Introduction to Computer Vision

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lecture 5

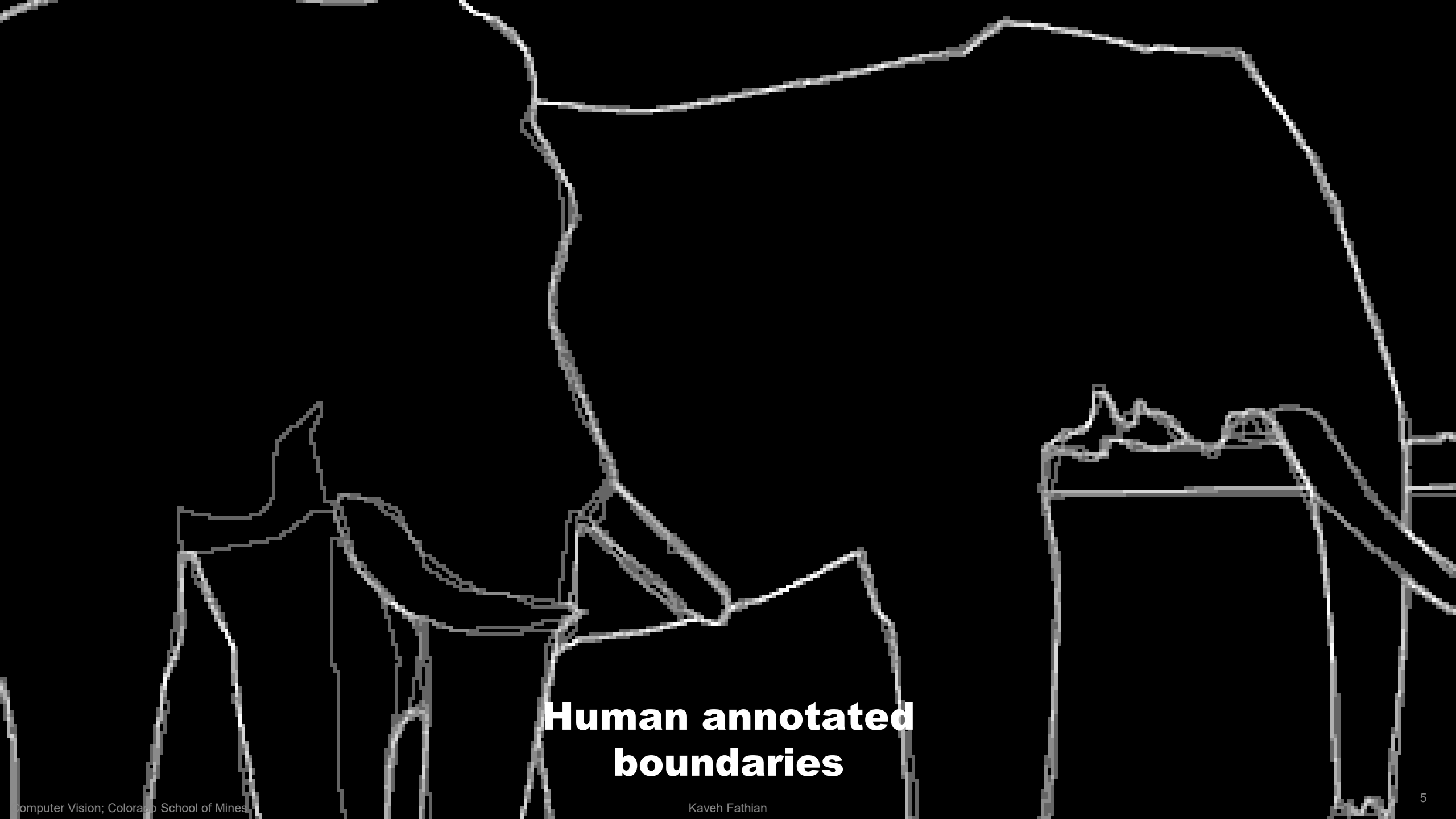
Edge Detection

Learning outcomes:

- Edges
- Edge Detection via Taking the Derivative
- 1D & 2D Gaussian Filters
- The 1st Derivative of a Gaussian
- Detection and Localization
- Canny Edge Detector
- Non-Maximum Suppression
- Hysteresis Thresholding

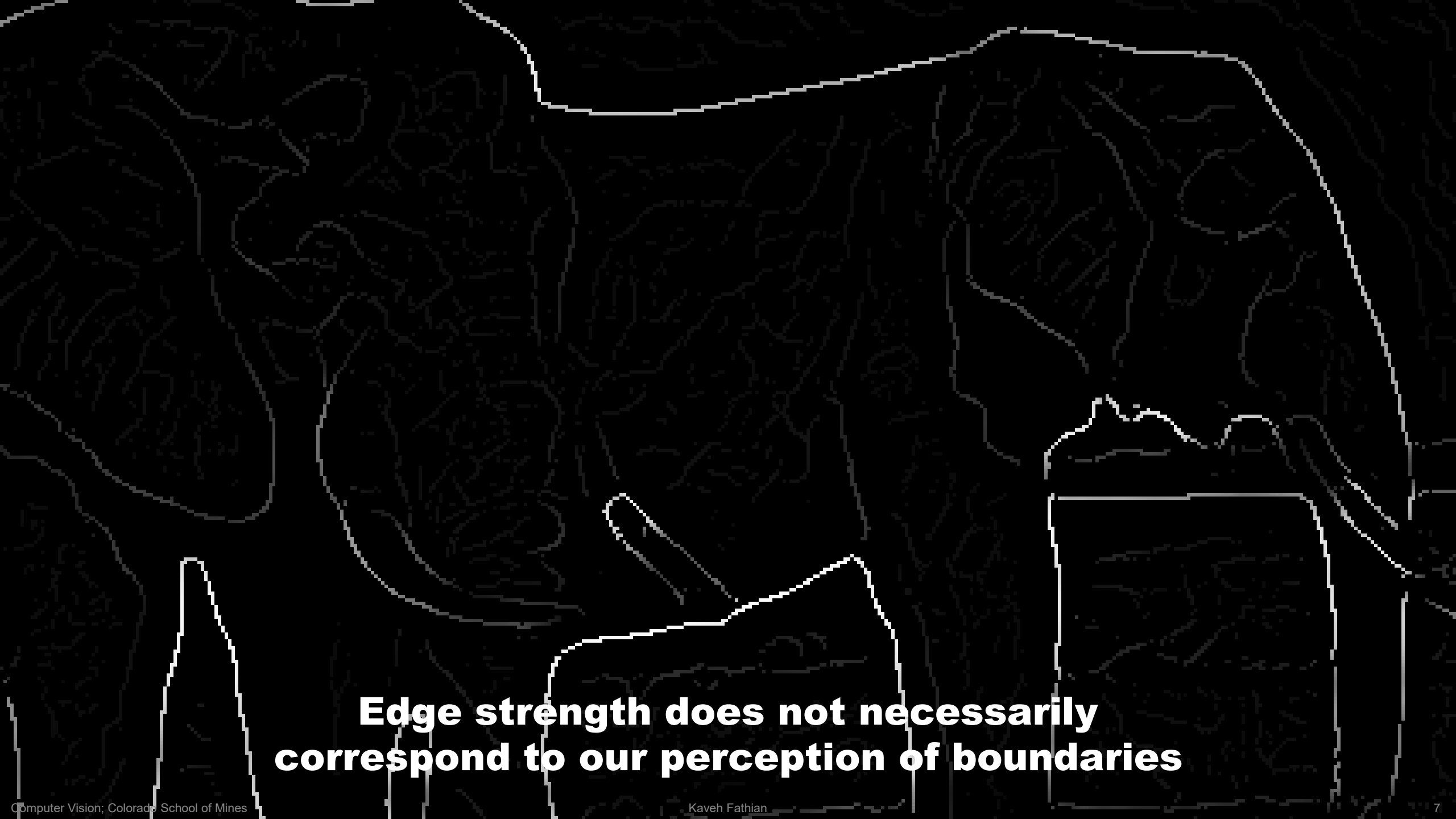
Edge Detection


Where are the object boundaries?



Human annotated boundaries

Edge detection



Edge strength does not necessarily correspond to our perception of boundaries

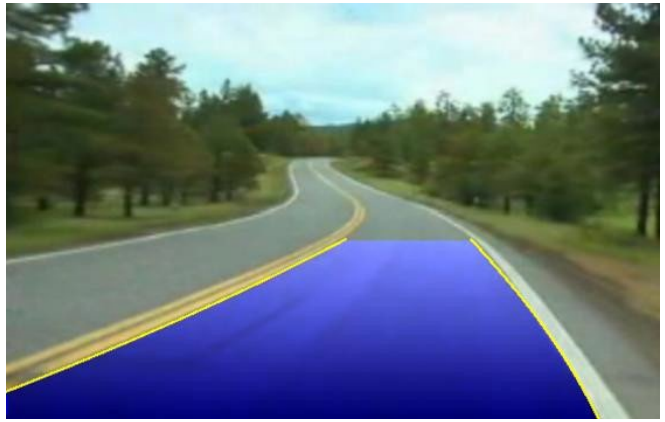

Where are the object boundaries?


Human annotated boundaries

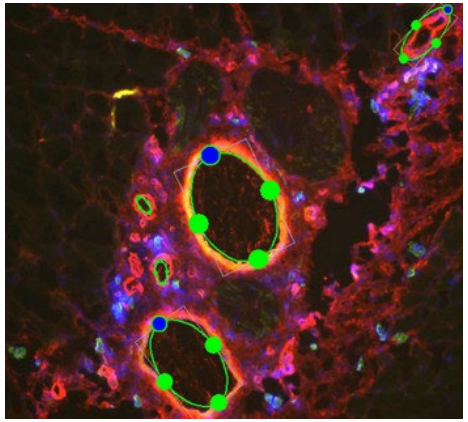
Edge detection


Defining boundaries are hard
for us too

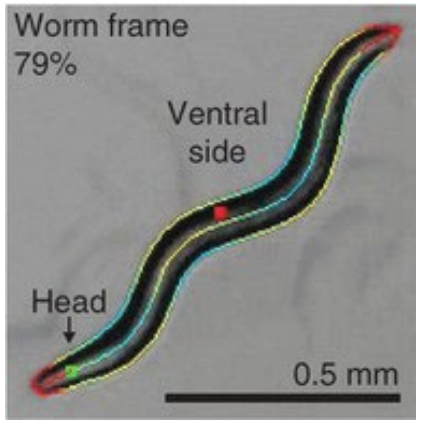
Applications



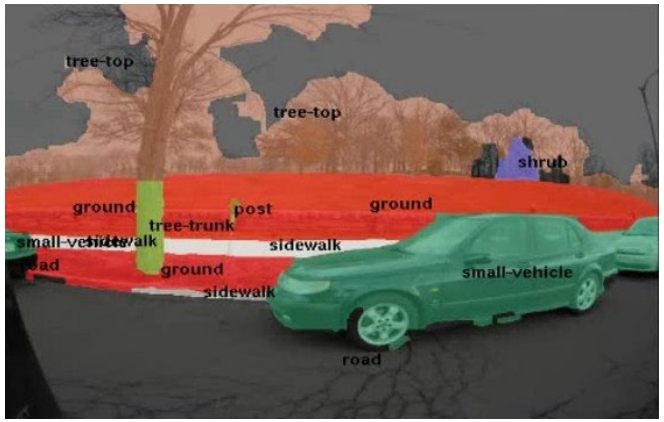
Autonomous Vehicles
(lane line detection)



tissue engineering
(blood vessel counting)



behavioral genetics
(earthworm contours)



Autonomous Vehicles
(semantic scene segmentation)

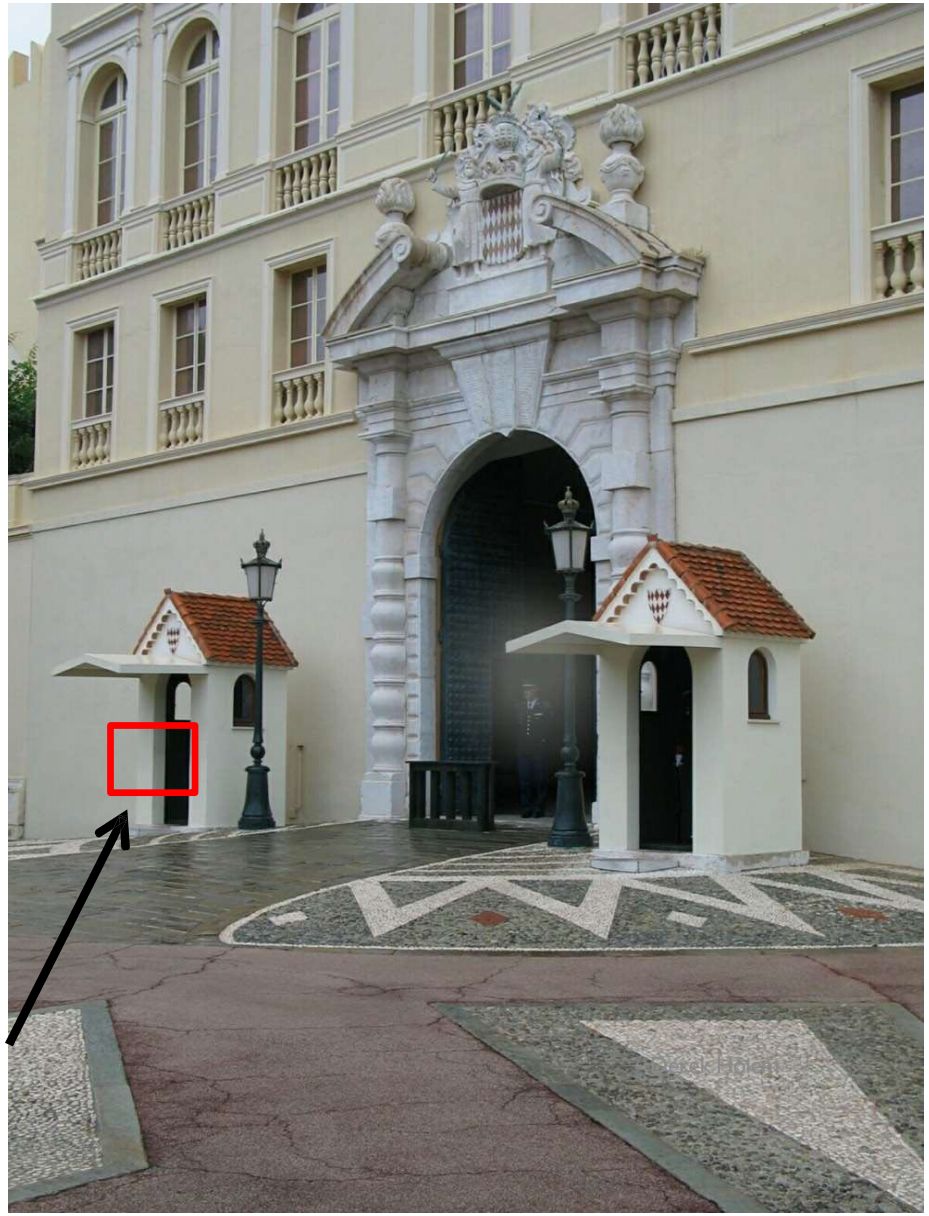
Computational Photography
(image inpainting)

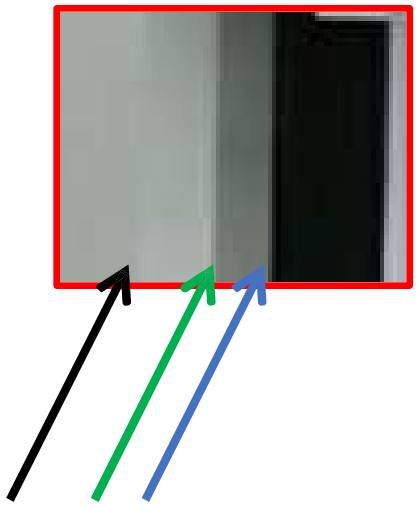
Image editing
(Photoshop smart select)

Closer Look at Edges

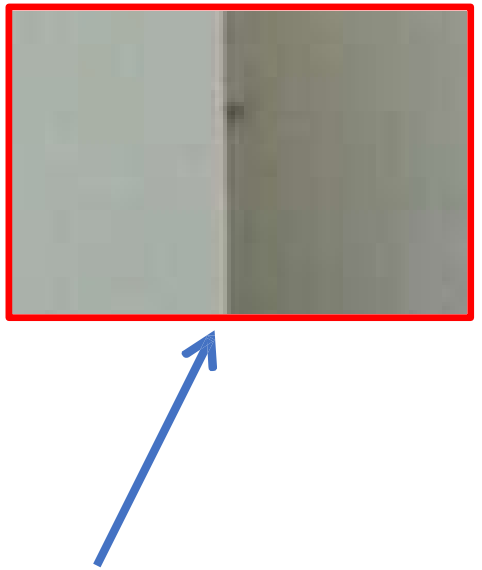
Derek Hoiem

Closer Look at Edges





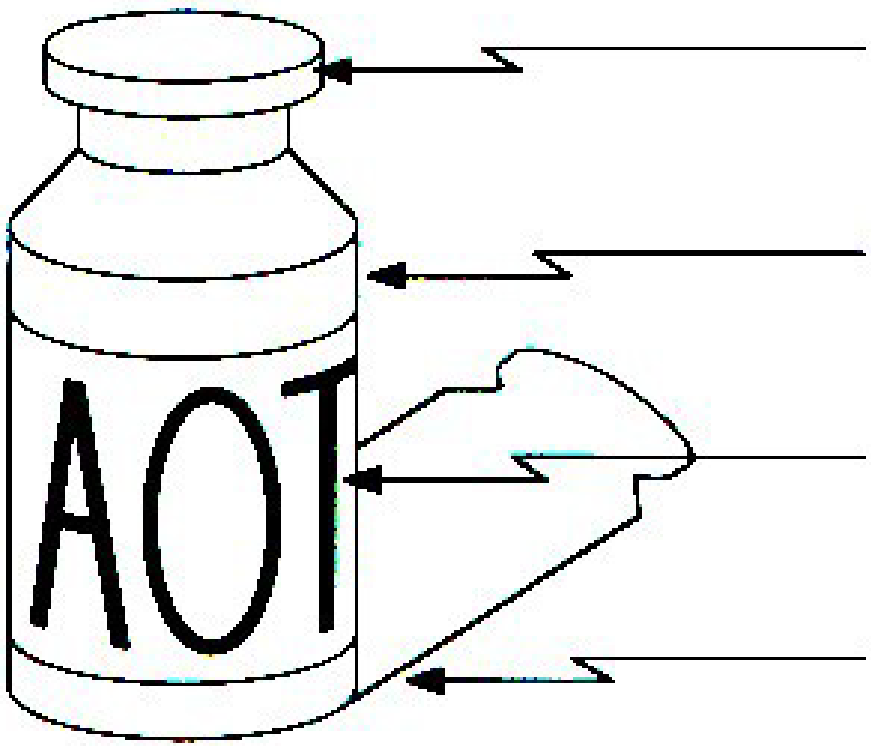
Closer Look at Edges



Closer Look at Edges

Causes of Edges

Edges are caused by a variety of factors



surface orientation discontinuity

depth discontinuity

surface color discontinuity

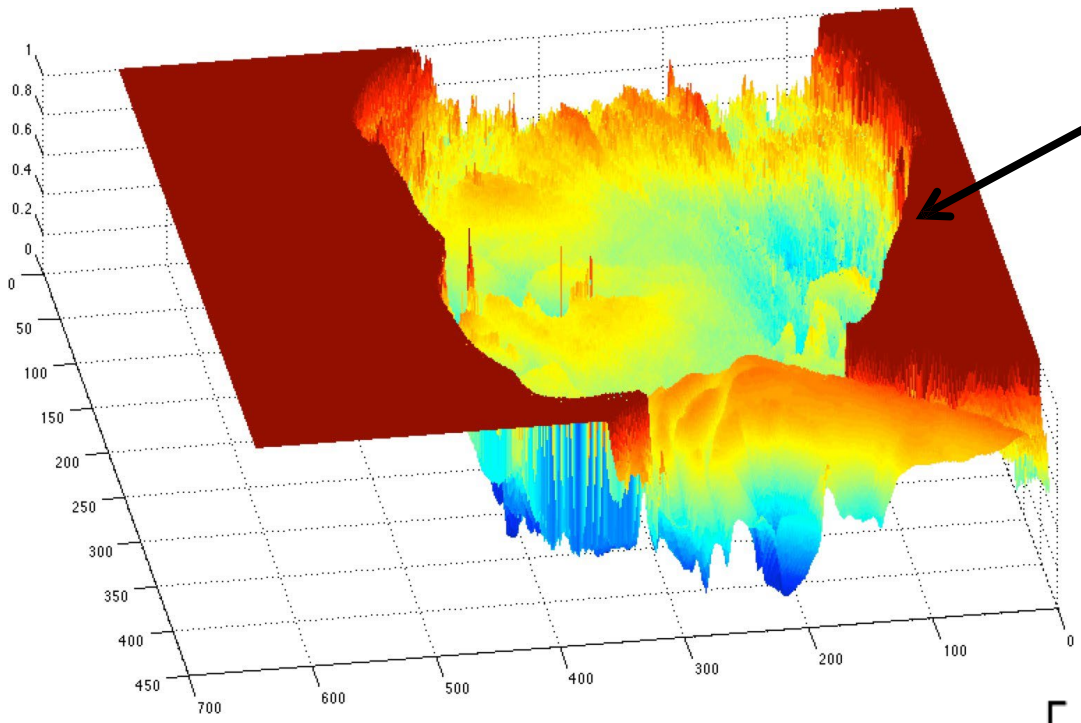
illumination discontinuity

Steve Seitz

What are image edges?

grayscale image

$f(\boldsymbol{x})$



Very sharp discontinuities in intensity.

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

Edge Detection

Definition

The process of identifying parts of a digital image with sharp changes (discontinuities) in image intensity.

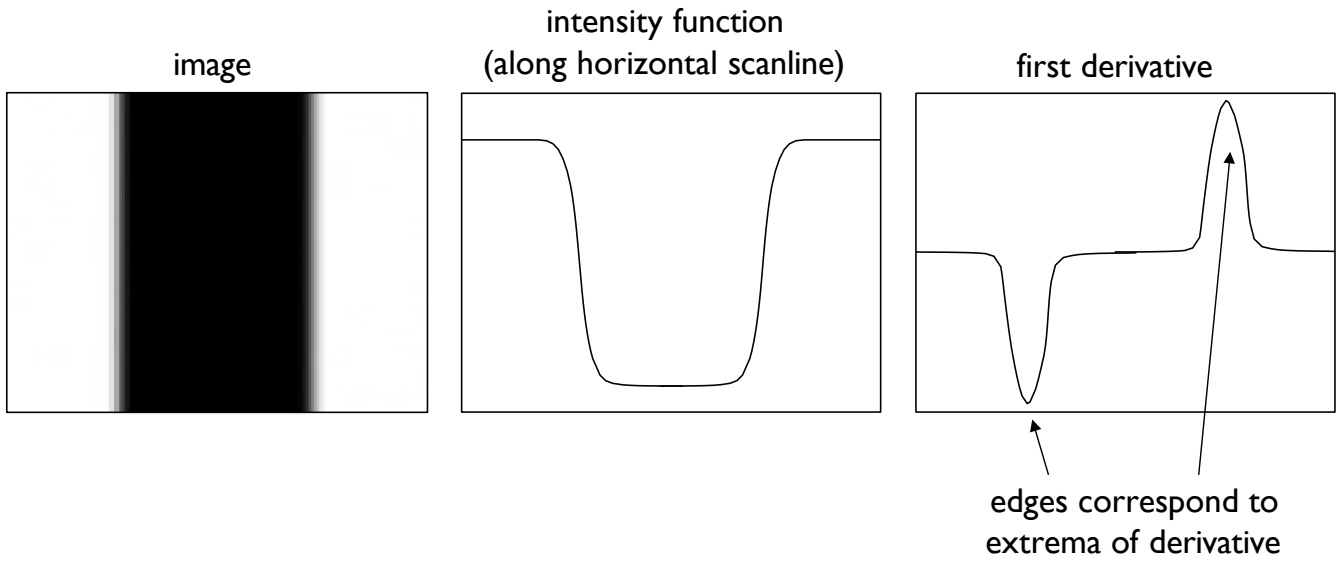
Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

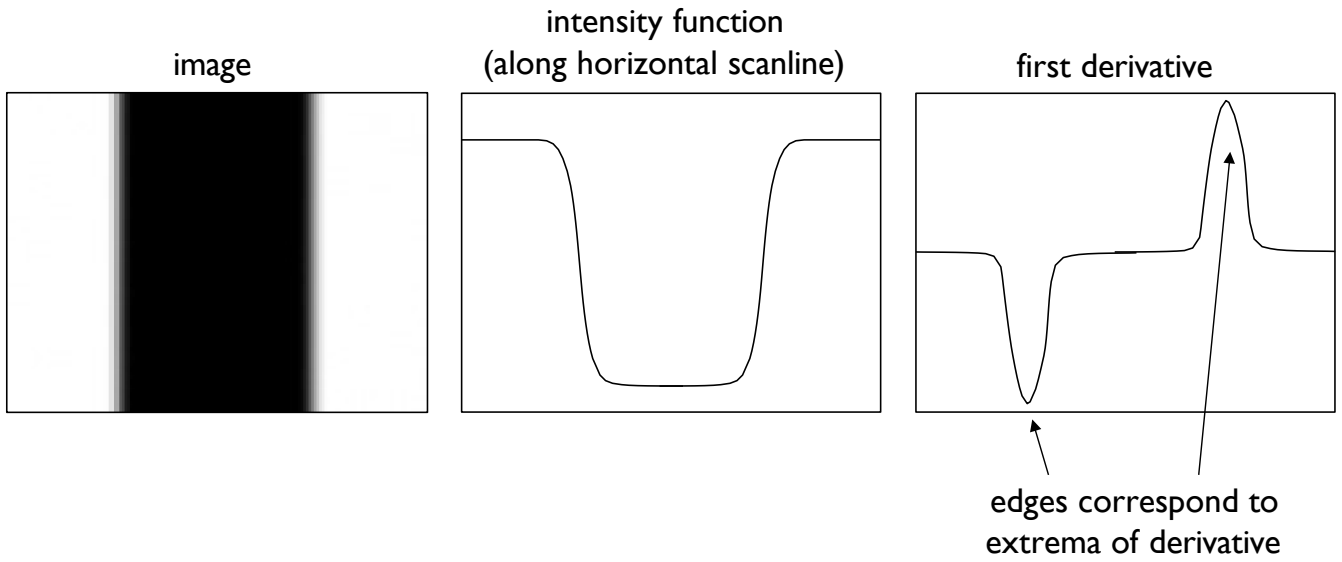


How do you differentiate a discrete image (or any other discrete signal)?

Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.



How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

What convolution kernel does this correspond to?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

-1	0	1	?

1	0	-1	?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

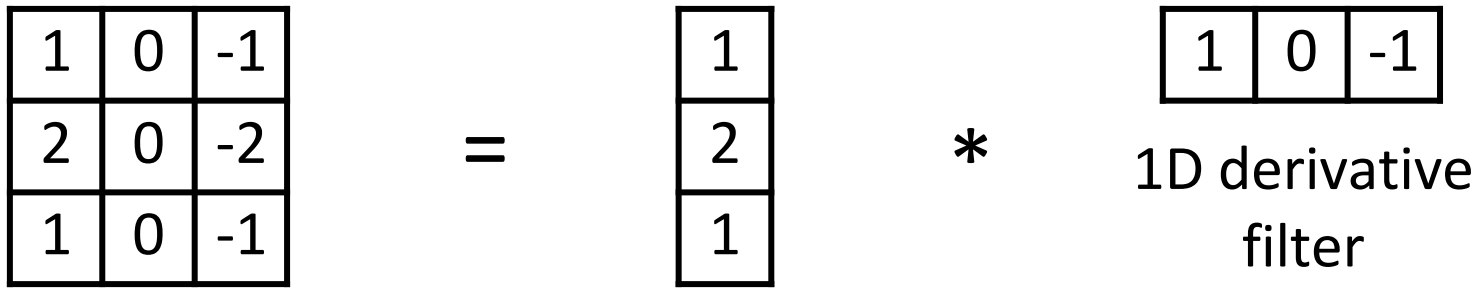
For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

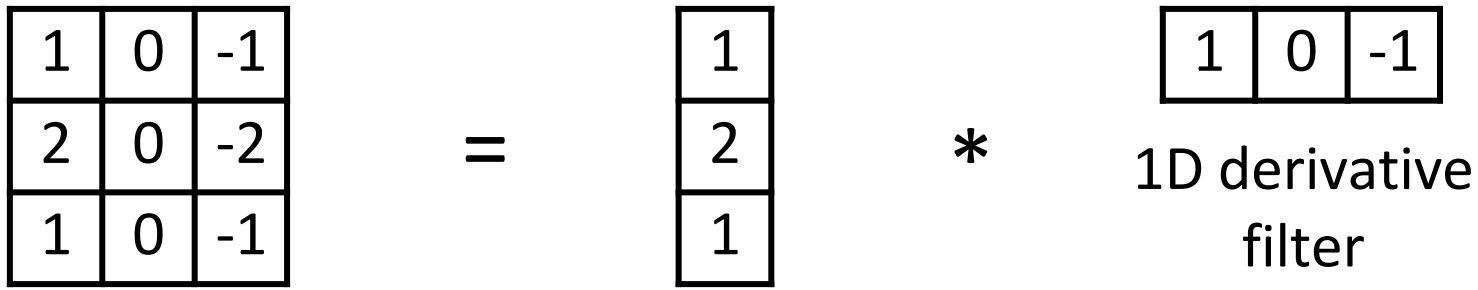
1D derivative filter

1	0	-1

The Sobel filter

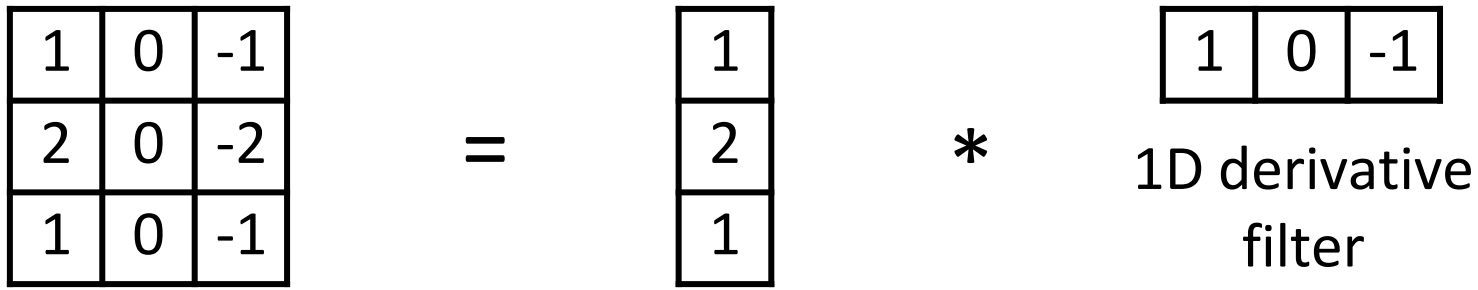


The Sobel filter



In a 2D image, does this filter responses along horizontal or vertical lines?

The Sobel filter



Does this filter return large responses on vertical or horizontal lines?

The Sobel filter

Horizontal Sober filter:

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

What does the vertical Sobel filter look like?

The Sobel filter

Horizontal Sober filter:

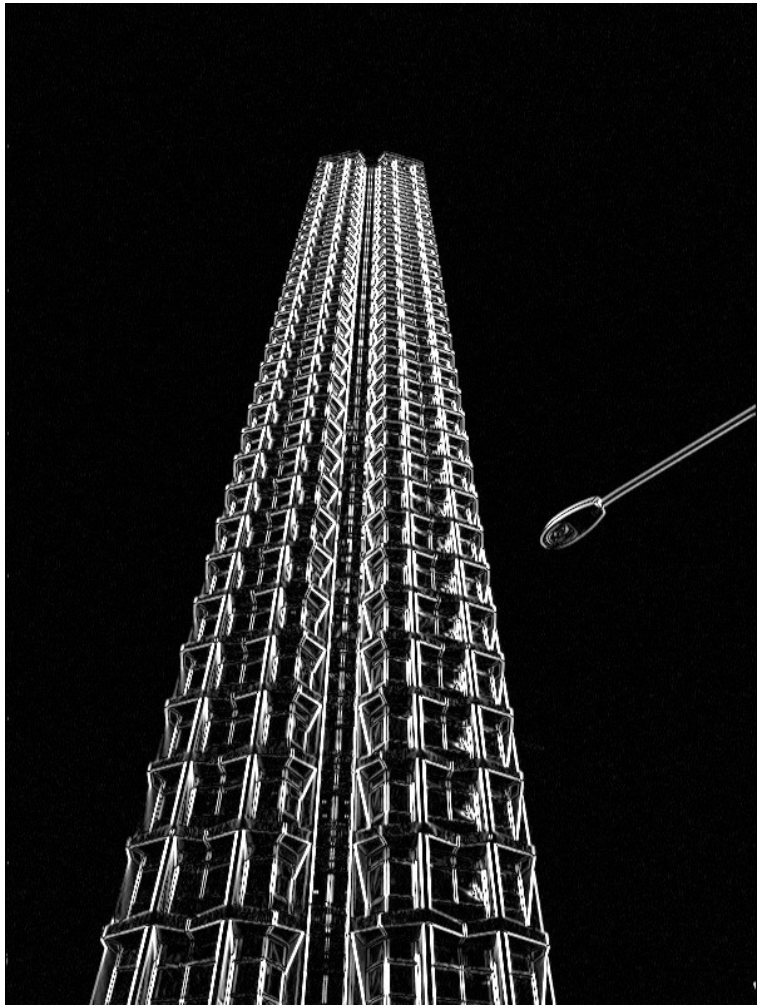
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Vertical Sobel filter:

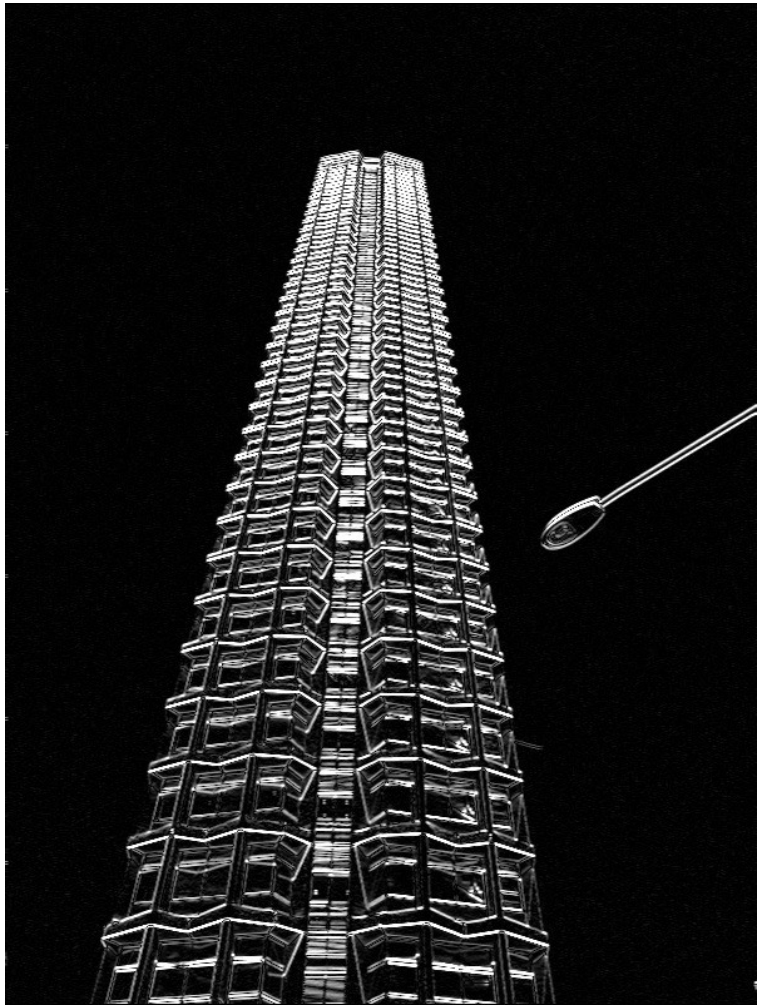
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Sobel filter example

original



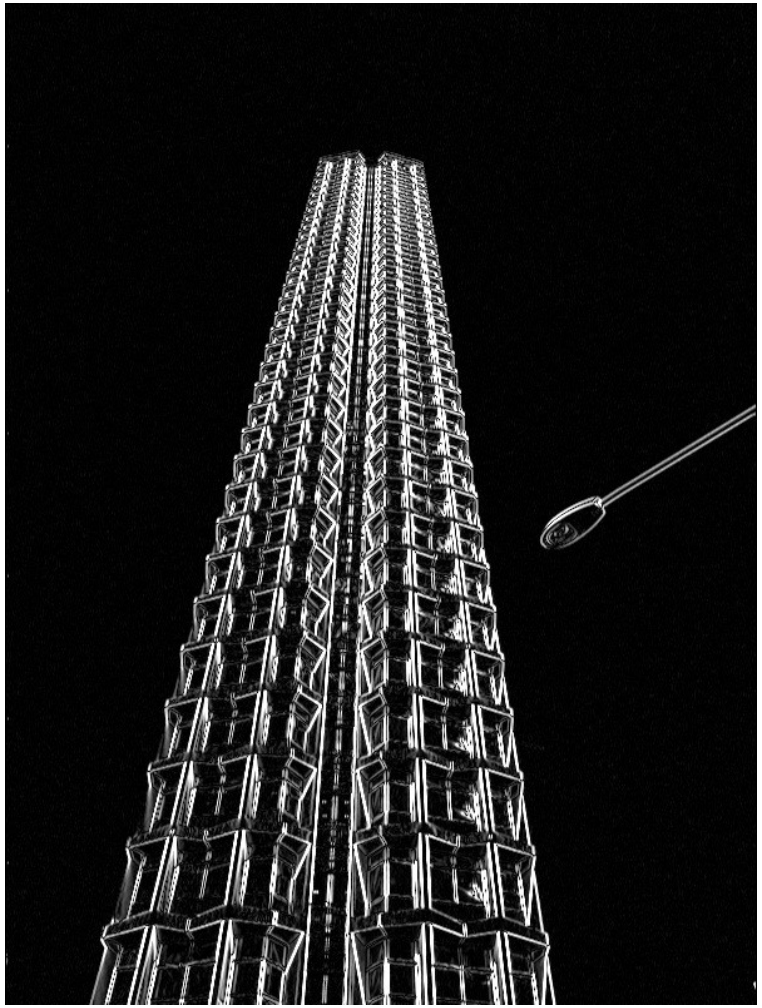
which Sobel filter?



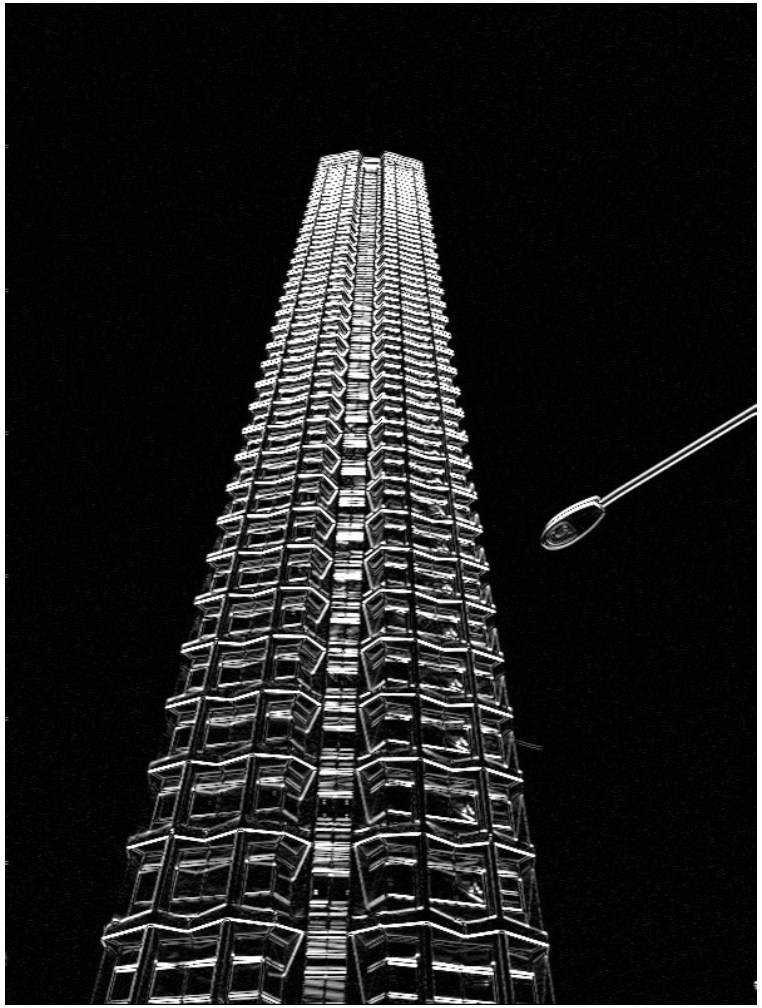
which Sobel filter?

Sobel filter example

original

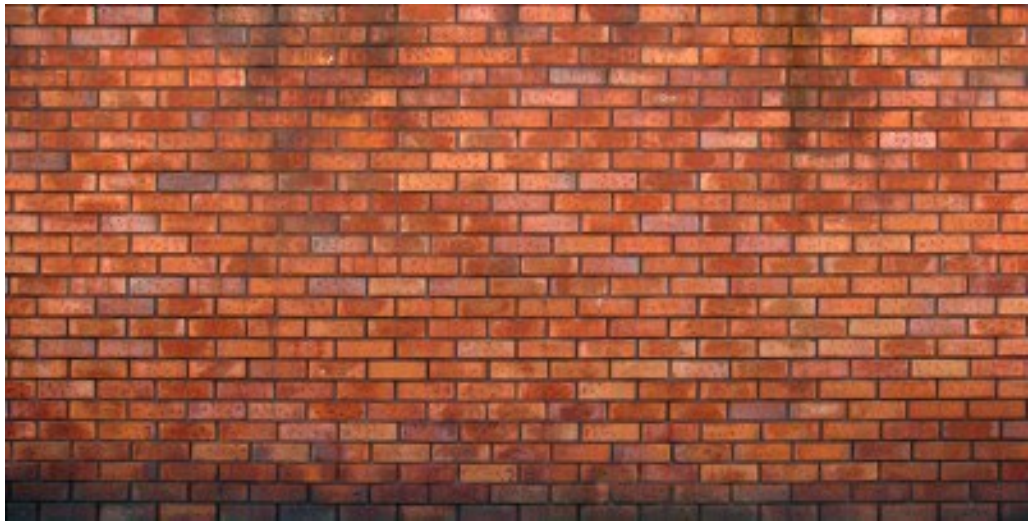


horizontal Sobel filter



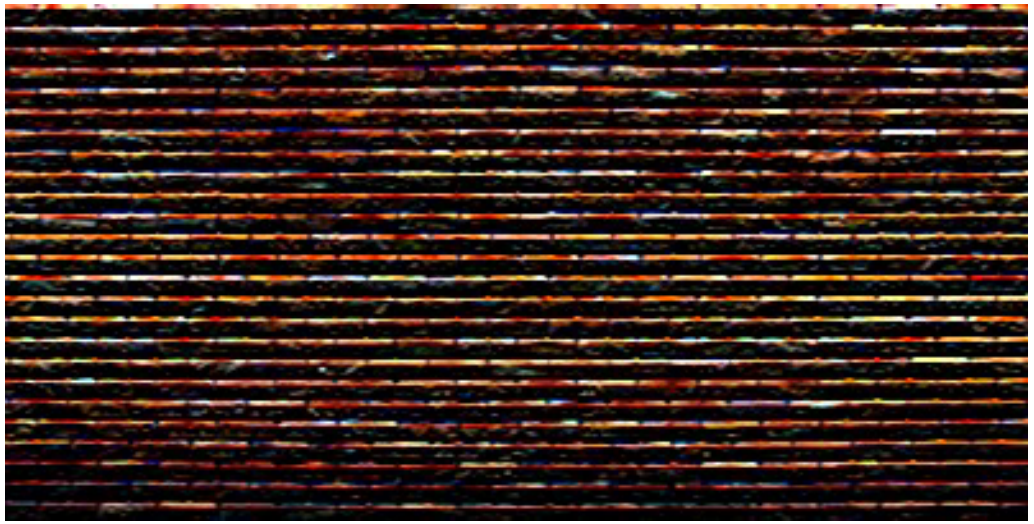
vertical Sobel filter

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Several derivative filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

- How are the other filters derived and how do they relate to the Sobel filter?
- How would you derive a derivative filter that is larger than 3x3?

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

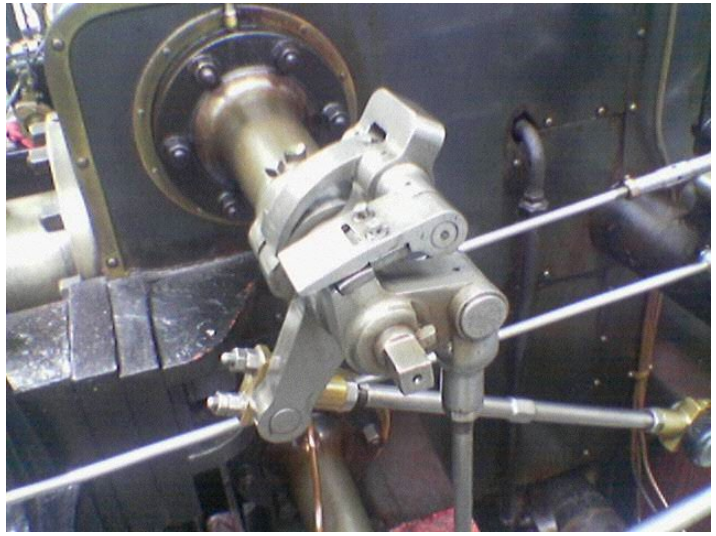
3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y} \right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

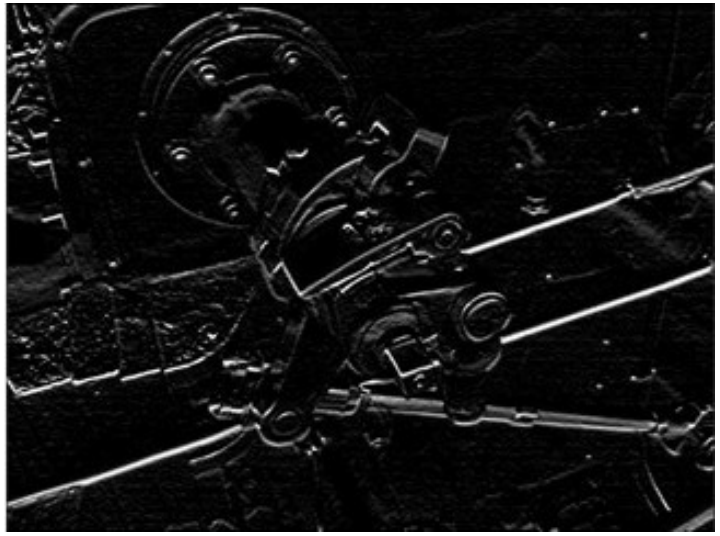
gradient direction amplitude

Image gradient example

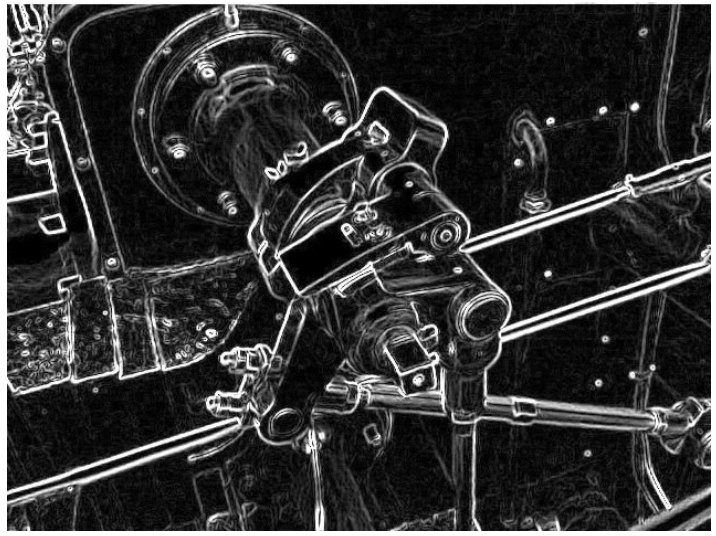
original



vertical derivative



gradient amplitude

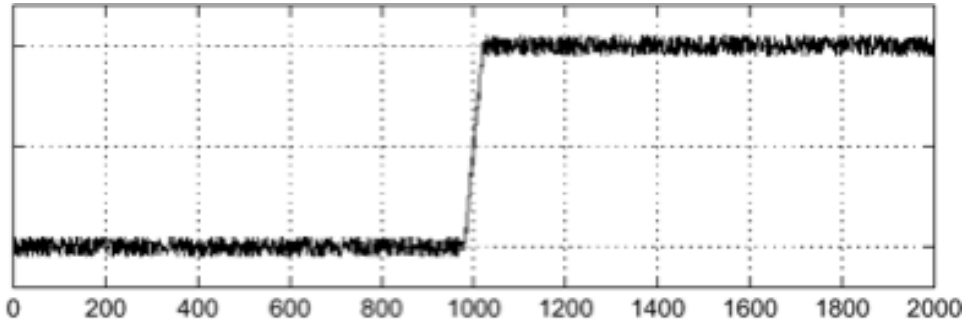


horizontal derivative

How does the gradient direction relate to these edges?

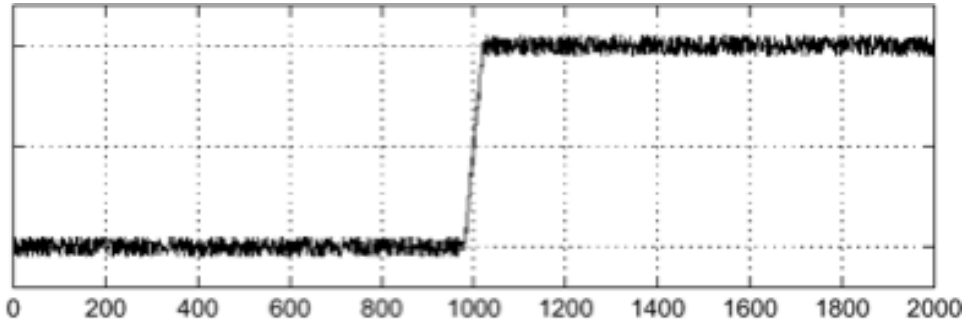
How do you find the edge of this signal?

intensity plot



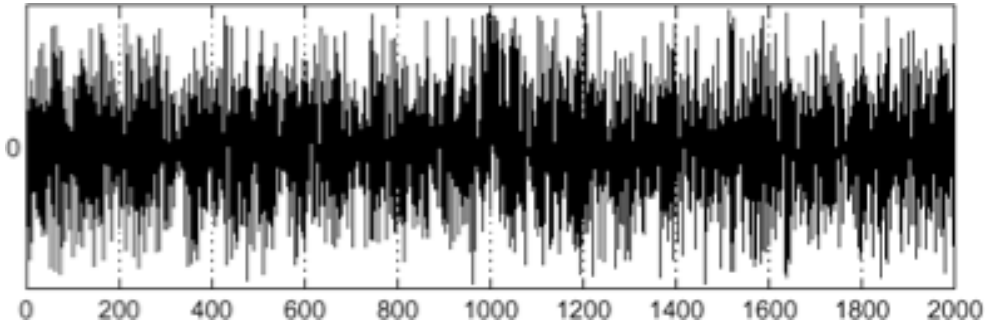
How do you find the edge of this signal?

intensity plot



Using a derivative filter:

derivative plot



What's the problem here?

Differentiation is very sensitive to noise

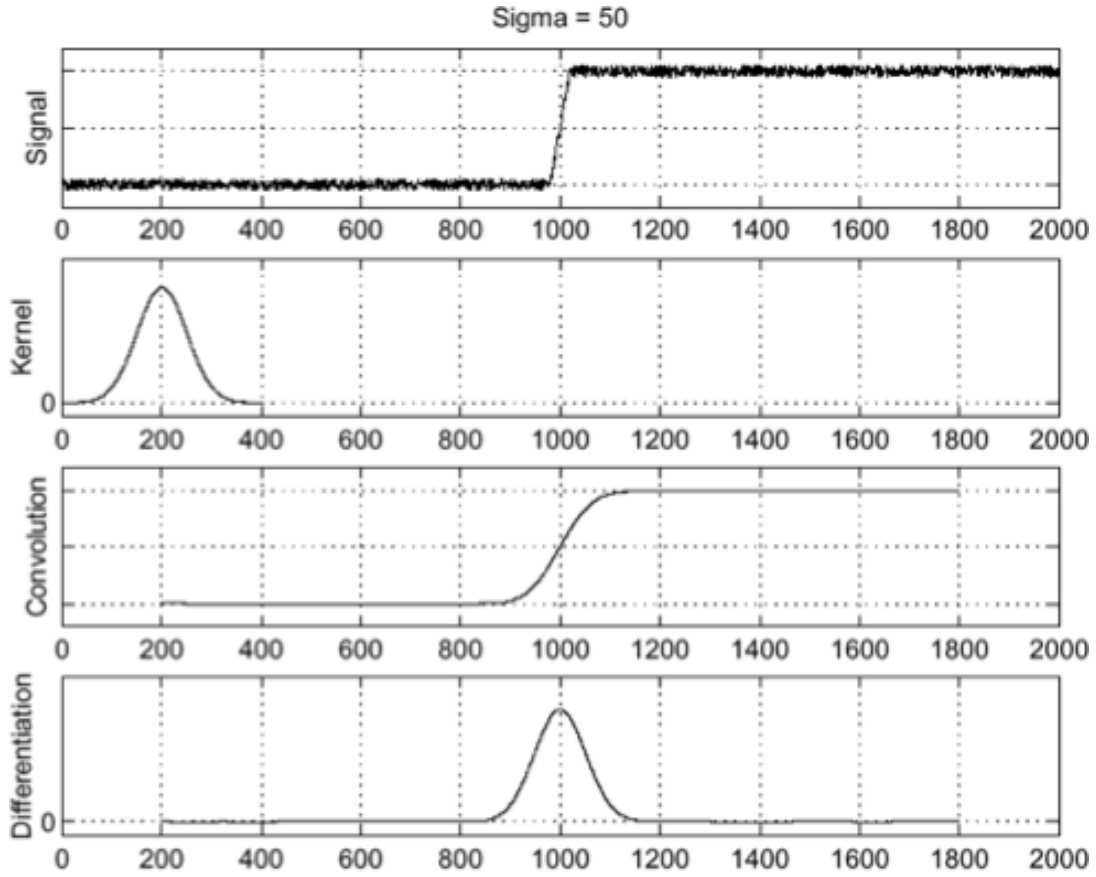
When using derivative filters, it is critical to blur first!

input

Gaussian

blurred

derivative of blurred



How much should we blur?

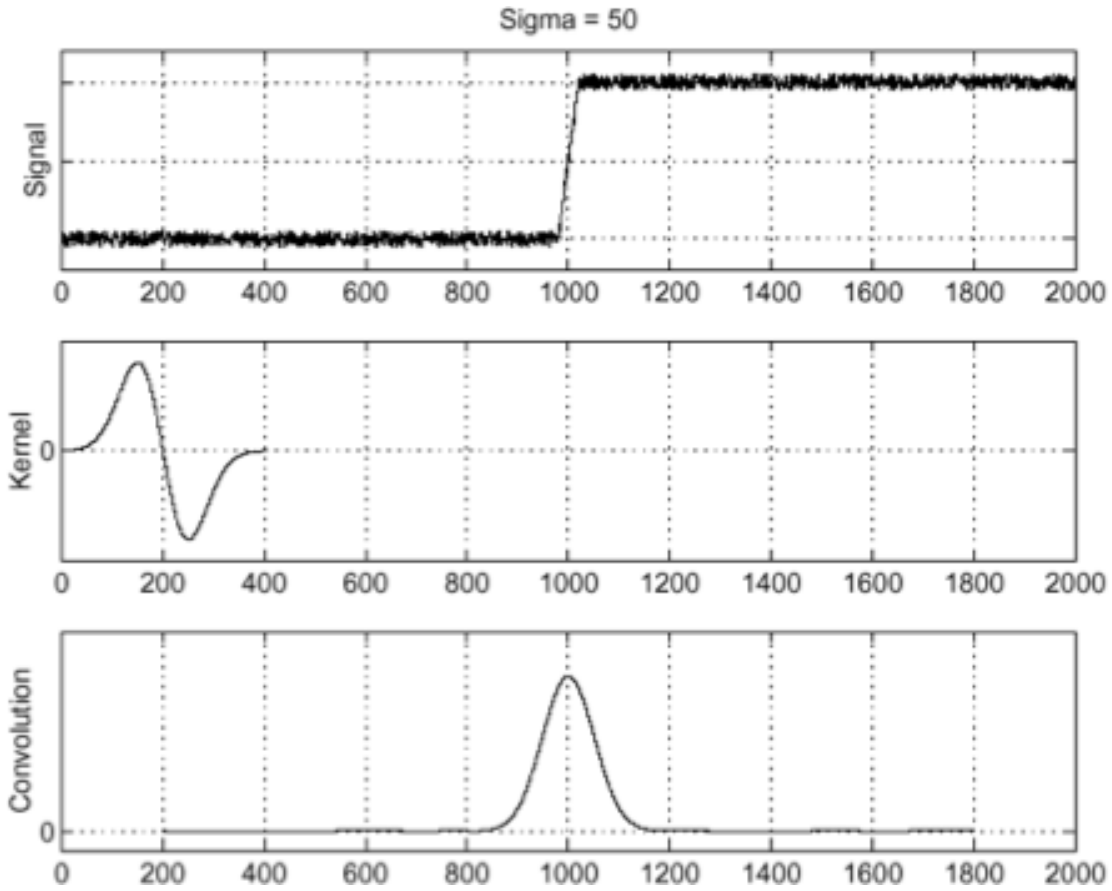
Derivative of Gaussian (DoG) filter

Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

input

derivative of Gaussian

output (same as before)



- How many operations did we save?
- Any other advantages beyond efficiency?

Laplace filter

Basically, a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

Laplace filter

?

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}$$

Laplace filter

1	-2	1

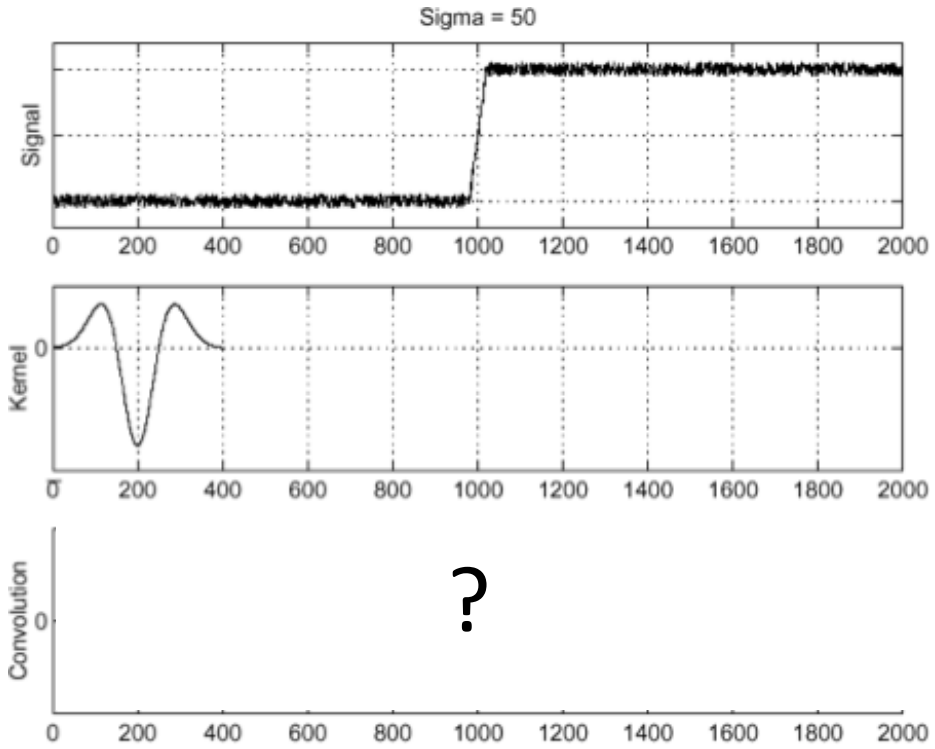
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



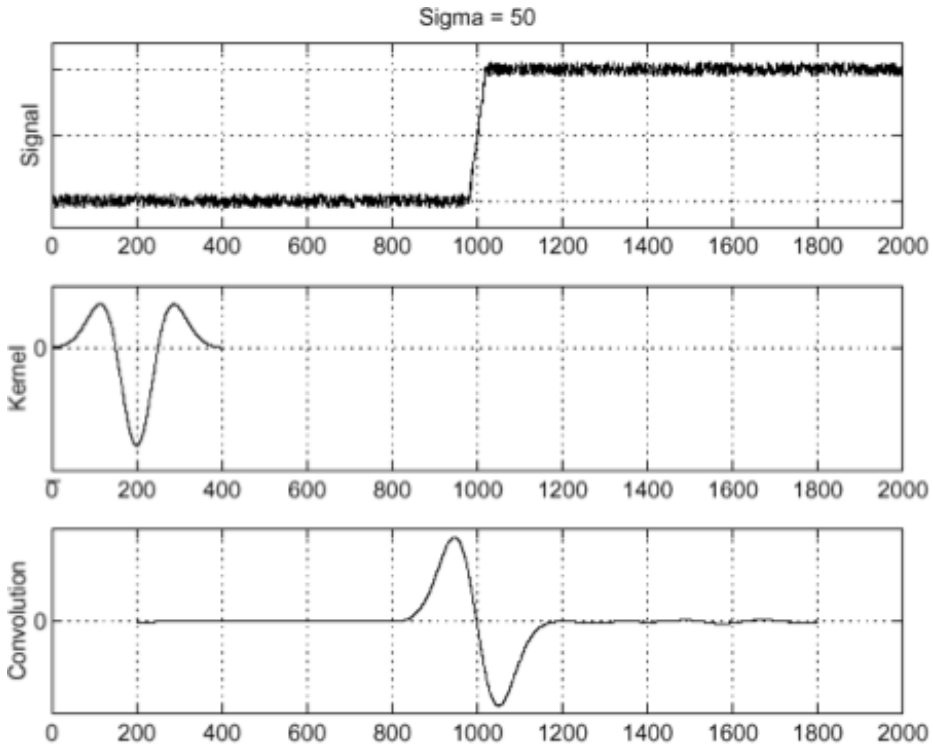
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



“zero crossings’’ at edges

Laplace and LoG filtering examples

Laplacian of Gaussian filtering

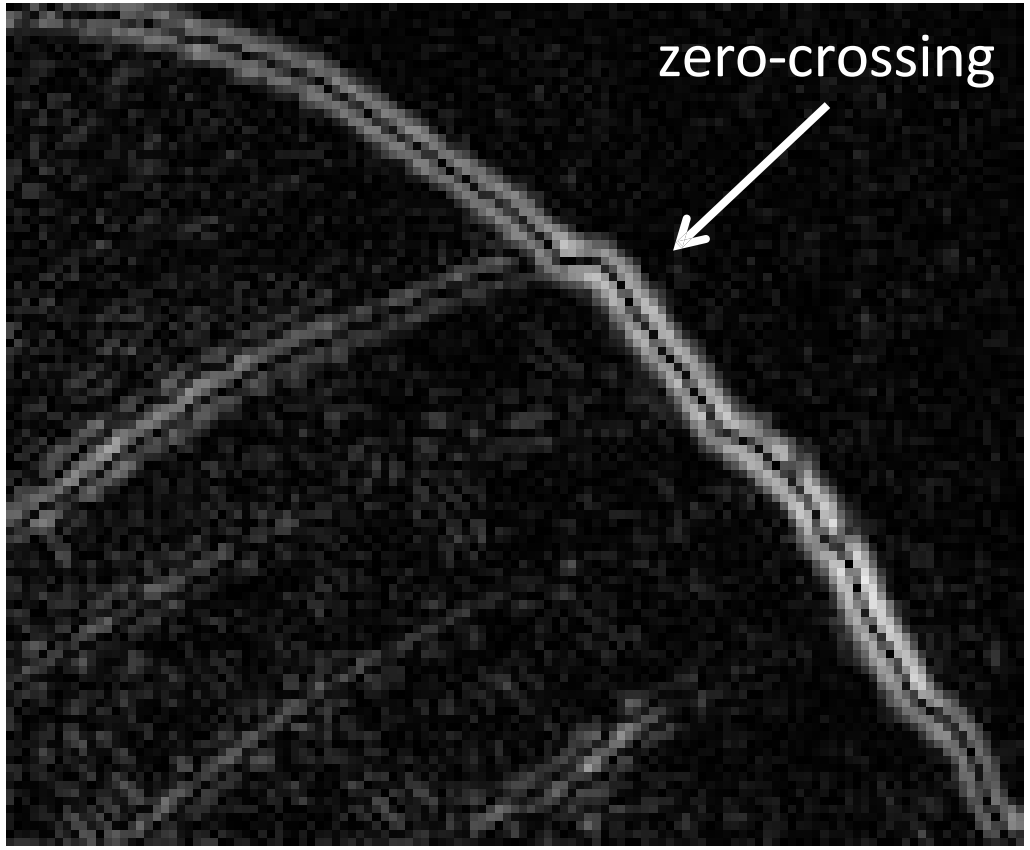
Laplace filtering

Laplacian of Gaussian vs Derivative of Gaussian

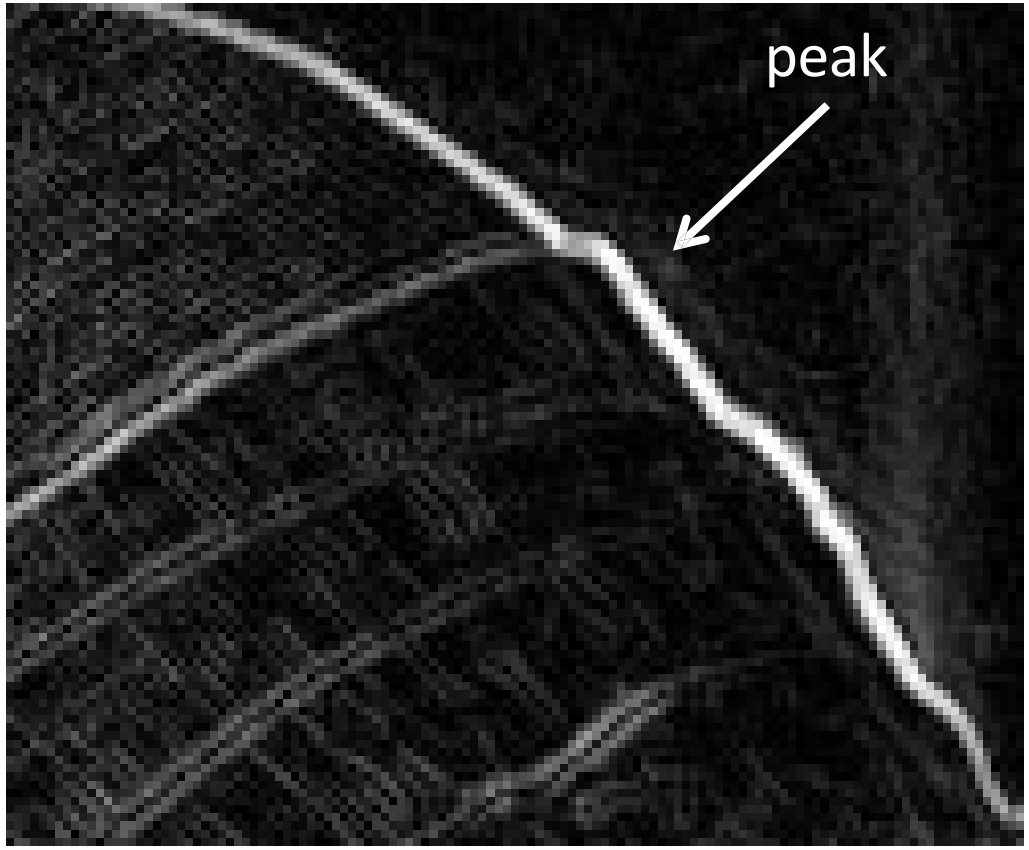
Laplacian of Gaussian filtering

Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian



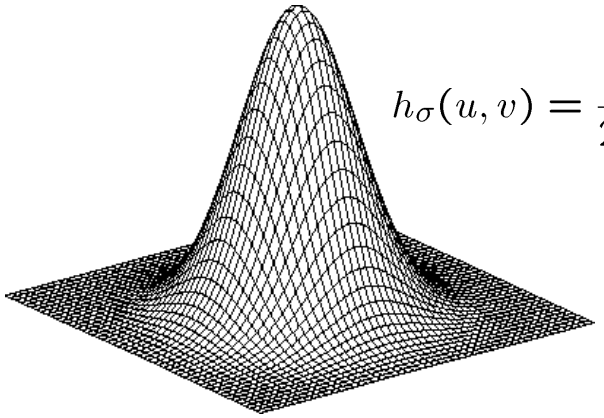
Laplacian of Gaussian filtering



Derivative of Gaussian filtering

Zero crossings are more accurate at localizing edges (but not very convenient).

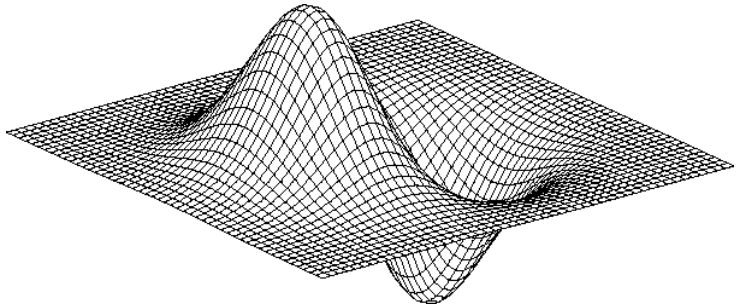
2D Gaussian filters



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

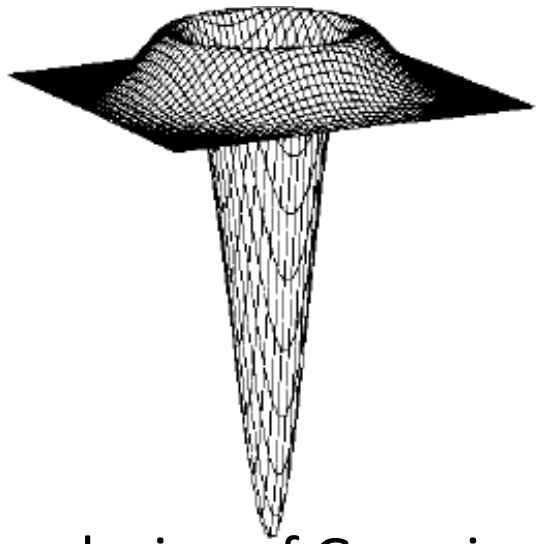
Gaussian

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



Derivative of Gaussian

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

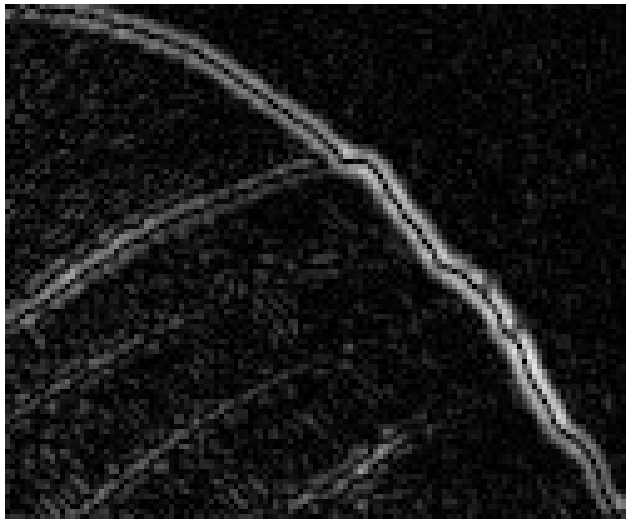


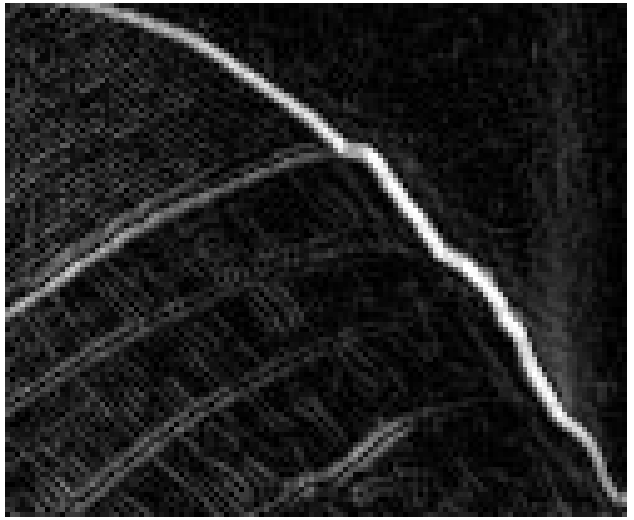
Laplacian of Gaussian

Designing an Edge Detector

Criteria for a good edge detector:

- **Good detection:** the optimal detector should find all real edges, ignoring noise or other artifacts
- **Good localization**
  - The edges detected must be as close as possible to the true edges
  - The detector must return one point only for each true edge point





Cues of edge detection

- Differences in color, intensity, or texture across the boundary
- Continuity and closure
- High-level knowledge

Canny Edge Detector

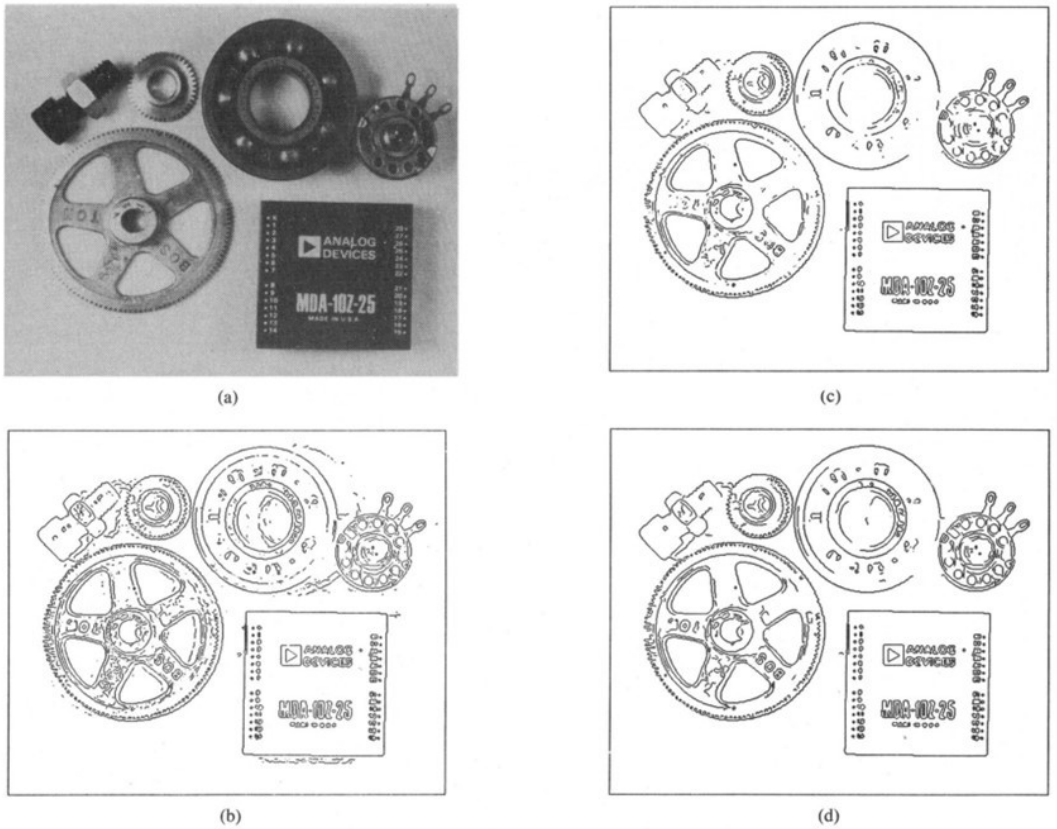
A Better Edge Detector

- 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?

Canny Edge Detector

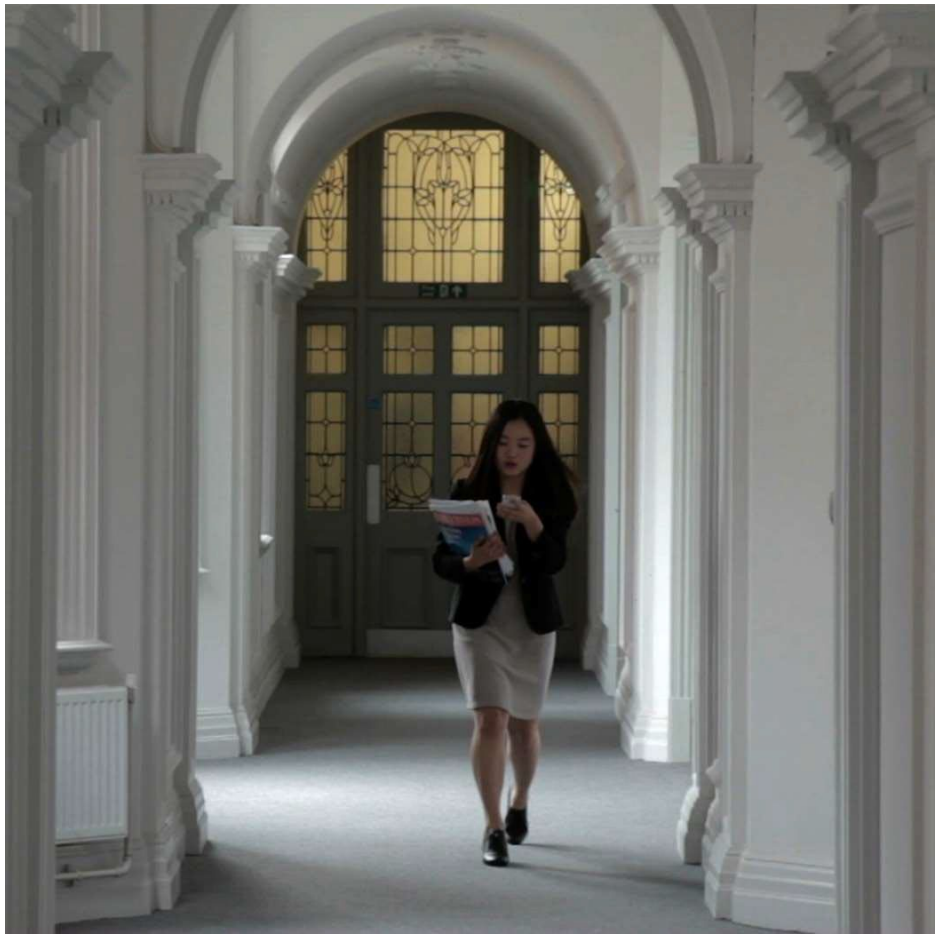
Probably the most widely used edge detector in computer vision.



J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

45,386 citations!

Demonstrator Image



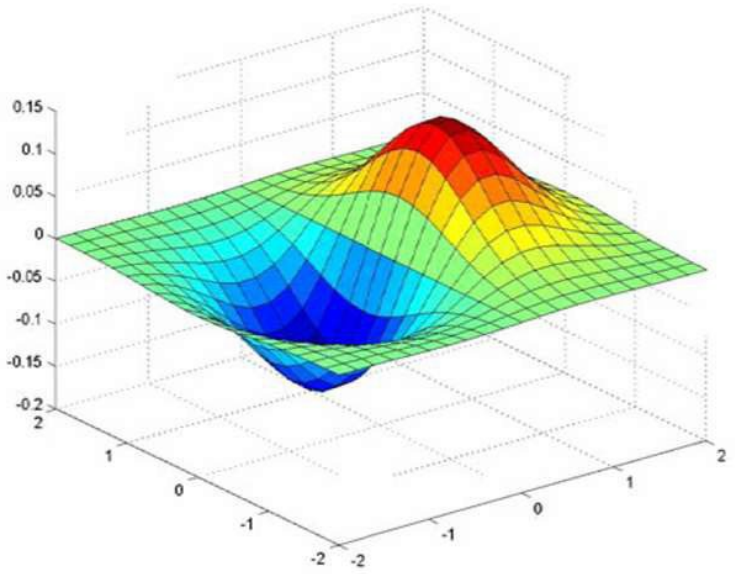
`rgb2gray('img.png')`

Canny Edge Detector

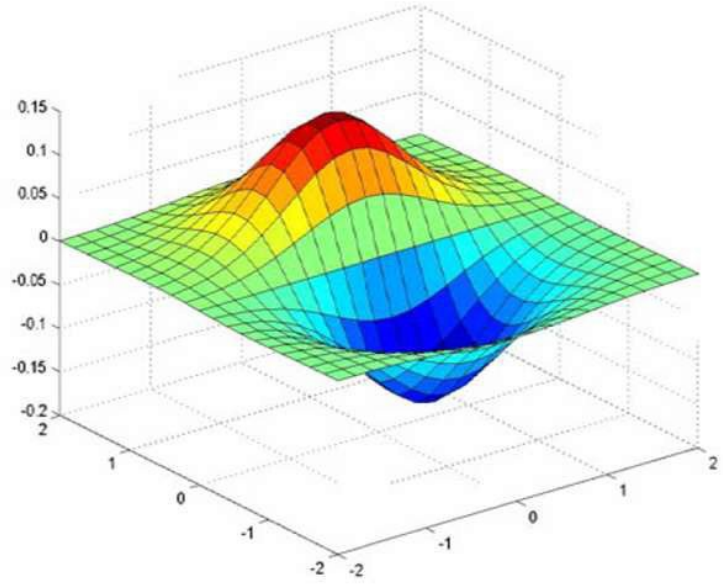
Algorithm

1. Filter image with x, y derivatives of Gaussian

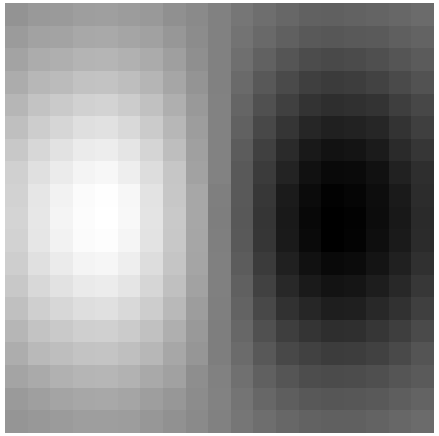
Derivative of Gaussian filter

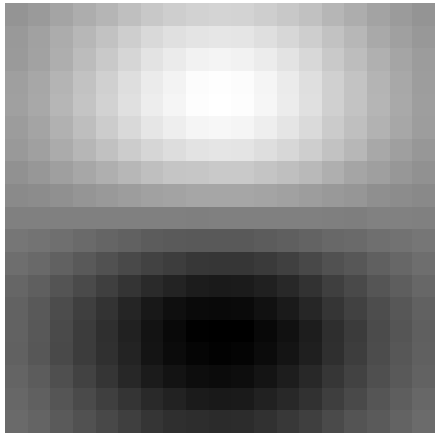


x-direction



y-direction

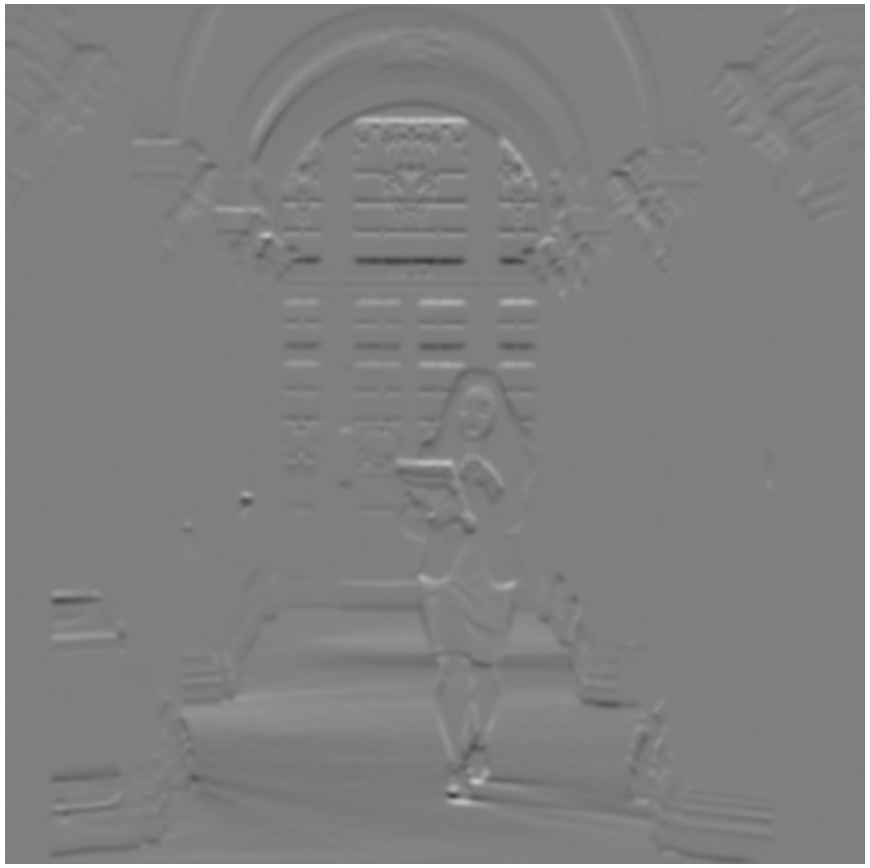




Compute Gradients

X' Derivative of Gaussian

Y Derivative of Gaussian



(x2 + 0.5 for visualization)

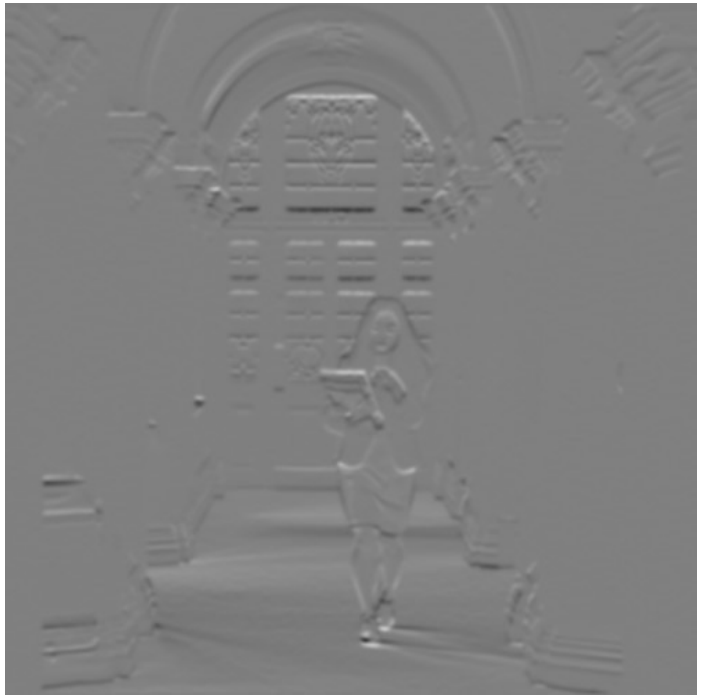
Canny Edge Detector

Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient

Compute Gradient Magnitude

sqrt(XDerivOfGaussian .^2 + YDerivOfGaussian .^2) = gradient magnitude

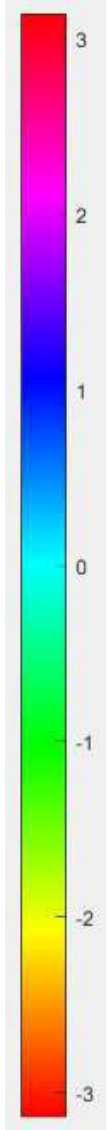


(x4 for visualization)

Compute Gradient Orientation

- Threshold magnitude at minimum level
- Get orientation via

```
theta = atan2(yDeriv, xDeriv)
```



Canny Edge Detector

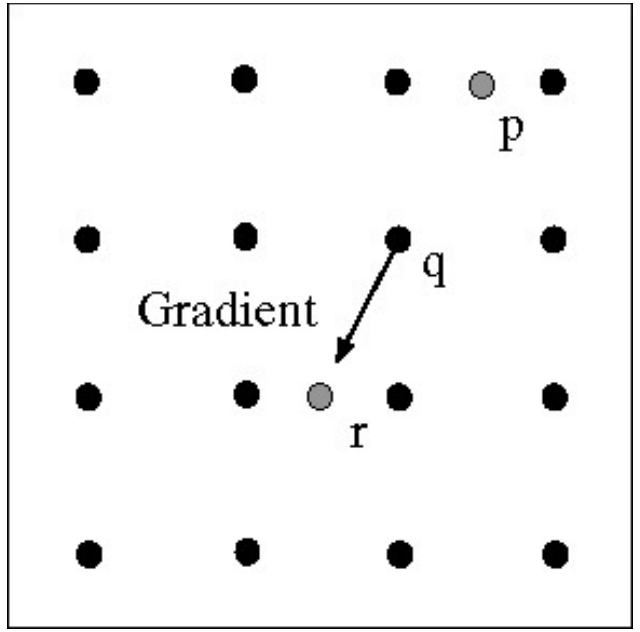
Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide "ridges" to single pixel width

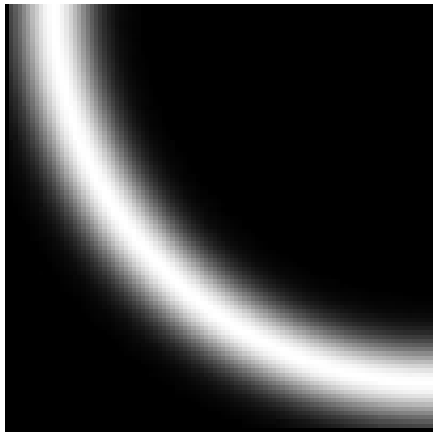
Non-Maximum Suppression for Each Orientation

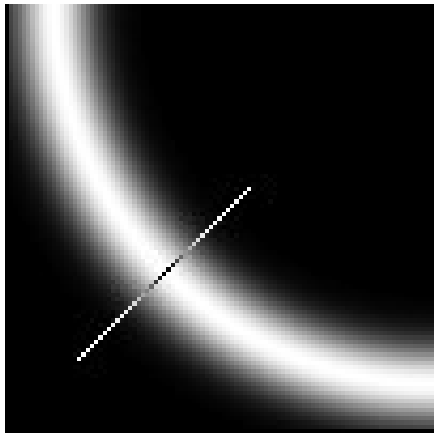
At pixel q:

We have a maximum if the value is larger than those at both p and at r.

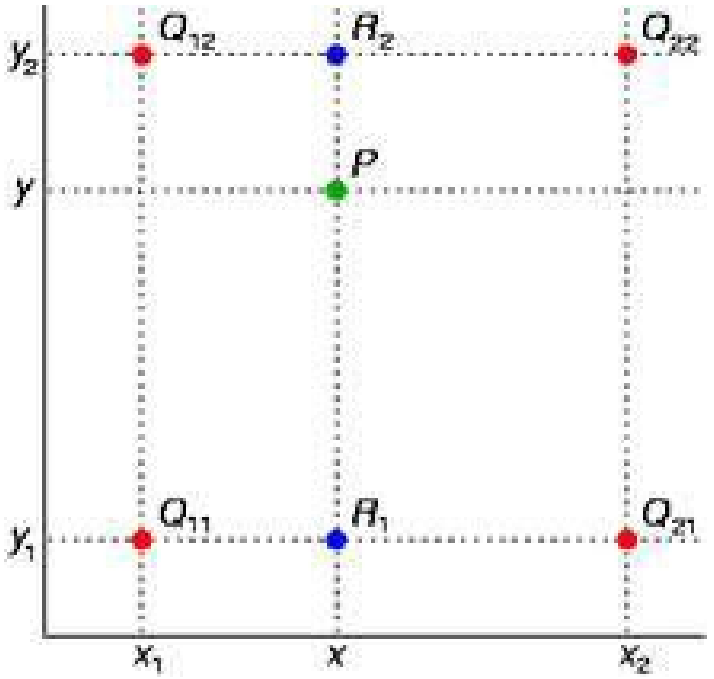


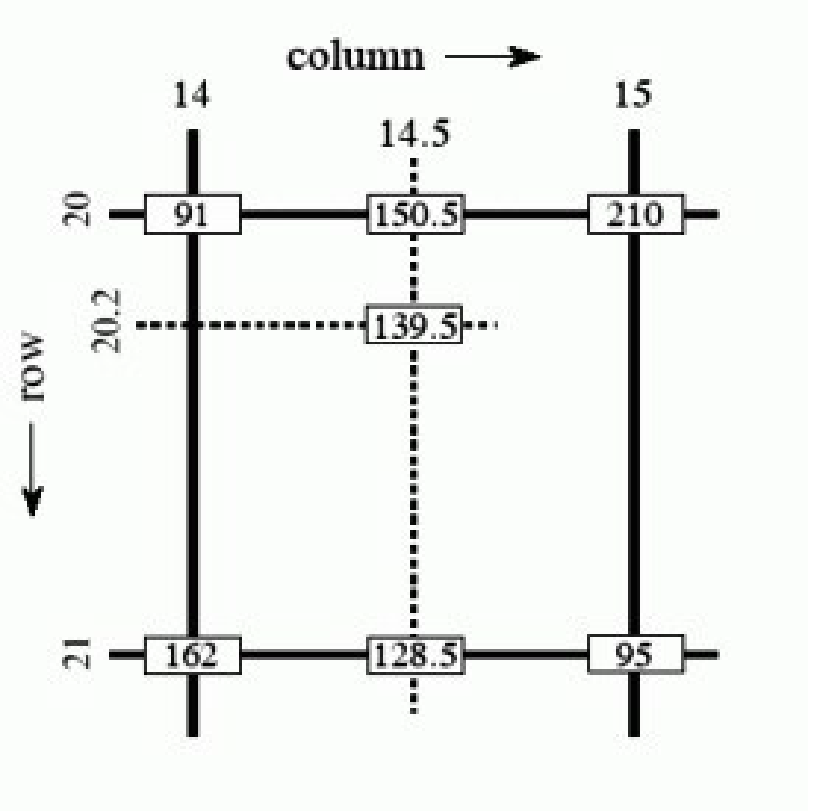
Interpolate along gradient direction to get these values.





Additional Slide: Bilinear Interpolation





$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

http://en.wikipedia.org/wiki/Bilinear_interpolation

Additional Slide: Interpolation Options

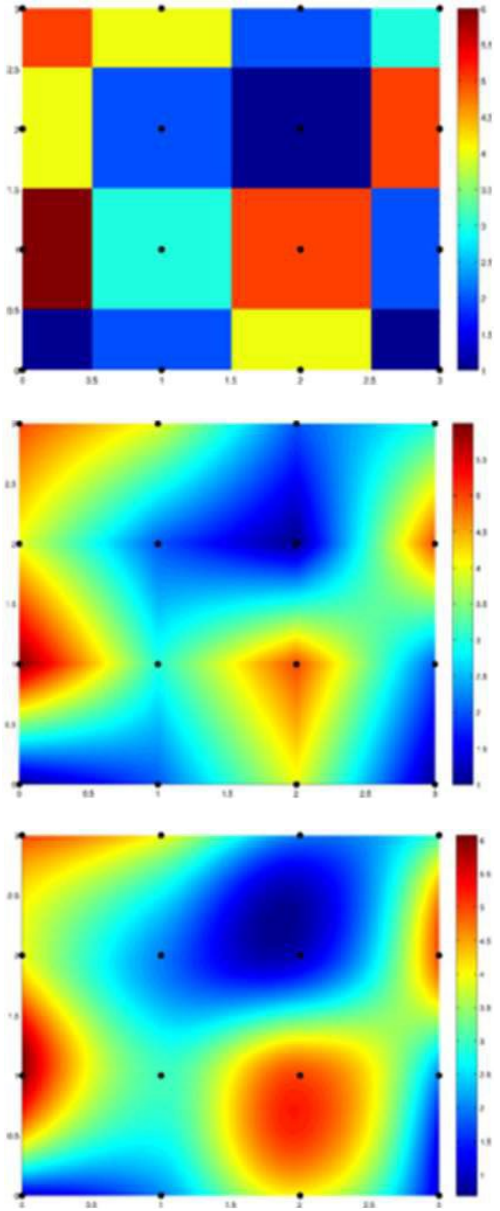
Image Scale

e.g., `skimage.transform.rescale(I, 2, order=x )`

x = 0 ➔ 'nearest neighbor'
- Copy value from nearest known
- Very fast but creates blocky edges

x = 1 ➔ 'bilinear' (default)
- Weighted average from four nearest known pixels
- Fast and reasonable results

x = 3 ➔ 'bicubic'
- Fit cubic spline to pixel intensities
- Non-linear interpolation over larger area (4x4)
- Slower, visually appealing, may create negative pixel values in cubic function fitting

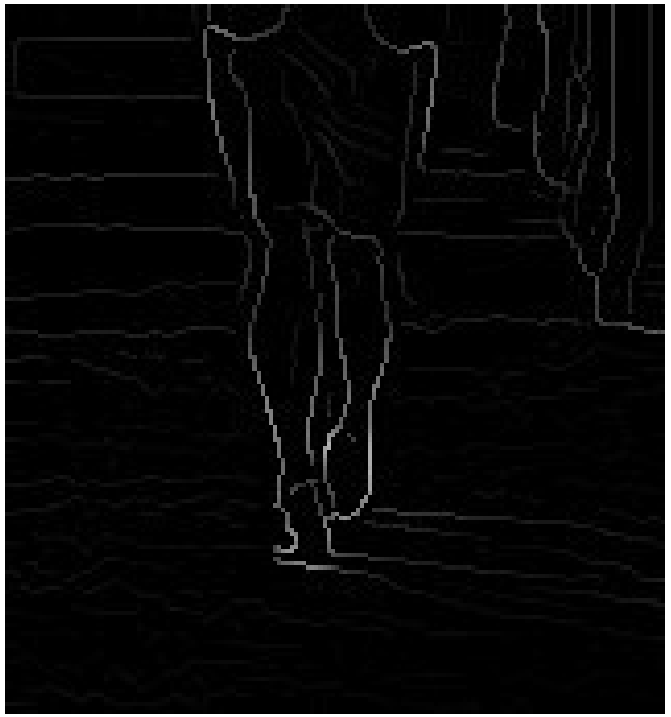


Examples from http://en.wikipedia.org/wiki/Bicubic_interpolation

Before Non-max Suppression

Gradient magnitude (x4 for visualization)

After Non-max Suppression



Gradient magnitude (x4 for visualization)

Canny Edge Detector

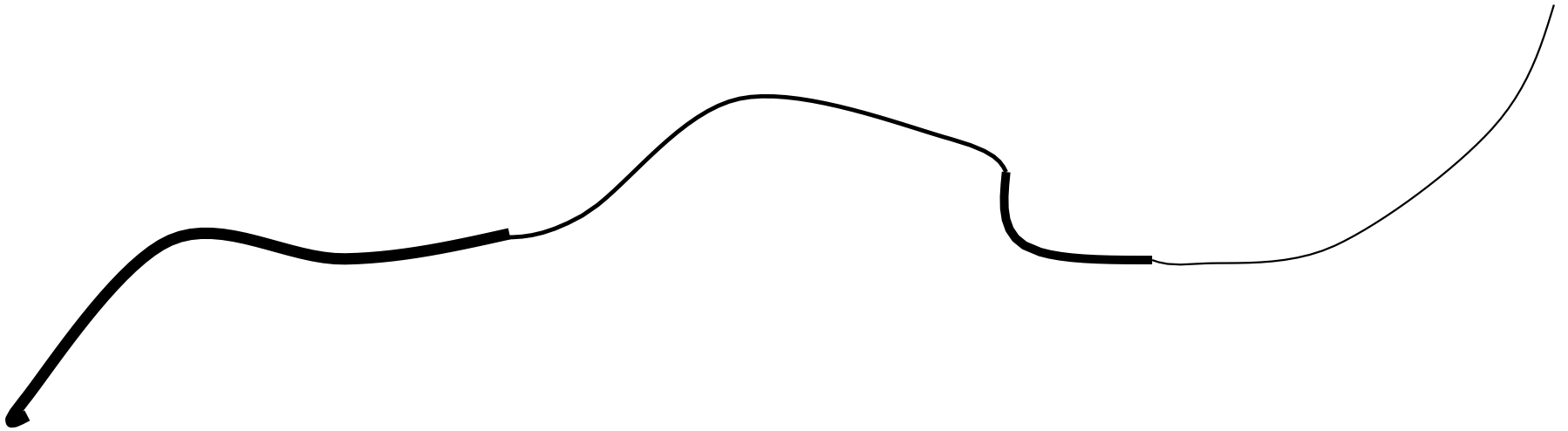
Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide "ridges" to single pixel width
4. 'Hysteresis' Thresholding

Hysteresis Thresholding

- Two thresholds – high and low
- Grad. mag. > high threshold? = strong edge
- Grad. mag. < low threshold? noise
- In between = weak edge

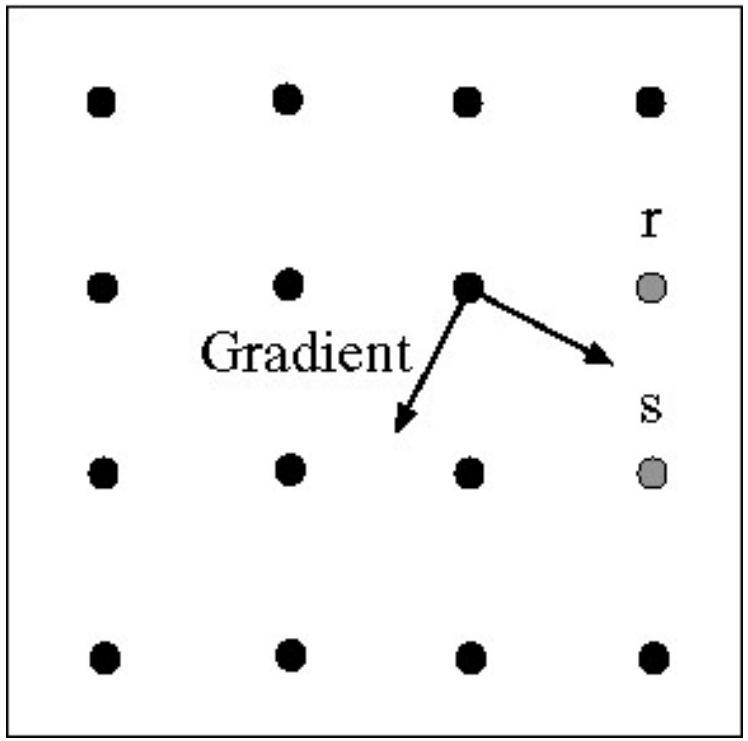
- **Edge linking:** ‘Follow’ edges starting from strong edge pixels
- Continue them into weak edges
- Connected components (Szeliski 3.3.4)

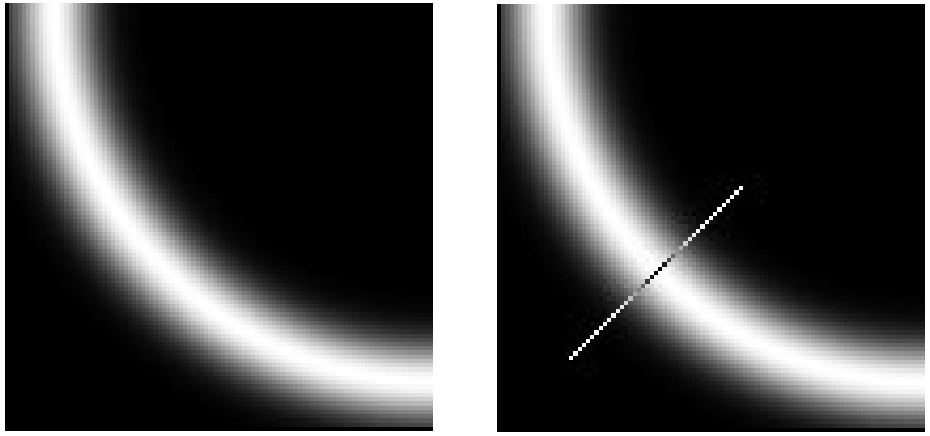


Steve Seitz

Additional Slide: Edge Linking

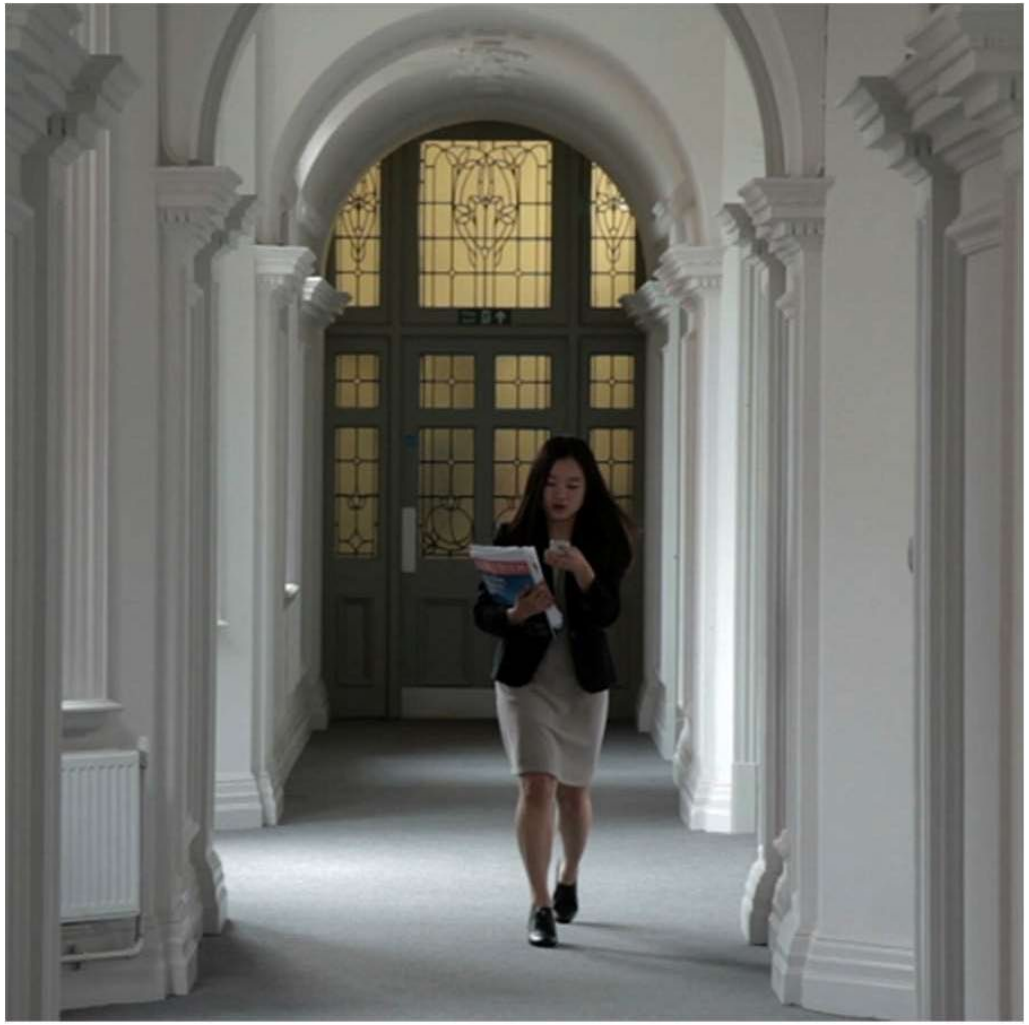
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s):





Final Canny Edges

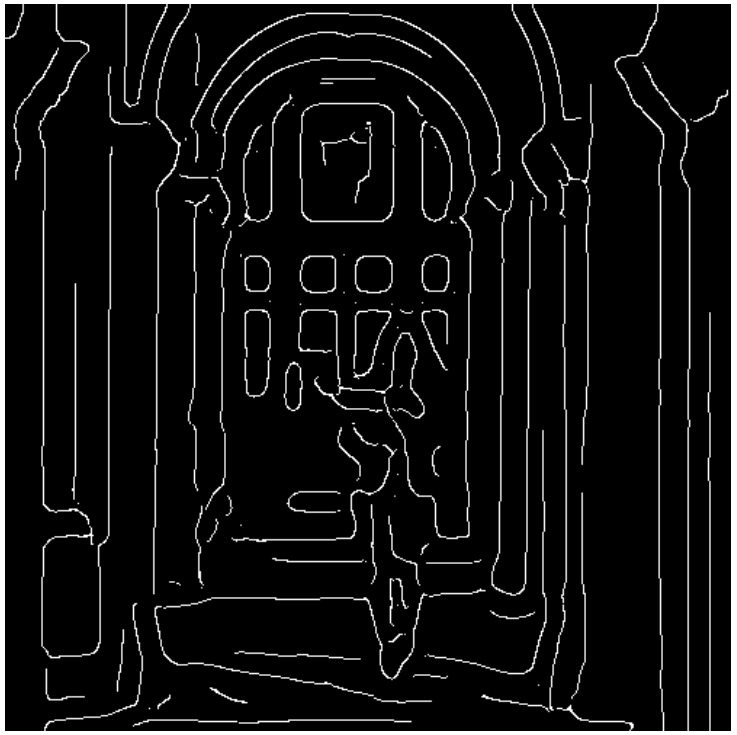
$$\sigma = \sqrt{2}, t_{\text{low}} = 0.05, t_{\text{high}} = 0.1$$



Effect of σ (Gaussian kernel spread/size)

Original

$\sigma = \sqrt{2}$



$\sigma = 4\sqrt{2}$

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

Canny Edge Detector

Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide "ridges" to single pixel width
4. 'Hysteresis' Thresholding:
   a. Define two thresholds: low and high
   b. Use the high threshold to start edge curves and the low threshold to continue them
   c. 'Follow' edges starting from strong edge pixels
      i. Connected components (Szeliski 3.3.4)

Demo

Controls:

- sigma: Gaussian sigma
- low_threshold: threshold on weak edges
- high_threshold: threshold on strong edges

```
skimage.feature.canny(image, sigma=1.0, low_threshold=None, high_threshold=None,
mask=None, use_quantiles=False, *, mode='constant', cval=0.0)          [source]
```

Edge filter an image using the Canny algorithm.

Parameters:

image : *2D array*

Grayscale input image to detect edges on; can be of any dtype.

sigma : *float, optional*

Standard deviation of the Gaussian filter.

low_threshold : *float, optional*

Lower bound for hysteresis thresholding (linking edges). If None, low_threshold is set to 10% of dtype's max.

high_threshold : *float, optional*

Upper bound for hysteresis thresholding (linking edges). If None, high_threshold is set to 20% of dtype's max.