Fitting: Voting and the Hough Transform (part 2)

Tues Feb 13, 2018

Kristen Grauman

UT Austin

Last time

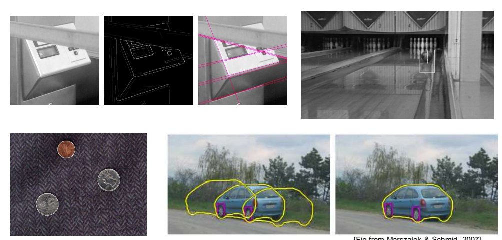
- **Fitting** problems require finding any supporting evidence for a model, even within clutter and missing features.
  - associate features with an explicit model
- **Voting** approaches, such as the Hough transform, find likely model parameters without searching all combinations of features.
  - Hough transform approach for lines, circles, …, arbitrary shapes defined by a set of boundary points, recognition from patches.

Outline

- Last time:
  - Fitting: voting and the Hough transform
    - For lines
    - For circles
- Today:
  - Review of Hough circles
  - Generalized Hough algorithm for any shape
  - Background subtraction

Fitting

- Want to associate a model with observed features



[Fig from Marszalek & Schmid, 2007]

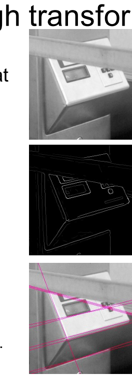
For example, the model could be a line, a circle, or an arbitrary shape.

Recall -- 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

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.

Finding lines in an image: Hough space

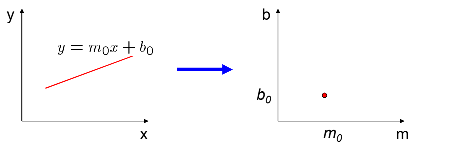


image space Hough (parameter) space

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

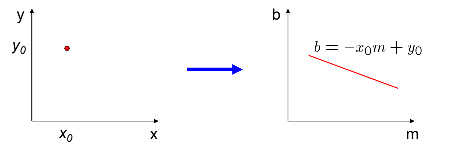


image space Hough (parameter) space

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

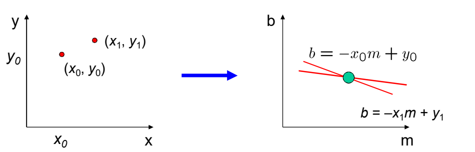


image space Hough (parameter) space

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

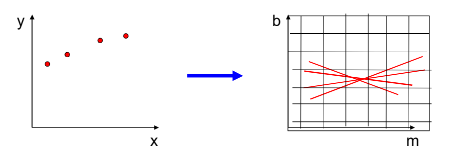


image space Hough (parameter) space

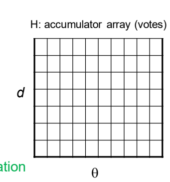
How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

- 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.

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
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)?

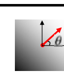
Source: Steve Seitz

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

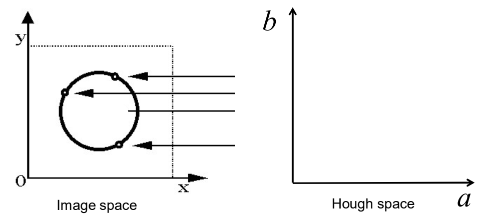
Extension 4

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

Source: Steve Seitz

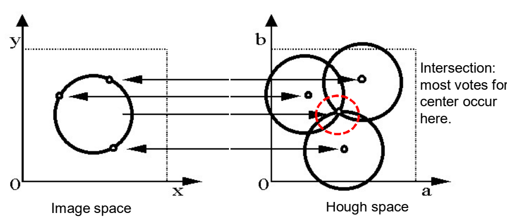
Hough transform for circles

- 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



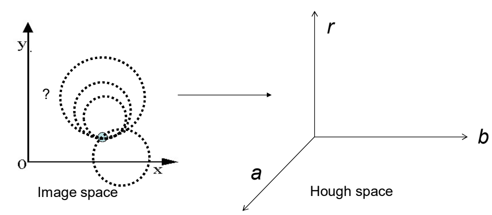
Hough transform for circles

- 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



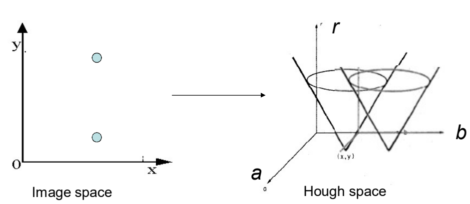
Hough transform for circles

- 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



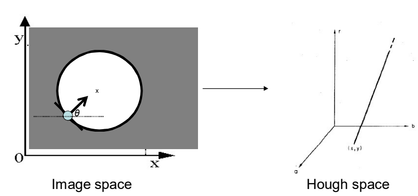
Hough transform for circles

- 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

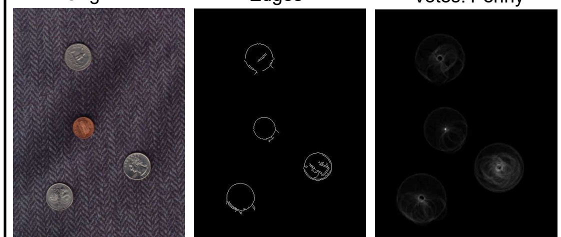
For every edge pixel (x,y) :
  For each possible radius value r:
    For each possible gradient direction θ:
    // or use estimated gradient at (x,y)
      $a = x + r\cos(\theta)$ // column
      $b = y - r\sin(\theta)$ // row
      H[a,b,r] += 1
  end
end

Time complexity per edge pixel?

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

Example: detecting circles with Hough

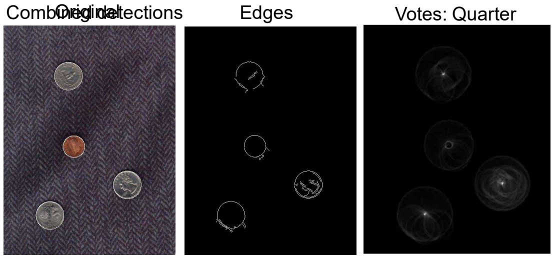
Original Edges Votes: Penny



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

Example: detecting circles with Hough

Combined detections Original 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

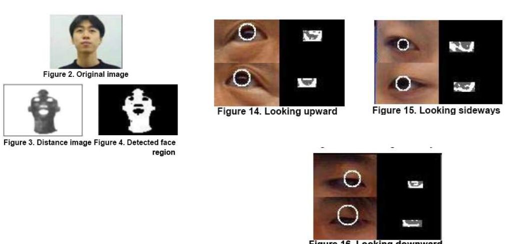


Figure 2. Original image
Figure 3. Distance image Figure 4. Detected face region
Figure 14. Looking upward
Figure 15. Looking sideways
Figure 16. Looking downward

- 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.

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

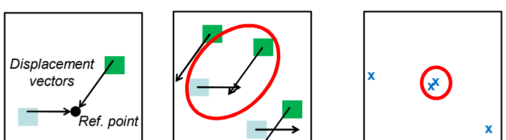
Outline

- Last time:
  - Fitting: voting and the Hough transform
    - For lines
    - For circles
- Today:
  - Review of Hough circles
  - Generalized Hough algorithm for any shape
  - Background subtraction

Generalized Hough Transform

- What if we want to detect arbitrary shapes?

Intuition:

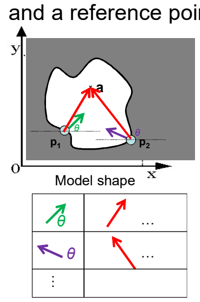


Model image Novel image Vote space

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: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$.

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

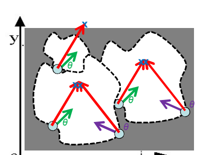
[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]

Generalized Hough Transform

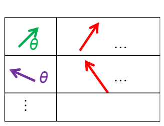
Detection procedure:

For each edge point:

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



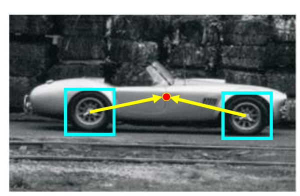
Novel image



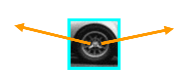
Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

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

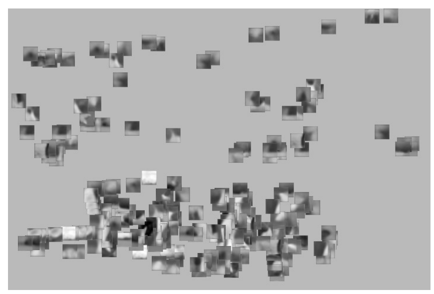
Example: Results on Cows

Interest points

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

Example: Results on Cows

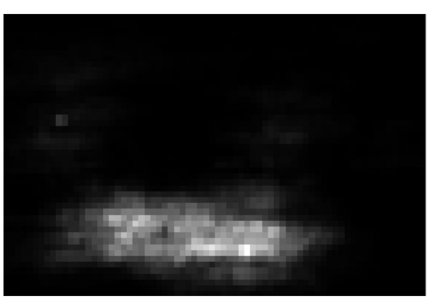


Matched patches

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

Example: Results on Cows



Votes 5

K. Grauman, B. Leibe

37

Visual Object Recognition Tutorial

Example: Results on Cows

1st hypothesis

K. Grauman, B. Leibe

38

Example: Results on Cows

2nd hypothesis

K. Grauman, B. Leibe

Visual Object Recognition Tutorial

39

Example: Results on Cows

3rd hypothesis

K. Grauman, B. Leibe

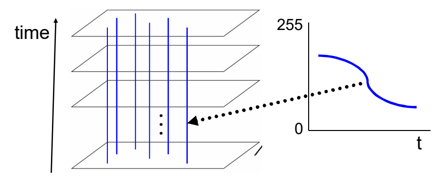
Visual Object Recognition Tutorial

40

Outline

- Last time:
  - Fitting: voting and the Hough transform
    - For lines
    - For circles
- Today:
  - Review of Hough circles
  - Generalized Hough algorithm for any shape
  - Background subtraction

Video as an “Image Stack”

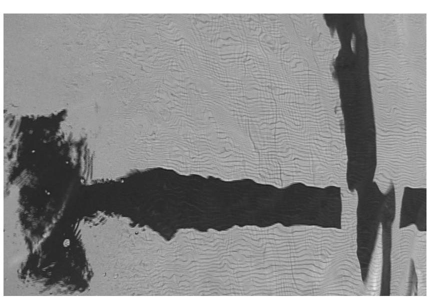


Can look at video data as a spatio-temporal volume

- If camera is stationary, each line through time corresponds to a single ray in space

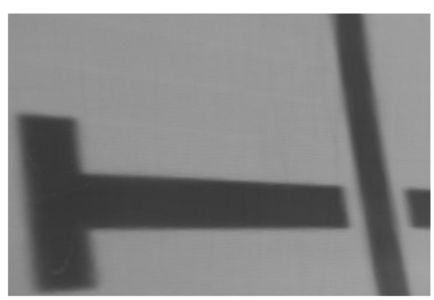
Alyosha Efros, CMU

Input Video



Alyosha Efros, CMU

Average Image



Alyosha Efros, CMU

Background Subtraction

- Given an image (mostly likely to be a video frame), we want to identify the **foreground objects** in that image!

$\Rightarrow$

Motivation

- In most cases, objects are of interest, not the scene.
- Makes our life easier: less processing costs, and less room for error.

Slide credit: Birgi Tamersoy

Background subtraction

- Simple techniques can do ok with static camera
- …But hard to do perfectly

- Widely used:
  - Traffic monitoring (counting vehicles, detecting & tracking vehicles, pedestrians),
  - Human action recognition (run, walk, jump, squat),
  - Human-computer interaction
  - Object tracking

Simple Approach

Image at time t: $I(x,y,t)$ $\Downarrow$

Background at time t: $B(x,y,t)$ $\Downarrow$

$|$  $-$  $| > Th$

1. Estimate the background for time t.
2. Subtract the estimated background from the input frame.
3. Apply a threshold, Th, to the absolute difference to get the **foreground mask**.

Slide credit: Birgi Tamersoy

Frame Differencing

- Background is estimated to be the previous frame. Background subtraction equation then becomes:

$$B(x,y,t) = I(x,y,t-1)$$
$$\Downarrow$$
$$|I(x,y,t) - I(x,y,t-1)| > Th$$

- Depending on the object structure, speed, frame rate and global threshold, this approach may or may **not** be useful (usually **not**).

$|$  $-$  $| > Th$

Slide credit: Birgi Tamersoy

Frame Differencing

$Th = 25$

$Th = 50$

$Th = 100$

$Th = 200$

Slide credit: Birgi Tamersoy

Mean Filter

- In this case the background is the mean of the previous n frames:

$$B(x,y,t) = \frac{1}{n}\sum_{i=0}^{n-1} I(x,y,t-i)$$
$$\Downarrow$$
$$|I(x,y,t) - \frac{1}{n}\sum_{i=0}^{n-1} I(x,y,t-i)| > Th$$

- For $n = 10$:

Estimated Background

Foreground Mask

Slide credit: Birgi Tamersoy

Median Filter

- Assuming that the background is more likely to appear in a scene, we can use the median of the previous n frames as the background model:

$$B(x,y,t) = median\{I(x,y,t-i)\}$$
$$\Downarrow$$
$$|I(x,y,t) - median\{I(x,y,t-i)\}| > Th \text{ where } i \in \{0,\ldots,n-1\}.$$

- For $n = 10$:

Estimated Background

Foreground Mask

Slide credit: Birgi Tamersoy

Average/Median Image

Alyosha Efros, CMU

Background Subtraction

-

=

Alyosha Efros, CMU

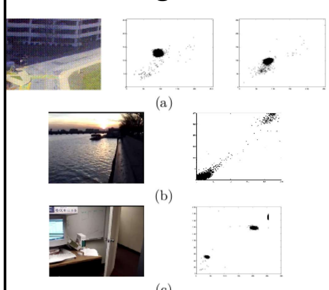
Pros and cons

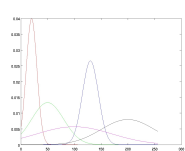
Advantages:
- Extremely easy to implement and use!
- All pretty fast.
- Corresponding background models need not be constant, they change over time.

Disadvantages:
- Accuracy of frame differencing depends on object speed and frame rate
- Median background model: relatively high memory requirements.
- Setting global threshold Th…

When will this basic approach fail?

Background mixture models





Idea: model each background pixel with a *mixture* of Gaussians; update its parameters over time.

Adaptive Background Mixture Models for Real-Time Tracking, Chris Stauffer & W.E.L. Grimson

Coming up

- Deformable contours