Linear Filters

Class 2

Artificial Vision

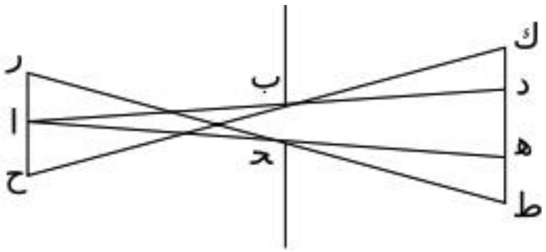
Most slides from: Prof. Kristen Grauman, UT-Austin

Today

- Image construction
- Spatial and photometric resolution
  - Histogram and image contrast enhancement
- Linear filters
  - Examples: smoothing filters
- Convolution / correlation
- Smoothing
- Linear filters with Gaussians

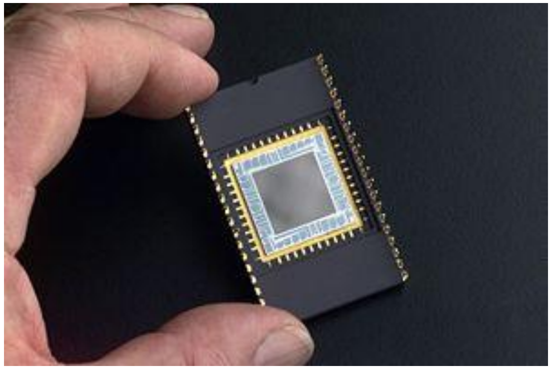
Historical context

- **Pinhole model:** Mozi (470-390 BCE), Aristotle (384-322 BCE)
- **Principles of optics (including lenses):** Alhacen (965-1039 CE)
- **Camera obscure:** Leonardo da Vinci (1452-1519)
- **First photo:** Joseph Nicephore Niepce (1822)
- **Cinema** (Lumière Brothers, 1895)
- **Color Photography** (Lumière Brothers, 1908)
- **Television** (Baird, Farnsworth, Zworykin, 1920s)
- **First consumer camera with charge-coupled device (CCD):** Sony Mavica (1981)
- **First fully digital camera:** Kodak DCS100 (1990)



Alhacen's notes

Niepce, "La Table Servie," 1822

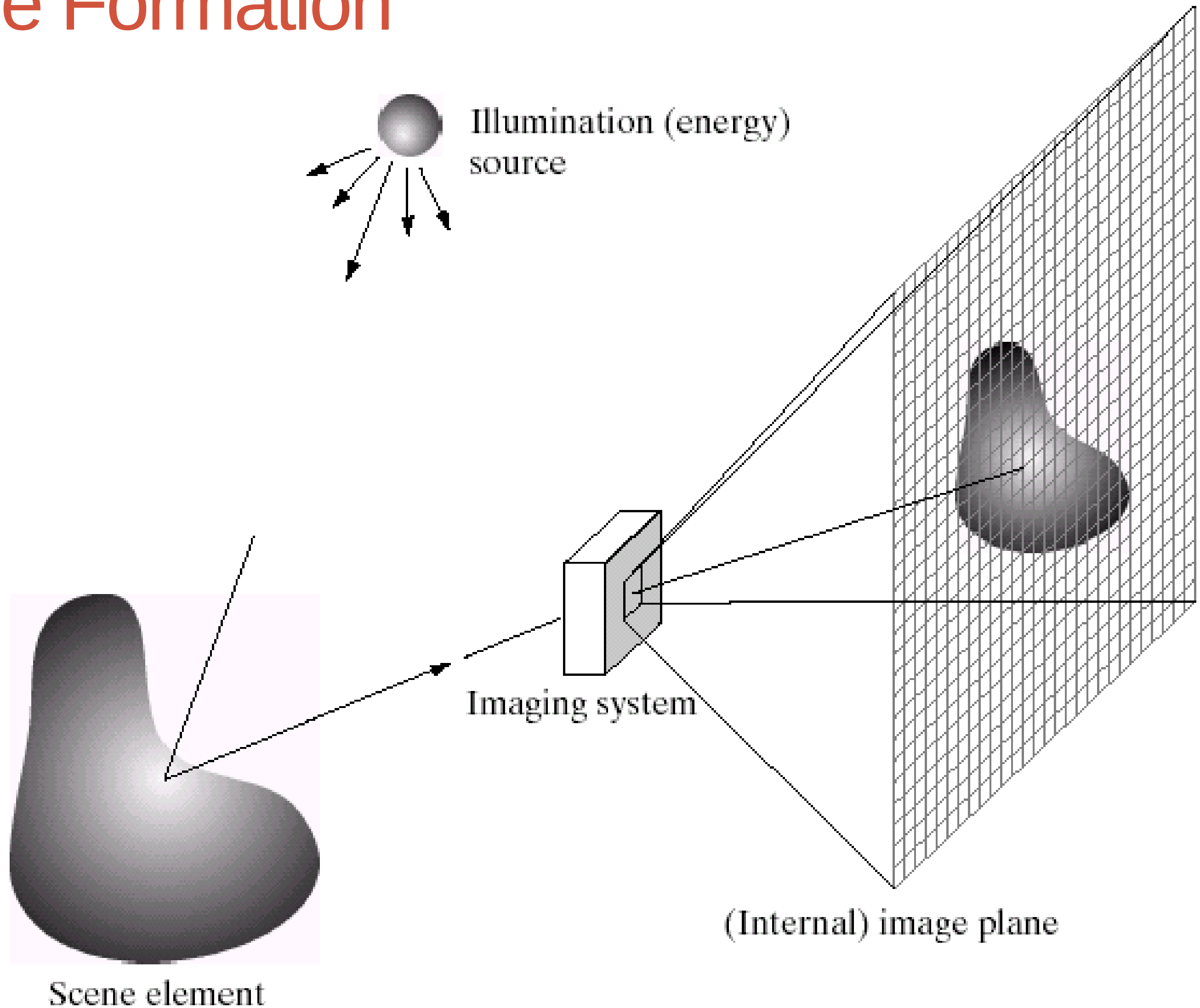


CCD chip

Slide credit: L. Lazebnik

K. Grauman

Image Formation



Slide credit: Derek Hoiem

Digital camera

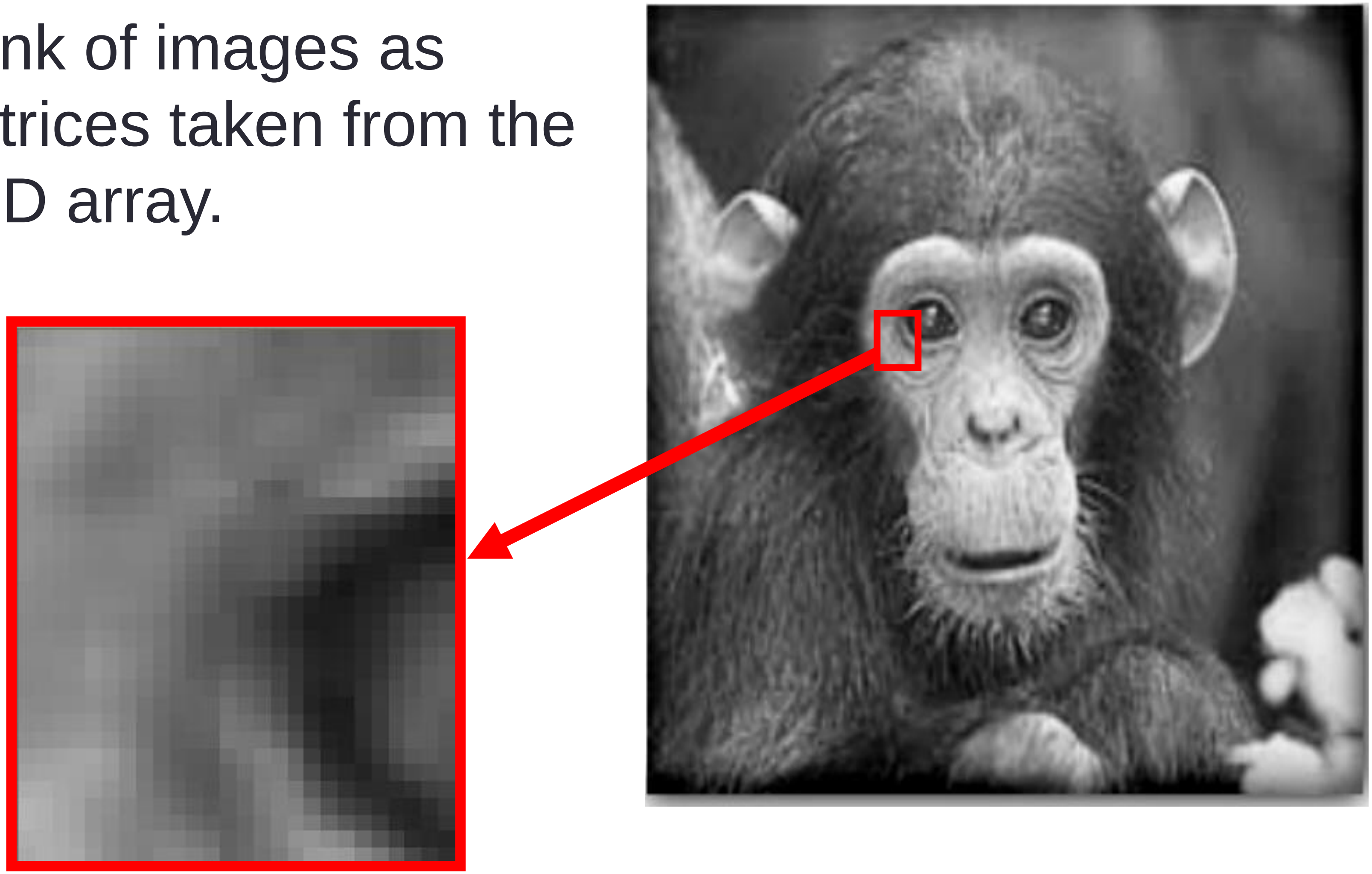
A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons

Slide by Steve Seitz

Digital images

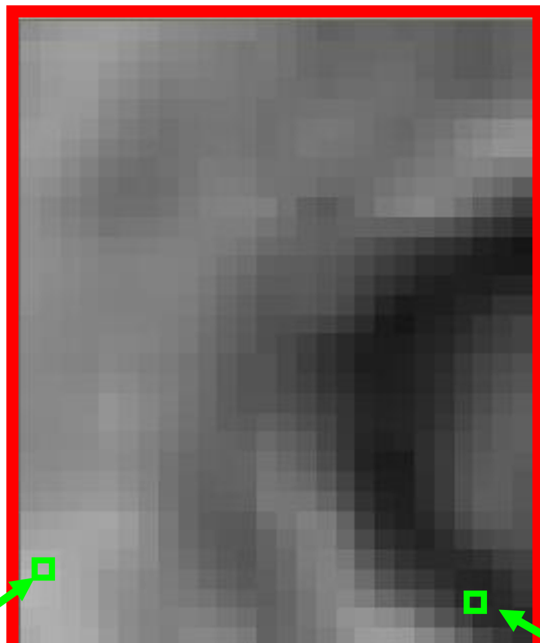
Think of images as matrices taken from the CCD array.

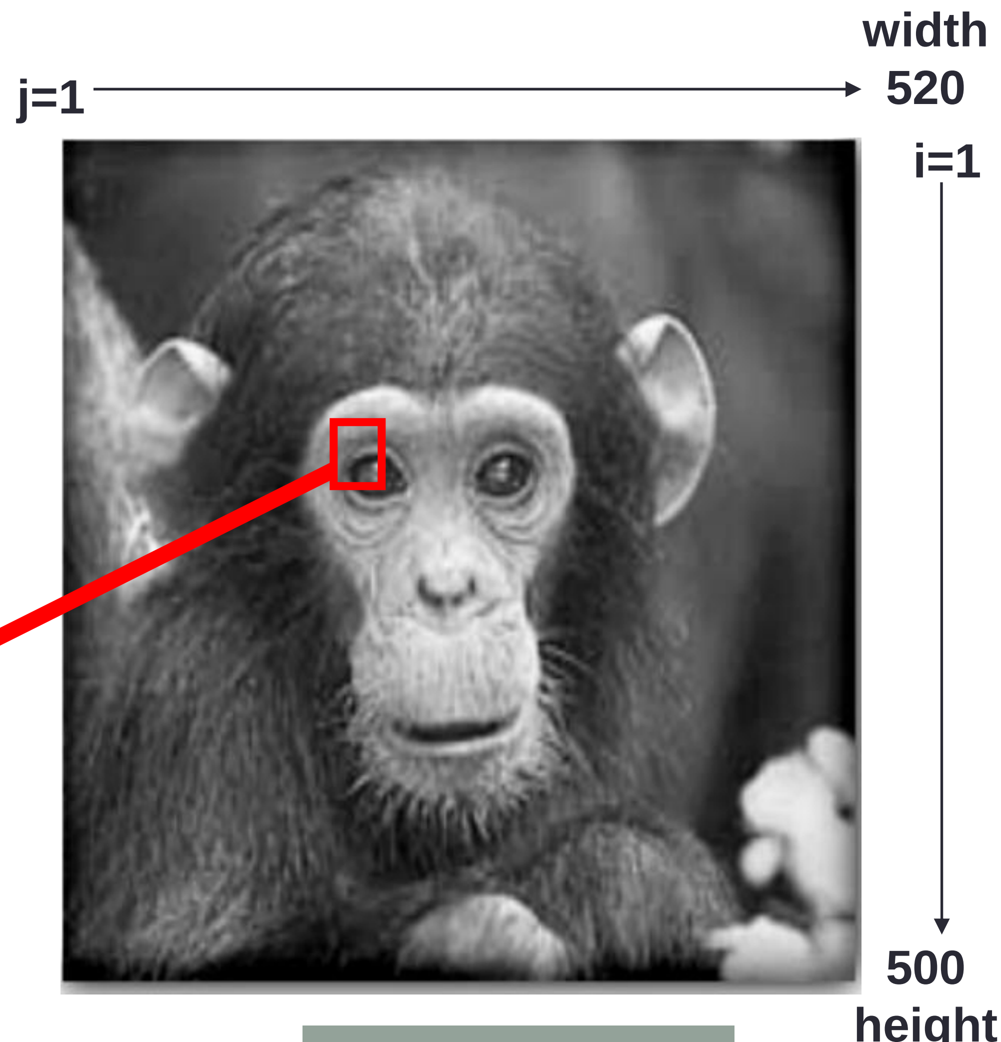


K. Grauman

Digital images

Intensity : [0,255]





im[176][201] has value 164

im[194][203] has value 37

Matlab:
>>im=imread(‘monkey.jpg’);
>>size(im)
ans= 500 520
>>im(10,20) % grey imatge
ans= 20

K. Grauman

Images in Matlab

- Images represented as a matrix
- Type (class) of images: double, uint8, indexed, binary.
- Rule:
  - When processing – convert into double.
    - im=zeros(256, 256, 'double')
  - When visualizing or saving – convert to uint8.
    - figure, imshow(uint8(im))
- How many values can have a double image? What is the maximal/minimal possible value of it?
- How many values can have an uint8 image? What is the maximal/minimal possible value of it?
- What is the value of pixel (1,1) in:

Matlab:

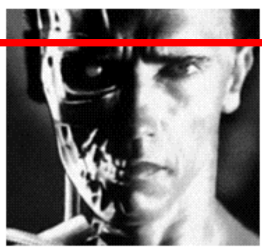
```
>>im=ones(256, 256);
>>im=uint8(im);
>>disp(im(1,1))

>>im(1,1)=256;
>>disp(im(1,1))

>>im(1,1)=1000;
>>disp(im(1,1))
```

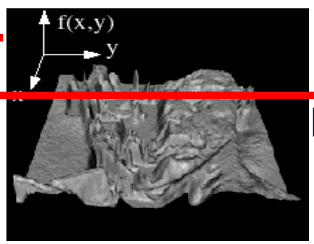
Slide credit: Derek Hoien

Digital images

- **Sample** the 2D space on a regular grid

- **Quantize** each sample (round to nearest integer)





j →

i ↓

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

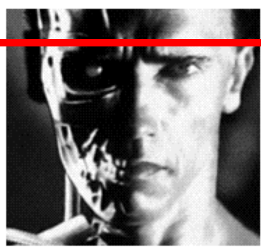
- Image is represented as a matrix of integer values.

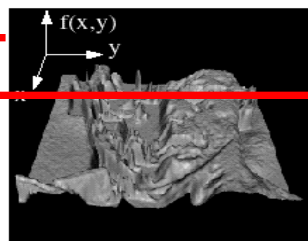
Adapted from S. Seitz

Digital images

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

- Image is represented as a matrix of integer values.



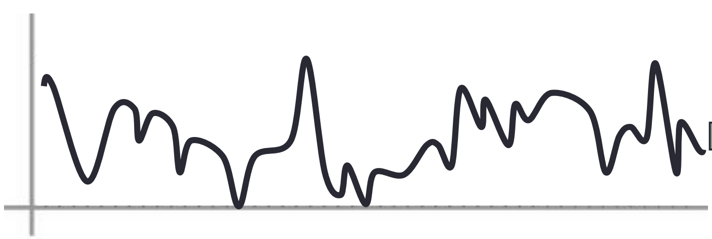


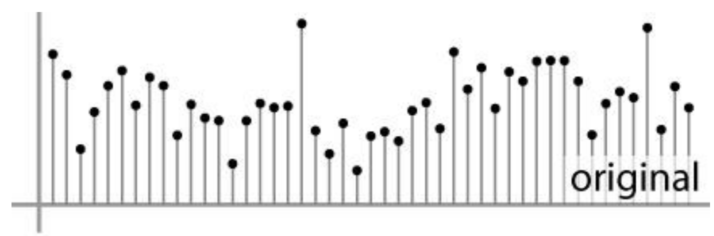
j

i

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



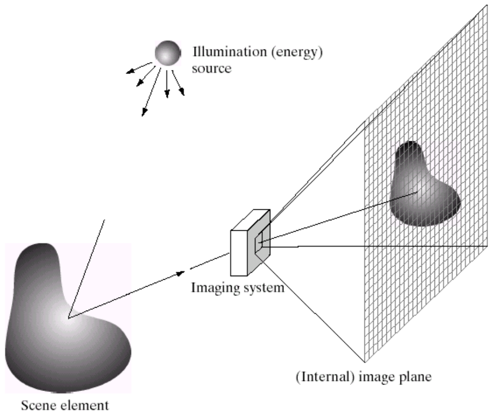


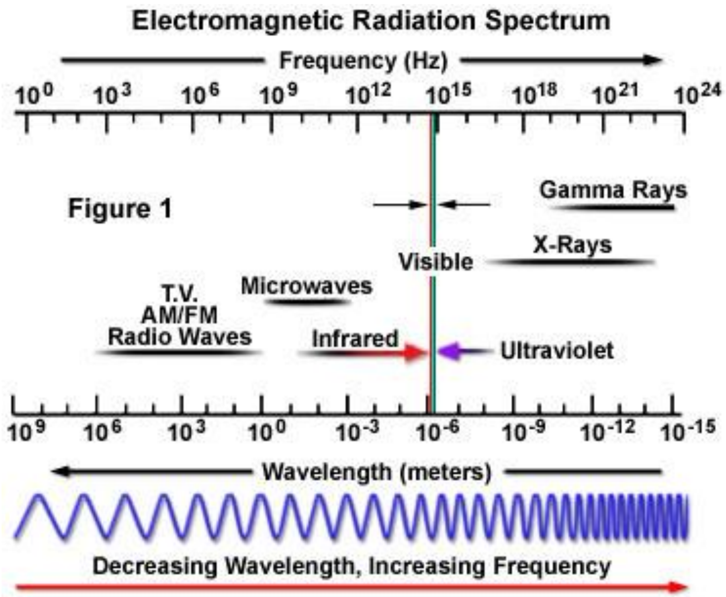
1D

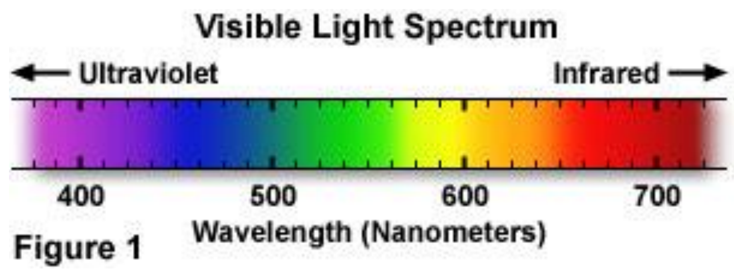
Adapted from S. Seitz

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!



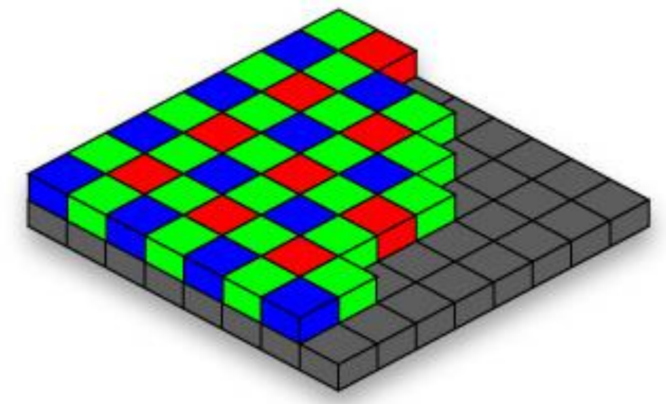


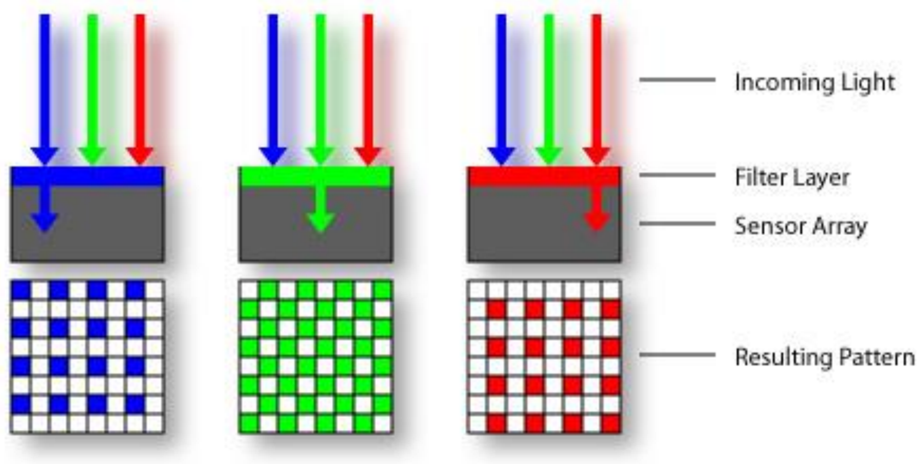


A typical human eye will respond to wavelengths from about 380 to 750 nm.

Color sensing in digital cameras

Bayer grid





Estimate missing components from neighboring values (demosaicing)

Source: Steve Seitz

Images in Matlab

- Images can be grey-value (1 channel) or color images (3 channels)

Matlab:
>>imGV=zeros(25,25)
>>imCOL=zeros(25,25,3)

- Suppose we have an NxM RGB image called "im"
  - im(1,1,1) = top-left pixel value in R-channel
  - im(y, x, 3) = y pixels down, x pixels to right in the bth channel
  - im(N, M, 3) = bottom-right pixel in B-channel

Matlab:
>>im=imread('flowers.jpg');
>>size(im)
ans= 500 520 3
>>im(10,20) % grey imatge
ans= 20
>>im(:,:,2) % G channel
.....

column

row

R

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

G

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

B

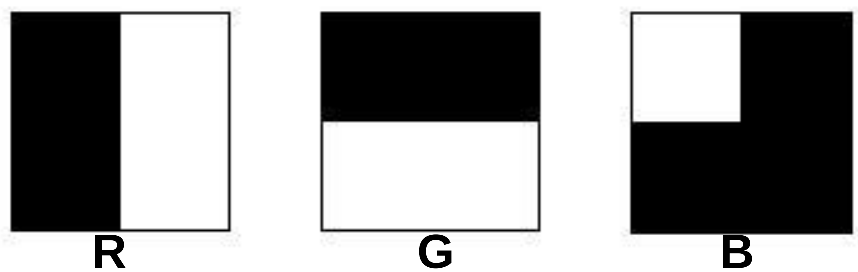
0.92	0.99
0.95	0.91
0.91	0.92
0.97	0.95
0.79	0.85
0.45	0.33
0.49	0.74
0.82	0.93
0.90	0.99

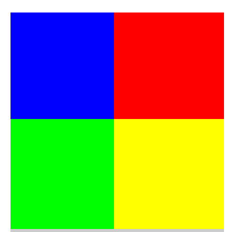
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

Slide credit: Derek Hoiem

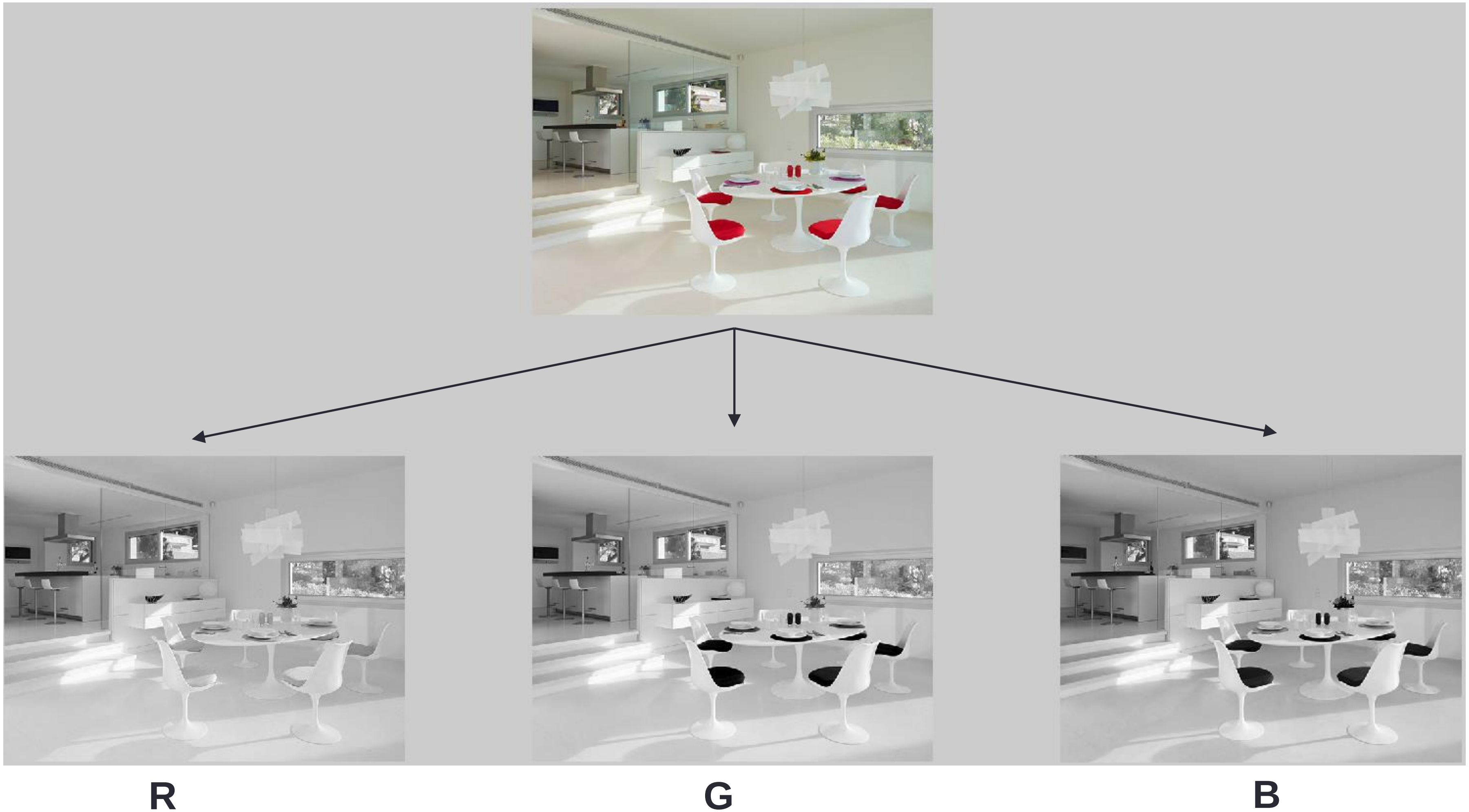
Exercise

- What are the colors of each quadrant, if we compose a color image with the following channels?





Color images, RGB color space



Why are the chairs tops appearing in black?
What are the values of the chair pixels in the color image?

Today

- Image construction
- Spatial and photometric resolution
  - Histogram and image contrast enhancement
- Linear filters
  - Examples: smoothing filters
- Convolution / correlation
- Smoothing
- Linear filters with Gaussians

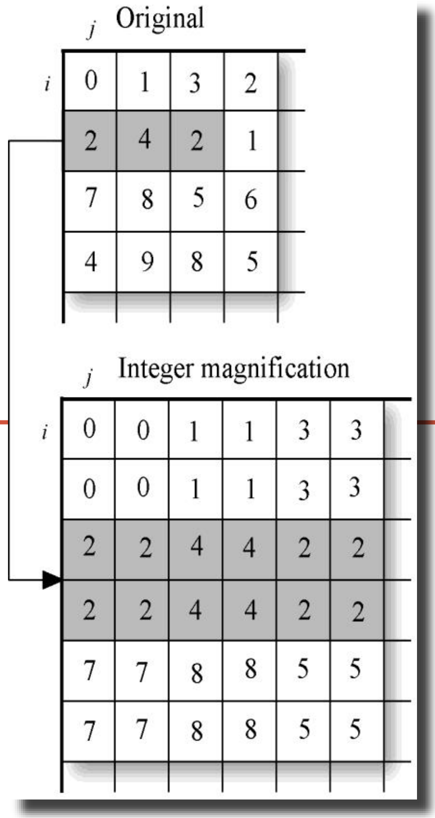
Spatial resolution

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

[fig from Mori et al]

Influences what analysis is feasible, it affects best representation choice.

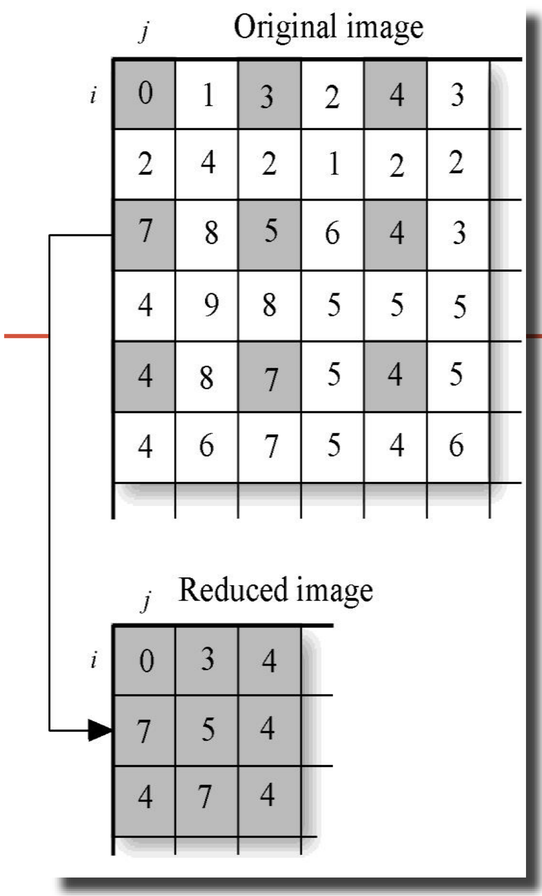
Image magnification

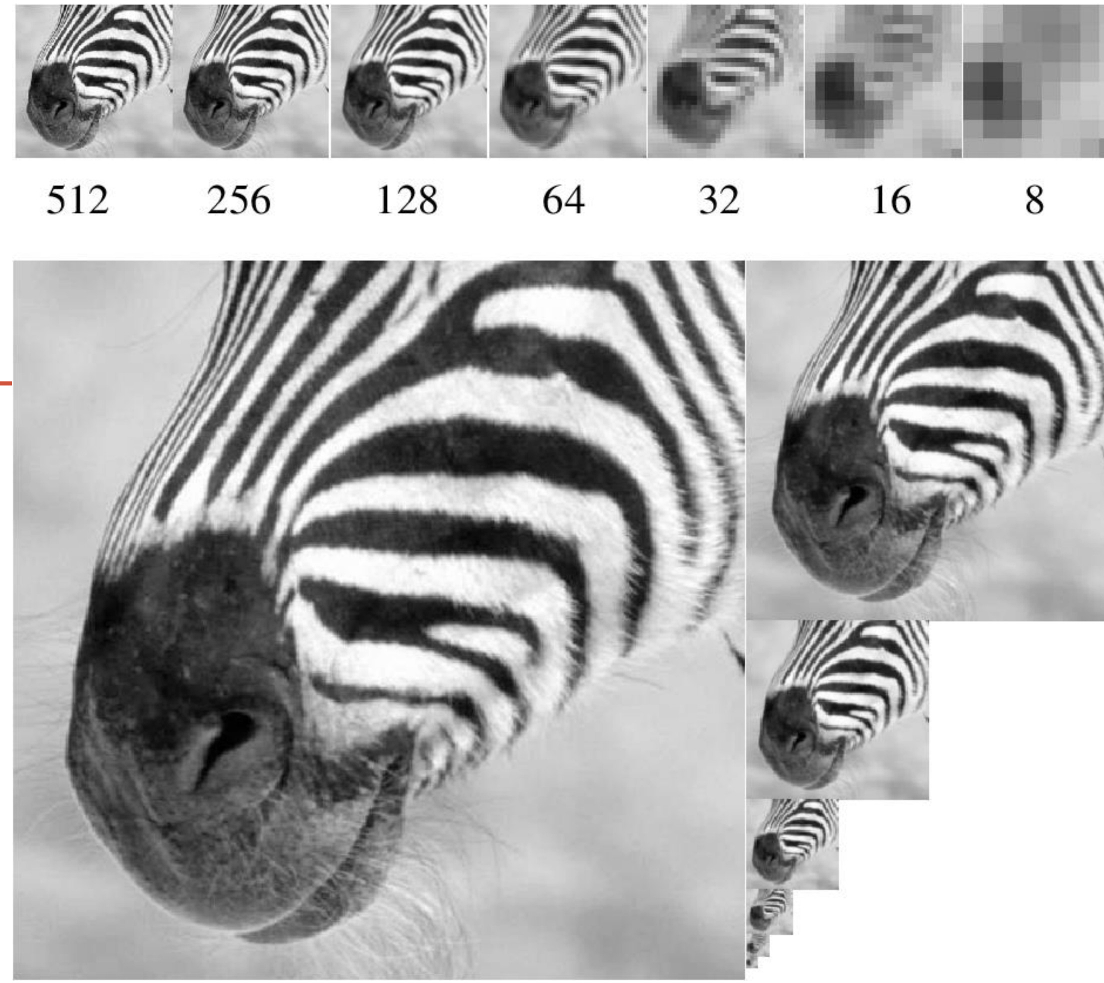


The number of pixels determines the spatial resolution of an image (imresize()).

Example: imresize(im,2) => ?
imresize(im,0.5) =>?

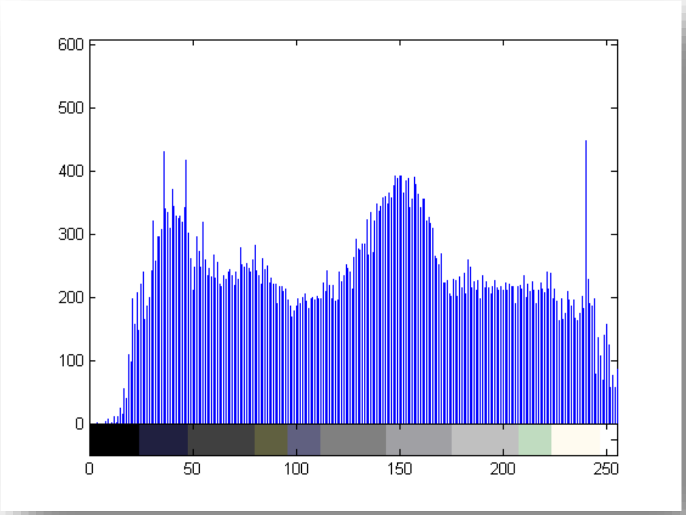
Image reduction





Matlab: imresize(im,0.5)

Photometric resolution



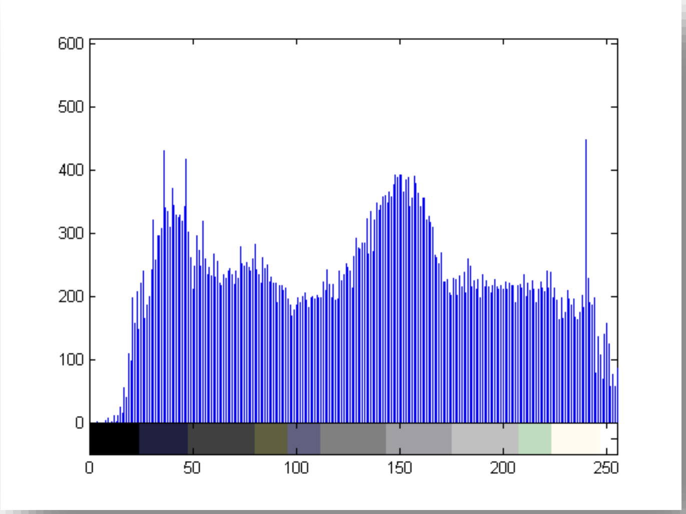
>>mm=zeros(256, 256, 'uint8'); %Creating an image of grey level ____?
Given an image of type uint8, how many grey levels we can have at most?

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



Matlab:

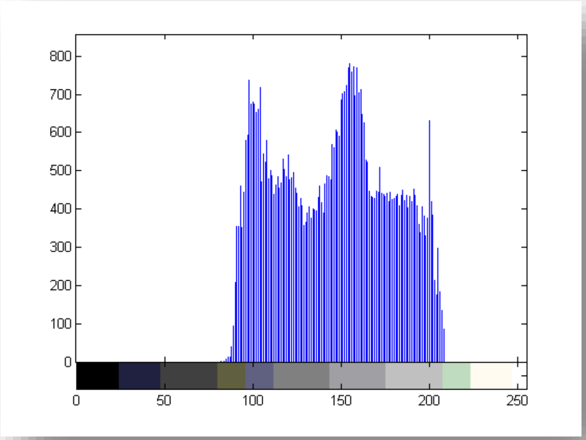
imhist(im)

[COUNTS,X] = imhist(im,n); % n=#bins

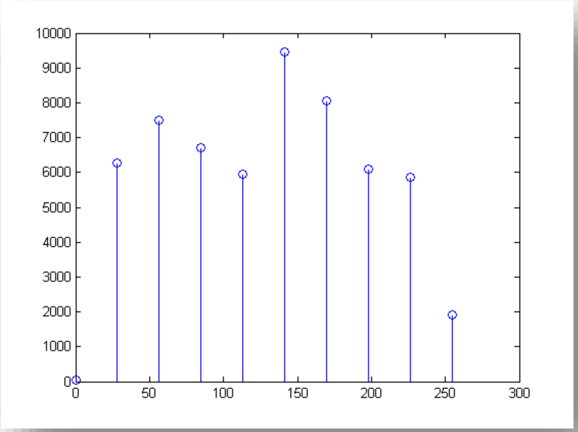
% returns the histogram counts in COUNTS and the bin locations in X so that stem(X,COUNTS) shows the histogram

stem(X,COUNTS) plots COUNTS according to bins X

Histogram



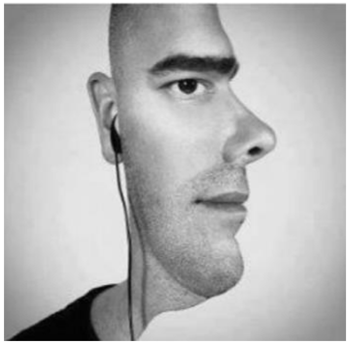
How will look the histogram of the right image?

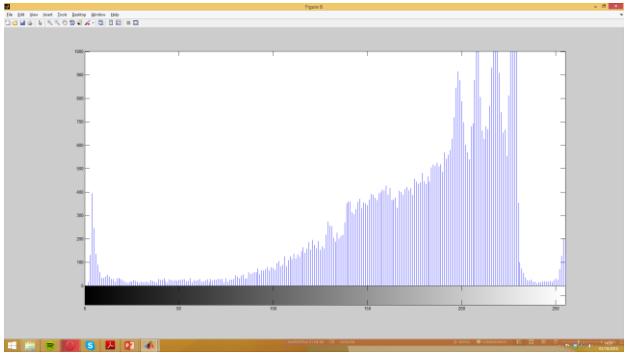


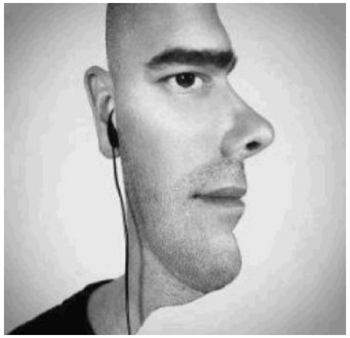
- Example: imhist(im(:,:,1)/2+80,10)

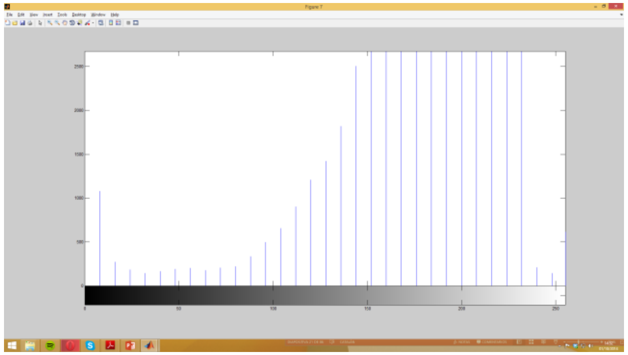
See Demo "Adjusting Pixel Intensity Values" of the "Image Processing Toolbox"

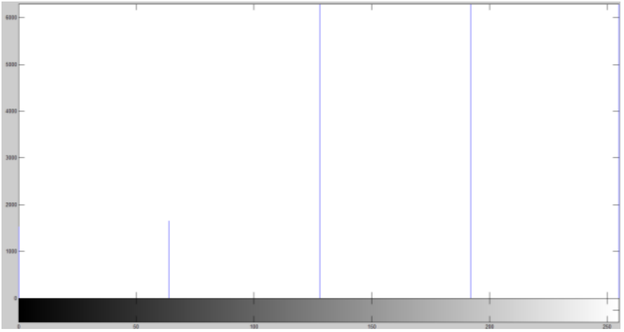
Histogram: How should the histograms look?



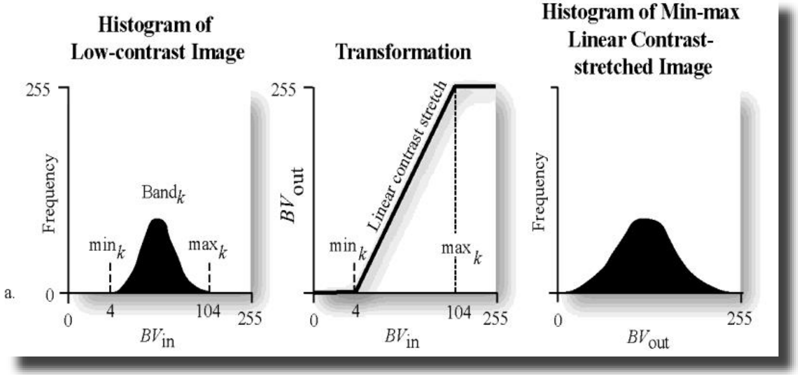








Histogram manipulation for contrast enhancement



Multiply the image to augment its contrast:

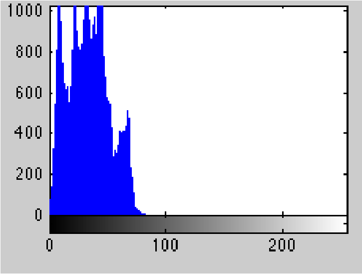
$$BV_{out} = \left(\frac{BV_{in} - \min_k}{\max_k - \min_k} \right) quant_k$$

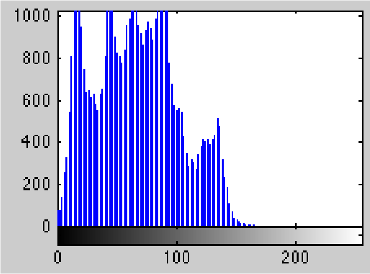
where:

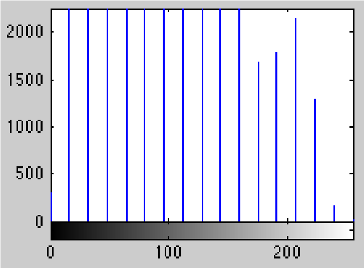
- BV_{in} is the original input brightness value (i.e. the original image)
- $quant_k$ is the range of the brightness values that can be displayed on the CRT (eg 255),
- min_k is the minimum value in the image,
- max_k is the maximum value in the image, and
- BV_{out} is the output brightness value.

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

- Image construction
- Spatial and photometric resolution
  - Histogram and image contrast enhancement
- Linear filters – mean filter
- Convolution / correlation
- Smoothing
- Median filter
- 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

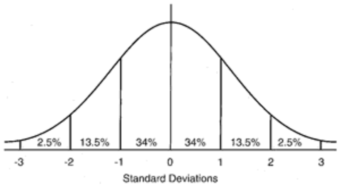
- **Filtering**: Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (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

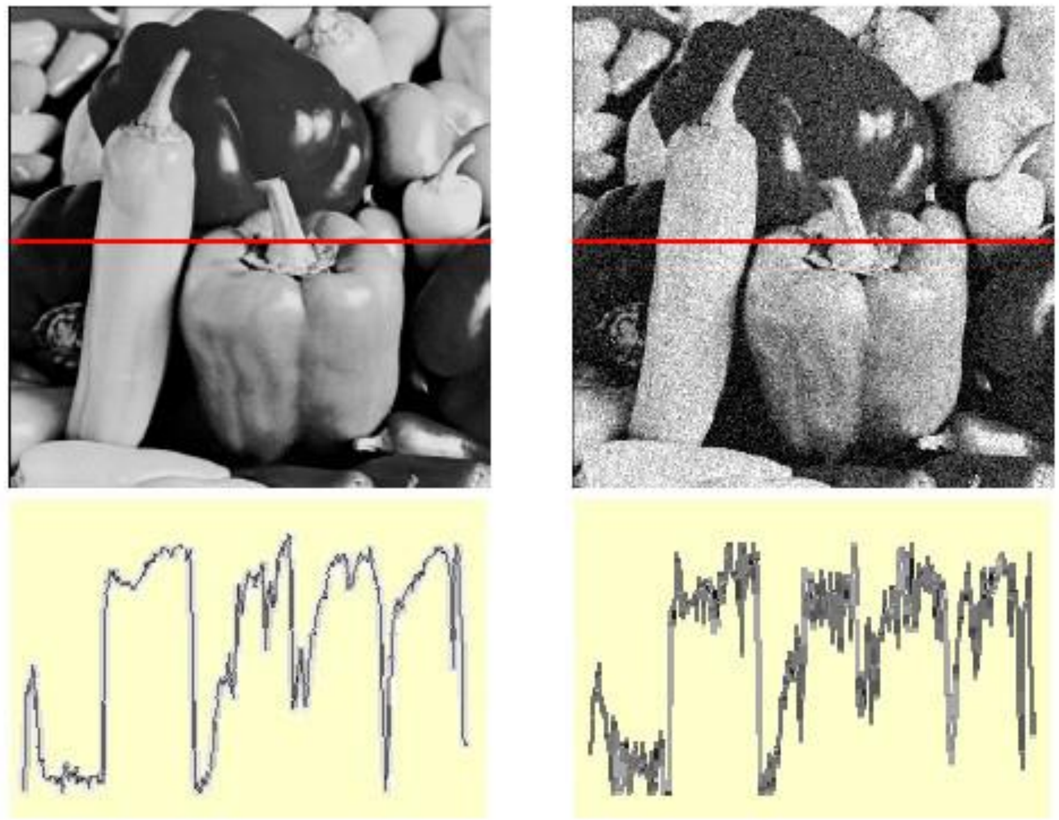
Salt and pepper noise

Impulse noise

Gaussian noise

Source: S. Seitz

Gaussian noise



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

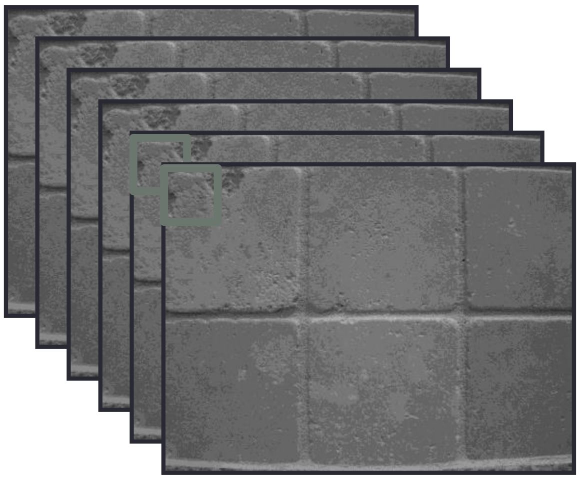
Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

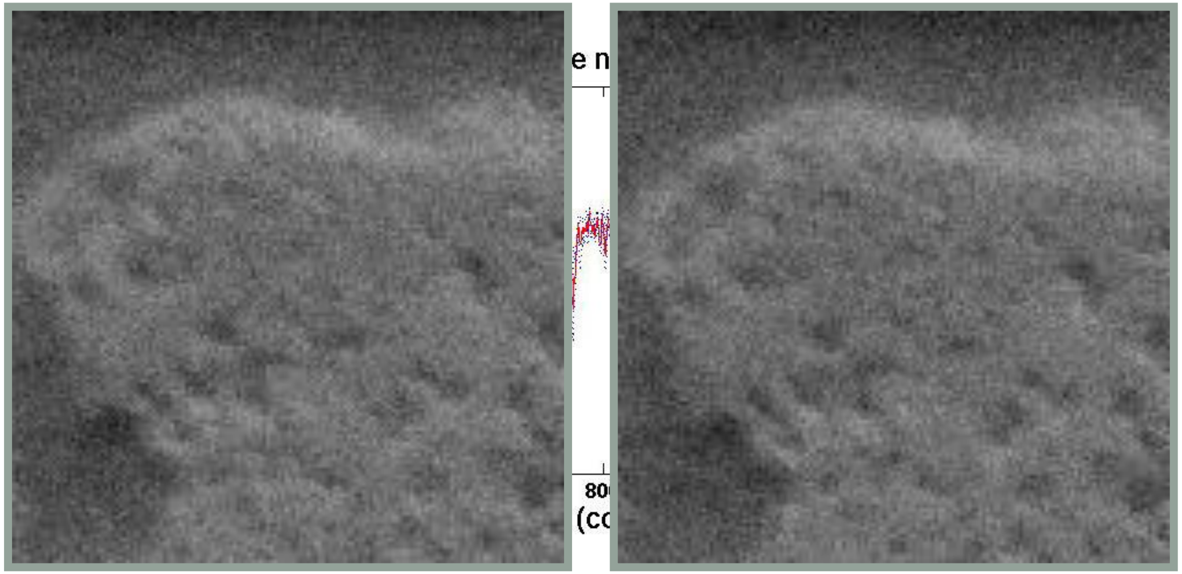
```
>> noise = randn(size(im)).*sigma;
>> output = im + noise;
```

What is the impact of sigma?

Fig: M. Hebert

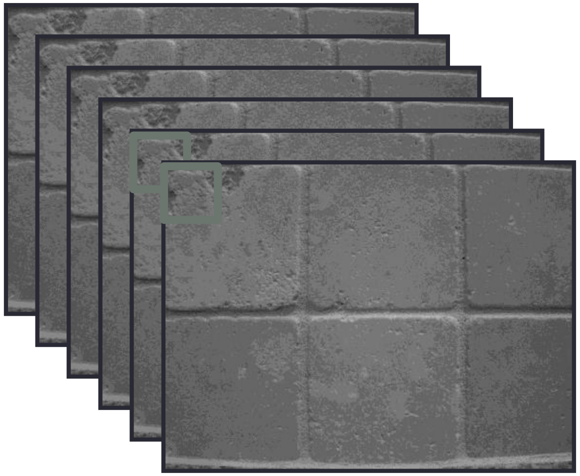
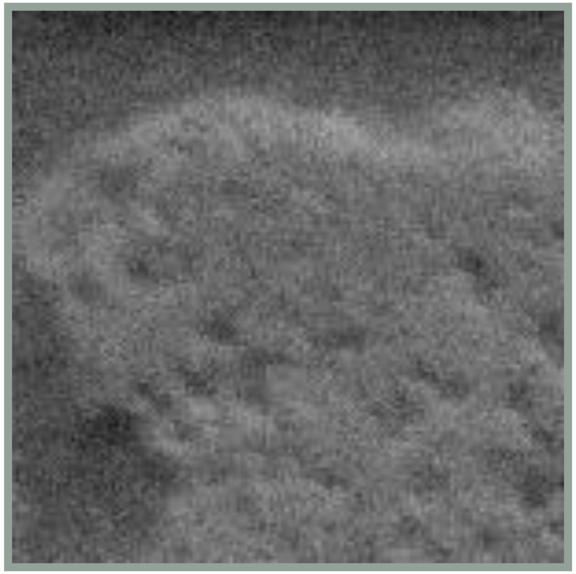
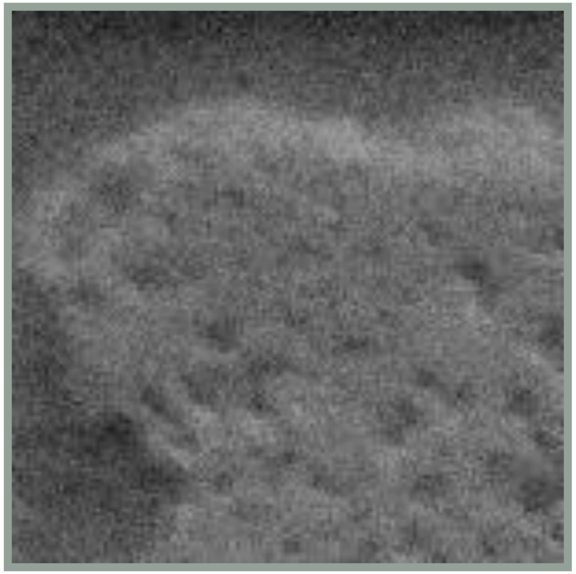
Motivation: noise reduction





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

Motivation: noise reduction

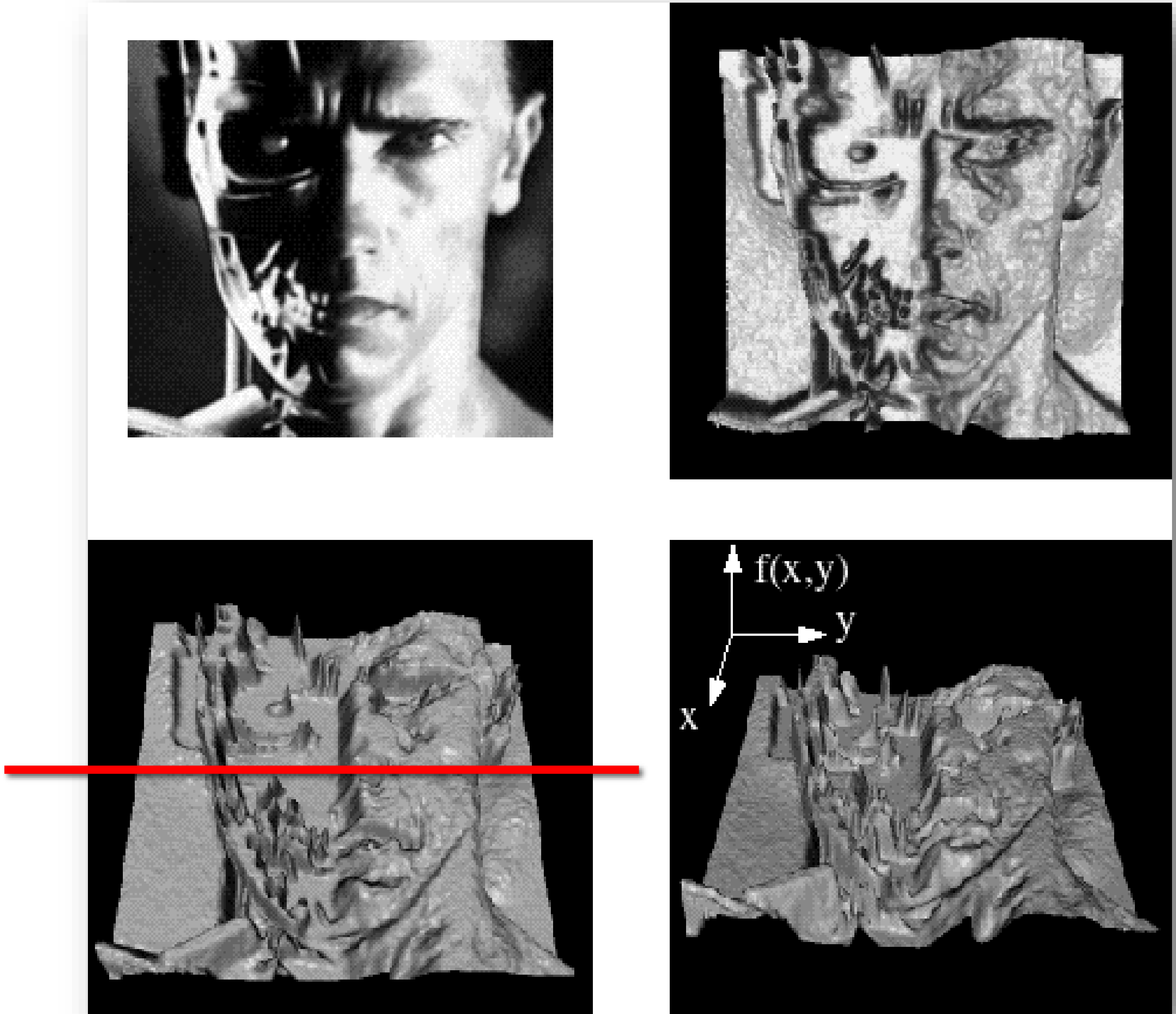
- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
  - Take the average of the grey values per pixel.

- **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

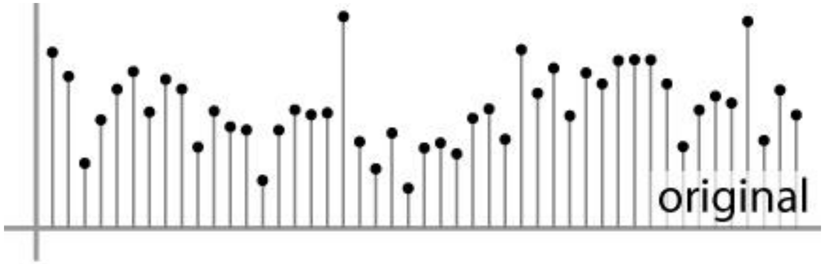
Remember: an image is a matrix & topographic map

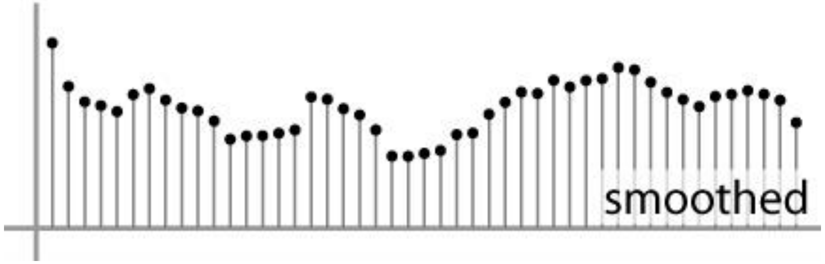


First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood

- Moving average in 1D:

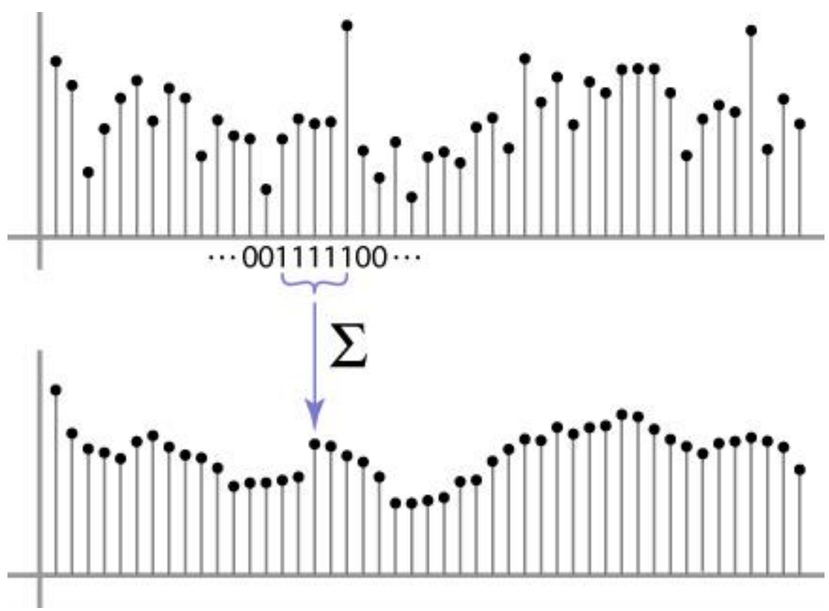




Source: S. Marschner

Weighted Moving Average – Mean filter

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

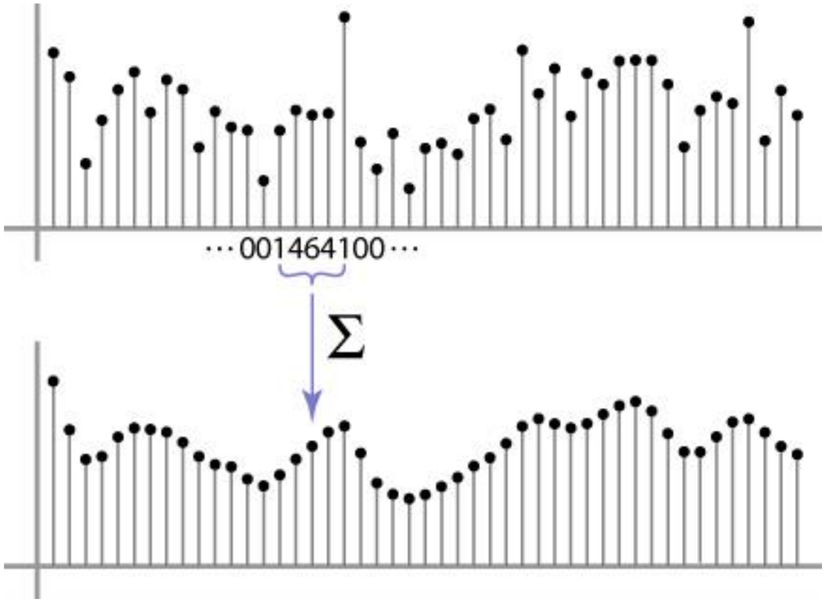


Can we add weights to our moving average? Why?

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

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

Convolutional filtering

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

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

Attribute uniform weight to each pixel

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

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

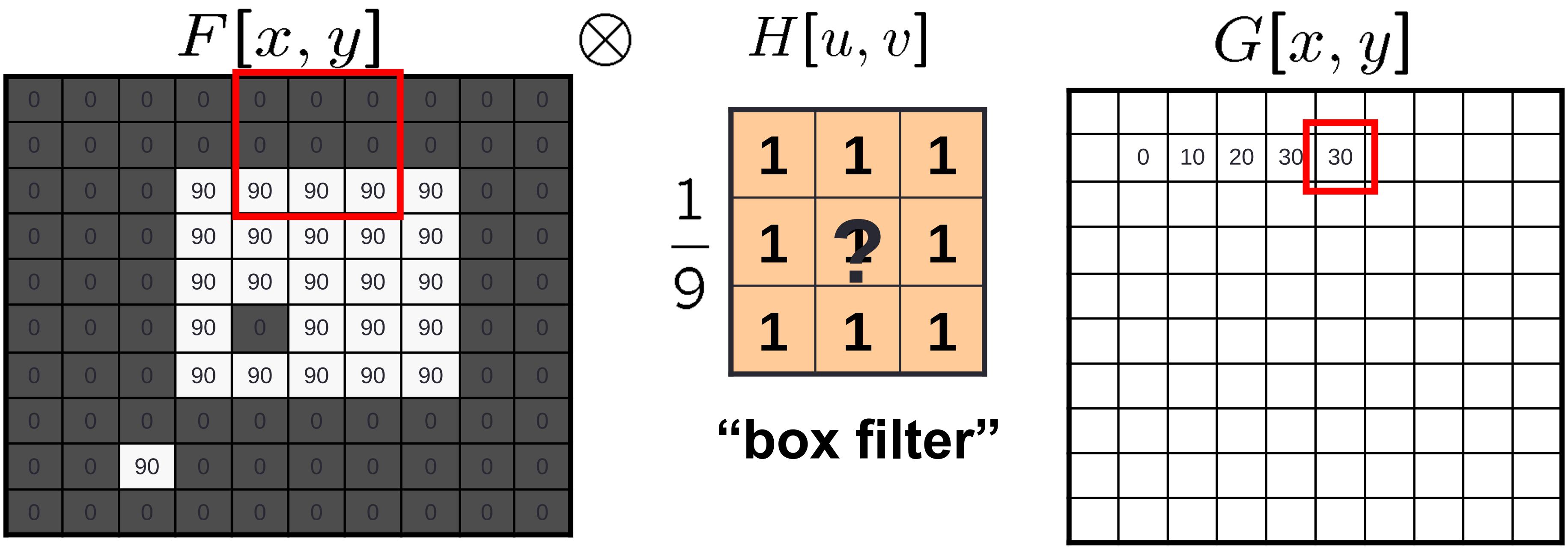
This is called **convolution**, denoted as: $G = H \otimes F$

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?



$$G = H \otimes F$$

Mean filter

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

$F[x,y]$

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

$\otimes$

$H[u,v]$

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

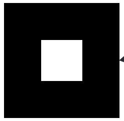
"box filter"

$G[x,y]$

	0	10	20	30	30				

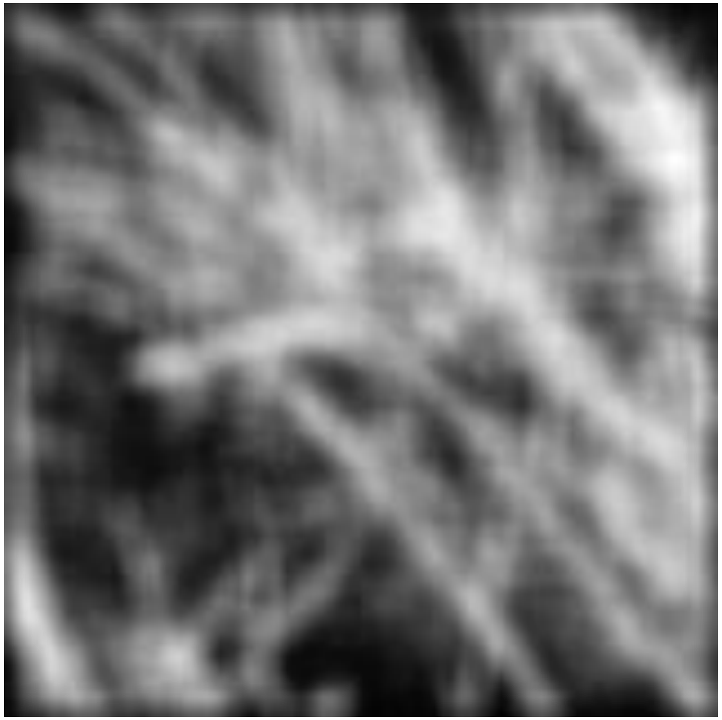
$$G = H \otimes F$$

Smoothing by averaging



depicts box filter:
white = high value, black = low value

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*?

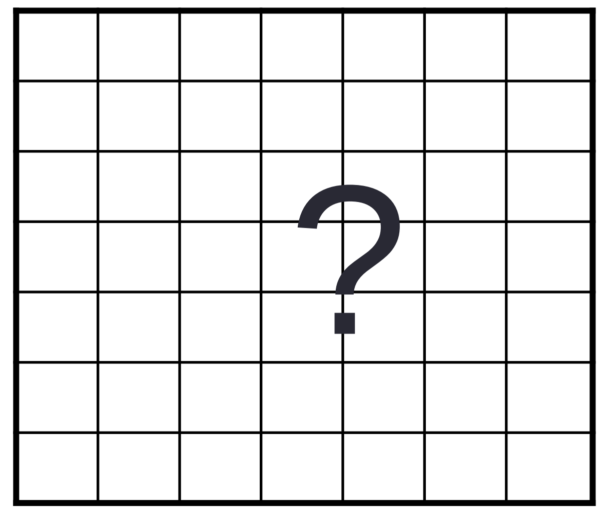
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x, y]$

$\otimes$

a	b	c
d	e	f
g	h	i

$H[u, v]$



$G[x, y]$

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

0	0	0
0	1	0
0	0	0

Original

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

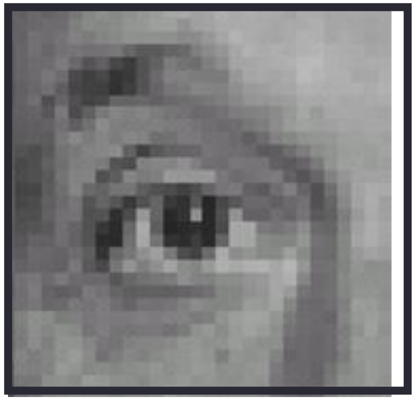
?

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

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

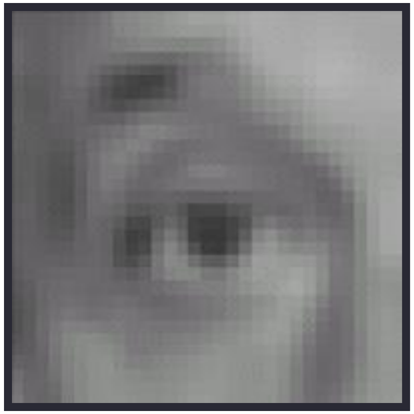
?

Source: D. Lowe

Practice with linear filters

Original

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



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

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

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

A

=

I[x,y]

h[i,j]

f[x,y]

Mask=

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

Matlab:
mask=[[1,1,1,1,1],[1,1,1,1,1],…]
f=conv2(mask,im)

Smoothing with a rectangular filter

I[x,y]

A

h[i,j]

1,1,1,1,1

=

f[x,y]

Smoothing with a rectangular filter

I[x,y]

A

h[i,j]

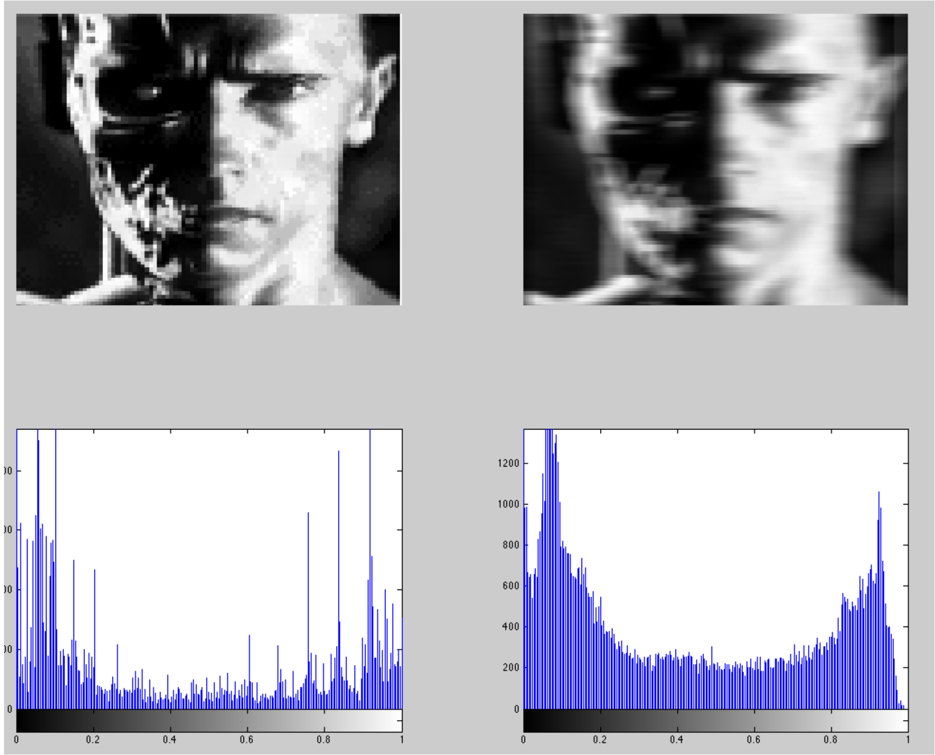
1,
1,
1,
1,
1

=

f[x,y]

Exercise

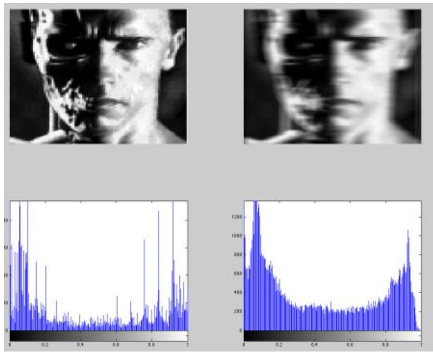
- Apply a smoothing on the Schwarzenegger image with a uniform masks.
- Compare the histograms of the original and resulting image.



Exercise

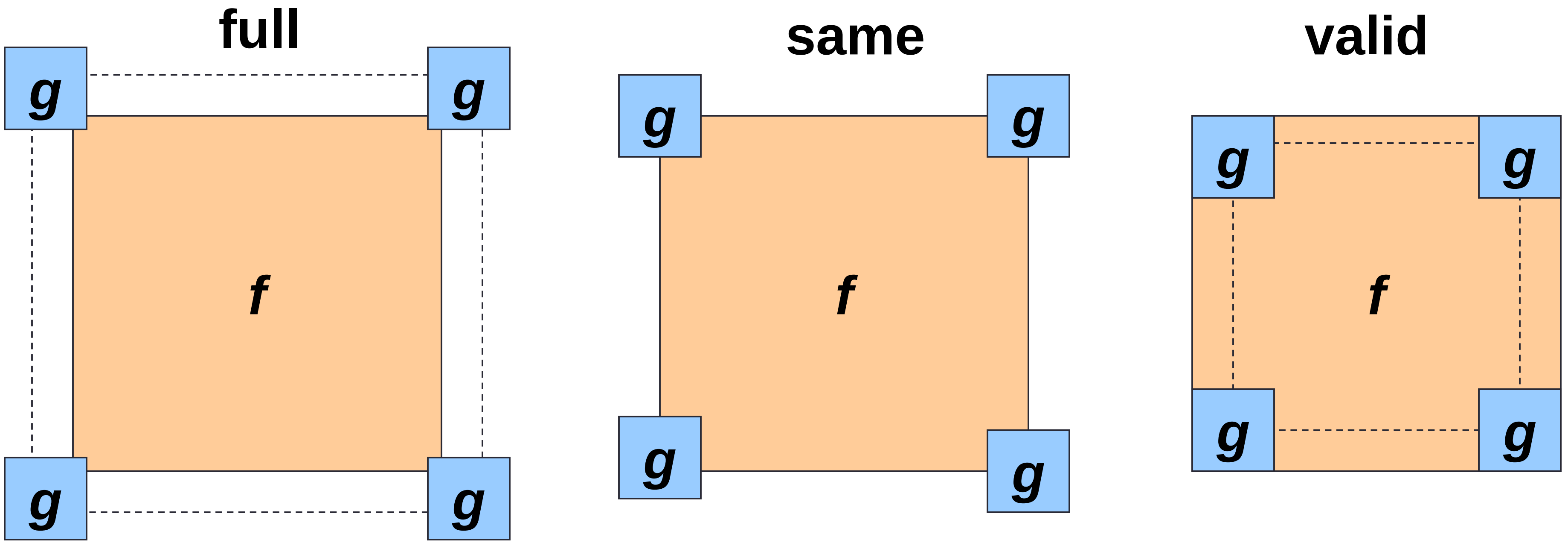
- Apply a smoothing on the Schwarzenegger image with a uniform masks.
- Compare the histograms of the original and resulting image.

```
function [ ] = test_conv( )
% This function illustrates the effect of smoothing with convolutions by a
% uniform mask
            close all;
            im=imread('swarz.png');

            mask=[[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]]/25.0;
            im2=double(im)/255.0;

            f=conv2(mask,im2);

            figure,
            subplot(2,2,1), imshow(im), subplot(2,2,2), imshow(uint8(255*f)),
            subplot(2,2,3), imhist(im2), subplot(2,2,4), imhist(f);

end
```



Boundary issues

- What is the size of the output?
- MATLAB: output size / “shape” options
  - *shape* = ‘full’: output size is sum of sizes of f and g
  - *shape* = ‘same’: output size is same as f
  - *shape* = ‘valid’: output size is difference of sizes of f and g



Source: S. Lazebnik

Boundary issues

- What about near the edge?
  - the filter window falls off the edge
  - need to extrapolate
  - methods:
    - clip filter (black)
    - copy edge
    - reflect across edge

Source: S. Marschner

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods (MATLAB):
    - clip filter (black): imfilter(f, g, 0)
    - copy edge: imfilter(f, g, ‘replicate’)
    - reflect across edge: imfilter(f, g, ‘symmetric’)
    - g -> is a mask (image)

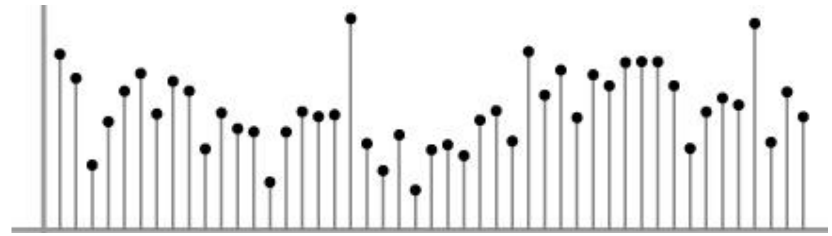
Source: S. Marschner

Today

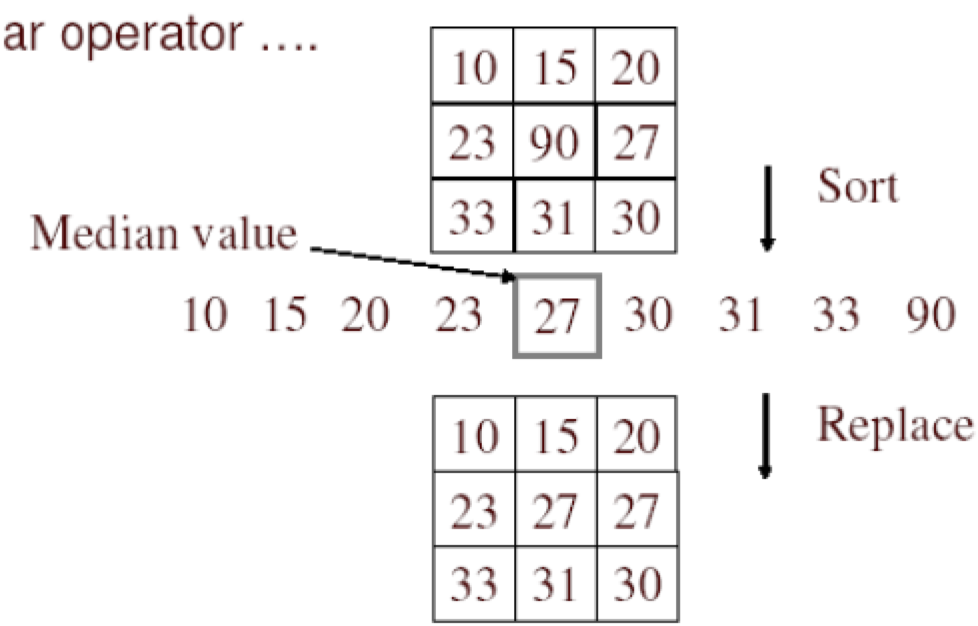
- Image construction
- Spatial and photometric resolution
  - Histogram and image contrast enhancement
- Linear filters – mean filter
- Convolution / correlation
- Smoothing
- Median filter
- Linear filters with Gaussians

Median filter

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



- No new pixel values introduced

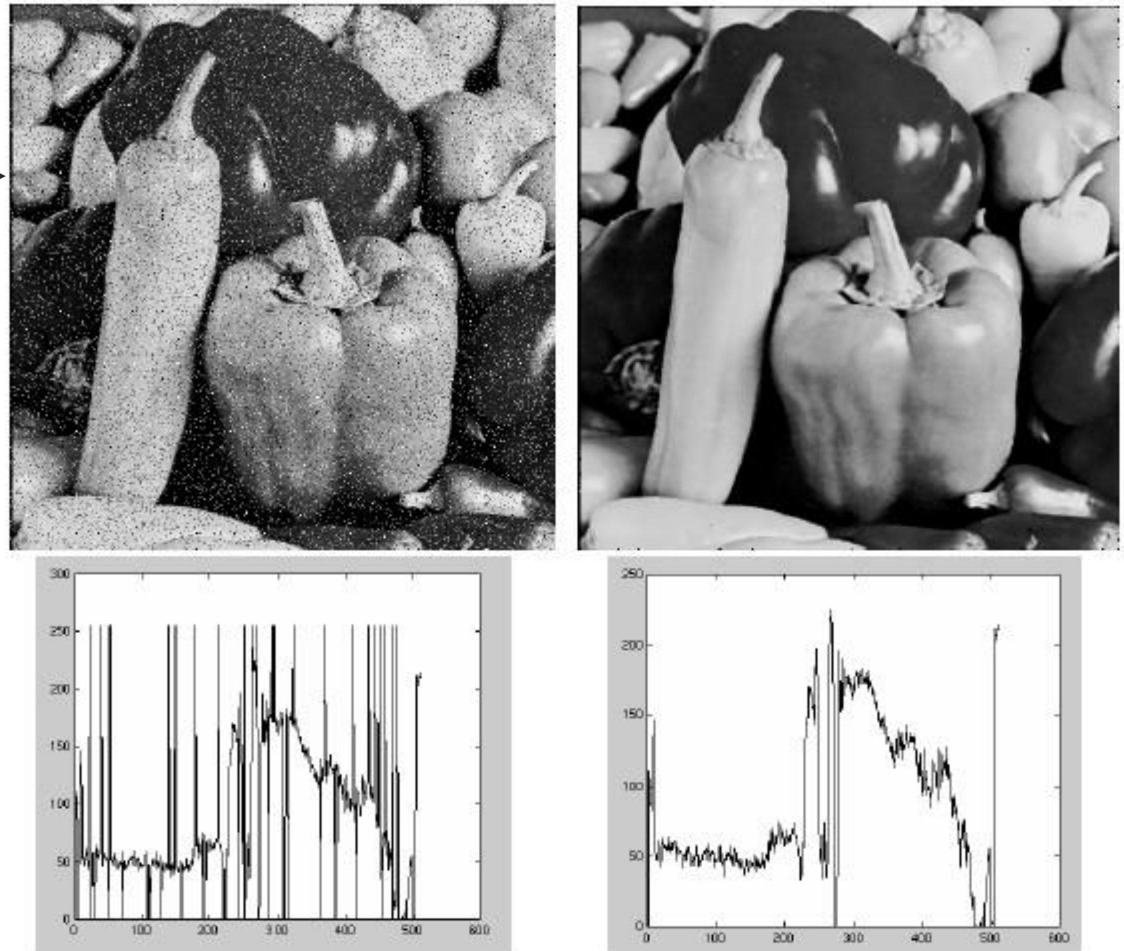


- Removes spikes: good for impulse, salt & pepper noise

•Non-linear filter (it can be proved)

Median filter

Salt and pepper noise



Median filtered

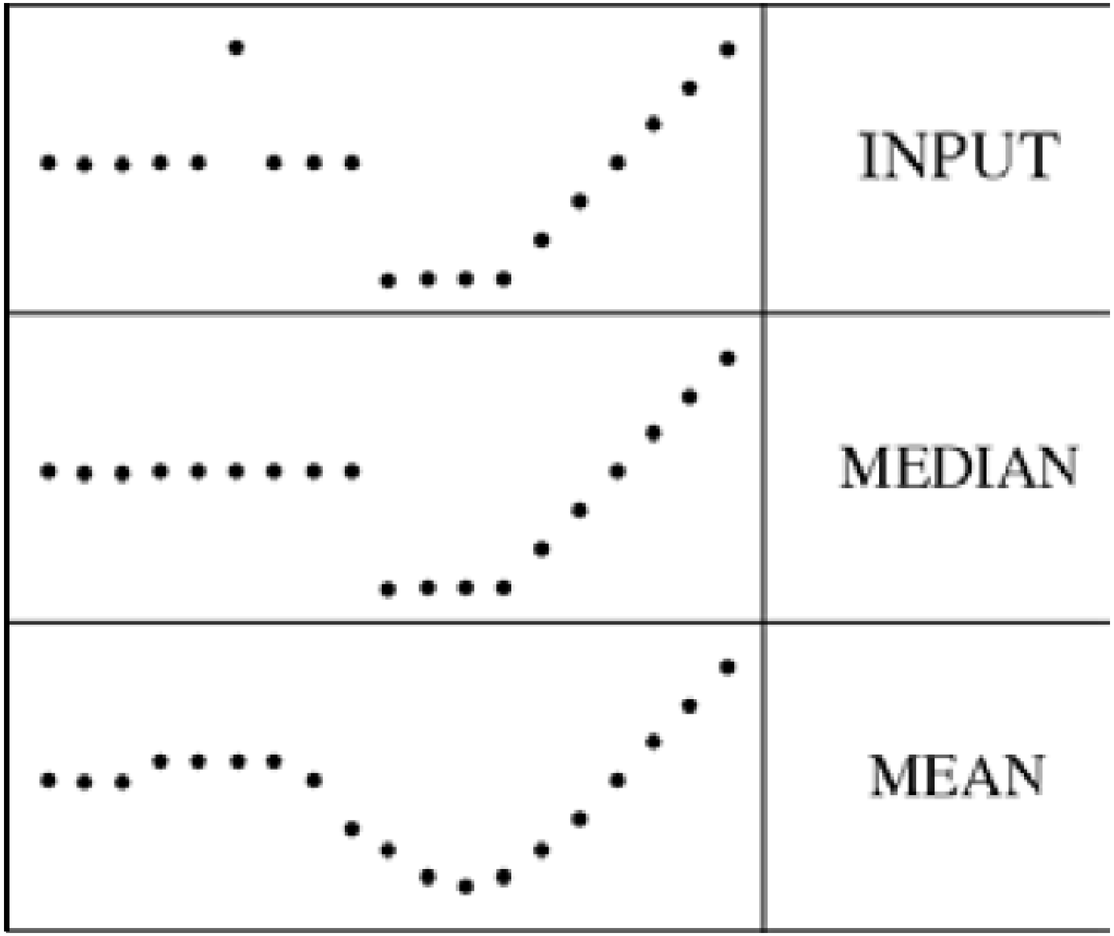
Plots of a row of the image

Matlab: output im = medfilt2(im, [h w]);

Source: M. Hebert

Median filter

- Median filter is edge preserving



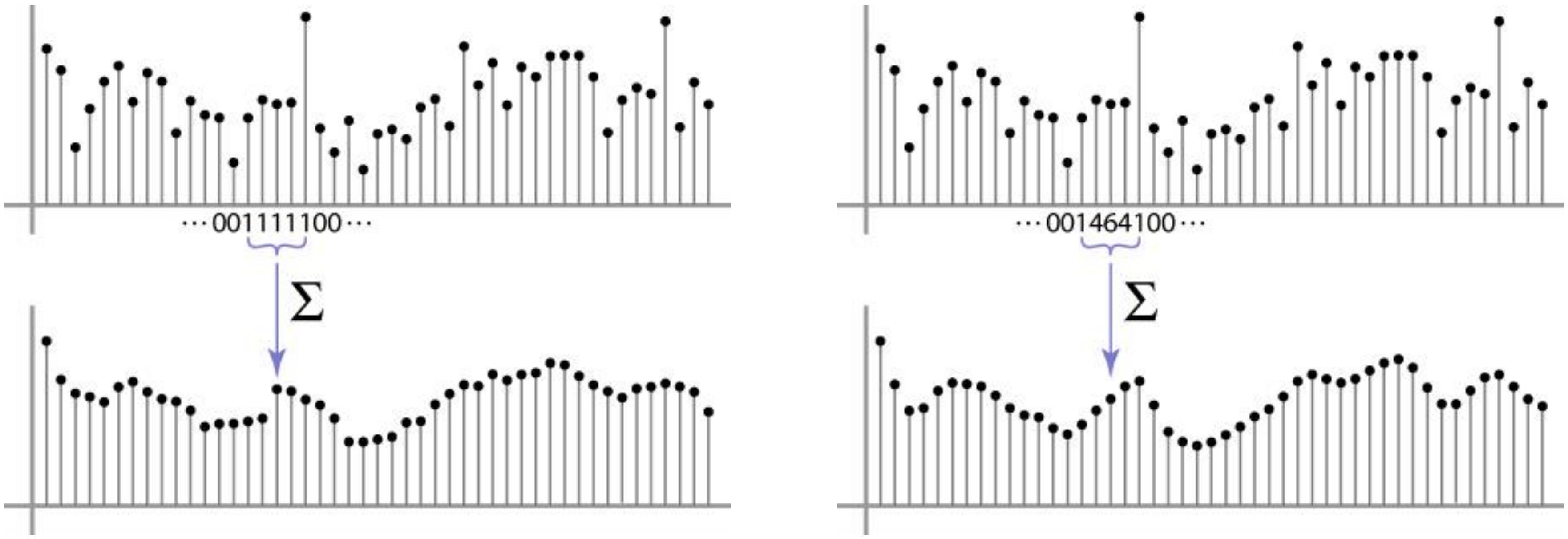
What would be the result of a mean filter?

Today

- Image construction
- Spatial and photometric resolution
  - Histogram and image contrast enhancement
- Linear filters
  - Examples: smoothing filters
- Convolution / correlation
- Smoothing
- 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

$F[x, y]$

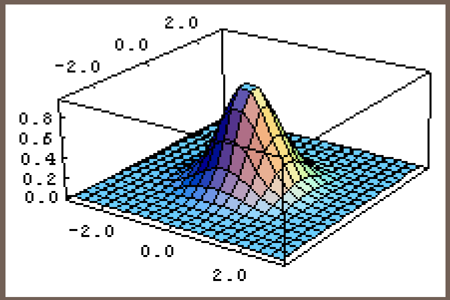
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

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

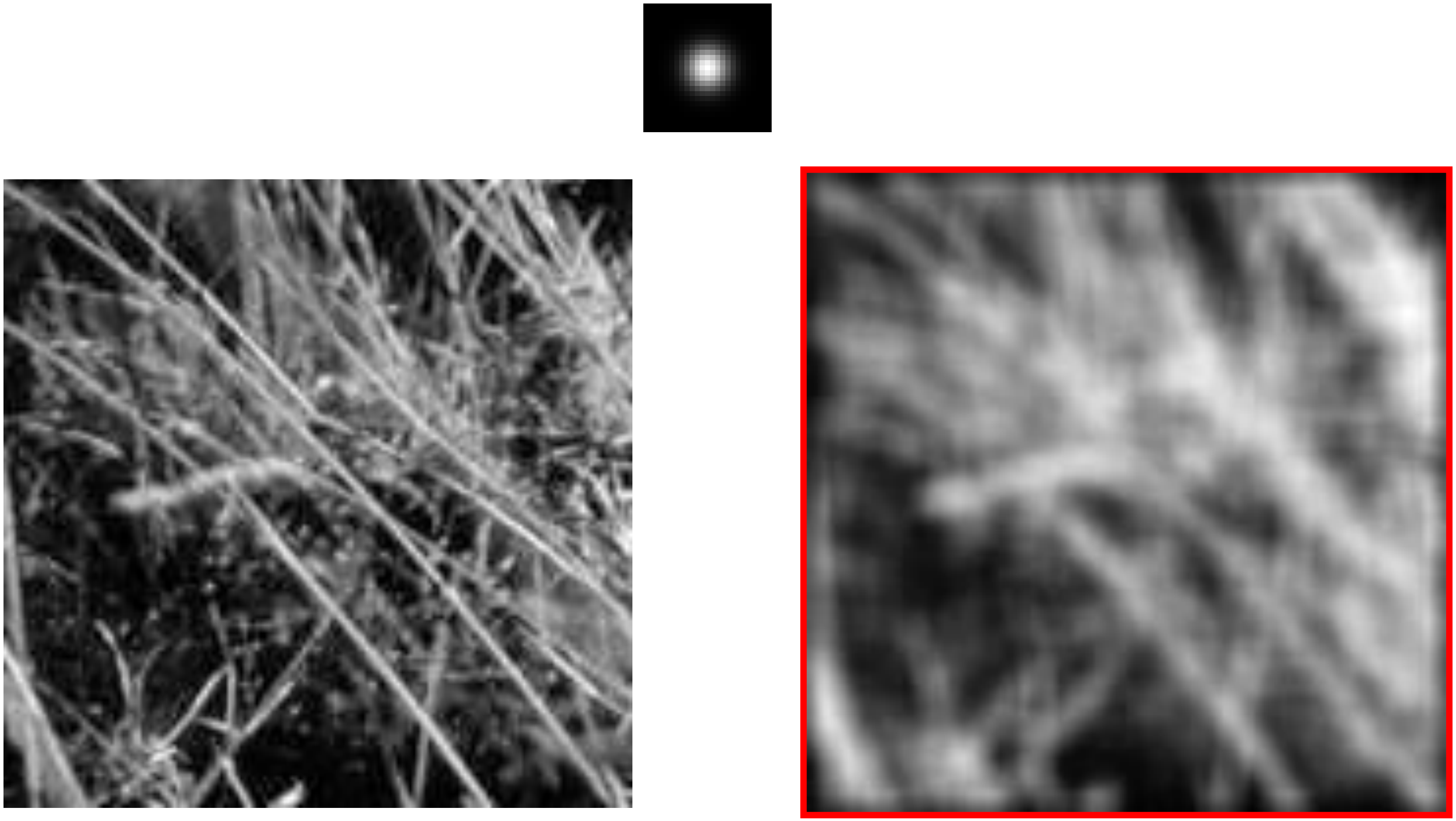
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



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

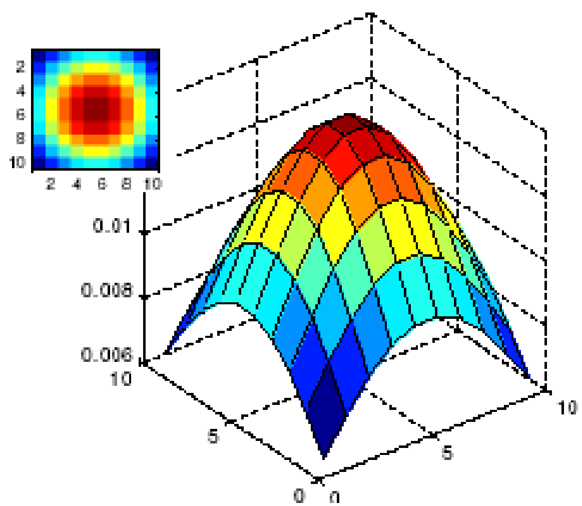
Source: S. Seitz

Smoothing with a Gaussian

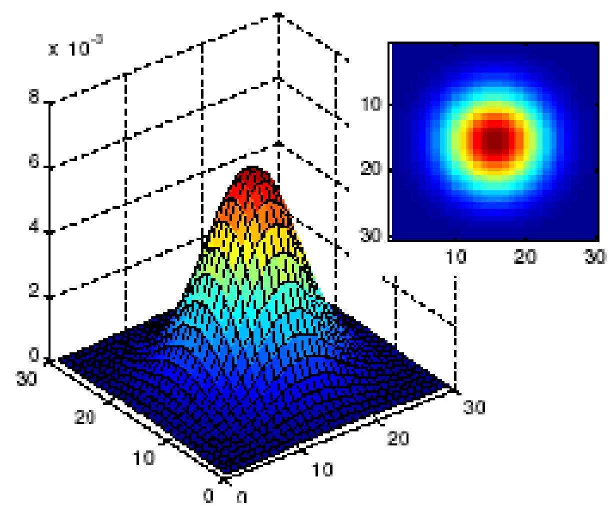


Gaussian filters

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



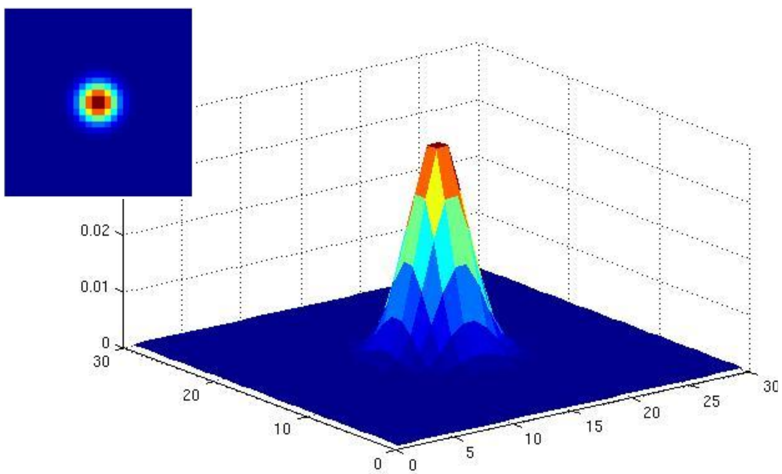
σ = 5 with 10 x 10 kernel



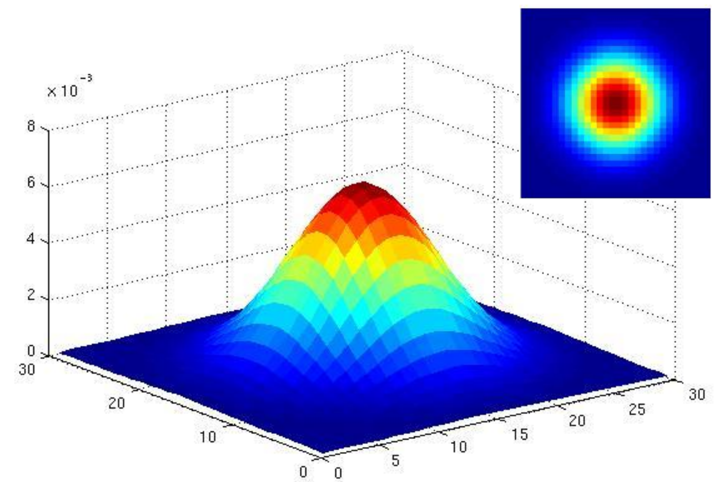
σ = 5 with 30 x 30 kernel

Gaussian filters

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



σ = 2 with 30 x 30 kernel

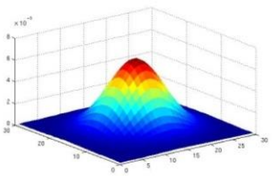


σ = 5 with 30 x 30 kernel

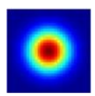
Matlab

```
>> hsize = 10;
>> sigma = 5;
>> h = fspecial(‘gaussian’ hsize, sigma);
```

```
>> mesh(h);
```



```
>> imagesc(h);
```

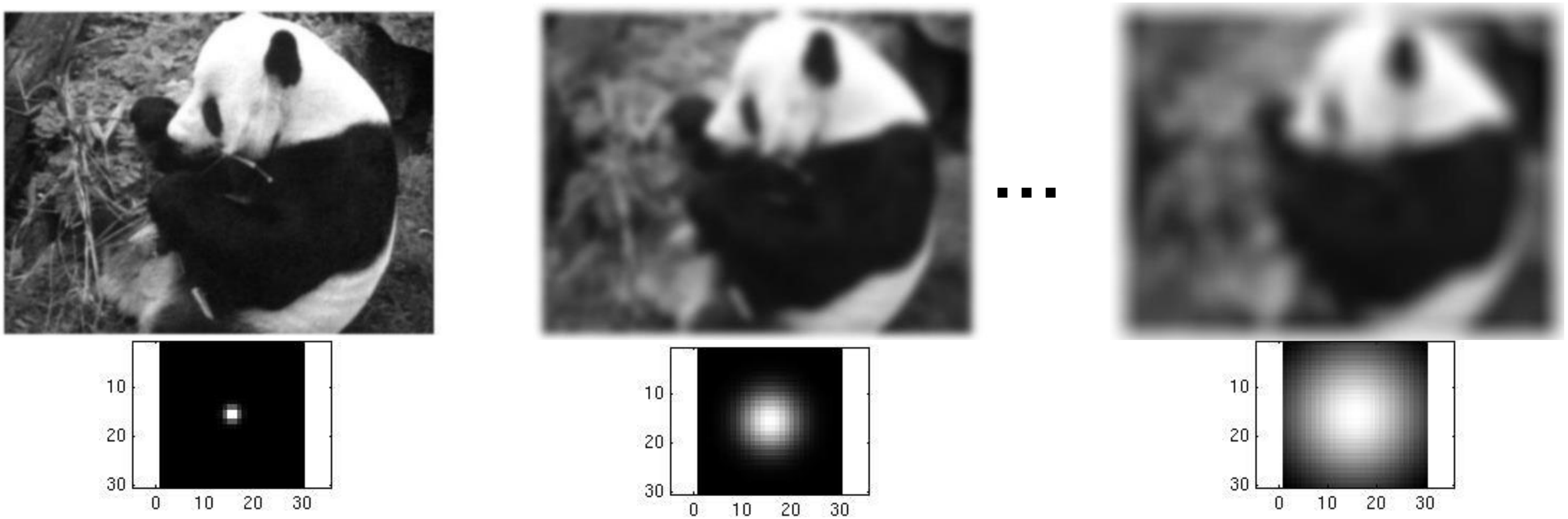


```
>> outim = imfilter(im, h); % convolution
>> imshow(outim);
```

outim

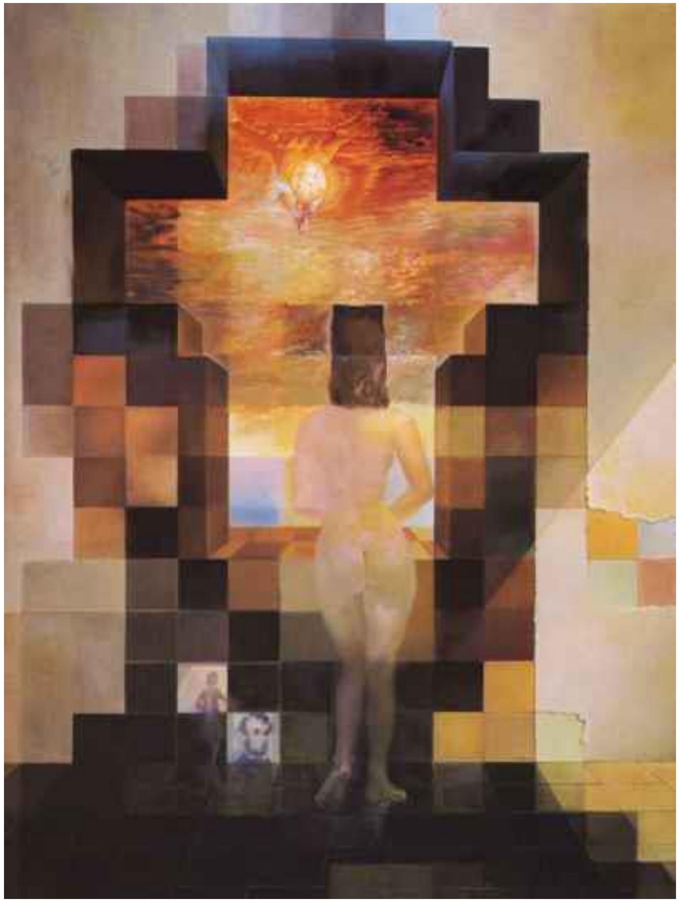
Smoothing with a Gaussian filter

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

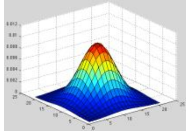



```
for sigma=1:3:10
  h = fspecial('gaussian‘, fsize, sigma);
  out = imfilter(im, h);
  imshow(out);
  pause;
end
```

Local vs global analysis



Dali



Properties of smoothing filters

- <u>Smoothing</u>
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

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