上海交通大学

计算机视觉

教师：赵旭

班级：AI4701

2024 春

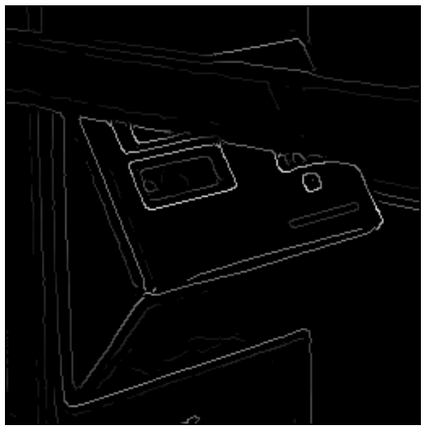
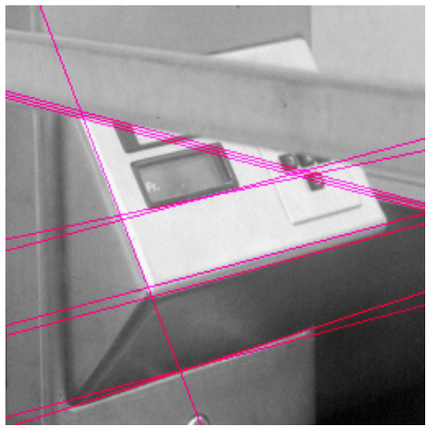
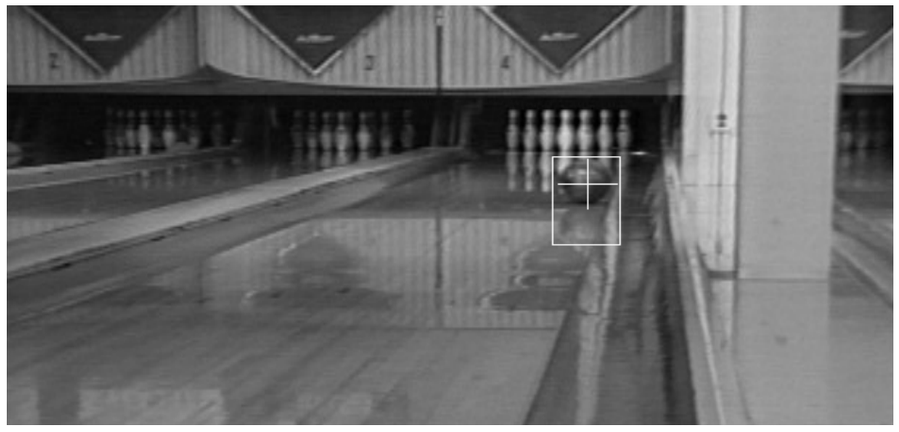
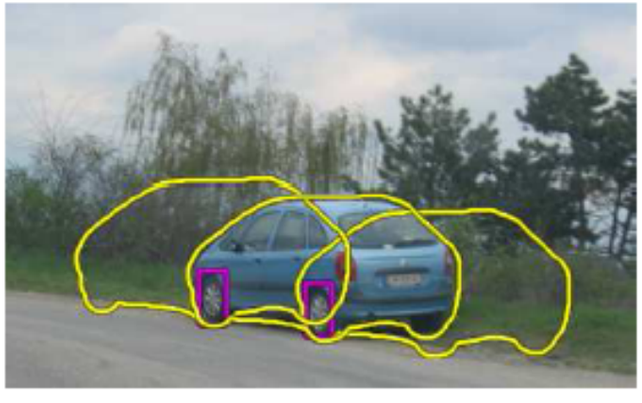
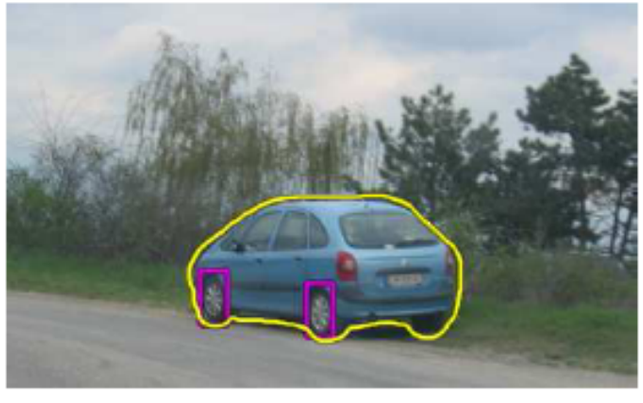
15. 模型拟合

Contents

- Fitting using Hough Transform
- Fitting using Probabilistic Models-EM

Fitting

Fitting: want to associate a model with observed features.
The model could be a line, a circle, or an arbitrary shape.

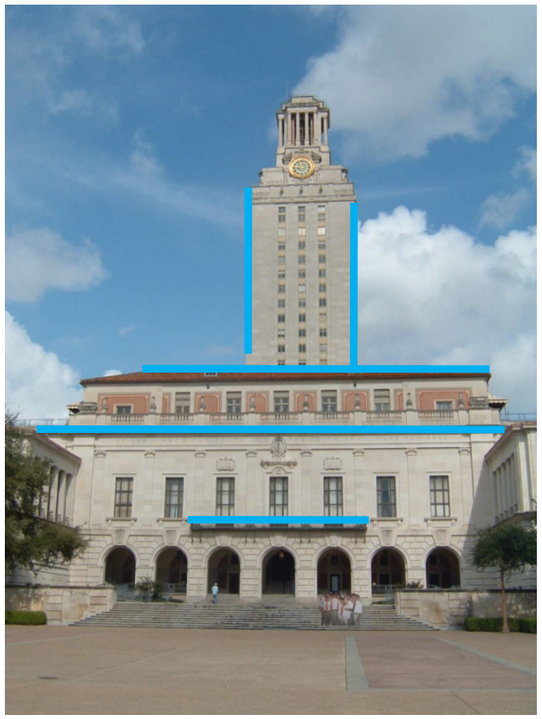

Slide credit: Kristen Gruman

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

Line fitting example

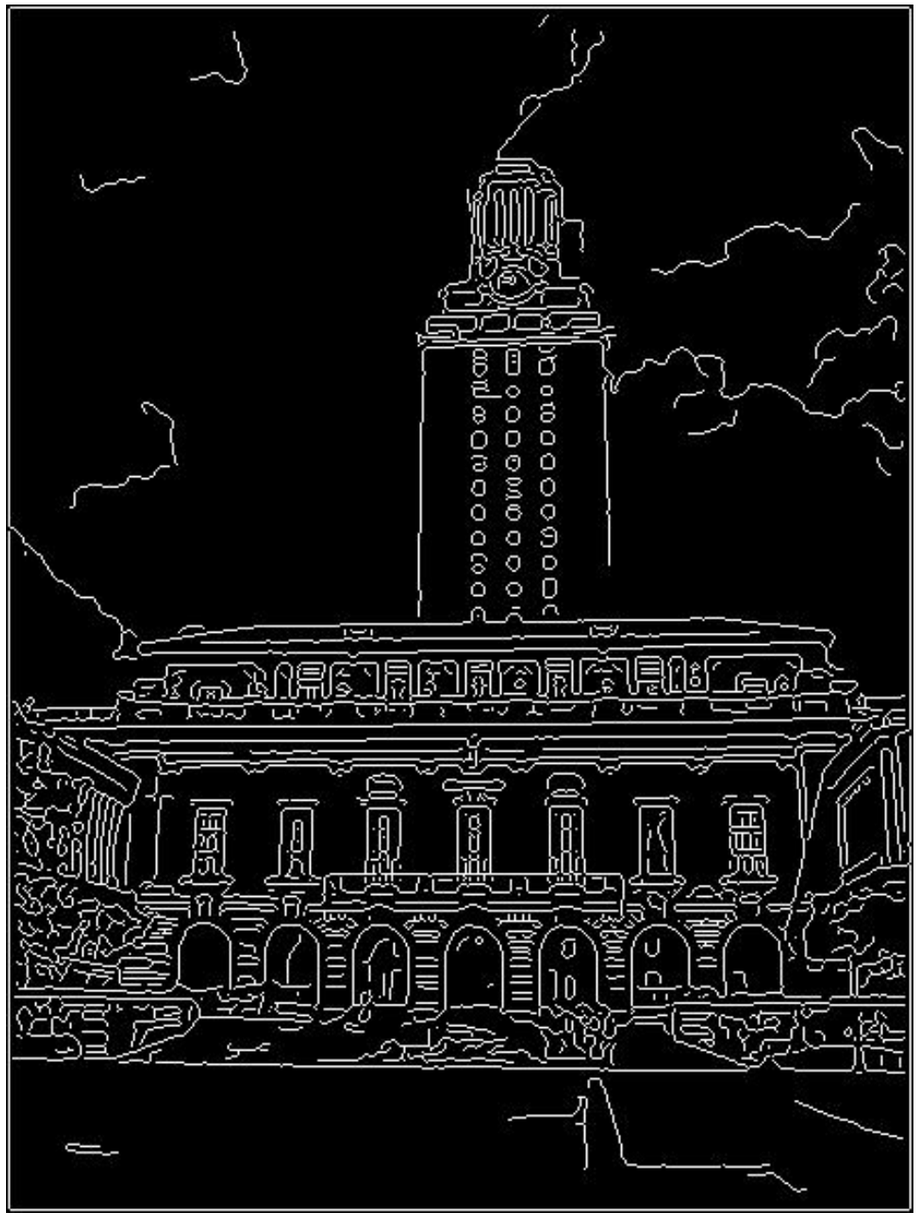


- Many objects characterized by presence of straight lines
- But why aren’t we done just by running edge detection?

Slide credit: Kristen Gruman

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?



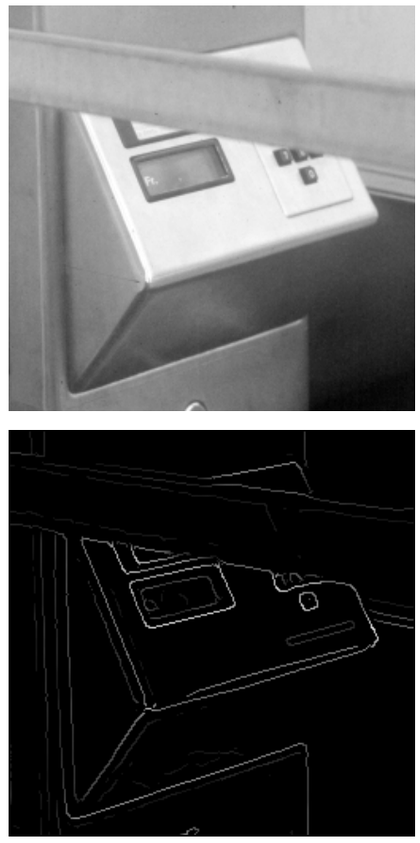
Slide credit: Kristen Gruman

Voting

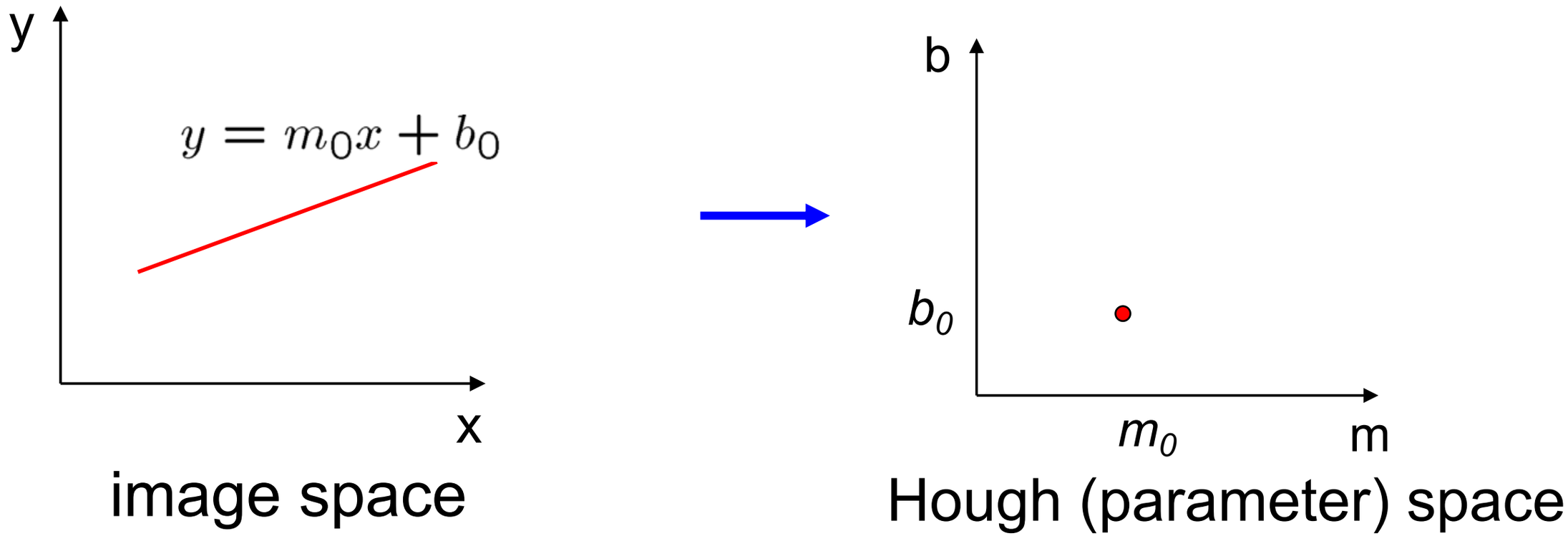
- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features *vote for all models that are compatible with it*.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters 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:
  - Record vote for each possible line on which each edge point lies.
  - Look for lines that get many votes.



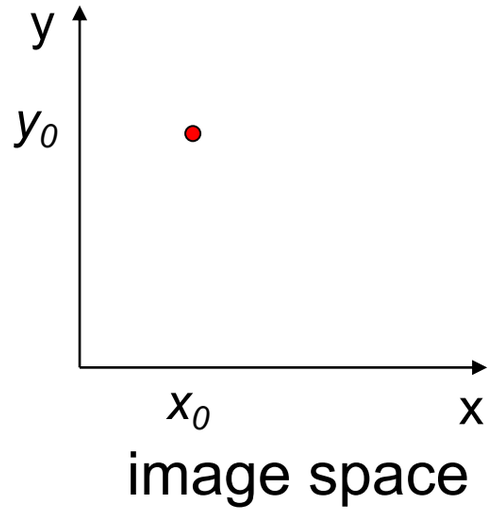
Finding lines in an image: Hough space

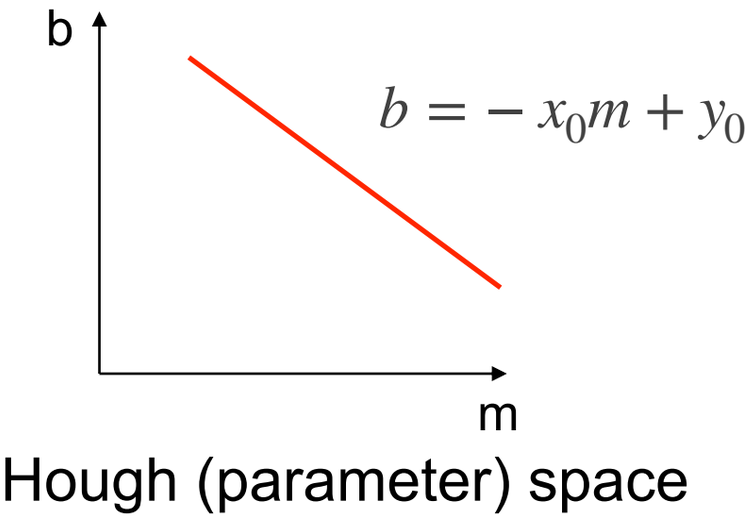


- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space
  - To go from image space to Hough space:
    - given a set of points (x,y), find all (m,b) such that y = mx + b

Slide credit: Steve Seitz

Finding lines in an image: Hough space

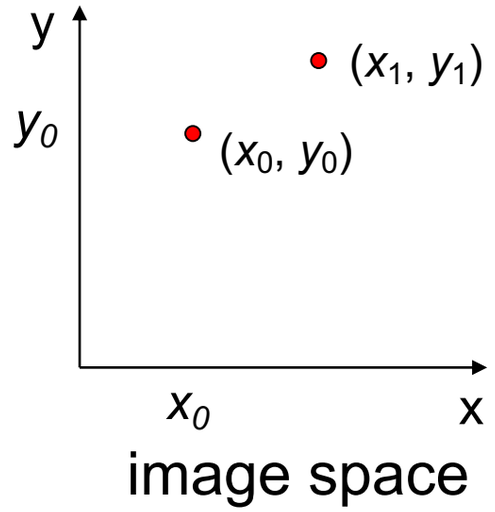


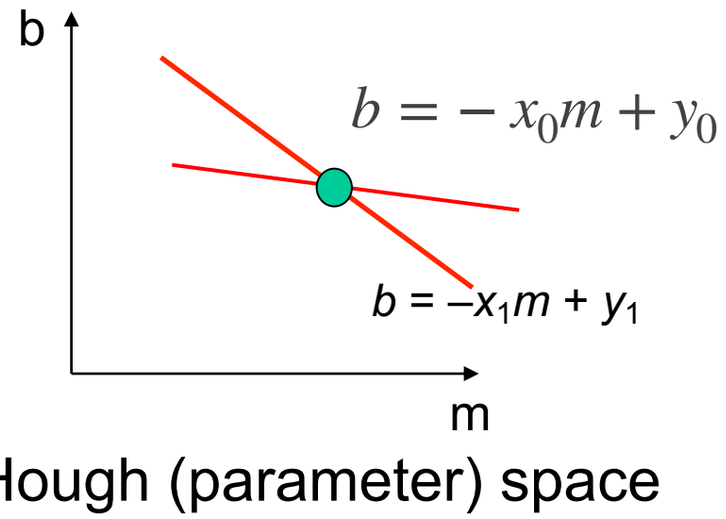


- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space
  - To go from image space to Hough space:
    - given a set of points (x,y), find all (m,b) such that y = mx + b
  - What does a point (x_0, y_0) in the image space map to?
    - Answer: the solutions of b = $-x_0$m + y_0
    - this is a line in Hough space

Slide credit: Steve Seitz

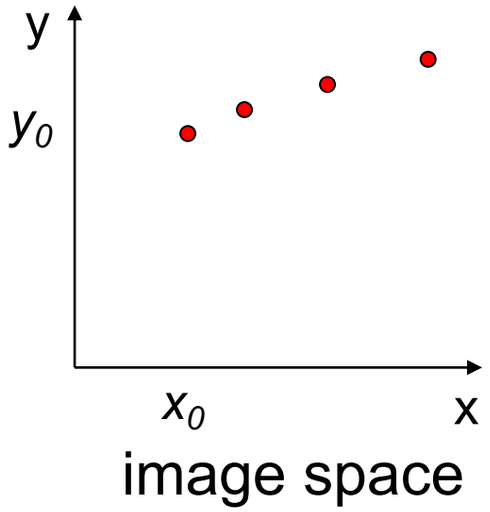
Finding lines in an image: Hough space

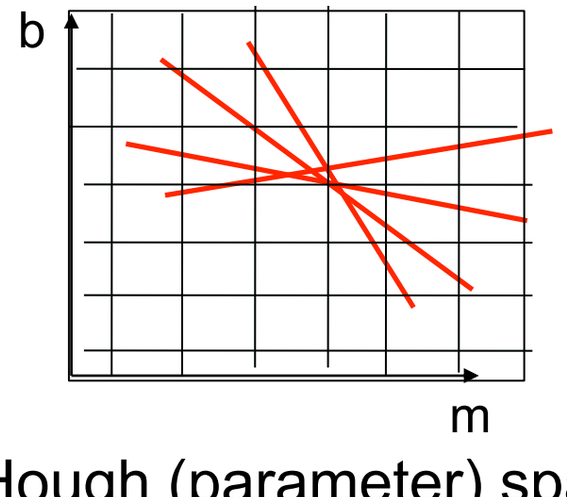




- What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$

Finding lines in an image: Hough space

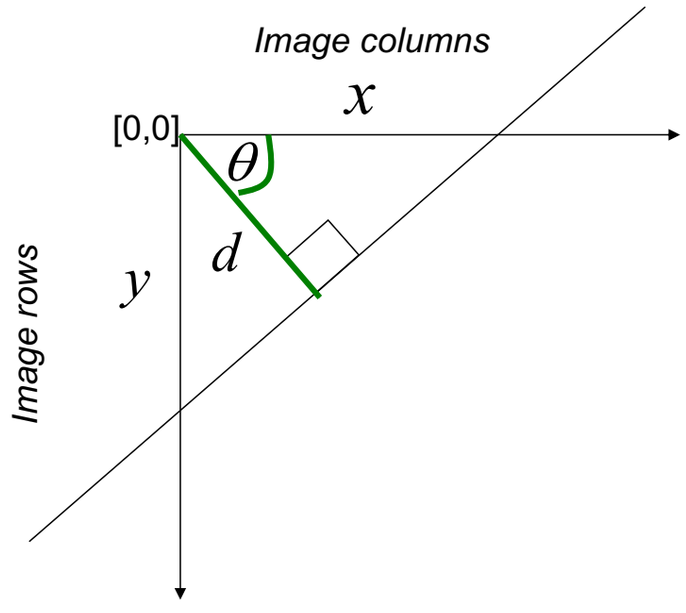




- How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?
  - Let each edge point in image space *vote* for a set of possible parameters in Hough space
  - Accumulate votes in discrete set of bins*; parameters with the most votes indicate line in image space

Polar representation for lines

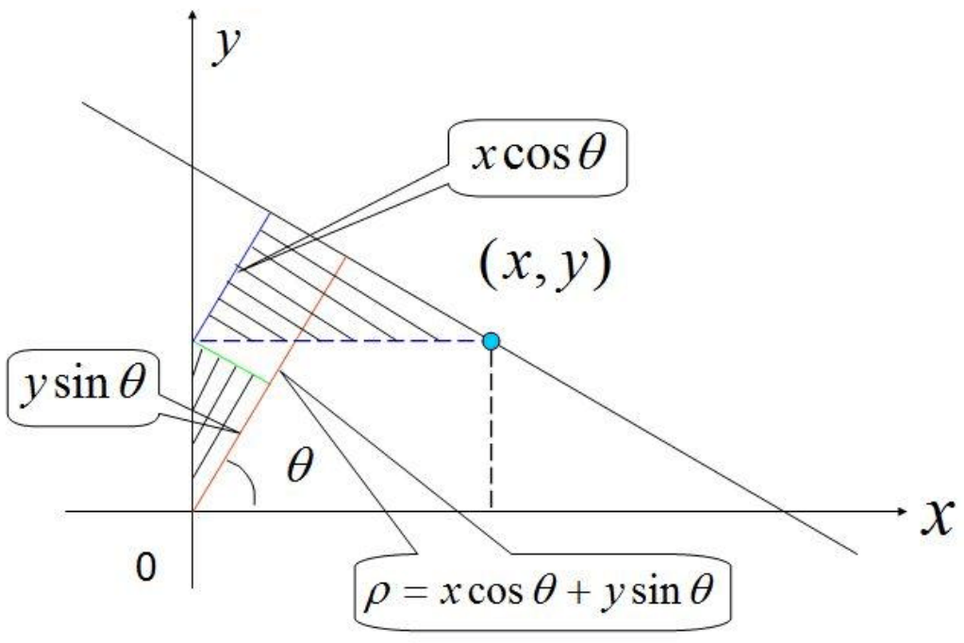
- Issues with usual (*m*,*b*) parameter space: can take on infinite values, undefined for vertical lines.



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

Hough transform algorithm

- Using the polar parameterization:

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

- Basic Hough transform algorithm

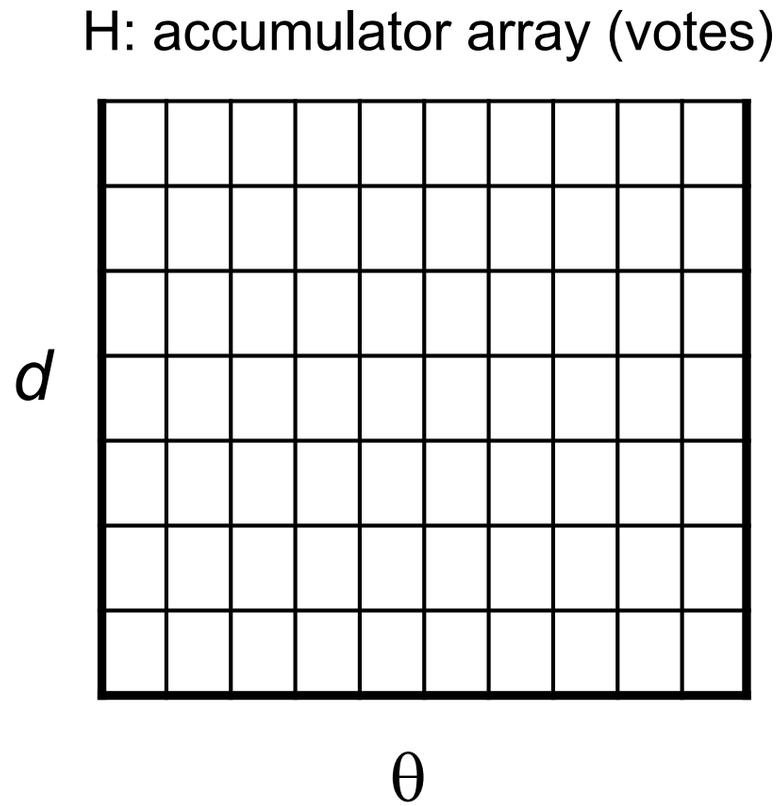
1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

   for θ = [θ_{min} to θ_{max}] // some quantization

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

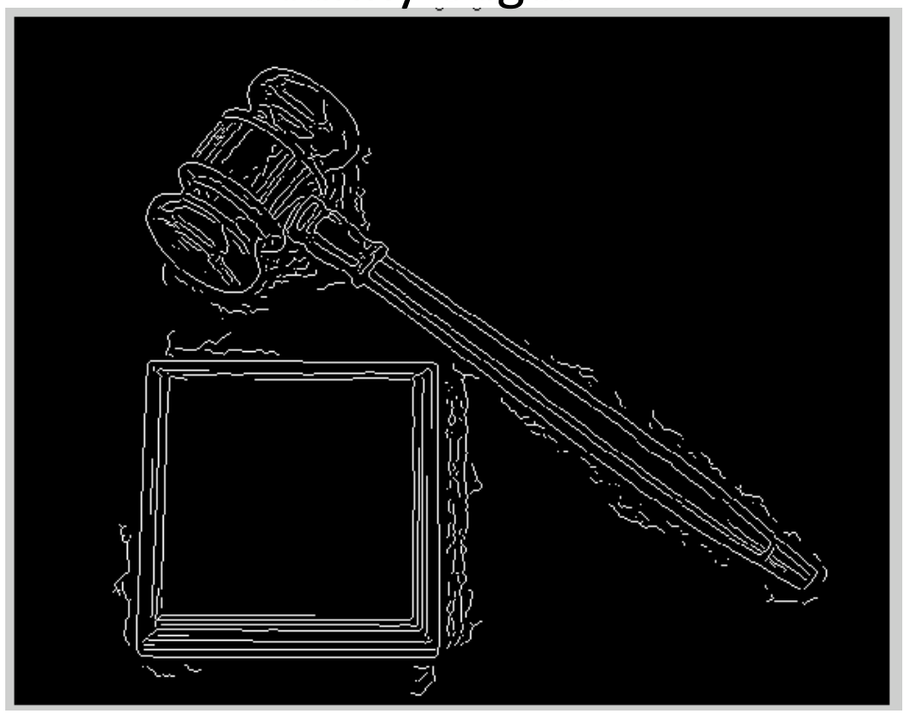
   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by

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

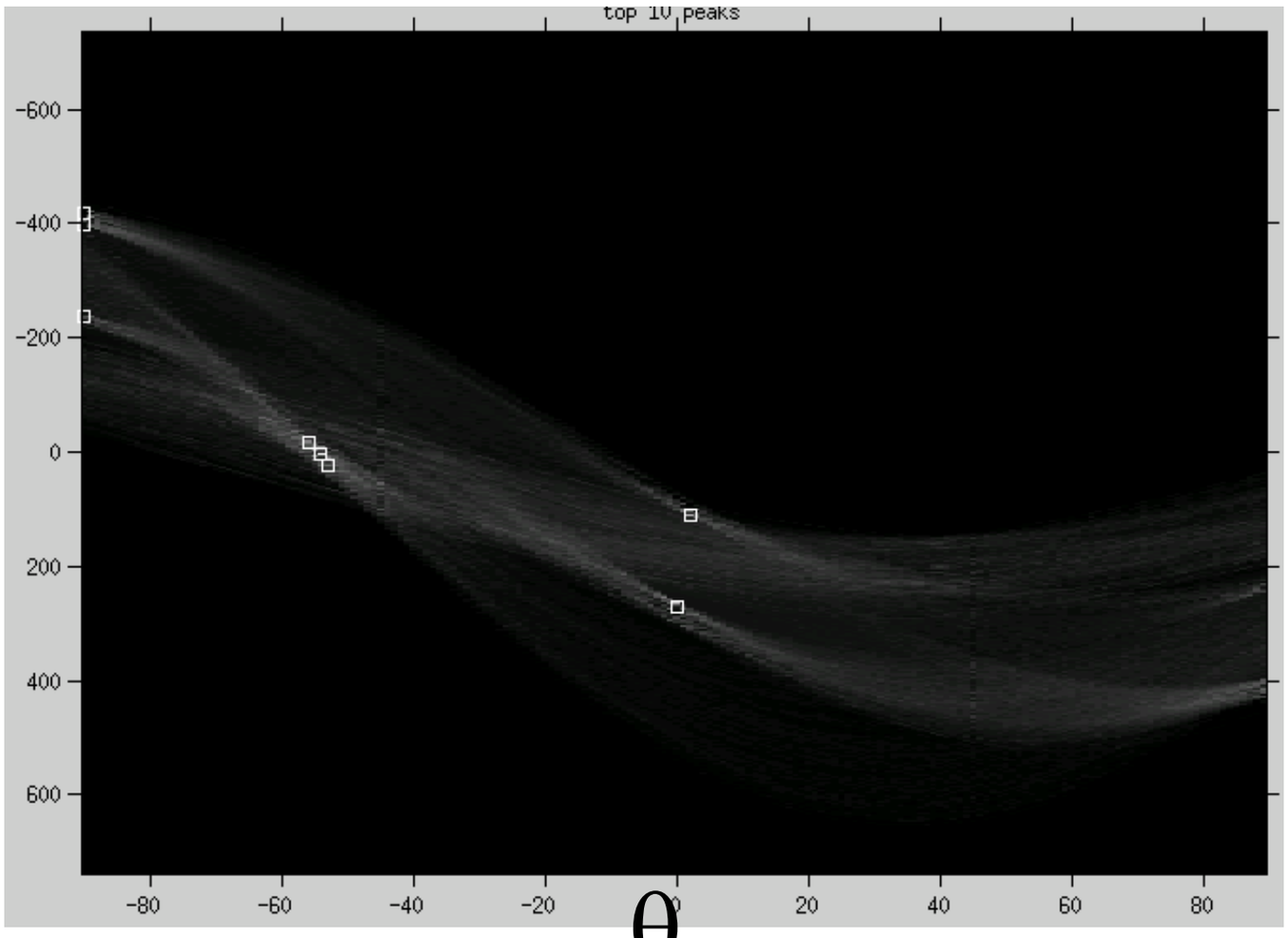


Original image

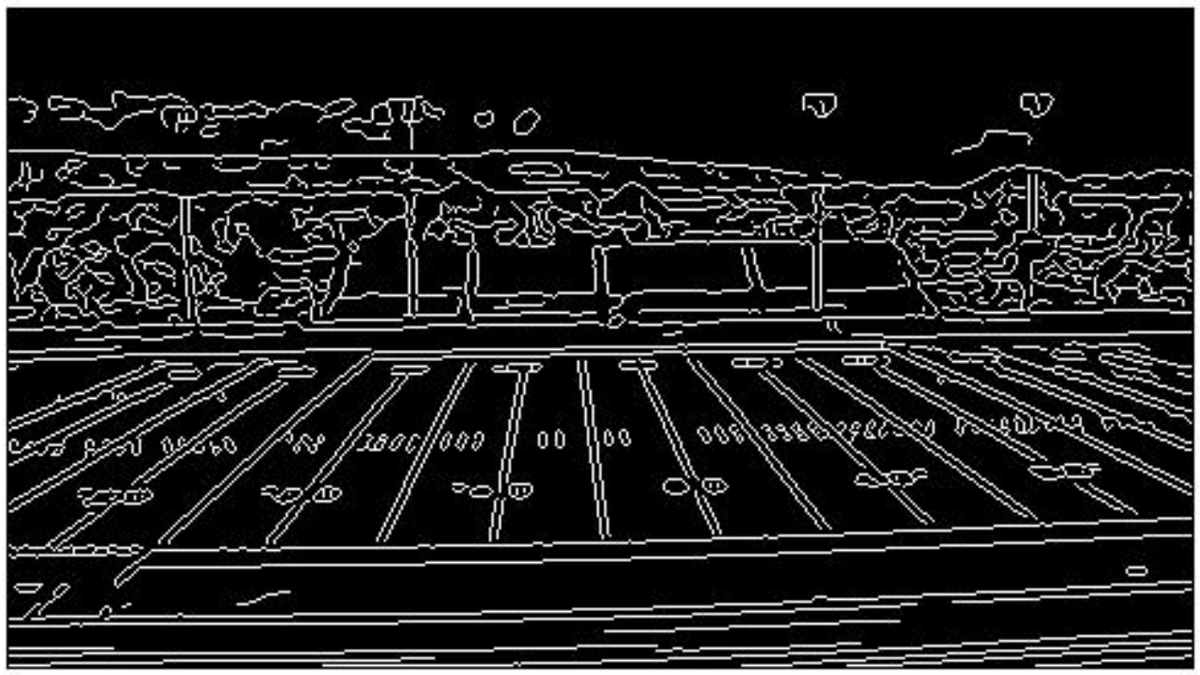
Canny edges

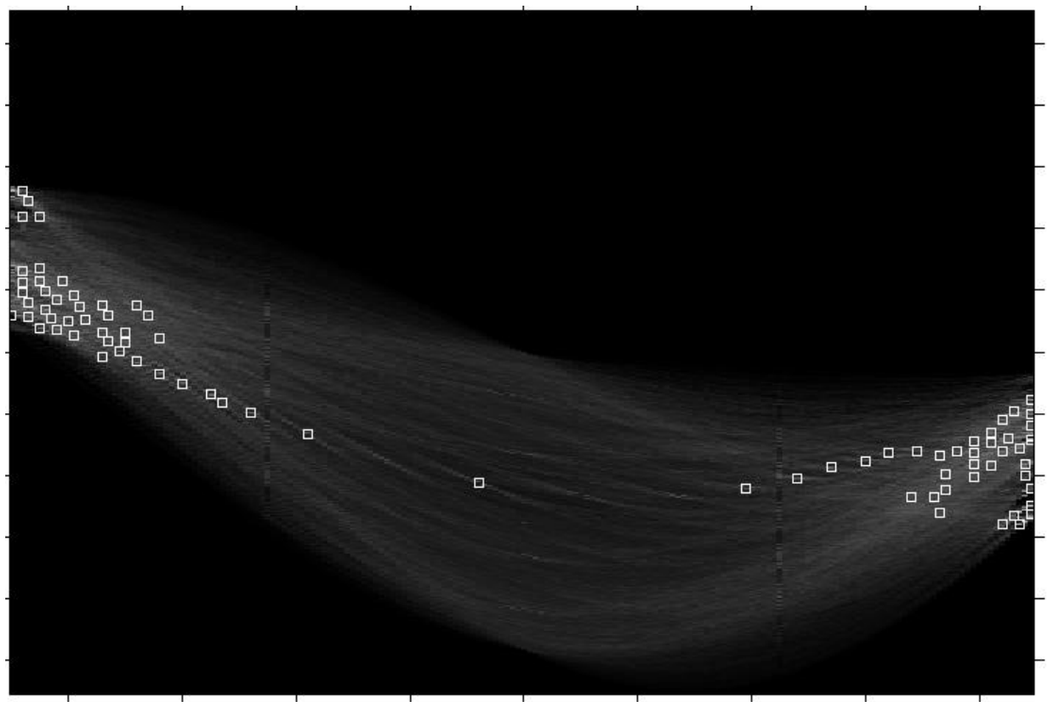


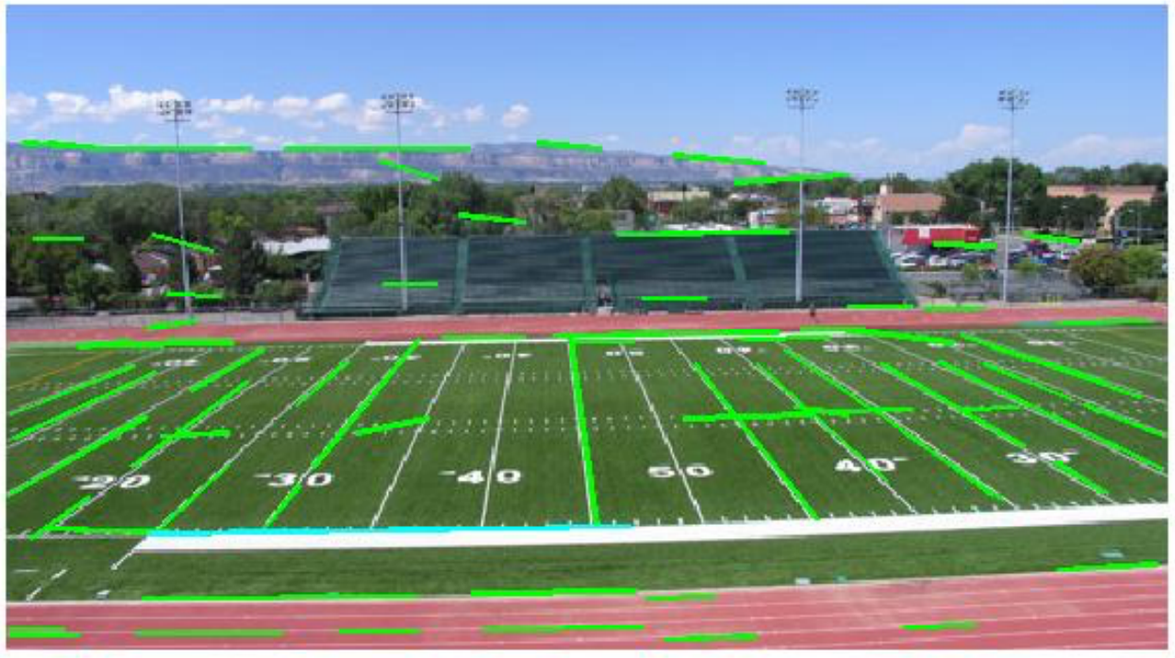
Decode the vote space.



Slide credit: Kristen Gruman







Showing longest segments found

Slide credit: Kristen Gruman

Impact of noise on Hough

- What difficulty does this present for an implementation?

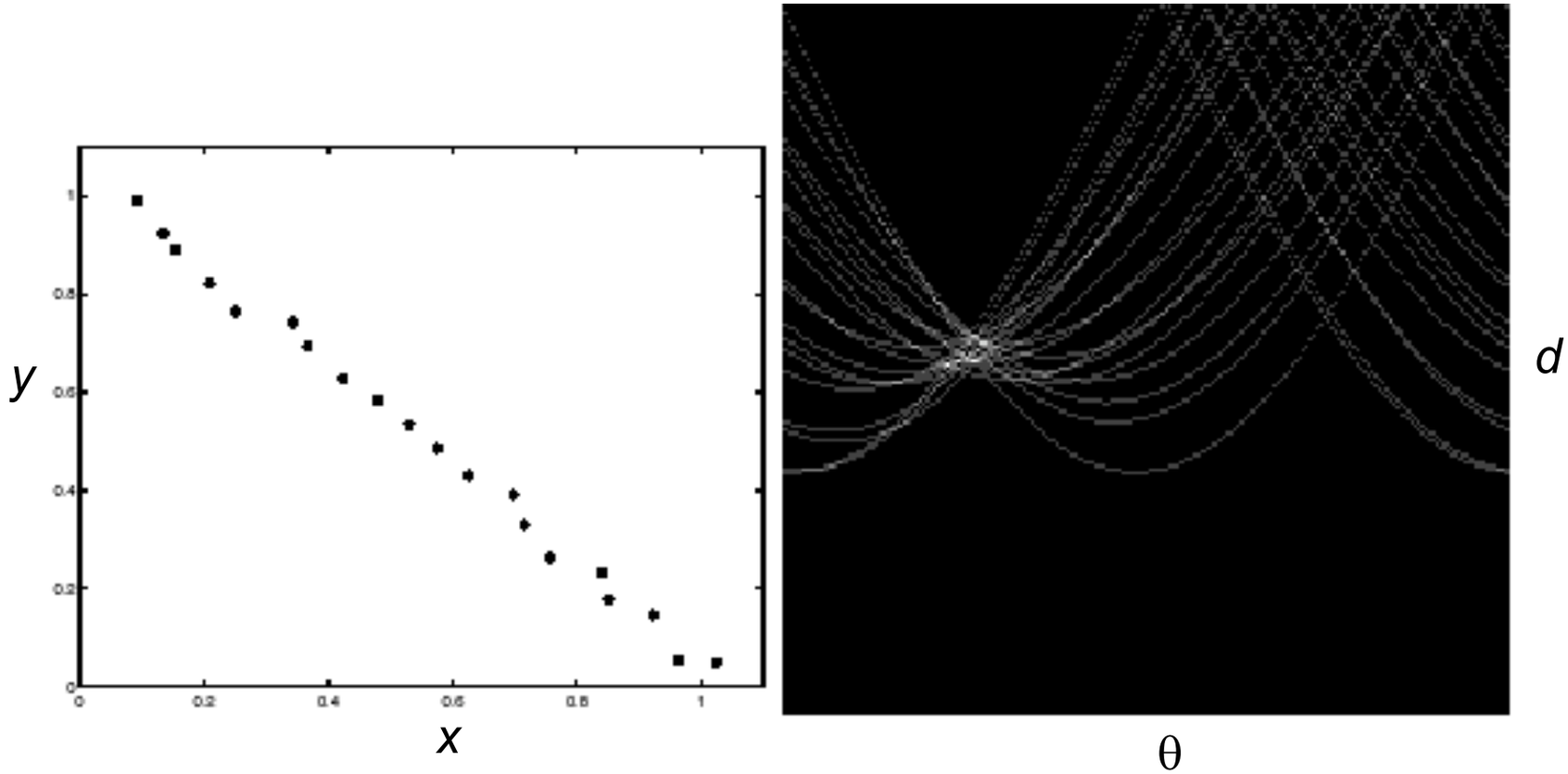


Image space
edge coordinates

Votes

Impact of noise on Hough

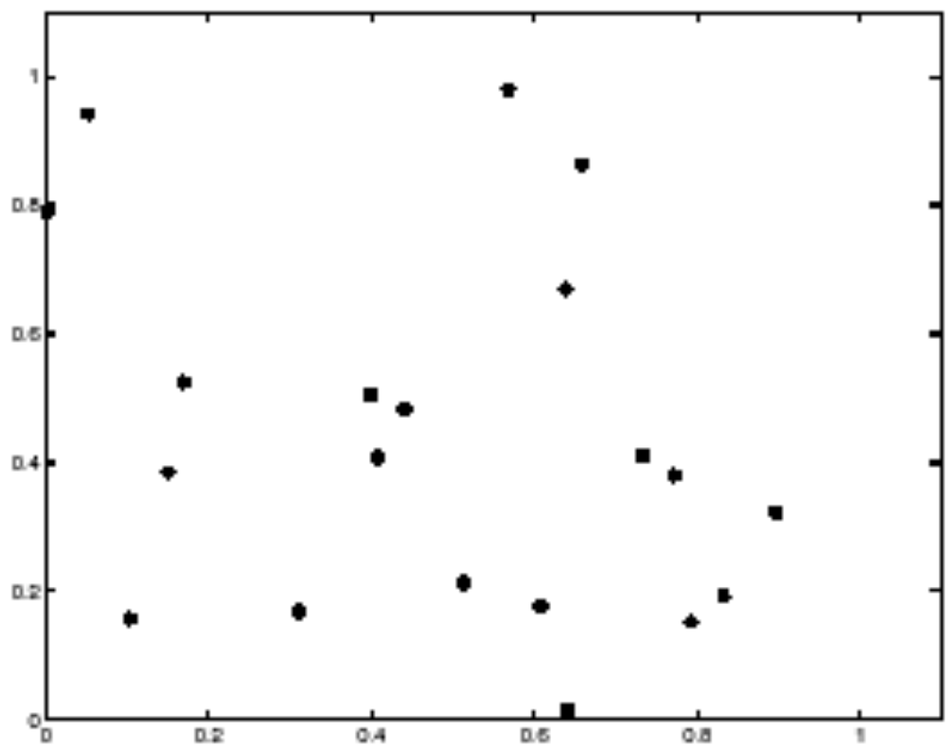
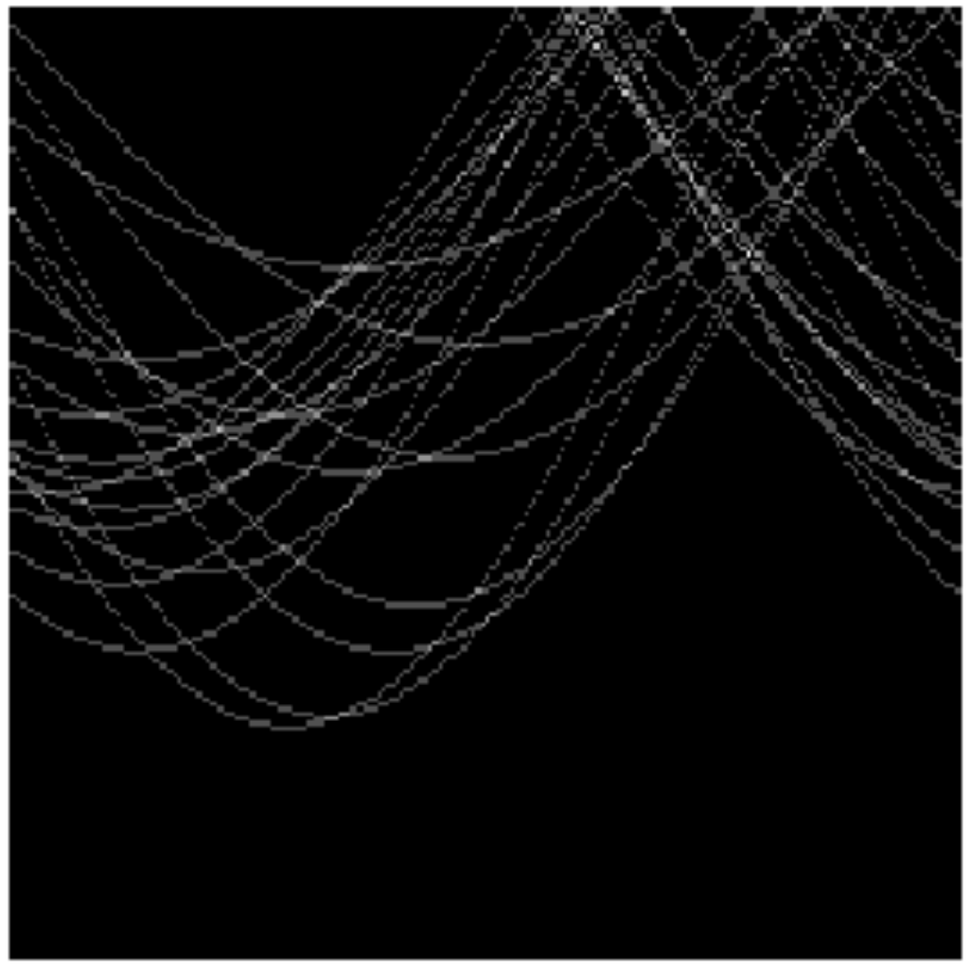


Image space edge coordinates



Votes

Extensions

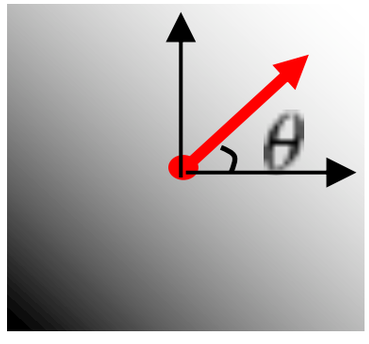
- Extension 1: Use the image gradient

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

     θ = gradient at (x,y)

     H[d, θ] += 1
  3. same
  4. same

  (Reduces degrees of freedom)



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

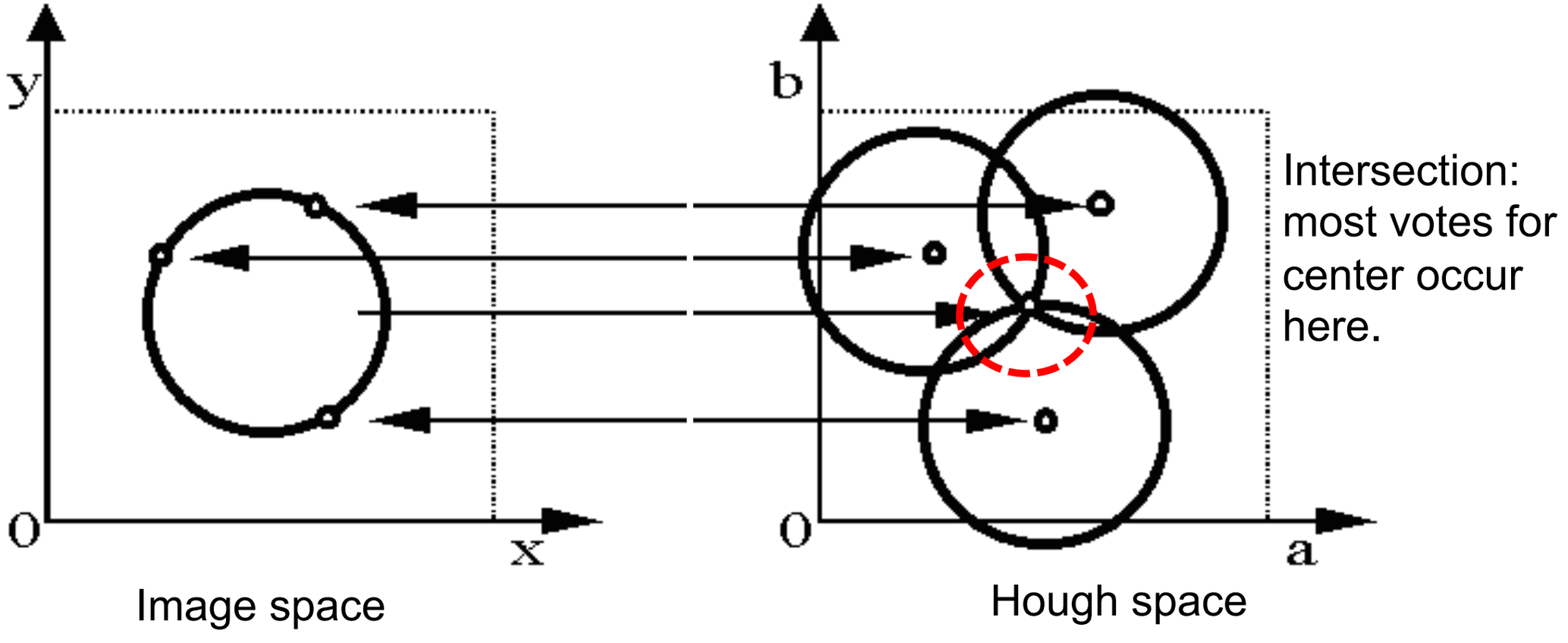
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Extension 2: give more votes for stronger edges
- 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

Slide credit: 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** gradients direction



Hough transform for circles

- Circle: center (a,b) and radius r $(x_i - a)^2 + (y_i - b)^2 = r^2$
- For **unknown** radius r, **unknown** gradients direction

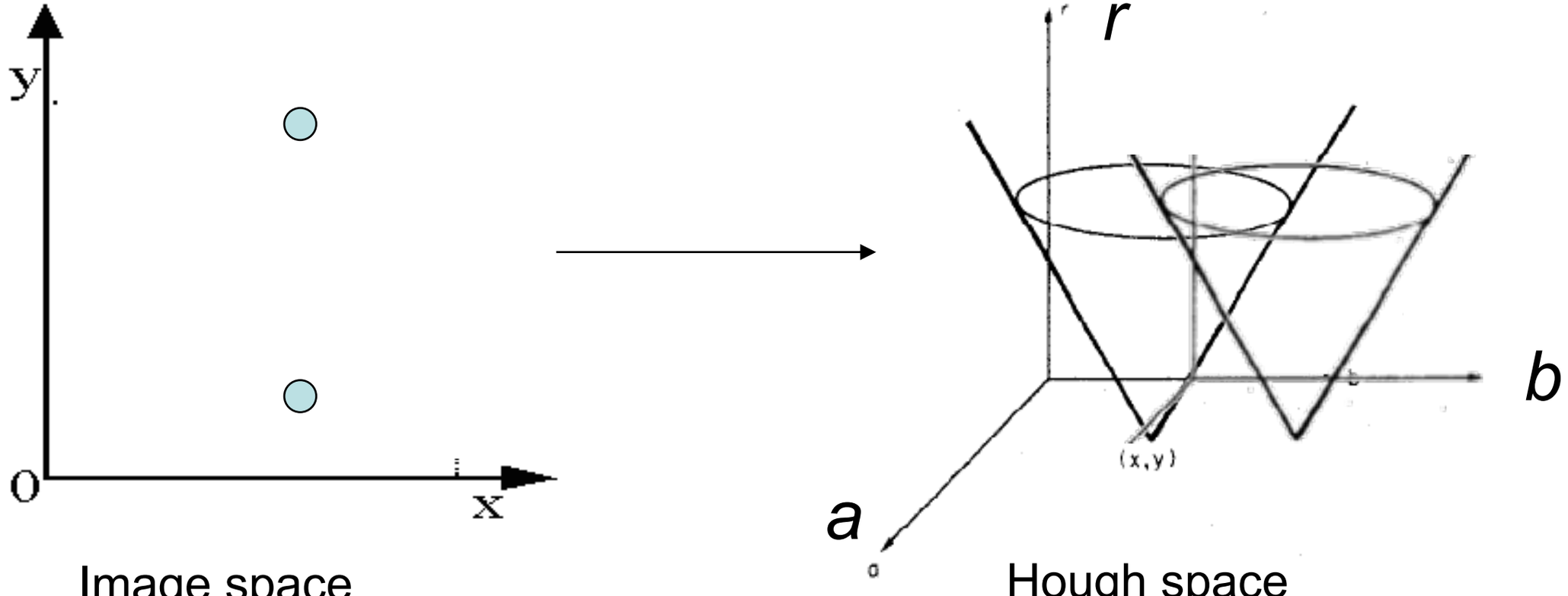


Image space

Hough space

Slide credit: Kristen Gruman

Hough transform for circles

- Circle: center (a,b) and radius r $(x_i - a)^2 + (y_i - b)^2 = r^2$
- For **unknown** radius r, **known** gradients direction

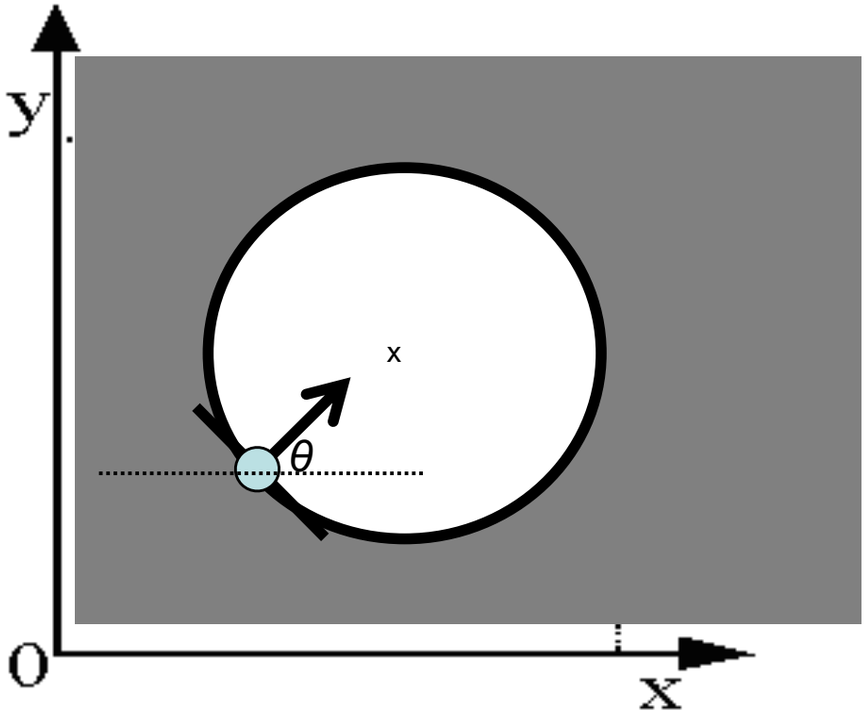
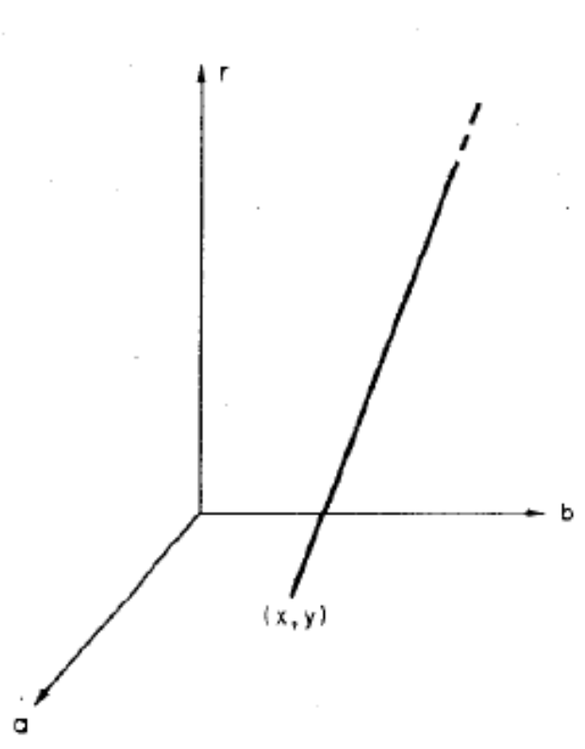


Image space



Hough space

Slide credit: Kristen Gruman

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

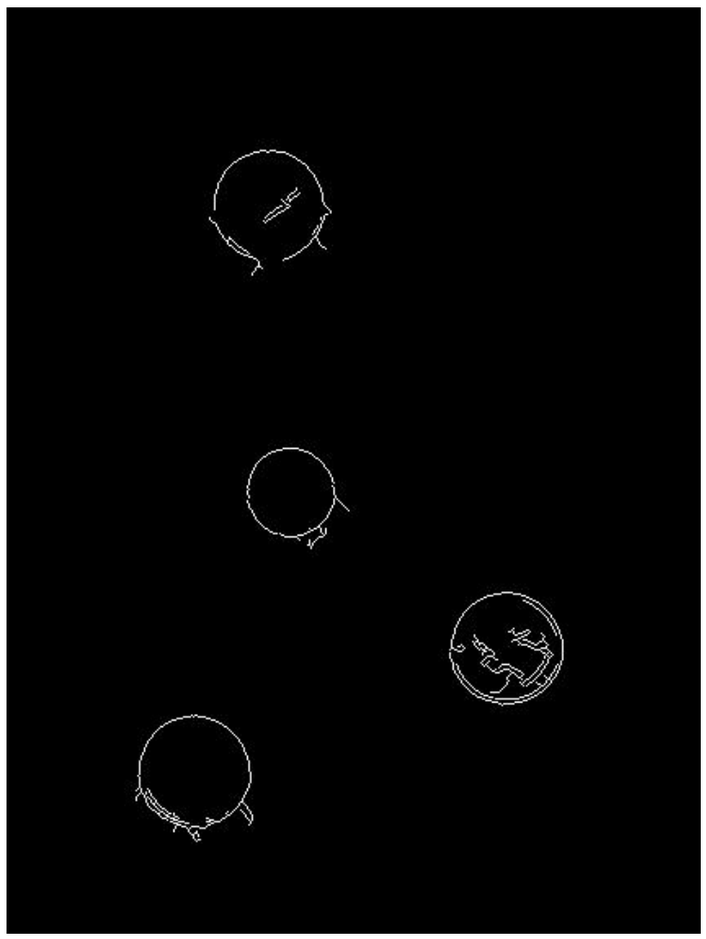
Time complexity per edge pixel?

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

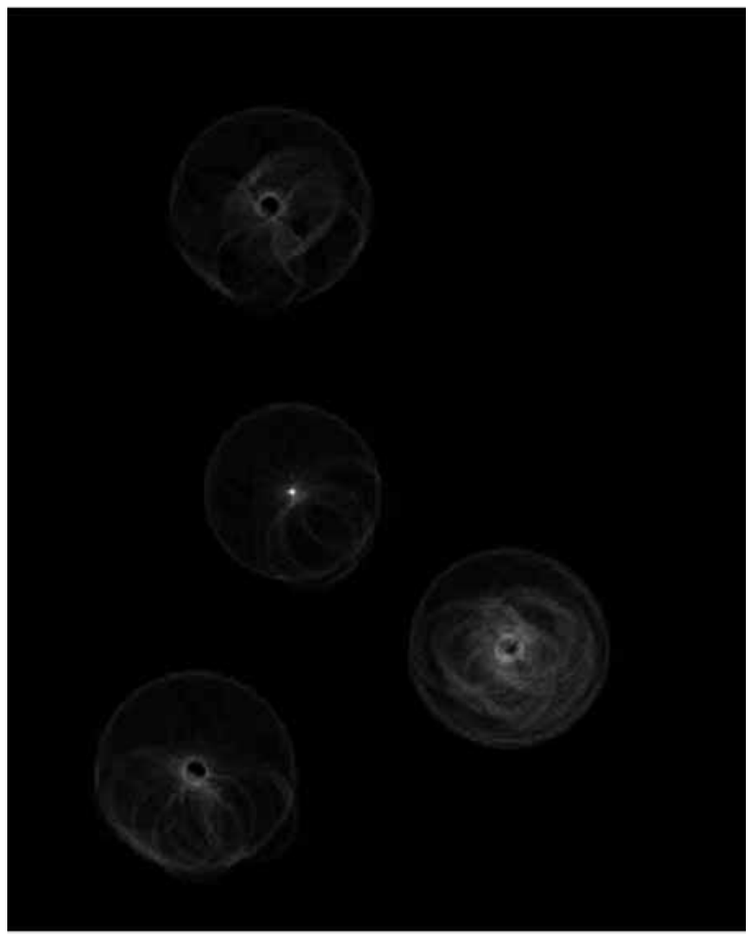
Example: detecting circles with Hough

Original

Edges



Votes: Penny

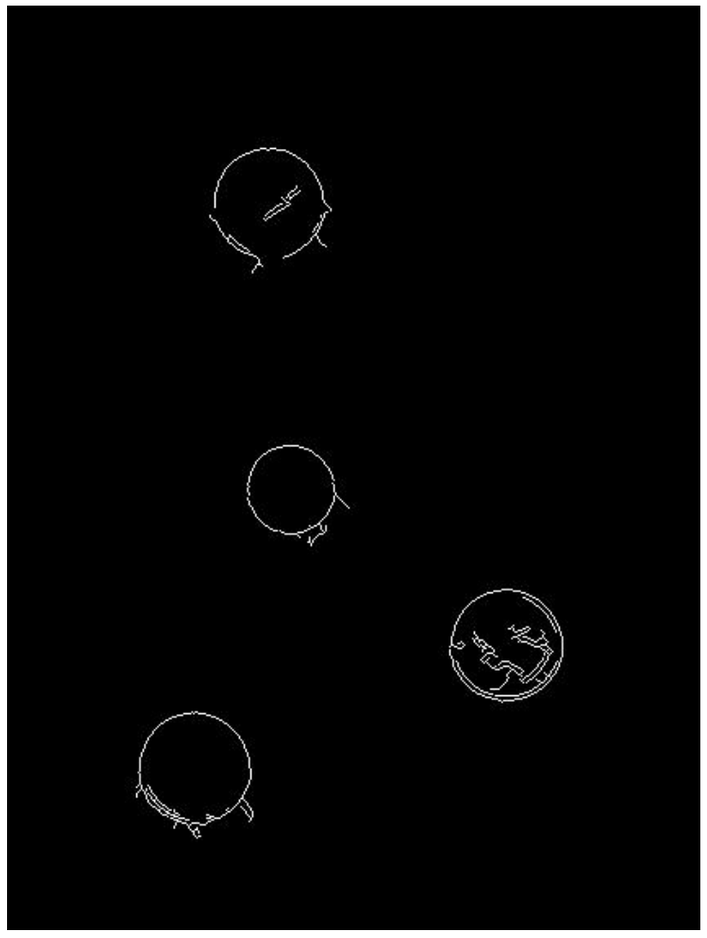


Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

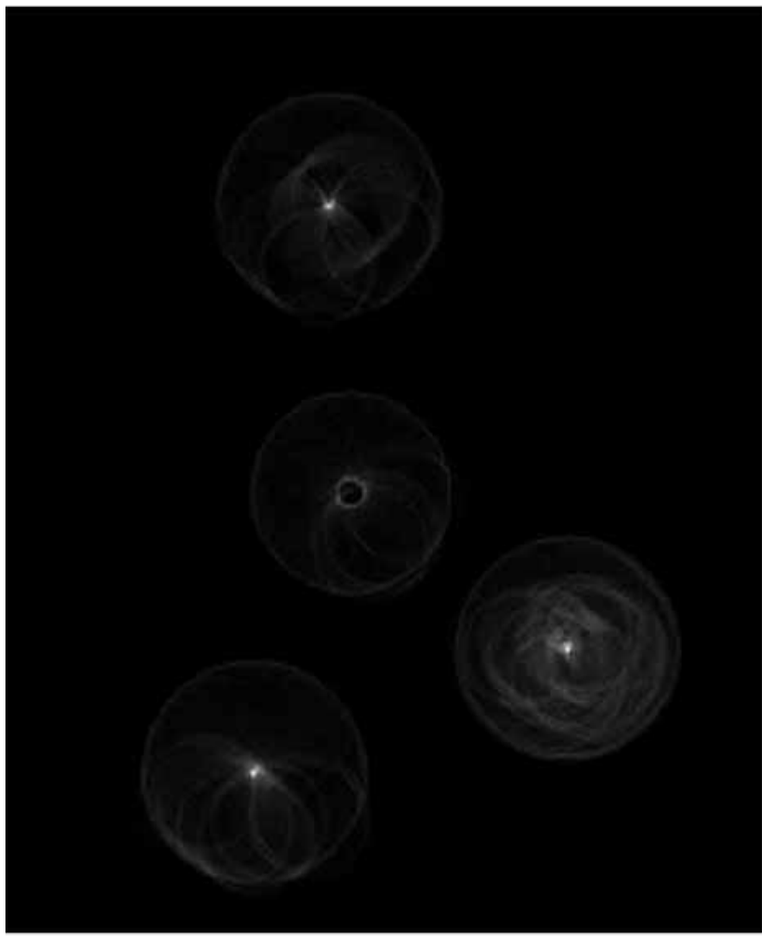
Example: detecting circles with Hough

Original

Edges



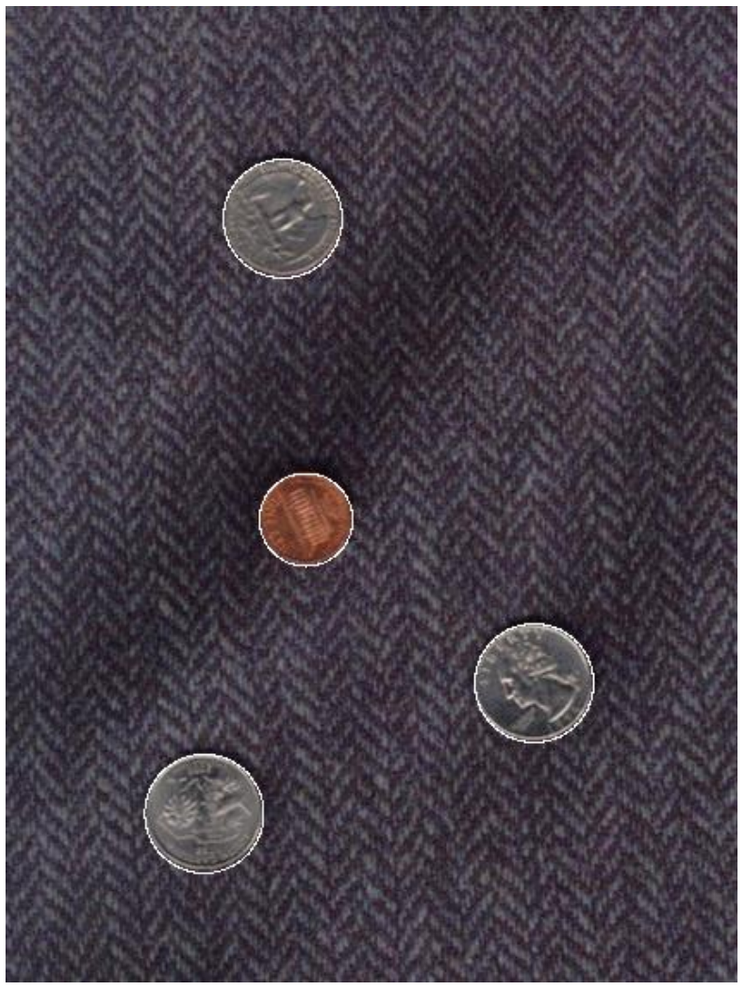
Votes: Quarter



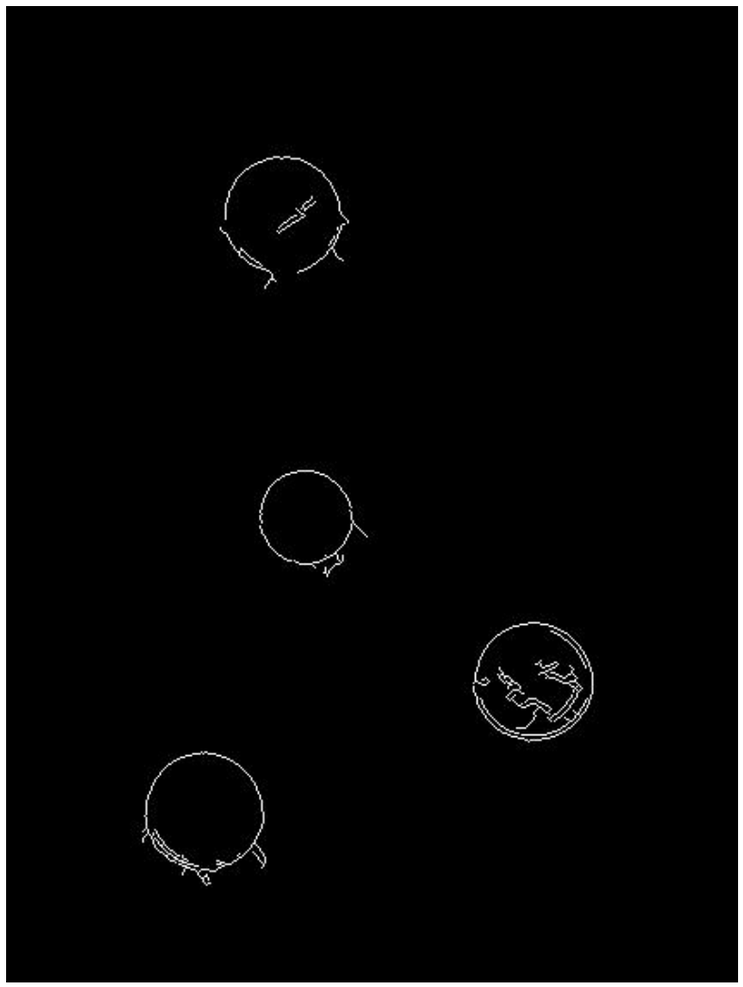
Coin finding sample images from: Vivek Kwatra

Example: detecting circles with Hough

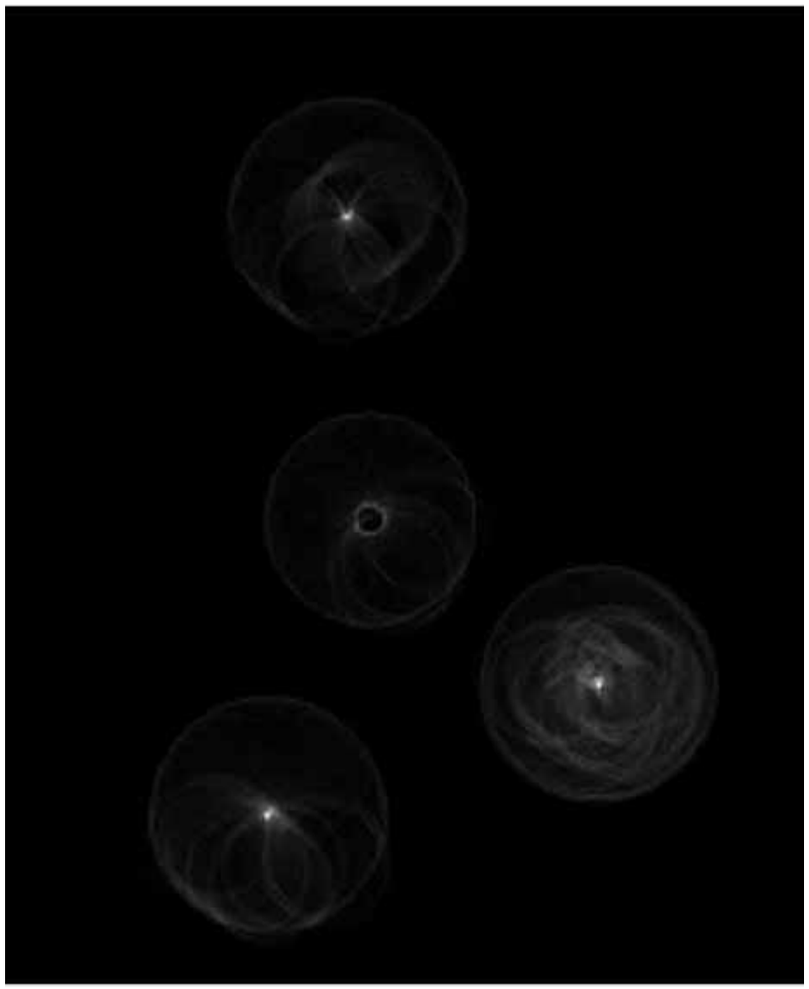
Combined detections



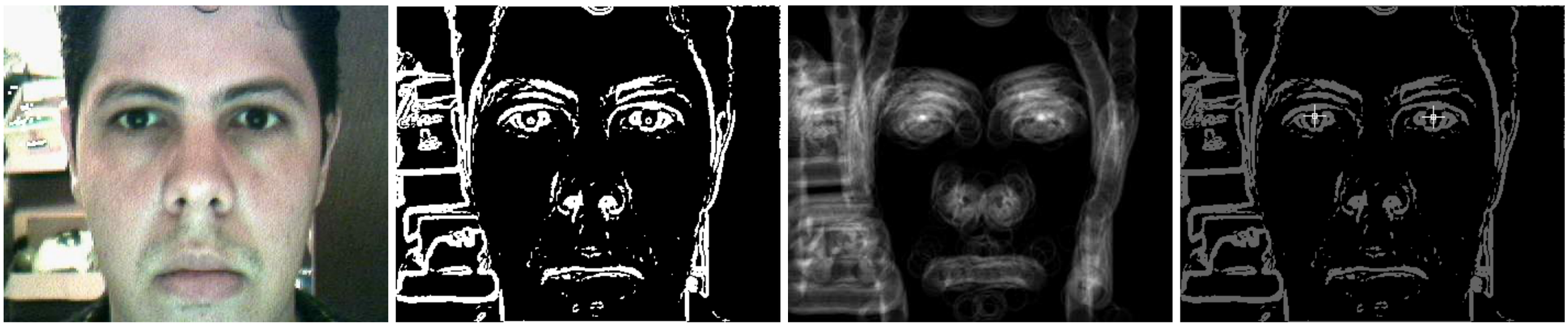
Edges



Votes: Quarter



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

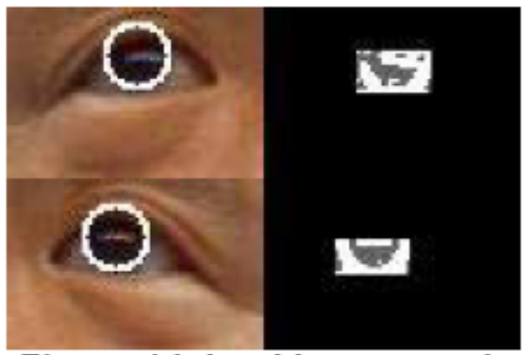


Figure 14. Looking upward

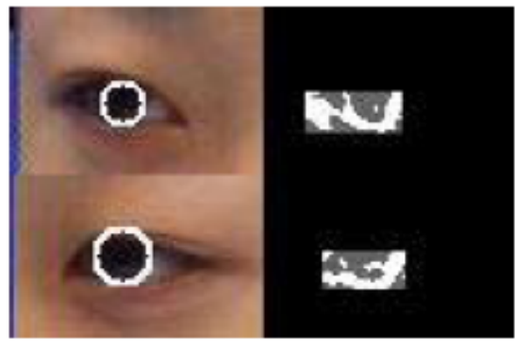


Figure 15. Looking sideways

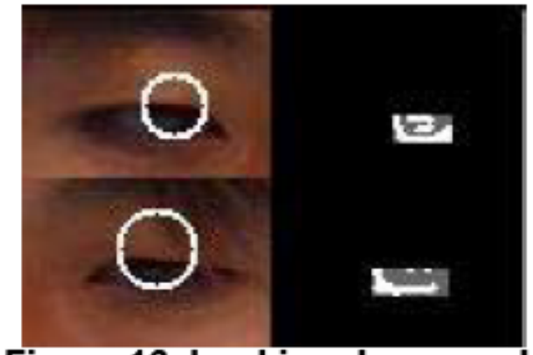


Figure 16. Looking downward

- 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

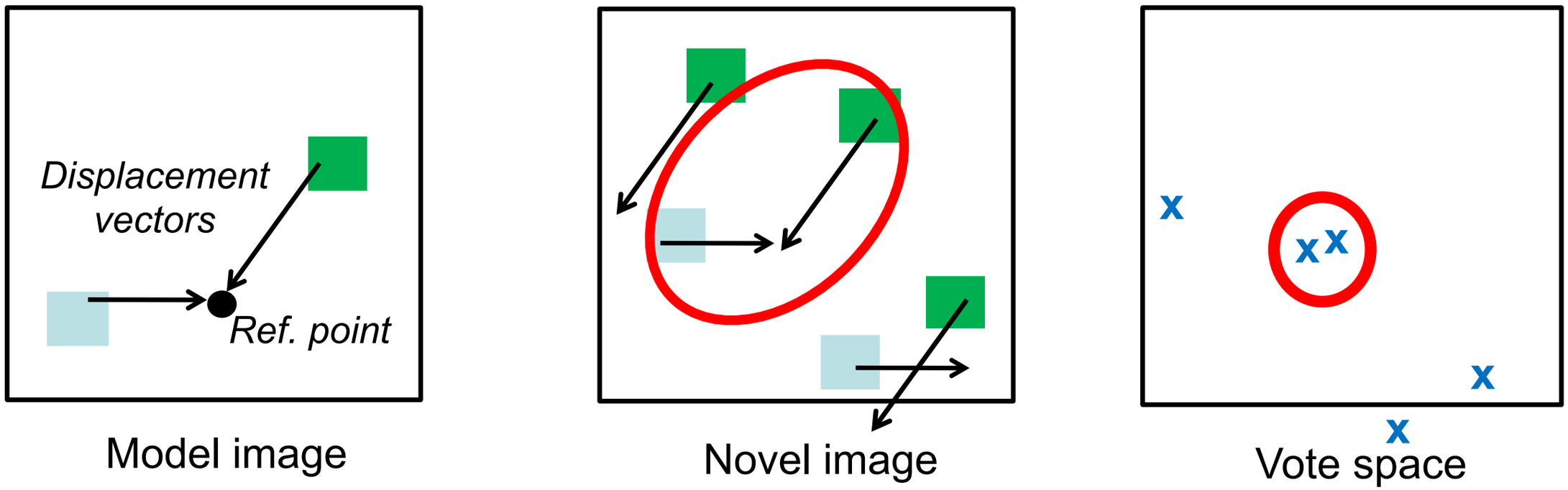
- Minimize irrelevant tokens first
- Choose a good grid / discretization
- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes

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

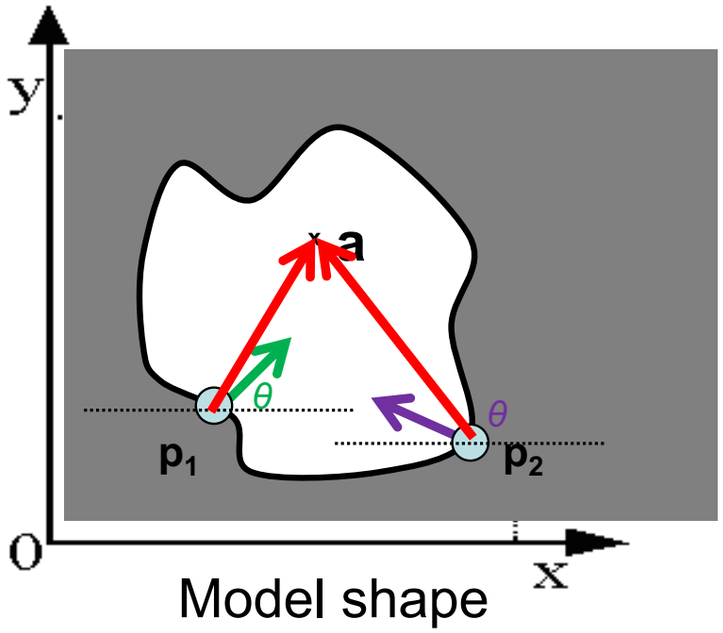
- What if we want to detect arbitrary shapes?



- Now suppose those colors encode gradient directions…

Generalized Hough Transform

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



Model shape

θ	…
θ	…
⋮	

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]

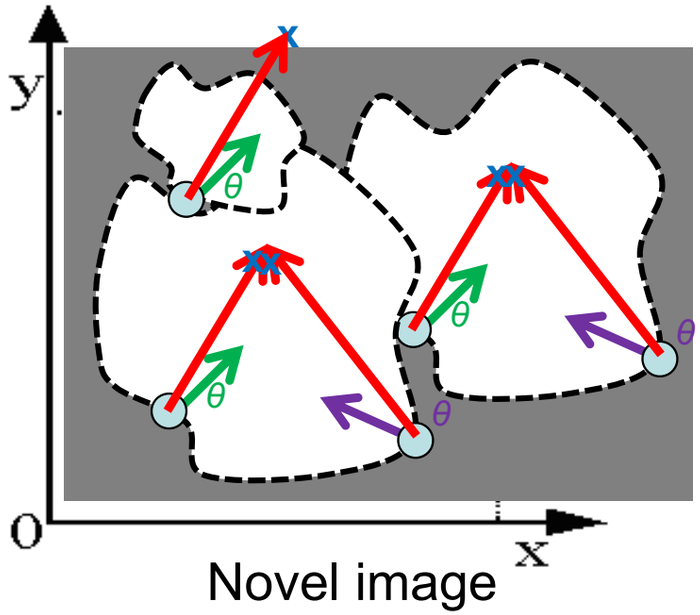
Generalized Hough Transform

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

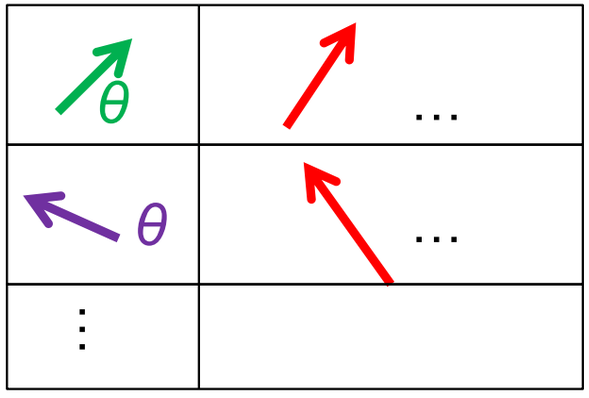
Detection procedure:

For each edge point:

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



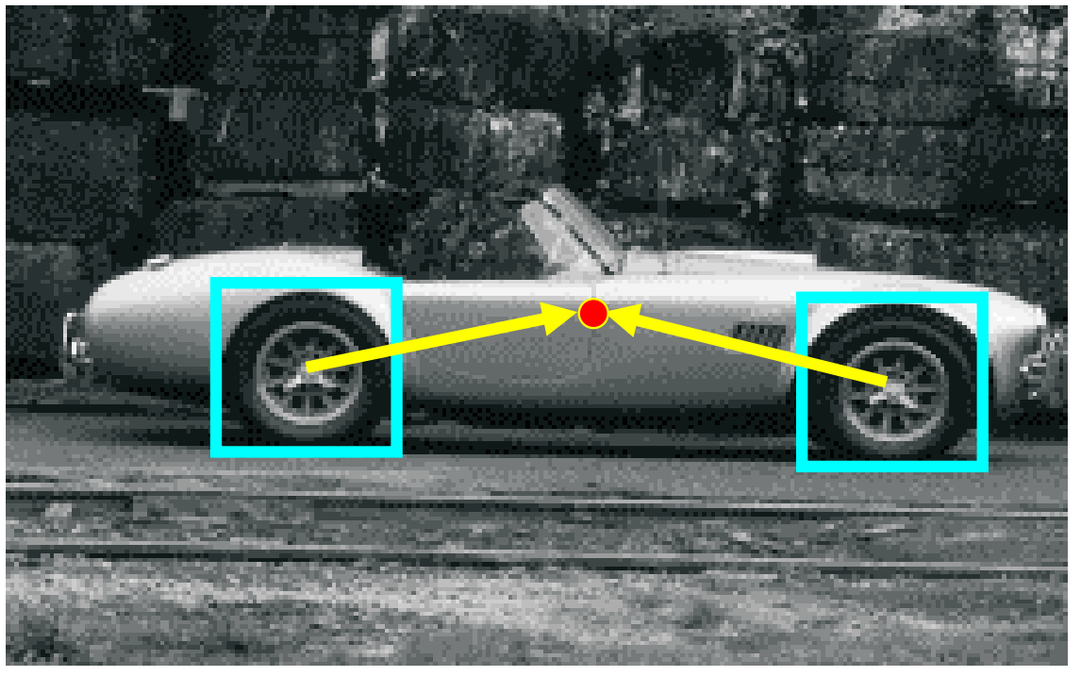
Novel image



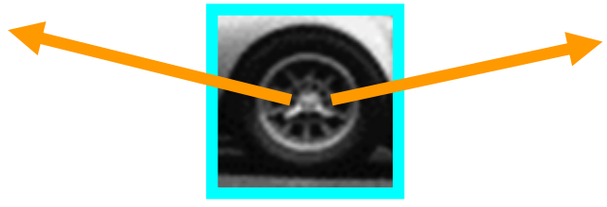
Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

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 matched local patterns.

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

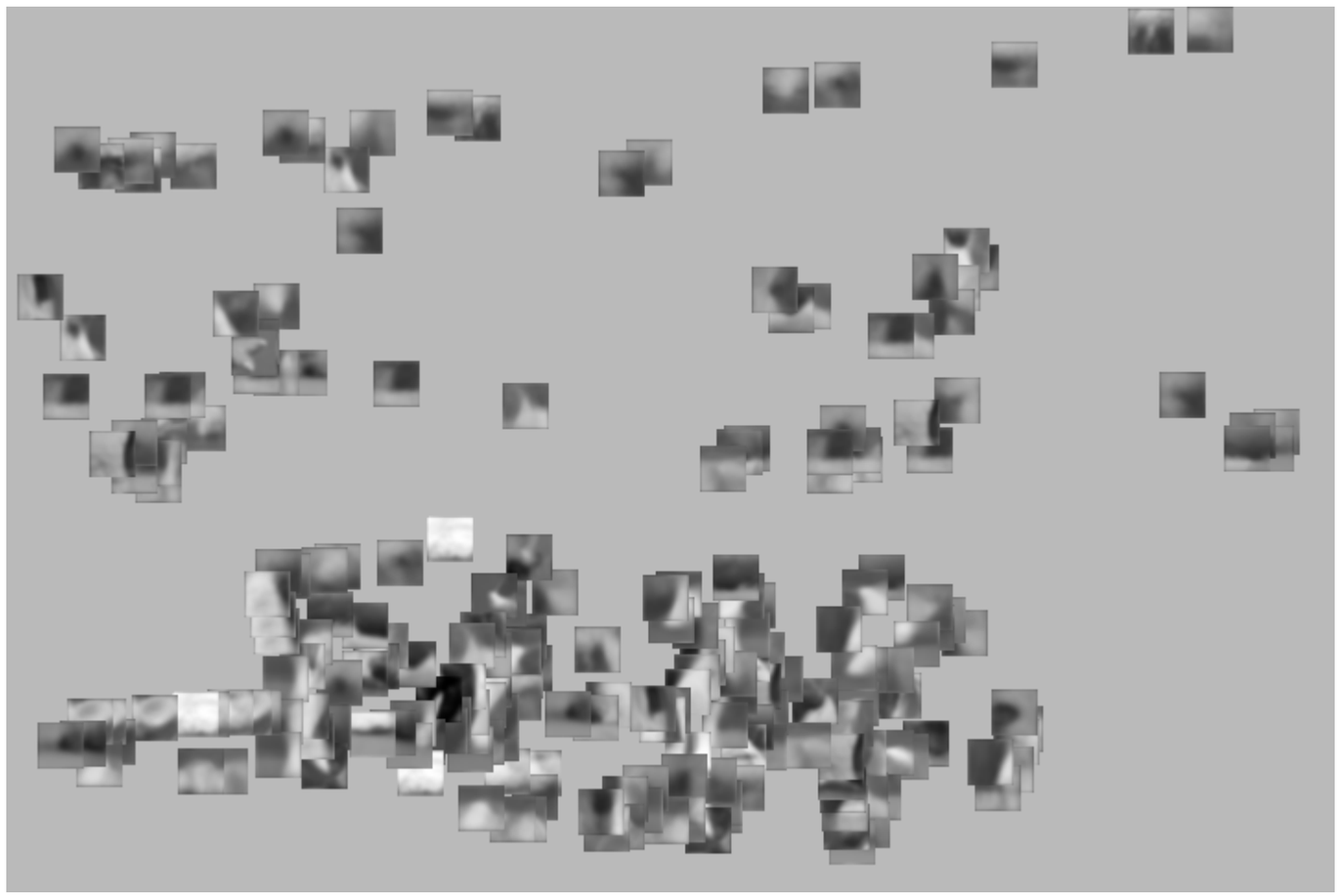
Example: Results on Cows

Original image

Example: Results on Cows

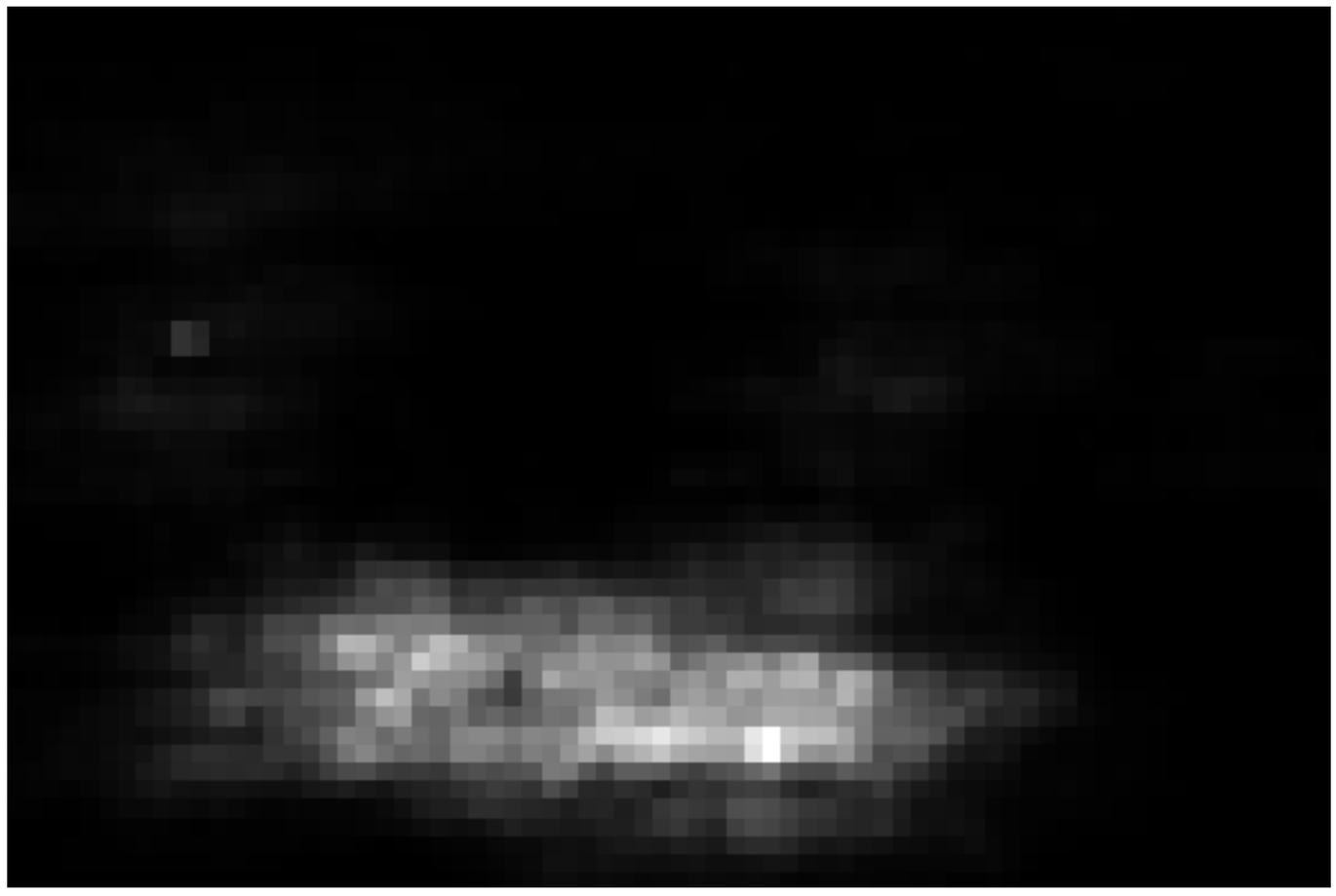
Interest points

Example: Results on Cows



Matched patches

Example: Results on Cows



Votes

Example: Results on Cows

1st hypothesis

Example: Results on Cows

2nd hypothesis

K. Grauman, B. Leibe

Example: Results on Cows

3rd hypothesis

Probabilistic model for fitting problem

- Idea: to fit the data to a model, firstly to make it clear how the data is generated
- Missing data problem: a statistical problem where some data is missing
  - some terms of the data vector are **missing**
  - simplify the problem by introducing **unknown** hidden variables (reveal hidden structure of the problem)

Example: line fitting and outliers

- To fit a line to a set of points $\mathbf{x}_i = (x_i, y_i)$. Among them, there are outliers but we don't know which ones are.
- The data generating process can be formulated as:

$$\begin{aligned} P(\boldsymbol{x}_i|a,b,c,\pi) &= P(\boldsymbol{x}_i, \text{line}|a,b,c,\pi) + P(\boldsymbol{x}_i, \text{outlier}|a,b,c,\pi) \\ &= P(\boldsymbol{x}_i|\text{line},a,b,c)P(\text{line}) + P(\boldsymbol{x}_i|\text{outlier},a,b,c)P(\text{outlier}) \\ &= P(\boldsymbol{x}_i|\text{line},a,b,c)\pi + P(\boldsymbol{x}_i|\text{outlier},a,b,c)(1-\pi). \end{aligned}$$

line equation: $ax + by + c = 0$ $P(\text{token comes from line}) = \pi$

- If we knew for every data item whether it came from the line or was an outlier, then fitting the line would be simple
- Data points without the labels: missing data problem

Mixture models and hidden variables

- Mixture model: a data item is generated by first choosing a mixture component (the line or the outlier), then generating the data item from that component
- Call the parameters for the lth component θ_l, the probability of choosing the lth component π_l, and write $\Theta = (\pi_1, \ldots, \pi_l, \theta_1, \ldots, \theta_l)$. Then, we can write the probability of generating x

$$p(\boldsymbol{x}|\Theta) = \sum_j p(\boldsymbol{x}|\theta_j)\pi_j$$

Mixture models and hidden variables

- The log-likelihood of the data for a general mixture model is

$$\mathcal{L}(\Theta) = \sum_{i \in \text{observations}} \log \left(\sum_{j=1}^{g} \pi_j p_j(\boldsymbol{x}_i | \theta_j) \right)$$

- Introduce a vector of indicator variables (one per component) that tells us from which component each data item came

$$\delta_{ij} = \begin{cases} 1 & \text{if item } i \text{ came from component } j \\ 0 & \text{otherwise} \end{cases}$$

- Maximize the complete data log-likelihood, to find the model paras.

$$\begin{aligned} \mathcal{L}_c(\Theta) &= \sum_{i \in \text{observations}} \log P(\boldsymbol{x}_i, \delta_i | \Theta) \\ &= \sum_{i \in \text{observations}} \log \prod_{j \in \text{components}} \left[p_j(\boldsymbol{x}_i | \theta_j) \pi_j \right]^{\delta_{ij}} \\ &= \sum_{i \in \text{observations}} \left(\sum_{j \in \text{components}} \left[(\log p_j(\boldsymbol{x}_i | \theta_j) \log \pi_j) \, \delta_{ij} \right] \right) \end{aligned}$$

The EM algorithm
Dempster *et al.*, 1977; McLachlan and Krishnan, 1997

- If we knew the missing data, we could estimate the parameters effectively. Similarly, if we knew the parameters, the missing data would follow. This suggests an iterative algorithm

    1. Obtain some estimate of the missing data using a guess at the parameters
    2. Form a maximum likelihood estimate of the free parameters using the estimate of the missing data.

- The line fitting problem

    1. Obtain some estimate of which points lie on the line and which are off lines, using an estimate of the line.
    2. Form a revised estimate of the line, using this information.

The EM (Expectation-Maximization) algorithm

- **E-step**: computing an expected value for the complete data log-likelihood using the incomplete data and the current value of the parameters

$$Q(\Theta; \Theta^{(s)}) = E_{\delta|\boldsymbol{x},\Theta^{(s)}} \mathcal{L}_c(\Theta)$$

- **M-step**: maximizing this object as a function of Θ

$$\Theta^{(s+1)} = \arg\max_{\Theta} Q(\Theta; \Theta^{(s)})$$

EM for Gaussian mixture model

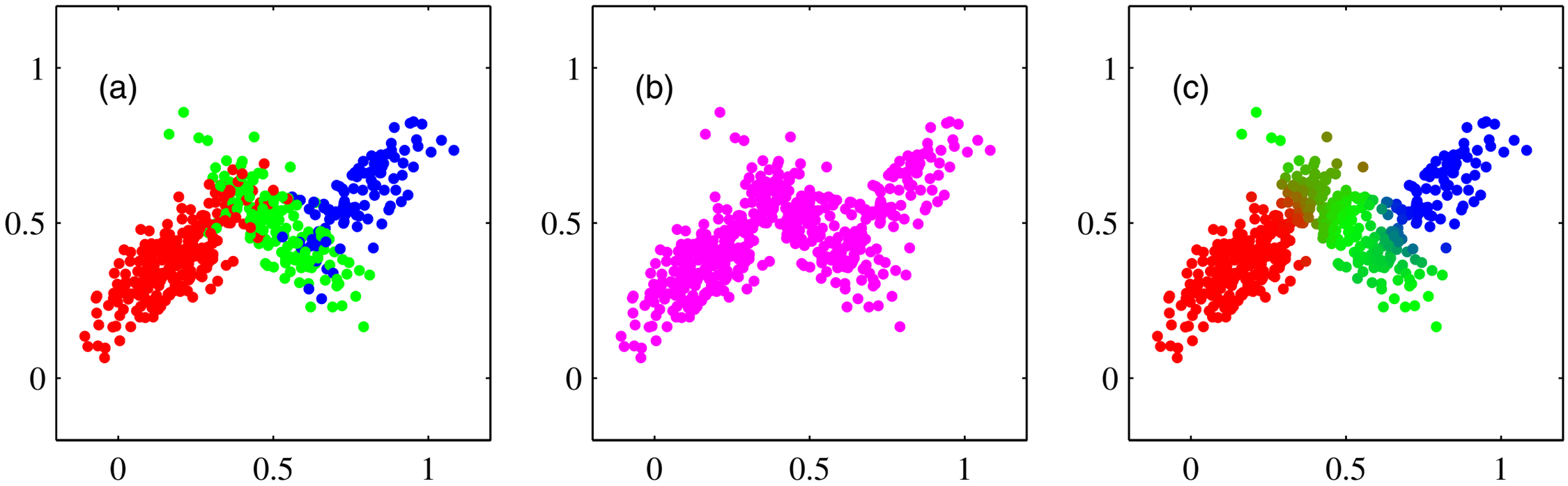


Figure 9.5 Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ in which the three states of $\mathbf{z}$, corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution $p(\mathbf{x})$, which is obtained by simply ignoring the values of $\mathbf{z}$ and just plotting the $\mathbf{x}$ values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point $\mathbf{x}_n$, obtained by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Bishop: PRML

EM for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{9.23}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

where

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}). \tag{9.27}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right\} \tag{9.28}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

EM for Gaussian mixture model

- Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients)

    1. Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k, and evaluate the initial value of the log likelihood.

2. **E step**. Evaluate the responsibilities using the current parameter values

3. **M step**. Re-estimate the parameters using the current responsibilities

4. Evaluate the log likelihood and check the convergence

Difficulties of EM

- EM is inclined to get stuck in local minima
- Some points will have extremely small expected weights

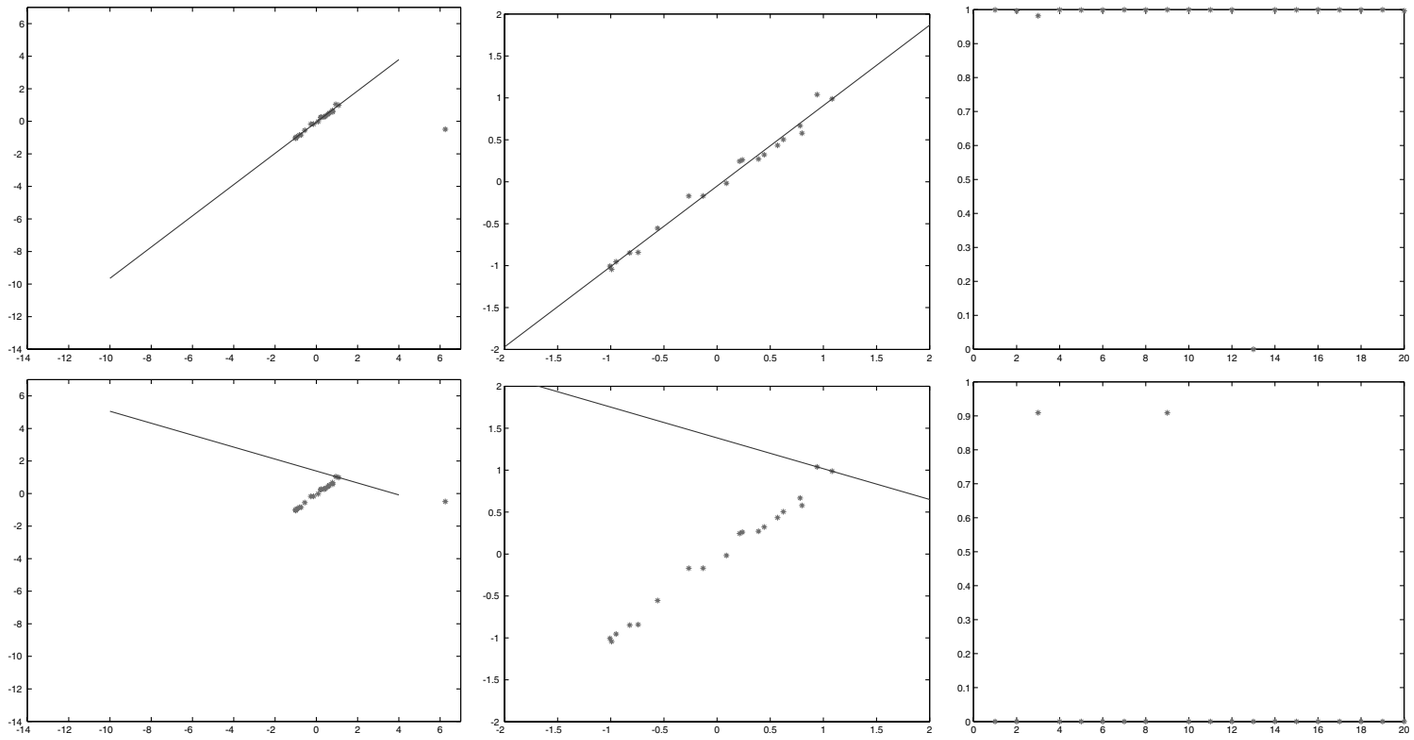


FIGURE 10.9: EM can be used to reject outliers. Here we demonstrate a line fit to the second dataset of Figure 10.5. The **top row** shows the correct local minimum, and the **bottom row** shows another local minimum. The **first column** shows the line superimposed on the data points using the same axes as Figure 10.5; the **second column** shows a detailed view of the line, indicating the region around the data points; and the **third column** shows a plot of the probability that a point comes from the line, rather than from the noise model, plotted against the index of the point. Notice that at the correct local minimum, all but one point is associated with the line, whereas at the incorrect local minimum, there are two points associated with the line and the others are allocated to noise.