COMPUTATIONAL VISION:

LINEAR FILTERS

Class 2: Computational Vision

Master in Artificial Intelligence

2021-2022

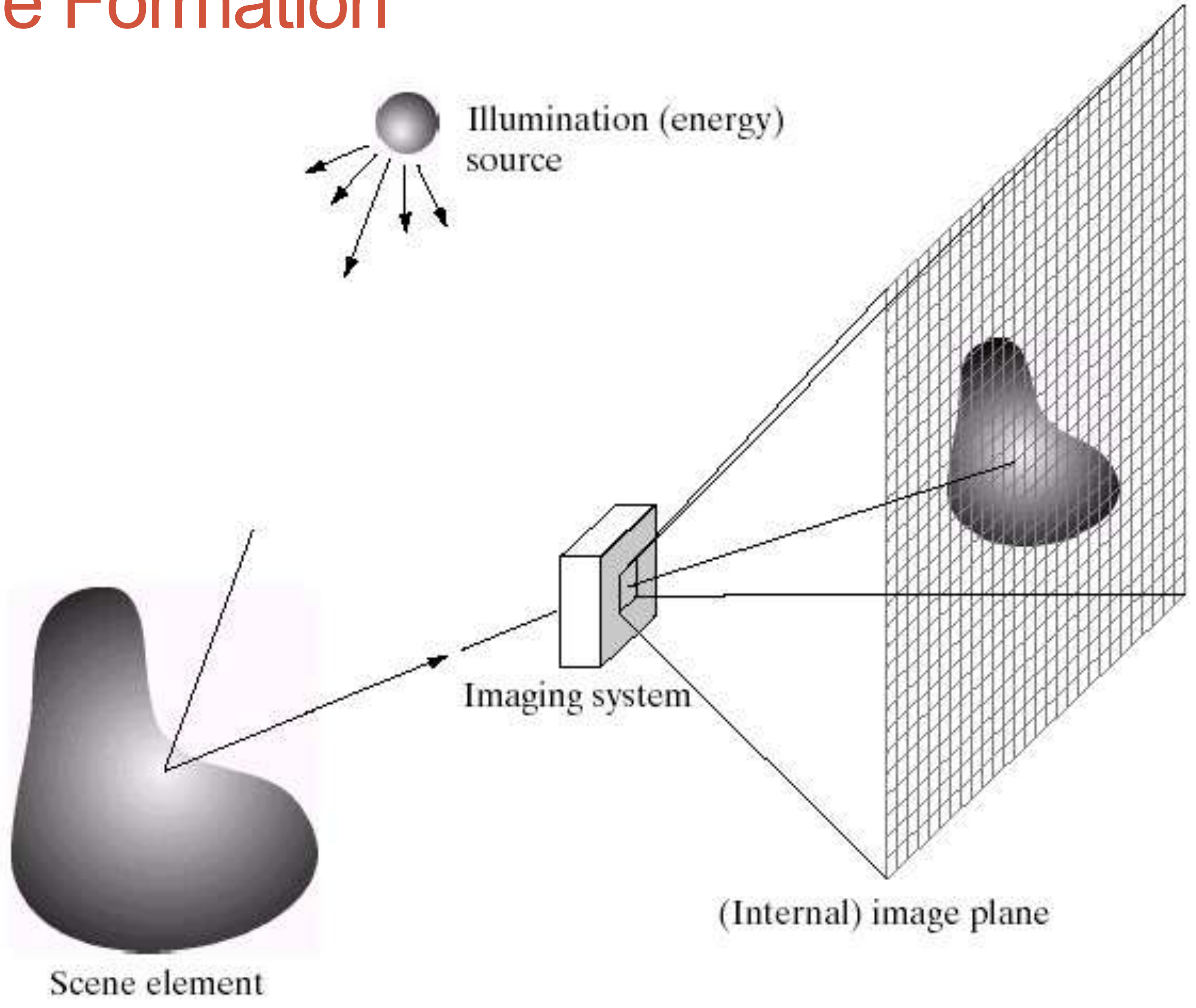
UNIVERSITAT DE BARCELONA

Outline

Digital images:

- Spatial and photometric resolution
- Histogram
- Image contrast enhancement
- Linear filters
- Examples: smoothing filters
- Convolution
- Gaussian filter

Image Formation



Slide credit: Derek Hoien

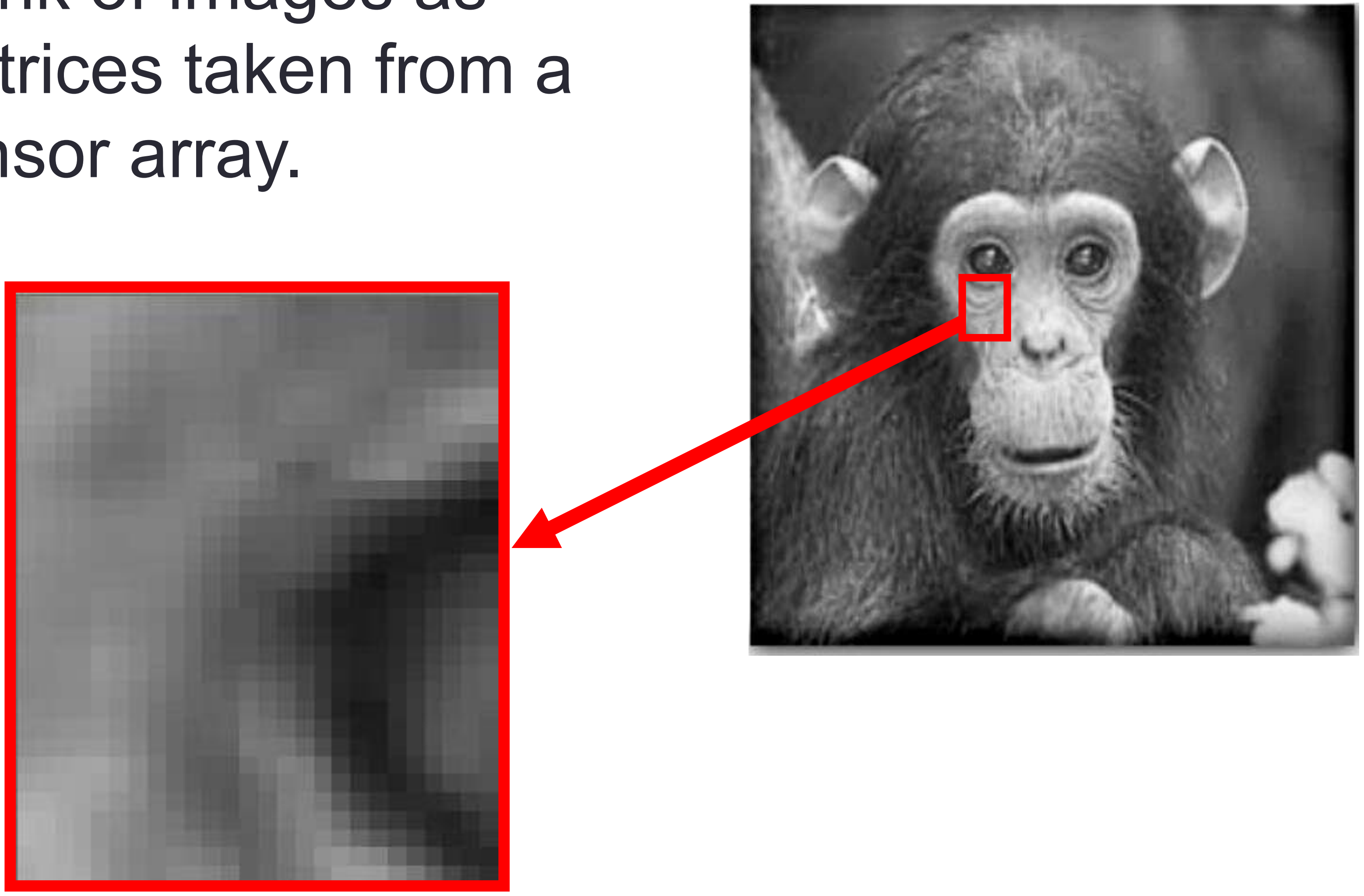
Digital camera

- The image I(x,y) measures how much light is captured
- at pixel (x,y): INTENSITY or BRIGHTNESS.
- • Proportional to the number of photons captured at the
- sensor element (CCD, CMOS, ..) in a time interval.
- 4

Slide by Steve Seitz

Digital images

Think of images as matrices taken from a sensor array.



K. Grauman

Digital images

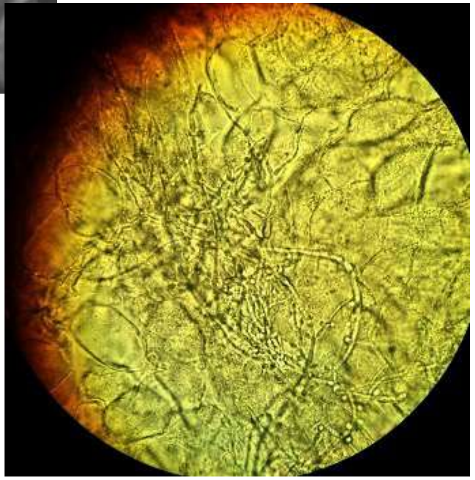
Different kinds of images:

•Natural images

• Other modalities:

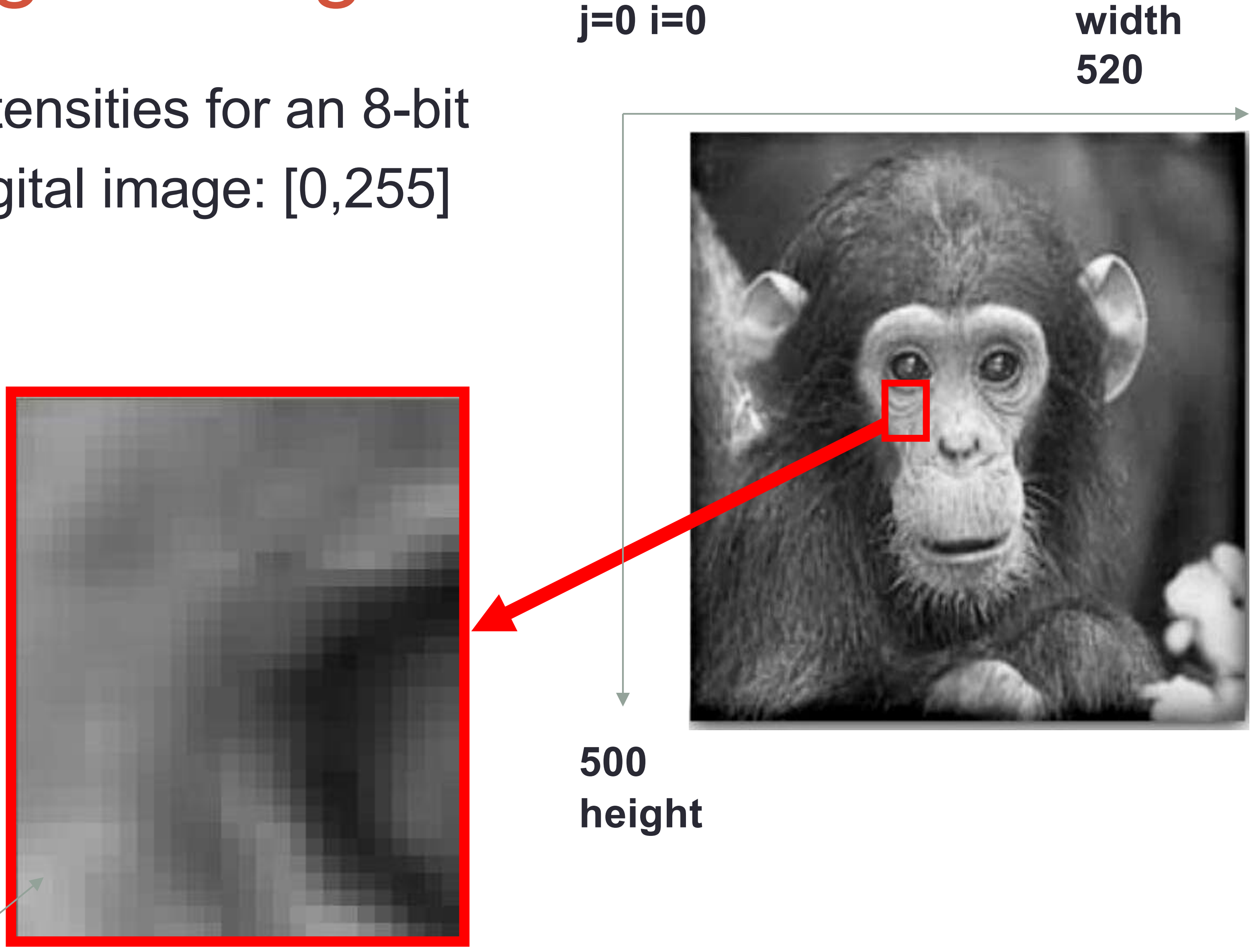
• X-rays, MRI…

• Light Microscopy, Electron Microscopy



Digital images

- Intensities for an 8-bit
- digital image: [0,255]

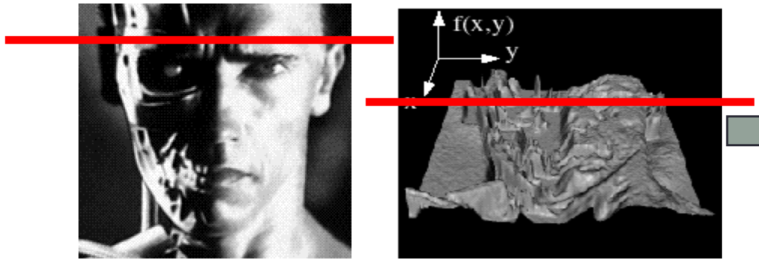


im[176,201] has value 164

K. Grauman

Digital images

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)



$j \rightarrow$

$i \downarrow$

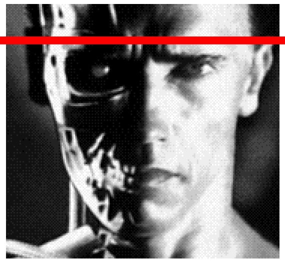
62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
~~10~~	~~58~~	~~107~~	~~46~~	~~46~~	~~0~~	~~0~~	~~48~~
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

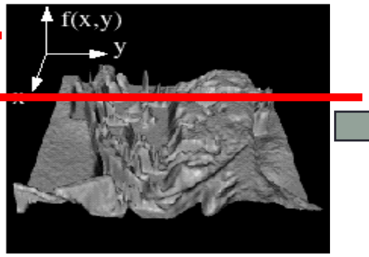
2D

Adapted from S. Seitz

Digital images

- Image is represented as a matrix of integer values.



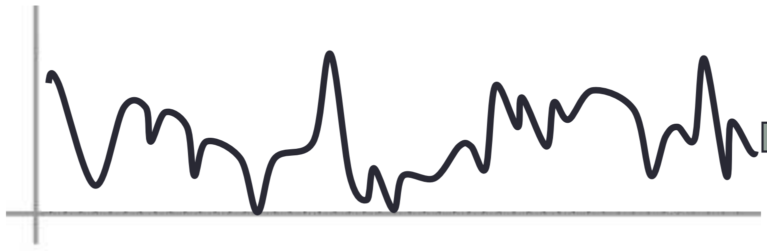


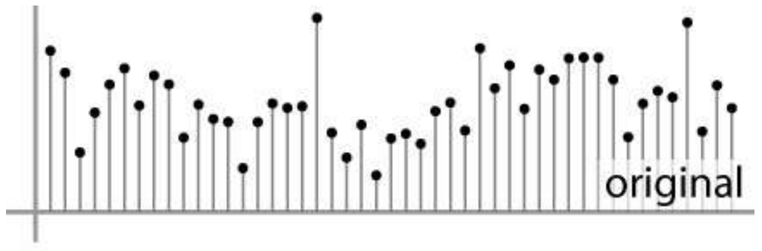
$j \rightarrow$

$i \downarrow$

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
~~10~~	~~58~~	~~107~~	~~46~~	~~46~~	~~0~~	~~0~~	~~48~~
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

2D



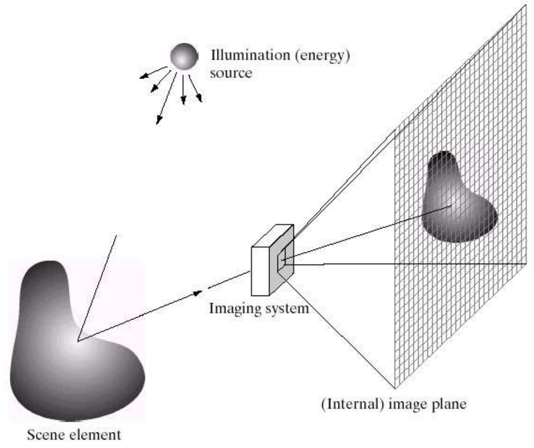


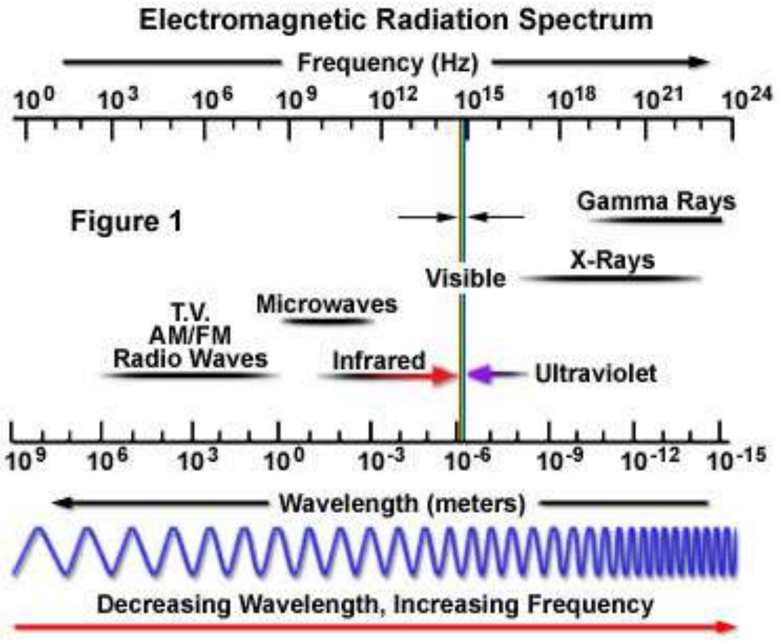
1D

Adapted from S. Seitz

How do we obtain color images?

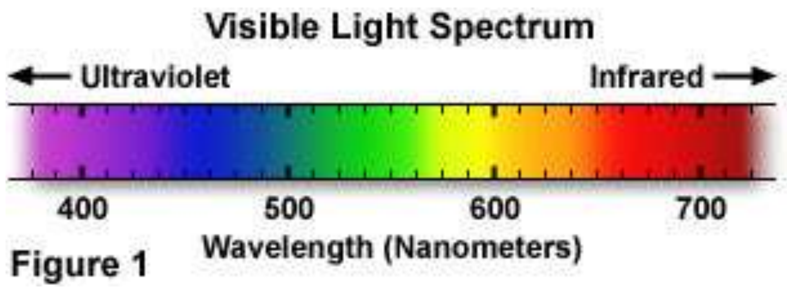
Light is an energy source that carries coded information about the world, which can be read from a distance through the images!





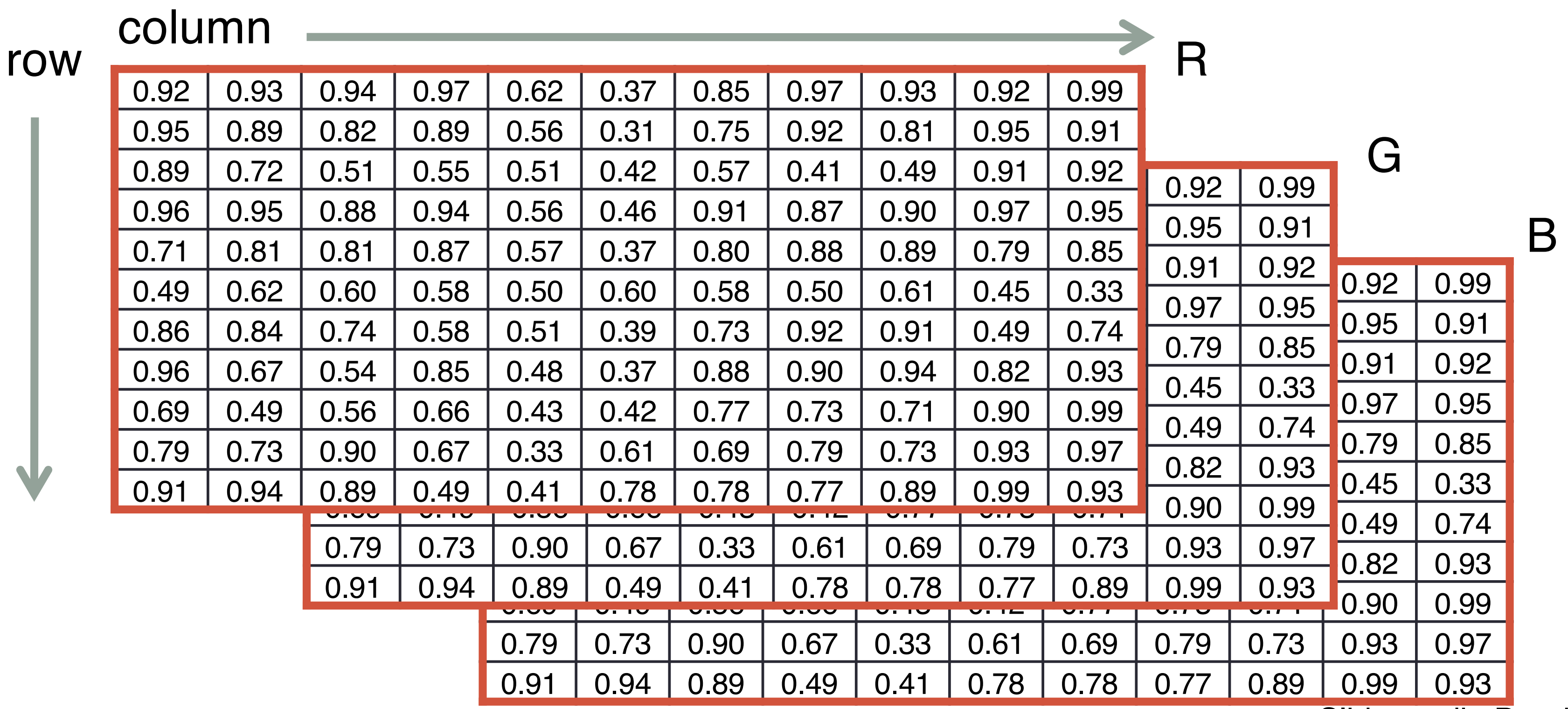
A typical human eye will respond to wavelengths from about 380 to 750 nm.

In order to get a full colour image, most sensors use filtering to look at the light in its three primary colours (Red, Green, Blue).
Once the camera records all three colours, it combines them to create the full spectrum.



Images in Skimage

- Images can be grey-value (1 channel) or color images (3 channels)
- Suppose we have an NxM RGB image called “im”
  - im[0,0,0] = top-left pixel value in R-channel
  - im[y, x, 2] = y pixels down, x pixels to right in the B-channel
  - im[N-1, M-1, 1] = bottom-right pixel in the G-channel



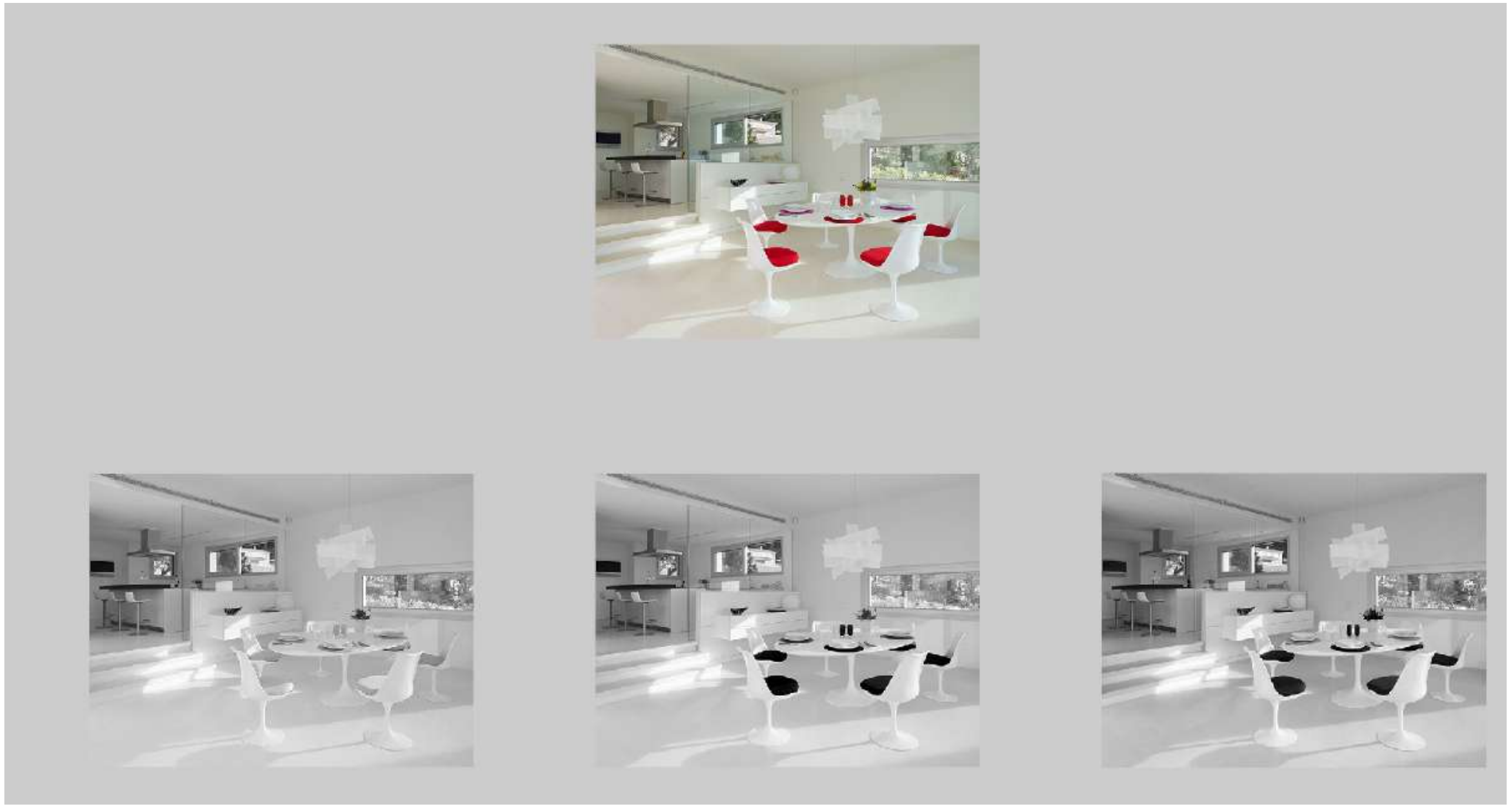
0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

									0.92	0.99
									0.95	0.91
									0.91	0.92
									0.97	0.95
									0.79	0.85
									0.45	0.33
									0.49	0.74
									0.82	0.93
									0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

									0.92	0.99
									0.95	0.91
									0.91	0.92
									0.97	0.95
									0.79	0.85
									0.45	0.33
									0.49	0.74
									0.82	0.93
									0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

Slide credit: Derek Hoien

Color images, RGB color space



R G B

Why are the chairs tops appearing in black in two of the channels?
What are the values of a white pixel in the color image?

Today

- What is an image?
  - Types, color vs gray level
- How to measure the quality of an image?
  - Spatial and photometric resolution
  - Histogram and image contrast enhancement
- How to process by a linear filter
  - Examples: smoothing filters
  - Convolution / correlation
  - Mean and median filters
  - Linear filters with Gaussians

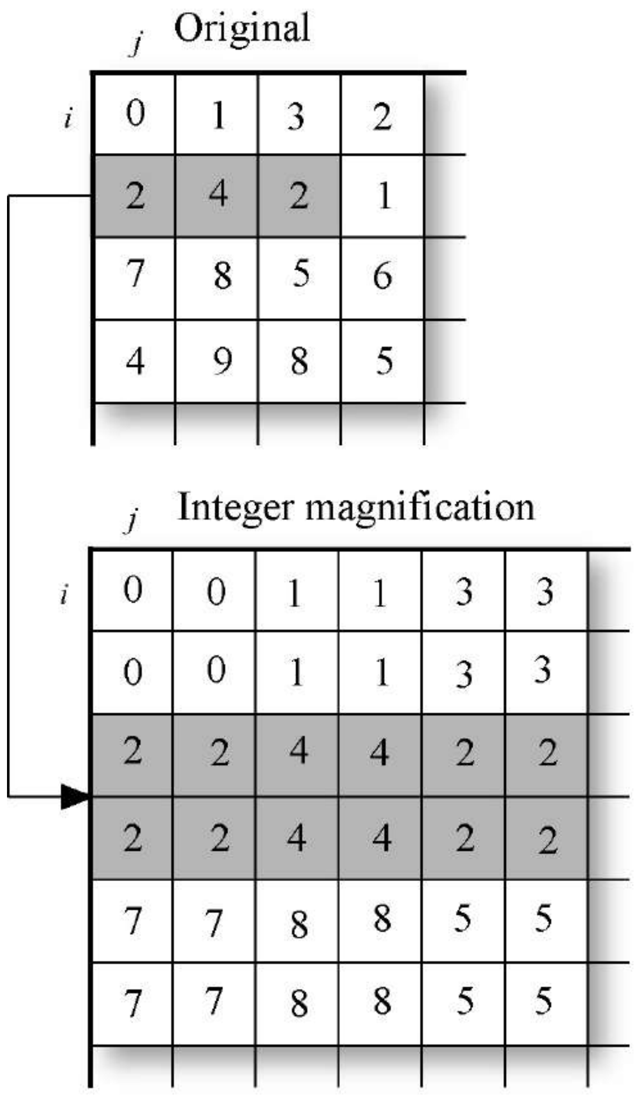
Spatial resolution

- **Sensor resolution**: size of real world scene element that images to a single pixel
- **Image resolution**: number of pixels

[fig from Mori et al]

It influences what analysis is feasible, it affects best representation choice.

Image Magnification



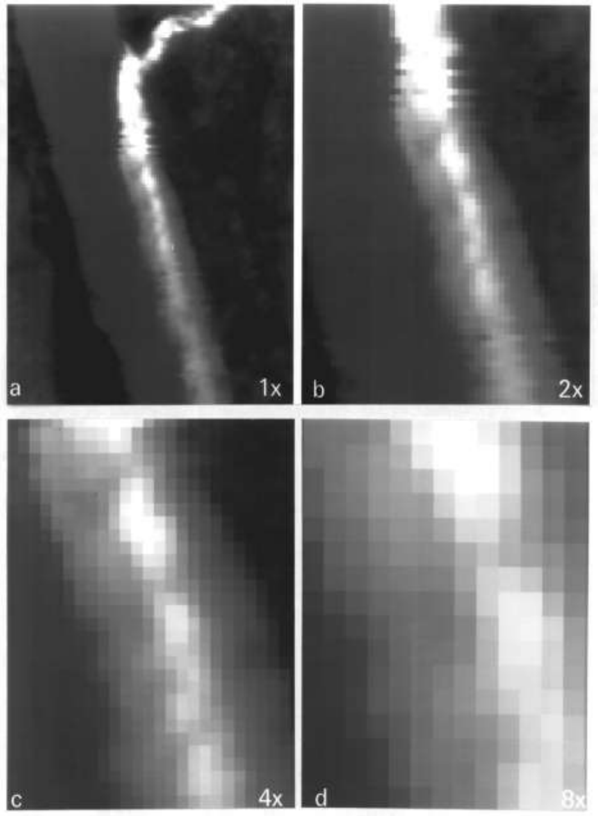
Original

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

Integer magnification

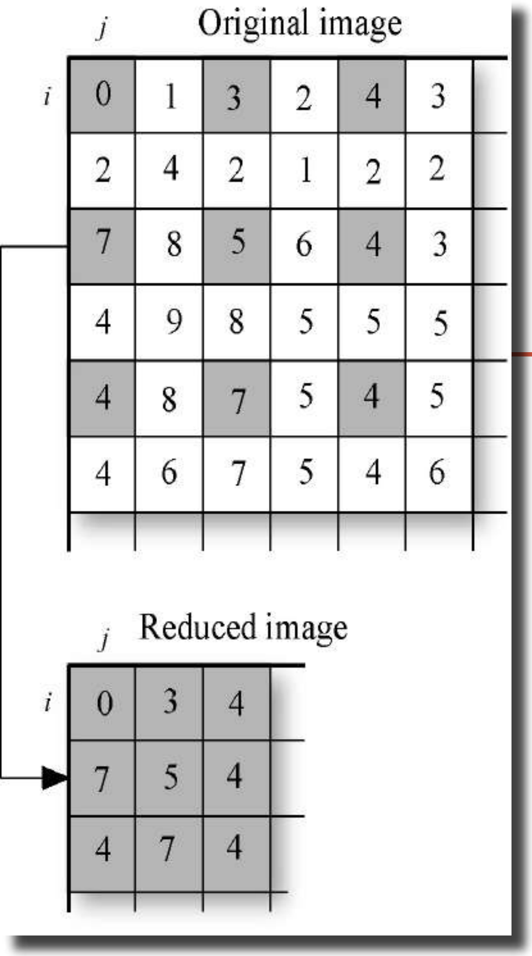
0	0	1	1	3	3
0	0	1	1	3	3
2	2	4	4	2	2
2	2	4	4	2	2
7	7	8	8	5	5
7	7	8	8	5	5

Magnification of Predawn Thermal Infrared Data
of A Thermal Plume in the Savannah River



rescale() in
scikit-image

Image reduction

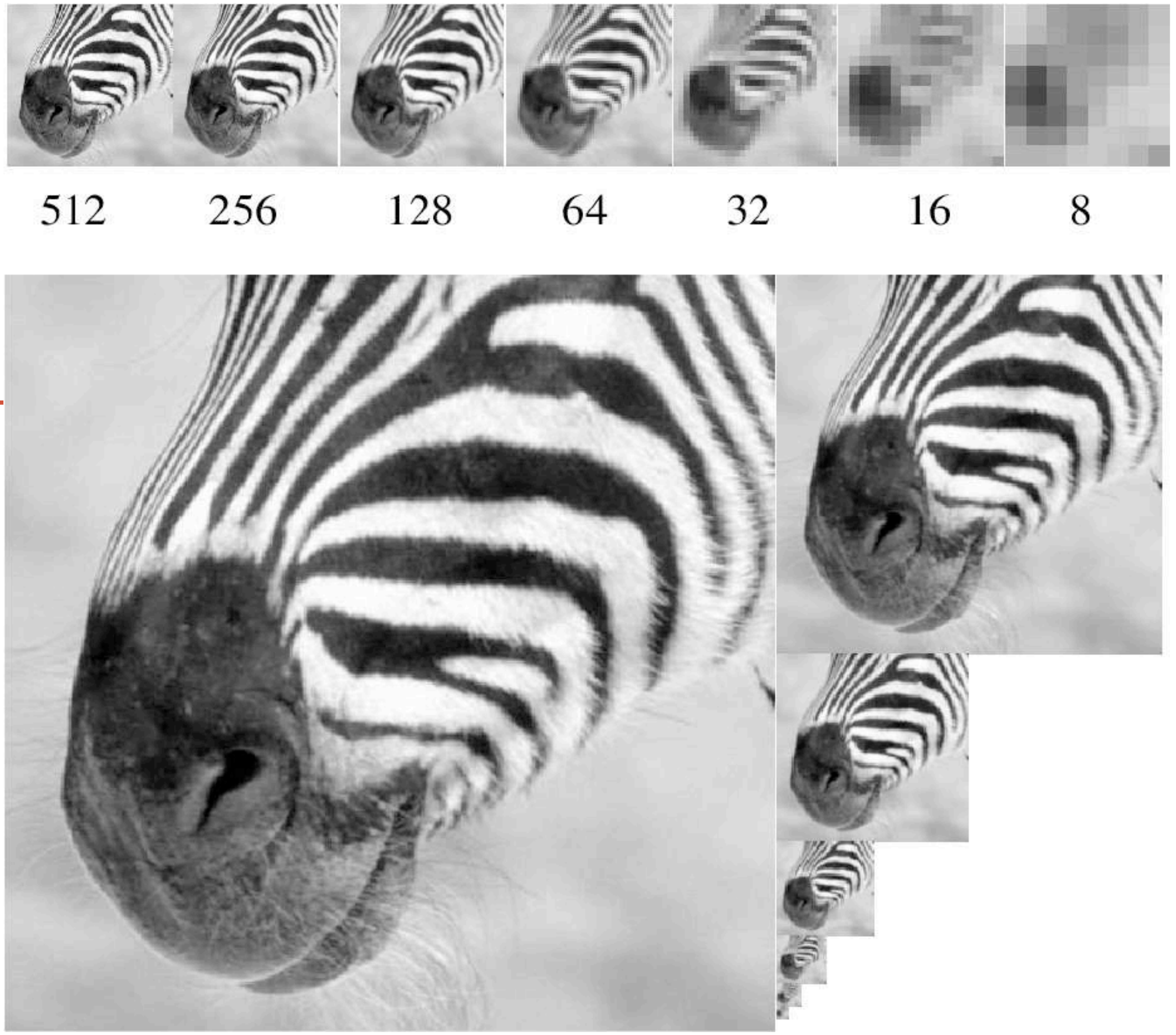


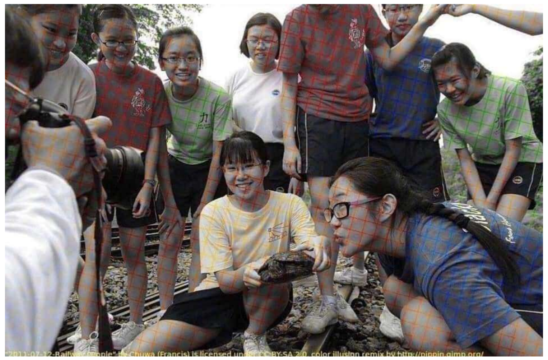
Original image

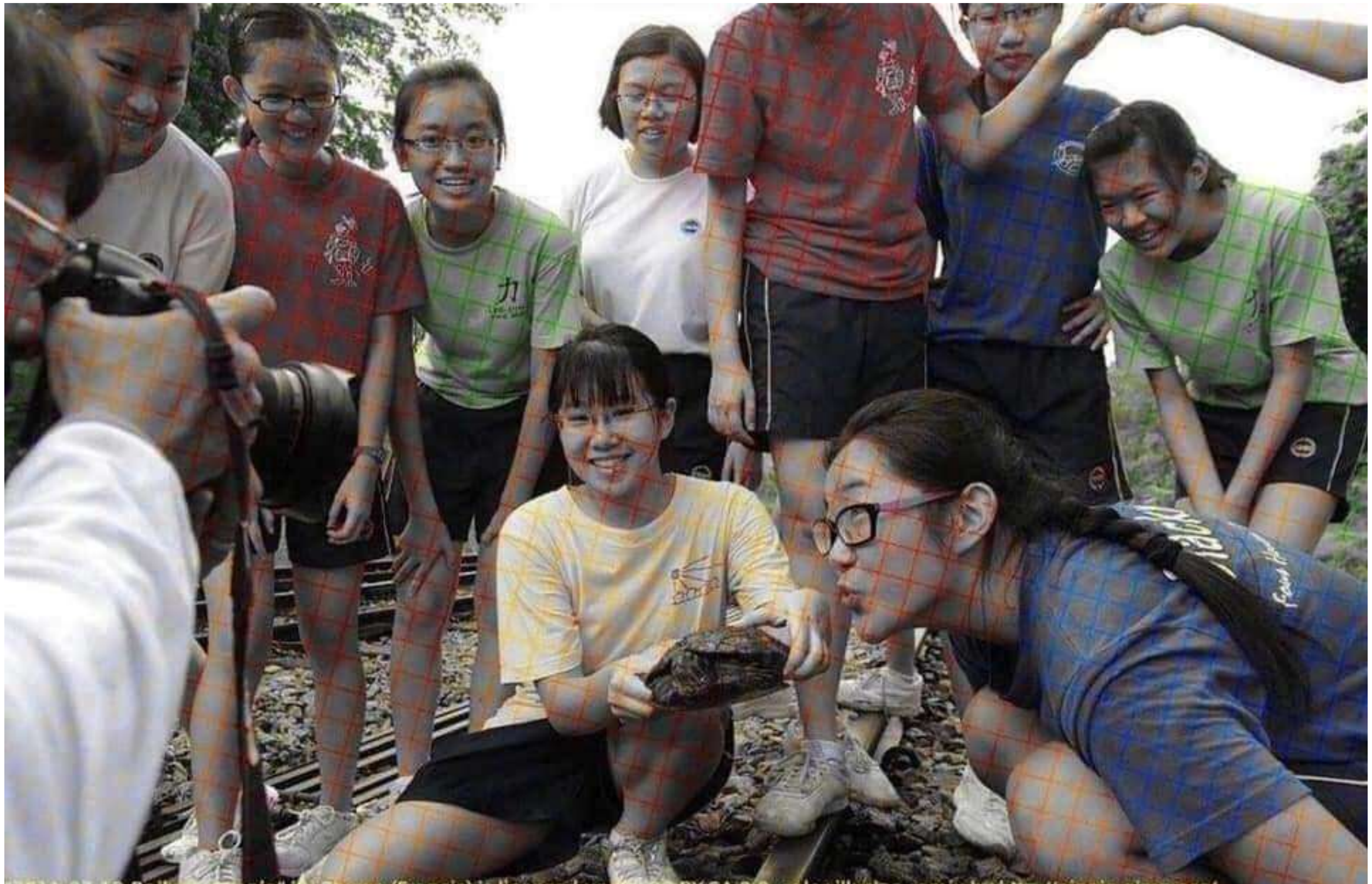
0	1	3	2	4	3
2	4	2	1	2	2
7	8	5	6	4	3
4	9	8	5	5	5
4	8	7	5	4	5
4	6	7	5	4	6

Reduced image

0	3	4
7	5	4
4	7	4

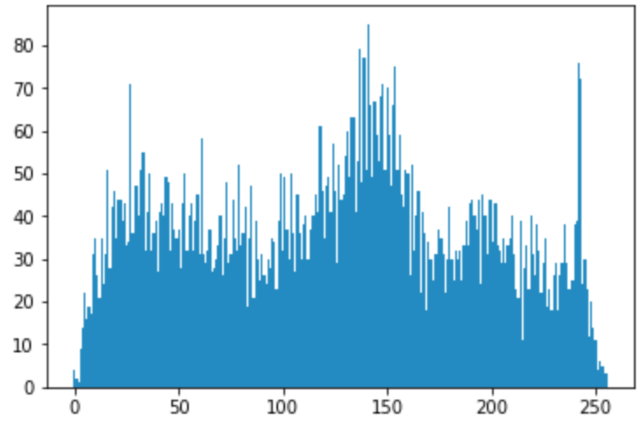



"2011-07-12-Railway-People" by Chuwa (Francis) is licensed under CC BY-SA 2.0, color illusion remix by http://pippin.gimp.org/



This is actually a black and white picture. An artist drew color grid lines and your brain is filling the colors in.

Photometric resolution



Given an image of type uint8, how many grey levels we can have at most?

A histogram of an image represents the frequencies of the image gray levels.

- Does it depend on the spatial distribution?
- Can it be considered as a measure of image quality?

The number of different grey levels (different pixel values in each color channel) determines the photometric resolution of the image.

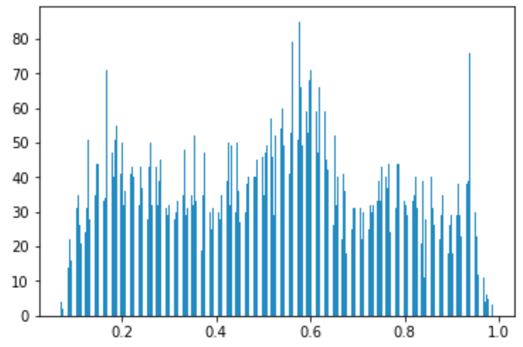
Histogram

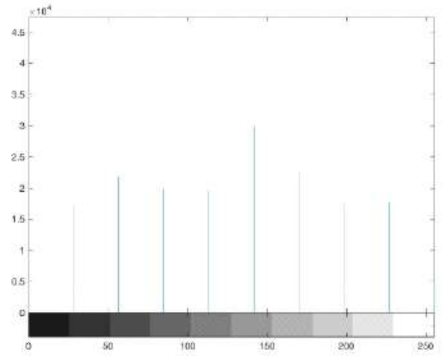
```
mm=np.zeros((256,
256,3) dtype=np.uint8)
```

-> Creates an image of what color?

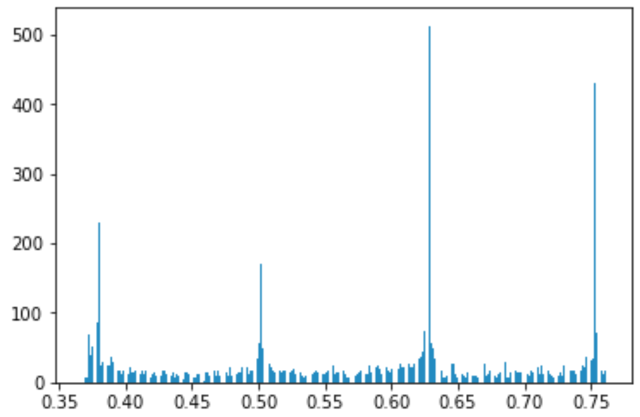
• Given an image of type uint8, how many grey levels we can have at most?

• We can change the number of bins of the histogram:



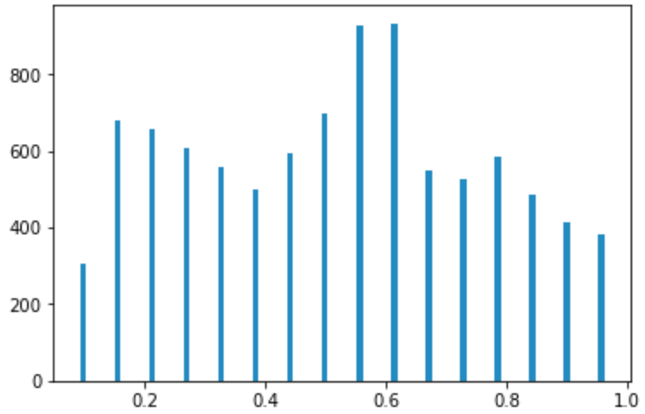


Histogram

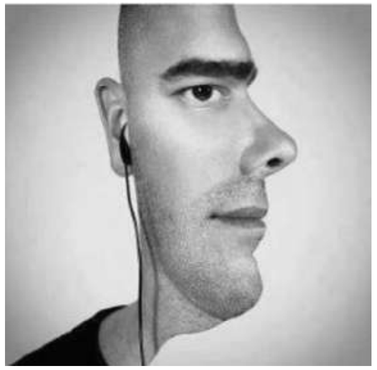
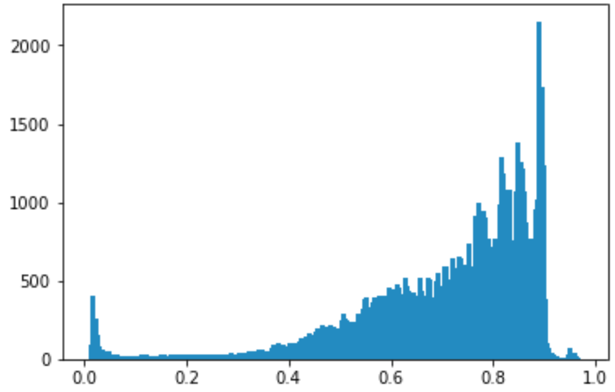


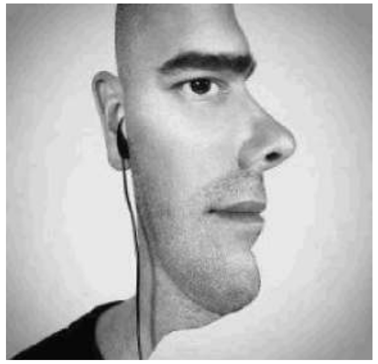
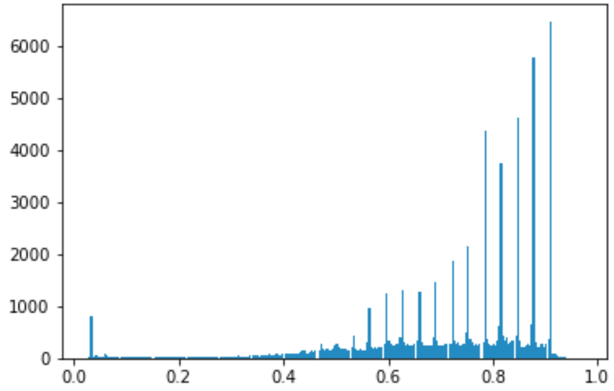
How will the histogram of the right image look?

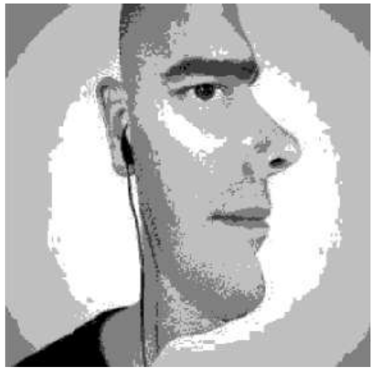
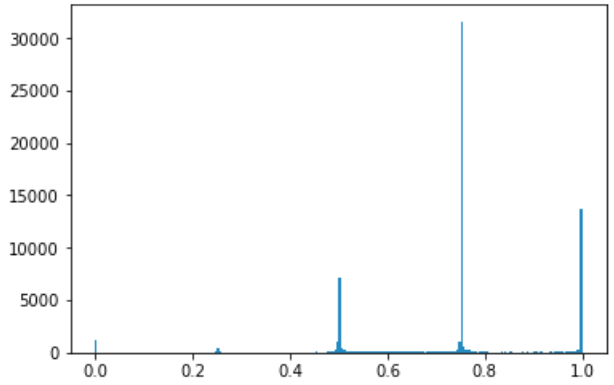
Histogram of 16 bins



Histogram: How should the histograms look?

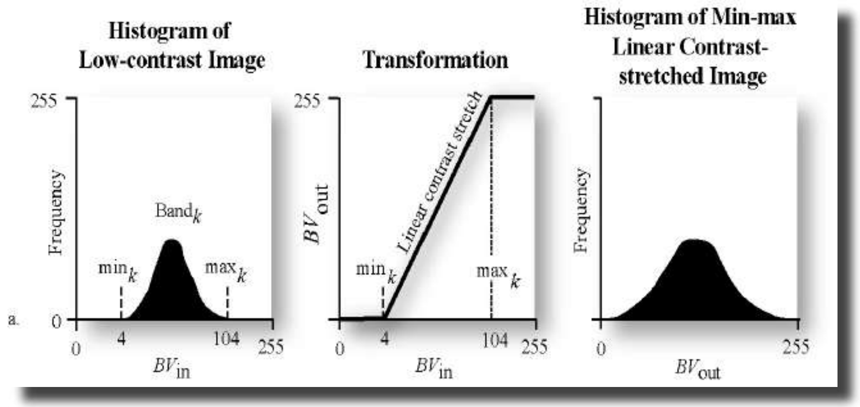



Histogram manipulation for image contrast enhancement

- Minimum-maximum contrast stretch
- Percentage linear contrast stretch

Histogram manipulation for contrast enhancement



Multiply the image to augment its contrast:

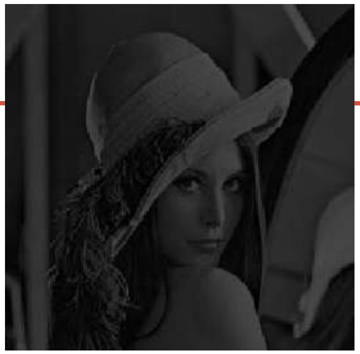
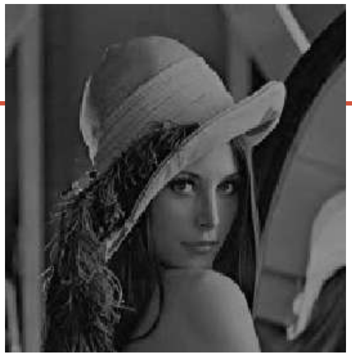
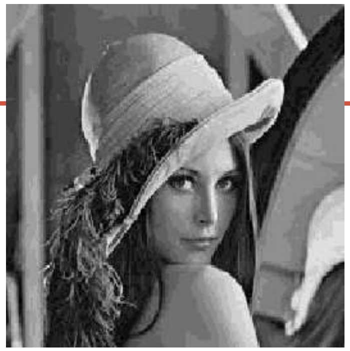
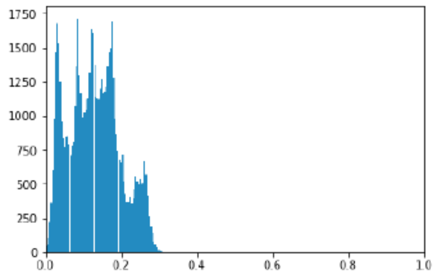
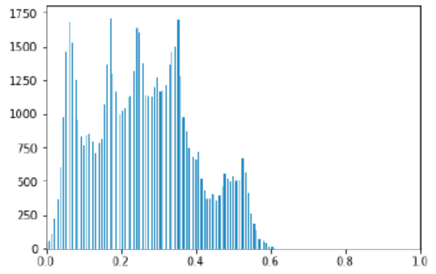
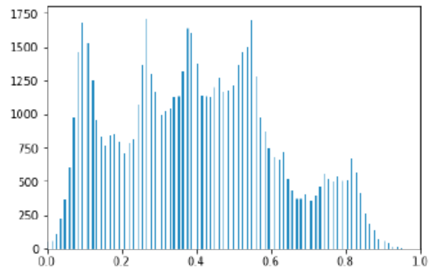
$$BV_{out} = \left(\frac{BV_{in} - \min_k}{\max_k - \min_k} \right) quant_k$$

where:

- BV_{in} is the original input brightness value (i.e. the original image)
- $quant_k$ is the range of the brightness values that can be displayed on the CRT (eg 255),
- min_k is the minimum value in the image,
- max_k is the maximum value in the image, and
- BV_{out} is the output brightness value.

Histogram manipulation for contrast enhancement

$$BV_{out} = \left(\frac{BV_{in} - \min_k}{\max_k - \min_k} \right) quant_k$$

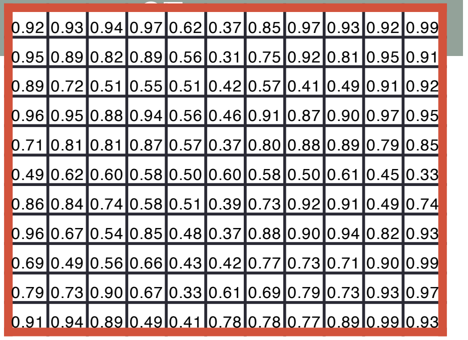







Did we augment the photometric quality really?

Today

- What is an image?
  - Types, color vs gray level
- How to measure the quality of an image?
  - Spatial and photometric resolution
  - Histogram and image contrast enhancement
- How to process by a linear filter
  - Examples: smoothing filters
  - Convolution / correlation
  - Mean and median filters
  - Linear filters with Gaussians

Image filtering



0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

- **Linear Filtering**: Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Soft blur, smoothing
  - Enhance an image (remove noise, denoise, etc)
  - Extract information (Sharpen or accentuate details (texture, edges, etc)
  - Detect patterns (feature detection and matching, template matching)

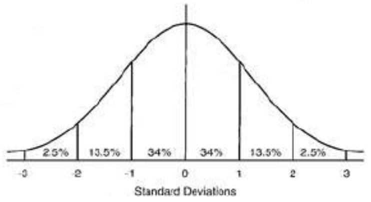
Adapted from Derek Hoiem

Common types of noise

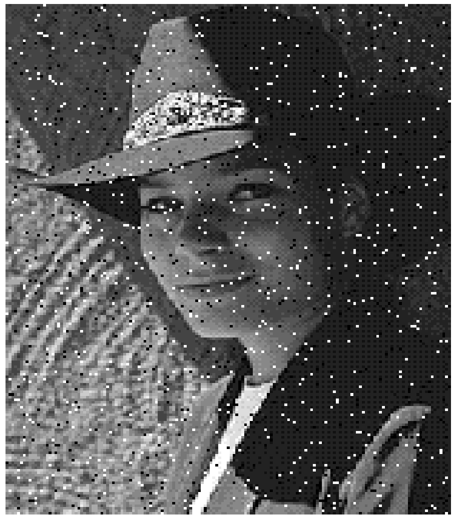
- **Salt and pepper noise**: random occurrences of black and white pixels

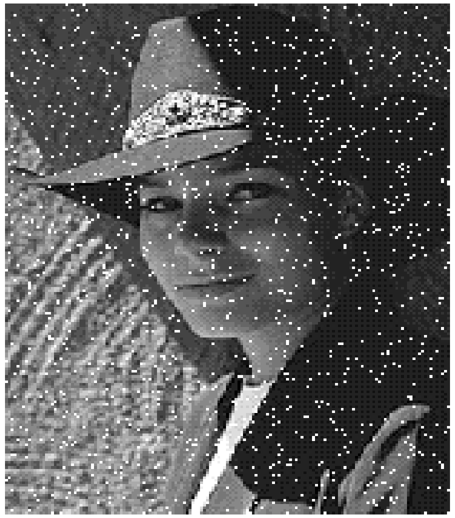
- **Impulse noise:** random occurrences of white pixels

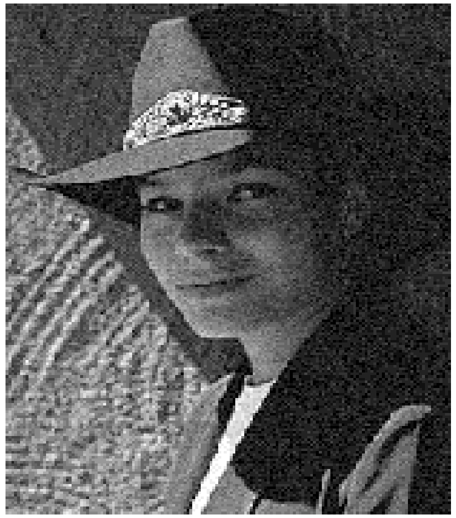
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



Original


Salt and pepper noise


Impulse noise


Gaussian noise

Source: S. Seitz

Lecture videos

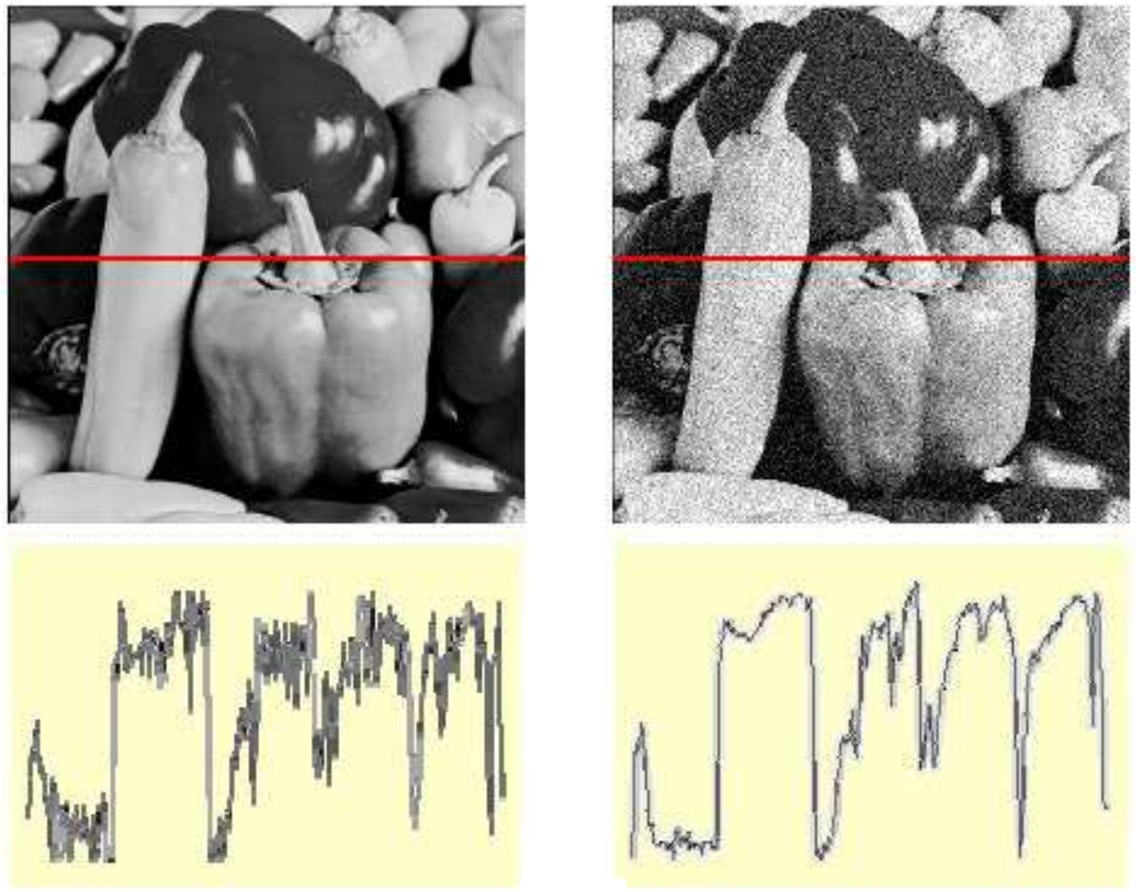
Lesson 2: 2A-L1 Images as Functions Part 1

- Until 9 - Compute Image Size
- 27 - Common Types of Noise
- 32 - Effect of Sigma on Gaussian Noise
- 36 - Adding Noise

From Introduction to Computer Vision:

https://www.udacity.com/course/introduction-to-computer-vision--ud810

Gaussian noise



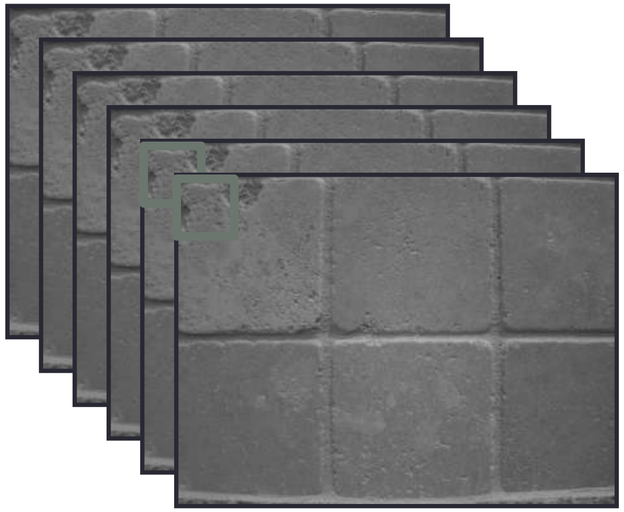
$$f(x,y) = \overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

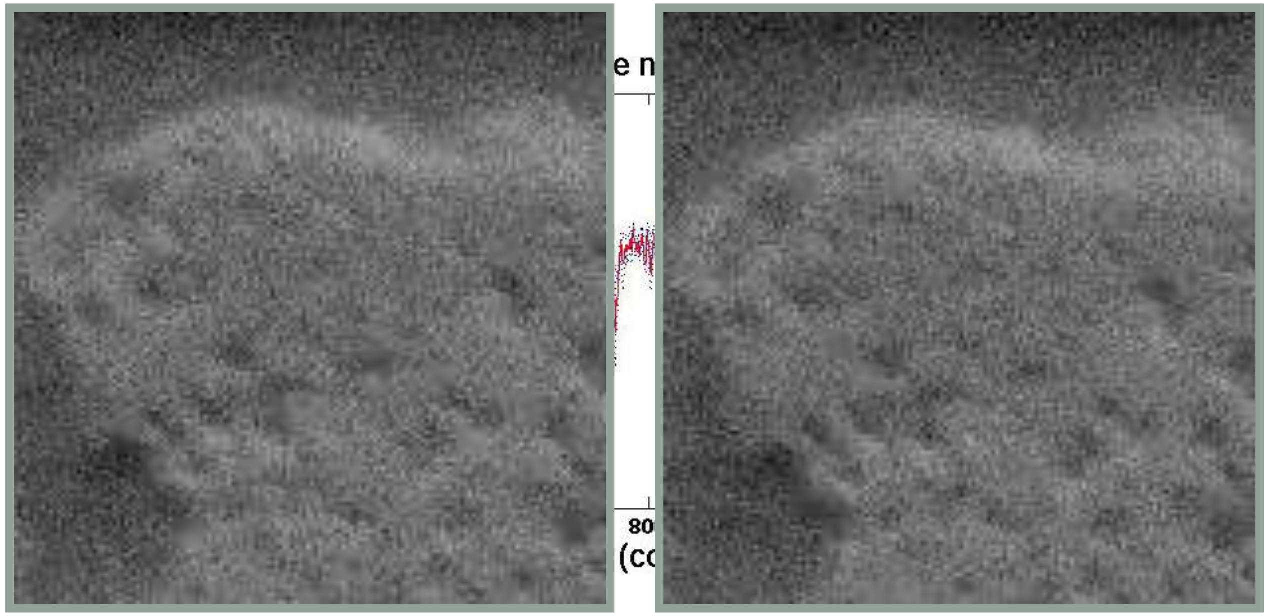
Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

What is the impact of sigma (var)?

Fig: M. Hebert

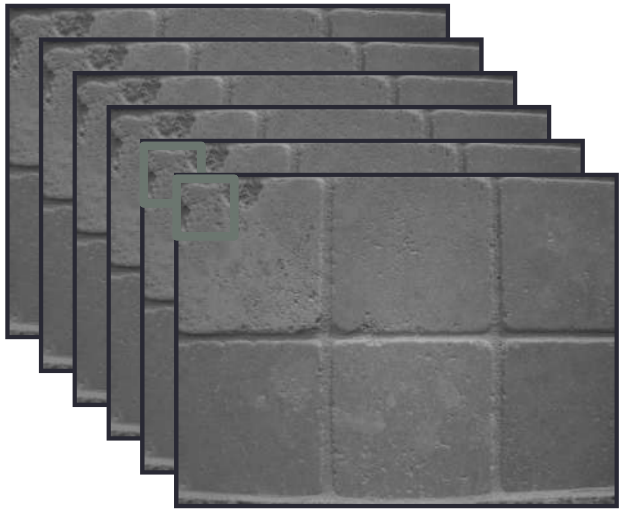
Motivation: noise reduction

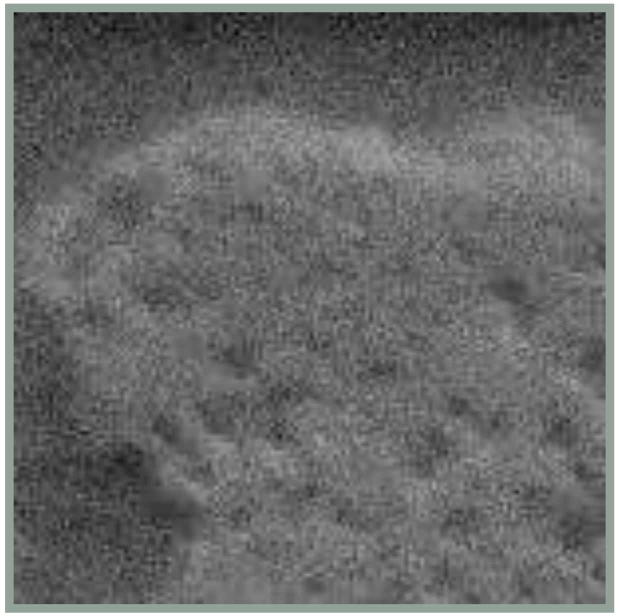




- Even multiple images of the **same static scene** will not be identical.

Motivation: noise reduction





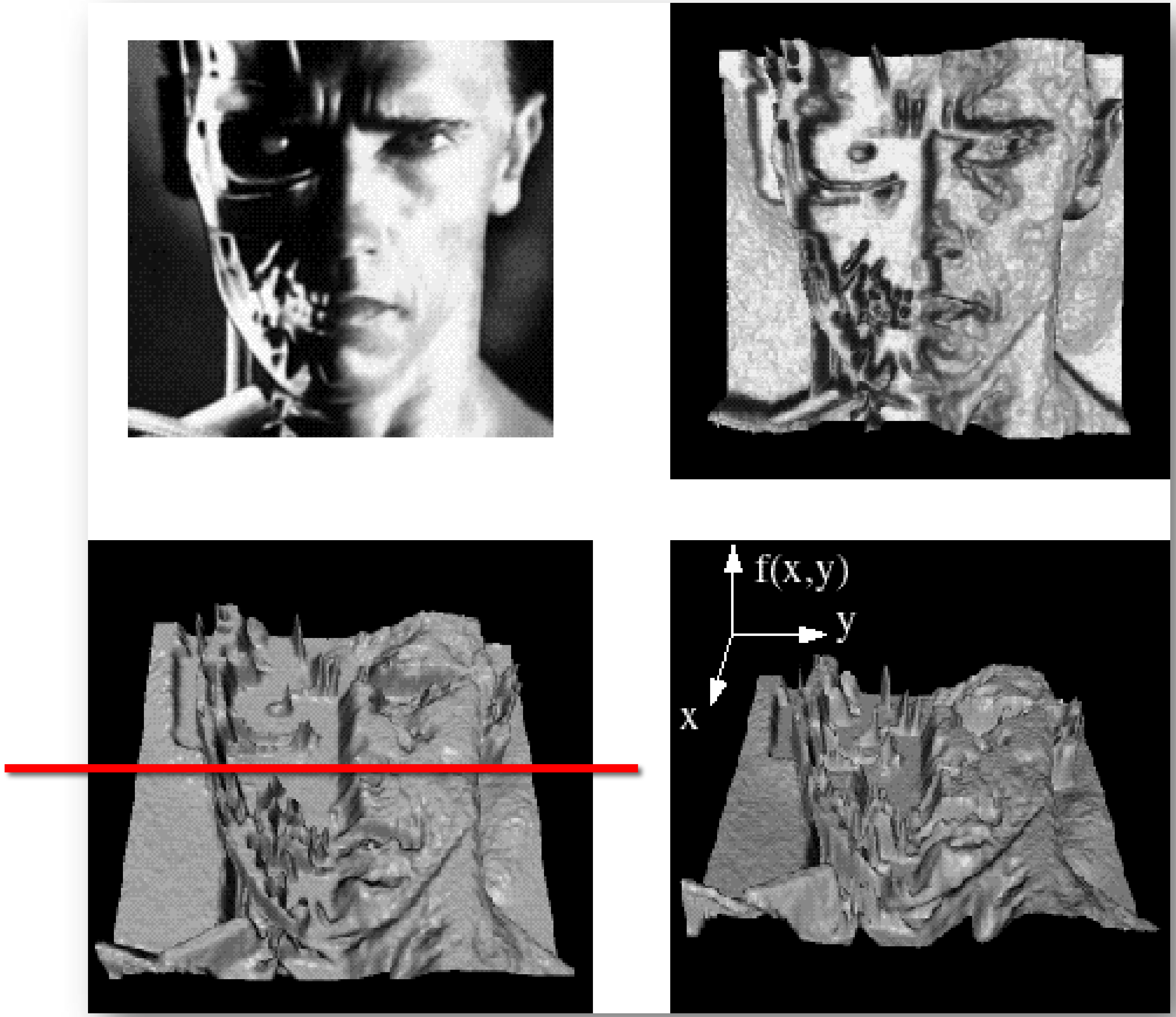
- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
  - Take the average of the grey values per pixel.

- **What if there's only one image?**

First attempt at a solution

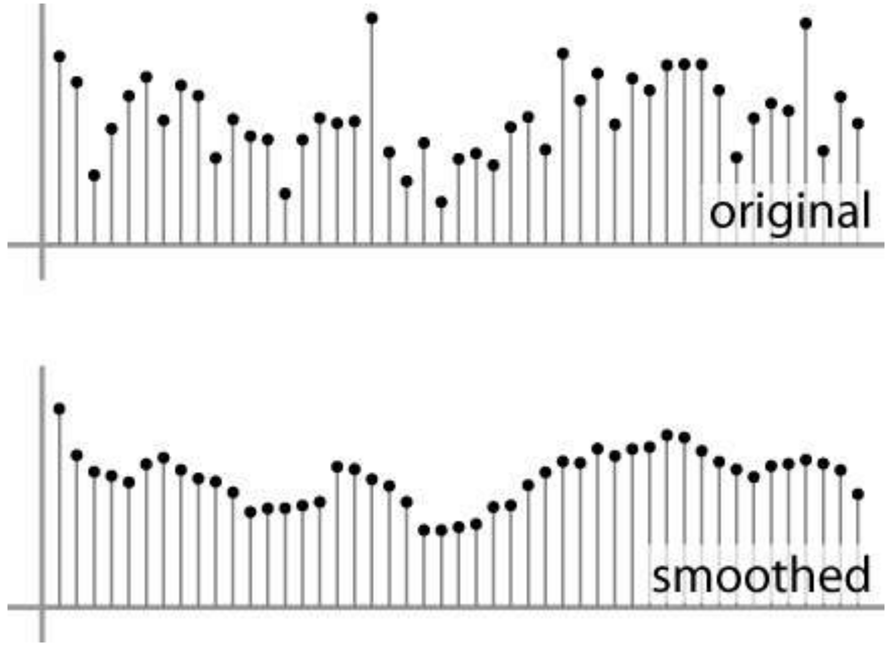
- Let’s replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Remember: an image is a matrix & topographic map



First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

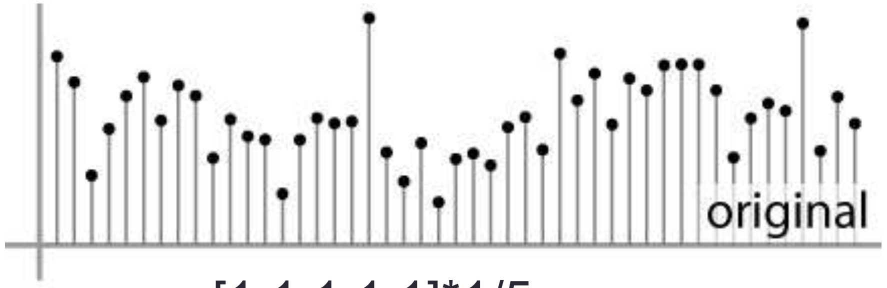


Source: S. Marschner

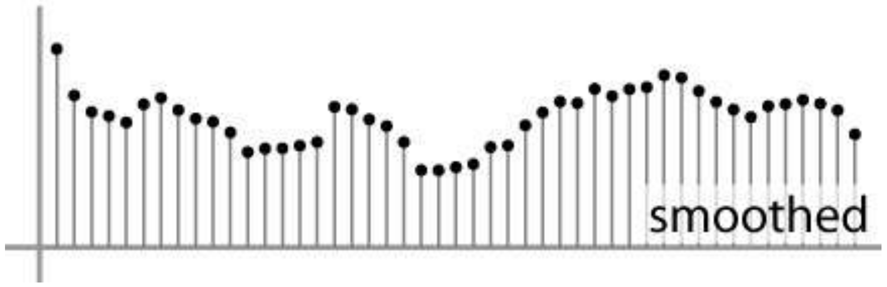
First attempt at a solution

- What is the average value?
- (F[i-2]+F[i-1]+F[i]+F[i+1]+ F[i+2])/5

- For each pixel i, multiply its neighbourhood by a mask:
  - [1,1,1,1,1]*1/5= [1/5,1/5,1/5,1/5,1/5,1/5]



[1,1,1,1,1]*1/5



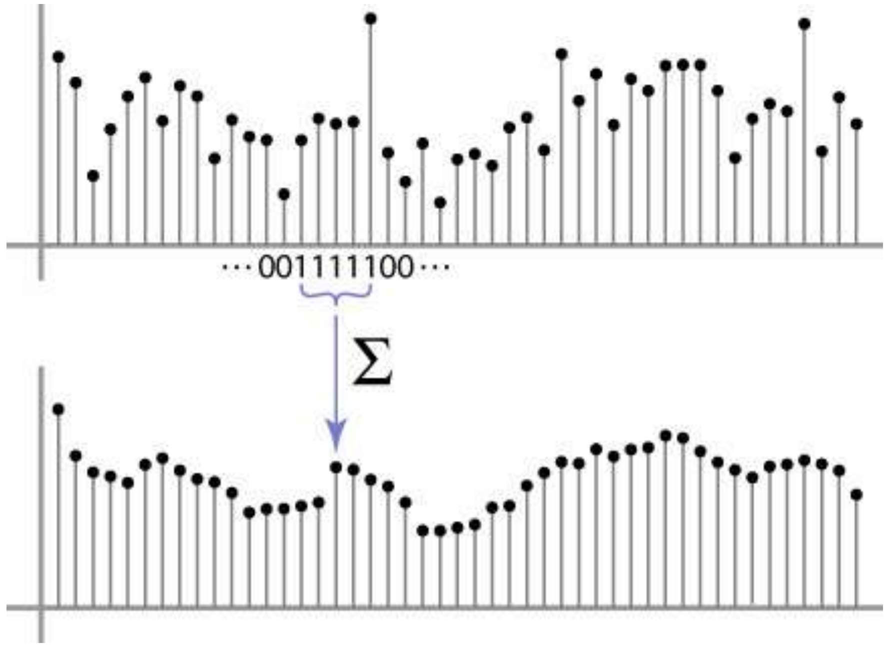
Definition: Convolution

$$(F*G)[i]=\sum_{m=-M}^{M} F[i+m]g[m]$$

Source: S. Marschner

Weighted Moving Average – Mean filter

- *Weights* [1, 1, 1, 1, 1] / 5
- Why are we dividing by 5?

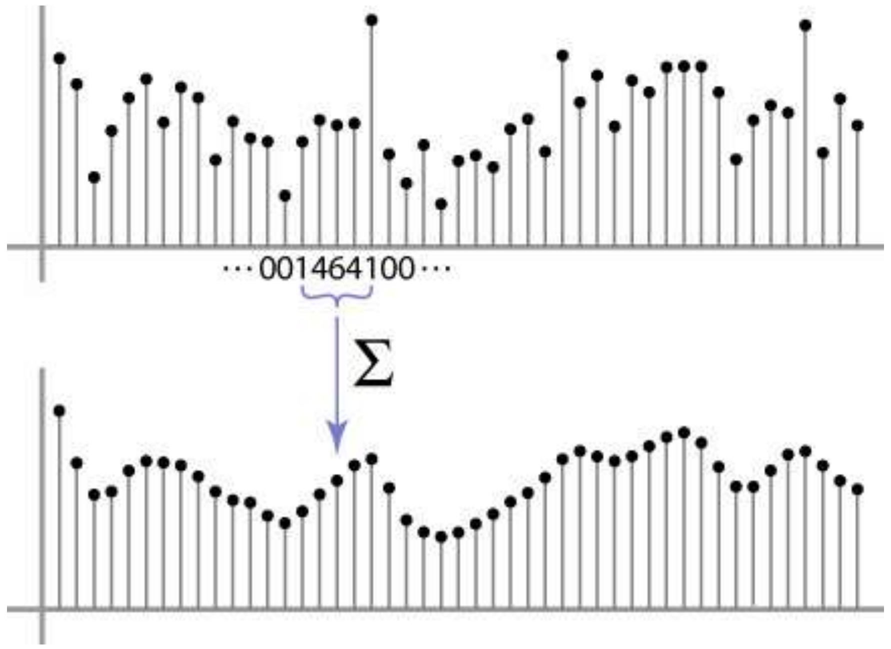


Can we add weights to our moving average? Why?

Source: S. Marschner

Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16

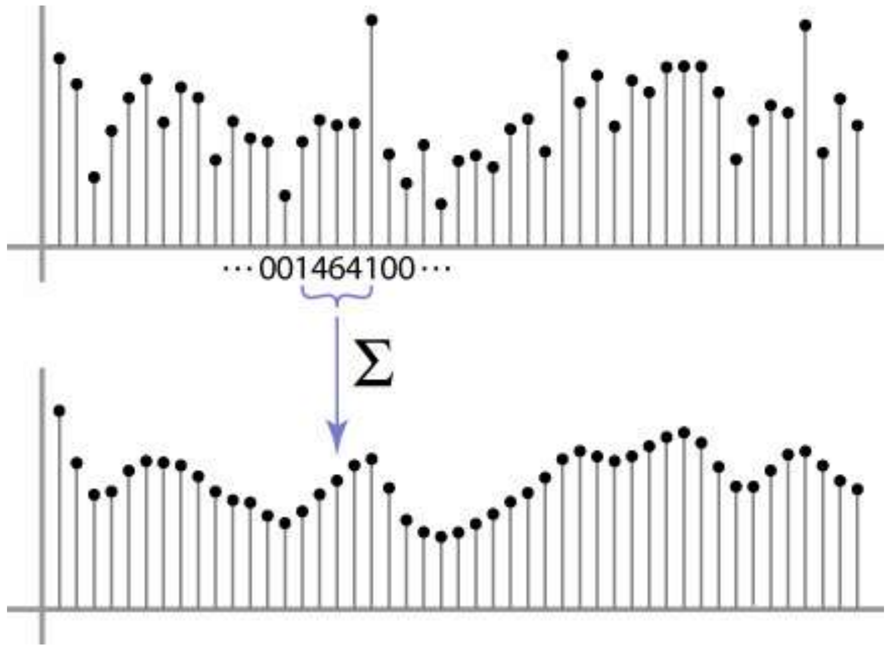


- What is the difference with the previous one?

Source: S. Marschner

Weighted Moving Average

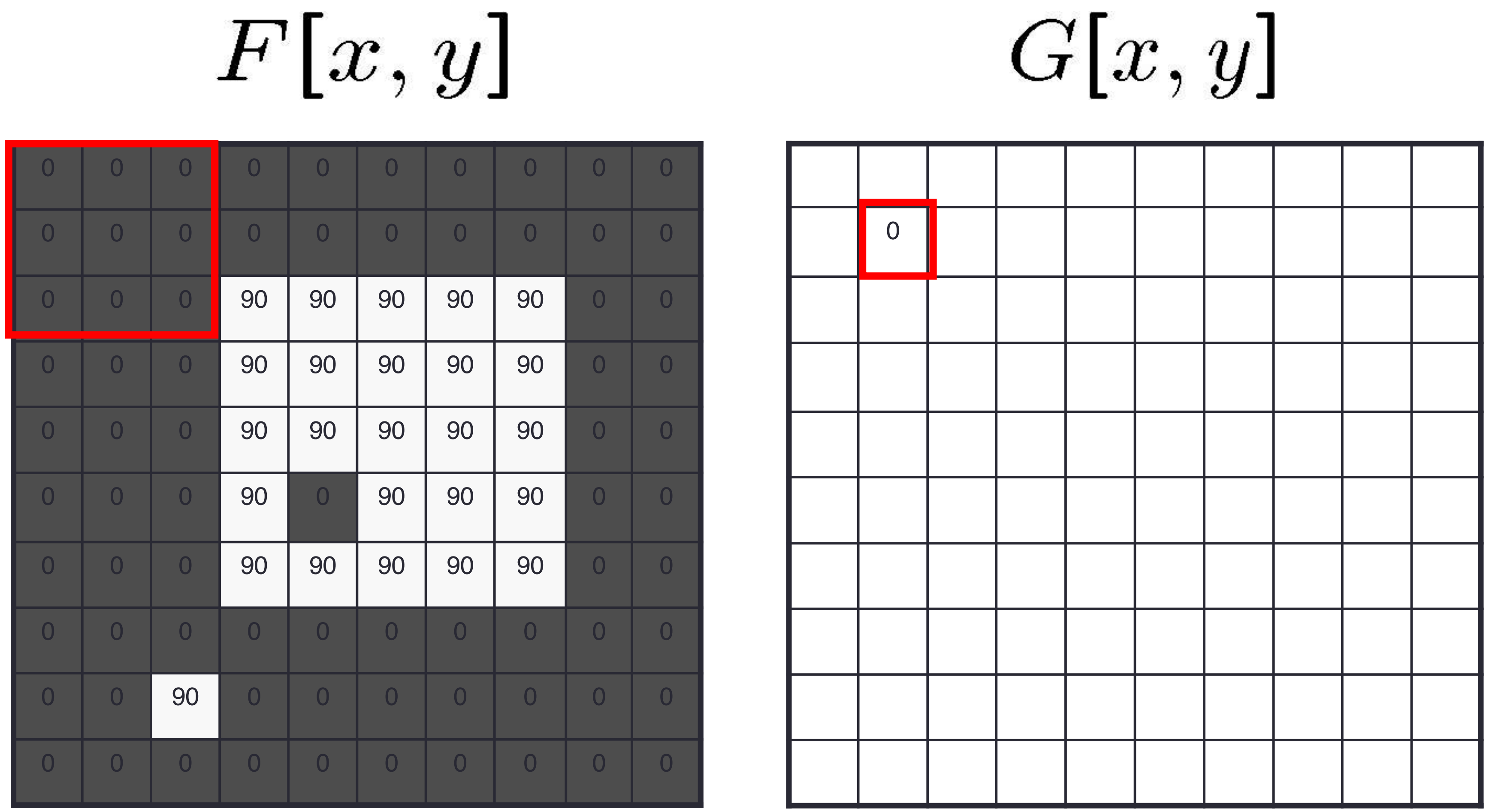
- Non-uniform weights [1, 4, 6, 4, 1] / 16



- What is the difference with the previous one?

Source: S. Marschner

Moving Average in 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

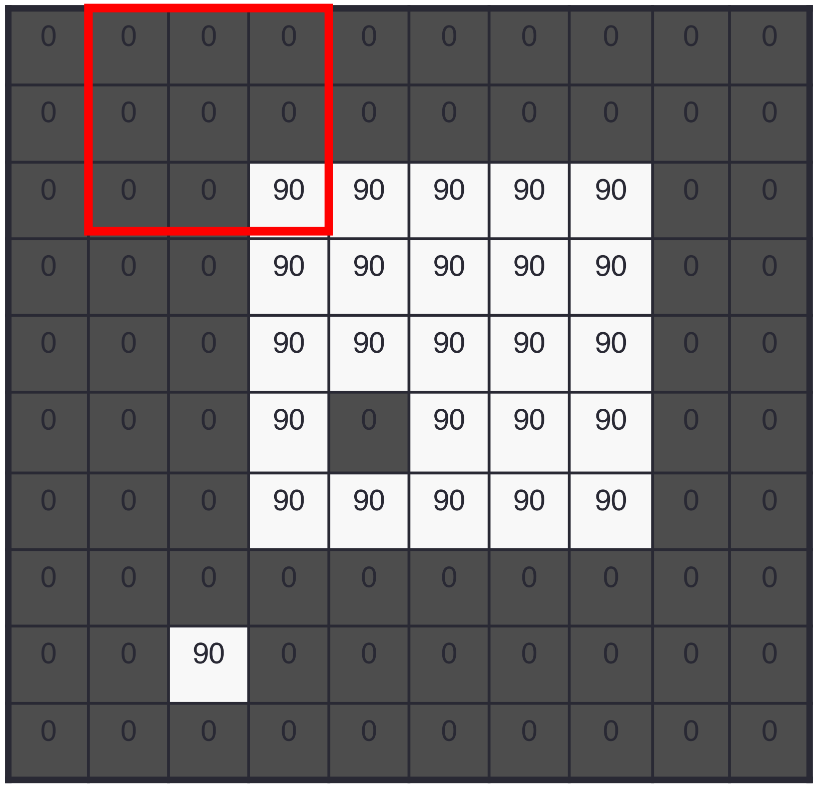
$G[x,y]$

	0								

Source: S. Seitz

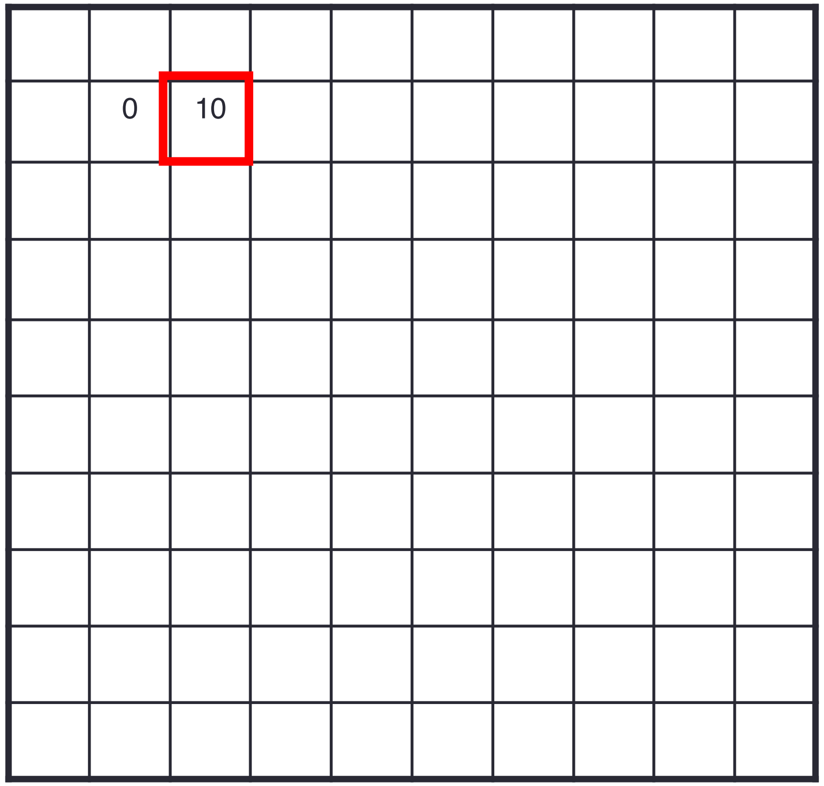
Moving Average in 2D

$F[x, y]$



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

$G[x, y]$

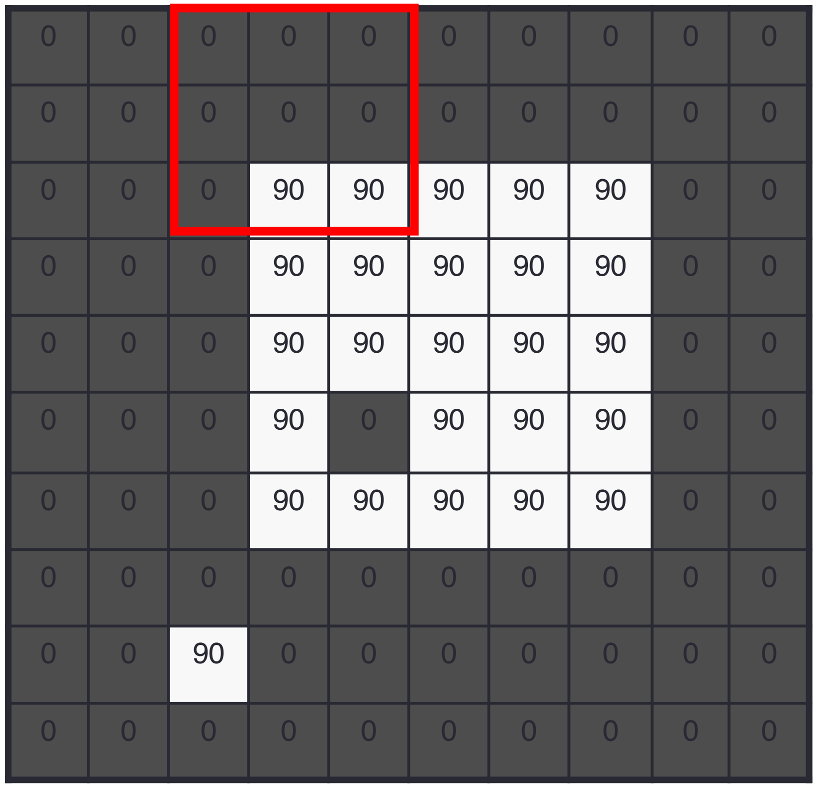


	0	10							

Source: S. Seitz

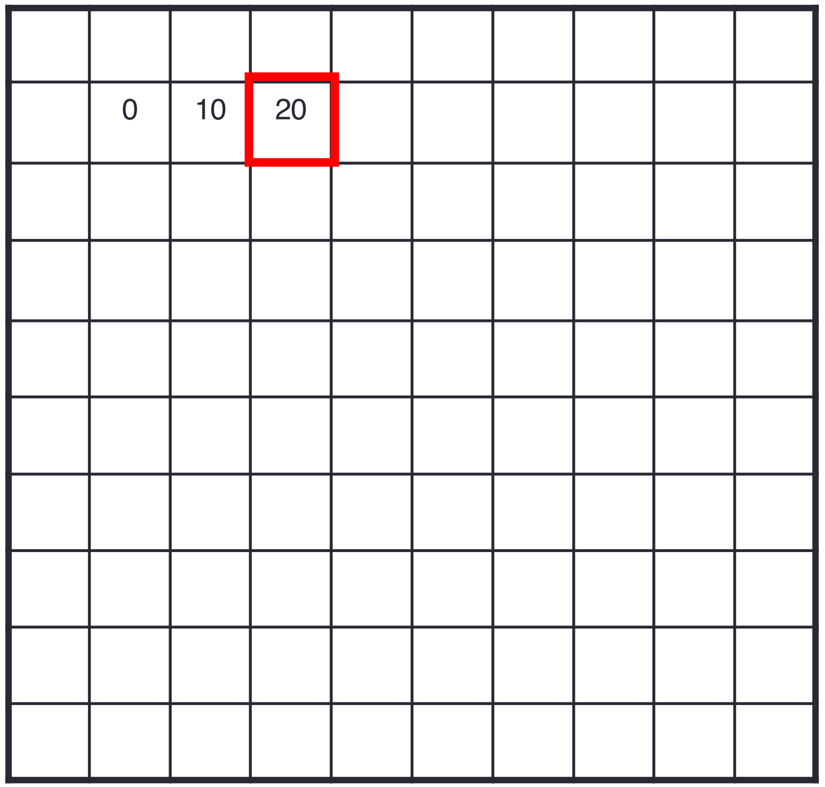
Moving Average in 2D

$F[x, y]$



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

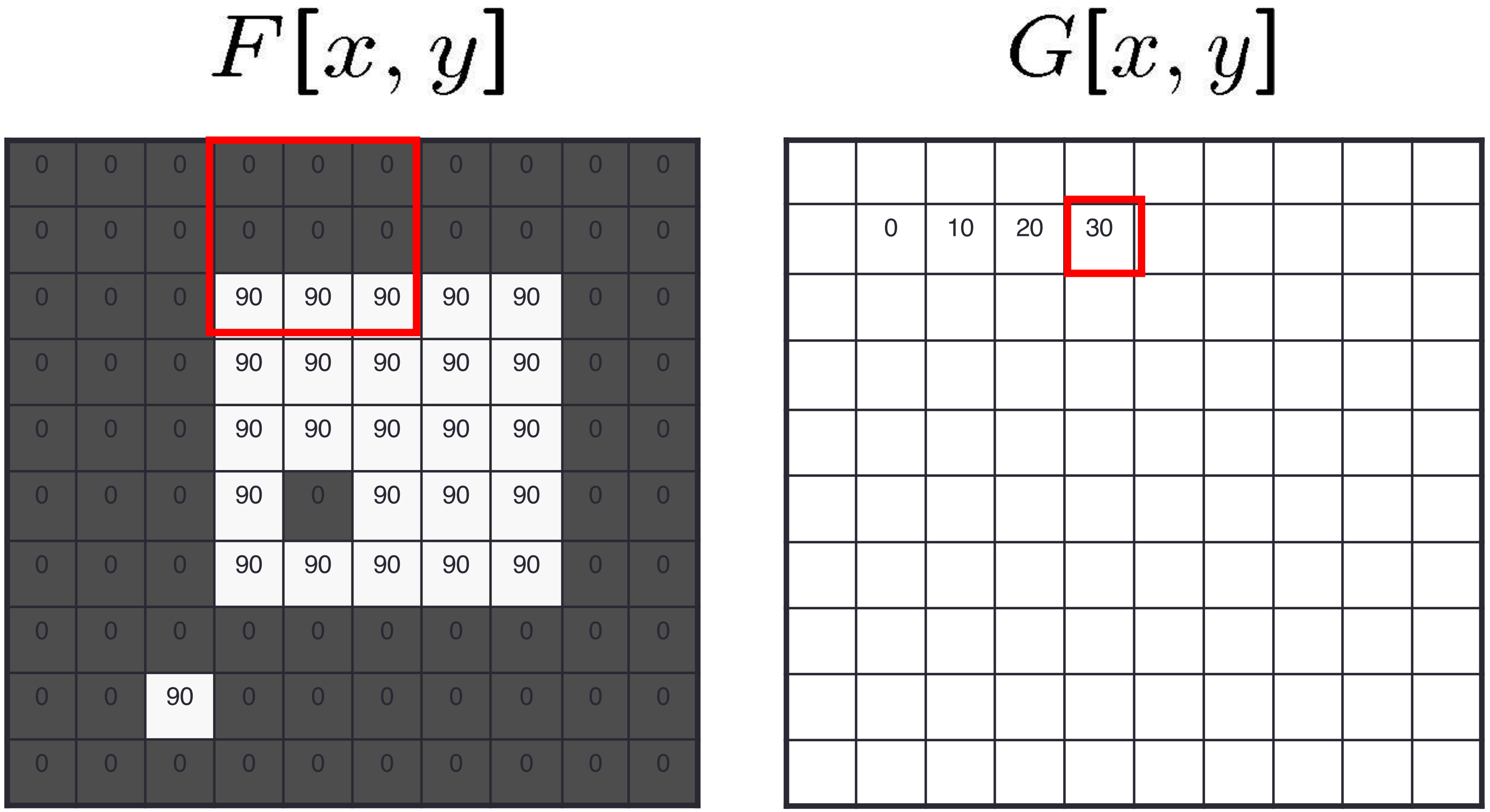
$G[x, y]$



	0	10	20						

Source: S. Seitz

Moving Average In 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

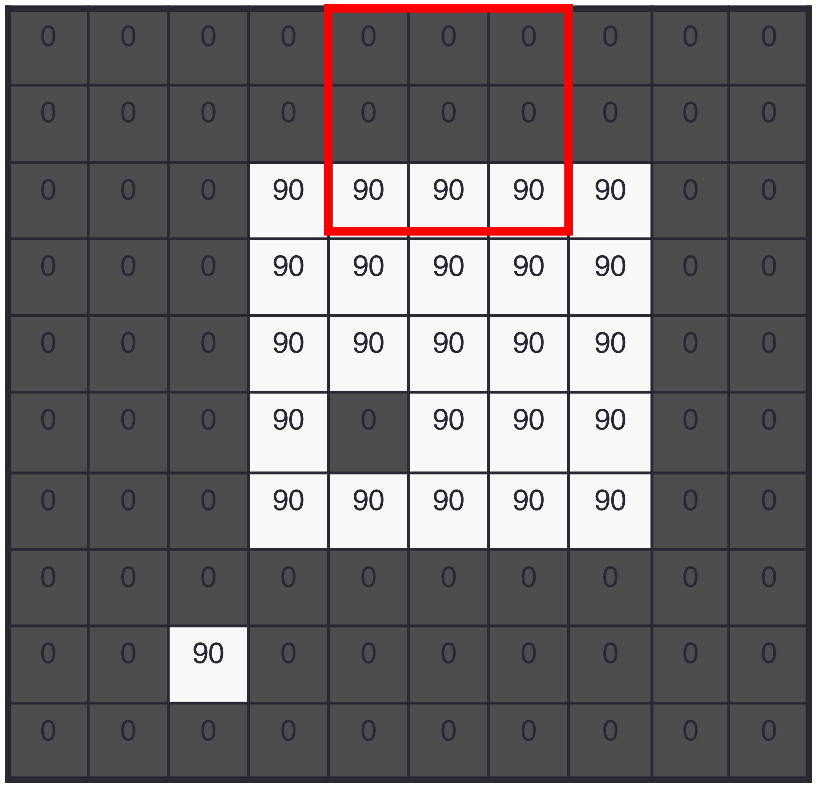
$G[x, y]$

	0	10	20	30					

Source: S. Seitz

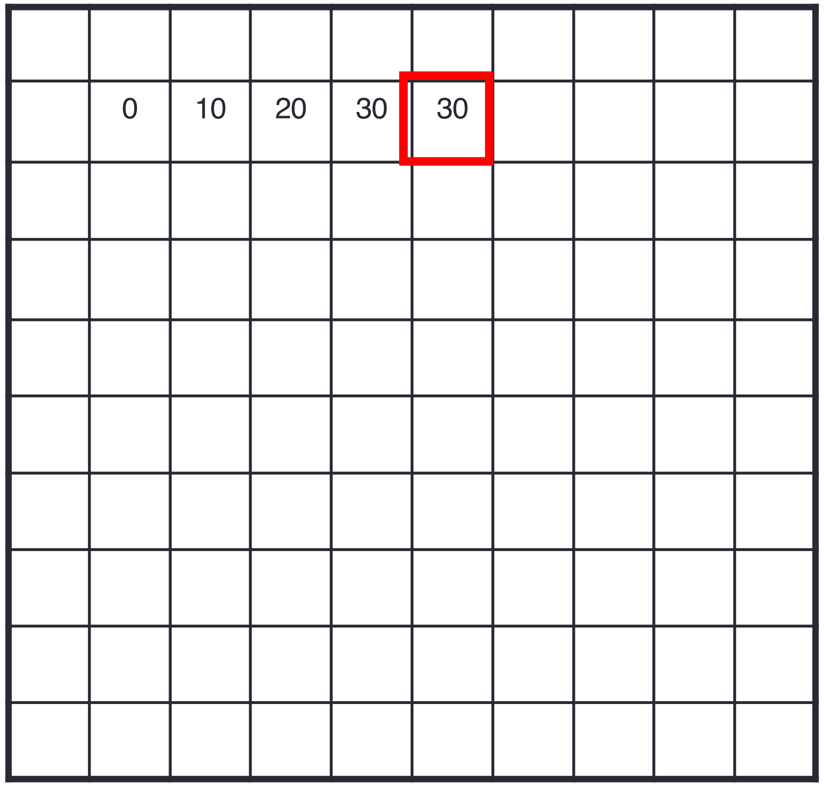
Moving Average in 2D

$$F[x,y]$$



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

$$G[x,y]$$

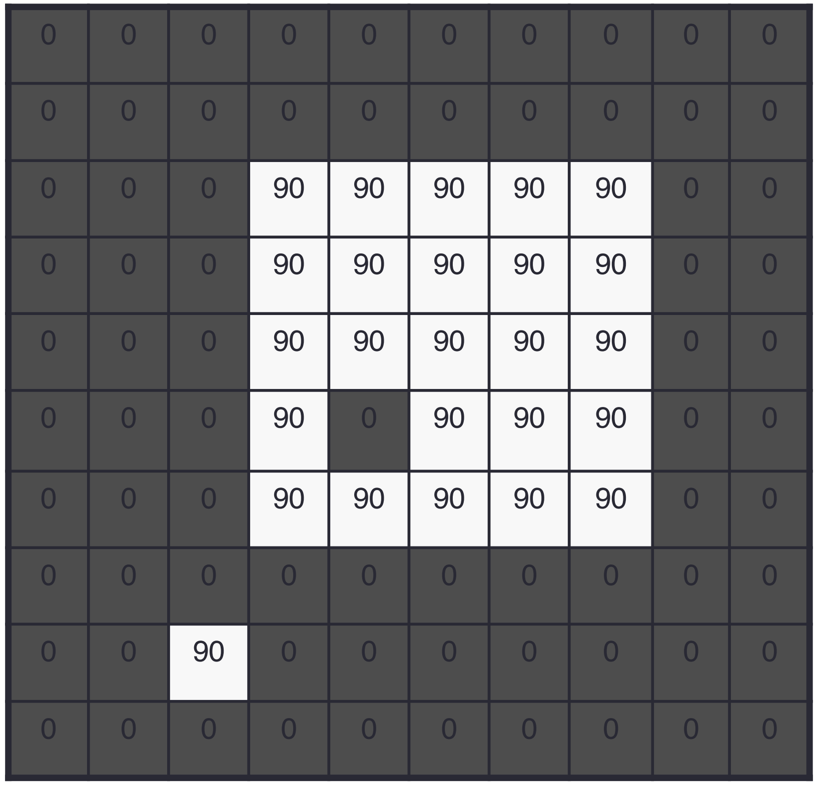


	0	10	20	30	30				

Source: S. Seitz

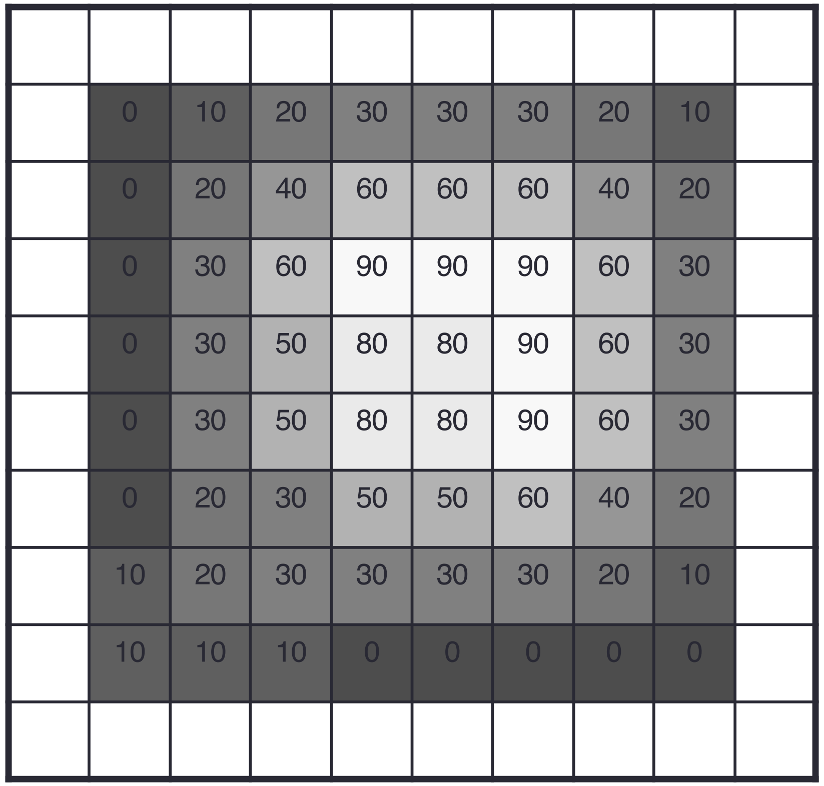
Moving Average in 2D

$F[x, y]$



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

Linear filter

$F[x,y]$ $\otimes$ $H[u,v]$

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

a	b	c
d	e	f
g	h	i

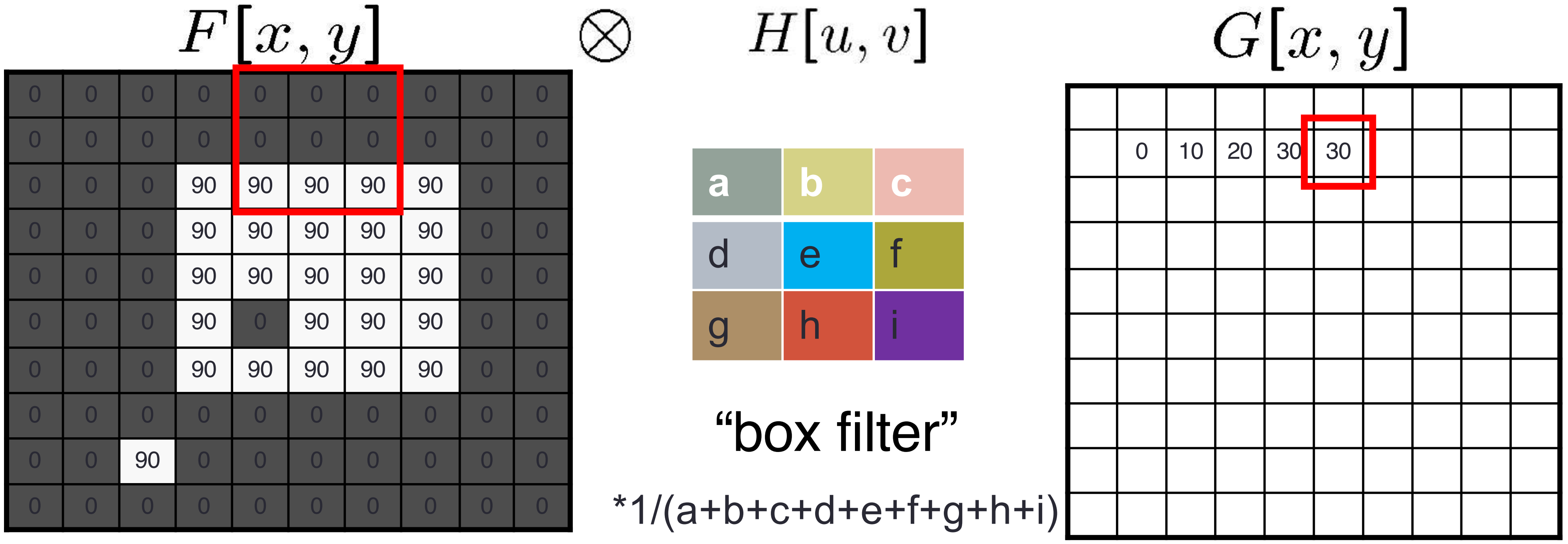
“box filter”

$$G = H \otimes F$$

$$df[i,j] = a * f[i-1 j-1] + b * f[i-1,j] + c * f[i-1,j+1] +$$
$$d * f[i,j-1] + e * f[i,j] + f * f[i,j+1] +$$
$$g * f[i+1,j-1] + h * f[i+1,j] + i * f[i+1,j+1]$$

Linear filter

- Normalization: divide the mask by (a+b+c+d+e+f+g+h+i)



$F[x,y]$ $\otimes$ $H[u,v]$ $G[x,y]$

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

a	b	c
d	e	f
g	h	i

"box filter"

*1/(a+b+c+d+e+f+g+h+i)

	0	10	20	30	30				

$$G = H \otimes F$$

Convolutional filtering

Say the averaging window size is (2k+1) x (2k+1):

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\textit{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\textit{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Convolutional filtering

Now generalize to allow different weights depending on neighboring pixel’s relative position:

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

Convolutional filtering

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

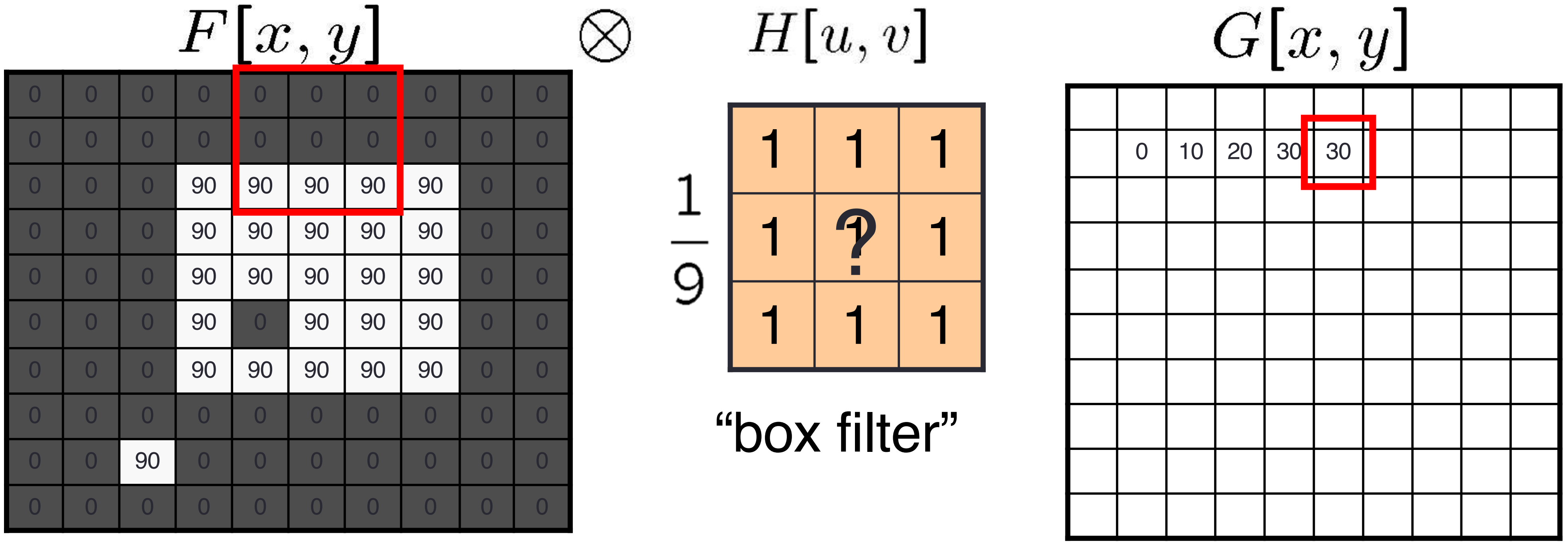
This is called convolution, denoted as: $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter “kernel” or “mask” *H*[*u,v*] is the prescription for the weights in the linear combination.

Mean filter

- What values do belong in the kernel *H* for the moving average example?



$F[x,y]$ $\otimes$ $H[u,v]$ $G[x,y]$

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

$\frac{1}{9}$

1	1	1
1	?	1
1	1	1

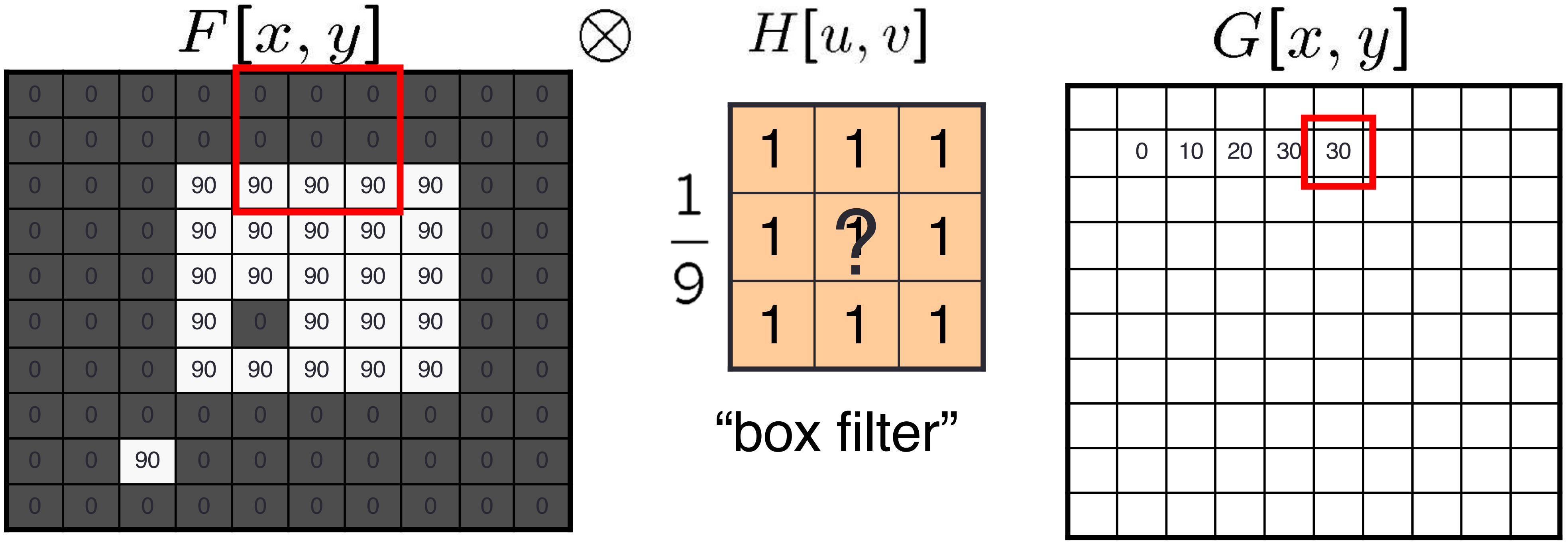
“box filter”

	0	10	20	30	30				

$$G = H \otimes F$$

Mean filter

- Normalization: why do we need to divide the mask by 9?



$F[x,y]$ $\otimes$ $H[u,v]$ $G[x,y]$

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

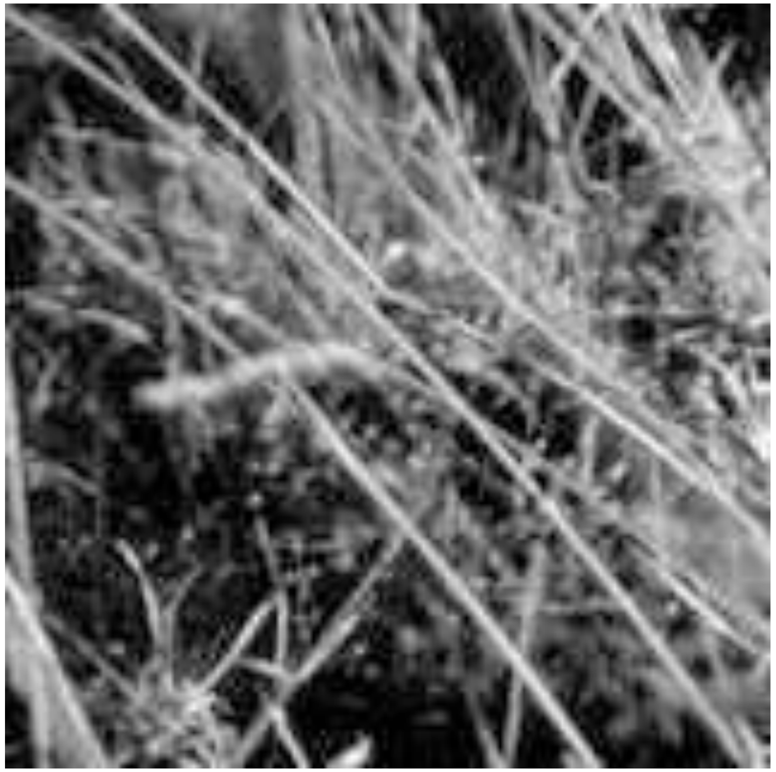
$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & ? & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

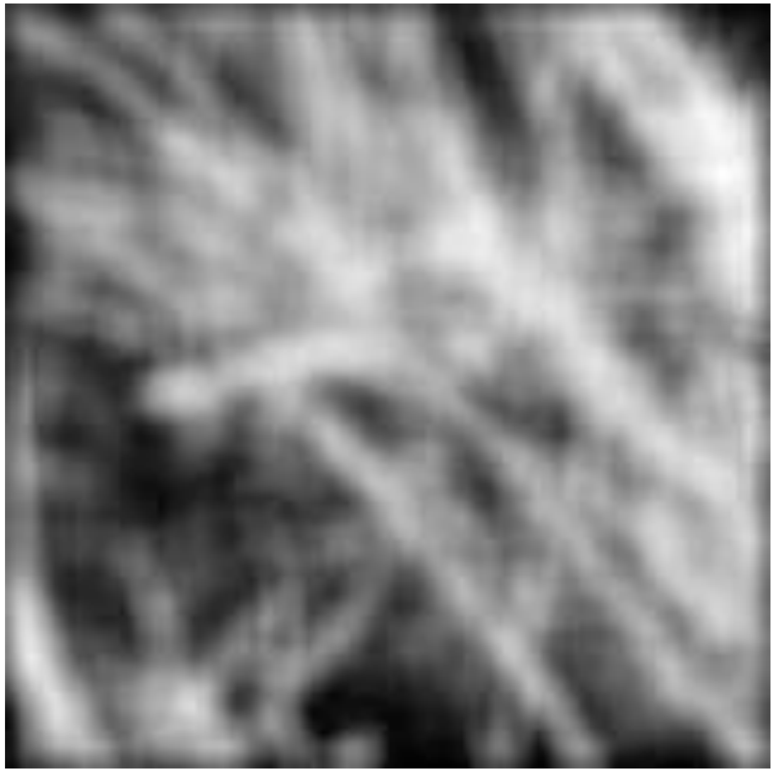
"box filter"

	0	10	20	30	30				

$$G = H \otimes F$$

Smoothing by averaging

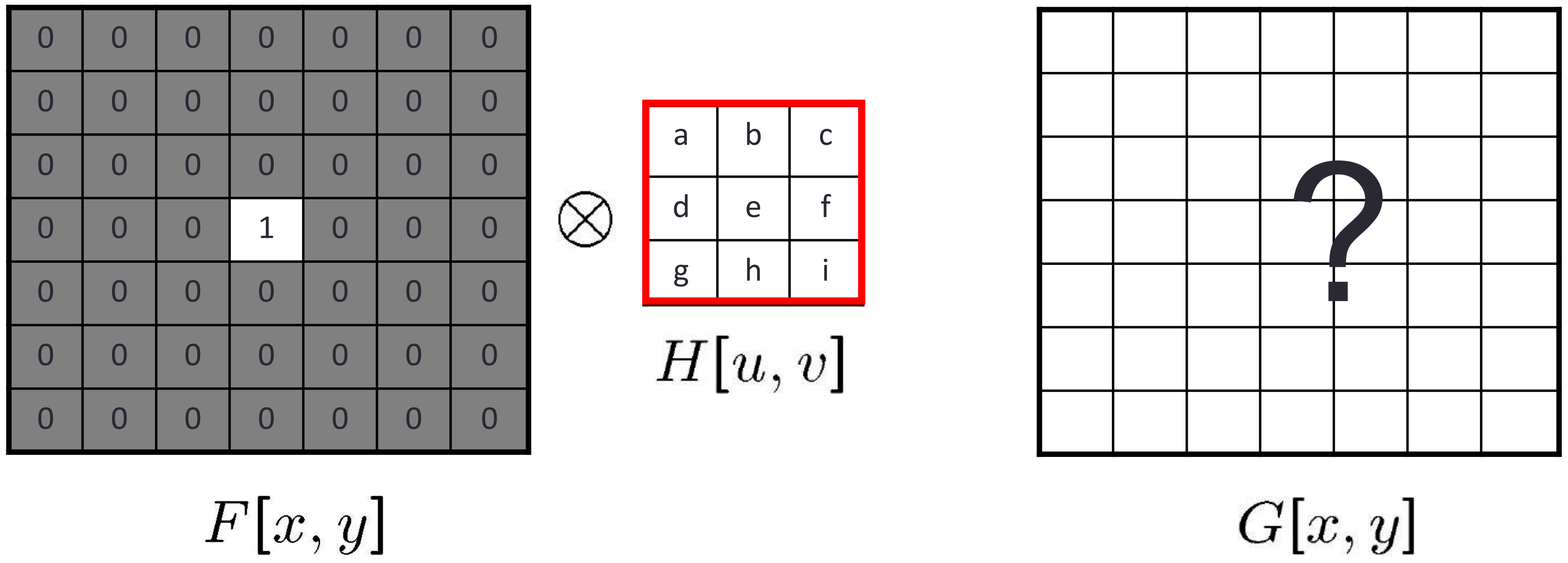

original


filtered

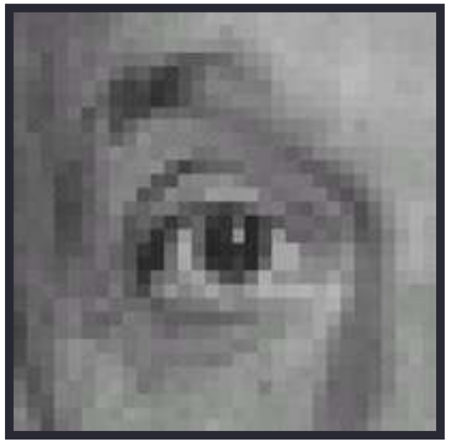
What is the effect if the filter size was 5 x 5 instead of 3 x 3?

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



Practice with linear filters

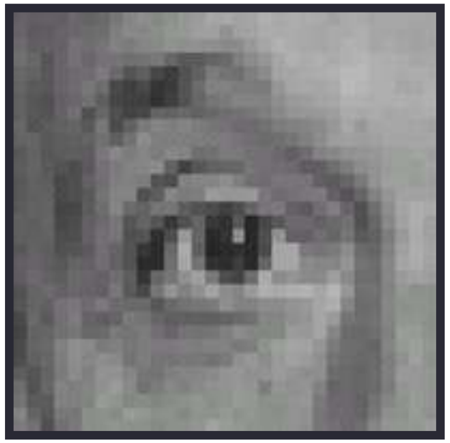

Original

0	0	0
0	1	0
0	0	0

?

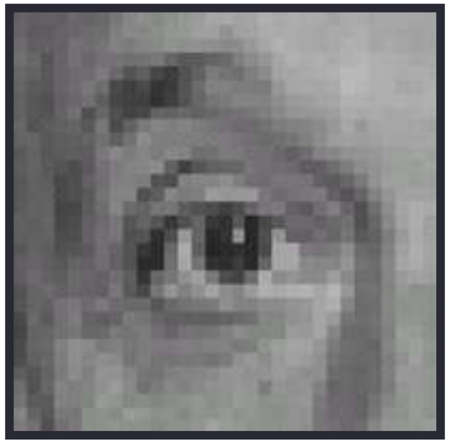
Source: D. Lowe

Practice with linear filters



Original

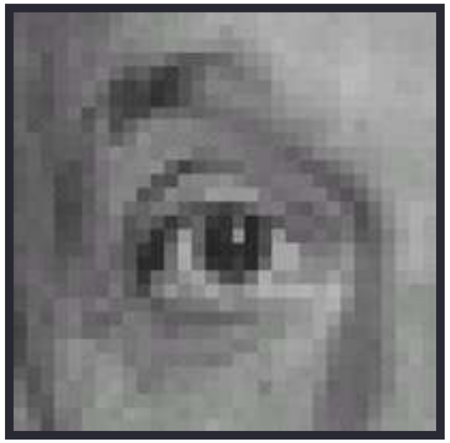
$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



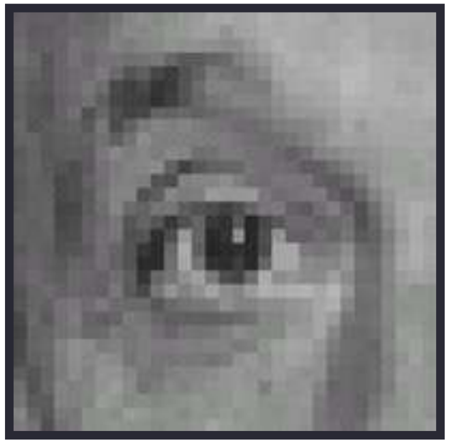
Original

0	0	0
0	0	1
0	0	0

?

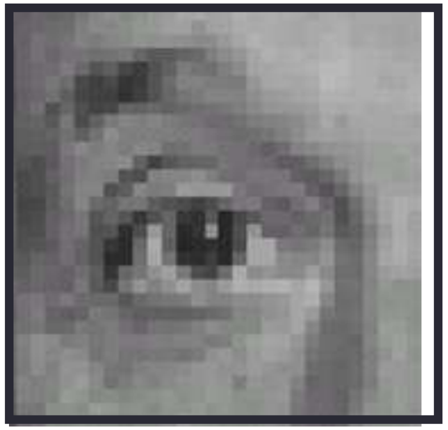
Source: D. Lowe

Practice with linear filters



Original

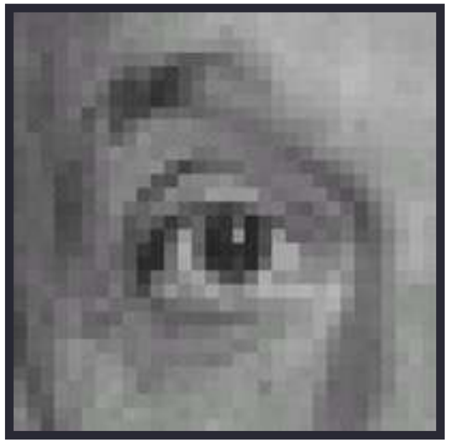
0	0	0
0	0	1
0	0	0



Shifted left by 1 pixel with correlation

Source: D. Lowe

Practice with linear filters



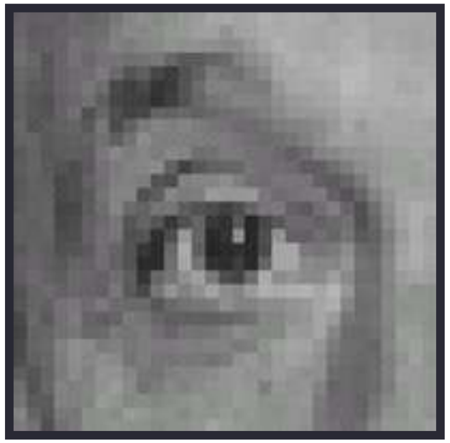
Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

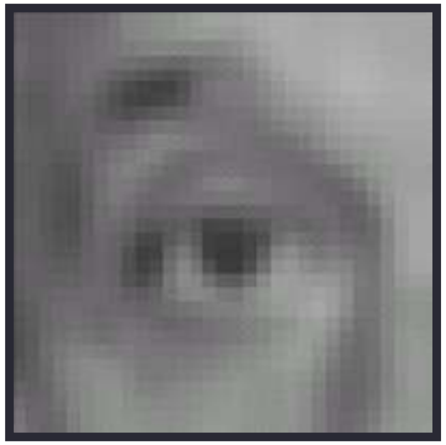
?

Source: D. Lowe

Practice with linear filters

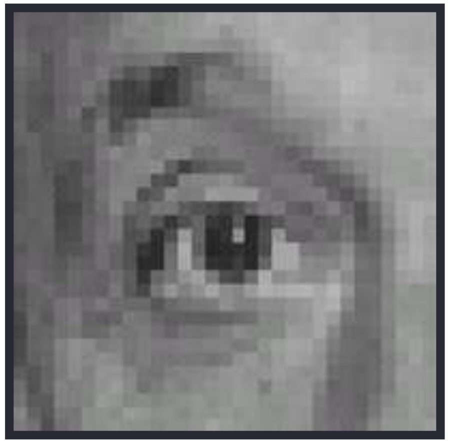

Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$


Blur (with a box filter)

Source: D. Lowe

Practice with linear filters



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

Original

Source: D. Lowe

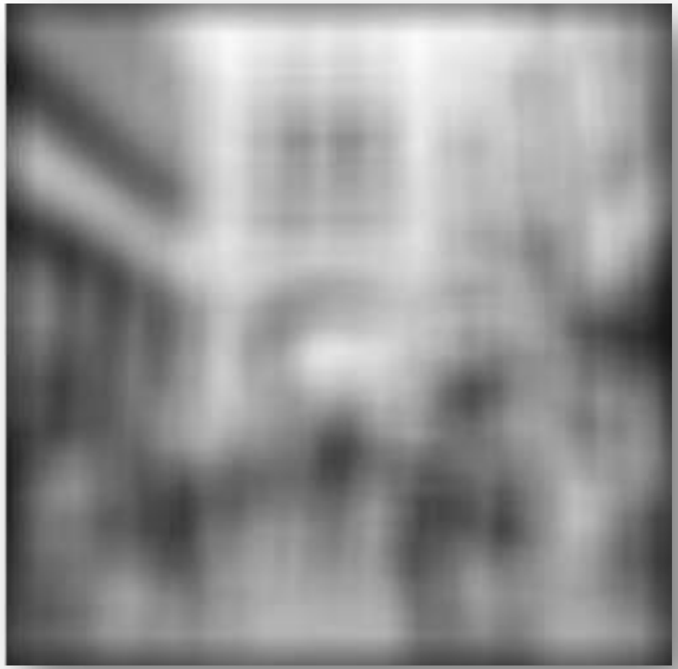
Properties of convolution

- **Shift invariant:**
  - Operator behaves the same everywhere, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.
- **Superposition:**
  - h * (f1 + f2) = (h * f1) + (h * f2)

Smoothing with a rectangular filter

$\otimes$

=



I[x,y]

h[i,j]

f[x,y]

$$\text{Mask}=1/25^* \begin{matrix} 1,1,1,1,1 \\ 1,1,1,1,1 \\ 1,1,1,1,1 \\ 1,1,1,1,1 \\ 1,1,1,1,1 \end{matrix}$$

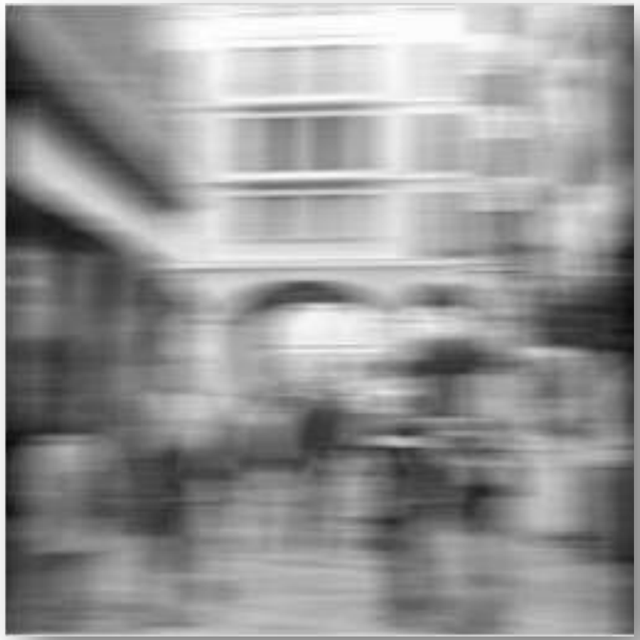
Smoothing with a rectangular filter

I[x,y]

$\otimes$

h[i,j]

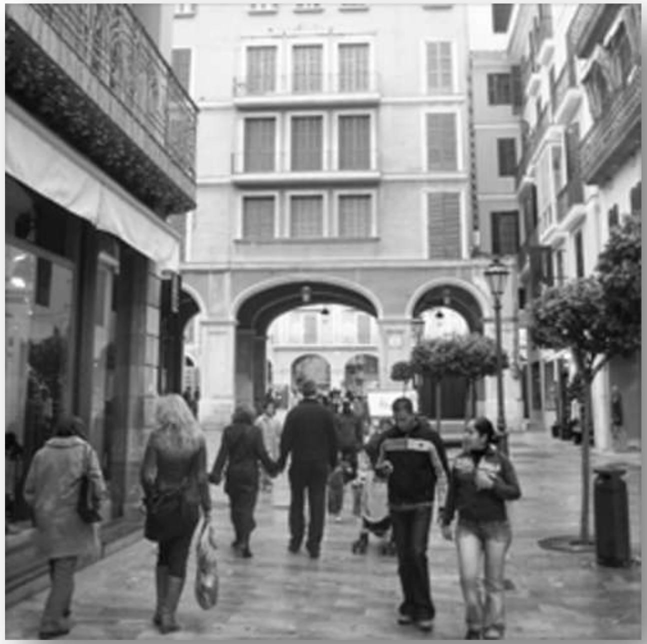
=



f[x,y]

Mask= 1/5*[1,1,1,1,1]

Smoothing with a rectangular filter

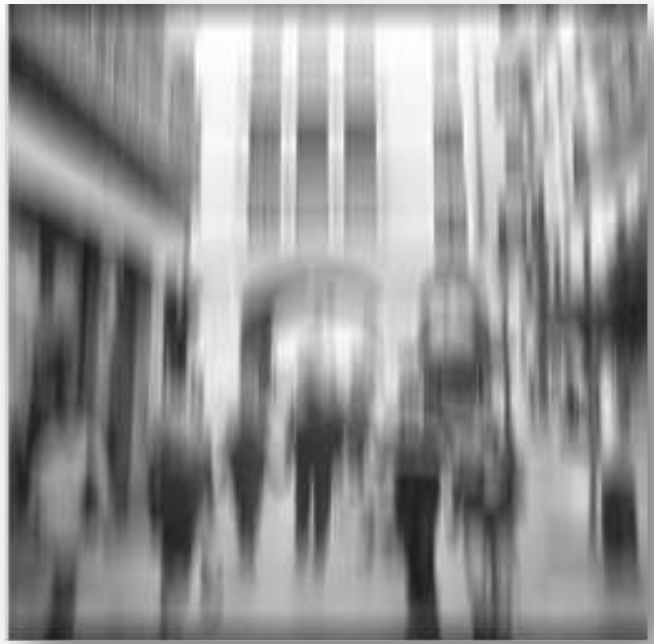


I[x,y]

$\otimes$

h[i,j]

=



f[x,y]

Mask=1/5*

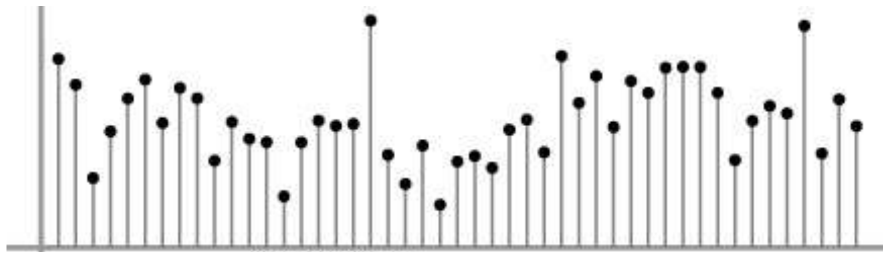
1,
1,
1,
1,
1

Today

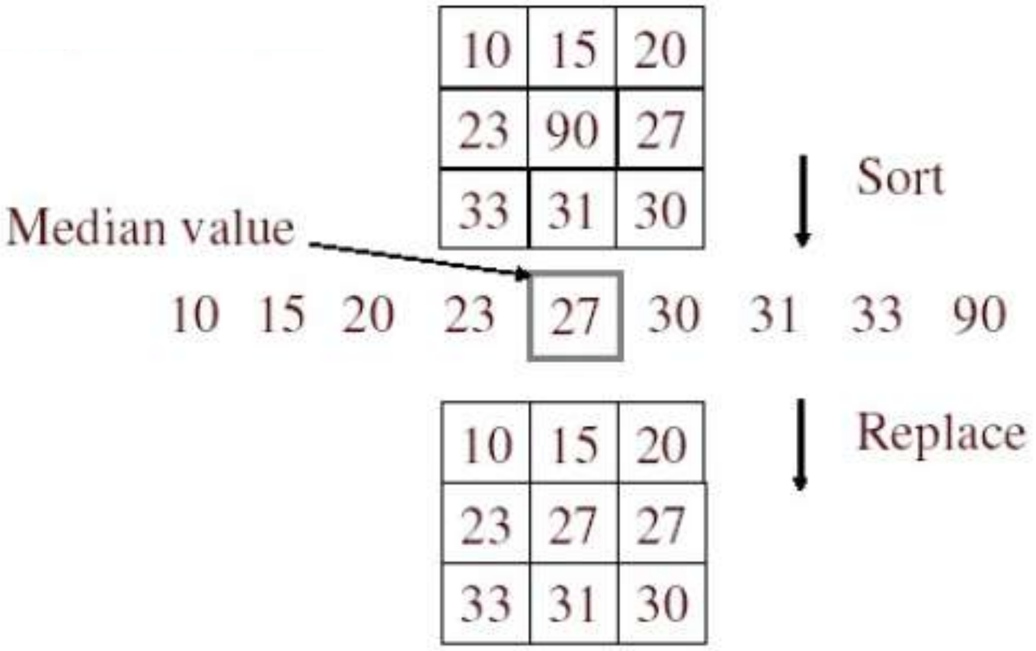
- What is an image?
  - Types, color vs gray level
- How to measure the quality of an image?
  - Spatial and photometric resolution
  - Histogram and image contrast enhancement
- How to process by a linear filter
  - Examples: smoothing filters
  - Convolution / correlation
  - Mean and median filters
  - Linear filters with Gaussians

Median filter

What is the behavior of the mean filter in the impulse noise pixels?



- No new pixel values introduced

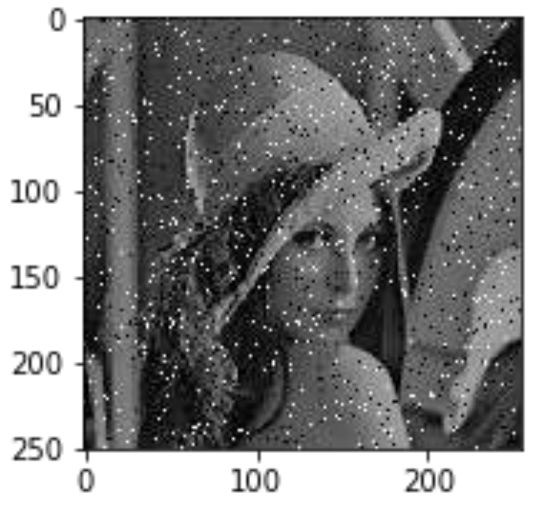


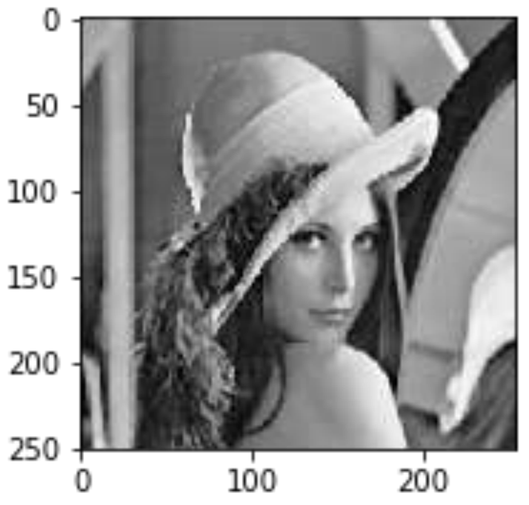
- Removes spikes: good for impulse, salt & pepper noise

•Non-linear filter (it can be proved)

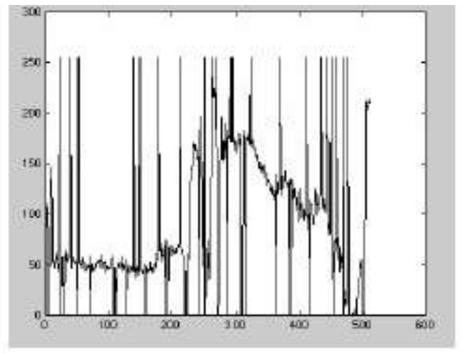
Median filter

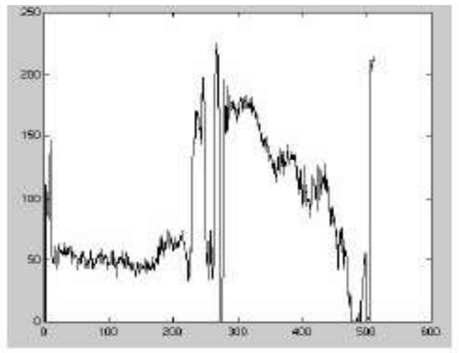
Salt and pepper noise





Median filtered



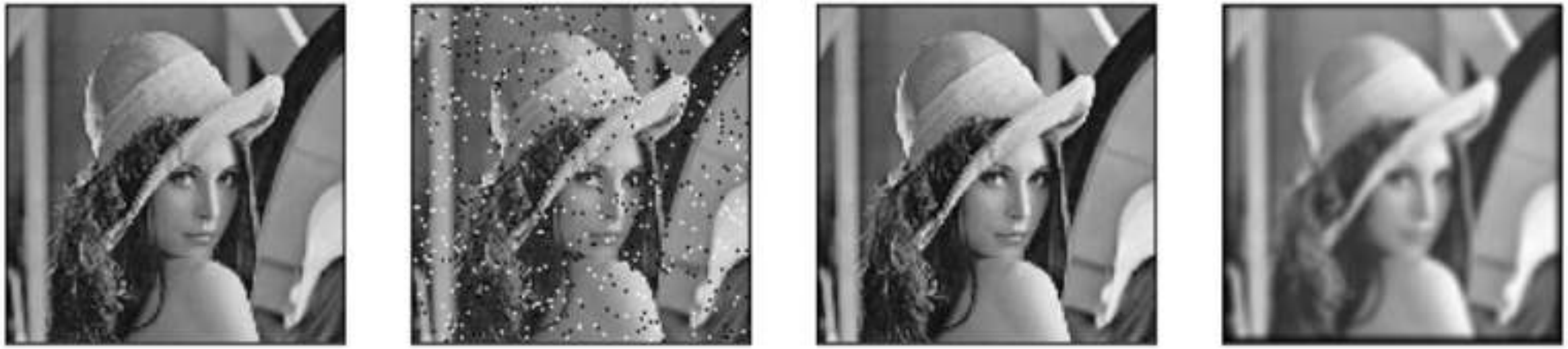


Plots of a row of the image

Source: M. Hebert

Median filter vs Mean filter

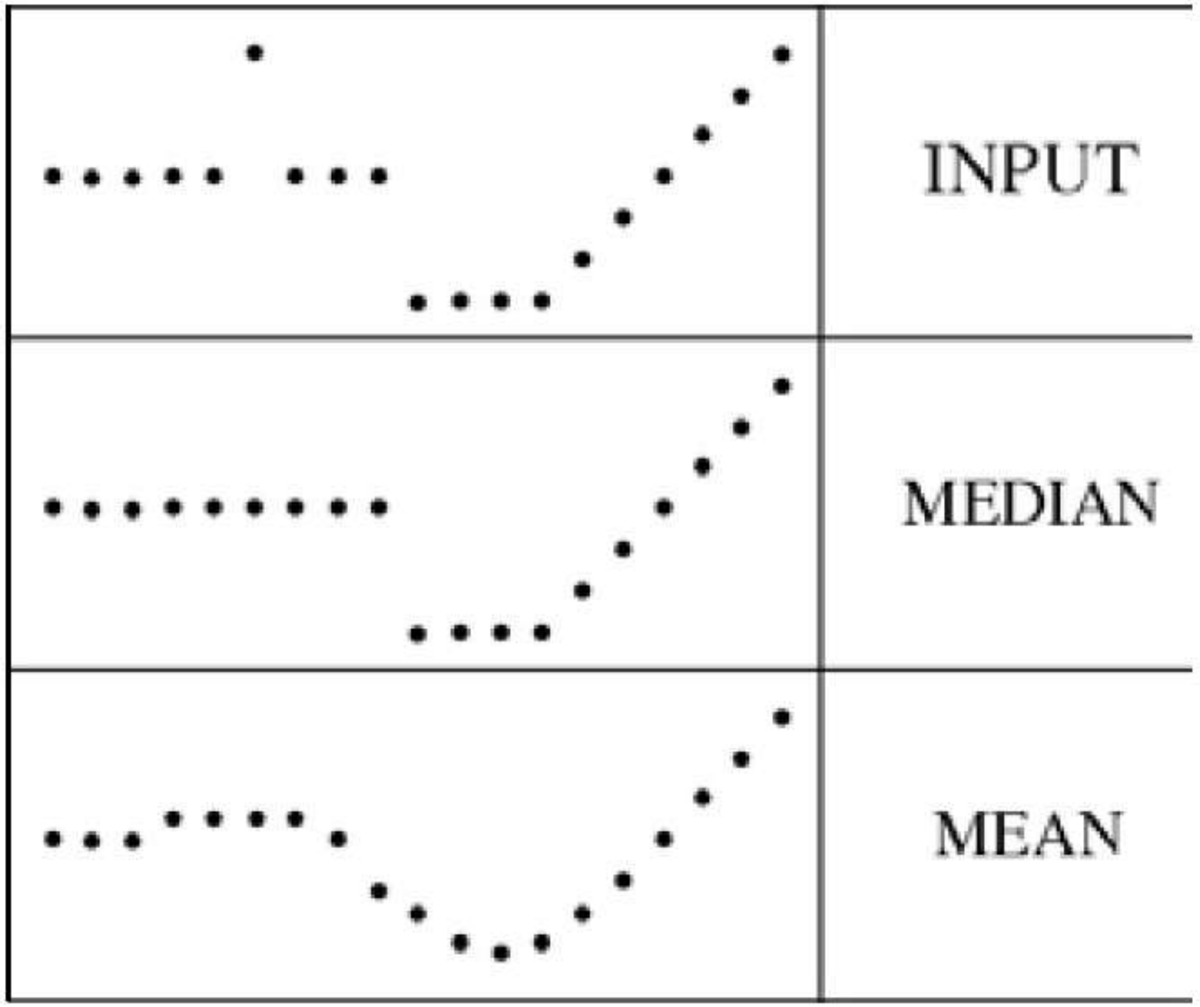
Introduce the captions of the images



Source: M. Hebert

Median filter

- Median filter is edge preserving



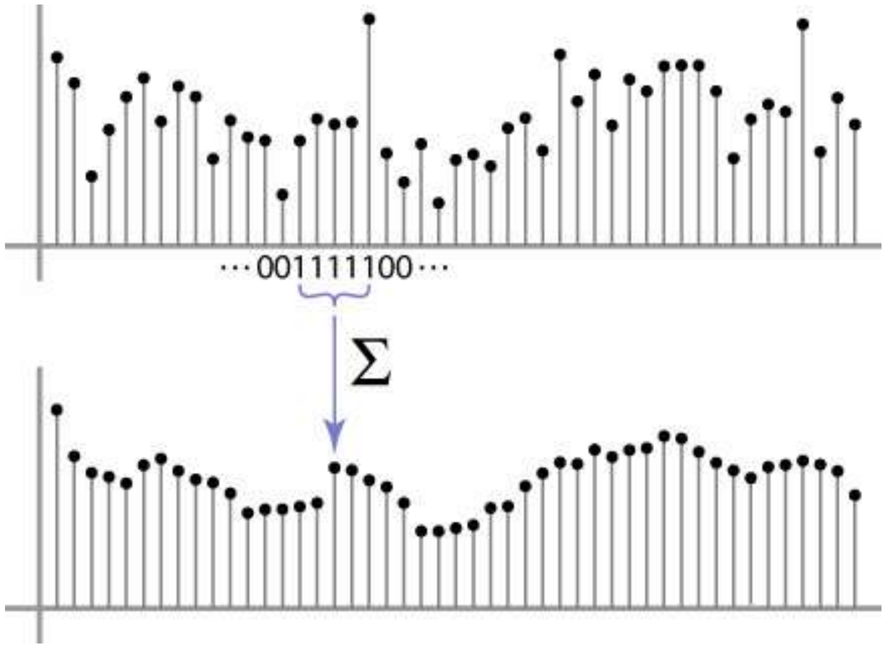
What would be the result of a mean filter?

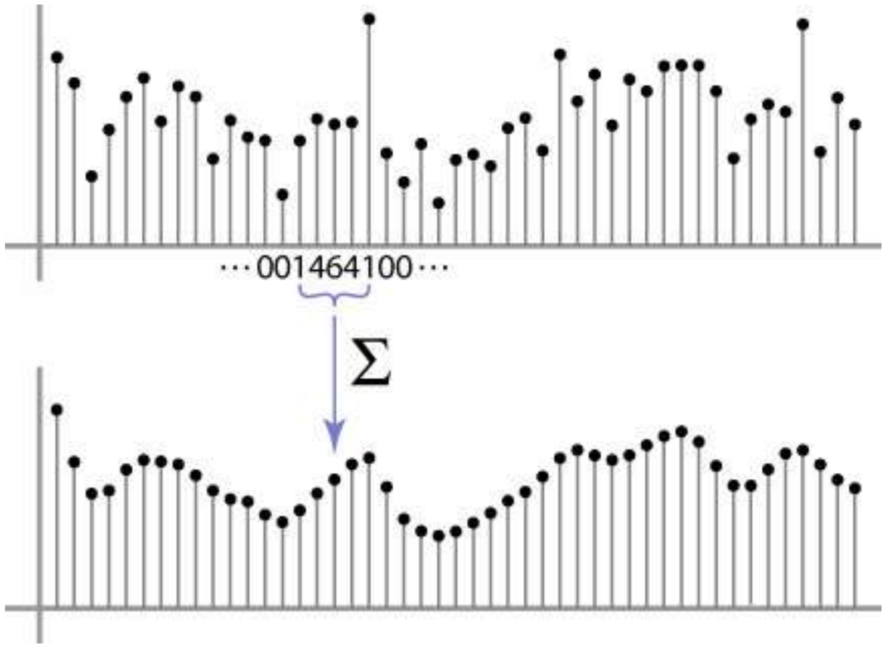
Today

- What is an image?
  - Types, color vs gray level
- How to measure the quality of an image?
  - Spatial and photometric resolution
  - Histogram and image contrast enhancement
- How to process by a linear filter
  - Examples: smoothing filters
  - Convolution / correlation
  - Mean and median filters
  - Linear filters with Gaussians

Weighted Moving Average

- *Weights* [1, 1, 1, 1, 1] / 5
- Non-uniform weights [1, 4, 6, 4, 1] / 16





Adding weights to our moving average? Why?

Source: S. Marschner

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

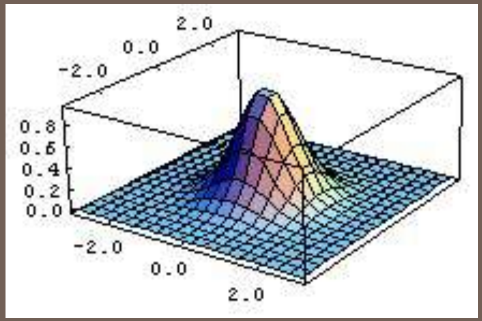
$F[x, y]$

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

$H[u, v]$

This kernel is an approximation of a 2D Gaussian function:

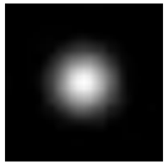
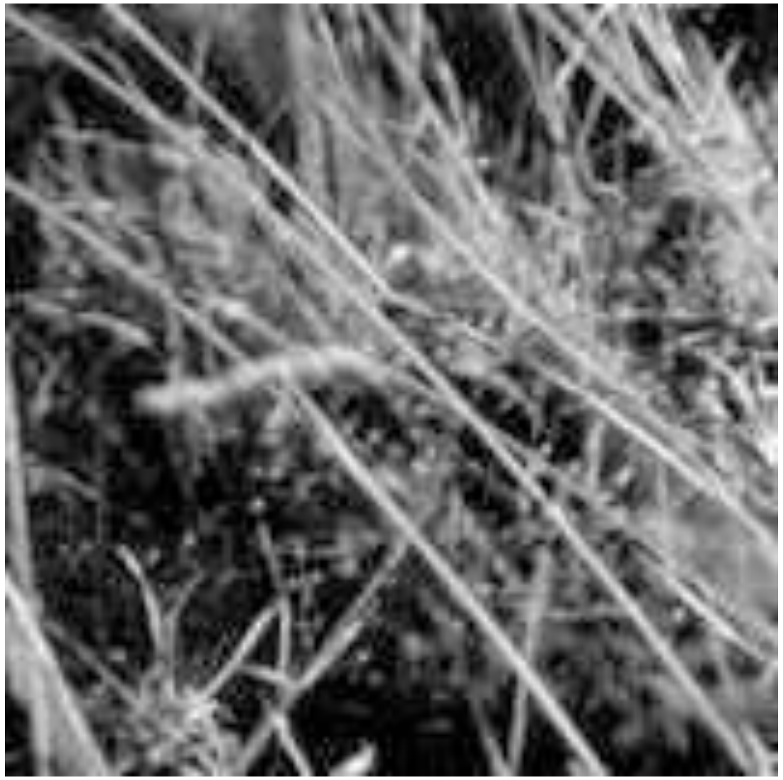
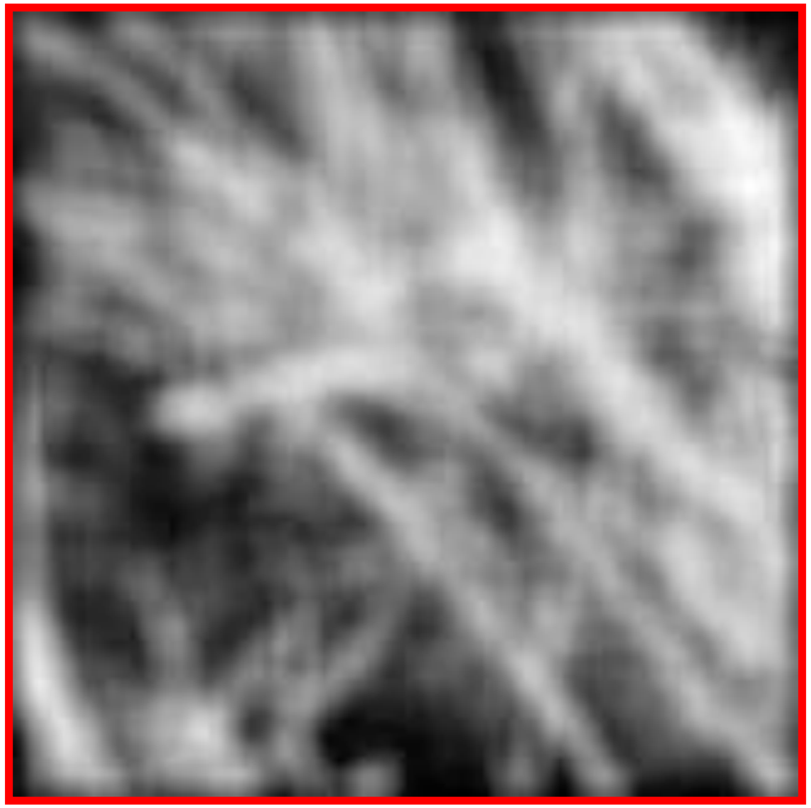
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Removes high-frequency components from the image (“low-pass filter”).

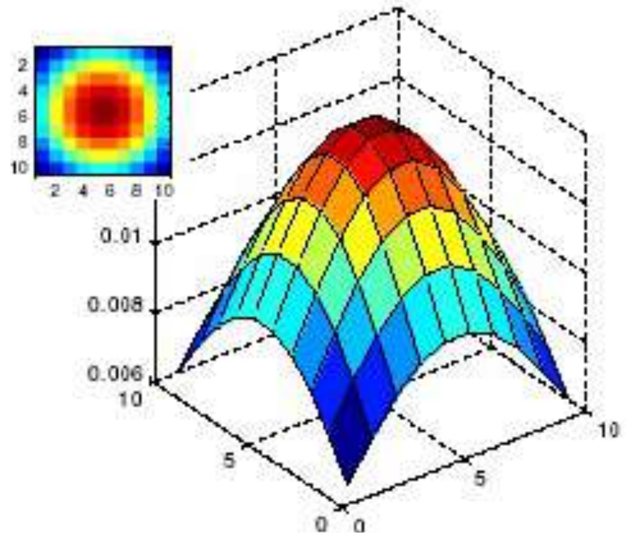
Source: S. Seitz

Smoothing with a Gaussian

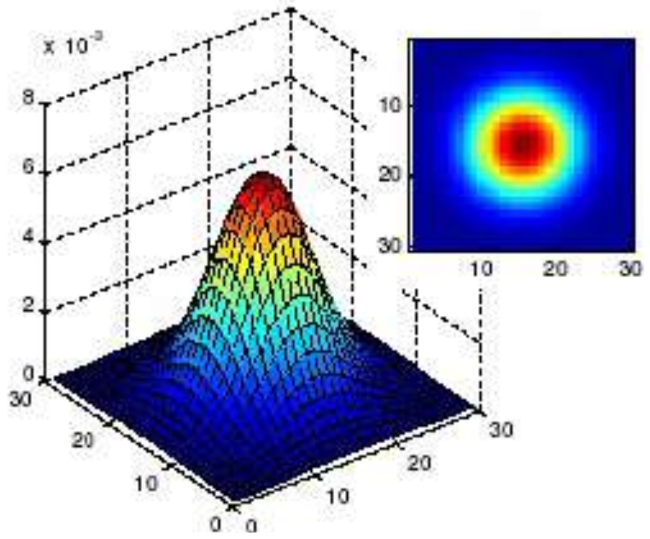




Gaussian filters

- What parameters do matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



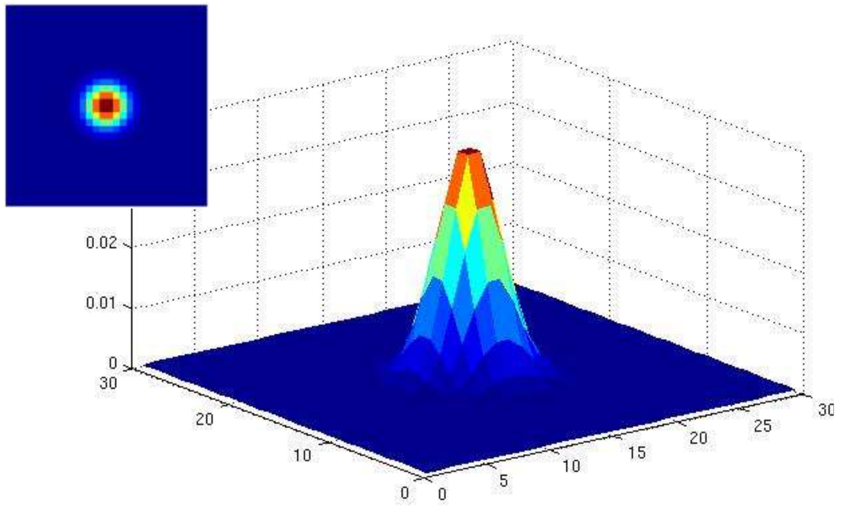
σ = 5 with 10 x 10 kernel



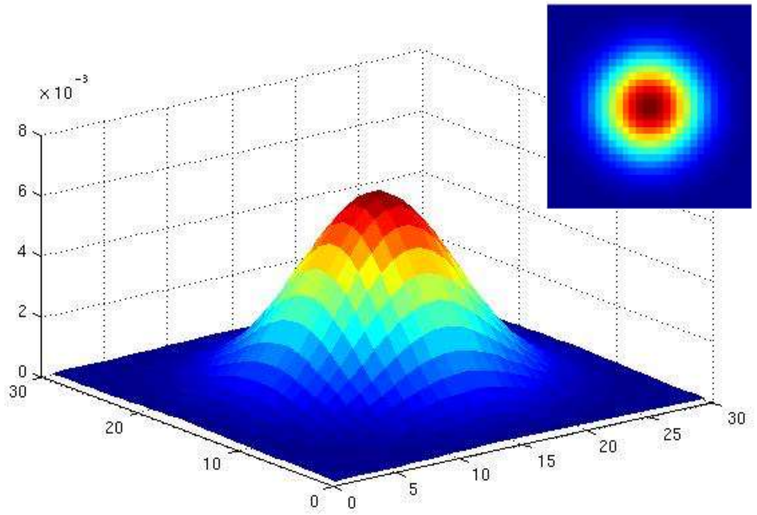
σ = 5 with 30 x 30 kernel

Gaussian filters

- What parameters do matter here?
- **Variance** of Gaussian: determines extent of smoothing



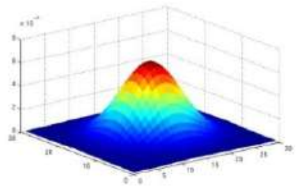
σ = 2 with 30 x 30 kernel

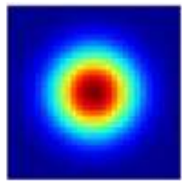


σ = 5 with 30 x 30 kernel

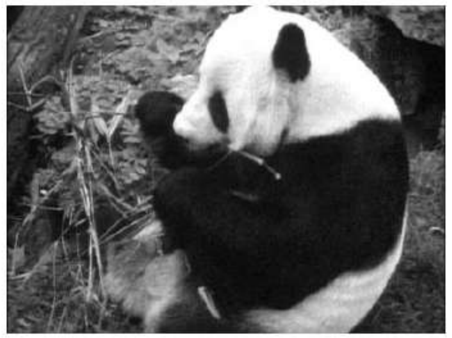
Smoothing with a Gaussian filter

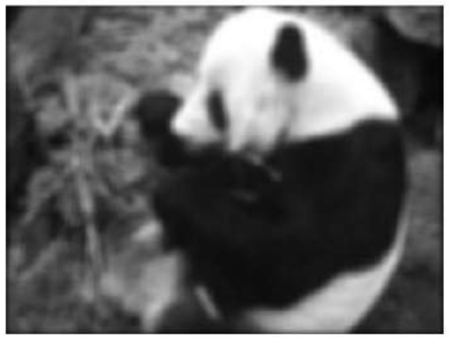
Applying a smoothing operation with a Gaussian with sigma=1





Gaussian filter

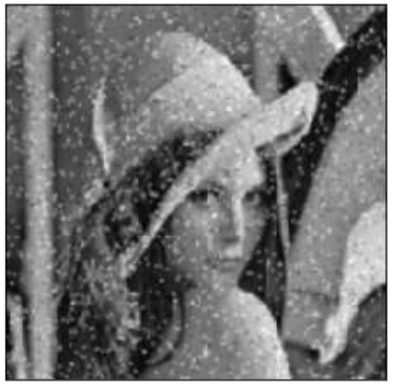


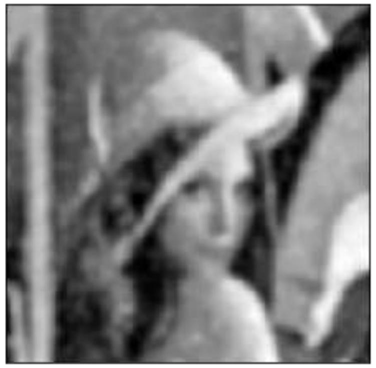


output

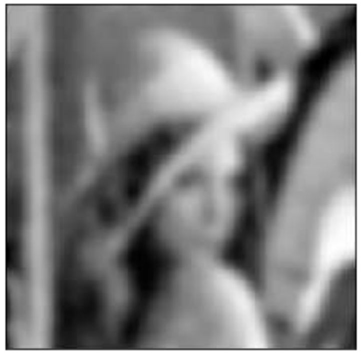
Smoothing with a Gaussian filter

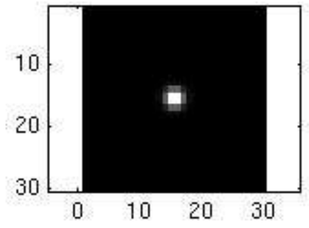
Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

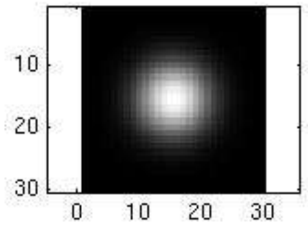


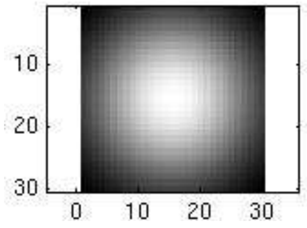


…



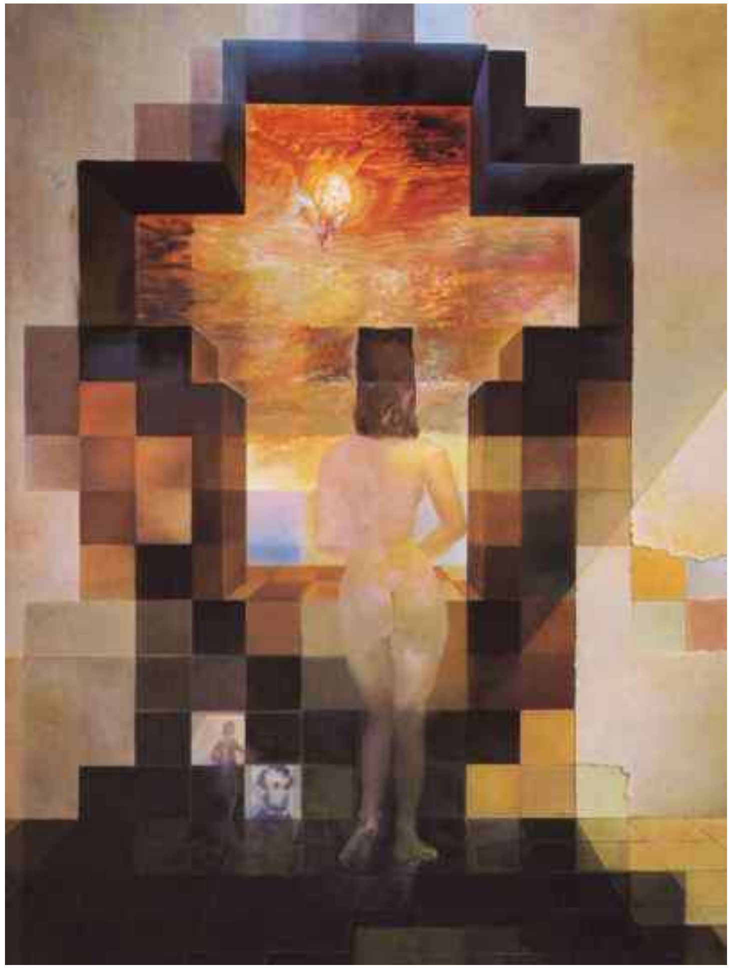


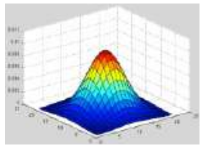


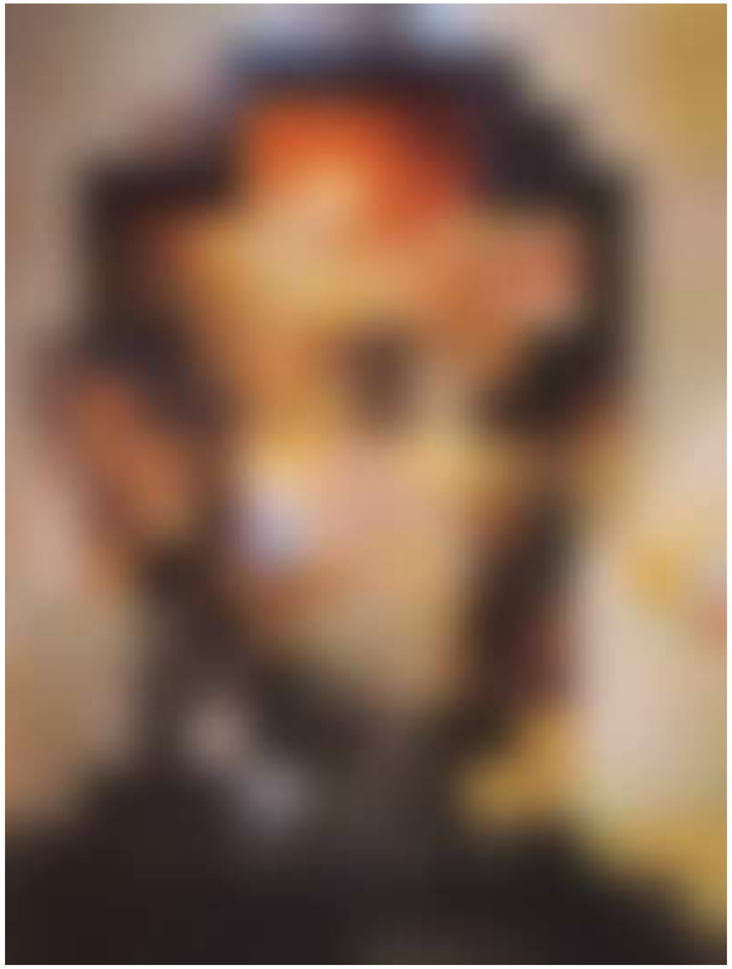


Applying Gaussian smothing with different sigma parameters

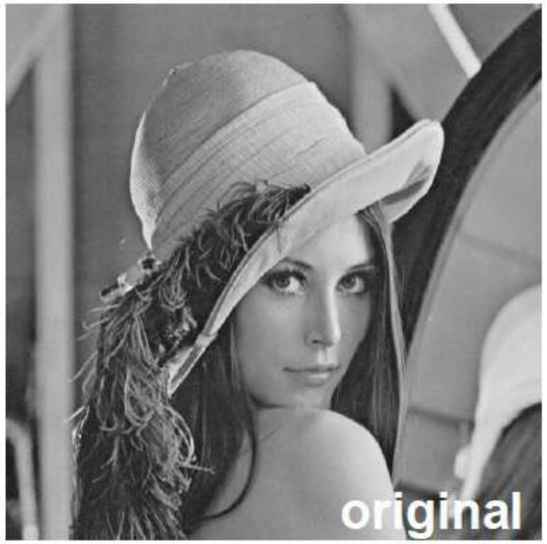
Local vs global analysis


Dali

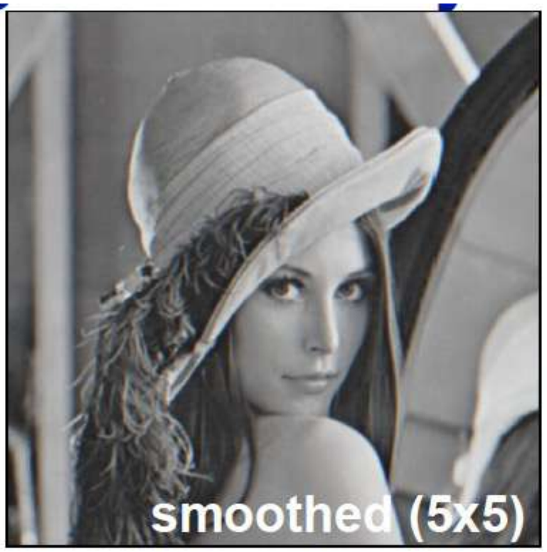




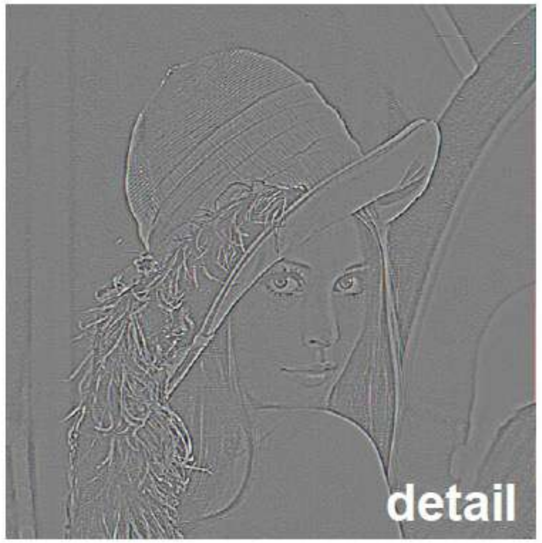
Sharpening



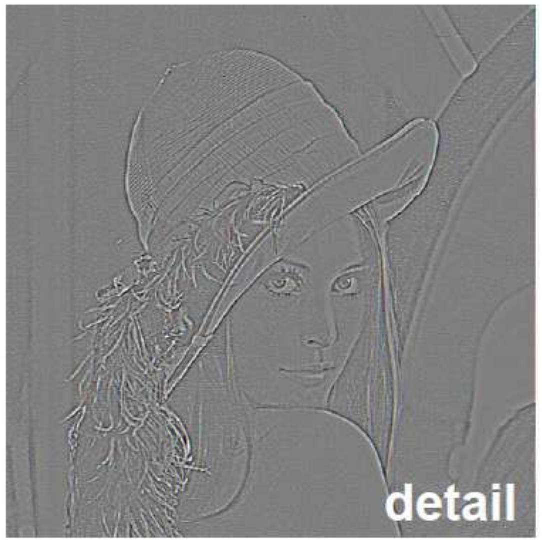
–



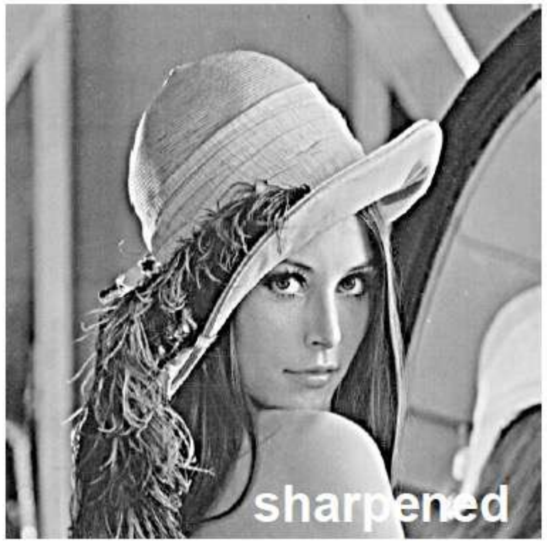
=



+ a



=



Summary

- Digital images: resolution, “noise”
- Histograms – a tool to visualize the statistical distribution of grey levels of pixels
- Linear filters and convolution useful for
  - Enhancing images (smoothing, removing noise)
    - Box filter
    - Impact of scale / width of smoothing filter
- Gaussians – how to use analytical functions to control processing scale
- Next: convolutions for image Gradient estimation

Properties of smoothing filters

- Smoothing
  - Values positive
  - Sum to 1 $\rightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Test:
  - Smoothing can improve ………. resolution and quality
  - Smoothing does not improve ……. resolution
  - ……………. resolution affects the speed of the algorithms
  - ……………. resolution does not affect the speed of the algorithms.

References

Source for slides are from the courses of:

- J. Malik, UBerkeley
- Prof. K. Grauman, UT-Austin
- D. Hoiem
- S. Marschner
- D. Lowe
- S. Seitz
- Some Figures from [Mori et al]

Lecture videos

- Lesson 3: 2A-L2 Filtering
  - 2 - Gaussian Noise
  - Until - Variance or Standard Deviation
  - 17 - Gaussian Filter Quiz
  - 18 - Keeping the Two Gaussians Straight

From Introduction to Computer Vision:

https://www.udacity.com/course/introduction-to-computer-vision--ud810