CS 4501: Introduction to Computer Vision

Filtering and Edge Detection

Connelly Barnes

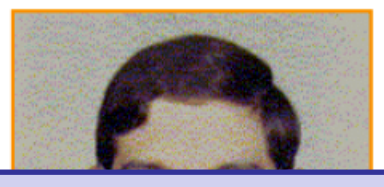
Slides from Jason Lawrence, Fei Fei Li, Juan Carlos Niebles, Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein, David Dobkin

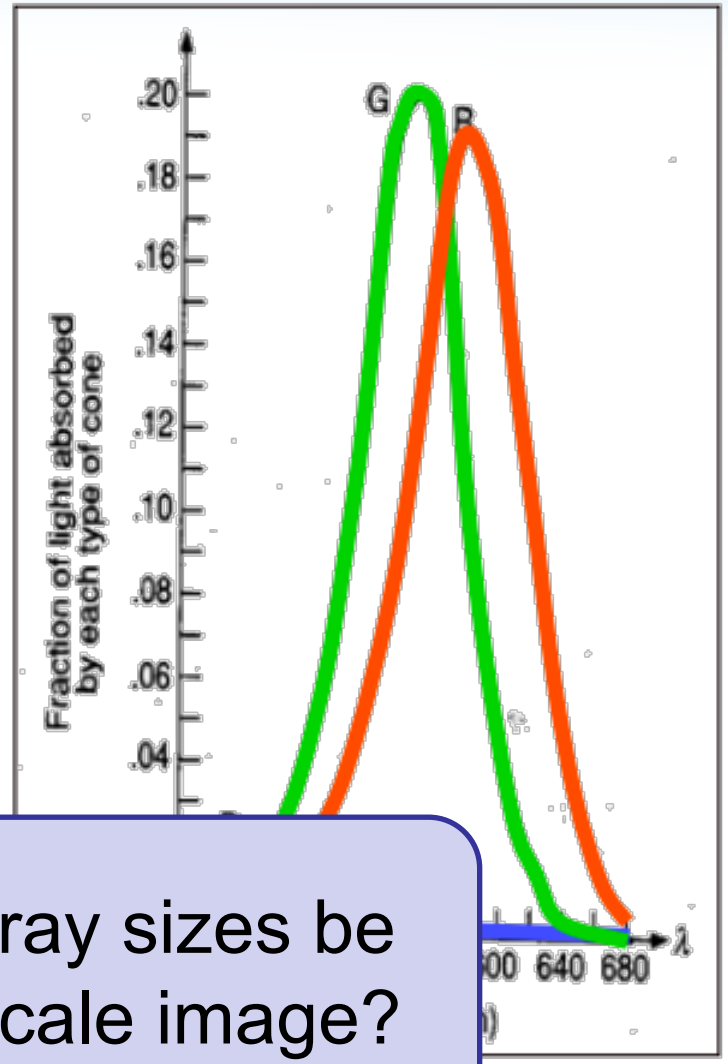
Outline

- **Simple image processing**
  - **Greyscale**
  - Brightness
  - Mirroring or “flipping”
- Filtering
  - Linear filters: cross-correlation and convolution
  - Gaussian filters
- Edge detection
  - Simple edge detector
  - Canny edge detector

Image processing: greyscale

- The human retina perceives red, green, blue.
- To compute luminance of a pixel, we need to take an average of the RGBs:
  - L = 0.21 R + 0.72 G + 0.07 B (ITU HDTV)
  - L = 0.3 R + 0.59 G + 0.11 B (W3C)





If represented as arrays, what would the array sizes be for the input RGB image? The output greyscale image?

Original Grayscale

FvDFH

Image processing: brightness

- Simply scale the array of RGB values.
  - Must clamp to valid range [0, 1]
  - $I_{\text{out}} = \min(\alpha\, I_{\text{in}}, 1)$
- Where is this operation used?
  - Photo adjustment, dataset augmentation

Original

Brighter

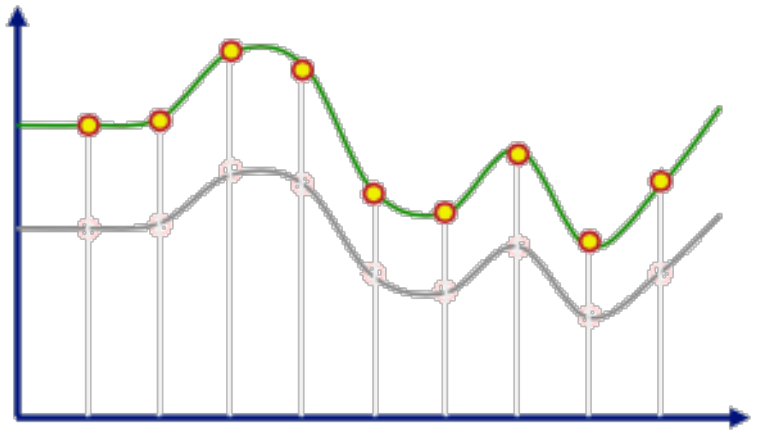
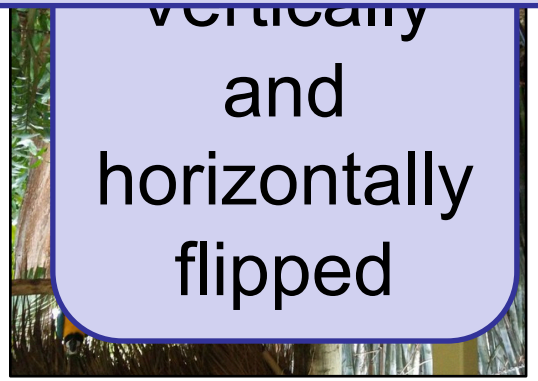


Image processing: mirroring or “flipping”

Mirrored or Horizontally

Why do we care?
Some linear filters involve “flipping” operations.
Another way to do dataset augmentation.

Vertically flipped



and horizontally flipped

Outline

- Simple image processing
  - Greyscale
  - Brightness
  - Mirroring or “flipping”
- **Filtering**
  - Linear filters: cross-correlation and convolution
  - Gaussian filters
- Edge detection
  - Simple edge detector
  - Canny edge detector

Image filtering

- **Filtering:**
  - Form a new image whose pixels are a combination of original pixel values.

- **Goals:**
  - Extract useful information from image
    - Features (corners, edges, blobs, …)

  - Enhance image properties
    - Remove noise, remove unwanted objects, …

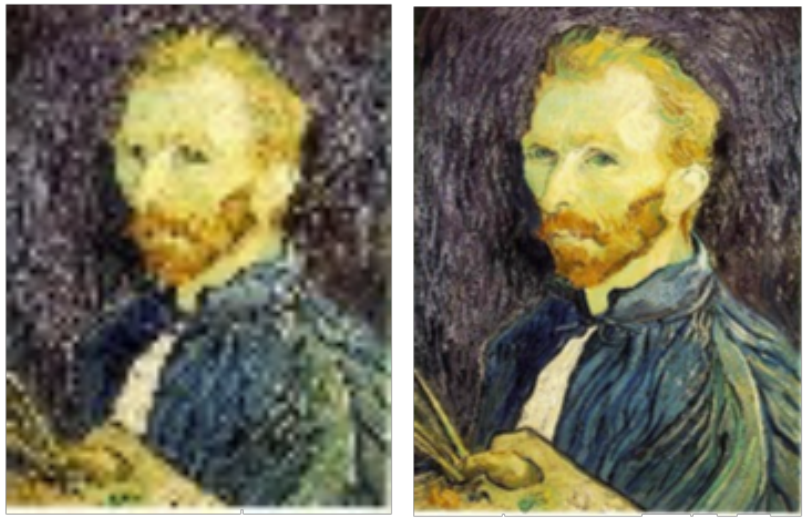
Slide from Fei Fei Li, Juan Carlos Niebles

Image filtering

De-noising

Salt and pepper noise

Super-resolution



In-painting

Bertamio et al

Slide from Fei Fei Li, Juan Carlos Niebles

Outline

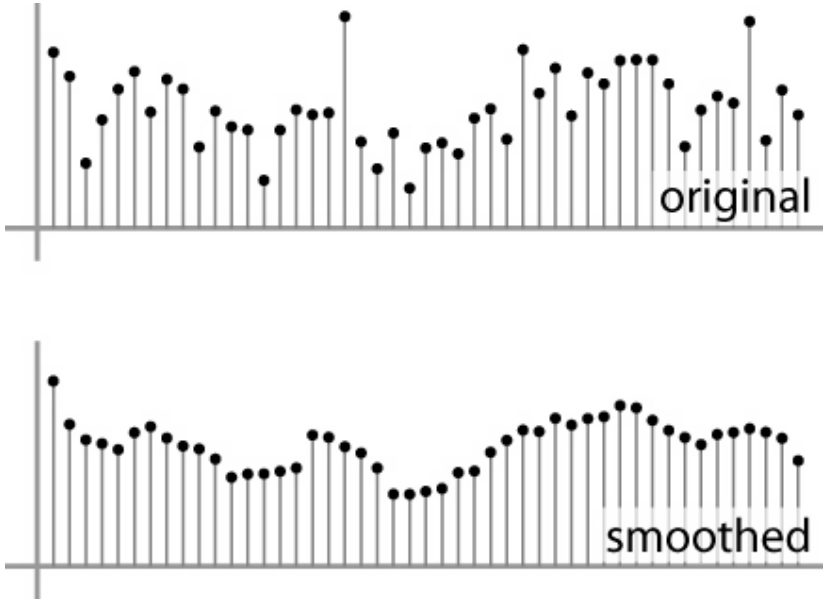
- Simple image processing
  - Greyscale
  - Brightness
  - Mirroring or “flipping”
- Filtering
  - **Linear filters: cross-correlation and convolution**
  - Gaussian filters
- Edge detection
  - Simple edge detector
  - Canny edge detector

Linear filtering: a key idea

- Transformations on signals; e.g.:
  - bass/treble controls on stereo
  - blurring/sharpening operations in image editing
  - smoothing/noise reduction in tracking
- Key properties
  - linearity: filter(f + g) = filter(f) + filter(g)
  - shift invariance: behavior invariant to shifting the input
    - delaying an audio signal
    - sliding an image around
- Can be modeled mathematically by *convolution*

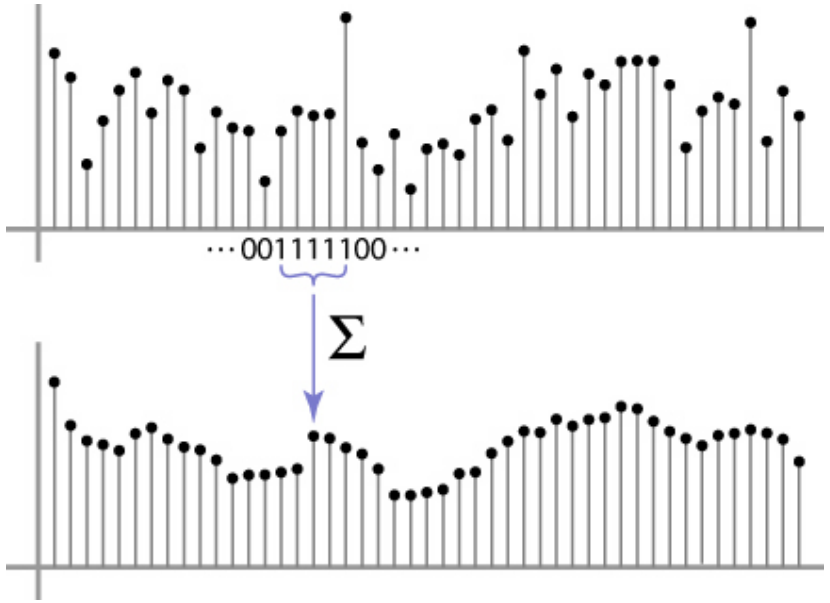
Moving Average

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing



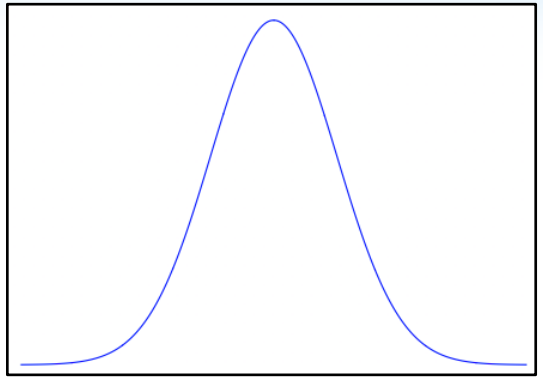
Weighted Moving Average

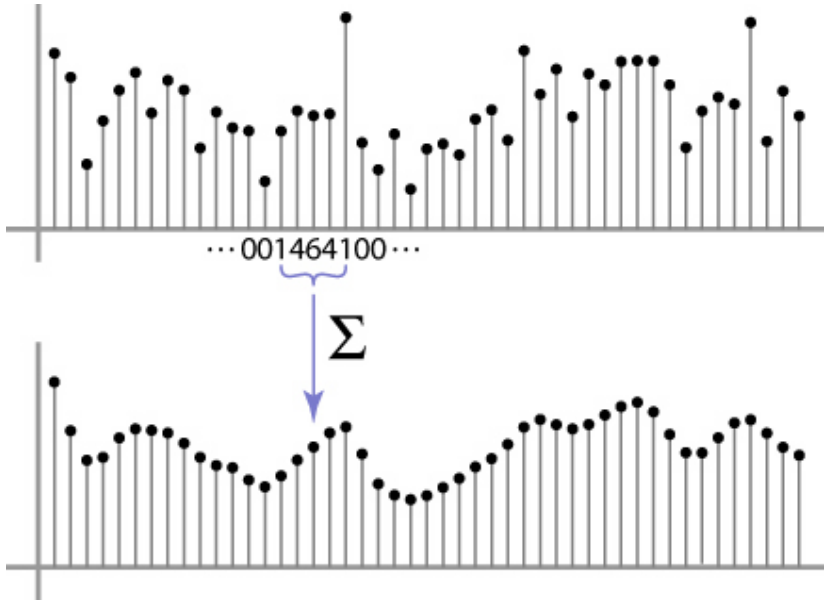
- Can add weights to our moving average
- *Weights* […, 0, 1, 1, 1, 1, 1, 0, …] / 5



Weighted Moving Average

- Bell curve (gaussian-like) weights […, 1, 4, 6, 4, 1, …]





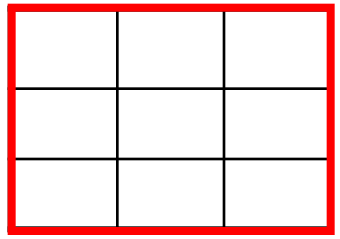
Moving Average In 2D

What are the weights H?

For uniform filter? (takes the mean)
For bell curve shaped filter?

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

Input image $F[x, y]$



$H[u, v]$

Slide by Steve Seitz

Cross-correlation filtering

- Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

- We can generalize this idea by allowing different weights for different neighboring pixels:

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

- This is called a **cross-correlation** operation and written:

$$G = H \otimes F$$

- H is called the "filter" or "kernel."

Slide by Steve Seitz

Gaussian filtering

- A Gaussian kernel gives less weight to pixels further from the center of the window

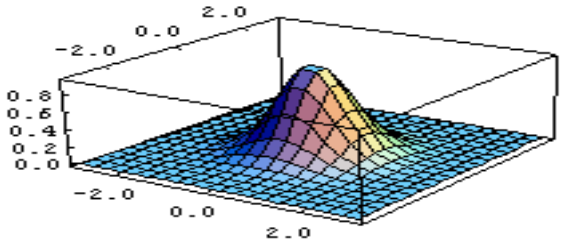
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

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

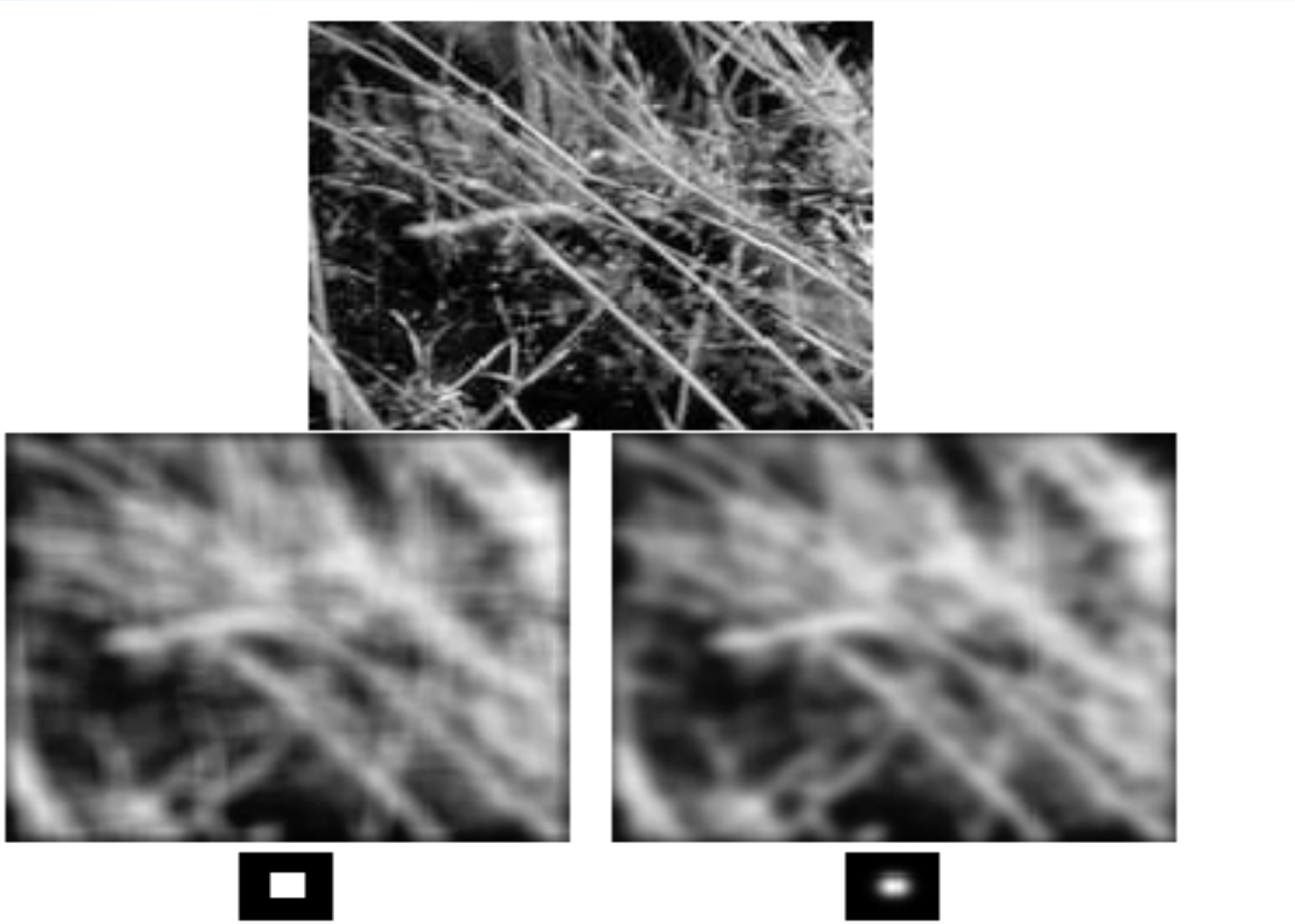
$H[u, v]$

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



Slide by Steve Seitz

Box Filter vs. Gaussian Filter



Slide by Steve Seitz

Convolution

- **Cross-correlation**:

$$G = H \otimes F$$

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

- **Convolution** is similar to cross-correlation, but the filter is flipped horizontally and vertically before being applied:

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

- It is written:

$$G = H * F$$

- Suppose H is a Gaussian or uniform (mean) kernel. How does convolution differ from cross-correlation?

Slide by Steve Seitz

Convolution is nice!

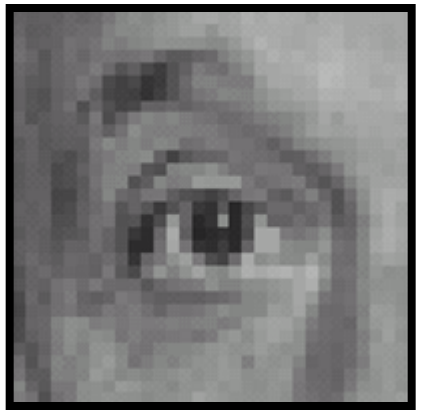
- Notation: $b = c * a$
- Convolution is a multiplication-like operation
  - commutative
  - associative
  - distributes over addition
  - scalars factor out
  - identity: unit impulse e = […, 0, 0, 1, 0, 0, …]

$$a * b = b * a$$
$$a * (b * c) = (a * b) * c$$
$$a * (b + c) = a * b + a * c$$
$$\alpha a * b = a * (\alpha b) = \alpha(a * b)$$
$$a * e = a$$

- Conceptually no distinction between filter and signal
- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Practice with linear filters

- Assume we are using **cross-correlation** filtering (filter is **not** flipped)



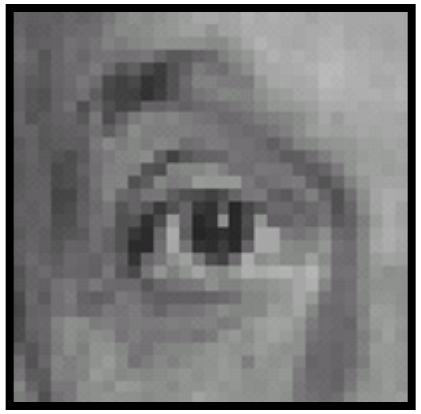
Original

0	0	0
0	1	0
0	0	0

?

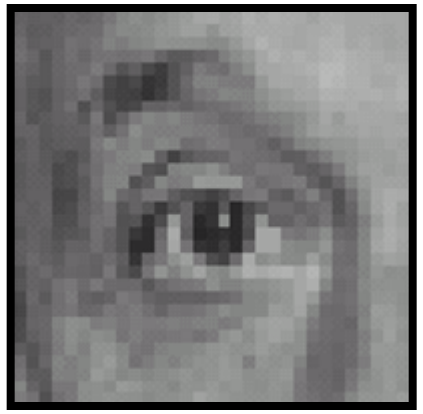
Source: D. Lowe

Practice with linear filters



Original

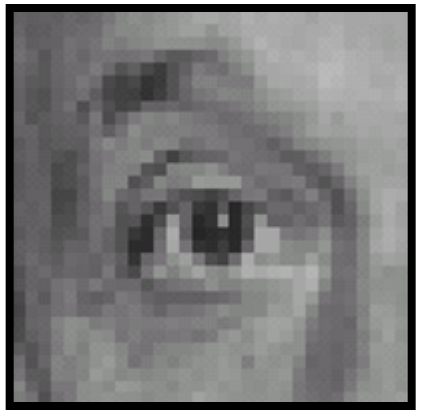
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



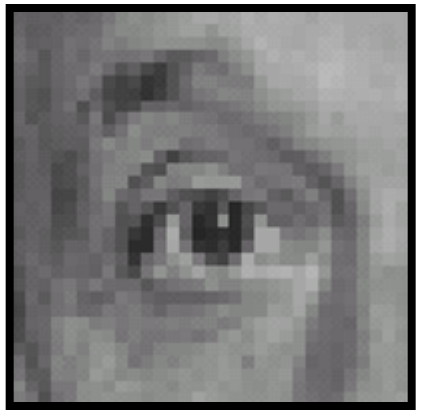
Original

0	0	0
0	0	1
0	0	0

?

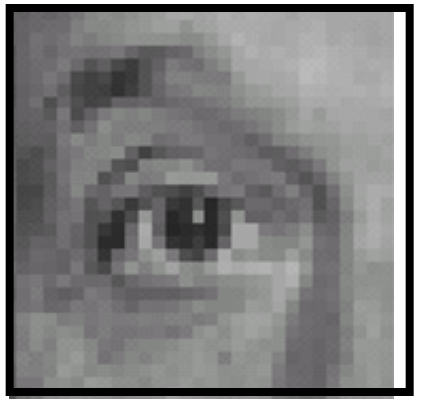
Source: D. Lowe

Practice with linear filters



Original

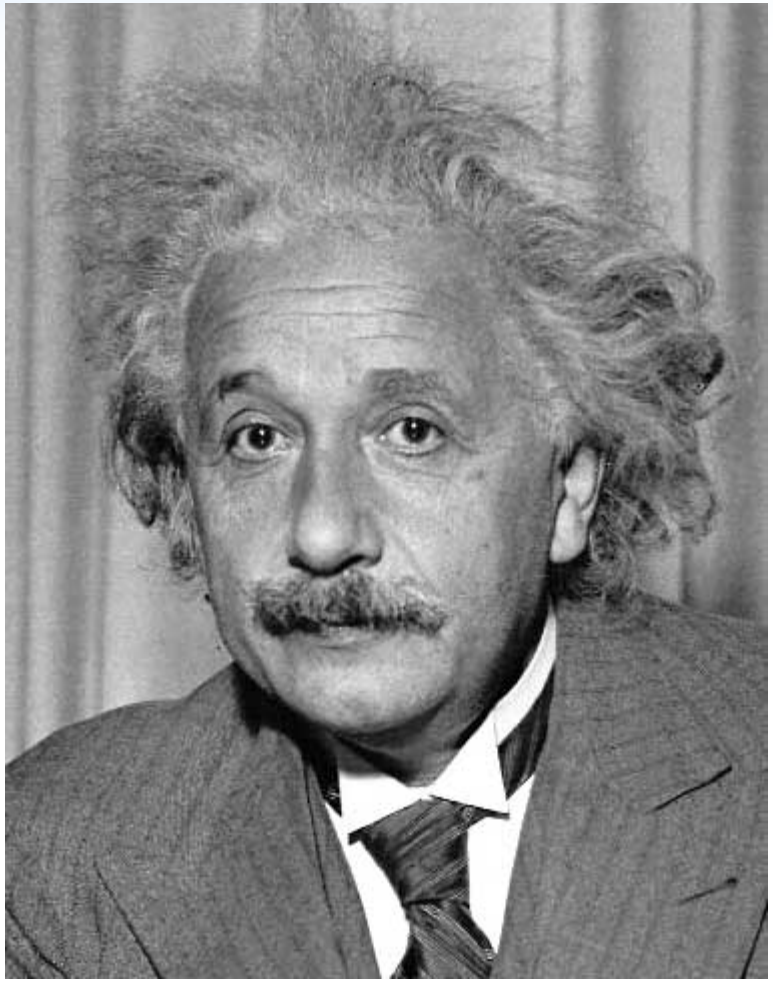
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

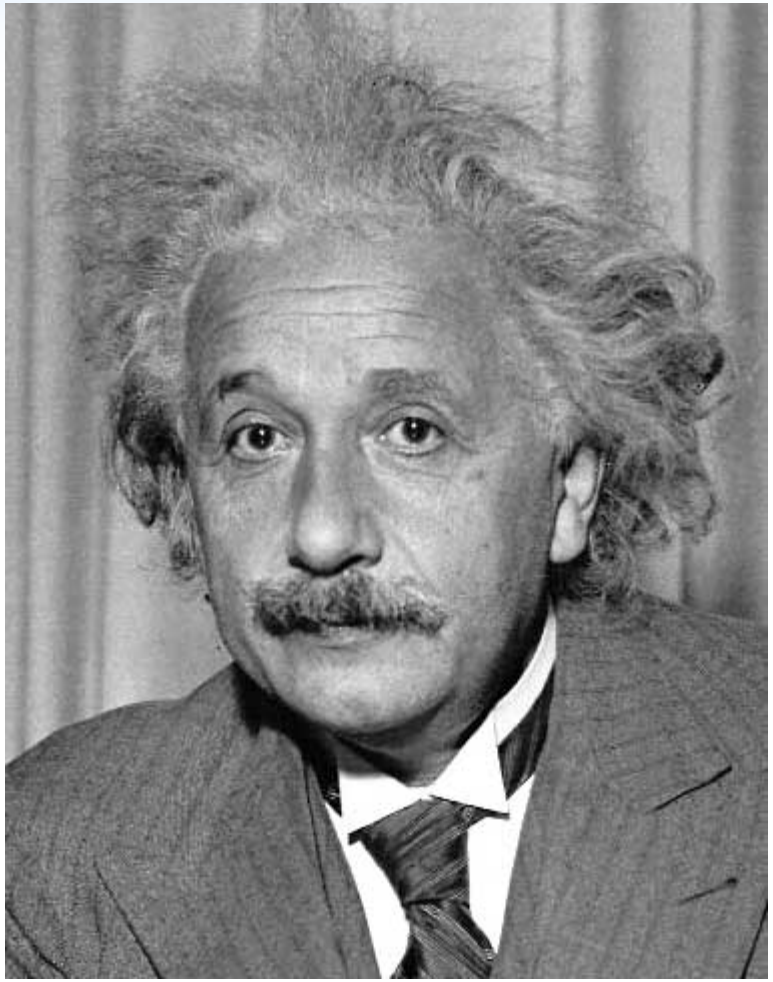


1	0	-1
2	0	-2
1	0	-1

Sobel

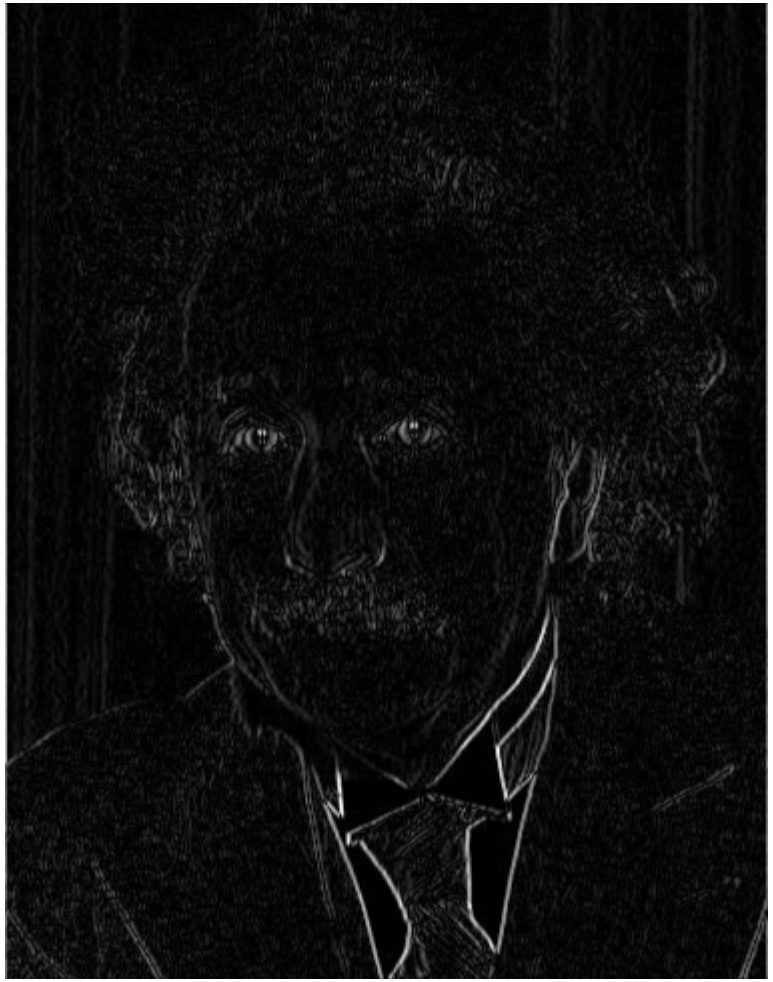
?

Practice with linear filters



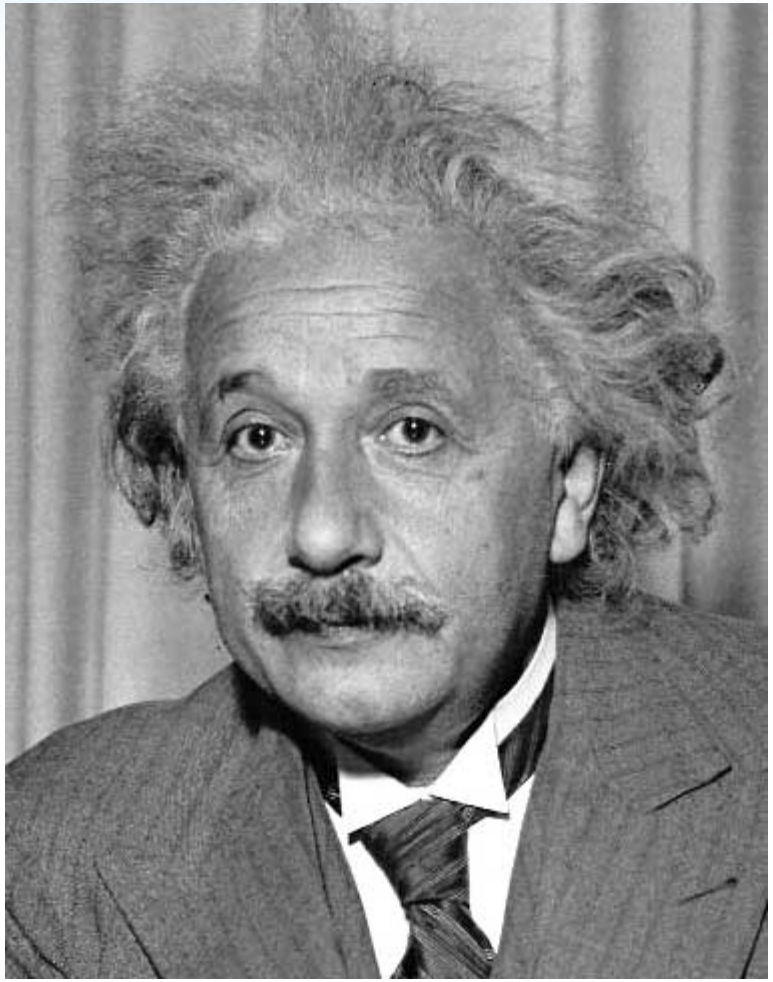
1	0	-1
2	0	-2
1	0	-1

Sobel



Vertical Edge
(absolute value)

Practice with linear filters

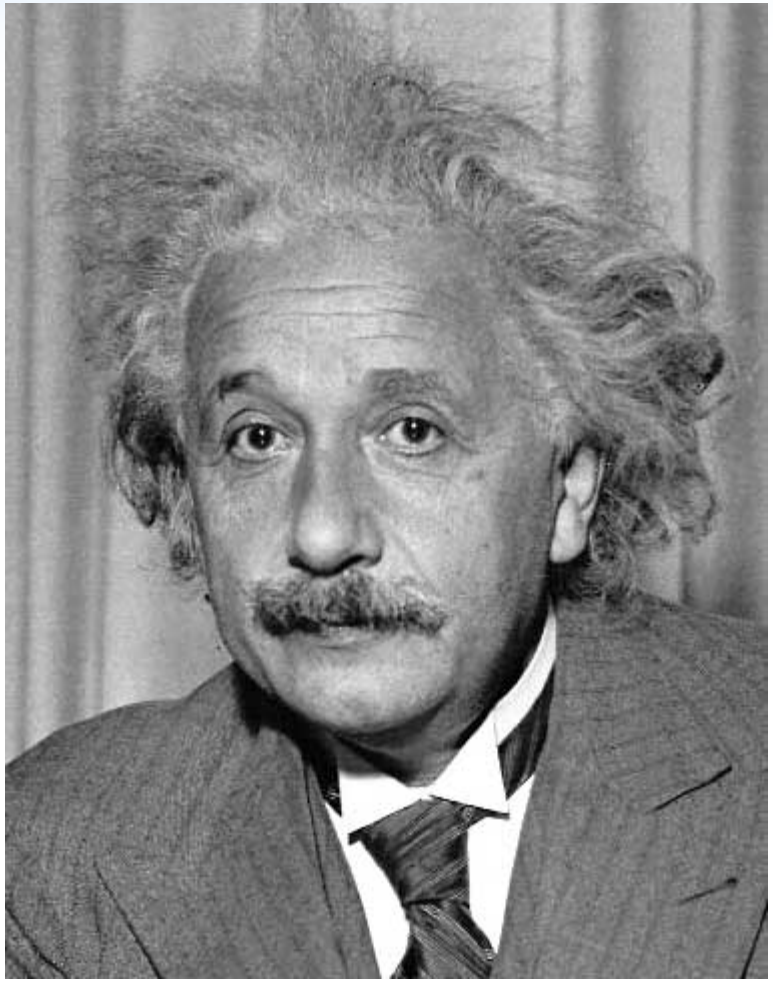


1	2	1
0	0	0
-1	-2	-1

Sobel

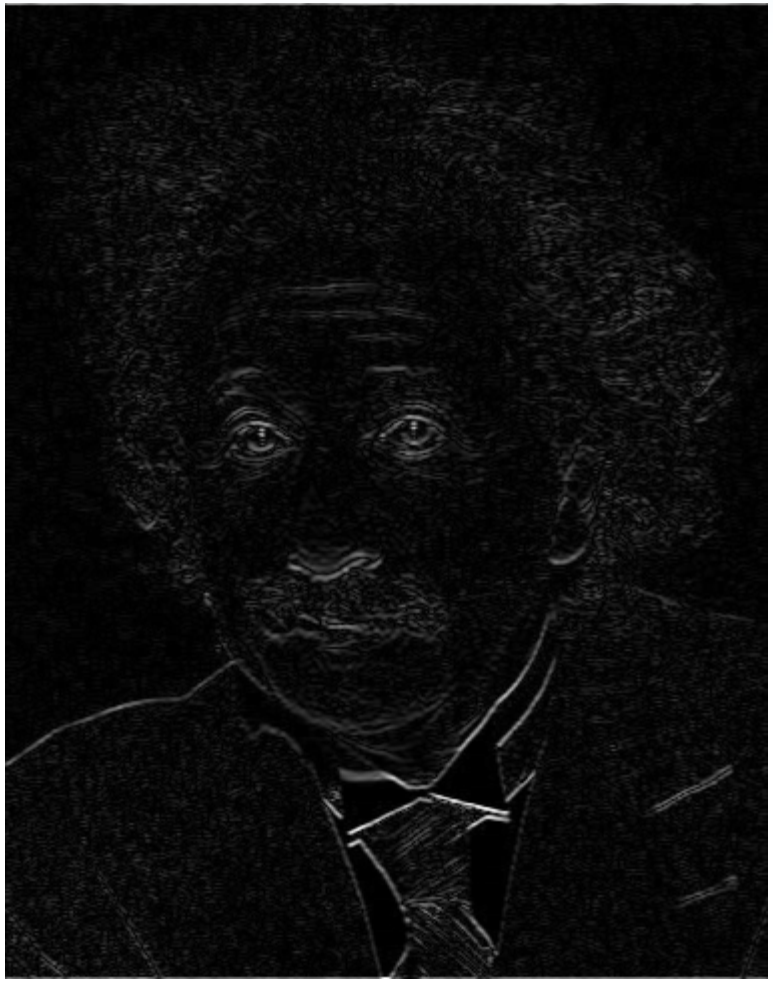
?

Practice with linear filters



1	2	1
0	0	0
-1	-2	-1

Sobel



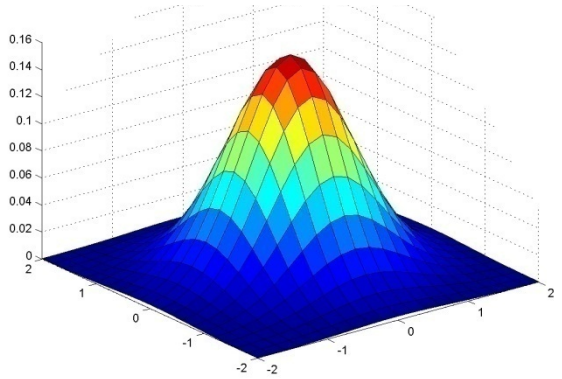
Horizontal Edge
(absolute value)

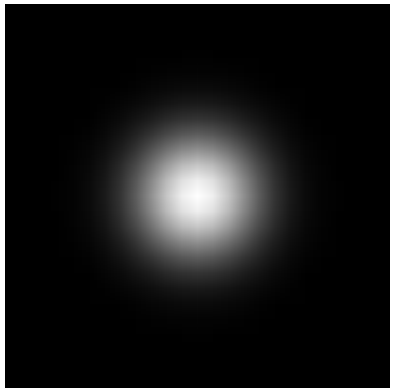
Outline

- Simple image processing
  - Greyscale
  - Brightness
  - Mirroring or “flipping”
- Filtering
  - Linear filters: cross-correlation and convolution
  - **Gaussian filters**
- Edge detection
  - Simple edge detector
  - Canny edge detector

Important linear filter: Gaussian

- Weight contributions of neighboring pixels by nearness





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

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

- Same shape in spatial and frequency domain (Fourier transform of Gaussian is Gaussian)

Slide credit: Christopher Rasmussen

Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

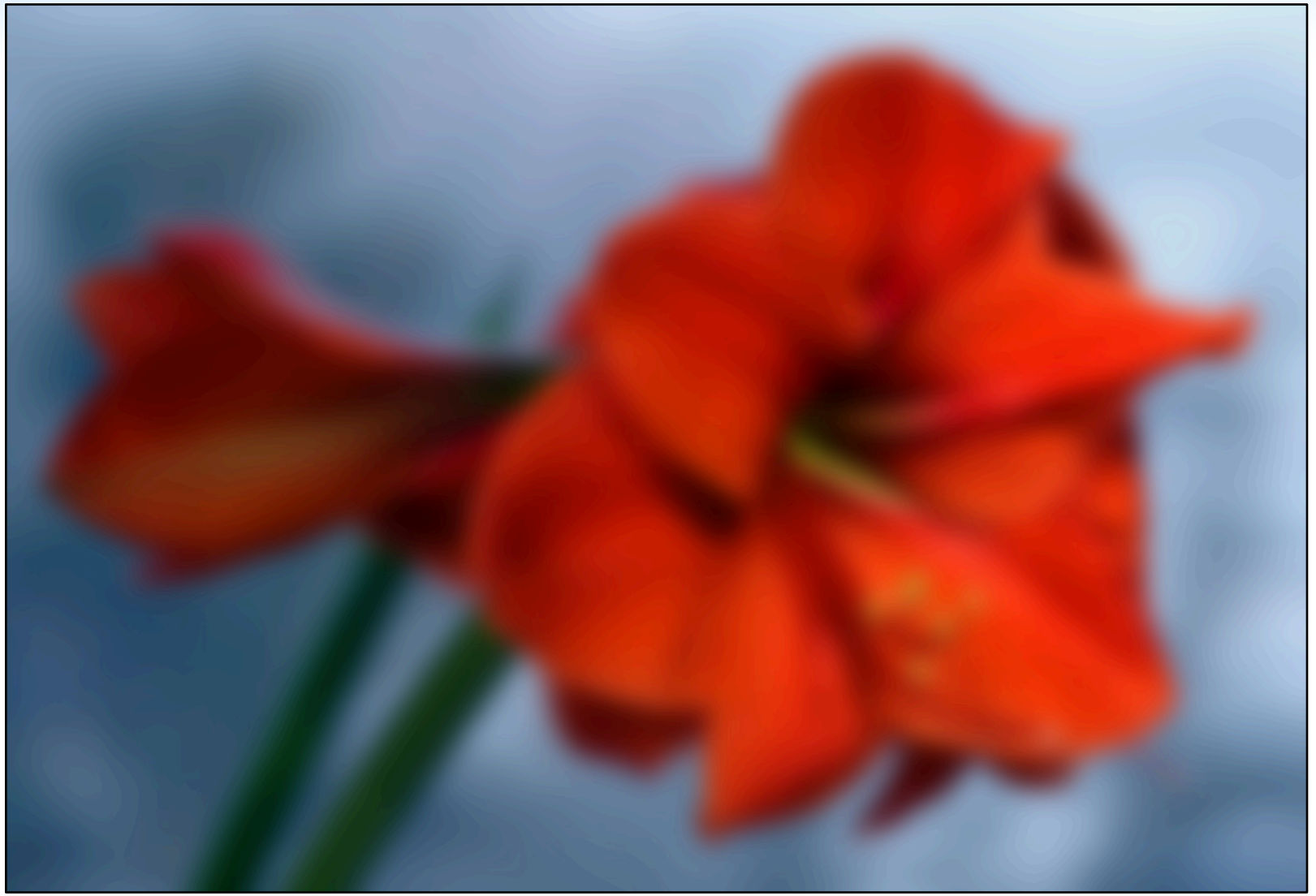
Gaussian filters

Input image (2048 x 1397)

Gaussian filters

Gaussian filtered (σ=5)

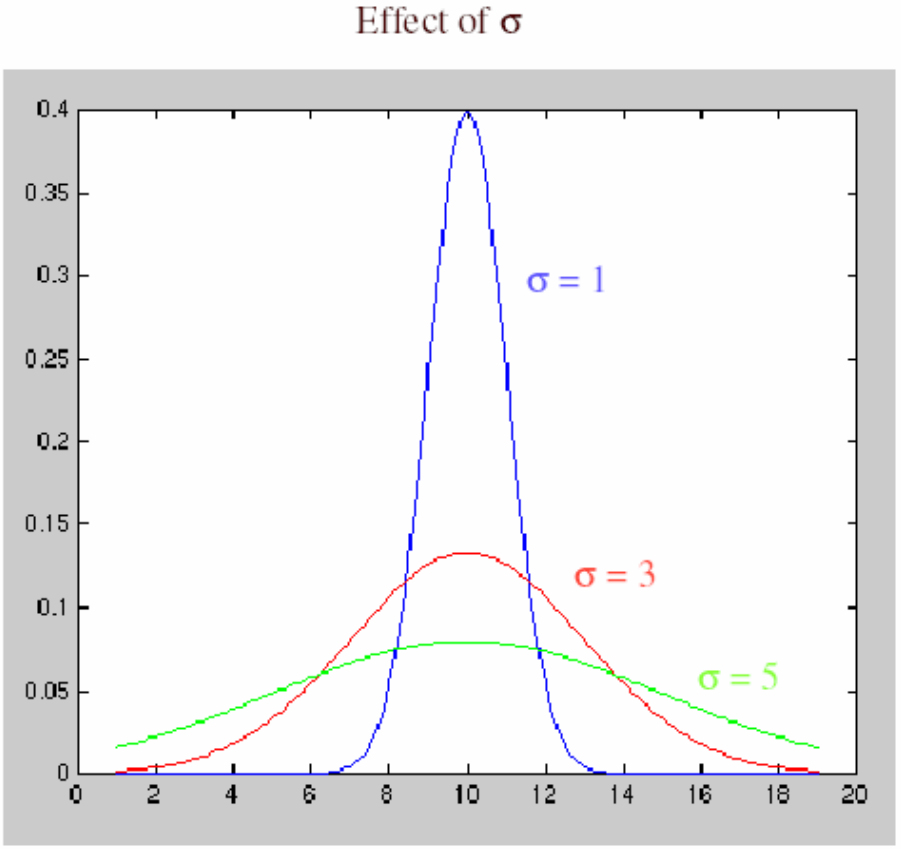
Gaussian filters



Gaussian filtered (σ=20)

Practical matters

- How big should the filter be?
  - Values at edges should be near zero
  - Rule of thumb for Gaussian: set filter half-width to about 3 σ
  - Normalize truncated kernel. Why?



Side by Derek Hoiem

Separable Filters

- Some kernels *K* can be written:

$$K = H * V, \qquad H \text{ is horizontal, } V \text{ is vertical}$$

- Example: 2D Gaussian

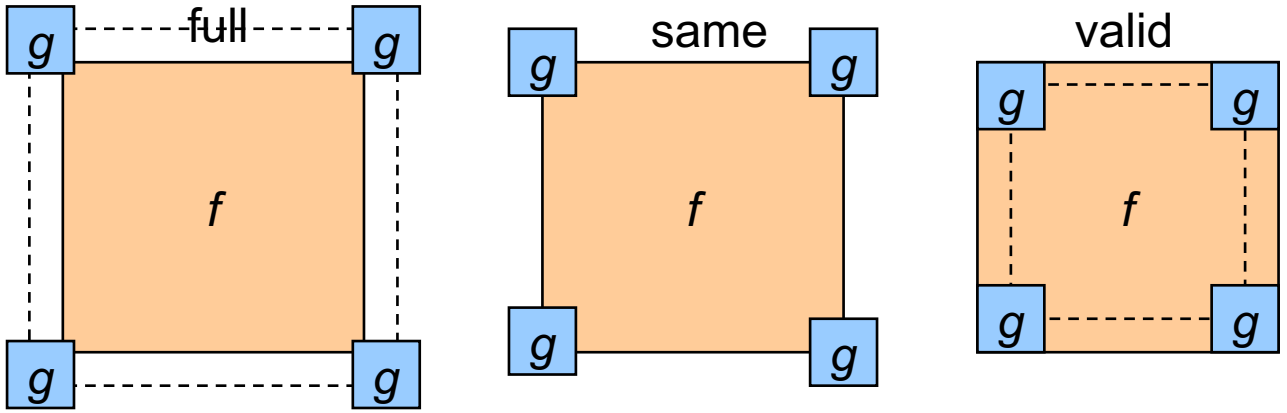
$$G(x,y) = \frac{1}{Z}\exp\left[\frac{-(x^2+y^2)}{2\sigma^2}\right]$$

$$= \frac{1}{Z}\exp\left[\frac{-x^2}{2\sigma^2}\right]\exp\left[\frac{-y^2}{2\sigma^2}\right] = \frac{1}{Z}H(x)V(y)$$

- Filter first by *H* then *V* (or vice versa)
- Why is this useful?

Size of Output

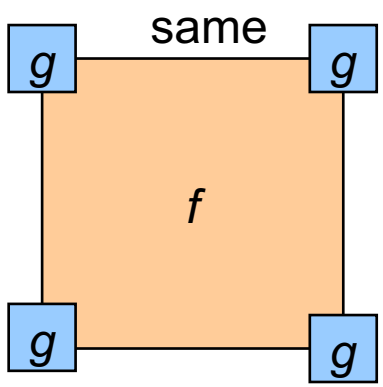
- MATLAB: conv2(g,f,shape)
- Python: scipy.signal.convolve2d(g,f,shape)
  - *shape* = ‘full’: output size is sum of sizes of f and g
  - *shape* = ‘same’: output size is same as f
  - *shape* = ‘valid’: output size is difference of sizes of f, g
- Easier for color images: scipy.ndimage.filters.convolve(g,f)



Source: S. Lazebnik

Python convolution (with SciPy)

- Python: scipy.signal.convolve2d(g,f,shape)
  - Convolves 2D images (e.g. greyscale)
- Python: scipy.ndimage.filters.convolve(g,f)
  - Convolves n-D images (e.g. greyscale, color)
  - But always uses ‘same’ size output
  - Can specify how to handle out of bounds pixels (e.g. ‘constant’, ‘reflect’)



Source: S. Lazebnik

Demo in Python

- Python (Jupyter) Notebook

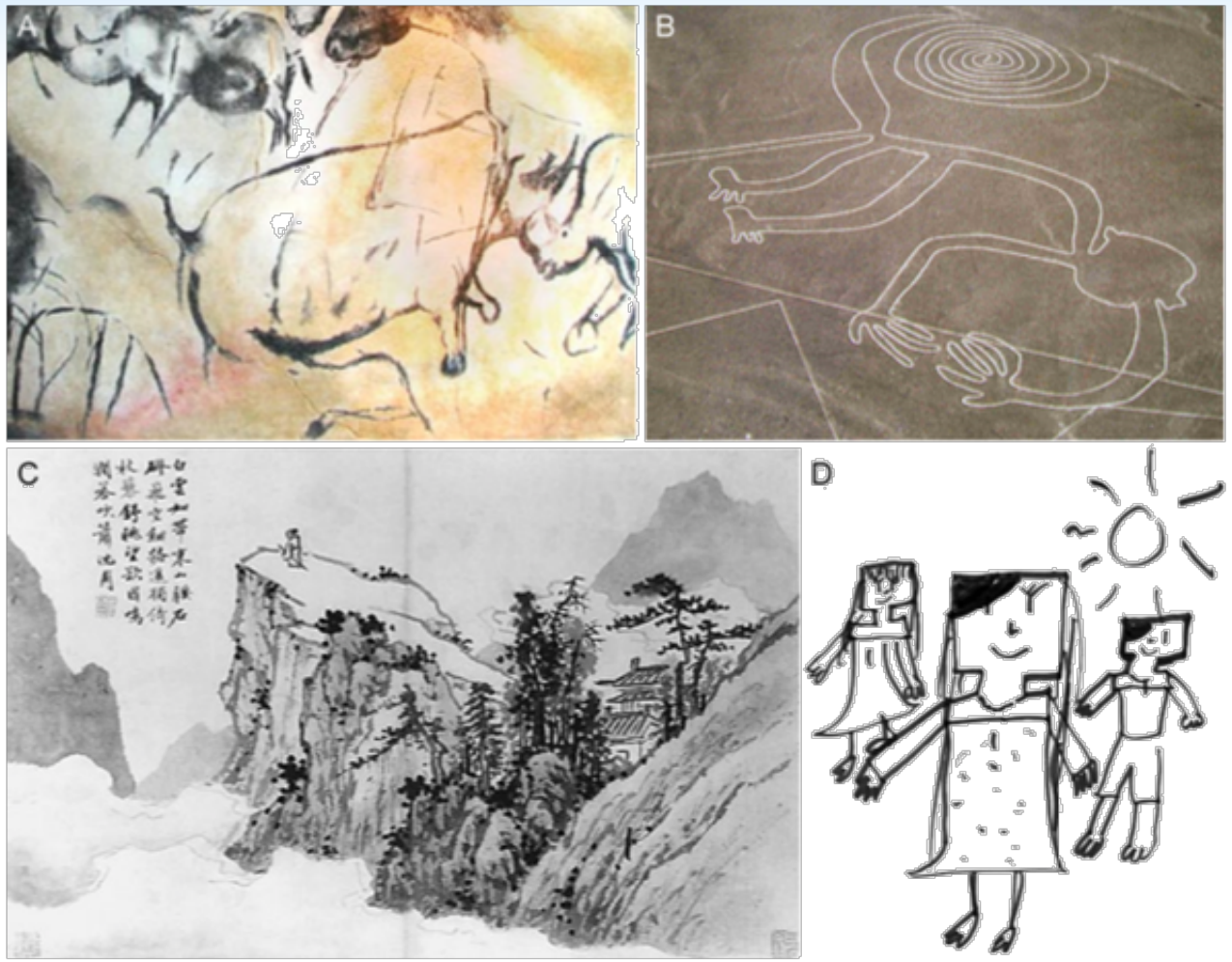
Python Environment for Programming Assignments

- Recommend Python (another option: MATLAB?)
- Set up Python (recommend Anaconda Python)
- Already included packages: SciPy, matplotlib, scikit-image.
- Recommend:
  - tensorflow (neural networks), ideally configured with GPU support
  - keras (neural networks)
- Can sign up to use department machines also,
  will send document for that.

Outline

- Simple image processing
  - Greyscale
  - Brightness
  - Mirroring or “flipping”
- Filtering
  - Linear filters: cross-correlation and convolution
  - Gaussian filters
- **Edge detection**
  - Simple edge detector
  - Canny edge detector

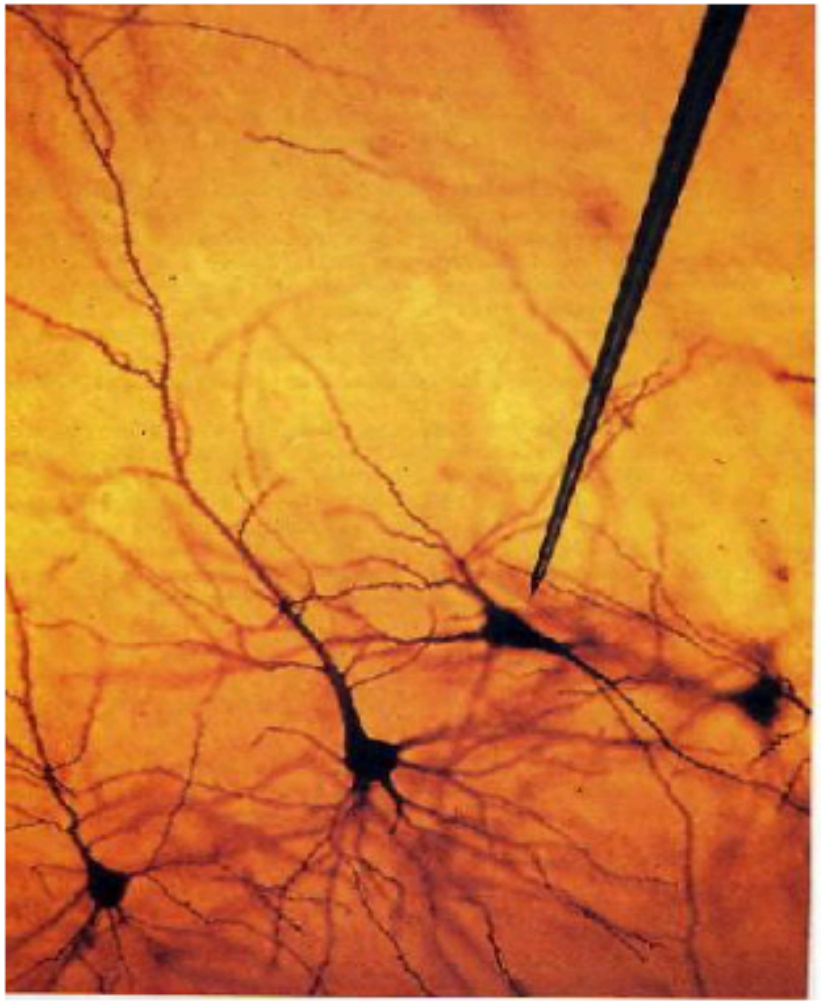
Edge Detection

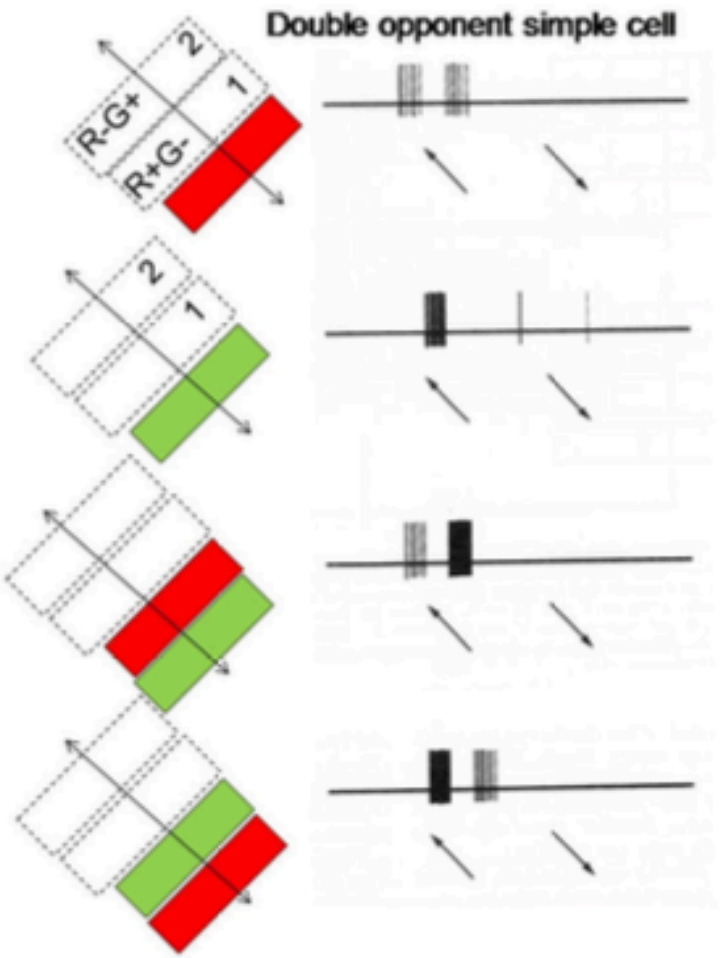


(A) Cave painting at Chauvet, France, about 30,000 B.C.;
(B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
(C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
(D) Line drawing by 7-year old I. Lleras (2010 A.D.).

Slide from Fei Fei Li, Juan Carlos Niebles

Edge Detection: Mammal Vision





Hubel & Wiesel, 1960s

Slide from Fei Fei Li, Juan Carlos Niebles

Edge Detection: Human Vision

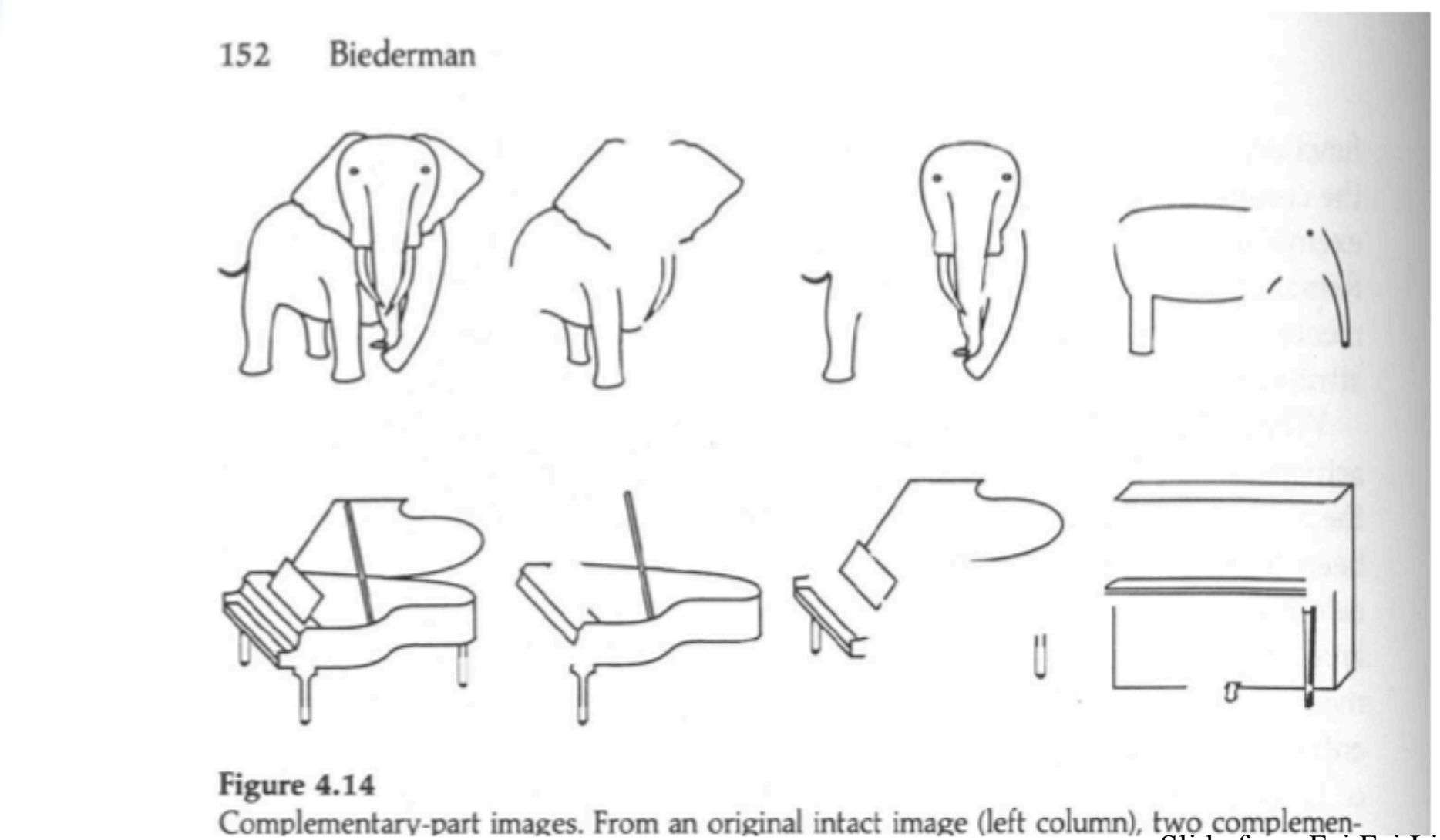
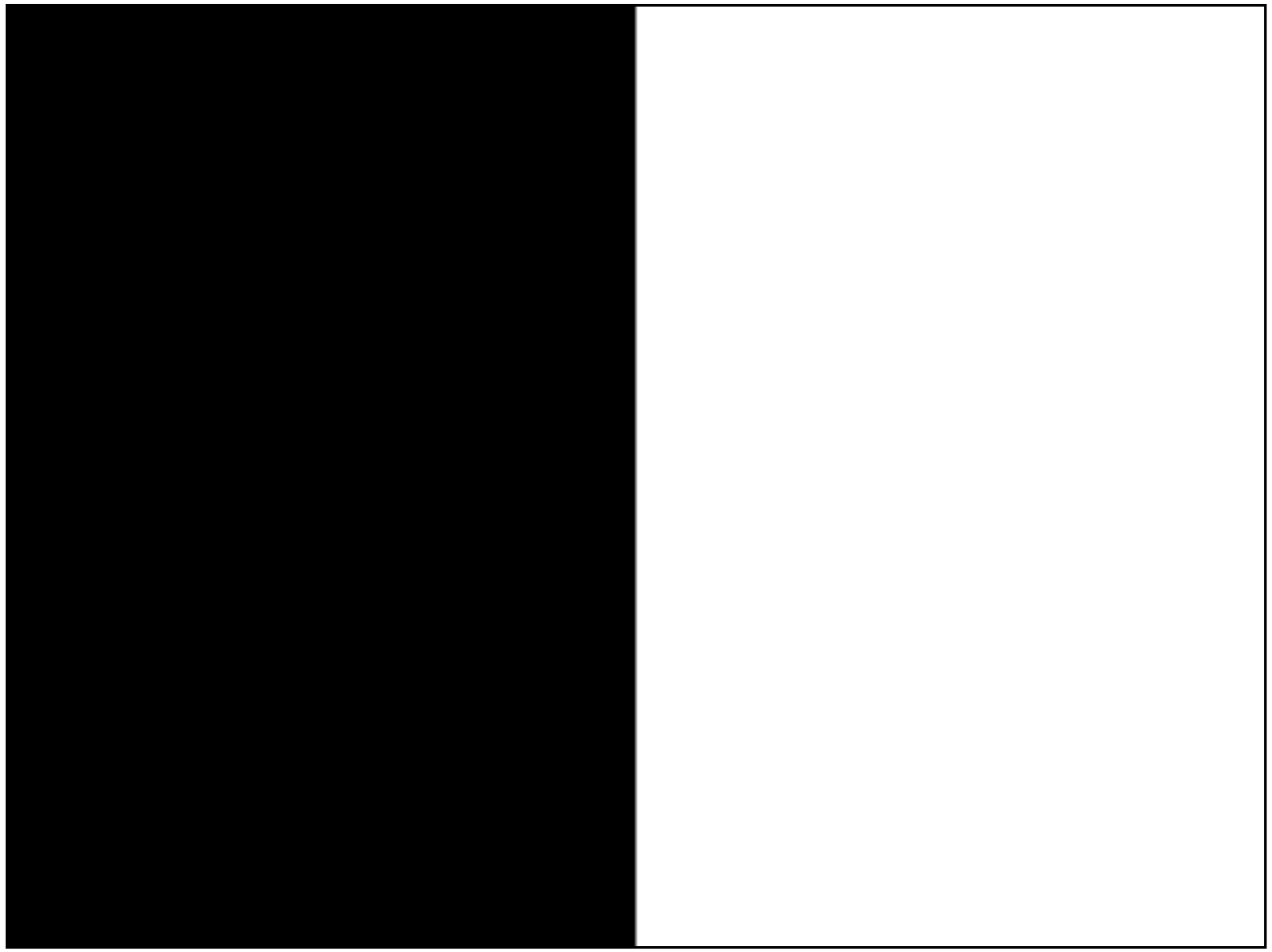


Figure 4.14
Complementary-part images. From an original intact image (left column), two complemen-

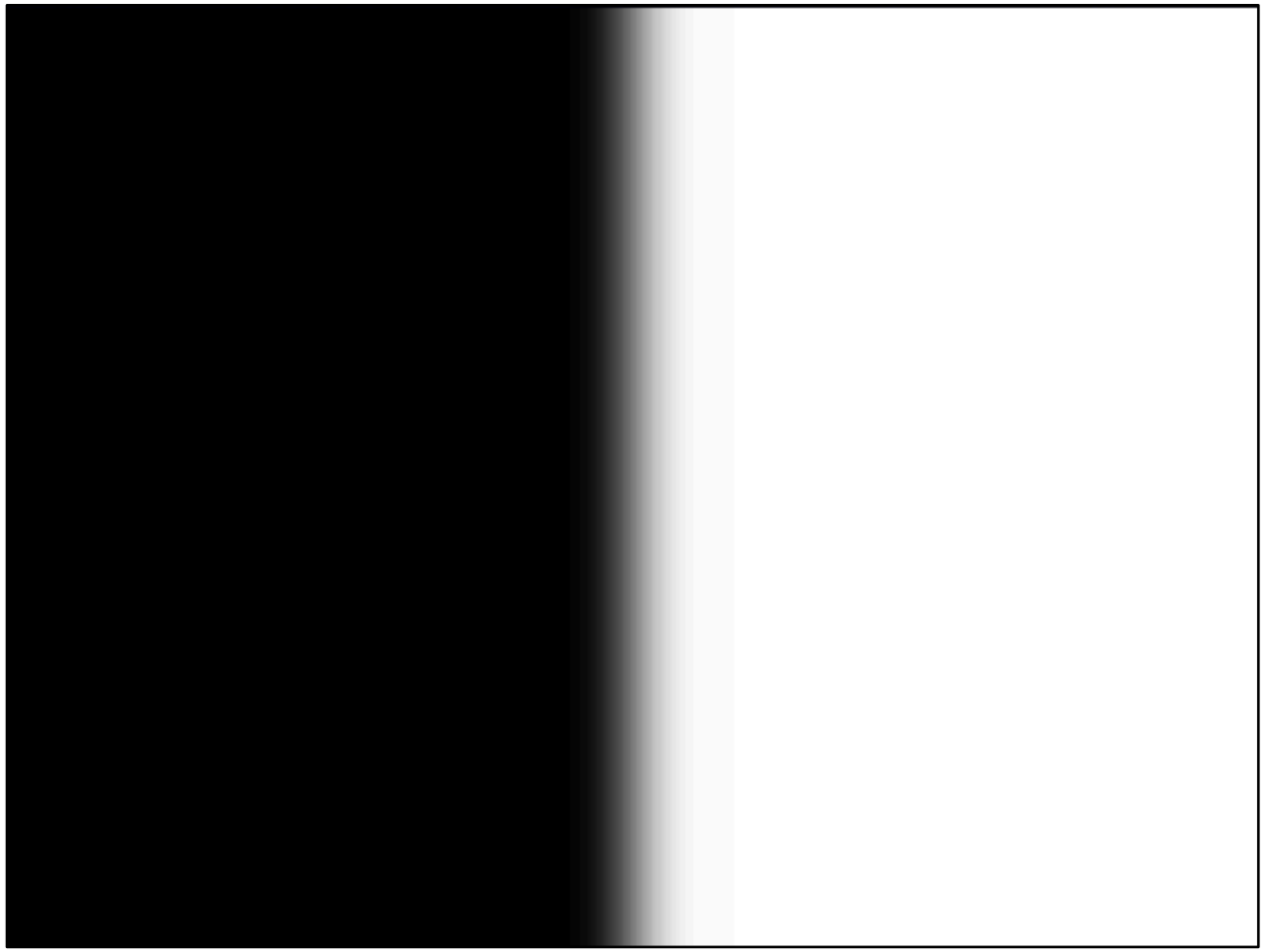
Slide from Fei Fei Li, Juan Carlos Niebles

What is an Edge?



Slide from Jason Lawrence

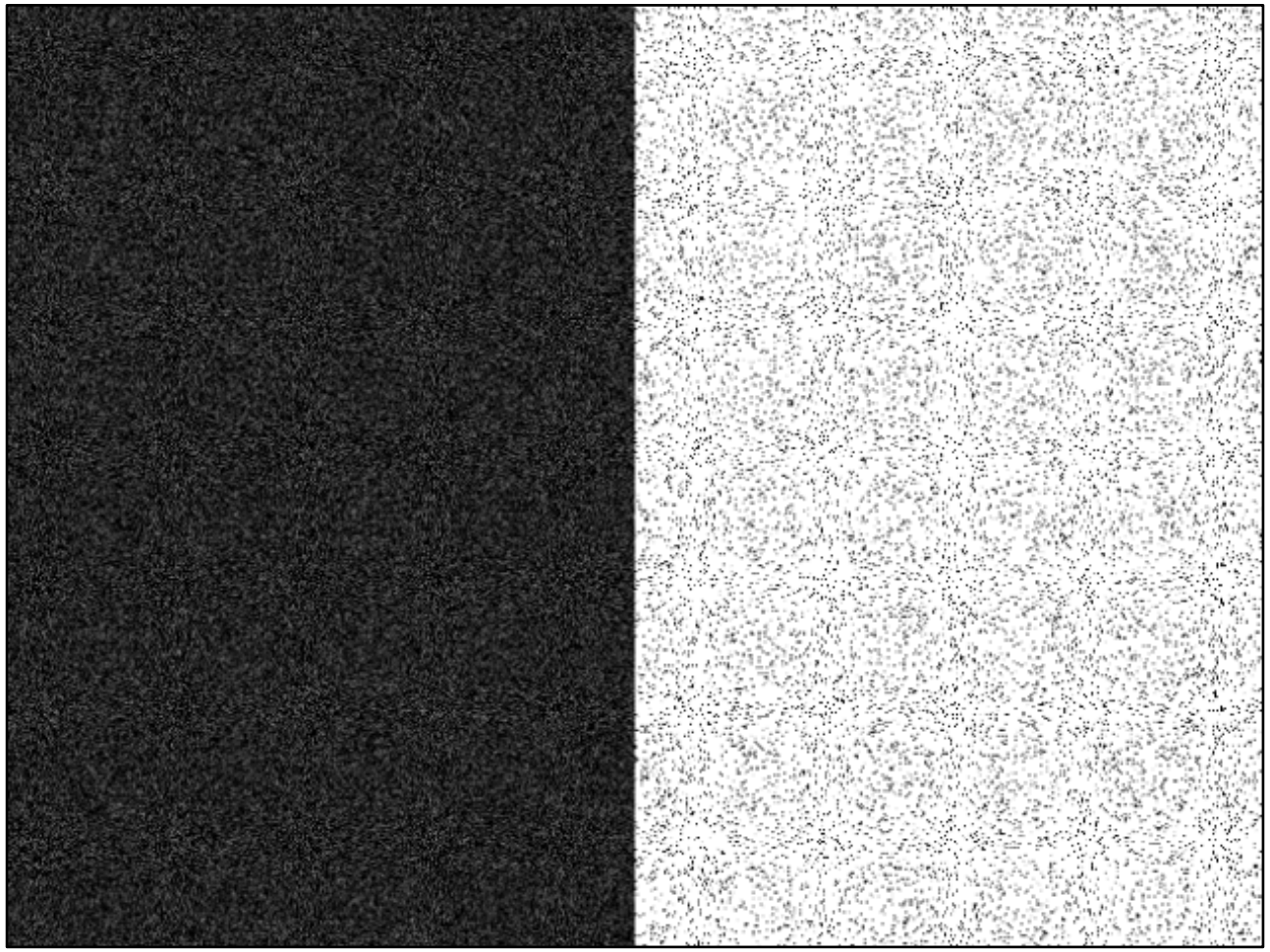
What is an Edge?



Challenge: blur

Slide from Jason Lawrence

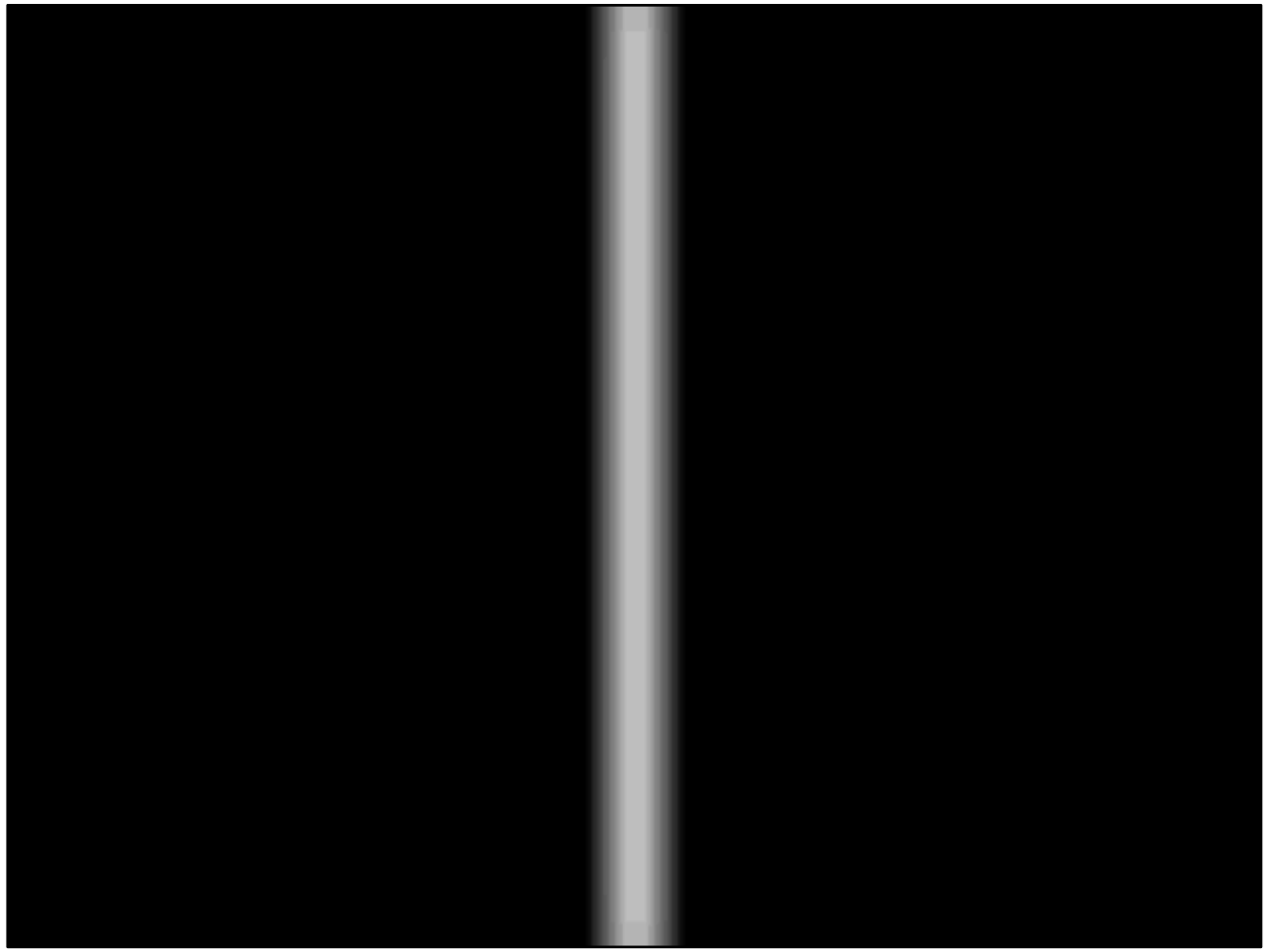
What is an Edge?



Challenge: noise

Slide from Jason Lawrence

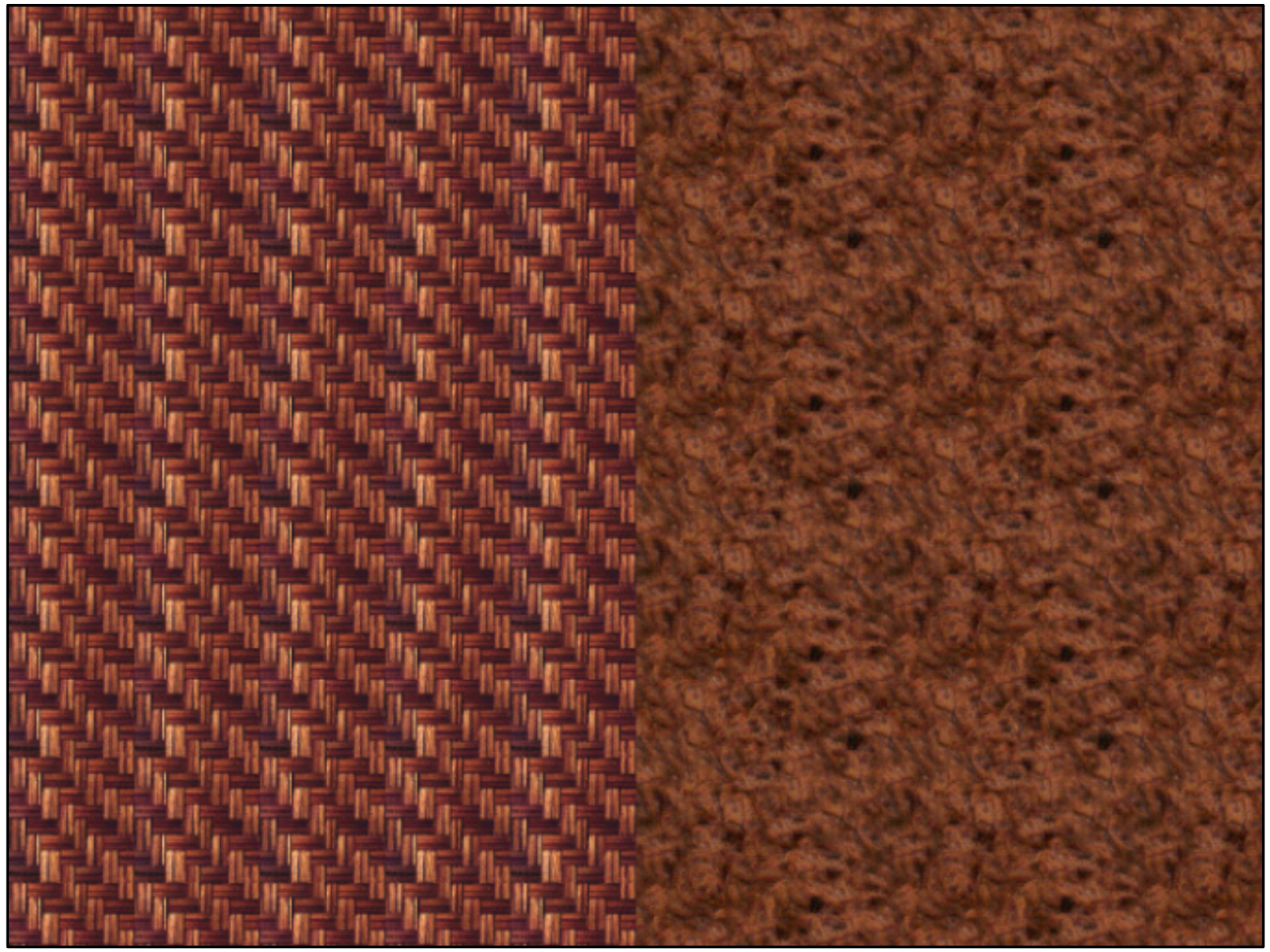
What is an Edge?



Is this one edge or two?

Slide from Jason Lawrence

What is an Edge?

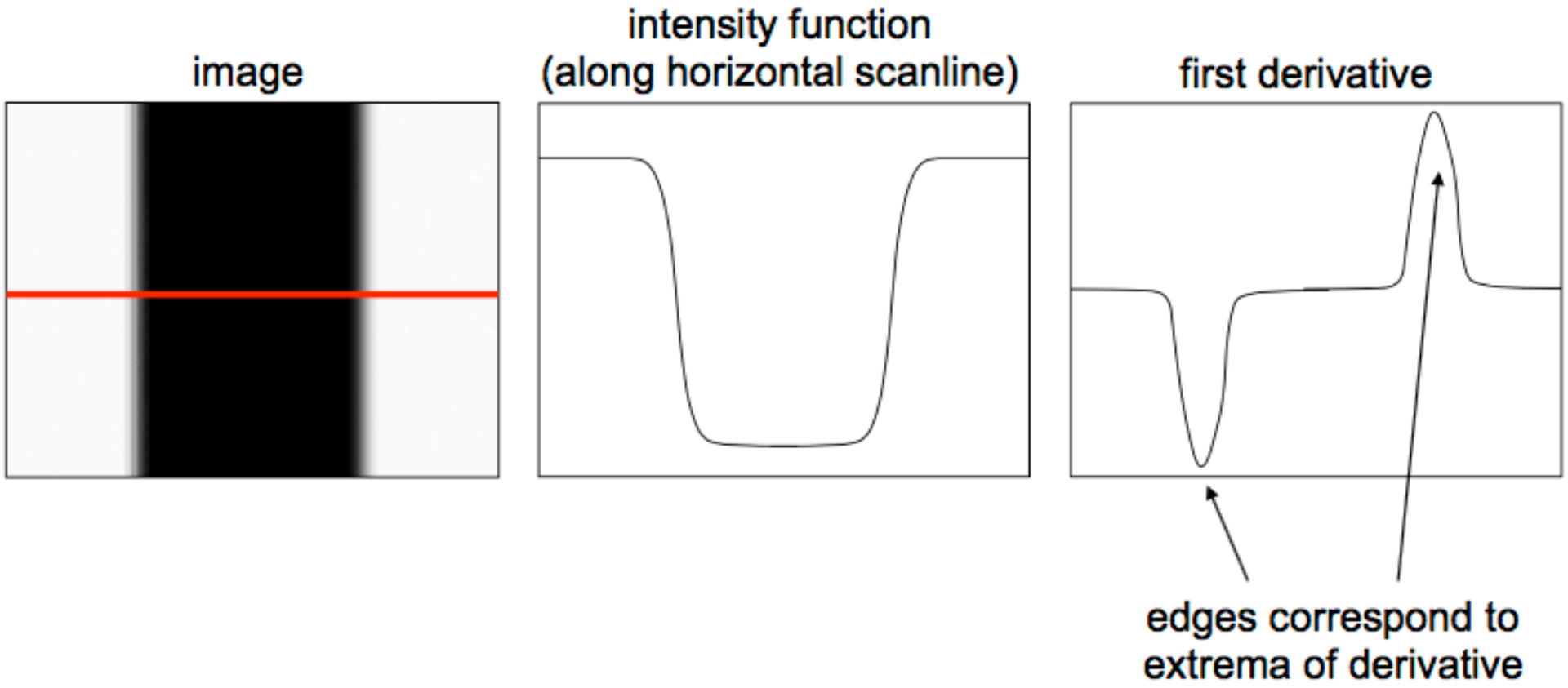


Where are the edges?

Slide from Jason Lawrence

Characterizing Edges

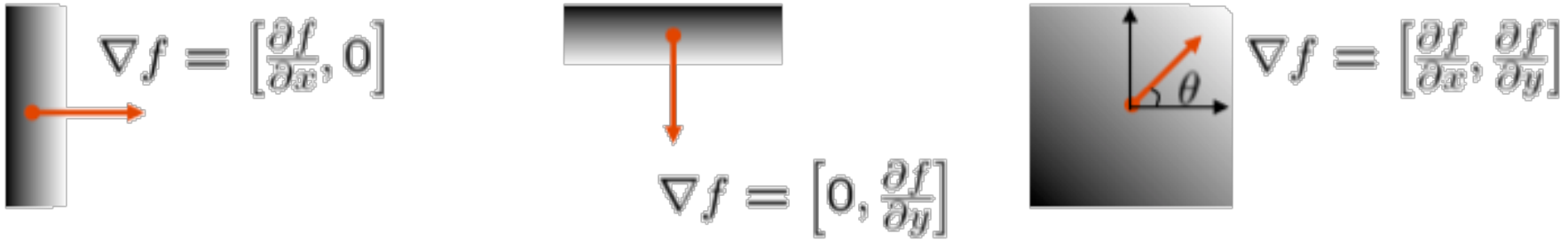
- An **edge** is a place of rapid change in the image intensity function.



Slide from Fei Fei Li, Juan Carlos Niebles

Image Gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



The gradient vector points in the direction of most rapid increase in intensity

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

- how does this relate to the direction of the edge?

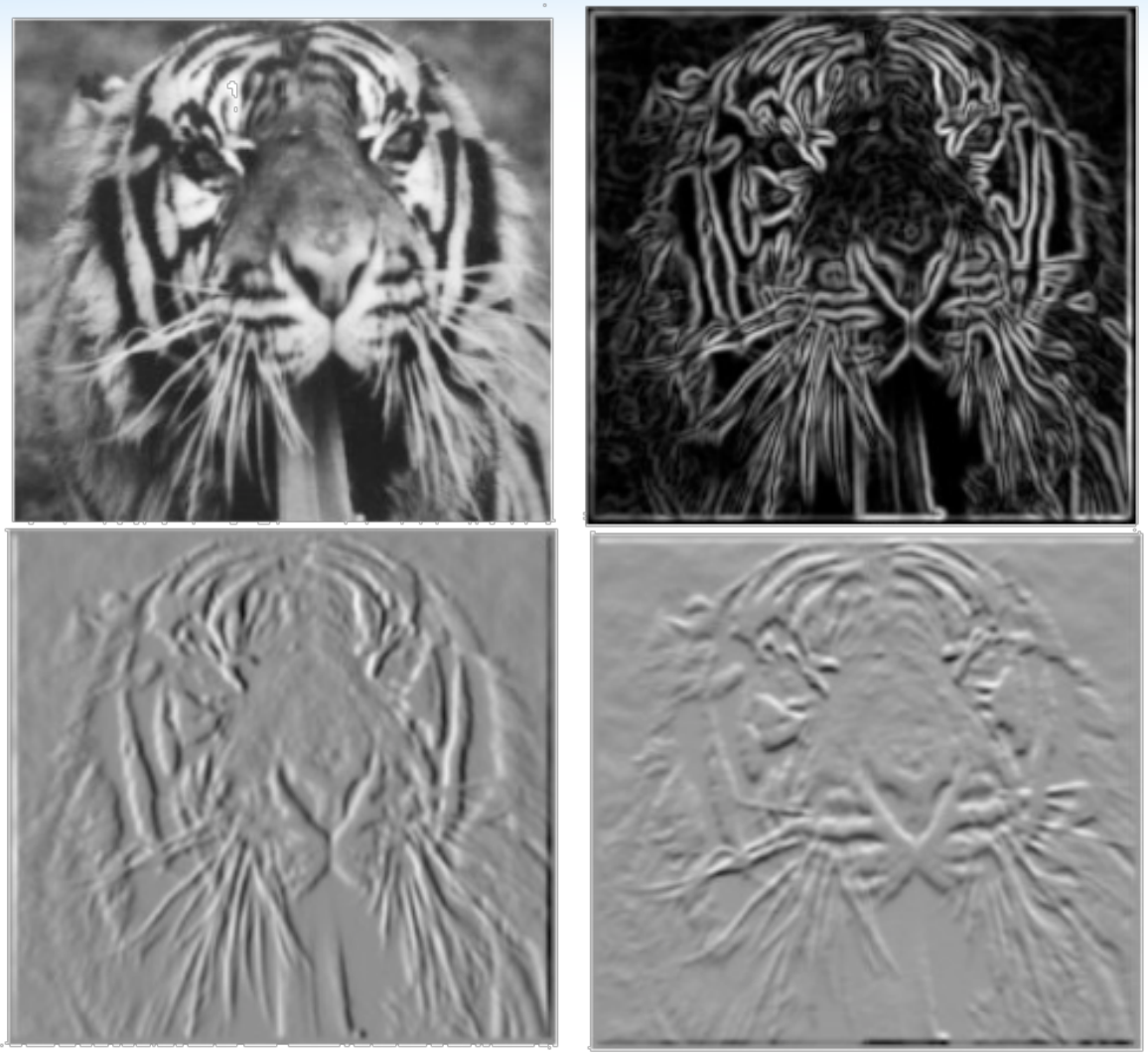
The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Image Gradient

Original Image



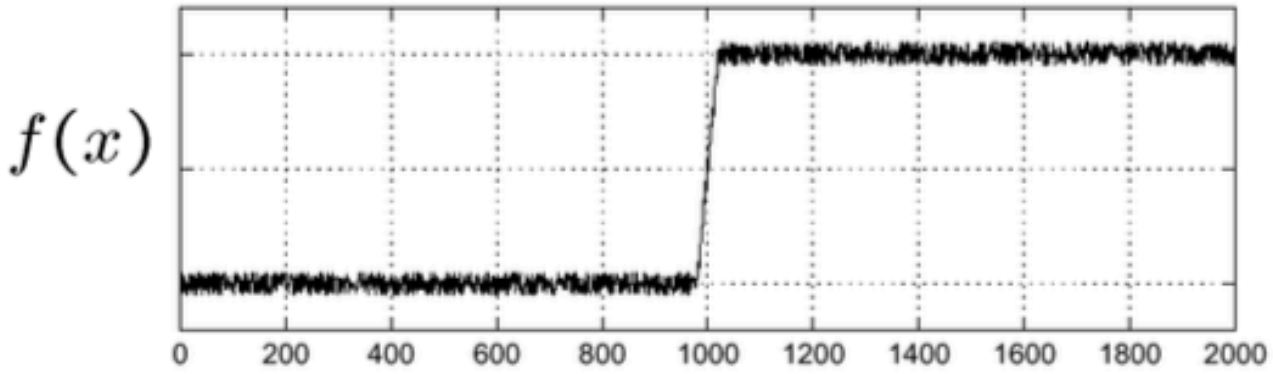
Gradient magnitude

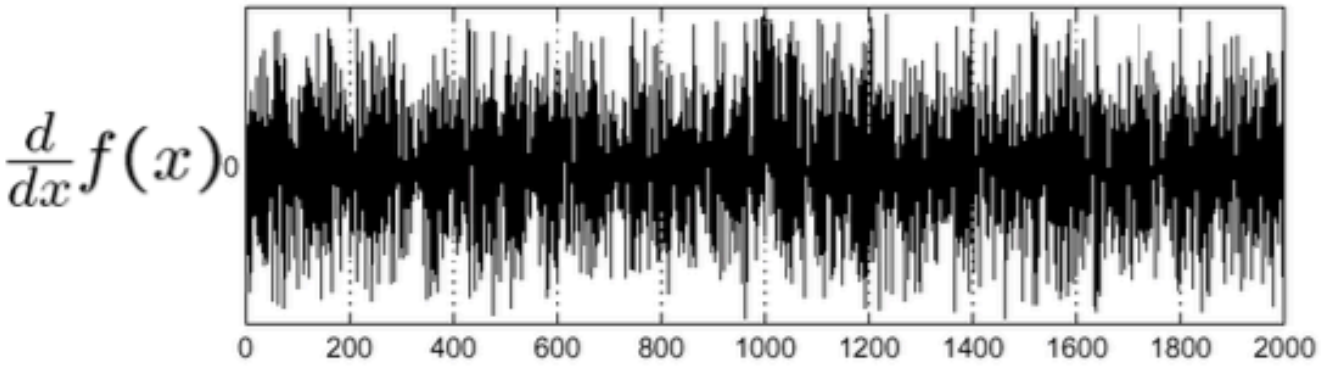
- Which one is the gradient in the x-direction? How about y-direction?

Slide from Fei Fei Li, Juan Carlos Niebles

Effects of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

Source: S. Seitz

Simple Edge Detector

Algorithm:

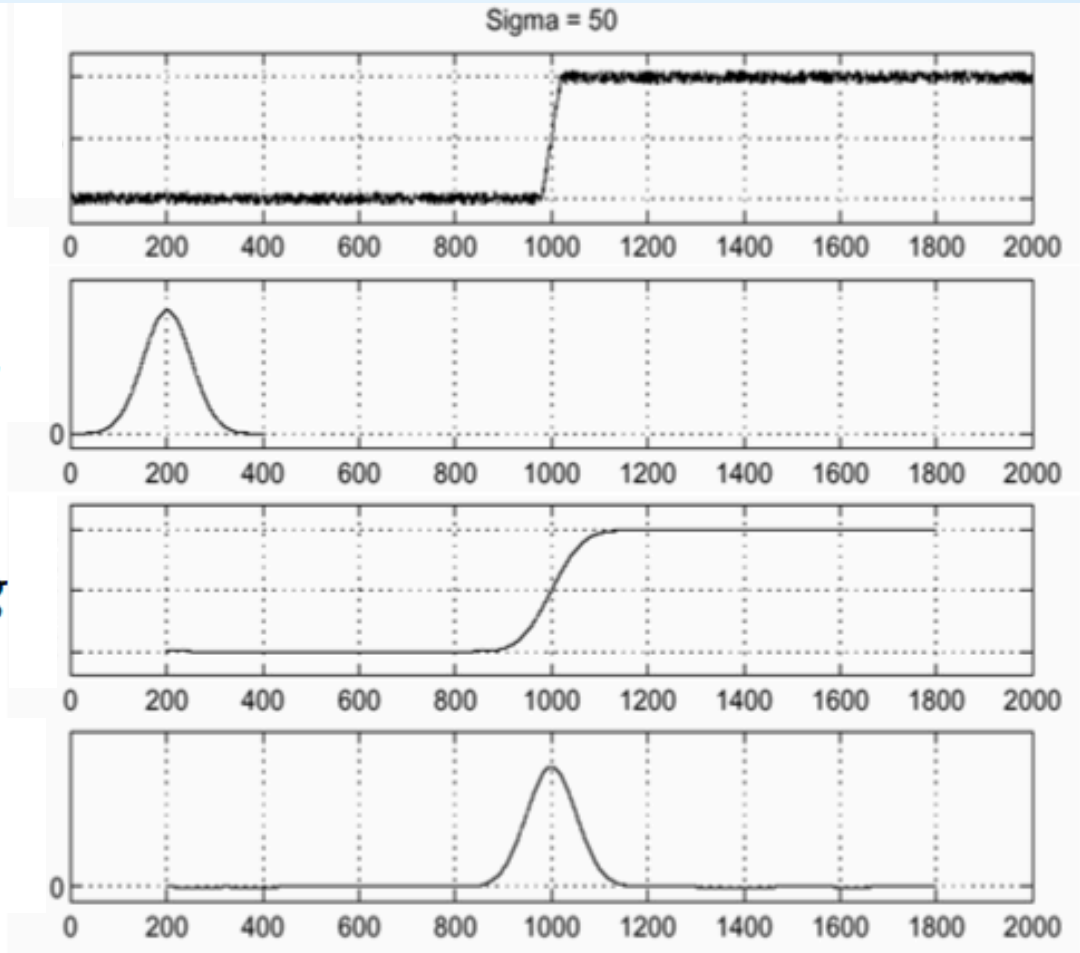
1. Blur using Gaussian filter
2. Find gradient magnitude

Input image: f

Gaussian kernel: g

Blurred image: $f * g$

Gradient: $\frac{d}{dx}(f * g)$



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Slide from Steve Seitz

Derivatives of Filters

- Can optionally combine the blurring and differentiation steps using the theorem:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

From Steve Seitz

Derivatives of Filters

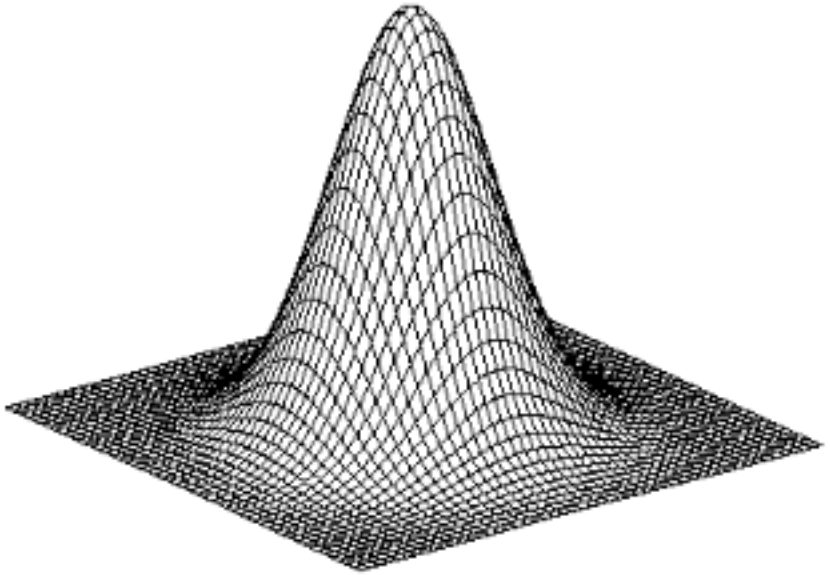
- Can optionally combine the blurring and differentiation steps using the theorem:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- **Algorithm 2 for simple edge detector**:
  1. Convolve with x derivative of Gaussian, gives E_x
  2. Convolve with y derivative of Gaussian, gives E_y
  3. Find gradient magnitude: E = $|| E_x^2 + E_y^2 ||$

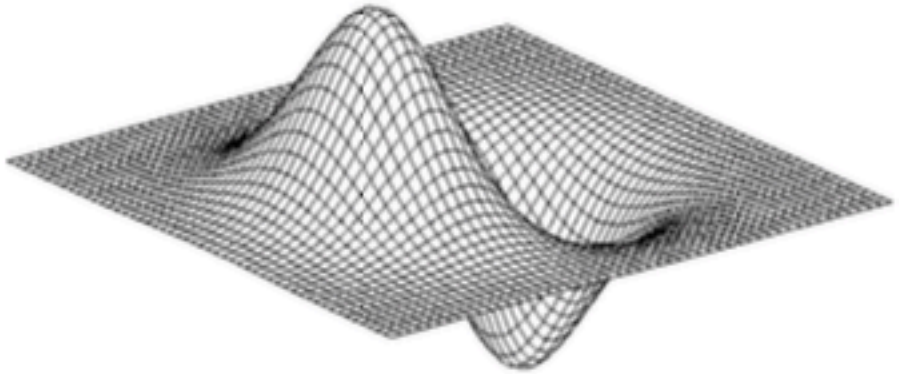
From Steve Seitz

Derivatives of Gaussian Filter



2D-gaussian

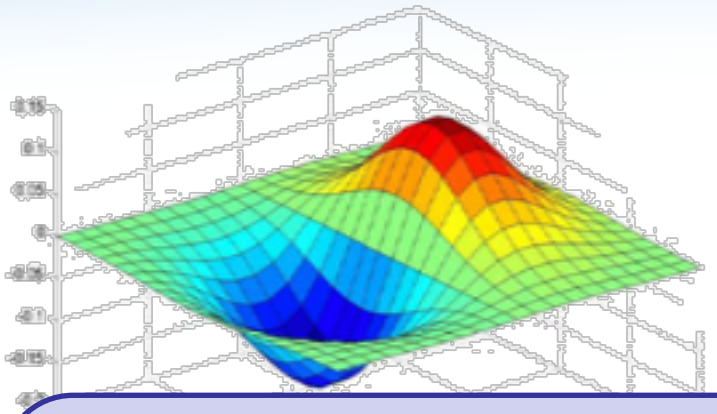
* [1 -1] =

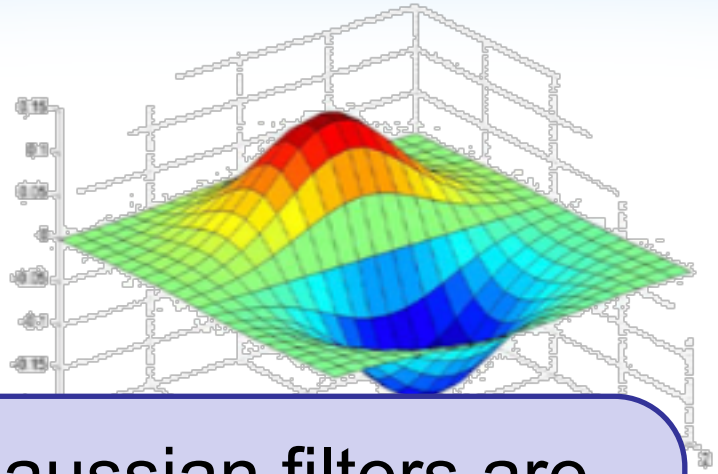


x - derivative

From Steve Seitz

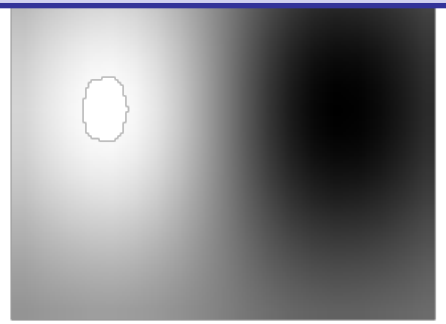
Derivatives of Gaussian Filter

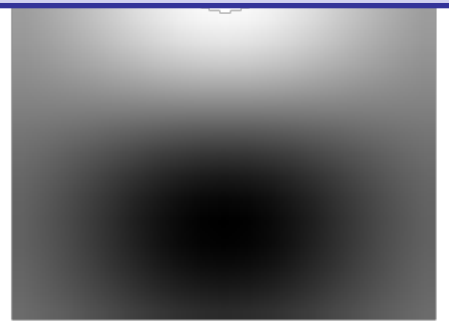




These derivative of Gaussian filters are separable, just like the Gaussian.

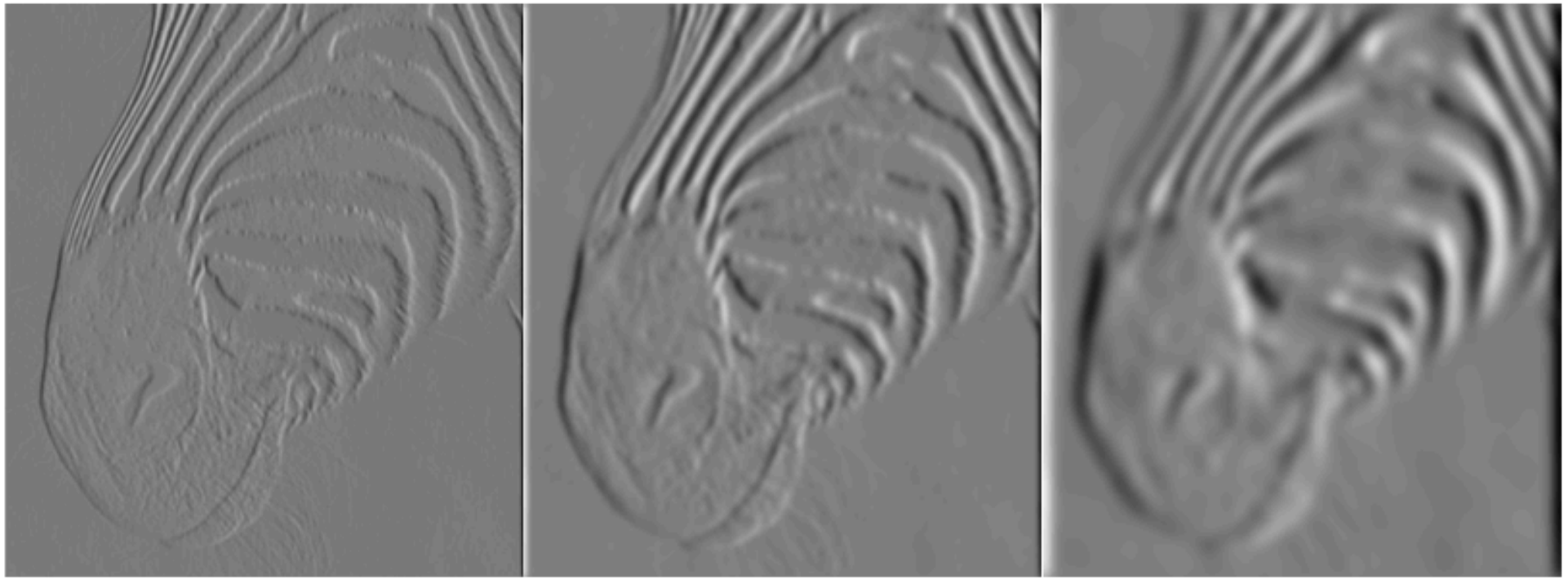
How does that help?





From Steve Seitz

Effect of Gaussian Filter Width (σ)



1 pixel 3 pixels 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

From Steve Seitz

Remaining Issues

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Outline

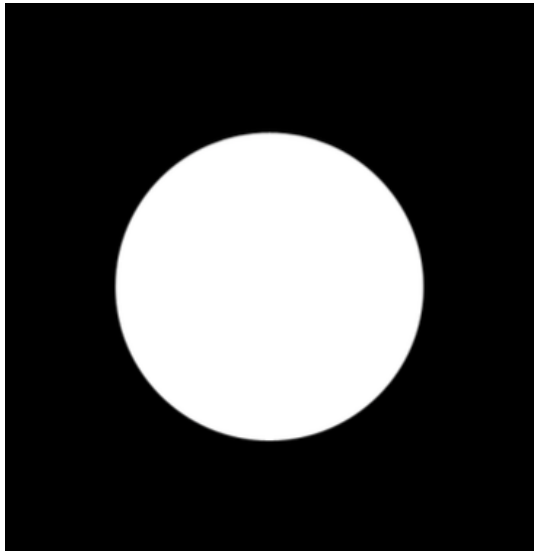
- Simple image processing
  - Greyscale
  - Brightness
  - Mirroring or “flipping”
- Filtering
  - Linear filters: cross-correlation and convolution
  - Gaussian filters
- Edge detection
  - Simple edge detector
  - **Canny edge detector**

Canny Edge Detector (in Project 1)

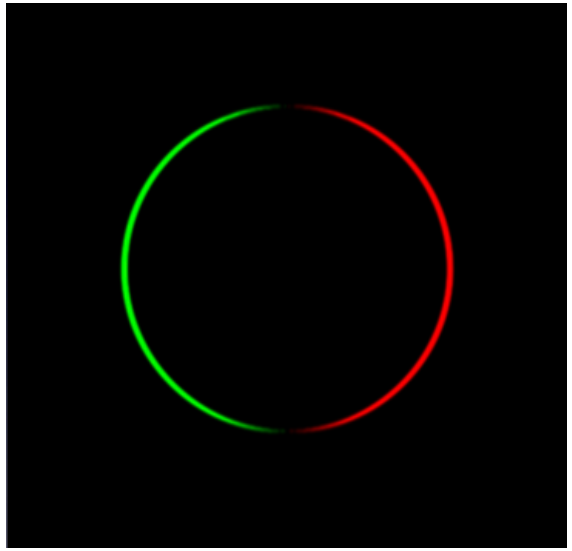
1. Smooth
2. Compute derivative
3. Non-maximum suppression
4. Thresholding

Canny Edge Detector

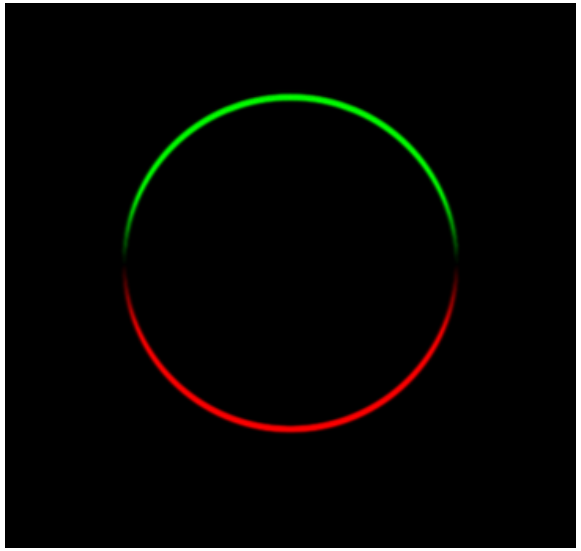
- First, smooth with a Gaussian with filter width σ
- Then compute *x* and *y* derivatives
- As we mentioned before the above 2 steps can be combined (using two derivative of Gaussian filters)



Input image



Smoothed x derivative



Smoothed y derivative

Canny Edge Detector

- **Non-maximum suppression**:
  - Eliminate all but local maxima in magnitude of gradient
  - At each pixel look along direction of gradient: if either neighbor is bigger, set to zero
  - In practice, quantize gradient directions to vertical, horizontal, two diagonals
  - Result: “thinned edge image.”

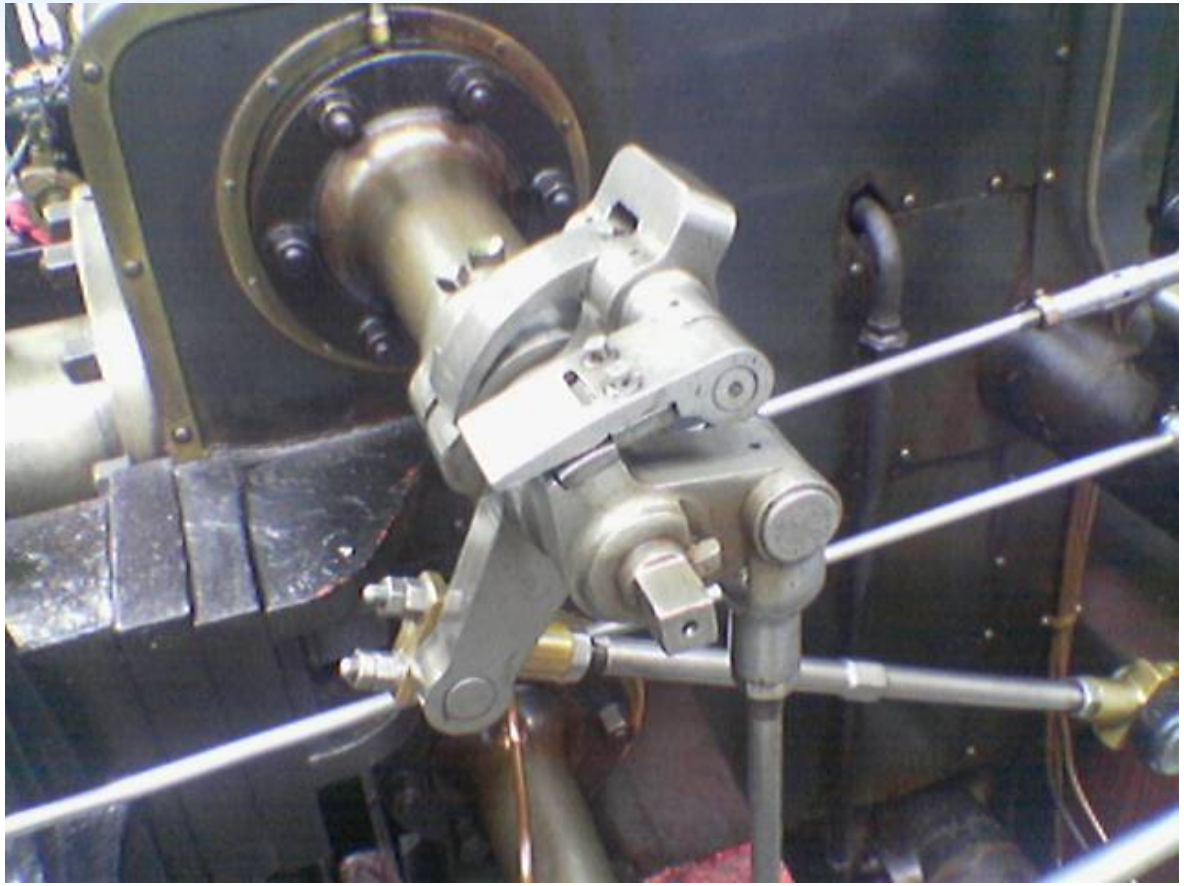
Canny Edge Detector

- Final stage**: thresholding**.
- Simplest: use a single threshold
- Better: use two thresholds
  - Mark pixels as “definitely not edge” if less than τ_{low}.
  - Mark pixels as “strong edge” if greater than τ_{high}.
  - Mark pixels as “weak edge” if within $[\tau_{low}, \tau_{high}]$.
  - Strong pixels are definitely part of the edge.
  - Weak pixels are debatable

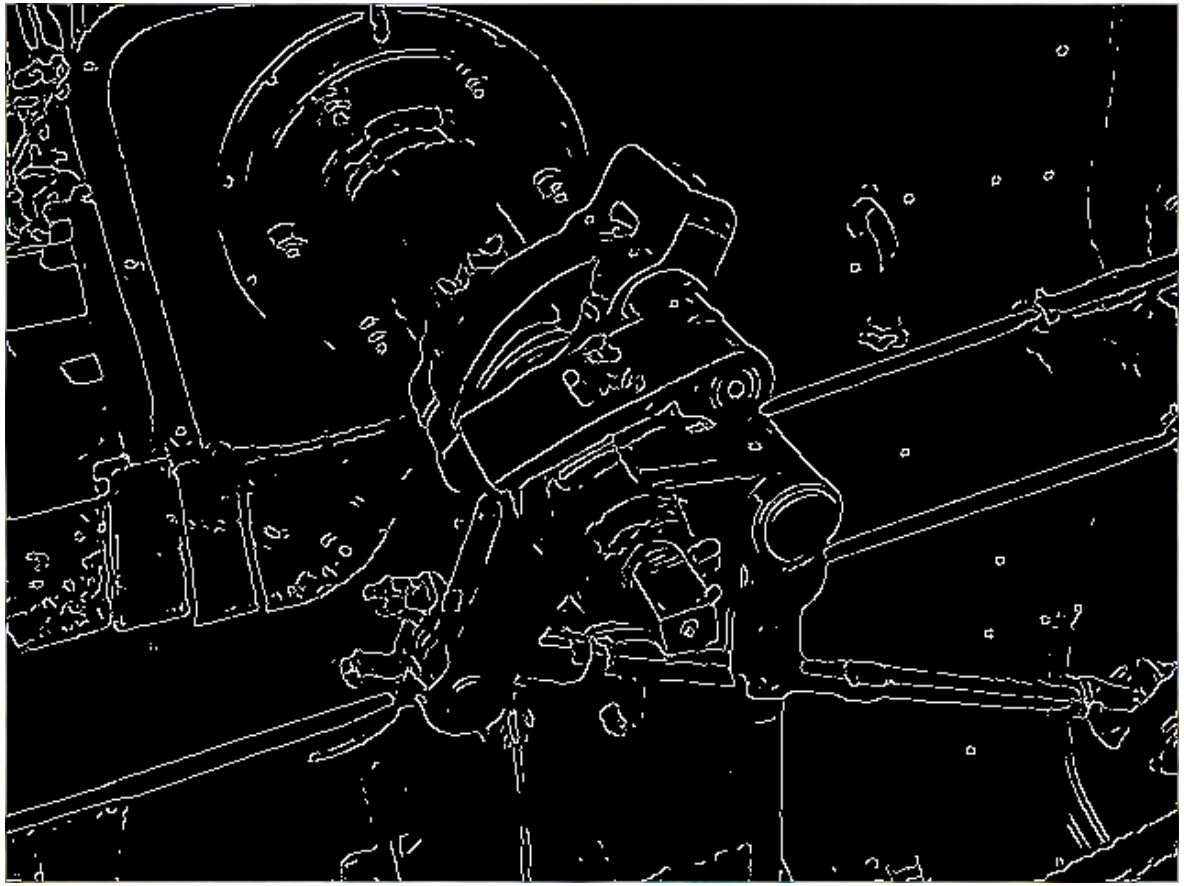
Canny Edge Detector

- Only include weak pixels connected in a chain to some strong pixel.
- How to do this?
  - Visit pixels in chains starting from the strong pixels. For each strong pixel, recursively visit the weak pixels that are in the 8 connected neighborhood around the strong pixel, and label those also as strong (and as edge).
  - Label as “not edge” any weak pixels that are not visited by this process.

Canny Edge Detector



Input image



Canny Edge Detector

From Wikipedia