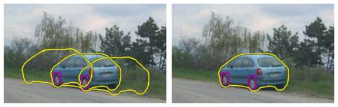


# Fitting:
## Voting and the Hough Transform

Thurs Feb 8, 2018

Kristen Grauman

UT Austin

## Last time

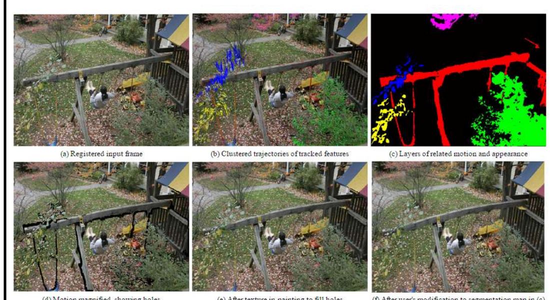
- Texture synthesis wrap up
- Optical flow: estimating motion in video

- Review:
  - What can we expect from an Nth order Markov field for texture synthesis (N > 1)?
  - What is the aperture problem?
  - What can cause flow errors at object boundaries?

## Recall: Motion estimation techniques

- Direct methods
  - Directly recover image motion at each pixel from spatio-temporal image brightness variations
  - Dense motion fields, but sensitive to appearance variations
  - Suitable for video and when image motion is small
- Feature-based methods
  - Extract visual features (corners, textured areas) and track them over multiple frames
  - Sparse motion fields, but more robust tracking
  - Suitable when image motion is large (10s of pixels)

## Motion magnification




Liu et al. SIGGRAPH 2005

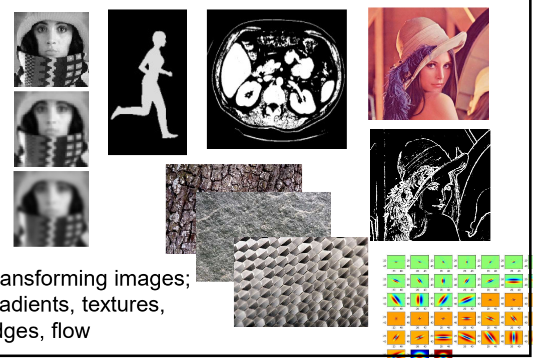
## original

## magnified

## Motion magnification


http://people.csail.mit.edu/mrub/vidmag/
Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, William T. Freeman. Eulerian Video Magnification for Revealing Subtle Changes in the World ACM Transactions on Graphics, Volume 31, Number 4 (Proc. SIGGRAPH), 2012

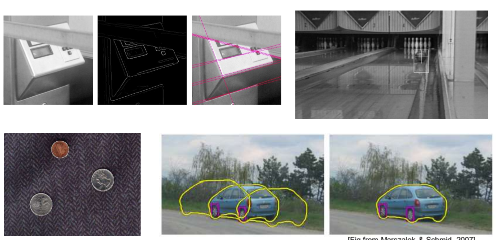
## So far: features and filters



Transforming images; gradients, textures, edges, flow

## 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.

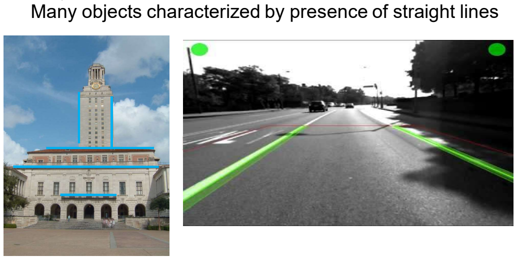
## Fitting: Main idea

- 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

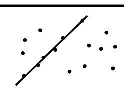
## Case study: Line fitting

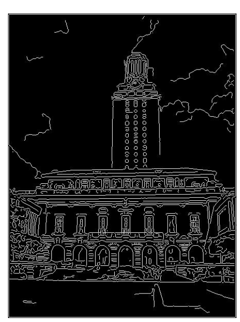
- Why fit lines?
  Many objects characterized by presence of straight lines



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

## Difficulty of line fitting



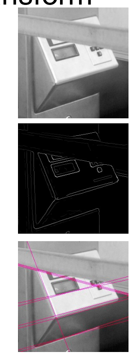


- **Extra** edge points (clutter), multiple models:
  – which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing:**
  – how to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
  – how to detect true underlying parameters?

## 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.

## 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.

## Finding lines in an image: Hough space

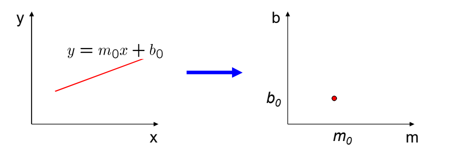



image space Hough (parameter) space

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

Slide credit: Steve Seitz

## Finding lines in an image: Hough space

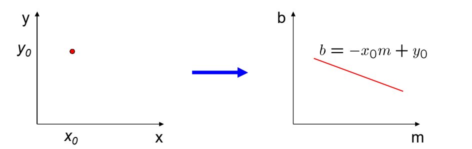



image space Hough (parameter) space

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

Slide credit: Steve Seitz

## Finding lines in an image: Hough space

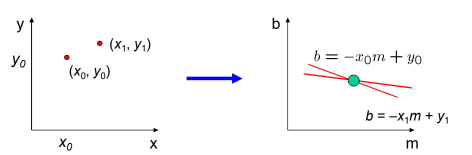



image space Hough (parameter) space

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

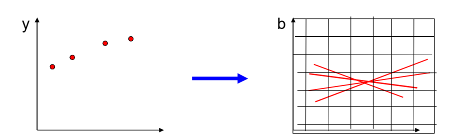
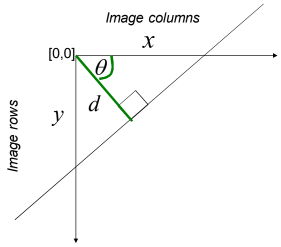



image space Hough (parameter) space

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

$\theta$: angle the perpendicular makes with the x-axis

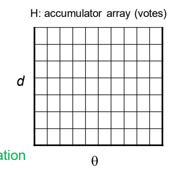
$$x\cos\theta - y\sin\theta = d$$

Point in image space → sinusoid segment in Hough space

## 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 = [$\theta_{min}$ to $\theta_{max}$ ] // 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

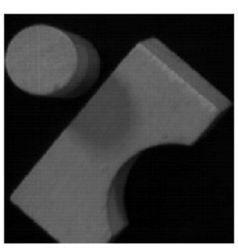

## Example: What was the shape?

Square :

*d*

θ

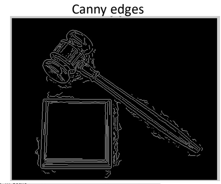
## Example: Hough transform for straight lines



Which line generated this peak?

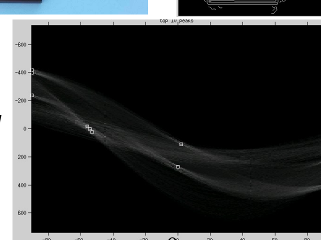
Original image

Canny edges

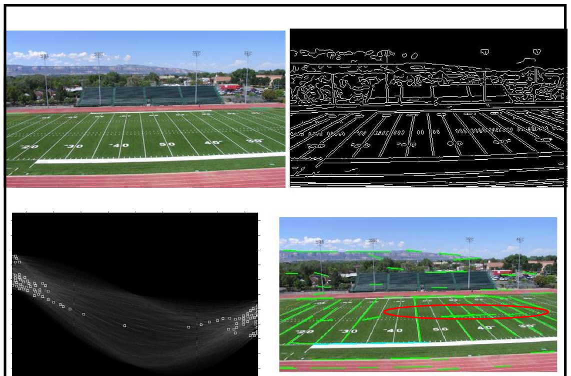


Decode the vote space.

*d*



θ



Showing longest segments found

## Impact of noise on Hough

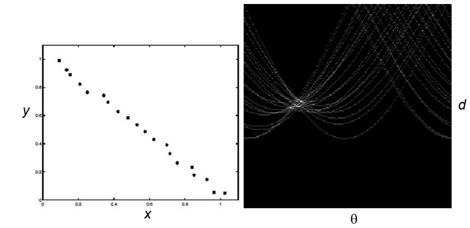



**Image space edge coordinates** **Votes**

What difficulty does this present for an implementation?

## Impact of noise on Hough

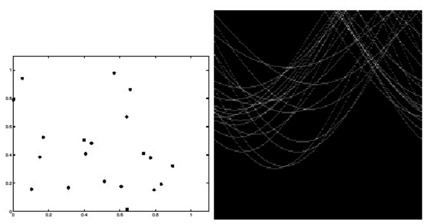


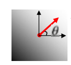
**Image space edge coordinates** **Votes**

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
   $\theta$ = gradient at (x,y)
   $d = x\cos\theta - y\sin\theta$
   H[d, $\theta$] += 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, $\theta$) based on image gradient at (x,y)
   H[d, $\theta$] += 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, $\theta$) 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

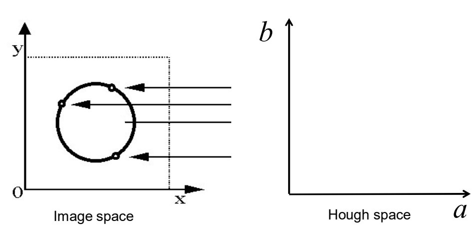


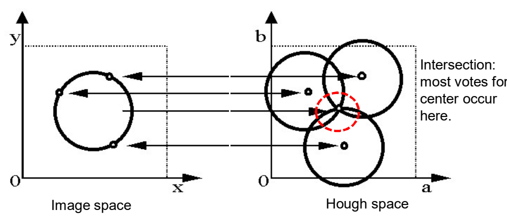

Image space Hough space

## 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

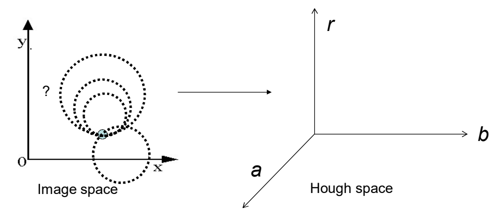



## Hough transform for circles

- 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

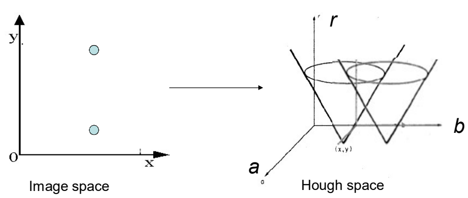



## Hough transform for circles

- 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

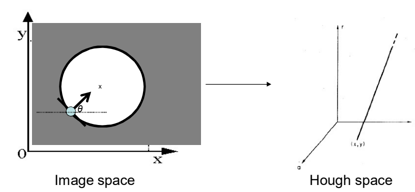



## Hough transform for circles

- 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




## Hough transform for circles

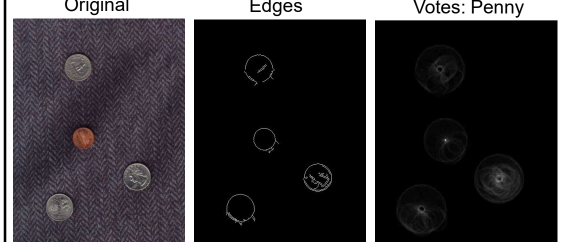
For every edge pixel $(x,y)$ :
  For each possible radius value $r$:
    For each possible gradient direction $\theta$:
     *// or use estimated gradient at (x,y)*
        $a = x + r \cos(\theta)$ // column
        $b = y - r \sin(\theta)$ // row
        $H[a,b,r]$ += 1
  end
end

Time complexity per edge pixel?

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

## Example: detecting circles with Hough

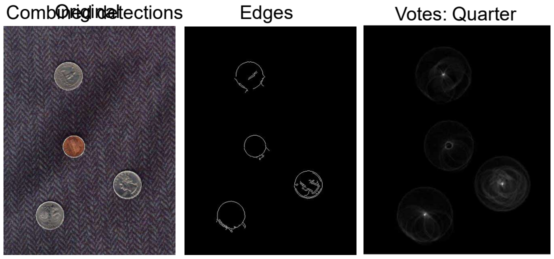
Original | Edges | Votes: Penny



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

## Example: detecting circles with Hough

Original / Combined detections Edges Votes: Quarter



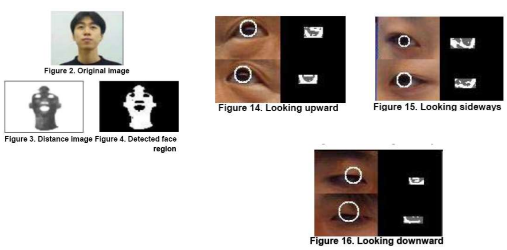
Coin finding sample images from: Vivek Kwatra

## 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




- 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.

## Hough transform: pros and cons

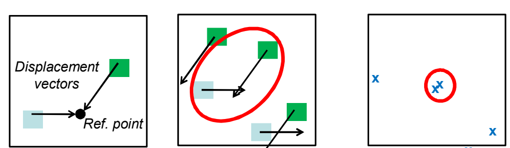
Pros
- All points are processed independently, so can cope with occlusion, gaps
- Some robustness to noise: noise points unlikely to contribute *consistently* to any single bin
- Can detect multiple instances of a model in a single pass

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

## Generalized Hough Transform

- What if we want to detect arbitrary shapes?

**Intuition:**

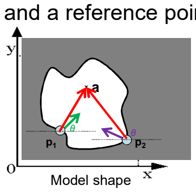



Model image Novel image Vote space

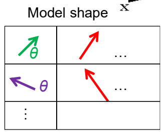
Now suppose those colors encode gradient directions…

## Generalized Hough Transform

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



Model shape




**Offline procedure:**

At each boundary point, compute displacement vector: **r = a – p$_i$**.

Store these vectors in a table indexed by gradient orientation $\theta$.

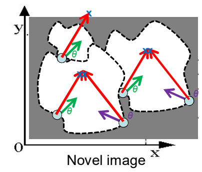
[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

## Generalized Hough Transform

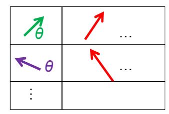
**Detection procedure:**

For each edge point:

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



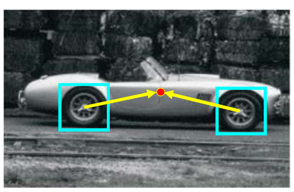
Novel image

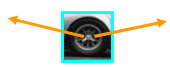



*Assuming translation is the only transformation here, i.e., orientation and scale are fixed.*

## 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

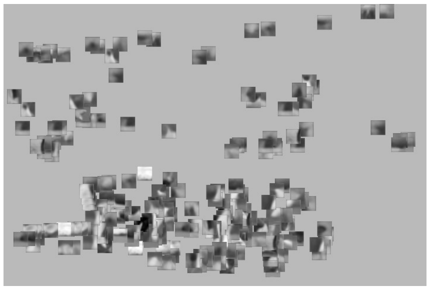
## Example: Results on Cows


**Interest points**

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

## Example: Results on Cows

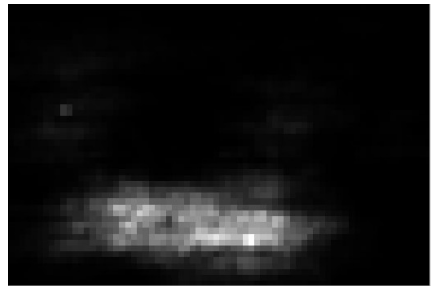


Matched patches

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

## Example: Results on Cows



Votes

K. Grauman, B. Leibe

52

Visual Object Recognition Tutorial

## Example: Results on Cows

1st hypothesis

K. Grauman, B. Leibe

53

Visual Object Recognition Tutorial

## Example: Results on Cows

2nd hypothesis

K. Grauman, B. Leibe

54

Visual Object Recognition Tutorial

## Example: Results on Cows

3rd hypothesis

K. Grauman, B. Leibe

55

## Summary

- **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, 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.