4. Lines & Hough Transform

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Week

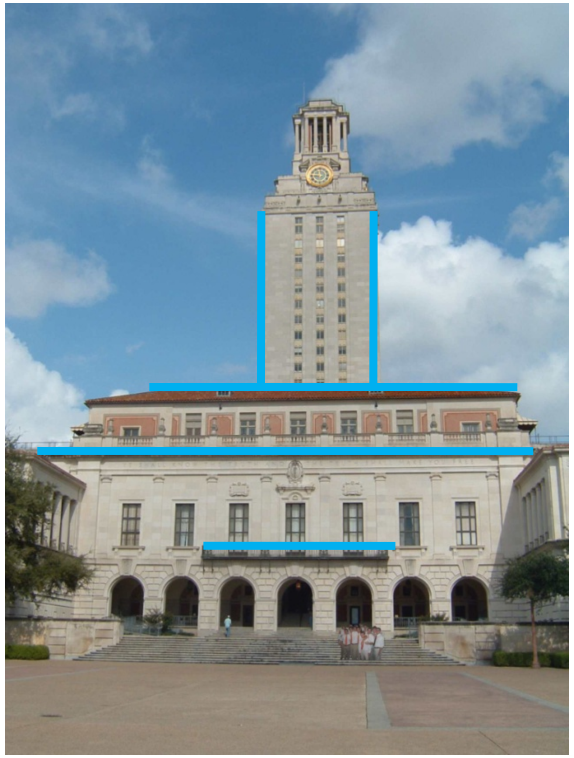
- Motivation for studying image edges
- Derivative filters to extract gradients
- Need for smoothing
- Canny edge detector

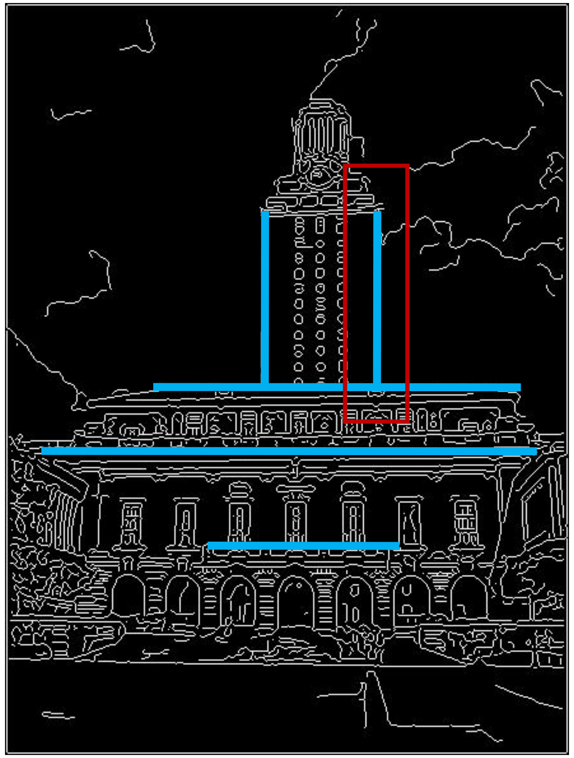
Today's Lecture

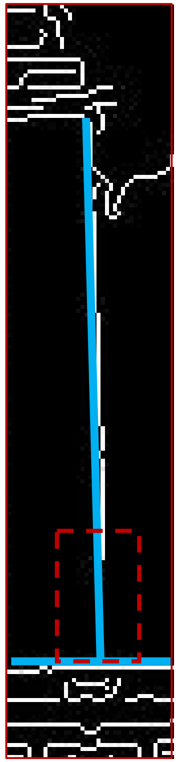
- Fitting straight lines in images
- Mathematics of lines
- Hough voting for straight lines
- Hough voting for circles
- Generalized Hough Transform

Motivation: Fitting (Straight) Lines

Many objects characterized by presence of straight lines.







Q: Doesn't edge detection already give us the lines?

A: Canny gives us **all** edges (lines, curves, noise, etc.)

Fitting Lines

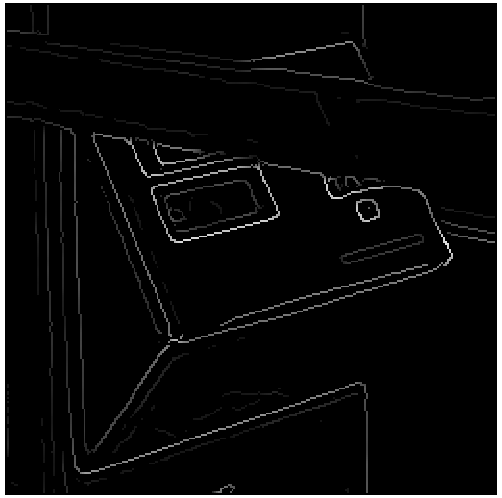
- Which points belong to which line?
- What about missing edge points and noise?

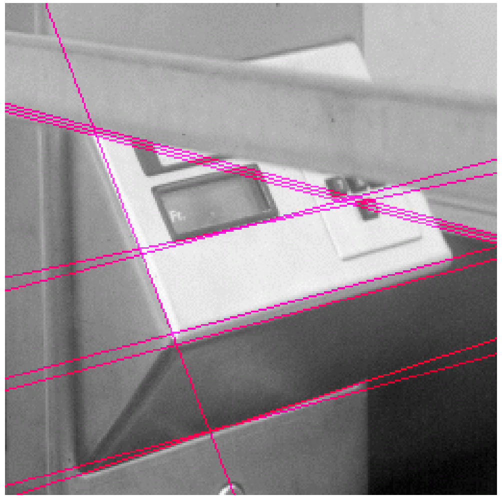
Adapted from K. Grauman

Hough Transform for Line Fitting

- 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 to answer all of these questions.
  - For each edge point, record a vote for each possible line that passes through that point
  - Look for lines which receive many votes.





Adapted from K. Grauman

The Mathematics of Lines

Slope-intercept form: y = mx + b

Normal form: $x \cos\theta + y \sin\theta = \rho$

Line Fitting

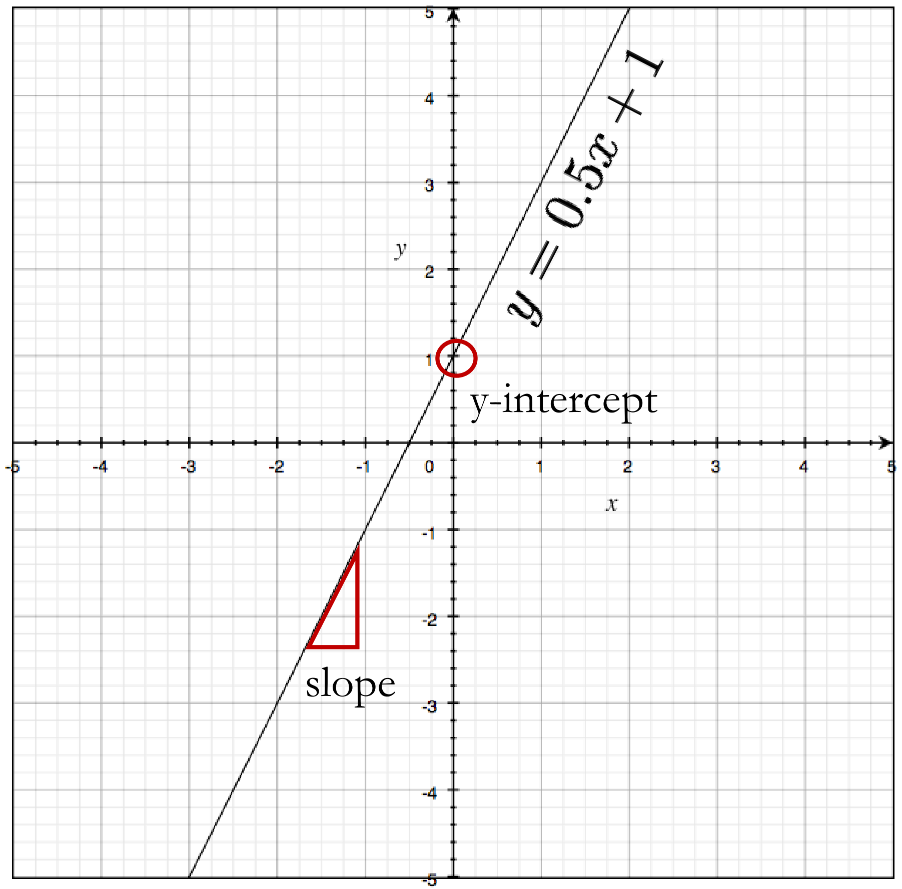
Slope Intercept Form

$$y = mx + b$$

slope   y-intercept

$$y = 0.5x + 1$$

How to get m & b?



Slide Credit: K. Kitani

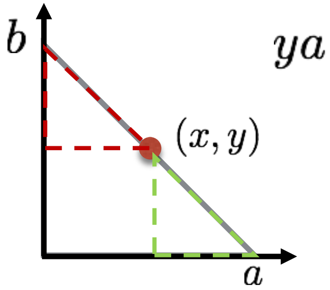
Double Intercept Form

$$\frac{x}{a} + \frac{y}{b} = 1$$

x-intercept

y-intercept

Derivation:

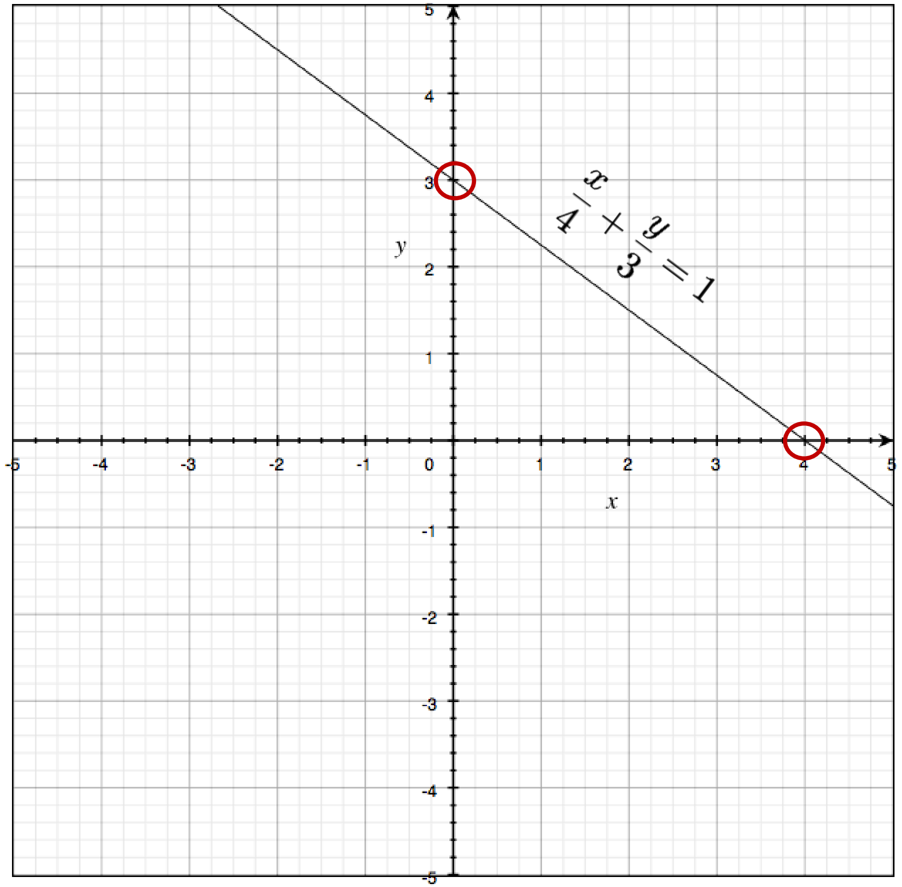


(Same slope) $$\frac{y-b}{x-0} = \frac{0-y}{a-x}$$

$$ya + yx - ba + bx = -yx$$

$$ya + bx = ba$$

$$\frac{y}{b} + \frac{x}{a} = 1$$



Slide Credit: K. Kitani

Normal Form

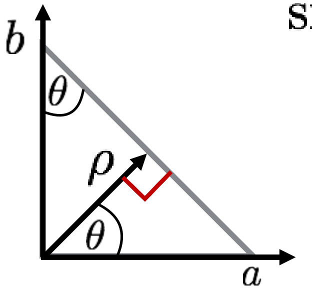
$$x \cos\theta + y \sin\theta = \rho$$

angle length

Derivation:

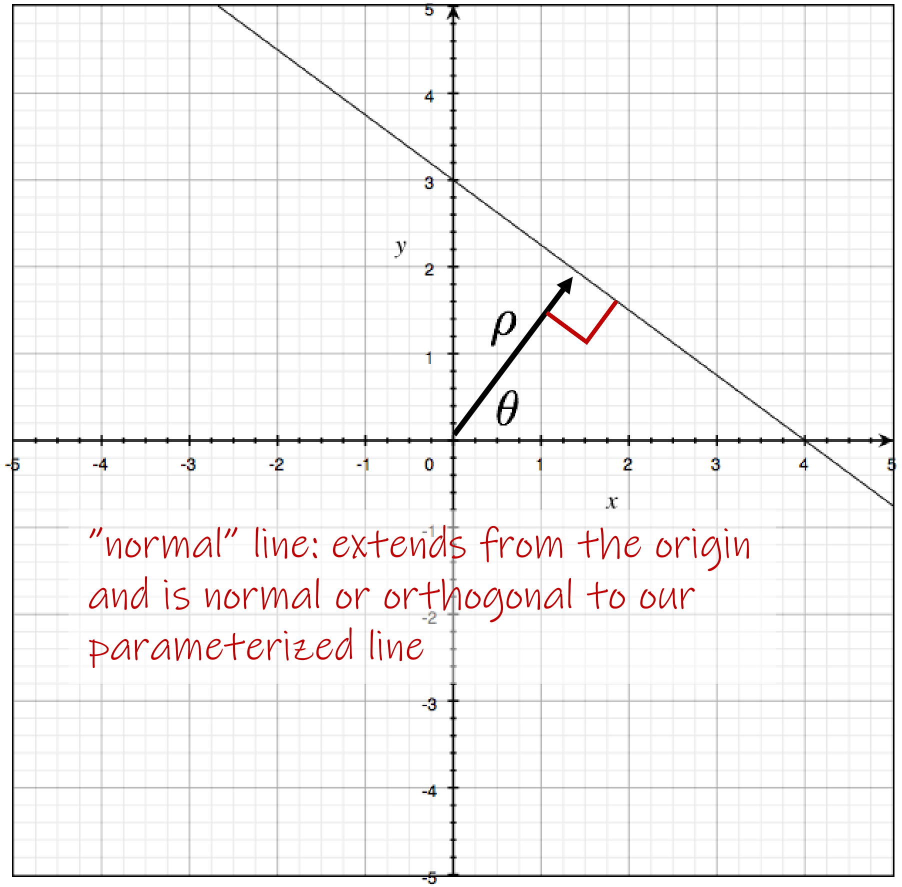
$$\cos\theta = \frac{\rho}{a} \rightarrow a = \frac{\rho}{\cos\theta}$$

$$\sin\theta = \frac{\rho}{b} \rightarrow b = \frac{\rho}{\sin\theta}$$



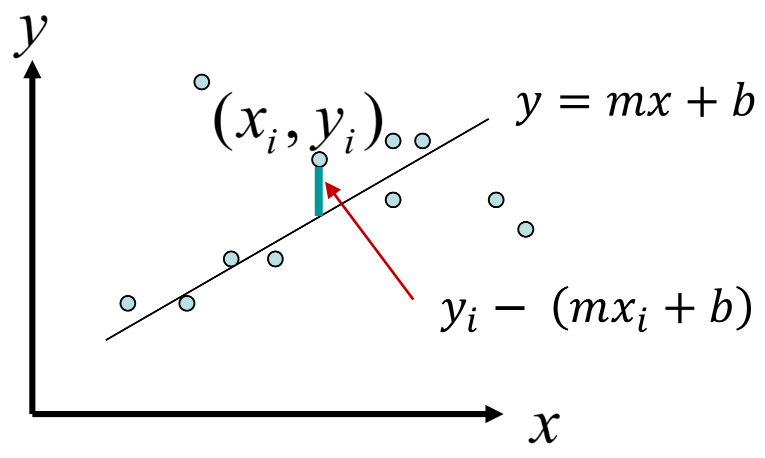
plug into: $$\frac{x}{a} + \frac{y}{b} = 1$$

$$x \cos\theta + y \sin\theta = \rho$$



Slide Credit: K. Kitani

Simple Line Fitting



Given: Many (x_i, y_i) pairs

point index

Find: Parameters (m, b)

Minimize: average square distance $E = \frac{1}{N}\sum_i [y_i - (mx_i + b)]^2$ $\quad (m, b) = \underset{m,b}{\operatorname{argmin}} E$

total number of points

Using: $\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial b} = 0$

$$b = \bar{y} - m\bar{x} \qquad \bar{y} = \frac{\sum_i y_i}{N} \qquad \bar{x} = \frac{\sum_i x}{N}$$

$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

Any problems with this approach?

https://mathworld.wolfram.com/LeastSquaresFitting.html

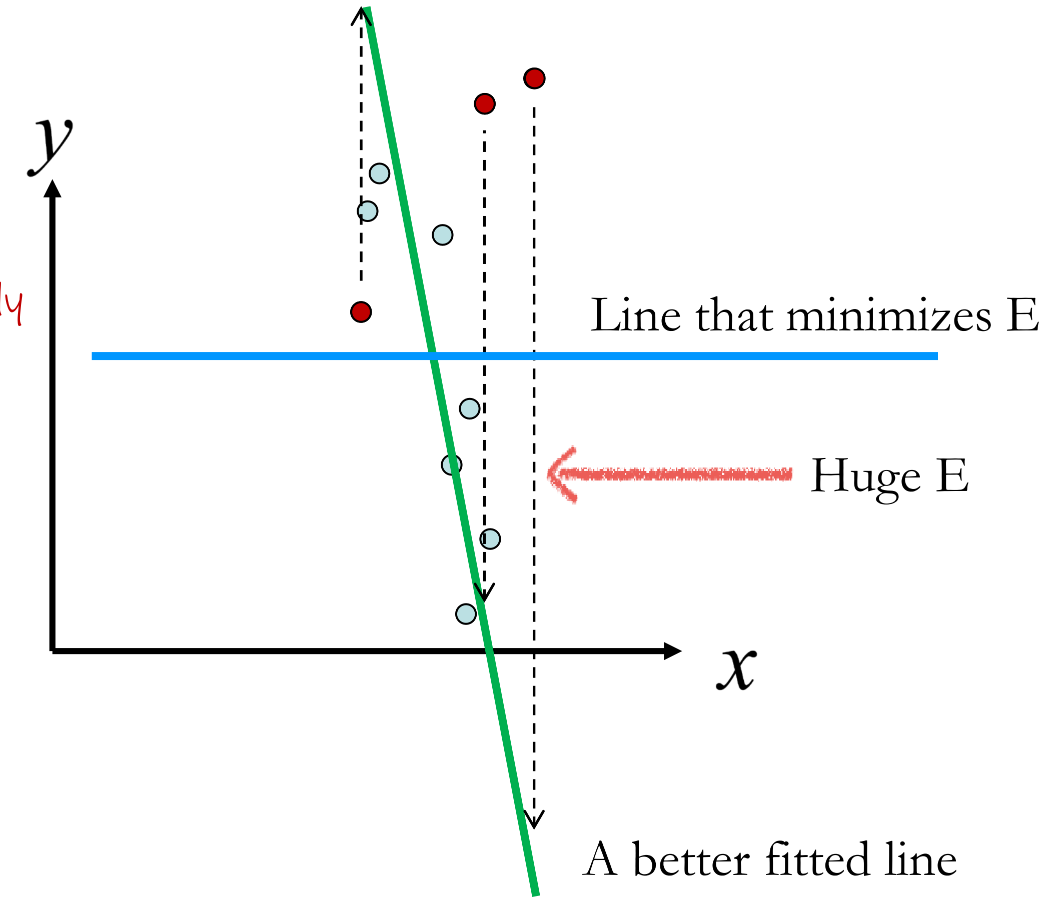
Modified from K. Kitani

Problems with Simple Line Fitting

Where is the line that minimizes E?

$$E = \frac{1}{N}\sum_i [y_i - (mx_i + b)]^2$$

Square term heavily penalizes outliers



- Error E must be formulated carefully.
- Reduce the impact of outliers
  - Voting-based approach
  - RANSAC (later lecture)

Modified from K. Kitani

Hough Transform for Lines

Image vs. parameter spaces

Normal parameterization for lines

Robustness to noise

Image Space vs. Parameter Space

variables

$$y = mx + b \xrightarrow{b = (-x)m+y} y - mx = b$$

parameters

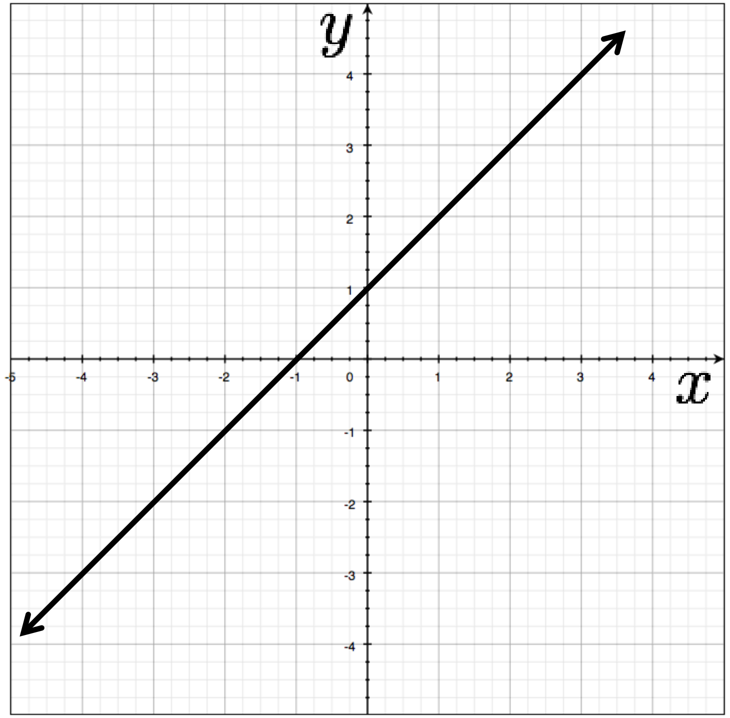
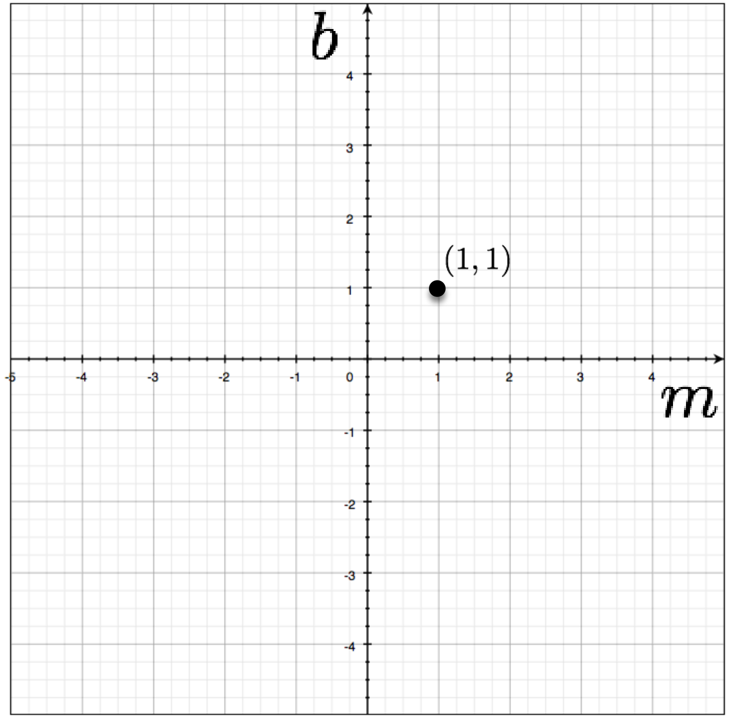


Image space

a line becomes a point



Parameter space

Slide Credit: K. Kitani

Image Space vs. Parameter Space

variables

$$y = mx + b$$

parameters

variables

$$y - mx = b$$

parameters

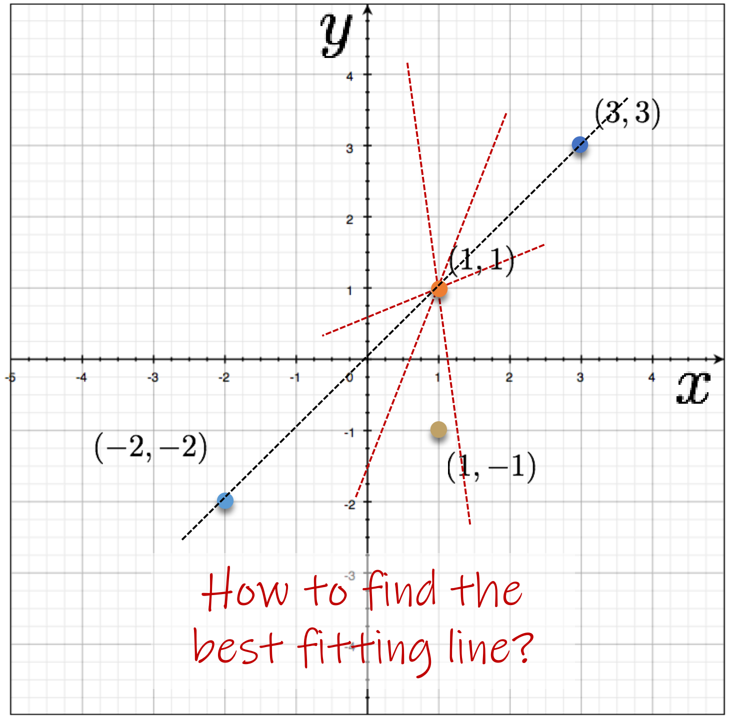
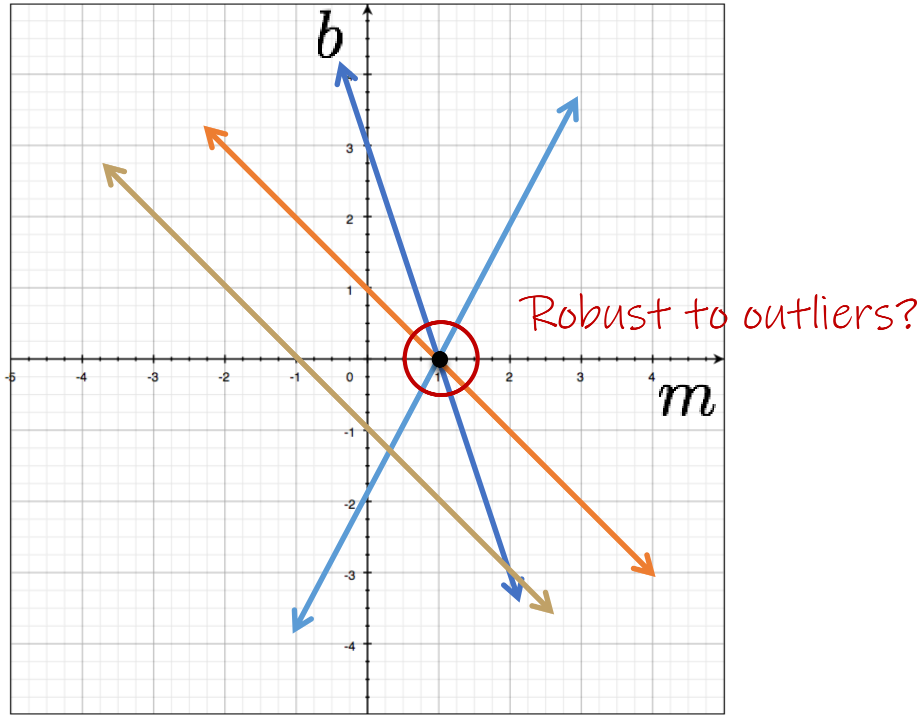


Image space

a point becomes a line



Parameter space

Slide Credit: K. Kitani

Line Detection with Hough Transform

Algorithm:

1. Quantize Parameter Space (m, b)

2. Create Accumulator Array $A(m, b)$

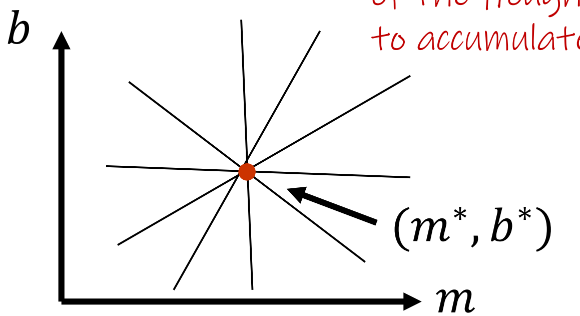
3. Set $A(m, b) = 0 \quad \forall\, m, c$

4. For each image edge point (x_i, y_i)
   For each element m
   Solve $b = x_i m + y_i$
   Increment $A(m, b) = A(m, b) + 1$

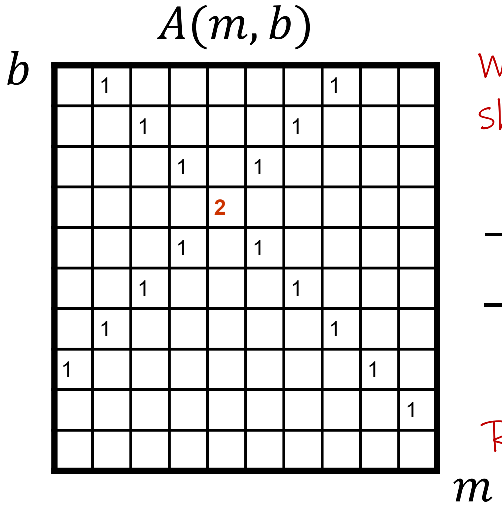
5. Threshold & find local maxima in $A(m, b)$

6. Detected line(s) given by $y = m^* x + b^*$

Accumulator is a discretization of the Hough parameter space to accumulate votes



Hough Parameter Space



What range of (m, b) should accumulator cover?

$-\infty \leq b \leq +\infty$

$-\infty \leq m \leq +\infty$

vertical line

Range is infinite!

Alternative parameter space?

Normal Parameterization

variables

$$y = mx + b$$

parameters

parameters

$$x \cos\theta + y \sin\theta = \rho$$

variables

Finite range

$0 \leq \theta \leq 2\pi$

$0 \leq \rho \leq \rho_{max}$

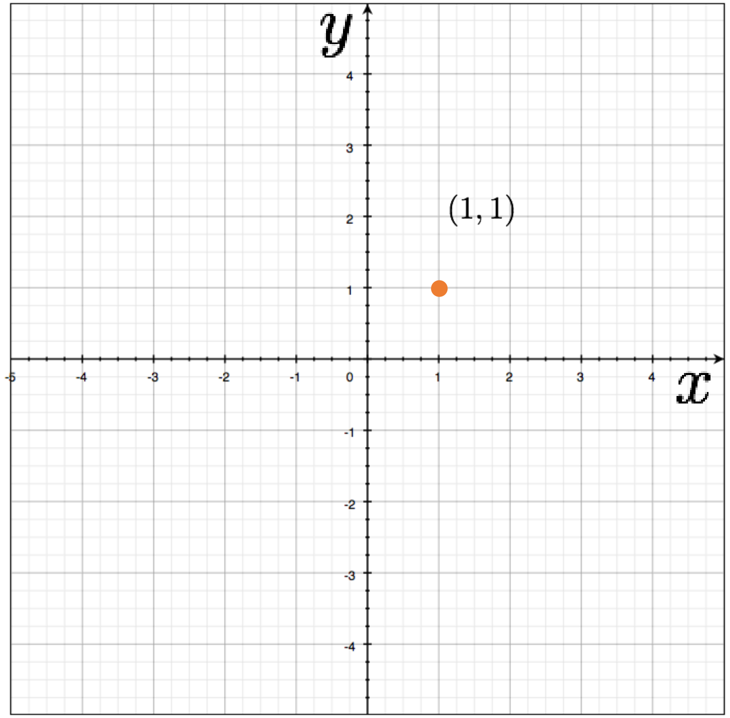
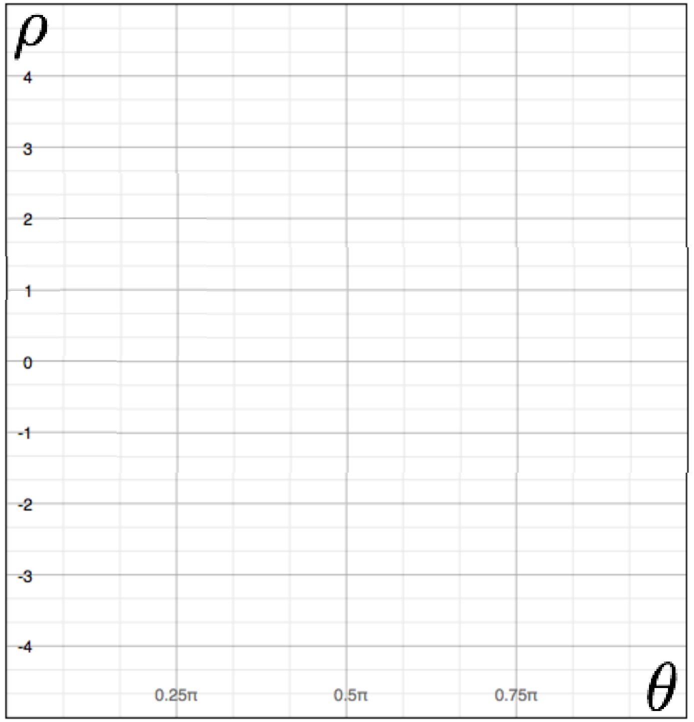


Image space

a point
becomes?
a sinusoid



Parameter space

Slide Credit: K. Kitani

A Point to a Sinusoid

variables

$$y = mx + b$$

parameters

parameters

$$x\cos\theta + y\sin\theta = \rho$$

variables

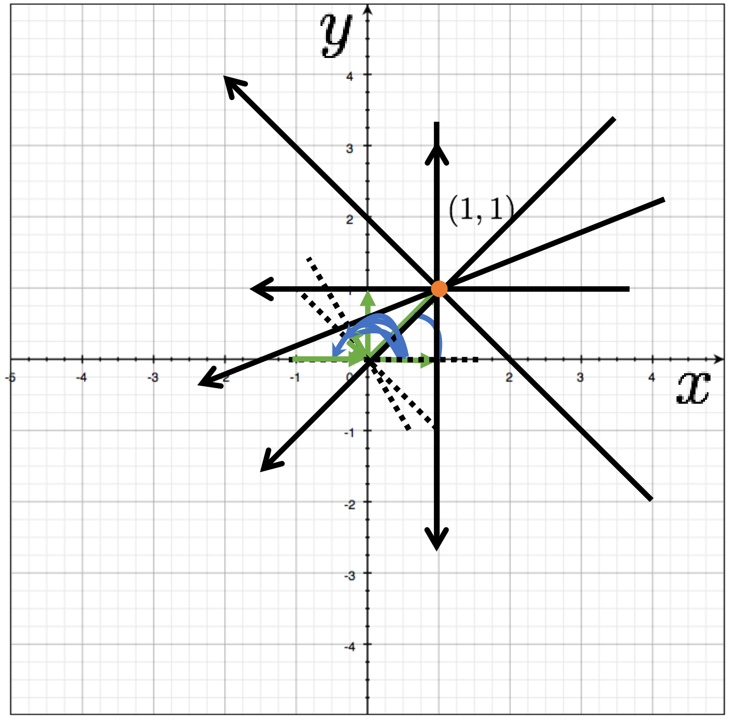
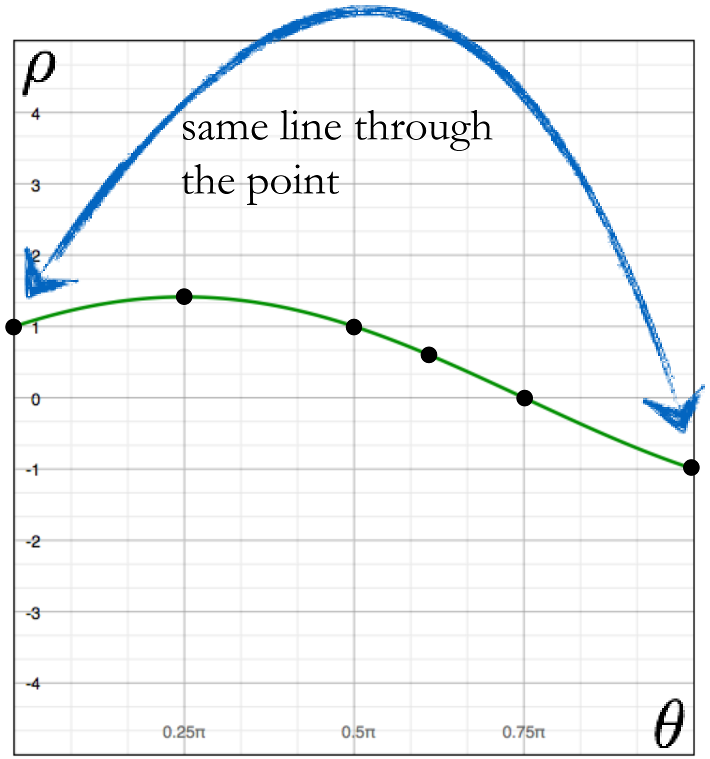


Image space

a point becomes?

a sinusoid



Parameter space

How can we have a negative rho?

Slide Credit: K. Kitani

There are two ways to write the same line:

Positive rho version:

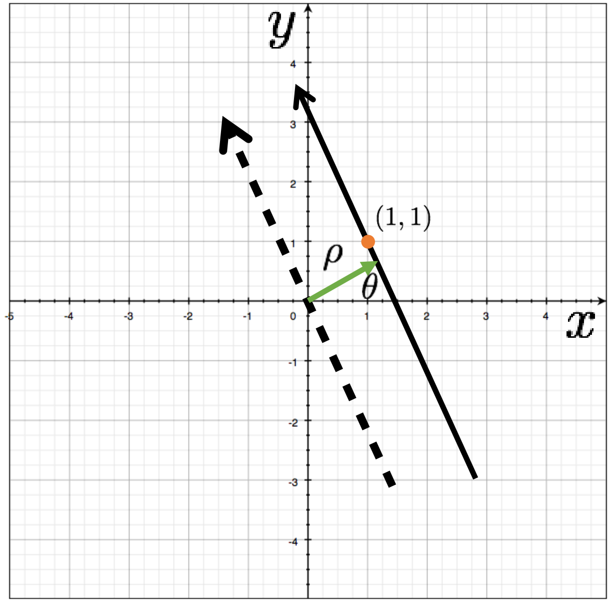
$$x \cos\theta + y \sin\theta = \rho$$

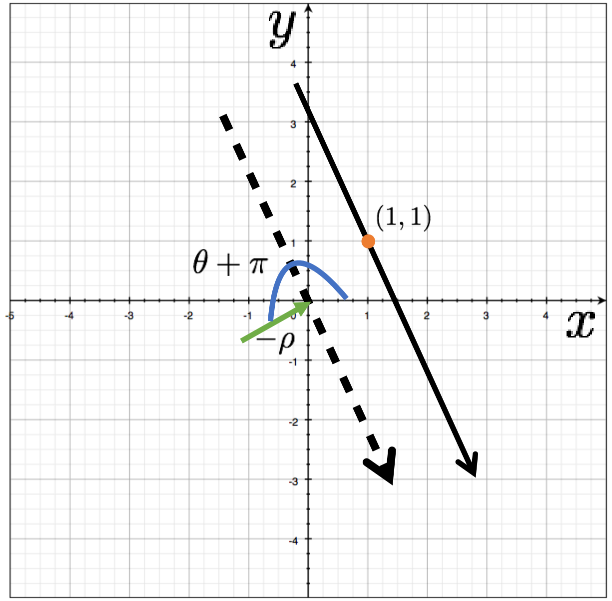
Recall:

$$\sin(\theta) = -\sin(\theta + \pi)$$
$$\cos(\theta) = -\cos(\theta + \pi)$$

Negative rho version:

$$x \cos(\theta + \pi) + y \sin(\theta + \pi) = -\rho$$





Slide Credit: K. Kitani

Image and (Normal) Parameter Space

variables

$$y = mx + b$$

parameters

parameters

$$x\cos\theta + y\sin\theta = \rho$$

variables

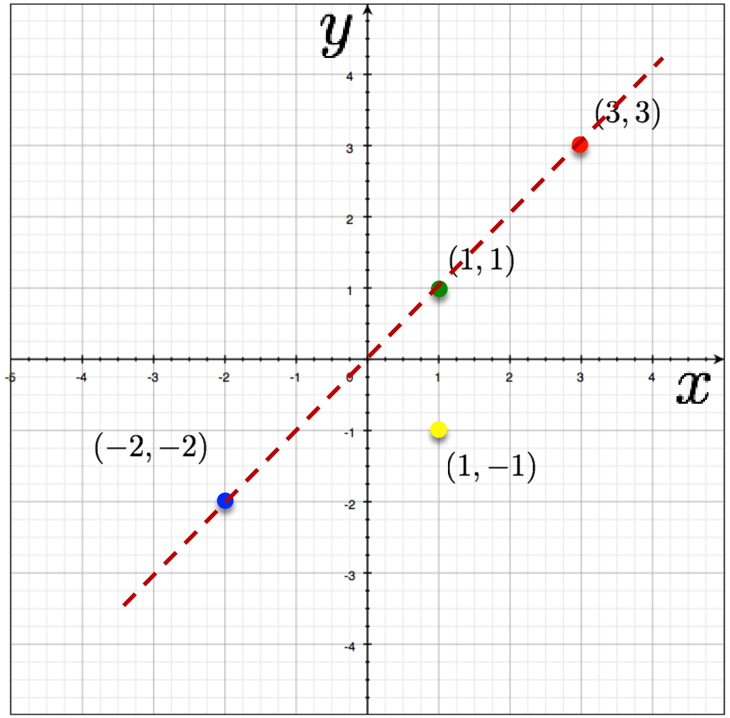
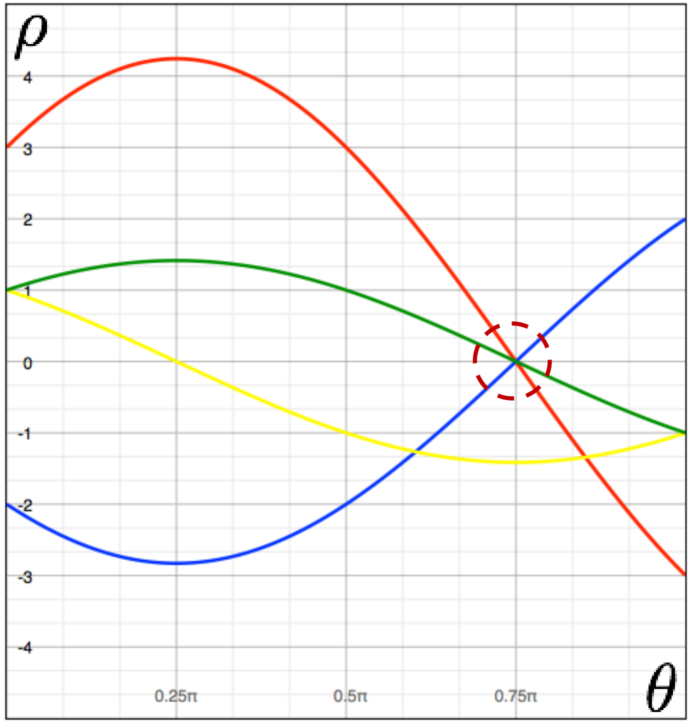


Image space

Points (on a line) become? Intersecting sinusoids



Parameter space

Slide Credit: K. Kitani

Normal Form

Line Detection with Hough Transform

Algorithm:

1. Quantize Parameter Space (θ, ρ)

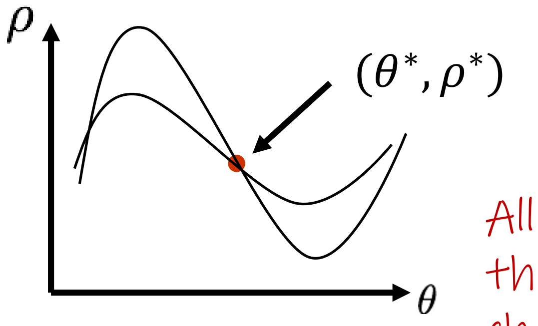
2. Create Accumulator Array $A(\theta, \rho)$

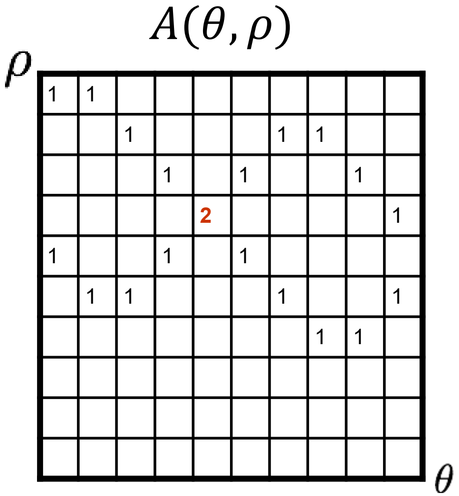
3. Set $A(\theta, \rho) = 0 \quad \forall \theta, \rho$

4. For each image edge point (x_i, y_i)
   For each element θ
   Solve $\rho = x_i \cos\theta + y_i \sin\theta$
   Increment $A(\theta,\rho) = A(\theta,\rho) + 1$

5. Threshold, find local maxima in $A(\theta,\rho)$

6. Detected line(s) given by $\rho^* = x\cos\theta^* + y\sin\theta^*$





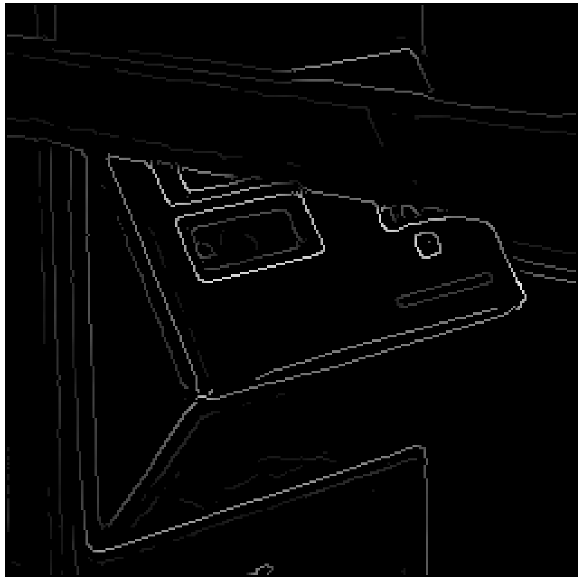
Hough Parameter Space

All steps remain the same, just a change in the parameterization of the Hough space!

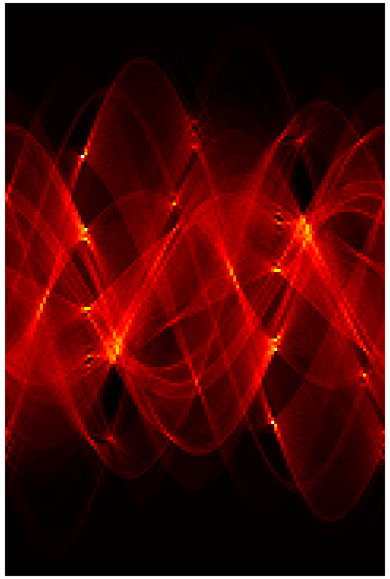
Real-world example

Where do "multiples" of the same line come from?

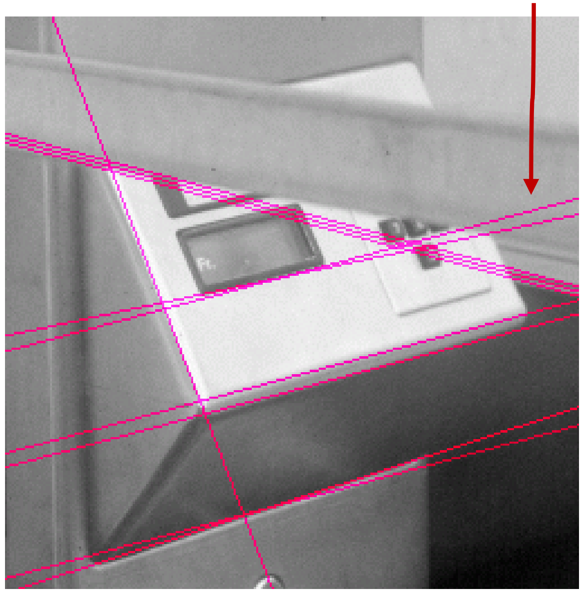
Original



Edges



Hough Space

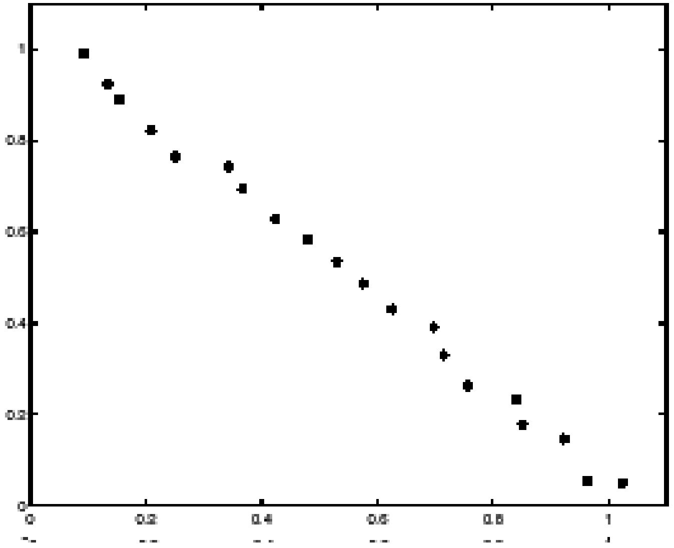


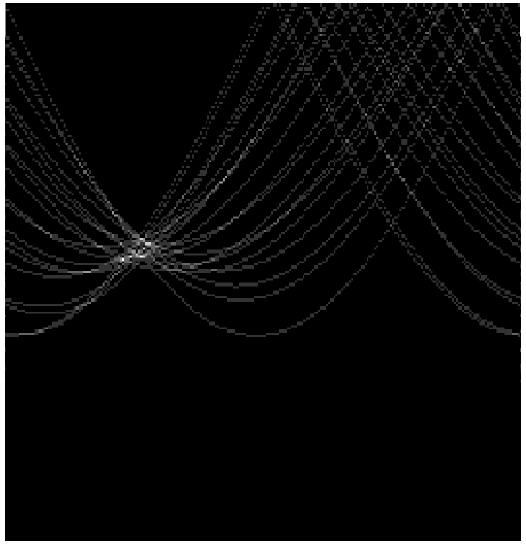
Hough Lines

Adapted from K. Grauman

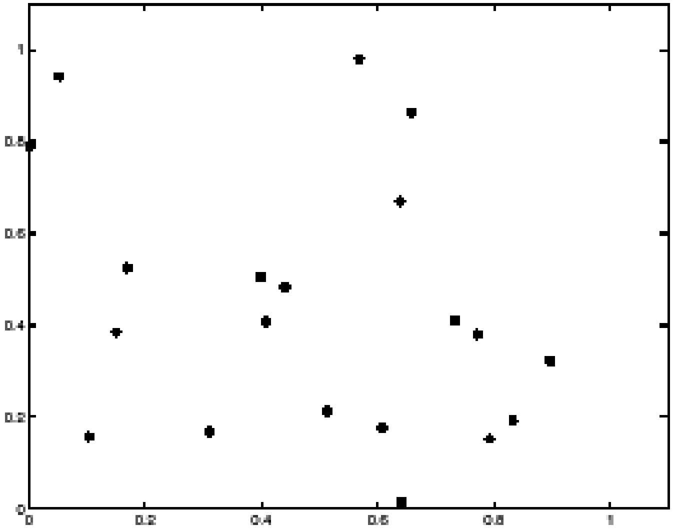
Hough Transform & Noise

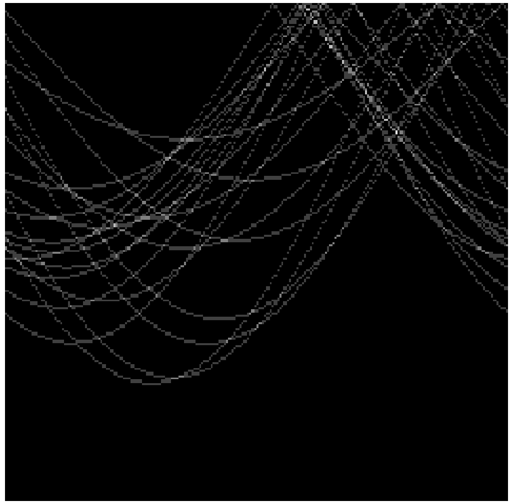
In practice, measurements are noisy…





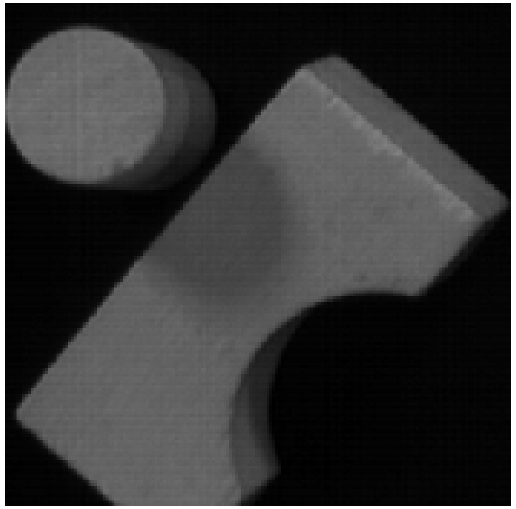
Everything looks like random edge points, but we can still discern peaks in the Hough space

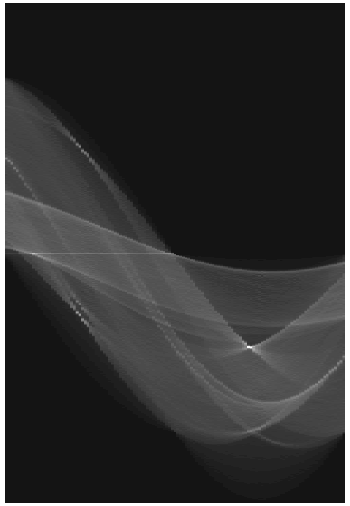


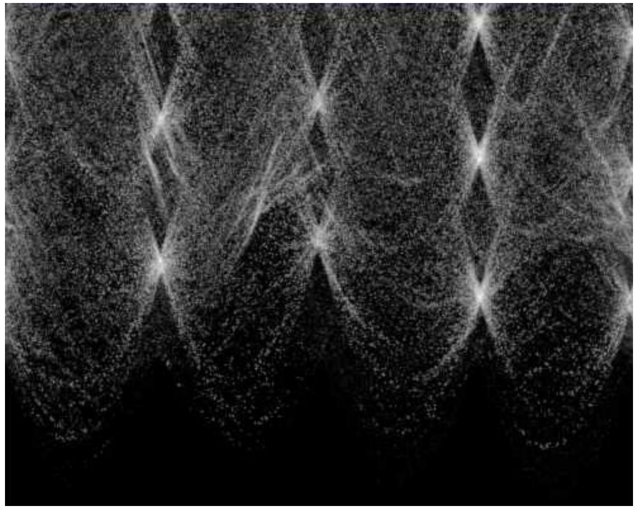


Adapted from K. Grauman

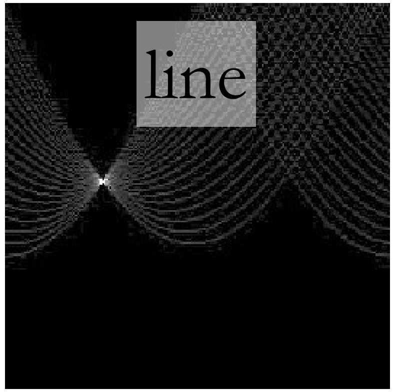
Shapes in Parameter Space

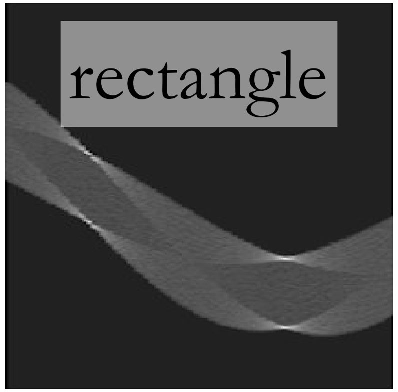


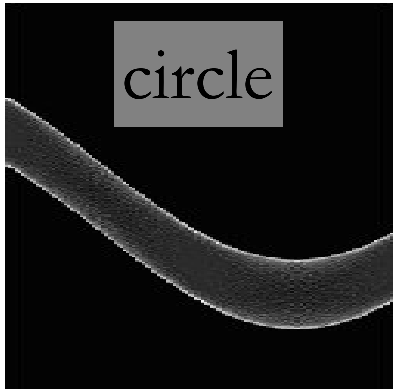




Can you guess the shape?







How to find a circle in an image? No longer a (single) local max. Alternative parameterization!

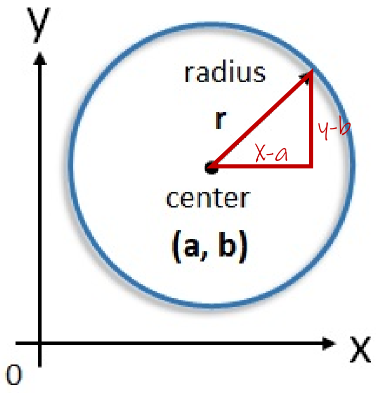
Adapted from K. Grauman

Hough Transform for Circles

Circle parameterization

Leveraging gradient information

Parameterizing a Circle



Let's assume the radius is known.

variables / parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

parameters / variables

$$(x - a)^2 + (y - b)^2 = r^2$$

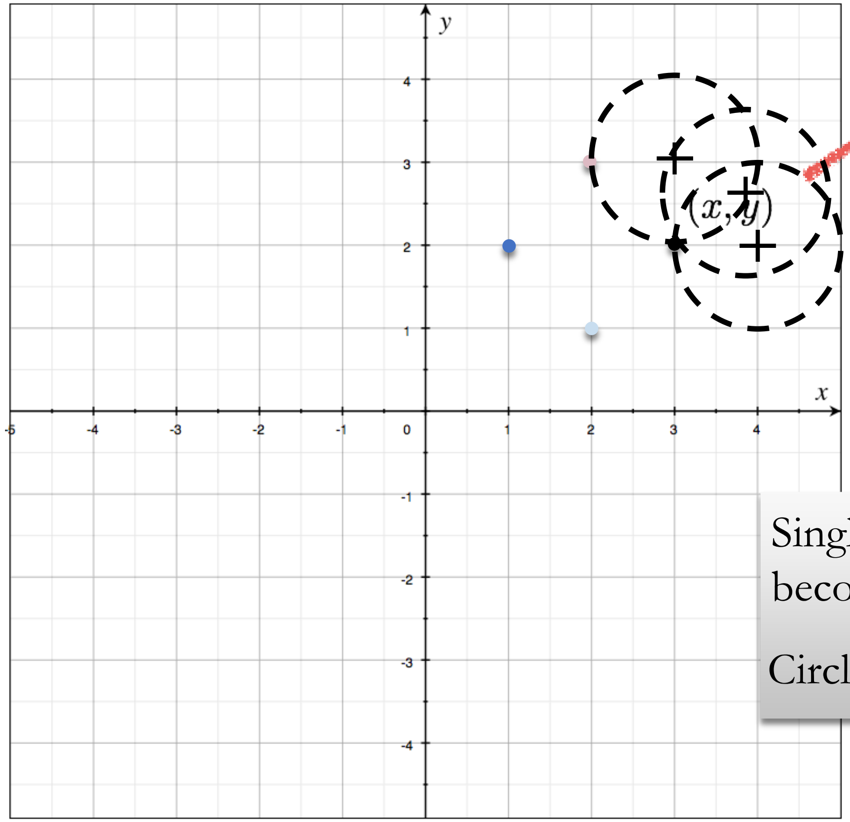
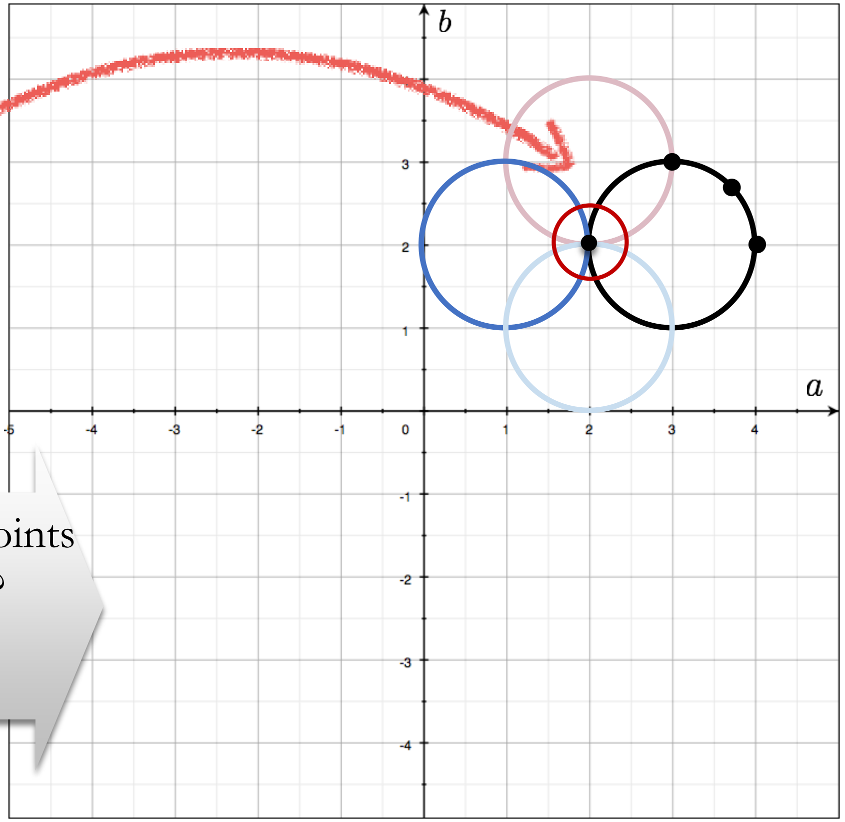


Image space

Single points become?

Circles



Parameter space

Slide Credit: K. Kitani

What if Radius is Unknown?

variables parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

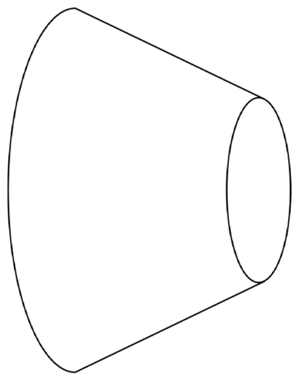
Radius is now a third parameter.

parameters variables

$$(x - a)^2 + (y - b)^2 = r^2$$

When considering the parameter space, this adds a third variable → makes the Hough space 3D.

- Augment accumulator array from $A(a, b)$ to $A(a, b, r)$
- In 2D parameter space, a point in image space corresponds to a circle (of radius r)
- In 3D parameter space, a point projected from image space also gets more complicated → cone



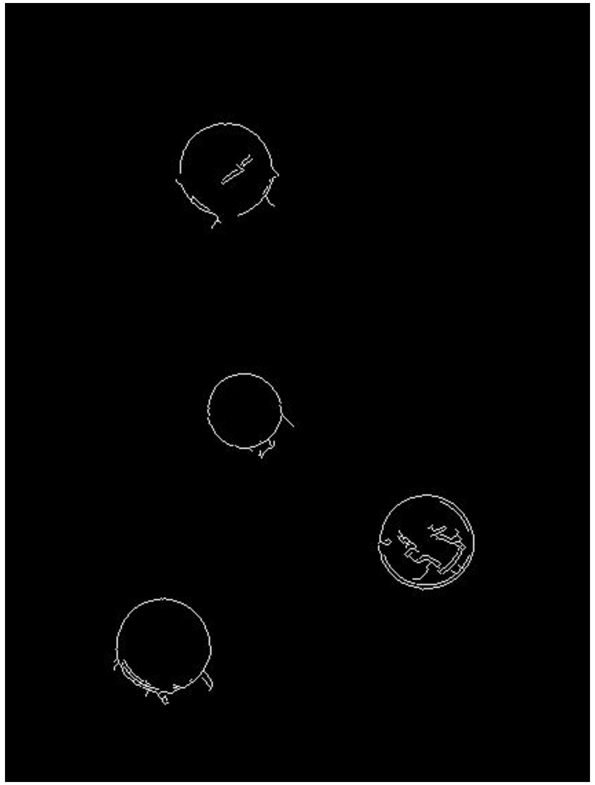
Adapted from K. Kitani

Example 1: Pennies & Quarters

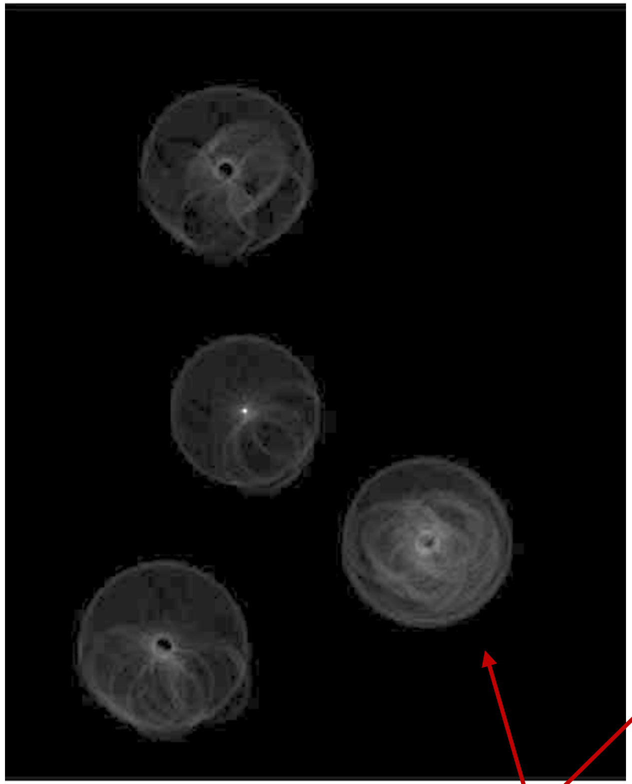
A different Hough accumulator was used for pennies vs. quarters (different radiuses).

Original

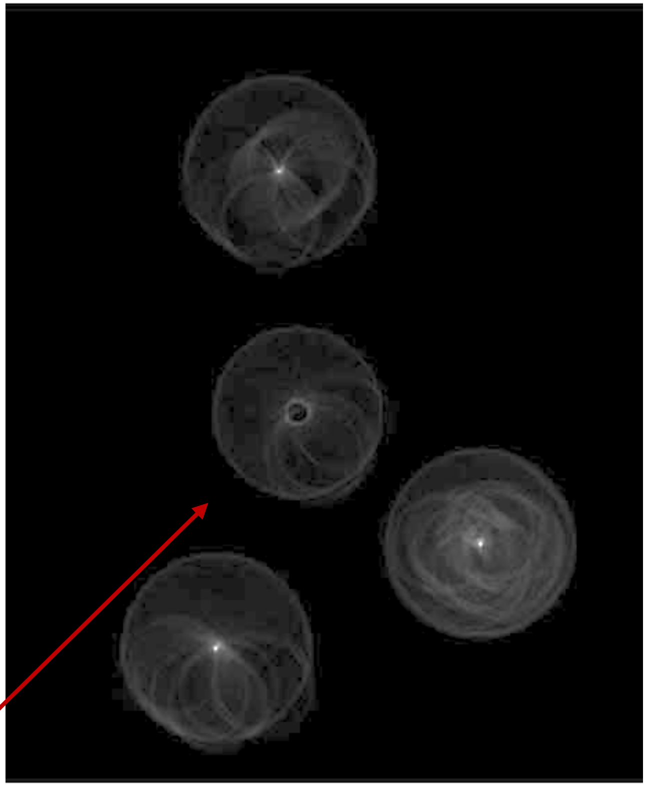
Edges



Votes: Penny



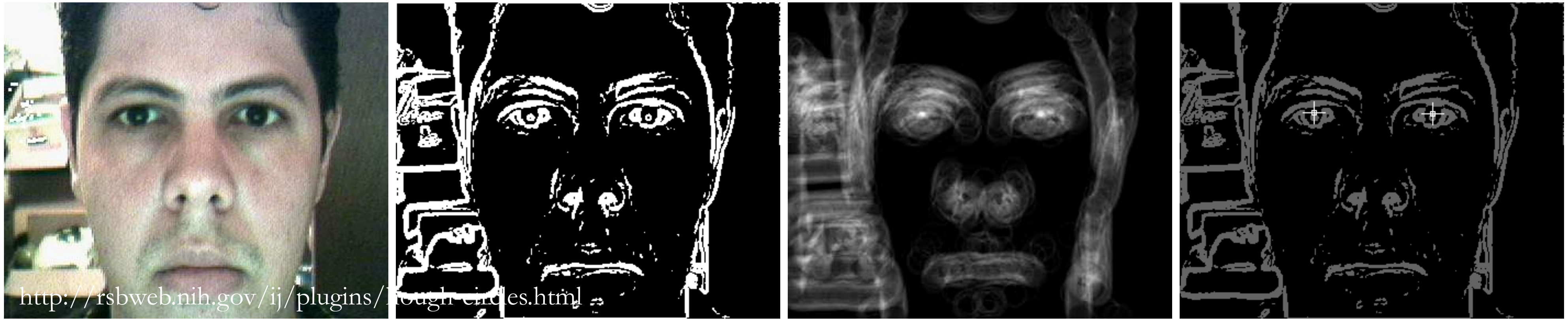
Votes: Quarter



Note how the coins which are too small or large (vs. assumed known radius) do not accumulate to a local maximum.

Coin finding sample images from: Vivek Kwatra

Example 2: Iris Detection



Gradient+threshold   Hough space (fixed radius)   Max detections

Figure 2. Original image

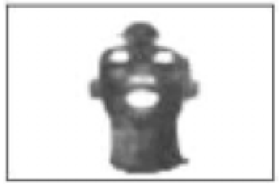


Figure 3. Distance image

Figure 4. Detected face region

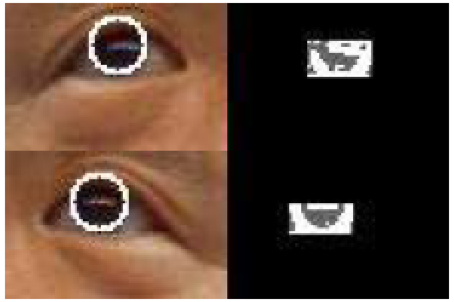


Figure 14. Looking upward

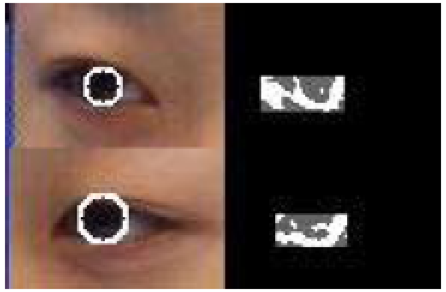


Figure 15. Looking sideways

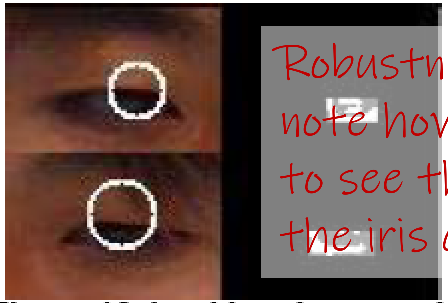


Figure 16. Looking downward

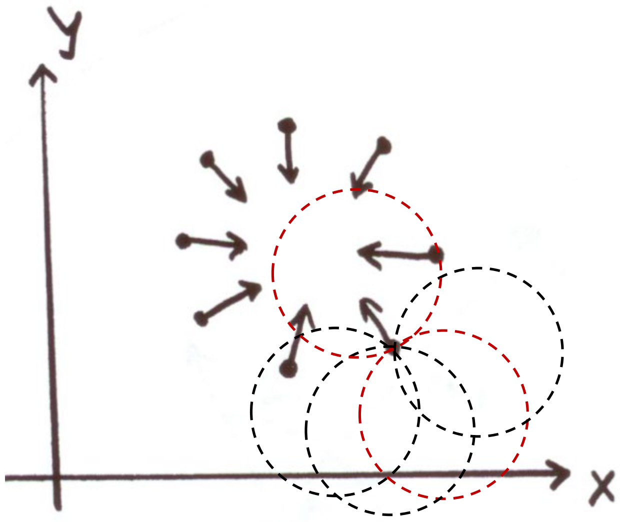
Robustness to occlusion – note how we don't need to see the full eye to find the iris center.

An Iris Detection Method Using the Hough Transform and Its Evaluation for Facial and Eye Movement. Kashima et al. ACCV 2002.

Adapted from K. Grauman

Leveraging Gradient Information

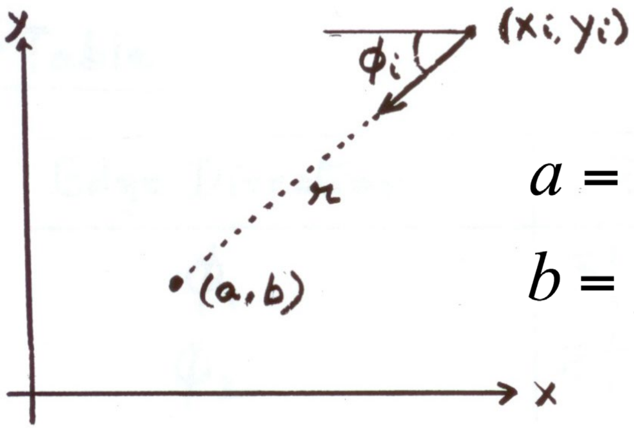
- Gradient information can save a lot of computation



Edge Location (x_i, y_i)

Edge Orientation ϕ_i

If we assume radius is known:



$$a = x - r\cos\phi$$

$$b = y - r\sin\phi$$

In this case, a single point in image space would vote for only two points in the Hough parameter space (instead of a full circle) → more efficient!

Adapted from S. Seitz

Leveraging Gradient Information

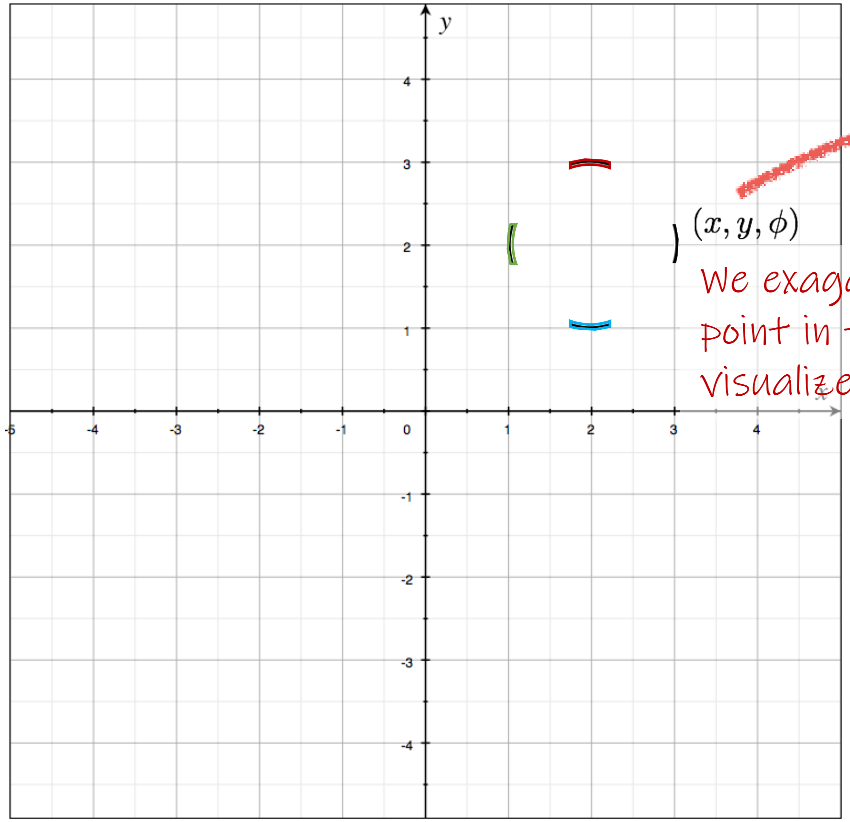
Let's assume the radius is known.

variables parameters

$$(x - a)^2 + (y - b)^2 = r^2$$

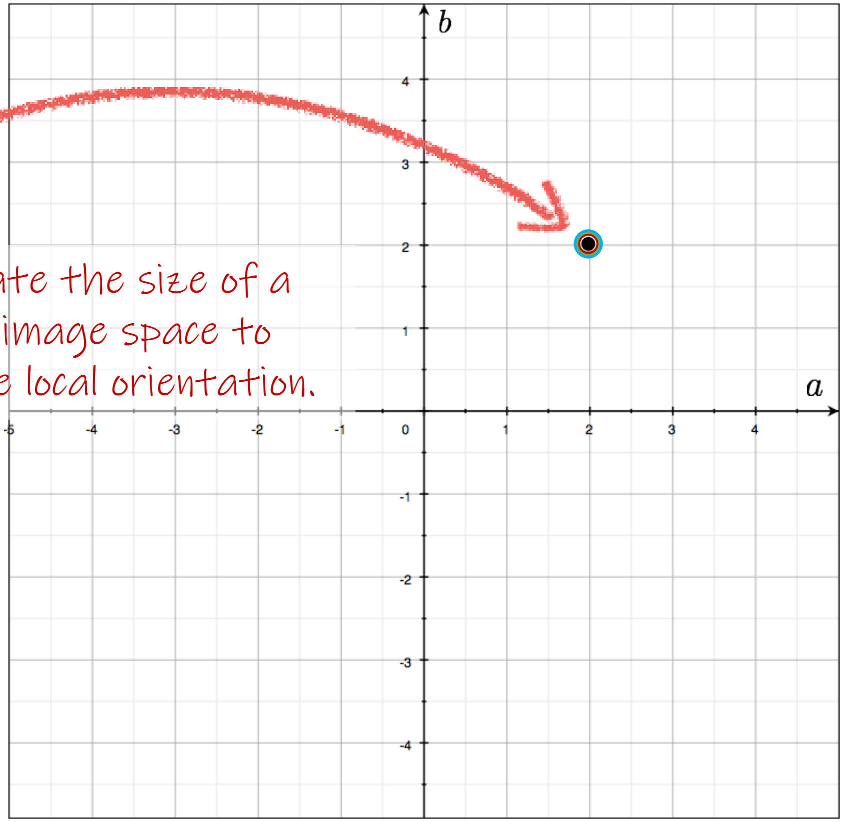
parameters variables

$$(x - a)^2 + (y - b)^2 = r^2$$



We exaggerate the size of a point in the image space to visualize the local orientation.

Image space



Parameter space

Slide Credit: K. Kitani

Generalized Hough Transform

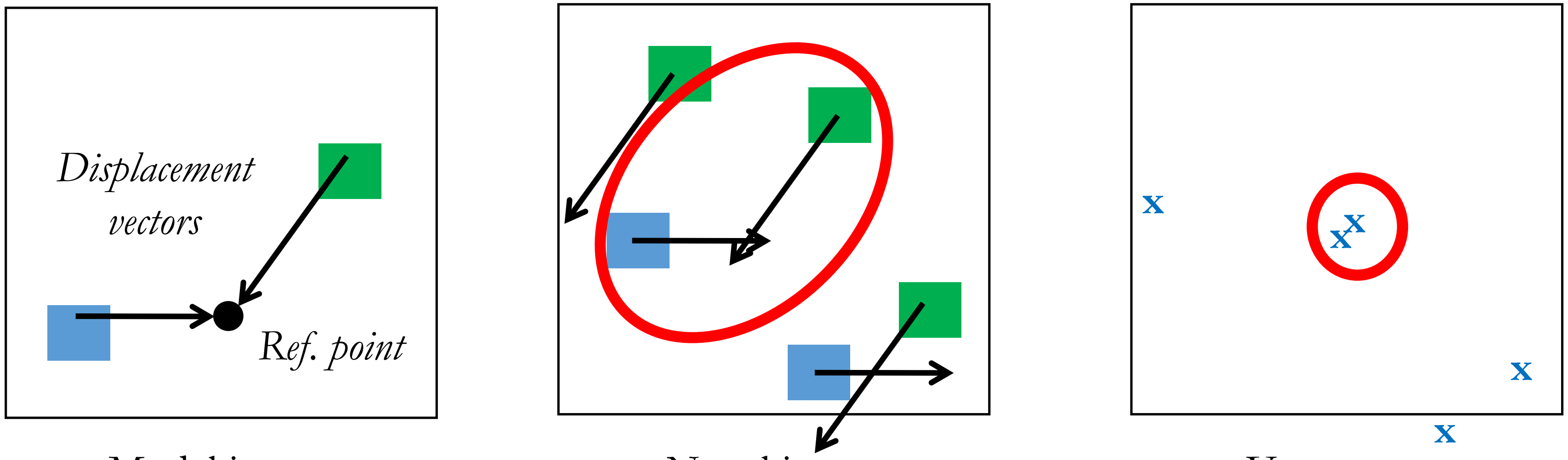
Arbitrary Shapes

Object Models

Generalized Hough Transform

How can we detect arbitrary shapes? What if these shapes that are not so easy to express in closed form equations?

Intuition:



Model image

Novel image

Vote space

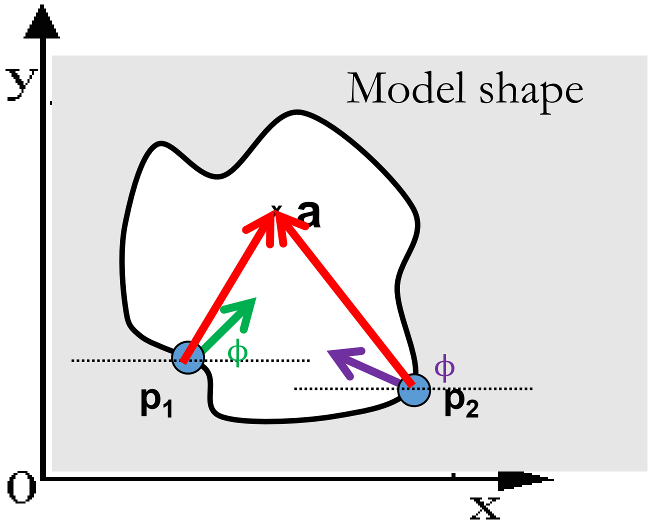
Now suppose the colors encode different gradient directions.

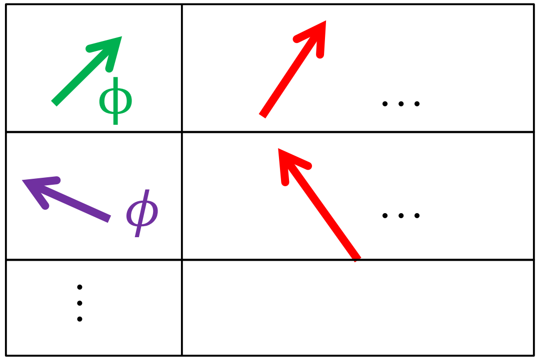
Slide Credit: K. Grauman

Generalized Hough Transform

Offline Modeling

- Define a model shape by its boundary points p_i and reference point a.
- At each boundary point, compute a displacement vector:
- Store vectors in a table, indexed by gradient orientation ϕ





For each row in the table, compute some averaging statistics, e.g. mean / mode(s)

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

Slide Credit: K. Grauman

Generalized Hough Transform

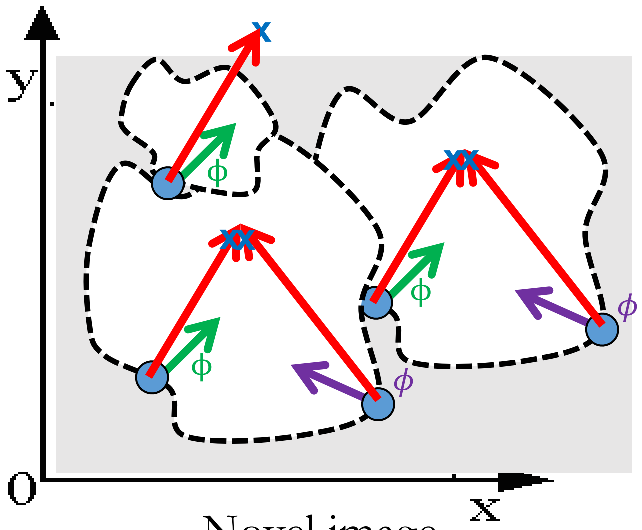
Detection Procedure:

For each edge point:

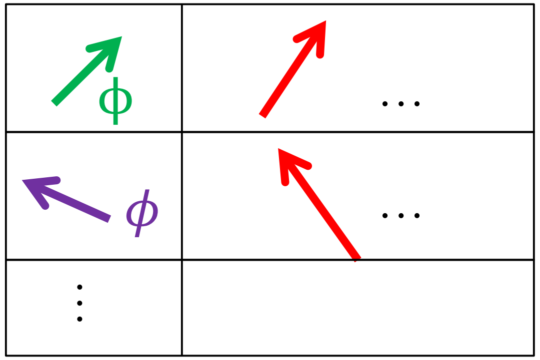
- Use its gradient orientation ϕ to index into stored table
- Use retrieved **r** vectors to vote for reference point

This procedure assumes that translation is the only transformation, i.e., orientation and scale of the arbitrary shape are fixed.

How can we account for orientation and scale?

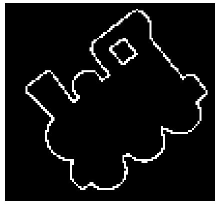


Novel image

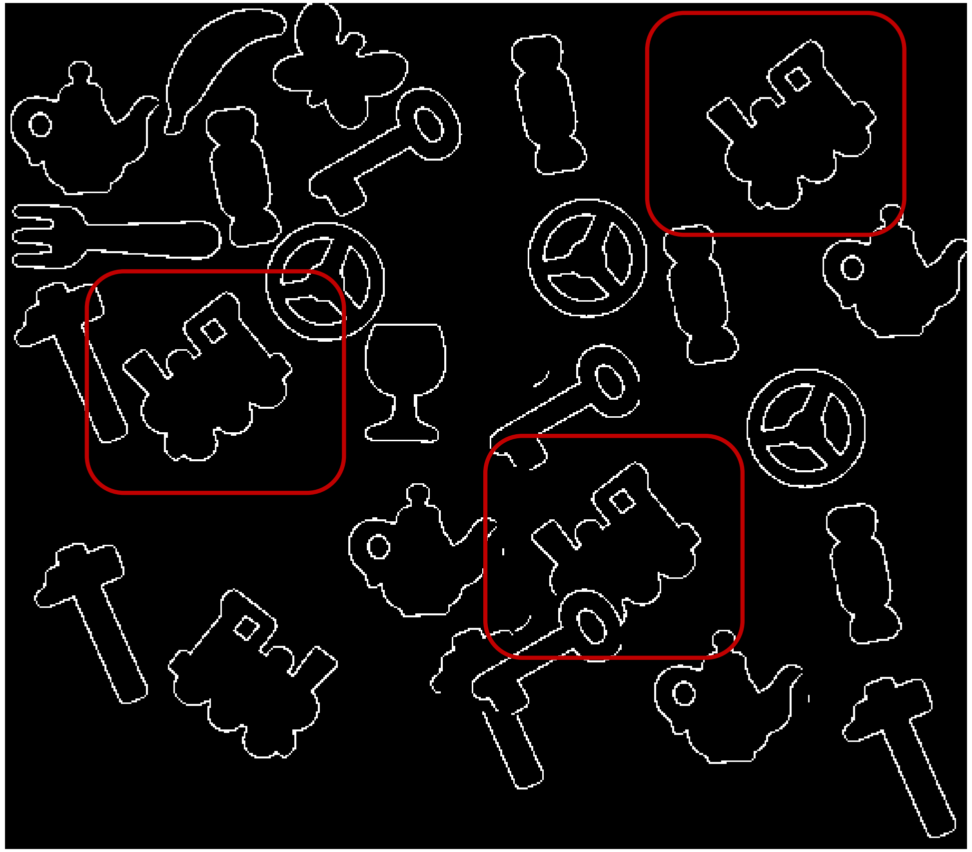


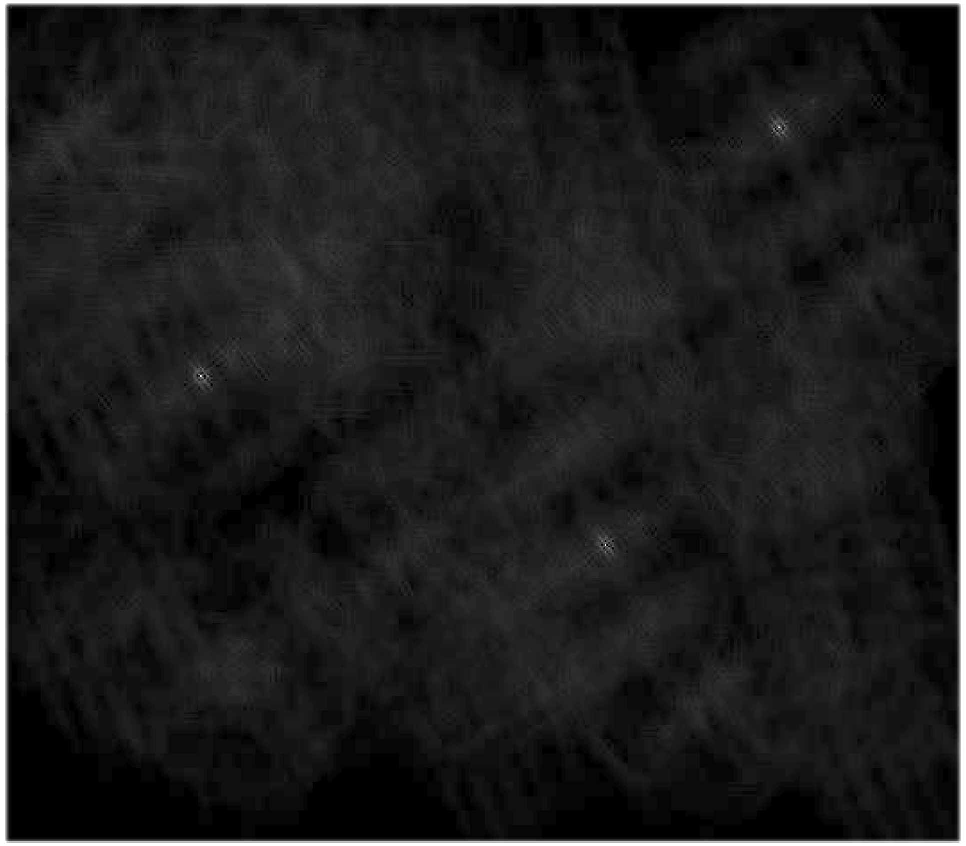
Slide Credit: K. Grauman

Generalized Hough Transform



reference template





https://mbcoder.com/generalized-hough-transform/

Hough Voting for Object Detection & More

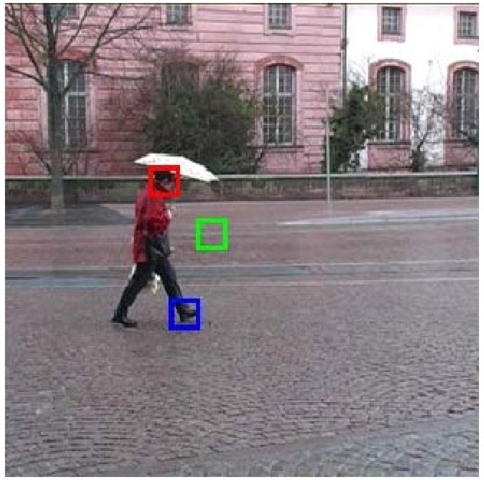
Instead of indexing displacements by gradient orientation, index by matched local patterns (visual codewords).

test image

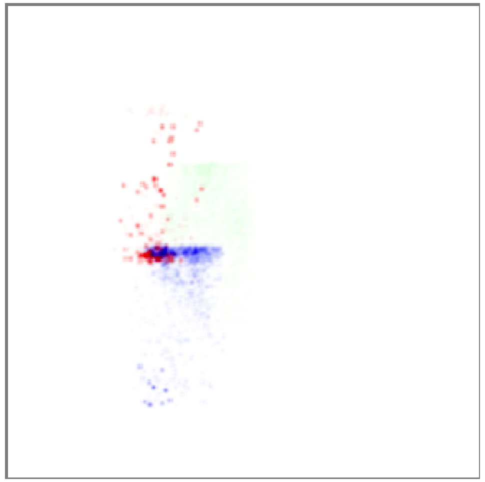
B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](), 2004

Slide credit: L. Lazebnik

Object Detection, Tracking, Actions & More

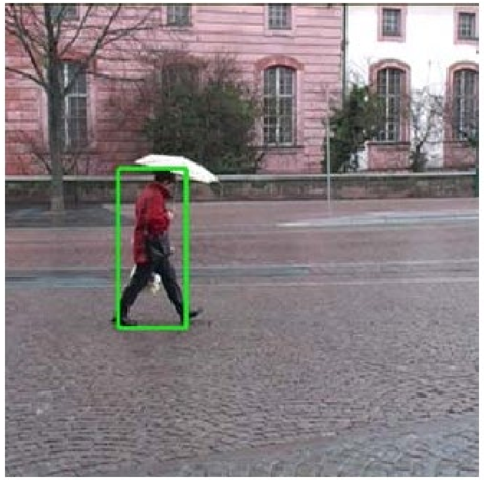


(a) – Original image with three sample patches emphasized

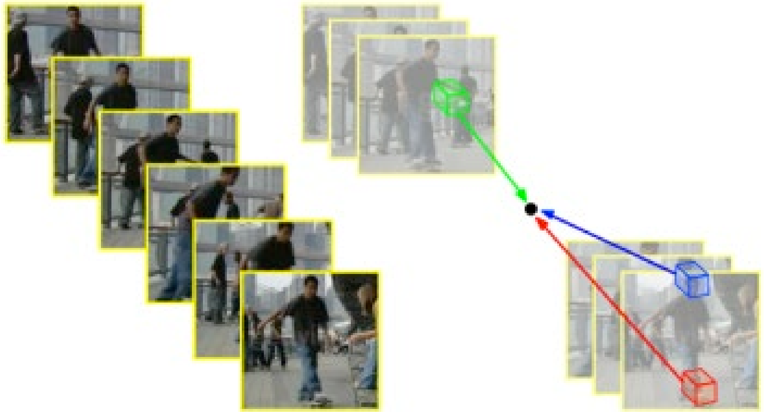


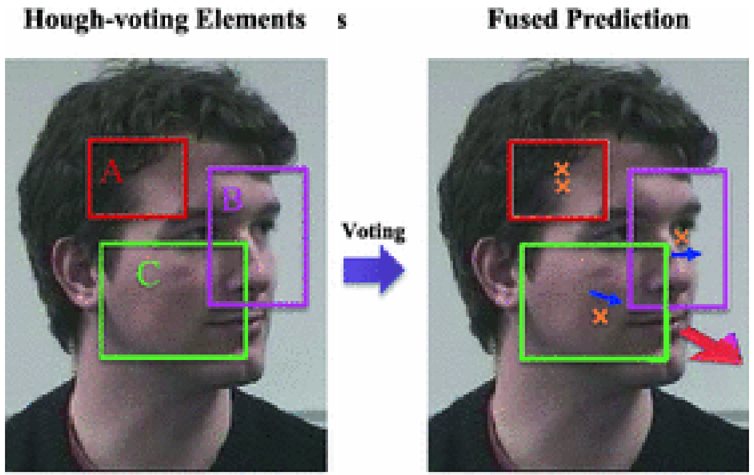
(b) – Votes assigned to these patches by the Hough forest

(c) – Hough image aggregating votes from all patches



(d) – The detection hypothesis corresponding to the peak in *(c)*





J. Gall, A. Yao, N. Razavi, L. Van Gool, & V. Lempitsky. Hough forests for object detection, tracking, and action recognition. PAMI 2011.

Model Fitting via Voting

- Infeasible to check all combos of features by fitting a model to each possible subset.
- **Voting:** general technique to let the features vote for all compatible models
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Features from noise, clutter, background, etc. will also cast votes
  - Typically their votes should be inconsistent with the majority of "good" features.

Adapted from K. Grauman

Voting: Practical Tips

- Minimize irrelevant tokens first, e.g. convert to edge image
- Choose a good grid / discretization

Too fine $\longleftarrow$? $\longrightarrow$ Too coarse

- Soft voting for neighbors (smoothing effect in accumulator array)
  - $(\theta, \rho) \rightarrow 0.25 * (\theta, \rho - 1), 0.5 * (\theta, \rho), 0.25 * (\theta, \rho + 1)$
- Limit voting extent from each token
  - Use direction of edge to reduce amount of cast votes
- Keep tags on votes to read back points contributing to local maxima

Adapted from K. Grauman

Hough Transform: Pros & Cons

Pros

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

Cons

- Search time complexity increases exponentially with the # 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: K. Grauman

Summary

- Mathematics of lines via different forms of parameterizations
  - Intercept form, normal form
- Hough voting for straight lines & circles
  - Iterate through each image edge point, cast a vote in the Hough accumulator for the corresponding set of line parameters
- Generalized Hough Transform
  - Arbitrary shapes can be defined e.g. by a set of oriented gradients and their corresponding displacements to some reference point
  - Generalize to other forms of local image evidence (visual codewords/patches) to vote for object detection, action recognition, etc.