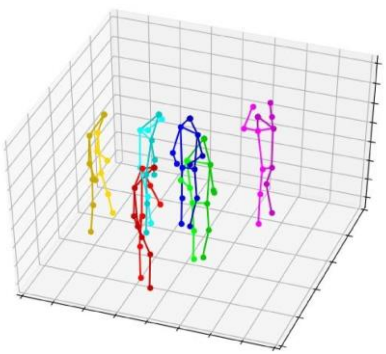
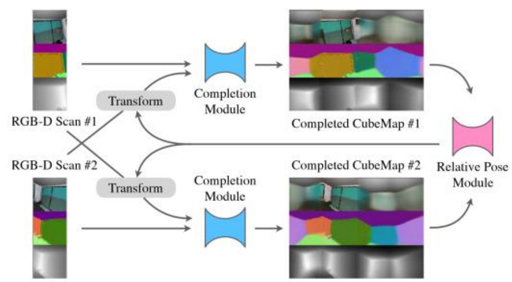
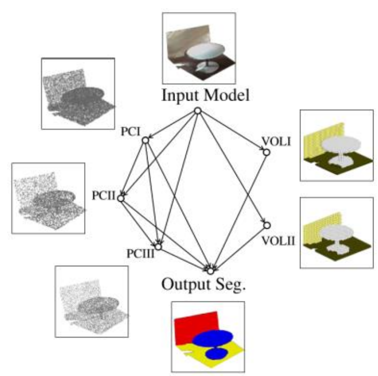
CS376 Computer Vision
Lecture 7: Hough Transform





Qixing Huang
Sep. 12th 2023



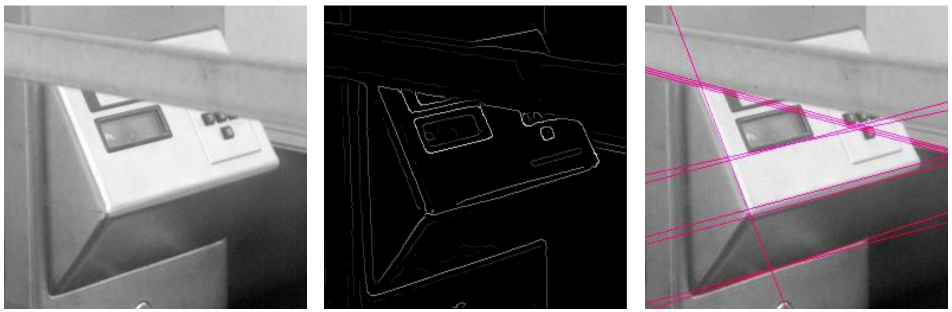
Review

- Image filters
- Edge detection
- Binary image analysis
- Texture
- Optical Flow

Local analysis

Now: 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.

Slide Credit: Kristen Grauman

Many Applications

- Vanishing point detection
- Segmentation/Detection
- 3D Vision
  - Calibration
  - Structure-from-motion

Fitting: Main Idea

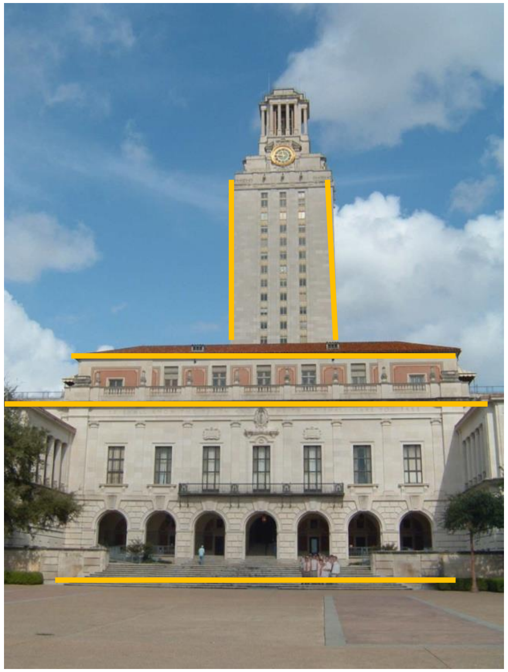
- Choose a parametric model to represent a set of features

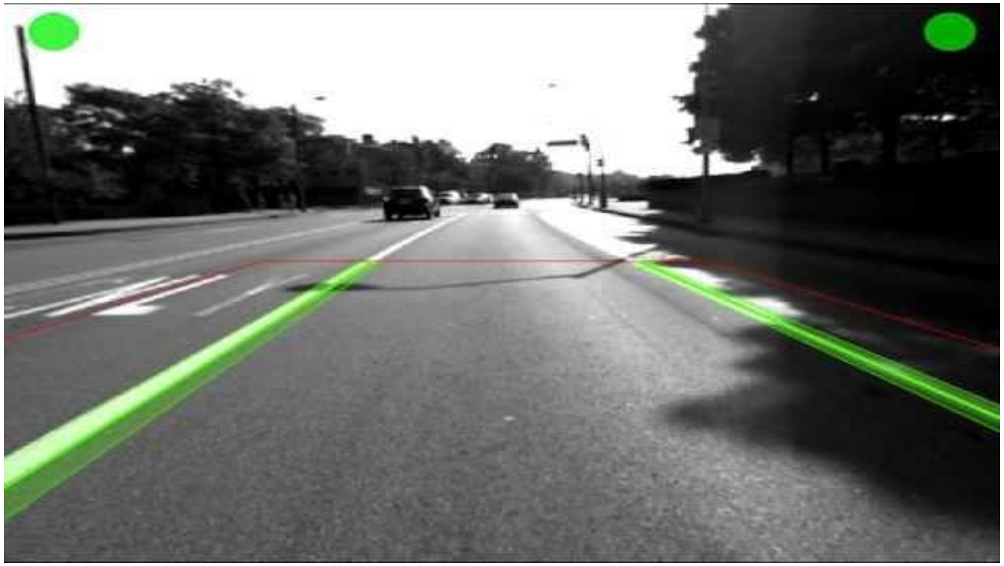
- Correlated problems
  - What are the models
  - Association between models and features
  - How to optimize the models

Case study: Line fitting

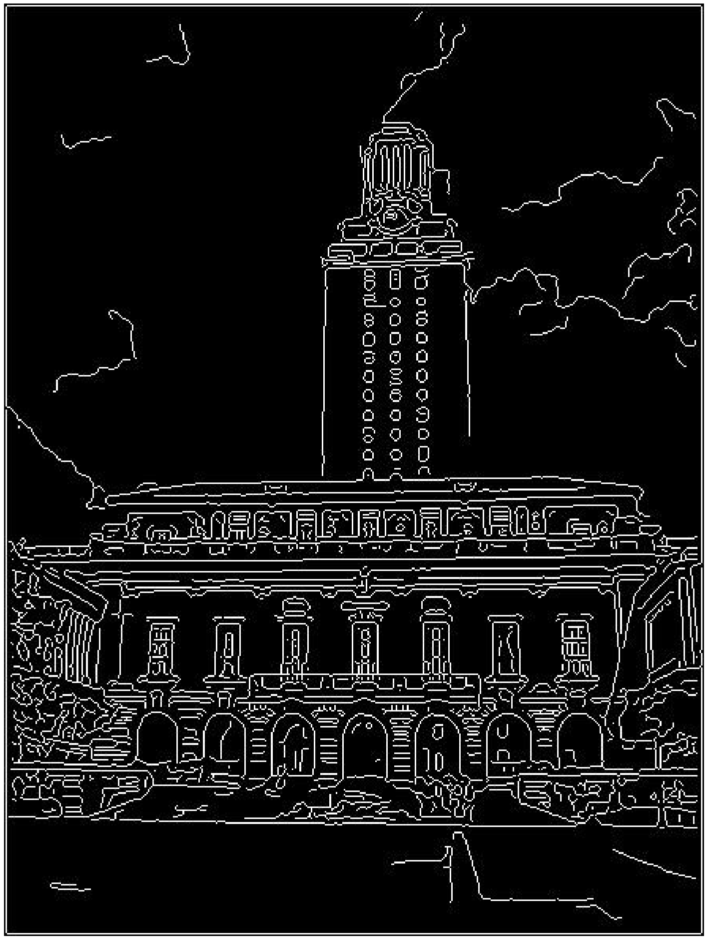
- Why fit lines?

  Line features are quite popular in natural images





Difficulty of line fitting



- Incomplete edge detections
- How many lines
- Not all edges are lines
- Noise in detected edges

Voting

- Impossible to test all combinations of features to extract the models

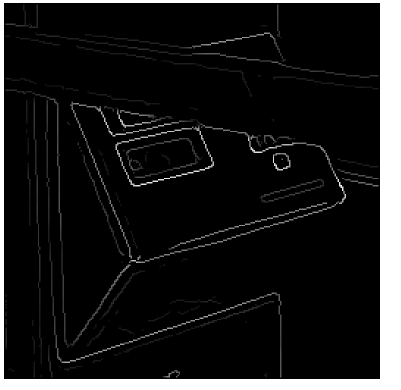
- Let features vote for the models
  - Cycle through features, cast votes for model parameters
  - Usually each model should be low-dimensional

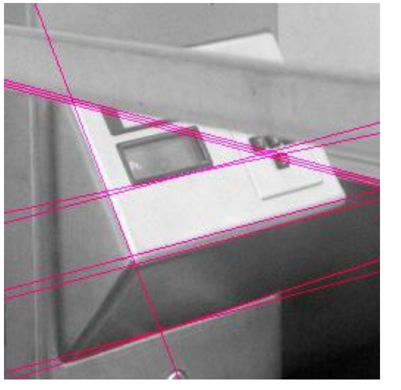
- Noise contribute less to the models

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:
  - Record vote for each possible line on which each edge point lies
  - Look for lines that get many votes





Basic Facts

- Not all the votes are correct, but the correct ones form ‘clusters’
- Depend on the representations of the models
- Depend on how we fit the models

Finding lines in an image: Hough space

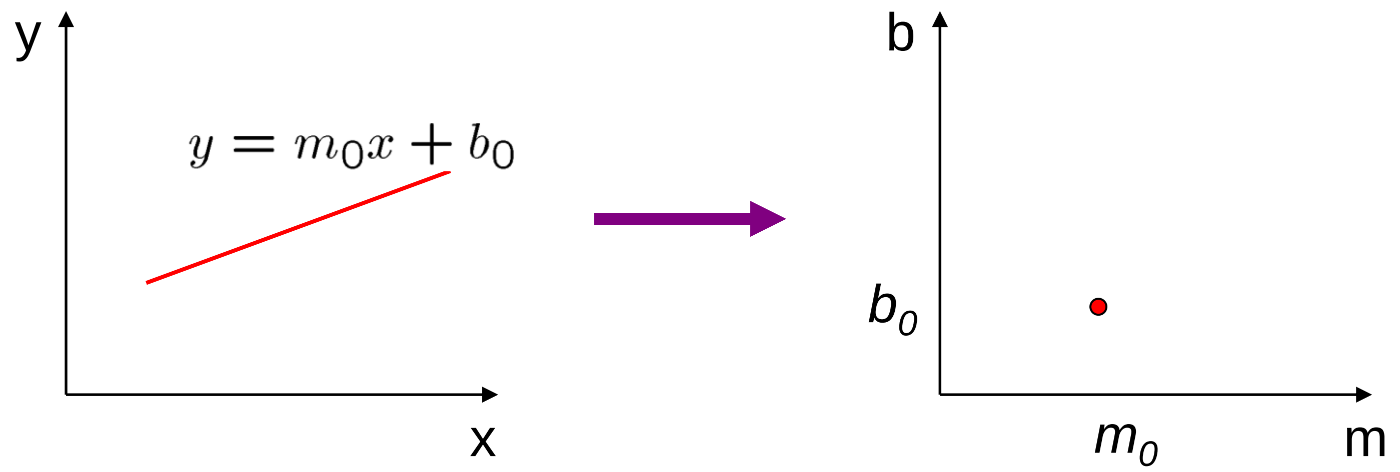


Image space

Parameter space

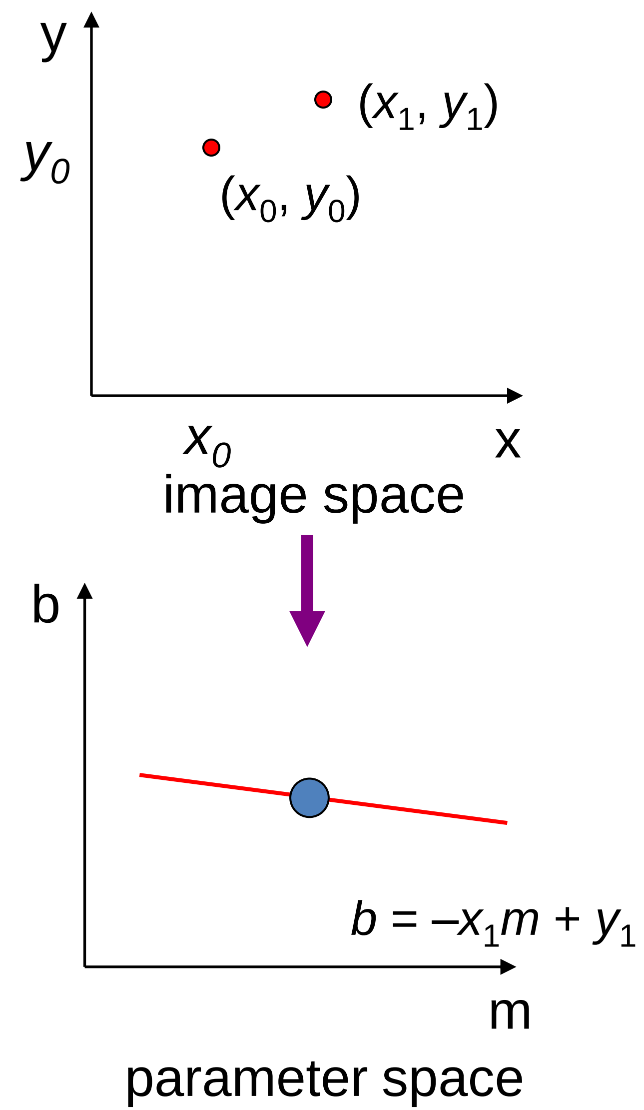
Connection between image (x,y) and parameter (m,b) spaces

- 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
  - This process is repeated many times

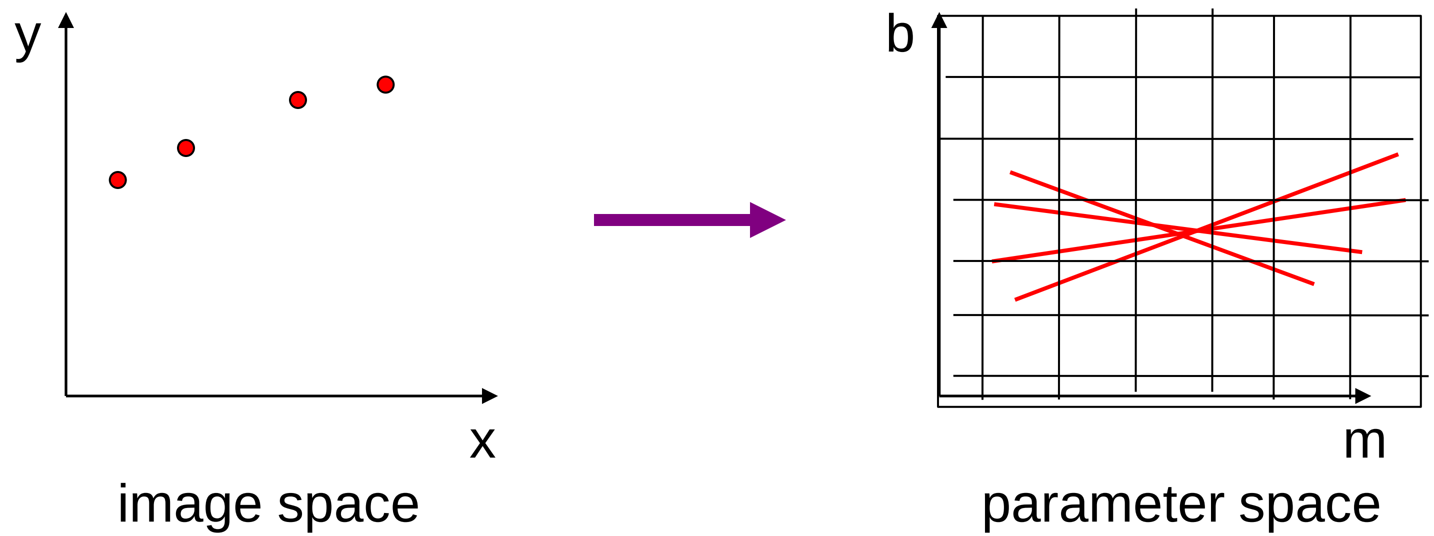
Going from point pairs to lines

- Each point in the image space corresponds to a line in the parameter space

- The lines that pass through two points in the image space corresponds to a point, which is the intersection of these two lines



Finding lines in an image: Hough algorithm



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

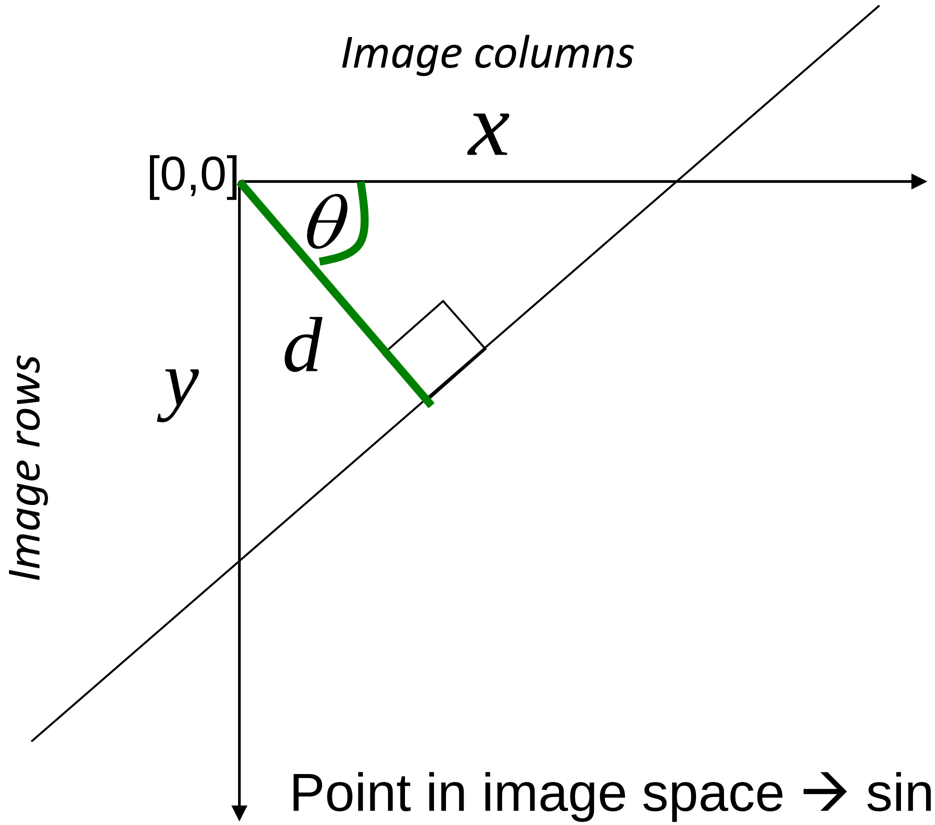
- Let each edge point in image space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins*; parameters with the most votes indicate line in image space.

Slide Credit: Kristen Grauman

Use a different representation

Finding lines in an image: Hough algorithm

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

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

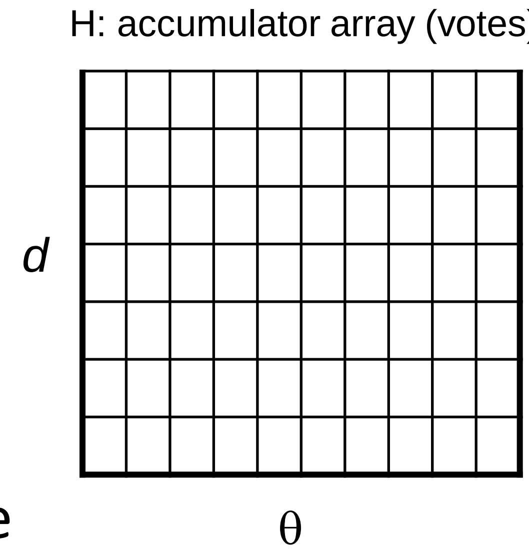
Point in image space → sinusoid segment in Hough space

Slide Credit: Kristen Grauman

Hough transform algorithm

Using the polar parameterization:

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



Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

   for θ = [θ_{min} to θ_{max}] // some quantization

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

   H[d, θ] += 1

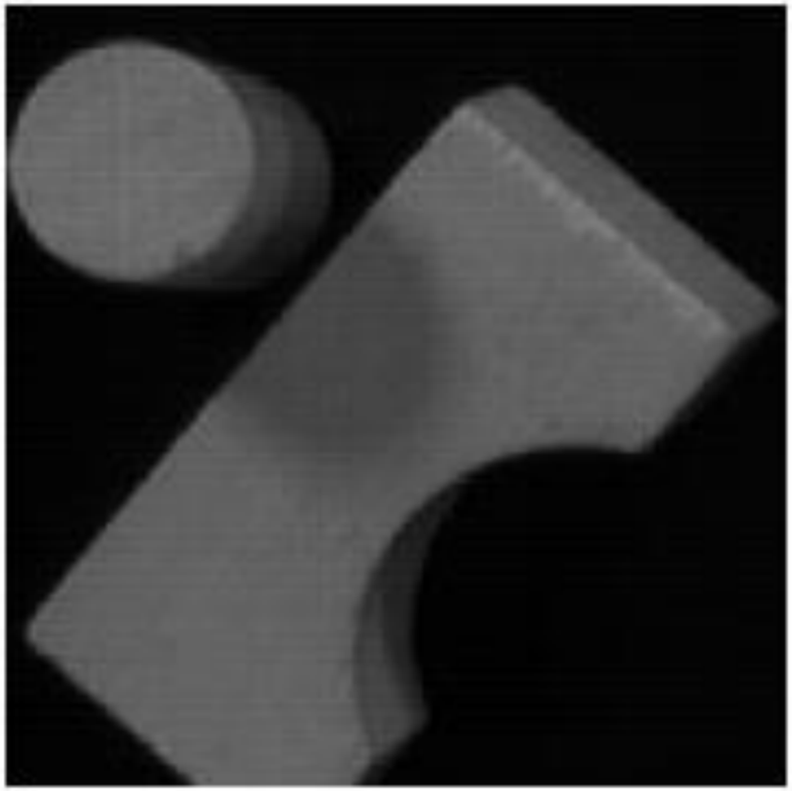
$$d = x\cos\theta - y\sin\theta$$

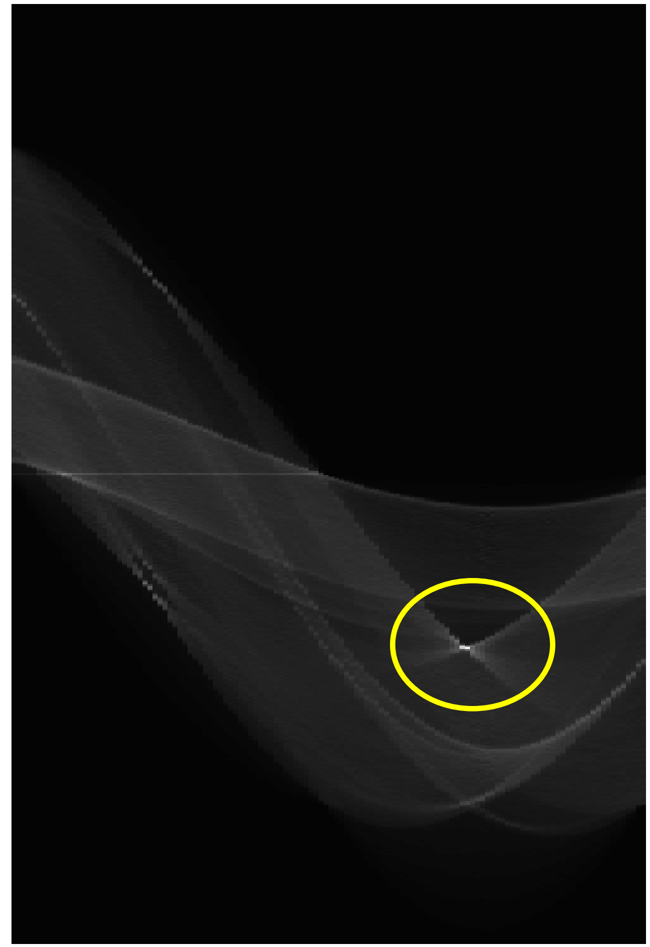
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by

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

Source: Steve Seitz

Example: Hough transform for straight lines



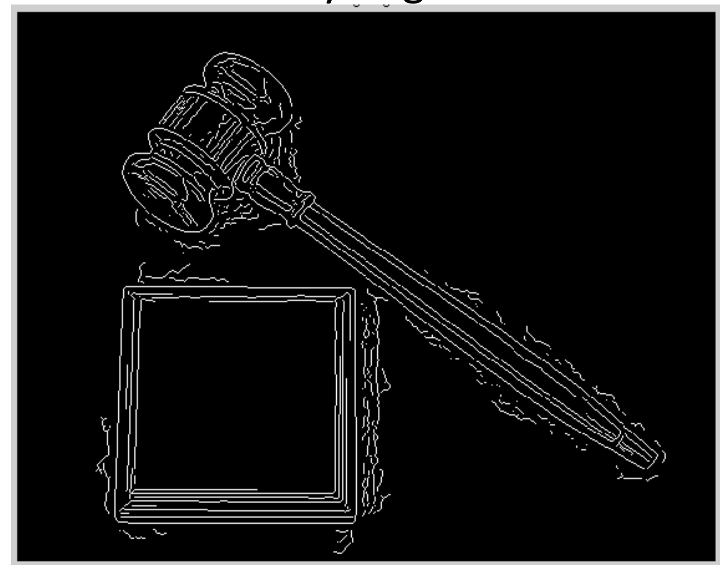


Which line generated this peak?

Slide Credit: Kristen Grauman

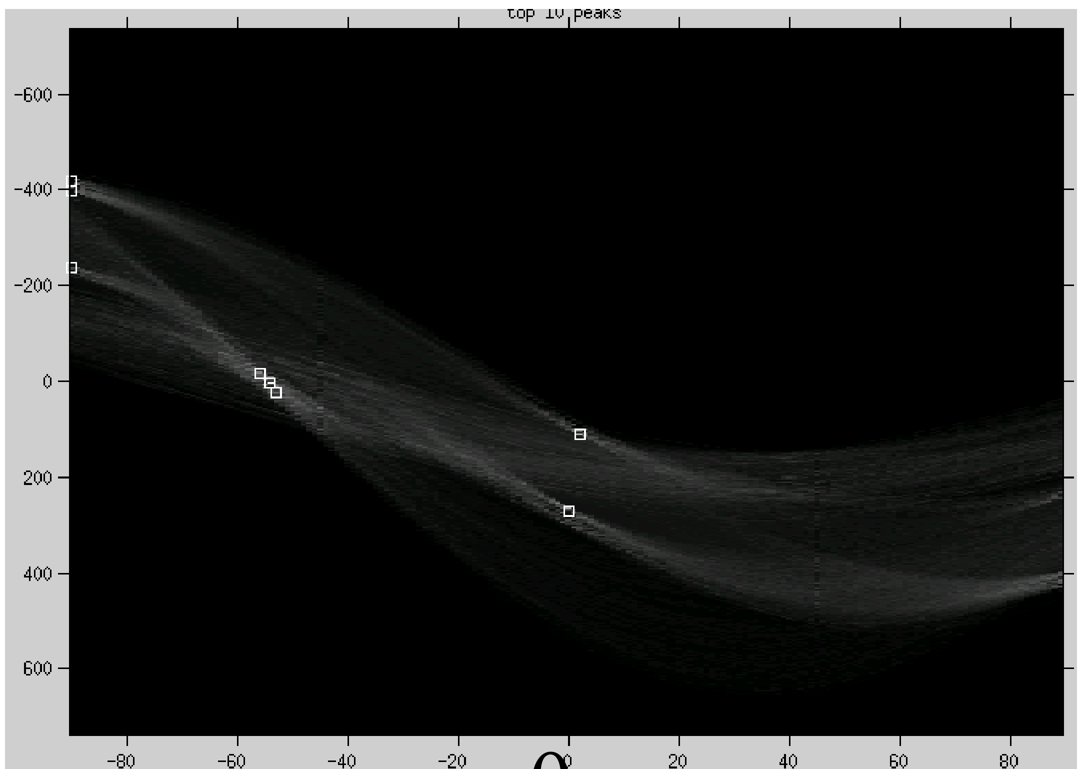
Original image

Canny edges

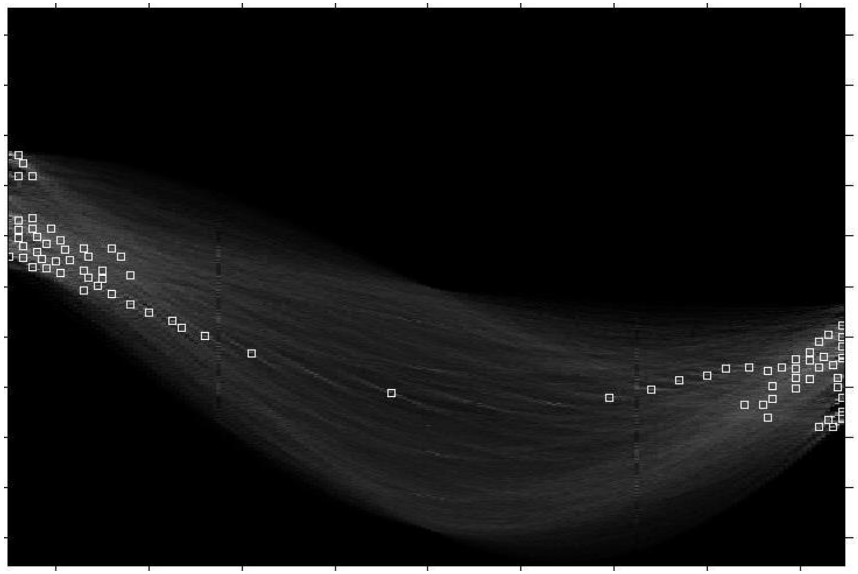


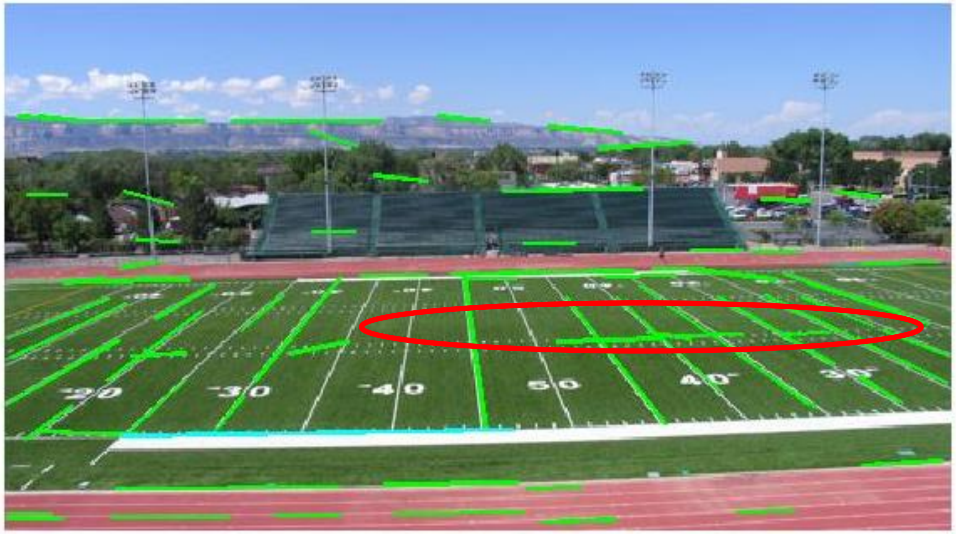
Decode the vote space.

d



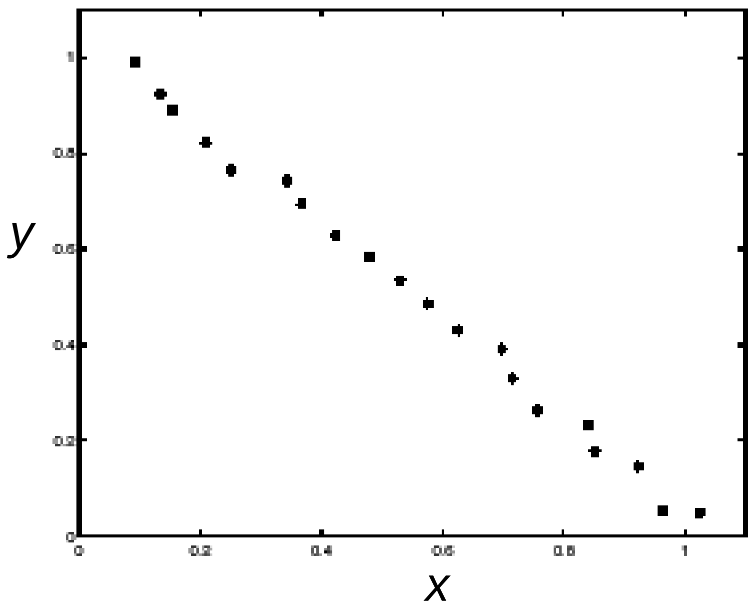
θ





Showing longest segments found

Impact of noise on Hough



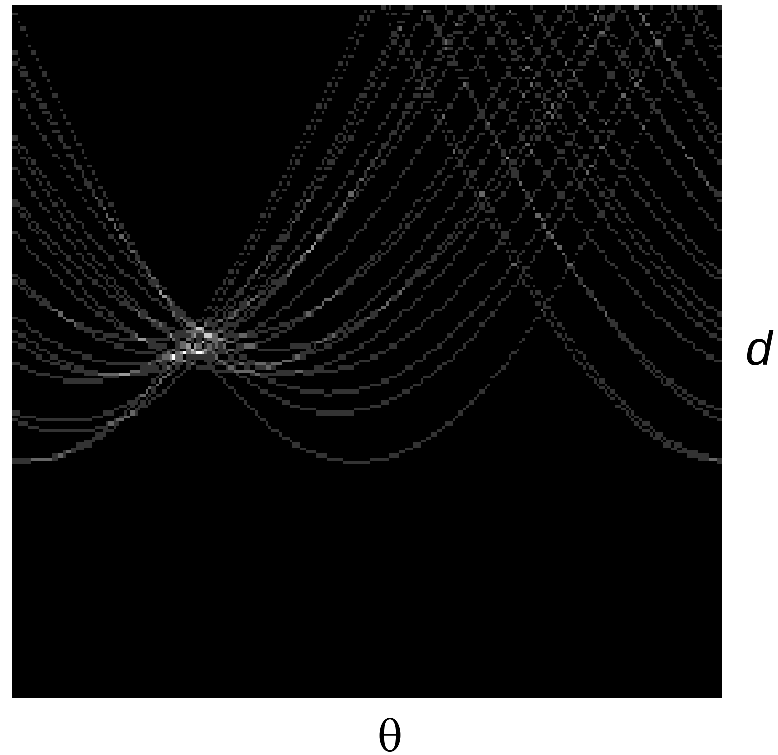


Image space
edge coordinates

Votes

What difficulty does this present for an implementation?

Impact of noise on Hough

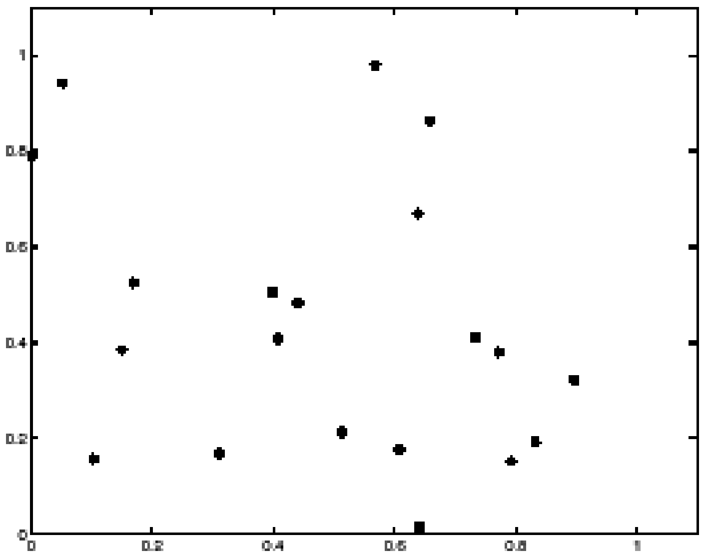
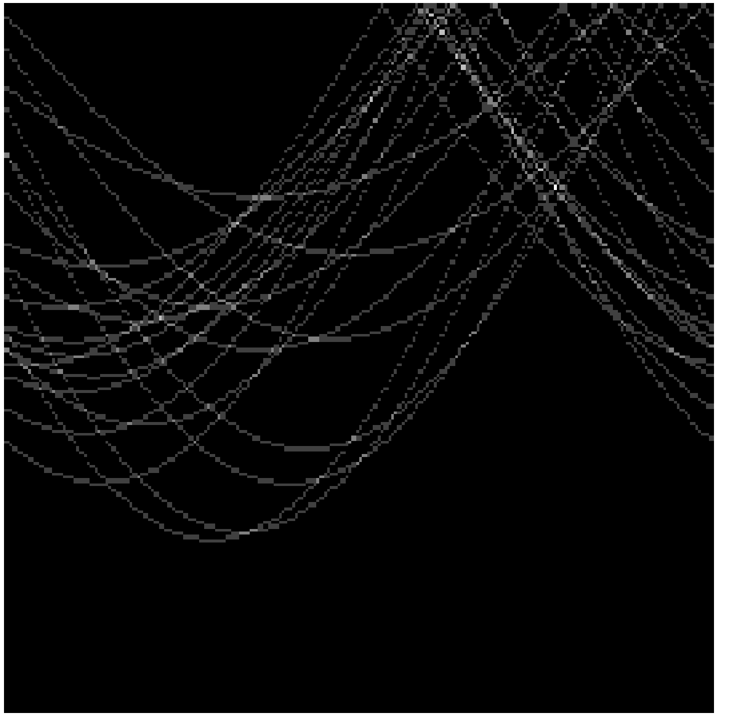


Image space edge coordinates



Votes

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

Extensions

Extension 1: Use the image gradient

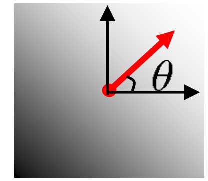
1. same
2. for each edge point I[x,y] in the image

   θ = gradient at (x,y)

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

   H[d, θ] += 1
3. same
4. same

(Reduces degrees of freedom)



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

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

Slide Credit: Kristen Grauman

Other extensions

- More votes for points on edges with large image gradients

- Vote for point pairs

- Before voting, check the candidacy of each point pair, e.g., distances to the line that pass through these two points

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

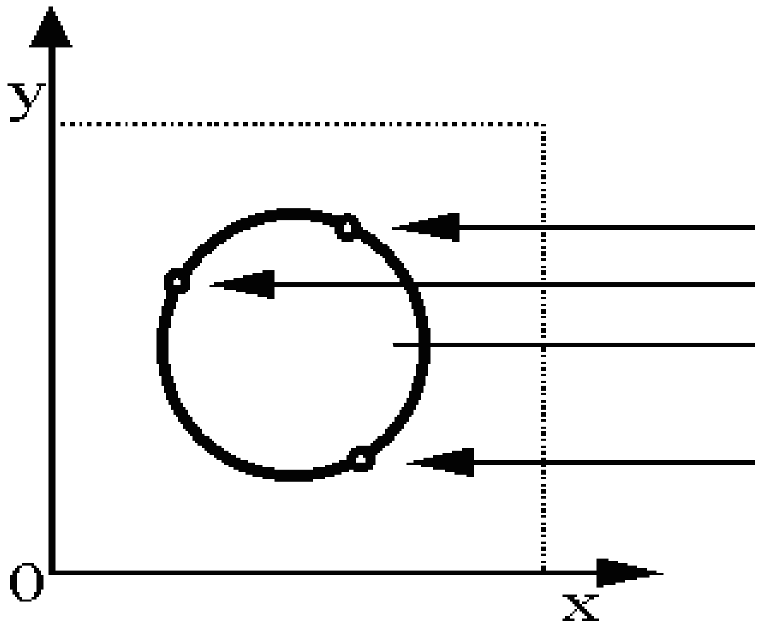
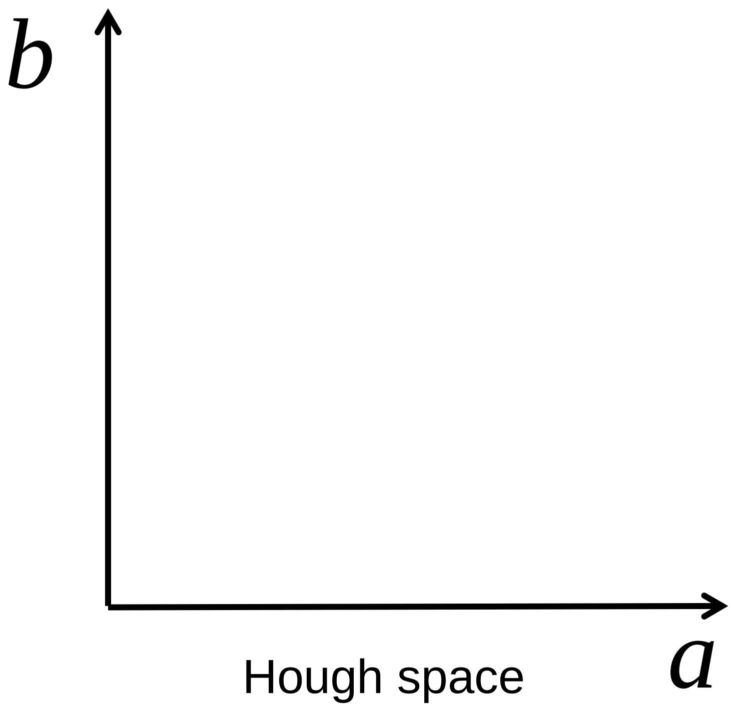


Image space



Hough space

Slide Credit: Kristen Grauman

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

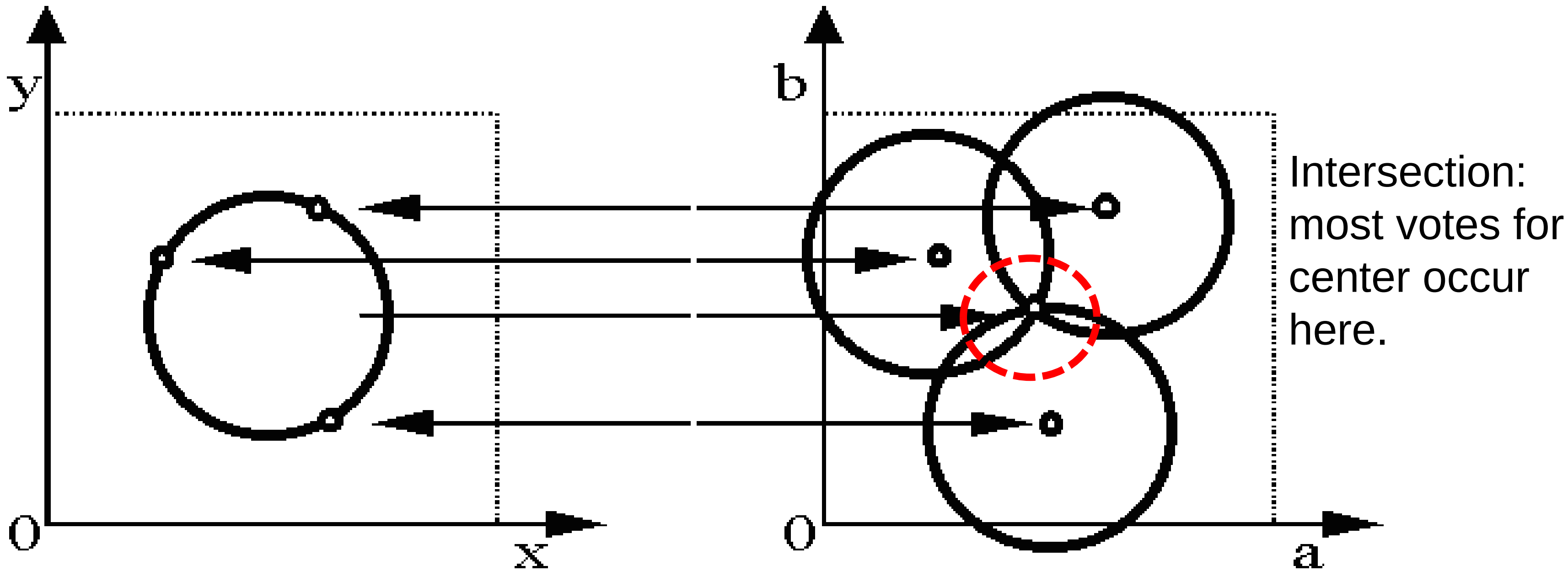


Image space

Hough space

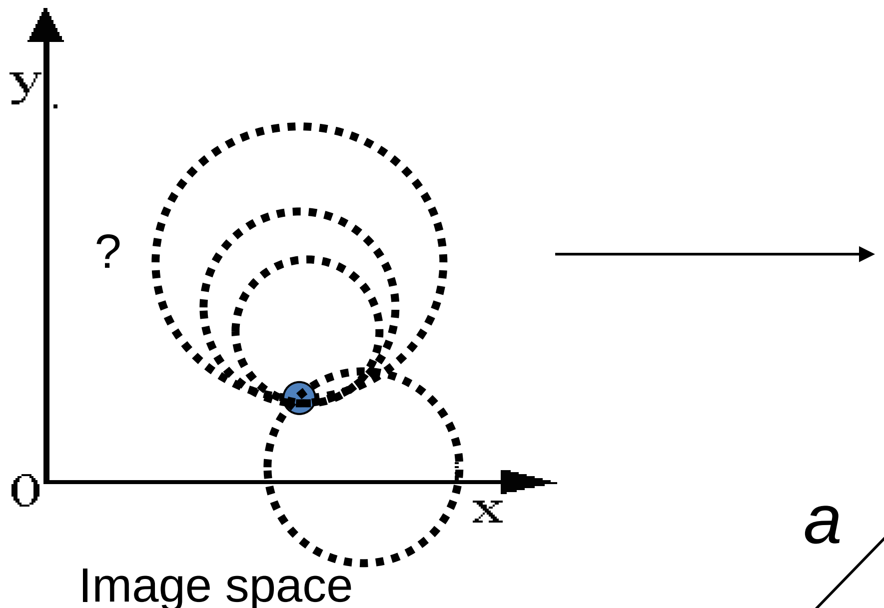
Slide Credit: Kristen Grauman

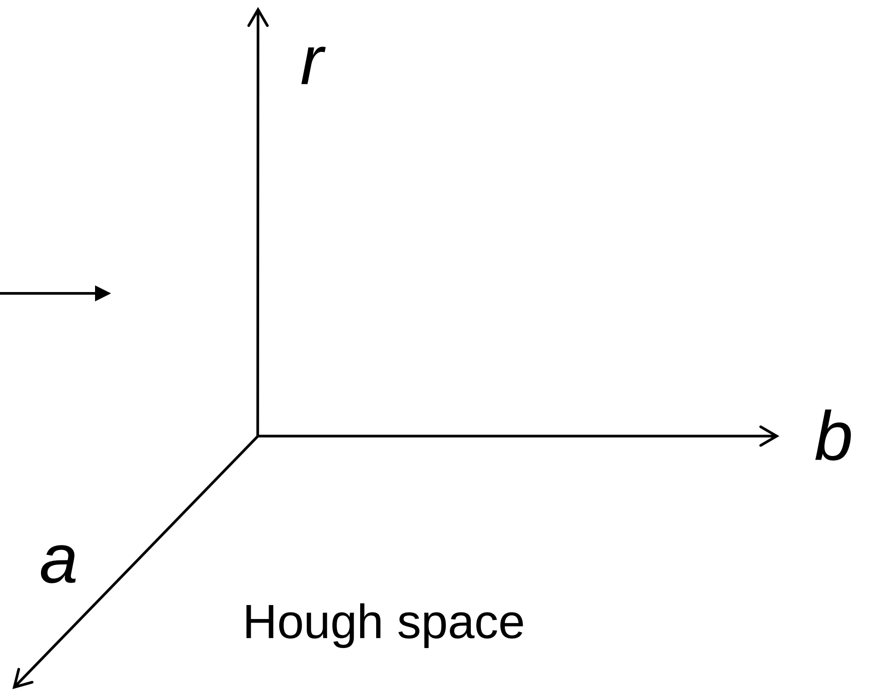
Hough transform for circles

- 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





Slide Credit: Kristen Grauman

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

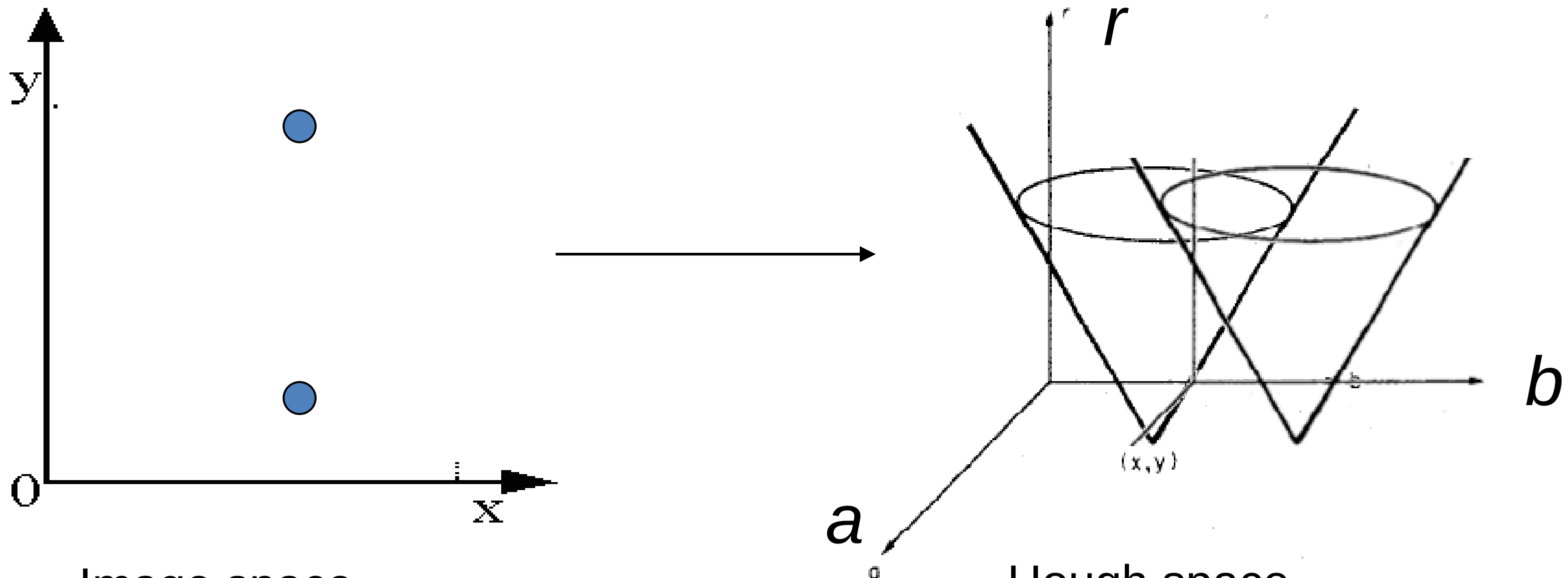


Image space

Hough space

Slide Credit: Kristen Grauman

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

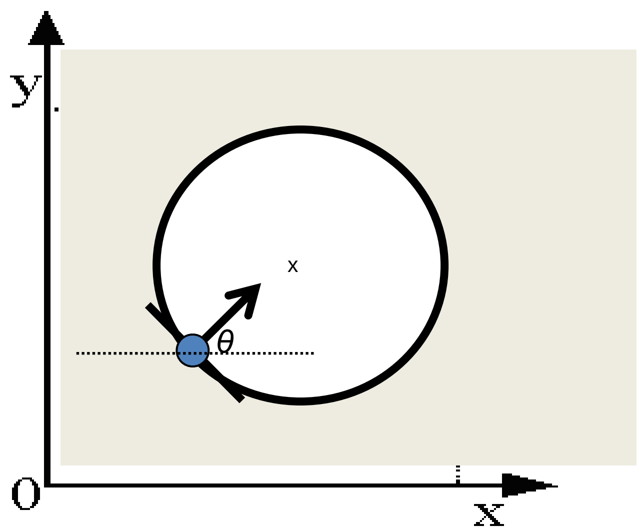
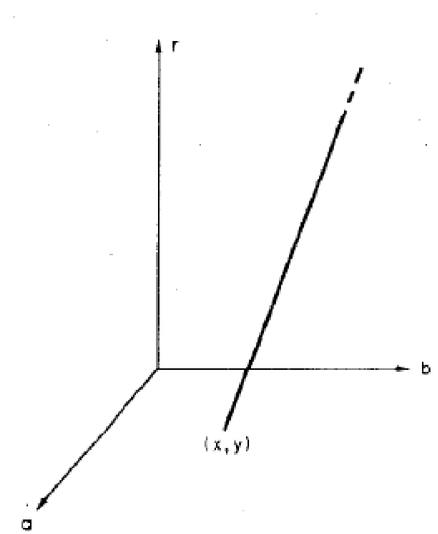


Image space



Hough space

Slide Credit: Kristen Grauman

Slide Credit: Kristen Grauman

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)

$a = x + r \cos(\vartheta)$ // column

$b = y - r \sin(\vartheta)$ // row

H[a,b,r] += 1

end

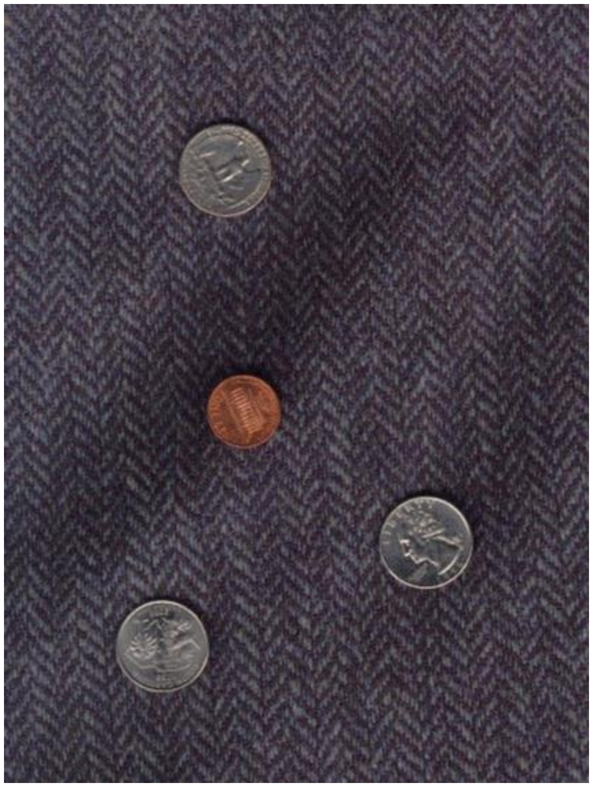
end

Time complexity per edge pixel?

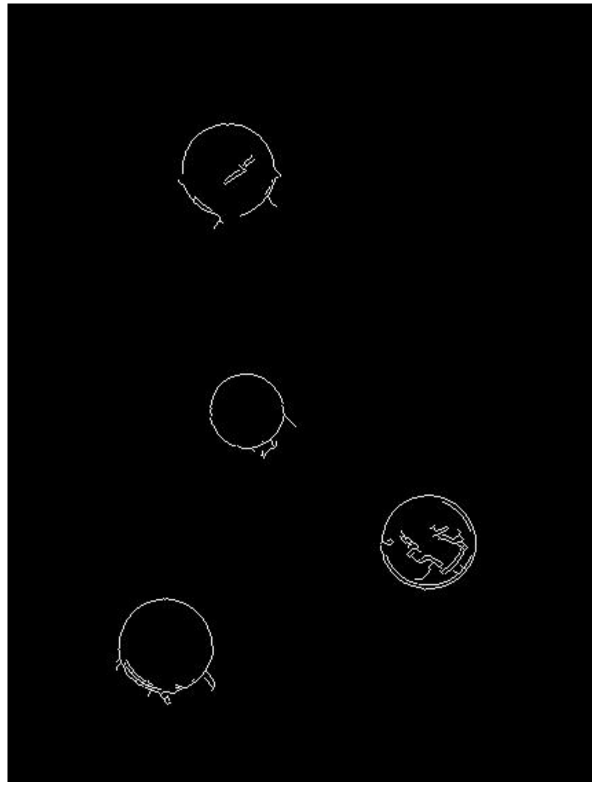
- Check out online demo : http://www.markschulze.net/java/hough/

Example: detecting circles with Hough

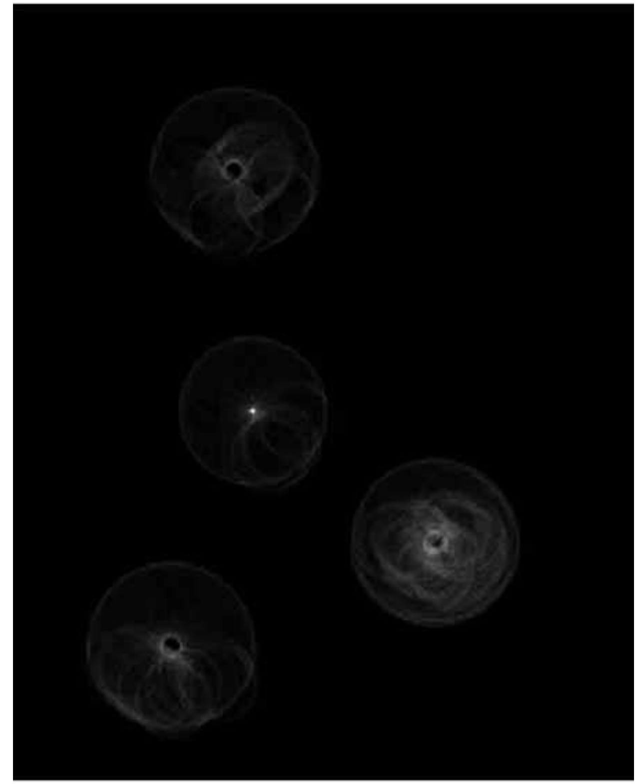
Original



Edges



Votes: Penny



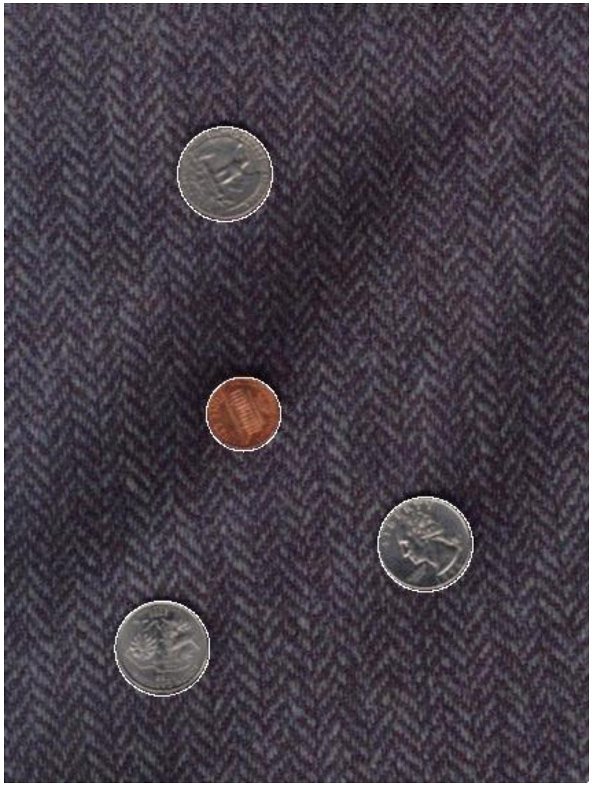
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Slide Credit: Kristen Grauman

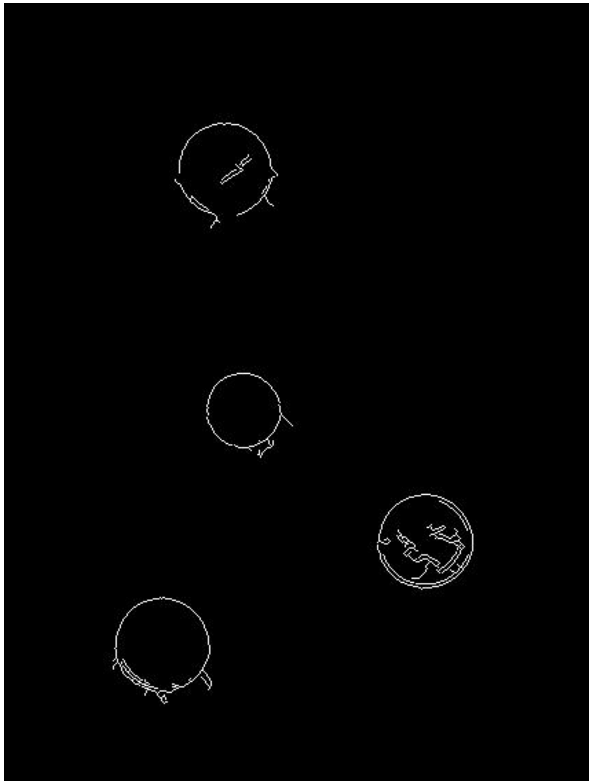
Example: detecting circles with Hough

Combined detections

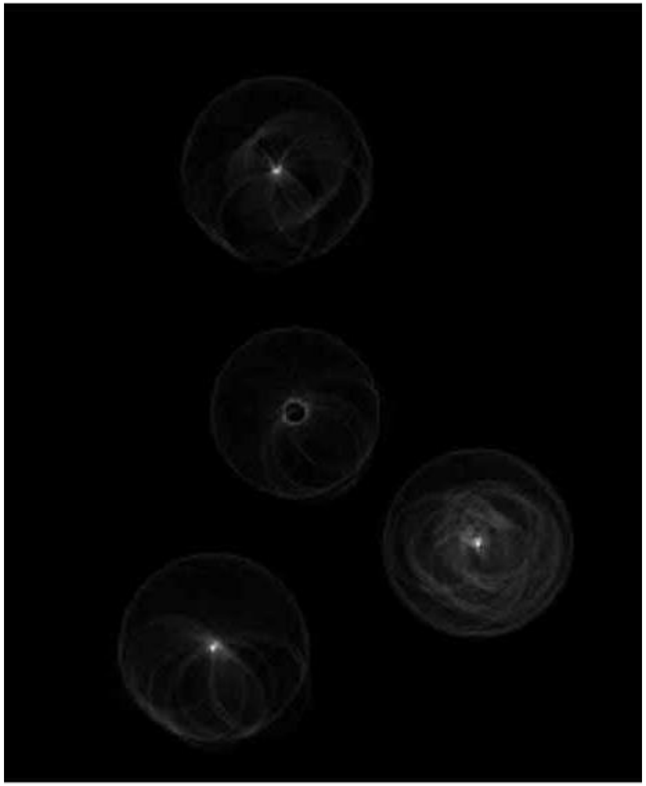
Original



Edges



Votes: Quarter



Coin finding sample images from: Vivek Kwatra

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

Slide Credit: Kristen Grauman