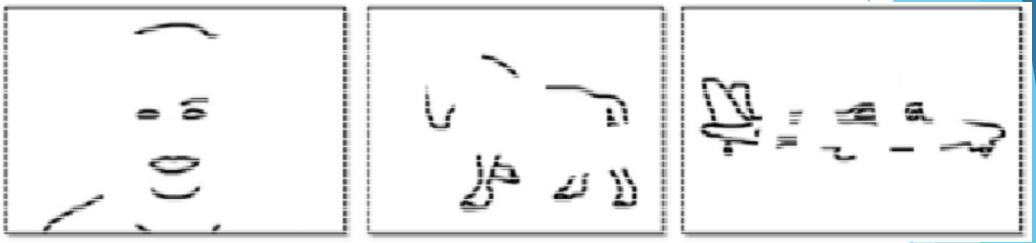
# Vision Systems

Lecture 6

# Part 1

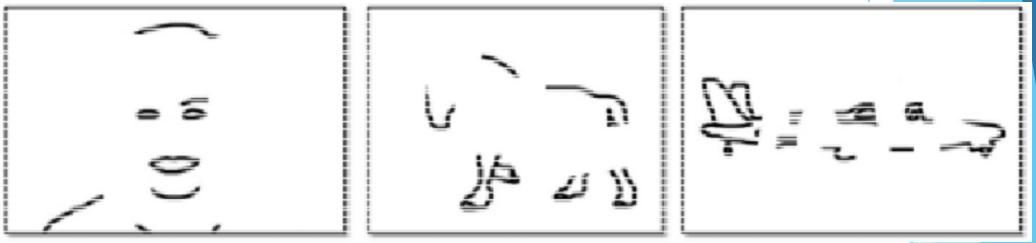
## Edge Detection

# Edge Detection



- Map image from 2D matrix of pixels to a set of curves or line segments or contours $\Rightarrow$ More compact representation than pixels
- **Key idea?**

# Edge Detection

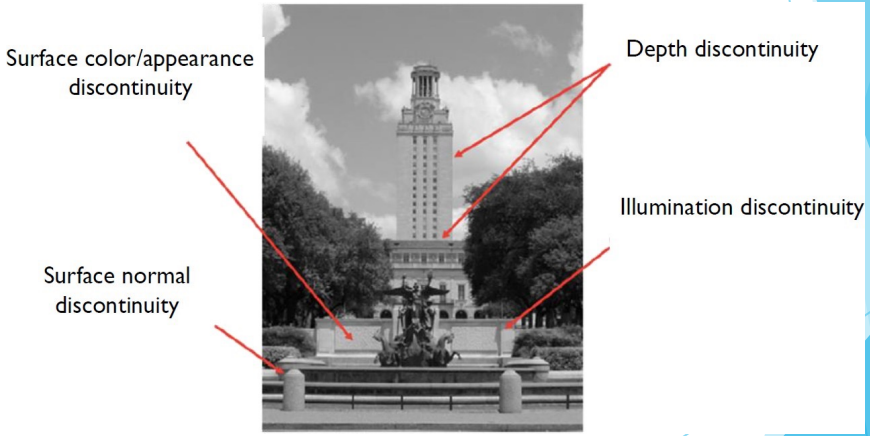


- Map image from 2D matrix of pixels to a set of curves or line segments or contours $\Rightarrow$ More compact representation than pixels
- **Key idea?** Look for strong gradients, and then post-process

*Source: Shotton, K Grauman, R Urtasun*

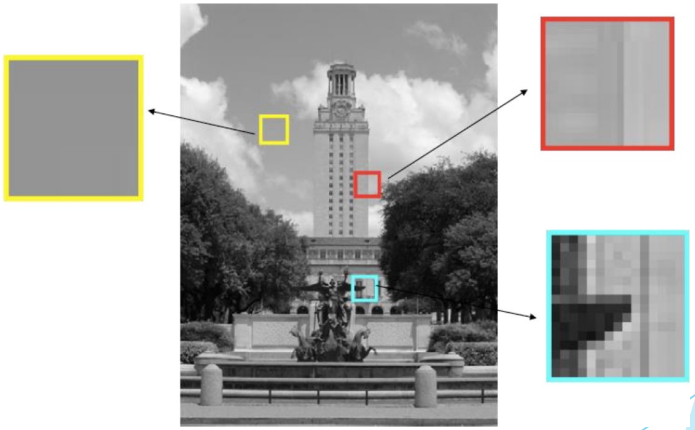
# How are Edges Caused?

- Variety of factors:




*Source: 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: Derek Hoiem*

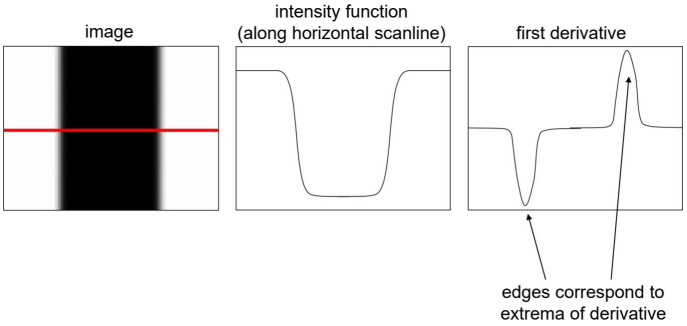
# Edges in Images as Functions

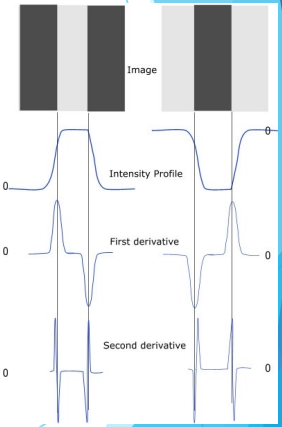
- Edges look like steep cliffs

*Source: N Snavely, R Urtasun*

# Derivatives and Edges

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




*Sources: L Lazebnik, K Grauman and https://mipav.cit.nih.gov/*

# 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:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

# 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

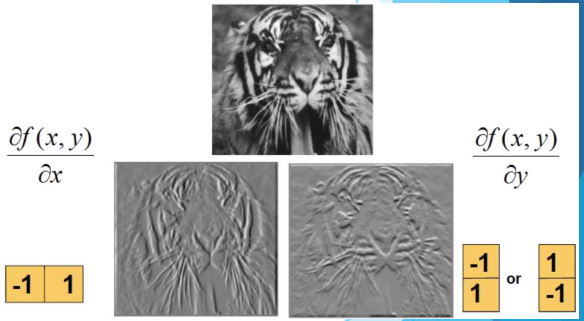
- For 2D function, $f(x, y)$, the partial derivative is:

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

- For discrete data, we can approximate using finite differences:

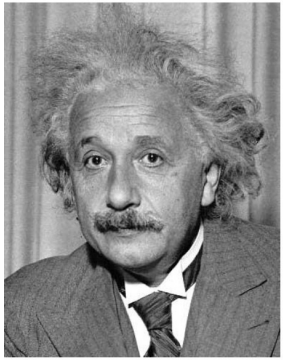
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

- To implement above as convolution, what would be the associated filter?



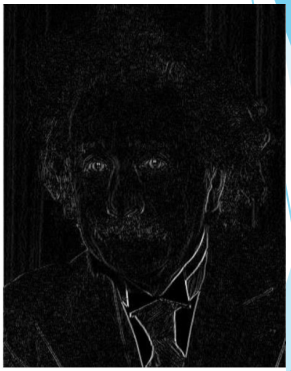

*Source: K Grauman*

# Sobel Edge Detection Filters



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

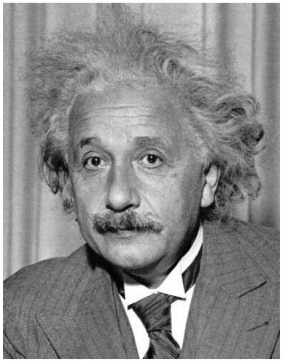
Sobel



Vertical Edge
(absolute value)

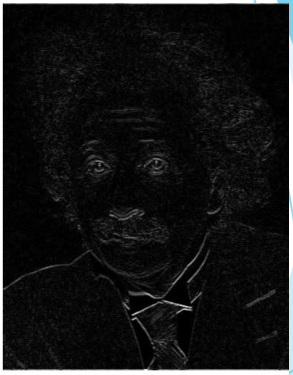
*Source: J Hays*

# Sobel Edge Detection Filters



| | | |
|---|---|---|
| 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 | $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ | $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$ |
| Sobel | $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ | $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ |
| Roberts | $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ | $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ |

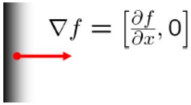
*Source: R Urtasun*

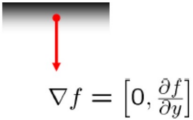
# Image Gradients

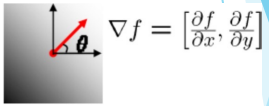
- The gradient of an image $\nabla f = \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$

# 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

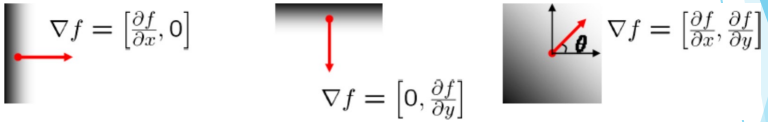


# Image Gradients

- 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

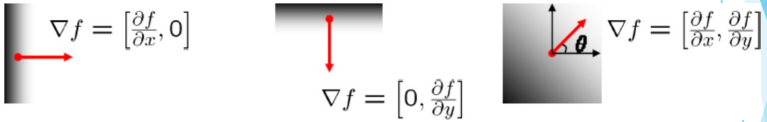



- The **gradient direction** (orientation of edge normal) is given by:

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

# 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:

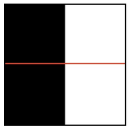
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

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

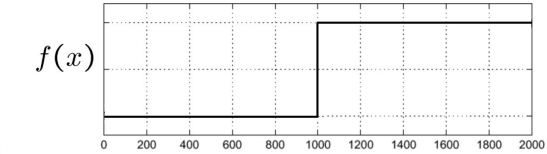
*Source: S Seitz, R Urtasun*

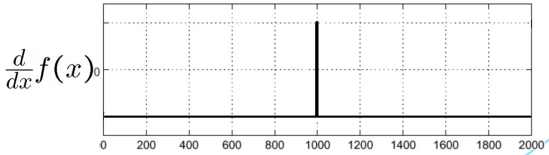
# Derivative with No Noise

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



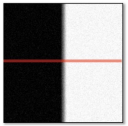
Input image with no noise



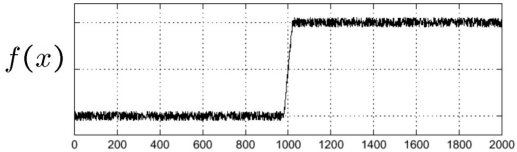



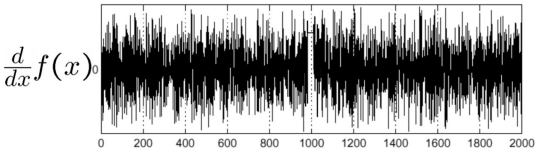

Where is the edge?

# Effect of Noise



Noisy input image



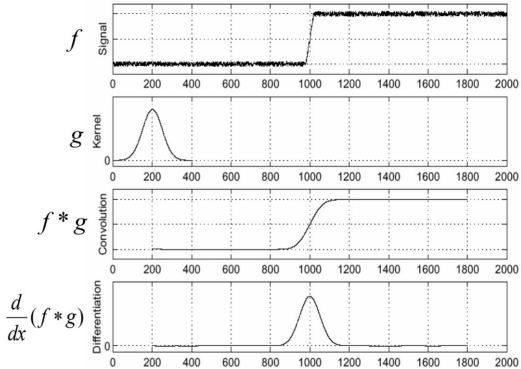




Now, where is the edge?

*Source: S Seitz, K Grauman*

# Effect of Noise

- Smooth first, and look for peaks in $\frac{d}{dx}(f * g)$



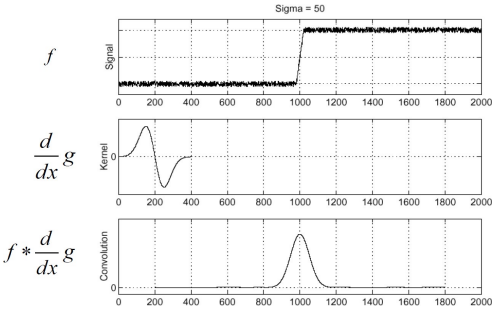

*Source: S Seitz, R Urtasun*

# Derivative theorem of Convolution

- Differentiation is achieved through convolution, and convolution is associative:

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

- This saves us an operation:




*Source: S Seitz, R Urtasun*

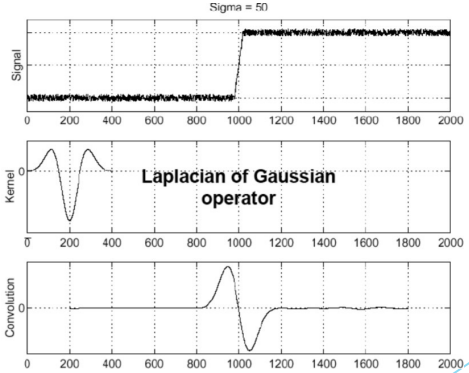
# What about Second Derivative?

- Edge by detecting **zero-crossing** of bottom graph

$f$

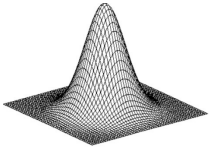
$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$



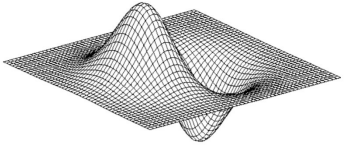

*Source: S Seitz, R Urtasun*

# Derivative and Laplacian of Gaussians



Gaussian

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

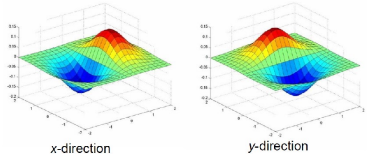


Derivative of Gaussian (x)

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

Laplacian of Gaussian

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



*x*-direction

*y*-direction

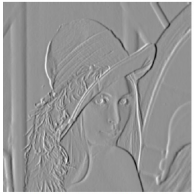
with $\nabla^2$ the Laplacian operator

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

Which one finds horizontal/vertical edges?

Source: S Seitz, R Urtasun

# Compute of Gradients



X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

*Source: S Seitz, R Urtasun*

# Properties of an Edge Detector

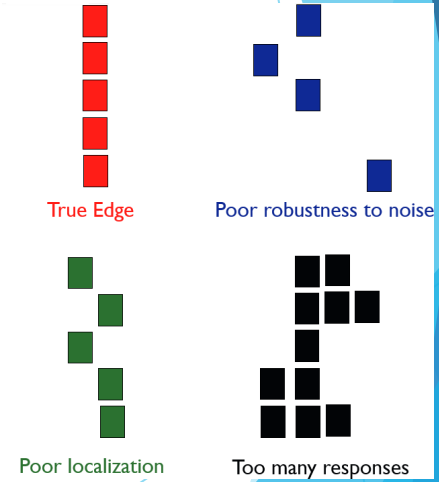
where is the edge?

# Properties of an Edge Detector

- Criteria for a good edge detector?

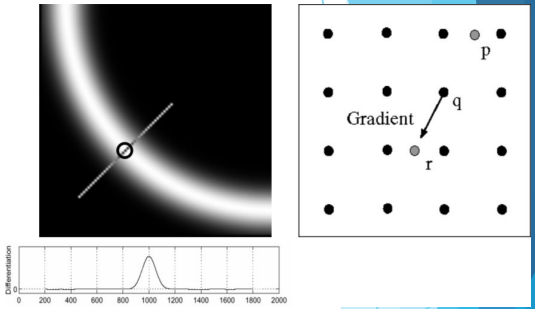
# Properties of an Edge Detector

- 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 true edges
  - **Single response:** return one point only for each true edge point




*Source: K Grauman*

# Non-Maxima Suppression


- Check if pixel is local maximum along gradient direction:
  - Could require checking interpolated pixels *p* and *r*

*Source: N Snavely, R Urtasun*

# Non-Maxima Suppression

Before and after non-maxima suppression

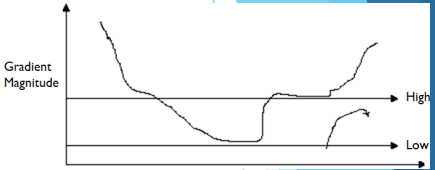
*Source: Derek Hoiem*

# Hysteresis Thresholding

- Check for well-connected edges. How?

# Hysteresis Thresholding

- Check for well-connected edges. How?
  - Use **hysteresis**: use a *high* threshold to start edge curves and a *low* threshold to continue them.
- How does it work?
  - If gradient at pixel > 'High' $\Rightarrow$ **'edge pixel'**
  - If gradient at pixel < 'Low' $\Rightarrow$ **'non-edge pixel'**
  - If gradient at pixel $\geq$ 'Low' and $\leq$ 'High' $\Rightarrow$ **'edge pixel'** iff it is connected to an 'edge pixel' directly or via pixels between 'Low' and 'High'



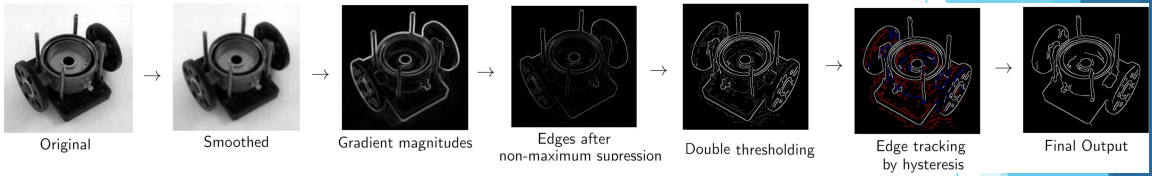

*Source: S Seitz, R Urtasun*

# Canny Edge Detector

- Probably the most widely used edge detector in computer vision (Canny 1986)
- Algorithm:
  1. Filter image with derivative of Gaussian
  2. Find magnitude and orientation of gradient
  3. Non-maximum suppression
  4. Linking and thresholding (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, R Urtasun*

# Canny Edge Pipeline and Examples




Canny Edges

*Source: Prem Kalra, R Urtasun, S Fidler*

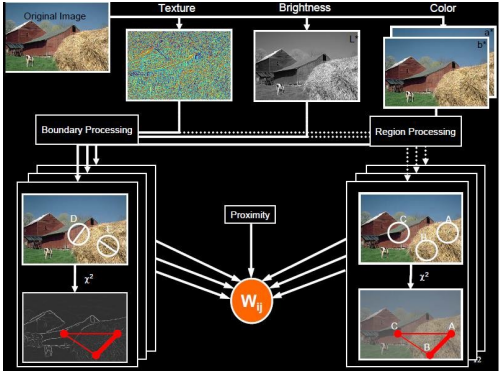
# Effect of $\sigma$ in Canny Edge Detector

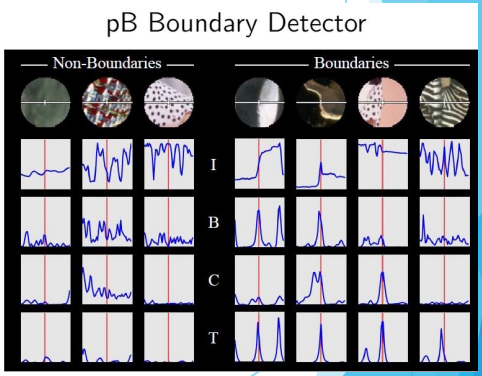
- The choice of $\sigma$ (Gaussian kernel spread/size) depends on desired behavior
  - large $\sigma$ detects large-scale edges
  - small $\sigma$ detects fine edges


*Source: S Seitz, R Urtasun*

# More Recent Methods in Edge Detection

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

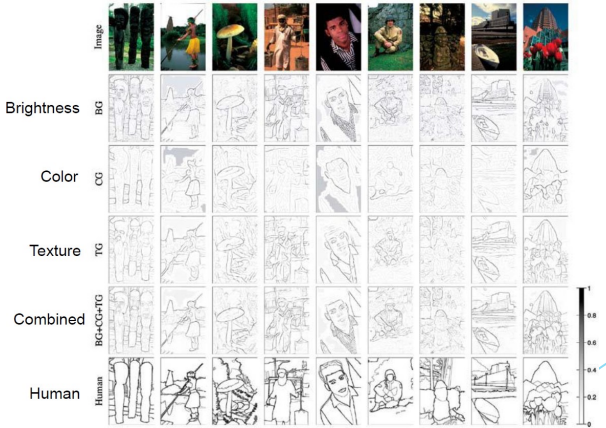



*Source: Derek Hoiem*

# More Recent Methods in Edge Detection

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

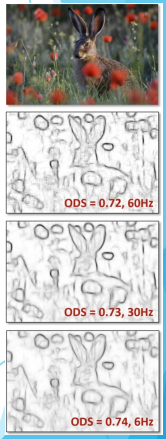



*Source: Derek Hoiem*

# More Recent Methods in Edge Detection

Structured Forests for Fast Edge Detection (Dollár et al, 2013)

- Goal: quickly predict whether each pixel is an edge
- Insights
  - Predictions can be learned from training data
  - Predictions for nearby pixels should not be independent
- Solution
  - Train structured random forests to split data into patches with similar boundaries based on features
  - Predict boundaries at patch level, rather than pixel level, and aggregate (average votes)

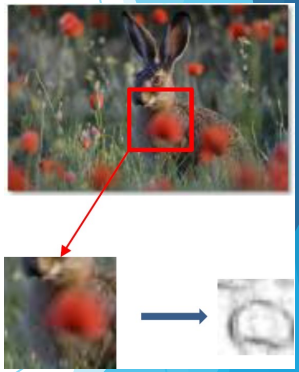



*Source: Derek Hoiem*

# More Recent Methods in Edge Detection

Structured Forests for Fast Edge Detection (Dollár et al, 2013)

- Algorithm
  1. Extract overlapping 32×32 patches at three scales
  2. Features are pixel values and pairwise differences in feature maps (LUV color, gradient magnitude, oriented gradient)
  3. Predict $T$ boundary maps in the central 16×16 region using $T$ trained decision trees
  4. Average predictions for each pixel across all patches

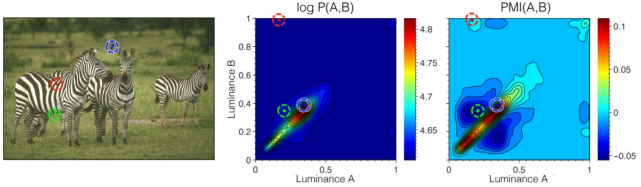


*Source: Derek Hoiem*

# More Recent Methods in Edge Detection

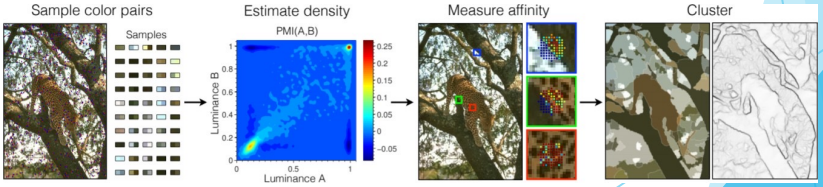
Crisp Boundary Detection using Pointwise Mutual Information (Isola et al, 2014)

- Pixel combinations that are unlikely to be together are edges




$$\mathrm{PMI}_\rho(A,B) = \log \frac{P(A,B)^\rho}{P(A)P(B)}$$

- Algorithm Pipeline:




*Source: Derek Hoiem*

# More Recent Methods in Edge Detection

Crisp Boundary Detection using Pointwise Mutual Information (Isola et al, 2014)

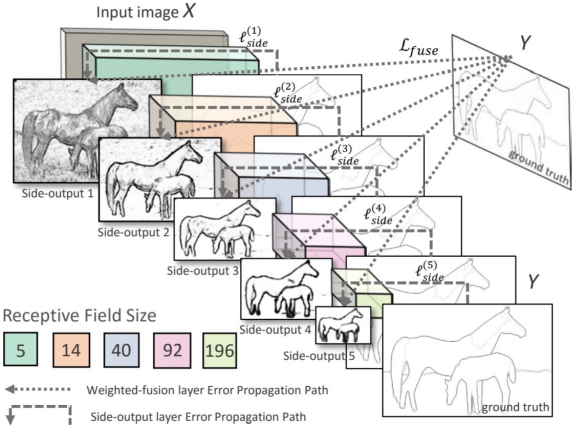

| Algorithm | ODS | OIS | AP |
|---|---|---|---|
| Canny [14] | 0.60 | 0.63 | 0.58 |
| Mean Shift [36] | 0.64 | 0.68 | 0.56 |
| NCuts [37] | 0.64 | 0.68 | 0.45 |
| Felz-Hutt [38] | 0.61 | 0.64 | 0.56 |
| gPb [1] | 0.71 | 0.74 | 0.65 |
| gPb-owt-ucm [1] | 0.73 | 0.76 | 0.73 |
| SCG [9] | **0.74** | 0.76 | 0.77 |
| Sketch Tokens [7] | 0.73 | 0.75 | 0.78 |
| SE [8] | **0.74** | 0.76 | 0.78 |
| Our method – SS, color only | 0.72 | 0.75 | 0.77 |
| Our method – SS | 0.73 | 0.76 | **0.79** |
| Our method – MS | **0.74** | **0.77** | 0.78 |

Evaluation on BSDS500

*Source: Derek Hoiem*

# More Recent Methods in Edge Detection

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

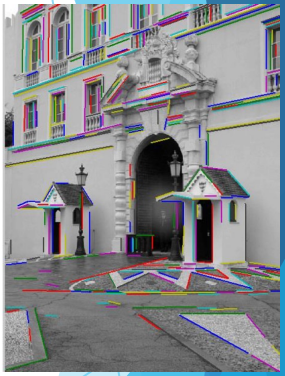



| | ODS | OIS | AP | FPS |
|---|---|---|---|---|
| Human | .80 | .80 | - | - |
| Canny | .600 | .640 | .580 | 15 |
| Felz-Hutt [9] | .610 | .640 | .560 | 10 |
| BEL [5] | .660* | - | - | 1/10 |
| gPb-owt-ucm [1] | .726 | .757 | .696 | 1/240 |
| Sketch Tokens [24] | .727 | .746 | .780 | 1 |
| SCG [31] | .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*

# Part A: Homework

- Readings
  - Chapter 2, Szeliski, *Computer Vision: Algorithms and*
- Questions
  - How do you go from Canny edges to straight lines? *(*



*Source: Derek Hoiem*

# Part 2

## From Edges to Blobs and Corners

# Review: Using Canny Edges to get Straight Lines

- Compute Canny edges
  - Compute $\nabla_x f$, $\nabla_y f$ (gradients in $x$, $y$ directions)
  - Compute $\vartheta = tan^{-1} \frac{\nabla_y f}{\nabla_x f}$
- Assign each edge to one of 8 directions. For each direction $d$, obtain *"edgelets"*:
  - Find connected components for edge pixels with directions in $\{d - 1, d, d + 1\}$

# Review: Using Canny Edges to get Straight Lines

- Compute Canny edges
  - Compute $\nabla_x f$, $\nabla_y f$ (gradients in $x$, $y$ directions)
  - Compute $\vartheta = tan^{-1} \frac{\nabla_y f}{\nabla_x f}$
- Assign each edge to one of 8 directions. For each direction $d$, obtain *"edgelets"*:
  - Find connected components for edge pixels with directions in $\{d-1, d, d+1\}$
- Compute straightness and orientation, $\vartheta$ of edgelets using eigenvectors and eigenvalues of second moment matrix, $M$, of edge pixels

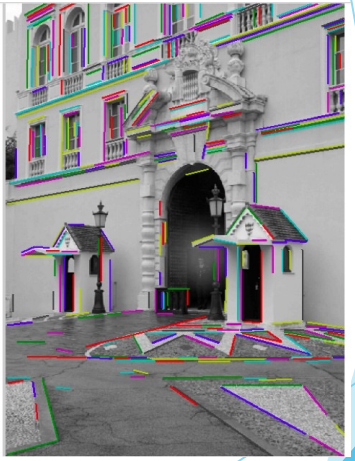
$$M = \begin{matrix} \sum (x-\mu_x)^2 & \sum (x-\mu_x)(y-\mu_y) \\ \sum (x-\mu_x)(y-\mu_y) & \sum (y-\mu_y)^2 \end{matrix} \qquad [\mathbf{v}, \lambda] = eig(M)$$

$\vartheta = tan^{-1}(\mathbf{v}_1/\mathbf{v}_0)$ where $\mathbf{v}_1$ is the 'larger' eigenvector; straightness $= \lambda_2/\lambda_1$

- Threshold straightness appropriately and store line segments

*Credit: Derek Hoiem*

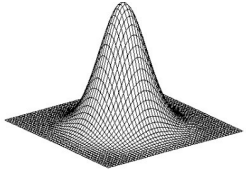
# Review: Using Canny Edges to get Straight Lines



*Credit: Derek Hoiem*

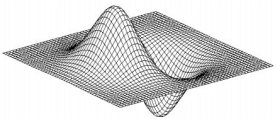
# Going Further to a Second Derivative

- What if we took the Laplacian of Gaussian?



Gaussian

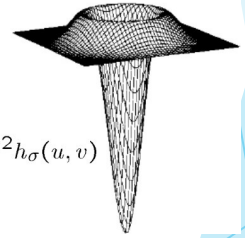
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



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

Laplacian $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 y}{\partial y^2}$ (1)

*Credit: S Seitz, K Grauman*

# Laplacian of Gaussian

Example of a $3 \times 3$ Laplacian of Gaussian filter:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

# Laplacian of Gaussian

Original Image

Laplacian

Laplacian of Gaussian

# Laplacian of Gaussian

Original Image

Laplacian

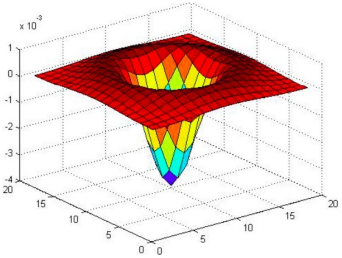
Laplacian of Gaussian

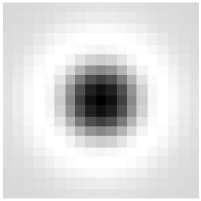
What else can LoG do?

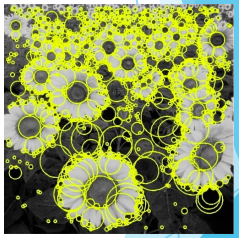
*Credit: K Grauman*

# LoG as a Blob Detector

- Recall that convolution with a filter can be viewed as comparing a little ”picture” of what you want to find against all local regions in the image.





- Observing the LoG filter matrix reveals that it is circularly symmetric. Thus it can be used for blob detection!

*Credit: S Lazebnik*

# From Blobs to Corners

- In the following image, what are some interesting features to choose?

*Credit: K Grauman, R Urtasun*

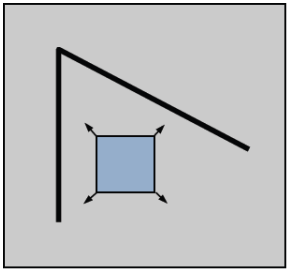
# From Blobs to Corners

- Look for image regions that are unusual. How to define ”unusual”?
- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the **aperture problem** (we’ll see next slide), i.e., it is only possible to align the patches along the direction normal to the edge direction.
- Gradients in at least two (significantly) different orientations are the easiest, e.g., corners.

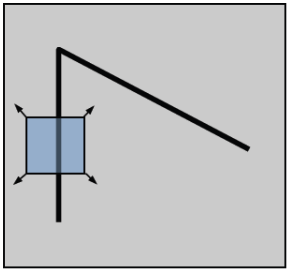
*Credit: R Urtasun*

# From Blobs to Corners

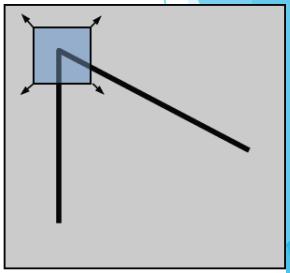
- Consider a small window of pixels. How does the window change when you shift it?



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

*Credit: S Seitz, D Frolova, D Simakova, R Urtasun*

# Autocorrelation

- In the previous slide, how to quantify the ”significant” change of the window?

# Part B: Homework

Reaadings

Chapter 2, Szeliski, *Computer Vision: Algorithms and Applications*

Questions: Linear Algebra review

Show that the trace of a matrix is the sum of its eigenvalues.

Show that the determinant of a matrix is the product of its eigenvalues.

# References


- John F. Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8 (1986), pp. 679–698.
- David Martin, Charless Fowlkes, and Jitendra Malik. “Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (June 2004), pp. 530–49.
- Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.
- Piotr Doll´ar and Lawrence Zitnick. “Structured Forests for Fast Edge Detection”. In: *Proceedings of the International Conference on Computer Vision*. IEEE, Dec. 2013.
- Phillip Isola et al. “Crisp Boundary Detection Using Pointwise Mutual Information”. In: *Proceedings of the European Conference on Computer Vision*. 2014.
- Saining Xie and Zhuowen Tu. “Holistically-Nested Edge Detection”. In: *International Journal of Computer Vision* 125 (2015), pp. 3–18.

# References


- C. Harris and M. Stephens. “A Combined Corner and Edge Detector”. In: *Proceedings of the 4th Alvey Vision Conference*. 1988, pp. 147–151.
- Bill Triggs. “Detecting Keypoints with Stable Position, Orientation, and Scale under Illumination Changes”. In: *Computer Vision - ECCV 2004*. Ed. by Tom´as Pajdla and Ji˘r´ıMatas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 100–113.
- M. Brown, R. Szeliski, and S. Winder. “Multi-image matching using multi-scale oriented patches”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 510–517 vol. 1.
- Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.
- David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. 2 edition. Boston: Pearson Education India, 2015.