Edges & Line Detection

Slides adapted from D.A. Forsyth, Lana Lazebnik, Li Fei-Fei, Kristen Grauman, D. Lowe, and Steve Seitz

Grouping & Fitting

- Previous Class: Template Matching
  - Slow
  - But, even if you could do this fast, it's only for a specific object
    - Different color, shape, etc.

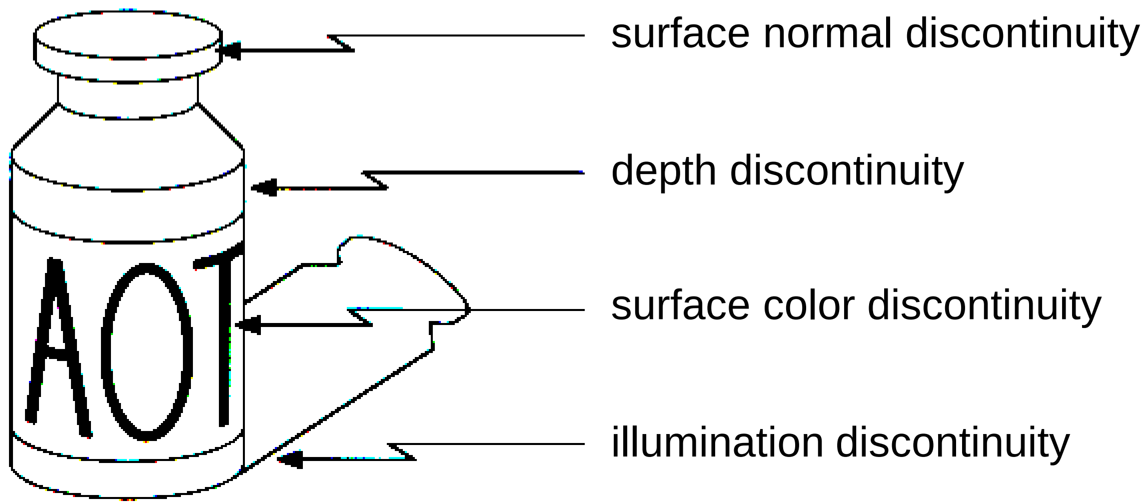
- We need a more general approach
- Insight: Much of the computation in template matching is wasted
- Plan: Focus on parts of the image that are interesting
  - *Issue: What parts of an image are interesting?*

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 are edges important?



- Edges are caused by a variety of factors
  - Something “interesting” is usually happening

Characterizing edges

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

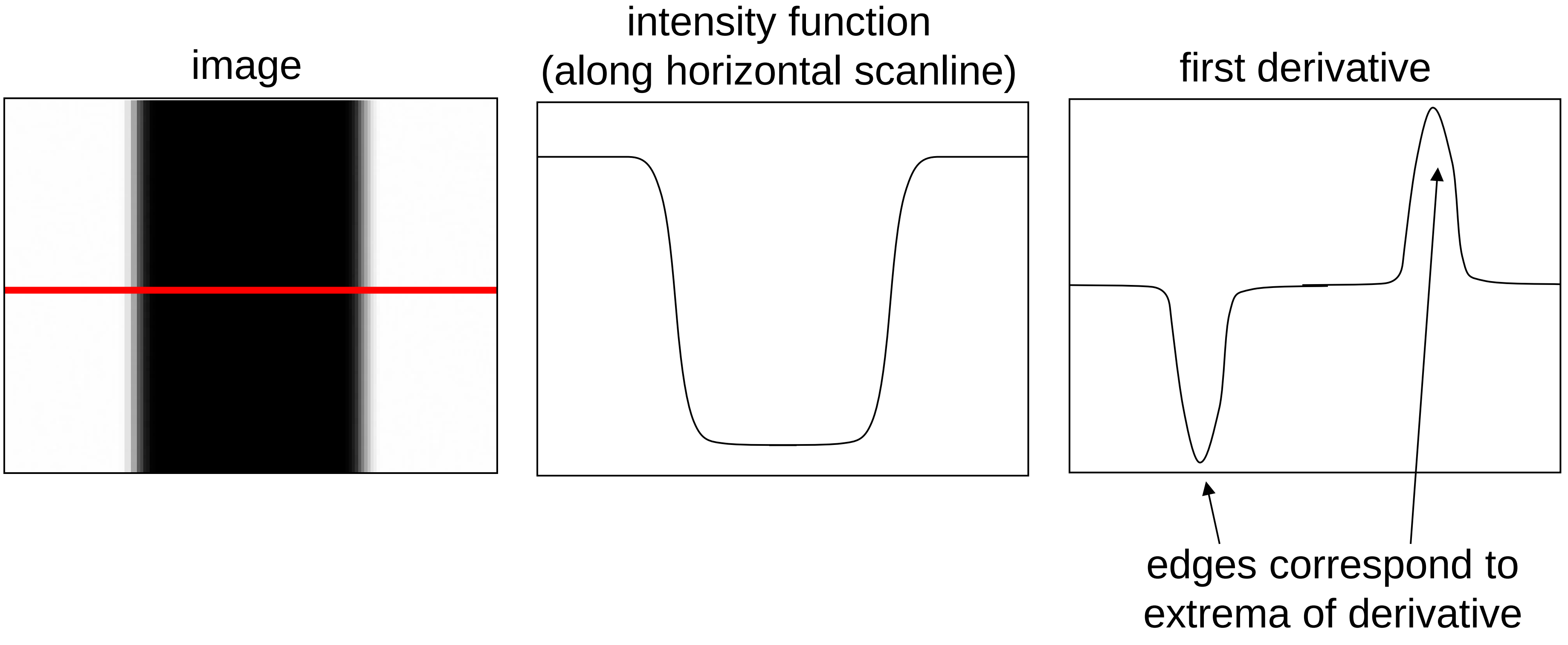
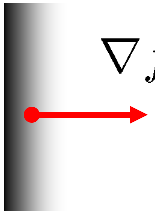
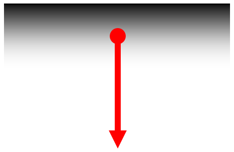


Image gradient

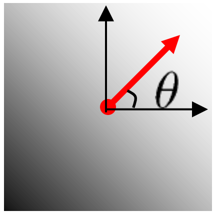
- The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- 

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



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



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

The gradient points in the direction of most rapid increase in intensity

The gradient direction is given by $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

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

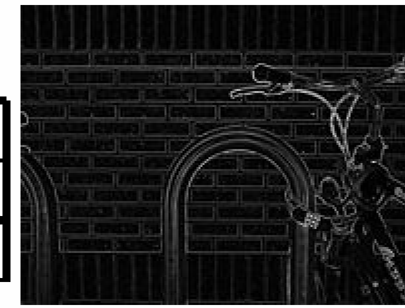
The *edge strength* is given by the gradient magnitude

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

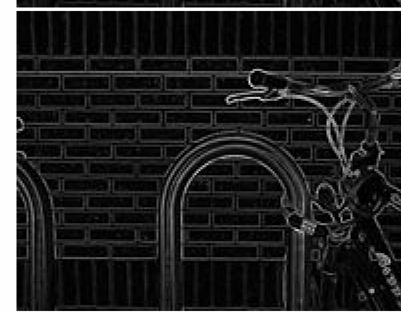
Source: Steve Seitz

Different Edge Filters

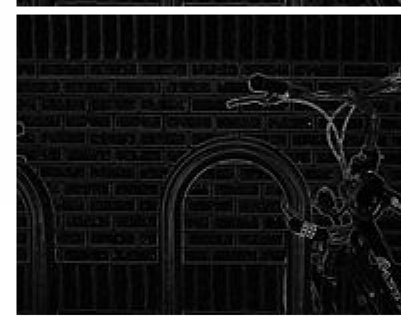
Prewitt: $$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



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



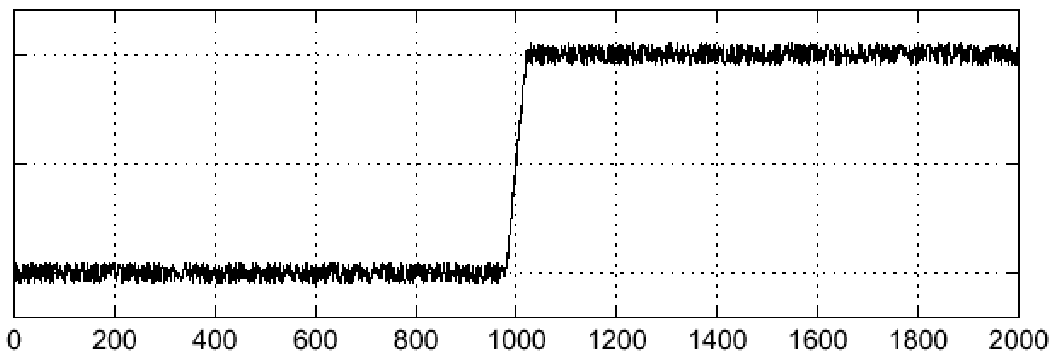
Roberts: $$M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$



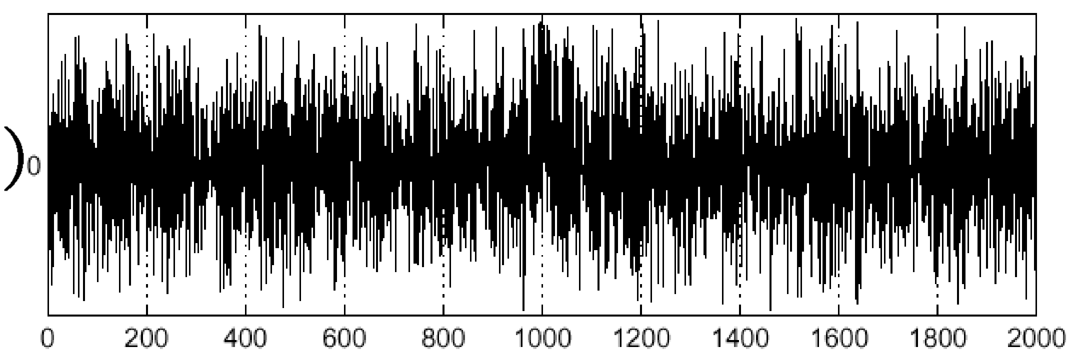
Effects of noise

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

$f(x)$

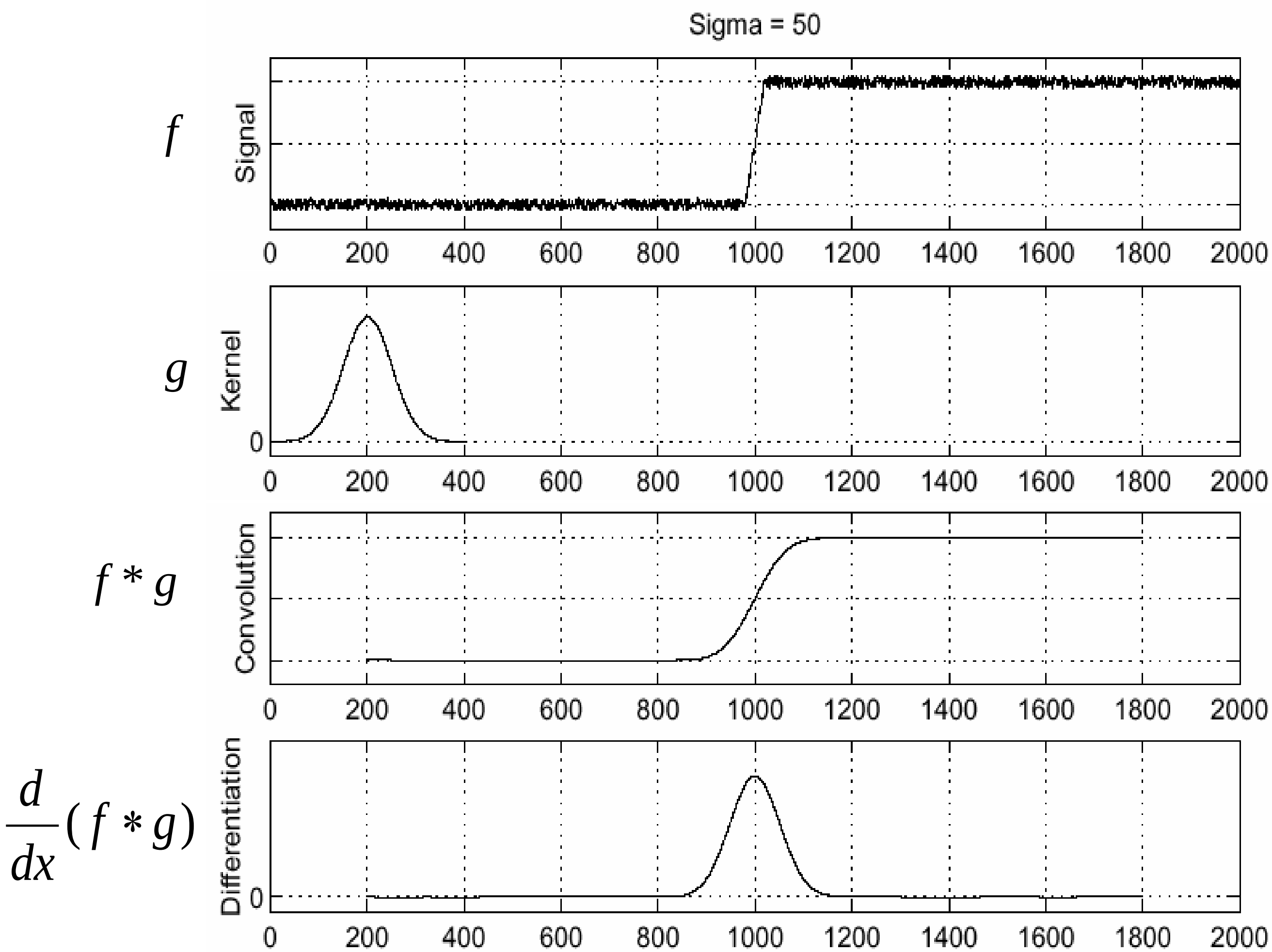


$\frac{d}{dx}f(x)$



Where is the edge?

Solution: smooth first



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

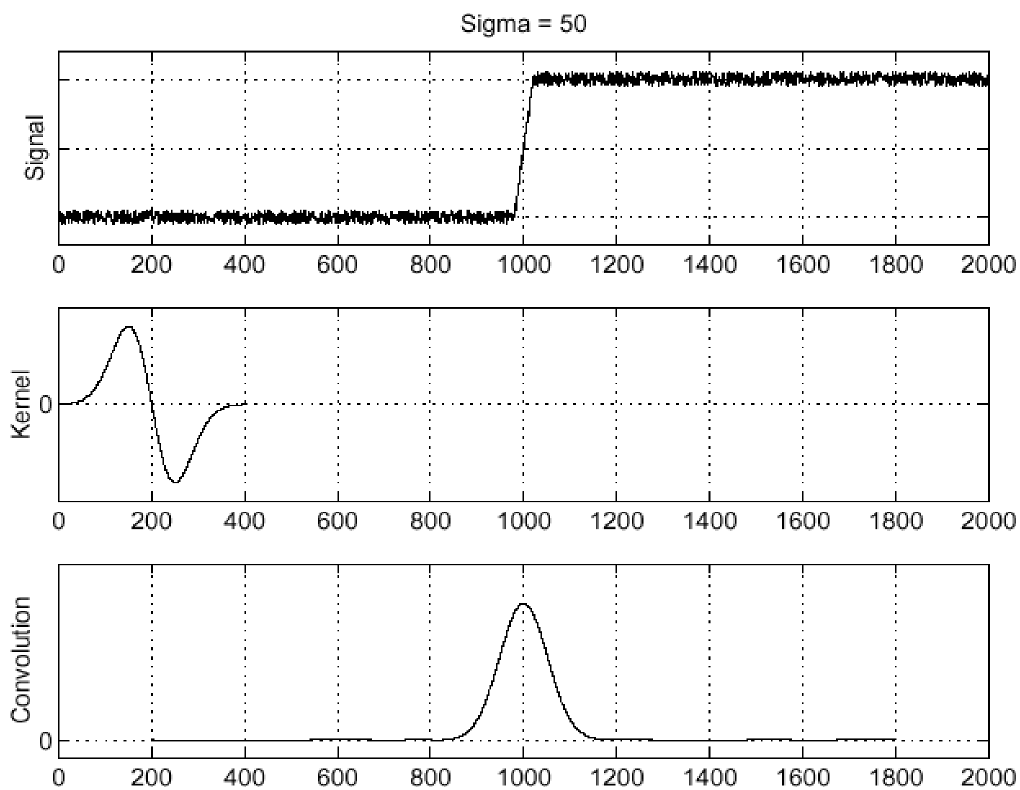
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$
- This saves us one operation:

f

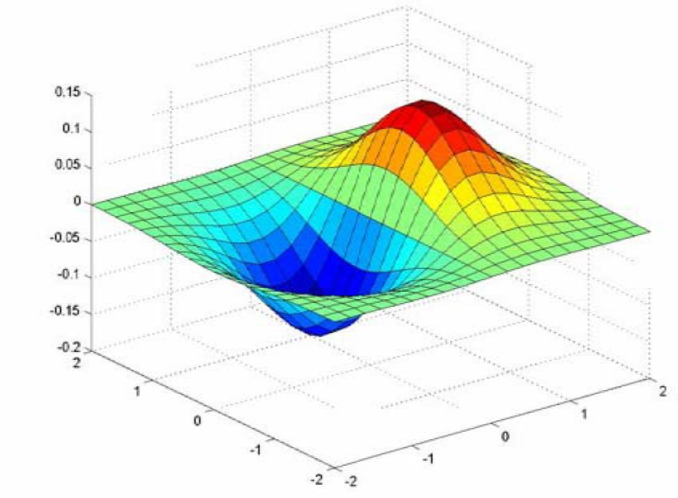
$\frac{d}{dx} g$

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

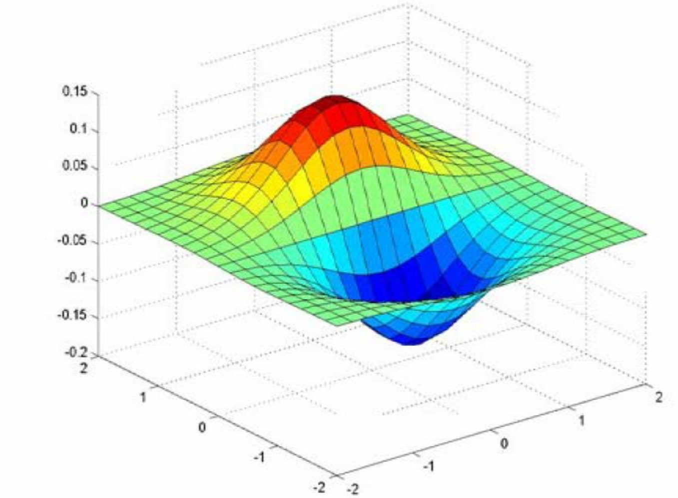


Source: S. Seitz

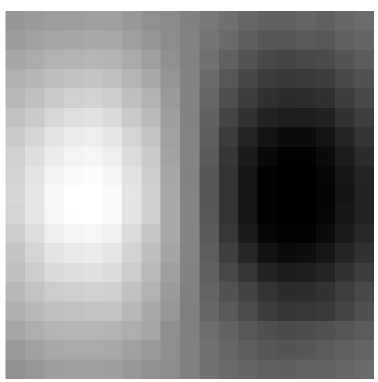
Derivative of Gaussian filter

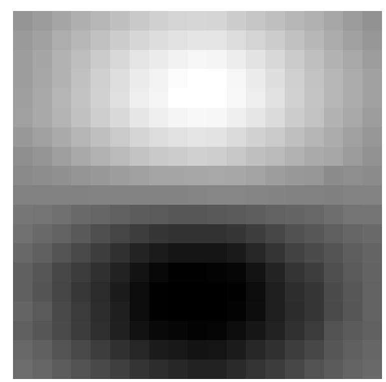


x-direction



y-direction





Implementation issues

Image

Gradient Magnitude

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Canny Edge Detector

Probably the most widely used edge detector in computer vision

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel 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’)

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

The Canny edge detector

- original image (Lena)

The Canny edge detector

magnitude of the gradient

The Canny edge detector

thresholding

The Canny edge detector

thinning

(non-maximum suppression)

Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

Source: L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

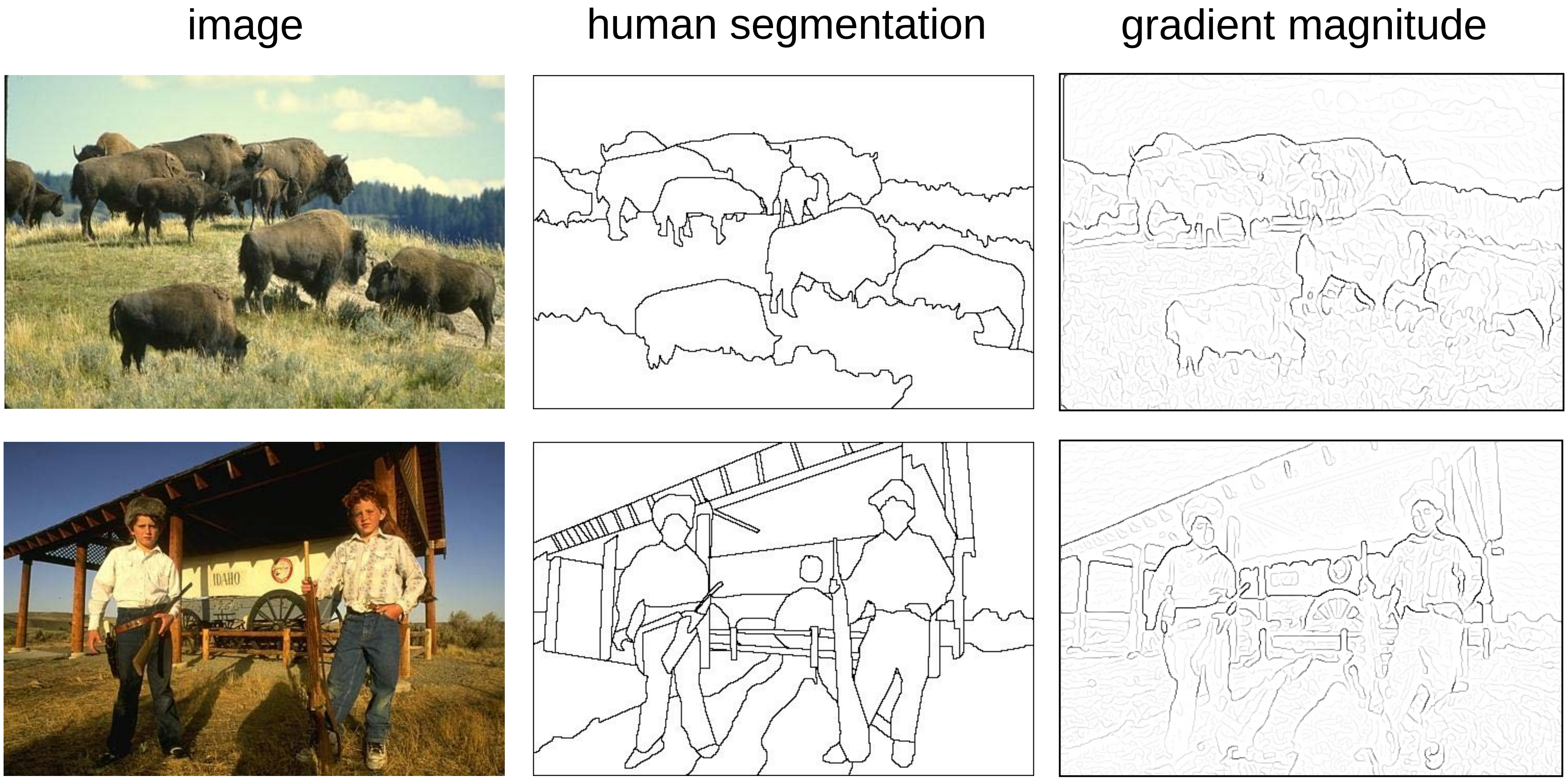
Canny with $\sigma = 2$

The choice of σ depends on desired behavior

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

Source: S. Seitz

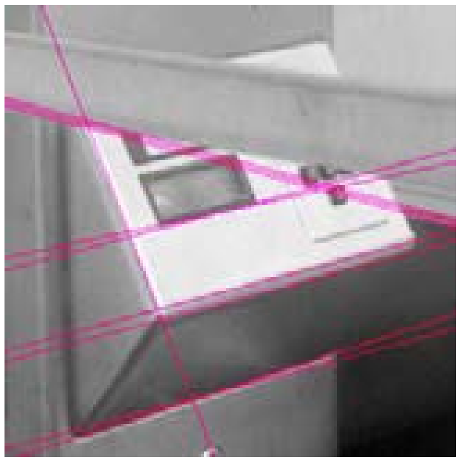
Edge detection is just the beginning…



- We want to go from edges to objects…
  - Higher-level, more compact representation of the features in the image
  - Grouping multiple features according to a simple model

Fitting

- Choose a parametric model to represent a set of features



simple model: lines

simple model: circles

complicated model: car

Fitting

- Membership criterion is not local
  - Can't tell whether a point belongs to a given model just by looking at that point
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features

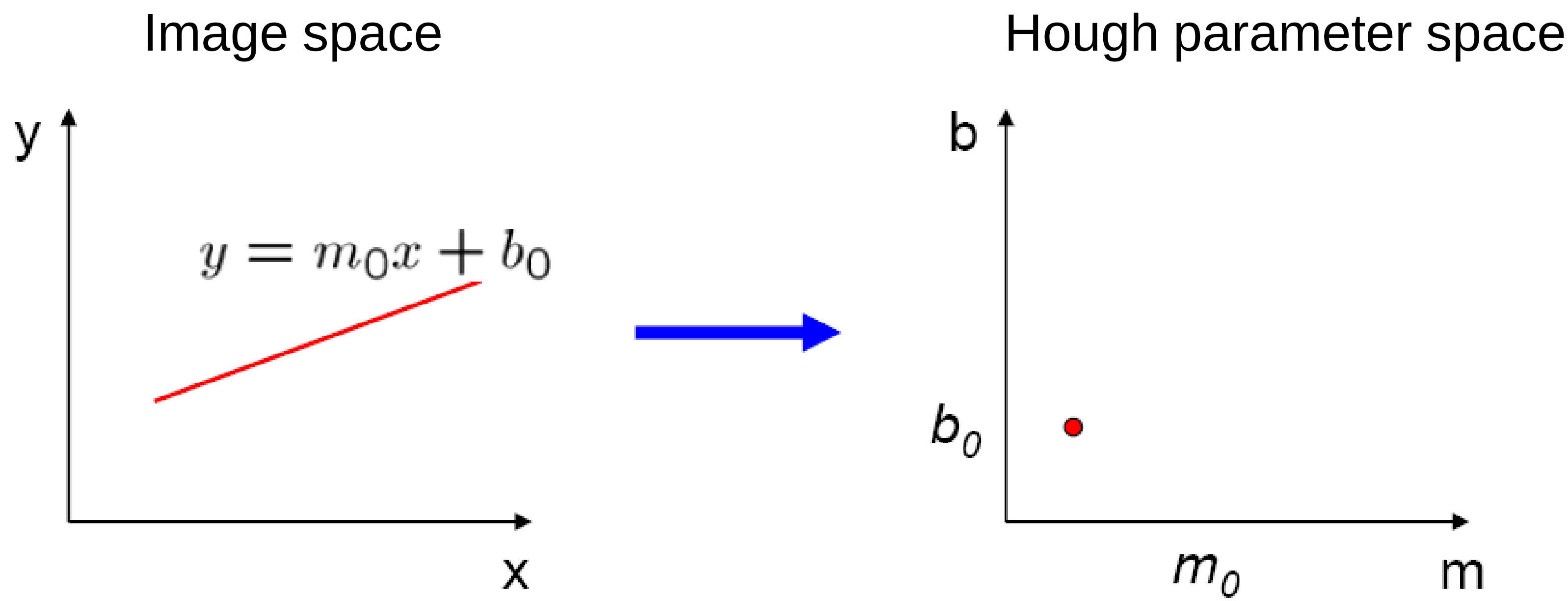
- Case Study: Line Fitting
  - Given edge points, find *real-world* lines in this image
  - Issues:
    - Noise in the measured feature locations
    - Clutter (outliers), multiple lines
    - Missing data: occlusions

Line Detection: Hough Transform

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model
- Missing data doesn’t matter as long as there are enough features remaining to agree on a good model

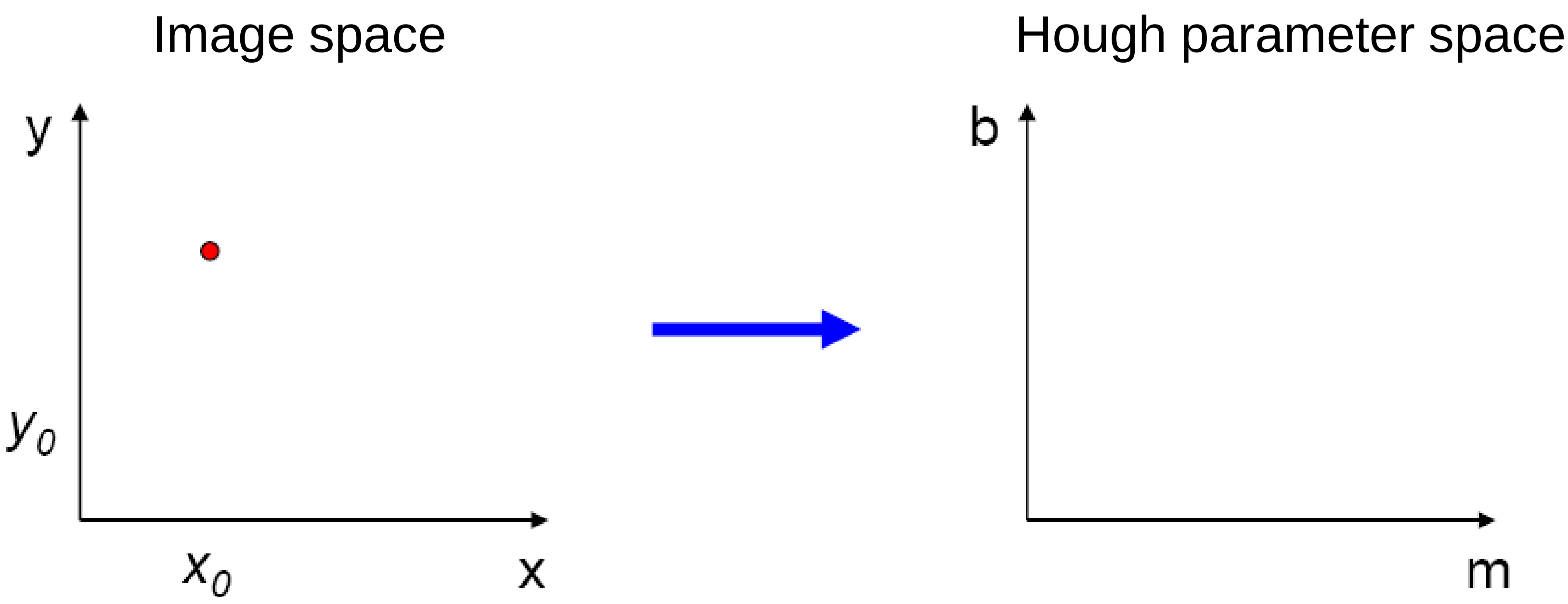
Parameter space representation

- A line in the image corresponds to a point in Hough space



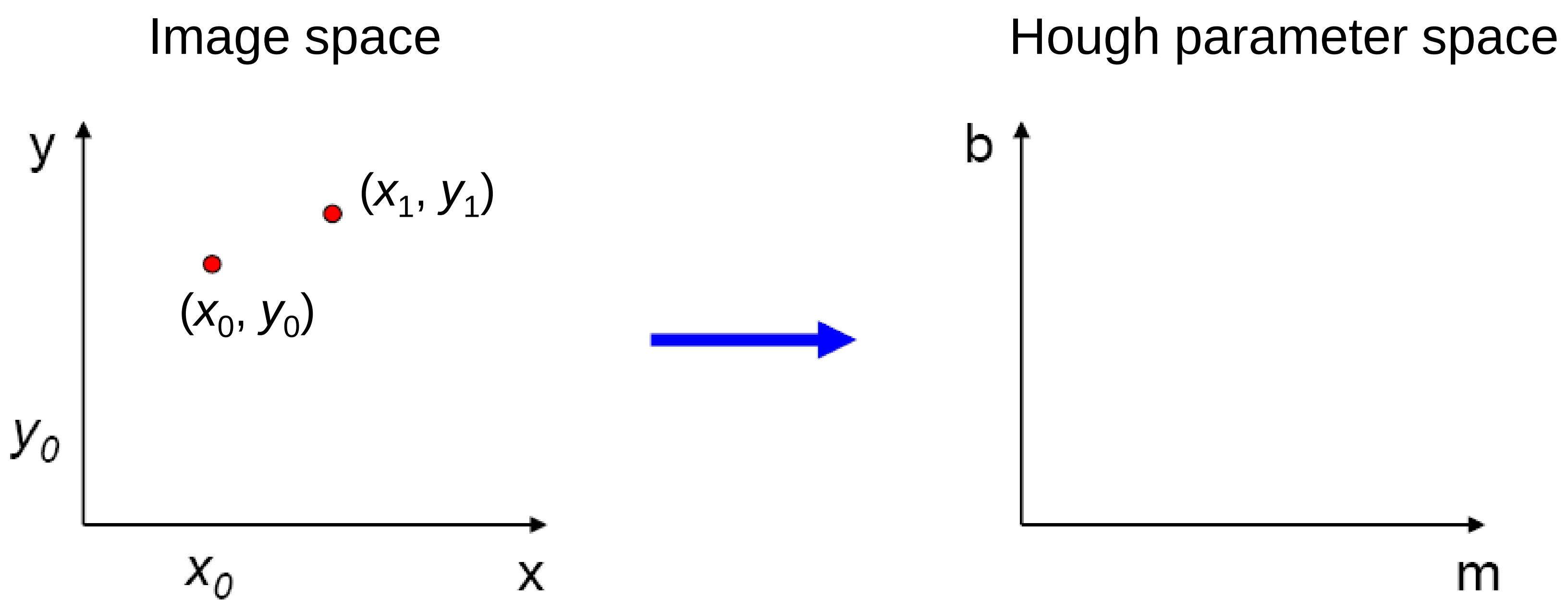
Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - This is a line in Hough space



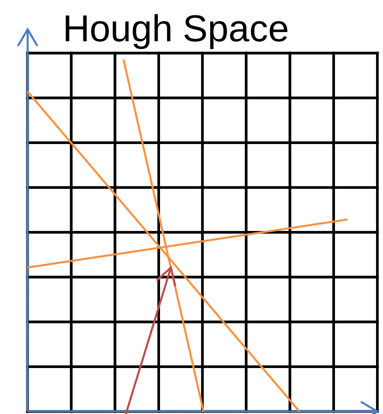
Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines $b = –x_0m + y_0$ and $b = –x_1m + y_1$



Basic Hough Transform

- Points which lie on the same line in image space will have lines which intersect at a point in Hough space
  - So, to find ***image line(s)***, we find the ***point(s)*** in Hough space where multiple Hough lines intersect.
  - Potential problem with this approach…
    - If the points don't lie *exactly* on the same line in image space, the lines in Hough space won't intersect
    - We can count intersections if they are "close" using by discretizing Hough space



Hough Accumulator

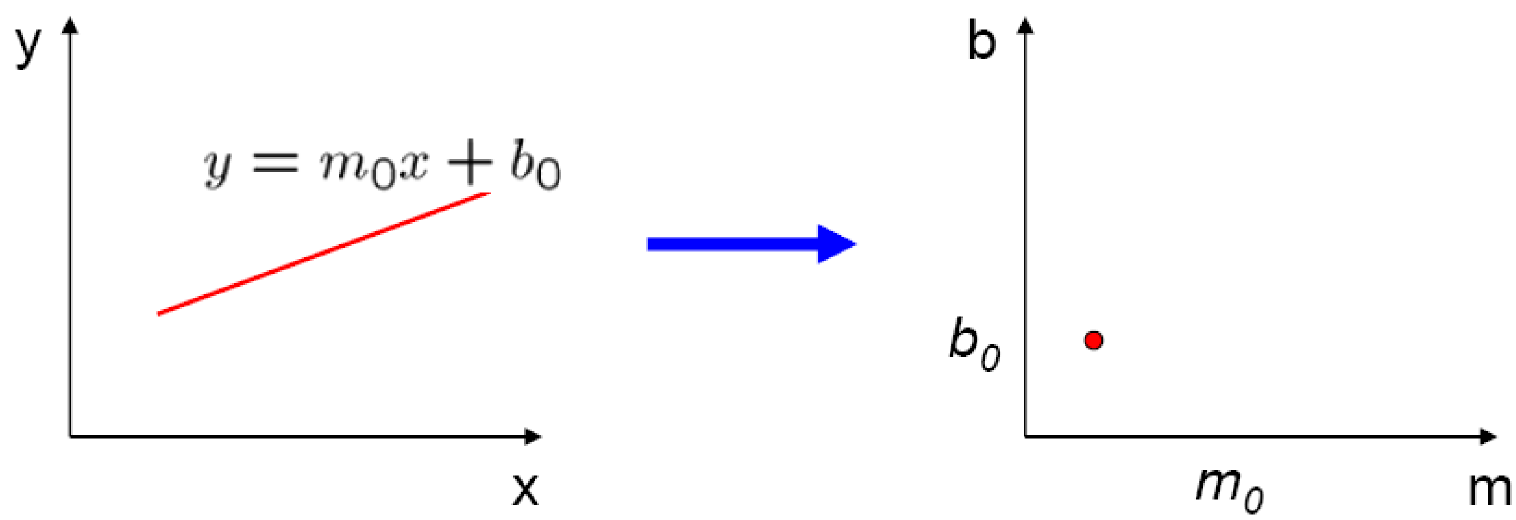
3 lines pass through this grid cell

More Problems

- What are some problems with the (m,b) Hough space:
  - Unbounded parameter domain
  - Vertical lines require infinite m

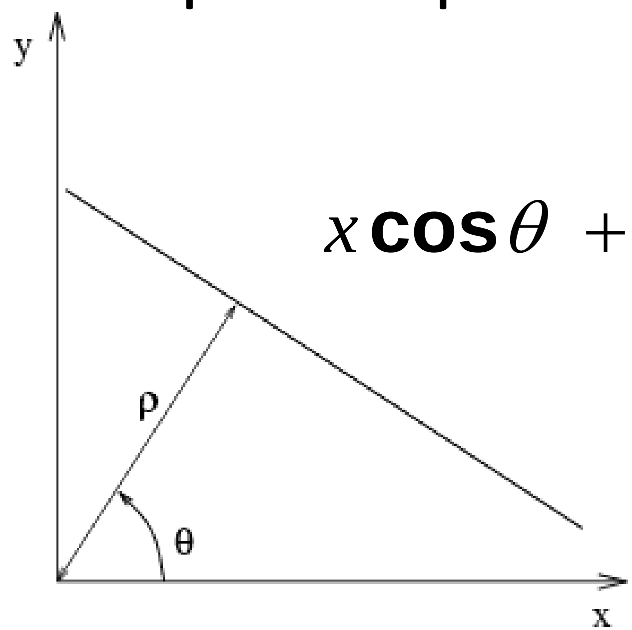
Image space

Hough parameter space



Parameter space representation

- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m
- Alternative: polar representation

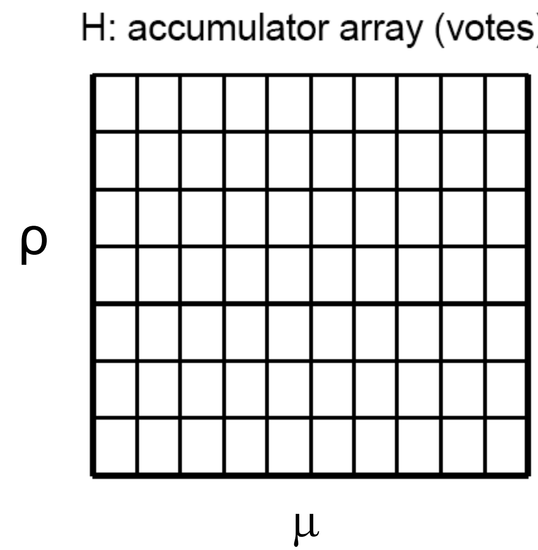


$$x\,\mathbf{cos}\,\theta + y\,\mathbf{sin}\,\theta = \rho$$

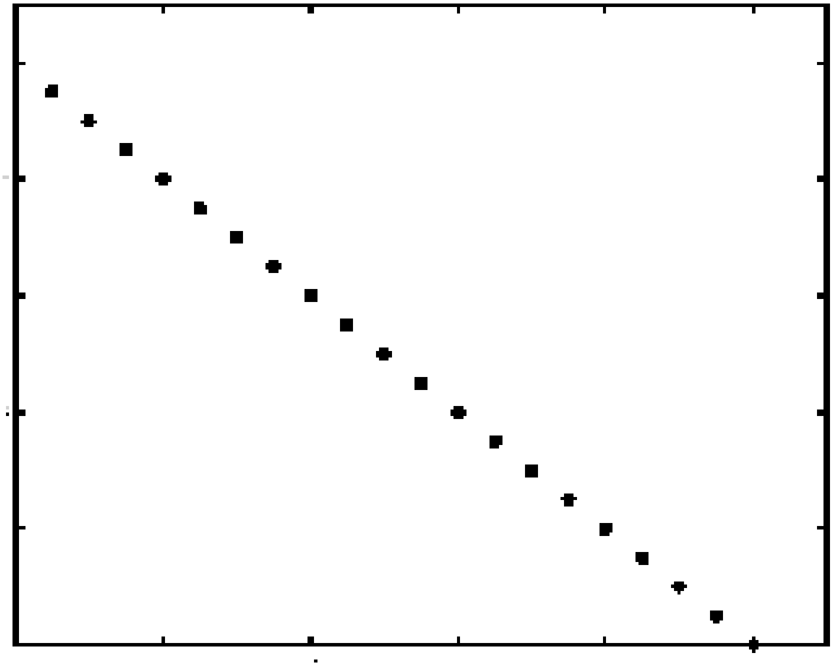
Each point will add a sinusoid in the (θ,ρ) parameter space

Algorithm outline

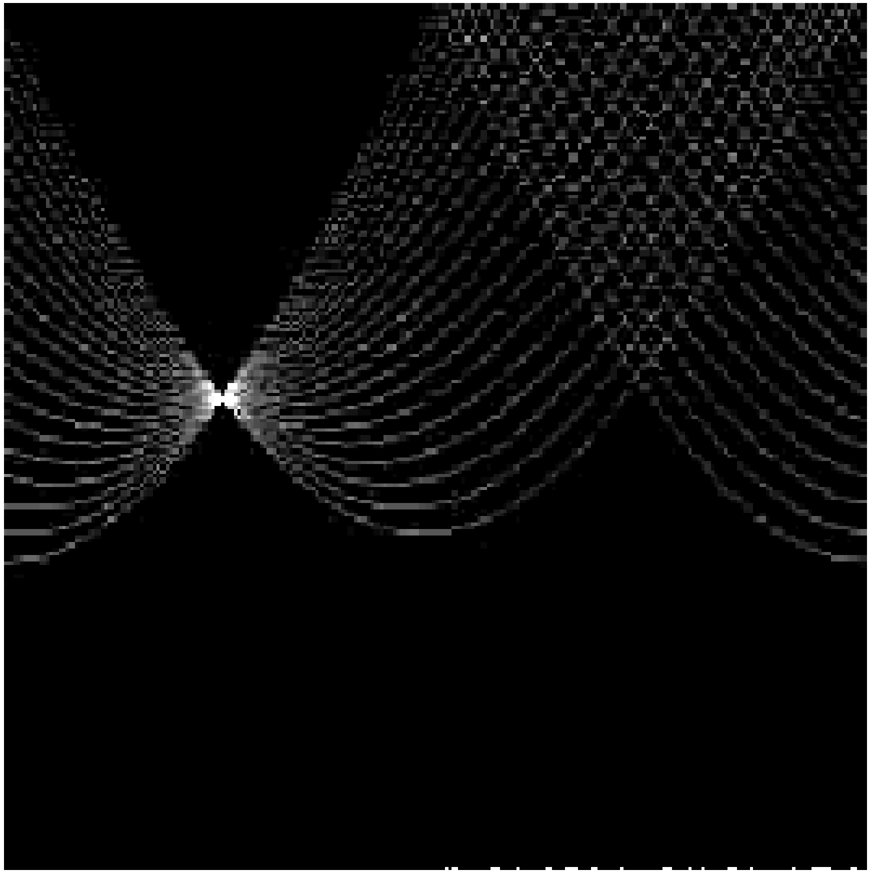
- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image
  For μ = 0 to 180
  $\rho = x \cos \mu + y \sin \mu$
  $H(\mu, \rho) = H(\mu, \rho) + 1$
  end
  end
- Find the value(s) of (μ, ρ) where $H(\mu, \rho)$ is a local maximum
  - The detected line in the image is given by $\rho = x \cos \mu + y \sin \mu$



Basic illustration



features

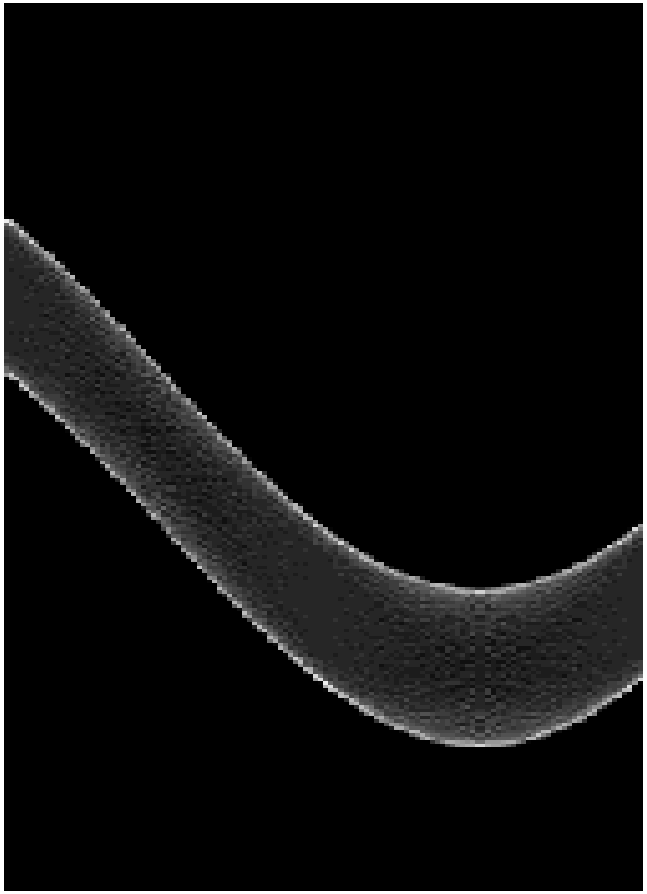


votes

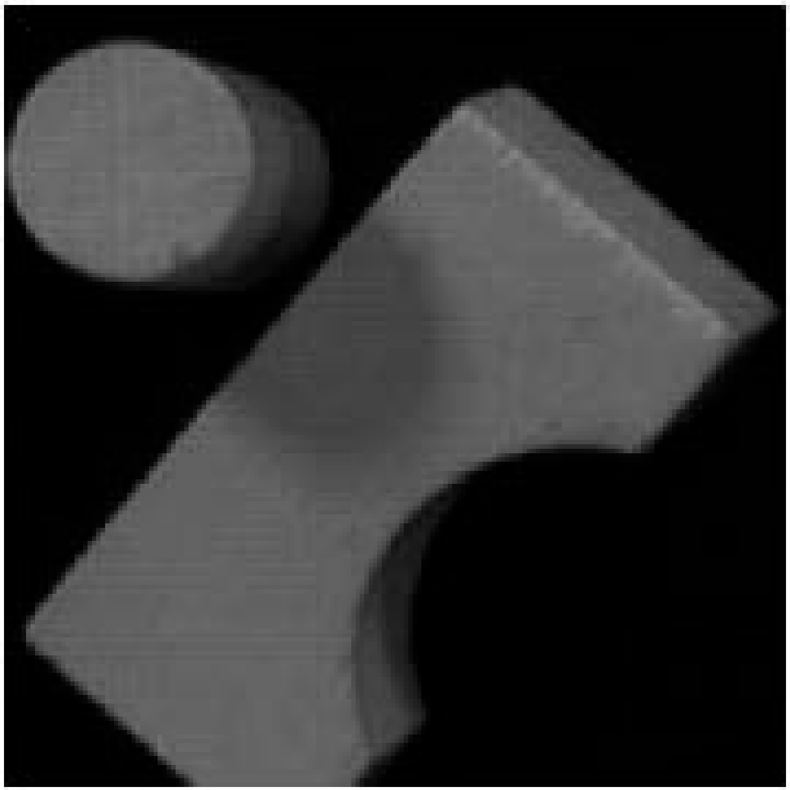
Other shapes in images

Square

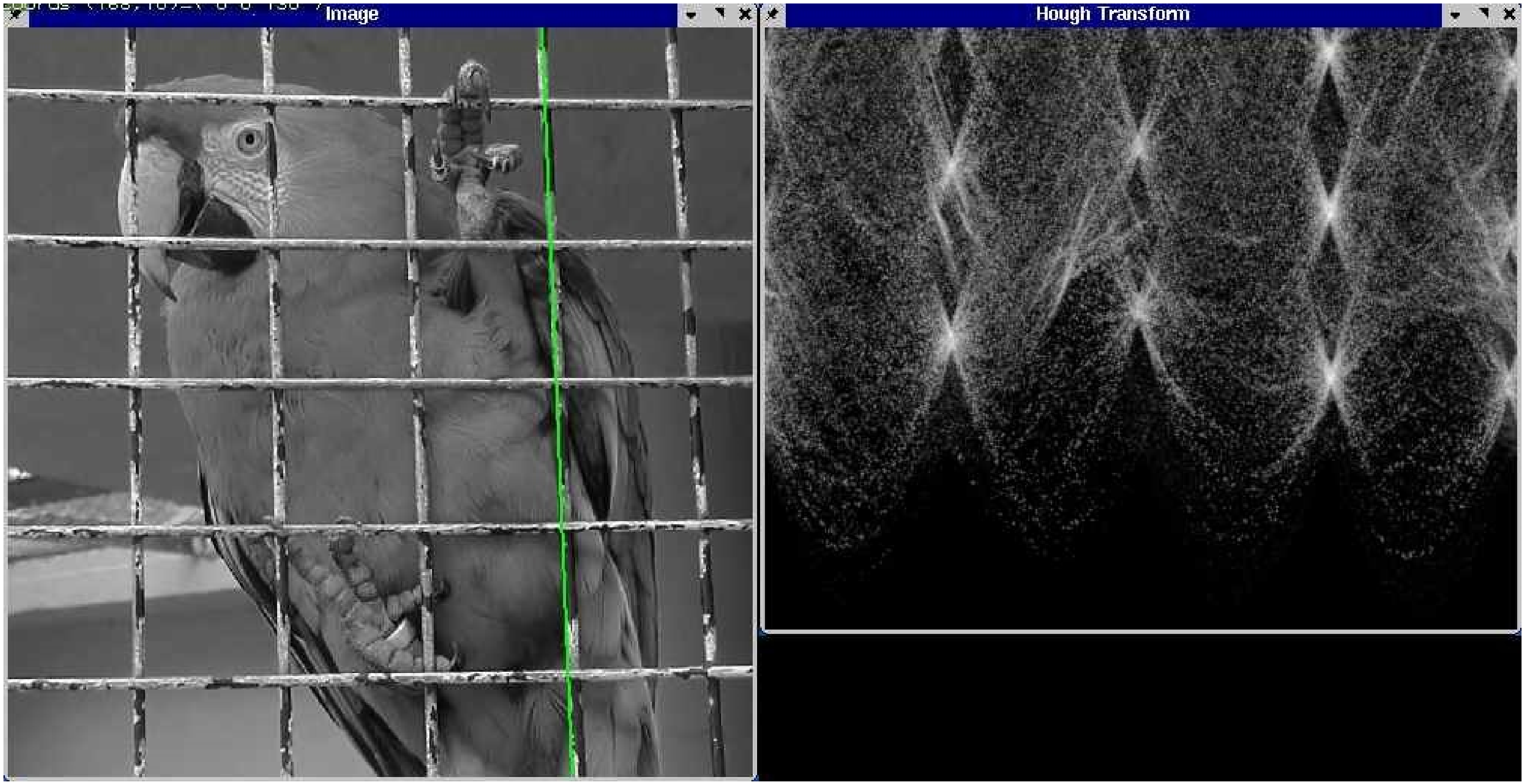
Circle



Several lines

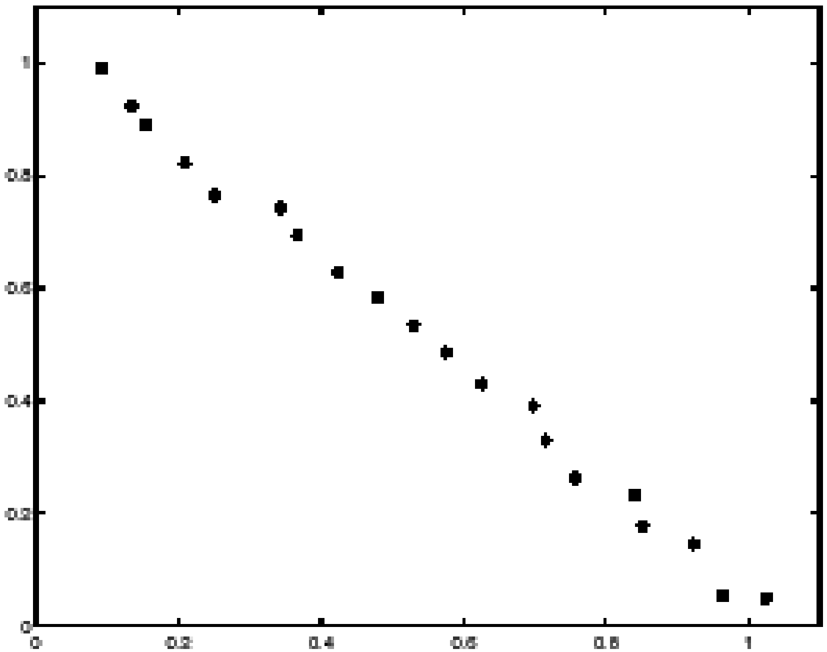


Real Image

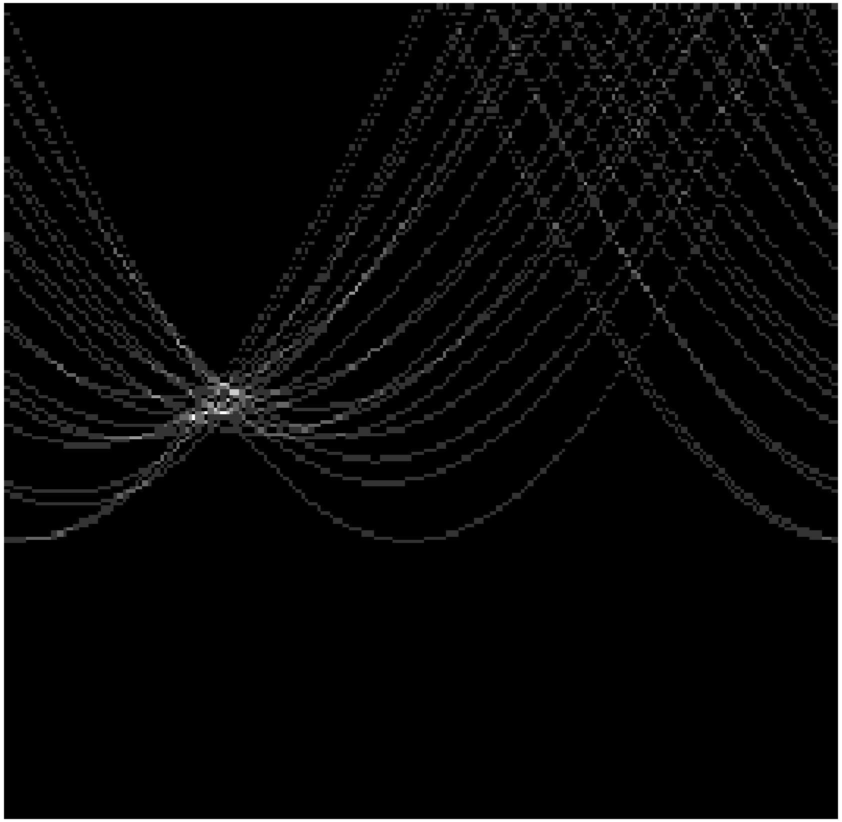


http://ostatic.com/files/images/ss_hough.jpg

Effect of noise



features

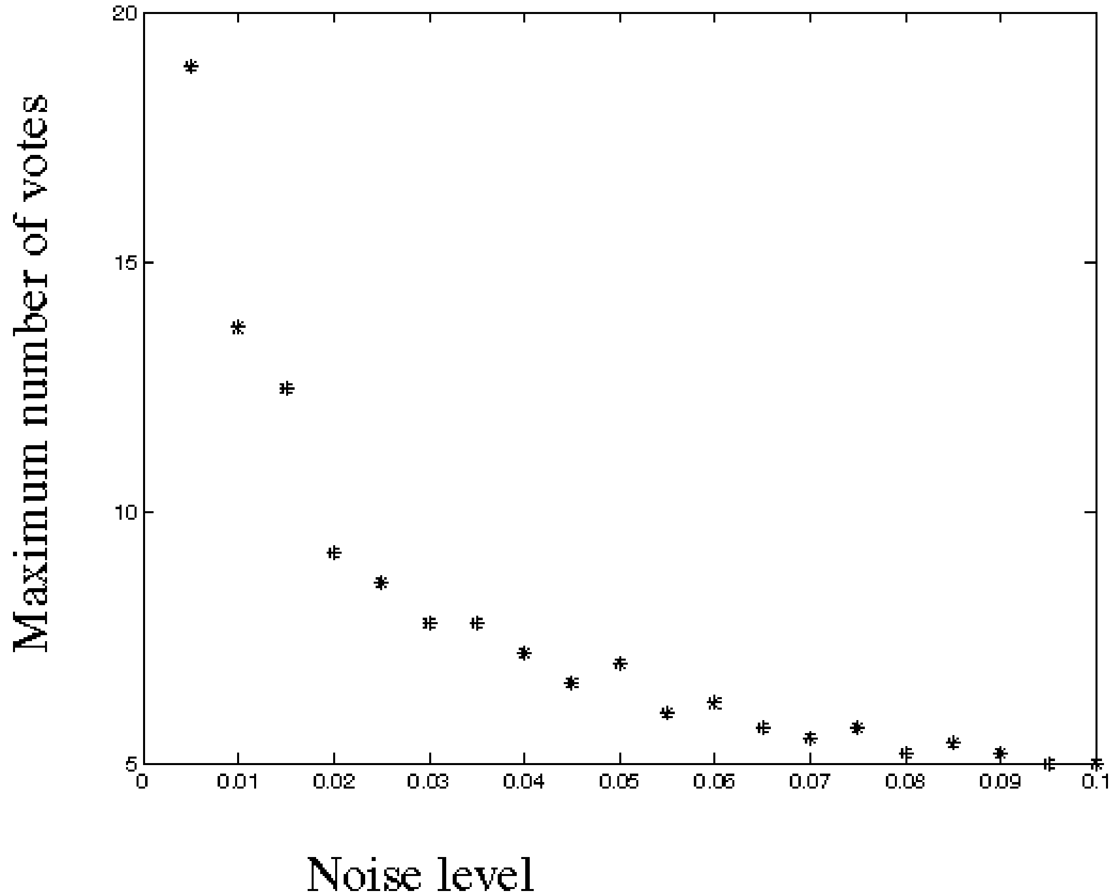


votes

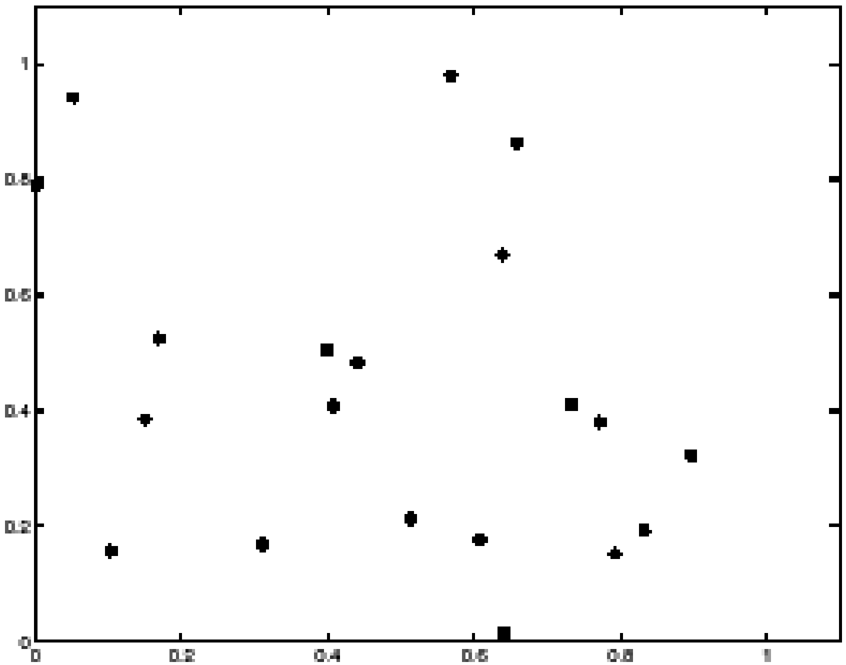
- Peak gets fuzzy and hard to locate

Effect of noise

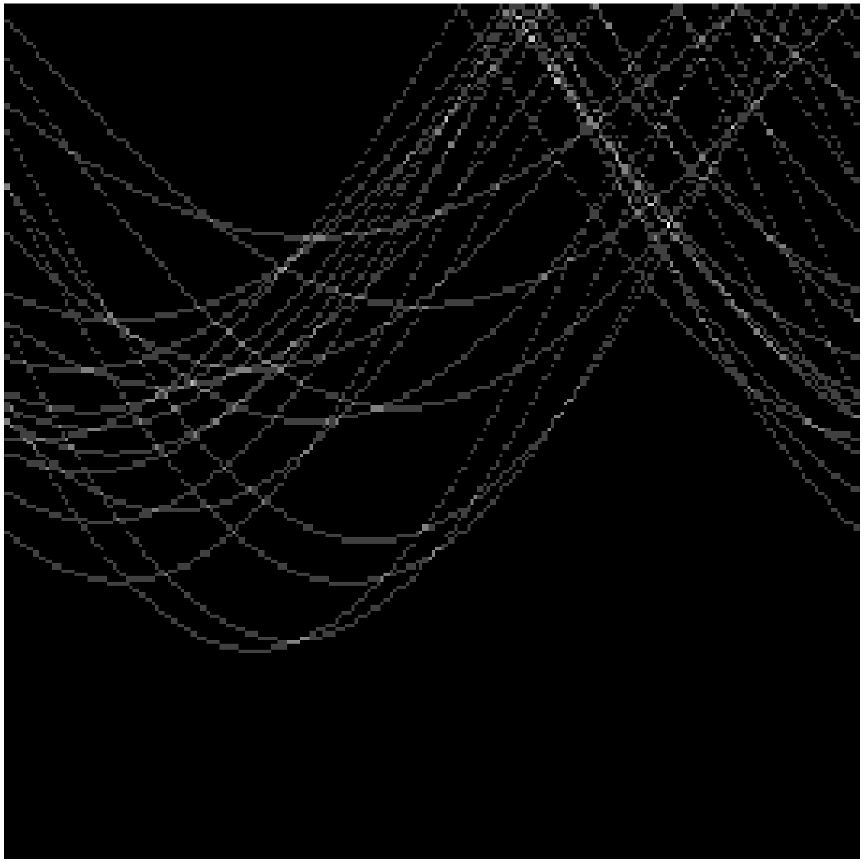
- Number of votes for a line of 20 points with increasing noise:



Random points



features

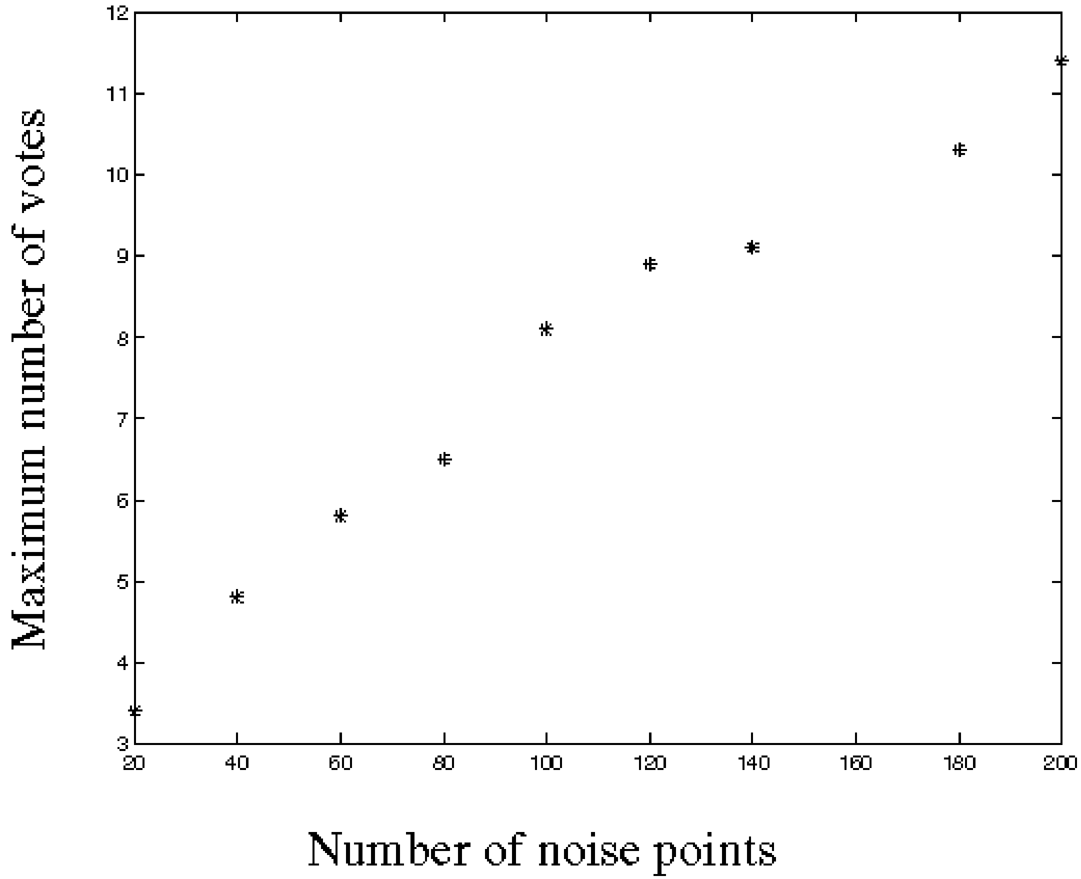


votes

- Uniform noise can lead to spurious peaks in the array

Random points

- As the level of uniform noise increases, the maximum number of votes increases too:



Practical details

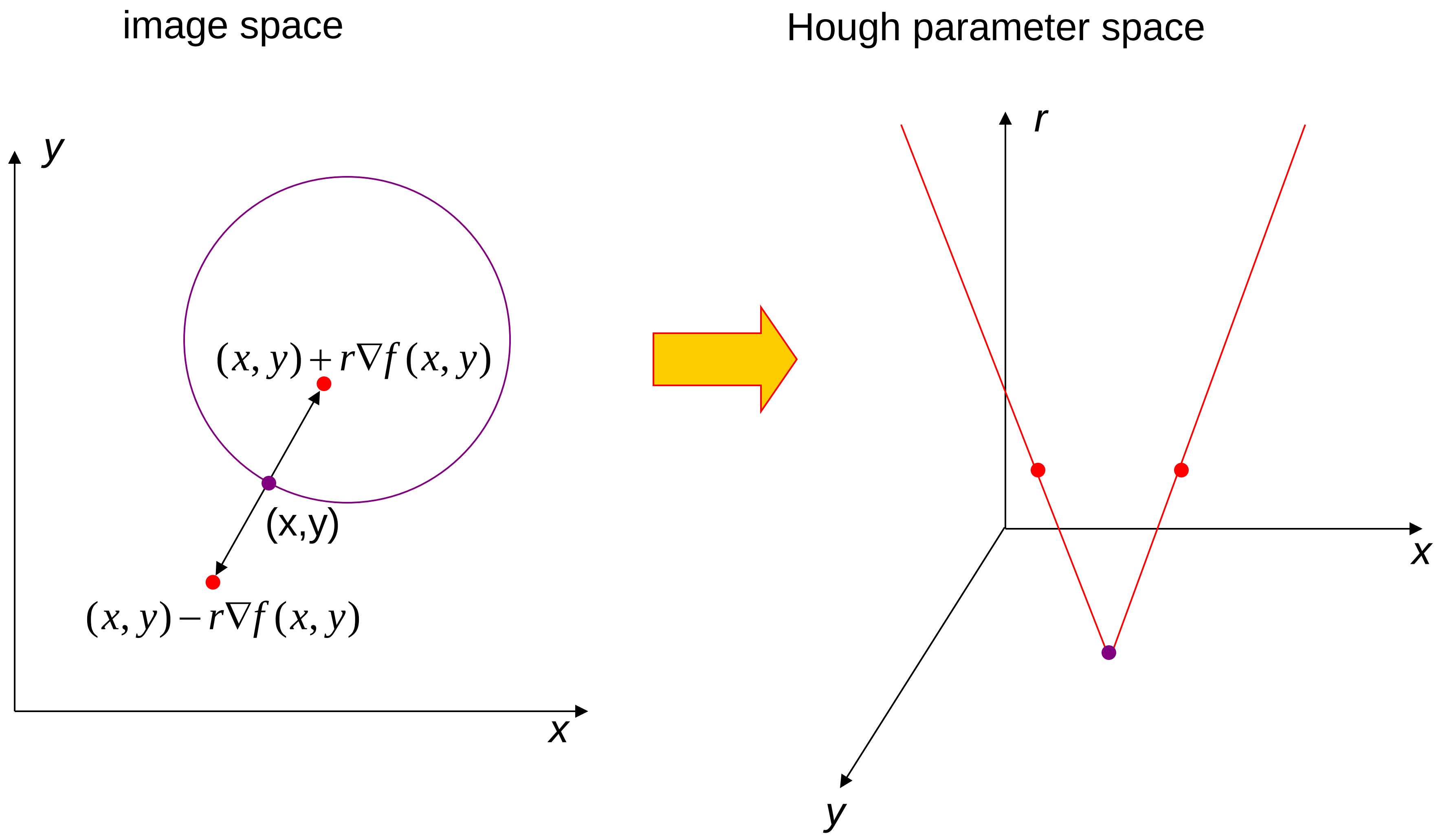
- Try to get rid of irrelevant features
  - Take only edge points with significant gradient magnitude
  - Use gradient magnitude to only vote for lines in perp. direction
- Choose a good grid / discretization
  - Too coarse: large votes obtained when too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Who belongs to which line?
  - Tag the votes

Hough Transform

- Not just for lines
  - Can be applied to *any* parametric model

- Hough Transform for Circles
  - How many dimensions will the parameter space have?
  - Given an oriented edge point, what are all possible bins that it can vote for?

Hough transform for circles



Hough transform: Pros

- Can deal with non-locality and occlusion
- Can detect multiple instances of a model in a single pass
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin

Hough transform: Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- It’s hard to pick a good grid size

Things to remember

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

- Hough Transform = points vote for shape parameters