CSCI 4830 / 5722
Computer Vision

University of Colorado **Boulder**

CSCI 4830 / 5722
Computer Vision

Dr. Ioana Fleming
Spring 2019
Lecture 12

University of Colorado **Boulder**

Reminders

Submissions:

- Homework 2: due Wed 2/13 at 11 pm

Readings:

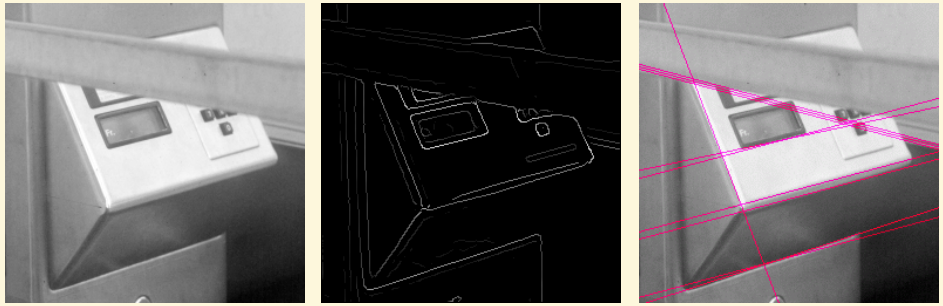
- Szeliski:
  - chapter 4.3 (Hough transform)
- P&F:
  - chapter 10.1 (Hough transform)

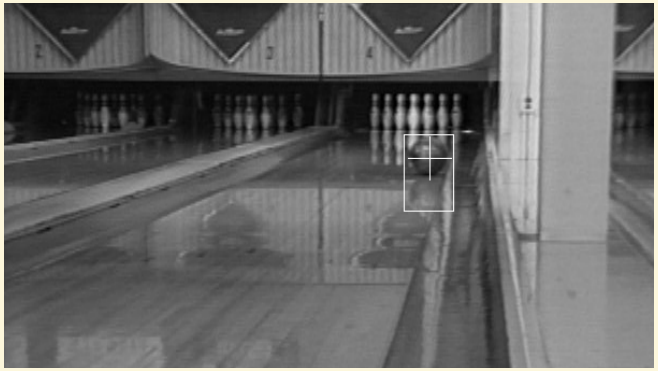
Today

- Fitting
- Hough transform

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.

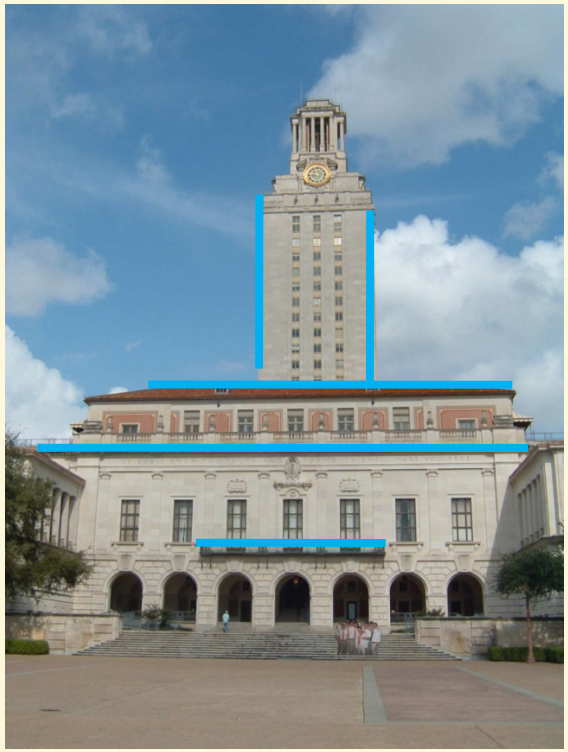
Fitting

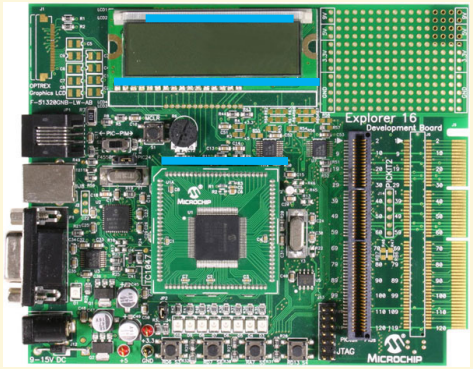
- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can't tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which 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

Source: L. Lazebnik

Example: Line fitting

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



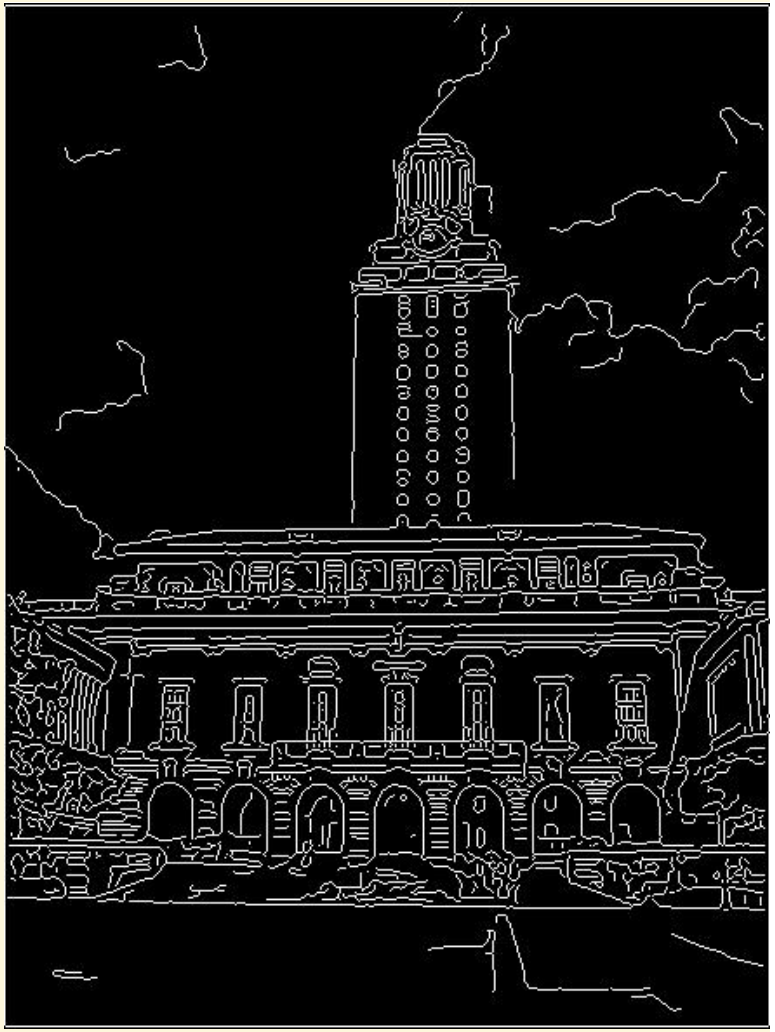


- Wait, why aren't we done just by running edge detection?

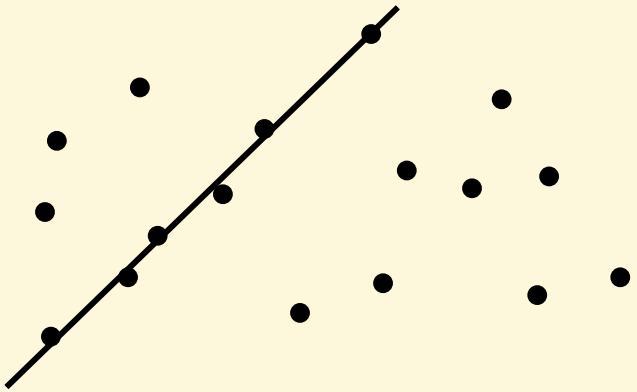
Line fitting

Three problems with line fitting

- Given the points that belong to a line, what is the line?
- Which points belong to which line?
- How many lines are there?



Challenges 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 is 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.

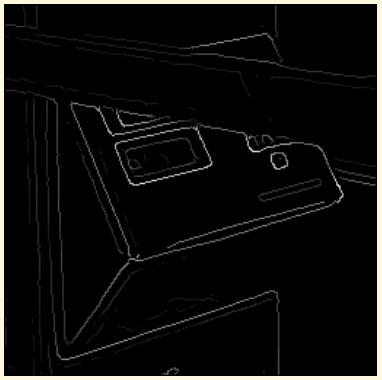
Fitting lines

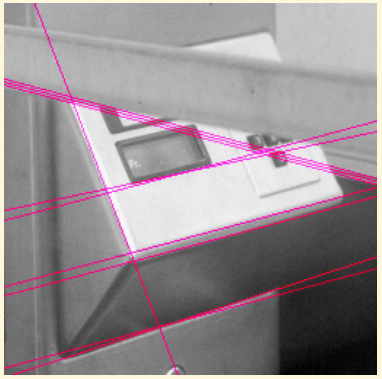
- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

- **Hough Transform** is a voting technique that can be used to answer all of these (rarely, in practice!)

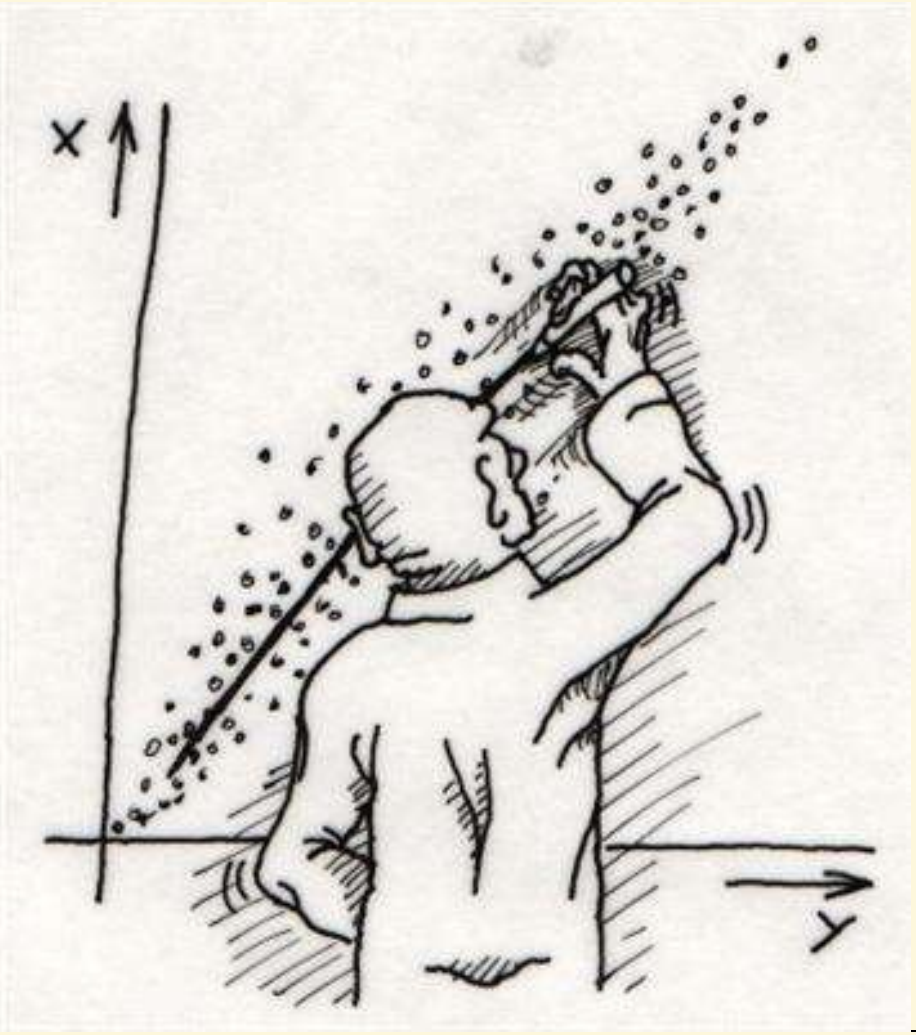
  Main idea:

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



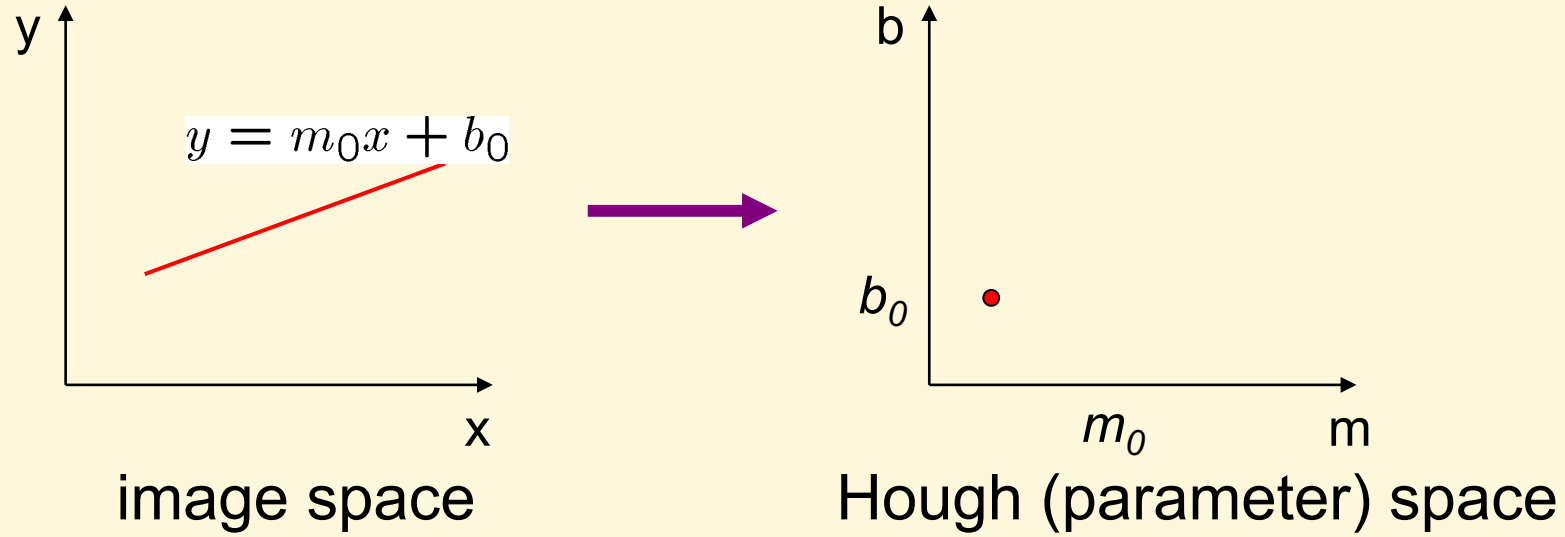


Fitting: The Hough transform



Bubble Chamber (1959)

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 one set of points (x,y), find all (m,b) such that y = mx + b***

Slide credit: Steve Seitz

Finding lines in an image: Hough space

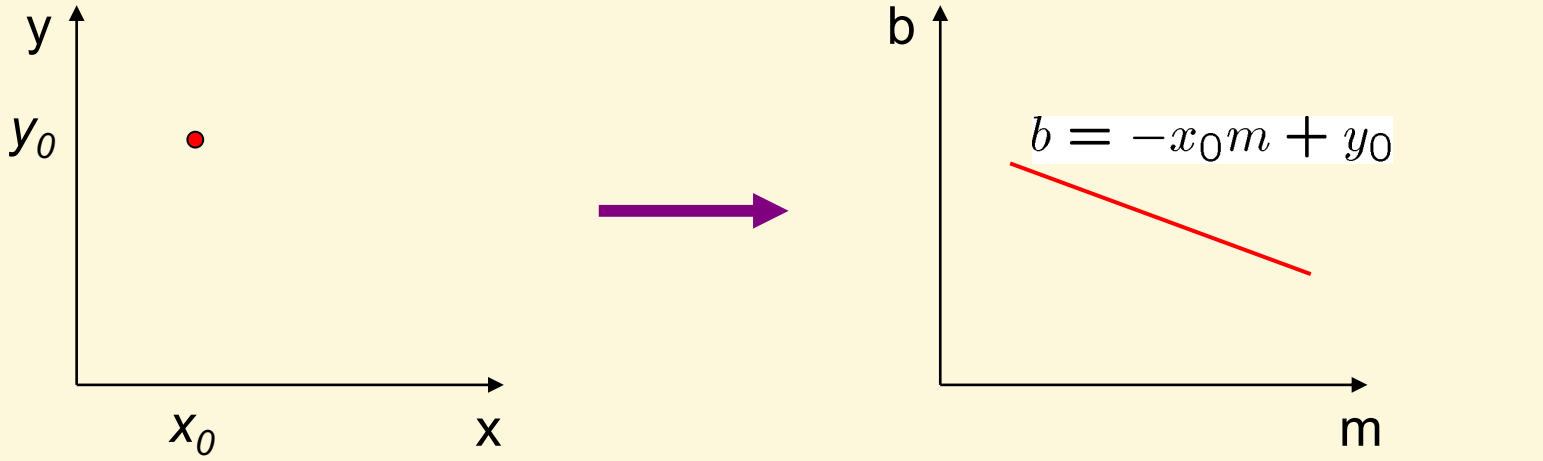


image space Hough (parameter) space

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

Slide credit: Steve Seitz

Finding lines in an image: Hough space

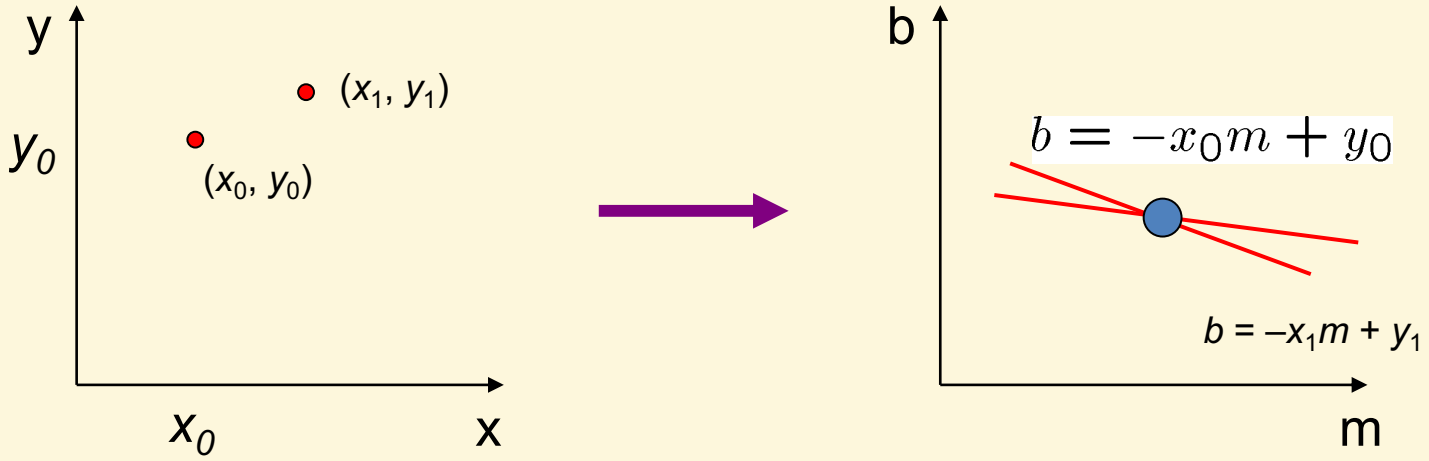


image space

Hough (parameter) space

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

- It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$

Finding lines in an image: Hough algorithm

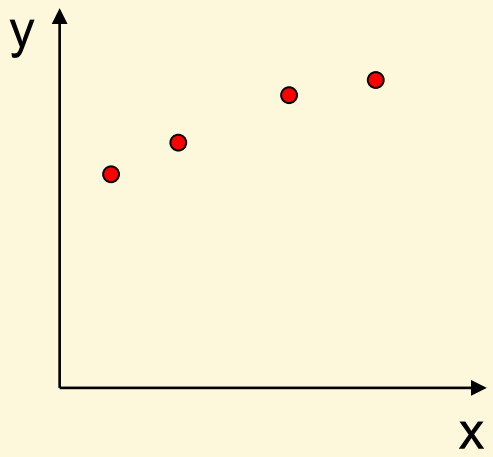
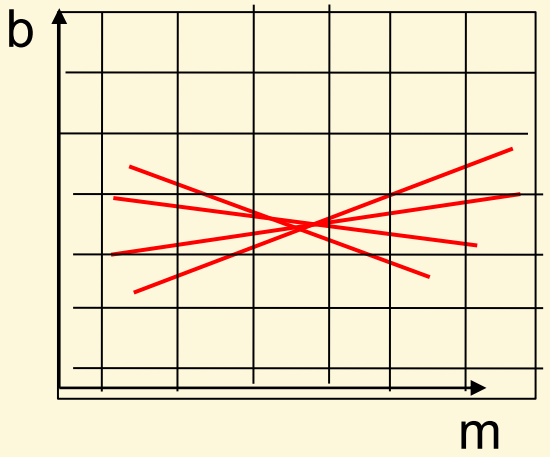


image space



Hough (parameter) space

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

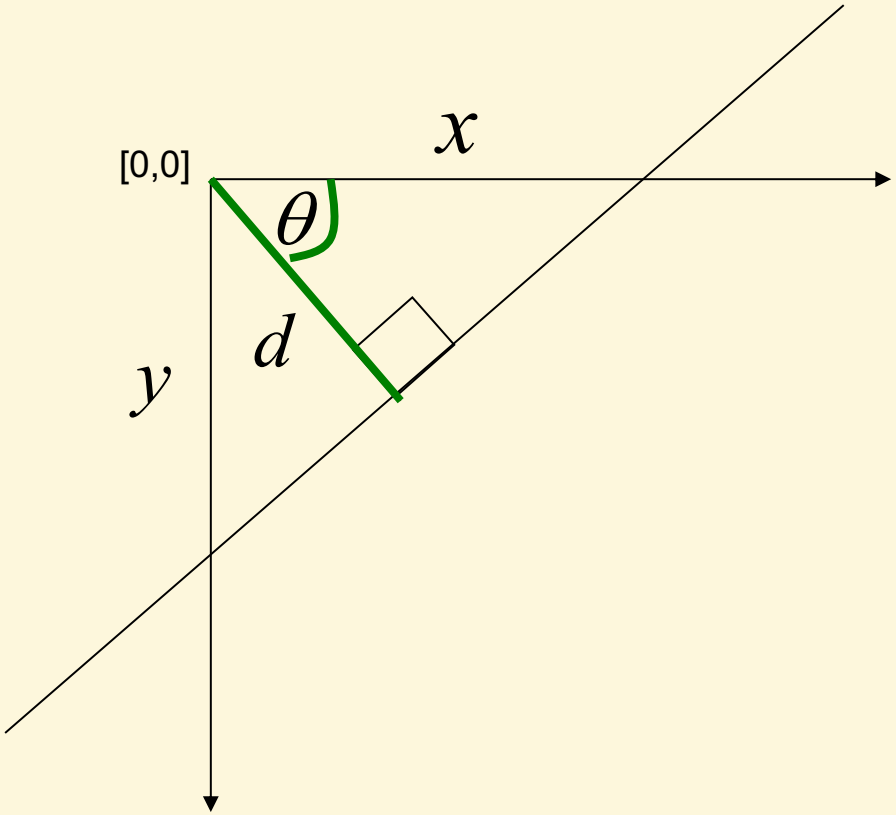
- Let each edge point in image space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Hough algorithm

Cons:

- Issues with usual (m,b) parameter space:
  - Can take on infinite values.
  - Undefined for vertical lines.

Polar representation for lines



d: perpendicular distance from line to origin

θ: angle the perpendicular makes with the x-axis

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

Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," 1972

Polar representation for lines

Given a single point in the plane, then the set of all straight lines going through that point corresponds to a *sinusoidal curve* in the (d,θ) plane, which is unique to that point. A set of two or more points that form a straight line will produce sinusoids which cross at the (d,θ) for that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves

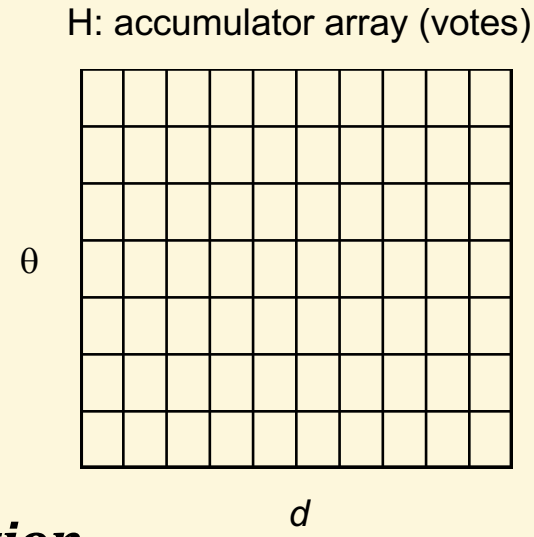
$$x\cos\theta - y\sin\theta = d$$

Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," 1972

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 = 0 to 2π // some quantization

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

   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by

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

Source: Steve Seitz

Example: Hough transform for straight lines

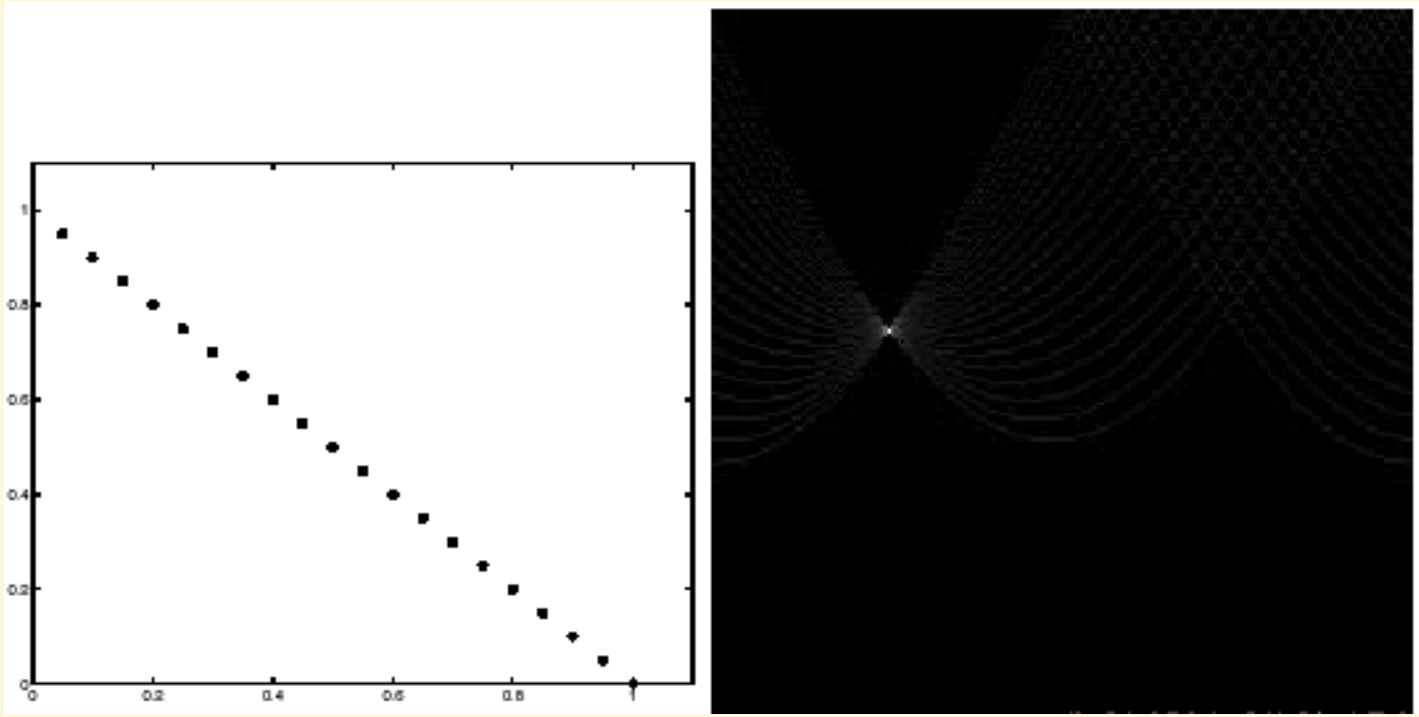


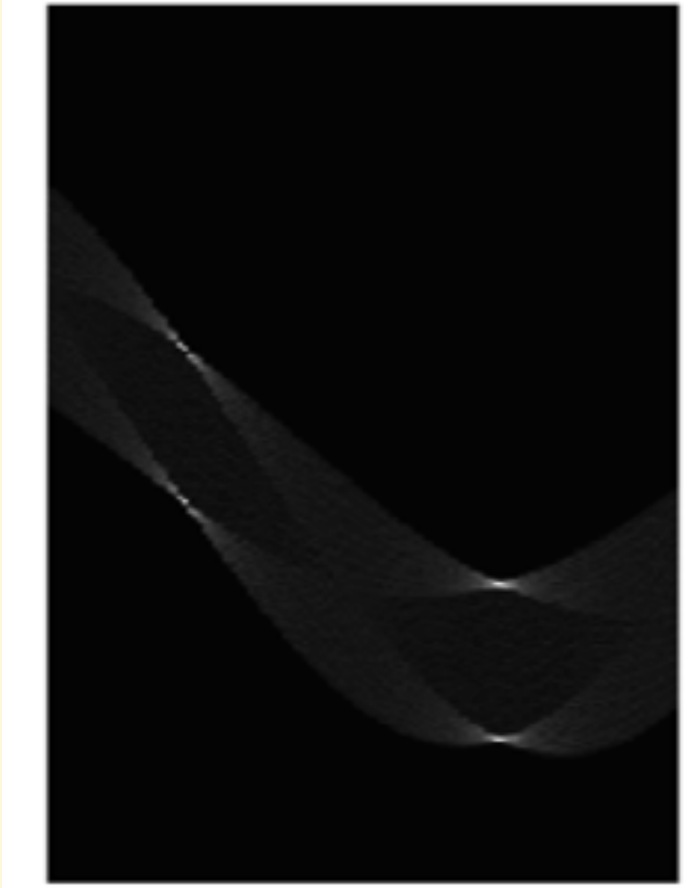
Image space
edge coordinates

Votes

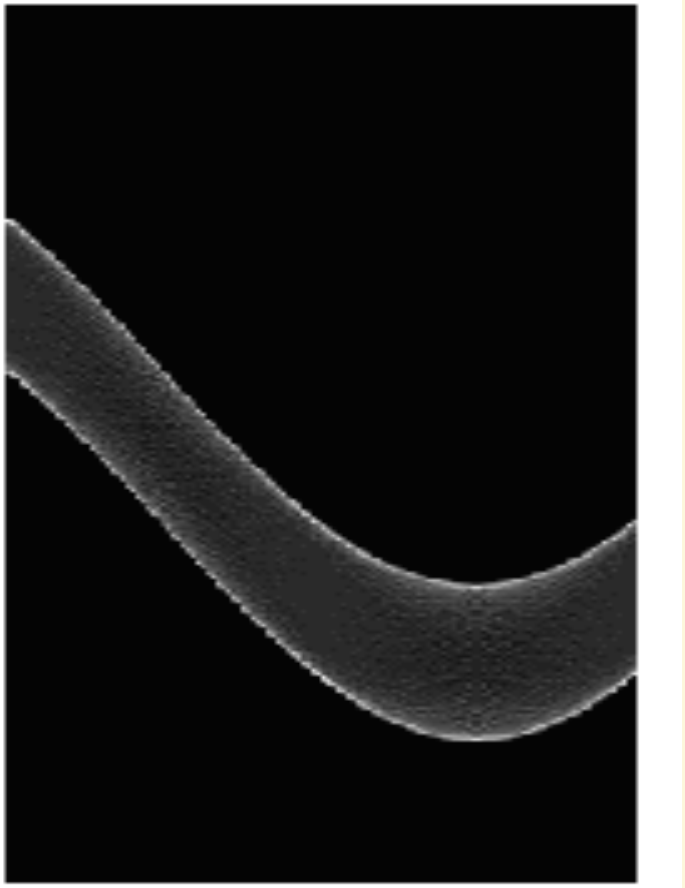
Bright value = high vote count
Black = no votes

Example: Hough transform for straight lines

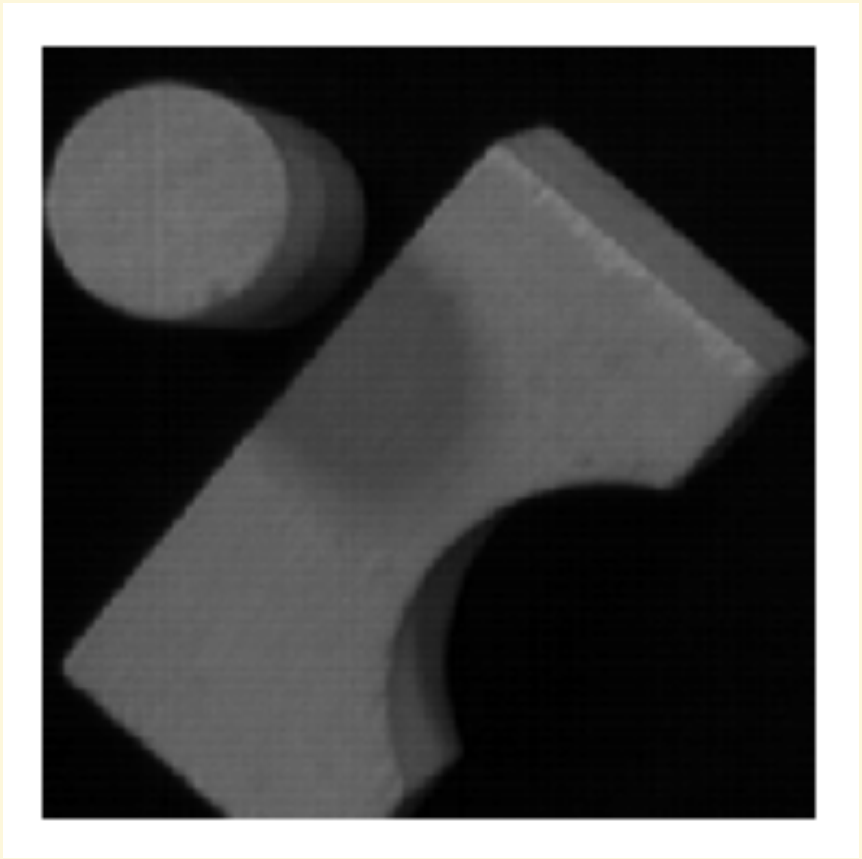
Square :

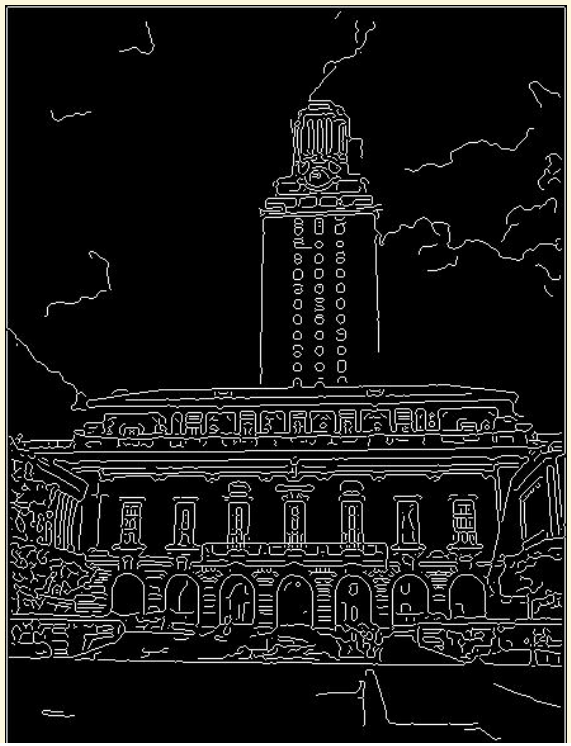


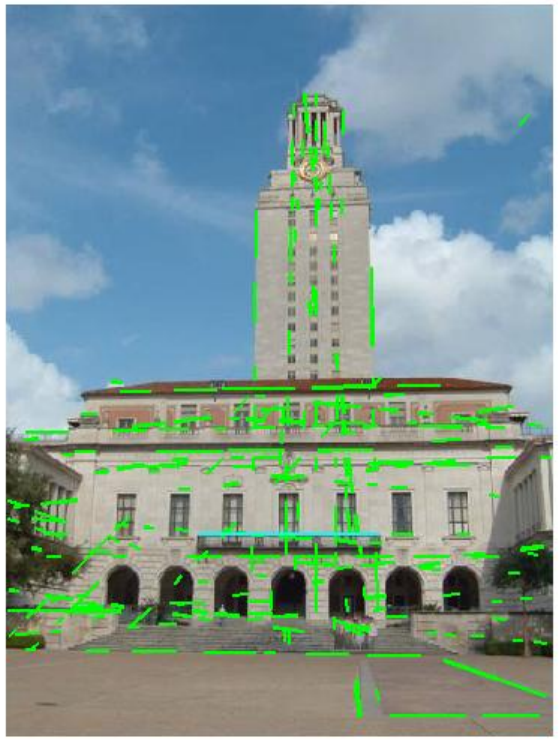
Circle :

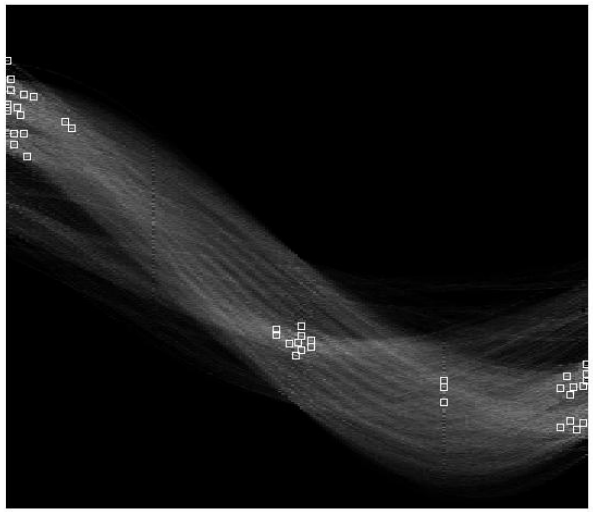


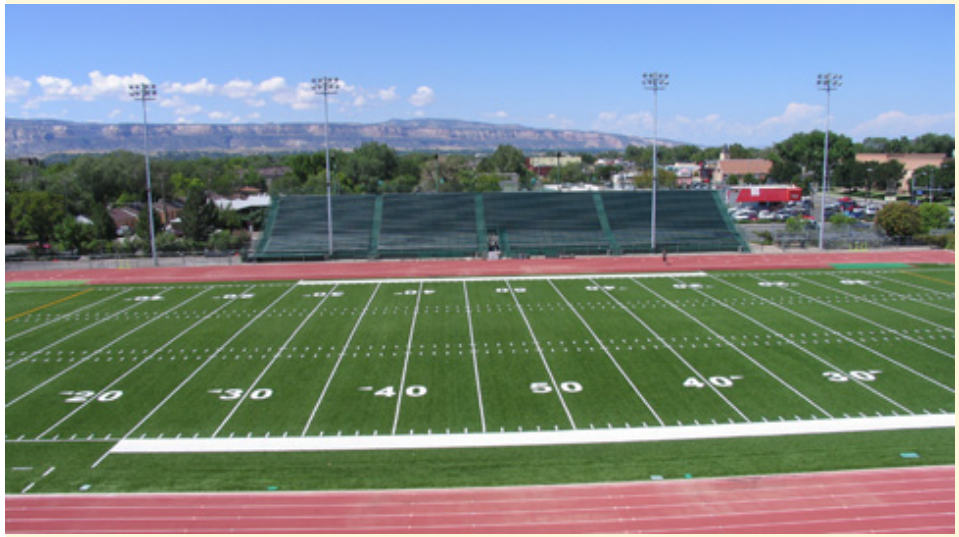
Example: Hough transform for straight lines

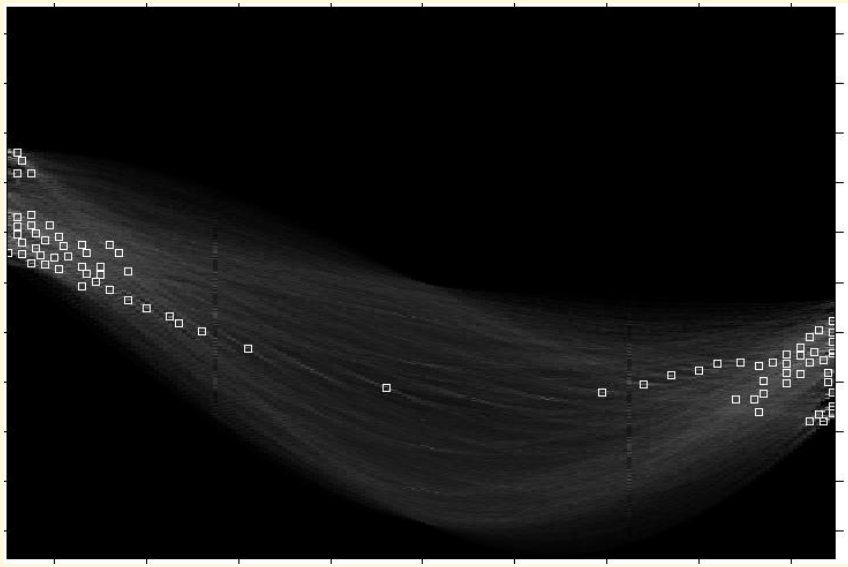


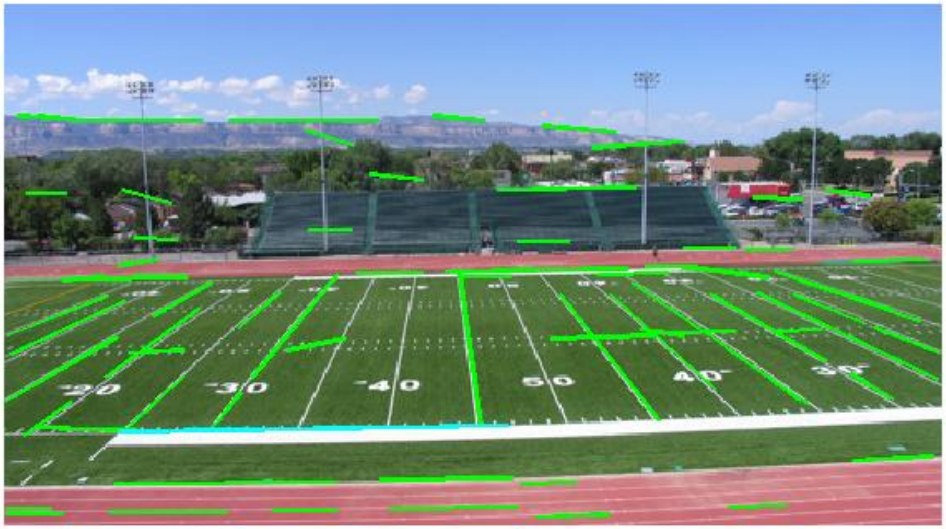





ρ
θ


-20
-30
-40
50
40
30




-20
-30
-40
50
40
30

Impact of noise on Hough

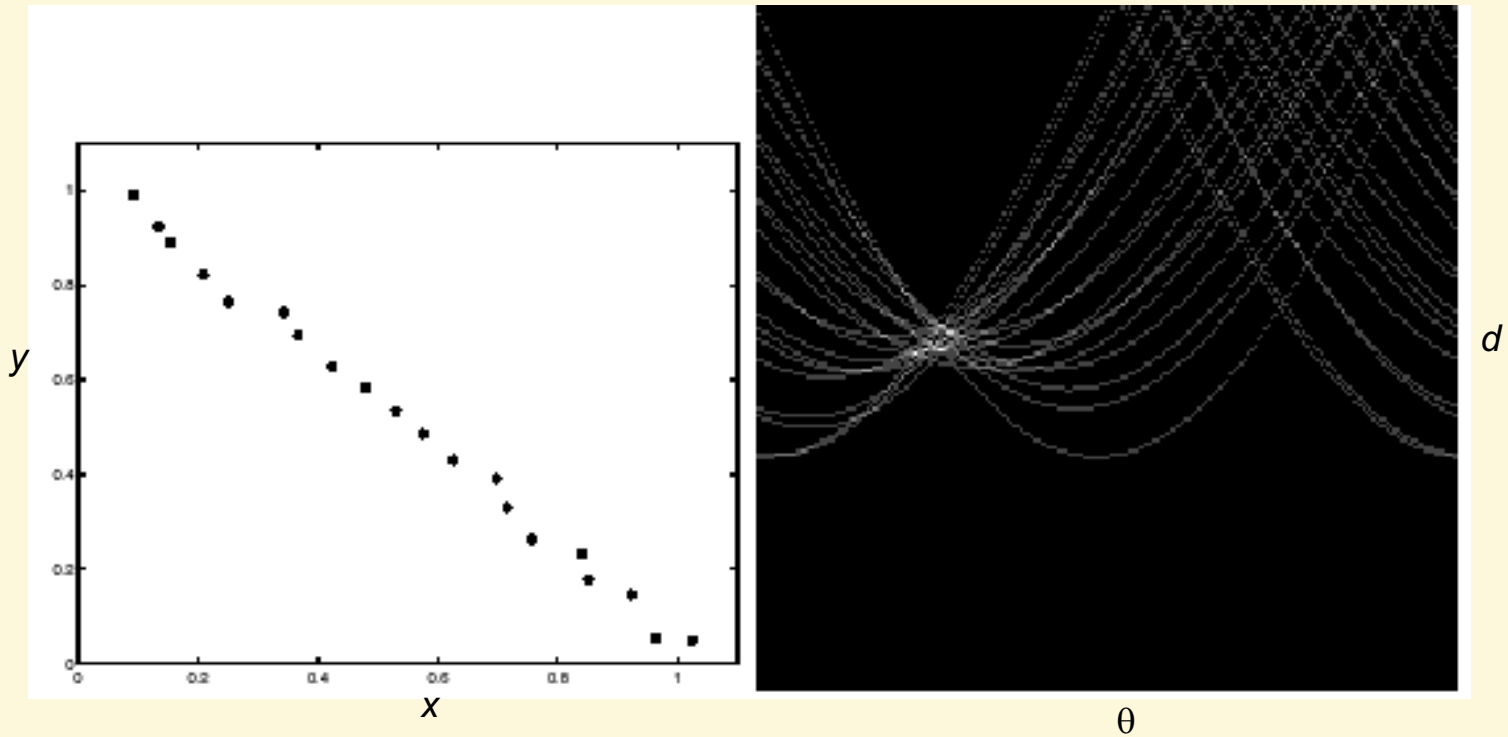


Image space edge coordinates

Votes

What difficulty does this present for an implementation?

Impact of noise on Hough

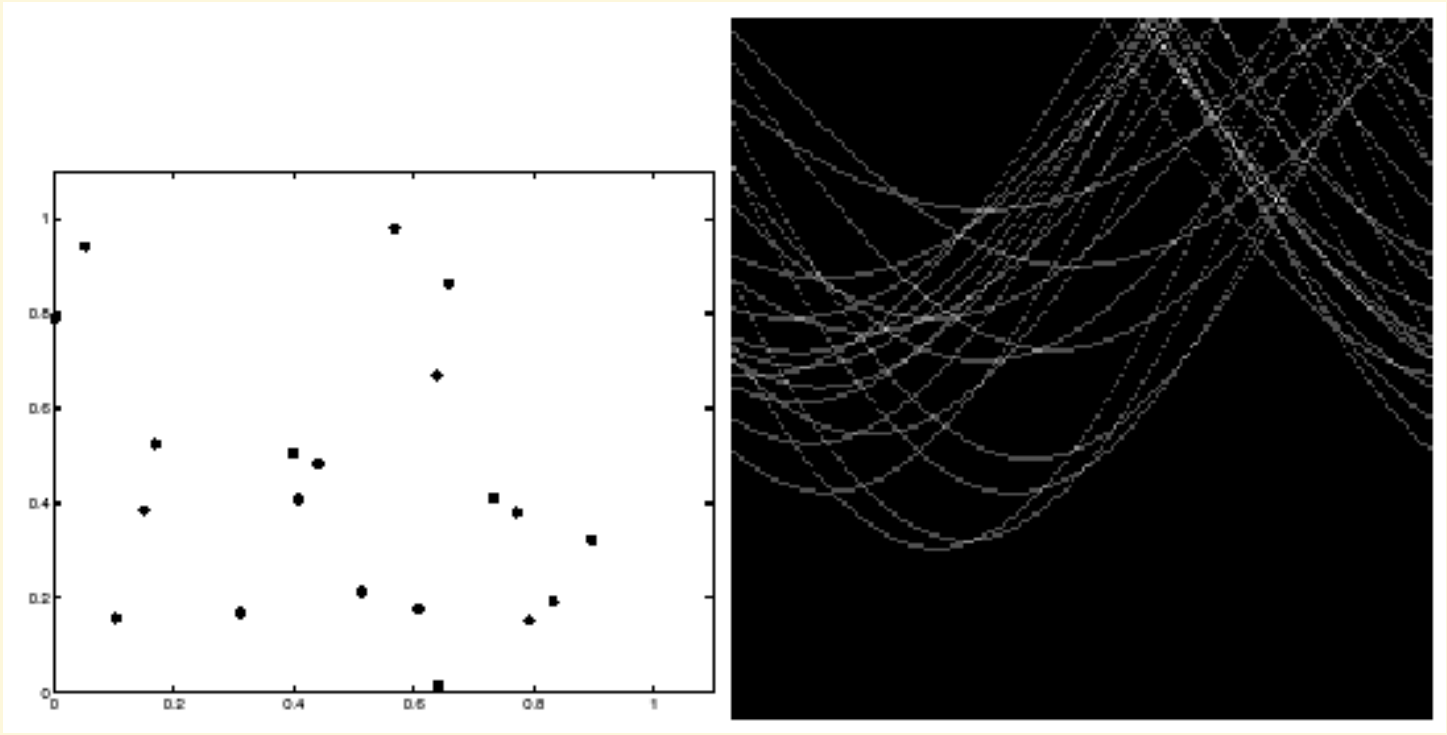


Image space edge coordinates

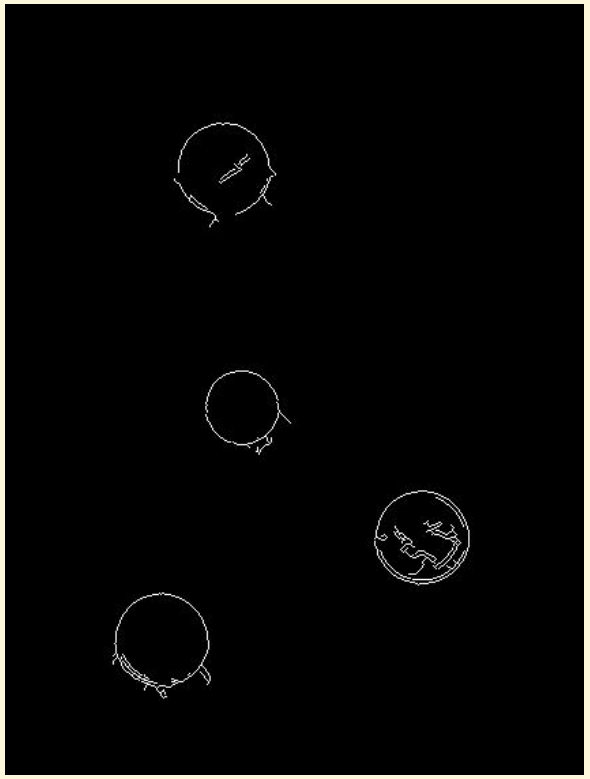
Votes

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

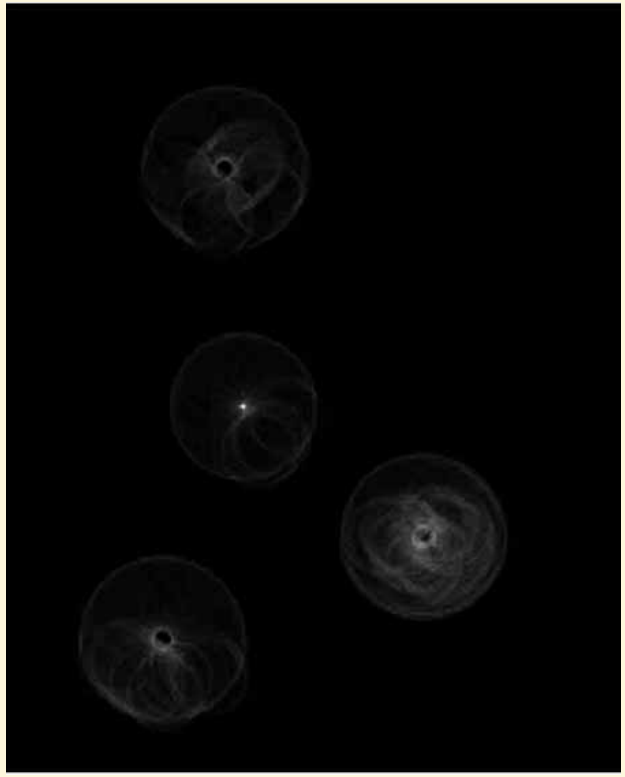
Example: detecting circles with Hough

Original

Edges



Votes: Penny

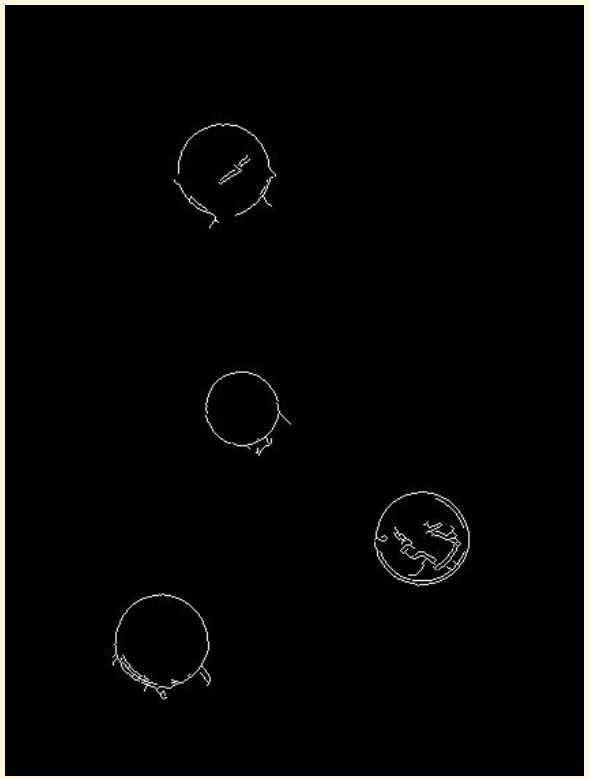


Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

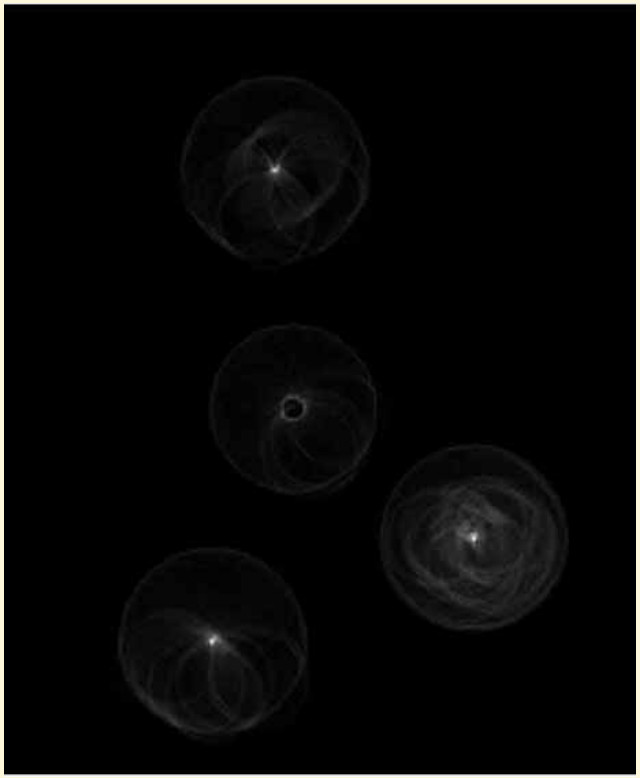
Example: detecting circles with Hough

Original

Edges



Votes: Quarter



Coin finding sample images from: Vivek Kwatra

Voting: practical tips

- Minimize irrelevant tokens first (take edge points with significant gradient magnitude)
- 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
- Vote for neighbors, also (smoothing in accumulator array)
- Utilize direction of edge to reduce the number of votes
- To read back which points voted for "winning" peaks, keep tags on the votes.

Hough transform: pros and cons

Pros

- All points are processed independently, so can cope with occlusion
- 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: hard to pick a good grid size

Generalized Hough transform

- What if want to detect arbitrary shapes? Model is defined by boundary points (p_i) and a reference origin point (a).

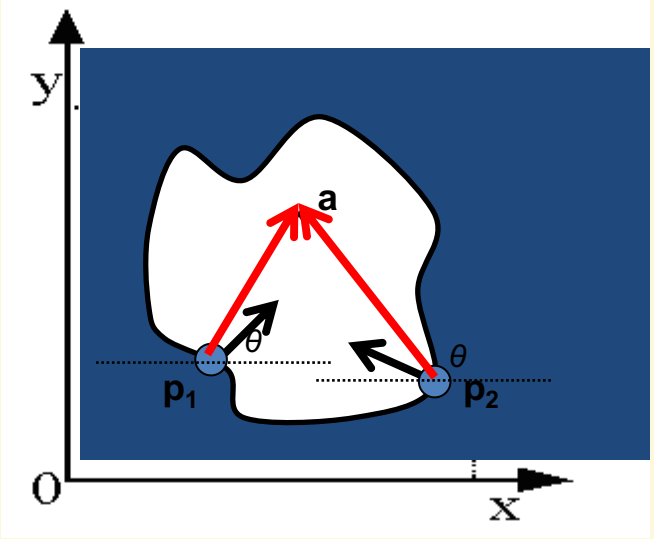


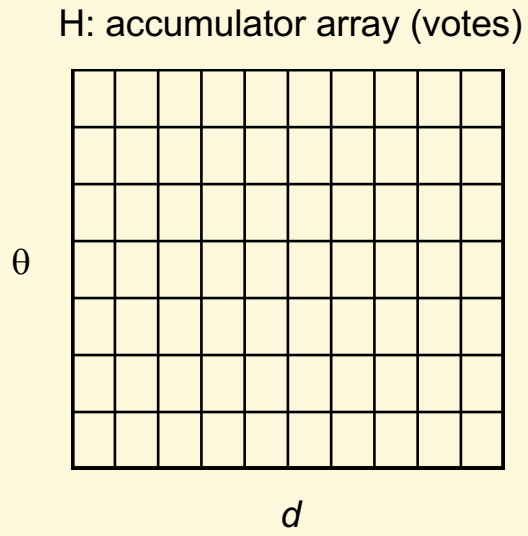
Image space

At each boundary point, compute displacement vector: **r = a – p_i.**

For a given model shape: store these vectors in a table/accumulator of **r vs. θ** (the gradient orientation).

[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

Hough transform algorithm



The Hough transform is only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected amid the background noise. This means that the bin must not be too small, or else some votes will fall in the neighboring bins, thus reducing the visibility of the main bin

Time complexity:

$\mathcal{O}\left(A^{m-2}\right)$, where A is the size of the image space and m is the number of parameters.
Thus, the Hough transform must be used with great care to detect anything other than lines or circles.

Source: Steve Seitz