TDS3651
Visual Information Processing

Edges
Lecture 4

Faculty of Computing and Informatics
Multimedia University

Previously created by John See, edited by Lai-Kuan, Wong, Yuen Peng, Loh
Some graphics and materials are taken from UTexas at Austin CS376 & CS381V notes, Stanford Uni's CS131 notes and Gonzalez & Wood's DIP textbook

Lecture Outline

- What makes an edge?
- Gradient-based Edge Detectors
- Canny Edge Detector

Revision: Filtering

- What is the output of the following filters?

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7 & 4 \\ \hline
1 & 5 & 8 & 9 & 3 & 1 \\ \hline
2 & 7 & 2 & 5 & 1 & 3 \\ \hline
0 & 1 & 3 & 1 & 7 & 8 \\ \hline
4 & 2 & 1 & 6 & 2 & 8 \\ \hline
2 & 4 & 5 & 2 & 3 & 9 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
\end{array}
$$

Revision: Filtering

- What is the output of the following filters?

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Revision: Filtering

- What is the output of the following filters?

3	0^{-1}	1^{0}	2^{1}	7	4
1	5^{-1}	8^{0}	9^{1}	3	1
2	7^{-1}	2^{0}	5^{1}	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4		

Revision: Filtering

- What is the output of the following filters?

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	

Revision: Filtering

- What is the output of the following filters?

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1^{-1}	7^{0}	8^{1}
4	2	1	6^{-1}	2^{0}	8^{1}
2	4	5	2^{-1}	3^{0}	9^{1}

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	7
3	2	3	16

Revision: Filtering

- What does the following filter do?

Filter 1

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

Filter 2

1	-2	1
-2	4	-2
1	-2	1

A. Smoothing
B. Sharpening
C. Extract details
D. Denoising

Revision: Filtering

- What does the following filter do?

Filter 1

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

Filter 2

1	-2	1
-2	4	-2
1	-2	1

A. Smoothing
B. Sharpening
C. Extract details
D. Denoising

Result

Recall: Unsharp Masking

Original

$$* \quad \frac{1}{9}\begin{bmatrix}-1&-1&-1\\-1&17&-1\\-1&-1&-1\end{bmatrix} \quad =$$

Sum to 1
(normalized)

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

Un-normalized filter

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} + \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

Details of Image

Recall: Image filtering

- Compute a function of the local neighbourhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbours
- Uses of filtering
  - Enhance an image (denoising, sharpening, etc)
  - **Extract information (edges, textures, etc)**

What makes an edge?

Edge Detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Most semantic and shape information from the image can be encoded in the edges
- **Ideal:** Artist's line drawing (but artist is also using object-level knowledge)

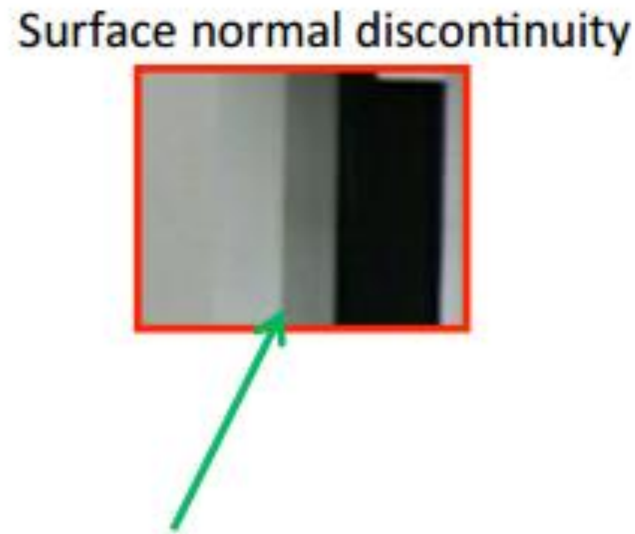
Source: D. Lowe

Edge Detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - ⇒ **Map image from 2D array of pixels to a set of curves or line segments/contours**
- **Main Idea:** **Look for strong gradients, post-process**

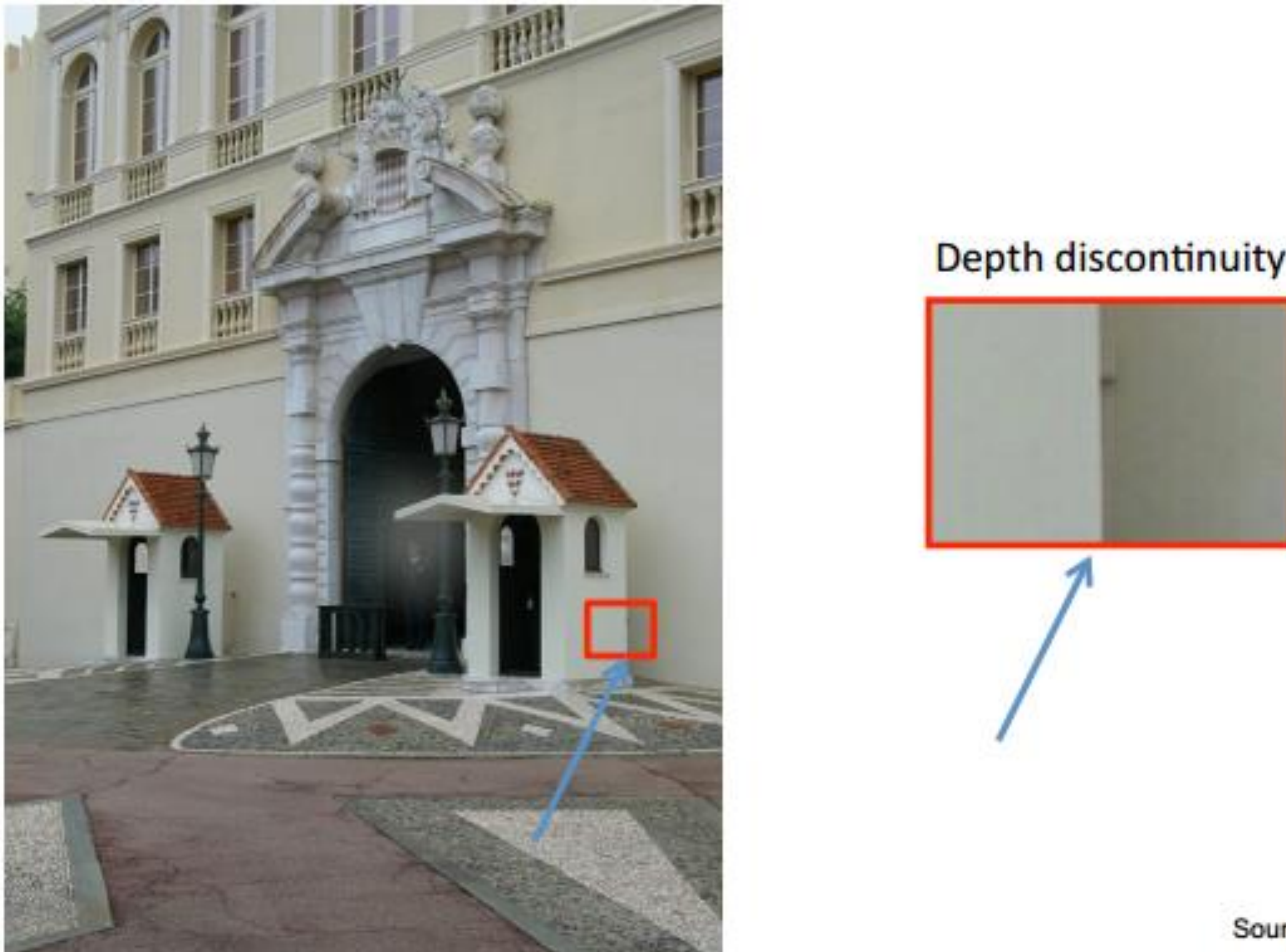
A close lookout for “edges”

A close lookout for “edges”



Source: D. Hoiem

A close lookout for “edges”



Source: D. Hoiem

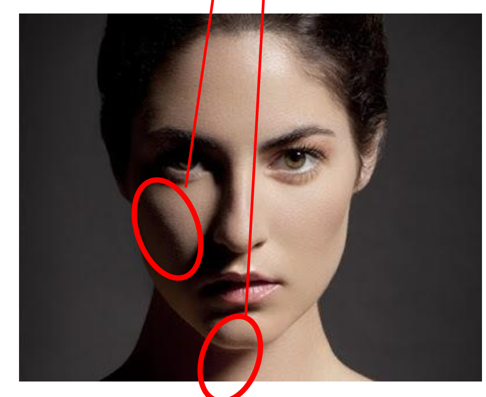
A close lookout for “edges”

Surface color discontinuity

Source: D. Hoiem

A close lookout for “edges”

Potential edges detected!

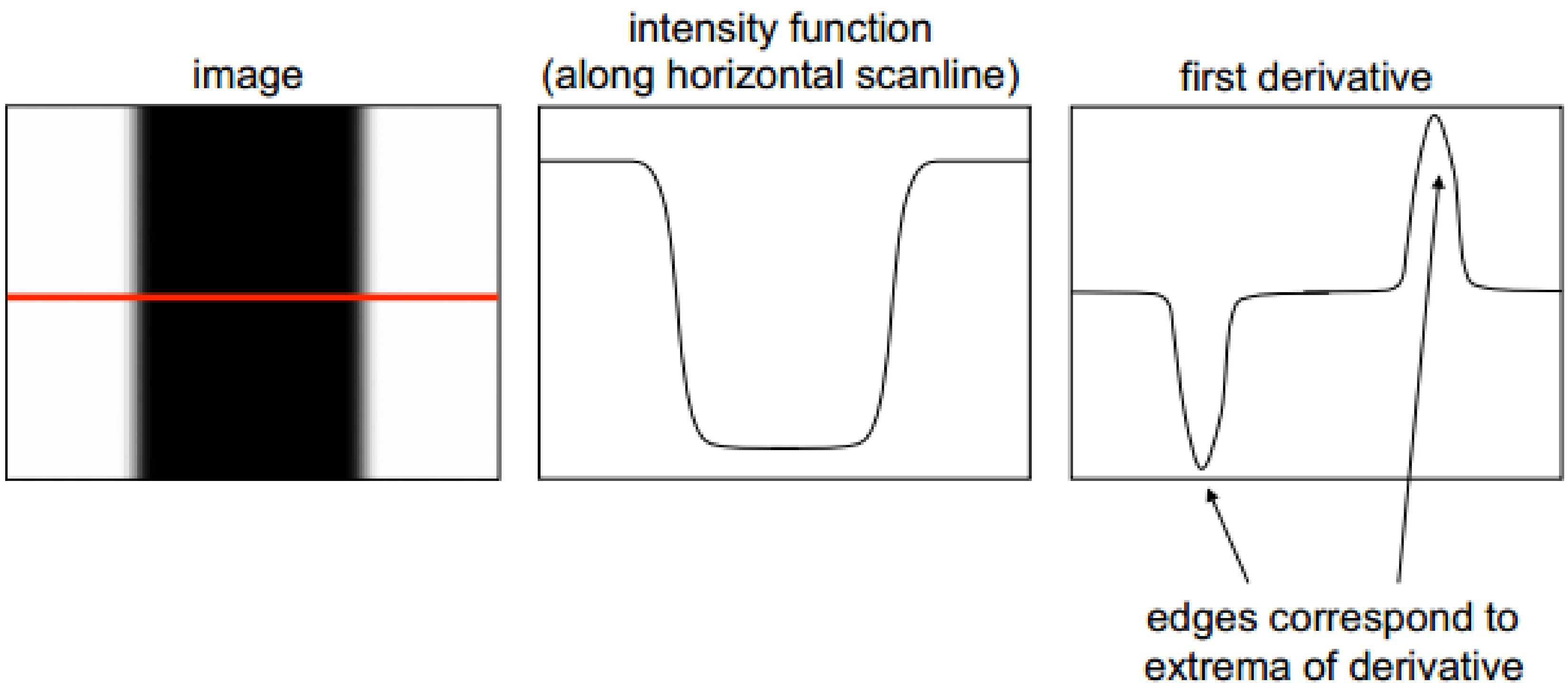


Illumination discontinuity ?

Gradient/Derivative-based edge detectors

Derivatives and 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:

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

- 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 it by convolution, what would be the associated filter?

Partial derivatives of an image

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

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

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

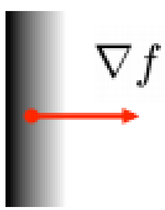
$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Image gradient

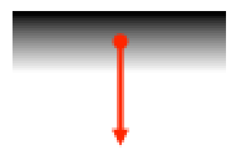
- The gradient of an image:

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

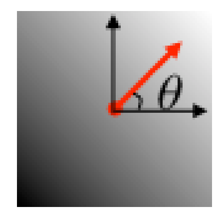
- Gradient – Points in the direction of most rapid increase in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

- **Direction**:

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

- **Magnitude** ("edge strength") $||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

Partial derivatives of an image

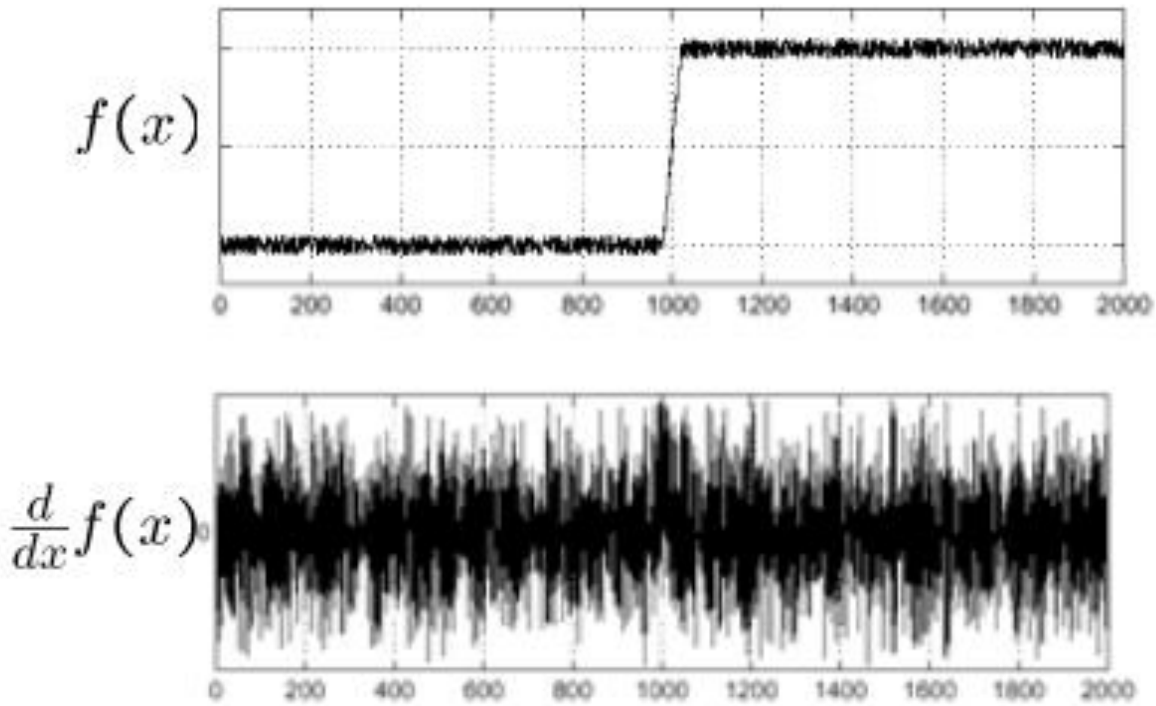
- Taking both partial derivatives in the x direction and y direction, we get the full set of derivatives or "gradients" corresponding to both directions

Looking at a row profile

Intensity profile

Effects of noise

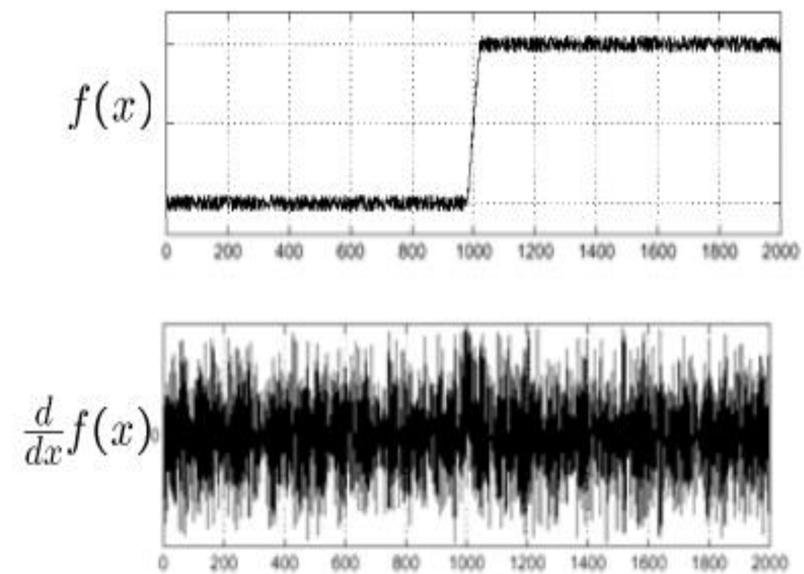
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Where's the edge??

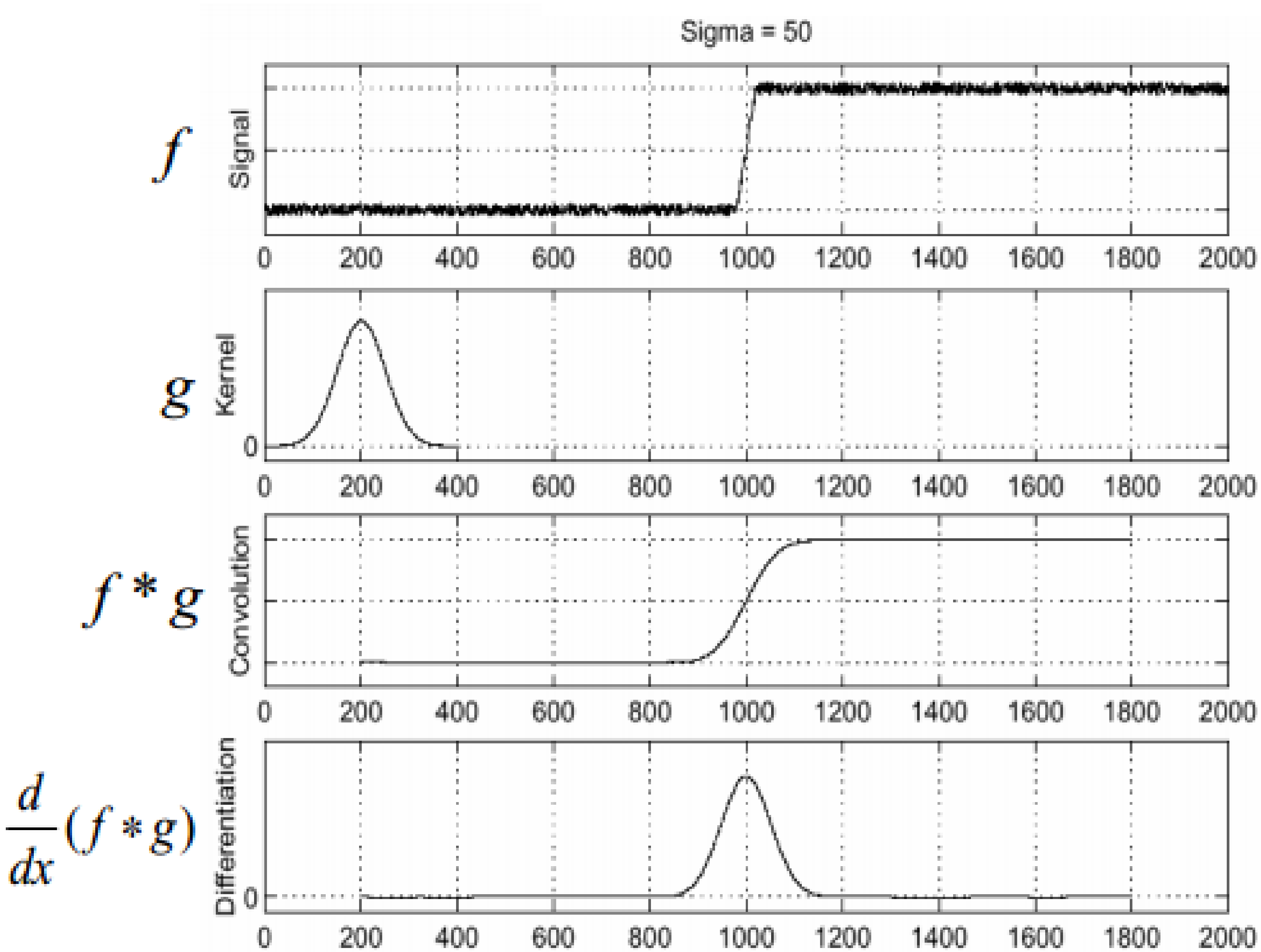
Effects of noise

- Gradient filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbours
  - What can be done?



A. Smoothing to make pixels look more like their neighbours
B. Thresholding to remove all the noise pixels
C. Filtering to transform the pixels into a clearer format
D. Nothing can be done

Solution:



Sigma = 50
f
Signal
g
Kernel
$f * g$
Convolution
$\frac{d}{dx}(f * g)$
Differentiation
0
200
400
600
800
1000
1200
1400
1600
1800
2000

Derivative theorem of convolution

- Differentiation is convolution, convolution is associative

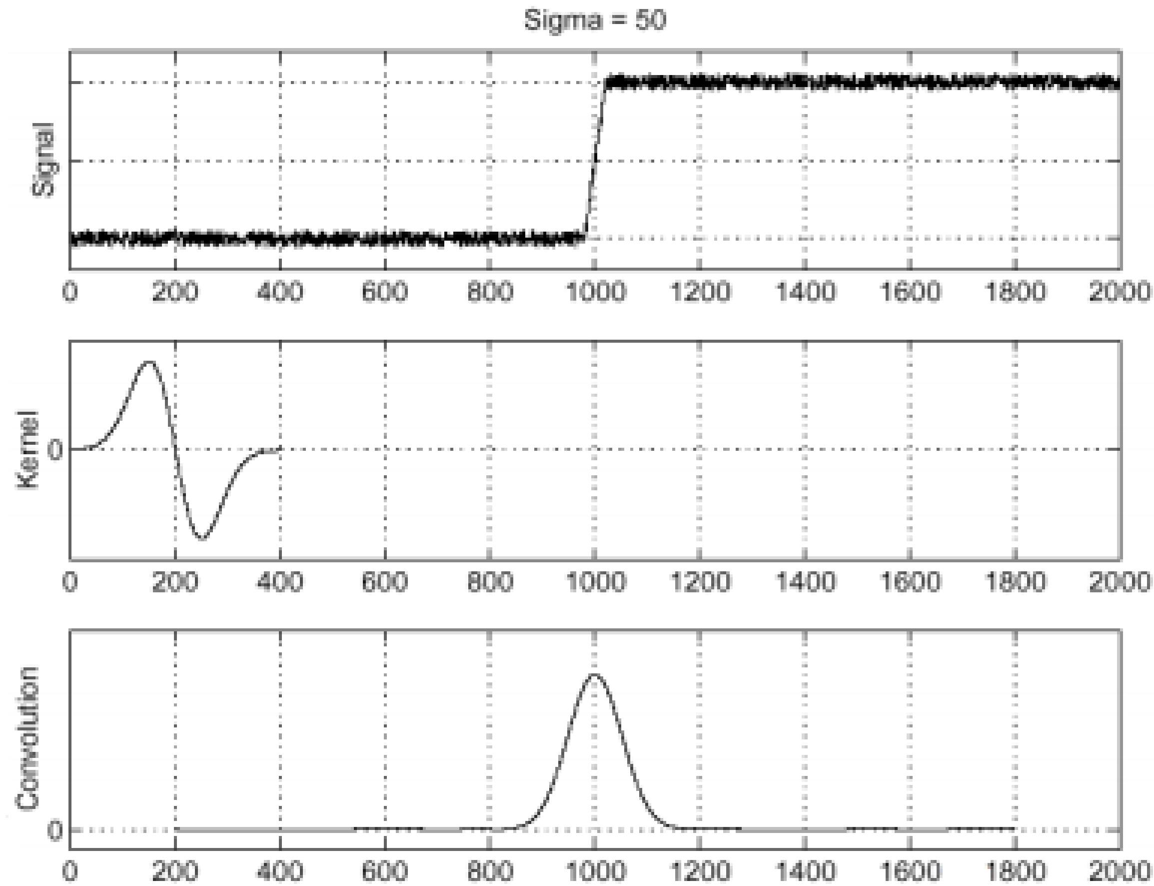
$$\frac{d}{dx}(f*g) = f*\frac{d}{dx}g$$

Saves one operation!

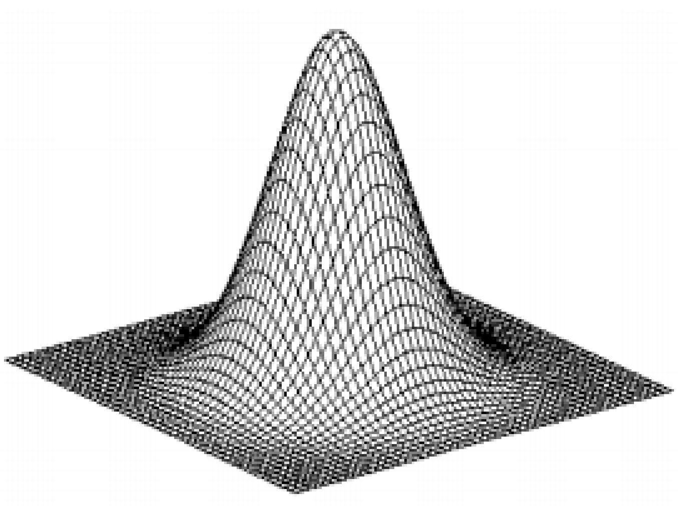
f

$\frac{d}{dx}g$

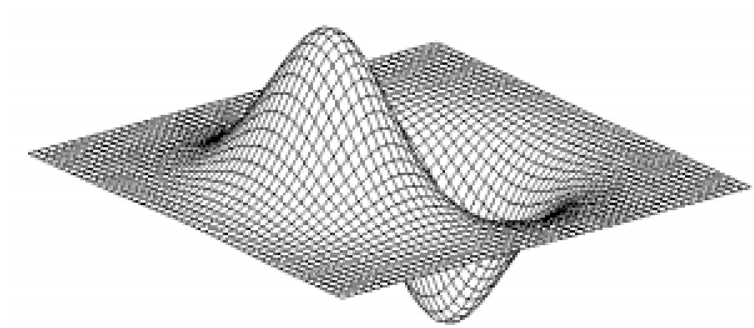
$f*\frac{d}{dx}g$



Derivative of Gaussian filter



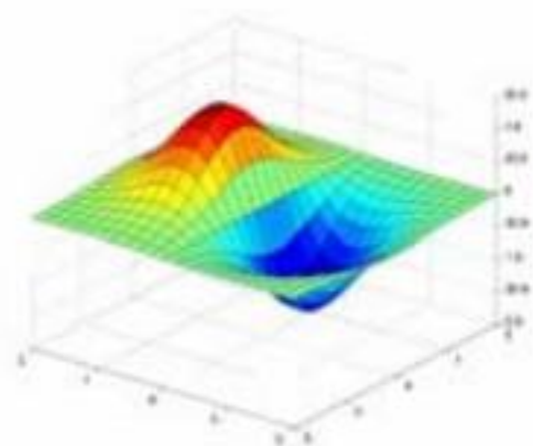
* [1 -1] =



Derivative of Gaussian filter

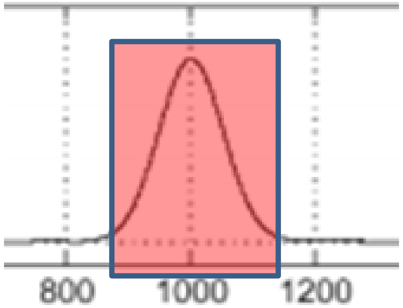
$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

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



Using 1st order gradient filter

Recall, 1st derivative of Gaussian

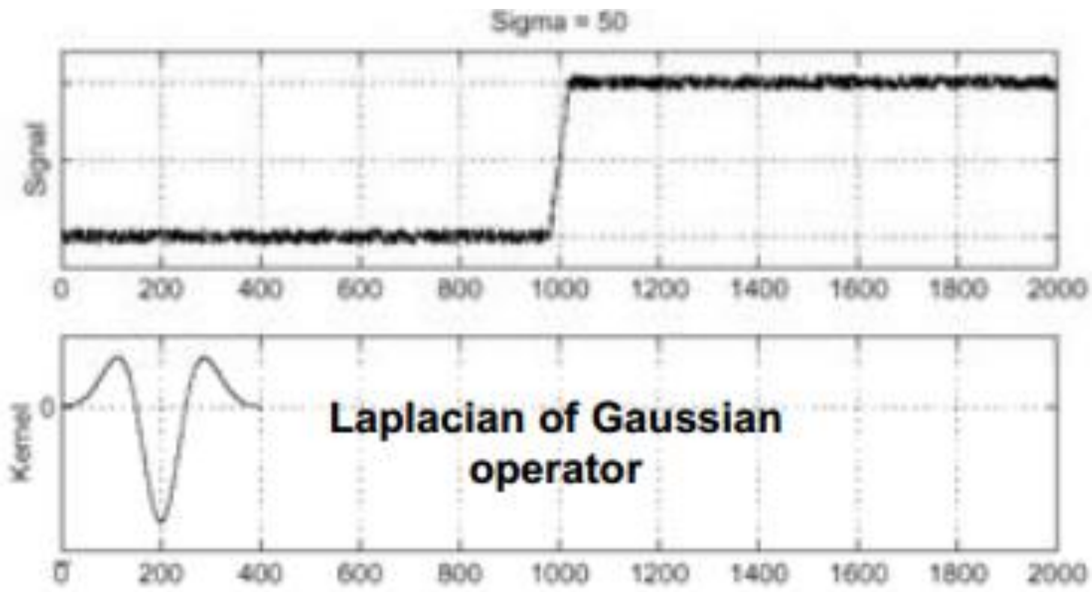


- Gradient magnitude is larger along a thick ridge
  - So, how to identify the actual edge points?
  - If we can, then how do we link the actual edge points to form curves?

Laplacian of Gaussian

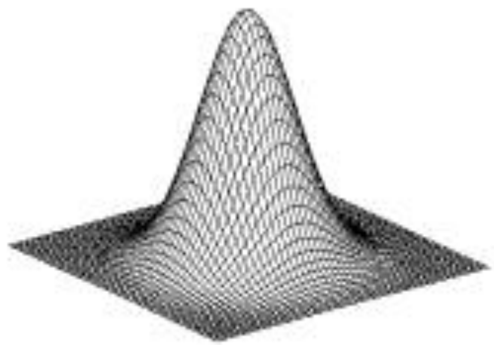
f

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



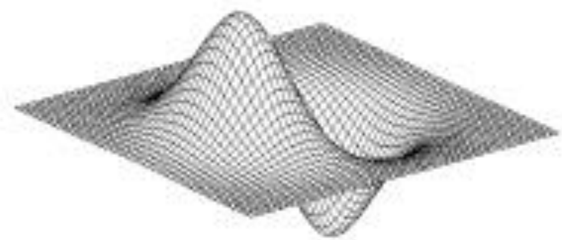
- 2nd derivative of the Gaussian
- Where is the edge? **Zero-crossings of last graph**

2D Gradient-based Edge Filters



Gaussian

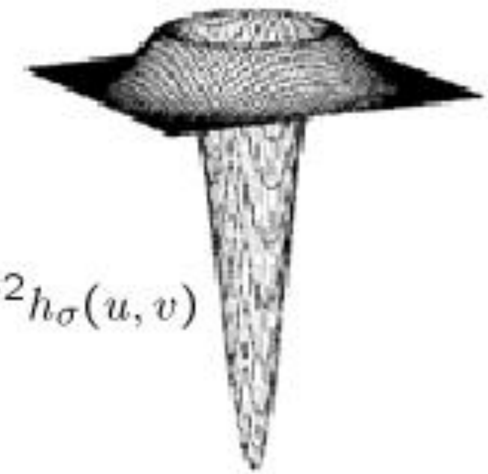
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



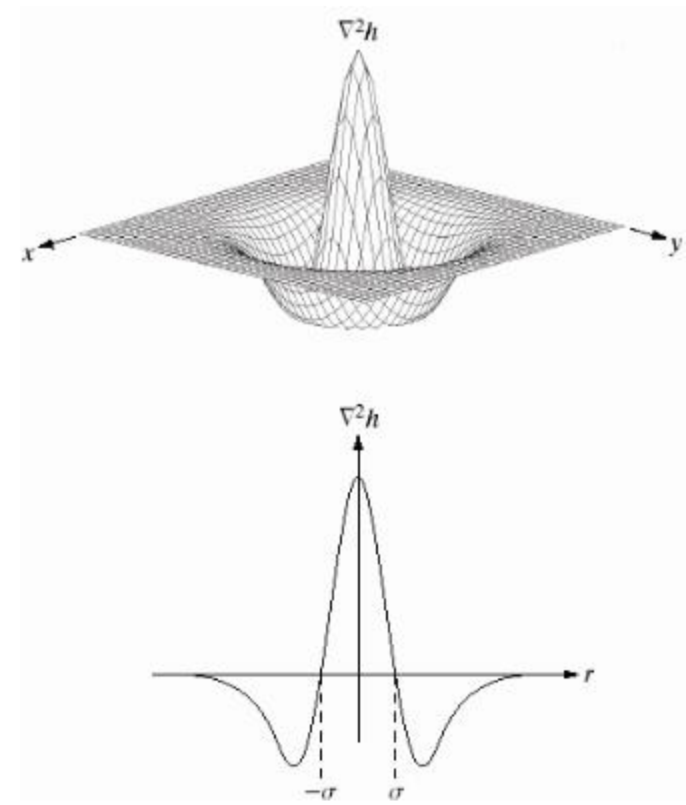
$$\nabla^2 h_\sigma(u,v)$$

∇^2 **Laplacian operator**

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

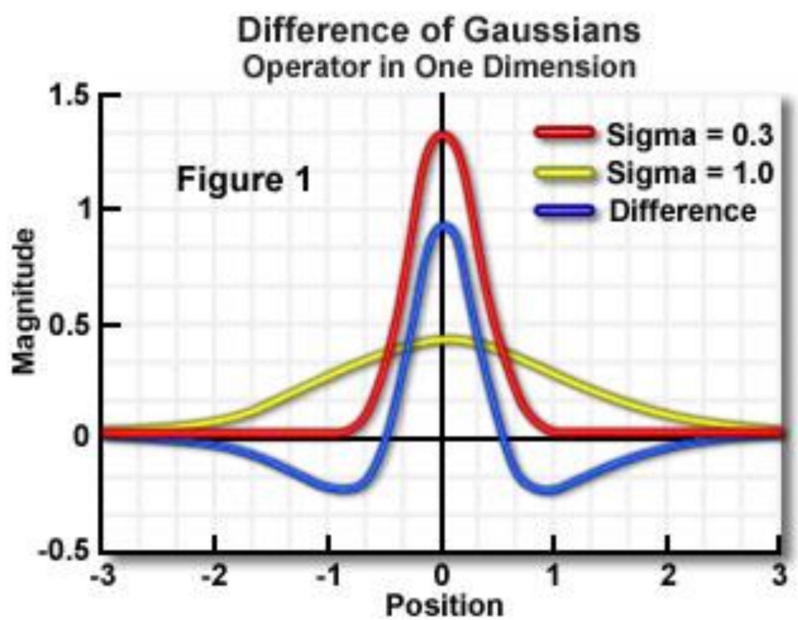
Laplacian of Gaussian (LoG) filter

- **LoG filter –** also known as Marr-Hildreth algorithm
  - Convolution of the image with the Laplacian of the Gaussian function
  - Zero crossings are detected to obtain edges
  - Its operator sometimes known as **Mexican hat operator**



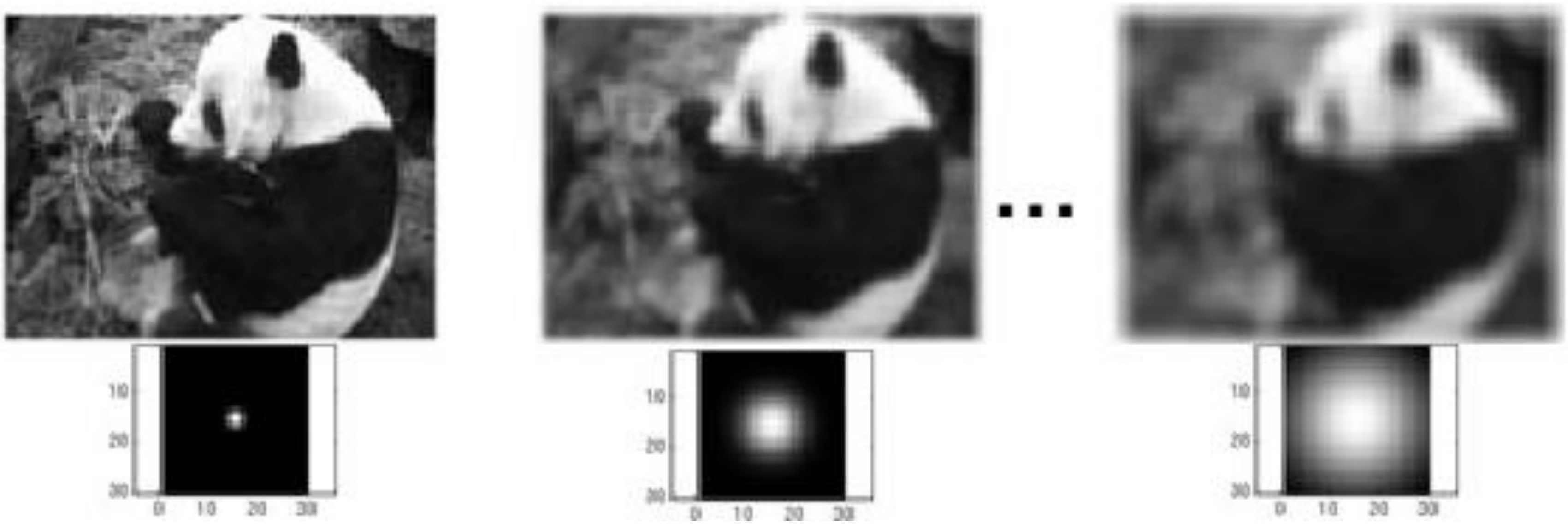
DoG $\Longrightarrow$ LoG

- **Difference of Gaussians (DoG) filter –** a fast approximation of the LoG filter
  - Subtract one blurred version of original image (by Gaussian filter) from another, less blurred version of original
    - $\Rightarrow$ Just use different σ values to create the two different blurred versions!



Smoothing with a Gaussian

- **Recall:** Parameter σ is the “scale” or “width” of the Gaussian kernel, controls amount of smoothing



Effect of σ on derivatives

- Edge structure differs depending on Gaussian kernel's scale parameter

- **Larger** values: thick, only important edges detected
- **Smaller** values: finer edges detected, but too much details
- So, what scale to choose?

Other finite difference filters

- “Finite difference filters” – approximates the gradient of image intensity function

Roberts

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Prewitt

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobel

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

Other finite difference filters

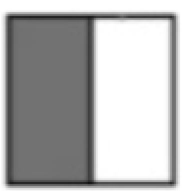
- "Finite difference filters" – approximates the gradient of image intensity function

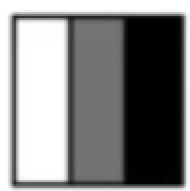
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

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

=





Other finite difference filters

- “Finite difference filters” – approximates the gradient of image intensity function

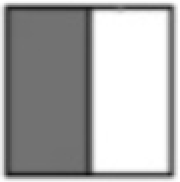
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

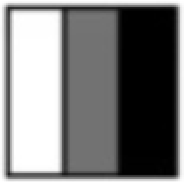
*

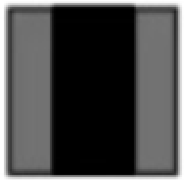
1	0	-1
1	0	-1
1	0	-1

=

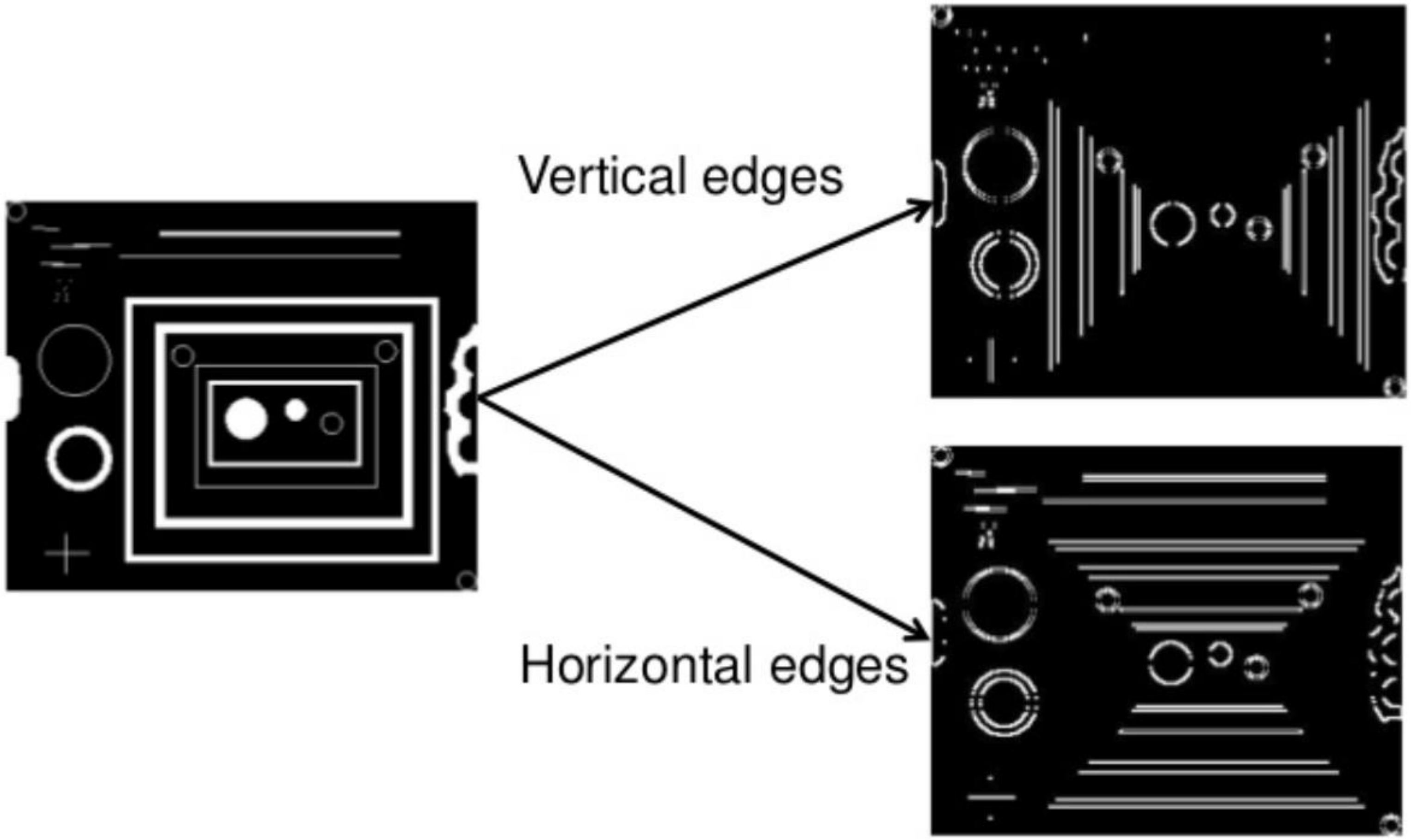
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0







Other finite difference filters



Mask properties

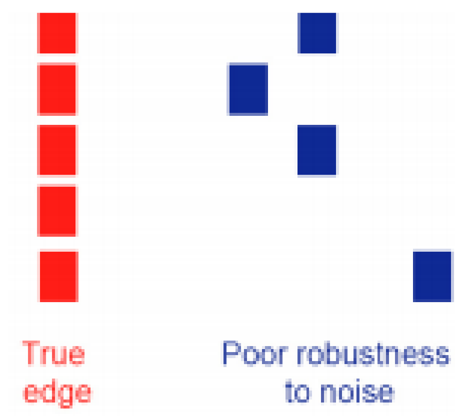
- **Smoothing**
  - Values positive
  - Sum to 1 $\Longrightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size

- **Edge Derivatives**
  - Different signs (positive, negative) are used to get high response in regions of high contrast
  - Sum to ? $\Rightarrow$ no response in constant regions
  - High absolute value at points of high contrast

A. 3
B. 2
C. 1
D. 0

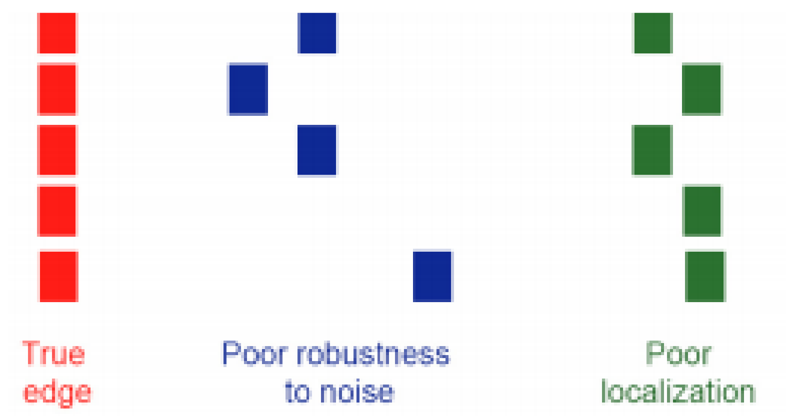
Designing an edge detector

- Criteria of an "optimal" edge detector
  - **Good detection**: Must minimize probability of false positives (spurious edges due to noise), and false negatives (missing real edges)



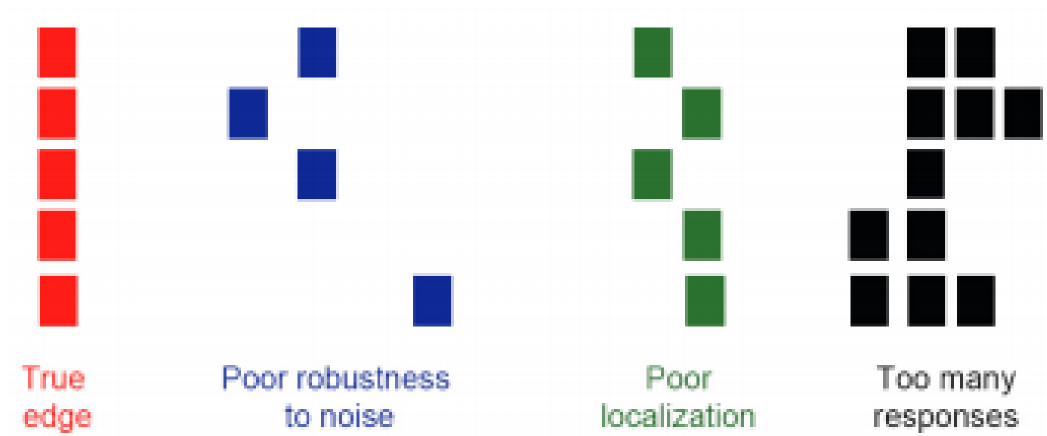
Designing an edge detector

- Criteria of an "optimal" edge detector
  - **Good detection**: Must minimize probability of false positives (spurious edges due to noise), and false negatives (missing real edges)
  - **Good localization**: Edges detected are as close as possible to true edges



Designing an edge detector

- Criteria of an “optimal” edge detector
  - **Good detection**: Must minimize probability of false positives (spurious edges due to noise), and false negatives (missing real edges)
  - **Good localization**: Edges detected are as close as possible to true edges
  - **Single response**: Detector must return one point only for each true edges



Designing an edge detector

- Primary edge detection steps:
  - **Smoothing:** Suppress noise
  - **Edge enhancement:** Filter for contrast
  - **Edge localization:** Determine which local maxima from filter output are actually edges
    - **Thresholding**

Thresholding

- Choose a threshold value ***t***
- Set any pixels less than *t* to **zero** (off)
- Set any pixels greater or equal to *t* to **one** (on)

Original image

Thresholding: Gradient magnitude image

Which threshold to use?

Lower threshold value

Higher threshold value

What can we conclude from using different threshold values?

A. A universal value can be used for the best thresholding.
B. Lower threshold will detect all edges we need.
C. Higher threshold may miss out on some edges.
D. A random value can be used to perform thresholding.

Canny edge detector

Canny edge detector

- Most widely used edge detector – Stable, consistent
- Steps:
  1. **Filter image with Gaussian filter**, find magnitude and orientation of gradient
  2. **Non-maximum suppression**
     - Thin wide "ridges" down to single pixel width (get rid of spurious responses)
  3. **Hysteresis thresholding and linking**
     - Define 2 thresholds: low and high (to mark strong and weak edges)
     - Start edge curves from the true edges and continue tracking based on weak edges that are connected to the true edges

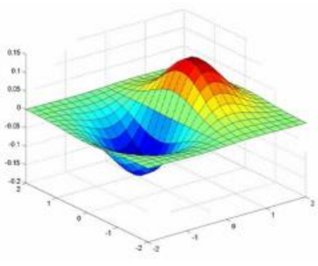
Normally we just threshold after filtering

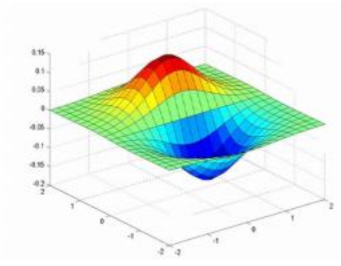
Canny edge detector

- Most widely used edge detector – Stable, consistent
- Steps:
  1. **Filter image with Gaussian filter**, find magnitude and orientation of gradient
  2. **Non-maximum suppression**
     - Thin wide “ridges” down to single pixel width (get rid of spurious responses)
  3. **Hysteresis thresholding and linking**
     - Define 2 thresholds: low and high (to mark strong and weak edges)
     - Start edge curves from the true edges and continue tracking based on weak edges that are connected to the true edges

Step 1

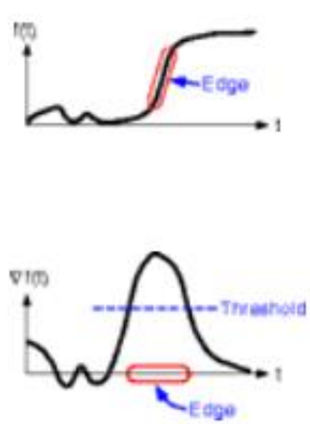
- Lena Example:
  - Step 1 can be done in one pass, by filtering with the derivative of Gaussian filter (both x and y directions)





2 Problems when thresholding

- Problem #1:
  - Edges from many gradient-based edge detectors are too thick



How to turn these thick regions into single pixel edges?

Canny edge detector

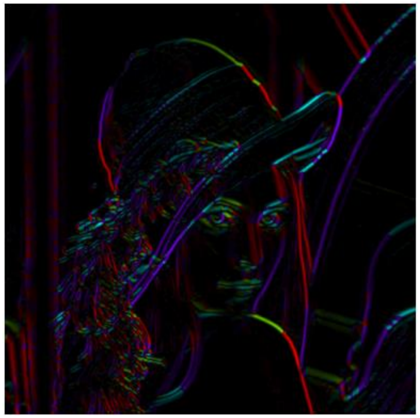
- Most widely used edge detector – Stable, consistent
- Steps:

1. **Filter image with Gaussian filter**, find magnitude and orientation of gradient
2. **Non-maximum suppression**
   - Thin wide “ridges” down to single pixel width (get rid of spurious responses)
3. **Hysteresis thresholding and linking**
   - Define 2 thresholds: low and high (to mark strong and weak edges)
   - Start edge curves from the true edges and continue tracking based on weak edges that are connected to the true edges

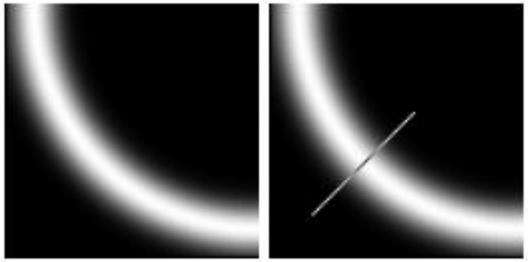
Step 2

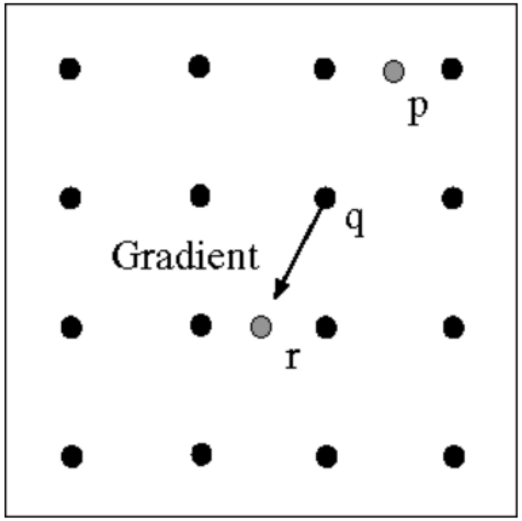
- Get orientation at each pixel

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



- **Non-maximum suppression**
  - Check if pixel is local maximum along gradient direction e.g. maximum at q if value larger than p and r (interpolated pixels). So, suppress p and r.





2 Problems when thresholding

- Problem #2:
  - Pixels along weak edges will not survive if standard thresholding is applied (using a threshold determined globally)

How can we keep the pixels along

A. The keep them.
B. Mul n them.
C. The weak.
D. All t ense.

Result

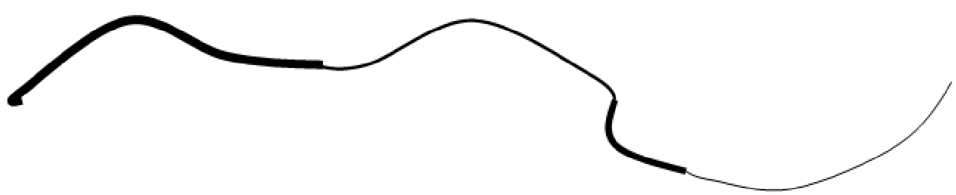
Canny edge detector

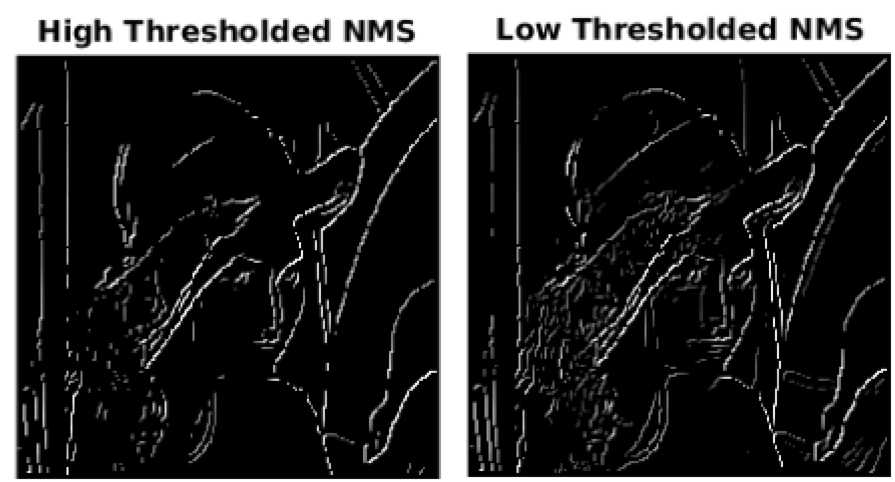
- Most widely used edge detector – Stable, consistent
- Steps:

1. **Filter image with Gaussian filter**, find magnitude and orientation of gradient
2. **Non-maximum suppression**
   - Thin wide "ridges" down to single pixel width (get rid of spurious responses)
3. **Hysteresis thresholding and linking**
   - Define 2 thresholds: low and high (to mark strong and weak edges)
   - Start edge curves from the true edges and continue tracking based on weak edges that are connected to the true edges

Step 3

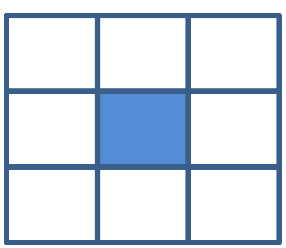
- Double threshold
  - **High threshold** – (Gradient higher than this threshold) ⟹ **strong** edges
  - **Low threshold** – (Gradient higher than low threshold but lower than high threshold) ⟹ **weak** edges
- The plan: Ensure continuity in edges of different strengths



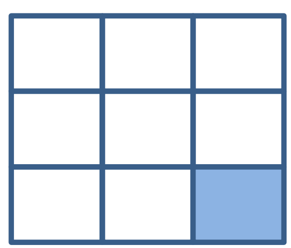


Step 3

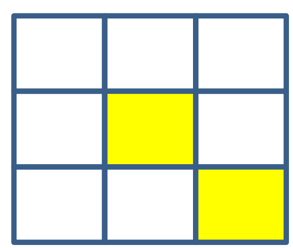
- To "link" edges, perform connected components, starting from strong edge pixels
  - Search 8-neighbourhood for weak edge pixels
  - Usually a weak edge pixel caused by true edges will be connected to a strong edge pixel while noise responses are unconnected.



Strong edge pixel



Weak edge pixel



Linked edges

Final Result

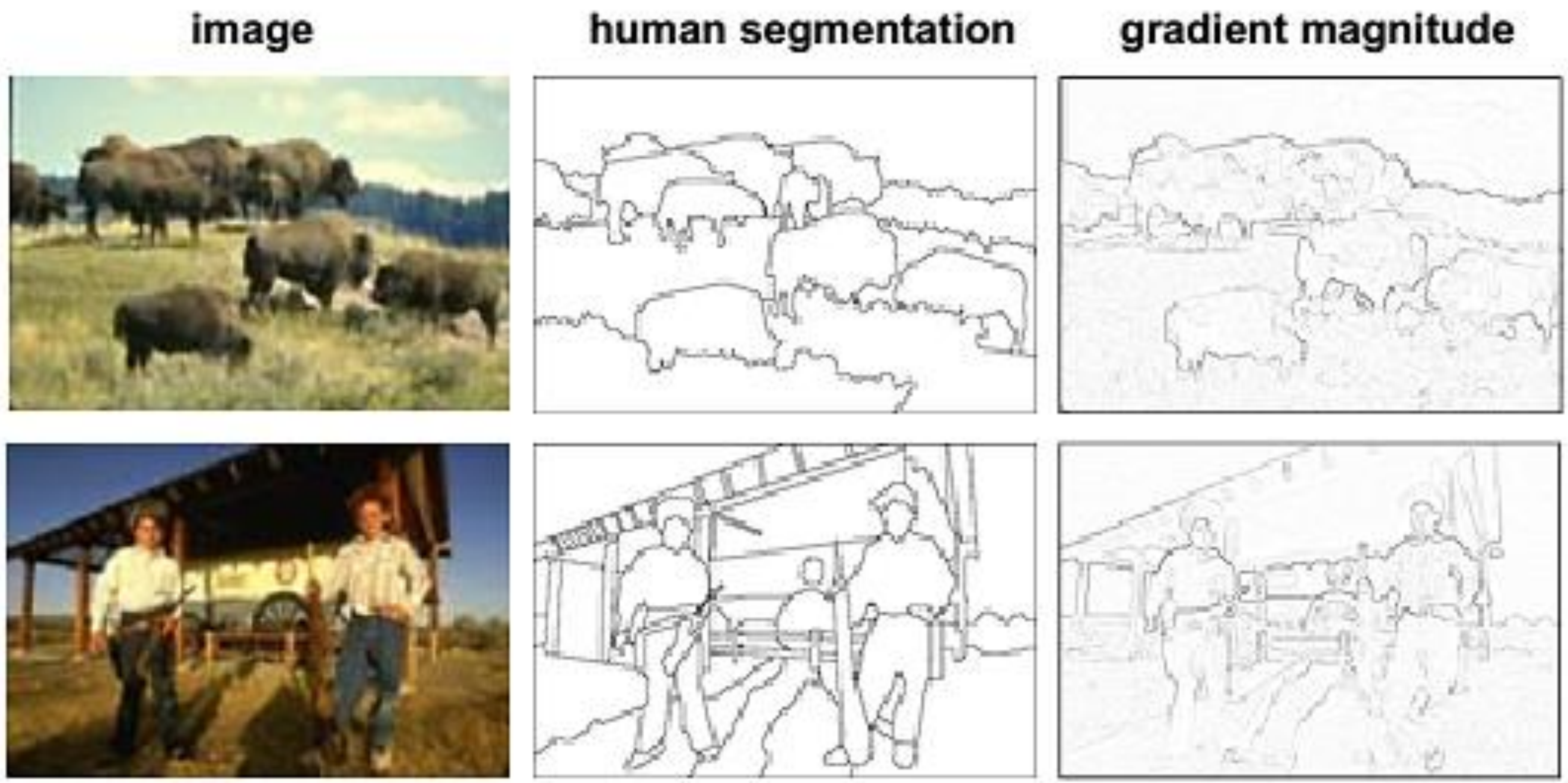
Effect of σ (Gaussian kernel width)

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

- The choice of σ depends on desired behaviour
  - Large σ detect large scale edges (less edges typically)
  - Small σ detects fine features (more edges typically)

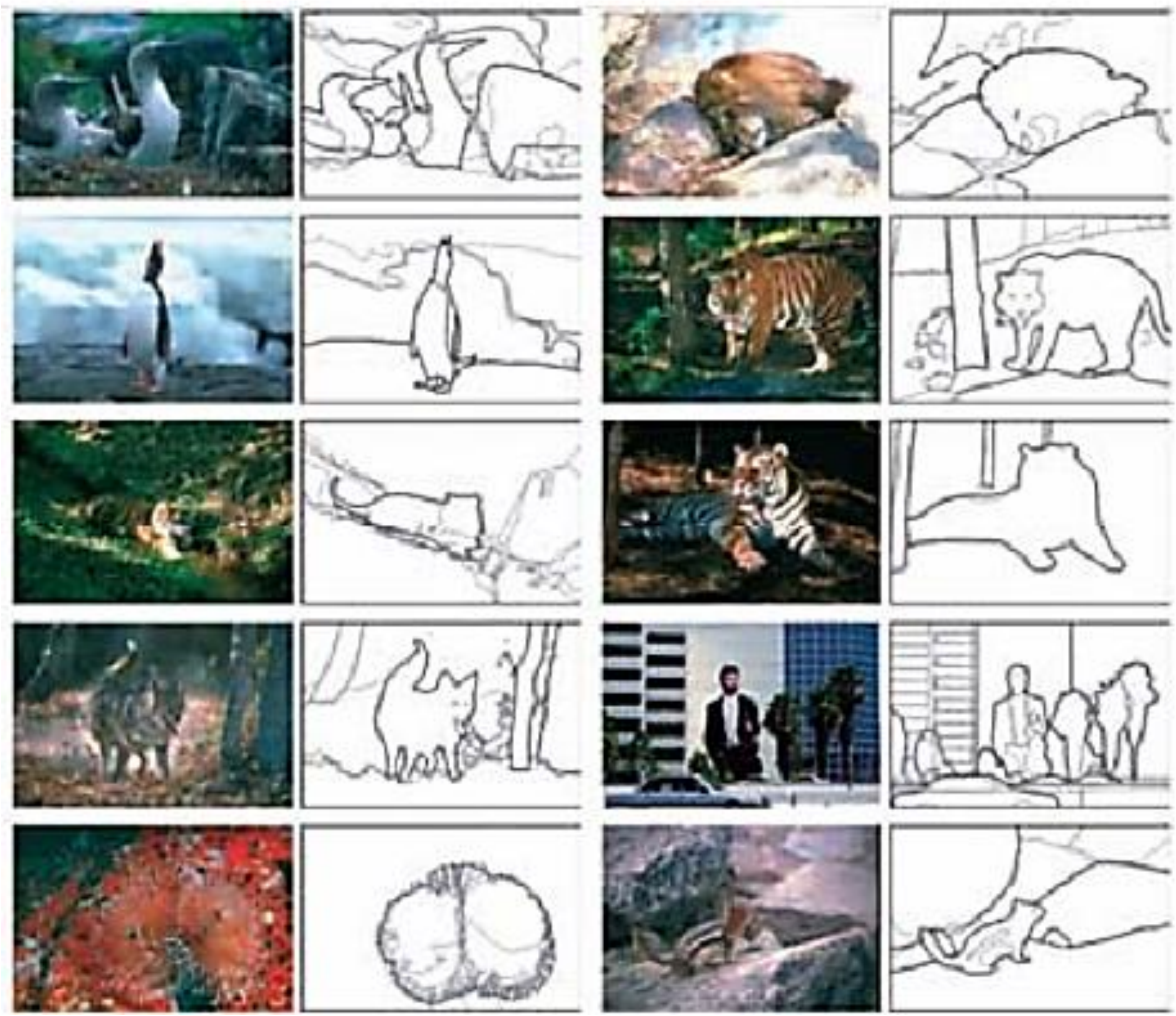
So…how do we know if the edges found are really the TRUE edges?

Gradient edges vs. Human segmentation

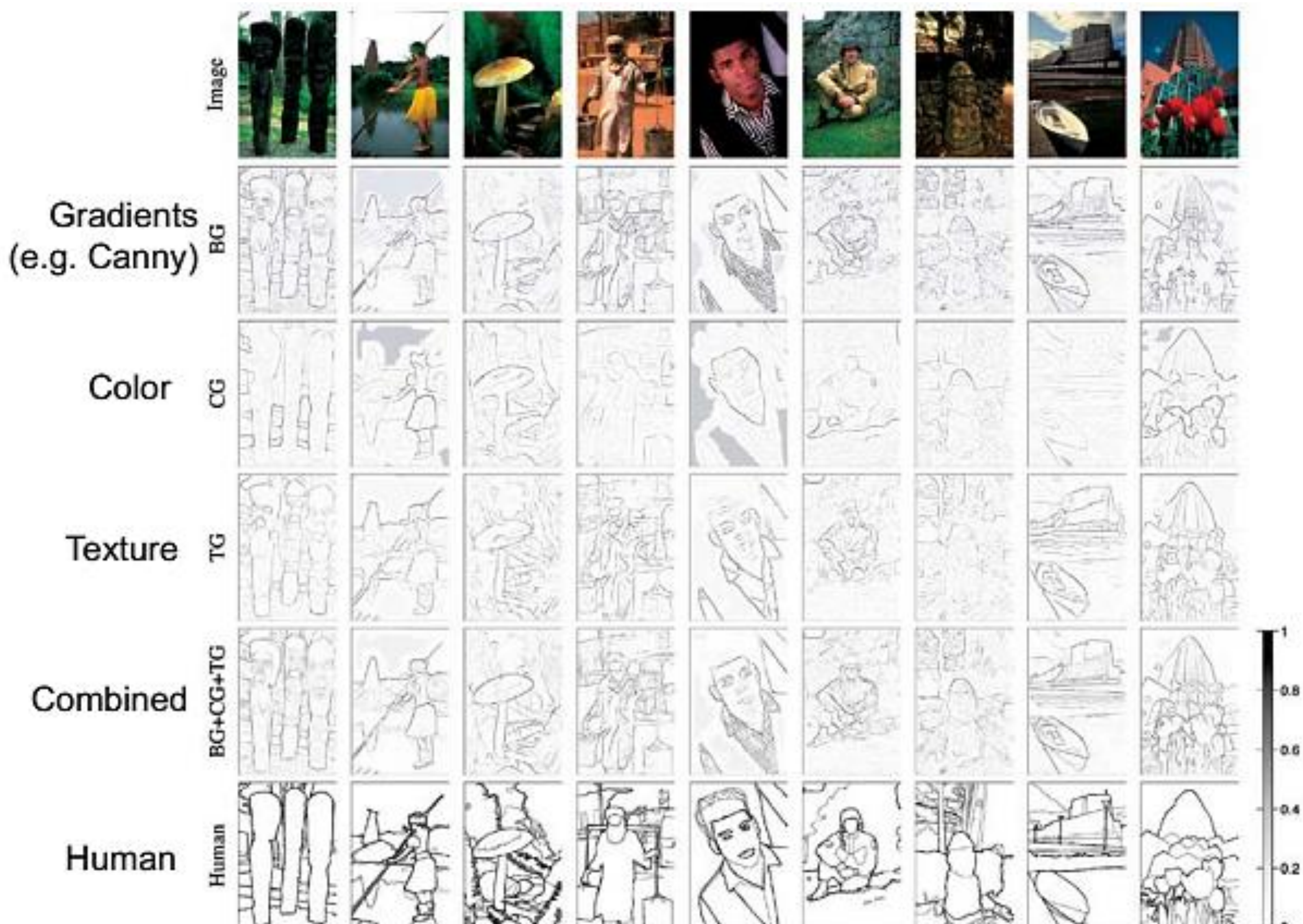


Berkeley segmentation database
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

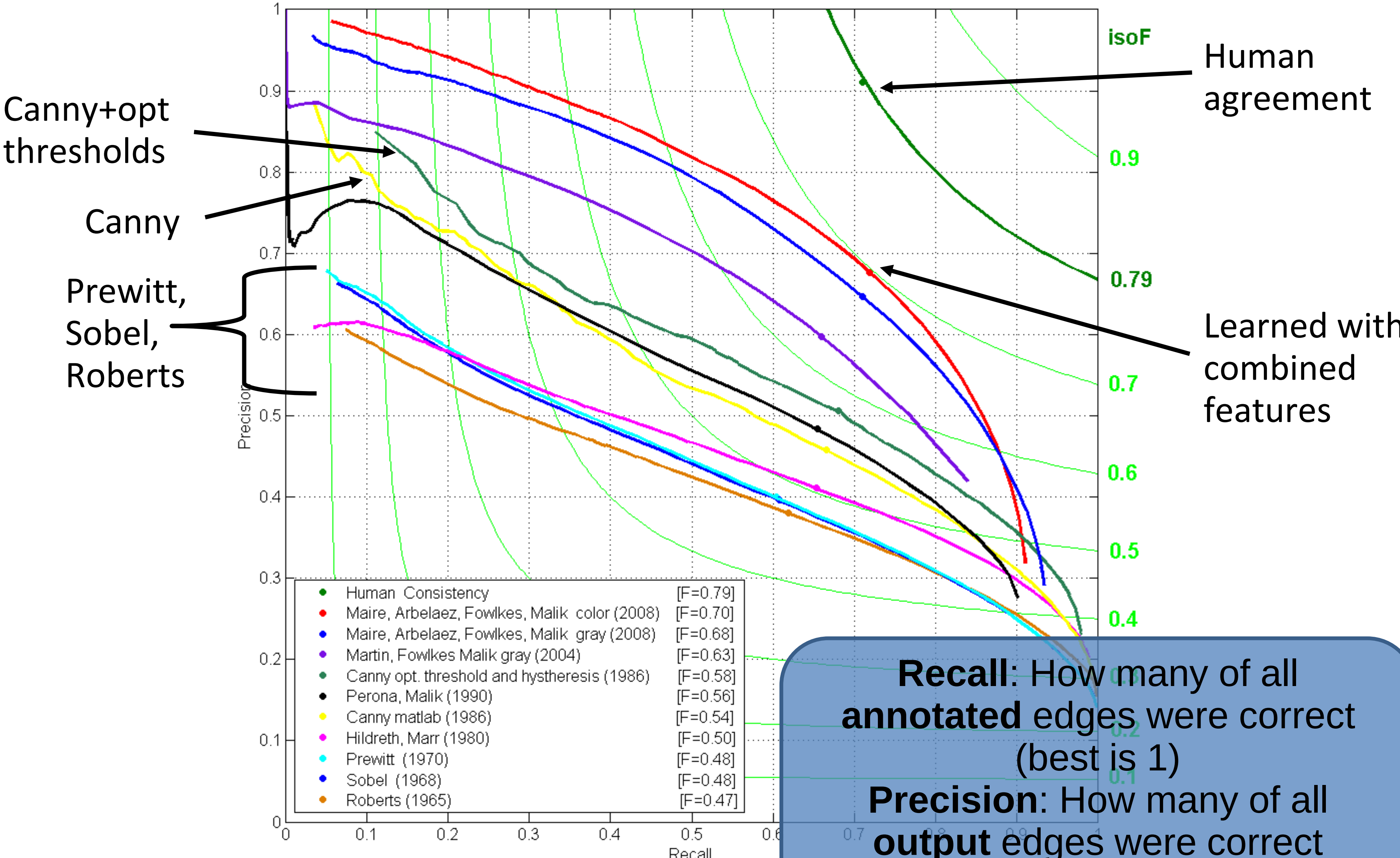
Human-perceived edges



Features used vs. Human-marked

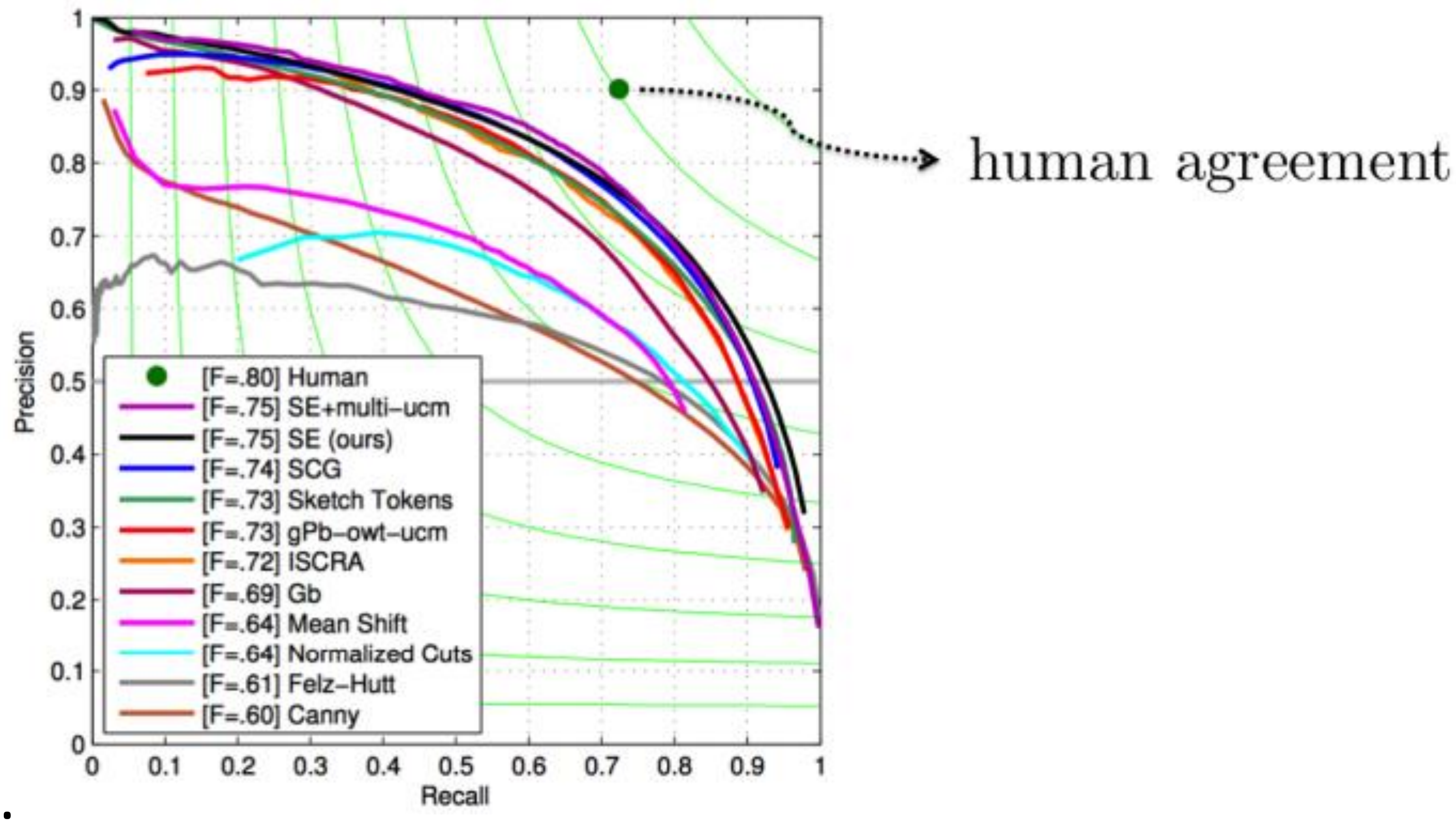


Contour Detection (CVPR 2008)



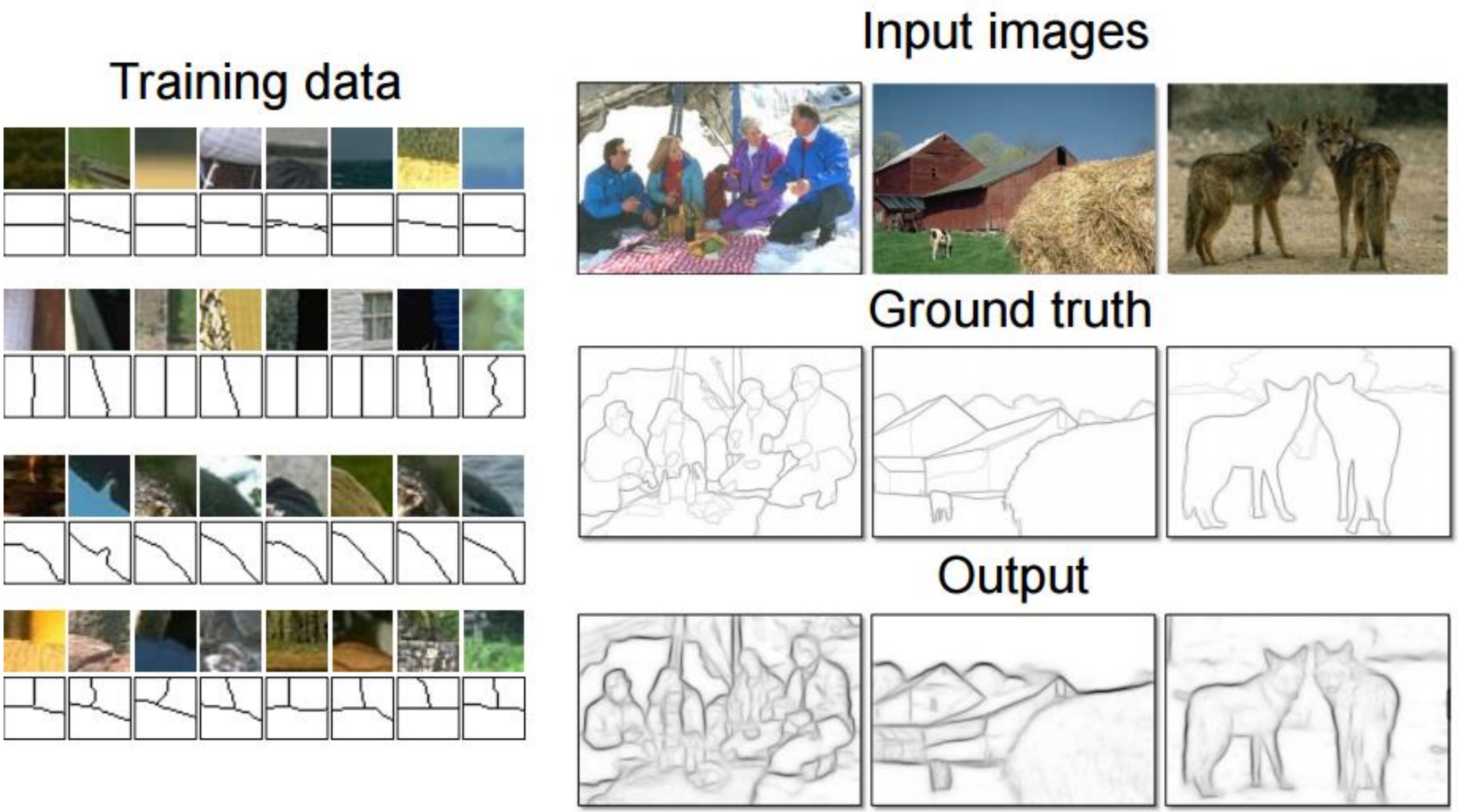
Recall: How many of all **annotated** edges were correct (best is 1)
Precision: How many of all **output** edges were correct (best is 1)

Contour Detection (ICCV 2013)



Best performing:
P. Dollar and C. Zitnick, **"Structured Forests for Fast Edge Detection"** (ICCV 2013)
Code:
Original (Matlab/C++): https://www.microsoft.com/en-us/download/details.aspx?id=52370
Python: https://github.com/ArtanisCV/StructuredForests
OpenCV (C++): https://docs.opencv.org/3.3.1/d0/da5/tutorial_ximgproc_prediction.html

Data-driven edge detection



Machine learning: How can we “train” an edge detector to learn what is an edge?

Deep Learning-based Edge detection

State-of-the-arts deep-learning based edge detection:

- **Holistically-Nested Edge Detection**
  https://ieeexplore.ieee.org/abstract/document/7410521/
- **Learning Relaxed Deep Supervision for Better Edge Detection**
  https://ieeexplore.ieee.org/document/7780401
- **DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detectionv**
  https://ieeexplore.ieee.org/document/7299024/

Summary

- **Edges – what makes an edge?**
- **Gradient-based edge detectors**
  - Using derivatives or gradients to find edges
  - LoG and DoG
- **Canny edge detector**
- **Knowing what are true edges**

Recommended Reading

- [Gonzalez & Woods] Chapter 10
- [Forsyth & Ponce] Chapter 8