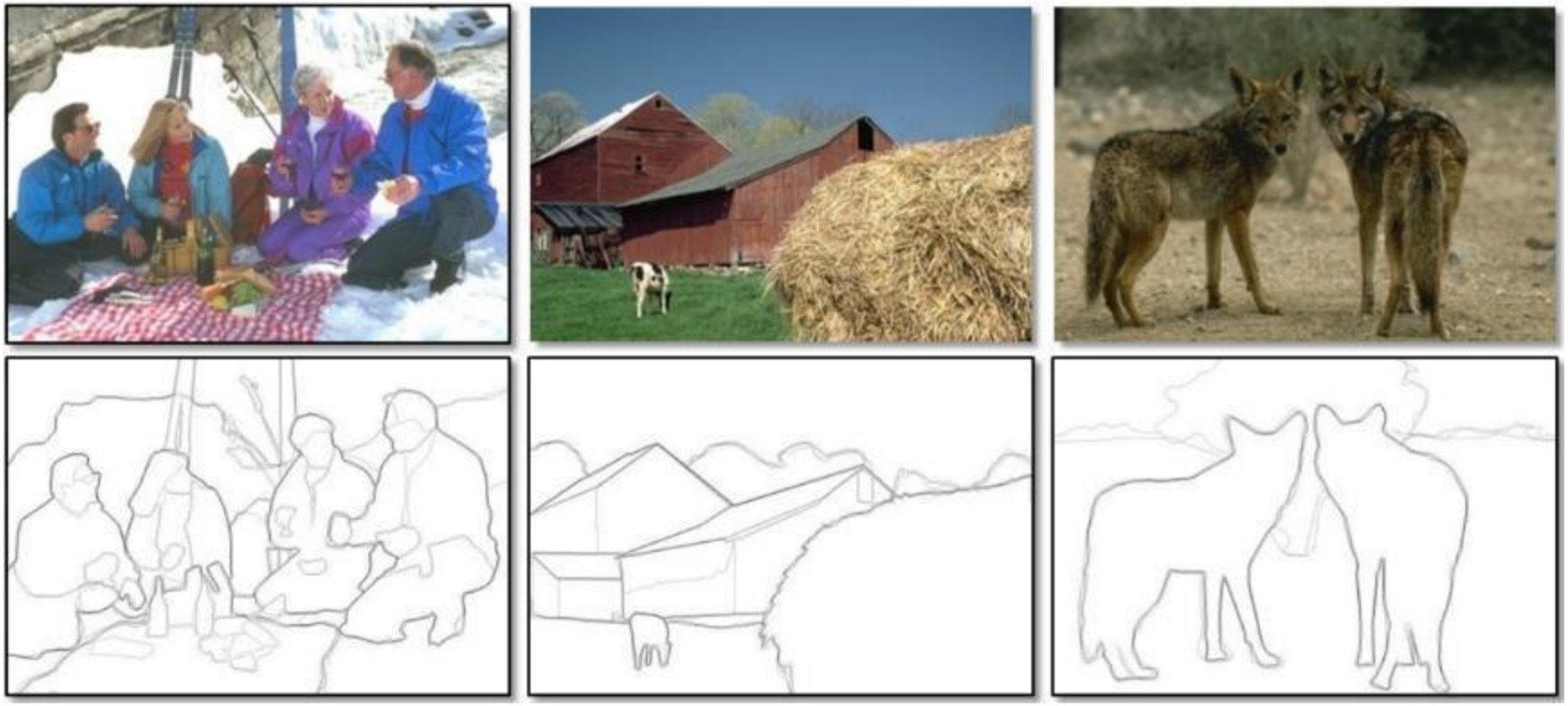
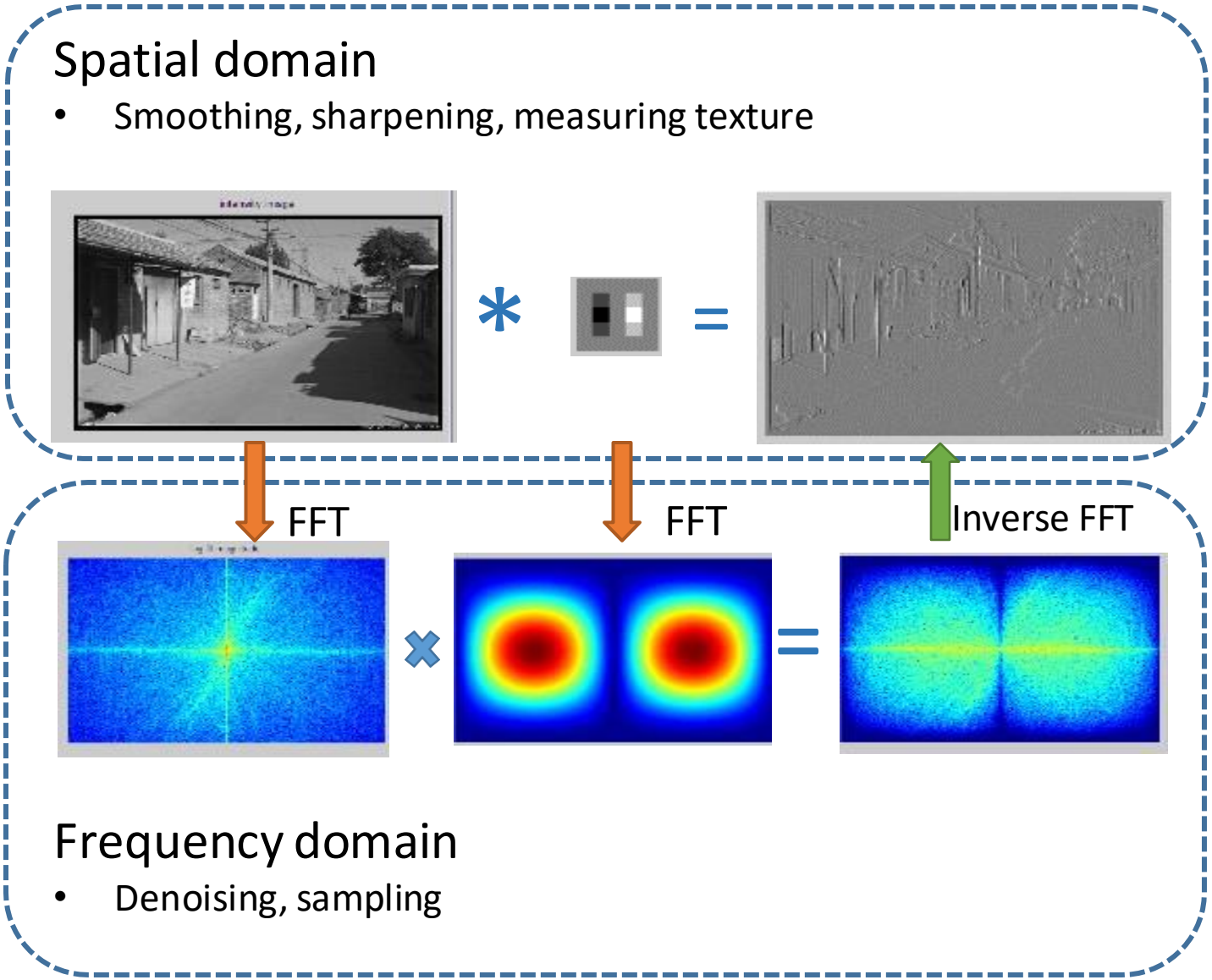
Edge Detection



Computer Vision

Adduru U G Sankararao, IIIT Sri City

Previous two classes: Image Filtering



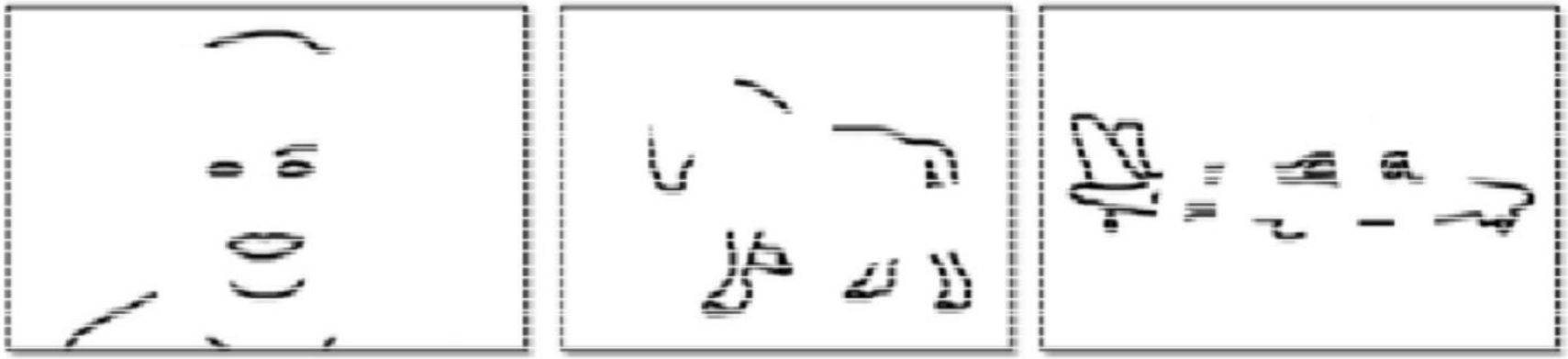
Today's class

- Detecting edges

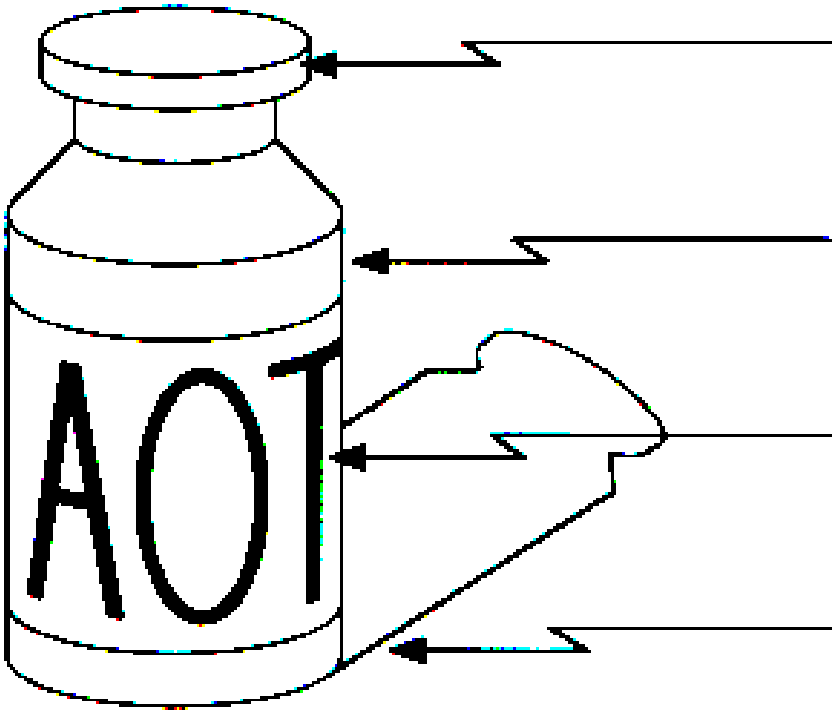
- Finding straight lines

Edge Detection

- Map image from 2D matrix of pixels to a set of curves or line segments or contours => More compact representation than pixels
- Key idea: Look for strong gradients, and then post-process



How edges are formed



surface normal discontinuity

depth discontinuity

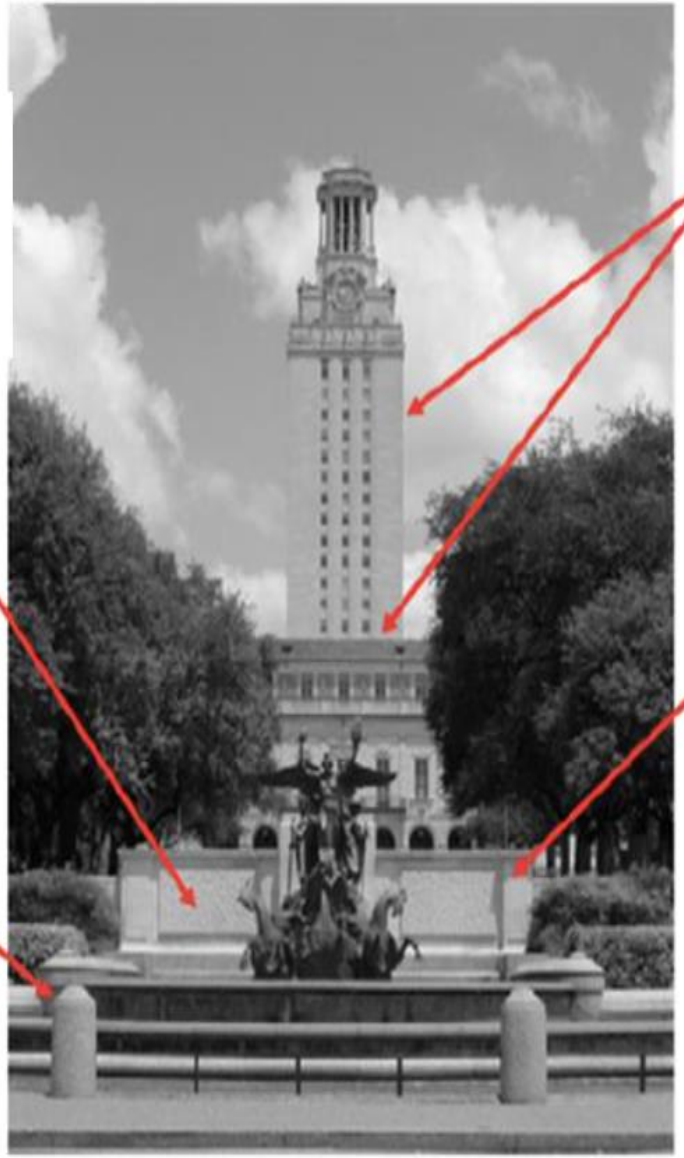
surface color discontinuity

illumination discontinuity

Edges are caused by a variety of factors

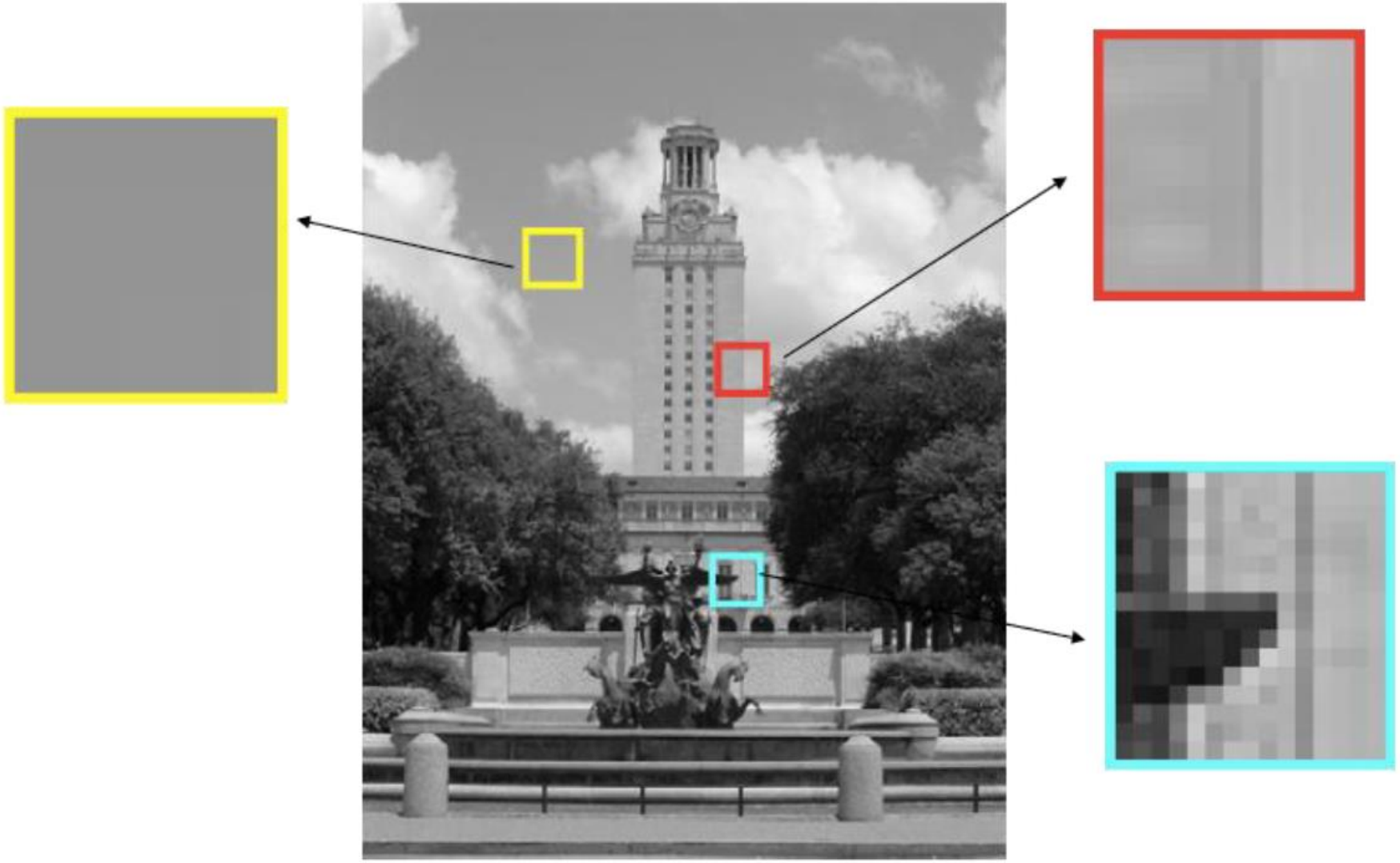
Source: Steve Seitz

How edges are formed



Source: K Grauman, R Urtasun

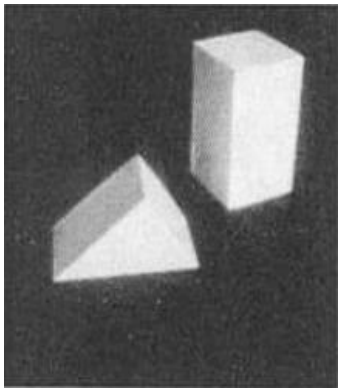
Looking more locally

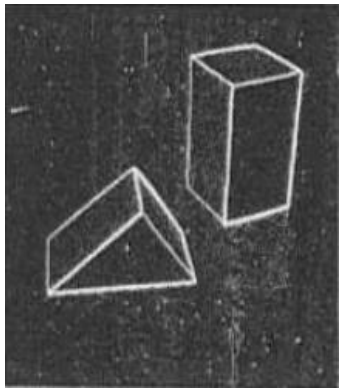


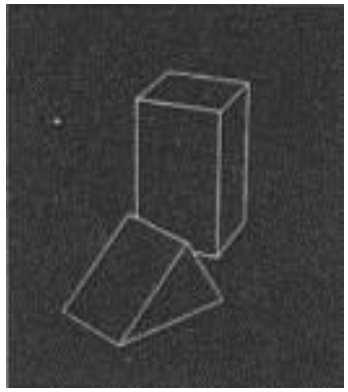
Source: K Grauman, R Urtasun

Why are Edges Important?

- Group pixels into objects or parts
- Allow us to track important features (e.g., corners, curves, lines).
- Cues for 3D shape
- Guiding interactive image editing



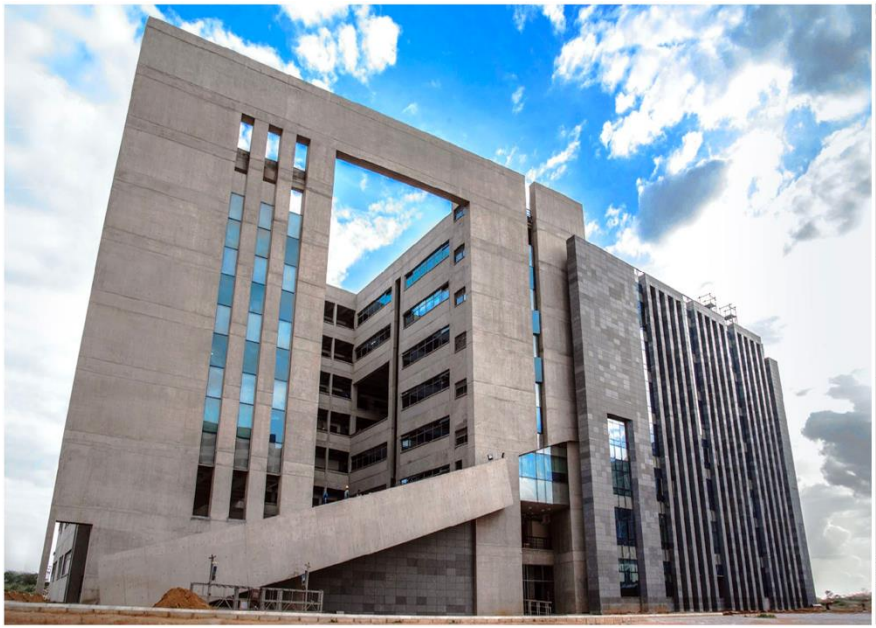


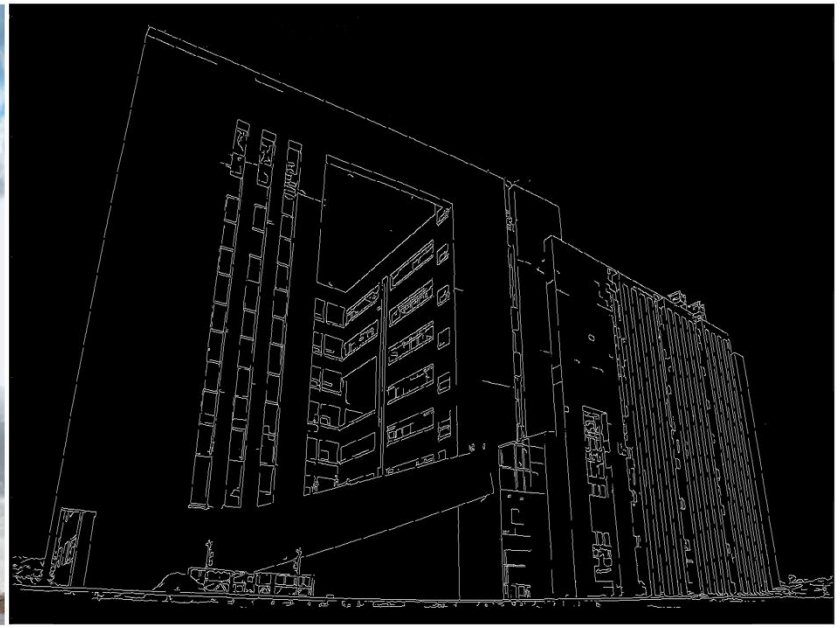


Source: K Grauman, R Urtasun

Why are Edges Important?

- Group pixels into objects or parts
- Allow us to track important features (e.g., corners, curves, lines).
- Cues for 3D shape
- Guiding interactive image editing

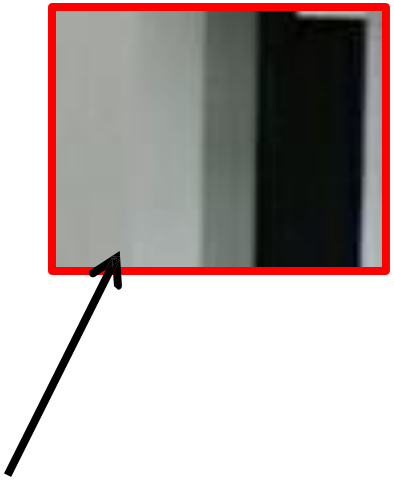




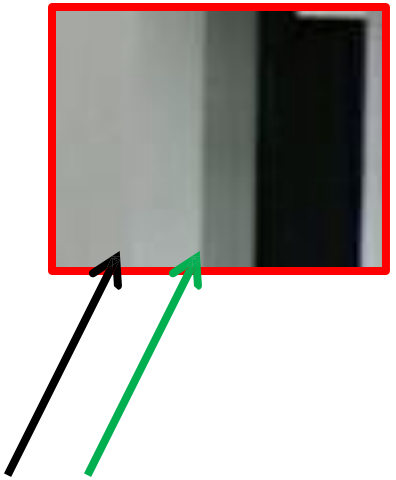
Source: K Grauman, R Urtasun

Closeup of edges

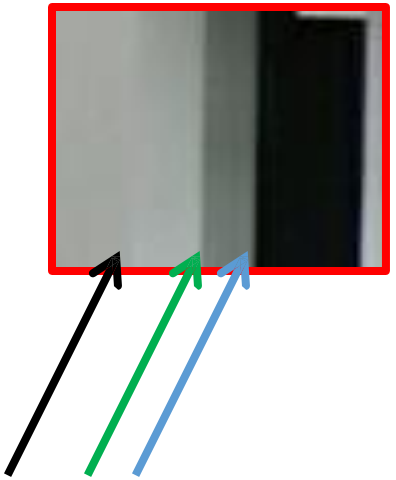
Closeup of edges



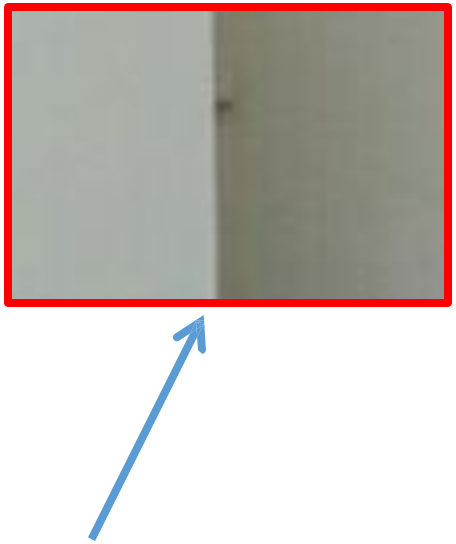
Closeup of edges



Closeup of edges



Closeup of edges

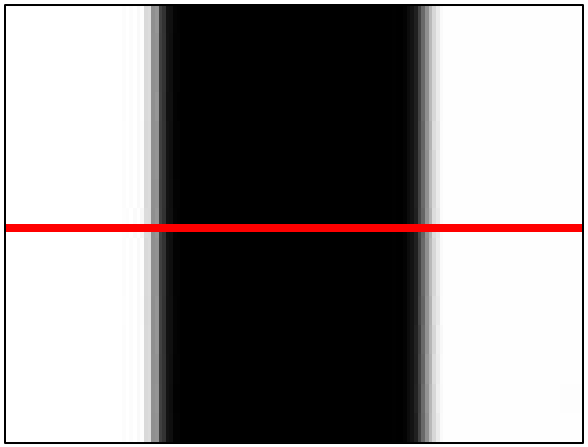


Closeup of edges

Characterizing edges

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

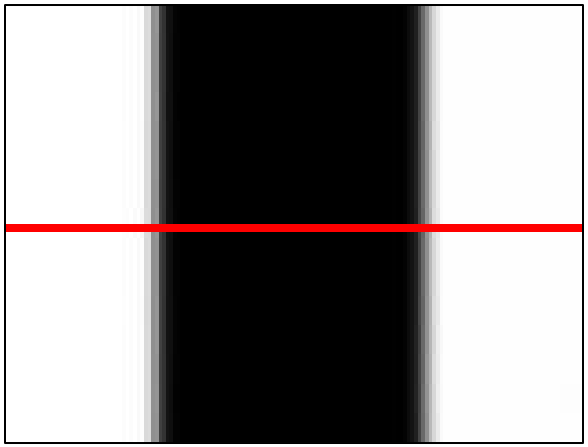
image



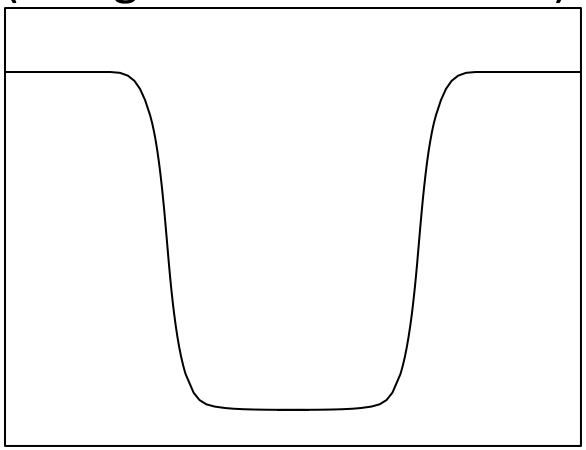
Derivatives & edges

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

image



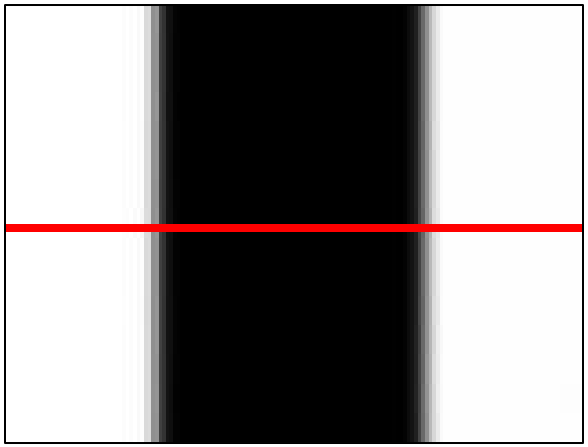
intensity function
(along horizontal scanline)



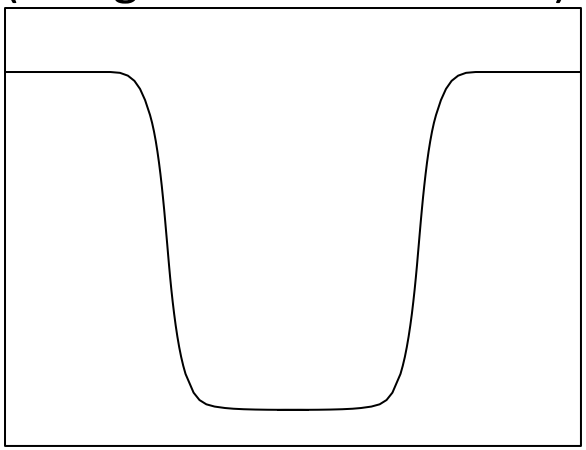
Derivatives & edges

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

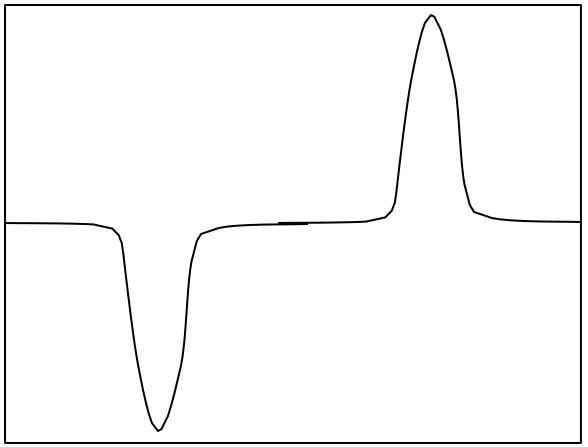
image



intensity function
(along horizontal scanline)

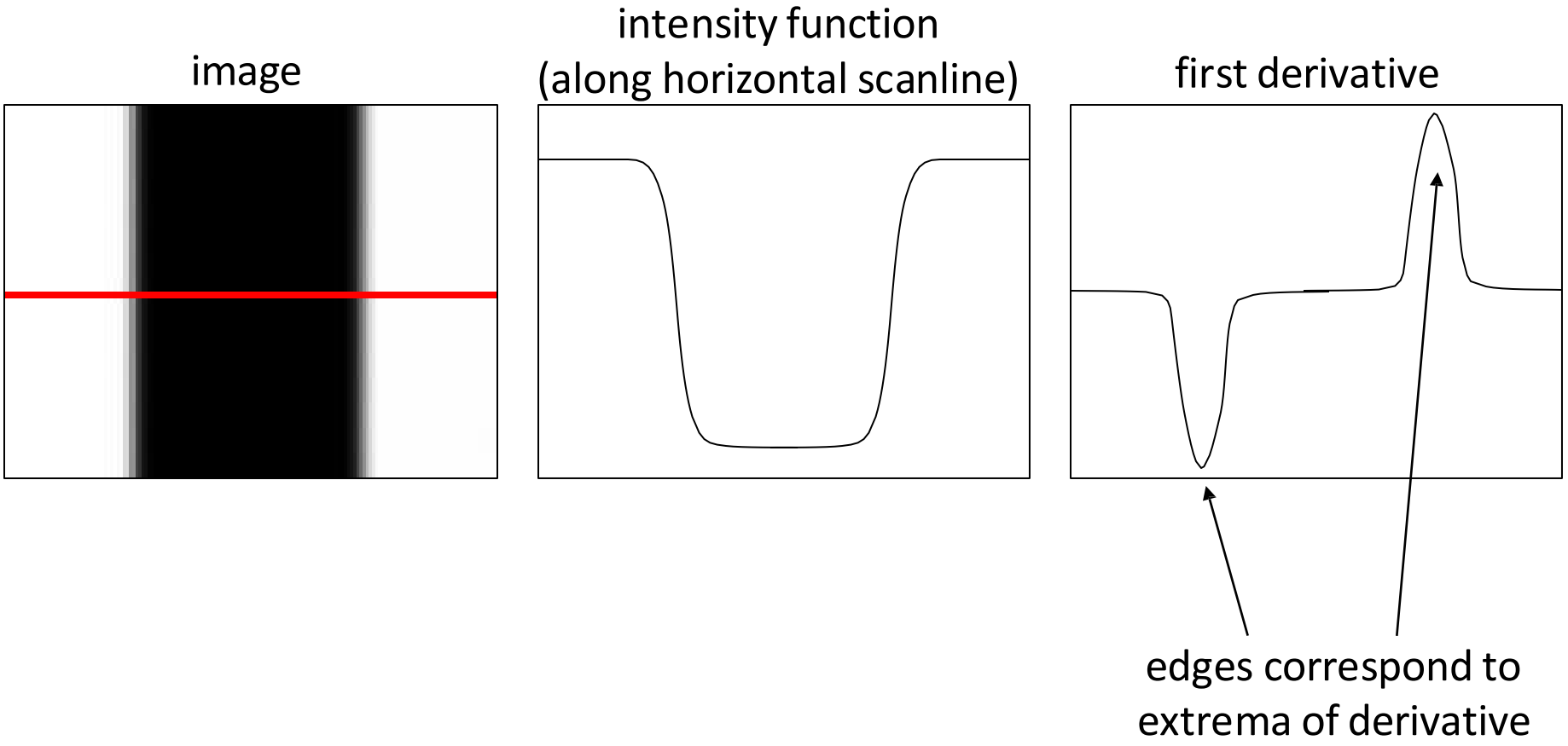


first derivative



Derivatives & edges

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



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?

Derivatives with Convolution

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

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

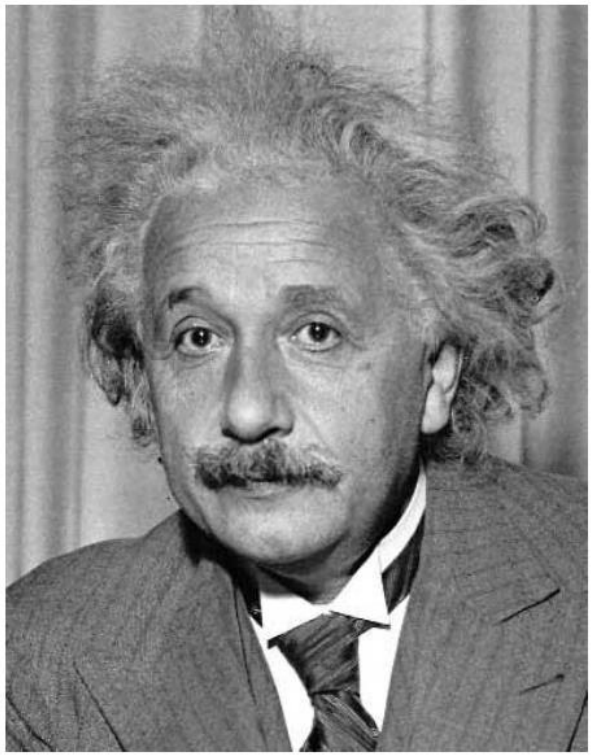
-1	1

-1
1

or

1
-1

Sobel Edge Detection Filter



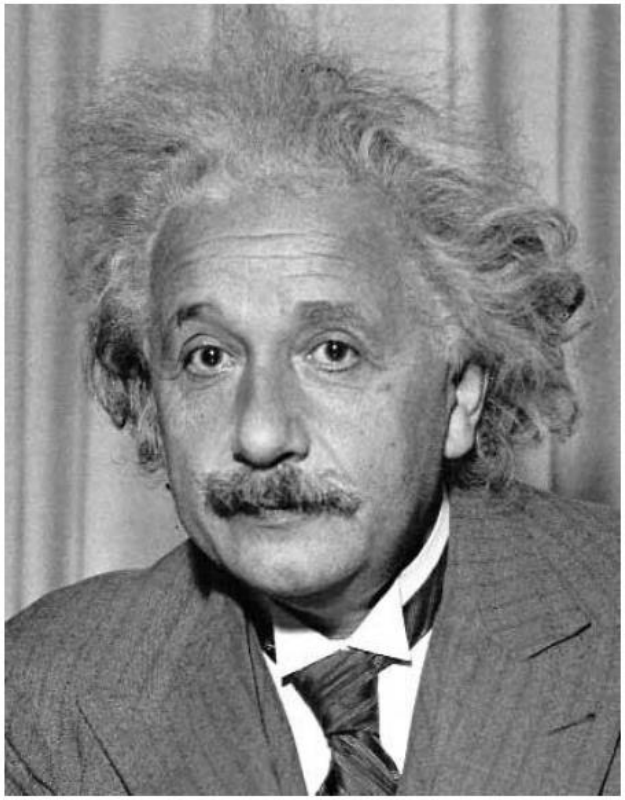
1	0	-1
2	0	-2
1	0	-1

Sobel

Vertical Edge
(absolute value)

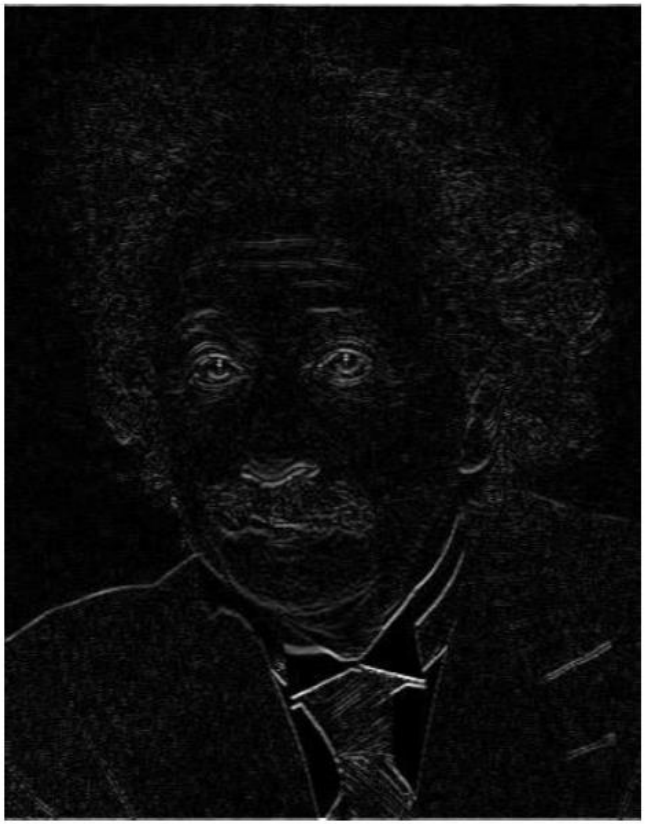
Source: J Hays

Sobel Edge Detection Filter



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

Sobel



Horizontal Edge
(absolute value)

Source: J Hays

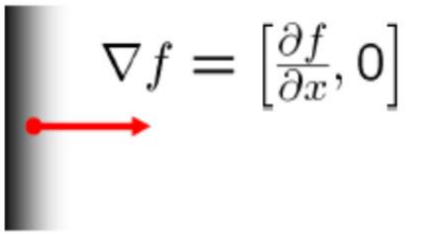
Finite Difference Filters

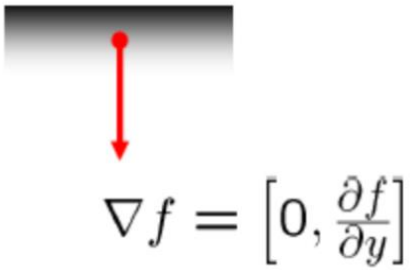
	M_x			M_y		
Prewitt	-1	0	1	1	1	1
	-1	0	1	0	0	0
	-1	0	1	-1	-1	-1
Sobel	-1	0	1	1	2	1
	-2	0	2	0	0	0
	-1	0	1	-1	-2	-1
Roberts	0	1		1	0	
	-1	0		0	-1	

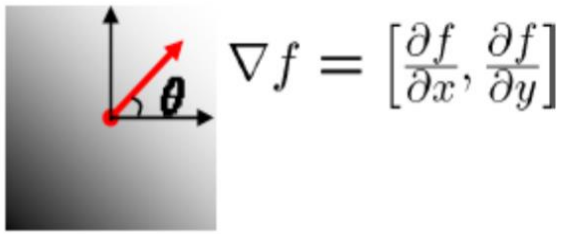
Source: R Urtasun

Image Gradients

- 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







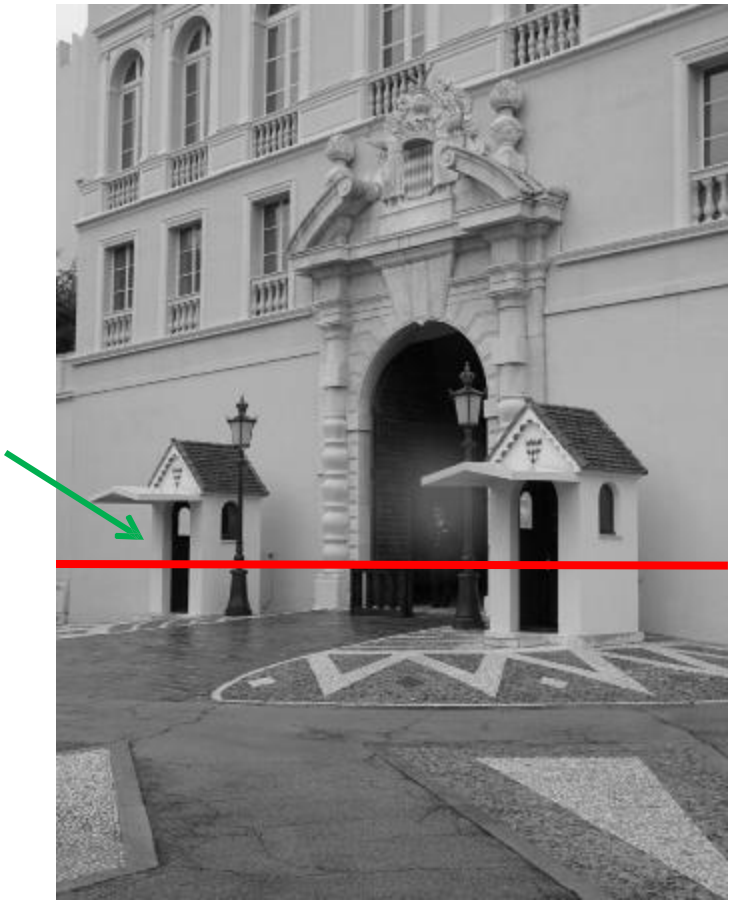
- The **gradient direction** (orientation of edge normal) is given by:

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

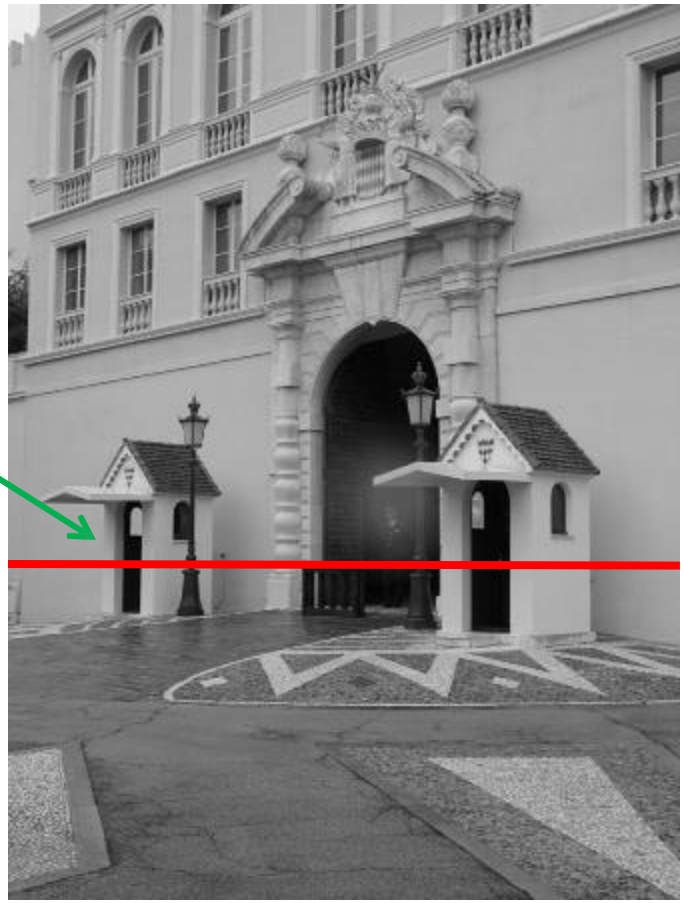
- The **edge strength** is given by the magnitude $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

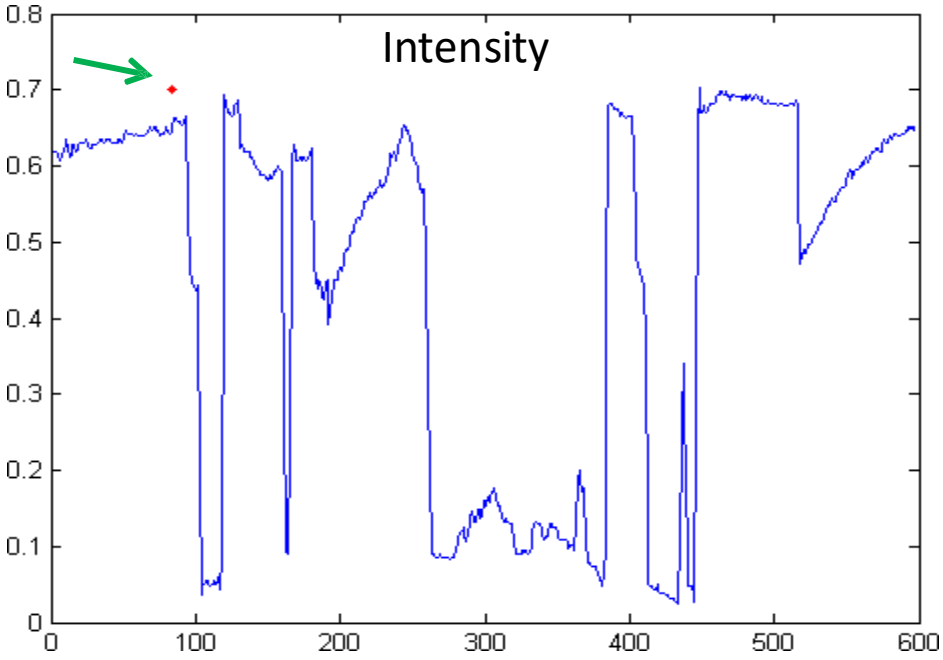
Source: R Urtasun

Intensity profile

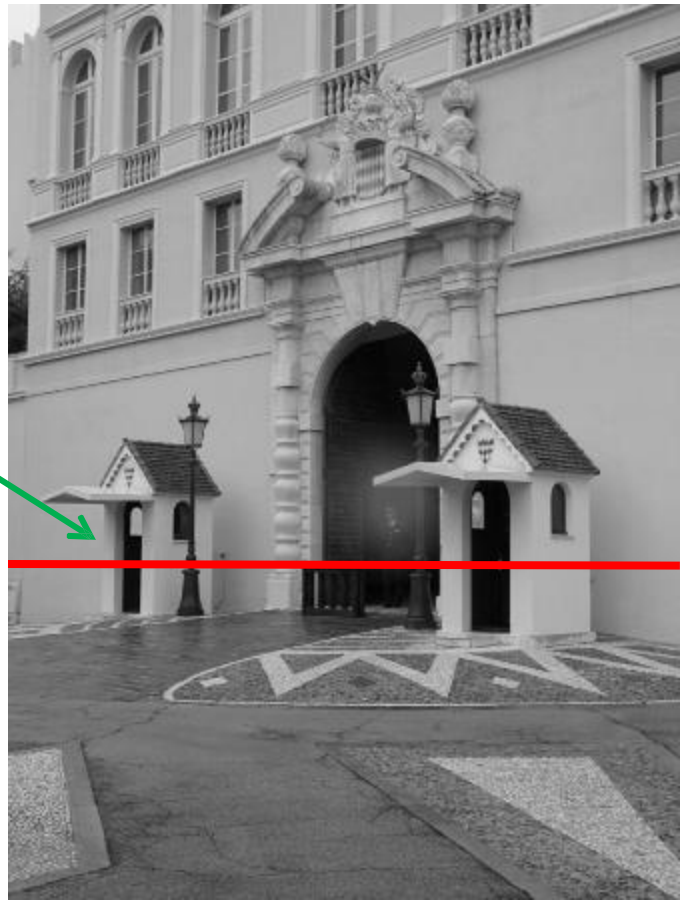


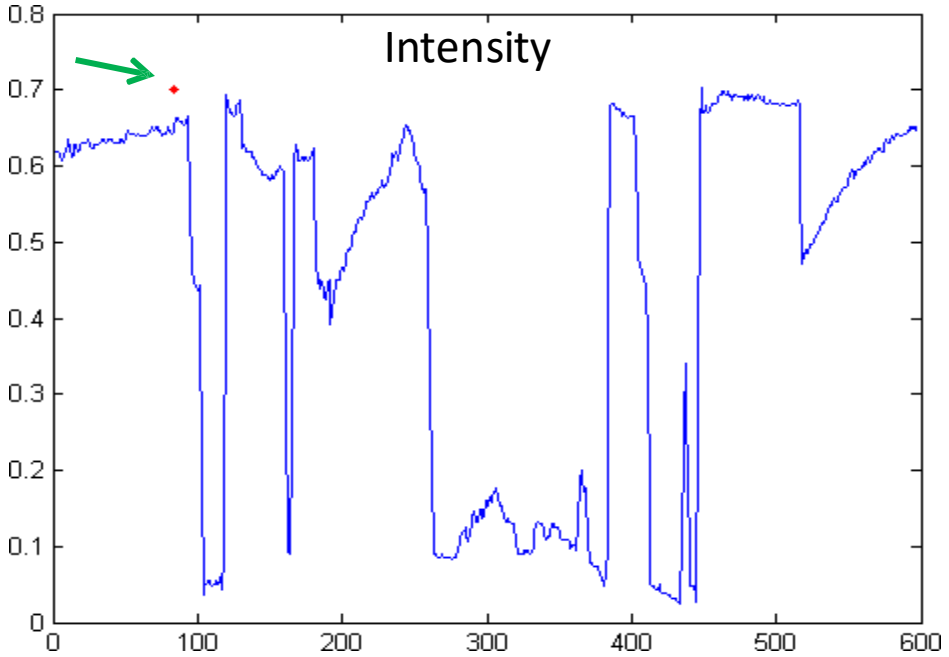
Intensity profile

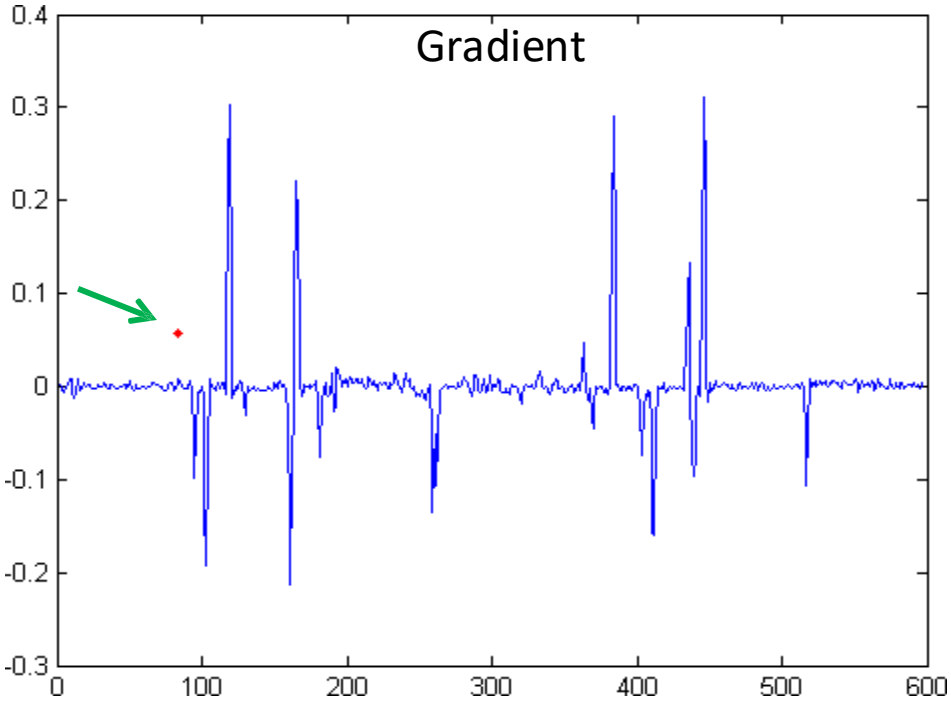




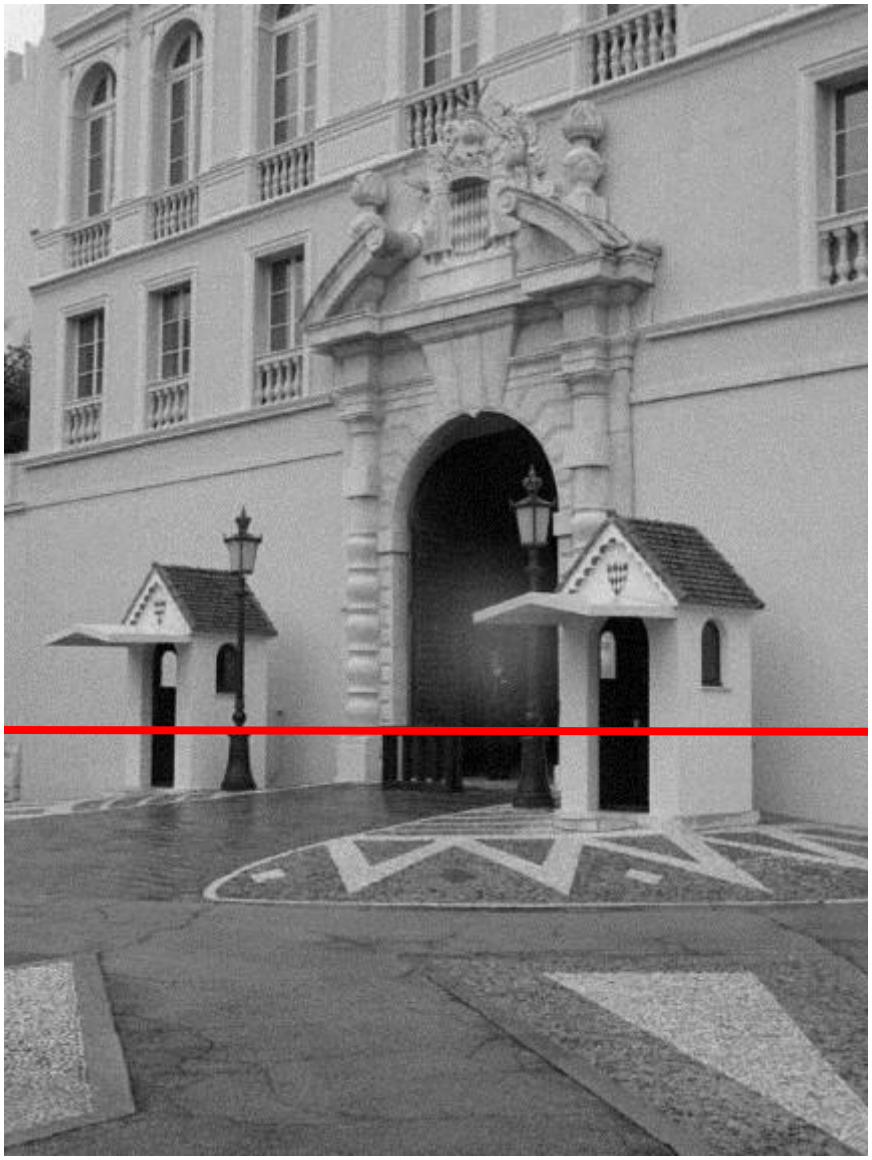
Intensity profile

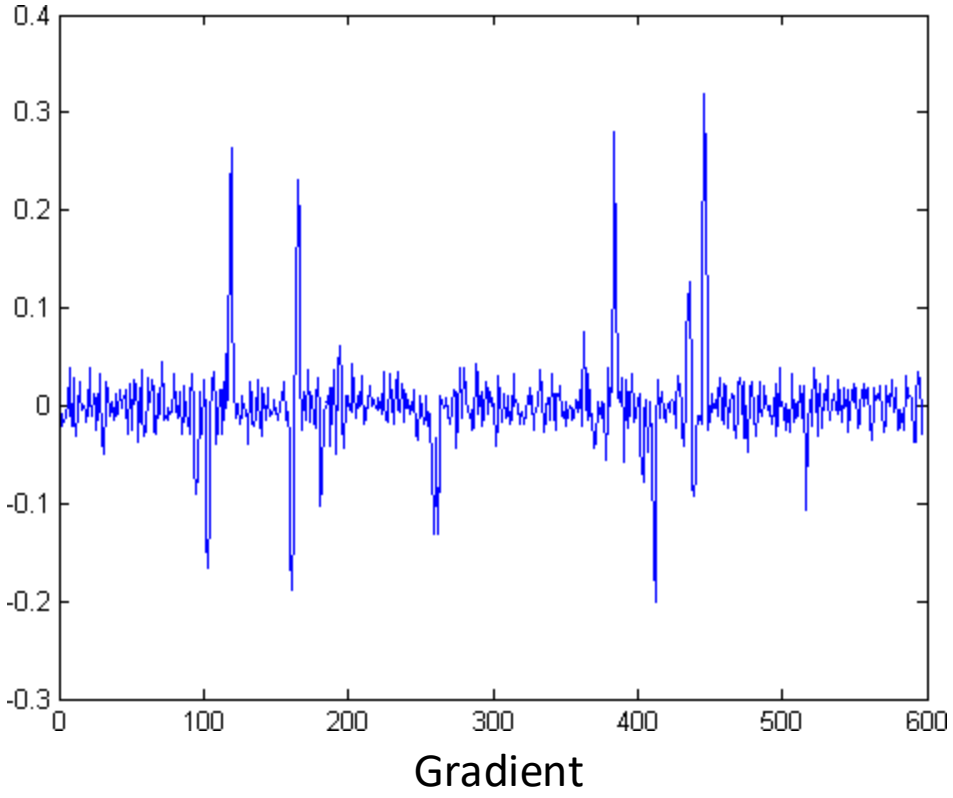






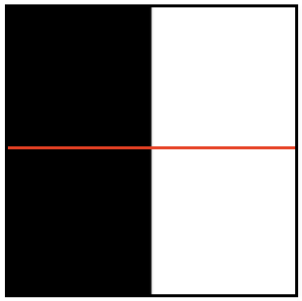
With a little Gaussian noise



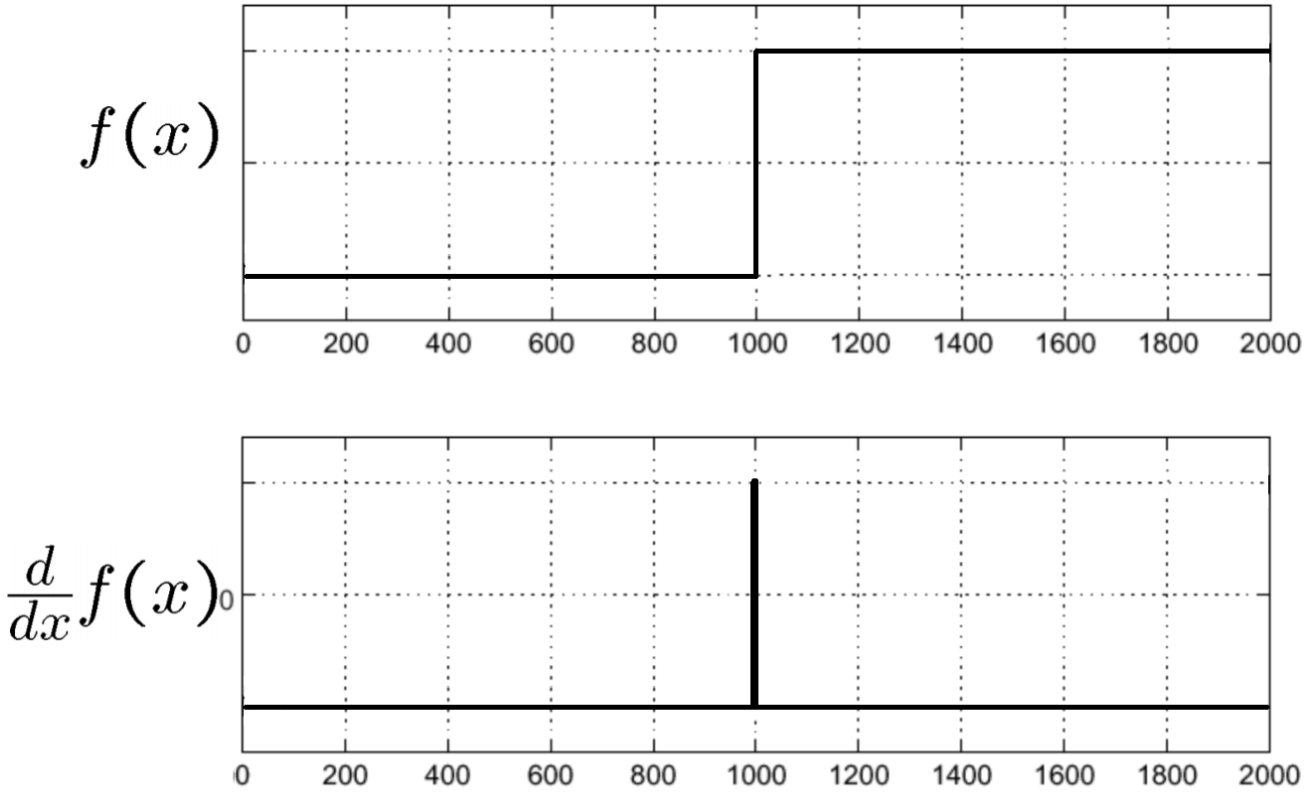


Derivative with no noise

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



Input image with no noise

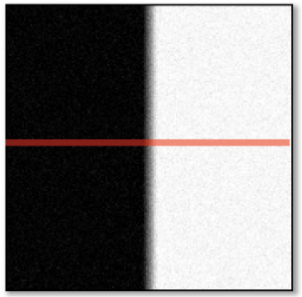


Where is the edge?

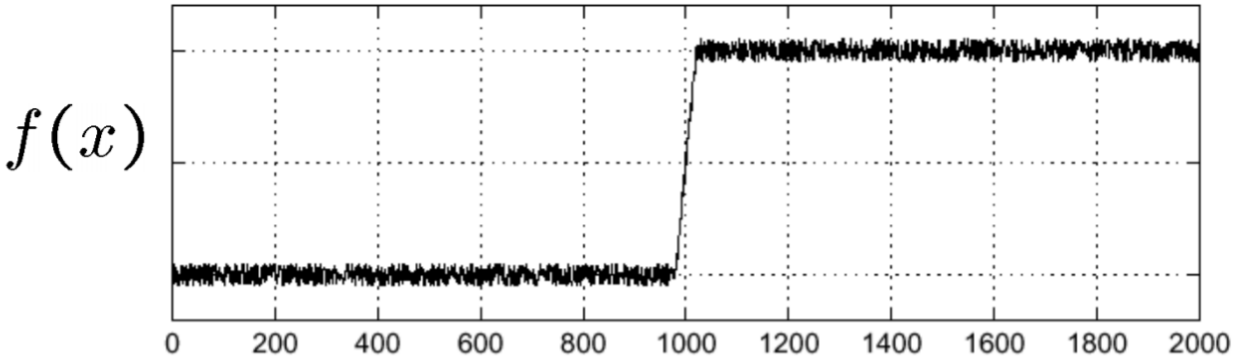
Source: S. Seitz

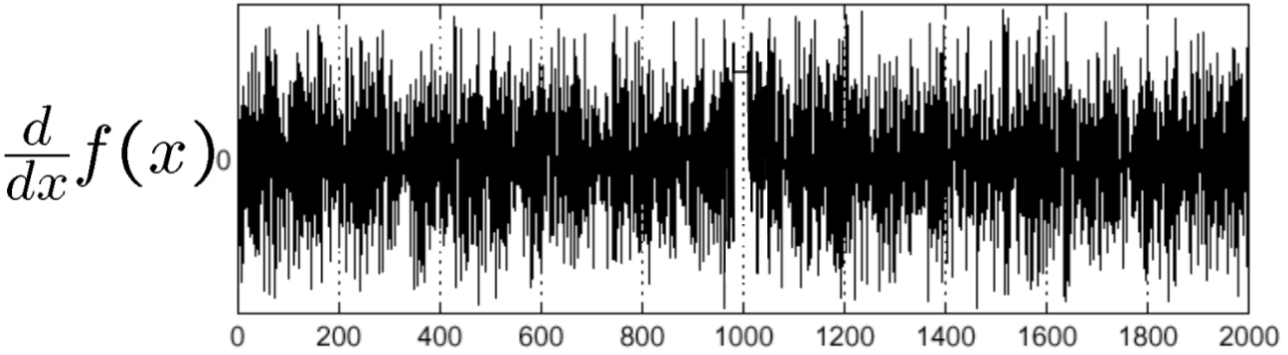
Effects of noise

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



Noisy input image





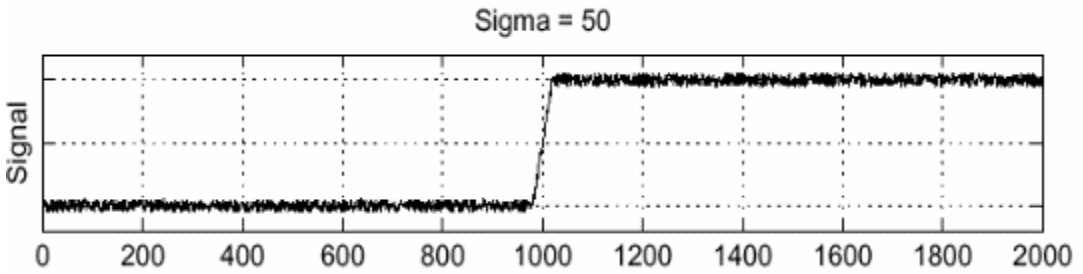
Where is the edge?

The larger the noise the stronger the response

- **What can we do about it?**

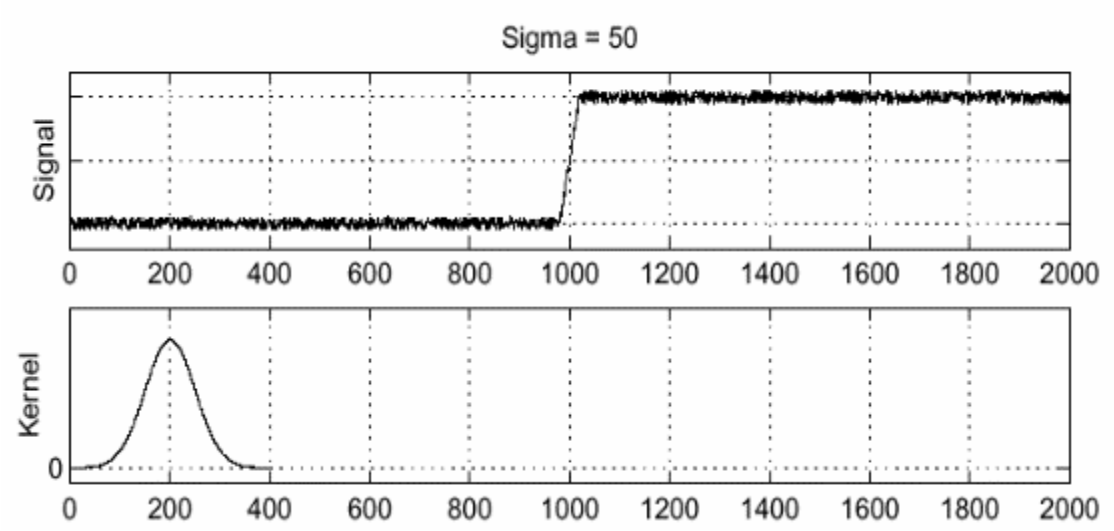
Solution: smooth first

f



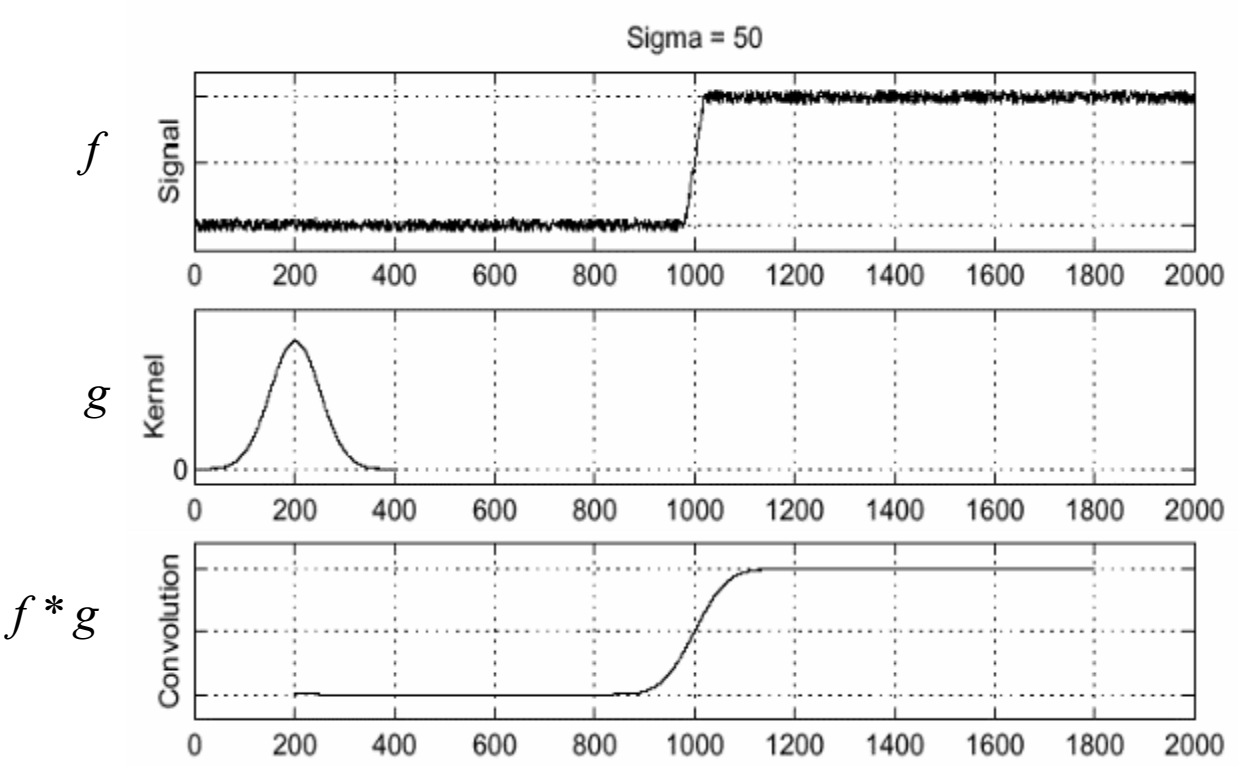
Source: S. Seitz

Solution: smooth first



Source: S. Seitz

Solution: smooth first



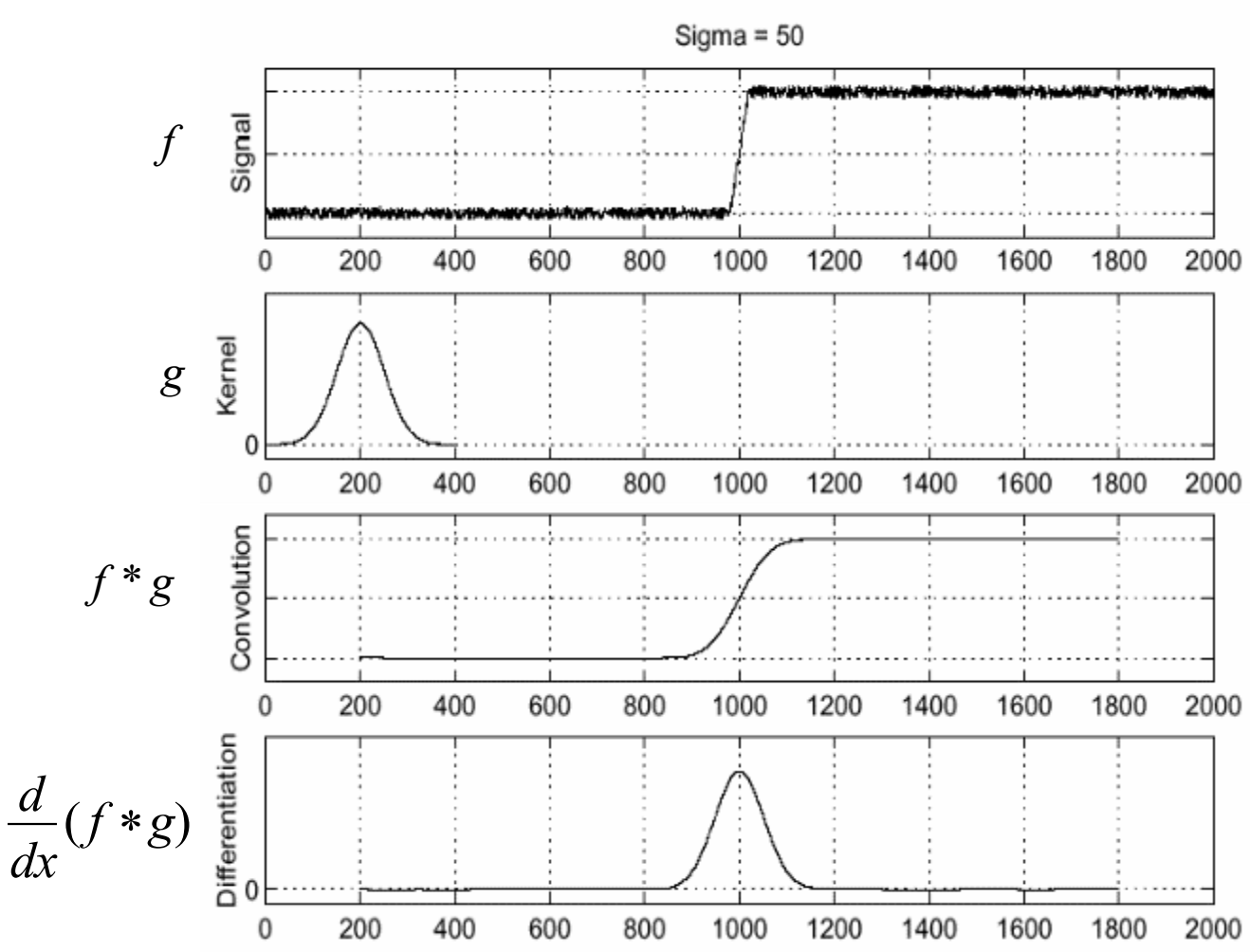
f

g

$f * g$

Source: S. Seitz

Solution: smooth first



f

g

$f * g$

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

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

Source: S. Seitz

Derivative theorem of convolution

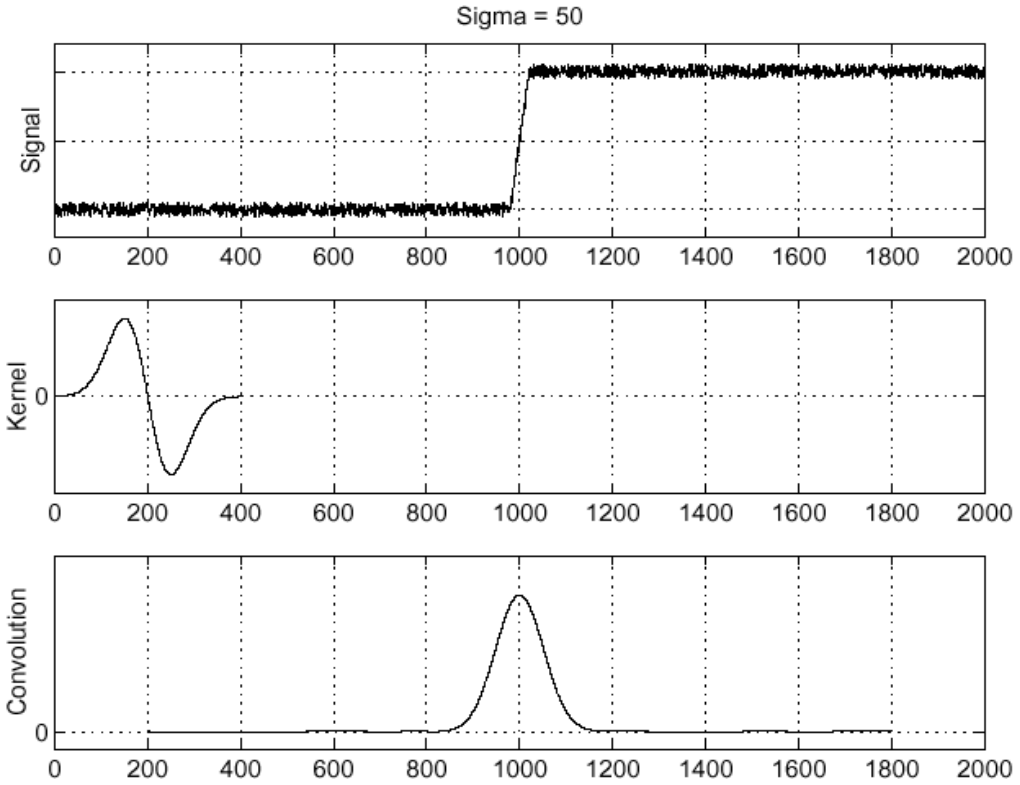
- Differentiation in convolution, and convolution is associative:
- This saves us one operation:

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

f

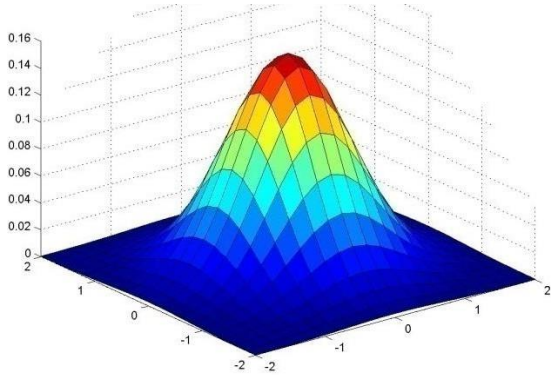
$\frac{d}{dx} g$

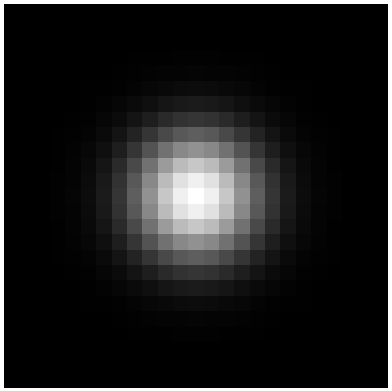
$f * \frac{d}{dx} g$



Source: S. Seitz

Gaussian Kernel

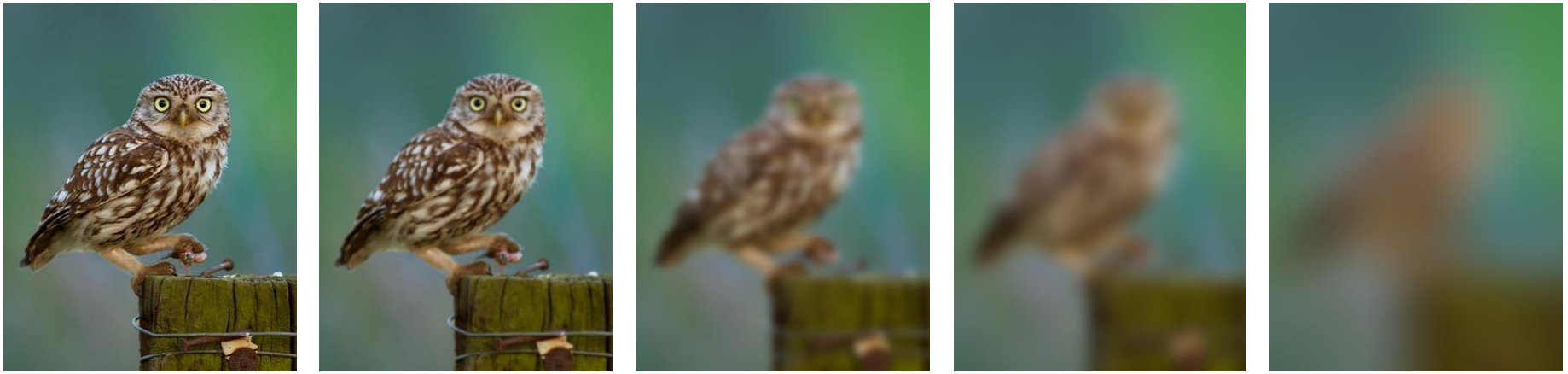




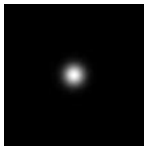
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

Gaussian filters

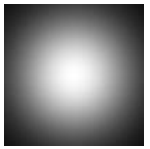


σ = 1 pixel



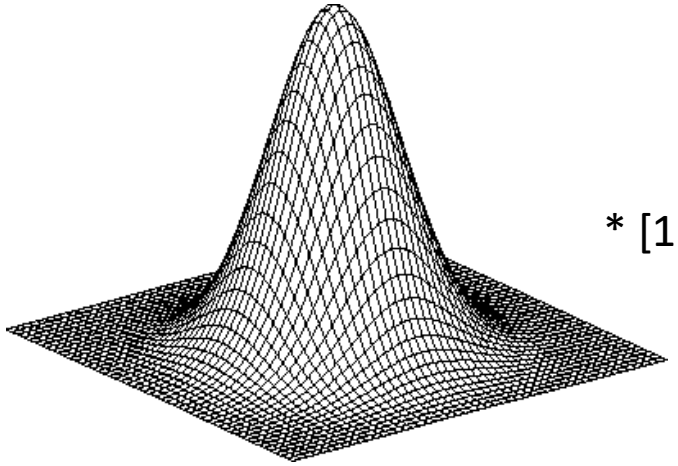
σ = 5 pixels

σ = 10 pixels



σ = 30 pixels

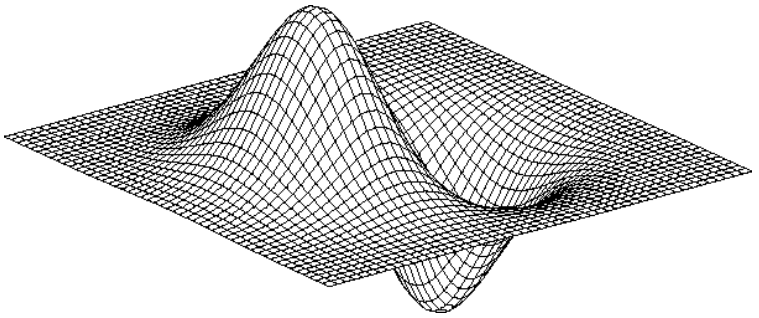
Derivative of Gaussian filter



Gaussian

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

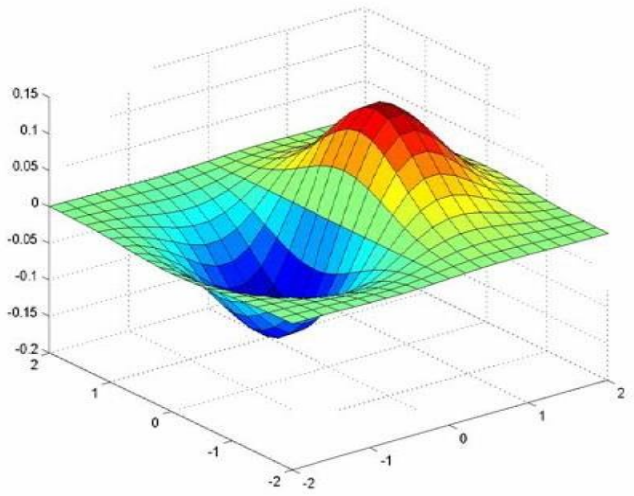
* [1 0 -1] =



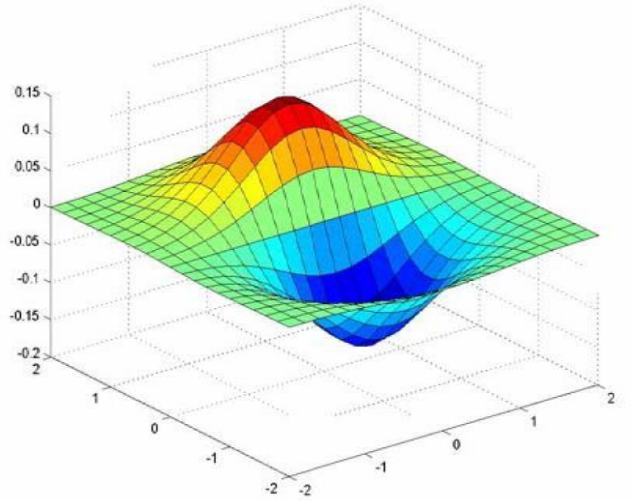
derivative of Gaussian (x)

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

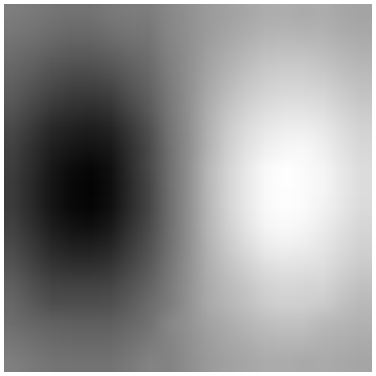
Derivative of Gaussian filter

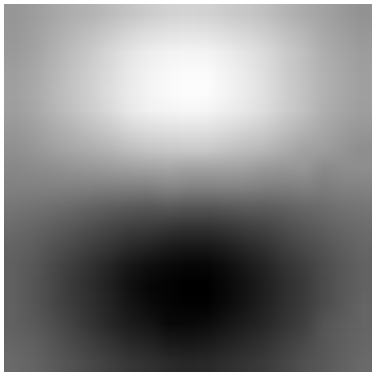


x-direction



y-direction



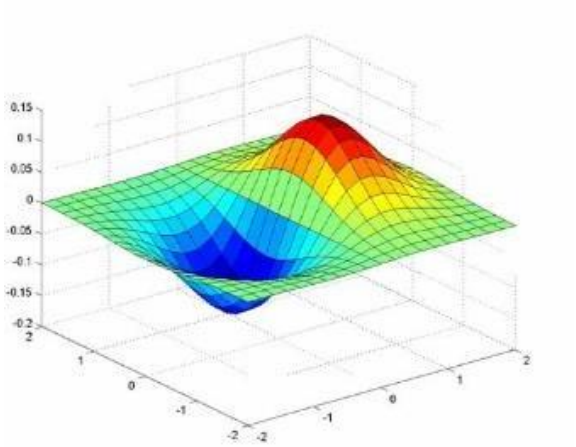


- Which one finds horizontal and vertical edges ?

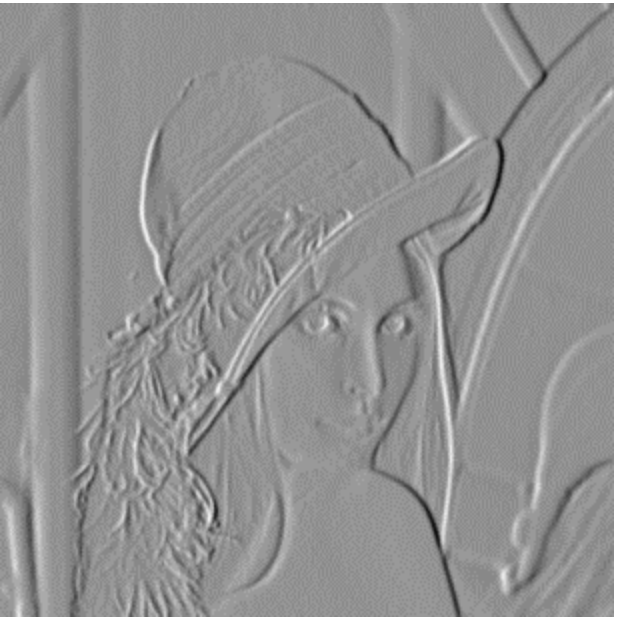
Example

input image (“Lena”)

Compute Gradients (DoG)

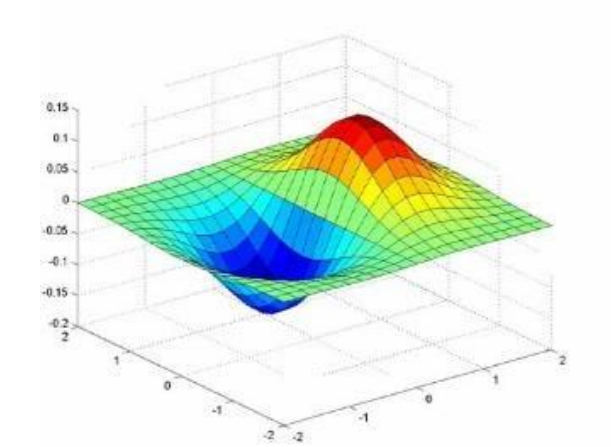


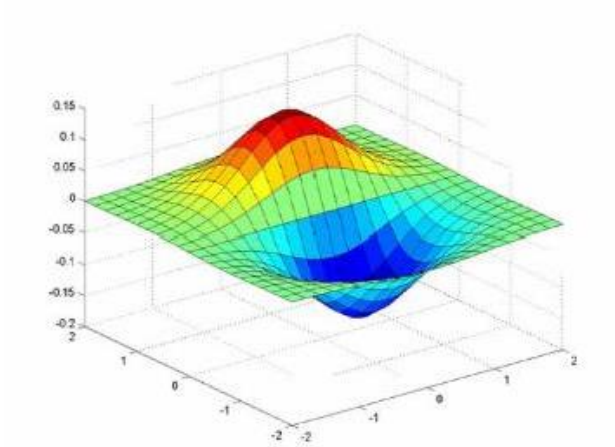
Input Image



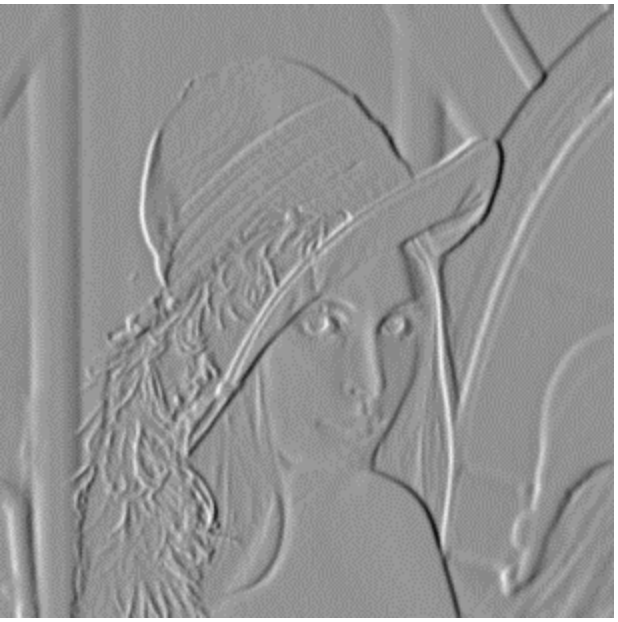
X-Derivative of Gaussian

Compute Gradients (DoG)





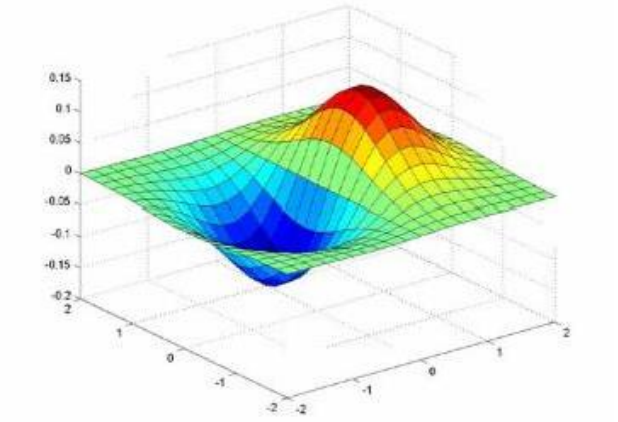
Input Image

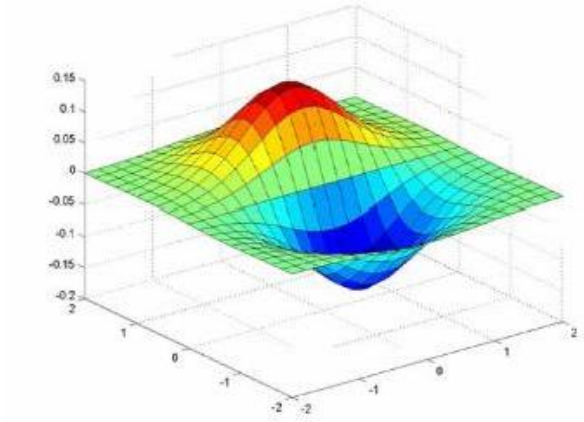


X-Derivative of Gaussian

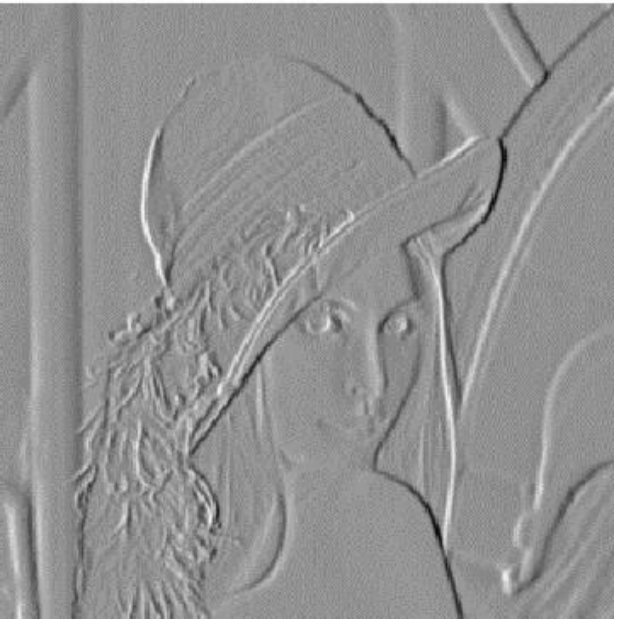
Y-Derivative of Gaussian

Compute Gradients (DoG)





Input Image



X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

Designing an edge detector

where is the edge?

Designing an edge detector

- Criteria for a good edge detector:

  - **Good detection:**
    - find all real edges, ignoring noise or other artifacts

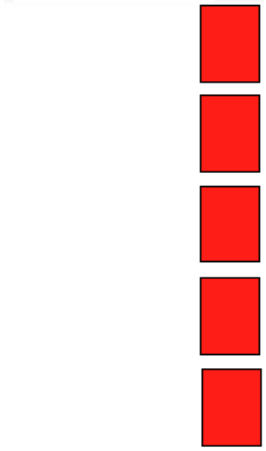
Source: L. Fei-Fei

Designing an edge detector

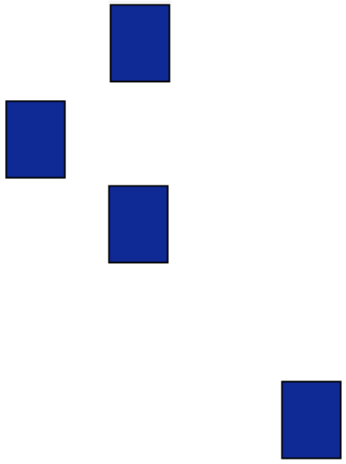
- Criteria for a good edge detector:
  - **Good detection:**
    - find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - detect edges as close as possible to the true edges
    - return one point only for each true edge point

Source: L. Fei-Fei

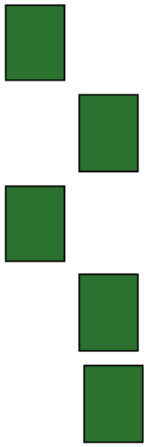
Designing an edge detector



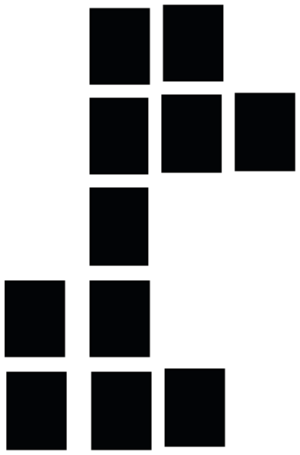
True Edge



Poor robustness to noise

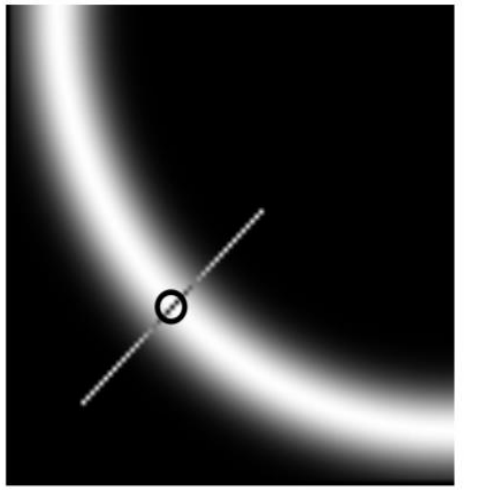


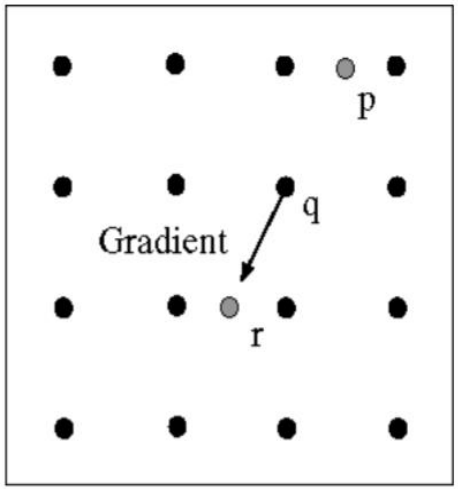
Poor localization

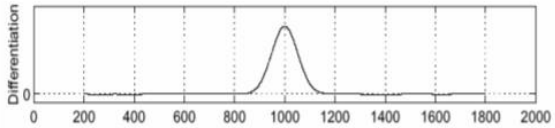


Too many responses

Non-Maxima Suppression



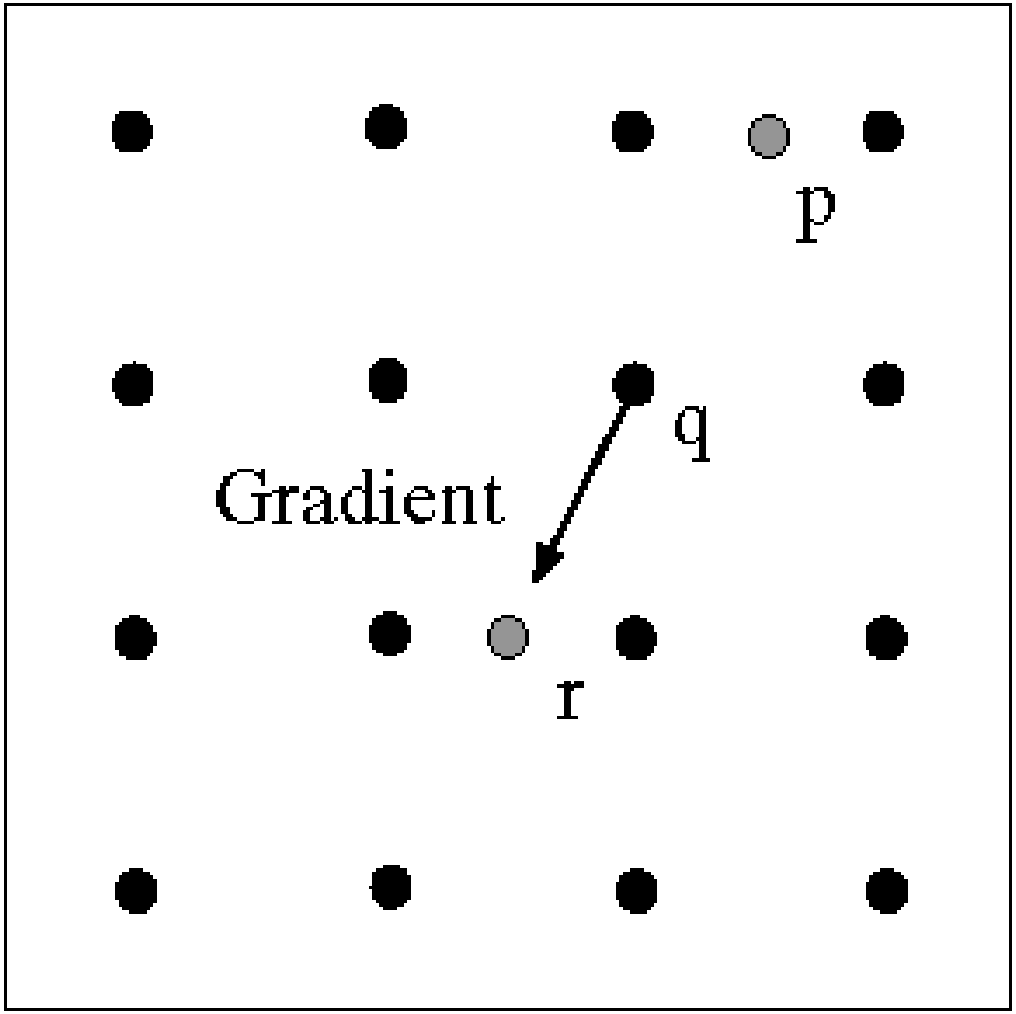




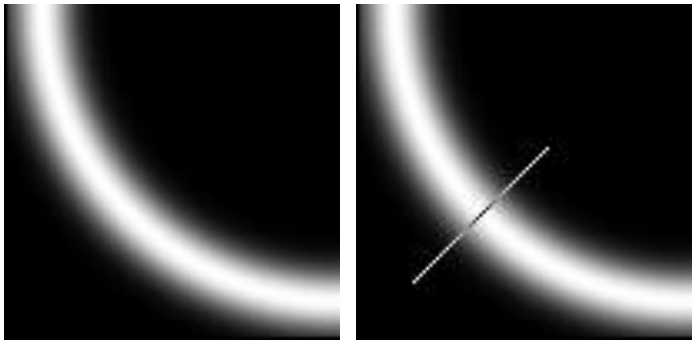
- Get orientation theta = arctan(gy, gx)
- Check if pixel is local maximum along gradient direction:
- Could require checking interpolated pixels p and r

Source: N Snavely, R Urtasun

Non-maximum suppression for each orientation



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Source: D. Forsyth

Before Non-max Suppression

After non-max suppression

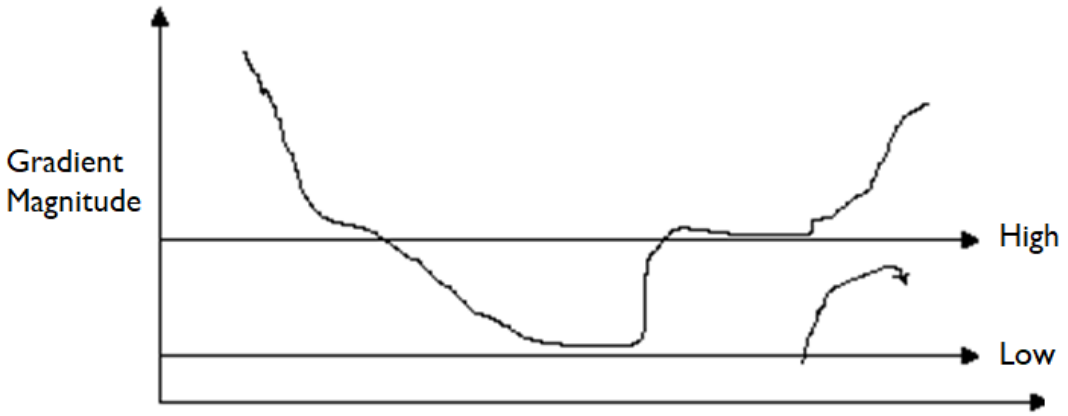
Hysteresis thresholding

- Check for well-connected edges
- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Hysteresis thresholding

- Check for well-connected edges. How?
- use a high threshold to start edge curves and a low threshold to continue them.
- How does it work?

  - If gradient at pixel > `High' => `edge pixel'
  - If gradient at pixel < `Low' => `non-edge pixel'
  - If gradient at pixel > `Low' and < `High' => `edge pixel' iff it is connected to an `edge pixel' directly or via pixels between `Low' and `High'



Source: S. Seitz

Final Canny Edges

Canny edge detector

- The most widely used edge detector

A computational approach to edge detection
J Canny - IEEE Transactions on pattern analysis and machine …, 1986 - ieeexplore.ieee.org
Abstract: This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for ...
Cited by 27743 Related articles All 27 versions Import into BibTeX Save More

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

Canny edge detector

- Probably the most widely used edge detector in computer vision (Canny 1986)

Source: D. Lowe, L. Fei-Fei

Canny edge detector

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

Source: D. Lowe, L. Fei-Fei

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” down to single pixel width

Source: D. Lowe, L. Fei-Fei

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” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei

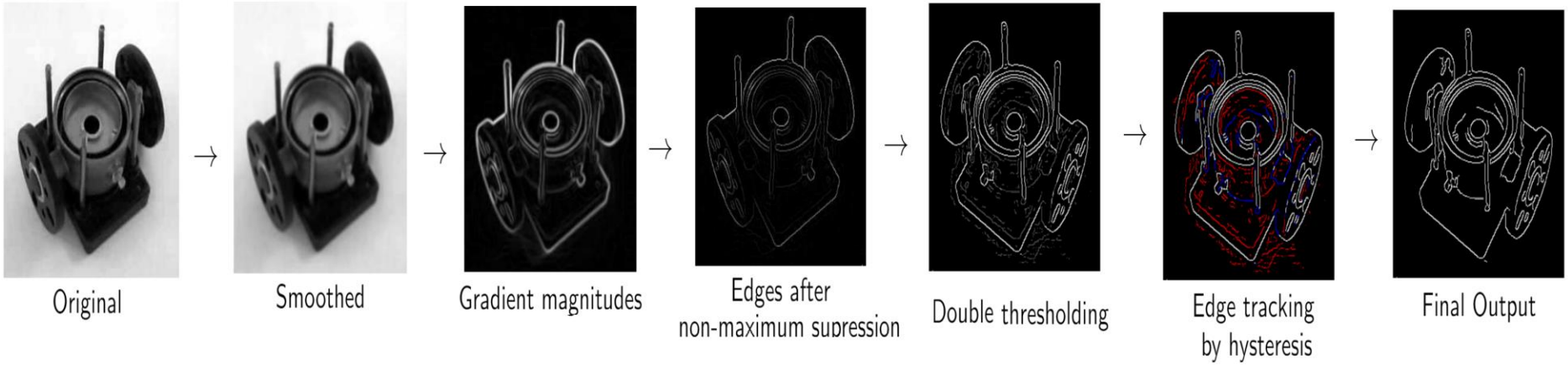
Canny edge detector

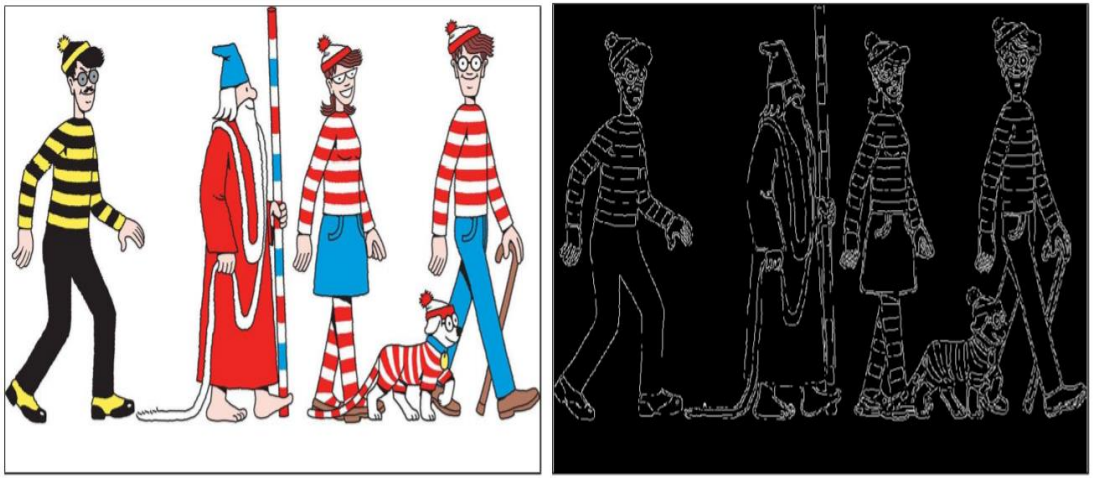
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: `edge(image, 'canny')`

Source: D. Lowe, L. Fei-Fei

Canny edge Pipeline: Example





Source: D. Lowe, L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

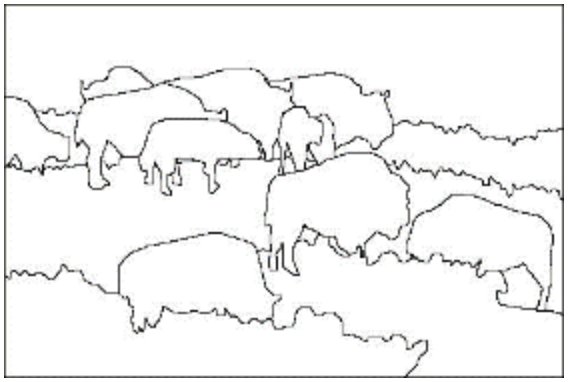
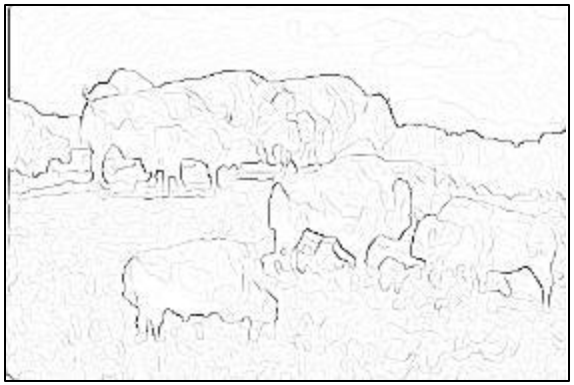
The choice of σ depends on desired behavior

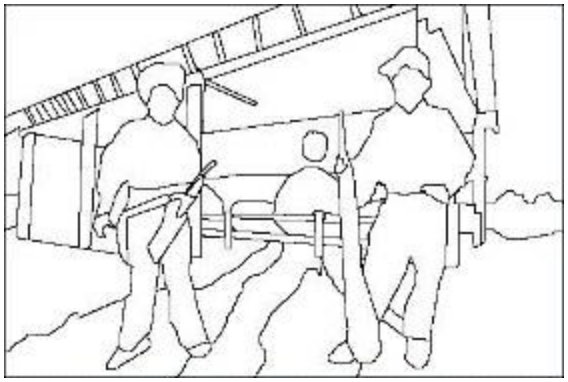
- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

Learning to detect boundaries

image | human segmentation | gradient magnitude

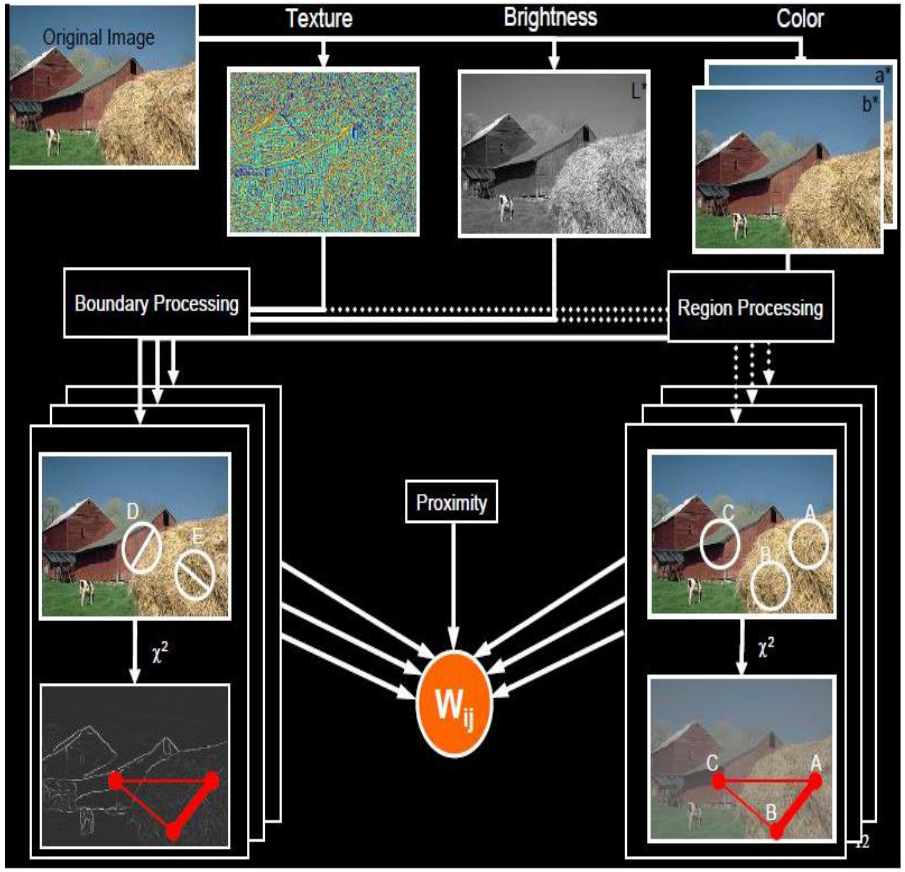




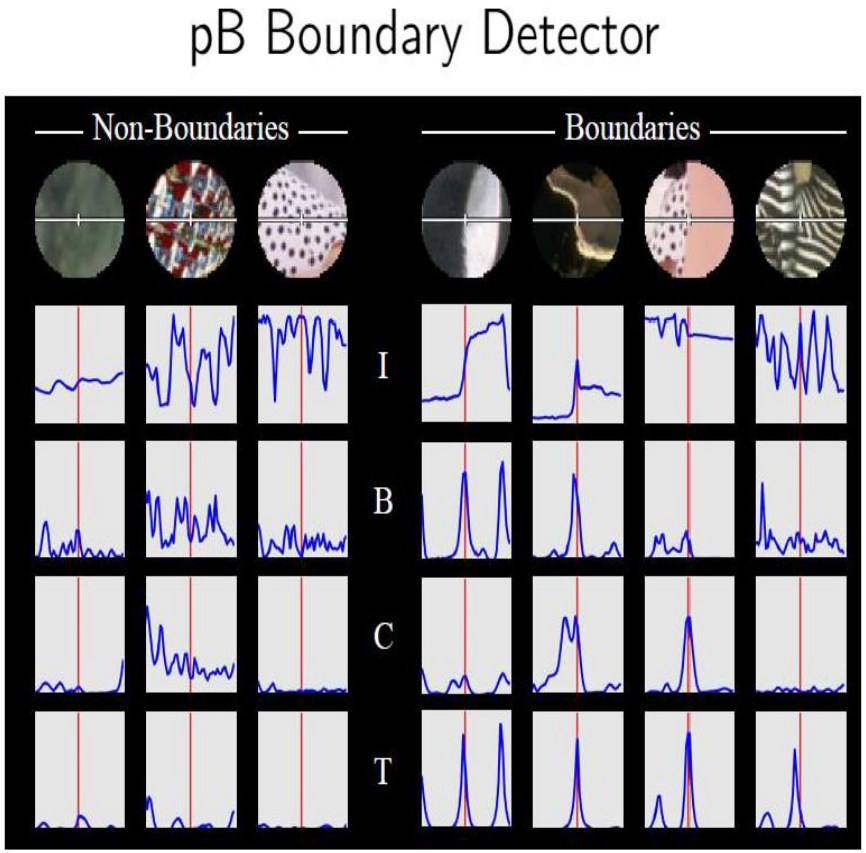



- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

More Recent Methods in Edge Detection

- Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues (Martin et al, 2004)



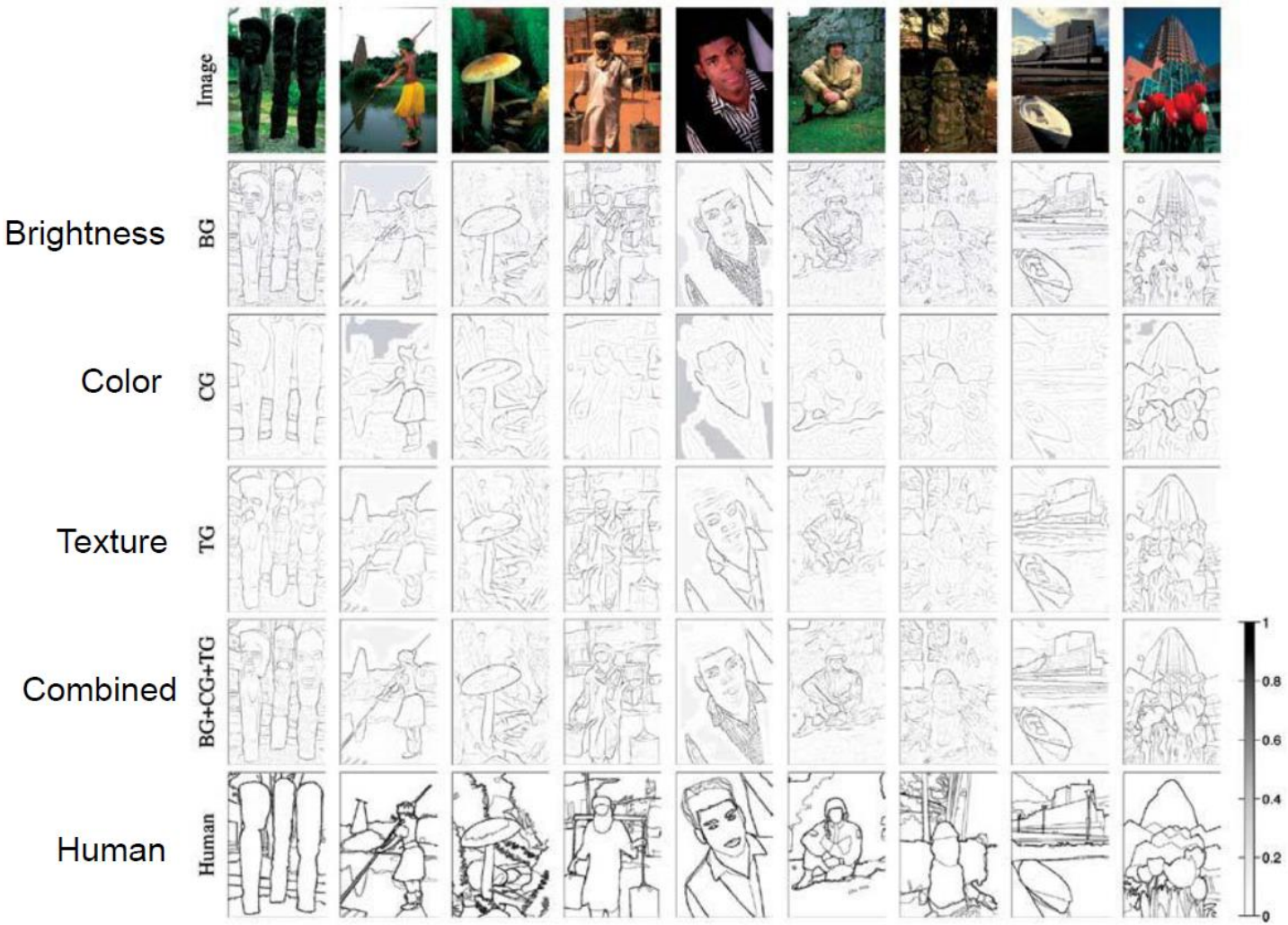


I-Intensity, B-Brightness gradient, C-Color gradient, T-Texture gradient

Source: Derek Hoiem

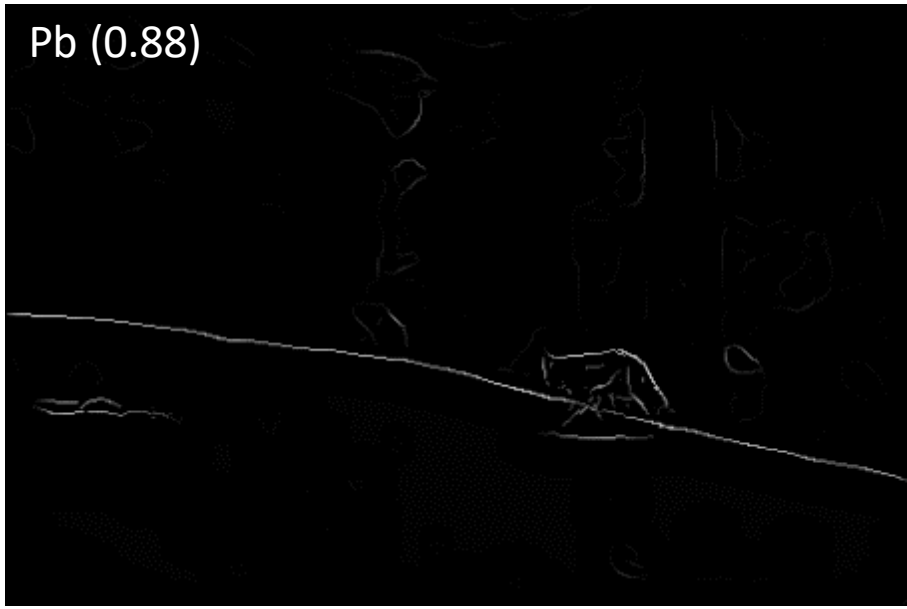
More Recent Methods in Edge Detection

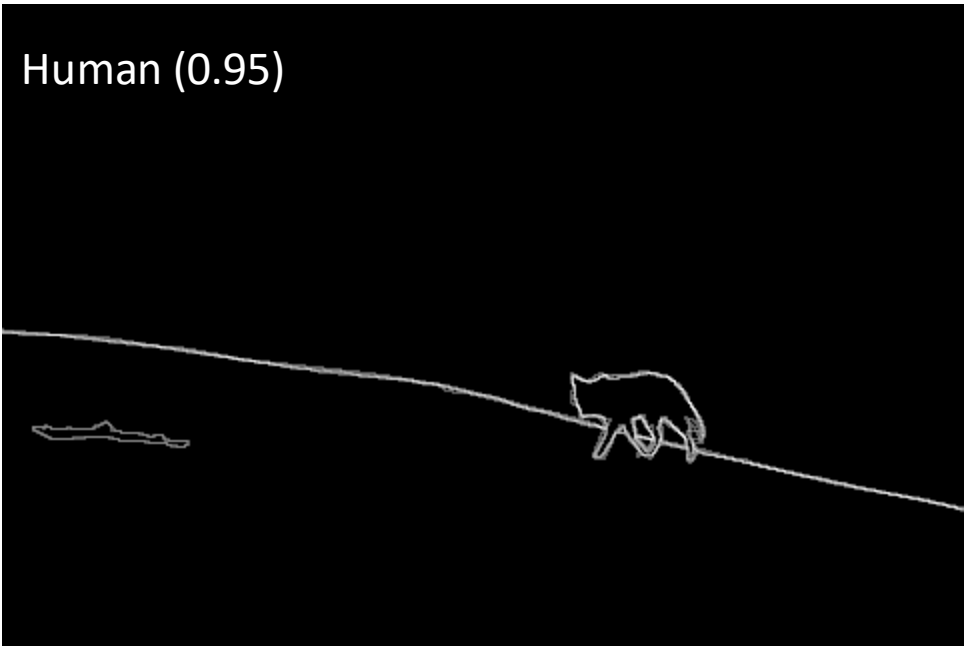
- Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues (Martin et al, 2004)



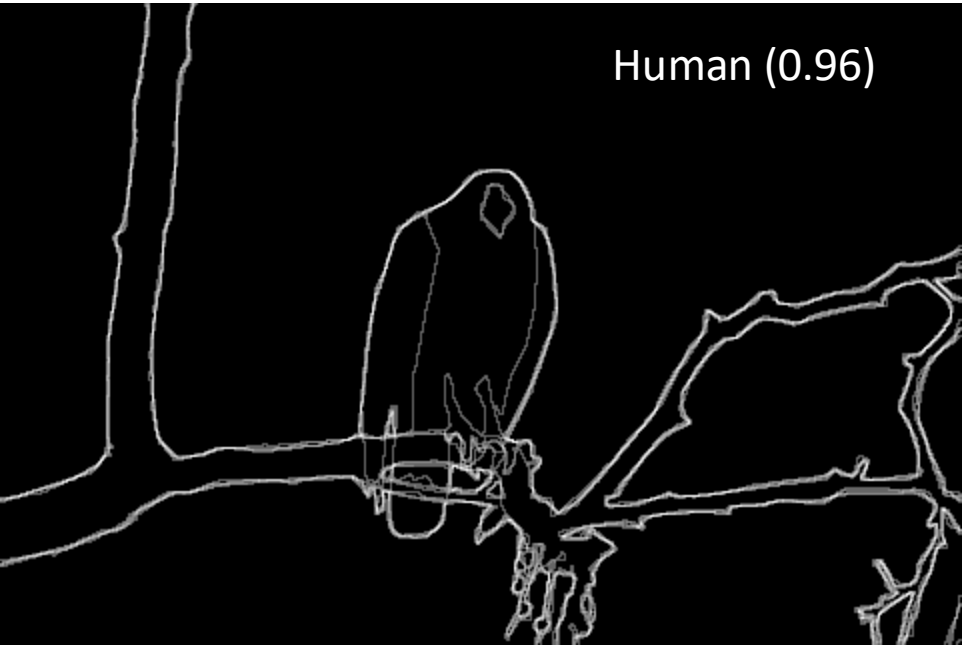
Source: Derek Hoiem

Results

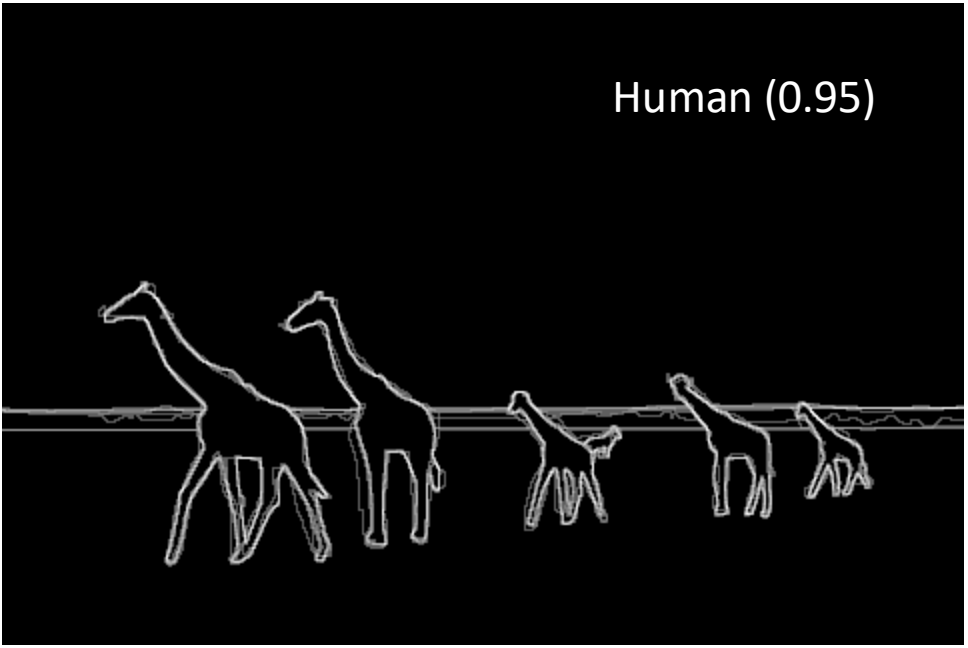




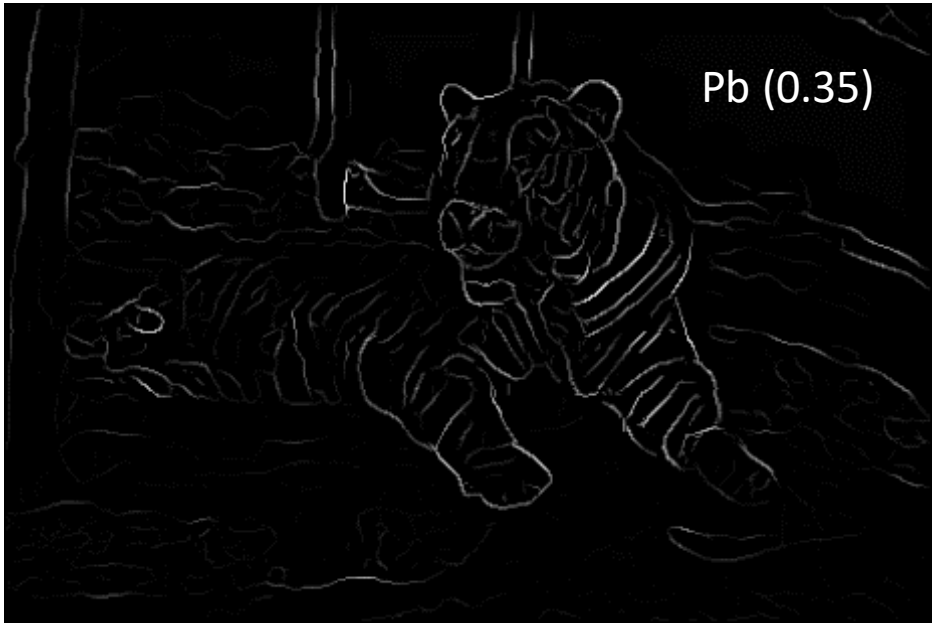
Results

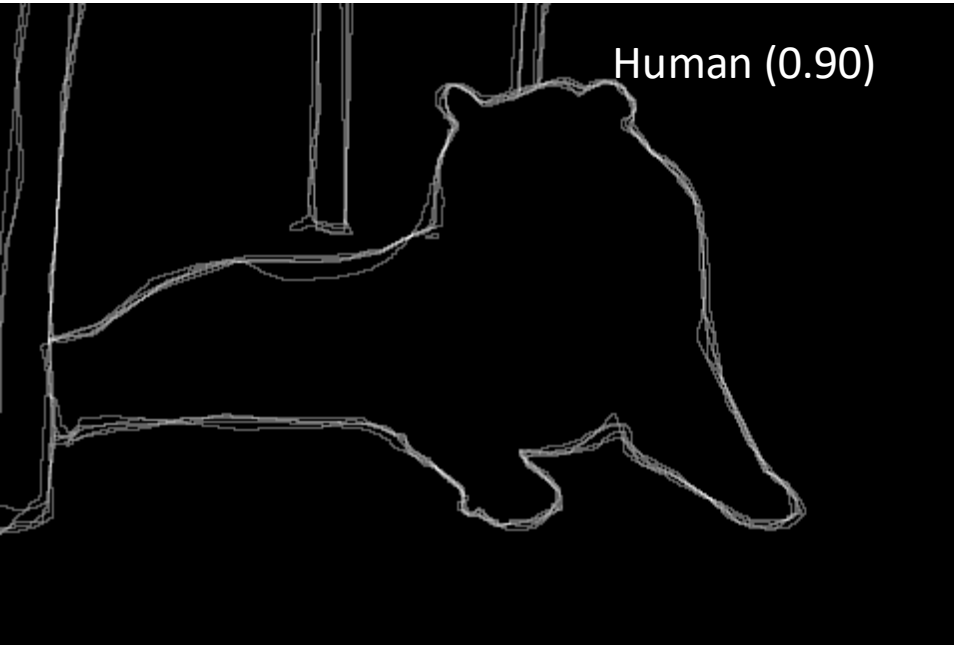


Results



Results





For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html

State of edge detection

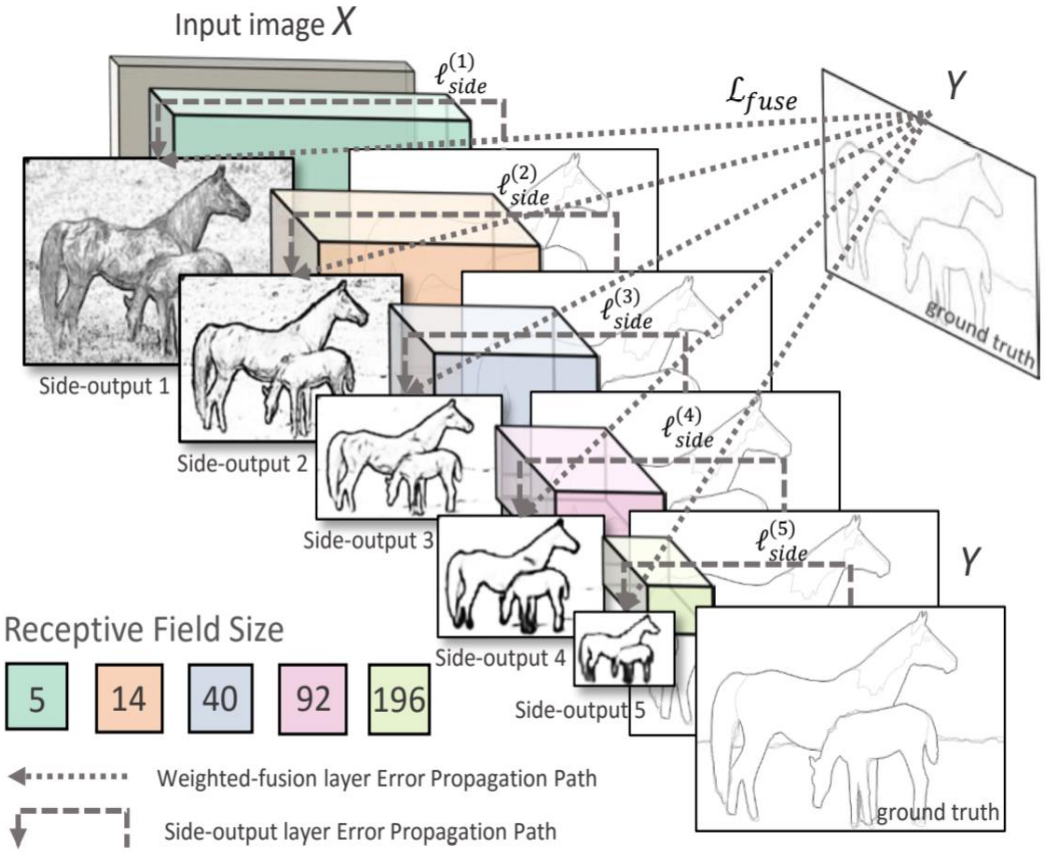
- Local edge detection is mostly solved
  - Intensity gradient, color, texture
- Often used in combination with object detectors or region classifiers
- Deep learning approach is more common nowadays

More Recent Methods in Edge Detection

- Structured Forests for Fast Edge Detection (Dollar et al, 2013)
- Crisp Boundary Detection using Pointwise Mutual Information (Isola et al, 2014)
- Holistically Nested Edge Detection (Xie et al, 2015)

More Recent Methods in Edge Detection

- Holistically Nested Edge Detection (Xie et al, 2015)



	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.600	.640	.580	15
Felz-Hutt [9]	.610	.640	.560	10
BEL [5]	.660*	-	-	1/10
gPb-owt-ucm [1]	.726	.757	.696	1/240
Sketch Tokens [24]	.727	.746	.780	1
SCG [31]	.739	.758	.773	1/280
SE-Var [6]	.746	.767	.803	2.5
OEF [13]	.749	.772	.817	-
DeepNets [21]	.738	.759	.758	1/5†
N4-Fields [10]	.753	.769	.784	1/6†
DeepEdge [2]	.753	.772	.807	$1/10^3$†
CSCNN [19]	.756	.775	.798	-
DeepContour [34]	.756	.773	.797	1/30†
HED (ours)	**.782**	**.804**	**.833**	2.5†, 1/12

Source: Derek Hoiem

Finding straight lines

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2^{nd} moment matrix of their points

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x-\mu_x)^2 & \sum (x-\mu_x)(y-\mu_y) \\ \sum (x-\mu_x)(y-\mu_y) & \sum (y-\mu_y)^2 \end{bmatrix} \qquad [v, \lambda] = \mathrm{eig}(\mathbf{M})$$

Larger eigenvector

$$\theta = \tan 2(v(2,2), v(1,2))$$

$$conf = \lambda_2 / \lambda_1$$

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x-\mu_x)^2 & \sum (x-\mu_x)(y-\mu_y) \\ \sum (x-\mu_x)(y-\mu_y) & \sum (y-\mu_y)^2 \end{bmatrix} \qquad [v, \lambda] = \text{eig}(\mathbf{M})$$

Larger eigenvector

$$\theta = \text{atan}\,2(v(2,2), v(1,2))$$

$$conf = \lambda_2 / \lambda_1$$

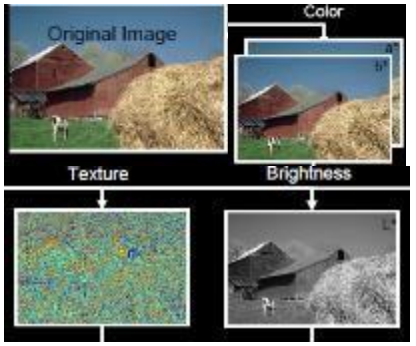
5. Threshold on straightness, store segment

Canny lines → … →straight edges

Things to remember

- Canny edge detector =
  smooth → derivative → thin → threshold → link

- Pb: learns weighting of gradient, color, texture differences



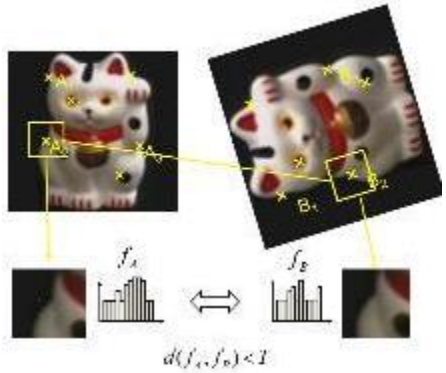
- Straight line detector =
  canny + gradient orientations → orientation binning → linking → check for straightness

Acknowledgements

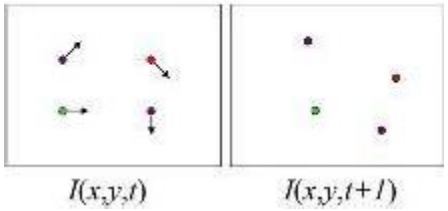
- Thanks to the following researchers for making their teaching/research material online
  - Forsyth
  - Steve Seitz
  - Noah Snavely
  - J.B. Huang
  - Derek Hoiem
  - D. Lowe
  - A. Bobick
  - S. Lazebnik
  - K. Grauman
  - R. Zaleski

Next classes: Correspondence and Alignment

- Detecting interest points



- Tracking points



- Object/image alignment and registration
  - Image stitching