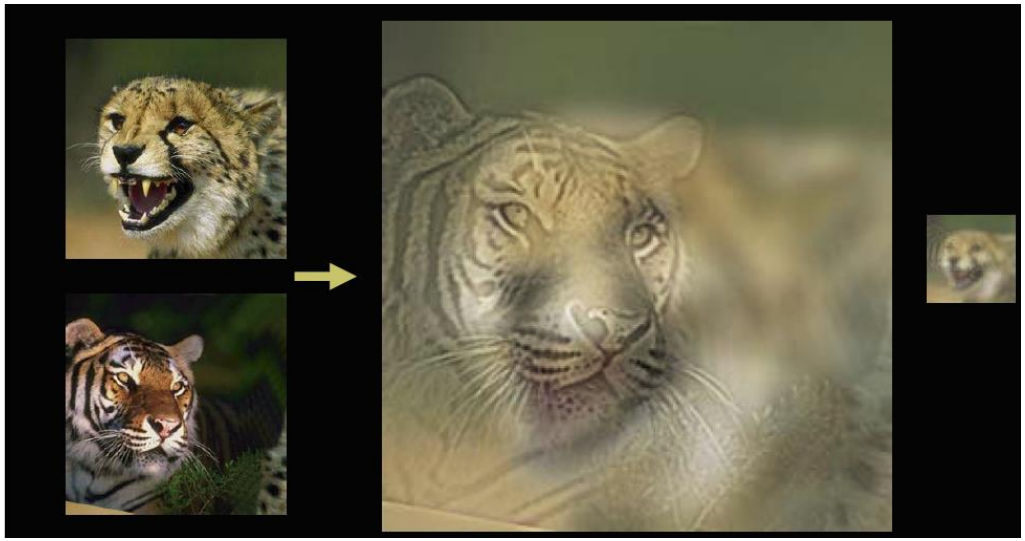


Computer Vision: Local transformations

Nicolás Pérez de la Blanca,
nicolas@decsai.ugr.es

Plan for today

- Image noise
- Local transformations
  - Examples: smoothing filters
- Convolution / correlation
- Edge detection

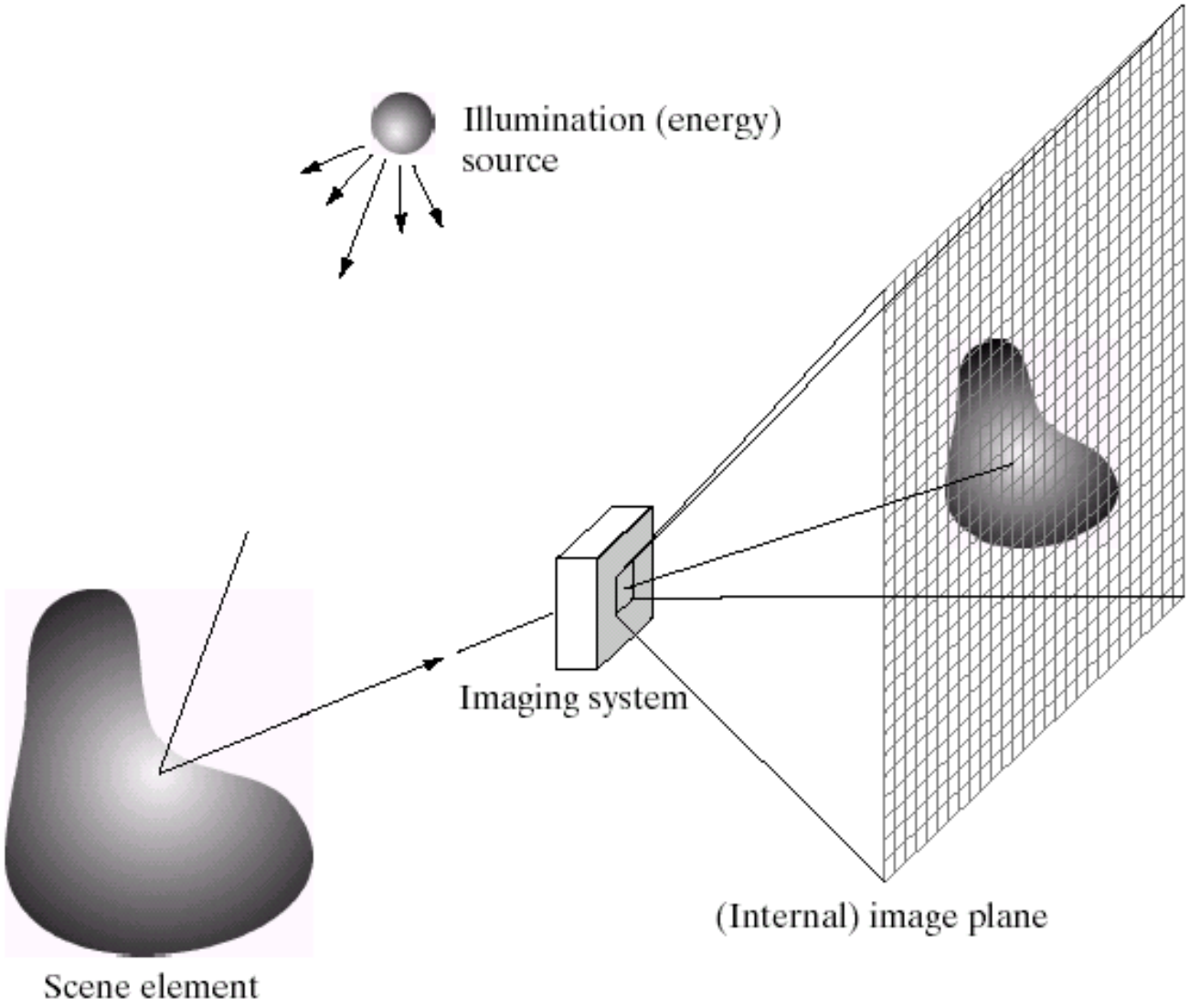
Readings for today

- ***Image Filtering***
  - Readings: Szel., Ch. 3.1-3.3
- ***Edge detection***
  - Readings: Szel., Ch. 4.2

What is an image?

Very complex data encoding multiples cues

Image Formation



Slide credit: Derek Hoiem

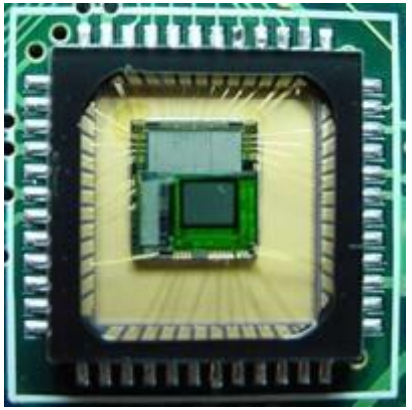
Digital camera

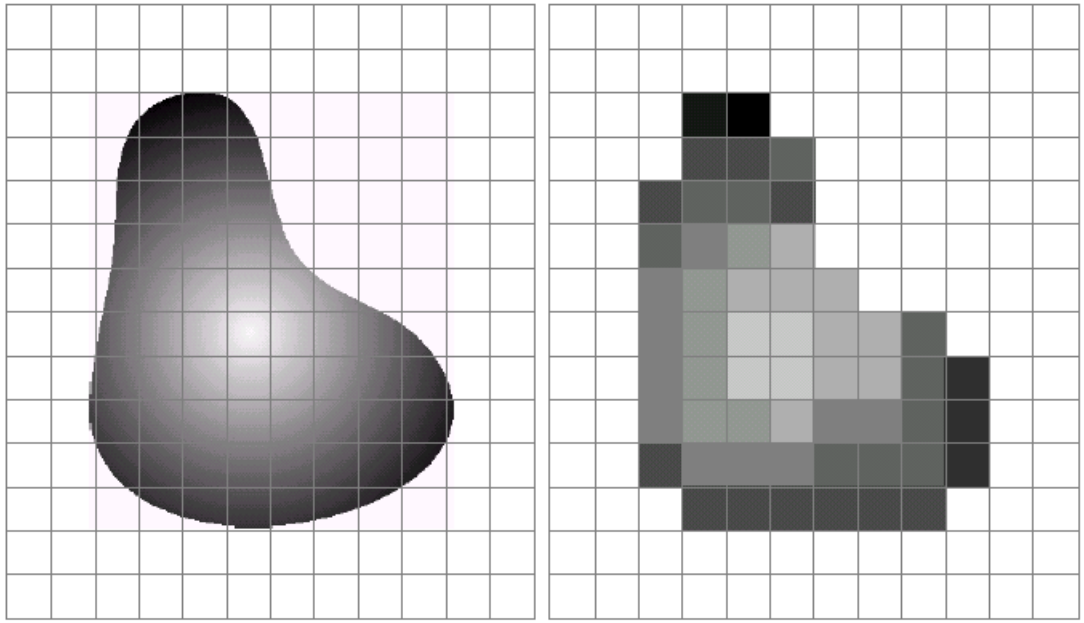
A digital camera replaces film with a sensor array

- Each cell in the array is light-sensitive diode that converts photons to electrons
- http://electronics.howstuffworks.com/digital-camera.htm

Slide by Steve Seitz

Digital images





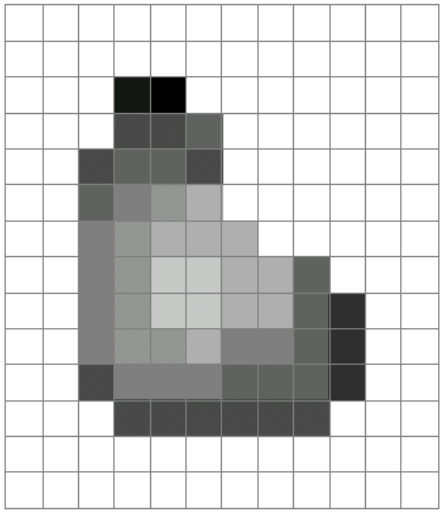
a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Slide credit: Derek Hoiem

What is an image?

- A grid (matrix) of intensity values



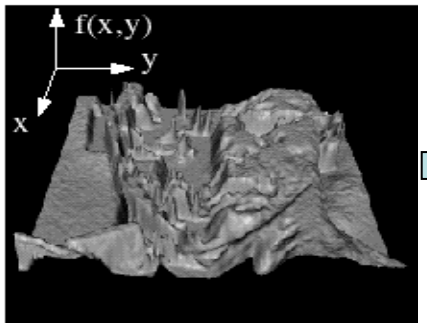
=

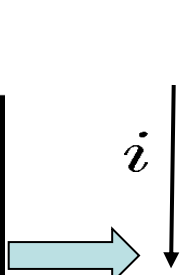
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

Digital images

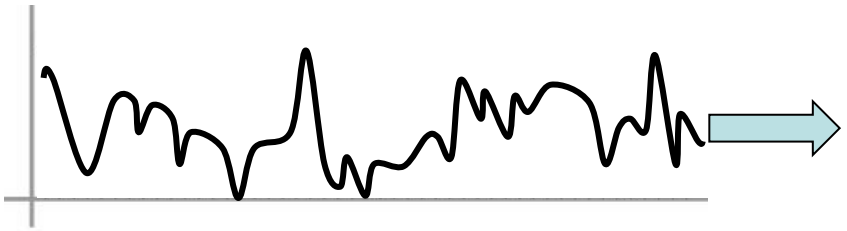
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

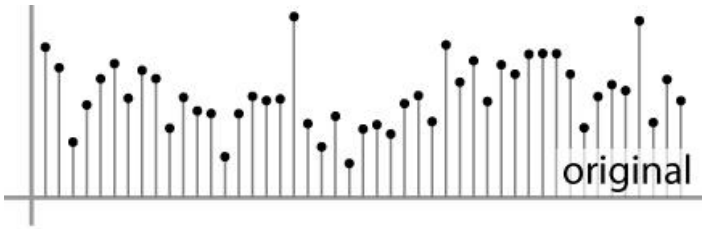




62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

2D





1D

Adapted from S. Seitz

Digital color images

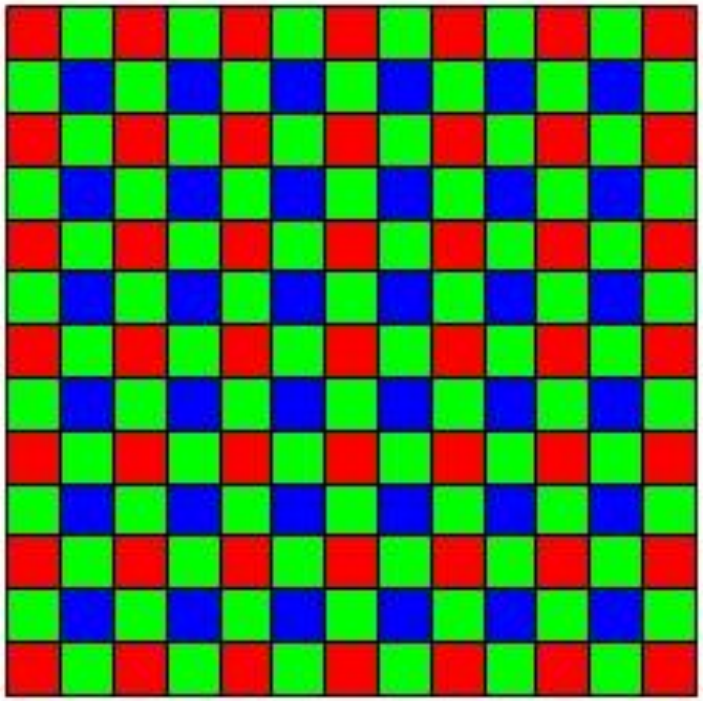
Color images, RGB color space

R

G

B

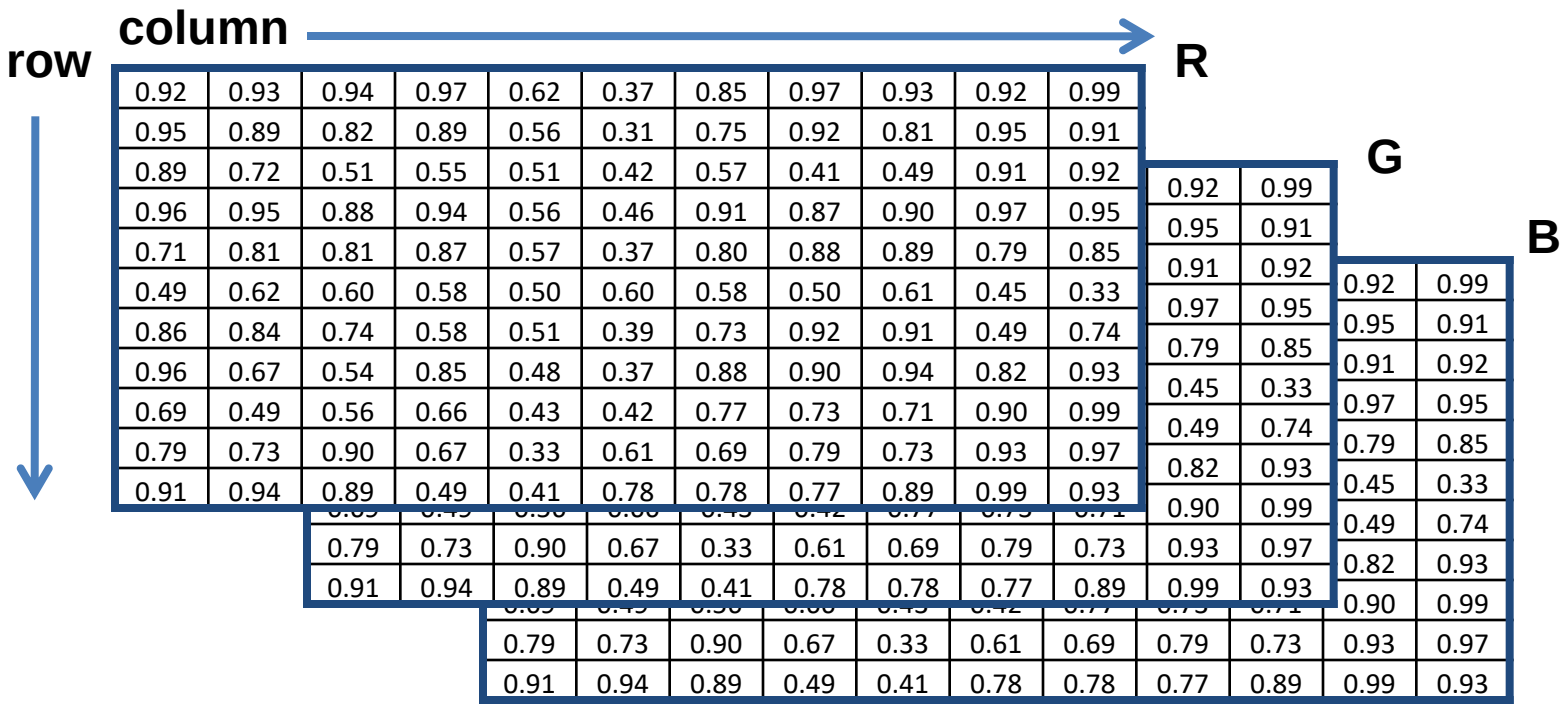
Digital color images



Bayer filter

Images in OpenCV

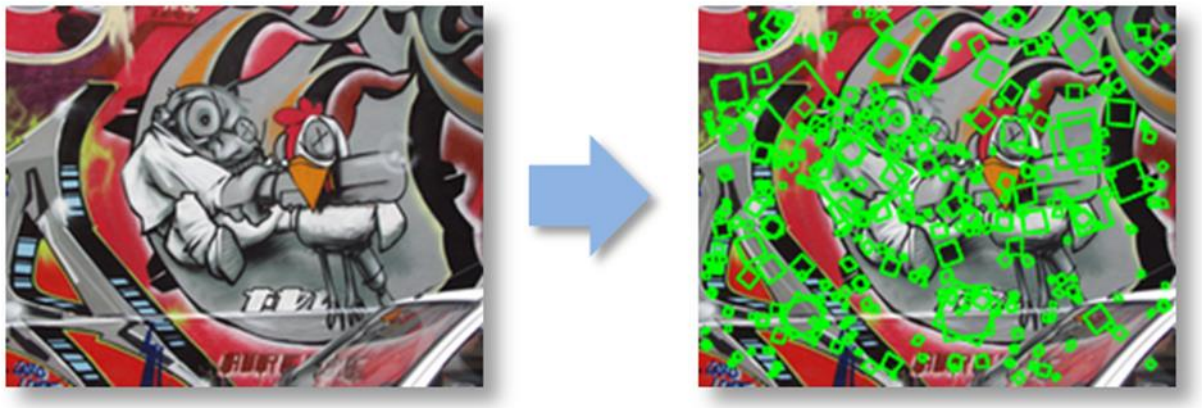
- Image represented as a matrix im
- im=imread(filename) returns a CV_8UC3 image (values 0 to 255)
  - Convert to float format with im.convertTo()



Slide credit: Derek Hoiem

What is the most relevant information cue ?

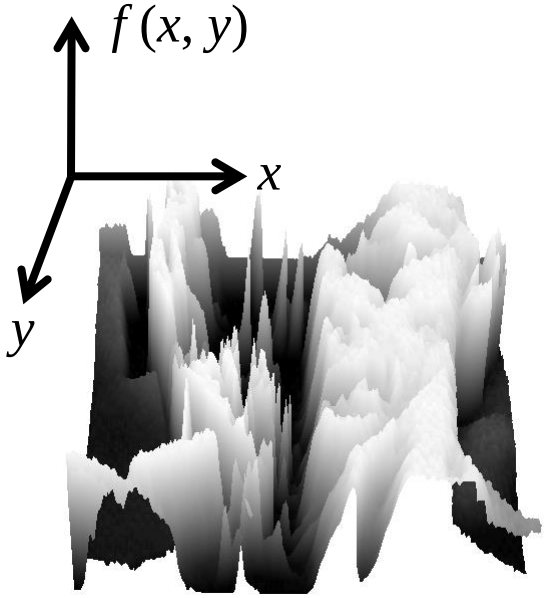
- Pixel and nearby regions show dependence between their values
  - No rule for distant pixels/regions (high spatial variability)
- Main approach: To characterize images from local informations.



The image as a function

- We can think of a (grayscale) image as a **function**, f, from R^2 to R:
  - $f(x,y)$ gives the **intensity** at position (x,y)

snoop



3D view

  - A **digital** image is a discrete (**sampled**, **quantized**) version of this function

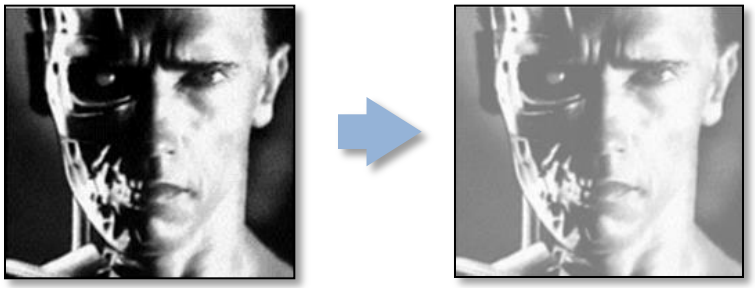
Basic Image Transformations

- In order to extract useful information to approach our goals we apply transformations on the image values

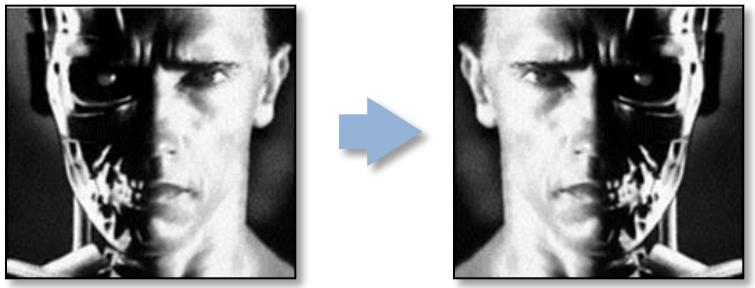
- Local : Image filtering using local masks
- Local/Global: Geometric transformations

Examples of transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

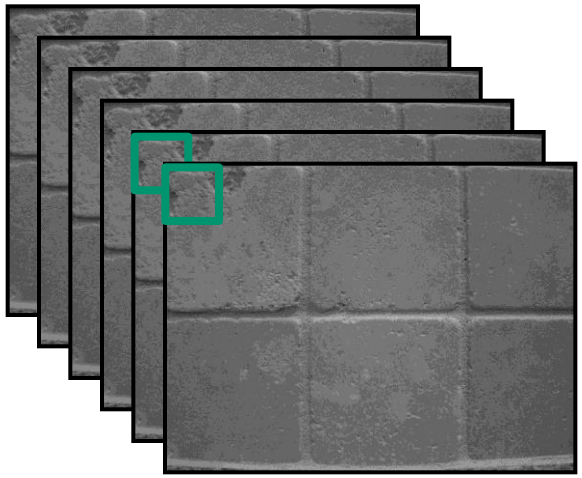
- We'll talk about a special kind of operator, *convolution* (Local transformation)

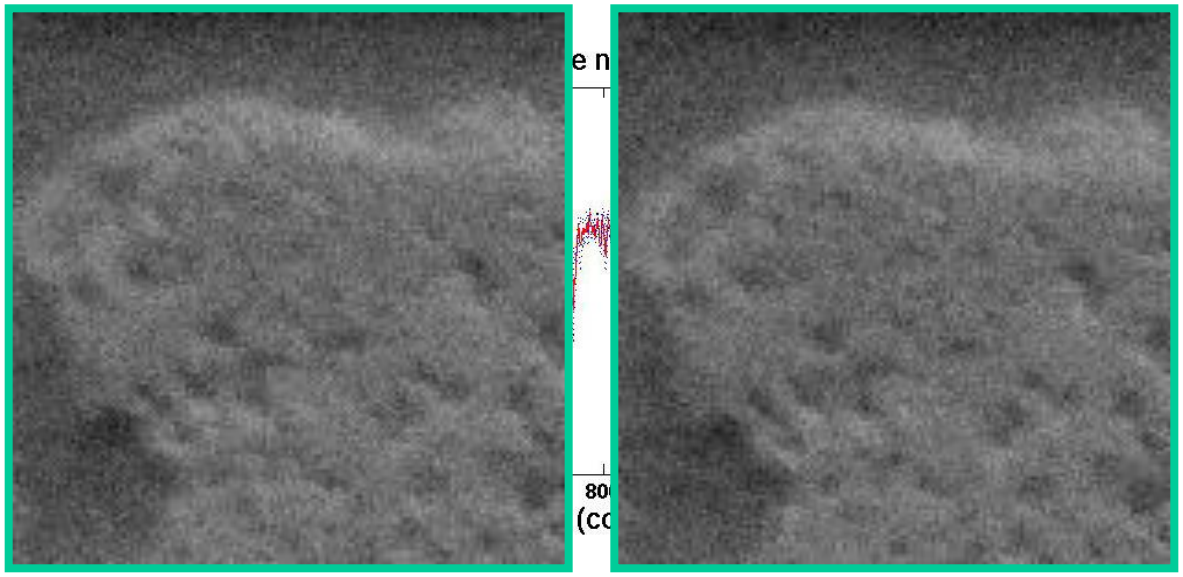
Image filtering

- 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.
  - We extract only a part of the image information
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Adapted from Derek Hoiem

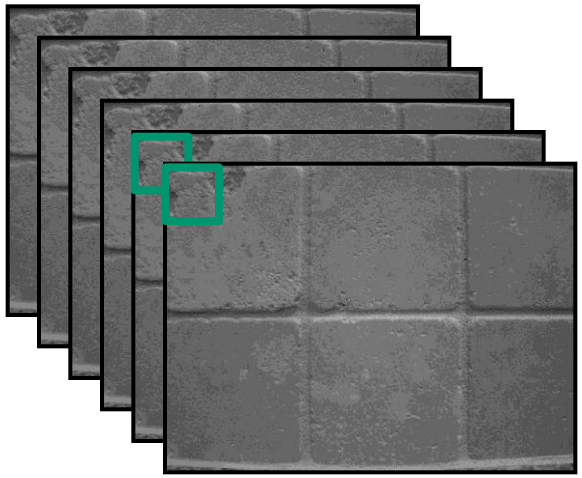
Simple case: noise reduction

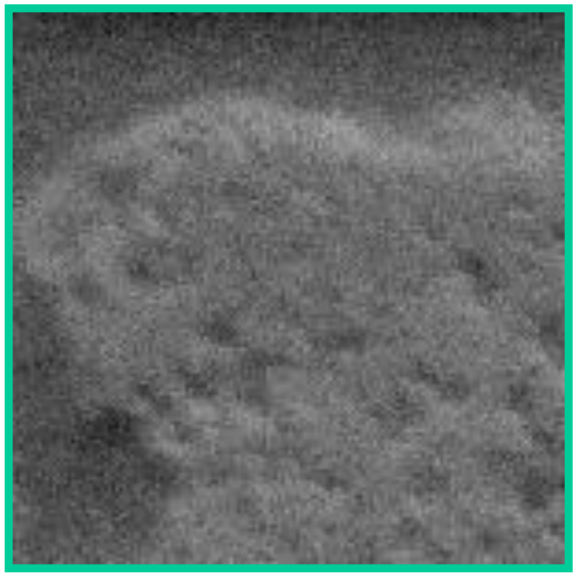


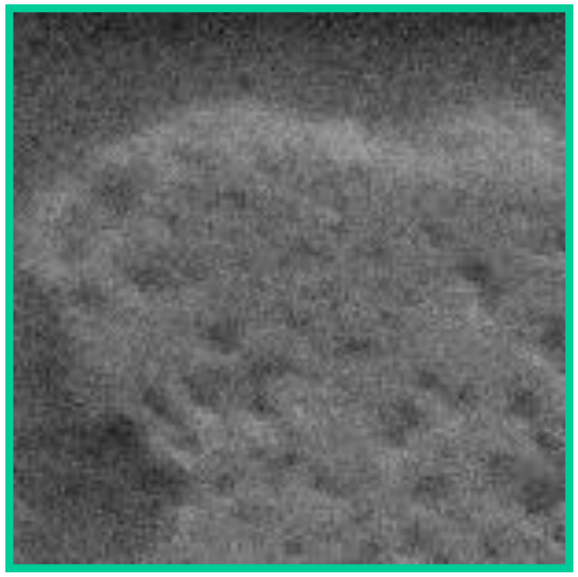


- Even multiple images of the **same static scene** will not be identical.
- Different types of noise need different transformations

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

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

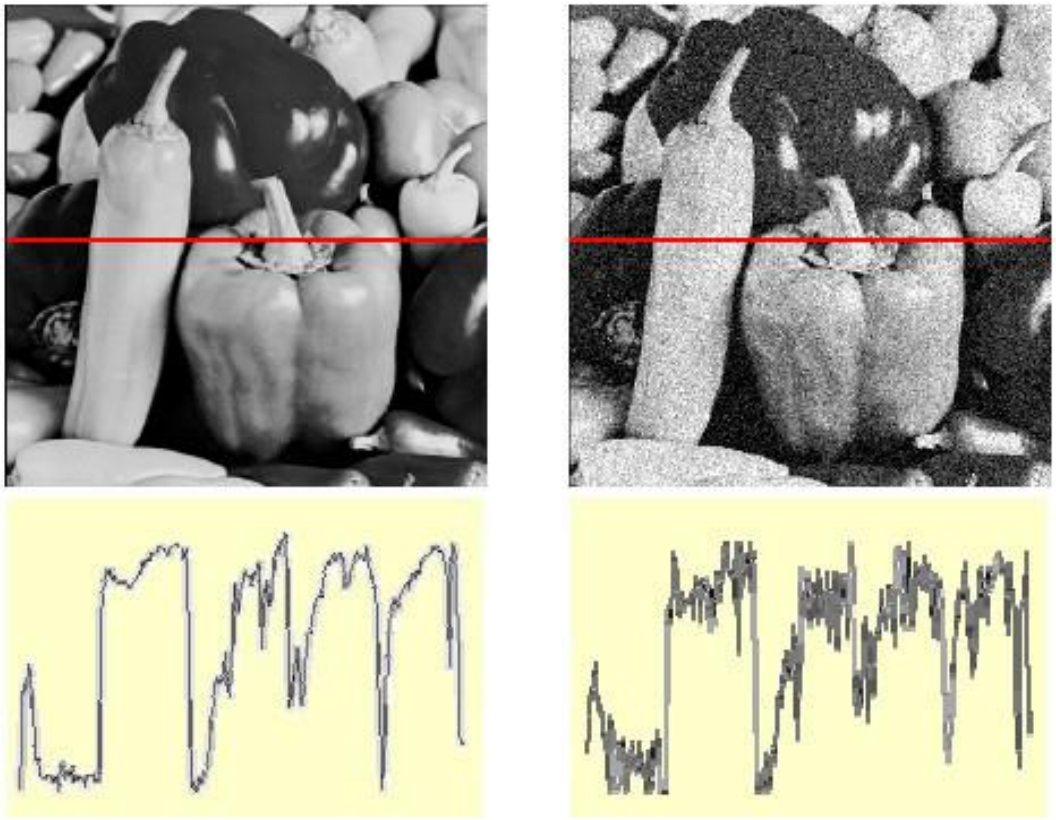
Salt and pepper noise

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:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

```
>> randn(noise,0,sigma);
>> output = im + noise;
```

What is impact of the sigma?

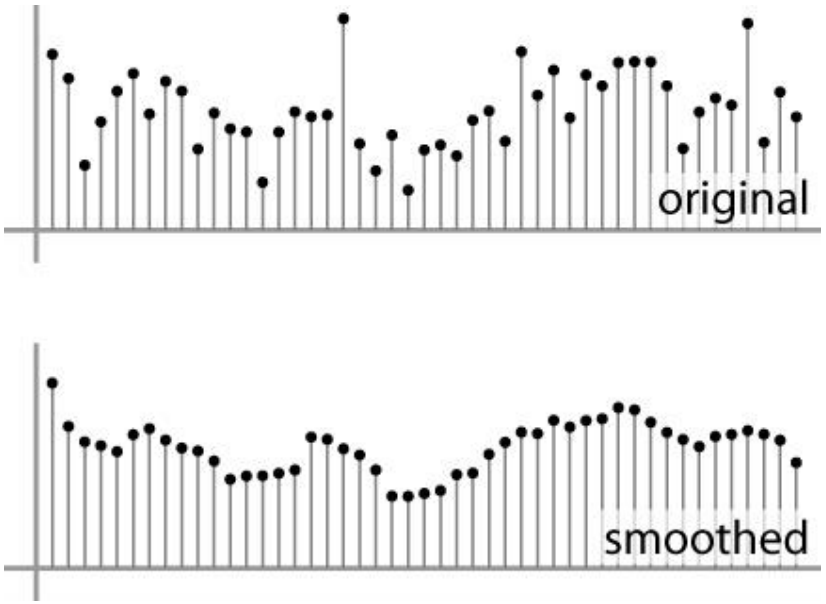
Fig: M. Hebert

First attempt at a solution

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

First attempt at a solution

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

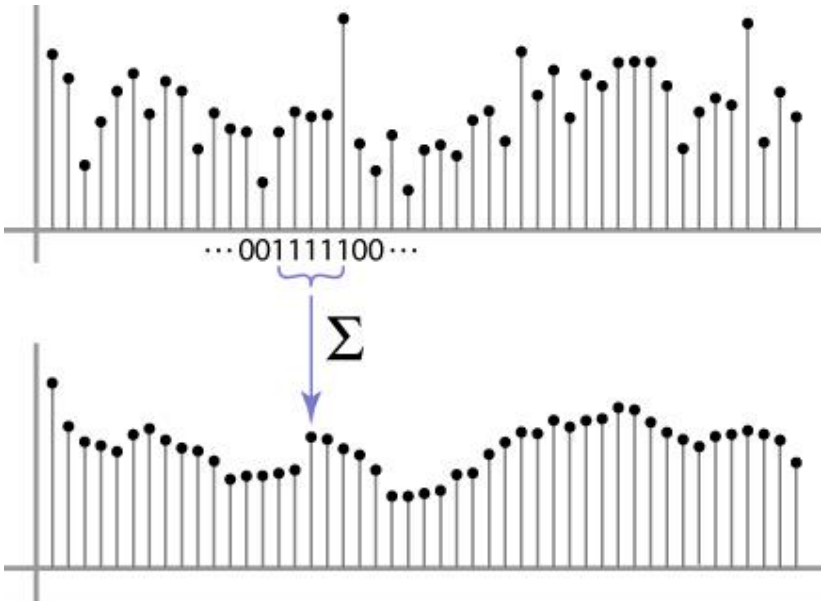


Source: S. Marschner

Weighted Moving Average

Can add weights to our moving average

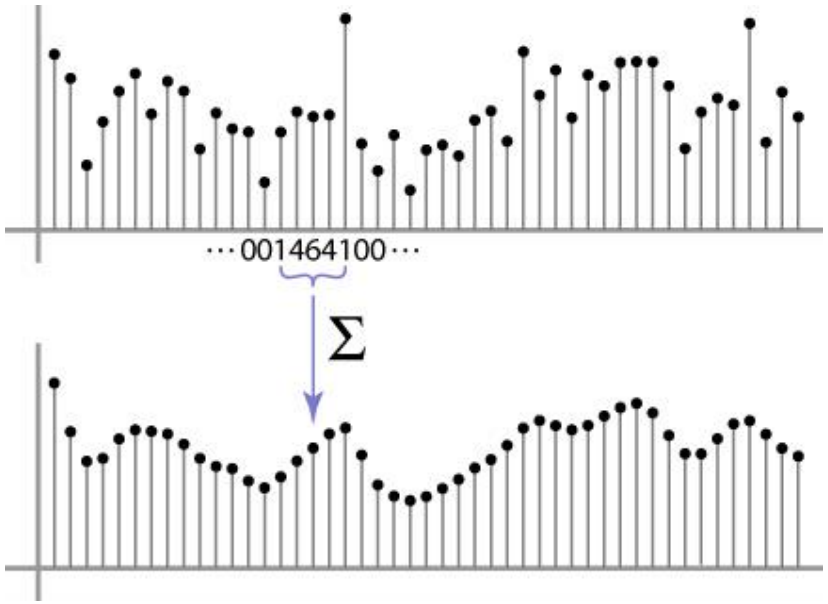
Weights [1, 1, 1, 1, 1] / 5



Source: S. Marschner

Weighted Moving Average

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



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

Correlation mask

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

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

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

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

Correlation mask

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

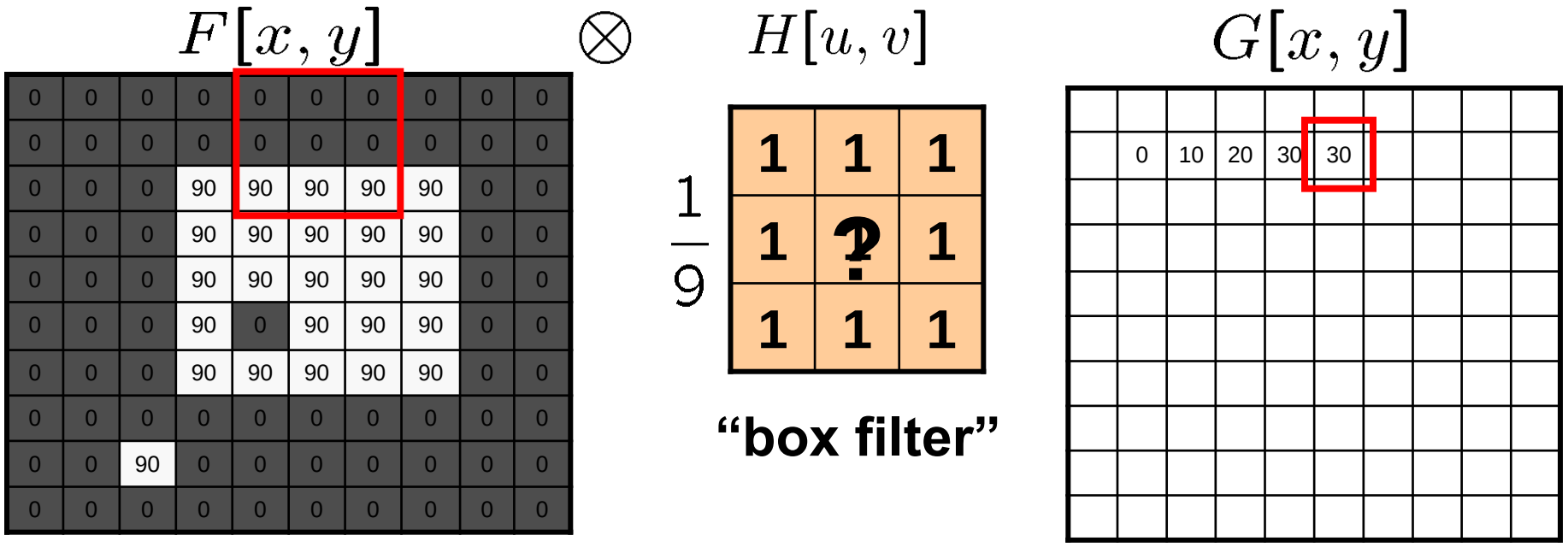
This is called **cross-correlation**, denoted $G = H \otimes F$

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.

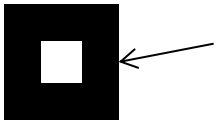
Averaging mask

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



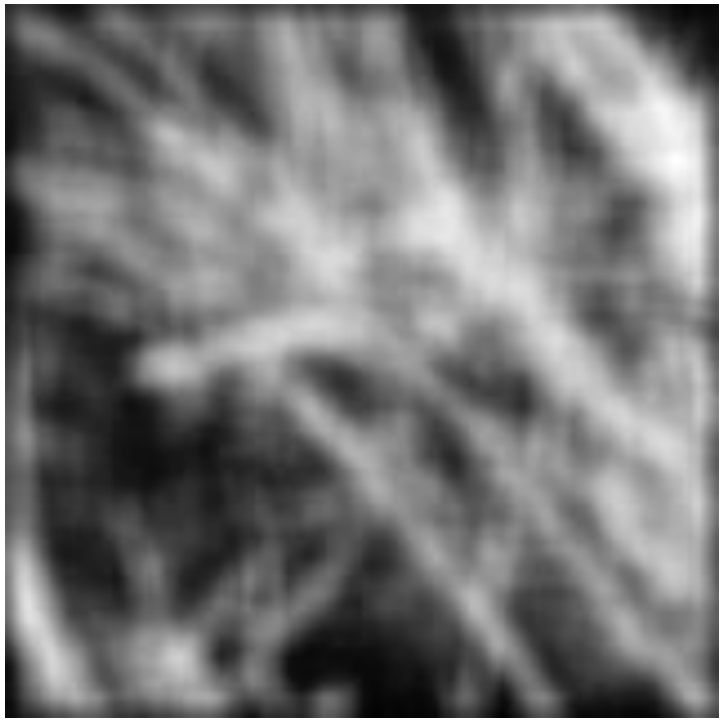
$$G = H \otimes F$$

Smoothing by averaging



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

original



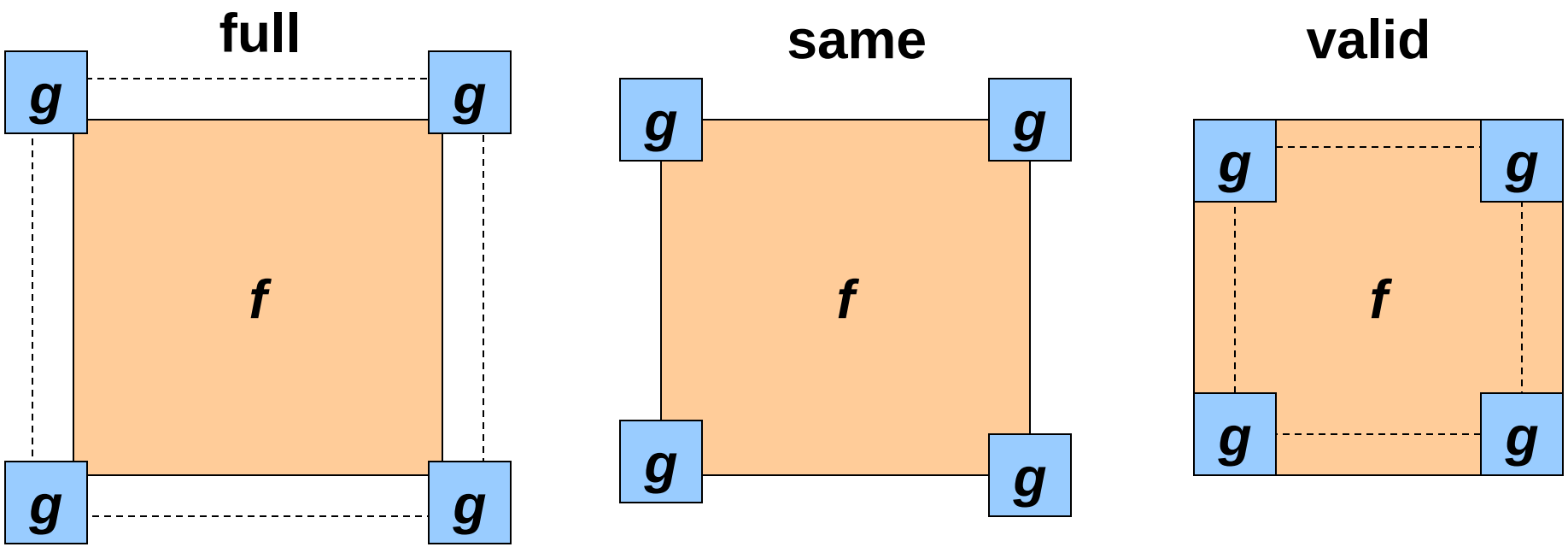
filtered

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

Boundary issues

What is the size of the output?

- 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 of the image
- need to extrapolate
- methods:
  - clip filter (black)
  - wrap around
  - 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
- Types (OpenCV):

  * *BORDER_REPLICATE: aaaaaa* | *abcdefgh* | *hhhhhhh*
  * *BORDER_REFLECT: fedcba* | *abcdefgh* | *hgfedcb*
  * *BORDER_REFLECT_101: gfedcb* | *abcdefgh*| *gfedcba*
  * *BORDER_WRAP: cdefgh* | *abcdefgh* | *abcdefg*
  * *BORDER_CONSTANT: iiiiii* | *abcdefgh* | *iiiiiii*

Gaussian mask

- 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

$F[x, y]$

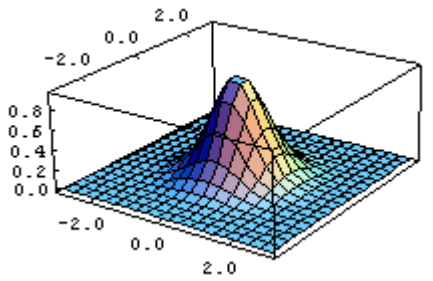
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$H[u, v]$

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

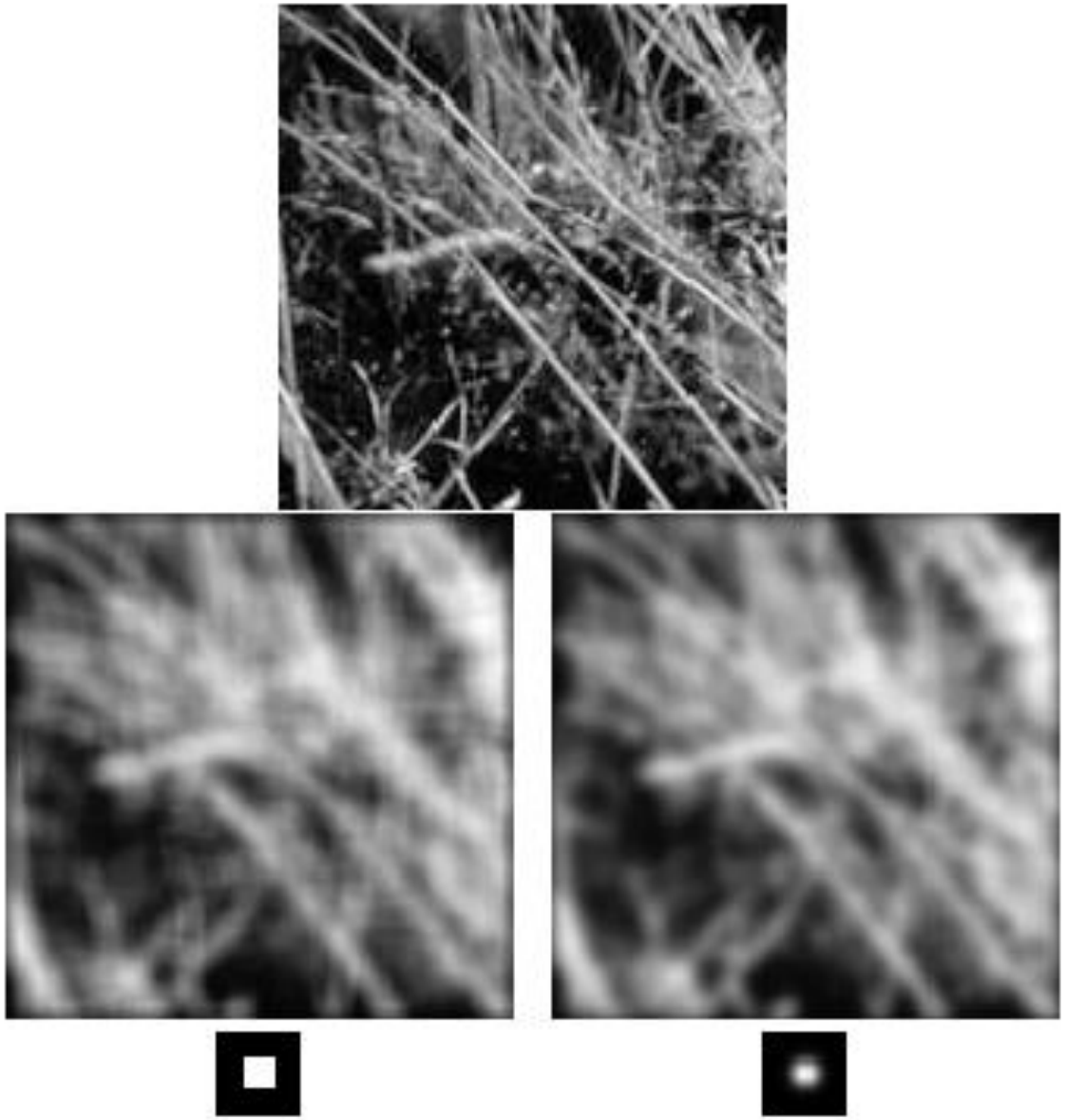
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



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

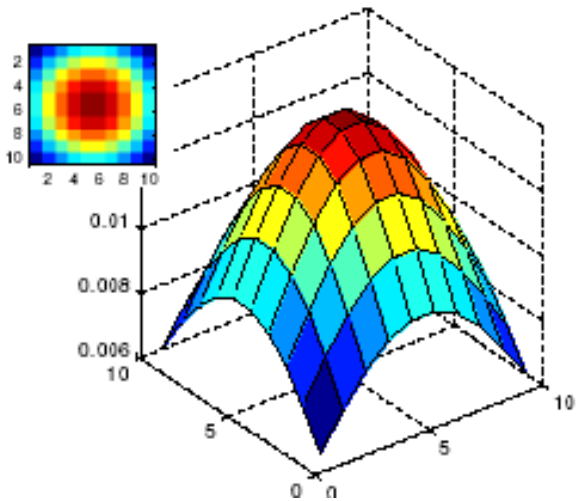
Source: S. Seitz

Gaussian vs. box mask

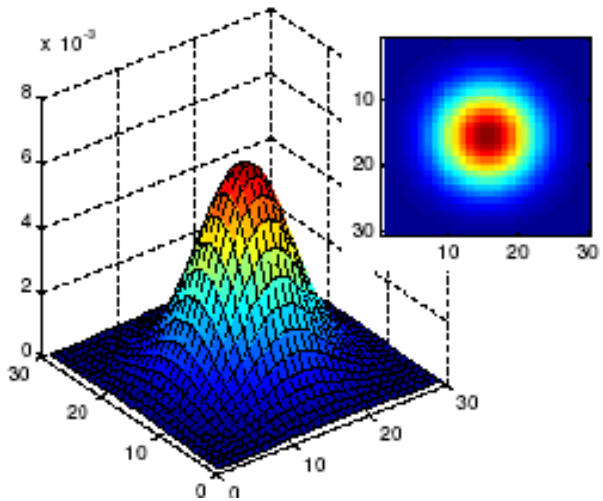


Gaussian masks

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



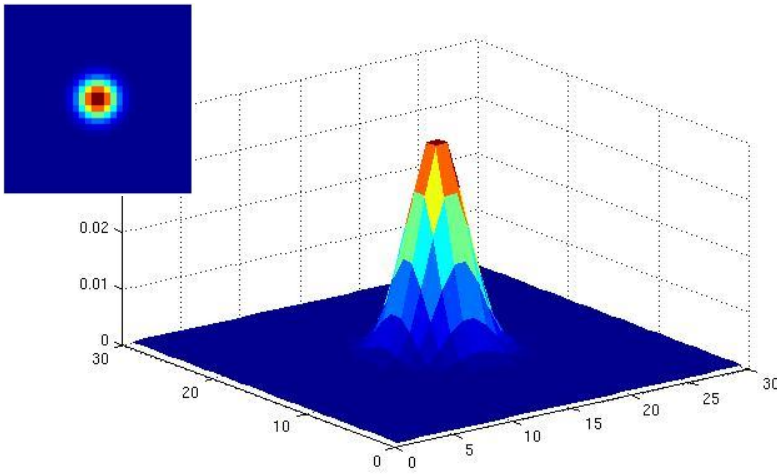
σ = 5 with 10 x 10 kernel



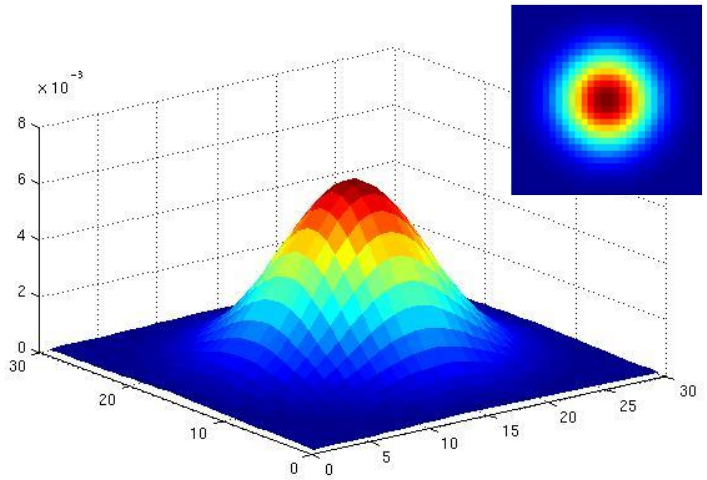
σ = 5 with 30 x 30 kernel

Gaussian masks

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



σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

OpenCV

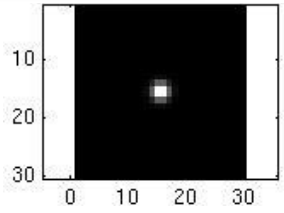
```
>> hsize = 10;
>> sigma = 5;
>> cv2.getGaussianKernel(hsize, sigma)
>> cv2.imshow(kernel);
```

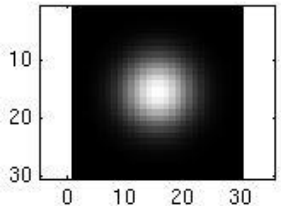
```
>> cv2.filter2D(im,outim,kernel); #correlation
>> cv2.imshow(outim);
```

outim

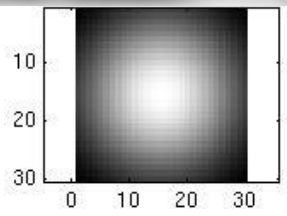
Smoothing with a Gaussian

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





...




```
for sigma in np.arange(1, 10, 3)
        cv2.getGaussianKernel(hsize, sigma)
        cv2.filter2D(im,outim,kernel); #correlation
        cv2.imshow(outim);
        cv2.waitkey()
```

Properties of smoothing masks

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - **Remove “high-frequency” components; “low-pass” filter**

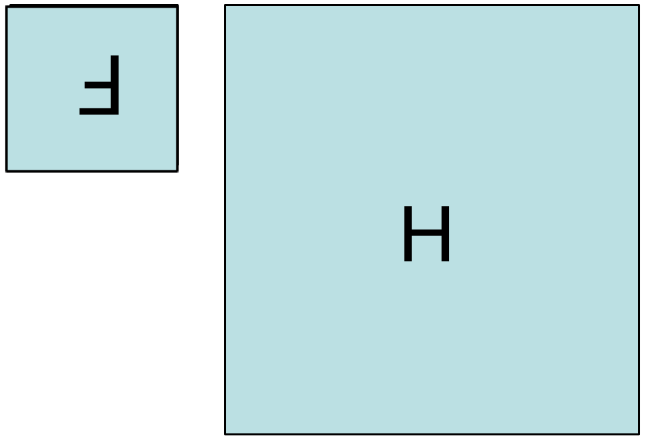
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

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

$$G = H \star F$$

Notation for convolution operator



Convolution vs. correlation

Convolution

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

$$G = H \star F$$

Cross-correlation

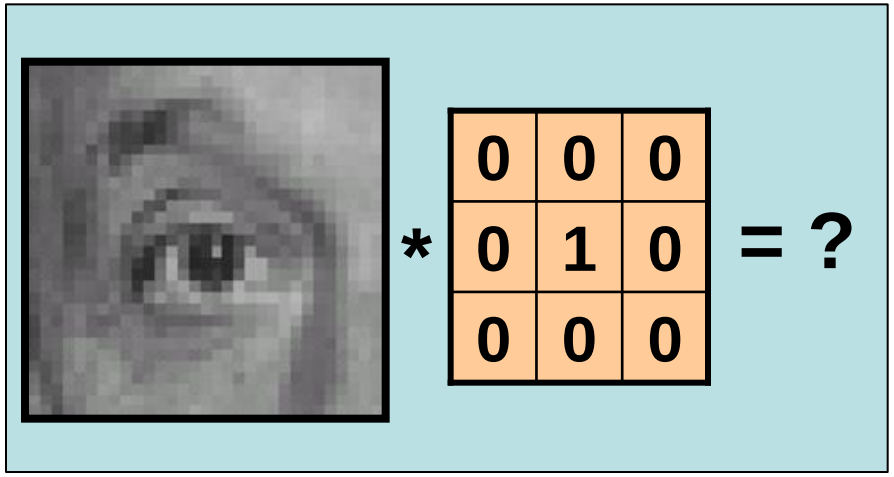
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

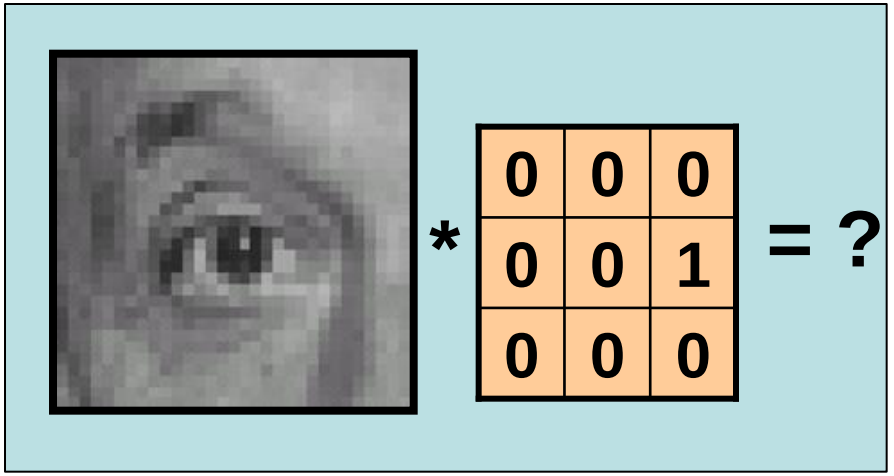
For a Gaussian or box filter, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

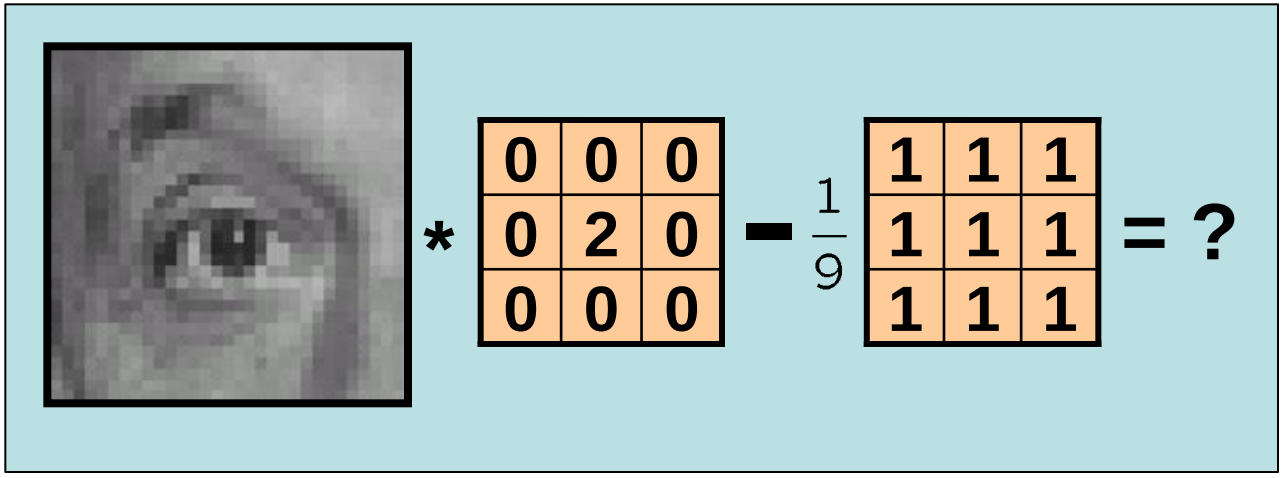
Predict the outputs using correlation filtering



$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$



$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$



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

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

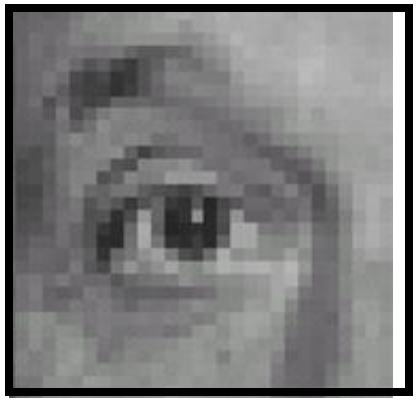
?

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

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

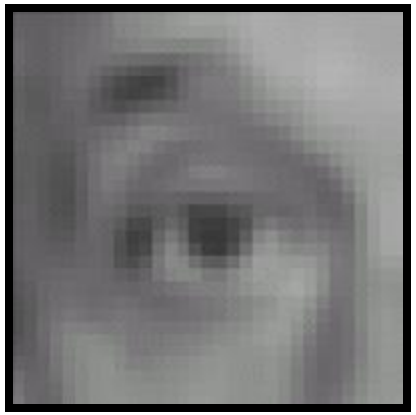
?

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

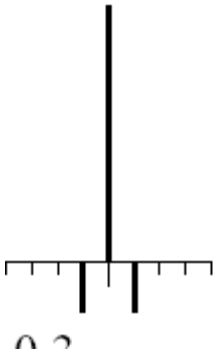
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

Source: D. Lowe

Practice with linear filters

Original

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} - \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

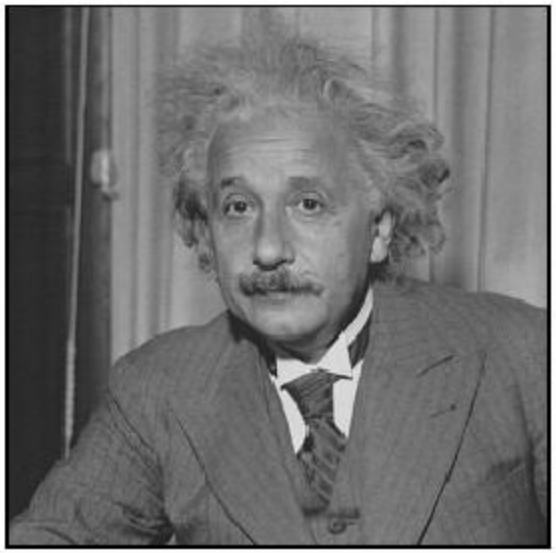


Sharpening filter:

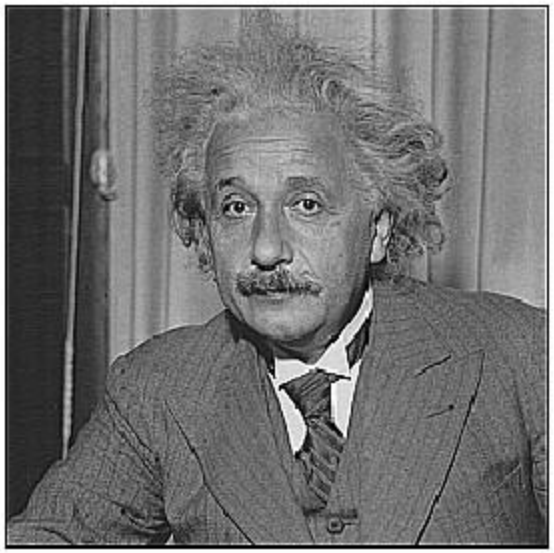
accentuates differences with local average

Source: D. Lowe

Filtering examples: sharpening



before



after

Sharpening

What does blurring take away?

–

=

Let’s add it back:

+

=

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)

Properties of convolution

- Commutative:

  $f * g = g * f$

- Associative

  $(f * g) * h = f * (g * h)$

- Distributes over addition

  $f * (g + h) = (f * g) + (f * h)$

- Scalars factor out

  $kf * g = f * kg = k(f * g)$

- Identity:

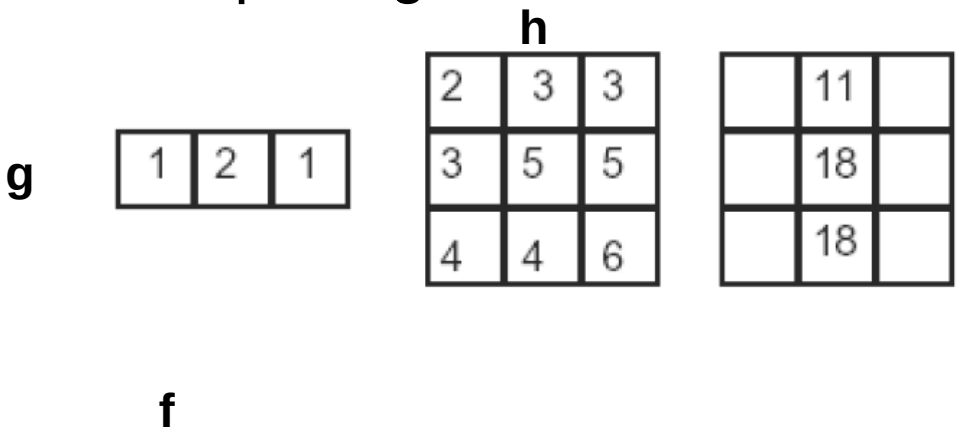
  unit impulse e = […, 0, 0, 1, 0, 0, …]. $f * e = f$

Separability

- In some cases, filter is separable, and we can factor into two steps:
  - Convolve all rows
  - Convolve all columns

Separability

- In some cases, filter is separable, and we can factor into two steps: e.g.,



What is the computational complexity advantage for a separable filter of size k x k, in terms of number of operations per output pixel?

f * (g * h) = (f * g) * h

Why is separability useful?

- Separability means that a 2D convolution can be reduced to two 1D convolutions (one among rows and one among columns)
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Gaussian filters

- Remove high-frequency components from the image (*low-pass filter*)
- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Discrete example:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Source: K. Grauman

Separability of the Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

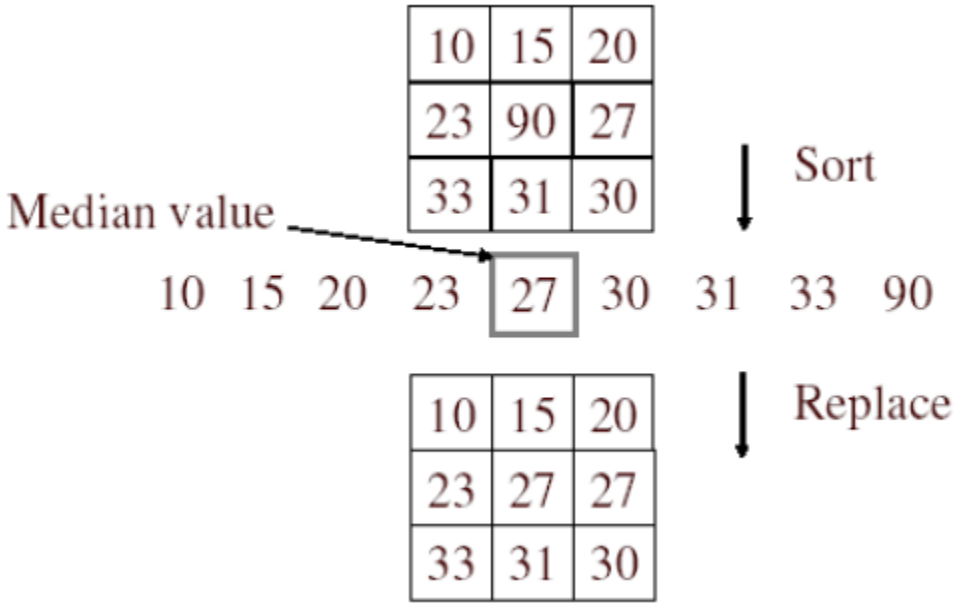
Effect of smoothing filters

5x5

Additive Gaussian noise

Salt and pepper noise

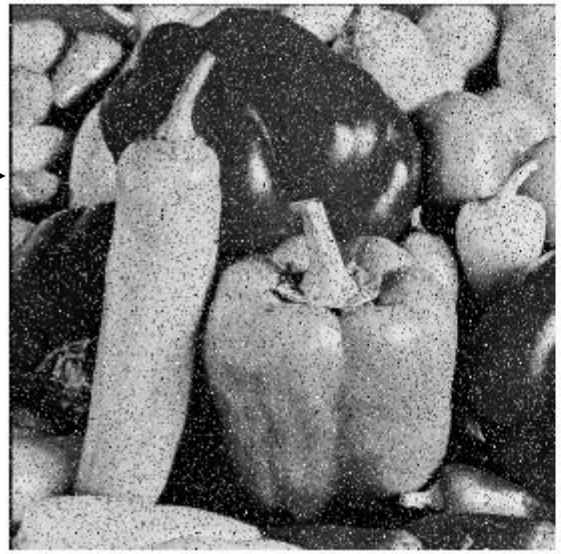
Median filter



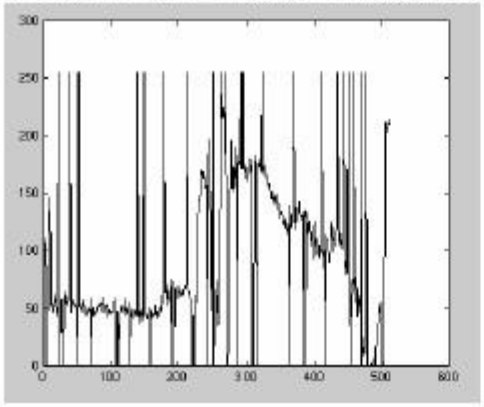
- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

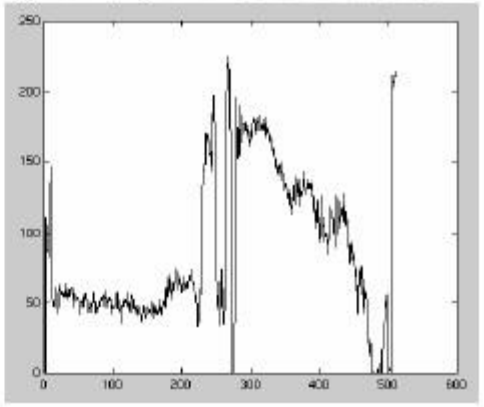
Median filter

Salt and pepper noise



Median filtered





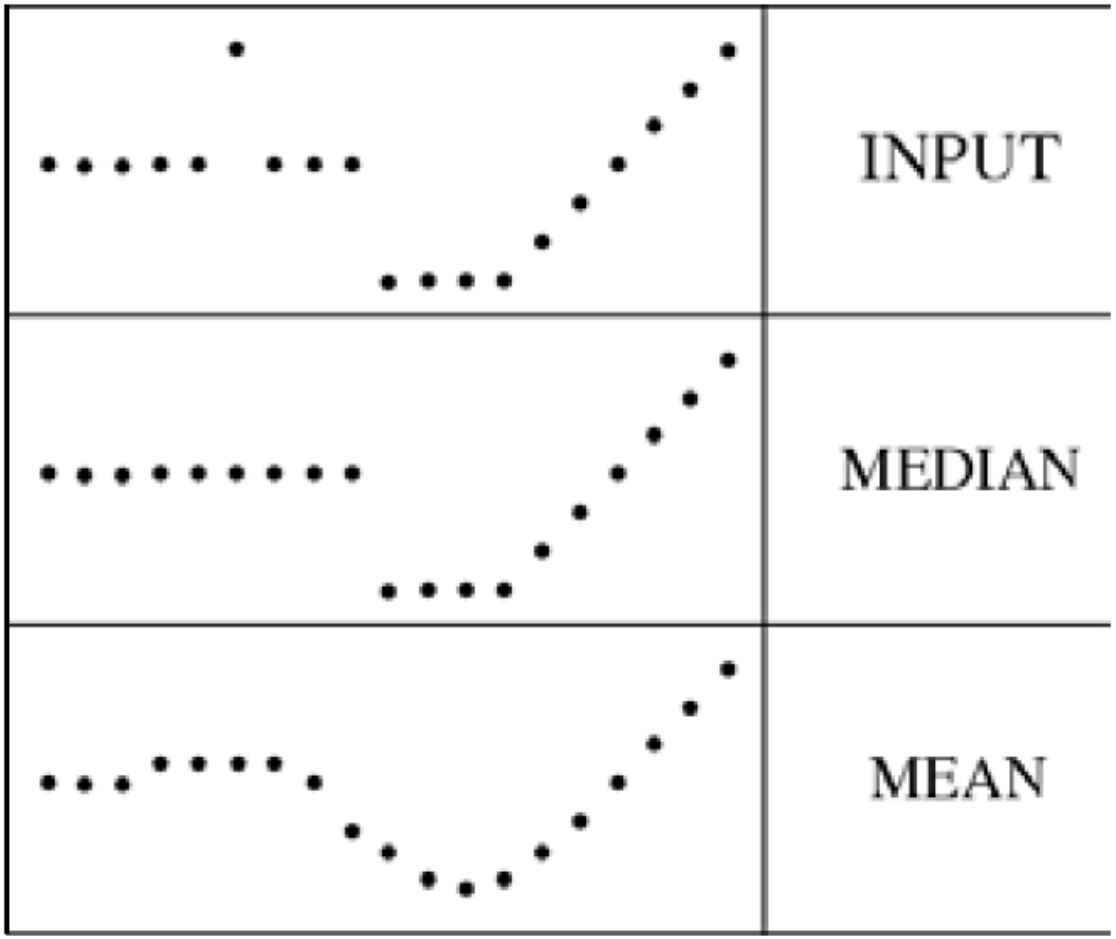
Plots of a row of the image

OpenCV: cv2.**medianBlur**(src, ksize[, dst])->dst

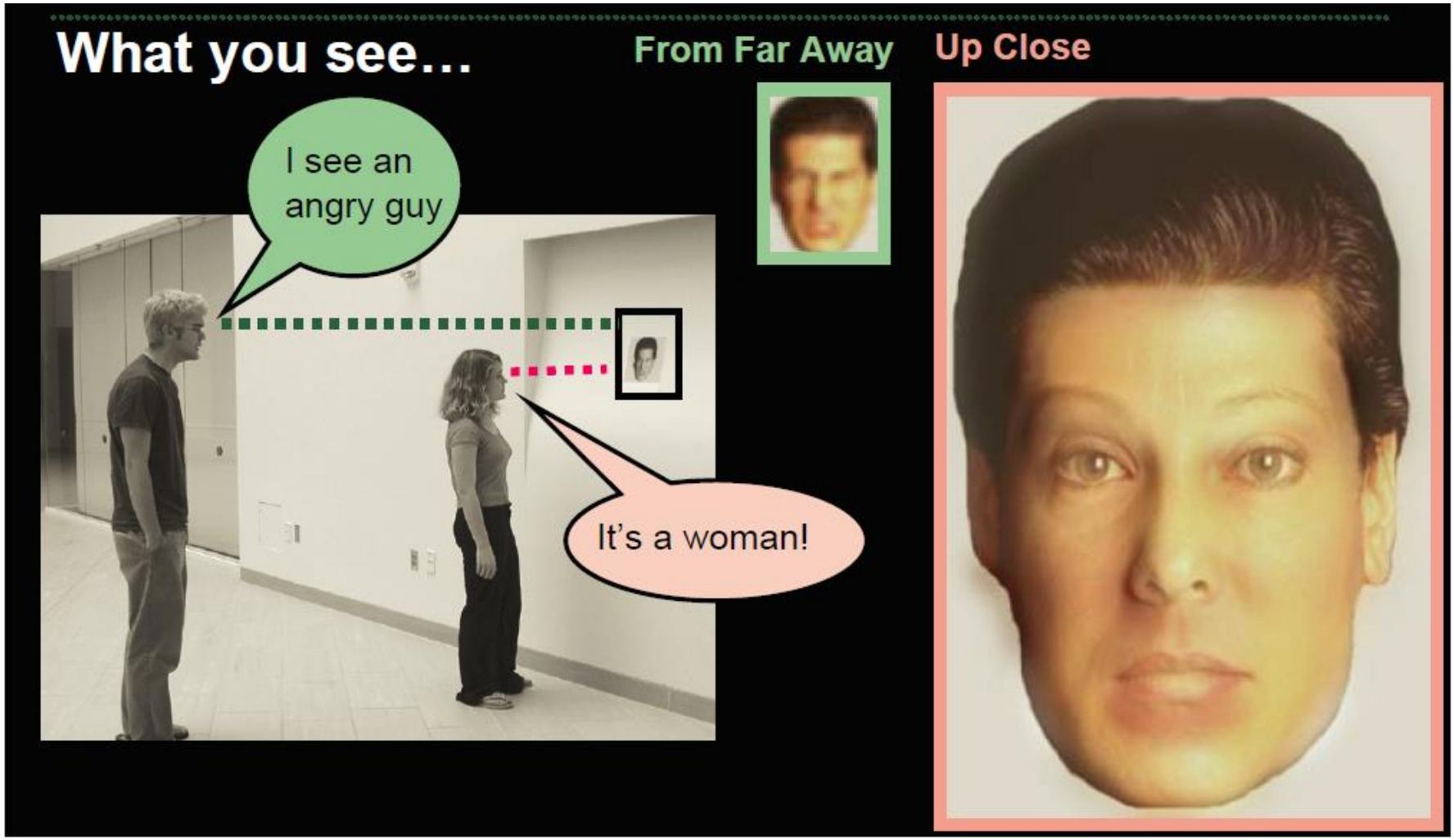
Source: M. Hebert

Median filter

- Median filter is edge preserving



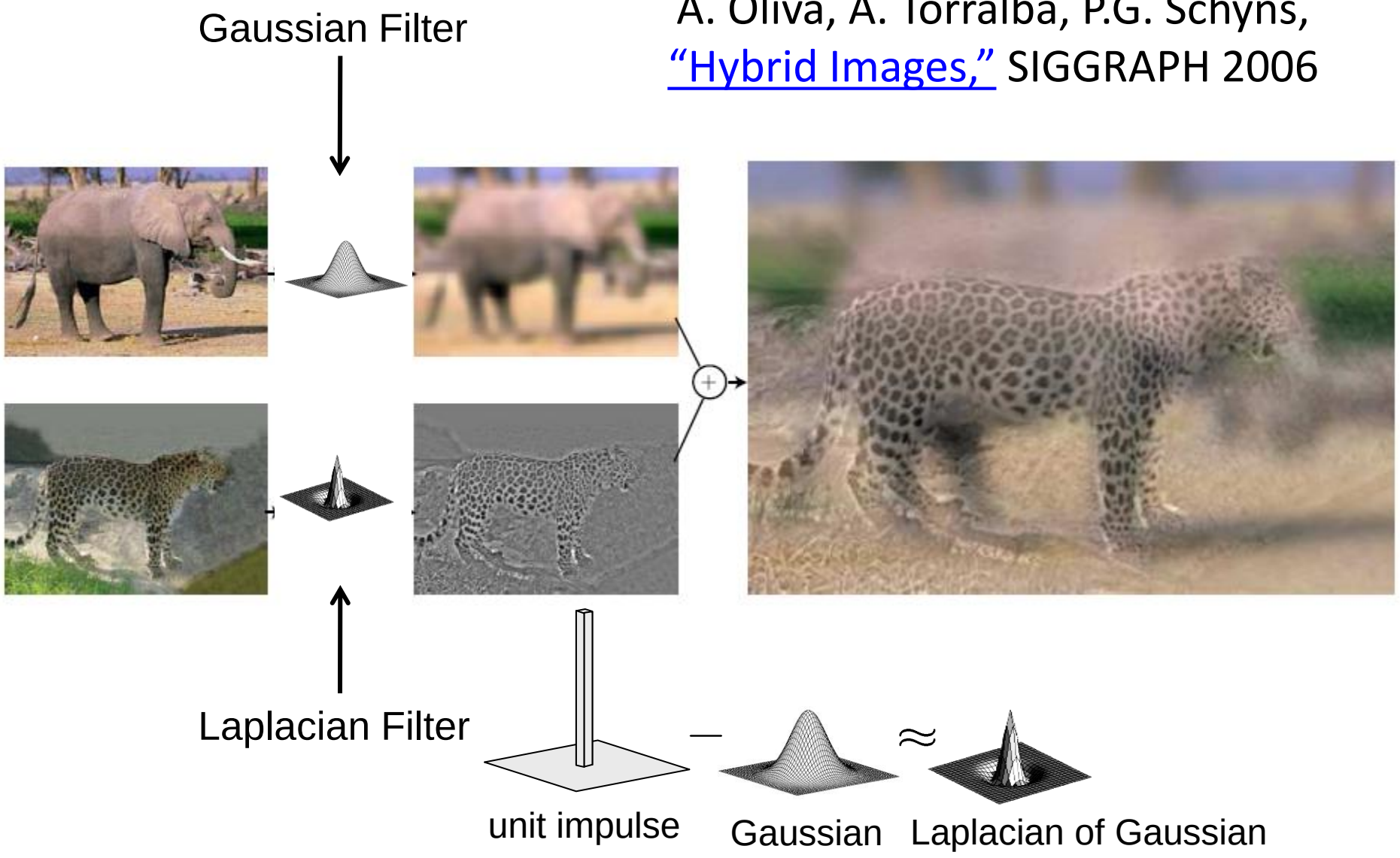
Filtering application: Hybrid Images

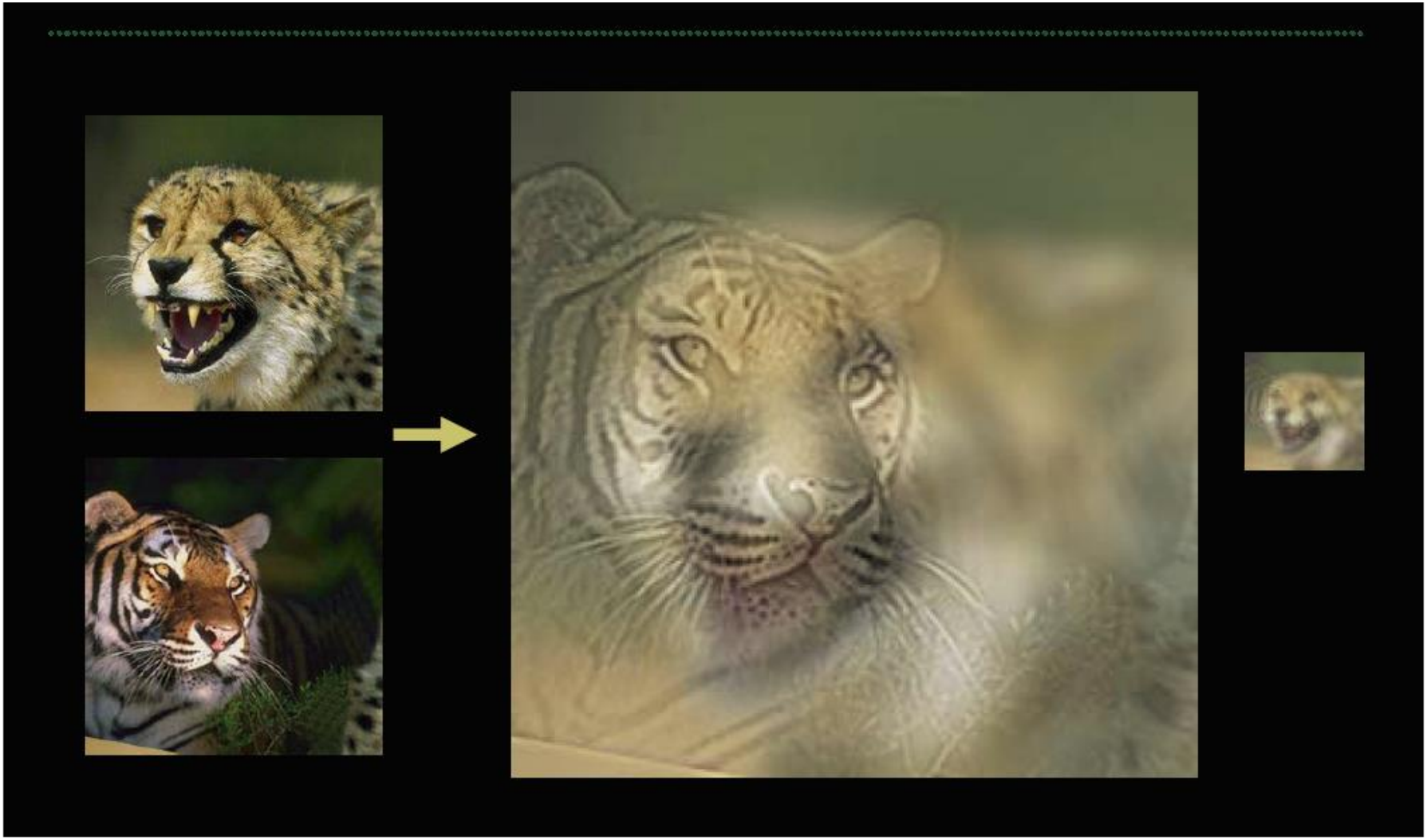


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Application: Hybrid Images

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





Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Changing expression

Sad ⟷ Surprised

Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Summary

- Image “noise”
- Linear filters and convolution useful for
  - Enhancing images (smoothing, removing noise)
    - Box filter
    - Gaussian filter
    - **Impact of scale / width of smoothing filter**
  - Detecting features (next time)
- Separable filters more efficient
- Median filter: a non-linear filter, edge-preserving
- Different frequency bands contribute different information.

Features: Edge detection

Edge detection

- **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
- **Why?**

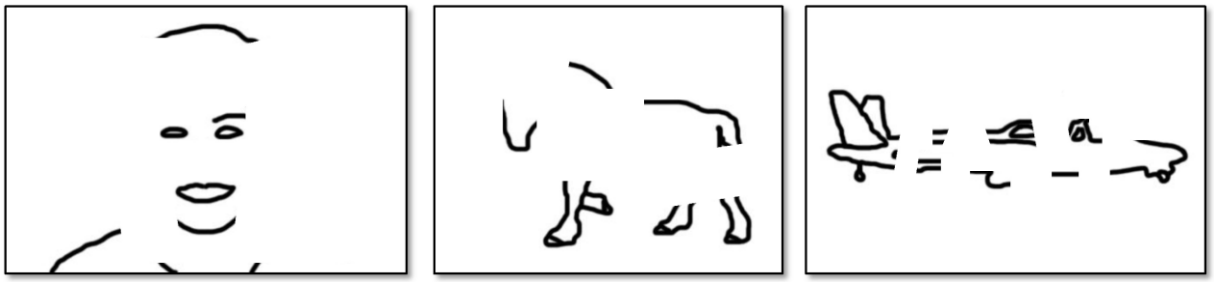
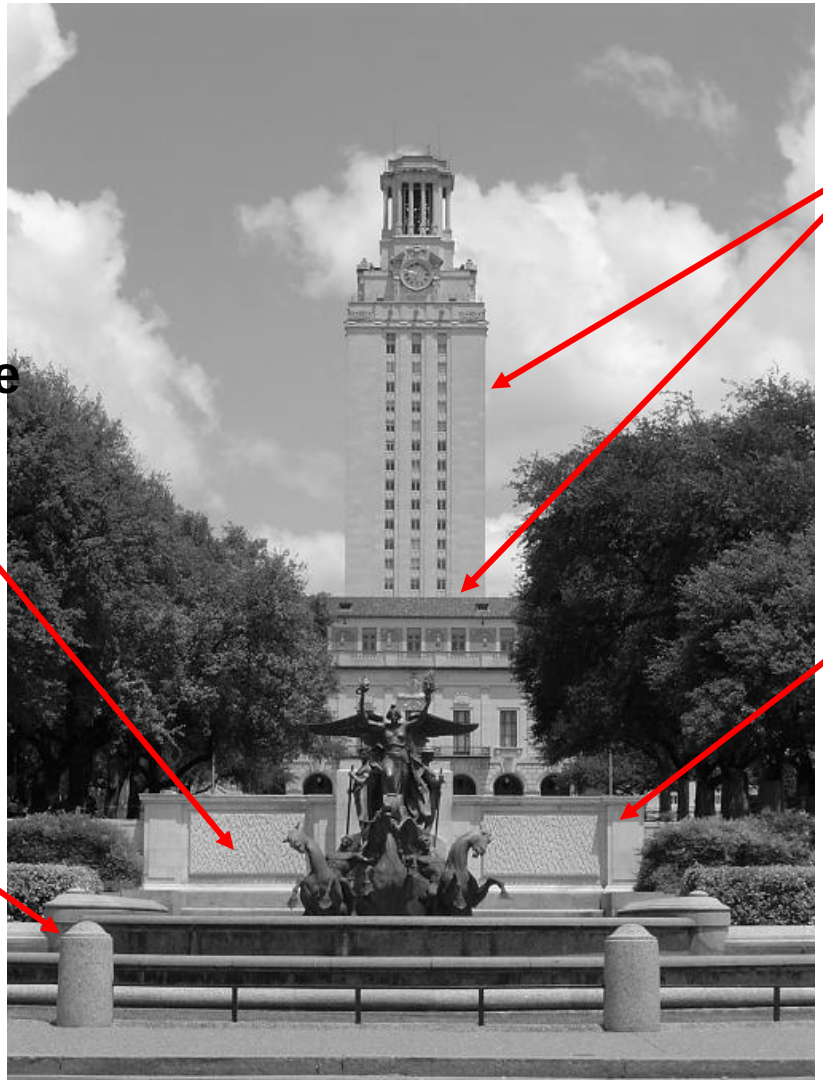


Figure from J. Shotton et al., PAMI 2007

- **Main idea**: look for strong gradients, post-process

What causes an edge?



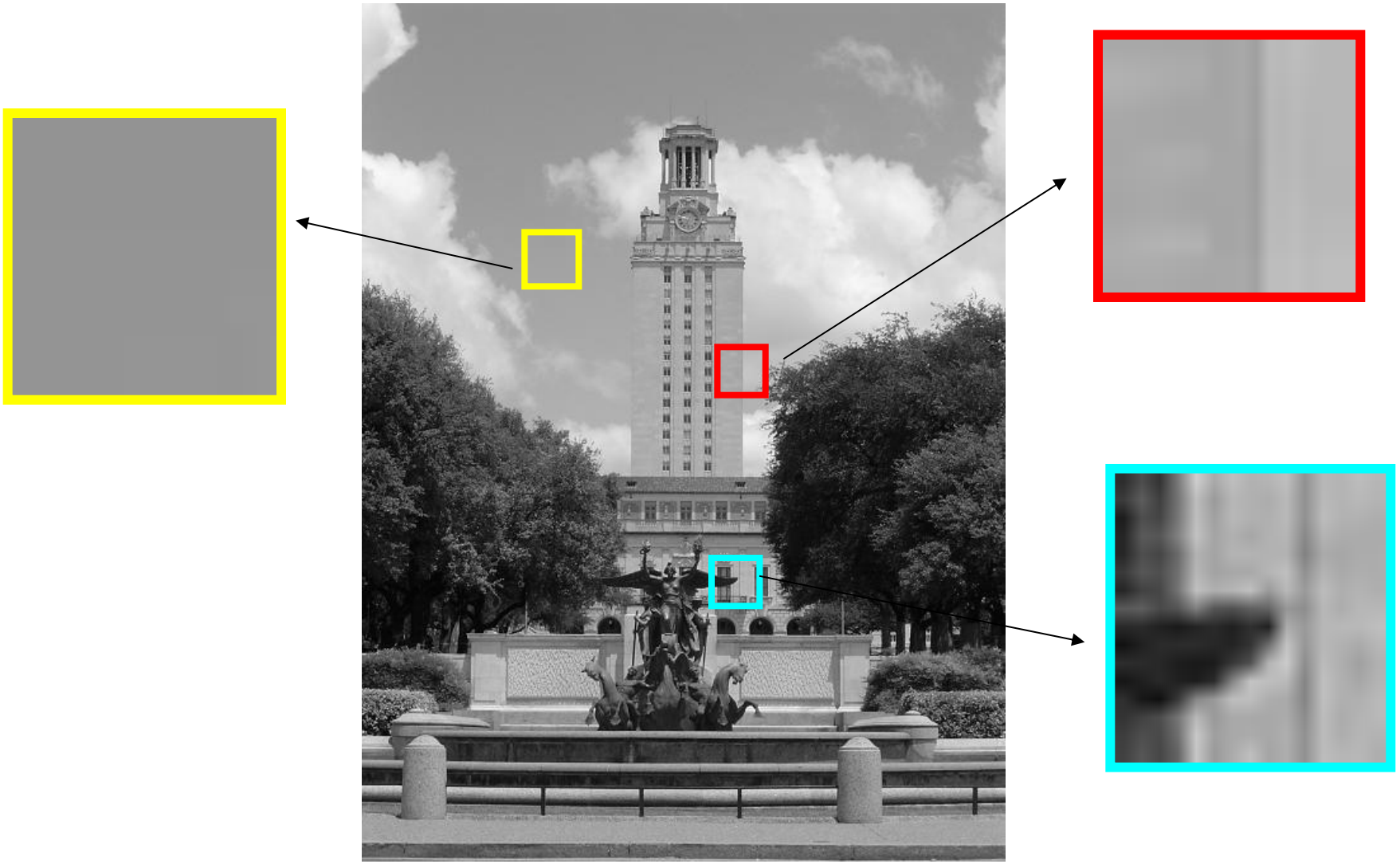
Reflectance change: appearance information, texture

Depth discontinuity: object boundary

Cast shadows

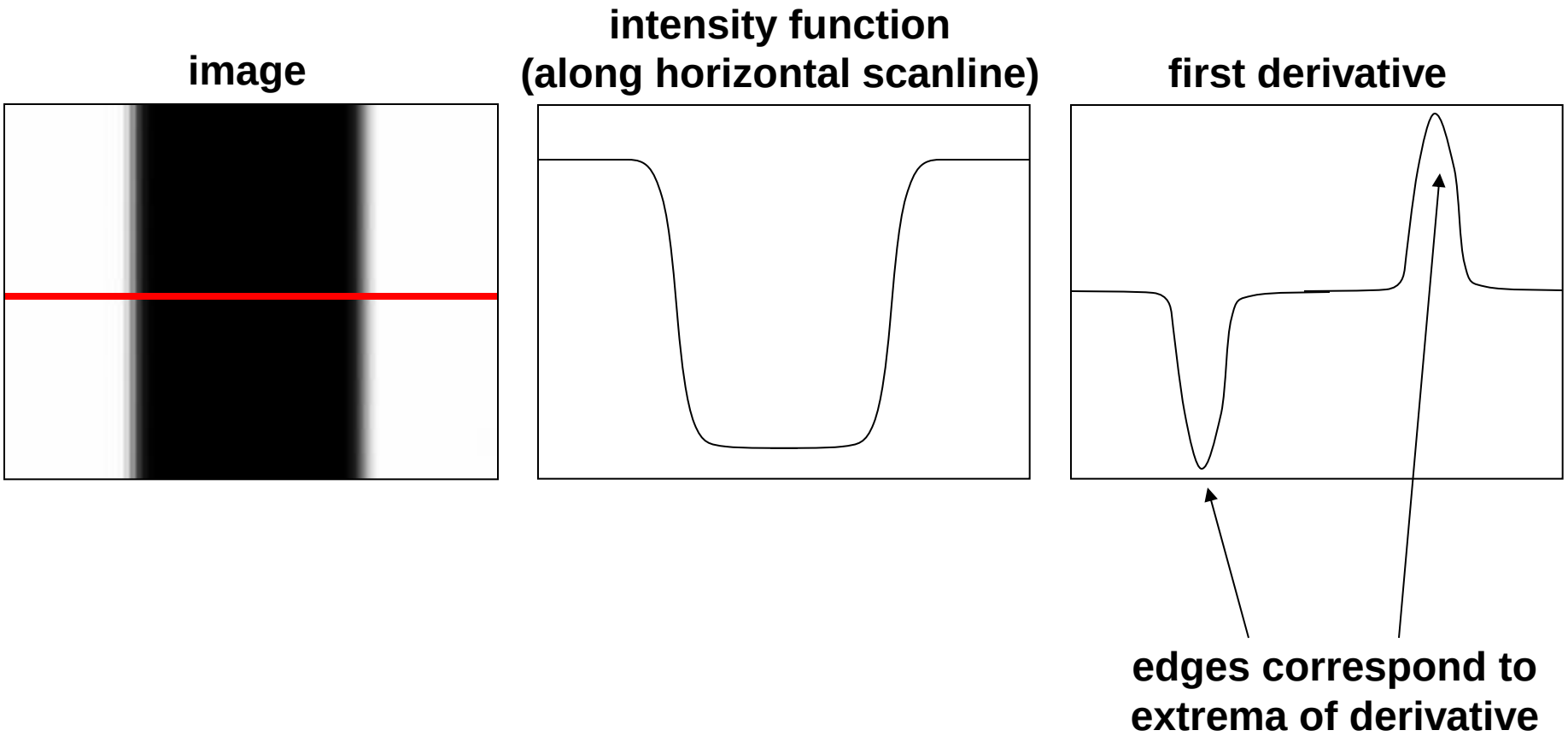
Change in surface orientation: shape

Edges/gradients and invariance



Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Source: L. Lazebni

Derivatives with convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

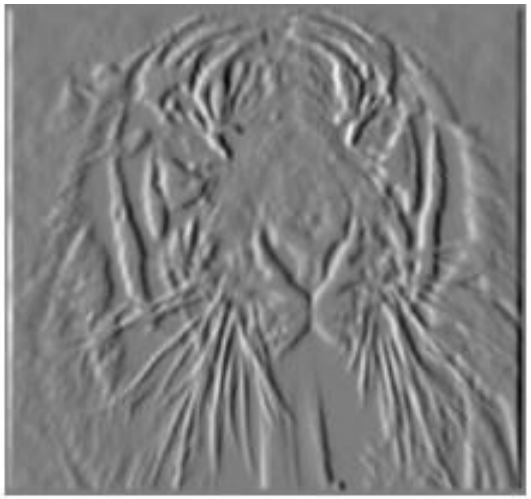
For discrete data, we can approximate using finite differences:

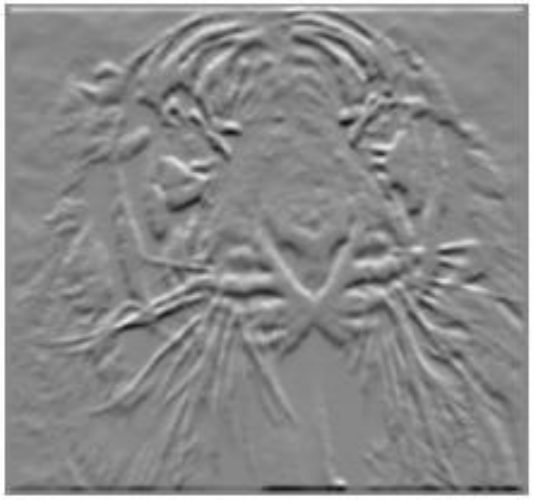
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$





$$\frac{\partial f(x,y)}{\partial y}$$

-1	1

-1
1

?

or

1
-1

Which shows changes with respect to x?

(showing filters for correlation)

Assorted finite difference filters

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

cv2.**Sobel**(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[, borderType]]]]])! ->dst

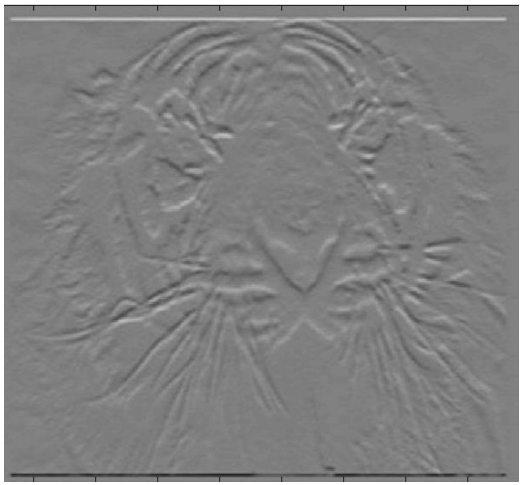
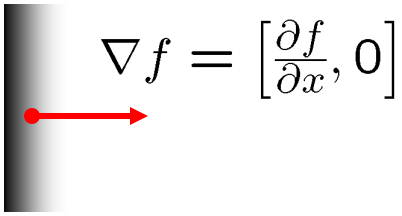


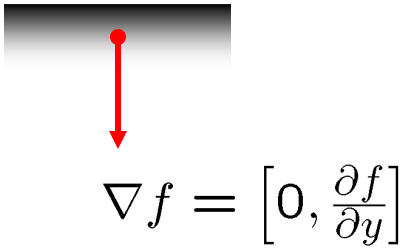
Image gradient

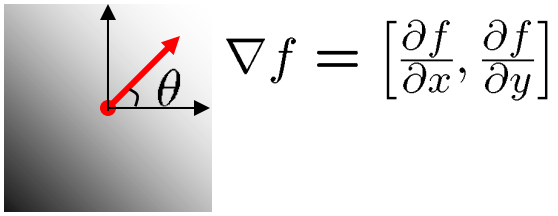
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity







The **gradient direction** (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

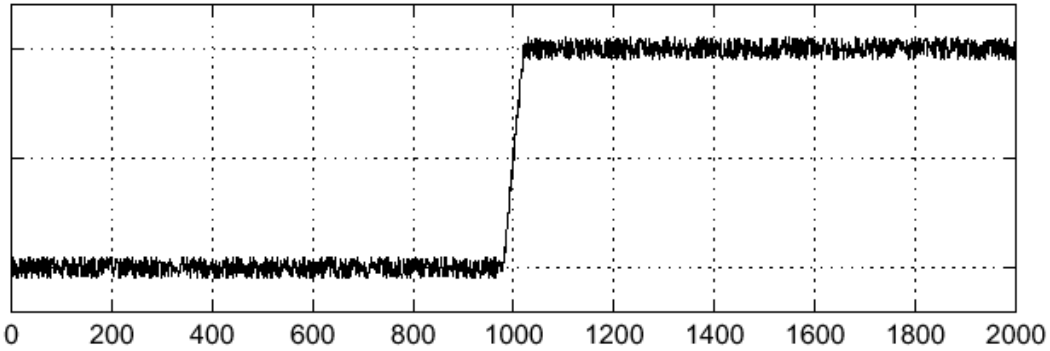
Slide credit Steve Seitz

Effects of noise

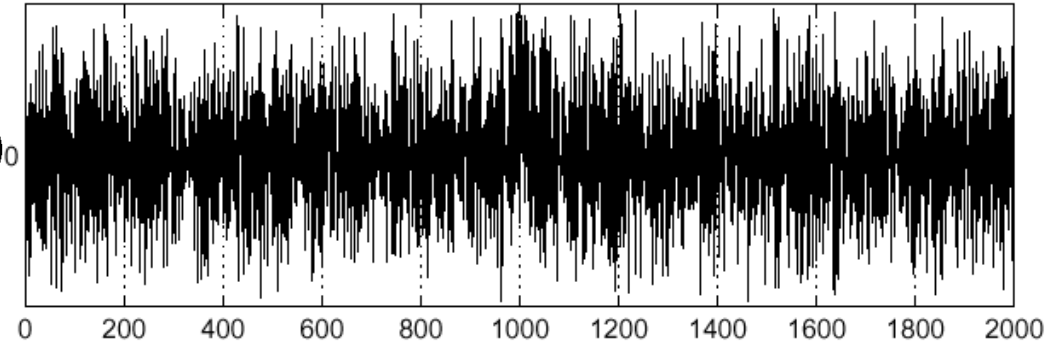
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx} f(x)$



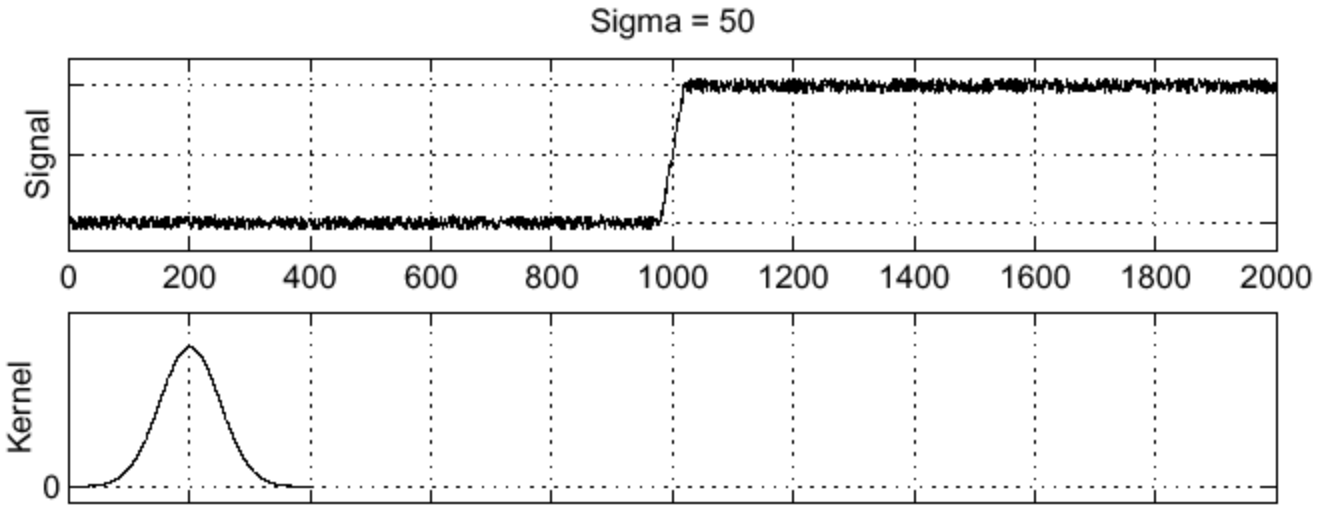
Where is the edge?

Slide credit Steve Seitz

Solution: smooth first

f

h



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

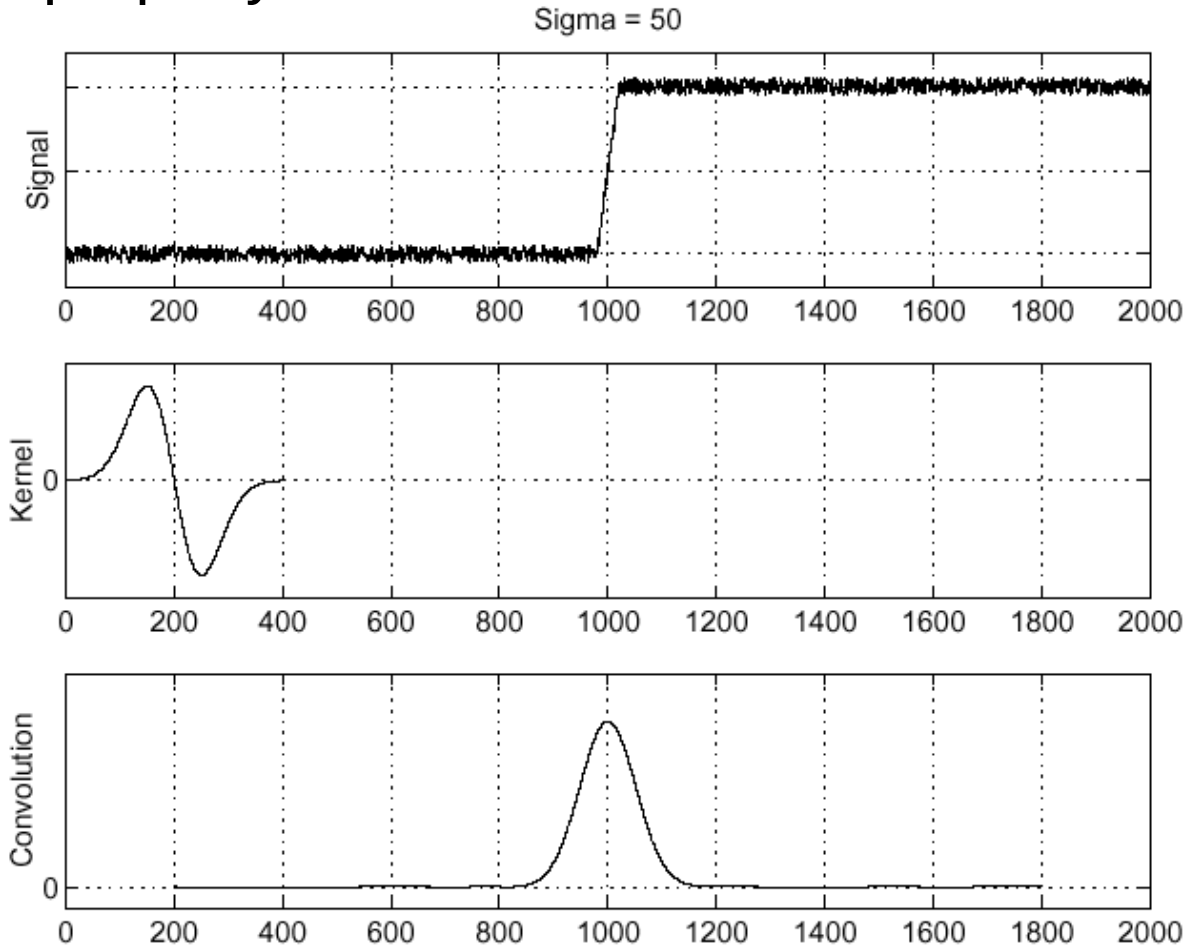
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



Slide credit Steve Seitz

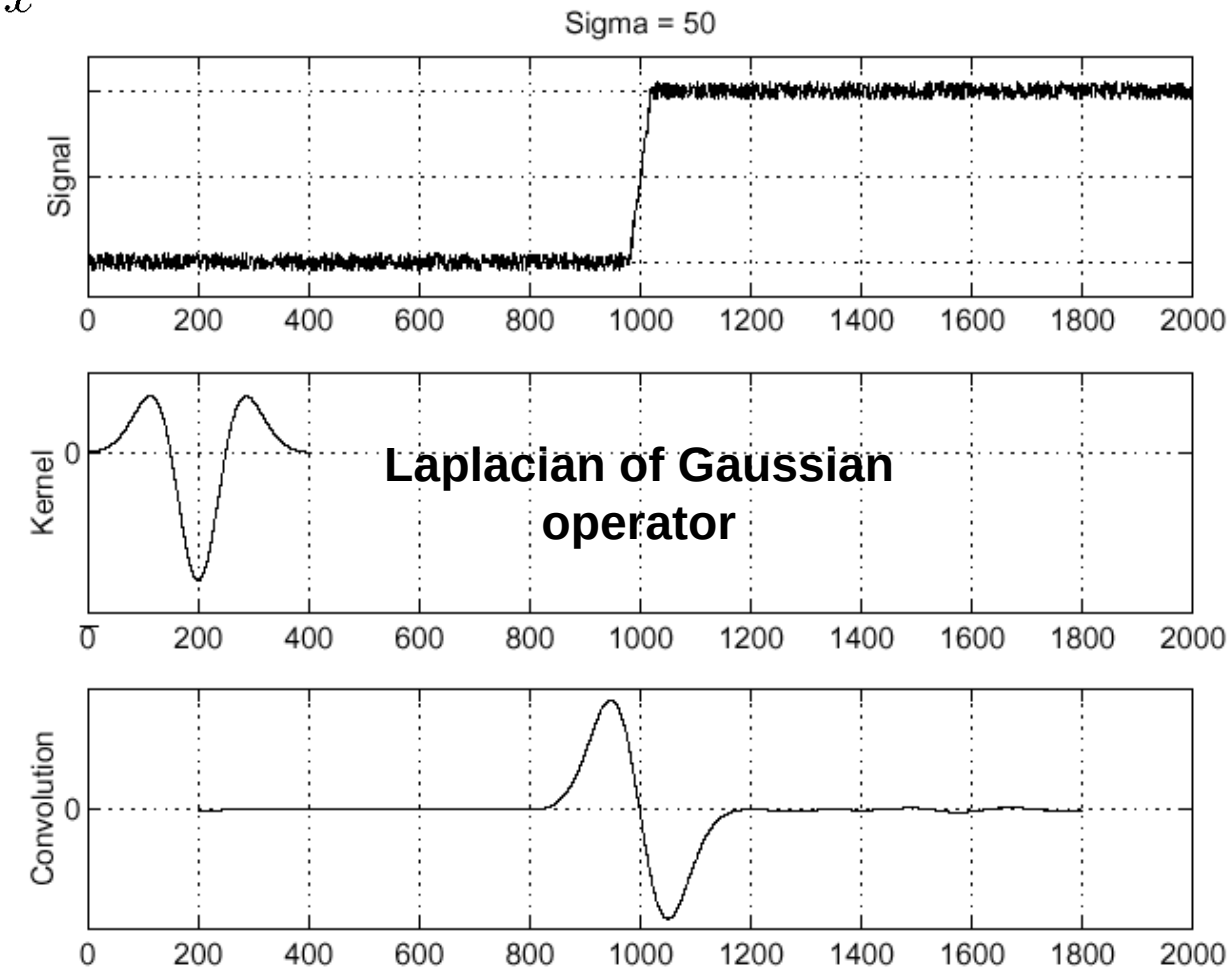
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$



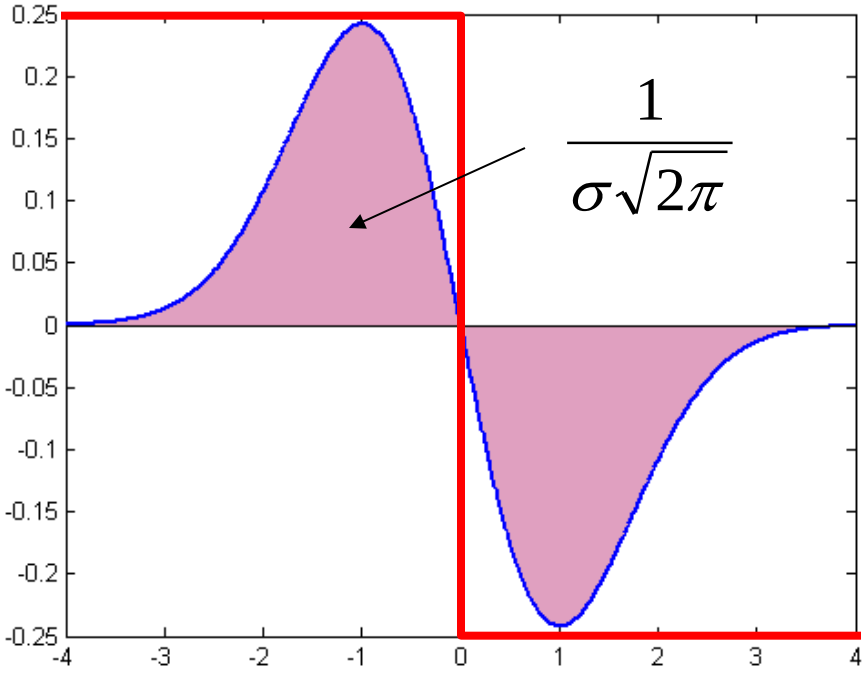
Where is the edge?

Zero-crossings of bottom graph

Slide credit: Steve Seitz

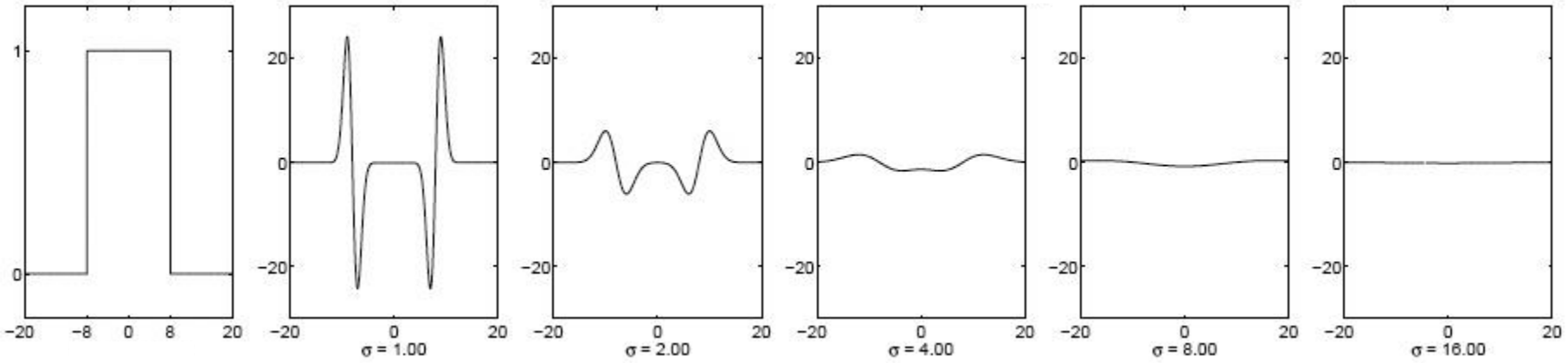
Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases



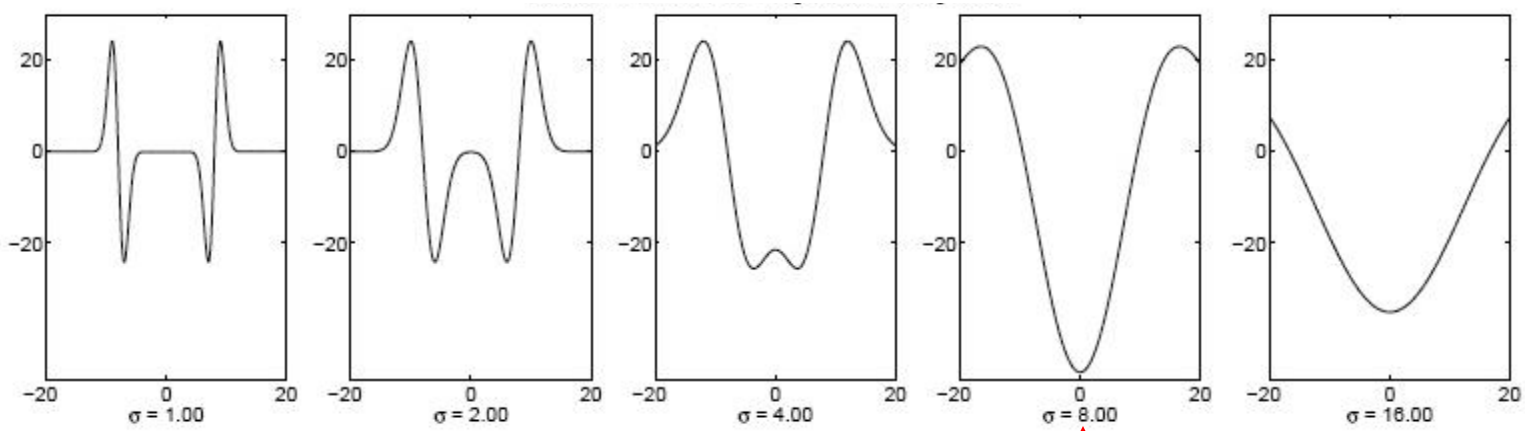
Effect of scale normalization

Original signal



Unnormalized Laplacian response

Scale-normalized Laplacian response

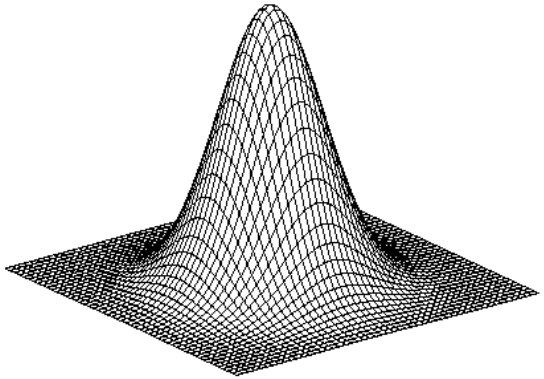


maximum

Scale normalization

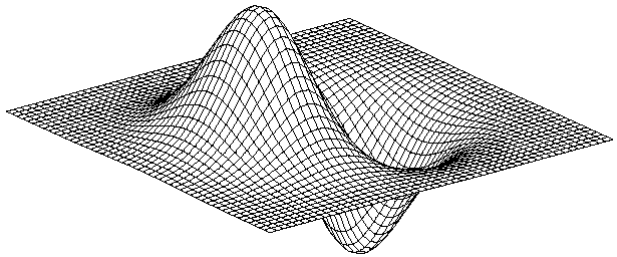
- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

2D edge detection filters



Gaussian

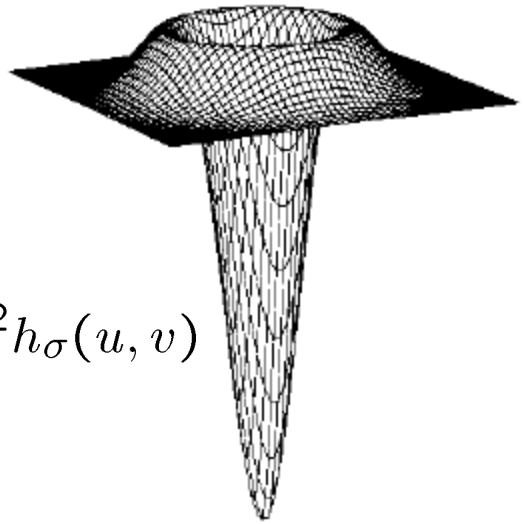
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

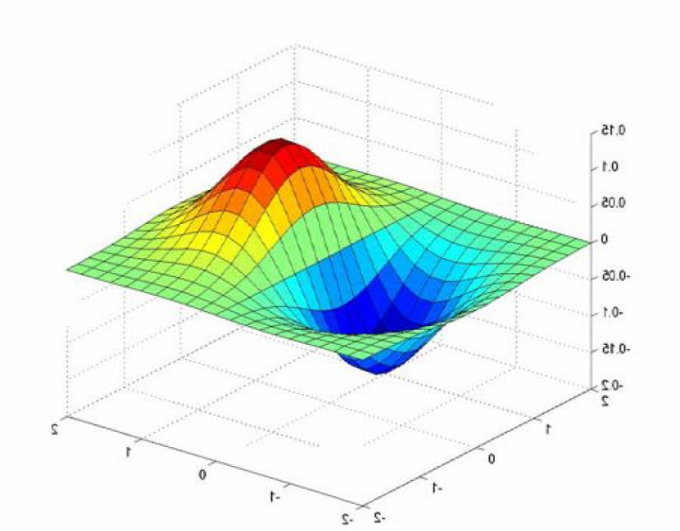
$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

Slide credit: Steve Seitz

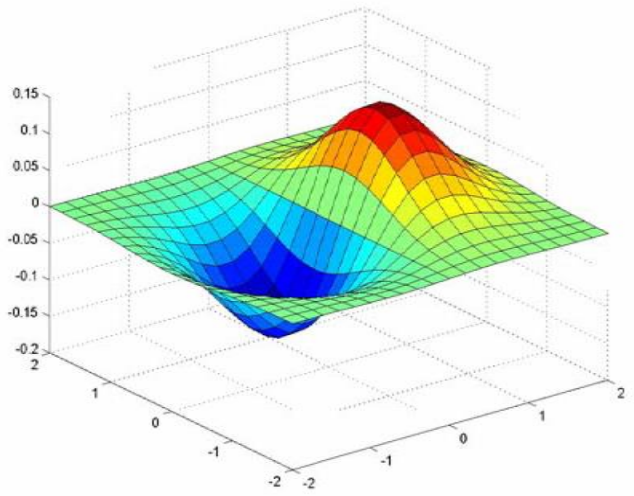
Derivative of Gaussian filters

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

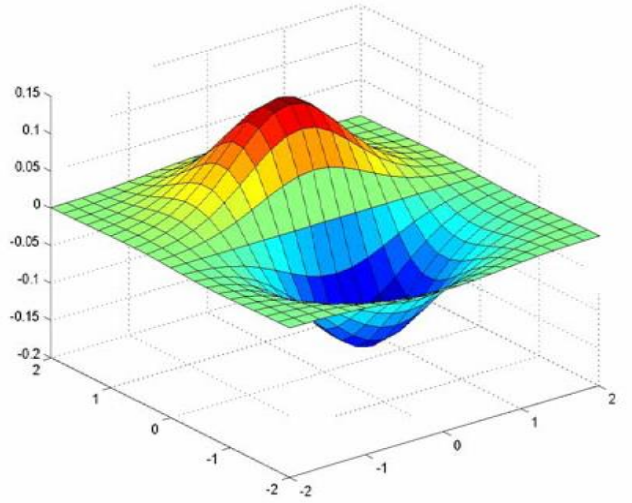
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



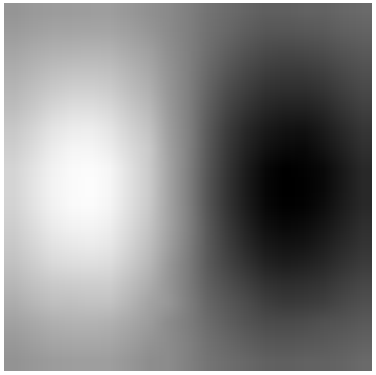
Derivative of Gaussian filters

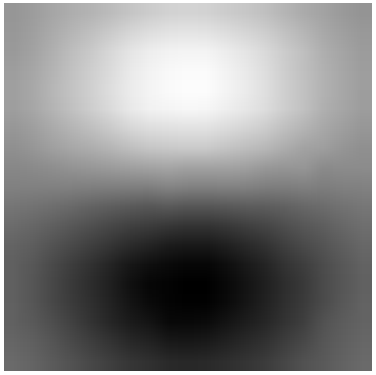


***x*-direction**



***y*-direction**

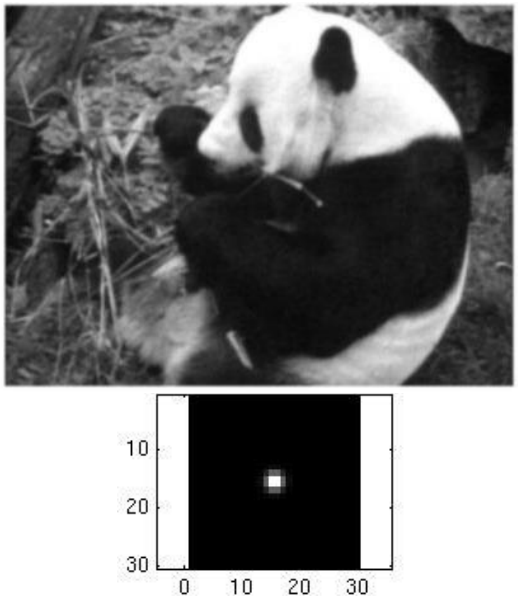


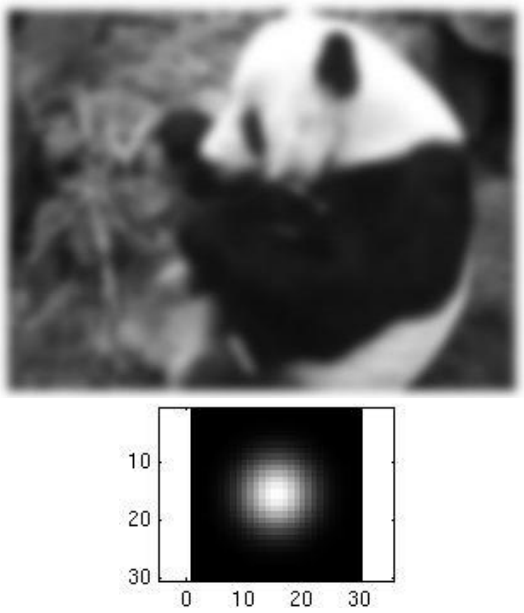


Source: L. Lazebnik

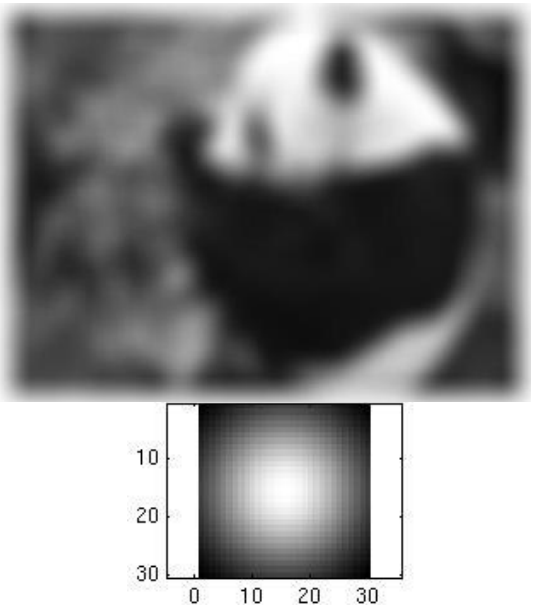
Smoothing with a Gaussian

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



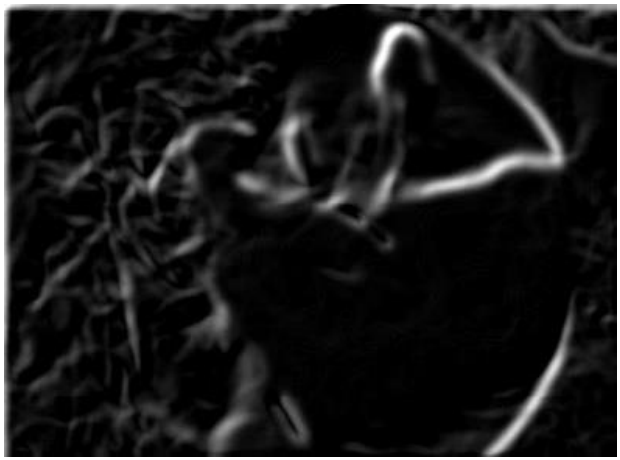


...



Effect of σ on derivatives

σ = 1 pixel



σ = 3 pixels

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected

So, what scale to choose?

It depends what we're looking for.

Mask properties

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Derivatives
  - Different signs used to get high response in regions of high contrast
  - Sum to zero → no response in constant regions
  - High absolute value at points of high contrast

Gradients -> edges

Primary edge detection steps:

1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

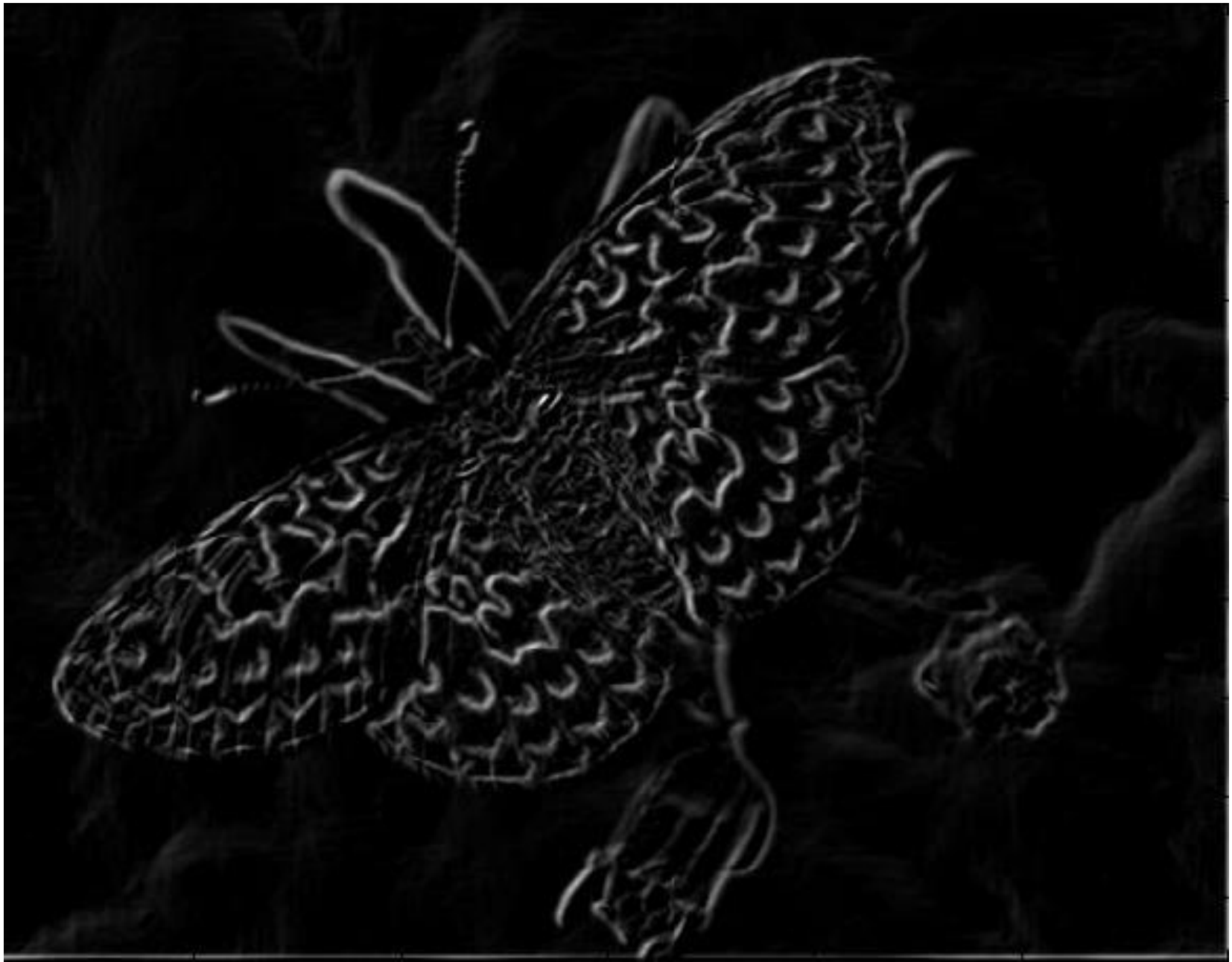
   Determine which local maxima from filter output are actually edges vs. noise

   - Threshold,
   - Thin

- No good results !

Original image

Gradient magnitude image



Thresholding gradient with a lower threshold

Thresholding gradient with a higher threshold

Low-level edges vs. perceived contours

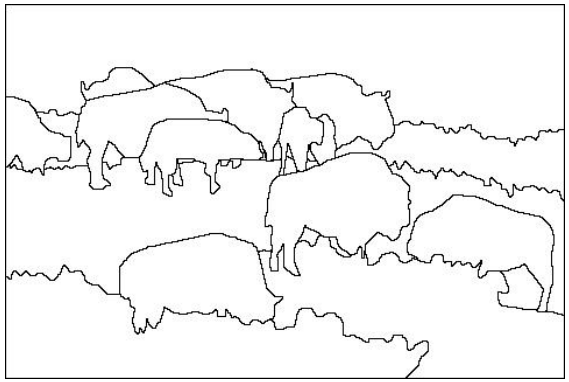
Background

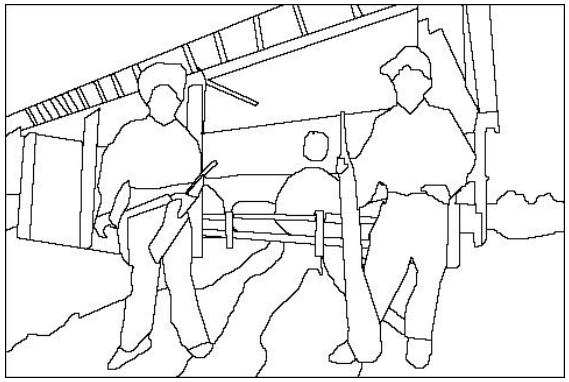
Texture

Shadows

Low-level edges vs. perceived contours

image | **human segmentation** | **gradient magnitude**

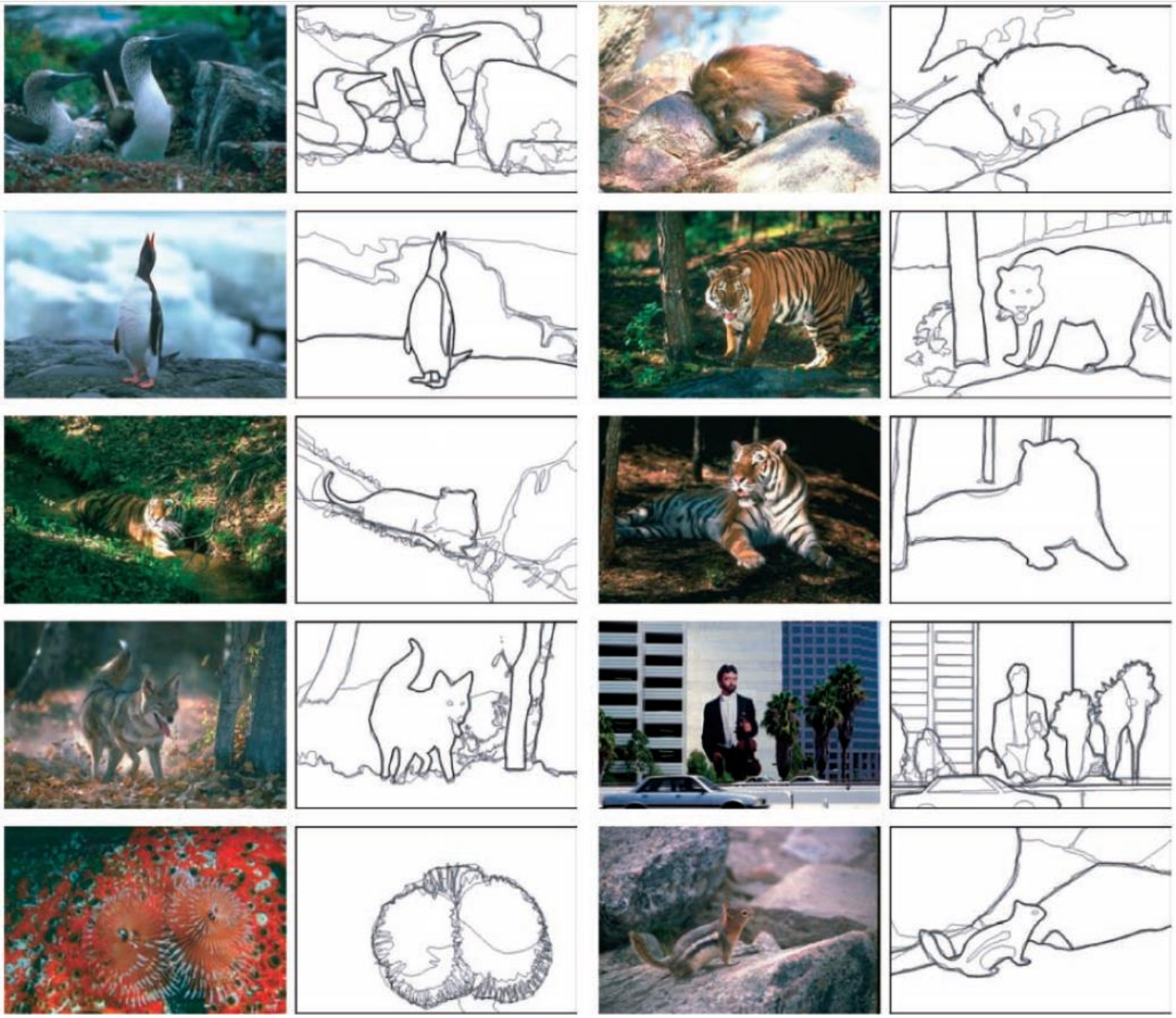






- Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Source: L. Lazebnik

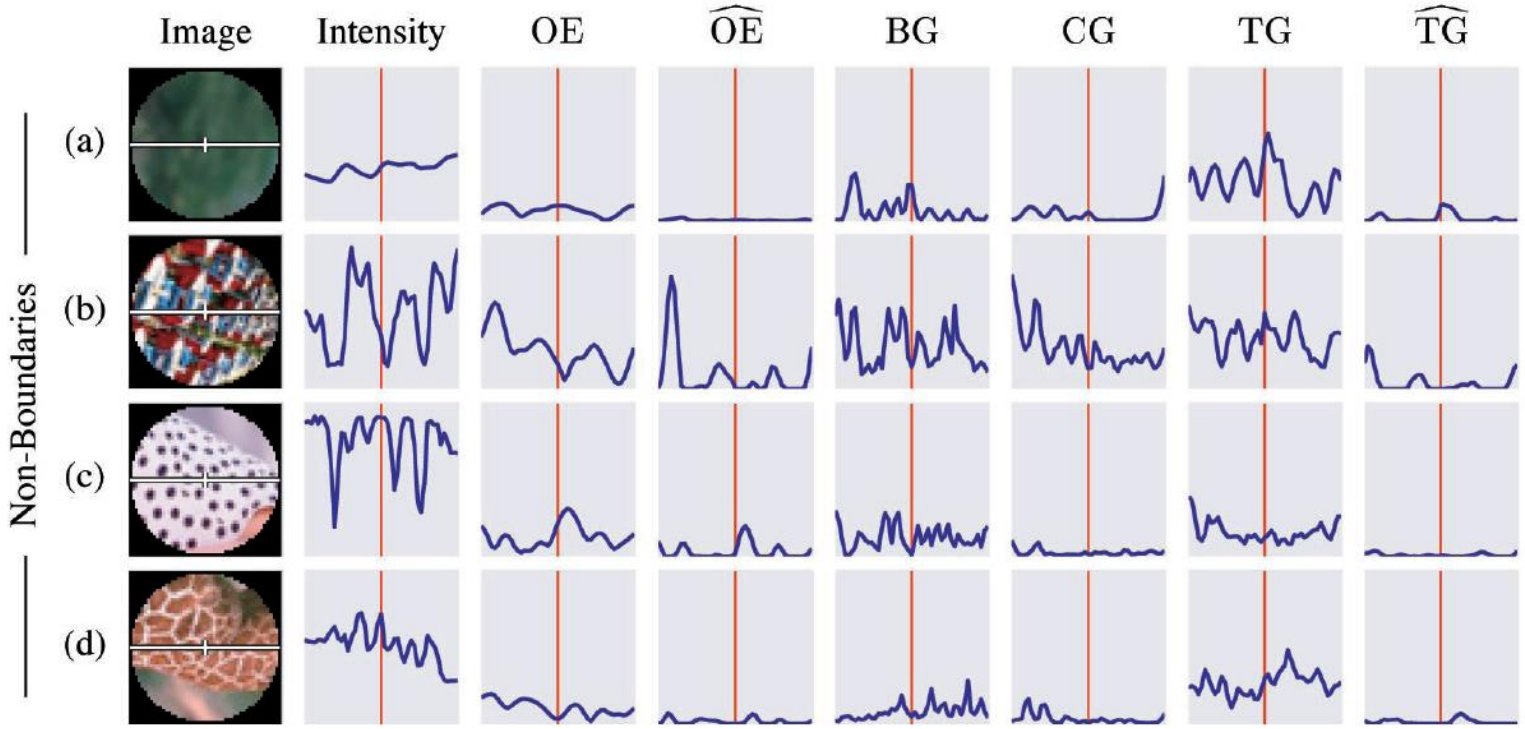
Learn from humans which combination of features is most indicative of a “good” contour?



[D. Martin et al. PAMI 2004]

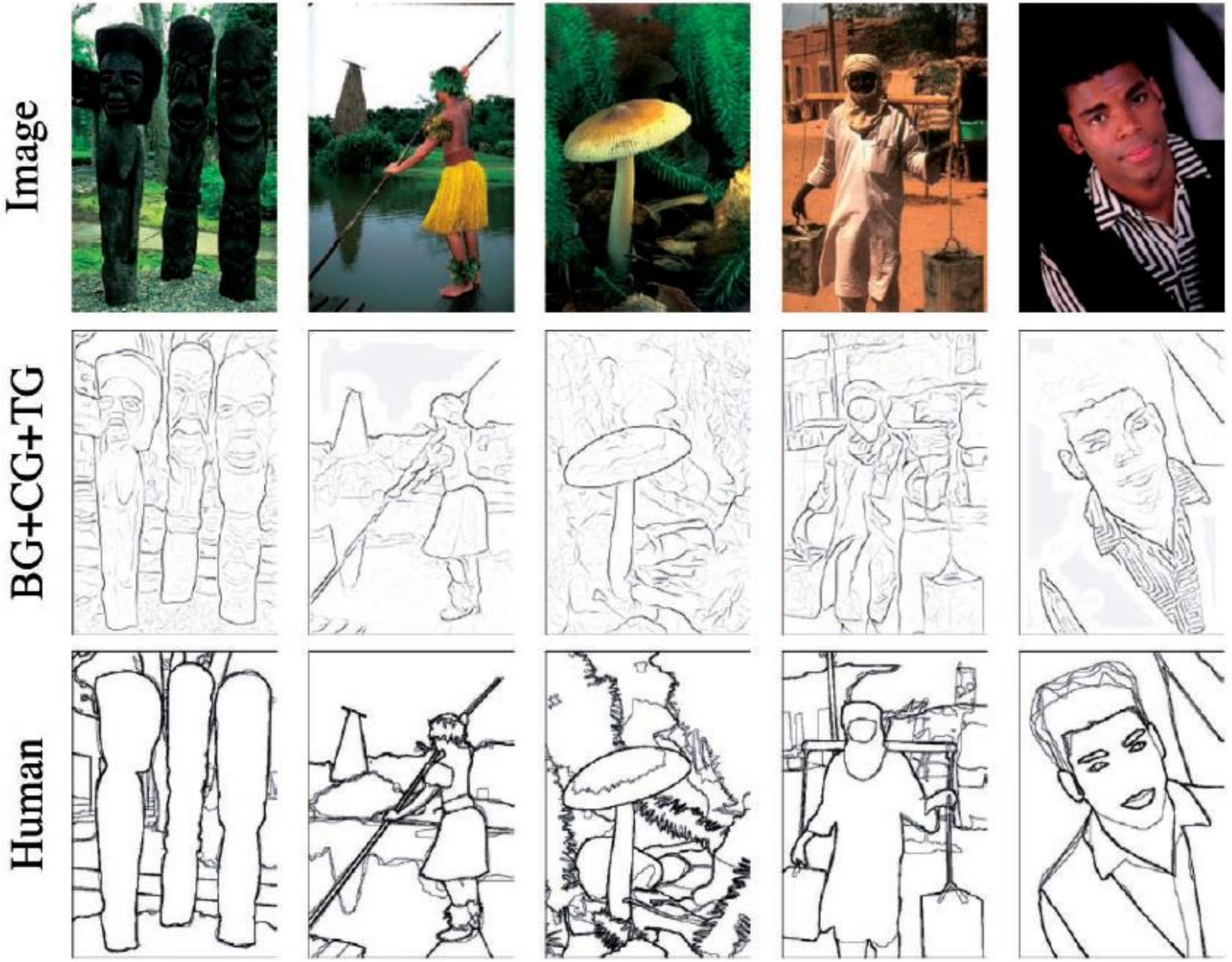
Human-marked segment boundaries

What features are responsible for perceived edges?



Feature profiles (oriented energy, brightness, color, and texture gradients) along the patch's horizontal diameter

[D. Martin et al. PAMI 2004]



[D. Martin et al. PAMI 2004]

Summary

- Filters allow local image neighborhood to influence our description and features
  - Smoothing to reduce noise
  - Derivatives to locate contrast, gradient
- Convolution properties will influence the efficiency with which we can process images.
  - Associative
  - Filter separability
- Edge detection processes the image gradient to find curves, or chains of edgels.

Slide Credits

- Slides Kristen Grauman, Steve Seitz
- and others, as marked…