

BBM 413

**Fundamentals of
Image Processing**

Nov. 27, 2012

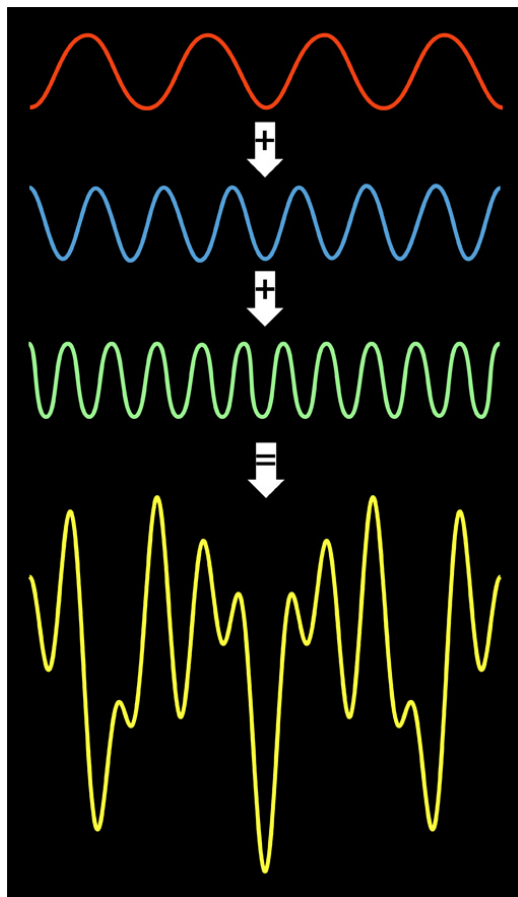
Erkut Erdem

Dept. of Computer Engineering
Hacettepe University

Edge Detection
Hough Transform

Review – Signals and Images

- A signal is composed of low and high frequency components



low frequency components: smooth /
piecewise smooth

Neighboring pixels have similar brightness values

You're within a region

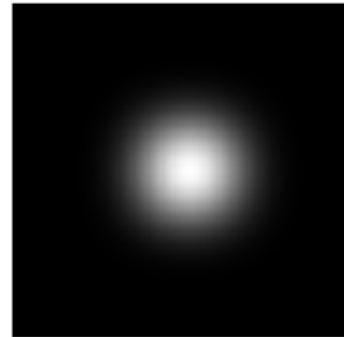
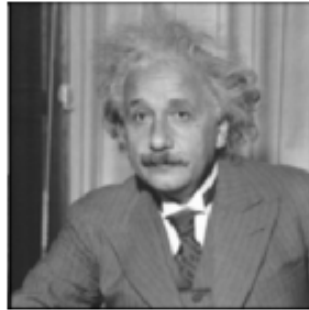
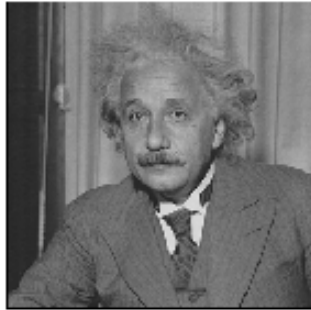
high frequency components: oscillatory

Neighboring pixels have different brightness values

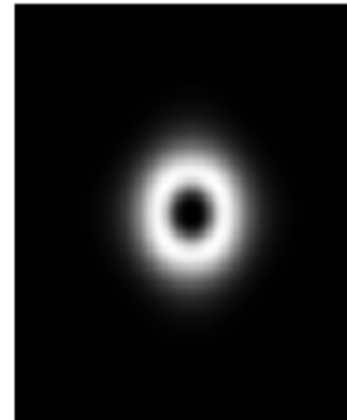
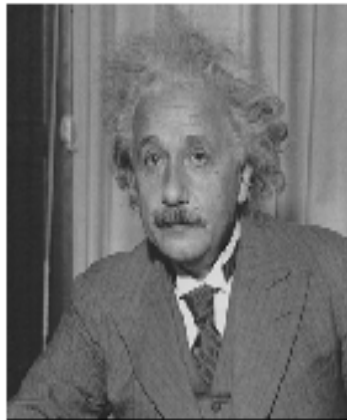
You're either at the edges or noise points

Review - Low-pass, Band-pass, High-pass filters

low-pass:



High-pass / band-pass:



Edge detection

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

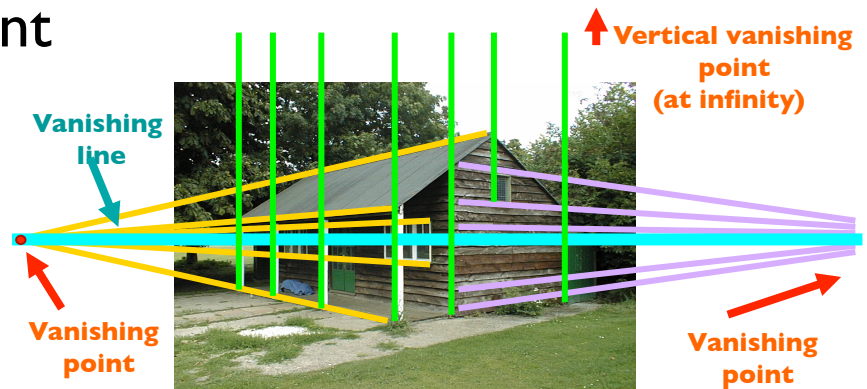


Why do we care about edges?

- Extract information, recognize objects



- Recover geometry and viewpoint

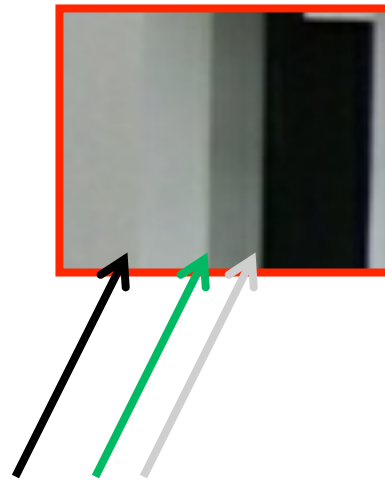


Closeup of edges



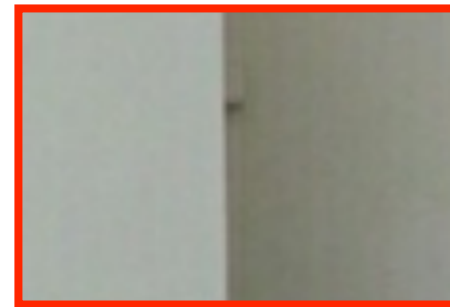
Slide credit: D. Hoiem

Closeup of edges



Slide credit: D. Hoiem

Closeup of edges



Slide credit: D. Hoiem

Closeup of edges

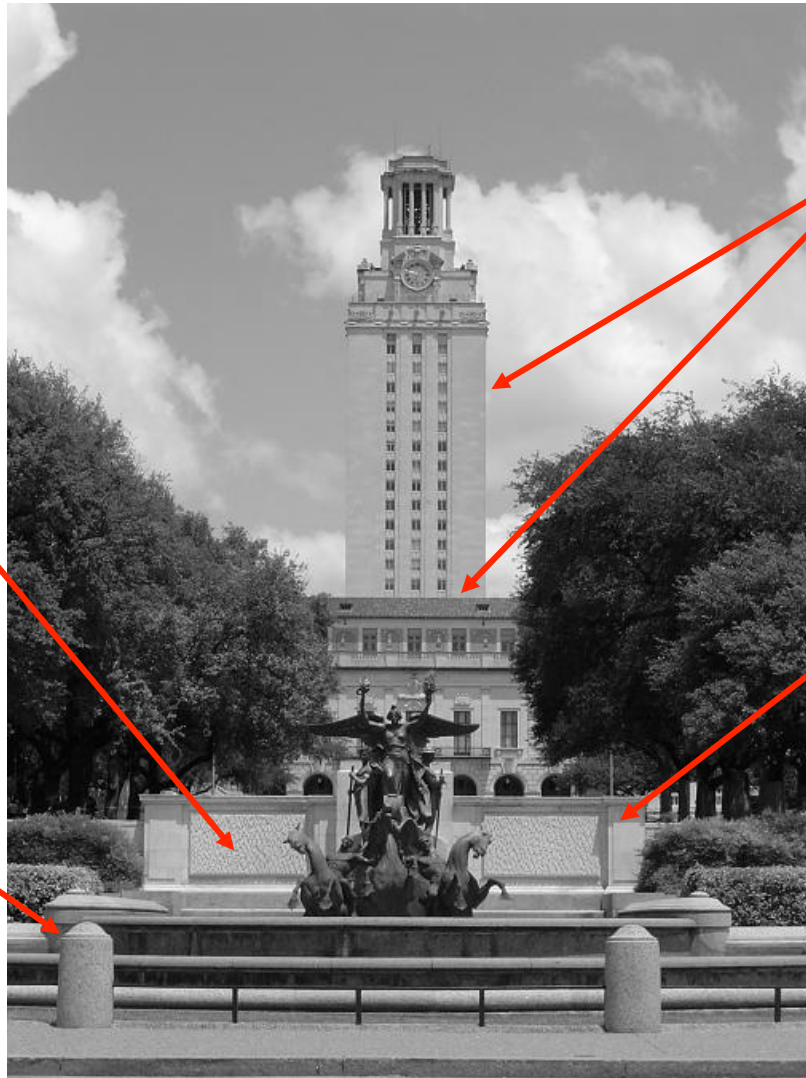


Slide credit: D. Hoiem

What causes an edge?

Reflectance change:
appearance
information, texture

Change in surface
orientation: shape

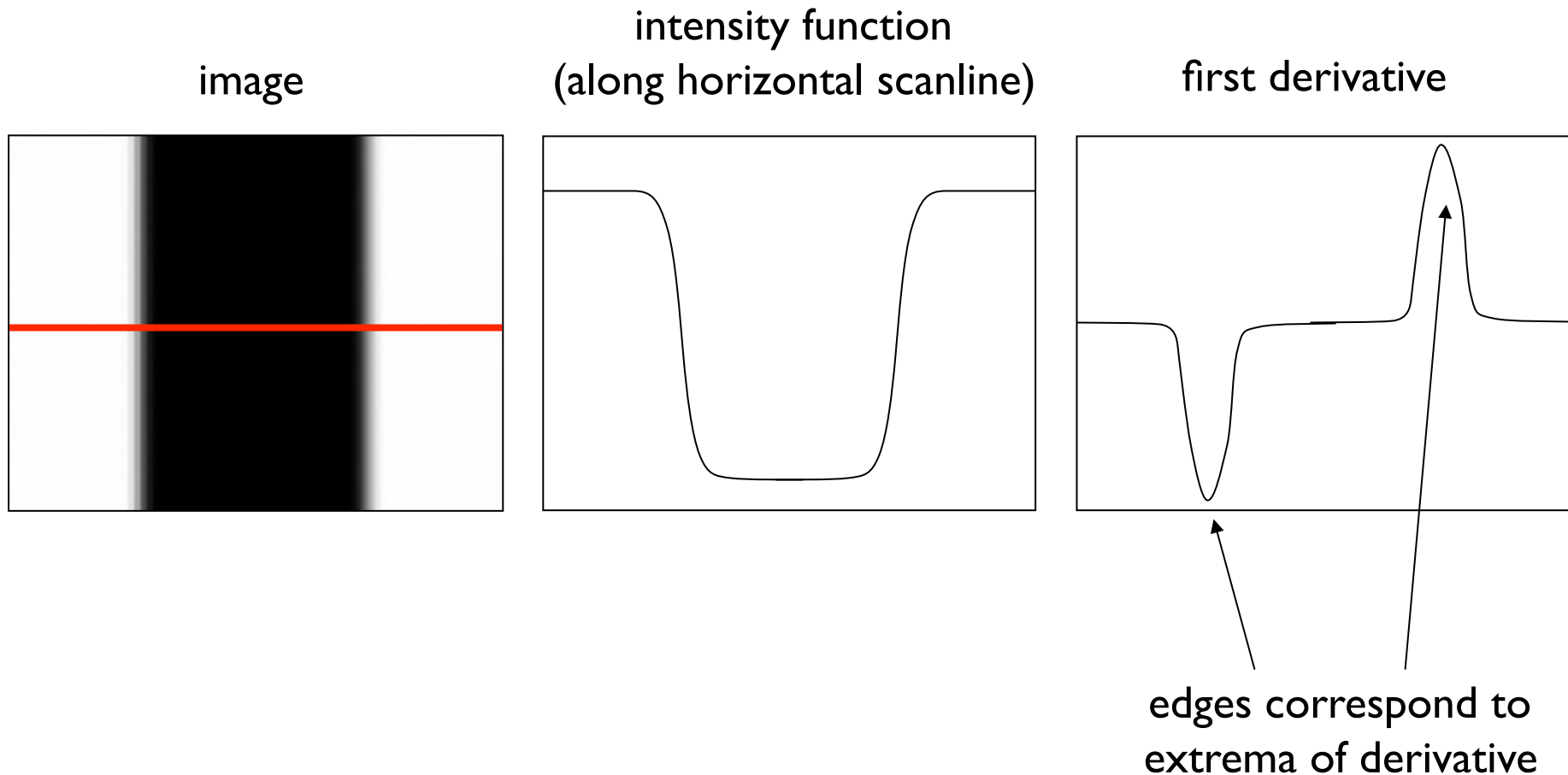


Depth discontinuity:
object boundary

Cast shadows

Characterizing 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:

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

For discrete data, we can approximate using finite differences:

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

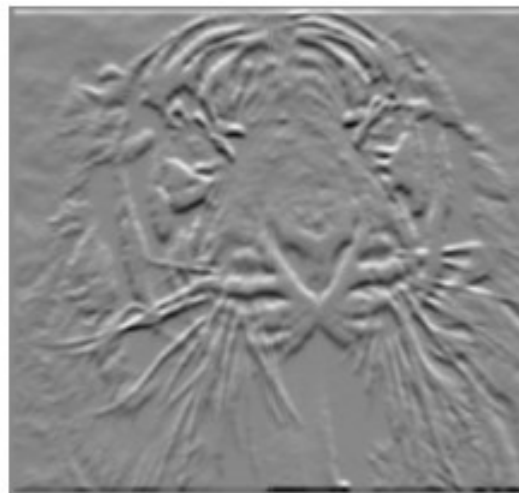
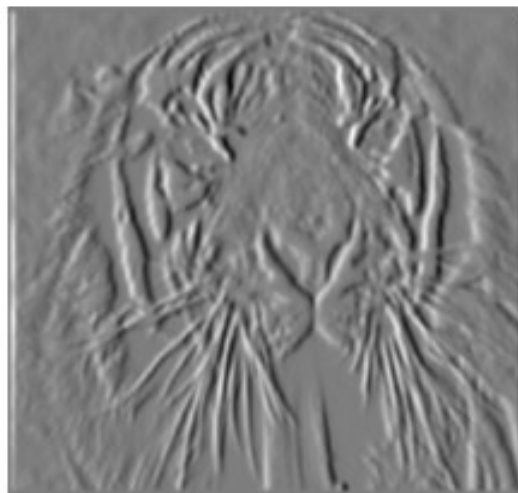
To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

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



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



-1	1
----	---

-1
1

Which shows changes with respect to x?

Assorted finite difference filters

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

```
>> My = fspecial('sobel');  
>> outim = imfilter(double(im), My);  
>> imagesc(outim);  
>> colormap gray;
```

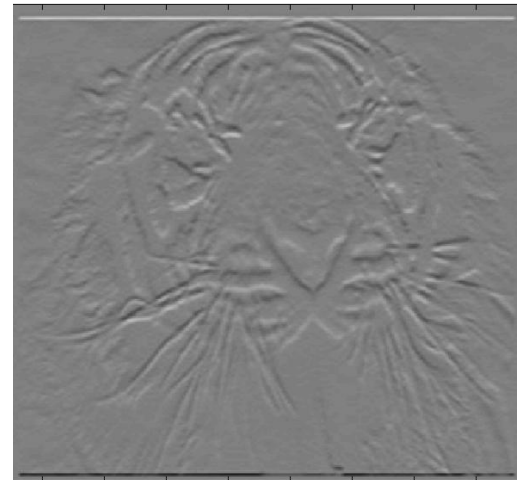
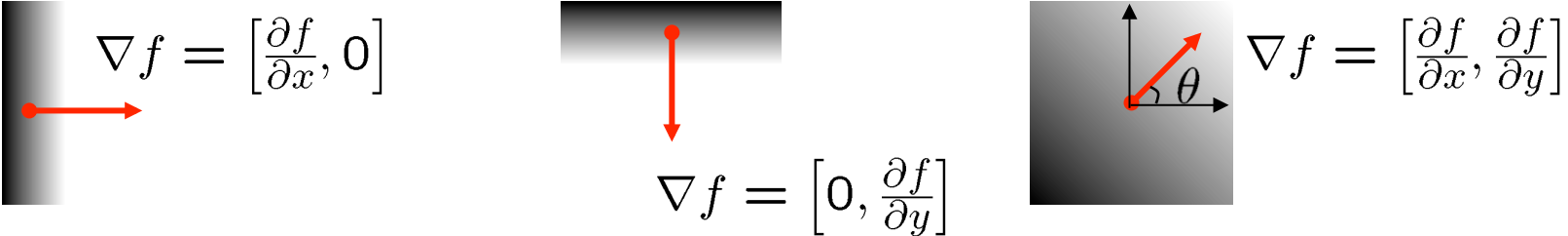


Image gradient

- 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 increase in intensity

- How does this direction relate to the direction of the edge?

The gradient direction 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 gradient magnitude

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

Original Image



Slide credit: K. Grauman

Gradient magnitude image



Slide credit: K. Grauman

Thresholding gradient with a lower threshold

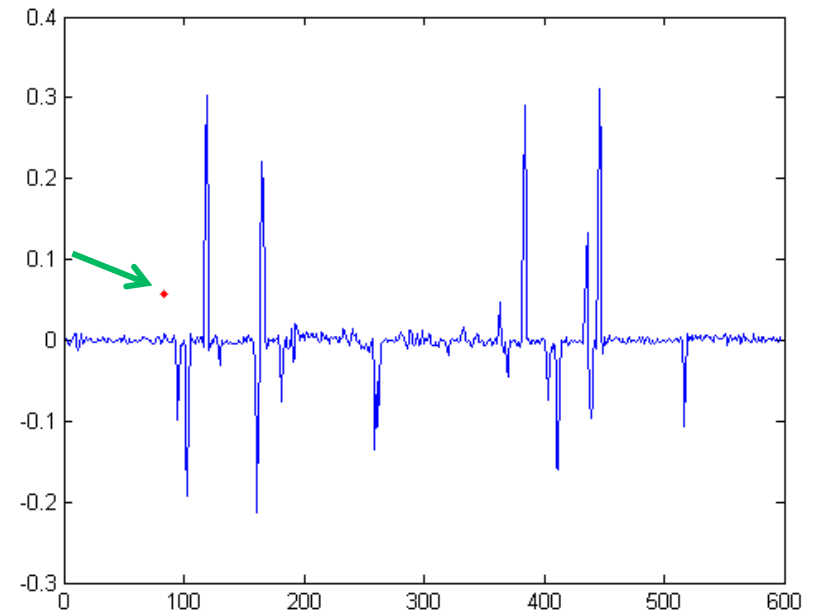
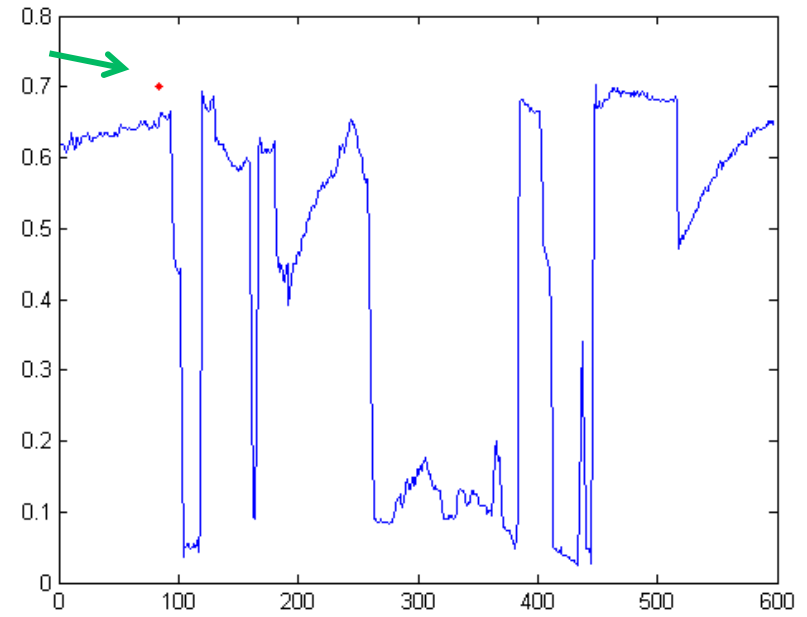
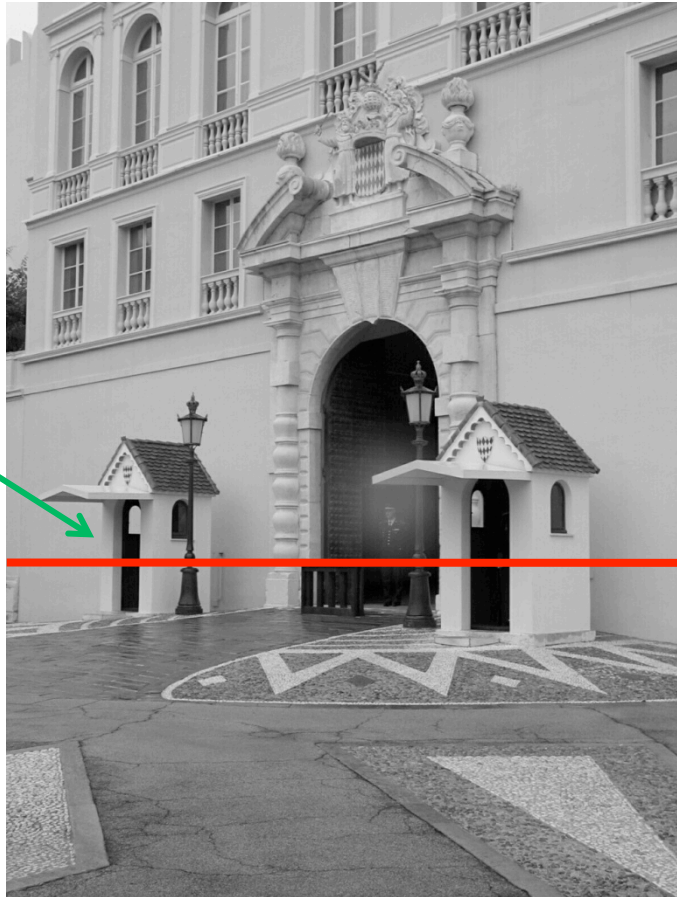


Thresholding gradient with a higher threshold

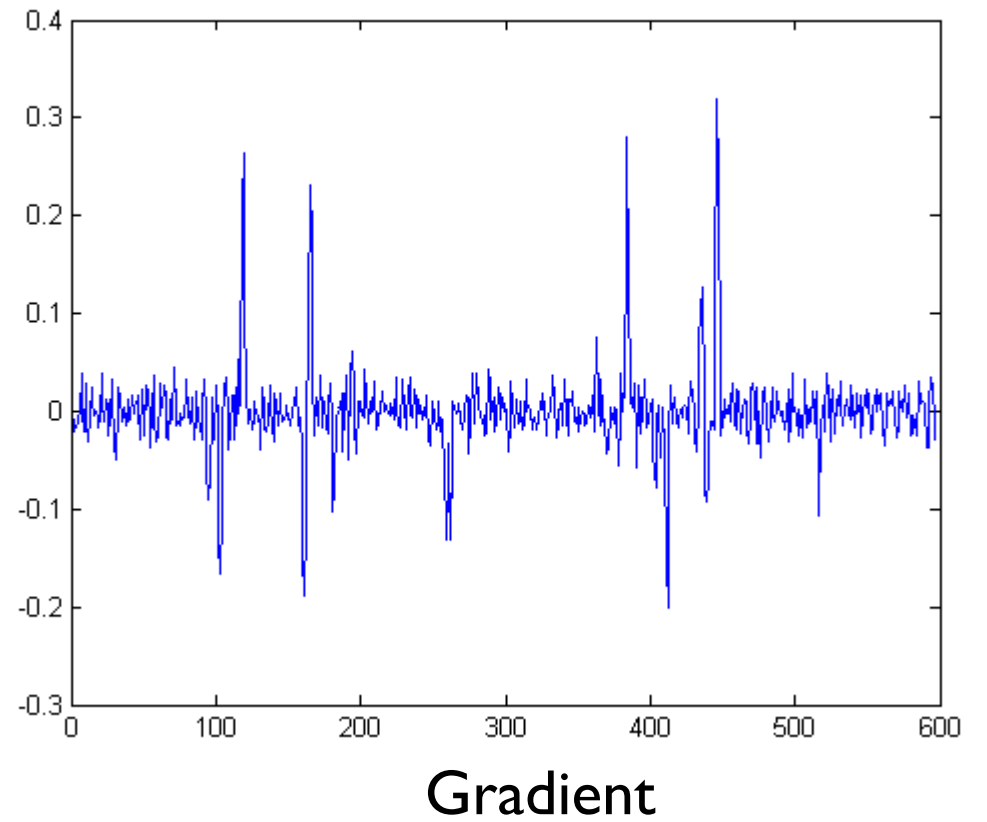
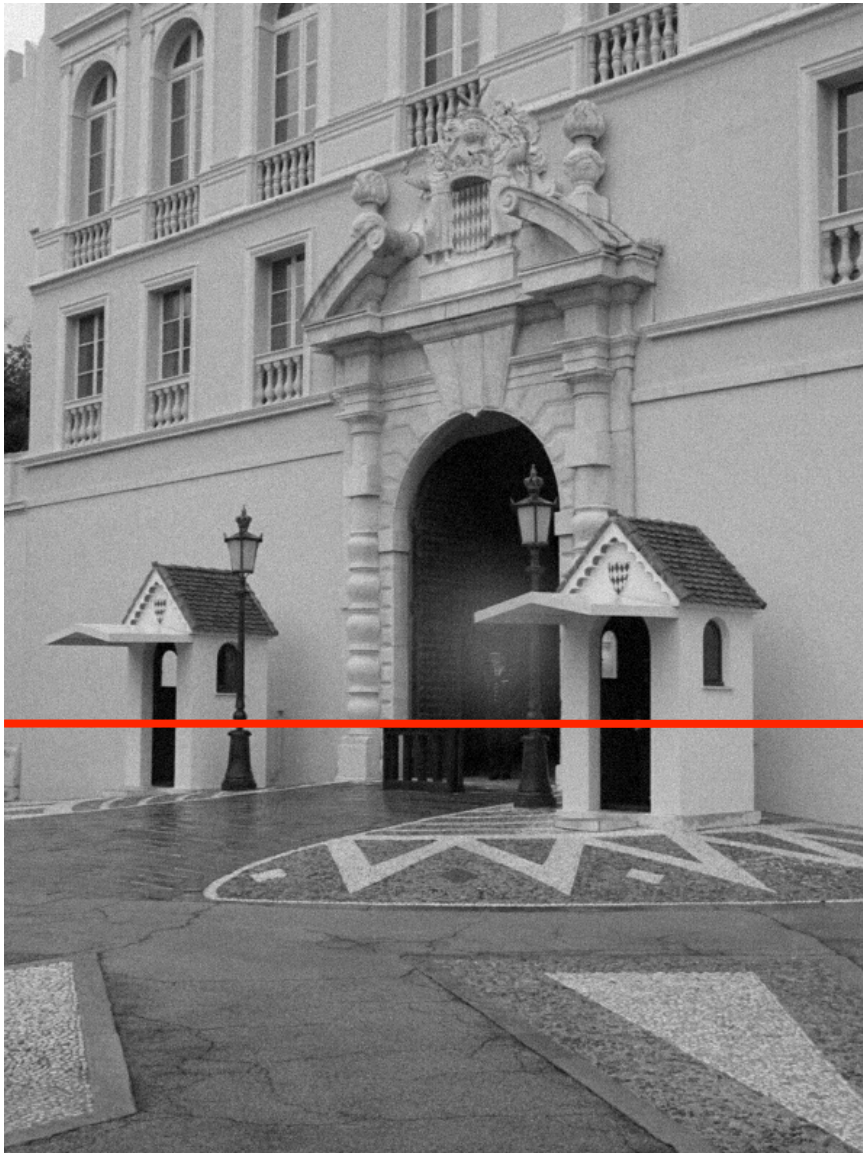


Slide credit: K. Grauman

Intensity profile



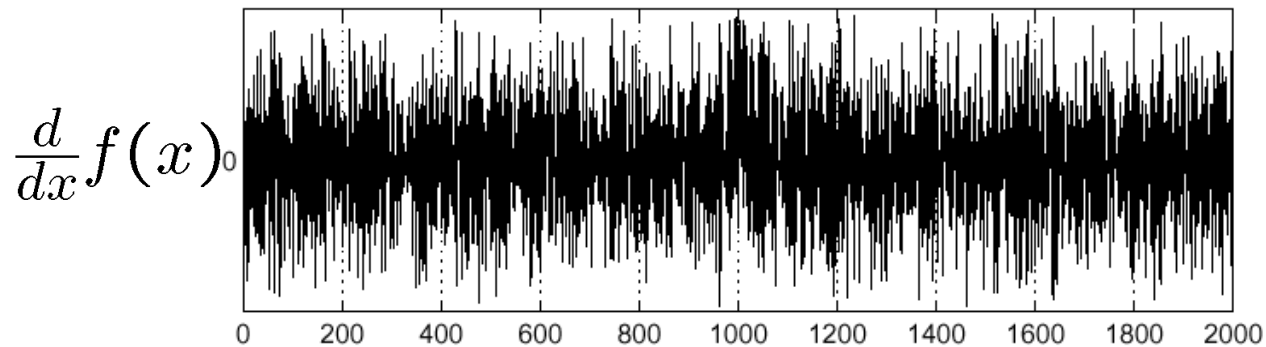
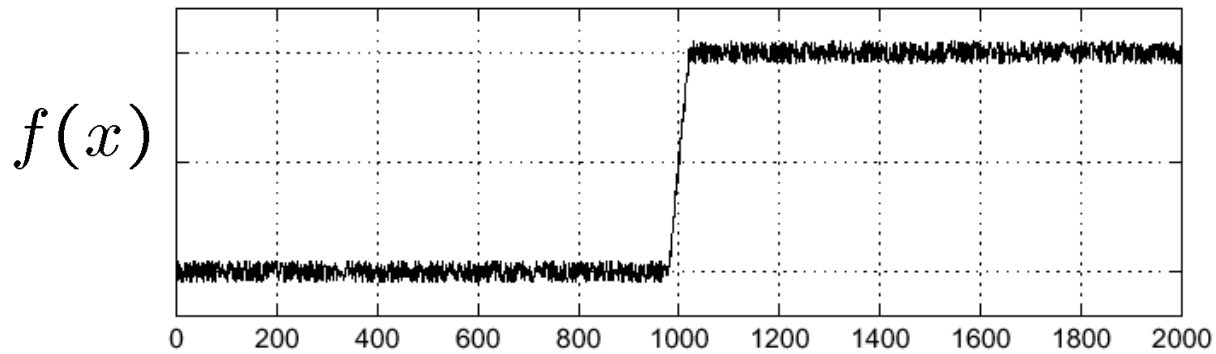
With a little Gaussian noise



Slide credit: D. Hoiem

Effects of noise

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

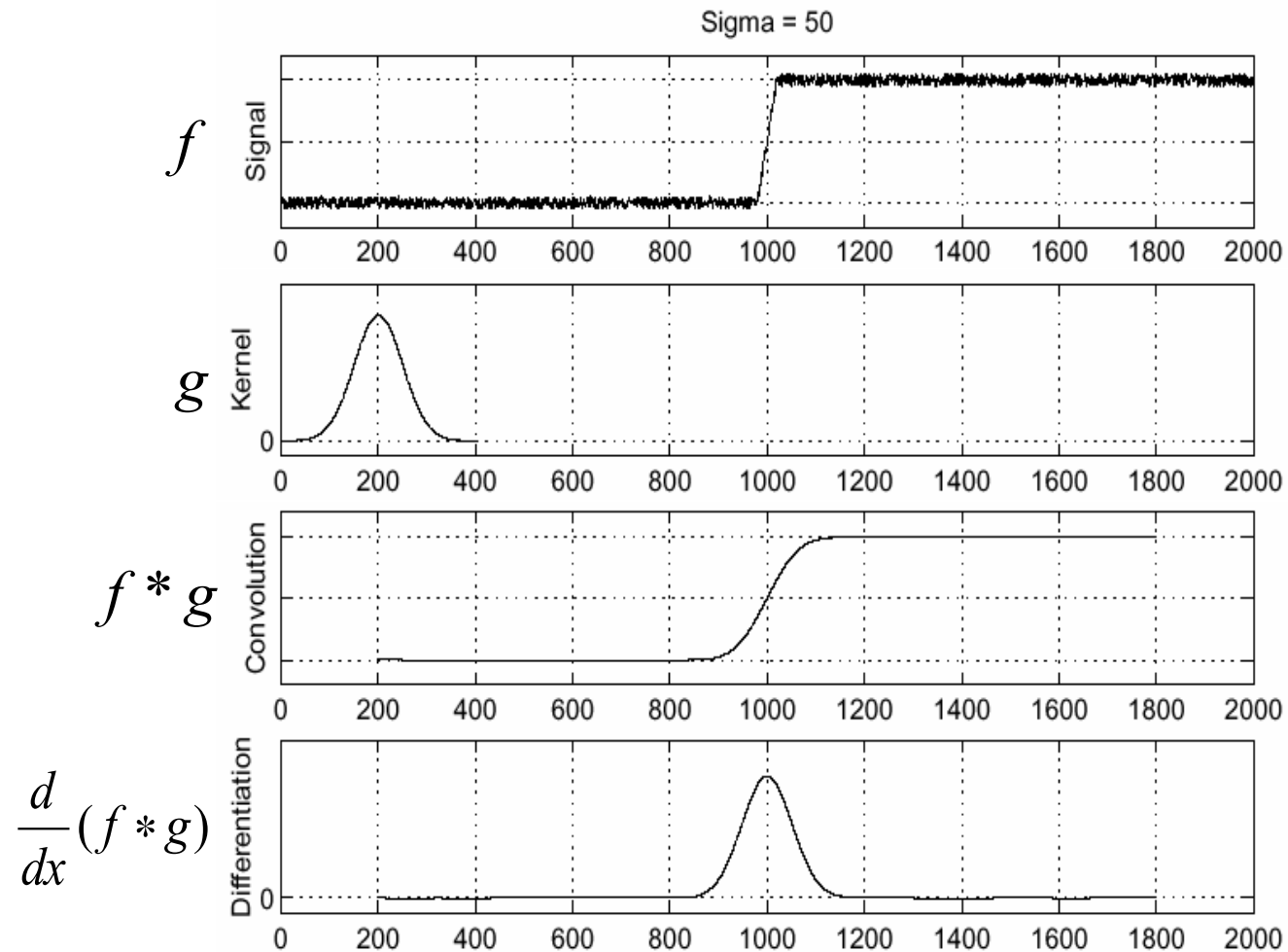


Where is the edge?

Effects of noise

- Difference filters respond strongly to noise
 - Image noise results in pixels that look very different from their neighbors
 - Generally, the larger the noise the stronger the response
- What can we do about it?

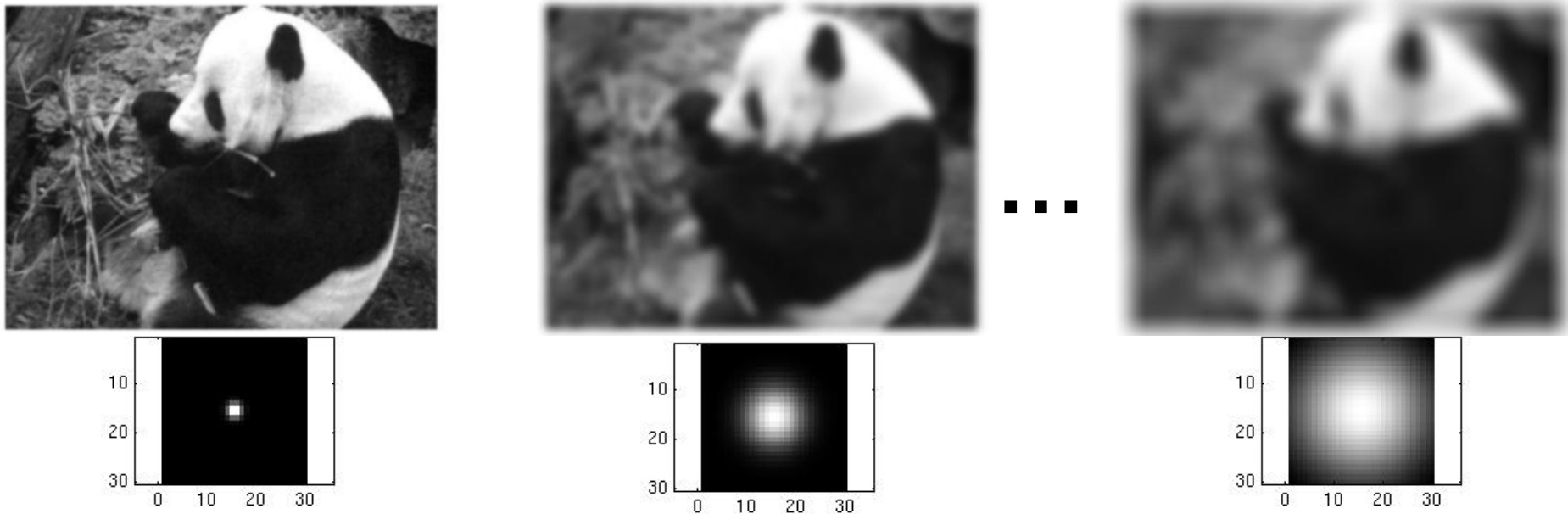
Solution: smooth first



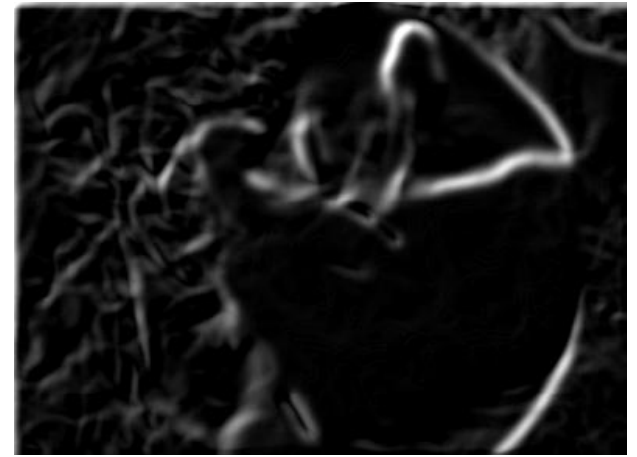
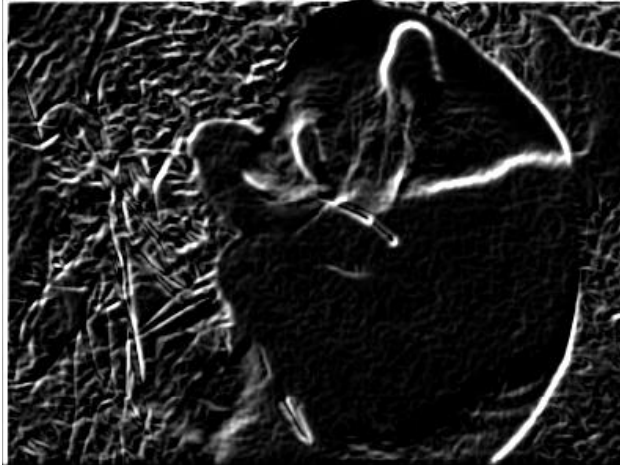
- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Smoothing with a Gaussian

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Effect of σ on derivatives



$\sigma = 1$ pixel

$\sigma = 3$ pixels

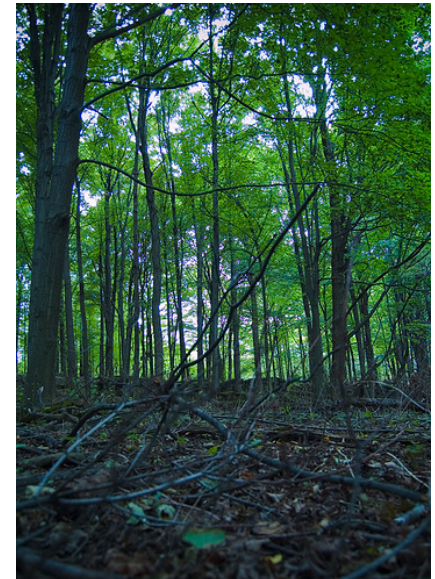
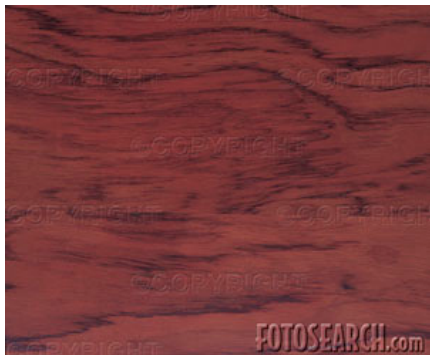
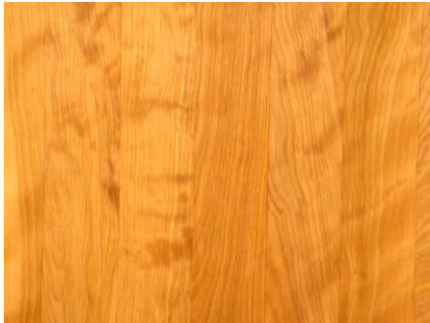
The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected

Smaller values: finer features detected

So, what scale to choose?

It depends what we're looking for.



Slide credit: K. Grauman

Smoothing and Edge Detection

- While eliminating noise via smoothing, we also lose some of the (important) image details.
 - Fine details
 - Image edges
 - etc.
- What can we do to preserve such details?
 - Use edge information during denoising!
 - This requires a definition for image edges.

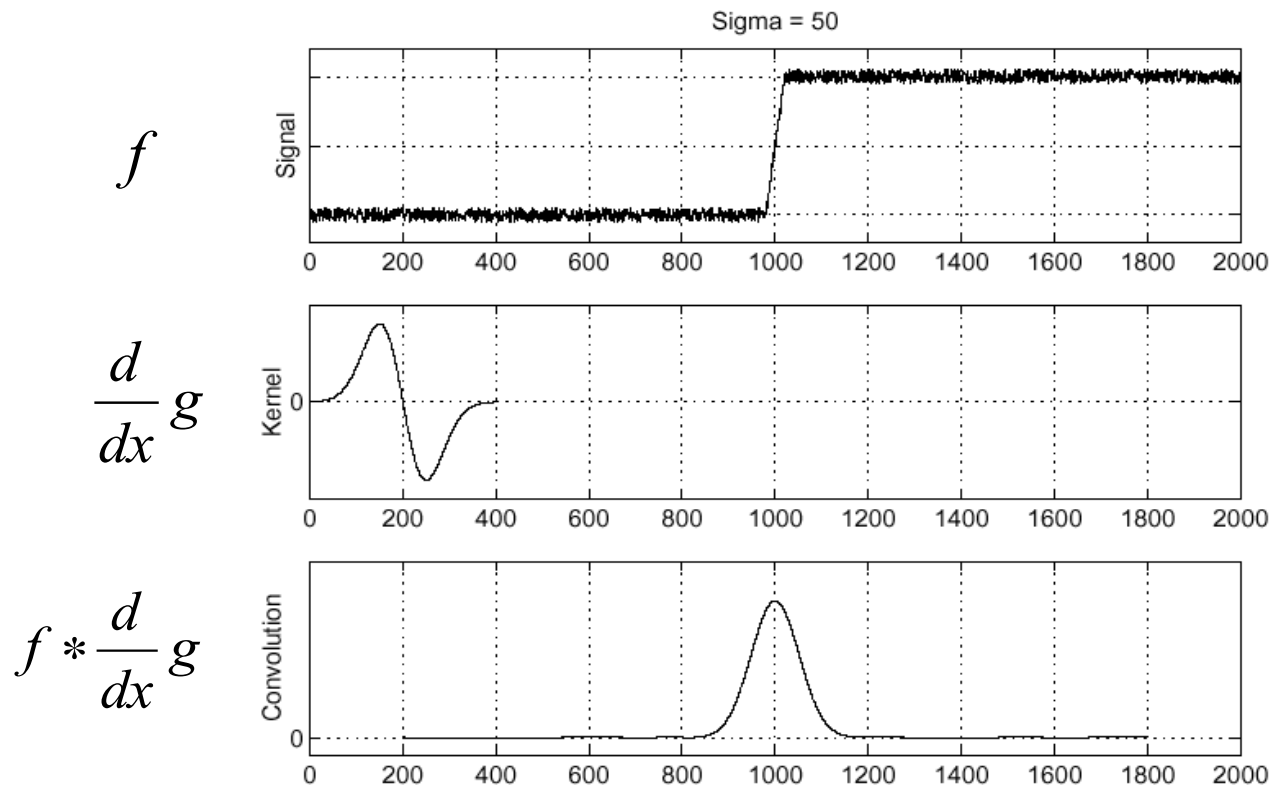
Chicken-and-egg dilemma!

- Edge preserving image smoothing (Next week's topic!)

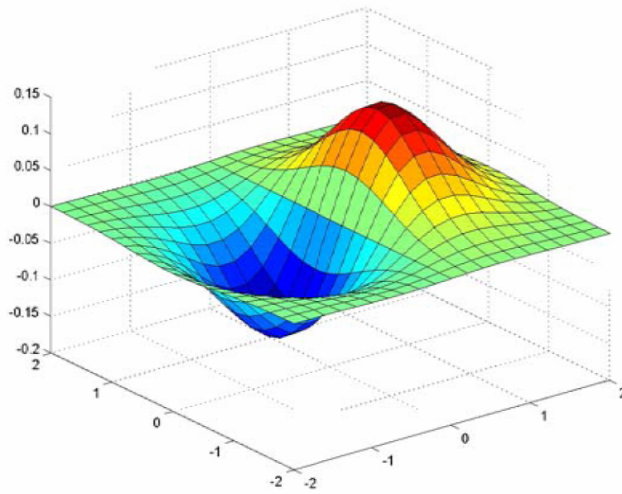
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

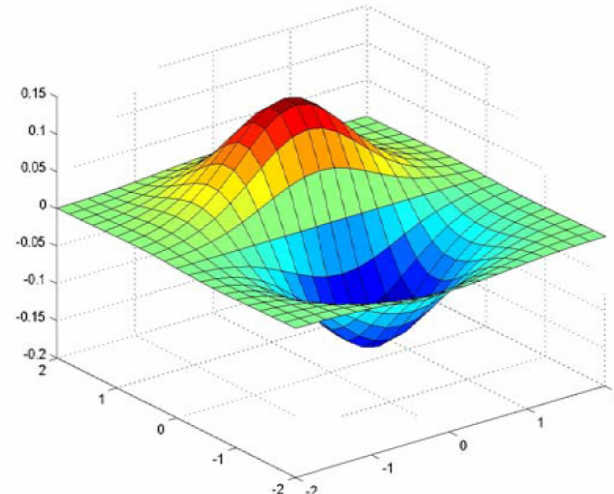
- This saves us one operation:
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$



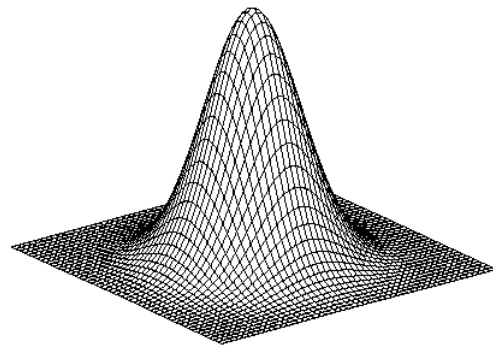
Derivative of Gaussian filter



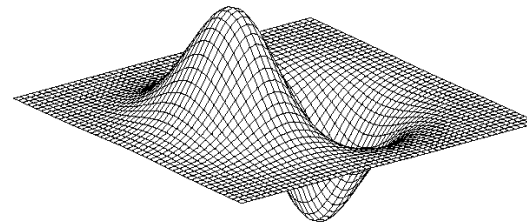
x-direction



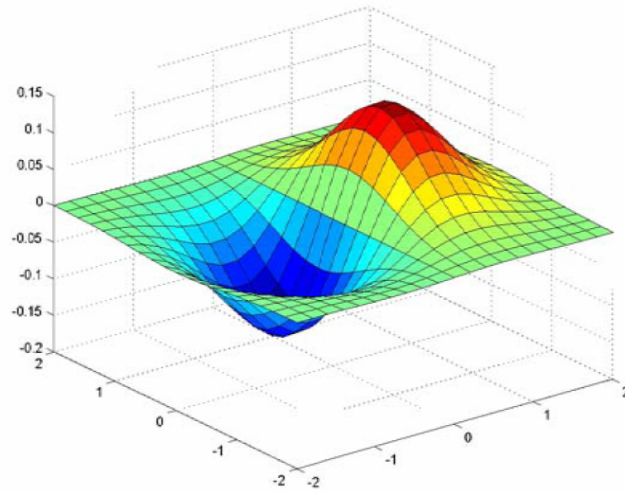
y-direction



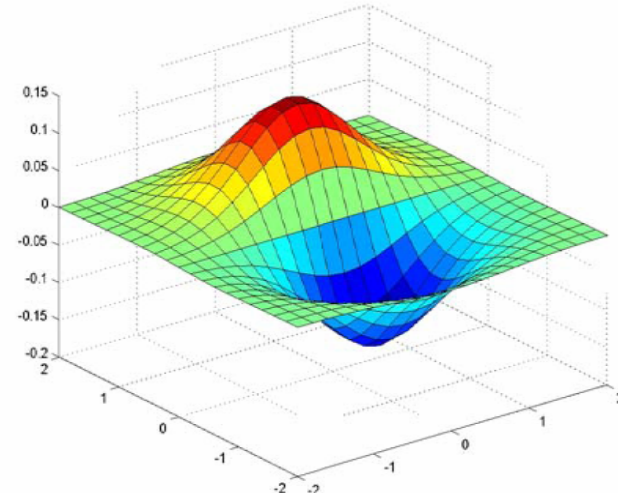
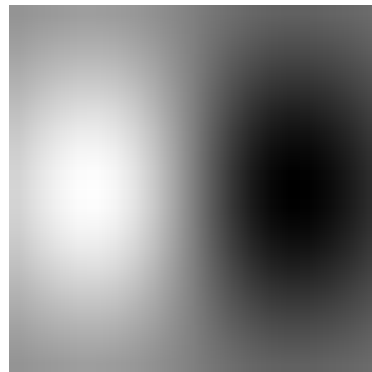
$$* [1 \ -1] =$$



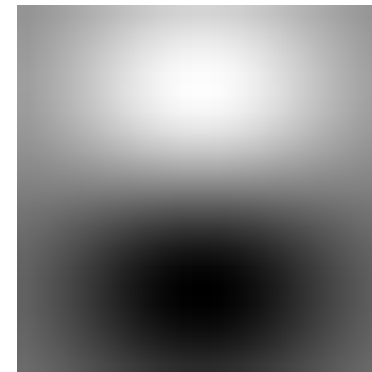
Derivative of Gaussian filter



x-direction



y-direction

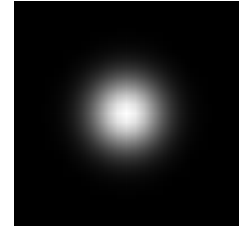


- Which one finds horizontal/vertical edges?

Smoothing vs. derivative filters

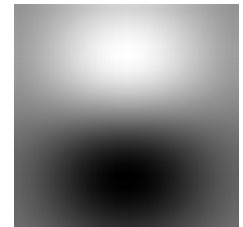
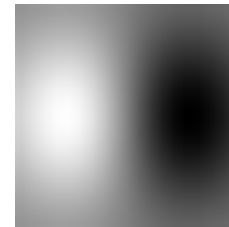
- Smoothing filters

- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
 - **One:** constant regions are not affected by the filter



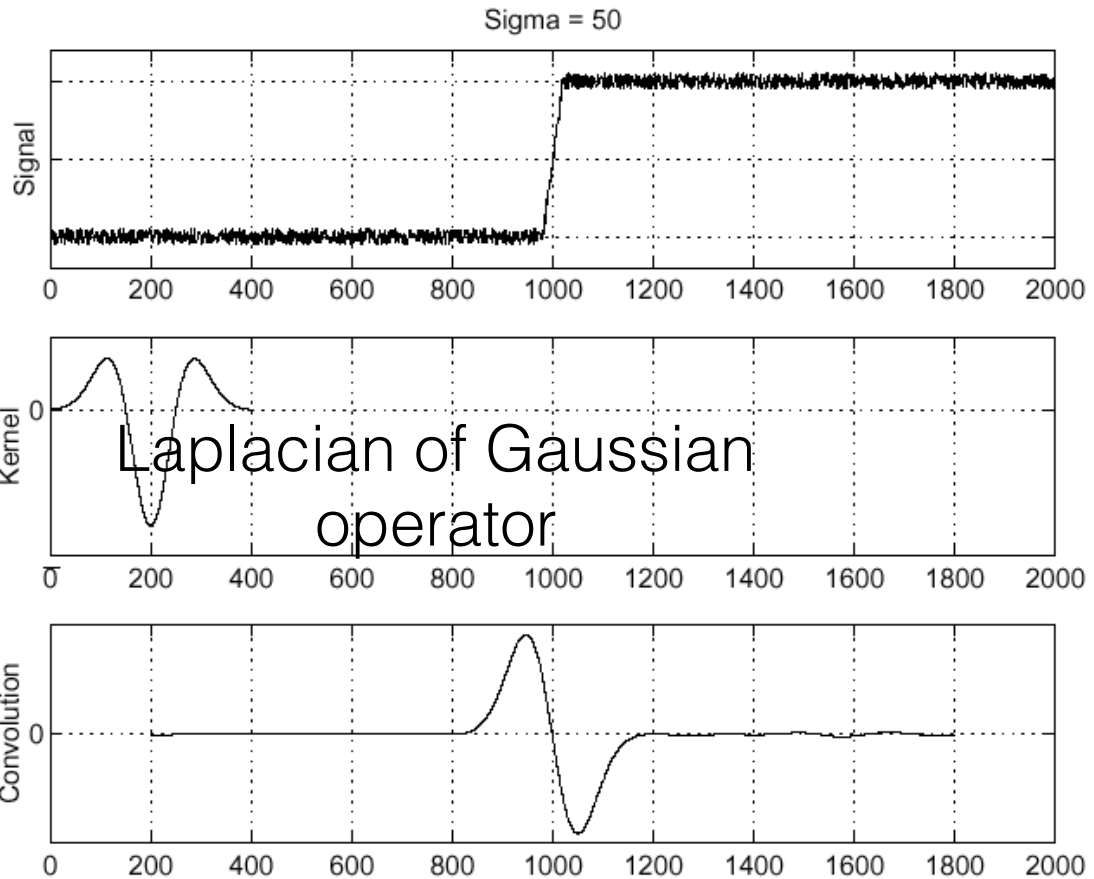
- Derivative filters

- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
 - **Zero:** no response in constant regions
- High absolute value at points of high contrast



Laplacian of Gaussian

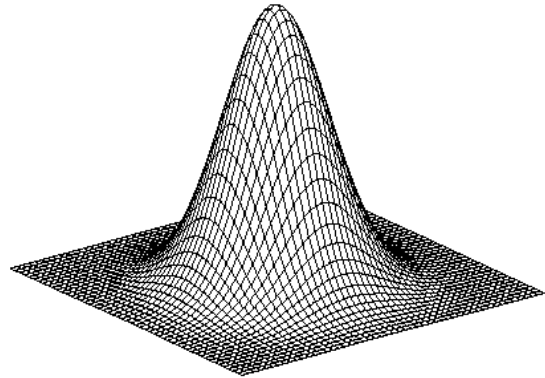
Consider $\frac{\partial^2}{\partial x^2}(h \star f)$



Where is the edge?

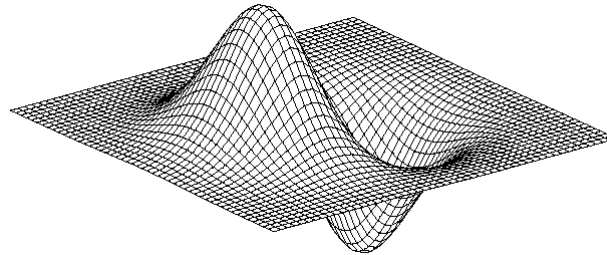
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

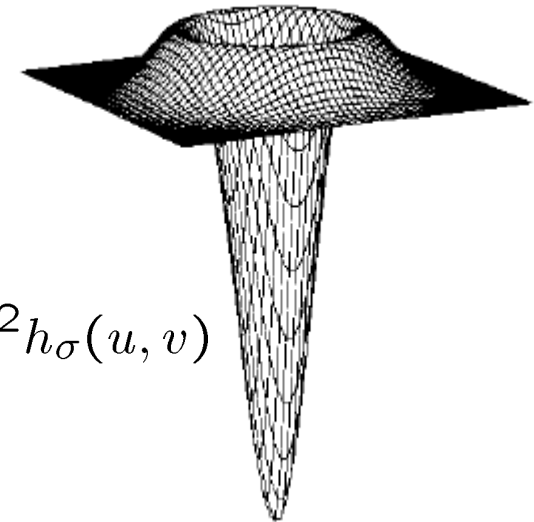
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



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

- The Laplacian operator:

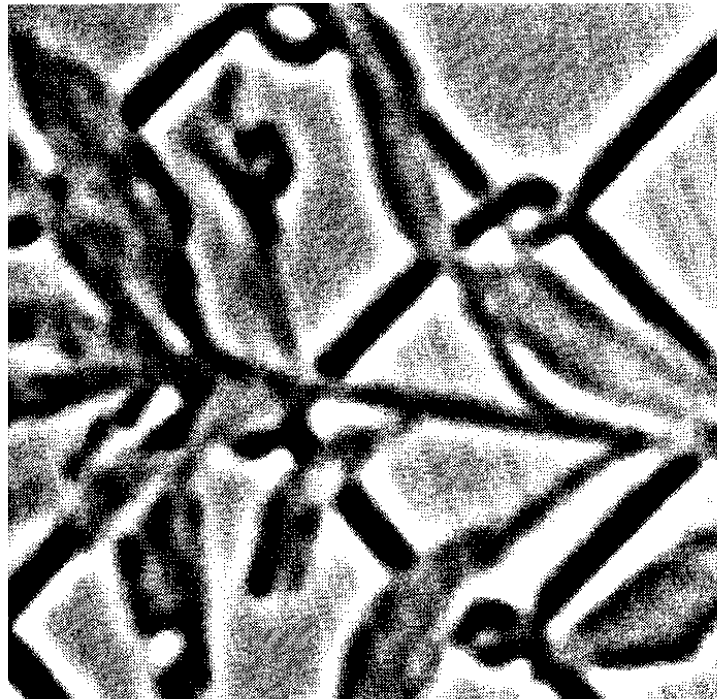
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian of Gaussian



original image

Laplacian of Gaussian



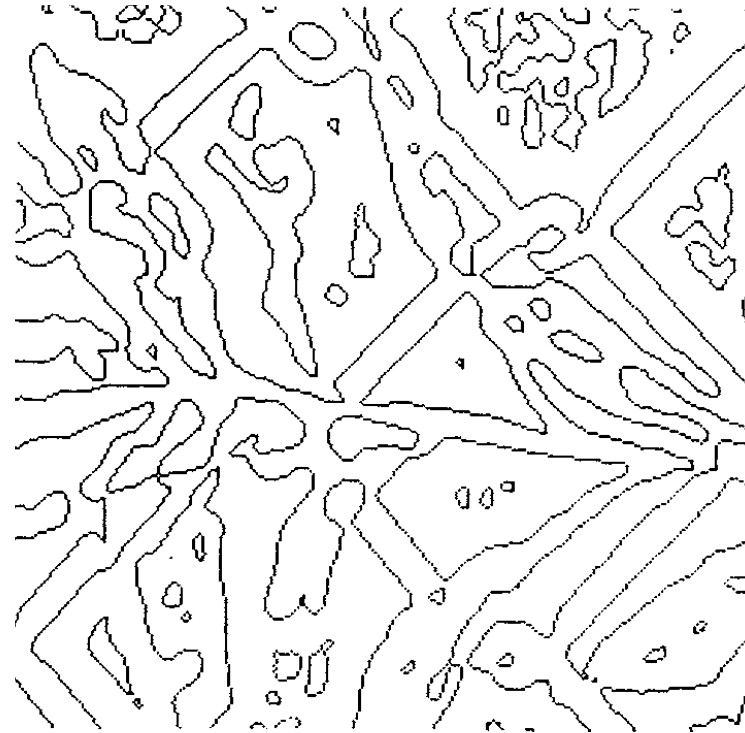
convolution with
 $\nabla^2 h_\sigma(u, v)$

Laplacian of Gaussian



convolution with
 $\nabla^2 h_\sigma(u, v)$
(pos. values – white, neg. values – black)

Laplacian of Gaussian



zero-crossings

Designing an edge detector

- Criteria for a good edge detector:
 - **Good detection:** the optimal detector should find all real edges, ignoring noise or other artifacts
 - **Good localization**
 - the edges detected must be as close as possible to the true edges
 - the detector must return one point only for each true edge point
- Cues of edge detection
 - Differences in color, intensity, or texture across the boundary
 - Continuity and closure
 - High-level knowledge

The Canny edge detector



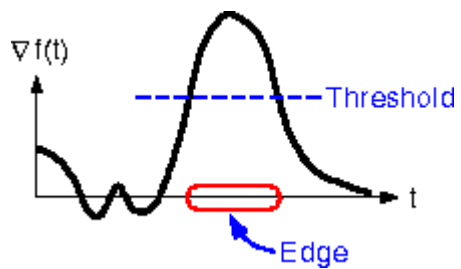
original image (Lena)

The Canny edge detector



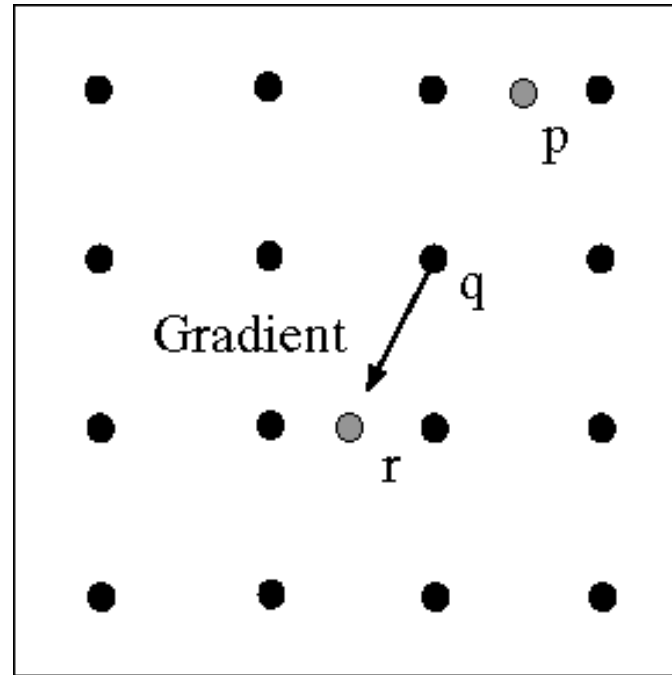
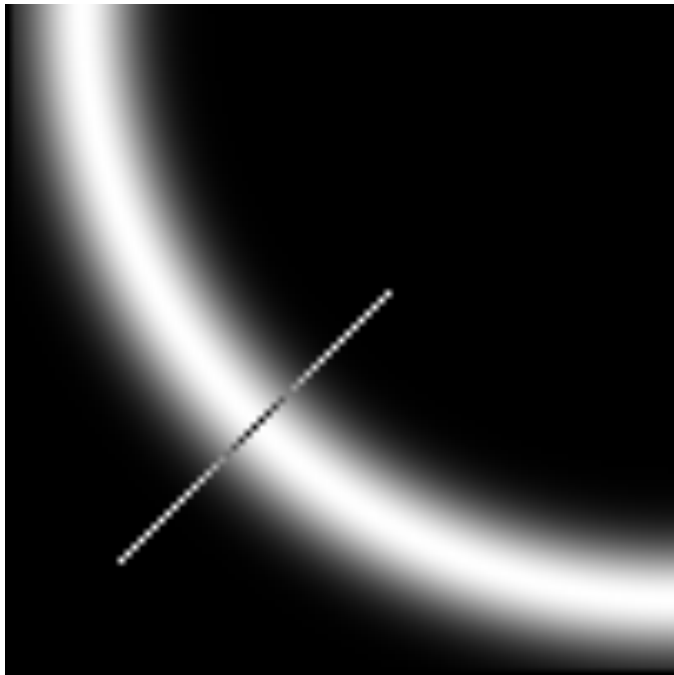
thresholding

The Canny edge detector



How to turn these thick regions of the gradient into curves?

Non-maximum suppression



Check if pixel is local maximum along gradient direction,
select single max across width of the edge
– requires checking interpolated pixels p and r

The Canny Edge Detector



Problem: pixels along this edge didn't survive the thresholding

thinning
(non-maximum suppression)

Slide credit: K. Grauman

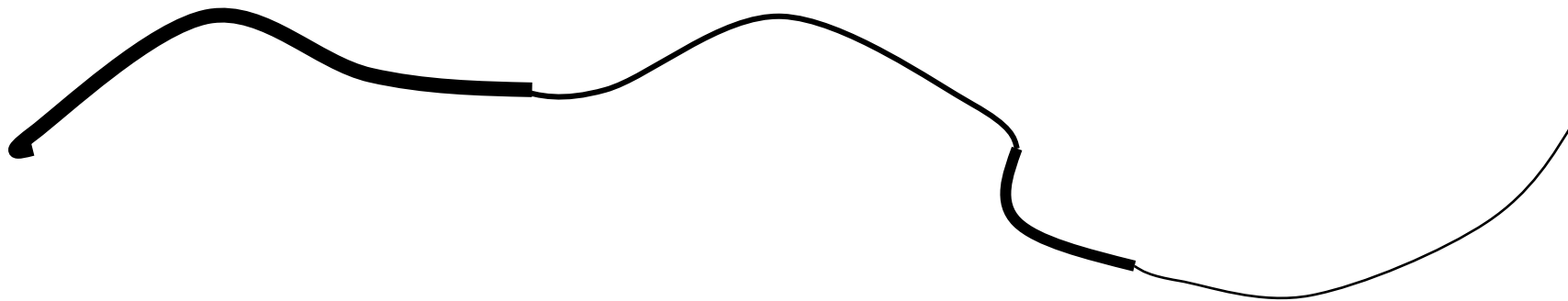
Hysteresis thresholding

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



Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
 - drop-outs? use **hysteresis**
 - use a high threshold to start edge curves and a low threshold to continue them.



Hysteresis thresholding



original image



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold

Hysteresis thresholding



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold

Recap: Canny edge detector

1. Filter image with derivative of Gaussian
 2. Find magnitude and orientation of gradient
 - 3. Non-maximum suppression:**
 - Thin wide “ridges” down to single pixel width
 - 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
- MATLAB: `edge(image, 'canny');`

Effect of σ (Gaussian kernel spread/size)



original

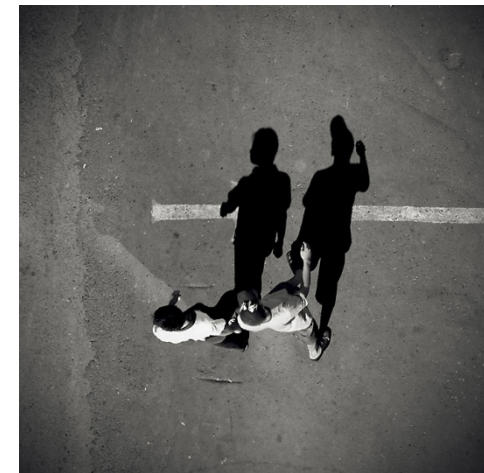
Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

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

Low-level edges vs. perceived contours



Background

Texture

Shadows

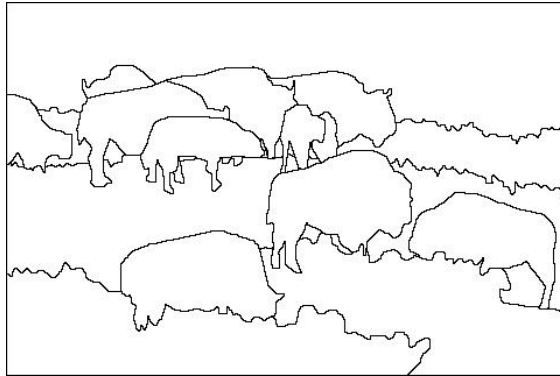
Slide credit: K. Grauman

Edge detection is just the beginning...

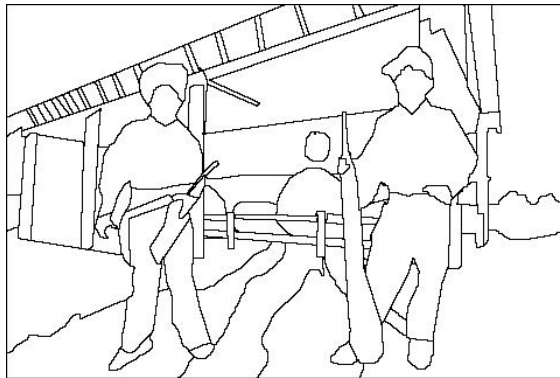
image



human segmentation



gradient magnitude

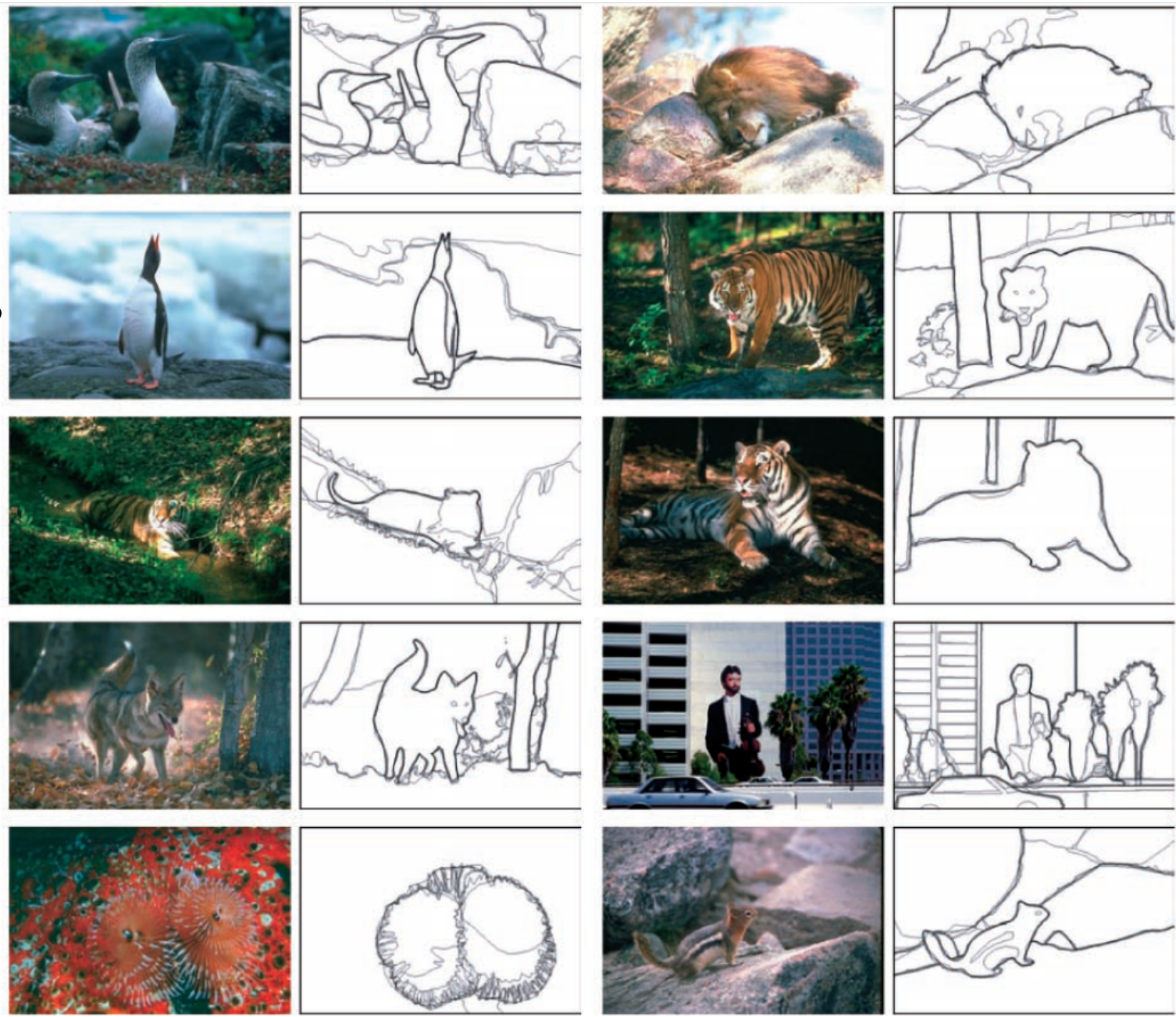


- Berkeley segmentation database:

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

Learn from humans which combination of features is most indicative of a “good” contour?

[D. Martin et al. PAMI 2004]



Slide credit: K. Grauman

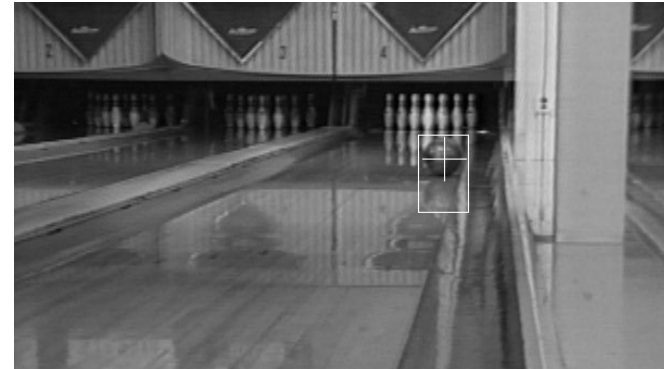
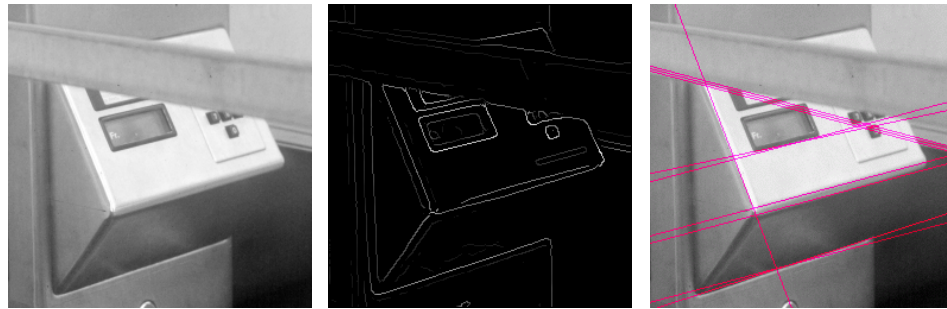
Human-marked segment boundaries

Edges vs. Boundaries

- Edges
 - abrupt changes in the intensity
 - discontinuities in intensity values
 - a local entity
- Edge detection may result in
 - Breaks in the edges due to non-uniform illumination
 - Spurious edges
- Boundaries
 - related to regions
 - a global entity
 - assemble of meaningful edge points
- Boundary detection requires grouping or fitting

Fitting

- Want to associate a model with observed features



[Fig from Marszalek & Schmid, 2007]

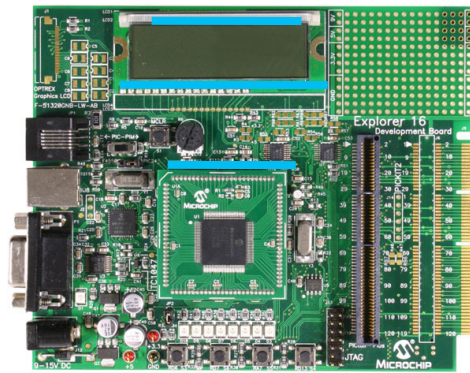
For example, the model could be a line, a circle, or an arbitrary shape.

Fitting: Main idea

- Choose a parametric model to represent a set of features
- Membership criterion is not local
 - Can't tell whether a point belongs to a given model just by looking at that point
- Three main questions:
 - What model represents this set of features best?
 - Which of several model instances gets which feature?
 - How many model instances are there?
- Computational complexity is important
 - It is infeasible to examine every possible set of parameters and every possible combination of features

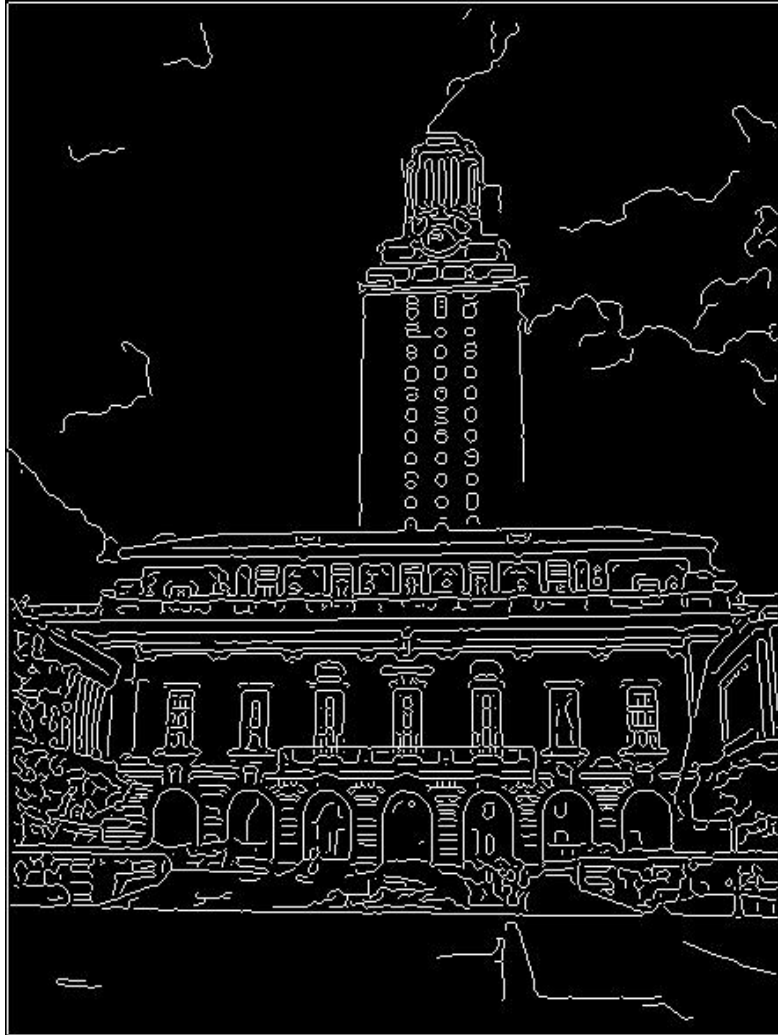
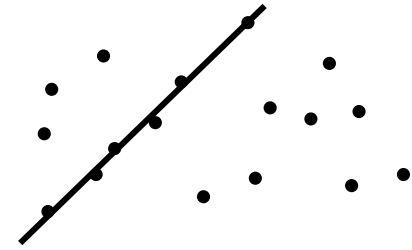
Example: Line fitting

- Why fit lines?
 - Many objects characterized by presence of straight lines



Wait, why aren't we done just by running edge detection?

Difficulty of line fitting



- **Extra** edge points (clutter), multiple models:
 - which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing**:
 - how to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
 - how to detect true underlying parameters?

Voting

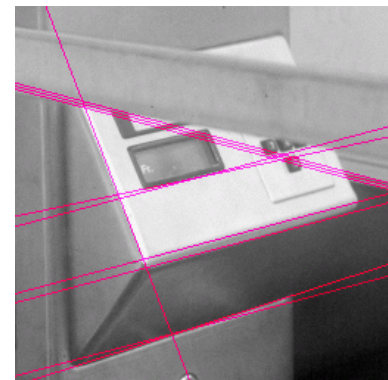
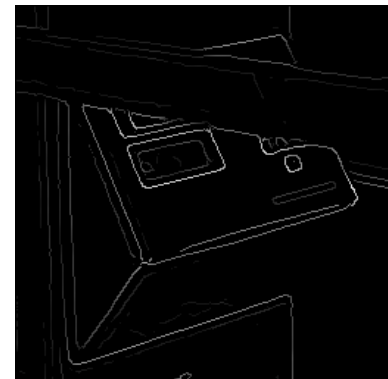
- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features *vote for all models that are compatible with it*.
 - Cycle through features, cast votes for model parameters.
 - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of “good” features.

Fitting lines: Hough transform

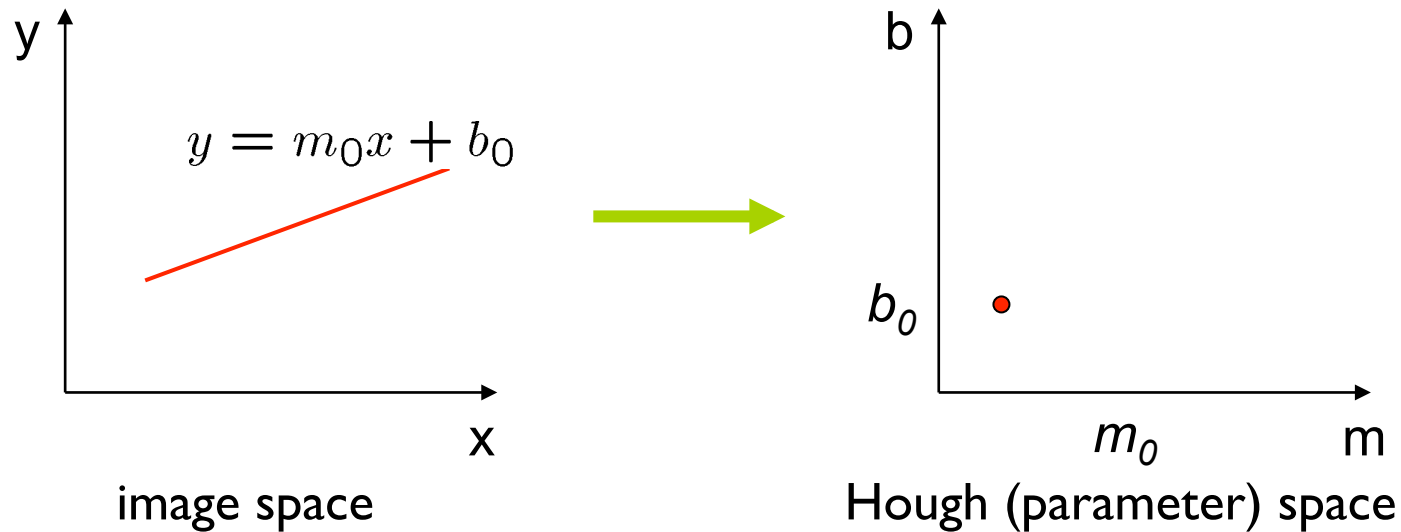
- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?
- **Hough Transform** is a voting technique that can be used to answer all of these questions.

Main idea:

1. Record vote for each possible line on which each edge point lies.
2. Look for lines that get many votes.



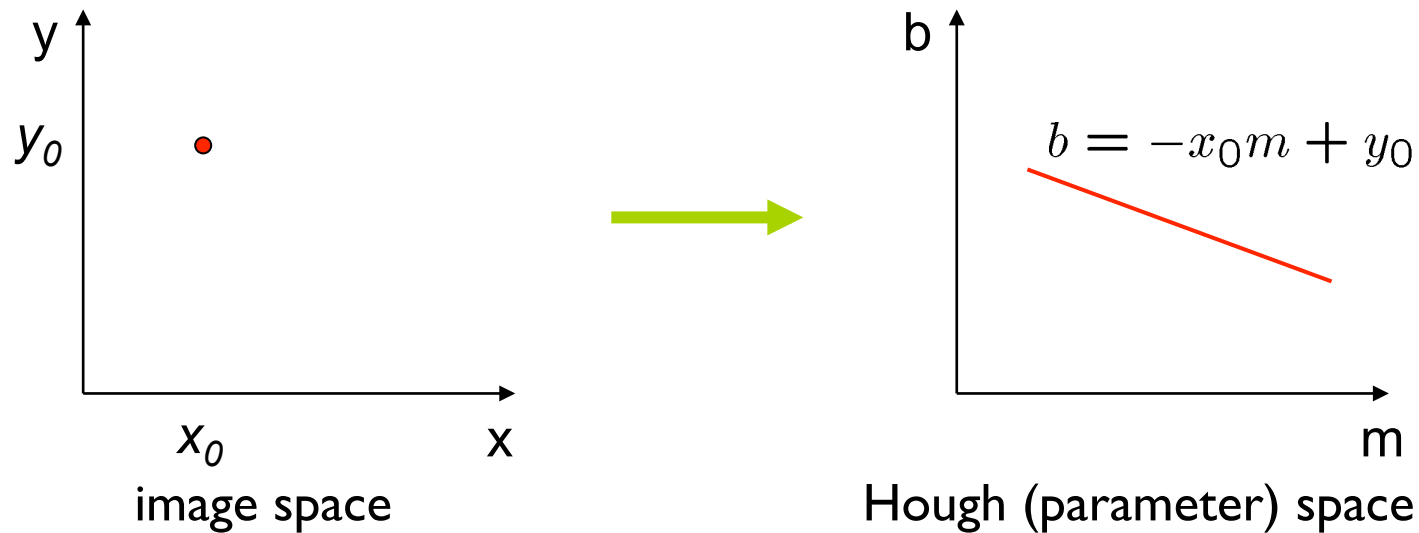
Finding lines in an image: Hough space



Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
 - given a set of points (x,y) , find all (m,b) such that $y = mx + b$

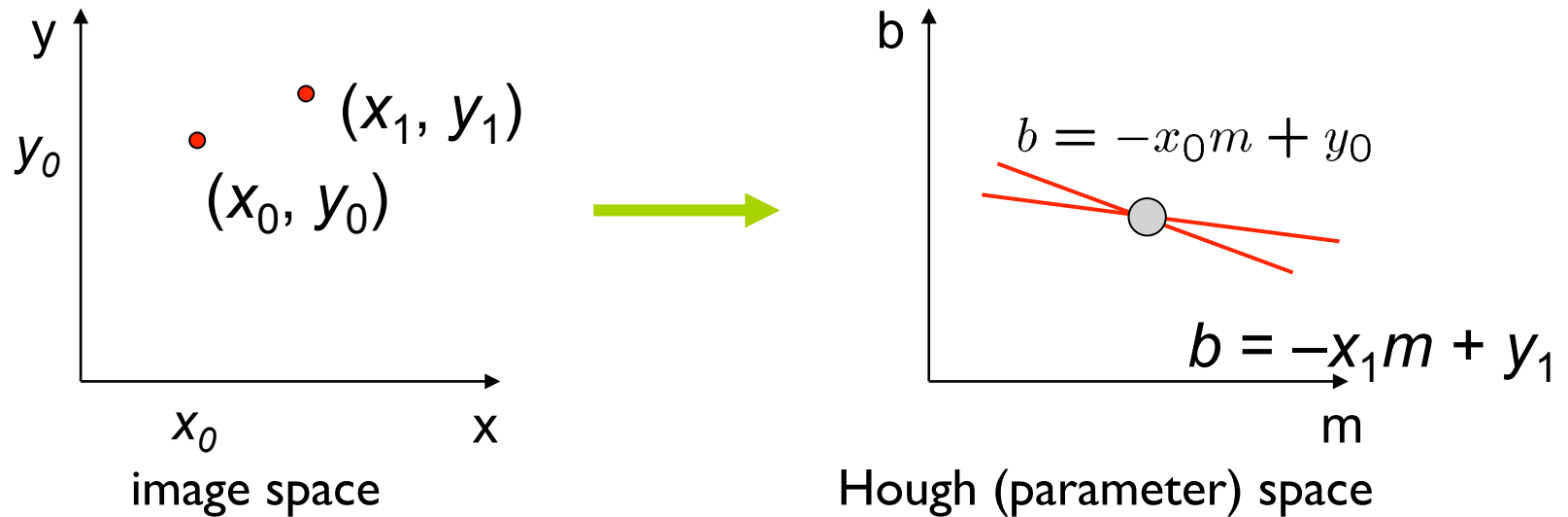
Finding lines in an image: Hough space



Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
 - given a set of points (x,y) , find all (m,b) such that $y = mx + b$
- What does a point (x_0, y_0) in the image space map to?
 - Answer: the solutions of $b = -x_0m + y_0$
 - this is a line in Hough space

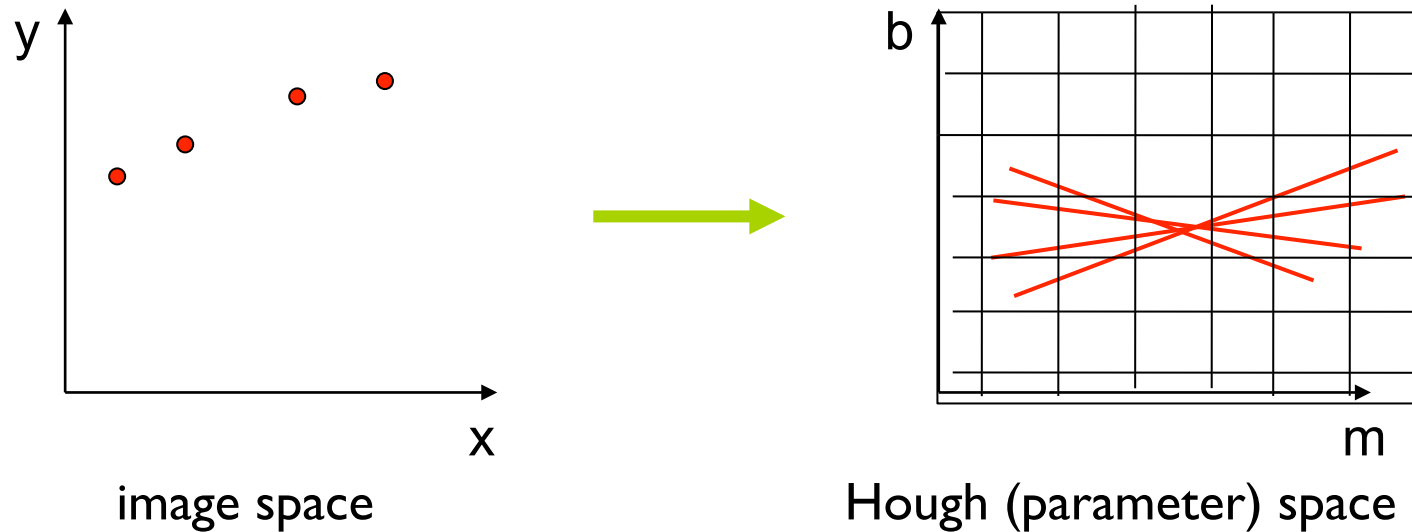
Finding lines in an image: Hough space



What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1) ?

- It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Finding lines in an image: Hough space

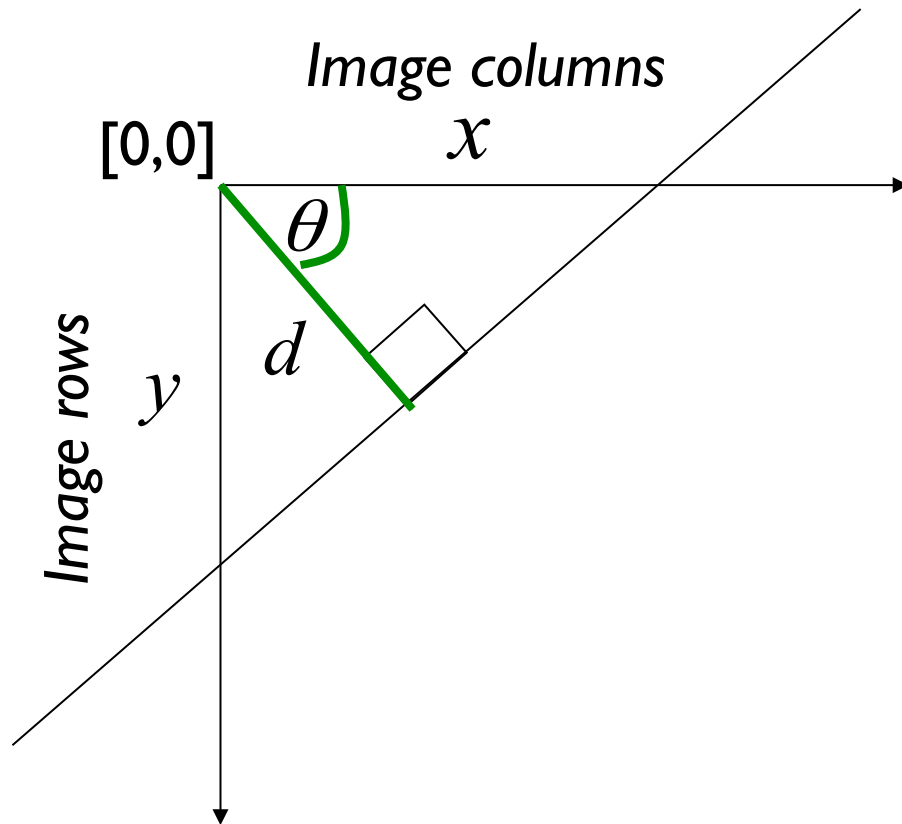


How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

- Let each edge point in image space vote for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Polar representation for lines

Issues with usual (m,b) parameter space: can take on infinite values, undefined for vertical lines.



d : perpendicular distance from line to origin

θ : angle the perpendicular makes with the x-axis

$$x \cos \theta - y \sin \theta = d$$

Point in image space \rightarrow sinusoid segment in Hough space

Hough transform algorithm

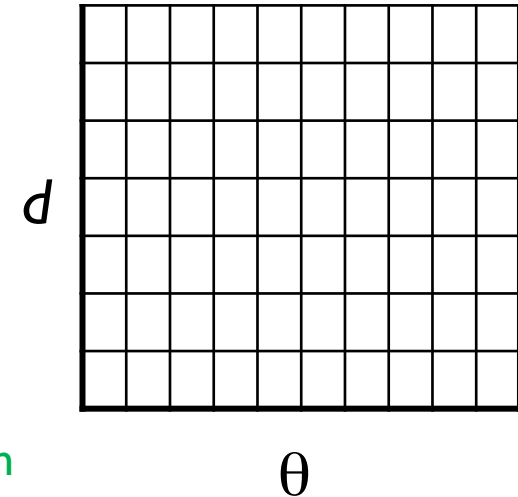
Using the polar parameterization:

$$x \cos \theta - y \sin \theta = d$$

Basic Hough transform algorithm

1. Initialize $H[d, \theta] = 0$
2. for each edge point $I[x, y]$ in the image
for $\theta = [\theta_{\min} \text{ to } \theta_{\max}]$ // some quantization
 $d = x \cos \theta - y \sin \theta$
 $H[d, \theta] += 1$
3. Find the value(s) of (d, θ) where $H[d, \theta]$ is maximum
4. The detected line in the image is given by $d = x \cos \theta - y \sin \theta$

H: accumulator array (votes)



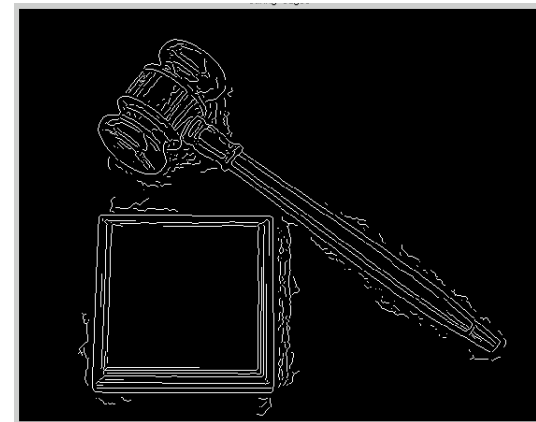
Time complexity (in terms of number of votes per pt)?

Hough transform algorithm

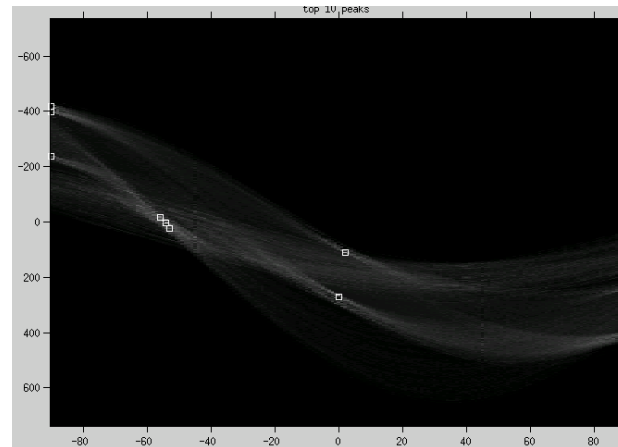
Original image



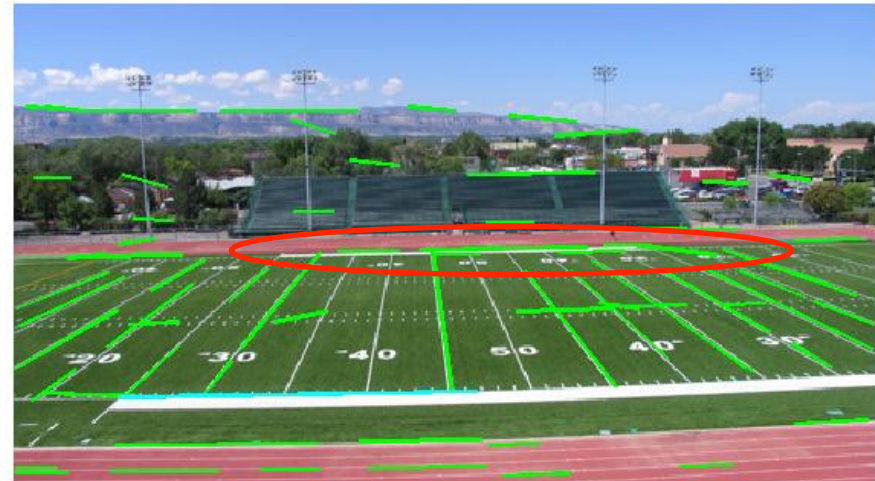
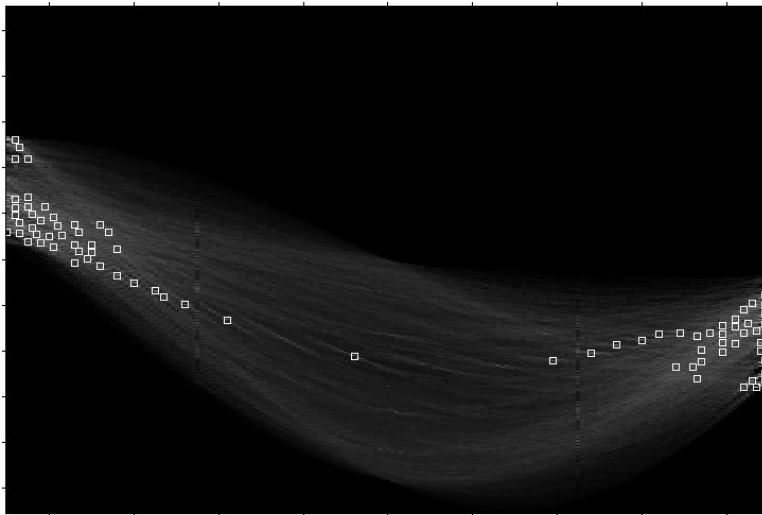
Canny edges



Vote space and top peaks



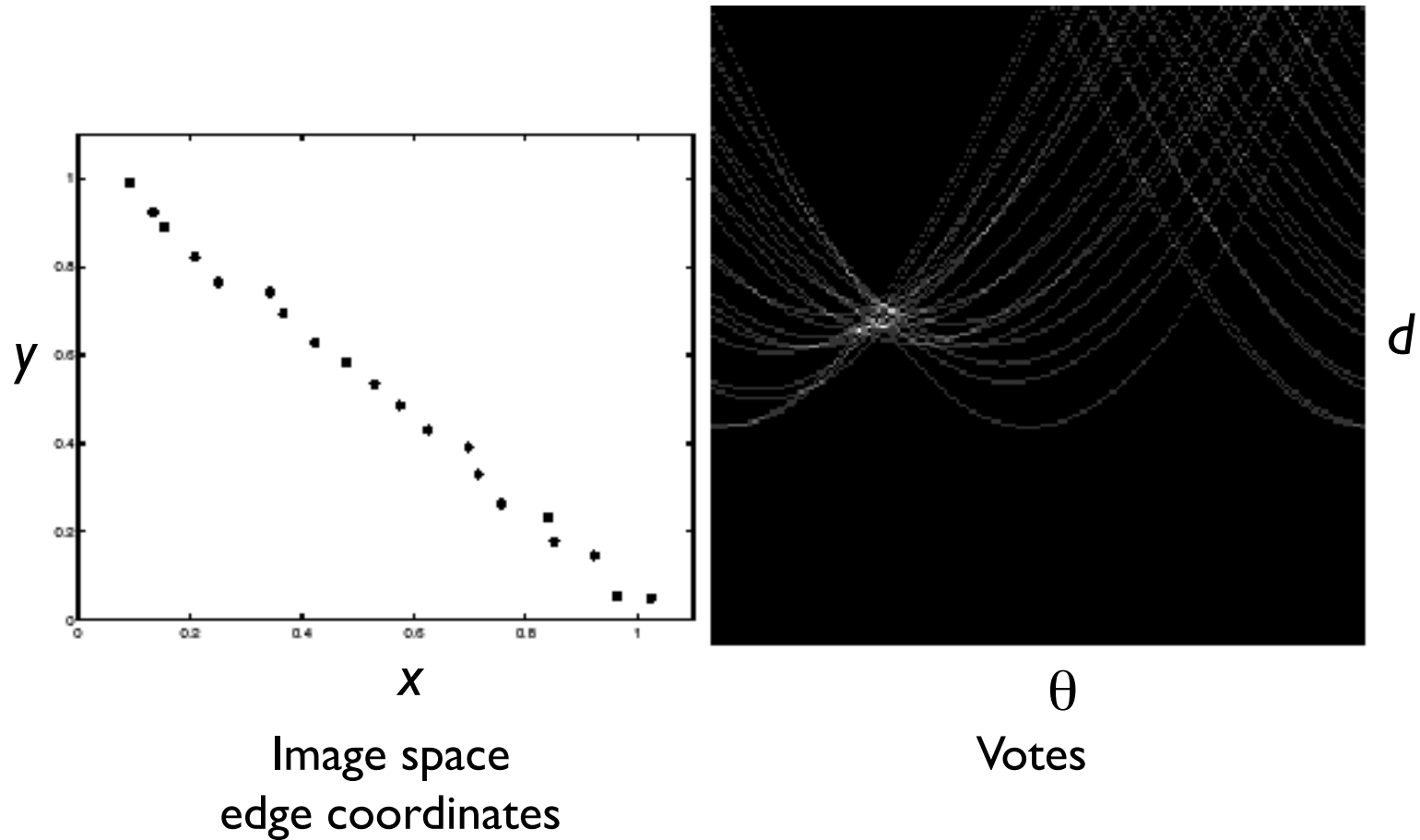
Hough transform algorithm



Showing longest segments found

Slide credit: K. Grauman

Impact of noise on Hough



What difficulty does this present for an implementation?

Impact of noise on Hough

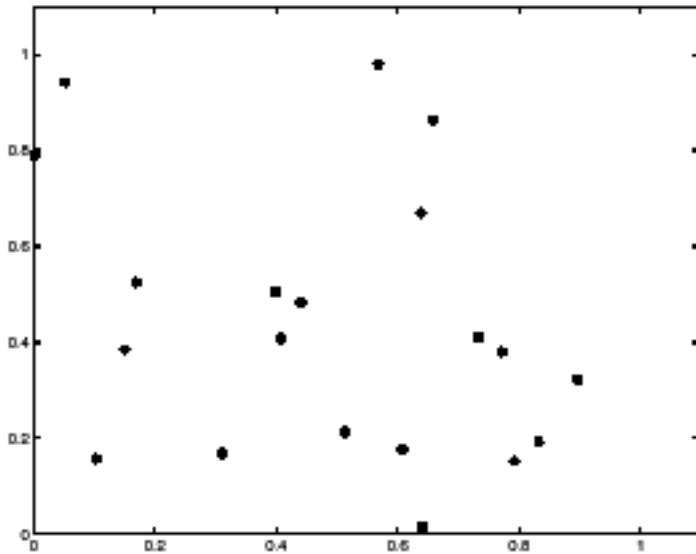
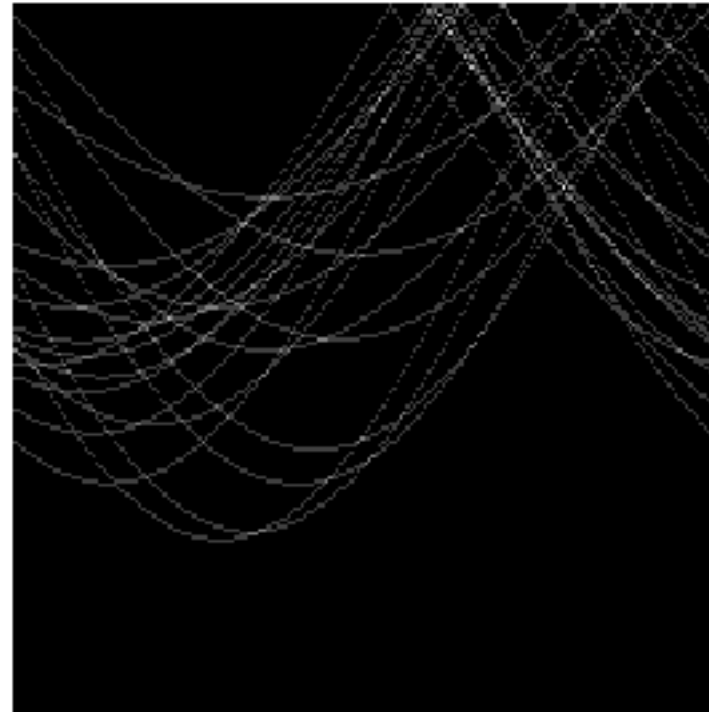


Image space
edge coordinates



Votes

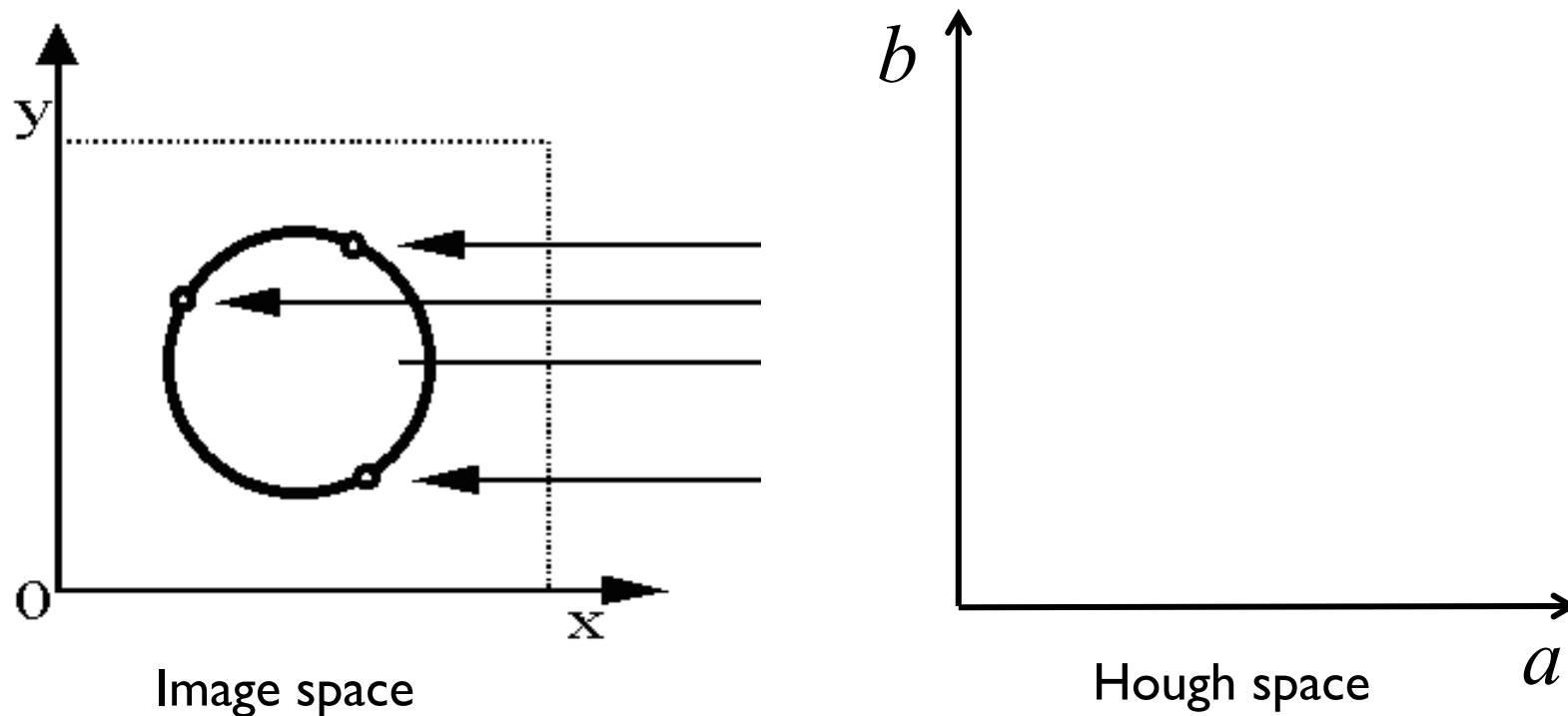
Here, everything appears to be “noise”, or random edge points, but we still see peaks in the vote space.

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r , unknown gradient direction

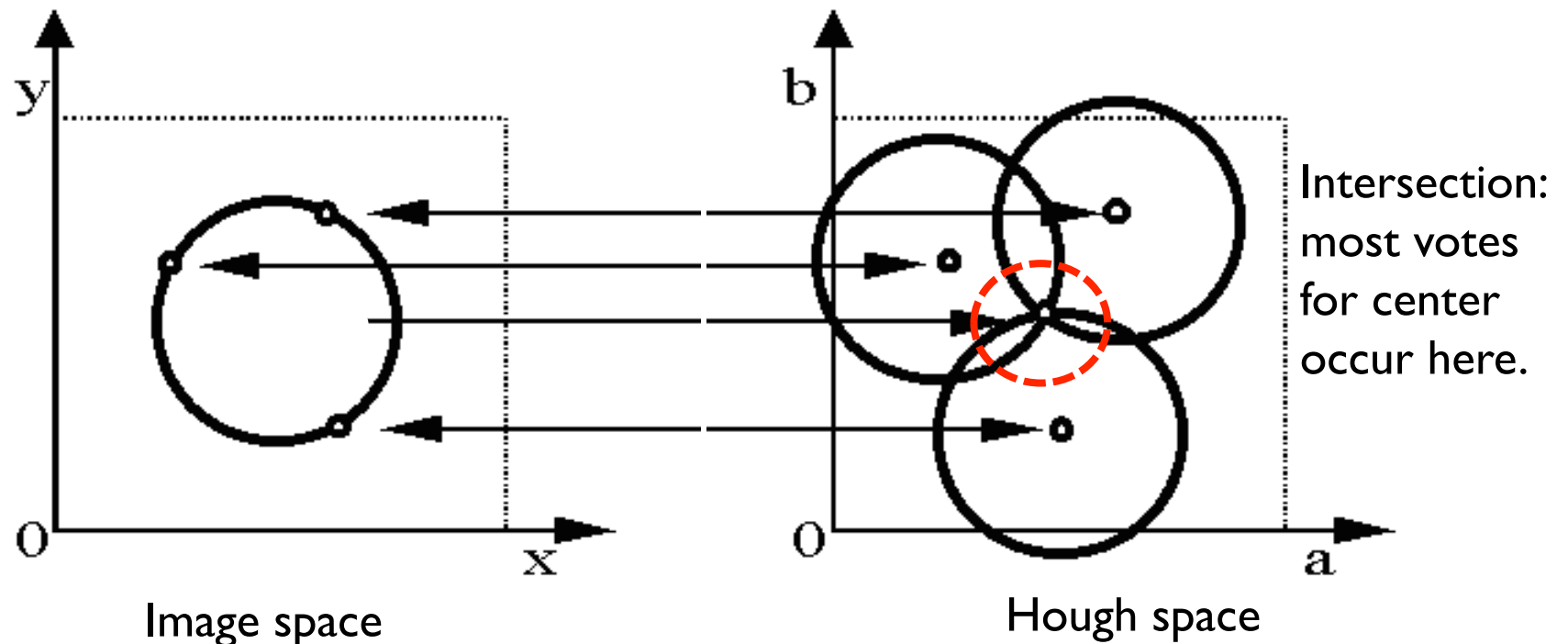


Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r , unknown gradient direction

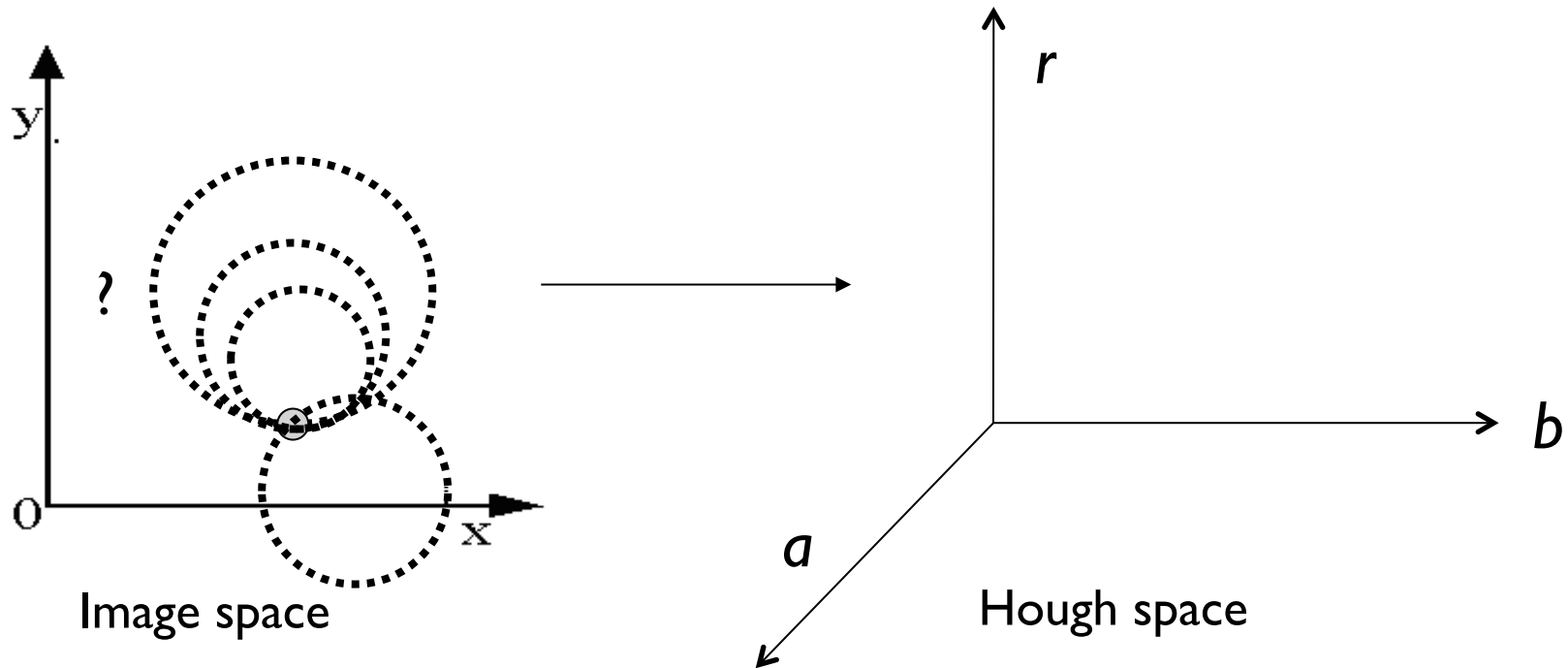


Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r , unknown gradient direction

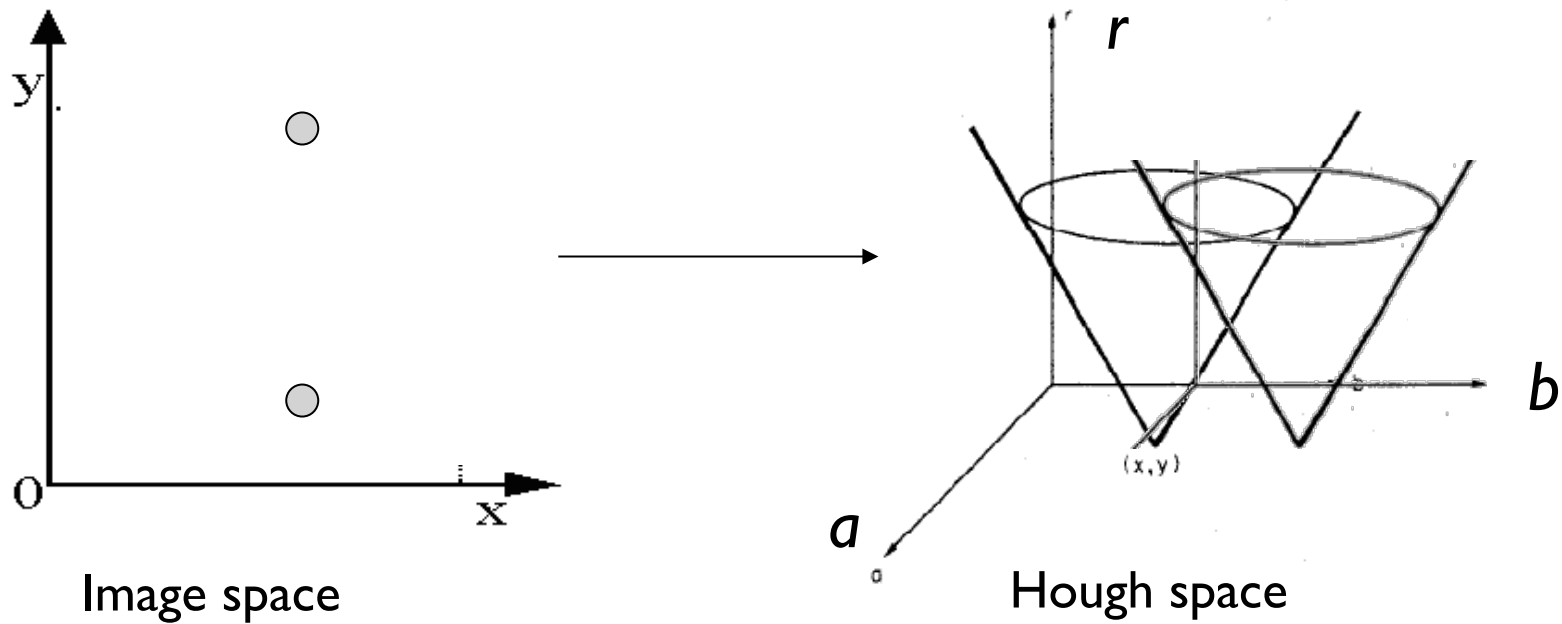


Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r , unknown gradient direction

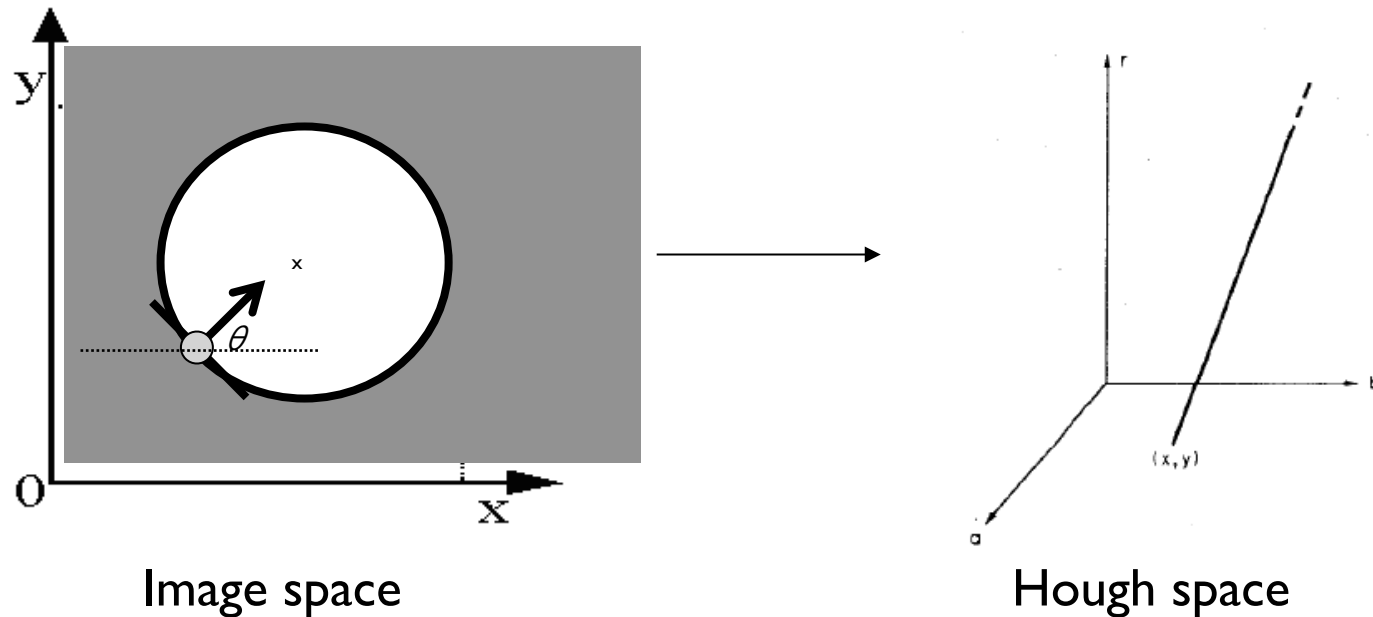


Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r , **known** gradient direction



Hough transform for circles

For every edge pixel (x,y) :

For each possible radius value r :

For each possible gradient direction θ :

// or use estimated gradient at (x,y)

$a = x - r \cos(\theta)$ *// column*

$b = y + r \sin(\theta)$ *// row*

$H[a,b,r] += 1$

end

end

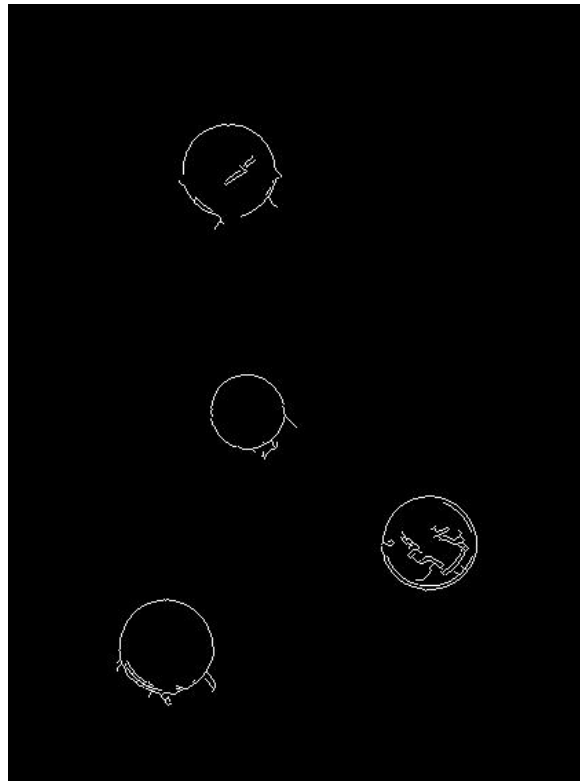
Time complexity per edgel?

Example: detecting circles with Hough

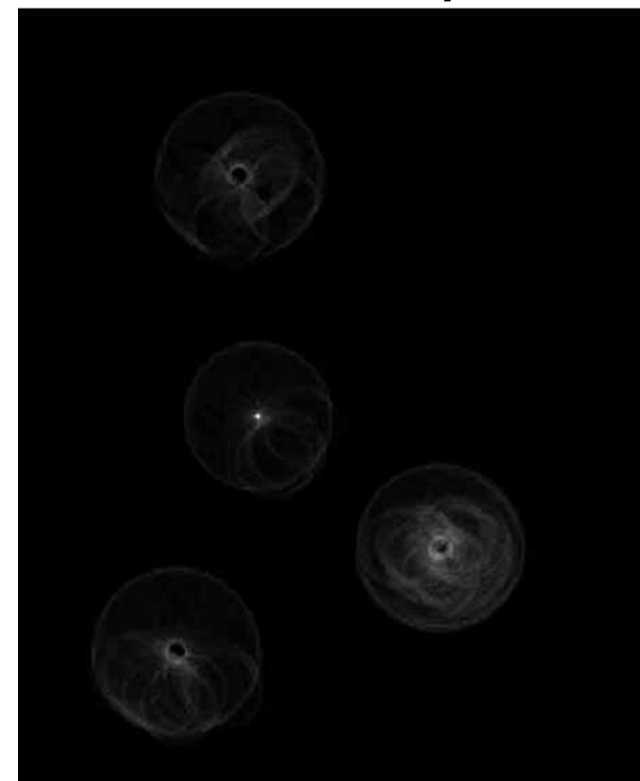
Original



Edges



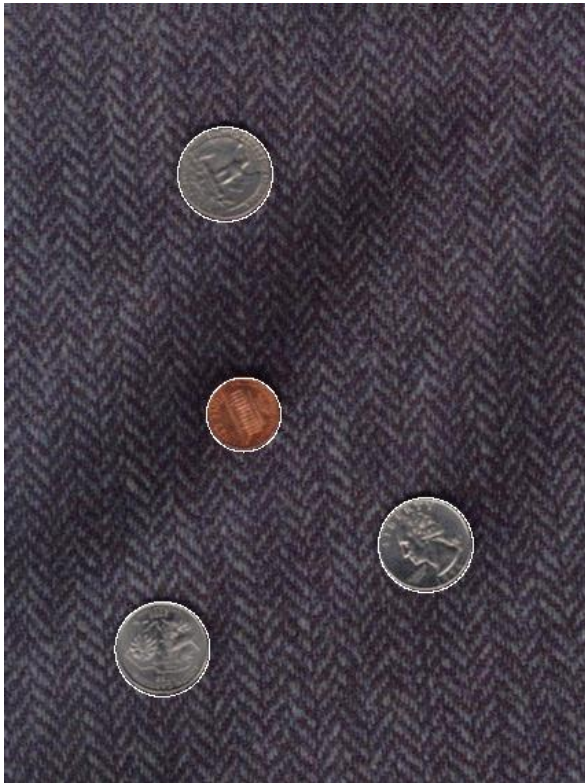
Votes: Penny



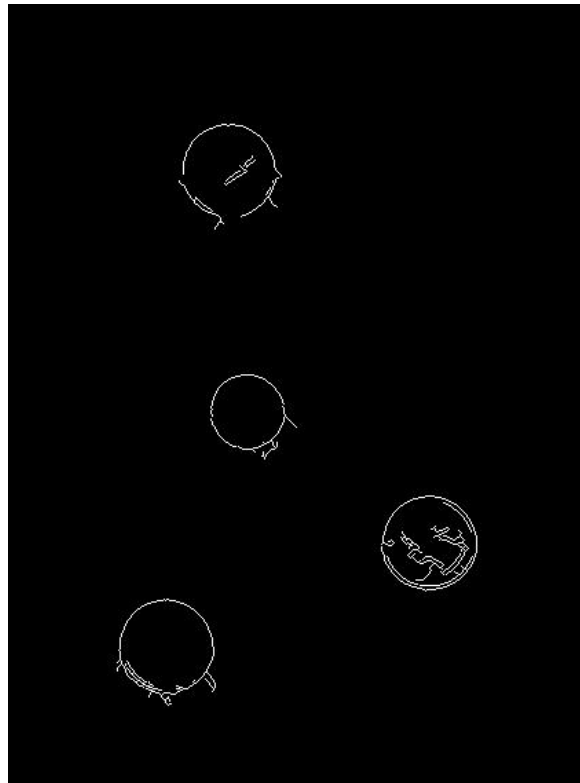
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Example: detecting circles with Hough

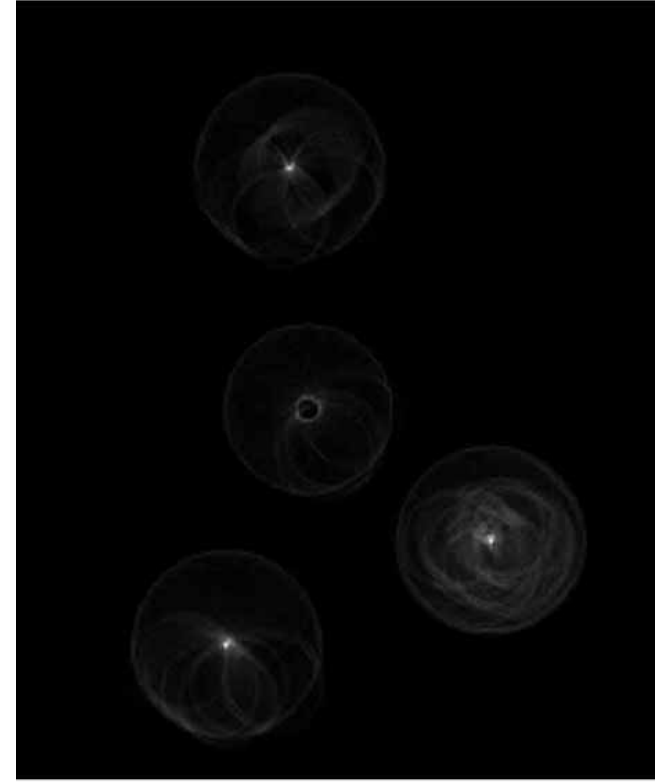
Original



Edges

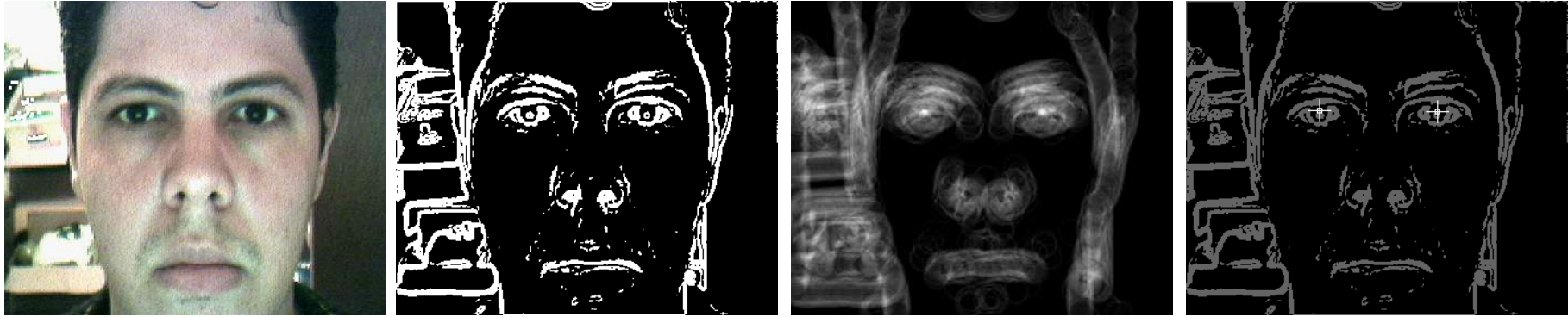


Votes: Quarter



Coin finding sample images from: Vivek Kwatra

Example: iris detection



Gradient+threshold

Hough space
(fixed radius)

Max detections

Hemerson Pistori and Eduardo Rocha Costa <http://rsbweb.nih.gov/ij/plugins/hough-circles.html>

Slide credit: K. Grauman

Hough transform: pros and cons

Pros

- All points are processed independently, so can cope with occlusion, gaps
- Some robustness to noise: noise points unlikely to contribute *consistently* to any single bin
- Can detect multiple instances of a model in a single pass

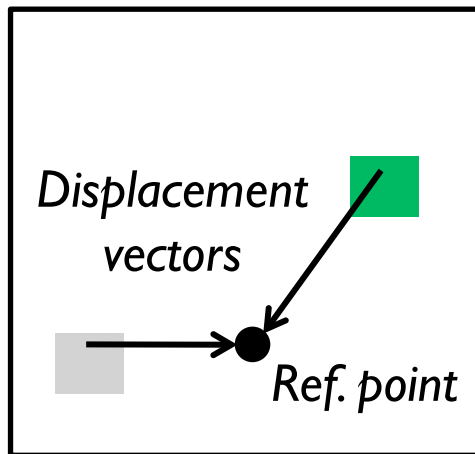
Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: can be tricky to pick a good grid size

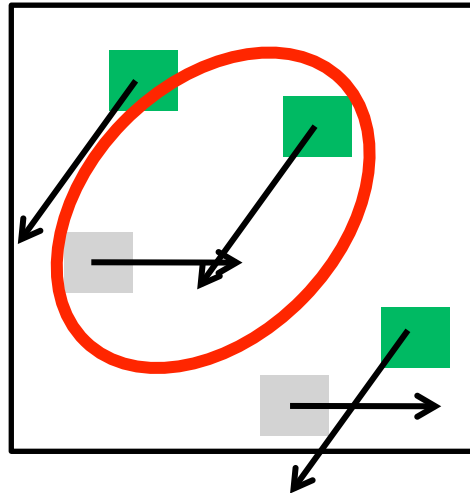
Generalized Hough Transform

- What if we want to detect arbitrary shapes?

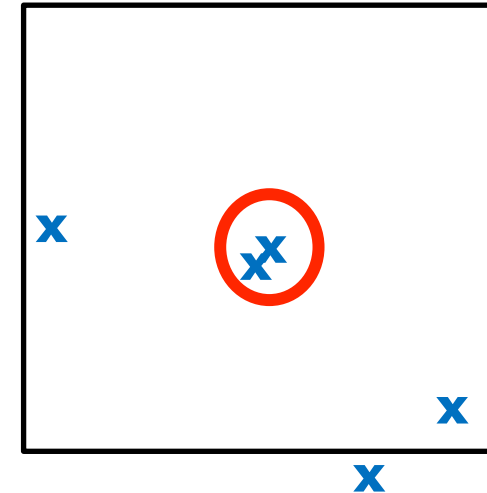
Intuition:



Model image



Novel image

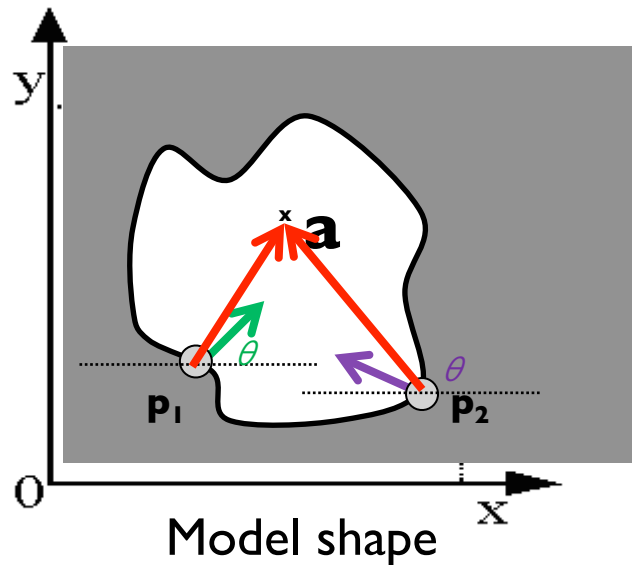


Vote space

Now suppose those colors encode gradient directions...

Generalized Hough Transform

- Define a model shape by its boundary points and a reference point.



Model shape

⋮	

Offline procedure:

At each boundary point, compute displacement vector: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$.





Store these vectors in a table indexed by gradient orientation θ .

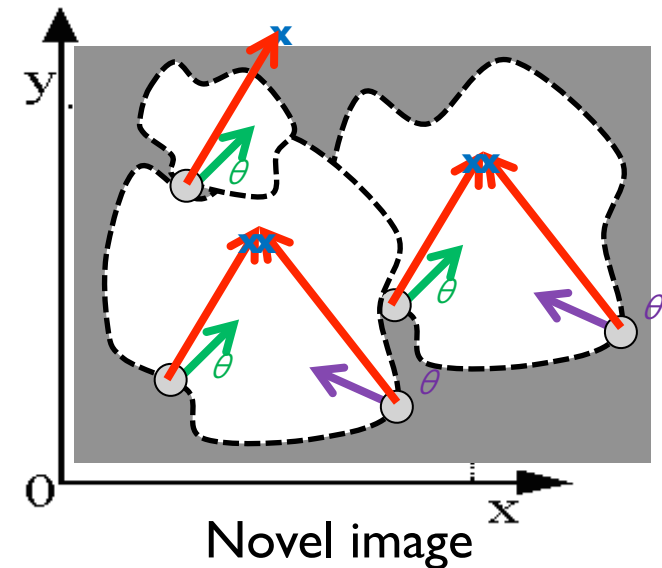
Generalized Hough Transform

Detection procedure:

For each edge point:

- Use its gradient orientation θ to index into stored table
- Use retrieved \mathbf{r} vectors to vote for reference point

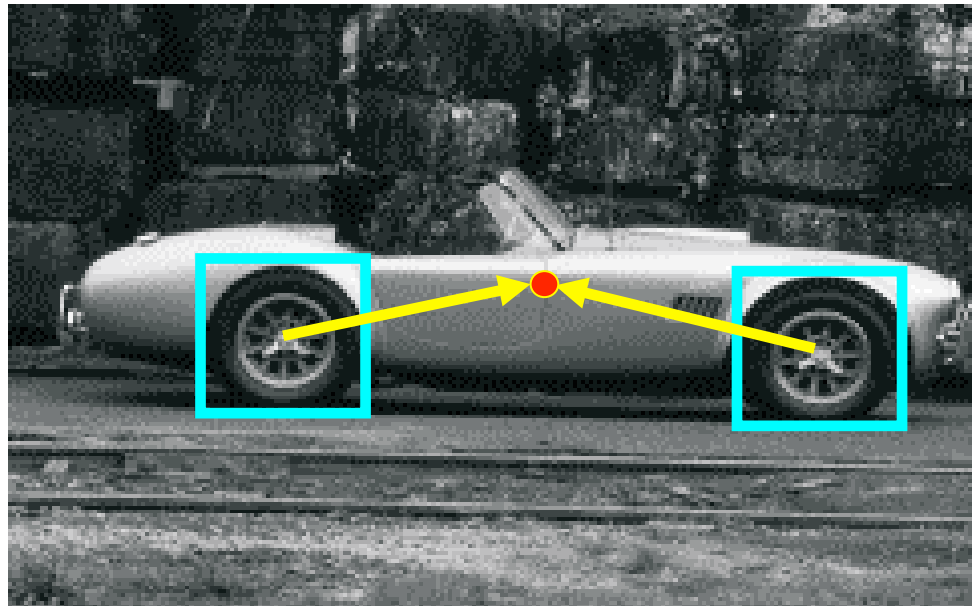
	 ...
	 ...
⋮	



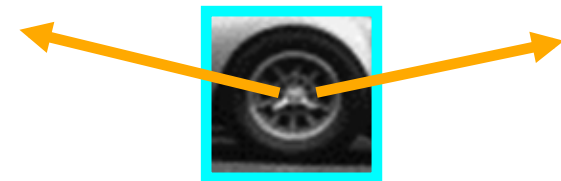
Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

Generalized Hough for object detection

- Instead of indexing displacements by gradient orientation, index by matched local patterns.



training image



“visual codeword” with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
[Combined Object Categorization and Segmentation with an Implicit Shape Model](#),
ECCV Workshop on Statistical Learning in Computer Vision 2004

Slide credit: S. Lazebnik

Generalized Hough for object detection

- Instead of indexing displacements by gradient orientation, index by “visual codeword”



test image

B. Leibe, A. Leonardis, and B. Schiele,
[Combined Object Categorization and Segmentation with an Implicit Shape Model](#),
ECCV Workshop on Statistical Learning in Computer Vision 2004

Slide credit: S. Lazebnik