COMPUTER VISION

LECTURE 3 11.9.2019

Laura Ruotsalainen, Associate Professor
Department of Computer Science

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

IMAGE PROCESSING, TODAY'S LECTURE

- Point processing, neighbourhood processing
- Convolution
- Box filter, Gaussian filter, other filters
- Fourier transformation

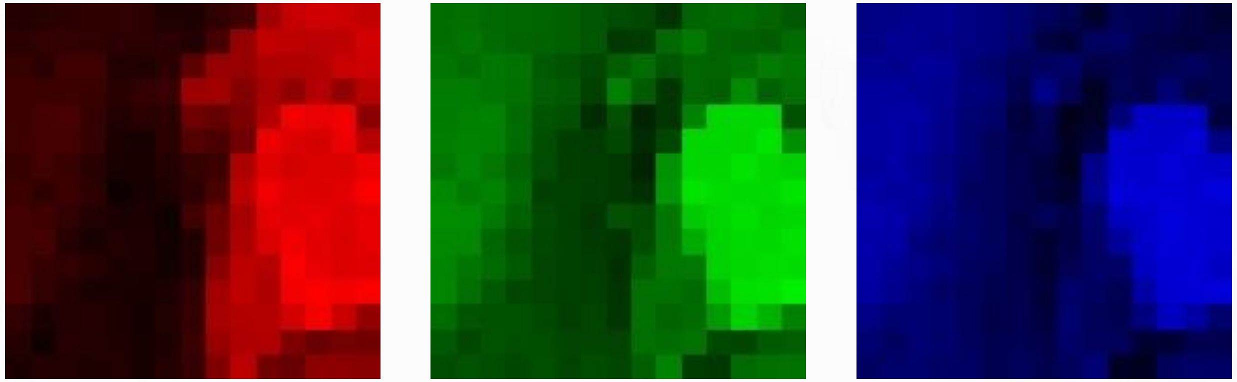
- Szeliski chapter 3
- However, Forsyth & Ponce provides much more comprehensive and logical explanation which is found from the slides

IMAGE PROCESSING

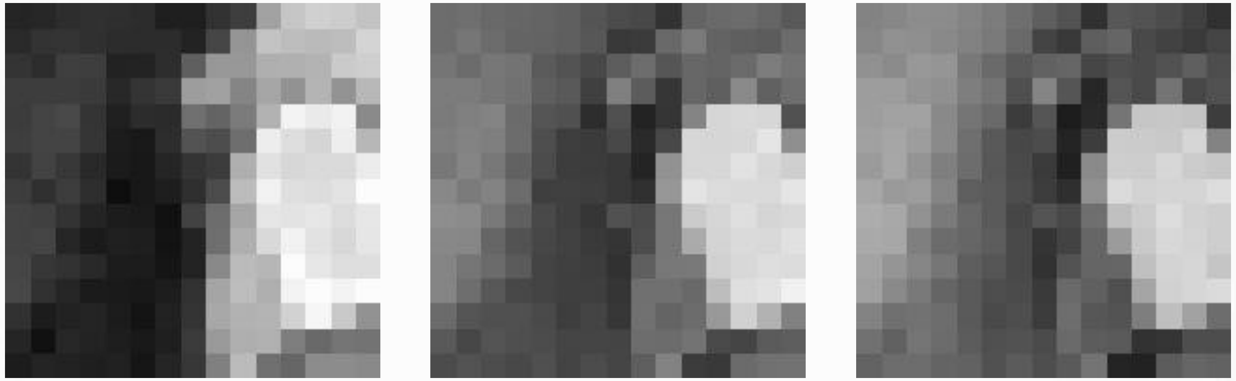
- Image = interaction of 3D scene objects, lighting, camera optics and sensors
- First step in computer vision is processing the image to be suitable for the computations needed => image processing
  - Exposure correction and color balancing
  - Noise reduction
  - Increasing sharpness
  - Rotating the image
- Point operators
- Neighborhood operators (linear filtering, non-linear filtering)
- Global operators

WHAT IS AN IMAGE?

red green blue



colorized for visualization



actual intensity values per channel

color image patch

How many bits are the intensity values?

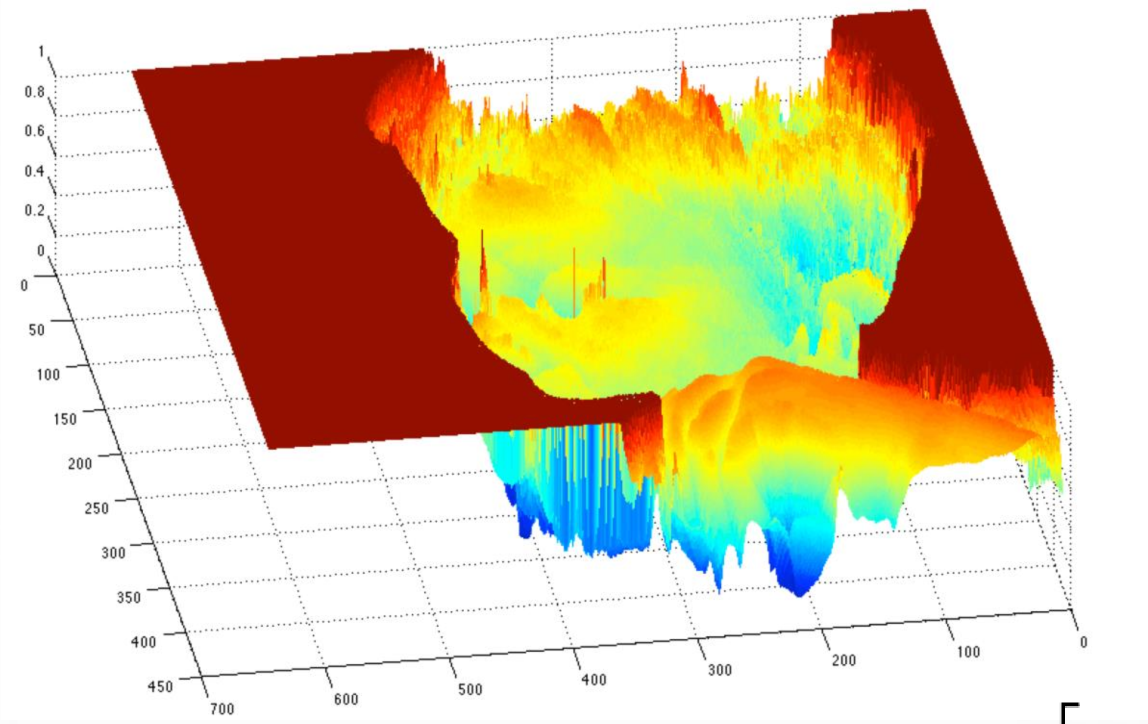
Each channel is a 2D array of numbers.

WHAT IS AN IMAGE?

grayscale image

What is the range of the image function f?

$f(\boldsymbol{x})$



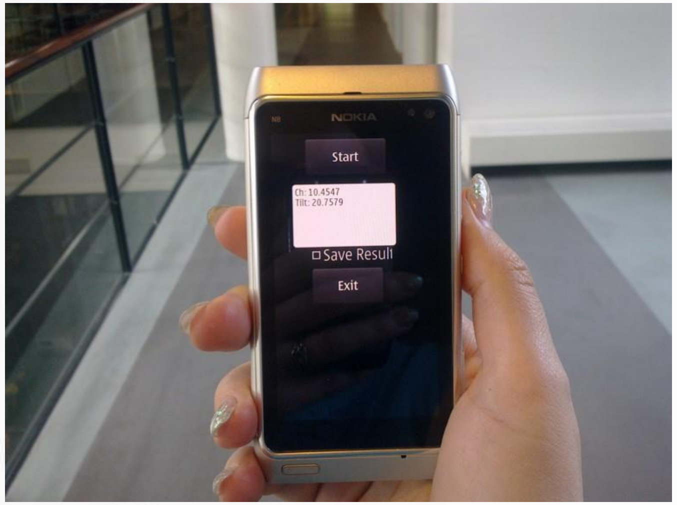
domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

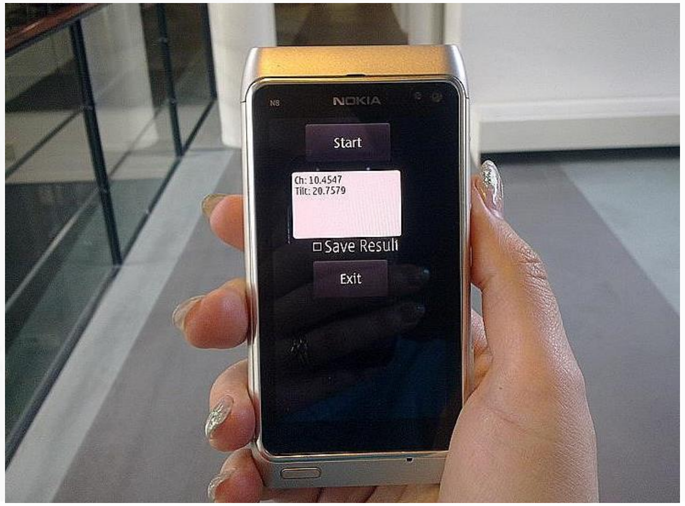
A (grayscale) image is a 2D function.

Slide: Gkioulekas

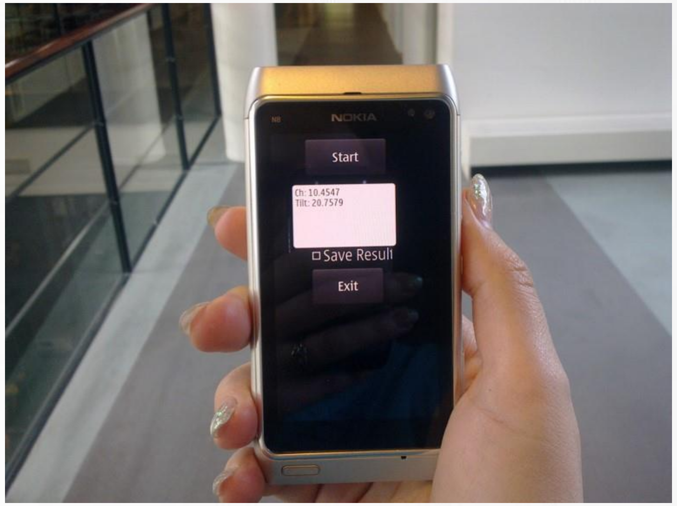
TYPES OF IMAGE TRANSFORMATIONS

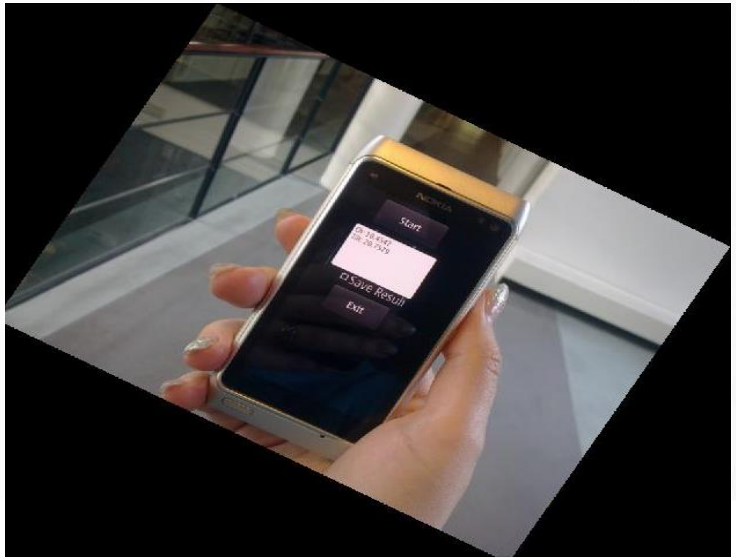
Filtering, changes pixel values





Warping, changes pixel locations

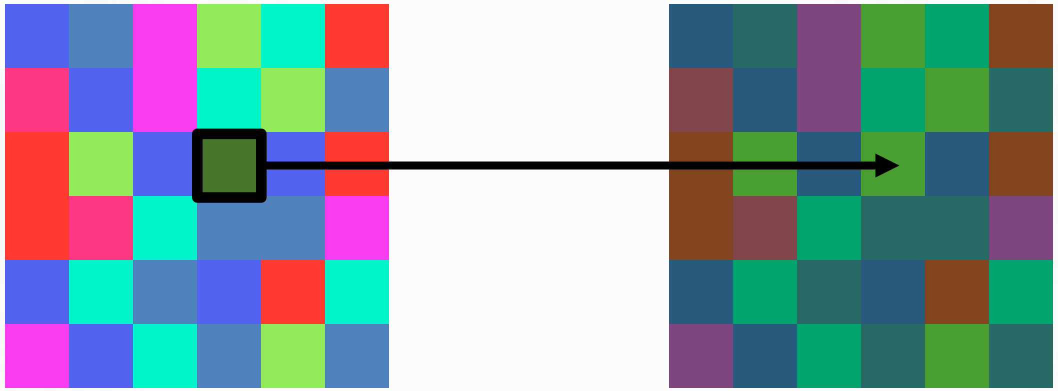




POINT VS NEIGHBORHOOD OPERATION

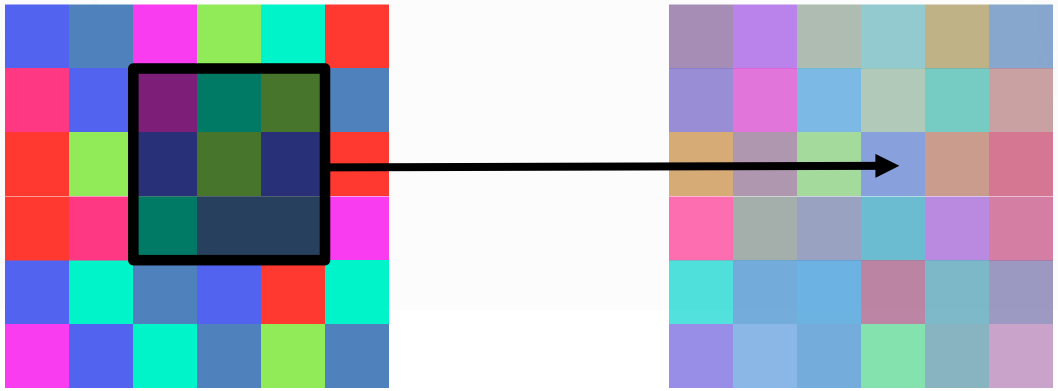
- Brightness and contrast adjustments
- Color transformations

Point Operation



point processing

Neighborhood Operation



“filtering”

POINT PROCESSES

- Image processing operator (*h*) takes an input image (*f*) and produces an output image (*g*)

$$g(i,j) = h(f(i,j)).$$

- Common point processes are multiplication and addition with a constant

$$g(x) = af(x) + b.$$

- *a* is called a gain parameter controlling the contrast and *b* is called a bias parameter controlling brightness
- Both can be spatially varying for e.g. cool effects

EXAMPLES OF POINT PROCESSING

$$x$$

darken

$$x - 128$$

lower contrast

$$\frac{x}{2}$$

non-linear lower contrast

$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

invert

$$255 - x$$

lighten

$$x + 128$$

raise contrast

$$x \times 2$$

non-linear raise contrast

$$\left(\frac{x}{255}\right)^{2} \times 255$$

Slide: Gkioulekas

MANY OTHER TYPES OF POINT PROCESSING

camera output

image after stylistic tonemapping

[Bae et al., SIGGRAPH 2006]

NEIGHBORHOOD OPERATIONS

- Filtering
  - Local tone adjustment
  - Adding soft blur
  - Sharpening details
  - Accentuate edges
  - Remove noise
- Linear filtering = weighted combinations of pixels
- Non-linear filtering = morphological filters, distance transforms

Szeliski

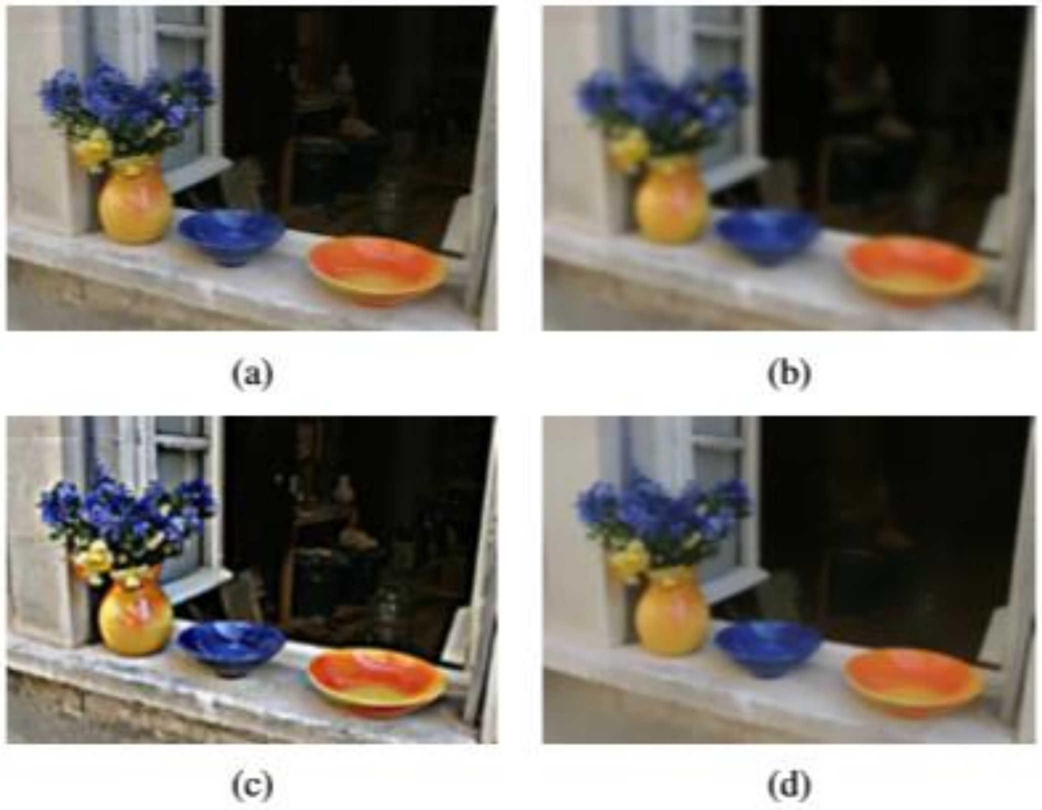


Figure 3.11 Some neighborhood operations: (a) original image; (b) blurred; (c) sharpened; (d) smoothed with edge-preserving filter

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

LINEAR FILTERING AND CONVOLUTION

- *Shift invariant filtering*
  - value of the output depends on the pattern in an image neihborhood and not in its location
  - linear means that the output for the sum of two images equals the sum of the outputs obtained for the images separately
- *Kernel (mask, filter)* = pattern of weights used for linear filtering (*h*)
- *Convolution* = process of applying the linear filter

$$G_{ij} = \sum_{u,v} F_{i-u,j-v} H_{u,v}$$

where *f* is the original image and *g* is the resulting image

DEFINING CONVOLUTION 1

- F is the image, H is the kernel and G is the output image
- Kernel size is 2k+1 x 2k+1

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

- This is a dot product between a certain local neighborhood and the kernel for each pixel

  - Cross-correlation

$$G = H \otimes F$$

DEFINING CONVOLUTION 2

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

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

- This is called a (discrete) **convolution** operation: $G = H * F$
- *H = impulse response function*, when kernel function H is convolved with impulse signal δ(i,j) => *H** δ=*H* whereas correlation produces the reflected signal

MORE PROPERTIES OF CONVOLUTION

Convolution is linear

Convolution is shift-invariant

Convolution is commutative (w*f = f*w)

Convolution is *associative* (v*(w*f) = (v*w)*f)

Every linear shift-invariant operation is a convolution

EXAMPLE: THE BOX FILTER

- also known as the 2D rect filter
- also known as the square mean filter

kernel $h[\cdot,\cdot] = \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$

- replaces pixel with local average
- has smoothing (blurring) effect

Slide: Gkioulekas

MEAN FILTERING

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

H

$*$

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

F

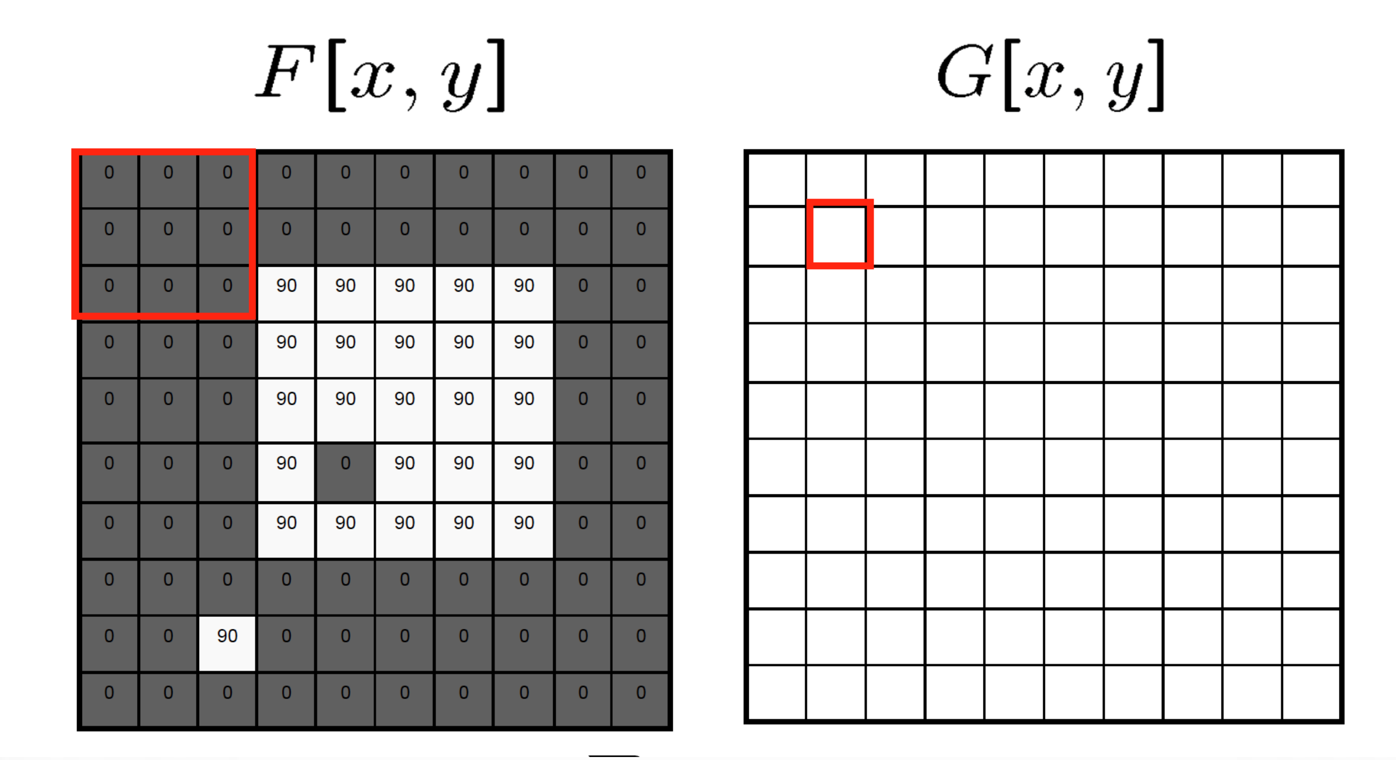
$=$

	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	

G

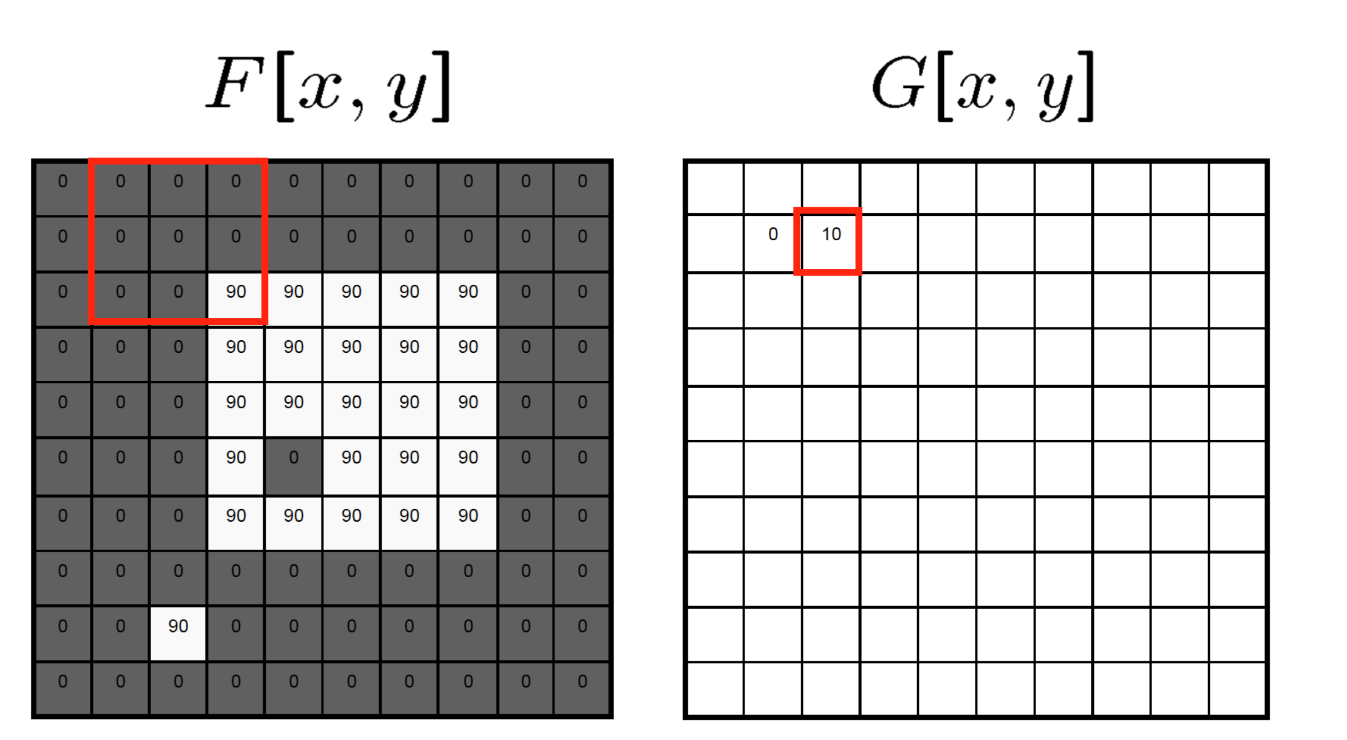
Slide: Noah Sively

MEAN FILTERING/MOVING AVERAGE



Slide: Noah Sively

MEAN FILTERING/MOVING AVERAGE



Slide: Noah Sively

MEAN FILTERING/MOVING AVERAGE

$F[x, y]$

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						

Slide: Noah Sively

MEAN FILTERING/MOVING AVERAGE

$F[x, y]$

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					

Slide: Noah Sively

MEAN FILTERING/MOVING AVERAGE

$F[x, y]$

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				

Slide: Noah Sively

MEAN FILTERING/MOVING AVERAGE

- Kernels are usually shift- invariant = behave similarly everywhere in the image
- However, shift-variant kernels may be used e.g.for blurring an image due to variable depth-dependent defocus

$F[x, y]$

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	

What about the edges?

COMPUTATIONAL REQUIREMENTS

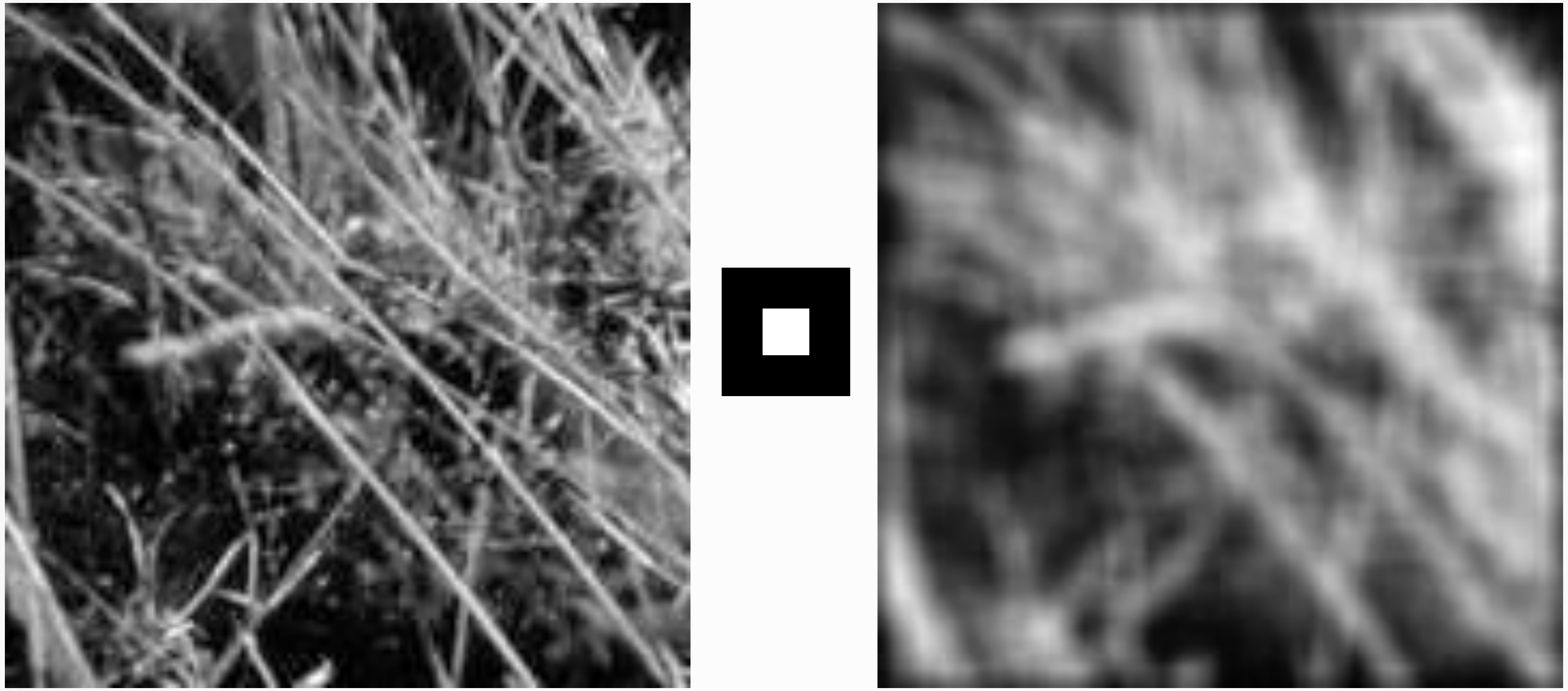
- Convolution requires K^2 multiply-add operations per pixel, K is the size (width w or height h) of the kernel => Every entry takes $O(k^2)$ operations, Total time complexity: $O(whK^2)$
- The process is much faster if first a one-dimensional horizontal convolution followed by vertical is performed => separable kernels required
- Horizontal kernel (h), vertical (v) => Kernel $K = vh^T$,time complexity of separable version : O(whk)
- Kernel is separable if only the first singular value σ_0 is non-zero when Singular Value Decomposition (SVD) is taken of the Kernel K.

$$K = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

w,h = Image size

- Vertical kernel v = $\sqrt{\sigma_0}\, \mathbf{u}_0$, horizontal kernel h = $\sqrt{\sigma_0} \mathbf{v}_0^T$

SMOOTHING WITH BOX FILTER, RESULT

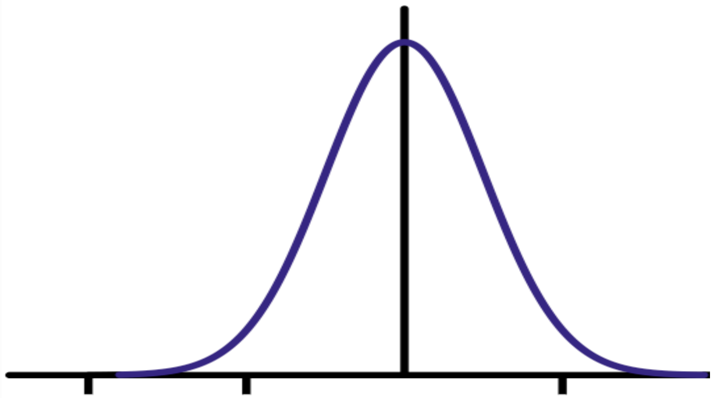


Source: D. Forsyth

GAUSSIAN FILTER

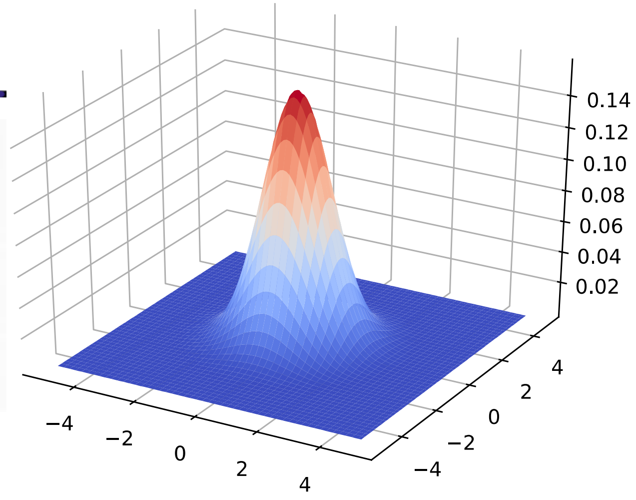
- Averaging gives as much emphasize for the distant neighbors as for the close ones => blurring
- Kernel values sampled from the 2D Gaussian function

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



Small std, weights of distant pixels very small => little effect

Large std, consensus of pixels => noise removed effectively but blurring exists



GAUSSIAN FILTER

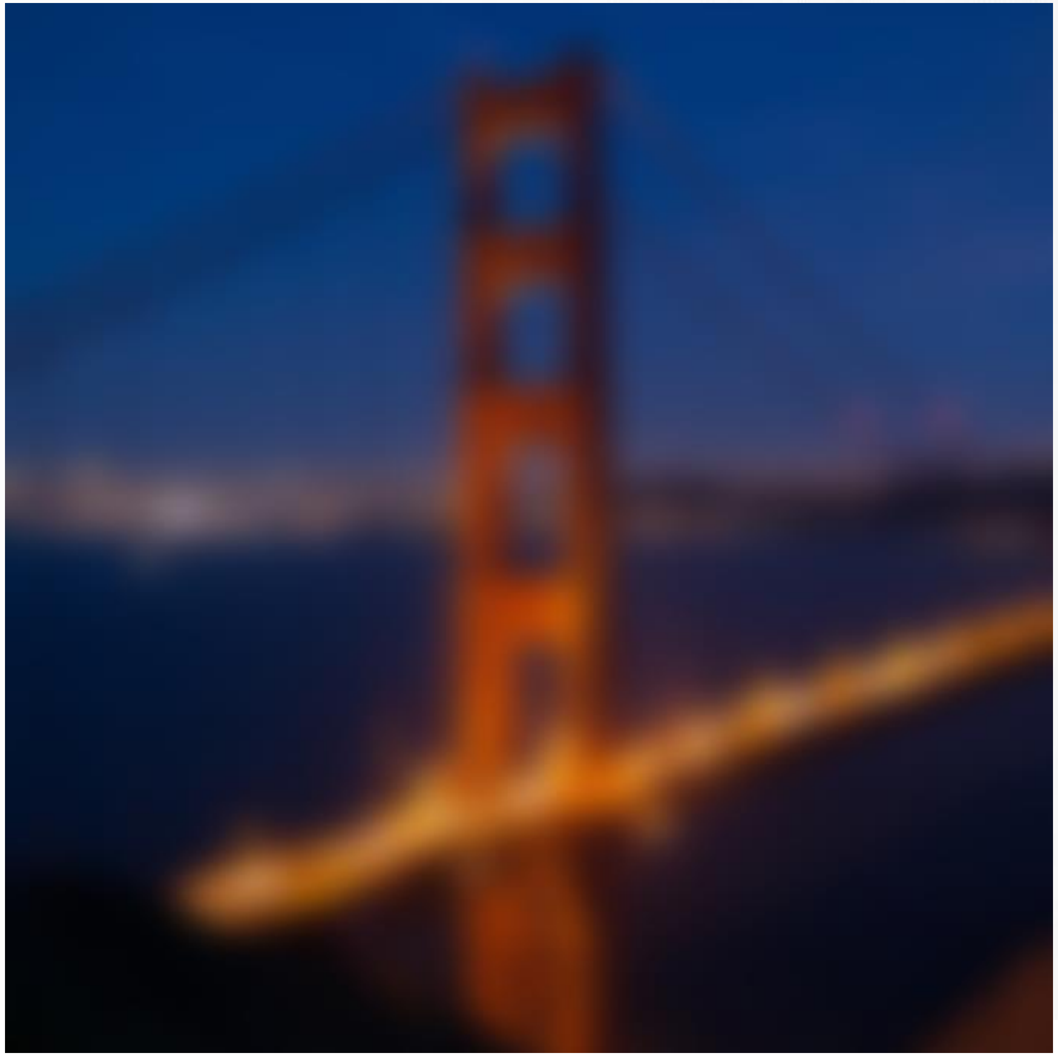
$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

Ignore factor in front, instead, normalize filter to sum to 1

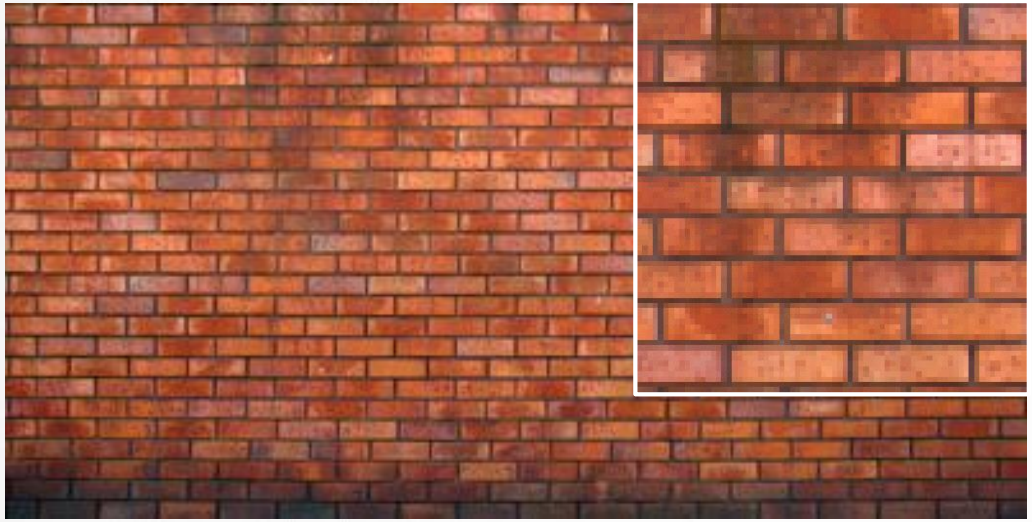
0.003	0.013	0.022	0.013	0.003
0.013	0.060	0.098	0.060	0.013
0.022	0.098	0.162	0.098	0.022
0.013	0.060	0.098	0.060	0.013
0.003	0.013	0.022	0.013	0.003

5x5, σ=1

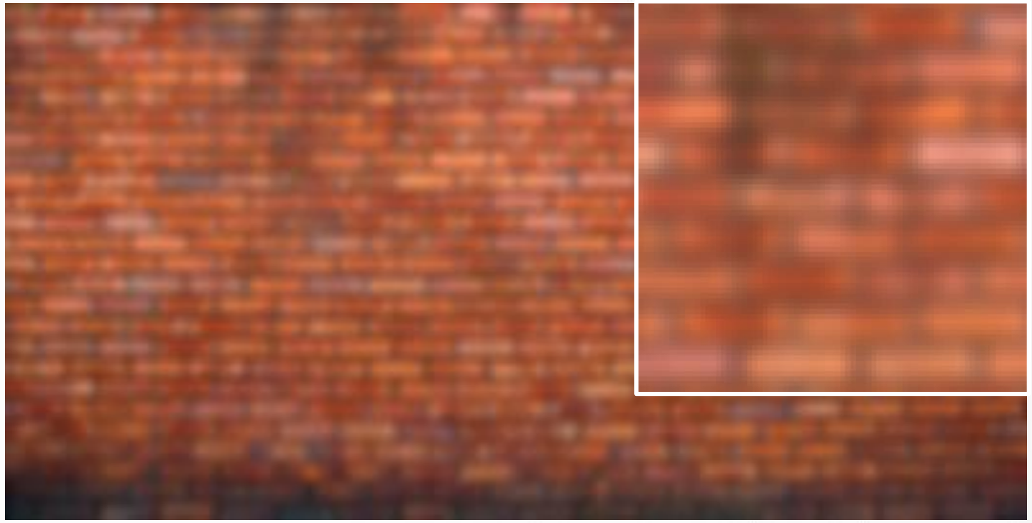
GAUSSIAN FILTERING EXAMPLE



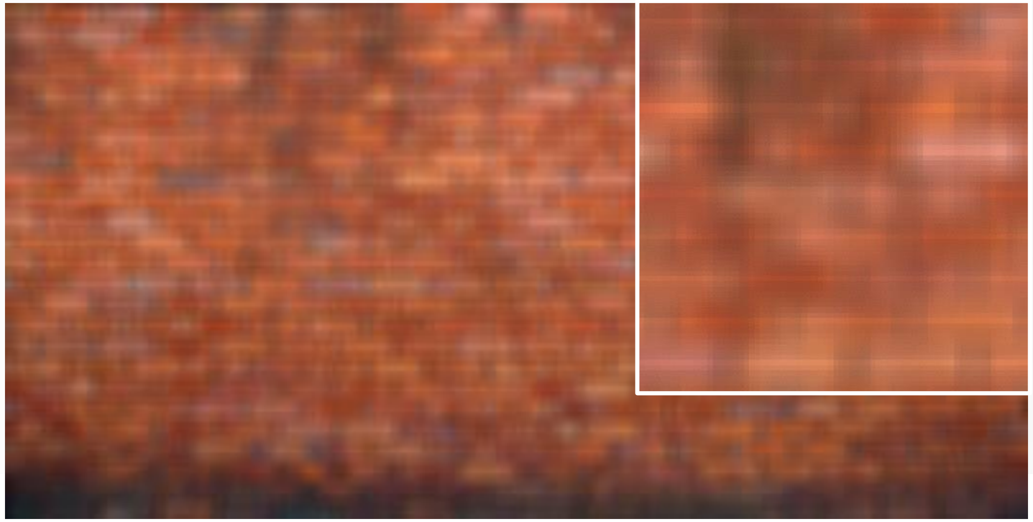
GAUSSIAN VS BOX FILTERING



original



7x7 Gaussian



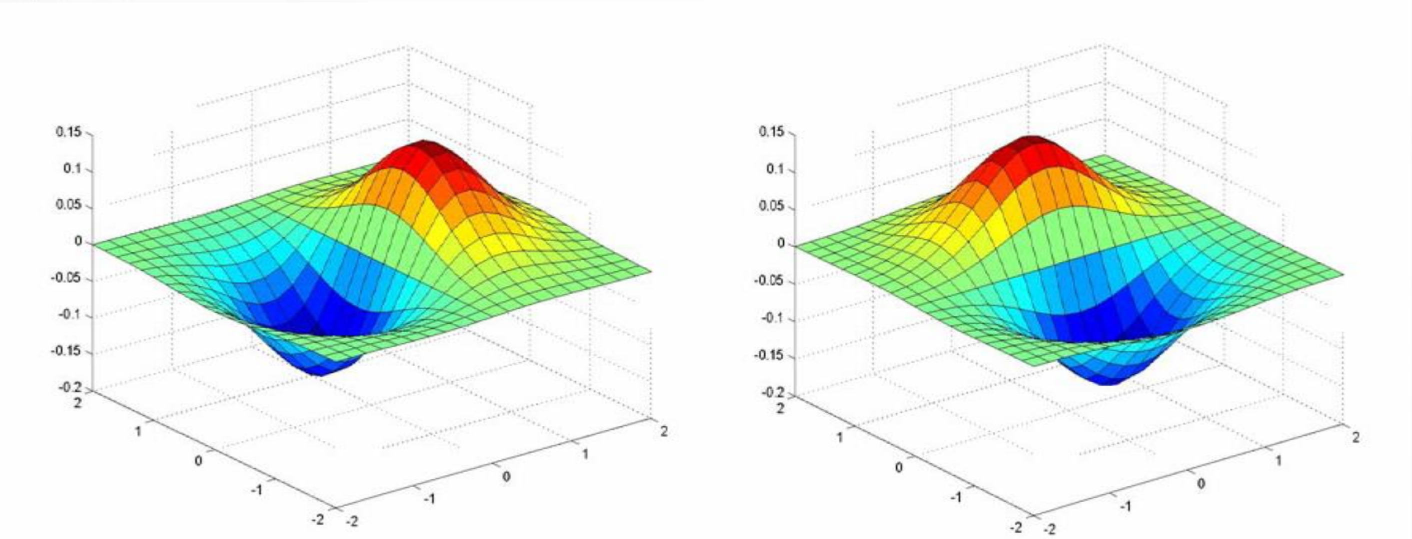
7x7 box

OTHER TYPES OF FILTERS

- Band-pass filtering (e.g. Sobel filters), sophisticated filters made by
  - First smoothing the image with a Gaussian filter
  - Then taking the first or second derivatives of the image (Laplacian operator)
- Non-linear filters for improved performance e.g. when the noise is not Gaussian distributed
  - e.g. Median filtering => median value from each pixels neighbourhood

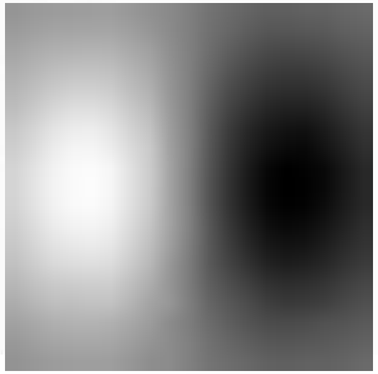
Shot noise | Gaussian filtered | Median filtered | Bilaterally filtered

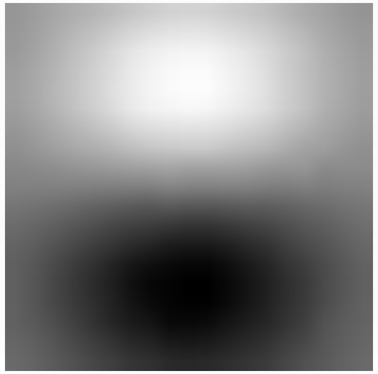
DERIVATIVE OF GAUSSIAN FILTER



x-direction

y-direction

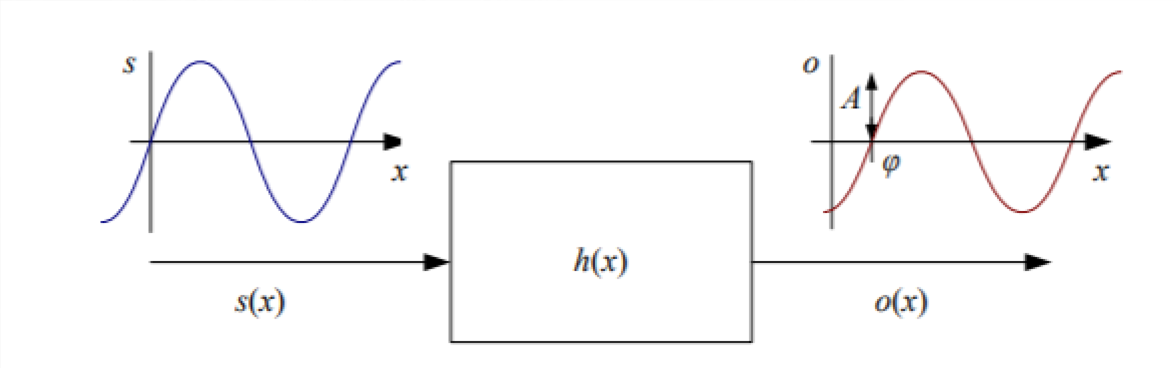




FOURIER TRANSFORMS 1

- So far the signal f(x,y) has been considered as being a weighted sum of a large on infinite number of small box functions
- We need to consider two problems:
  - we don’t know what is lost when representing the continuous signal with a discrete one => **sampling**
  - we don’t know how to shrink an image, can’t just take every kth pixel
- Problems are related to fast changes in the image => something important might be missed
- These can be studied by the change of basis => **Fourier transform**
  - basis will be a set of sinusoids, signal infinite weighted sum of an infinite number of sinusoids

FOURIER TRANSFORMS 2



The Fourier transform as a response of a filter h(x) to an input sinusoid s(x) yielding an output sinusoid o(x)

$$H(k) = \frac{1}{N} \sum_{x=0}^{N-1} h(x) e^{-j\frac{2\pi k x}{N}},$$

Discrete Fourier Transform (DFT)

- Fourier transform
  - used to understand the difference between continuous and discrete images => what has been lost
  - is simply a tabulation of the magnitude and phase response at each frequency

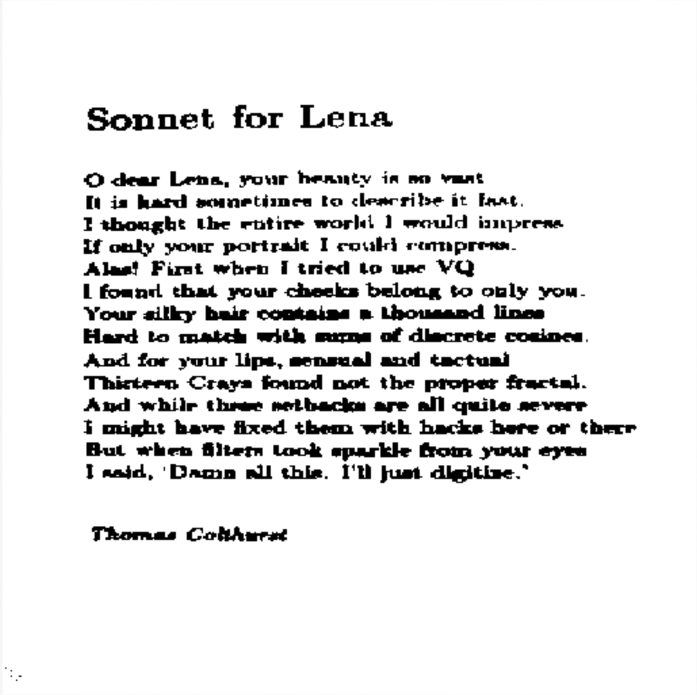
FOURIER TRANSFORMS 3

- A video showing how Fourier Transforms are used for images

  https://www.youtube.com/watch?v=gwaYwRwY6PU

FOURIER TRANSFORMS 4

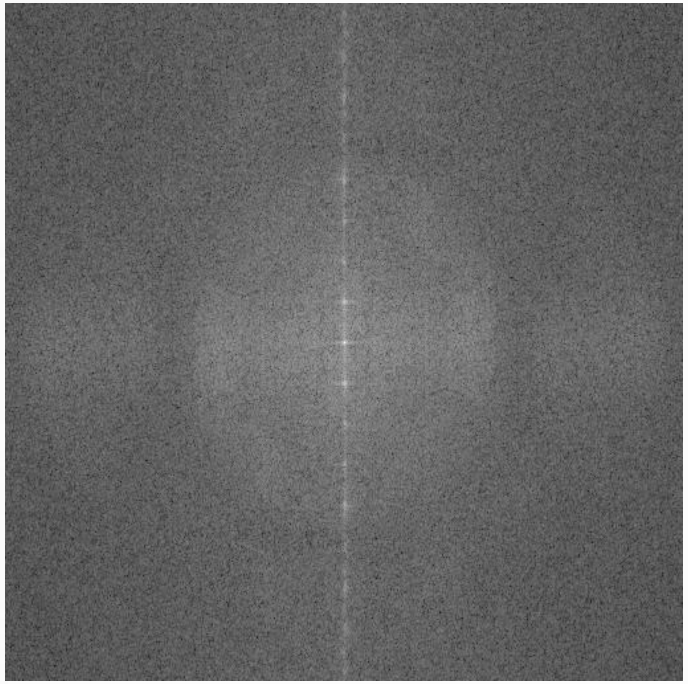
- A good tutorial is found from http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm



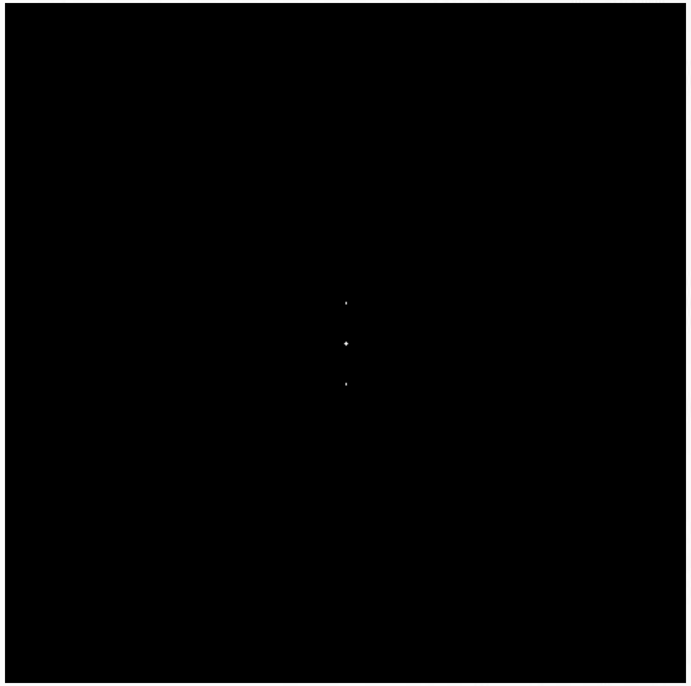
Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst



Logarithm of DFT's magnitude



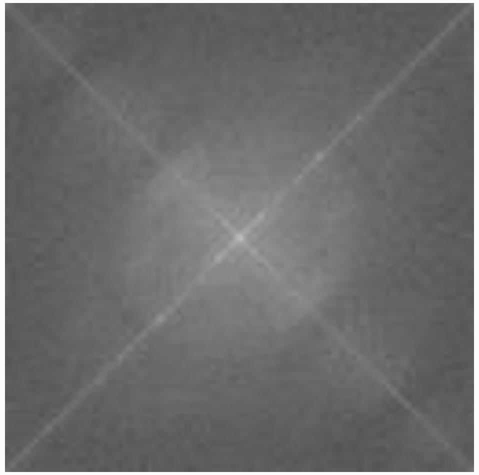
Thresholded magnitude of the Fourier image

For a square image of size N×N, the two-dimensional DFT is given by:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi \left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

FOURIER TRANSFORMS 5

- Rotate the image 45 degrees



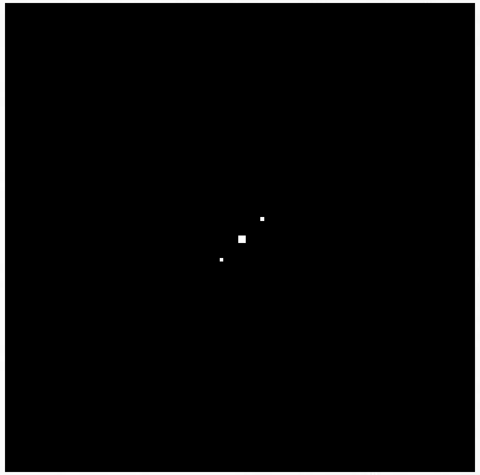


IMAGE SCALING

This image is too big to fit on the screen. How can we generate a half-sized version?

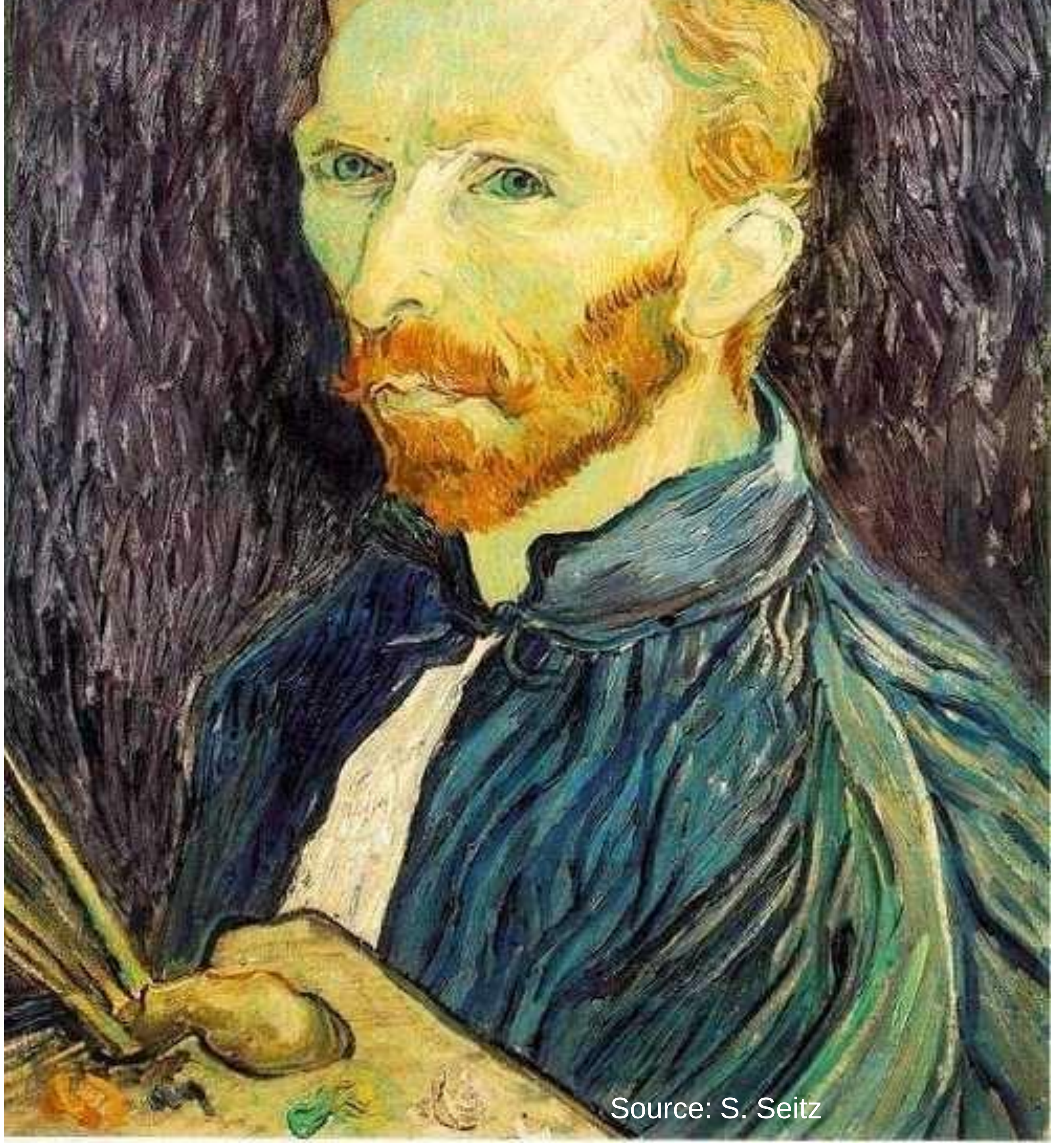
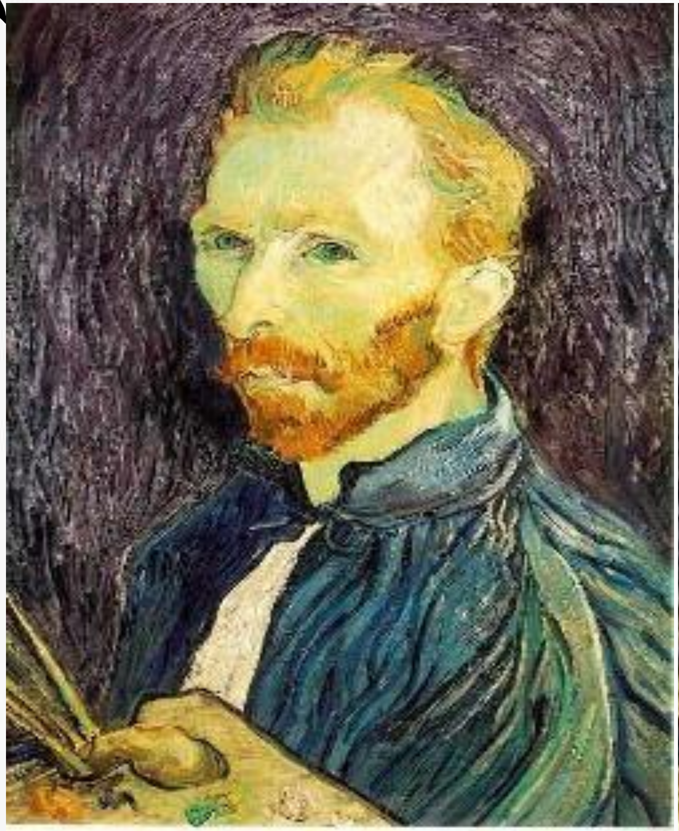


IMAGE SUB-SAMPLING



1/2

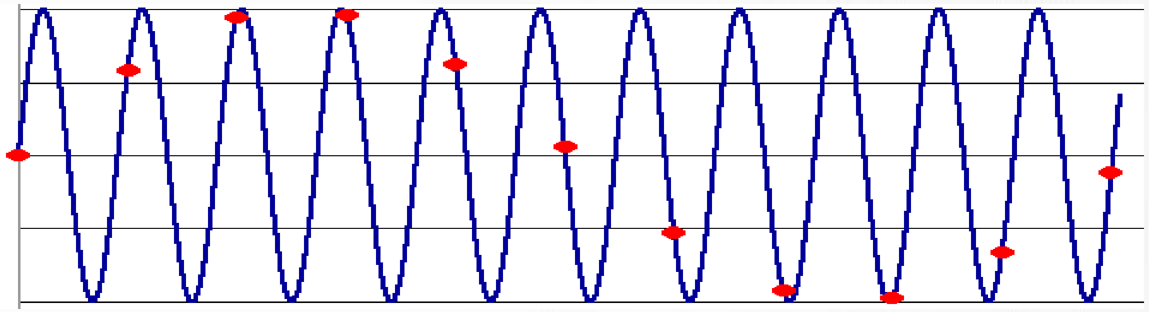
1/4 (2x zoom)

1/8 (4x zoom)

Throw away every other row and column to create a 1/2 size image: called *image sub-sampling*

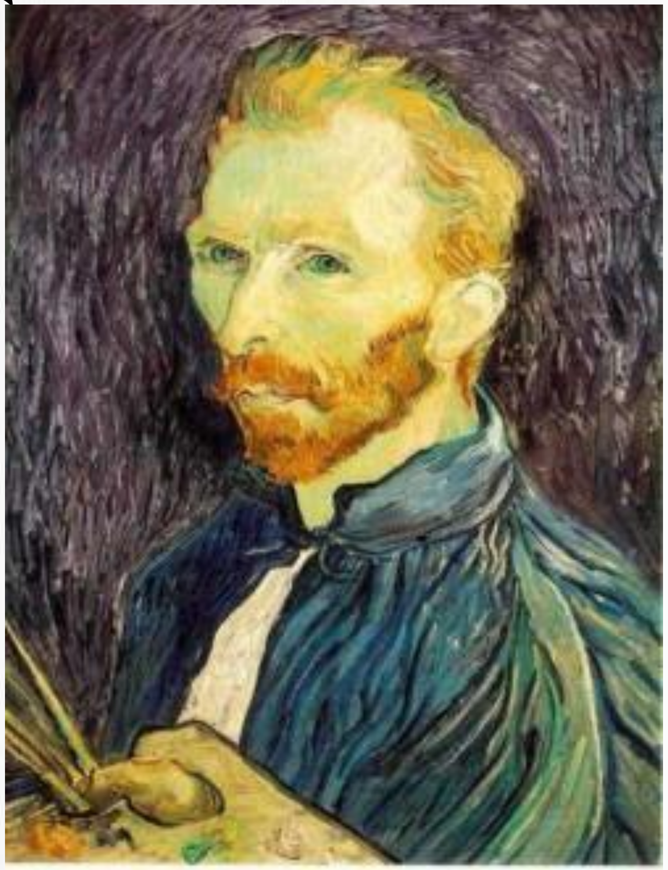
Source: S. Seitz

ALIASING



- Occurs when sampling rate is not high enough to capture the amount of detail in your image
- Can give the wrong signal/image—an *alias*
- To do sampling right, need to understand the structure of the signal/image
- To avoid aliasing:
  - Sampling rate ≥ 2 * max frequency in the image = ≥ two samples per cycle
  - This minimum sampling rate is called the **Nyquist rate**
  - k in DFT should be in the range k ϵ [- N/2, N/2]
- When downsampling by a factor of two the original image has frequencies that are too high

SUBSAMPLING WITH GAUSSIAN PRE-FILTERING



Gaussian 1/2

G 1/4

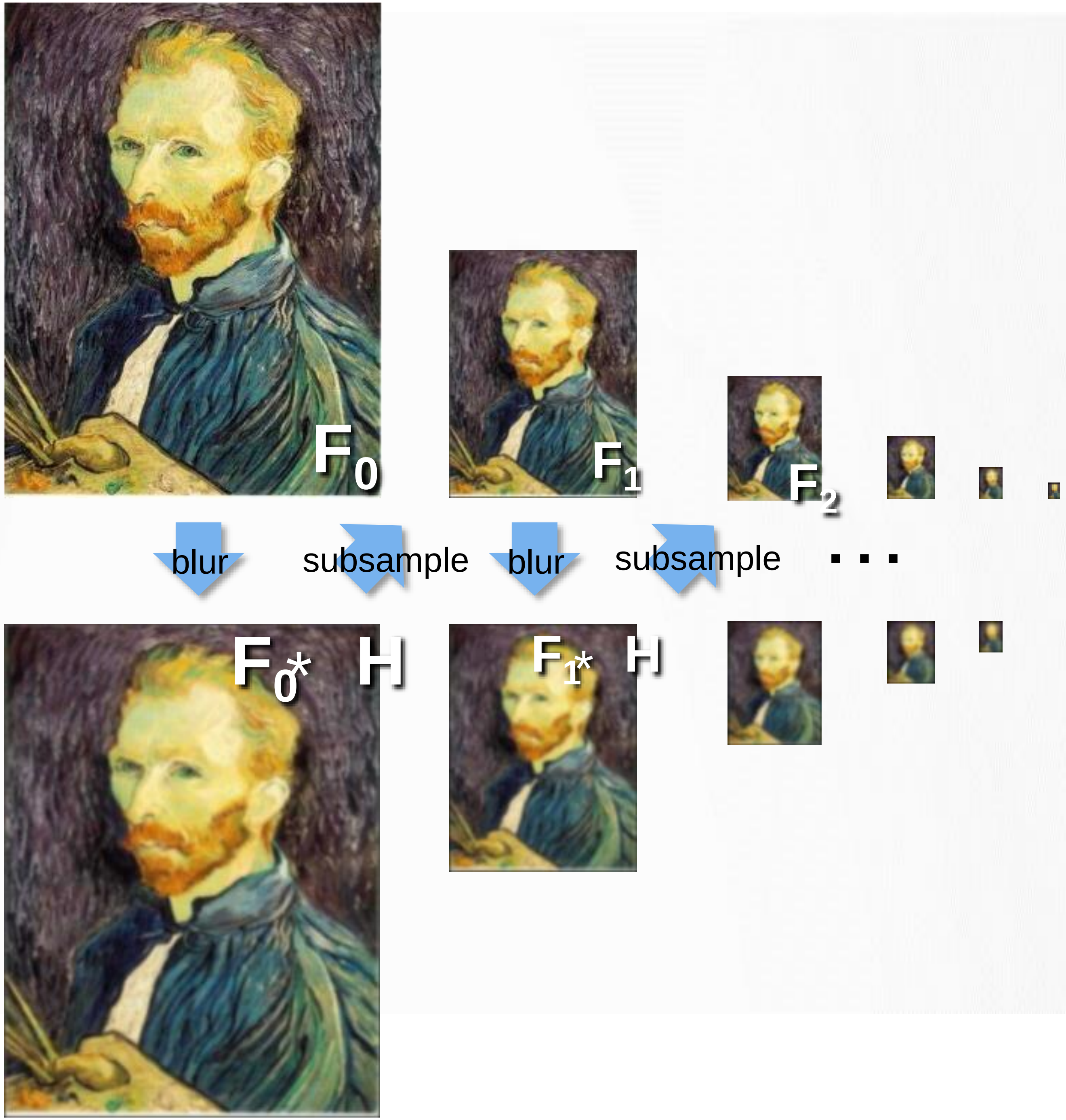
G 1/8

- Solution: filter the image, *then* subsample

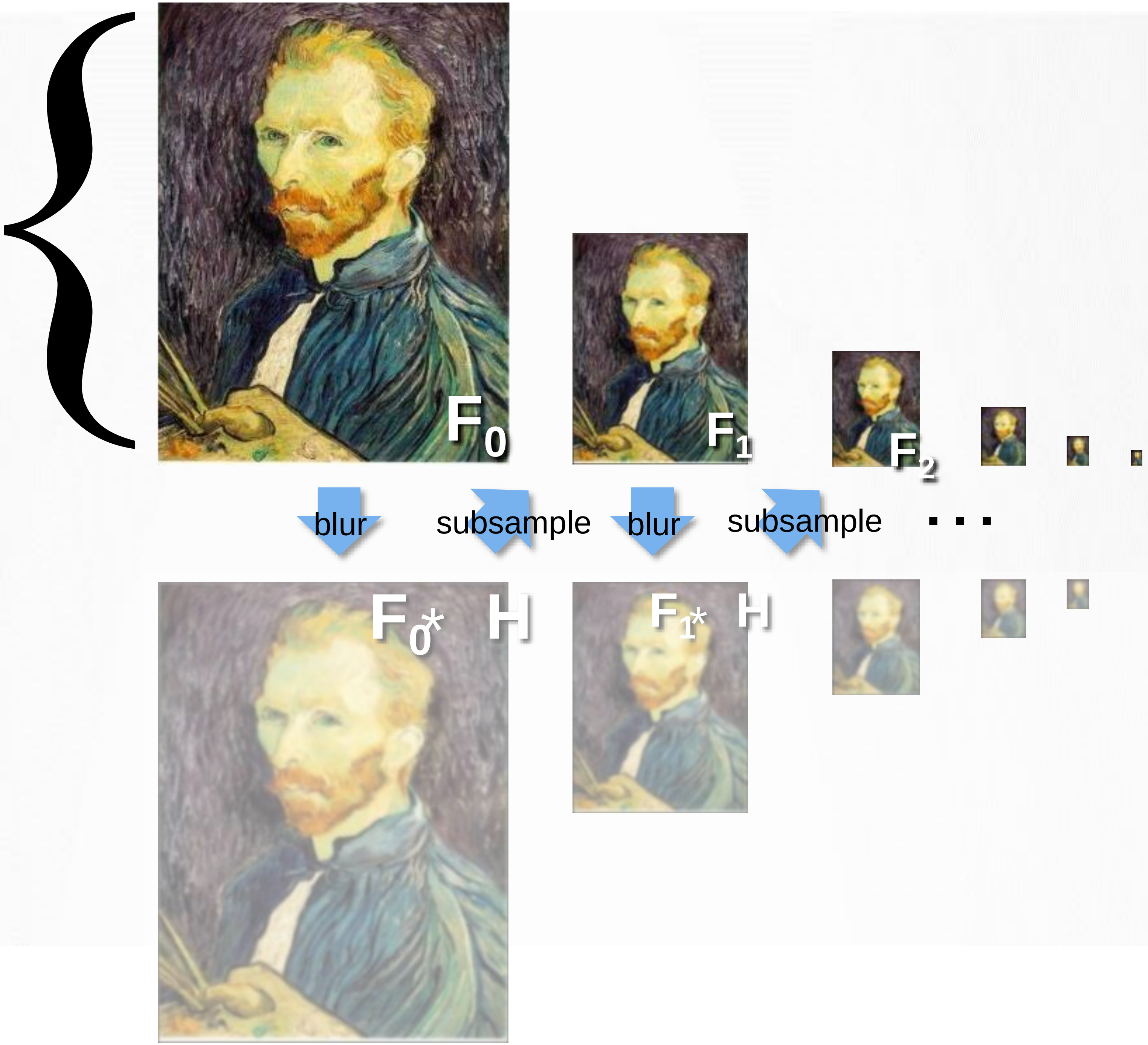
Source: S. Seitz

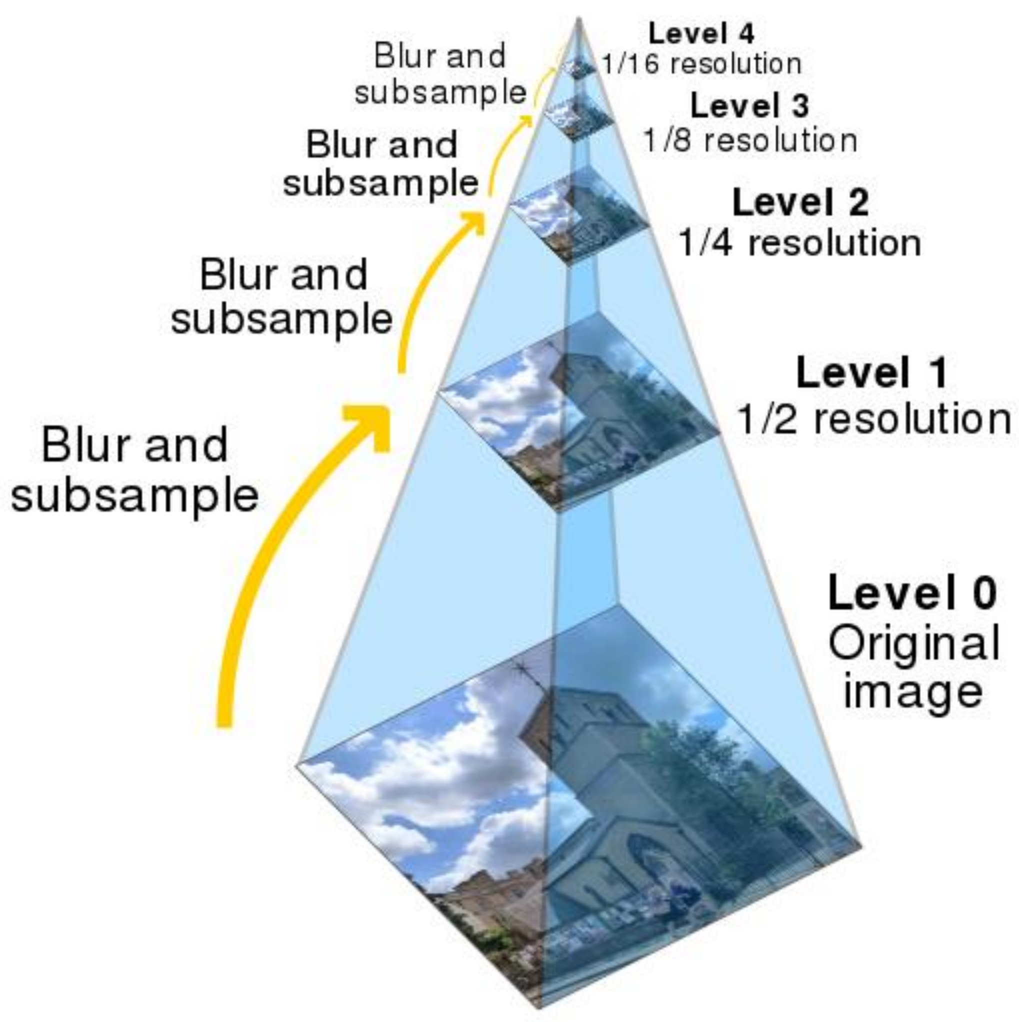
GAUSSIAN PRE-FILTERING

- Solution: filter the image, *then* subsample

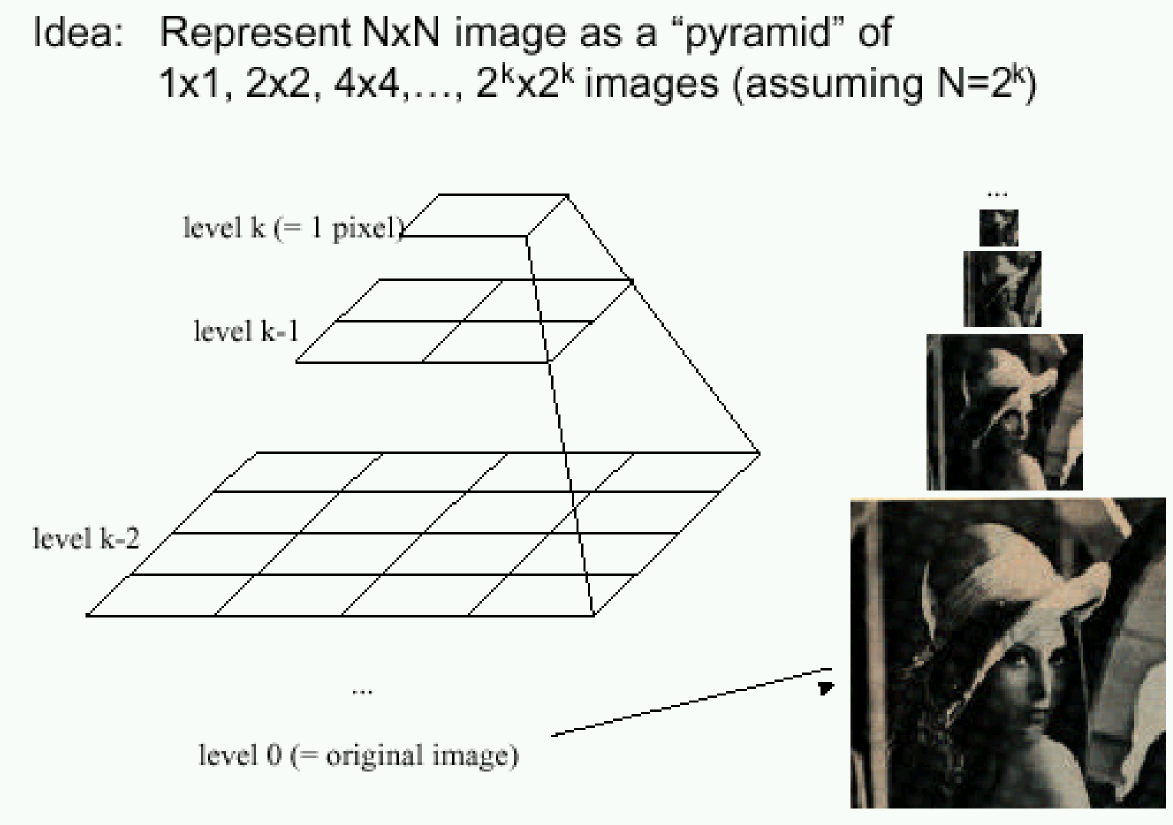


Gaussian pyramid




Blur and subsample
Blur and subsample
Blur and subsample
Blur and subsample
Level 4
1/16 resolution
Level 3
1/8 resolution
Level 2
1/4 resolution
Level 1
1/2 resolution
Level 0
Original image

GAUSSIAN PYRAMIDS [BURT AND ADELSON, 1983]



- A precursor to *wavelet transform*
- Gaussian Pyramids have all sorts of applications in computer vision

Source: S. Seitz

UPSAMPLING

This image is too small for this screen:

How can we make it 10 times as big?

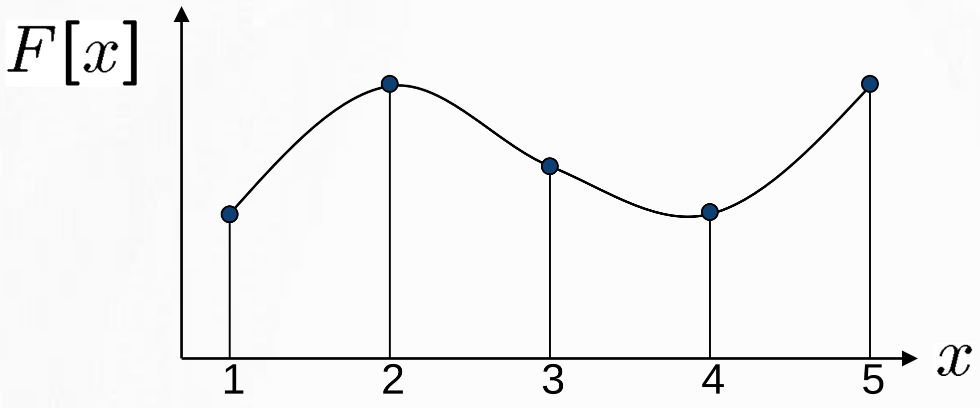
Simplest approach:

repeat each row

and column 10 times

(“Nearest neighbor interpolation”)

IMAGE INTERPOLATION



d = 1 in this example

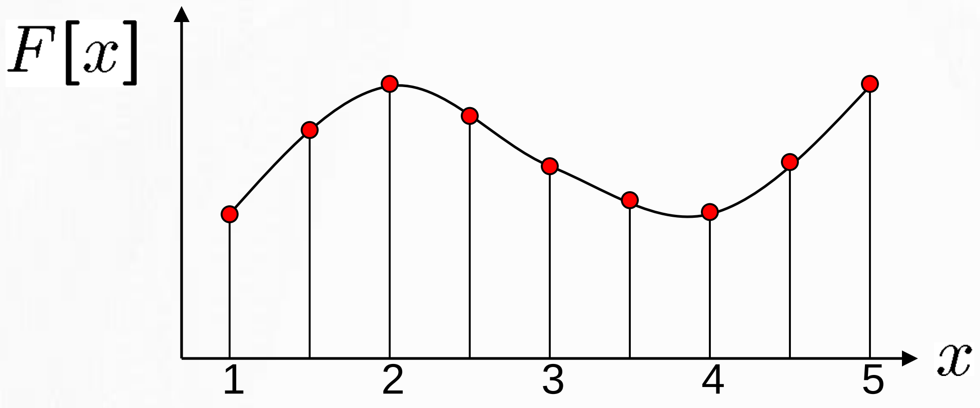
Digital images are formed as follows:

$$F[x,y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

IMAGE INTERPOLATION



d = 1 in this example

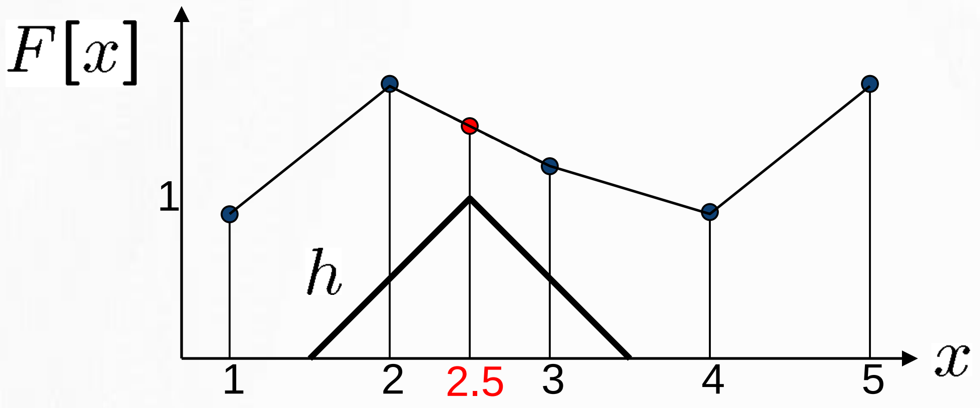
Digital images are formed as follows:

$$F[x,y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

IMAGE INTERPOLATION



d = 1 in this example

- What if we don't know f ?
  - Guess an approximation: $\tilde{f}$
  - Can be done in a principled way: filtering
  - Convert F to a continuous function:

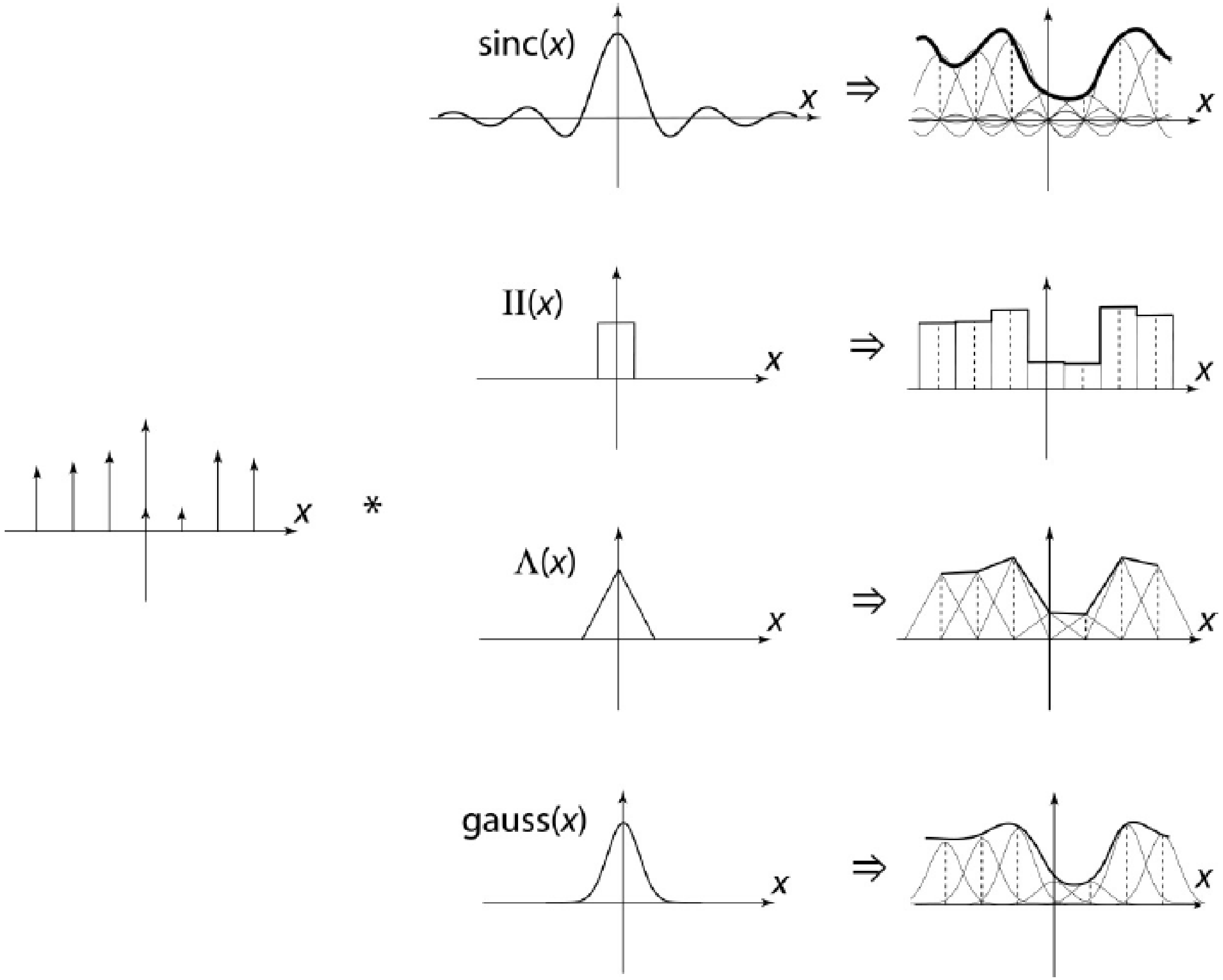
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - Reconstruct by convolution with a *reconstruction filter, h*

$$\tilde{f} = h * f_F$$

Adapted from: S. Seitz

Image interpolation



"Ideal" reconstruction

Nearest-neighbor interpolation

Linear interpolation

Gaussian reconstruction

Source: B. Curless

Image interpolation

Original image:  x 10

Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation


60˚ 10 1.2 N, 24˚ 57 18 E