Fitting:
Voting and the Hough Transform (part 2)

Kristen Grauman
UT Austin

Feb 16, 2017

Announcements

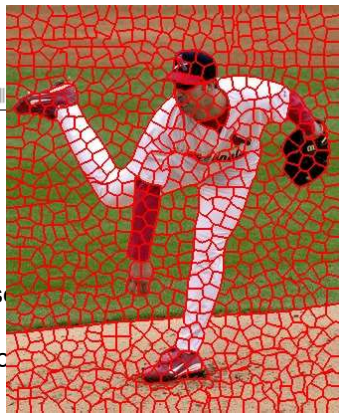
- A2 due next Friday
- Anonymous course survey – see link on Piazza
  - Please respond by next Thursday 2/23

Review: graph-based clustering

- Assuming we use a fully connected graph, what is the time complexity of computing the affinities for a graph cuts-based segmentation?

- Example affinity measure:

$$w_{ij} = e^{\frac{-\|\boldsymbol{F}(i)-\boldsymbol{F}(j)\|_2^2}{\sigma_I}} * \begin{cases} e^{\frac{-\|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|}{\sigma_X}} \\ 0 \end{cases}$$



X(i) is position of node i

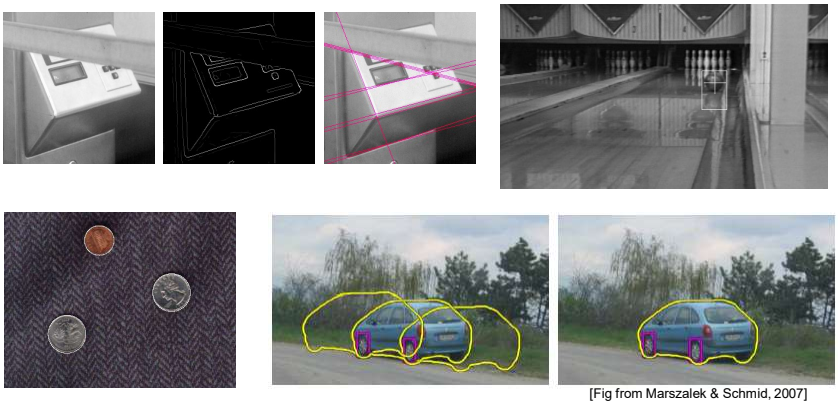
F(i) is a feature vector for node i bas

This affinity measure limits connectic

Slide credit: Kristen Grauman

Now: Fitting

- Want to associate a model with observed features



[Fig from Marszalek & Schmid, 2007]

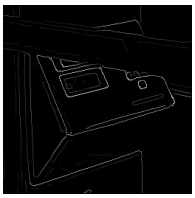
For example, the model could be a line, a circle, or an arbitrary shape.

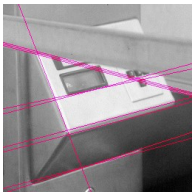
Slide credit: Kristen Grauman

Recall: 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.

Recall: Fitting lines: Hough transform





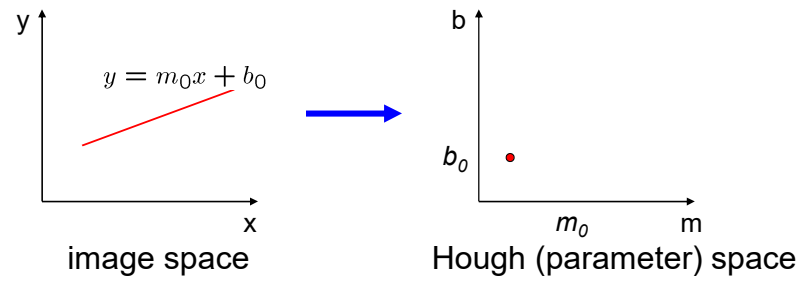
- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?
- **Hough Transform** is a voting technique that can be used to answer all of these questions.

  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

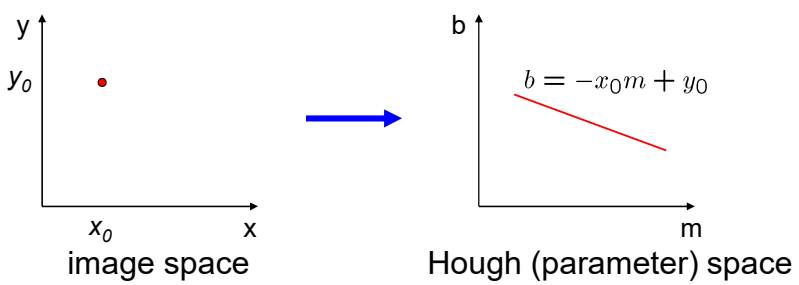


Connection between image (x,y) and Hough (m,b) spaces

- 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 $b = -x_0m + y_0$
  - this is a line in Hough space

Slide credit: Steve Seitz

Finding lines in an image: Hough space

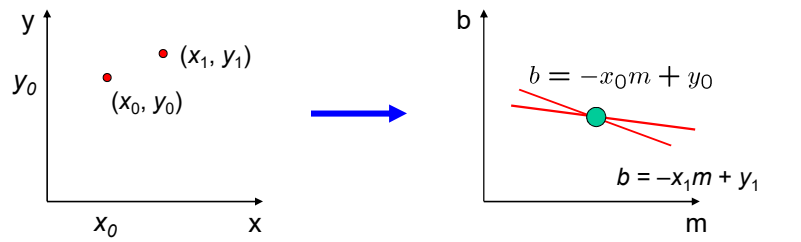


image space Hough (parameter) space

What are the 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

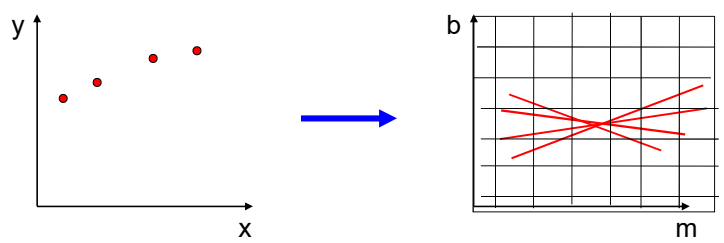


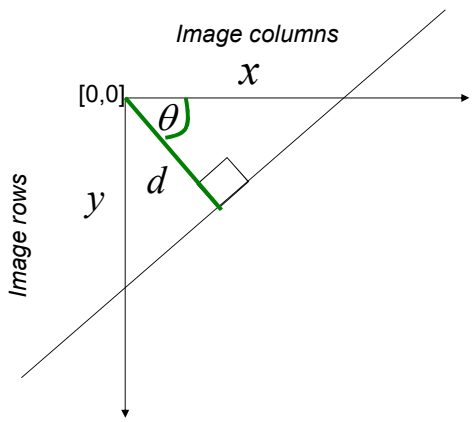
image space Hough (parameter) space

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

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

Written in image coordinates

Point in image space → sinusoid segment in Hough space

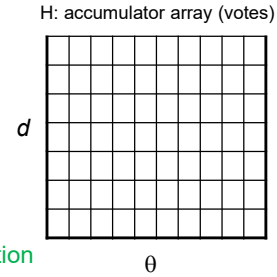
Slide credit: Kristen Grauman

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)?

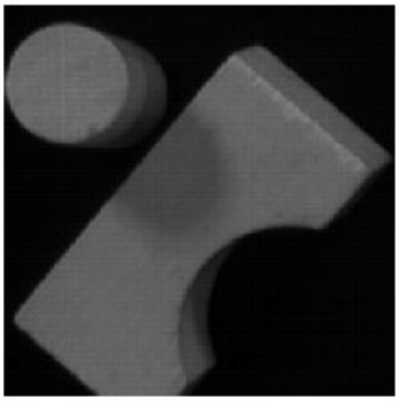
Source: Steve Seitz

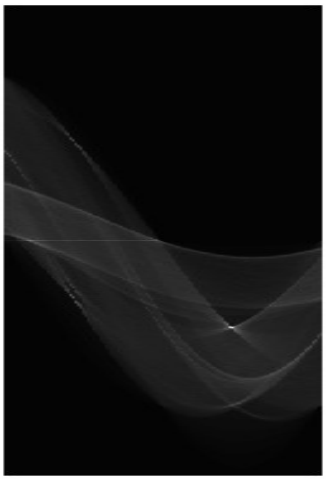
- https://www.youtube.com/watch?v=ebfi7qOFLuo

Example: What's in the image?

Square :

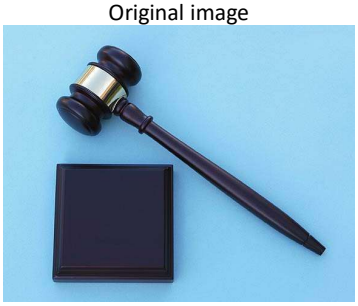
Example: Hough transform for straight lines



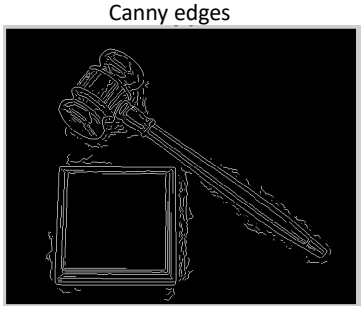


Which line do you think generated the brightest peak?

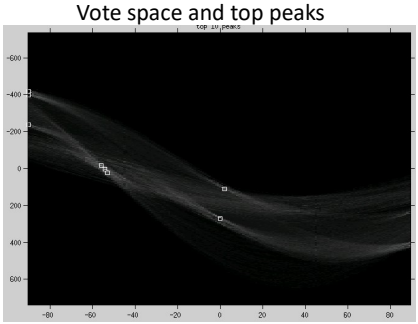
Original image



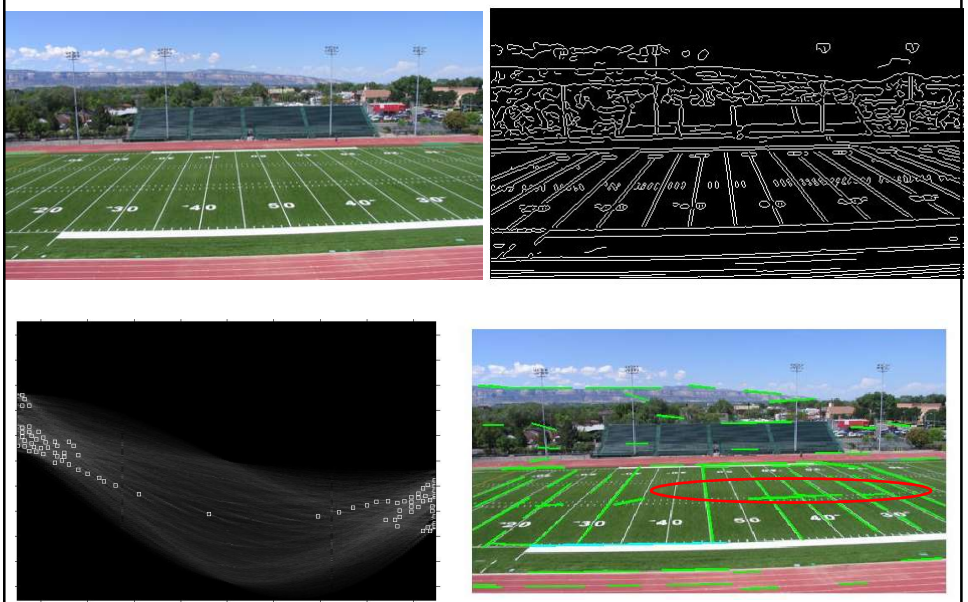
Canny edges



Vote space and top peaks



Slide credit: Kristen Grauman



Showing longest segments found

Slide credit: Kristen Grauman

Impact of noise on Hough

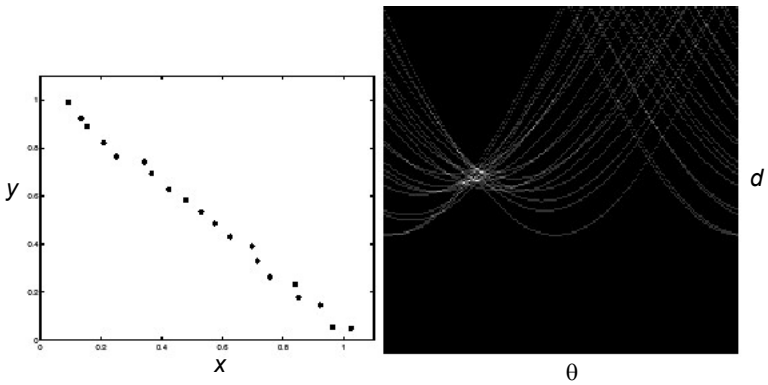


Image space edge coordinates

Votes

What difficulty does this present for an implementation?

Impact of noise on Hough

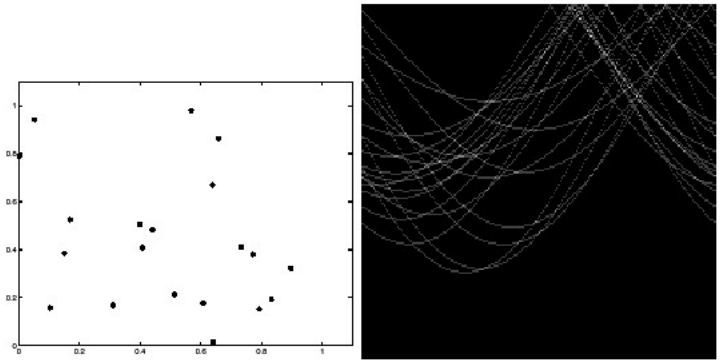


Image space edge coordinates **Votes**

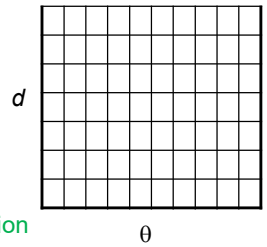
Here, everything appears to be “noise”, or random edge points, but we still see peaks in the vote space.

From before: 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 $d = x\cos\theta - y\sin\theta$

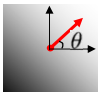
Source: Steve Seitz

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…

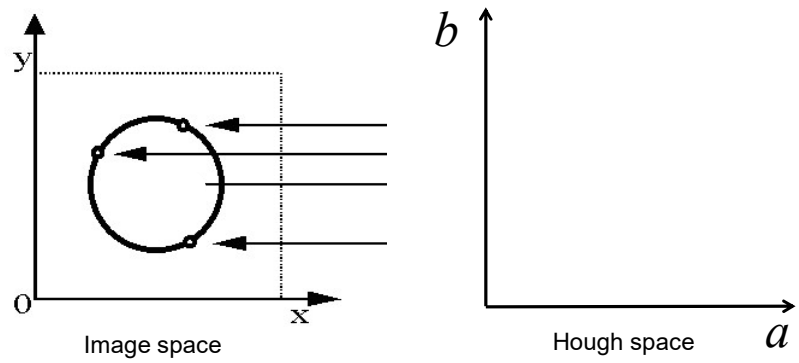
Source: Steve Seitz

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



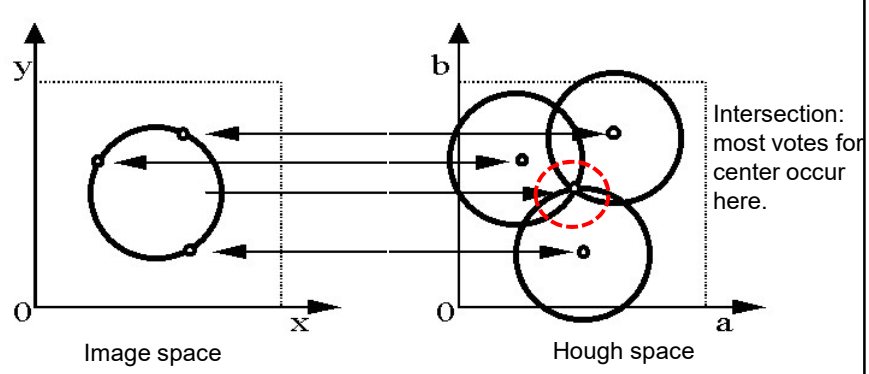
Slide credit: Kristen Grauman

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



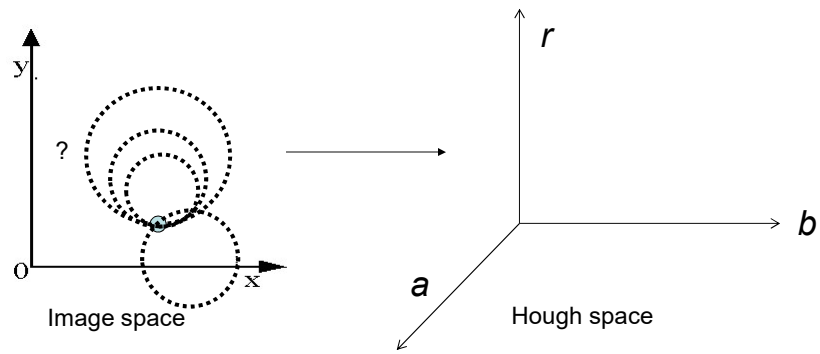
Slide credit: Kristen Grauman

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



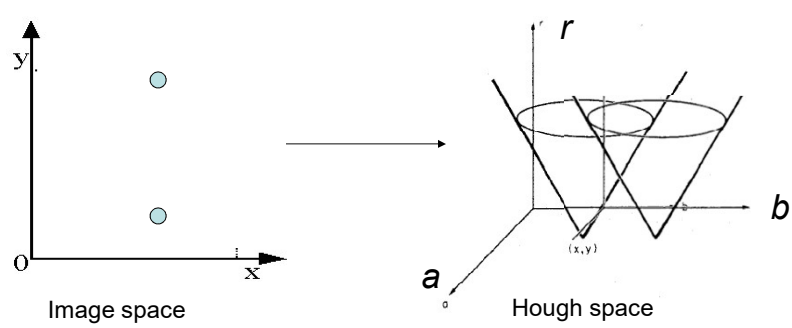
Slide credit: Kristen Grauman

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



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

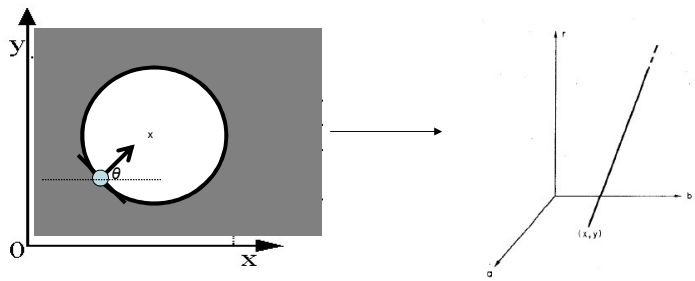


Image space Hough space

Slide credit: Kristen Grauman

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(*θ*) // column
      b = *y* - *r* sin(*θ*) // row
      H[*a,b,r*] += 1
  end
end

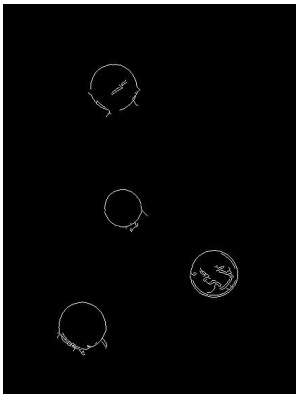
Time complexity per edgel?

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

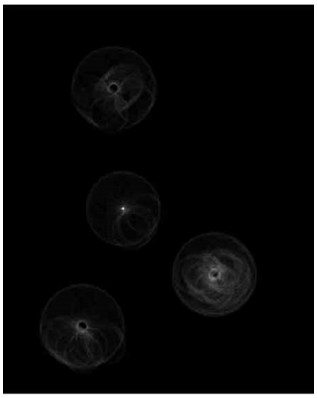
Example: detecting circles with Hough

Original

Edges



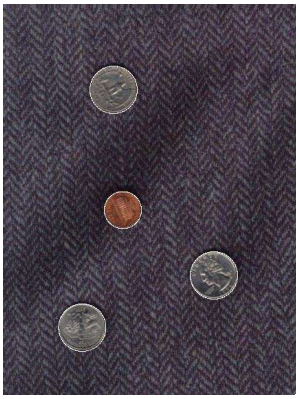
Votes: Penny



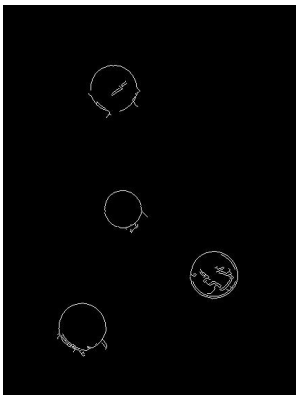
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Example: detecting circles with Hough

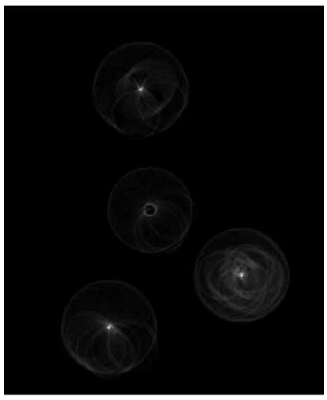
Combined detections



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

Example: iris detection

Figure 2. Original image

Figure 3. Distance image

Figure 4. Detected face region

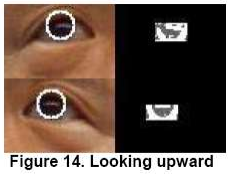


Figure 14. Looking upward

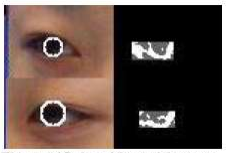


Figure 15. Looking sideways

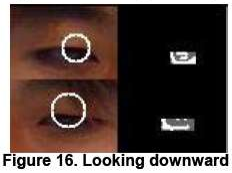


Figure 16. Looking downward

- An Iris Detection Method Using the Hough Transform and Its Evaluation for Facial and Eye Movement, by Hideki Kashima, Hitoshi Hongo, Kunihito Kato, Kazuhiko Yamamoto, ACCV 2002.

Voting: practical tips

- Minimize irrelevant tokens first
- Choose a good grid / discretization

Too fine ? Too coarse

- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes.

Slide credit: Kristen 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

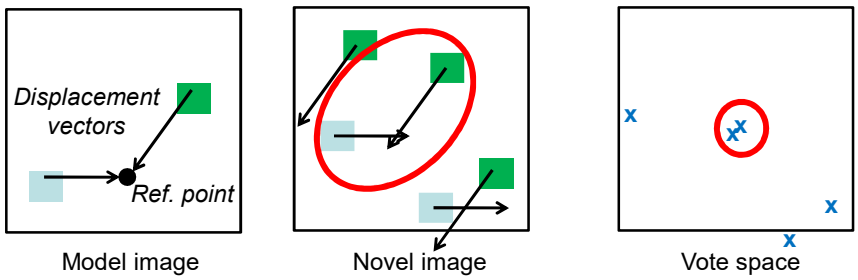
- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: can be tricky to pick a good grid size

Slide credit: Kristen Grauman

Generalized Hough Transform

- What if we want to detect arbitrary shapes?

Intuition:

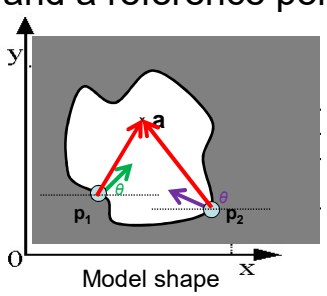


Now suppose those colors encode gradient directions…

Slide credit: Kristen Grauman

Generalized Hough Transform

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



θ	…
θ	…
⋮	

<u>Offline procedure:</u>

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

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

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

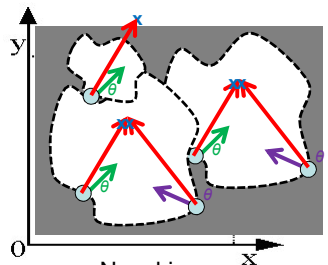
Slide credit: Kristen Grauman

Generalized Hough Transform

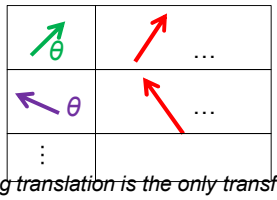
Detection procedure:

For each edge point:

- Use its gradient orientation θ to index into stored table
- Use retrieved **r** vectors to vote for reference point



Novel image

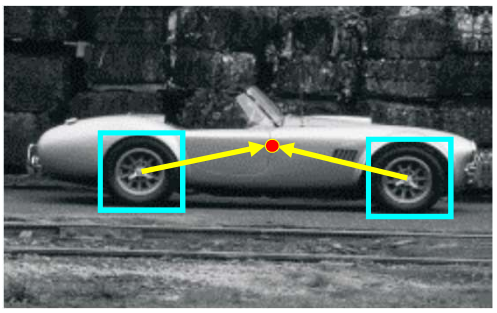


Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

Slide credit: Kristen Grauman

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

Source: L. 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

Source: L. Lazebnik

Visual Object Recognition Tutorial

Example: Results on Cows

Original image

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

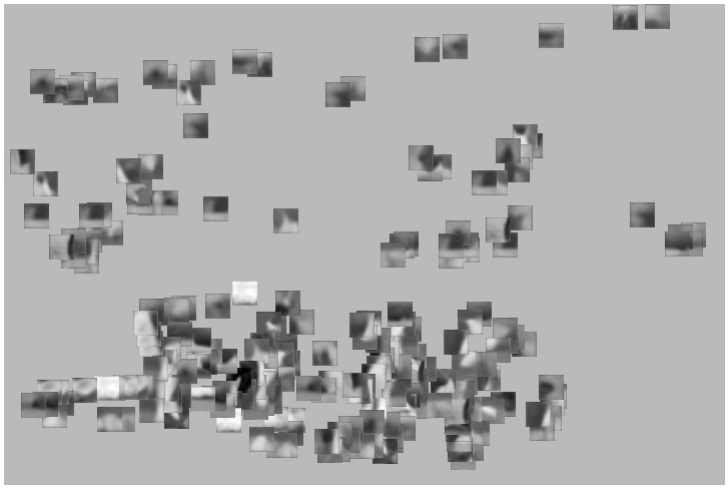
Example: Results on Cows

Interest points

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

Example: Results on Cows

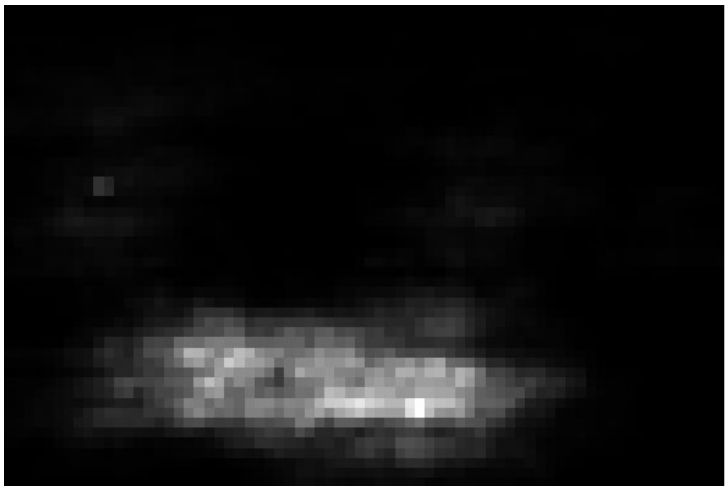


Matched patches

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

Example: Results on Cows



Votes

K. Grauman, B. Leibe

43

Visual Object Recognition Tutorial

Example: Results on Cows

1st hypothesis

K. Grauman, B. Leibe

44

Visual Object Recognition Tutorial

Example: Results on Cows

2nd hypothesis

K. Grauman, B. Leibe

45

Visual Object Recognition Tutorial

Example: Results on Cows

3rd hypothesis

K. Grauman, B. Leibe

46

Summary

- **Grouping/segmentation** useful to make a compact representation and merge similar features
  - associate features based on defined similarity measure and clustering objective
- **Fitting** problems require finding any supporting evidence for a model, even within clutter and missing features.
  - associate features with an explicit model
- **Voting** approaches, such as the **Hough transform**, make it possible to find likely model parameters without searching all combinations of features.
  - Hough transform approach for lines, circles, …, arbitrary shapes defined by a set of boundary points, recognition from patches.

Slide credit: Kristen Grauman

Coming up

Fitting with deformable contours