Fitting:

Voting and the Hough Transform

Tues Feb 14

Kristen Grauman

UT Austin

Today

- Grouping : wrap up clustering algorithms
  - See slides from last time
- Fitting : introduction to voting

Slide credit: Kristen Grauman

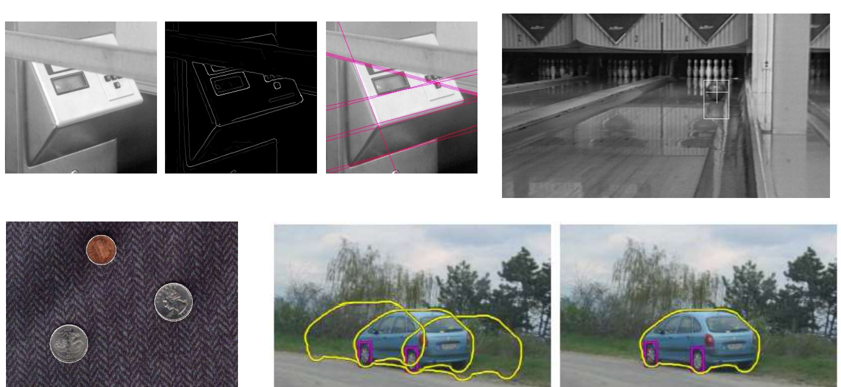
Last time

- What are grouping problems in vision?
- Inspiration from human perception
  - Gestalt properties
- Bottom-up segmentation via clustering
  - Algorithms:
    - Mode finding and mean shift: k-means, mean-shift
    - Graph-based: normalized cuts
  - Features: color, texture, …
    - Quantization for texture summaries

Slide credit: Kristen Grauman

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

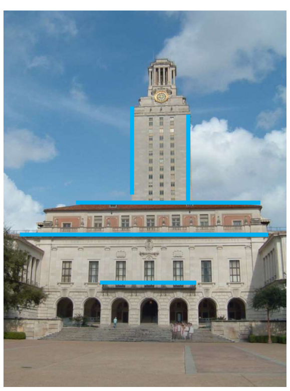
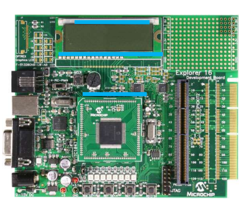
Fitting: Main idea

- 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

Slide credit: L. Lazebnik

Example: Line fitting

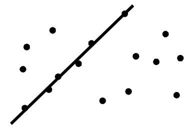
- Why fit lines?
  Many objects characterized by presence of straight lines

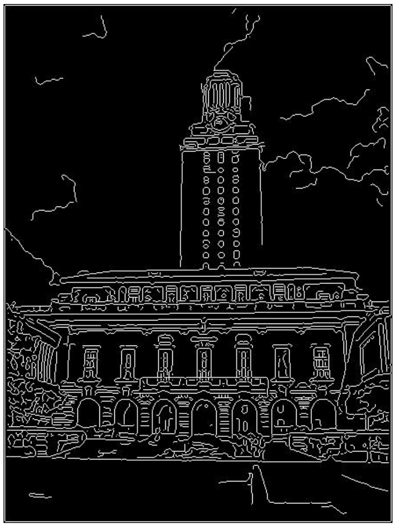




- Wait, why aren't we done just by running edge detection?

Slide credit: Kristen Grauman

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?

Slide credit: Kristen Grauman

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.

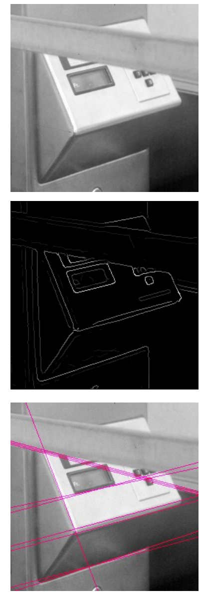
Slide credit: Kristen Grauman

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?

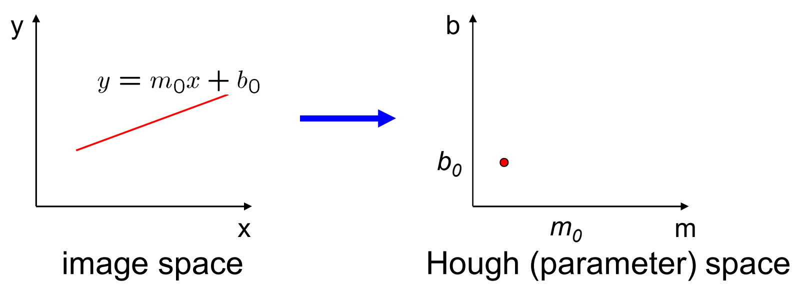
- **Hough Transform** is a voting technique that can be used to answer all of these questions.

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



Slide credit: Kristen Grauman

Finding lines in an image: Hough space

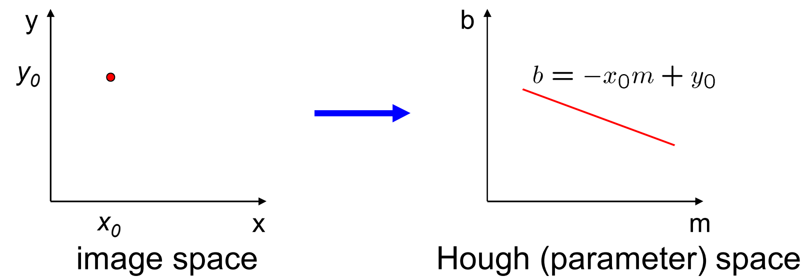


Connection between image (x,y) and Hough (m,b) spaces

- 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

Slide credit: Steve Seitz

Finding lines in an image: Hough space

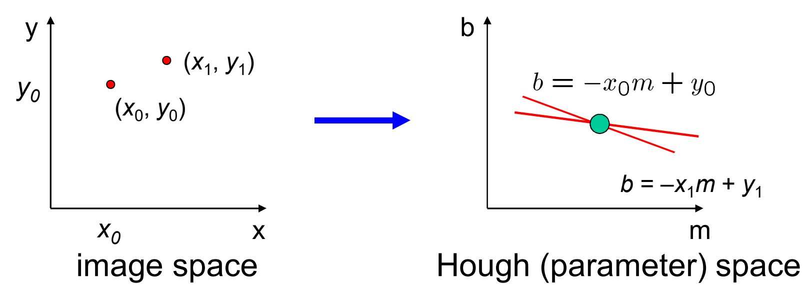


Connection between image (x,y) and Hough (m,b) spaces

- 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 $b = -x_0 m + y_0$
  - this is a line in Hough space

Slide credit: Steve Seitz

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 algorithm

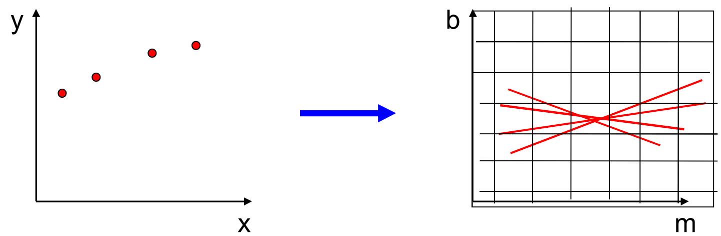


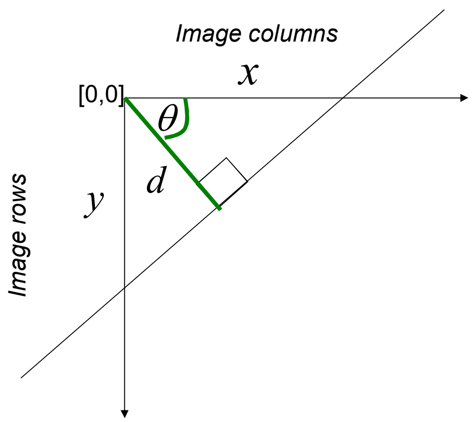
image space Hough (parameter) space

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.

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 → sinusoid segment in Hough space

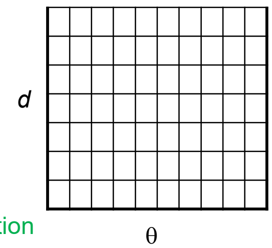
Slide credit: Kristen Grauman

Hough transform algorithm

Using the polar parameterization:

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

H: accumulator array (votes)



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

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

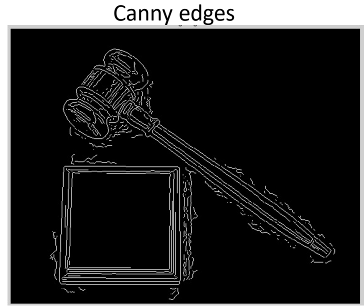
   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta - y\sin\theta$

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

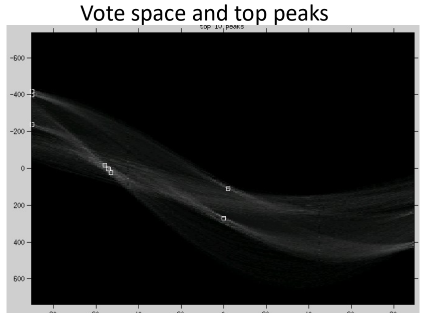
Source: Steve Seitz

Original image

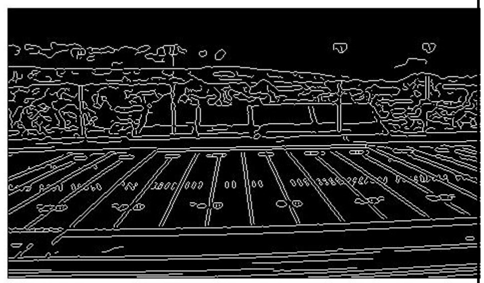
Canny edges

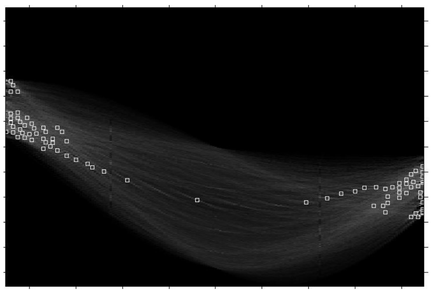


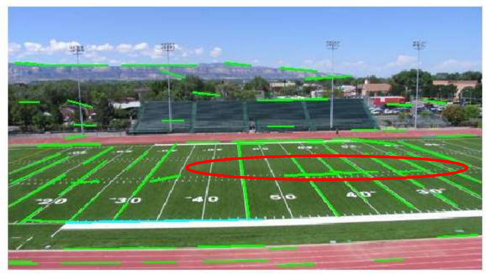
Vote space and top peaks



Slide credit: Kristen Grauman





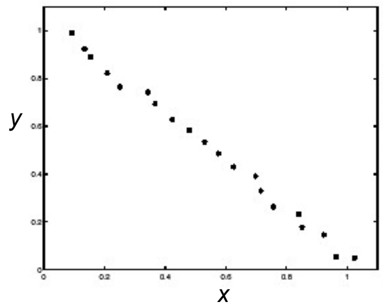


Showing longest segments found

Slide credit: Kristen Grauman

- https://www.youtube.com/watch?v=ebfi7qOFLuo

Impact of noise on Hough



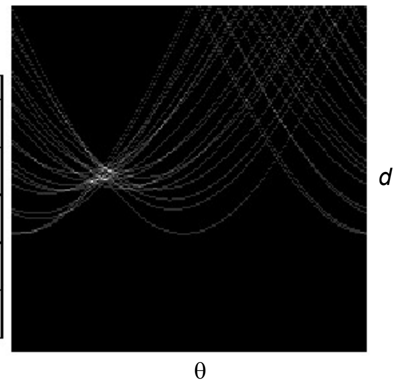
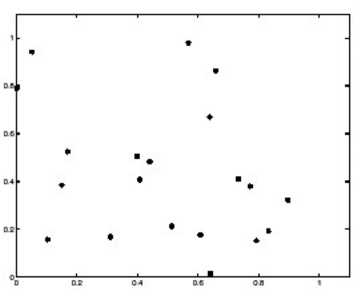


Image space edge coordinates

Votes

What difficulty does this present for an implementation?

Impact of noise on Hough



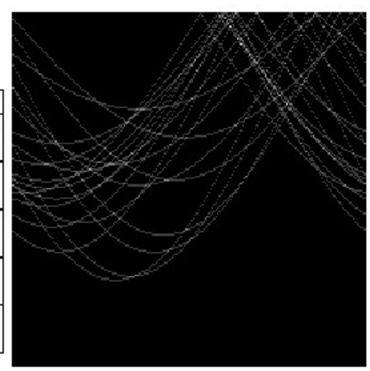


Image space edge coordinates

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

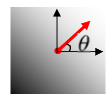
1. same
2. for each edge point I[x,y] in the image

   θ = gradient at (x,y)

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

   H[d, θ] += 1
3. same
4. same

(Reduces degrees of freedom)



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

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

Extensions

Extension 1: Use the image gradient

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

   compute unique (d, θ) based on image gradient at (x,y)

   H[d, θ] += 1
3. same
4. same

(Reduces degrees of freedom)

Extension 2

- give more votes for stronger edges (use magnitude of gradient)

Extension 3

- change the sampling of (d, θ) to give more/less resolution

Extension 4

- The same procedure can be used with circles, squares, or any other shape…

Source: Steve Seitz

Summary

- Clustering and segmentation algorithms
  - Kmeans
  - Mean shift
  - Normalized cuts
  - MRF for interactive
- Quantizing features
  - Summarize spatial statistics over prototypical feature
- Fitting via voting
  - Fitting vs. grouping
  - Hough Transform for lines

Coming up

- Thursday: More on Hough transform
  - Circles, arbitrary shapes
- Reminder: A2 is due next Friday