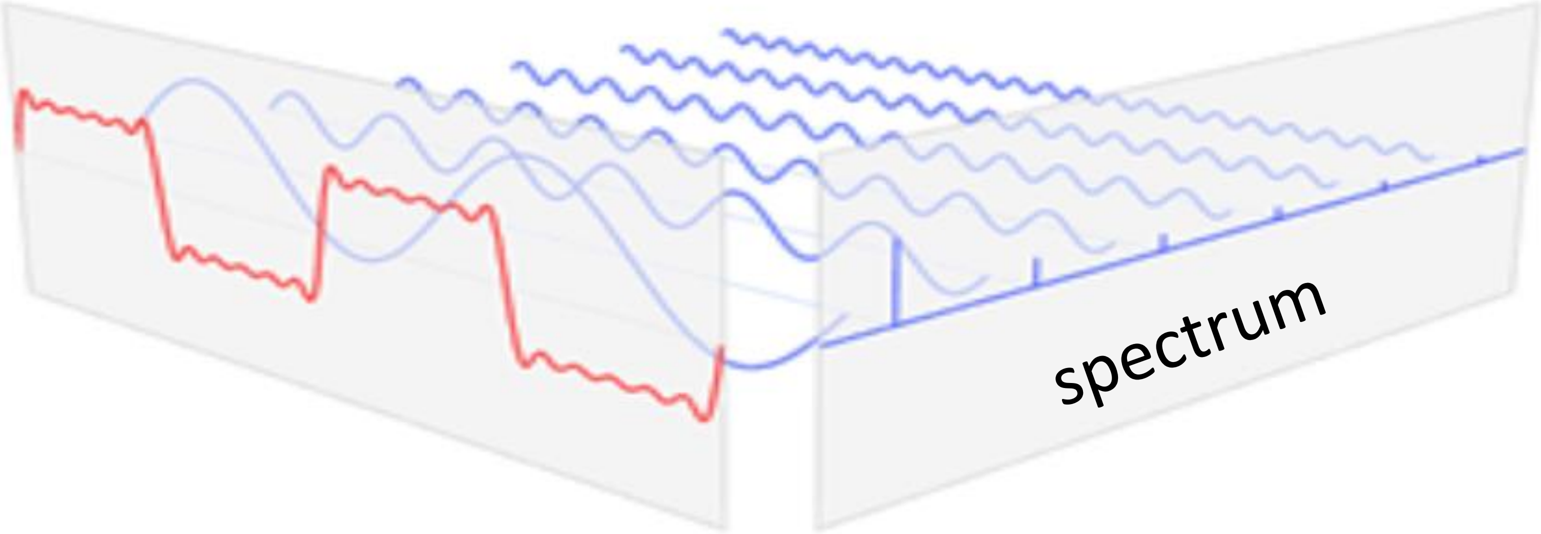
- Convolution manipulates the image spectrum
  - Enhancing/suppressing frequency bands in image.

Recall the Fourier transform

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



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

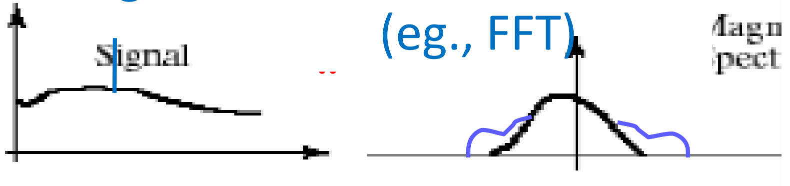


Convolution: removing noise

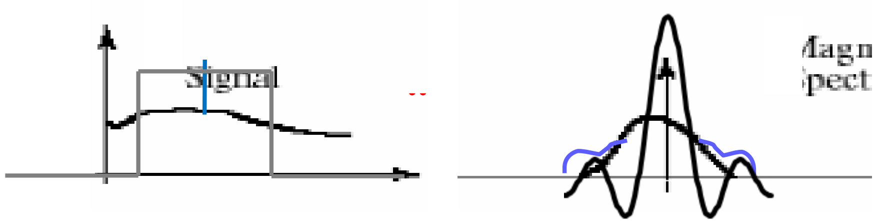
- Noise corresponds to adding high frequencies. To remove these, we apply a *low-band pass* filter.
- The spatial box filter transforms to a sinc in frequency space, causing artefacts (side lobes).
- A Gaussian maintains a compact support in both image and frequency space. Hence, it’s more appropriate as a low-band-pass filter.

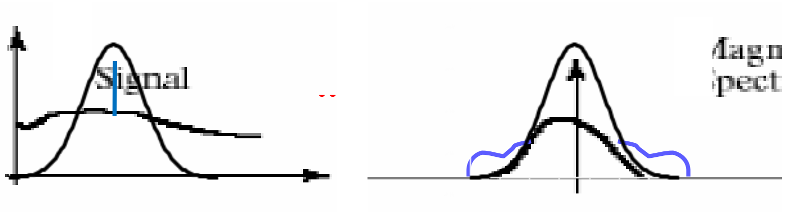
signal

Frequency spectrum (eg., FFT)



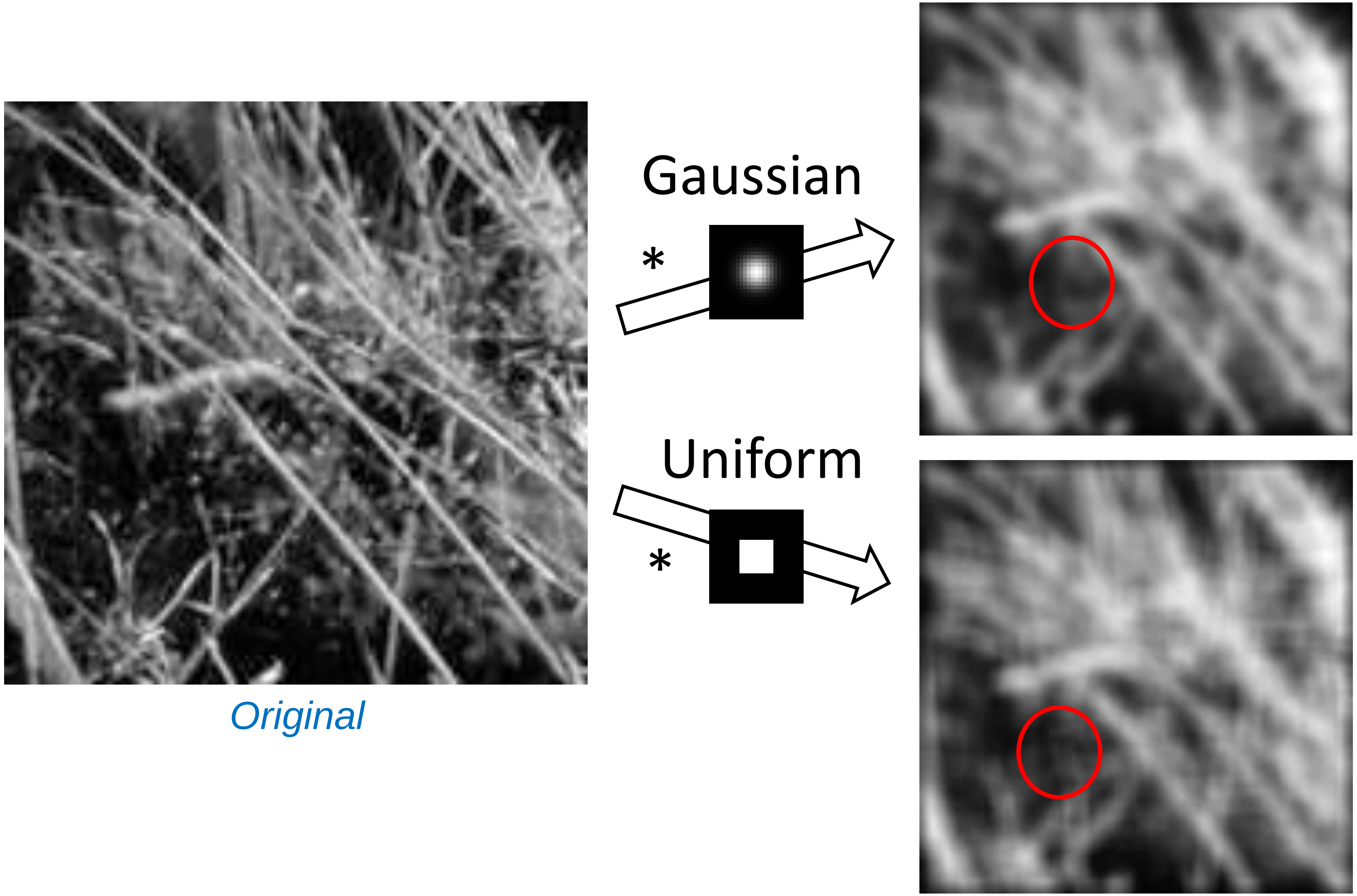
$$\mathcal{F}(f * g) = \mathcal{F}(f) \odot \mathcal{F}(g)$$





See Forsyth, Ponce: Computer vision, a modern approach

Strange artefacts in convolution results…



Original

Filtered

Filter does not introduce high frequencies

Filter introduces high frequencies

Linear filters in practice

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

Original

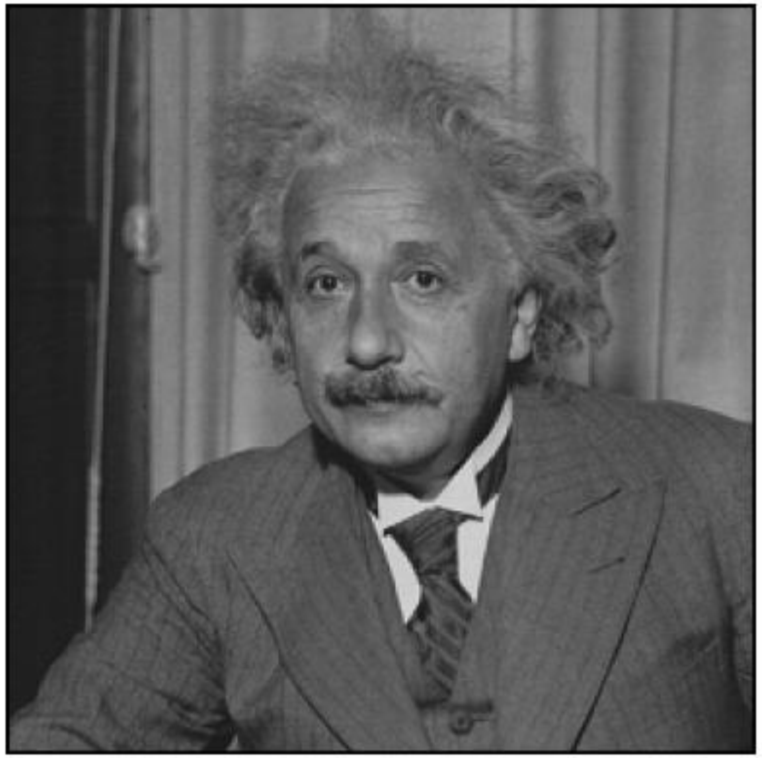
Sharpening filter:

Enhances differences by local averaging.

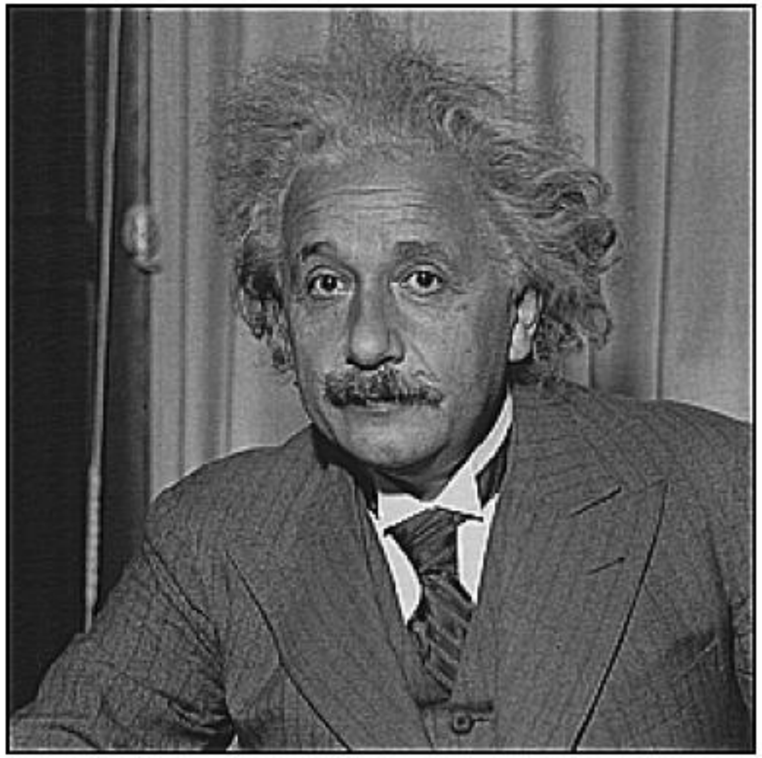
To explain this, think about what happens in frequency domain.

Sharpening filter

before



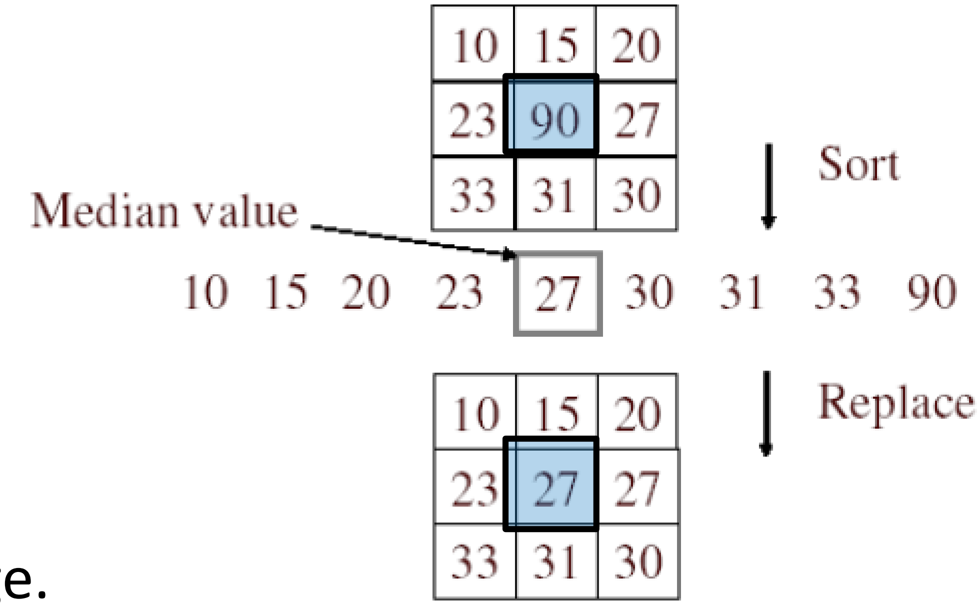
after



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

Nonlinear filters: Median filter

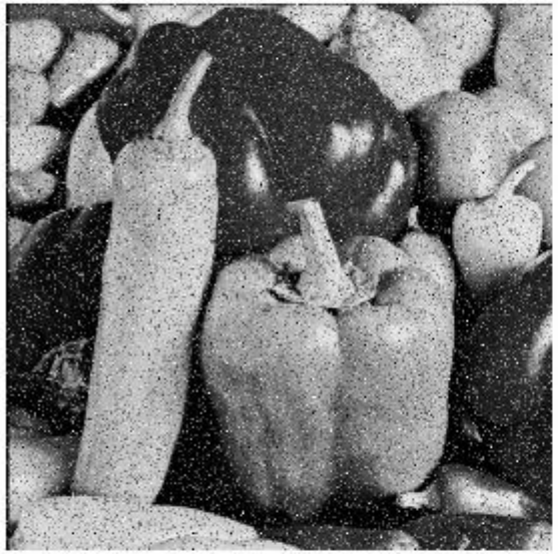
- Basic idea
  - Replace the pixel intensity by a median of intensities within a small patch.



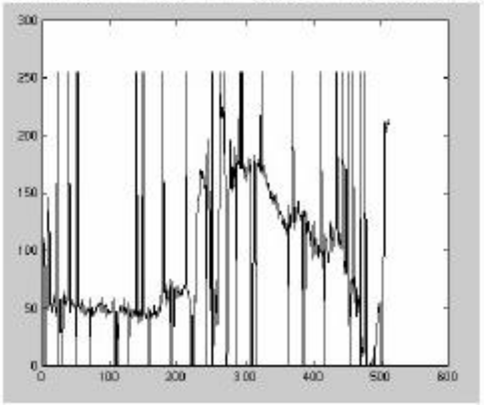
- Properties
  - Does not add new gray-levels into the image.
  - Removes outliers: appropriate for impulse noise and salt&pepper noise removal.

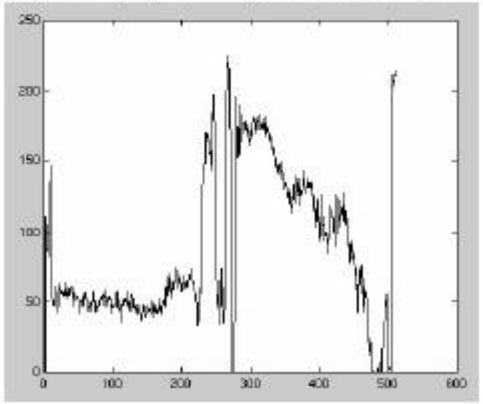
The Median filter

Salt&pepper noise



After median filtering





Plot of a line in the image

Median vs. Gaussian

3x3 5x5 7x7

Gaussian filter

Median filter

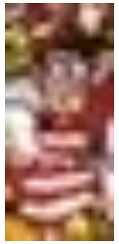
Slide credit: Svetlana Lazebnik

Machine perception

LINEAR FILTERING AS TEMPLATE MATCHING

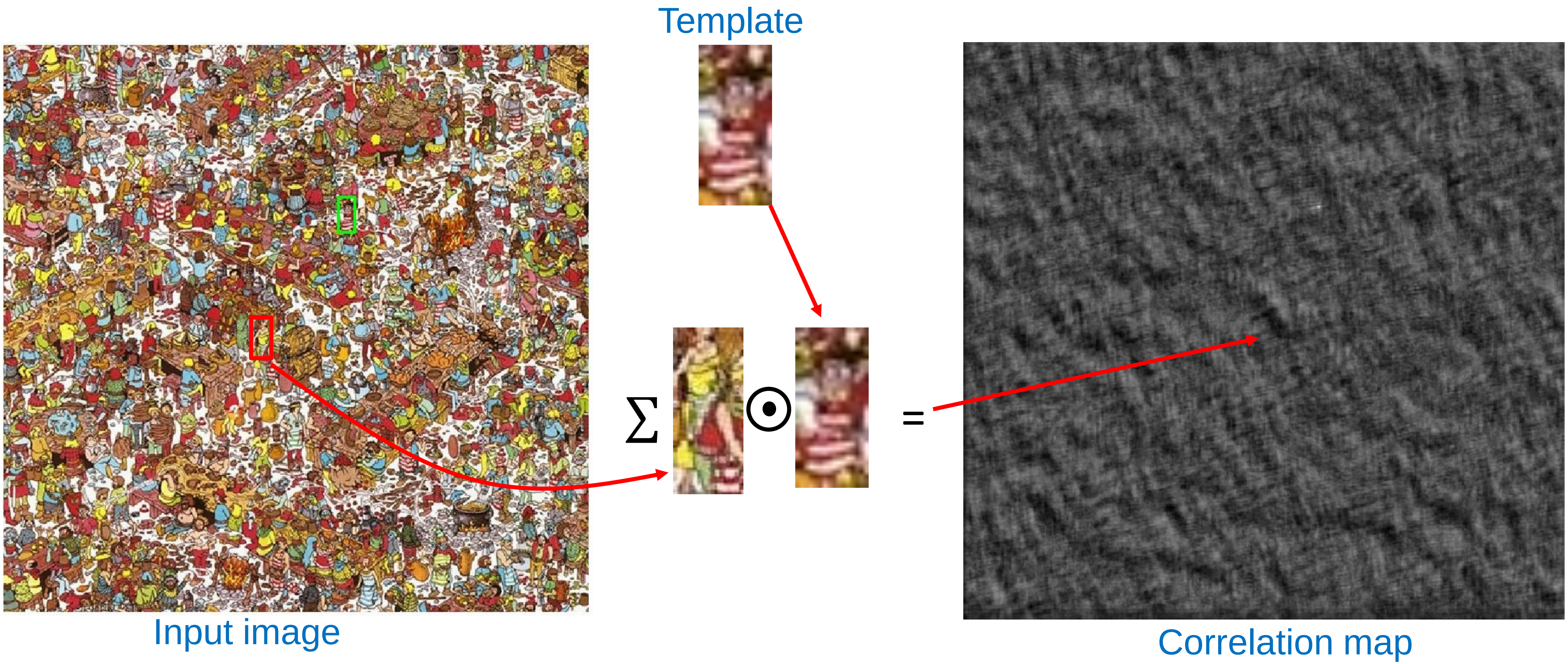
Filtering as template matching

Where's waldo?



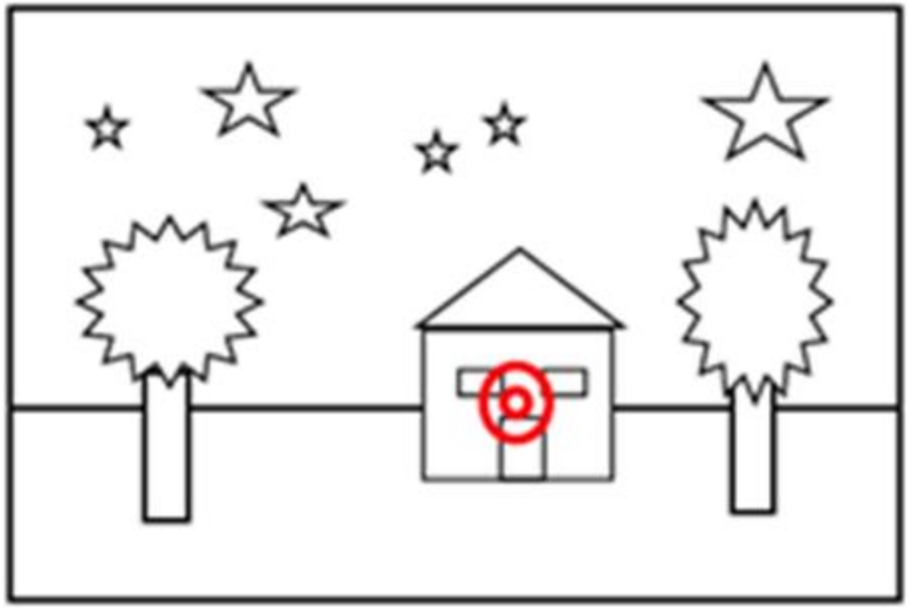
Template

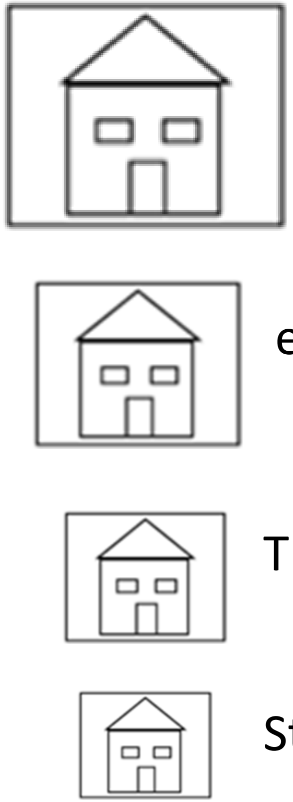
Apply correlation with template



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...





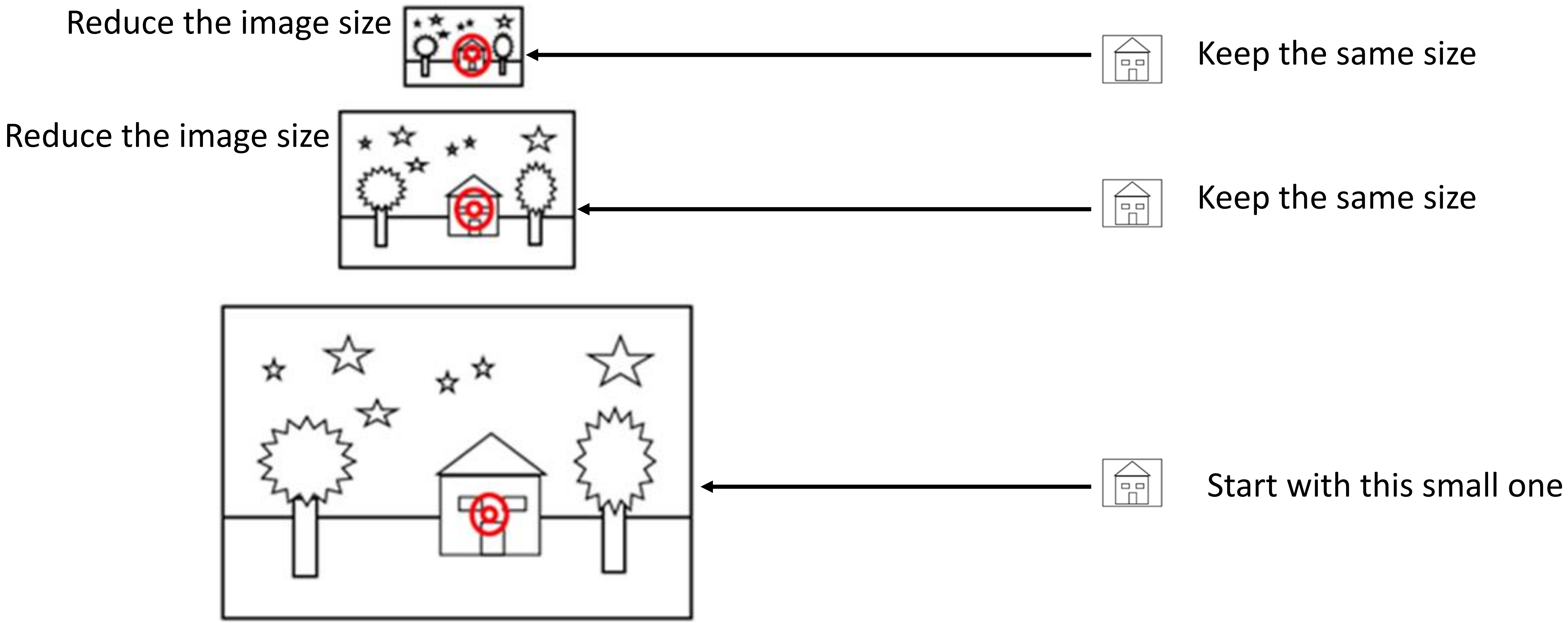
etc. ...

Then with this one

Start with this small one

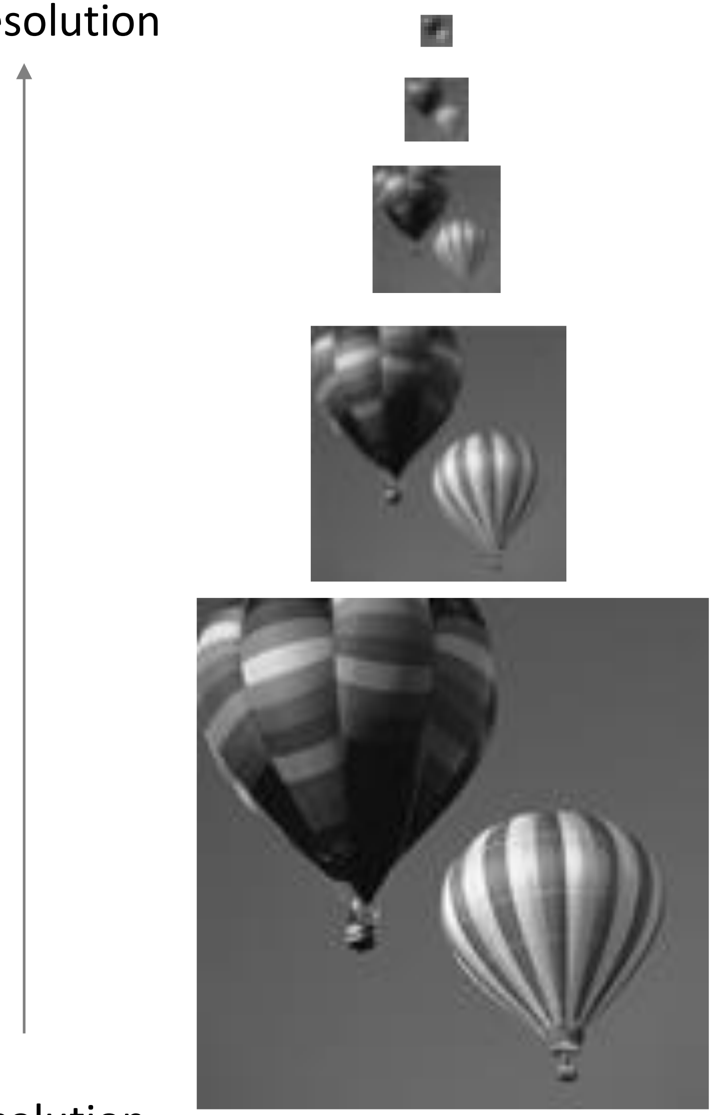
Template matching in scale space

- But rather than template, we scale the input image

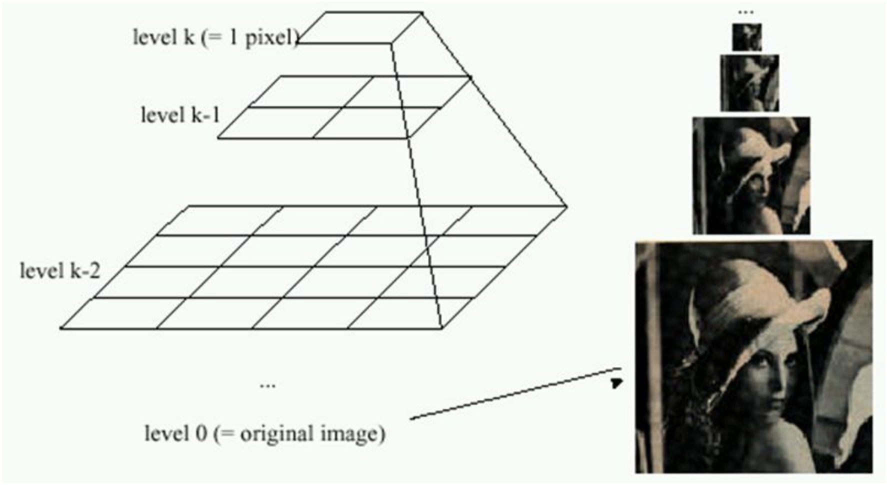


Efficient resizing: Image pyramids

Low resolution



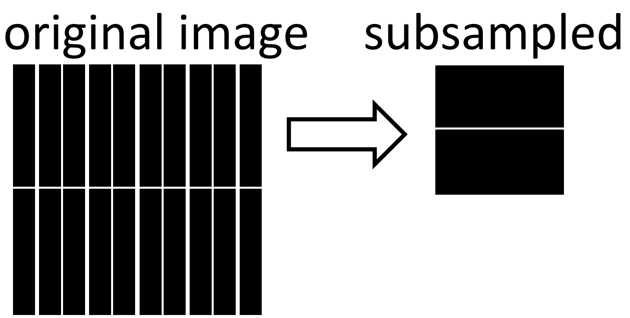
High resolution



Reduce (resample) the image!

How do we reduce an image?

- Naive:
  - Remove every second pixel...

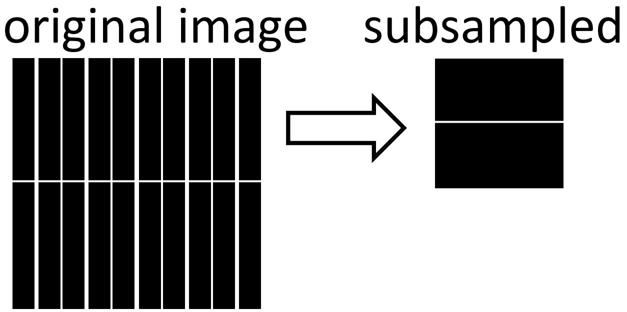


  - Problem: *the structures in image change!*
- This effect is called *Aliasing.*
- Look into frequency domain to explain this (Forsyth-Ponce Book)

Avoiding aliasing

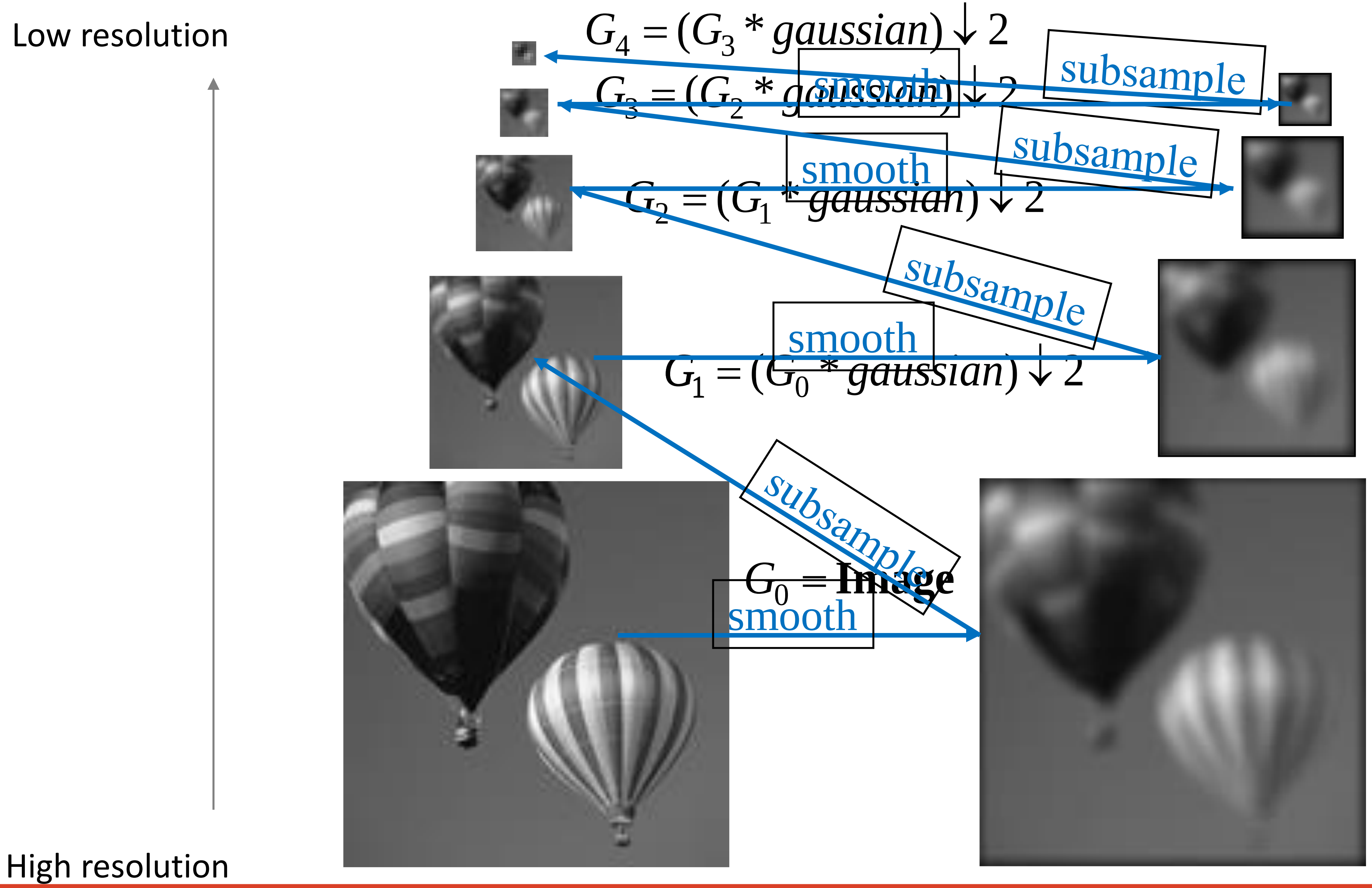
- Nyquist theorem:
  - If we want to reconstruct all frequencies up to *f*, we have to sample the signal by at least a frequency equal to *2f*.

- Meaning: we cannot reconstruct some of the high frequencies when subsampling!



- Solution: Remove the high frequencies that cannot be reconstructed, then subsample.

Gaussian pyramid



Summary: Gaussian pyramid

- Construction: get a new level directly from the previous
  - Smooth by a small filter and resample

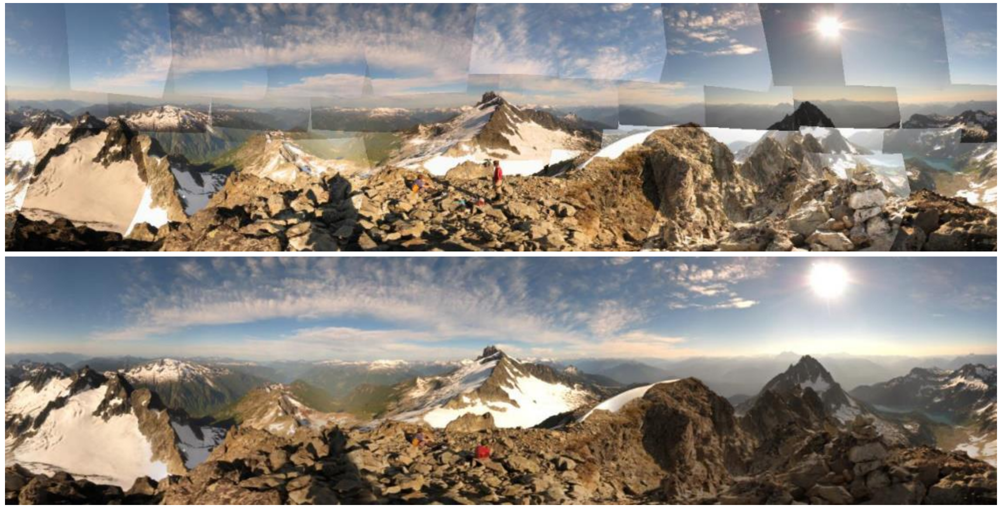
- Reasons for Gaussian smoothing...
  - Convolution Gauss*Gauss = new Gauss
  - $G(\sigma_1^2) * G(\sigma_2^2) = G(\sigma_1^2 + \sigma_2^2)$

- 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.

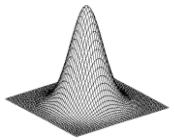
Why a pyramid?

- 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

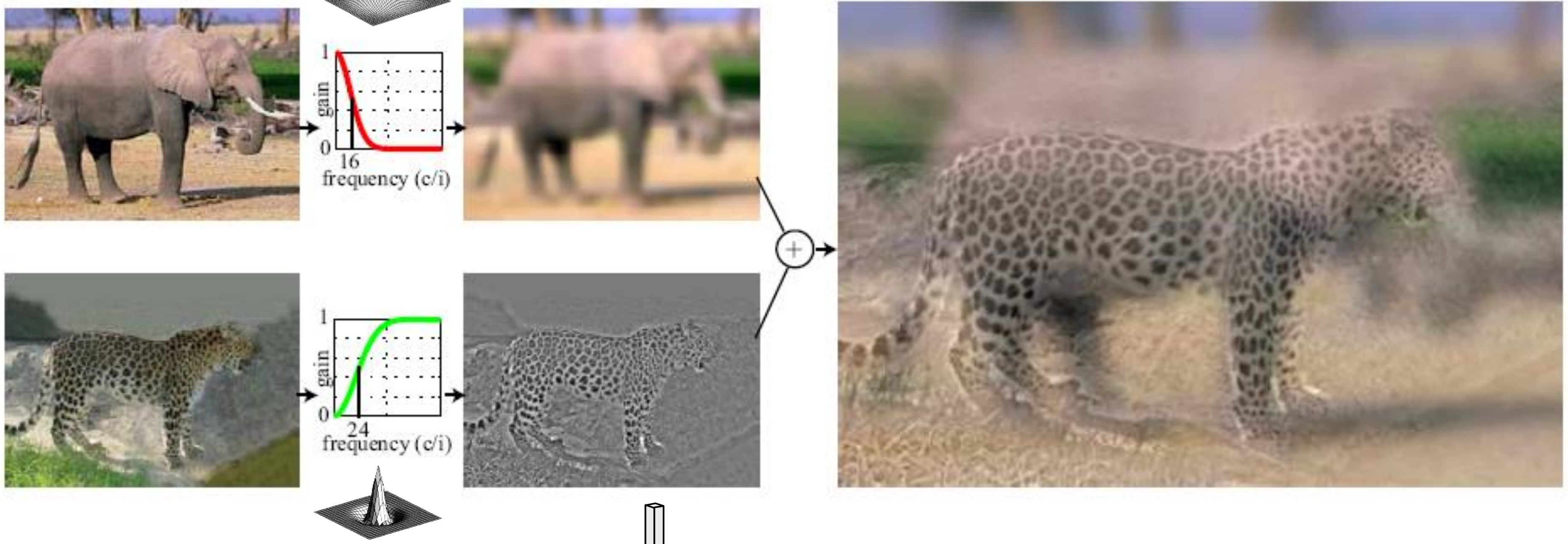


Fun with hybrid images...

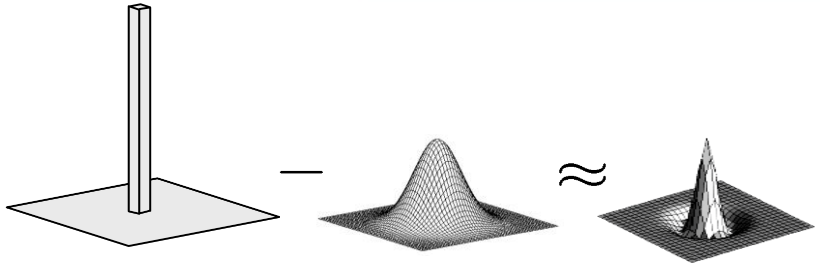
Gaussian filter



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