

**Spring 2022**

# INTRODUCTION TO COMPUTER VISION

---

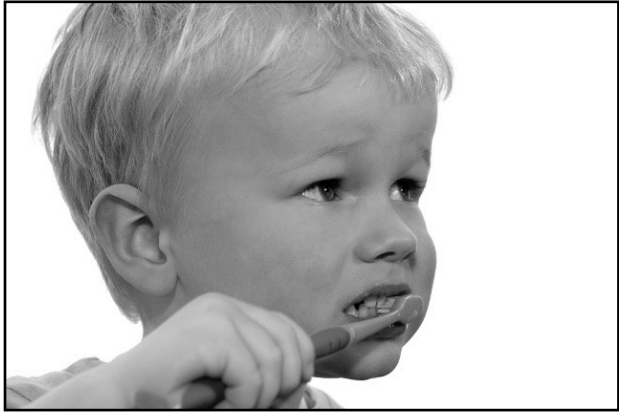
**Atlas Wang**

Assistant Professor, The University of Texas at Austin

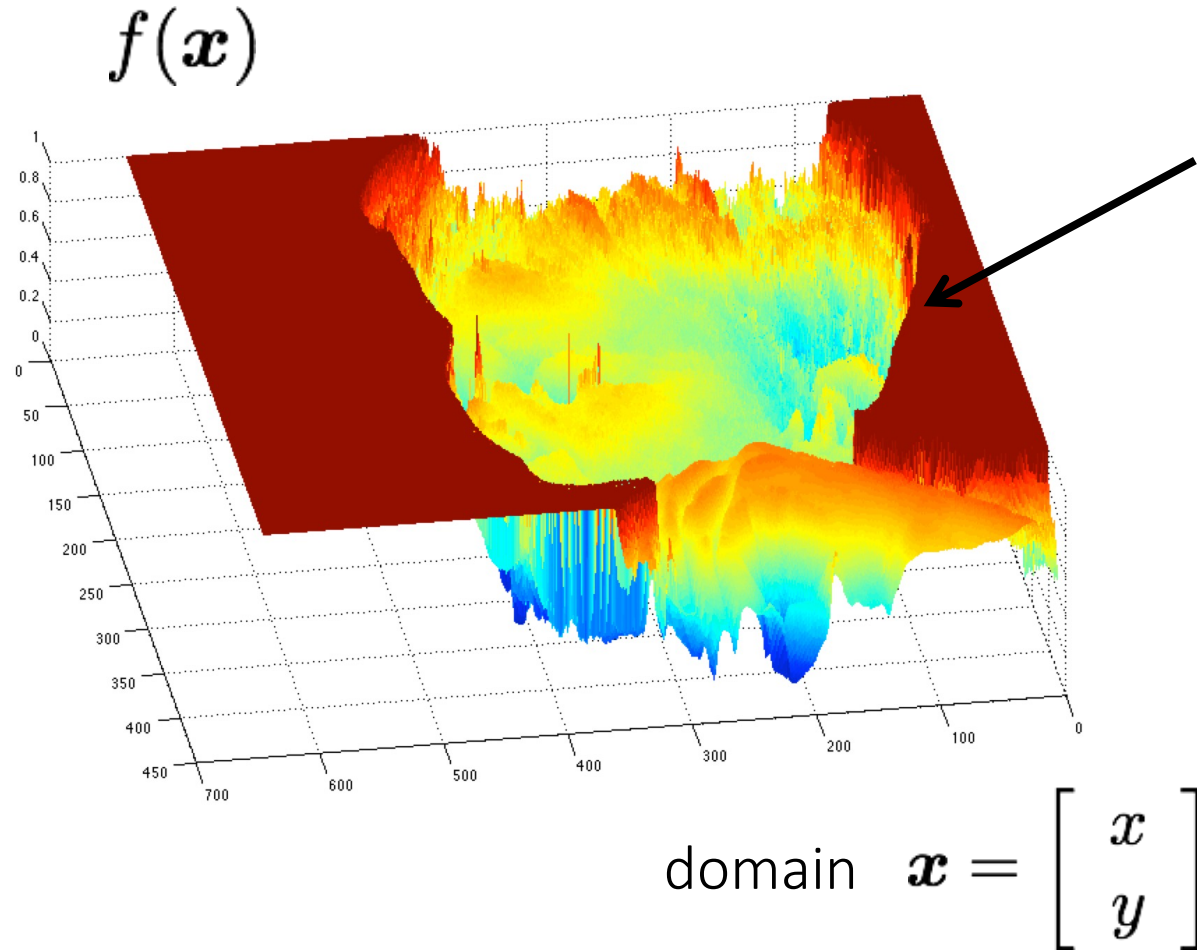
**Visual Informatics Group@UT Austin**  
<https://vita-group.github.io/>

Many slides here were adapted from CMU 16-385 Computer Vision

# What are image edges?



grayscale image



# 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

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

# Finite differences

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

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

Alternative: use central difference

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

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

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

Alternative: use central difference

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

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

Vertical Sober filter:

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

=

1
2
1

\*

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

“Derivative”

“Blurring”

Horizontal Sobel filter:

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

=

1
0
-1

\*

1	2	1
---	---	---

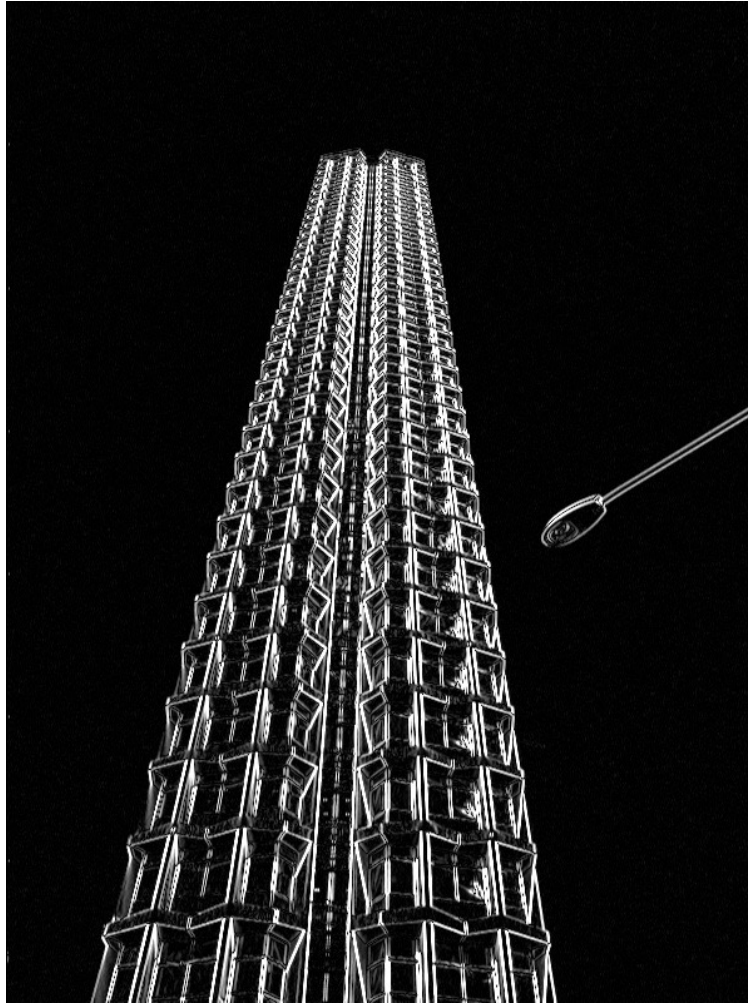
“Blurring”

“Derivative”

# Sobel filter example



original



which Sobel filter?



which Sobel filter?

# Computing image gradients

1. Select your favorite derivative filters.

$$S_x = \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$

$$S_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

# Computing image gradients

1. Select your favorite derivative filters.

$$\mathbf{S}_x = \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\mathbf{S}_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial f}{\partial x} = \mathbf{S}_x \otimes f$$

$$\frac{\partial f}{\partial y} = \mathbf{S}_y \otimes f$$

# Computing image gradients

1. Select your favorite derivative filters.

$$\mathbf{S}_x = \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\mathbf{S}_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \mathbf{f}}{\partial x} = \mathbf{S}_x \otimes \mathbf{f}$$

$$\frac{\partial \mathbf{f}}{\partial y} = \mathbf{S}_y \otimes \mathbf{f}$$

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

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

gradient

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

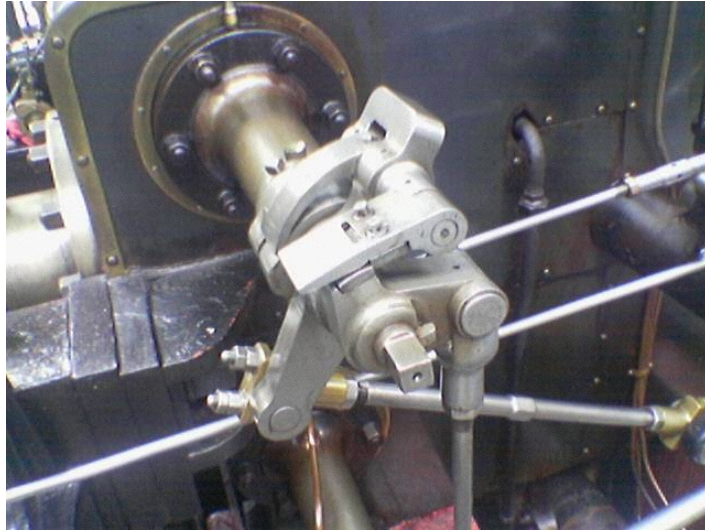
direction

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

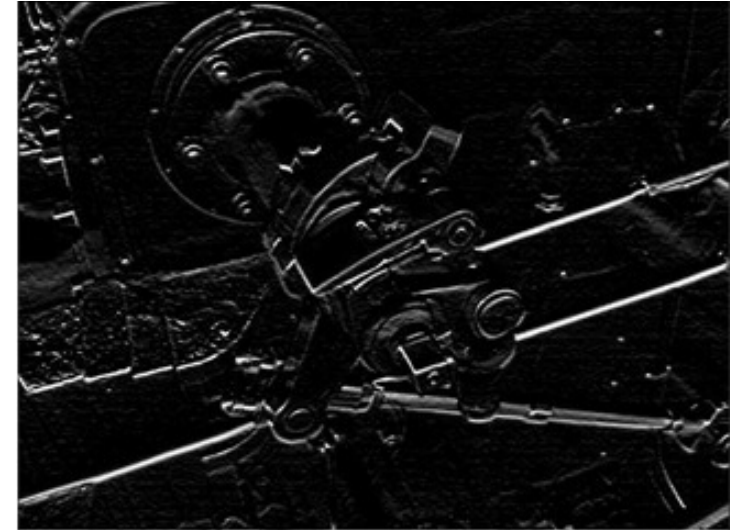
amplitude

# Image gradient example

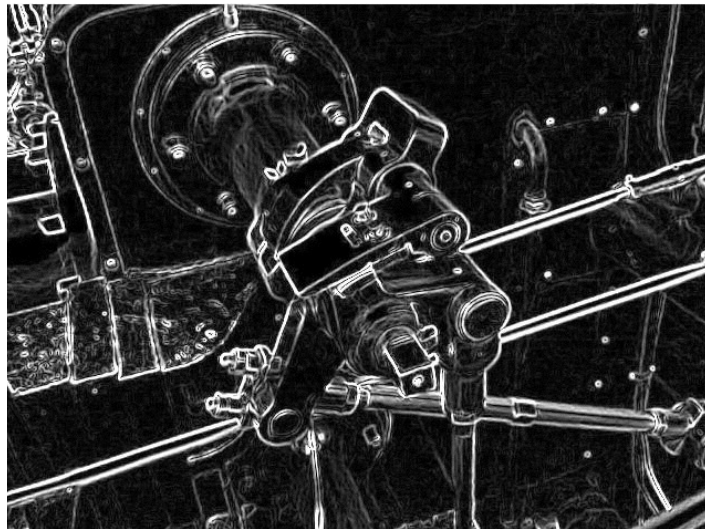
original



vertical  
derivative



gradient  
amplitude



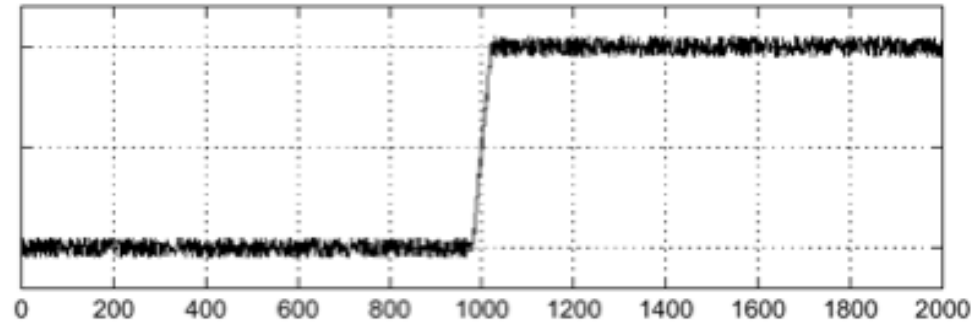
horizontal  
derivative



How does the gradient direction relate to these edges?

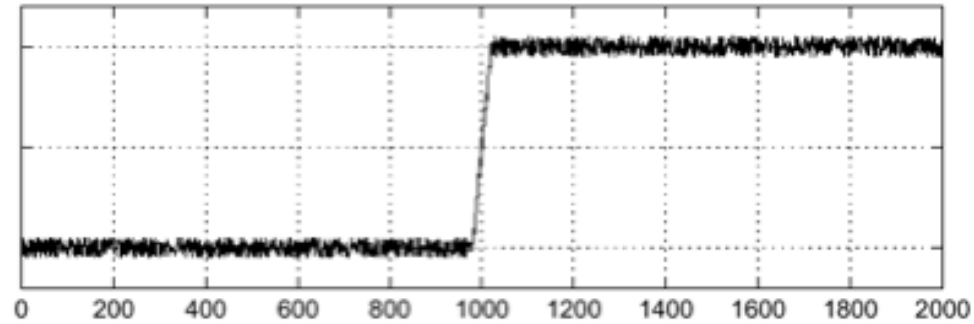
# How do you find the edge of this signal?

intensity plot



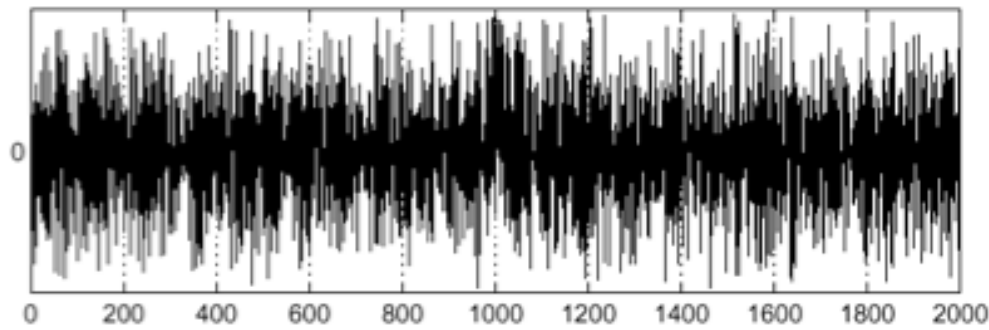
# How do you find the edge of this signal?

intensity plot



Using a derivative filter:

derivative plot

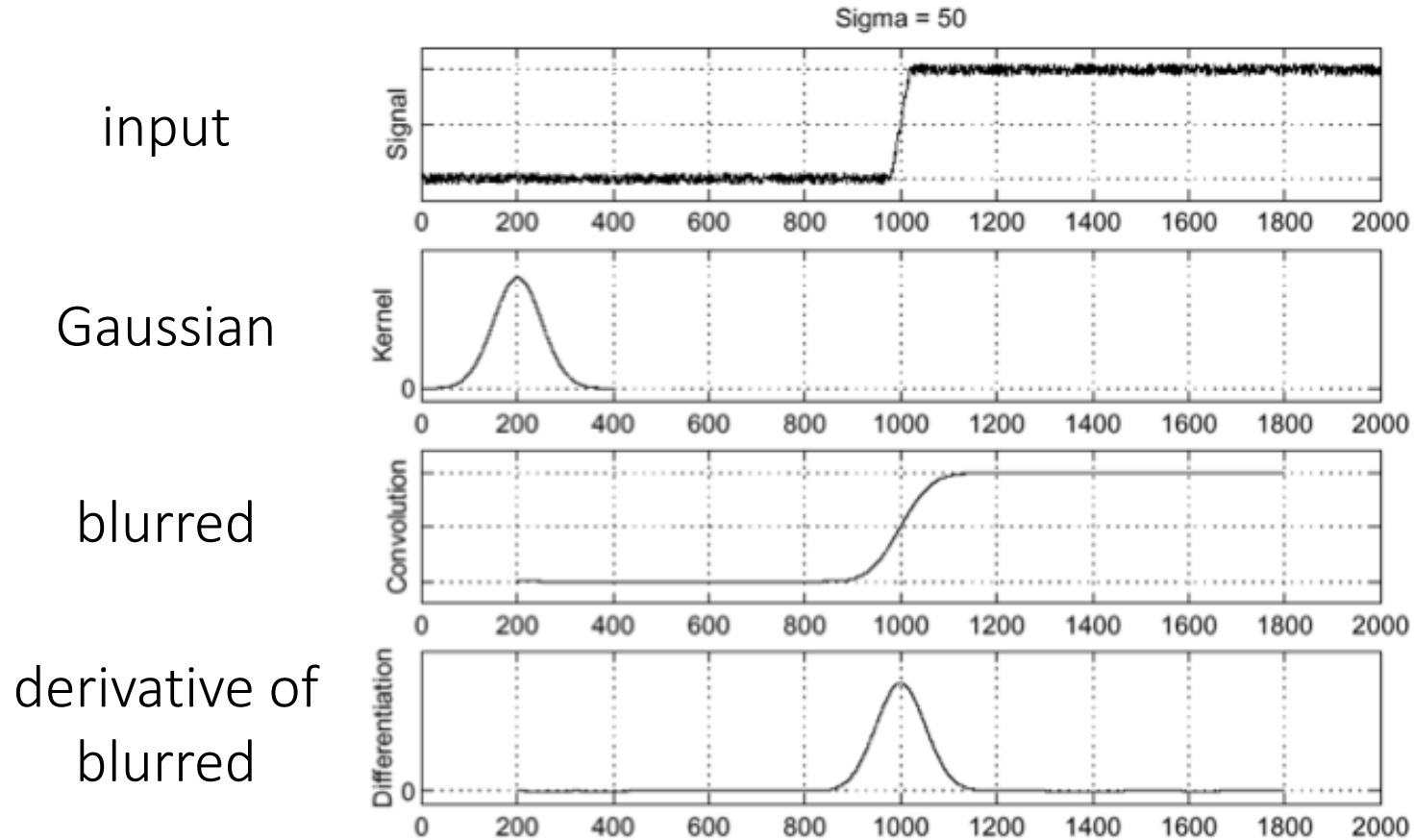


What's the problem here?



# Differentiation is very sensitive to noise

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

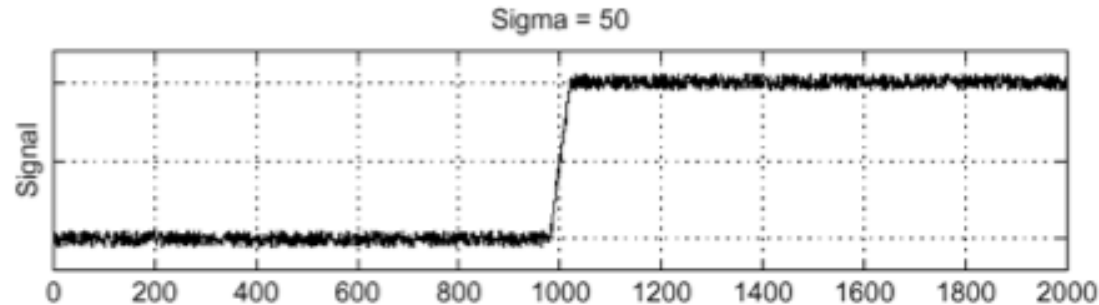


How much  
should we blur?

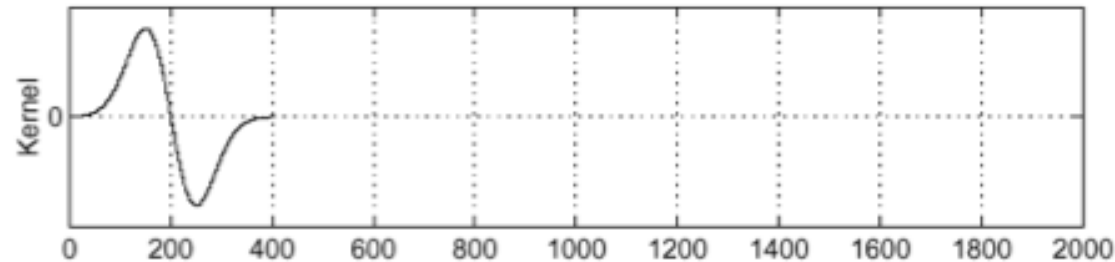
# Derivative of Gaussian (DoG) filter

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

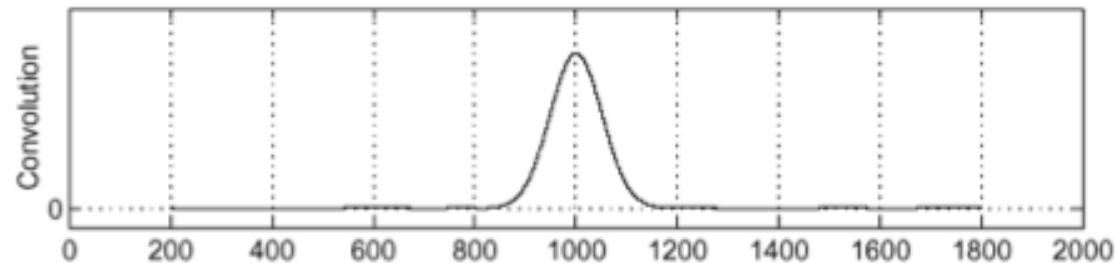
input



derivative of  
Gaussian

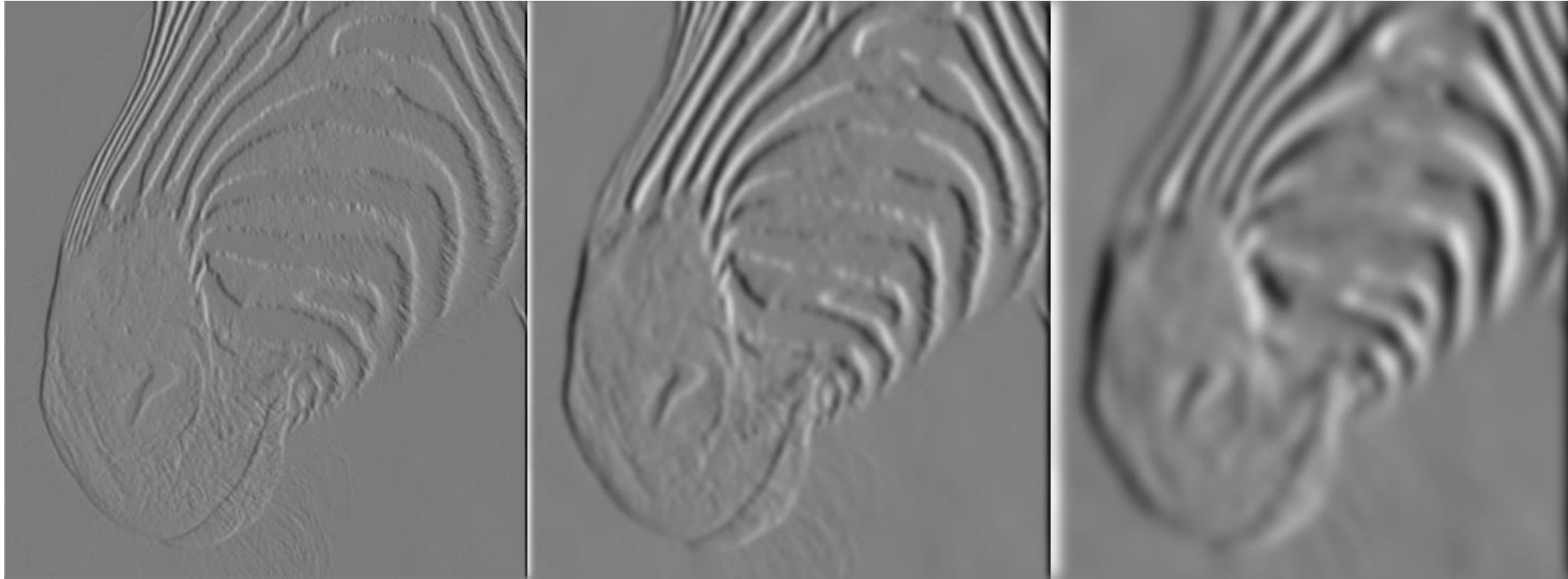


output (same  
as before)



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

# Tradeoff between smoothing and localization



1 pixel

3 pixels

7 pixels

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

# Laplace filter

Basically a second derivative filter.

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

first-order  
finite difference

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

→

1D derivative filter

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

second-order  
finite difference

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

→

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

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

→

1D derivative filter

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

second-order  
finite difference

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

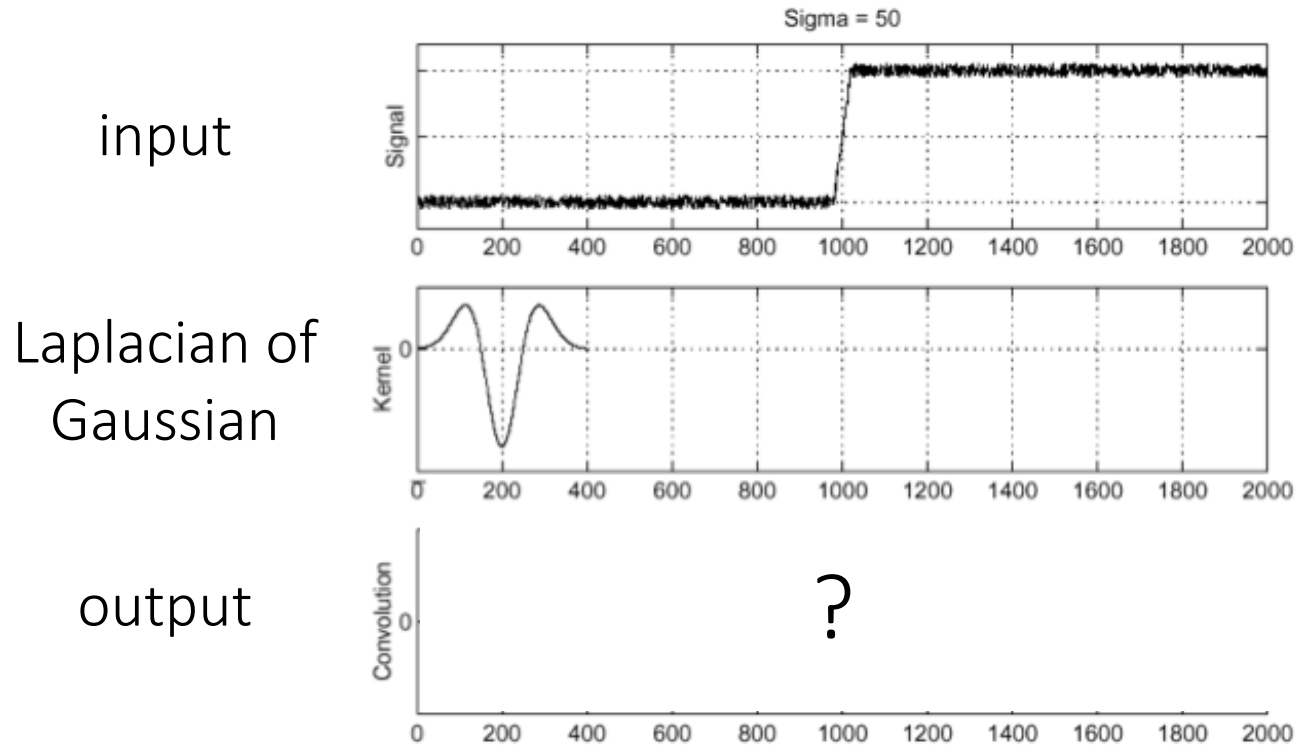
→

Laplace filter

1	-2	1
---	----	---

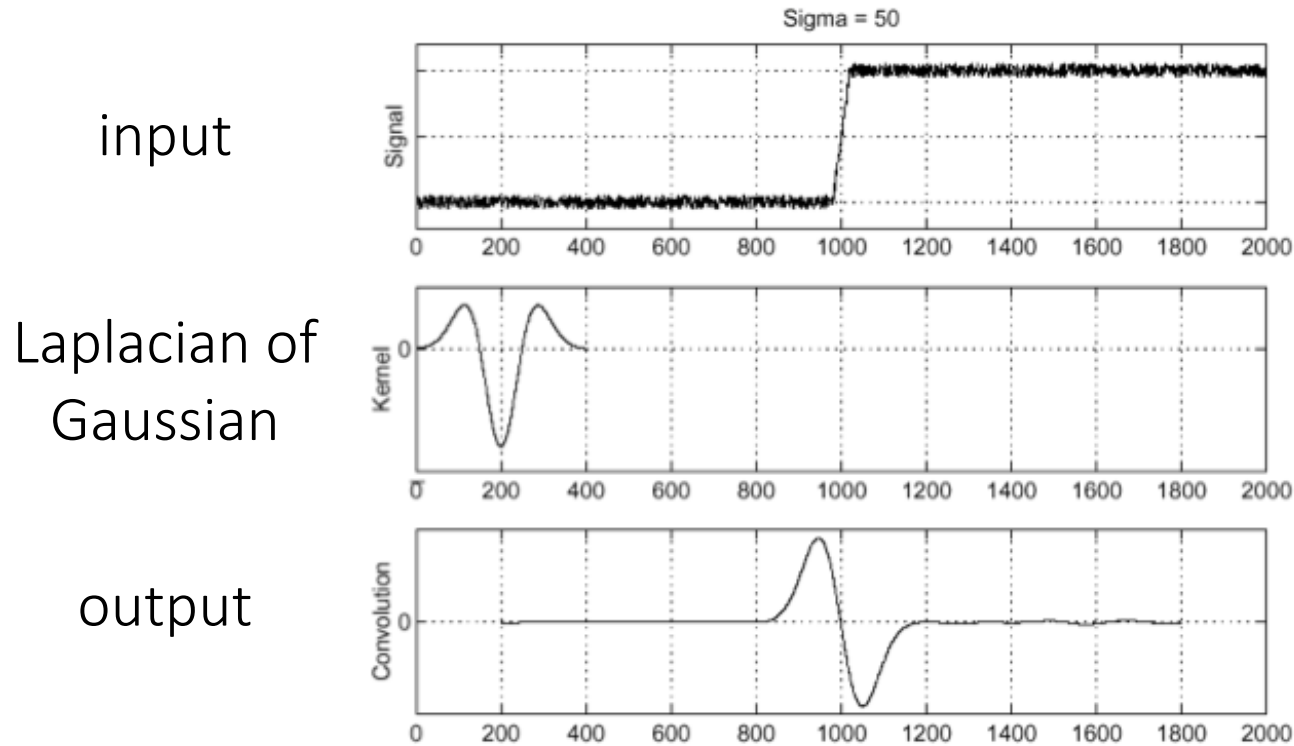
# Laplacian of Gaussian (LoG) filter

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



# Laplacian of Gaussian (LoG) filter

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



“zero crossings” at edges

# Laplace and LoG filtering examples



Laplacian of Gaussian filtering



Laplace filtering



# Laplacian of Gaussian vs Derivative of Gaussian

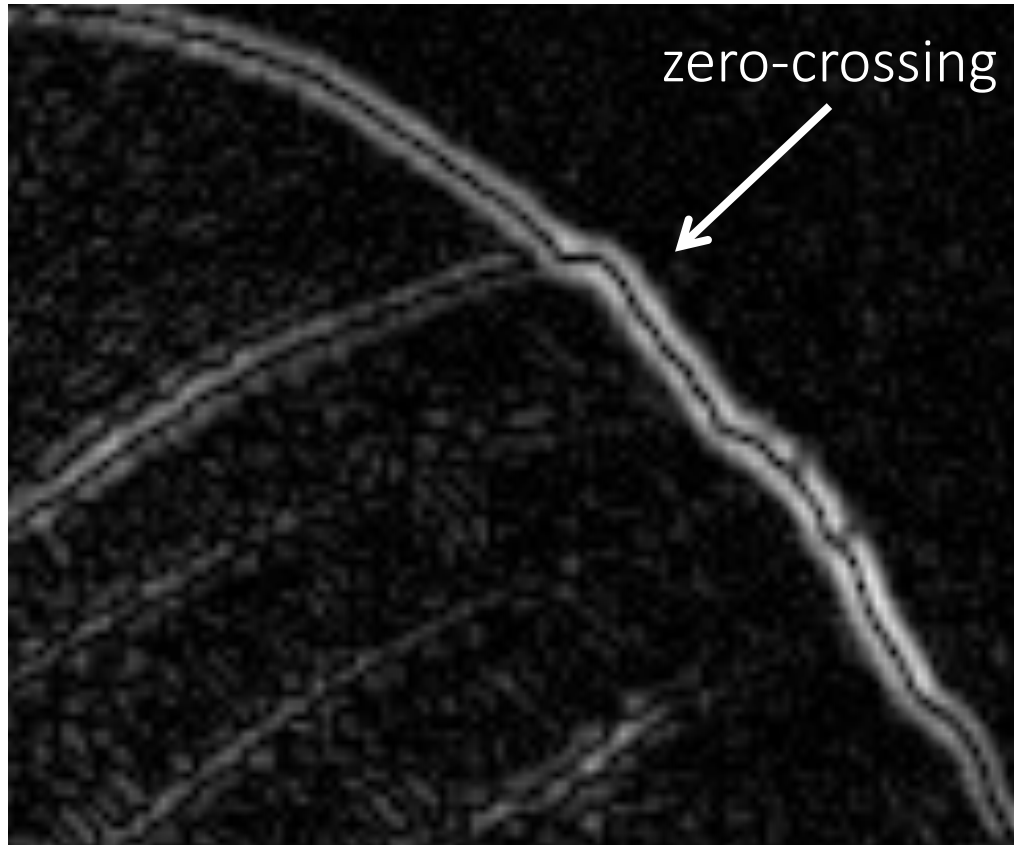


Laplacian of Gaussian filtering

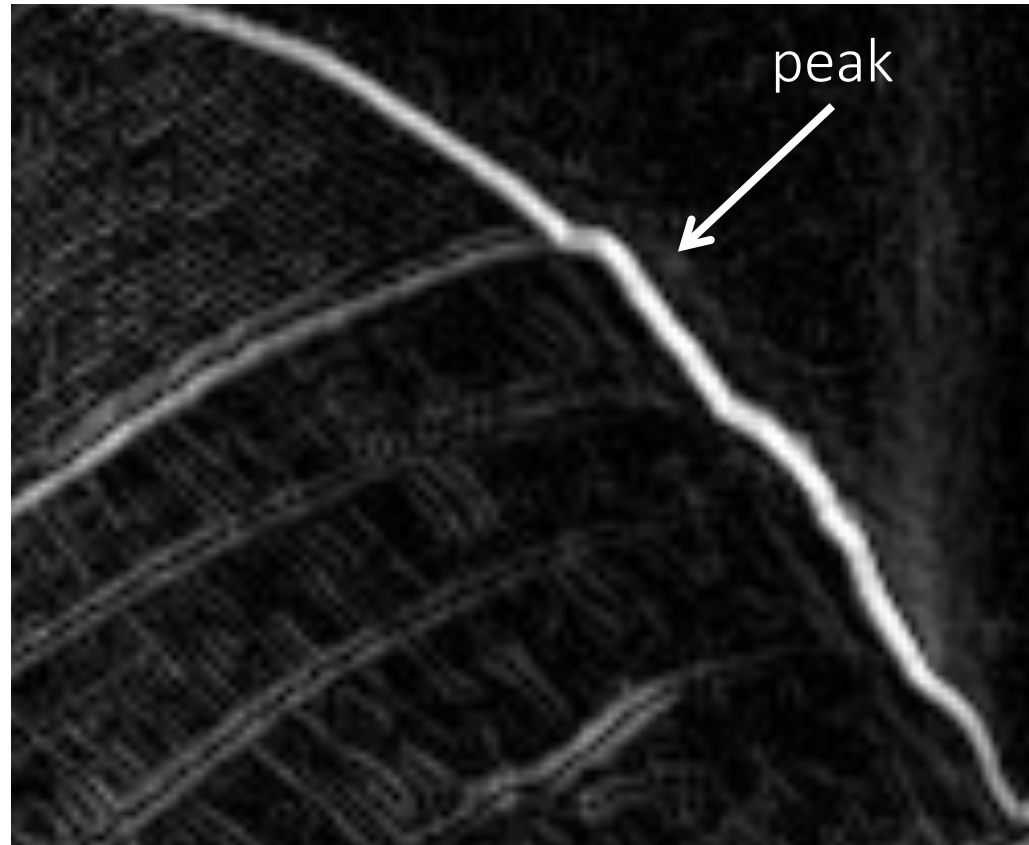


Derivative of Gaussian filtering

# 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).

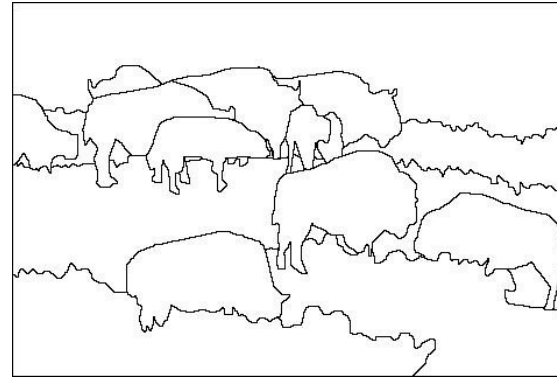
# But Wait ... Is Pixel Difference the Final Answer?

*Where do  
humans see  
boundaries?*

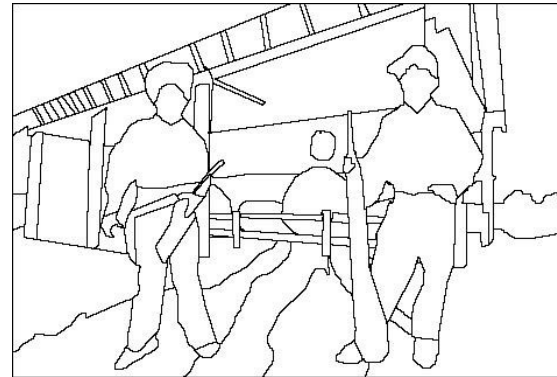
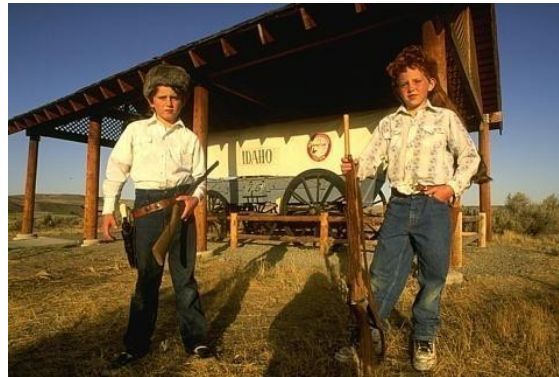
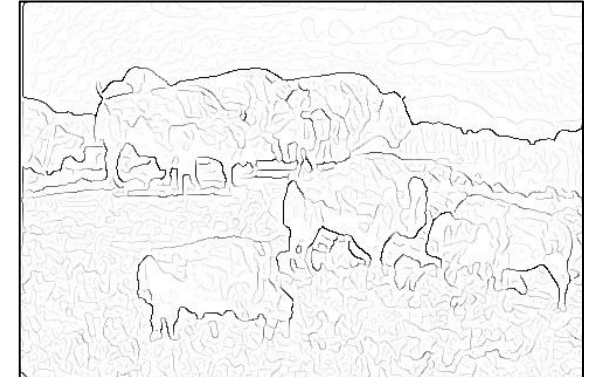
image



human segmentation



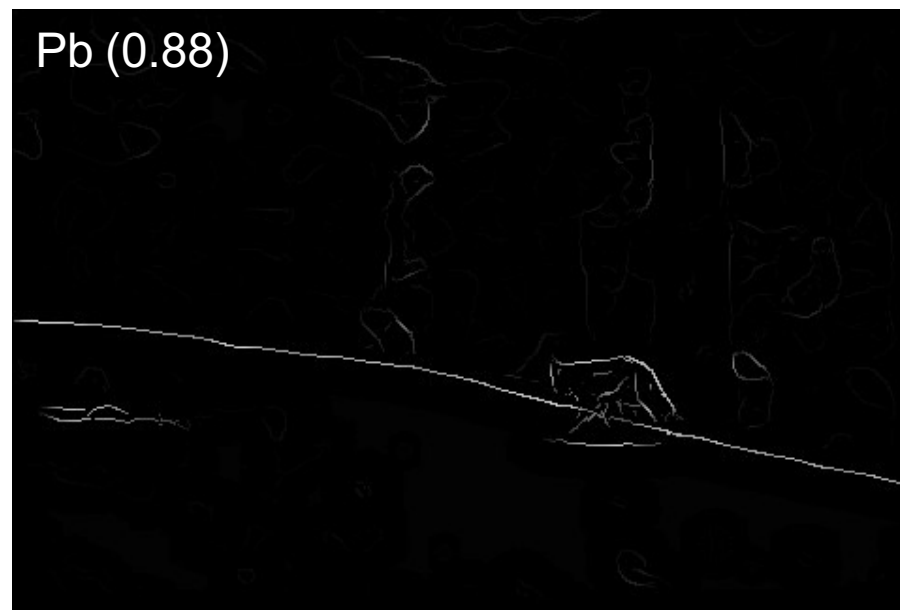
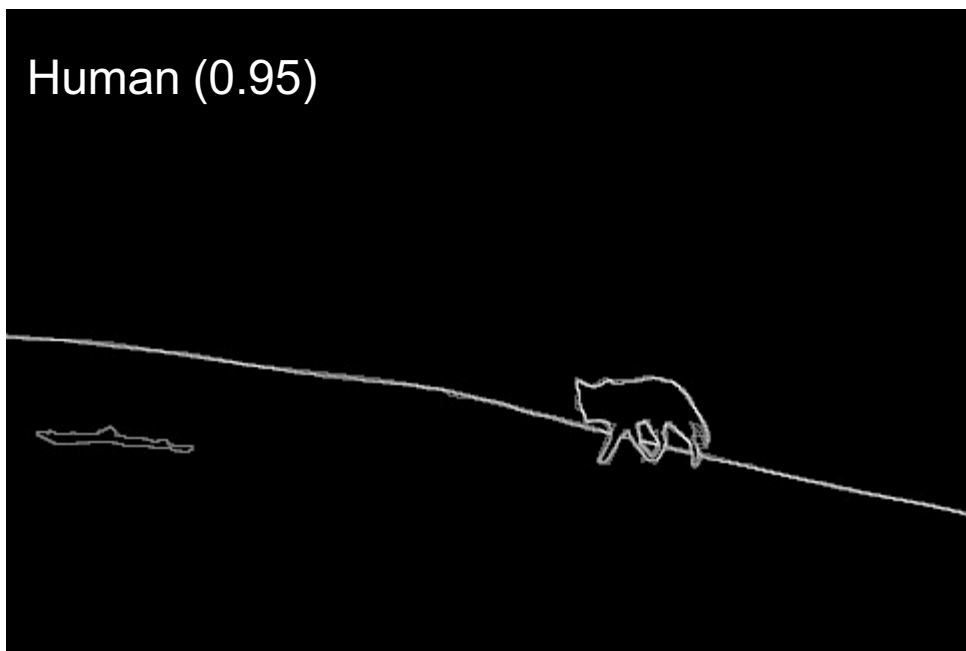
gradient magnitude



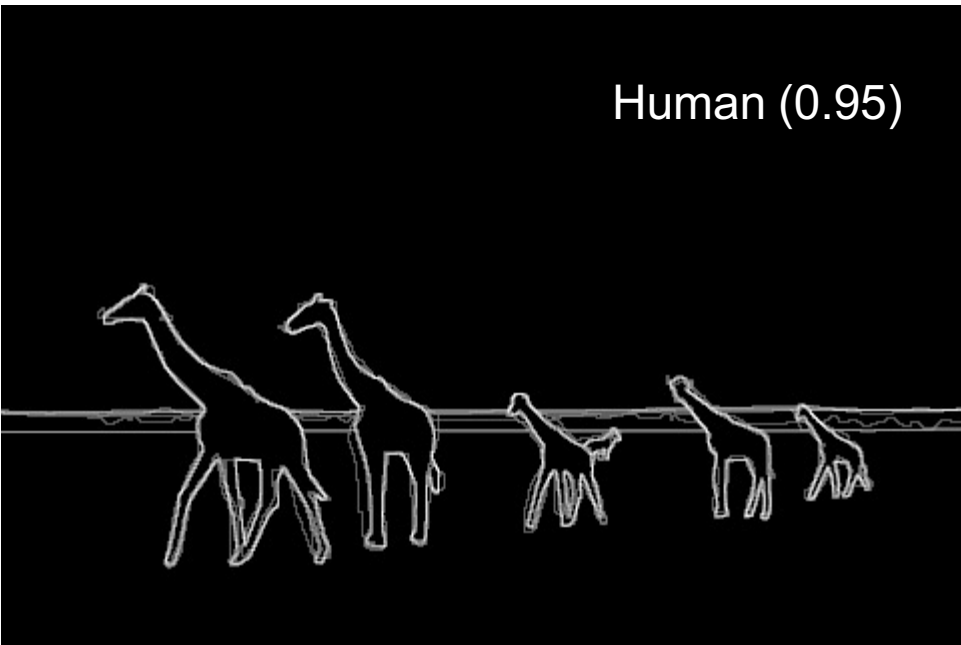
- Berkeley segmentation database:  
<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/>



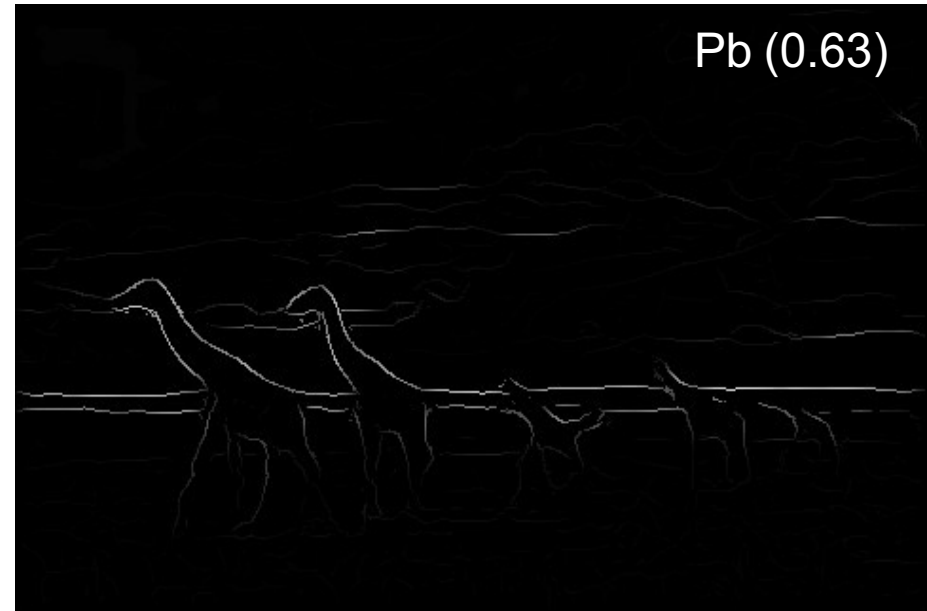
Score = confidence of edge.  
For humans, this is averaged across  
multiple participants.







Score = confidence of edge.  
For humans, this is averaged across  
multiple participants.



# Canny Edge Detector

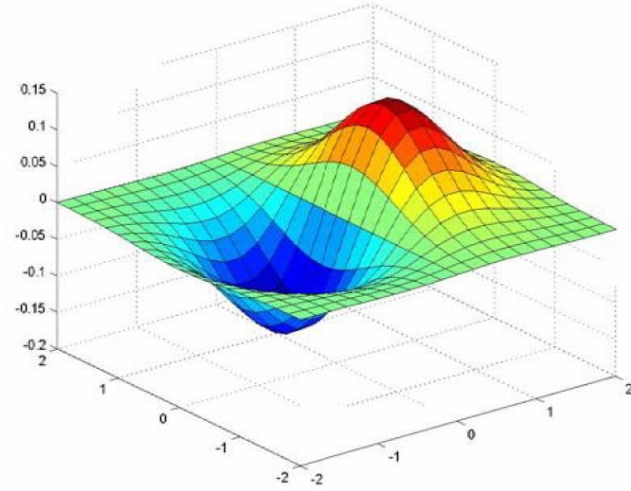
- Arguably **the most widely used** edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise



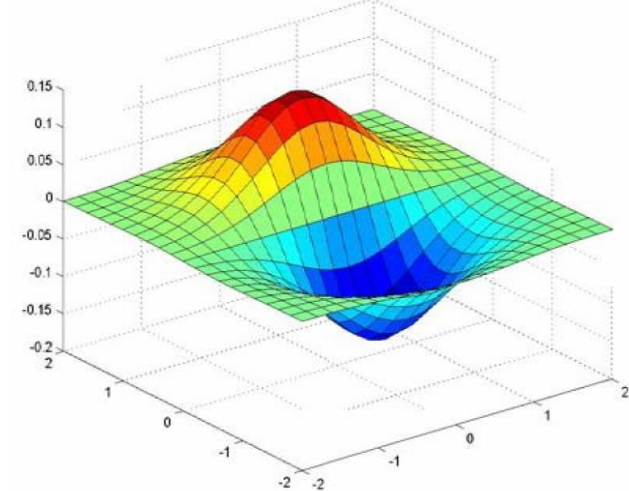
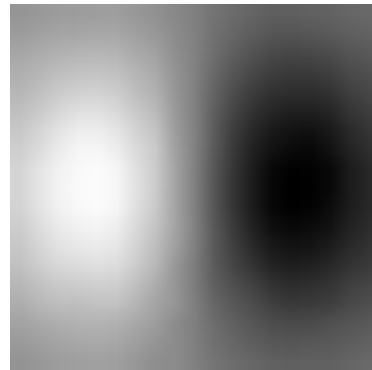
# Canny edge detector

1. Filter image with x, y derivatives of Gaussian

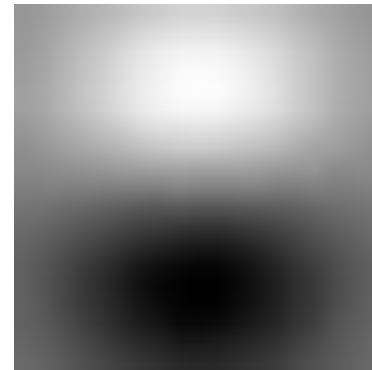
# Derivative of Gaussian filter



x-direction



y-direction





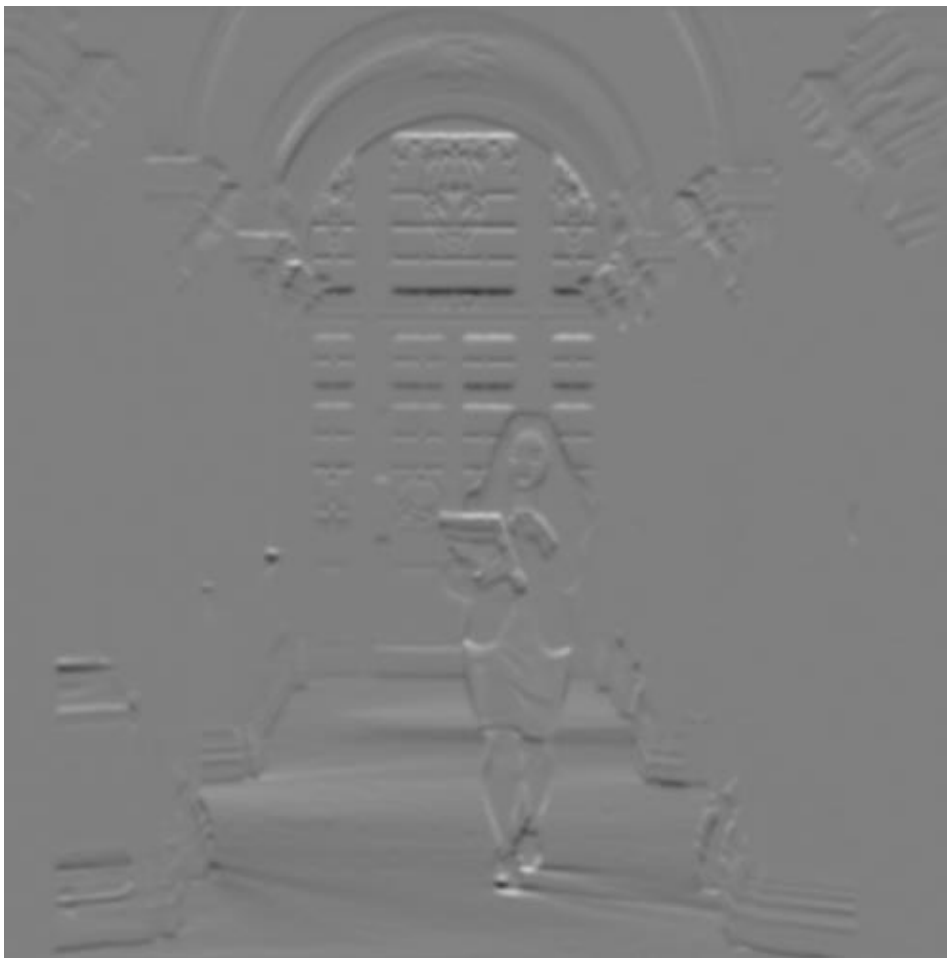
# Compute Gradients



X Derivative of Gaussian



Y Derivative of Gaussian



(x2 + 0.5 for visualization)

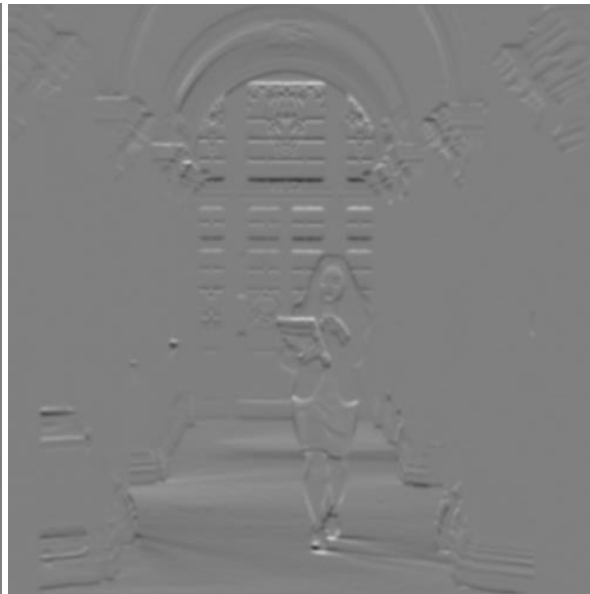
# Canny edge detector

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

# Compute Gradient Magnitude



$$\text{sqrt}( \text{XDerivOfGaussian}.^2 + \text{YDerivOfGaussian}.^2 ) = \text{gradient magnitude}$$



# Compute Gradient Orientation

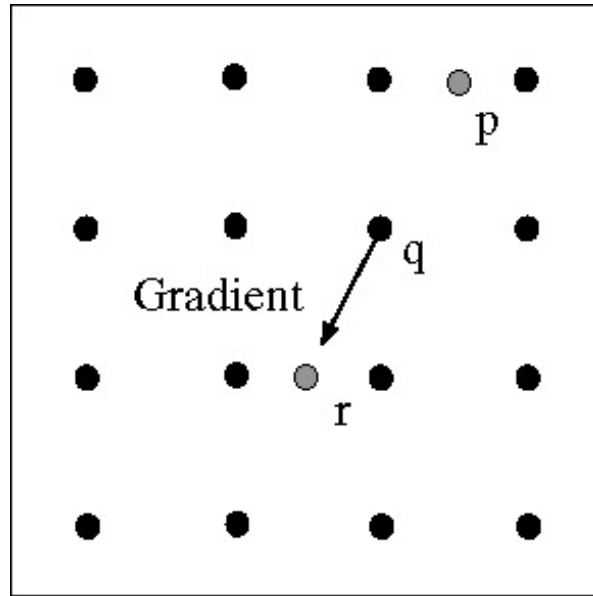
- Threshold magnitude at minimum level
- Get orientation via  $\theta = \text{atan2}(y\text{Deriv}, x\text{Deriv})$



# Canny edge detector

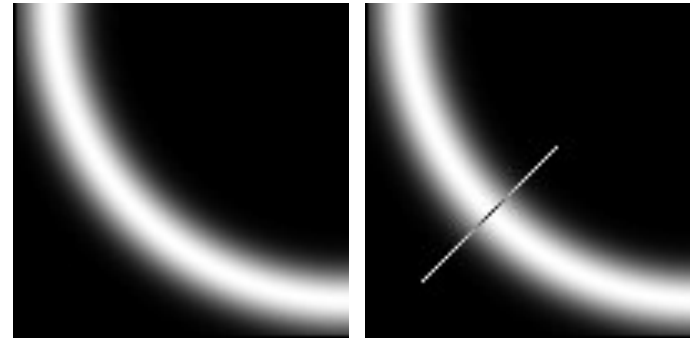
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
  - Thin multi-pixel wide “ridges” to single pixel width

# Non-maximum suppression for each orientation

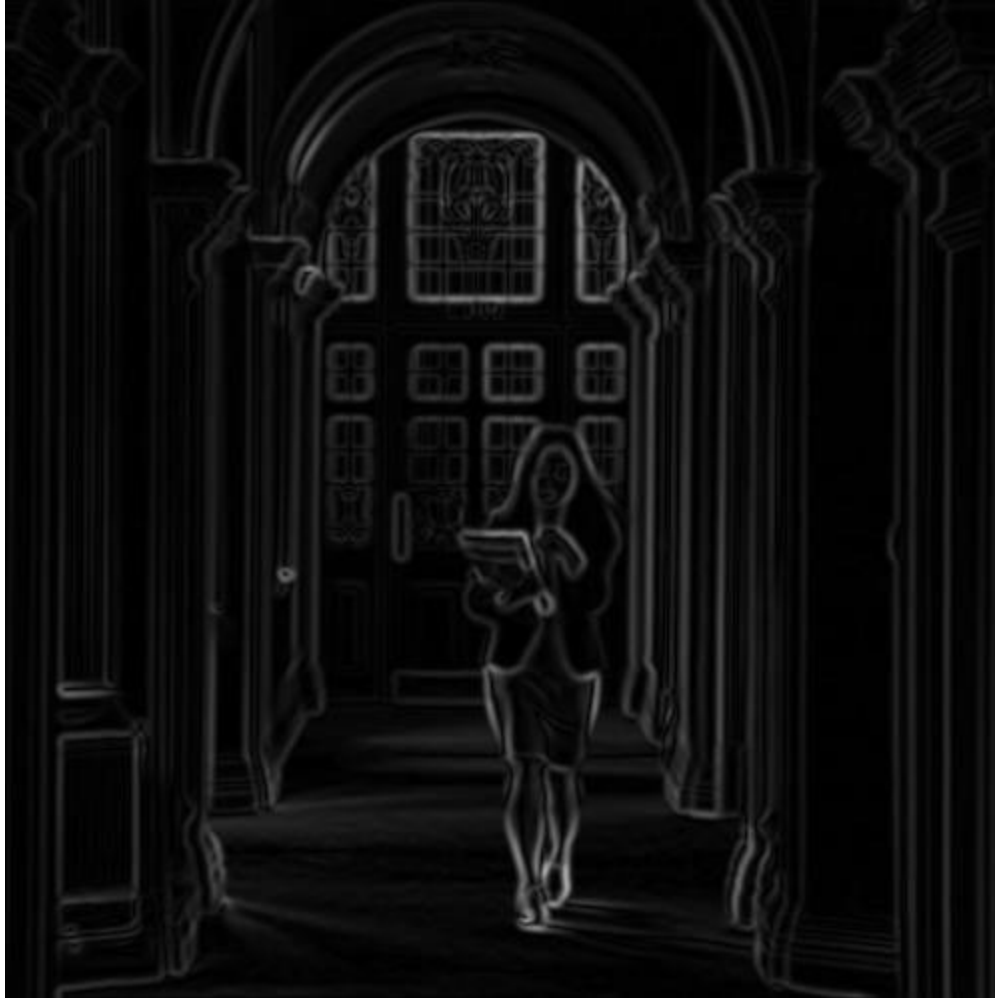


At pixel q:  
We have a maximum if the  
value is larger than those at  
both p and at r.

Interpolate along gradient  
direction to get these values.

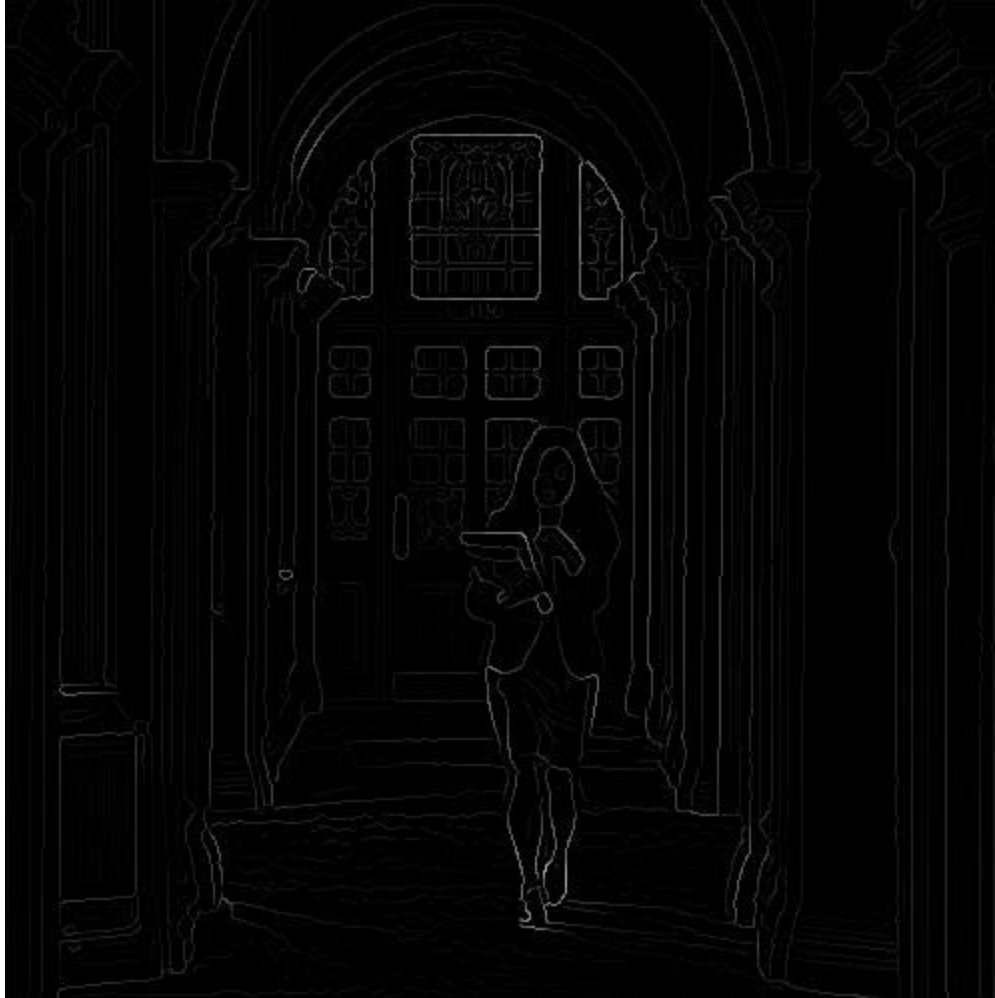


# Before Non-max Suppression



Gradient magnitude (x4 for visualization)

# After non-max suppression



Gradient magnitude (x4 for visualization)

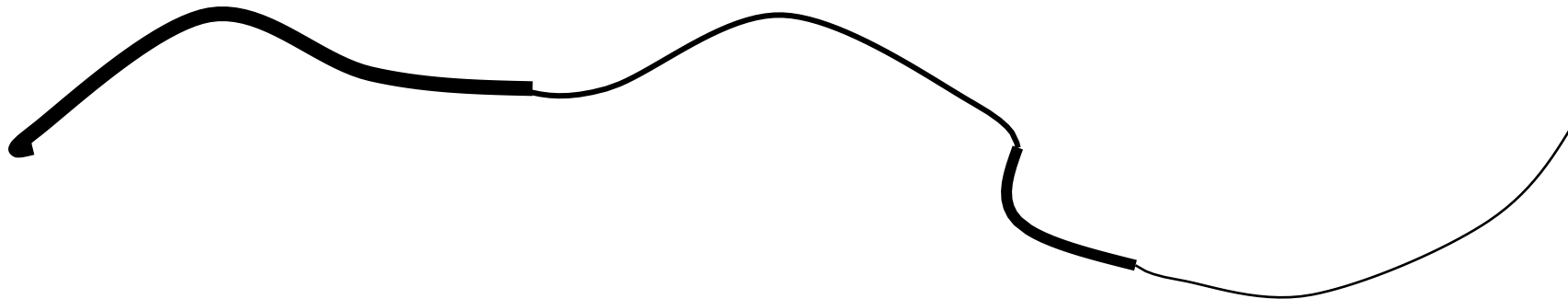


# Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
  - 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

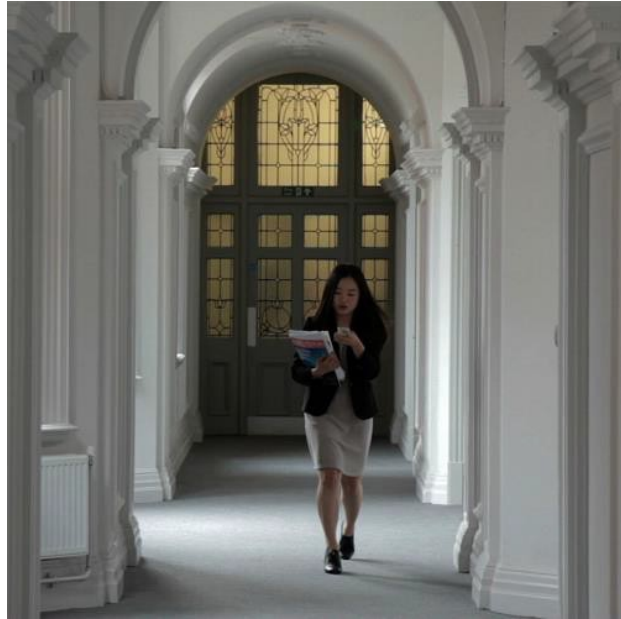


# Final Canny Edges

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



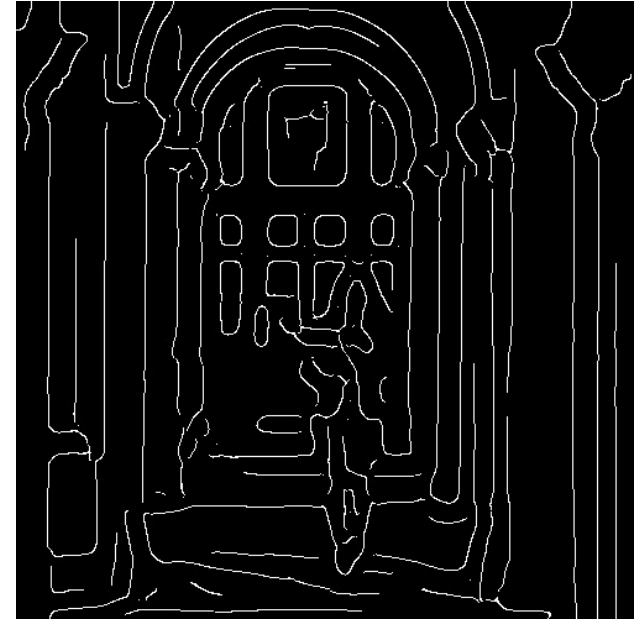
# Effect of $\sigma$ (Gaussian kernel spread/size)



Original



$\sigma = \sqrt{2}$



$\sigma = 4\sqrt{2}$

The choice of  $\sigma$  depends on desired behavior

- large  $\sigma$  detects large scale edges
- small  $\sigma$  detects fine features

# Canny edge detector

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

**Python: e.g., `skimage.feature.canny()`**



The University of Texas at Austin  
**Electrical and Computer  
Engineering**  
*Cockrell School of Engineering*