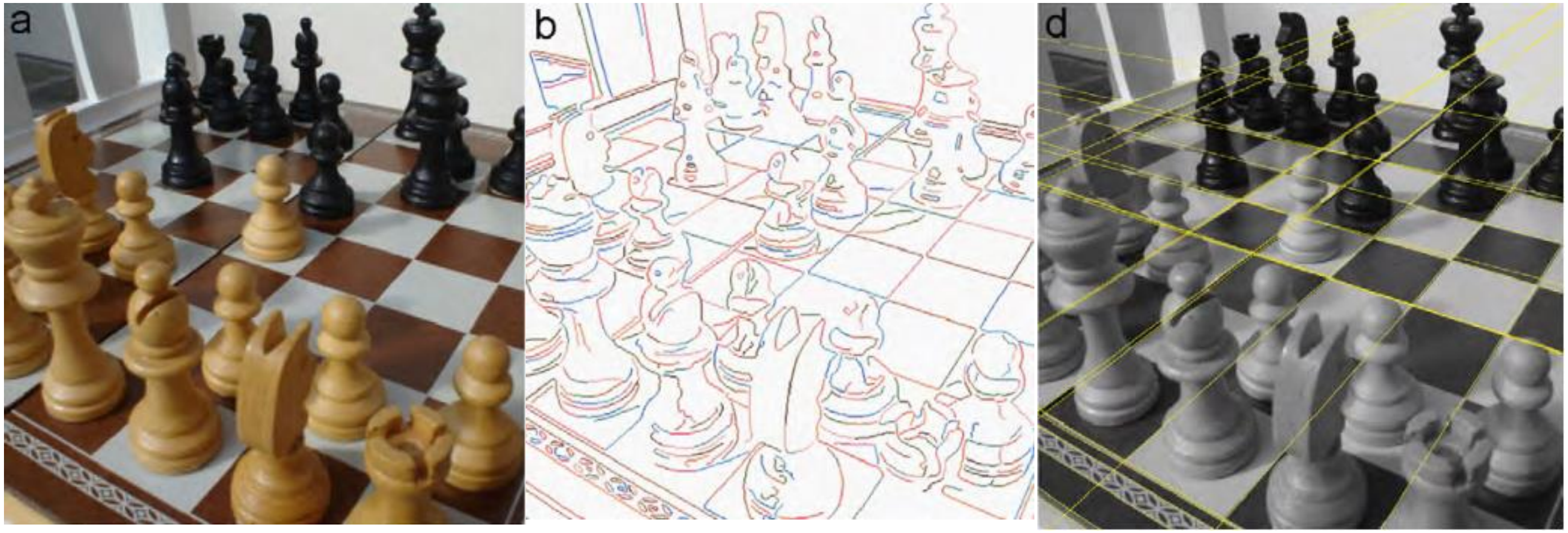
Lecture 9:
Hough Transform and Thresholding

Saad Bedros
sbedros@umn.edu

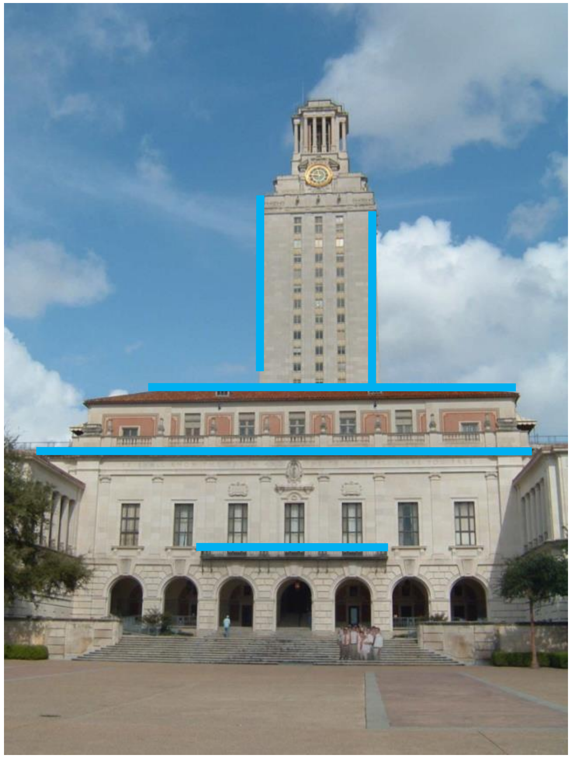
Hough Transform

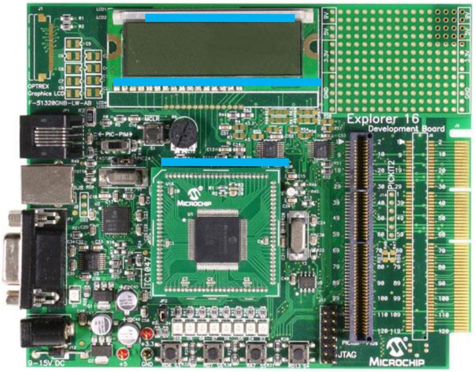
- Robust method to find a shape in an image
- Shape can be described in parametric form
- A voting scheme is used to determine the correct parameters



Example: Line fitting

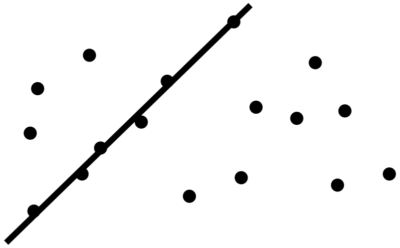
- Why fit lines?
  Many objects characterized by presence of straight lines

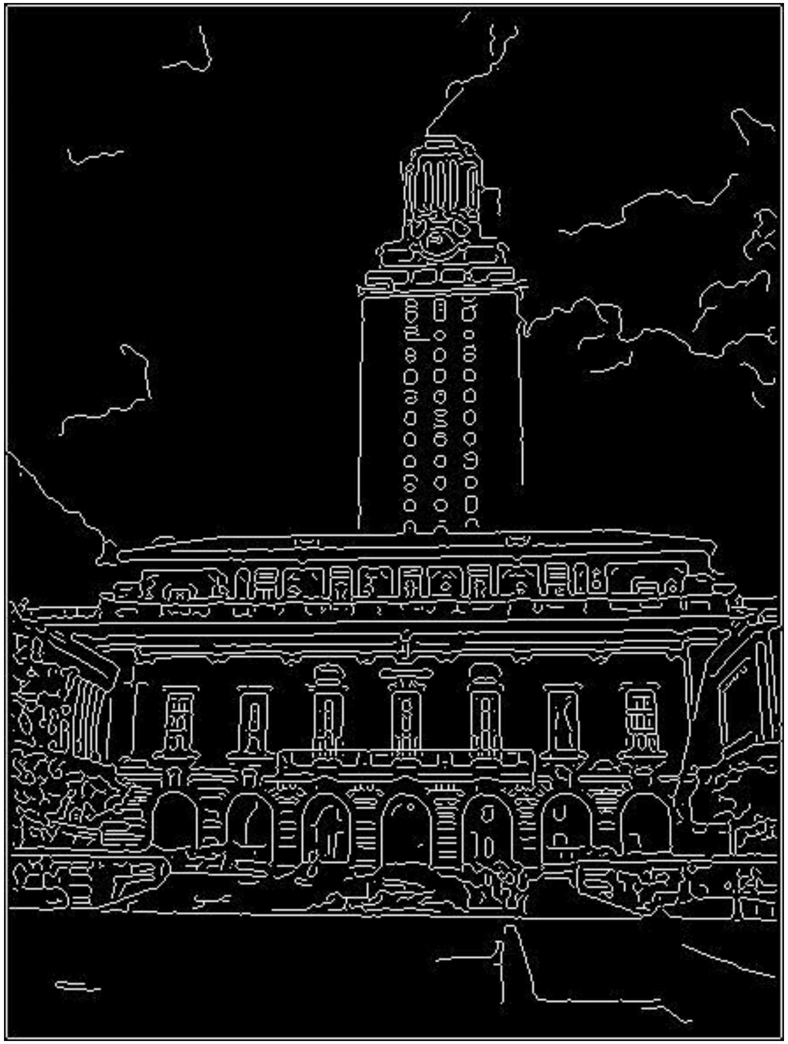




- Can we do it with edge detection? Use edge information

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.

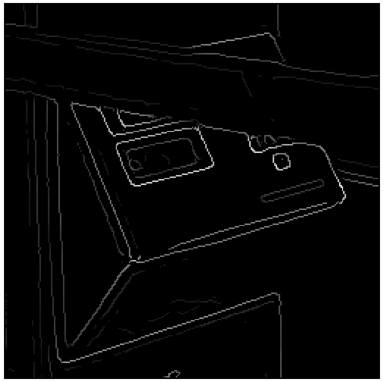
Fitting lines: Hough transform

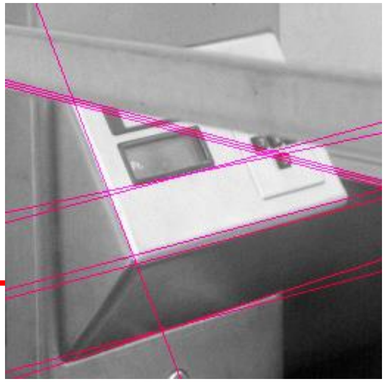
- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?

- **Hough Transform** is a voting technique that can be used to answer all of these questions.

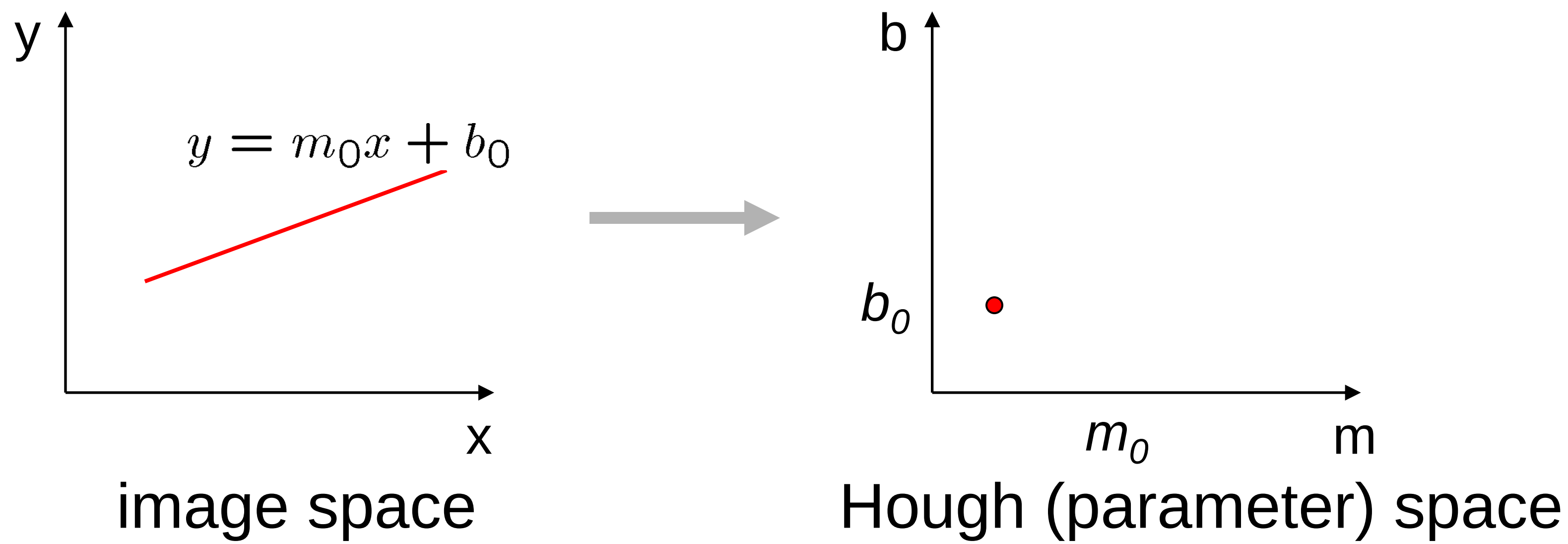
  Main idea:

  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

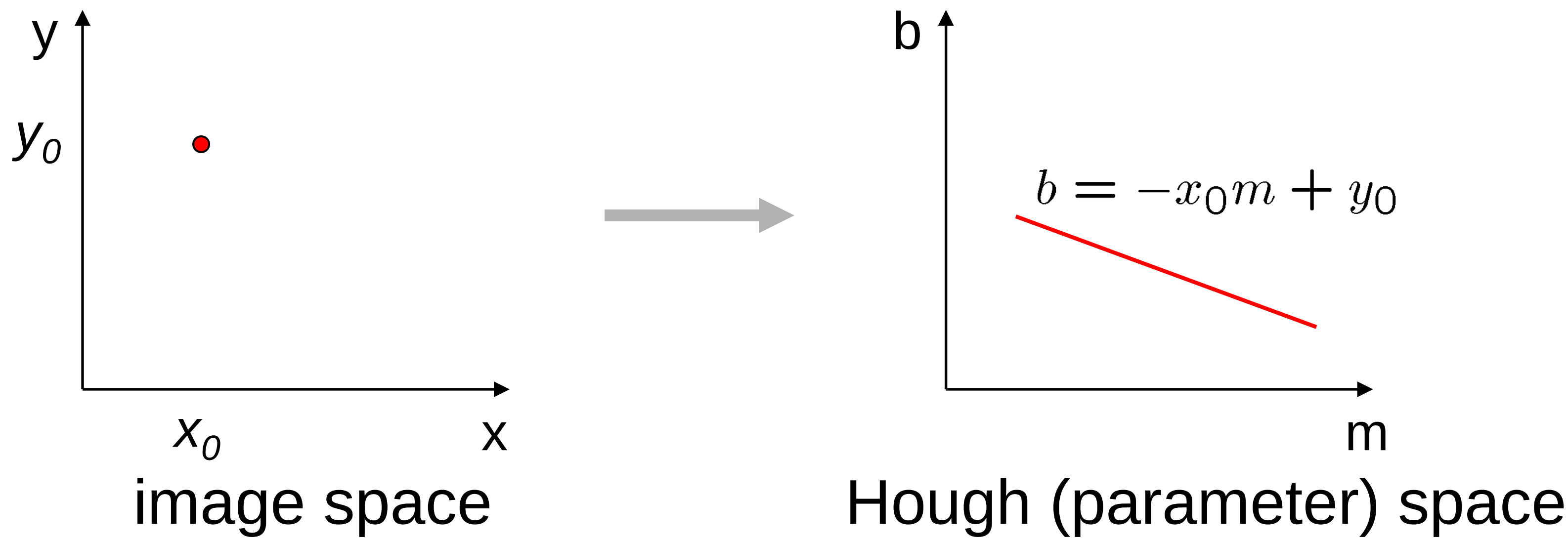


Connection between image (x,y) and Hough (m,b) spaces

- 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



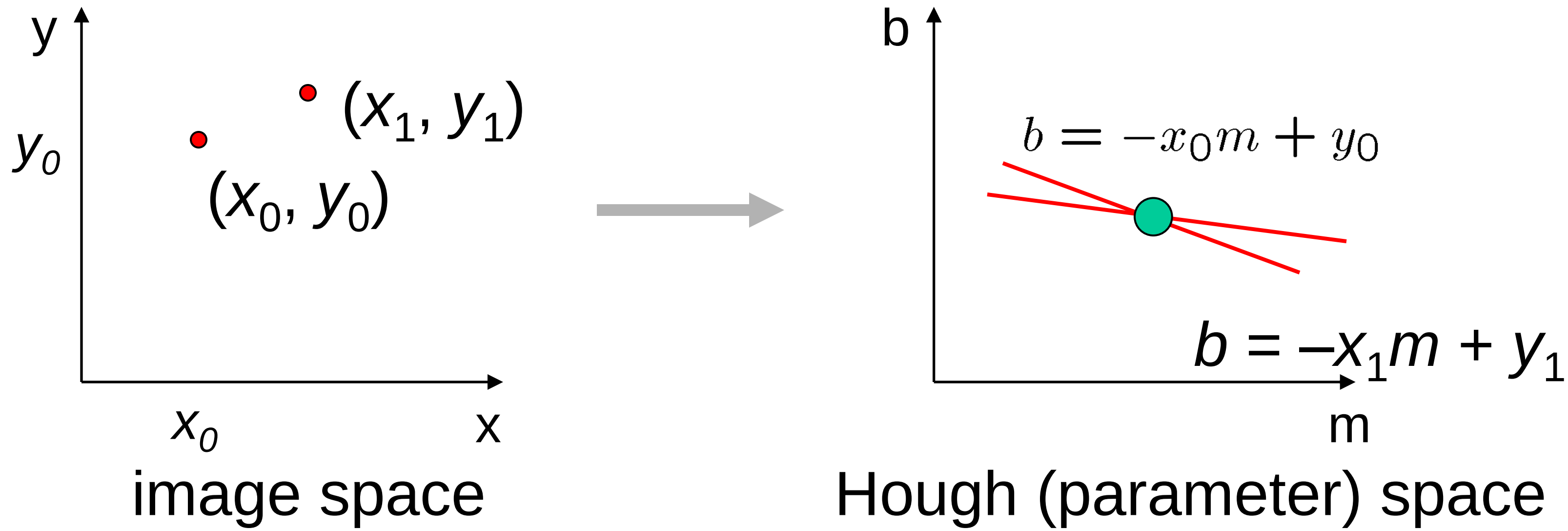
Connection between image (x,y) and Hough (m,b) spaces

- 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 $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Finding lines in an image: Hough algorithm

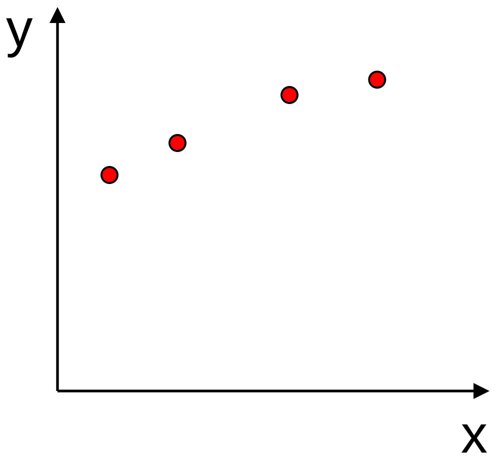
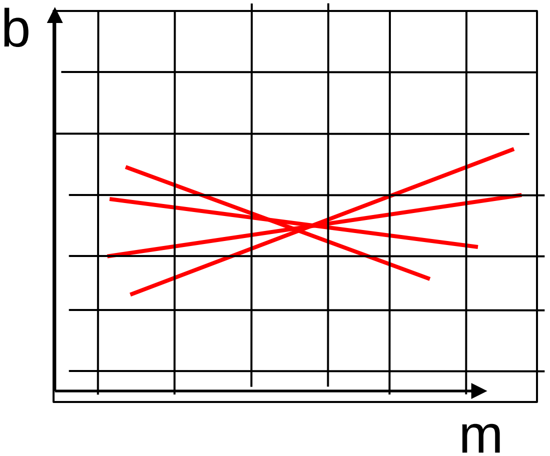


image space



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 for Line Detection

Find a subset of n points on an image that lie on the same straight line.

Write each line formed by a pair of these points as

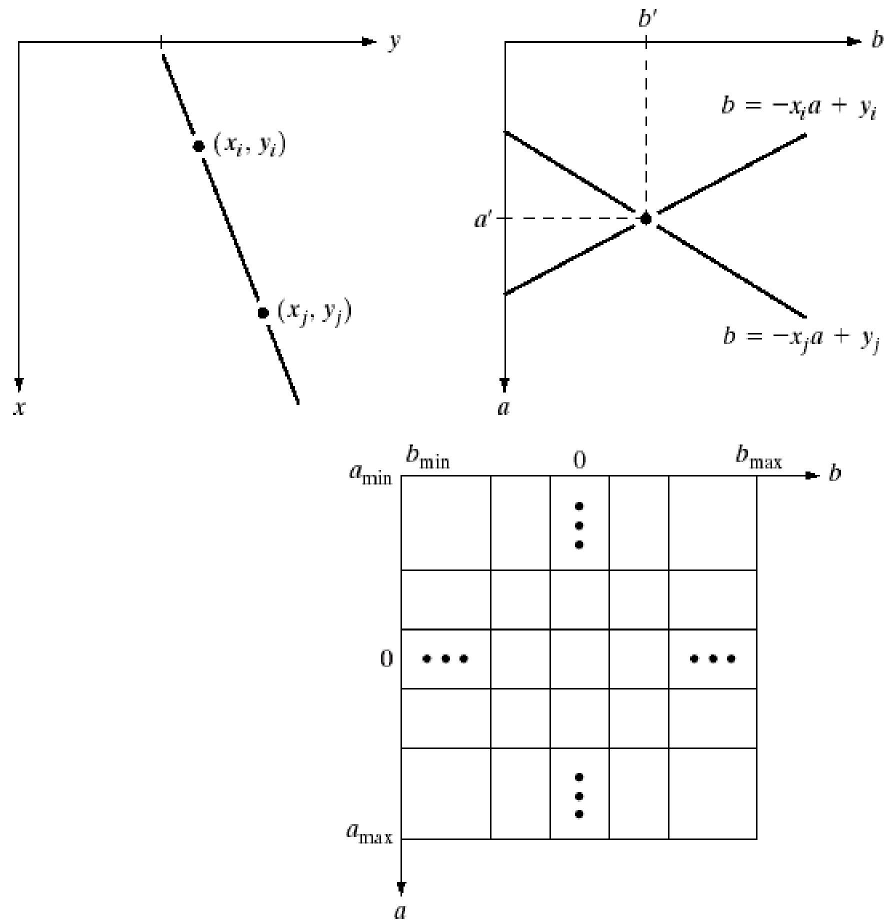
$$y_i = ax_i + b$$

Then plot them on the parameter space (a, b):

$$b = x_i\, a + y_i$$

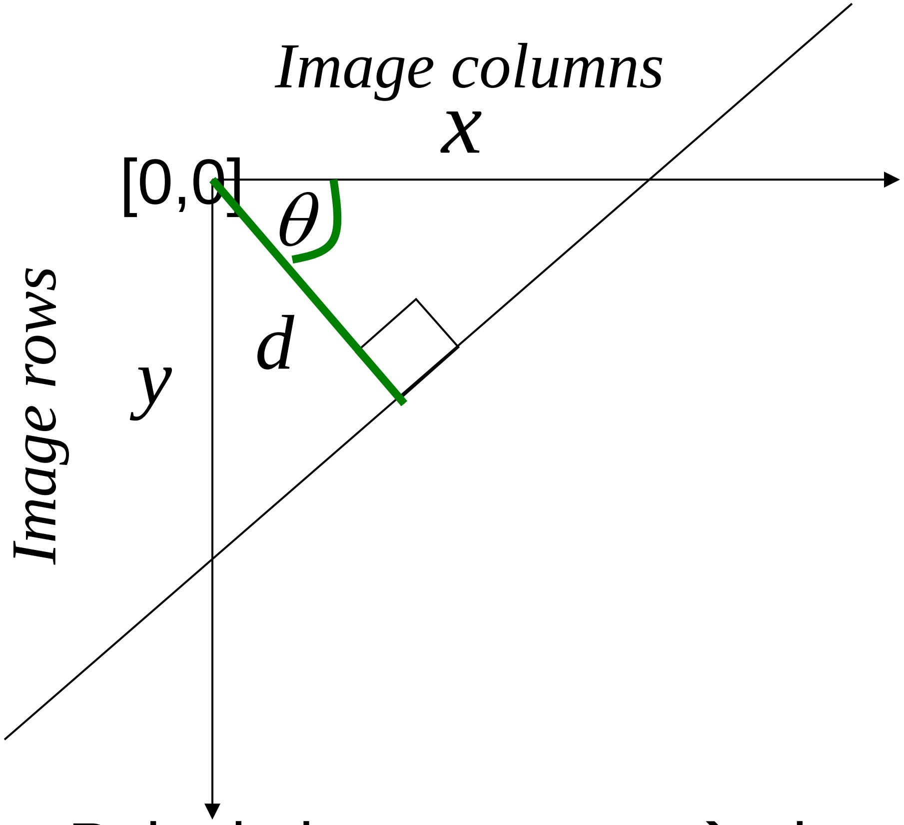
All points (x_i, y_i) on the same line will pass the same parameter space point (a, b).

Quantize the parameter space and tally # of times each points fall into the same accumulator cell. The cell count = # of points in the same line.



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

Kristen Grauman

Hough Transform in (ρ, θ) plane

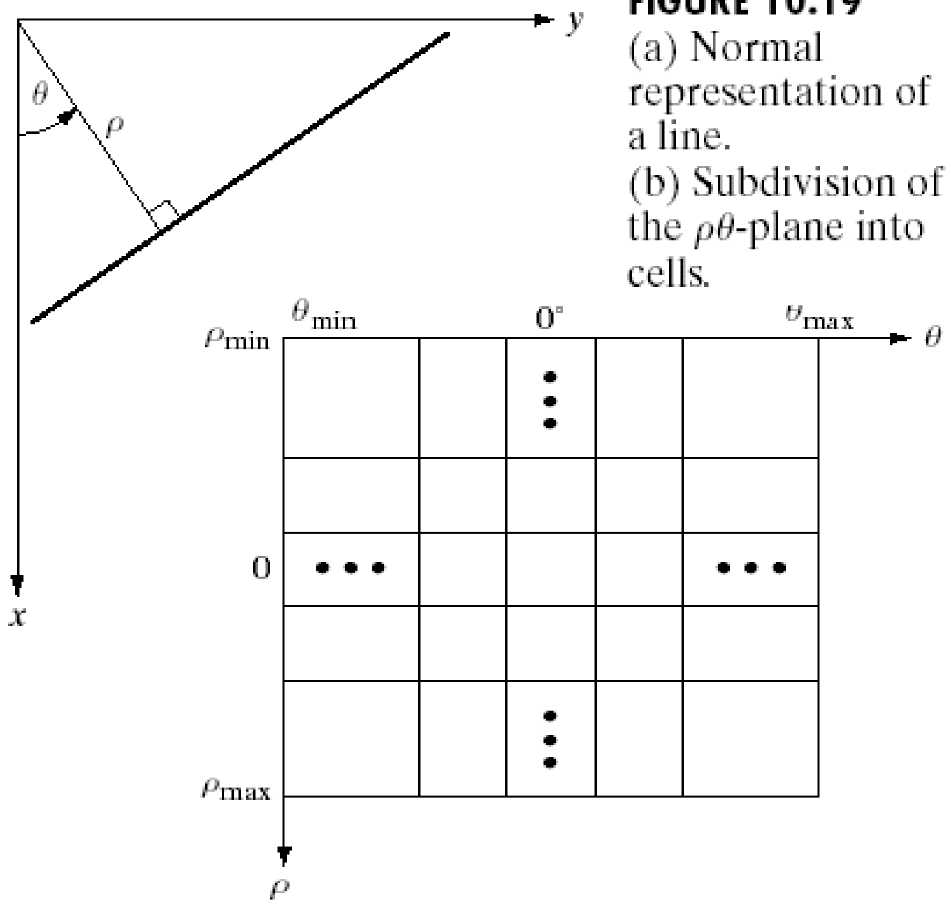


FIGURE 10.19
(a) Normal representation of a line.
(b) Subdivision of the $\rho\theta$-plane into cells.

To avoid infinity slope, use polar coordinate to represent a line.

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

Q points on the same straight line gives Q sinusoidal curves in (ρ, θ) plane intersecting at the same (ρ_i, θ_i) cell.

Hough transform algorithm

Using the polar parameterization:

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

Basic Hough transform algorithm

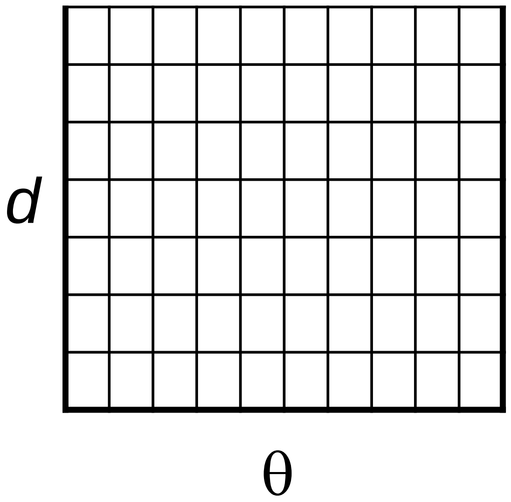
1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

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

H: accumulator array (votes)

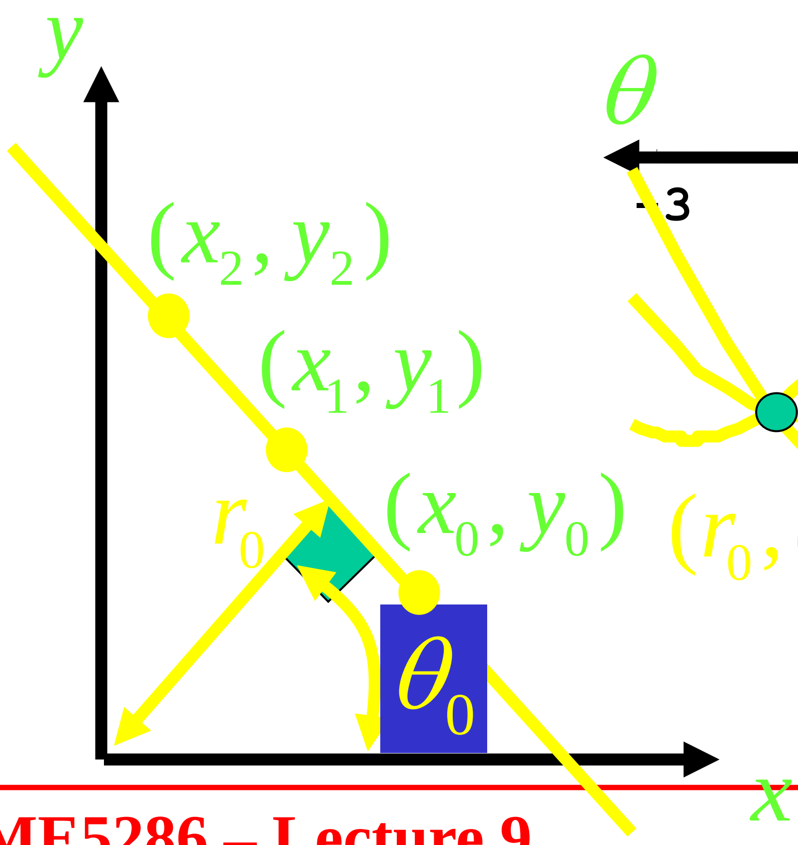


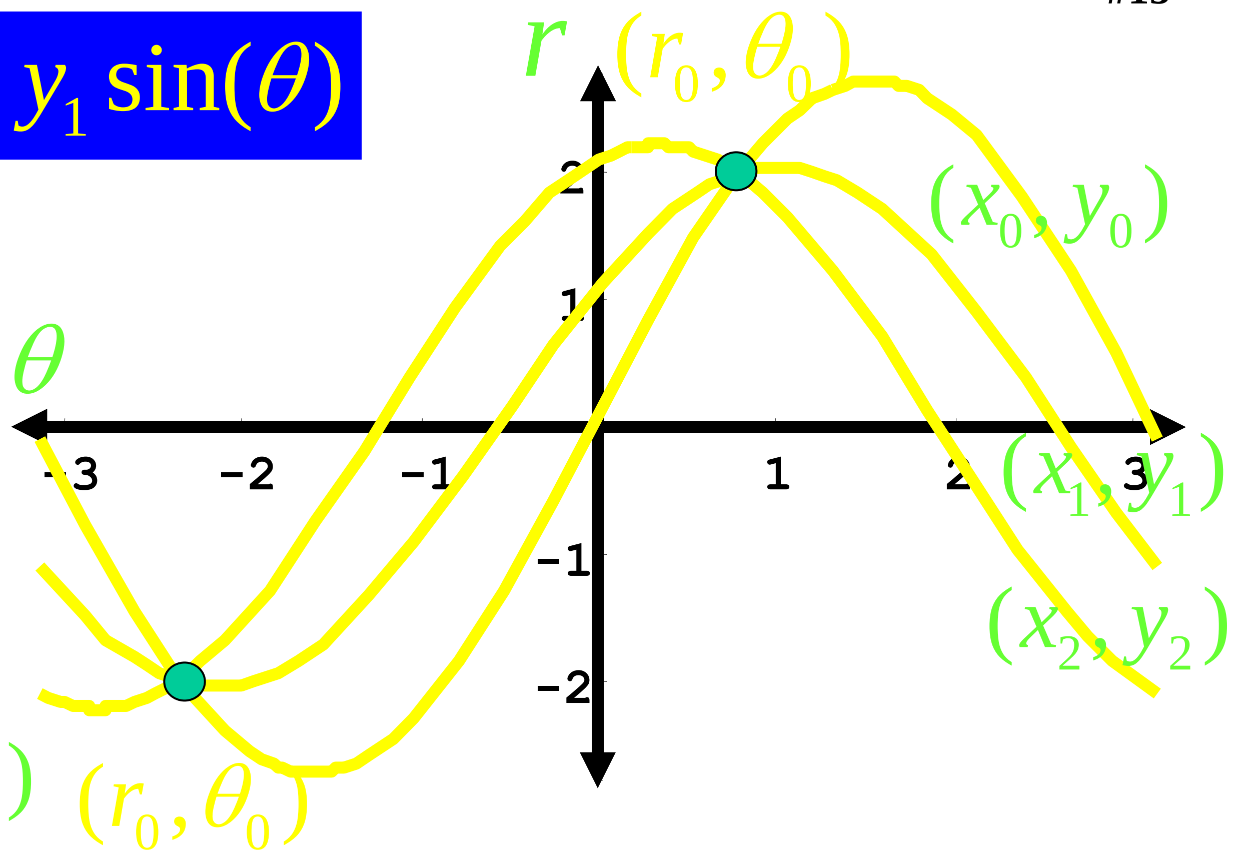
Time complexity (in terms of number of votes per pt)?

Source: Steve Seitz

Hough Transform for Lines

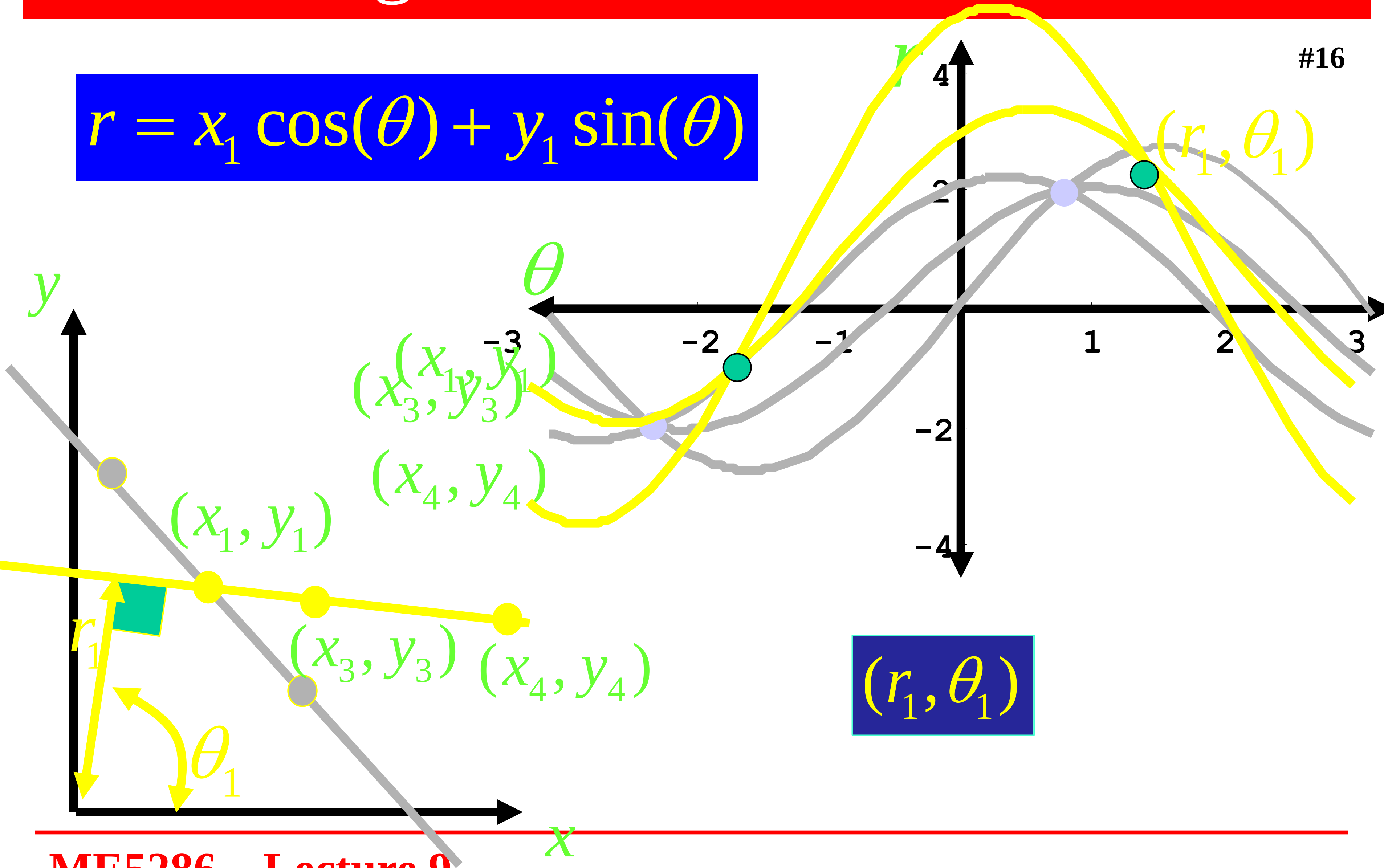
$$r = x_1 \cos(\theta) + y_1 \sin(\theta)$$



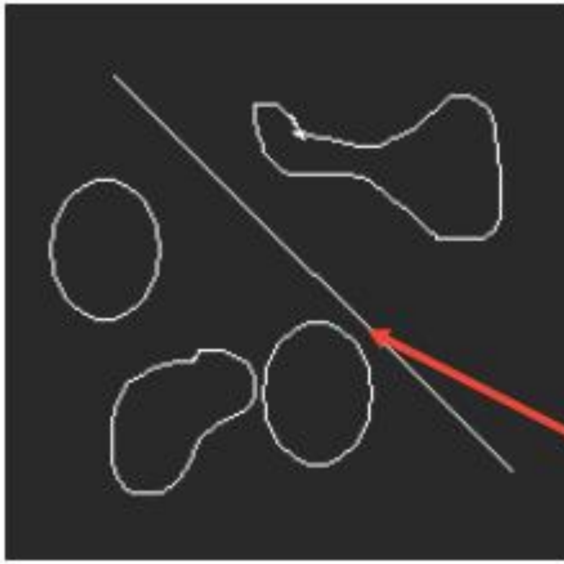


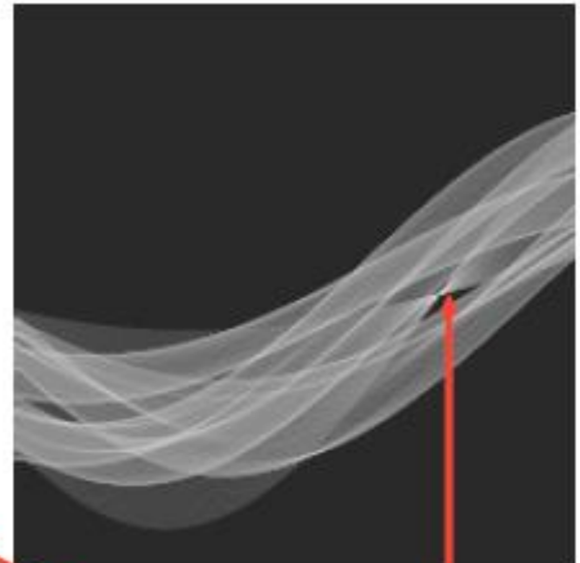
Hough Transform for Lines

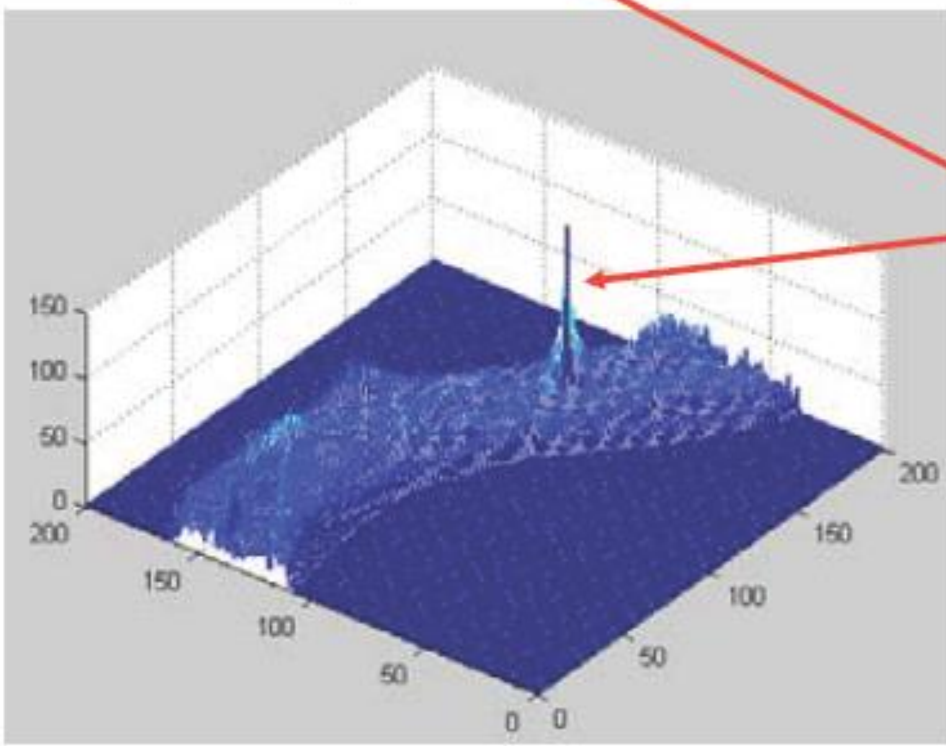
$$r = x_1 \cos(\theta) + y_1 \sin(\theta)$$



(r_1, θ_1)







Peak in the parametric space that corresponds to the line

Hough Transform for Lines

- Domain of the parametric space:

$$r \in \left[-\sqrt{M^2 + N^2}, \sqrt{M^2 + N^2} \right], \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2} \right]$$

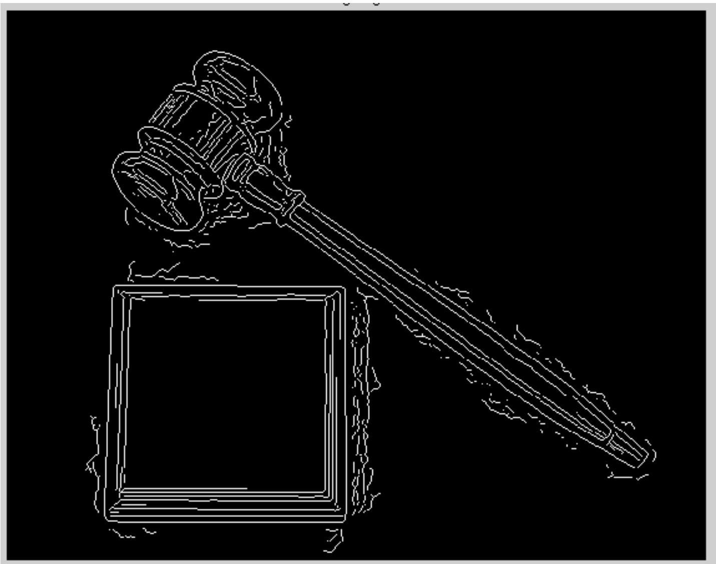
M and N image resolution

Not just lines, any parametric curve!

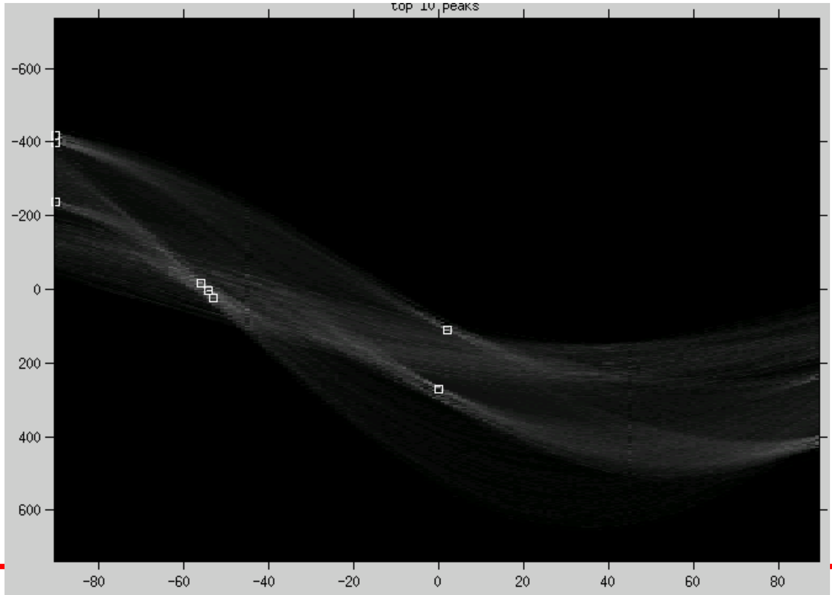
However increase of dimensions of the parametric space

Original image

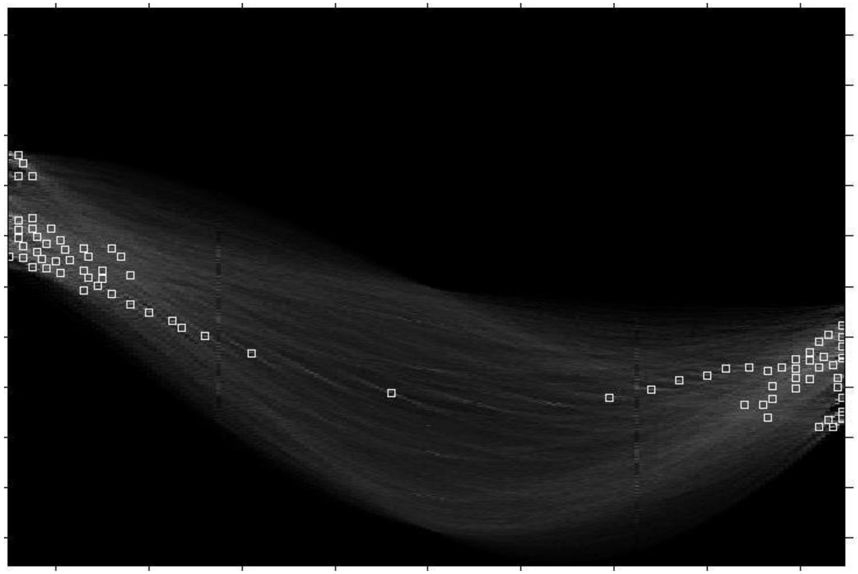
Canny edges

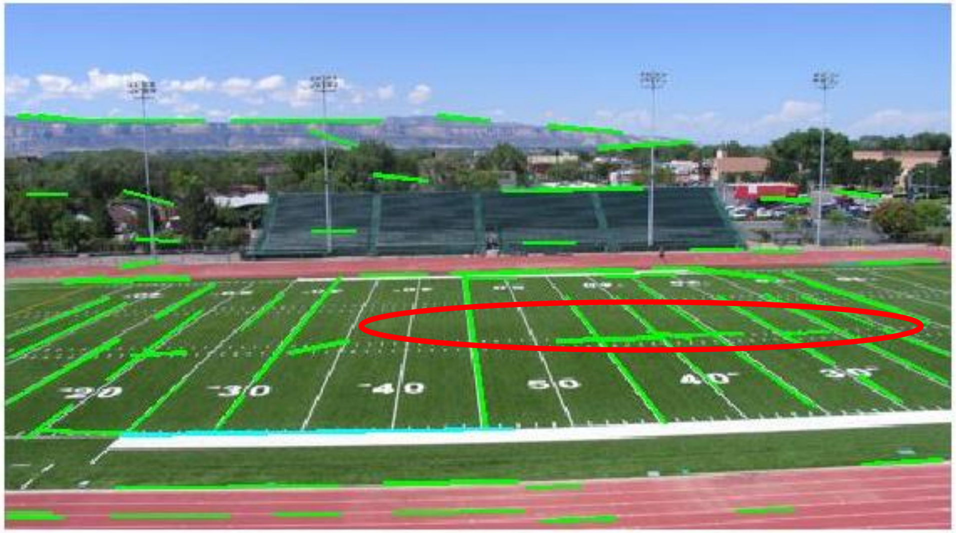


Vote space and top peaks



Kristen Grauman





Showing longest segments found

Kristen Grauman

Impact of noise on Hough

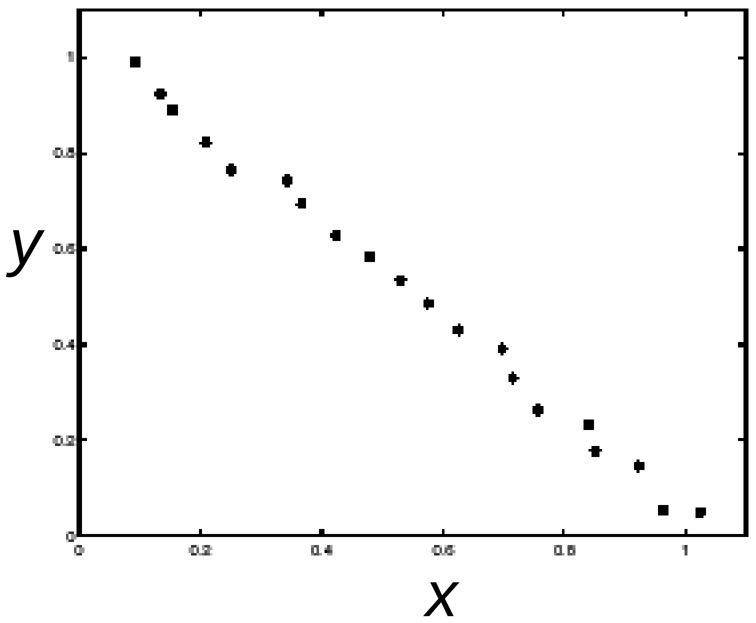
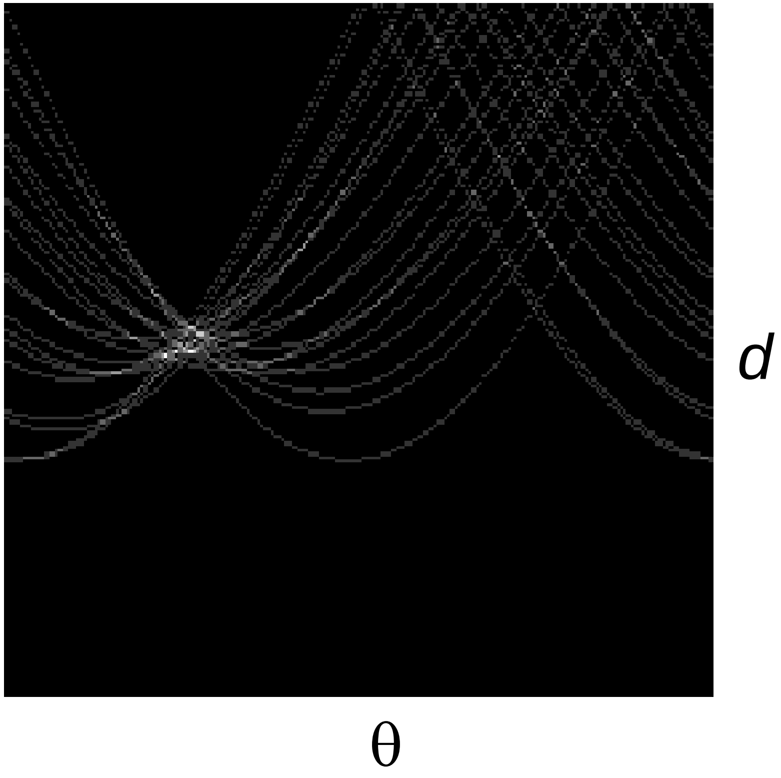


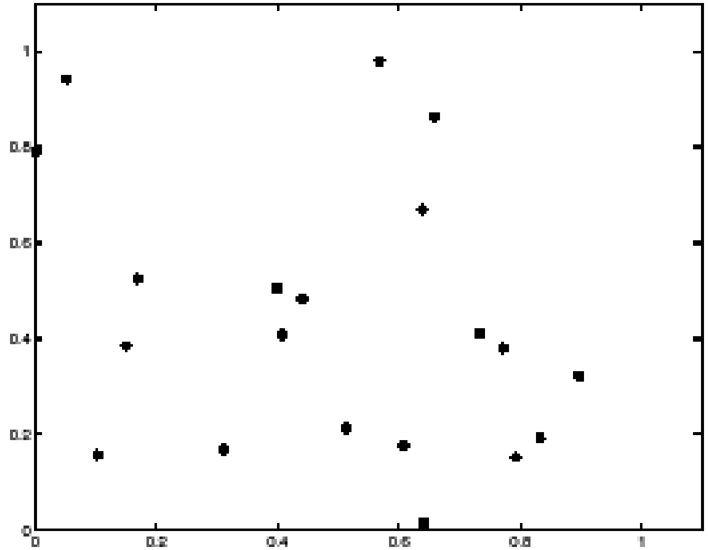
Image space
edge coordinates



Votes

What difficulty does this present for an implementation?

Impact of noise on Hough



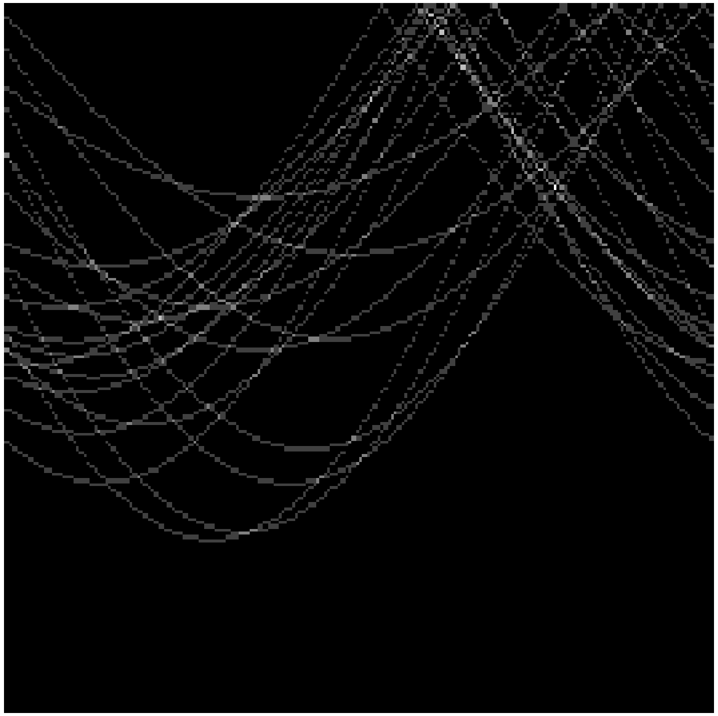


Image space edge coordinates

Votes

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

Extensions

Extension 1: Use the image gradient

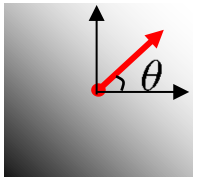
1. same
2. for each edge point I[x,y] in the image

   θ = gradient at (x,y)

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

   H[d, θ] += 1
3. same
4. same

(Reduces degrees of freedom)



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

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

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)

    $H[d, \theta] += 1$

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, 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

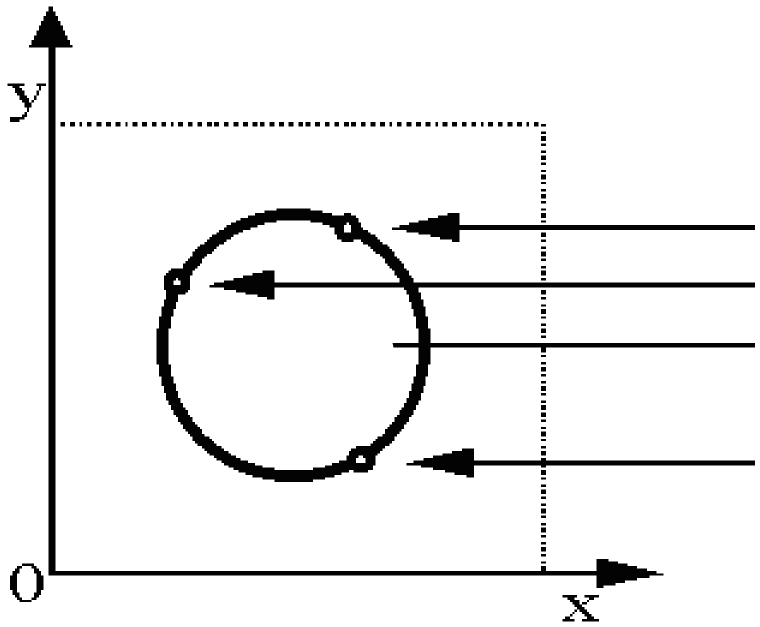
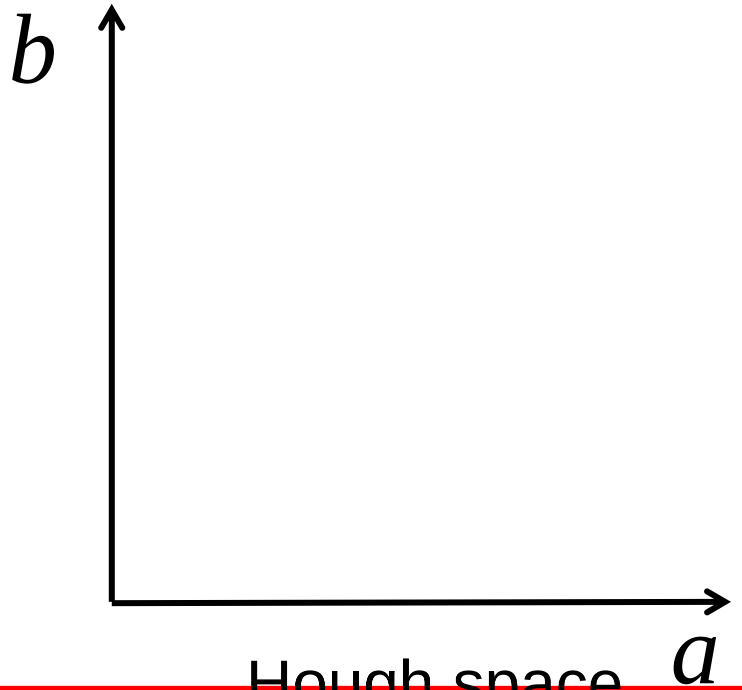


Image space



Hough space

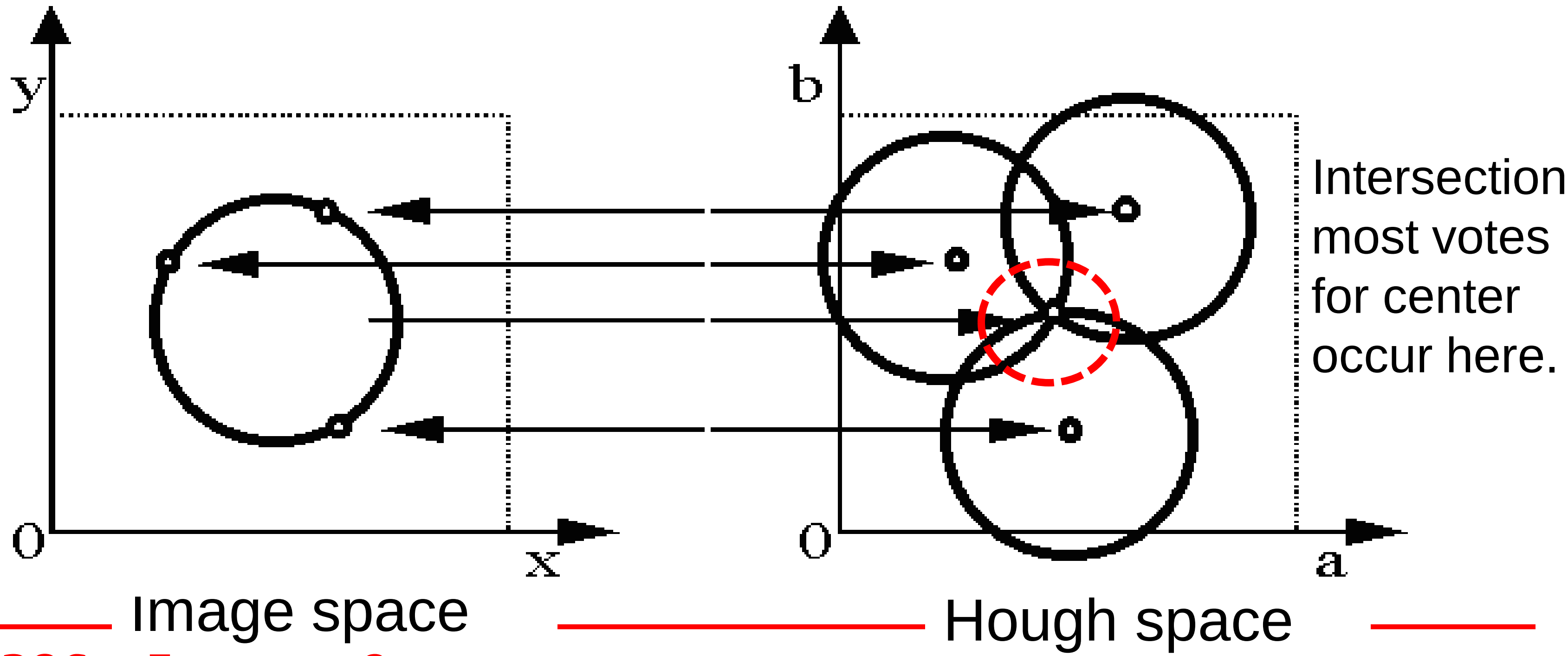
Kristen Grauman

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r, unknown gradient direction



Kristen Grauman

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, unknown gradient direction

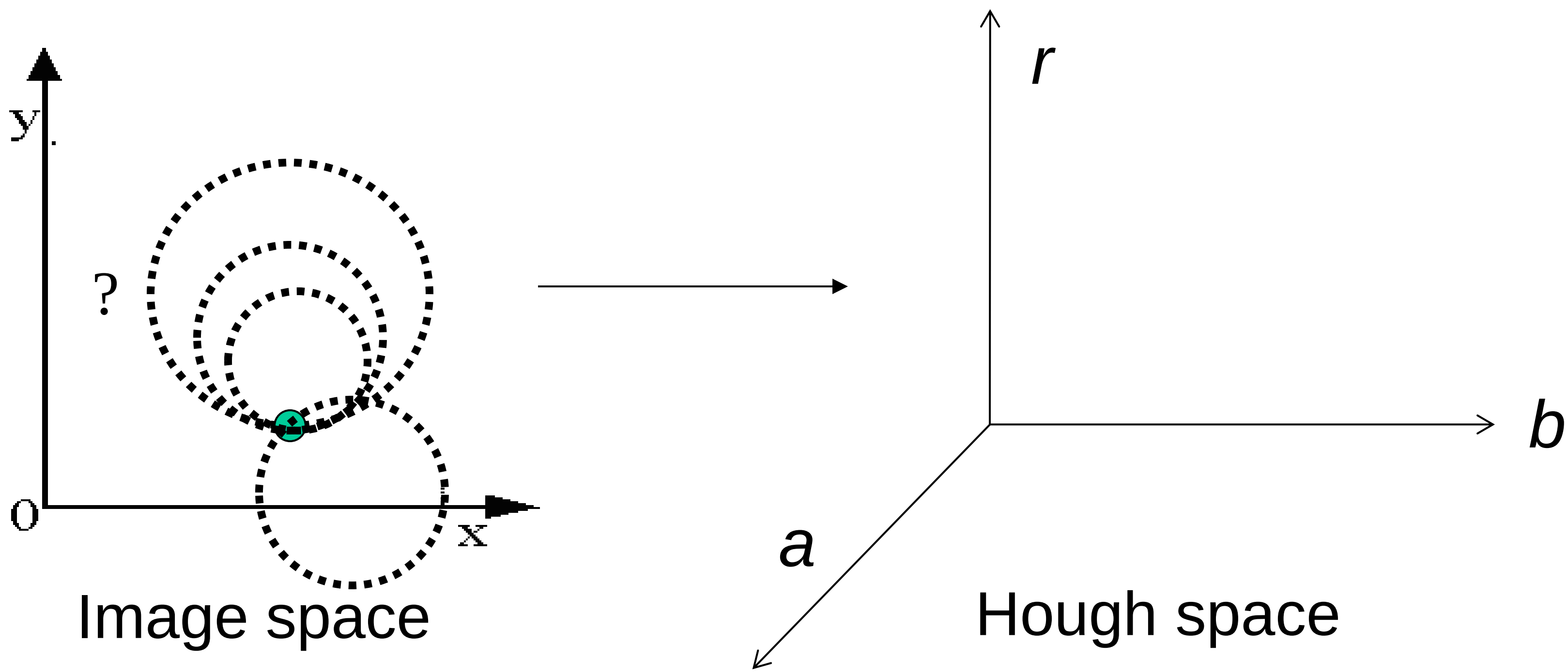


Image space

Hough space

Kristen Grauman

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, unknown gradient direction

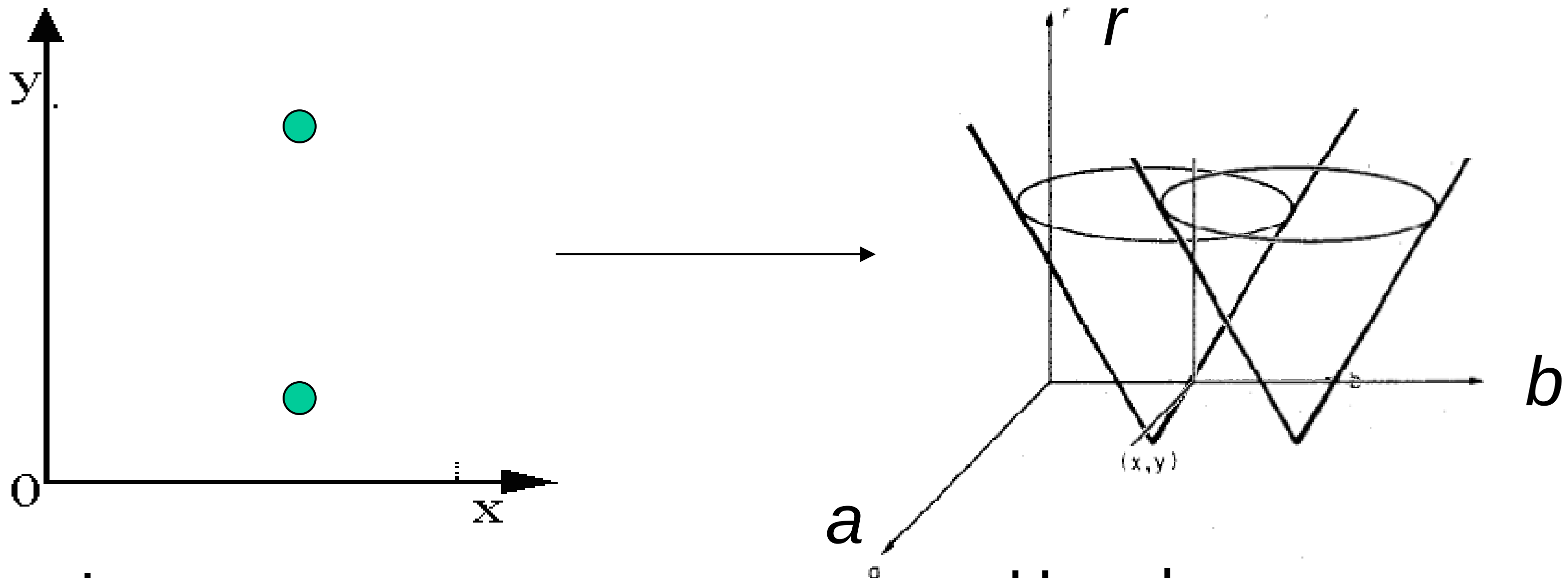


Image space

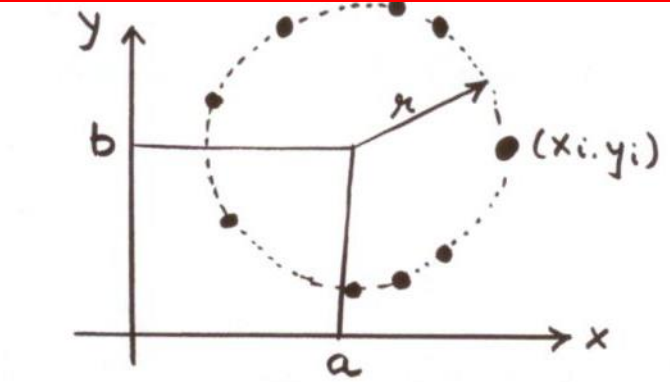
Hough space

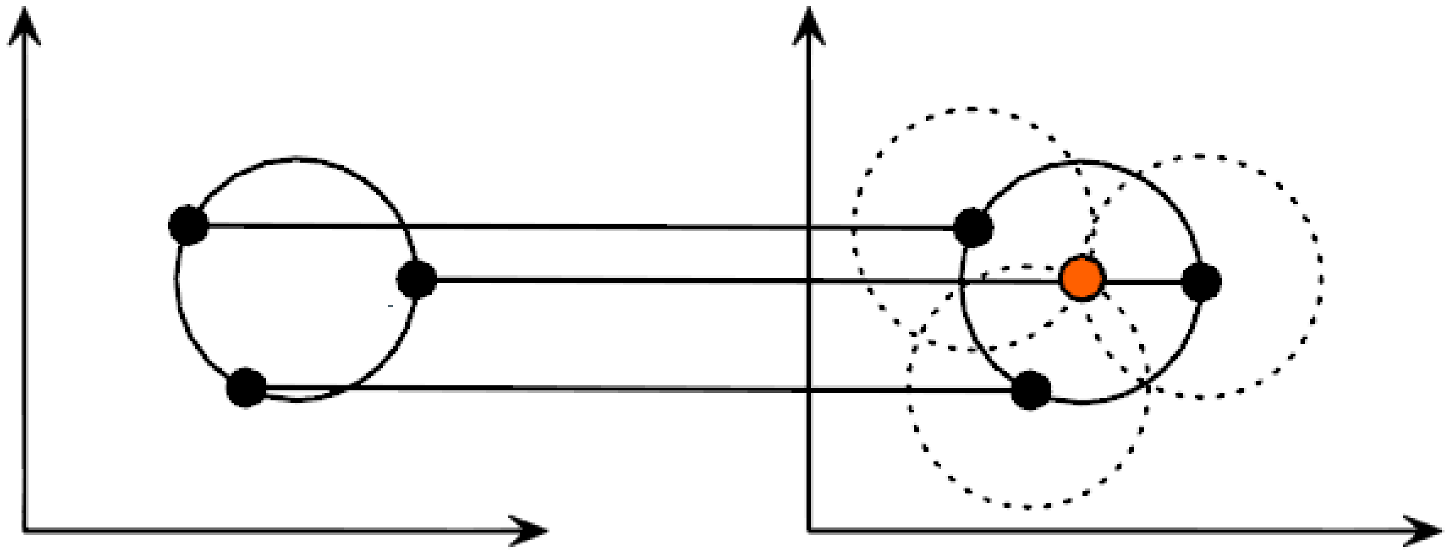
Kristen Grauman

HT for Circles: Search with fixed R

Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

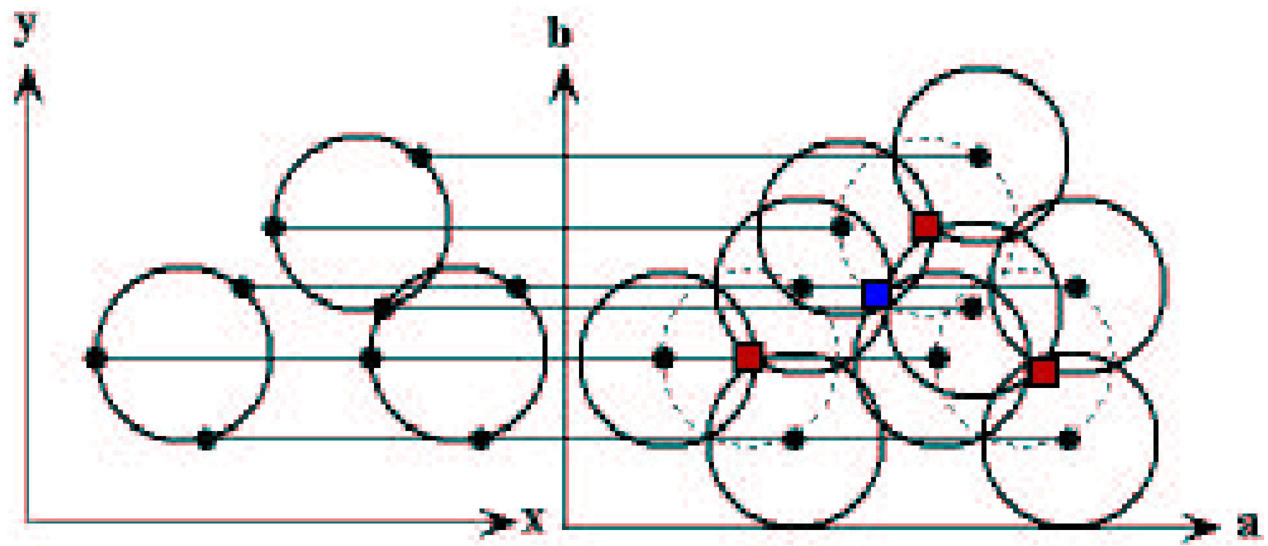




Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

Multiple Circles with known R

- Multiple circles with the same radius can be found with the same technique. The centerpoints are represented as **red cells** in the parameter space drawing.
- Overlap of circles can cause spurious centers to also be found, such as at the **blue cell**. Spurious circles can be removed by matching to circles in the original image.



Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

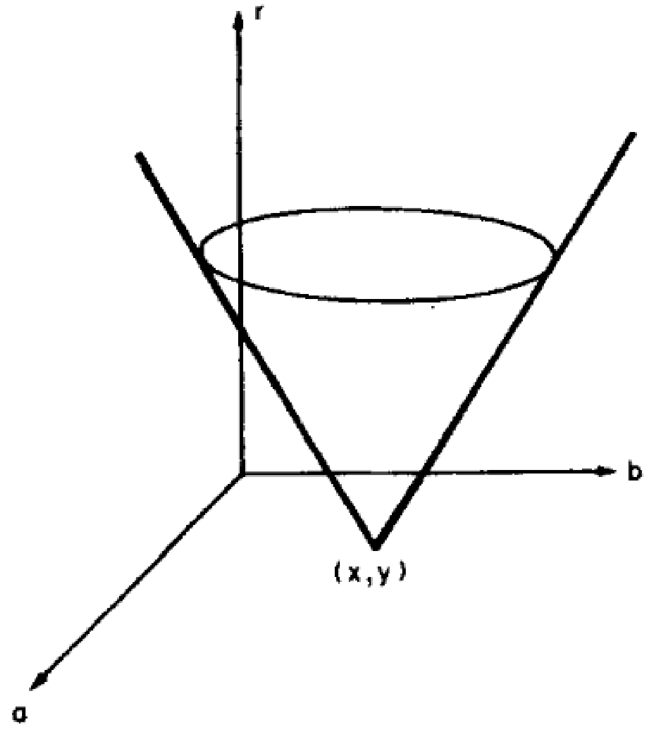
HT for Circles: Search with unknown R

Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

If radius is not known: 3D Hough Space!

Use Accumulator array $A(a, b, r)$



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

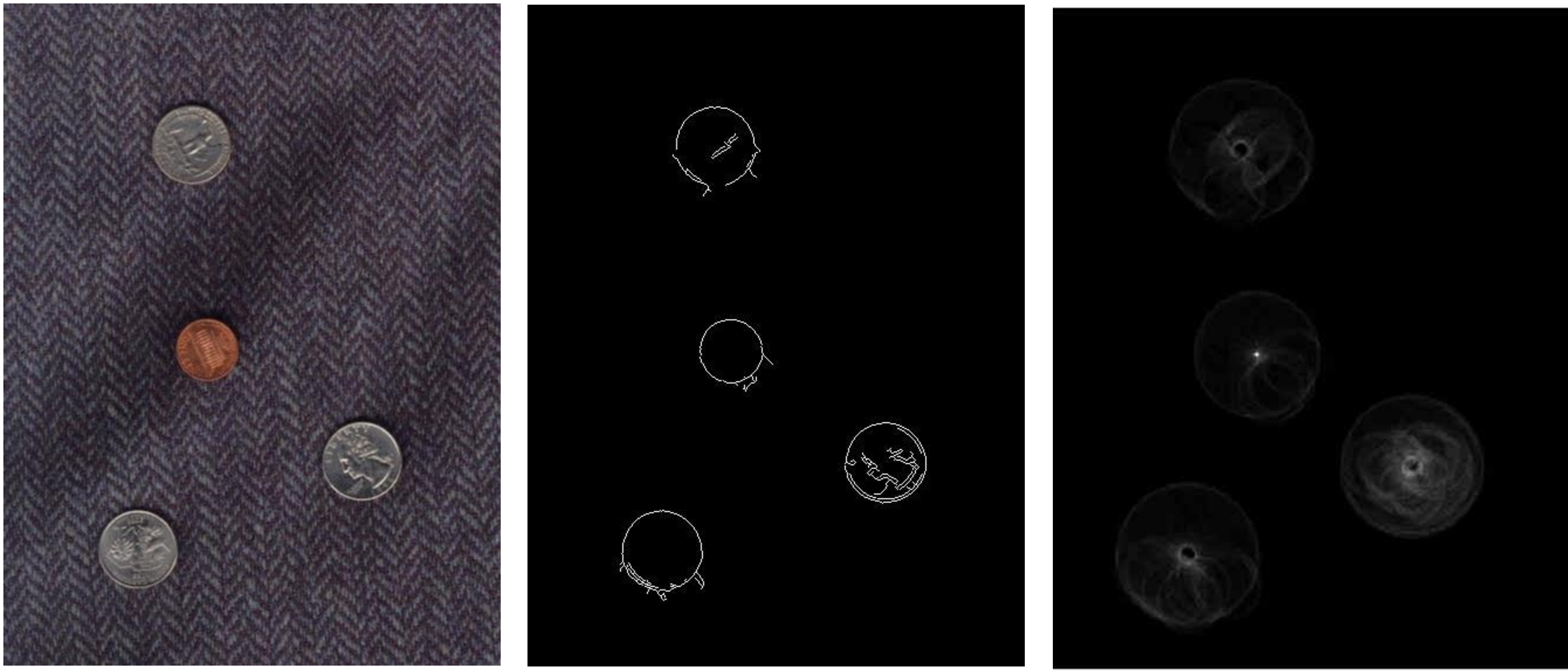
$H[a,b,r] \mathrel{+}= 1$

end

end

Example: detecting circles with Hough

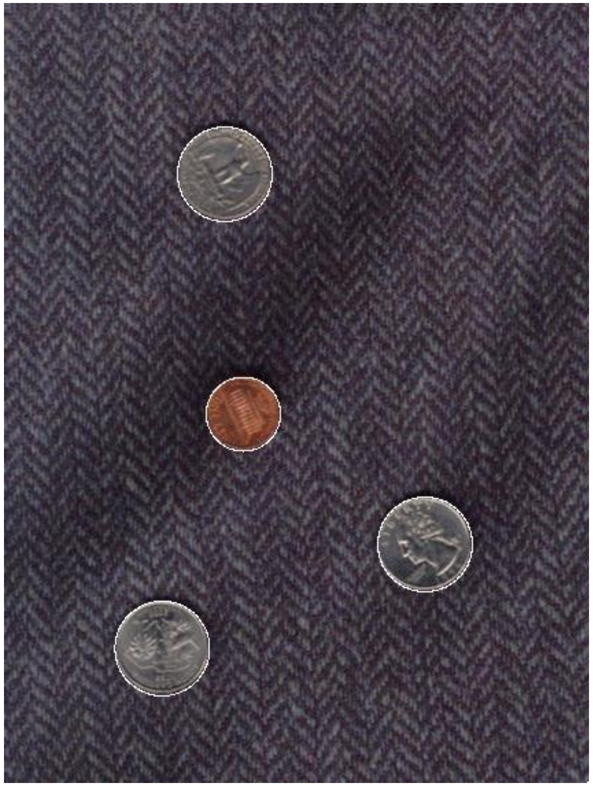
Original | Edges | Votes: Penny



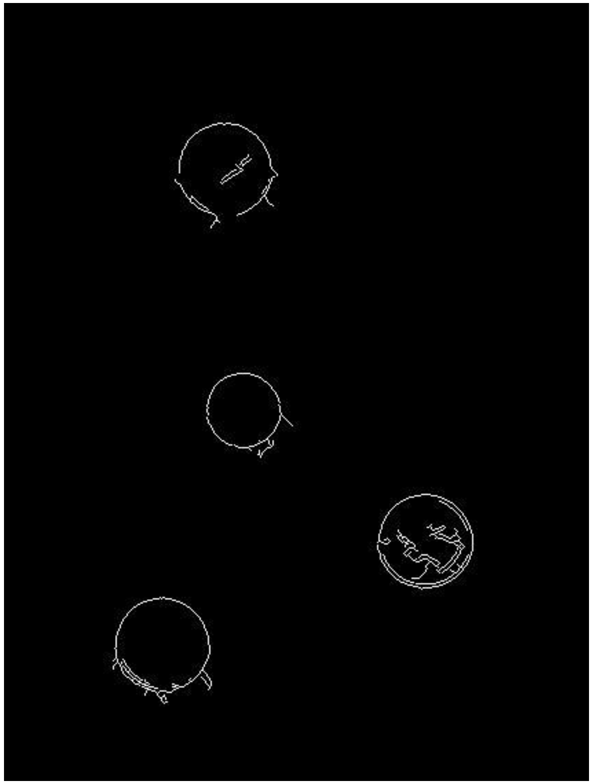
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Example: detecting circles with Hough

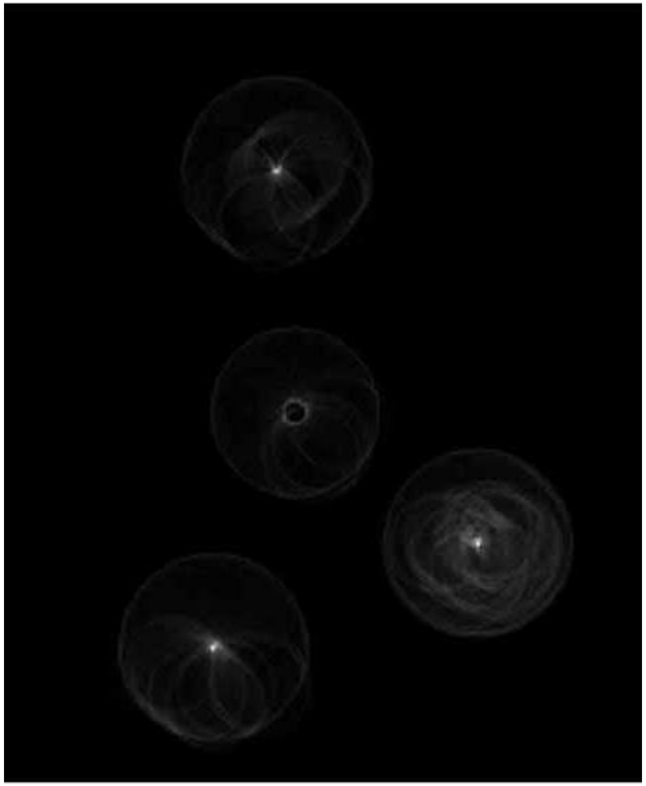
Original



Edges



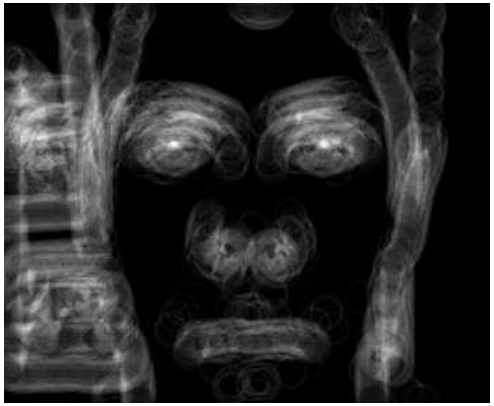
Votes: Quarter



Combined detections

Example: iris detection

Gradient+threshold



Hough space (fixed radius)

Max detections

Kristen Grauman

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.

Kristen Grauman

Hough transform: pros and cons

Pros

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

Cons

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

Kristen Grauman

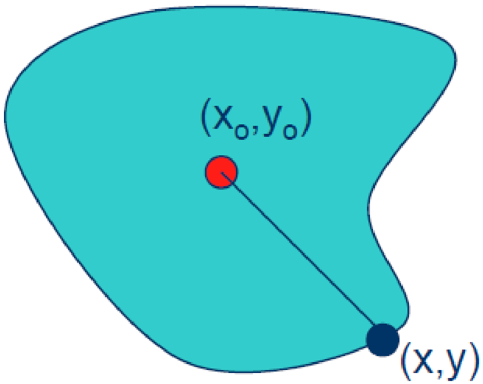
Generalized Hough Transform

- What if we want to detect arbitrary shapes?
- Detect any arbitrary shape
  - Requires specification of the exact shape of the object

- Compute centroid
- For each edge compute its distance to centroid

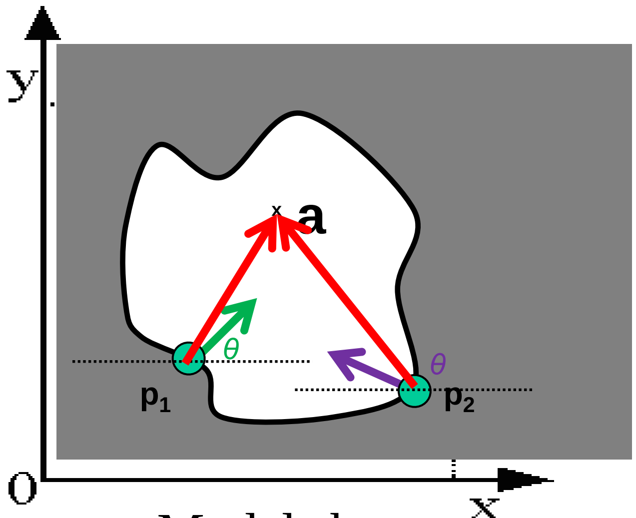
$$r = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

- Find edge orientation (gradient angle)
- Construct a table of angles and *r* values

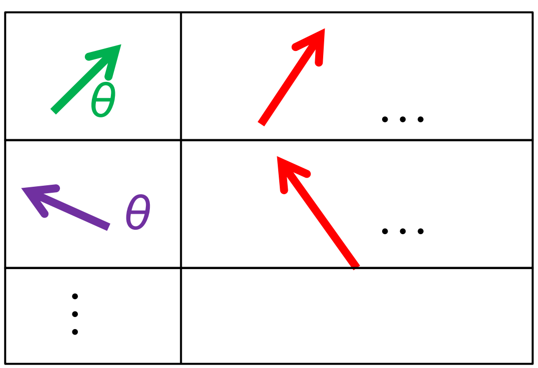


Generalized Hough Transform

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



Model shape



<u>Offline procedure:</u>

At each boundary point, compute displacement vector: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$.

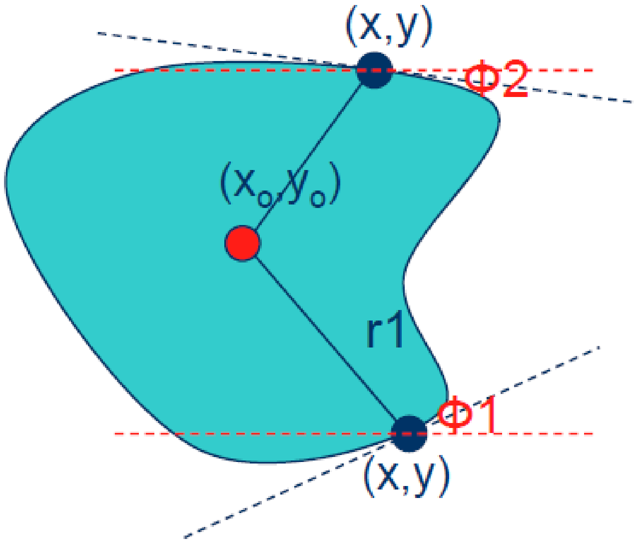
Store these vectors in a table indexed by gradient orientation θ.

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

Generalized Hough Transform

R-Table

Φ1	r1, r2, r3 …
Φ2	r14, r21, r23 …
Φ3	r41, r42, r33 …
Φ4	r10, r12, r13 …

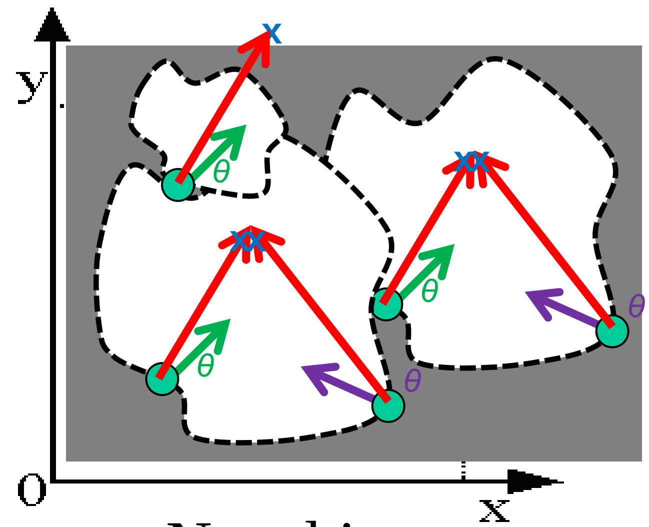


Generalized Hough Transform

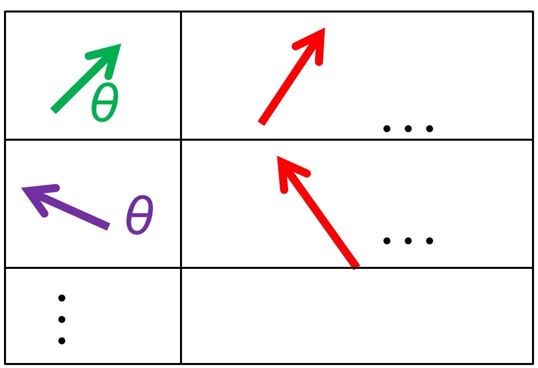
Detection procedure:

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 Transform

- known
  - Edge points (x,y)
  - Gradient angle at every edge point θ
  - R-table of the shape needs to be determined
- For each edge point find θ store it in corresponding row of R-table
- Create an accumulator array of 2D (x,y)

Generalized Hough Transform

1. Quantize the parameter space $P[x_{cmin}, \ldots\ldots, x_{cmax}, y_{cmin}, \ldots\ldots, y_{cmax}]$.

2. For each edge point (x, y) do
   compute $\phi(x, y)$
   for each table entry for ϕ do

$$x_c = x + x' \tag{4.13}$$

$$y_c = y + y' \tag{4.14}$$

   $P[x_c, y_c] = P[x_c, y_c] + 1.$

3. Find the local maxima in the parameter space.

Figure 4.8: Generalized Hough transform algorithm.

Rotation and Scale Solution

- Rotation around Z-axis

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = x\sin\alpha + y\cos\alpha$$

- Scaling

$$x' = sx$$
$$y' = sy$$

- Rotation+scaling

$$x' = s(x\cos\alpha - y\sin\alpha)$$
$$y' = s(x\sin\alpha + y\cos\alpha)$$

Rotation and Scale Solution

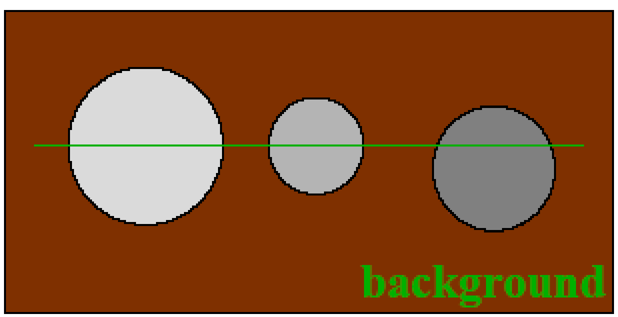
- Replace equations 4.13 and 4.14 in Algorithm 4.8 by following and loop for scale and rotation angles.

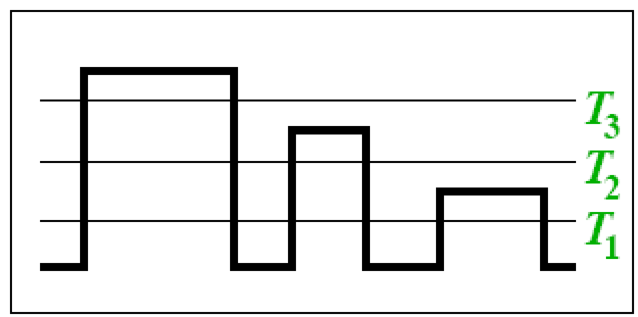
$$\begin{aligned} x_c &= x + s_x(x' \cos\theta + y' \sin\theta) \\ y_c &= y + s_y(-x' \sin\theta + y' \cos\theta) \end{aligned}$$

Segmentation of Objects Using Thresholding Method

Thresholding based Segmentation

- Goal is to identify an object based on uniform intensity
- Use the Histogram to compute the best threshold that can separate the object intensity





Thresholding Methods

1. Principles of greyvalue thresholding
2. Histogram-based thresholding
3. Methods for automatic threshold selection
   - Otsu's method
   - Histogram modelling by Gaussian distributions
4. Examples and analysis of thresholding
   - Examples of thresholding
   - Analysis of thresholding

Thresholding Principles

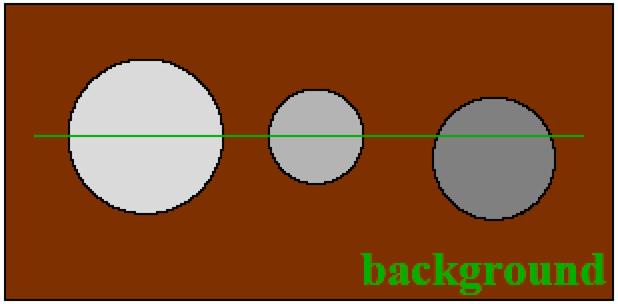
- Basic **image segmentation** technique
- Assumes following **conditions**
  - scene contains uniformly illuminated, flat surfaces
  - image is set of approximately uniform regions
- **Goal**
  - set one or more **thresholds** which split intensity range into intervals
  - ⇒ define **intensity classes**
- **Result**
  - objects labelled by classifying pixel intensities into classes
  - ⇒ objects separated from background

Thresholding Example

- Set $N - 1$ thresholds T_k, $k = 1, \ldots, N-1$, $N \geq 2$, so that pixel $f(x, y)$ is classified into class n if

$$T_{n-1} \leq f(x,y) < T_n, \quad n = 1, \ldots, N$$

- By definition, $T_0 = 0$ and $T_N = G_{max} + 1 = 256$



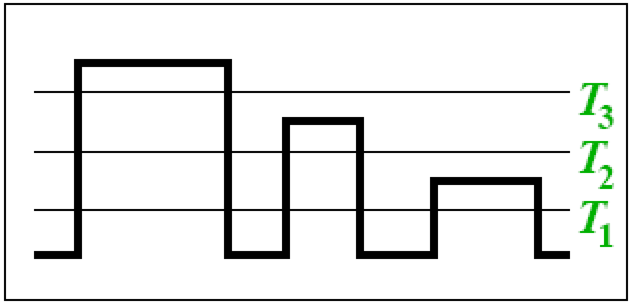
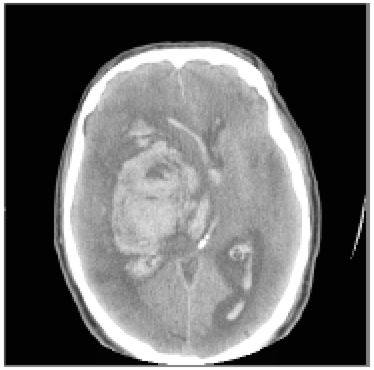
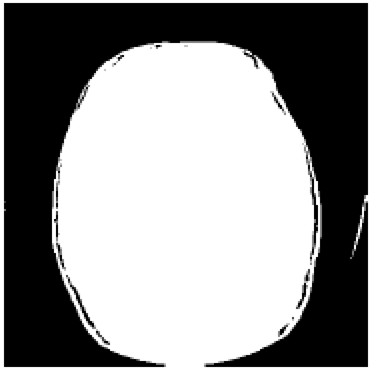


Illustration of 4-level thresholding. $T_0 = 0$ *and* $T_4 = 256$.
First level is background.

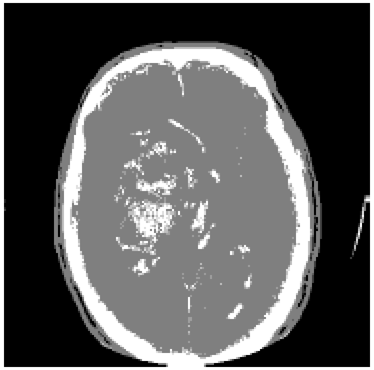
Thresholding Examples



original image



bilevel thresholding



trilevel thresholding

- Single threshold: $N = 2$
  - **bilevel** (binary) thresholding, or **binarisation**

  $\Rightarrow$ considered in this course
- **Multilevel** thresholding: $N > 2$
  - case $N = 3$ often called **trilevel**

Histogram Calculation

- Occurrence probability of greyvalue k in image

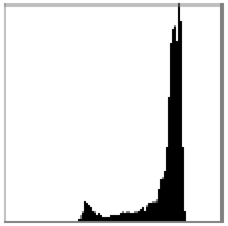
$$P(k) = \frac{n_k}{n}$$

  - n_k is number of pixels with greyvalue $k = 0, 1, \ldots, 255$
  - n is total number of pixels in image

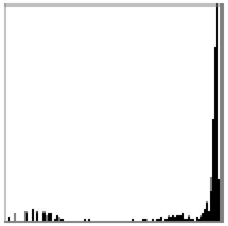
  $\Rightarrow$ $P(k)$ shows how frequently k occurs in image

- Calculation simple and fast
  - initialise $p[k] = 0$
  - scan image, for greyvalue k set $p[k] \leftarrow p[k] + 1$
  - after scan, normalise $P[k] = p[k]/n$

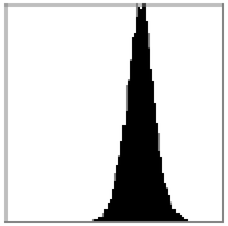
Histogram Profiles



bimodal



close to limit



unimodal

- Desirable histogram shape
  - bimodal with distinct modes and valley between modes

  ⇒ minimum of valley separates classes
- Undesirable histogram shapes
  - **mode at limit** of intensity range

  ⇒ modelling the histogram difficult
  - **mode not distinct**

  ⇒ setting good threshold not easy
  - **unimodal**

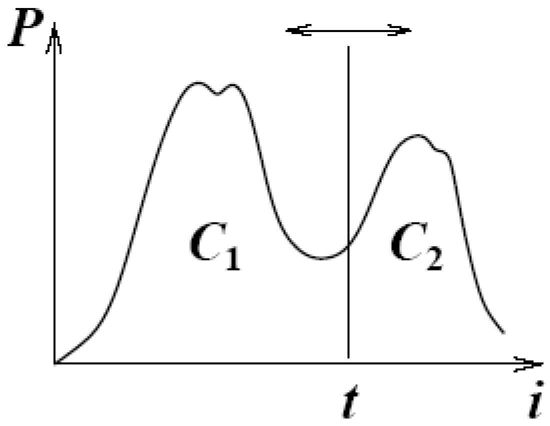
  ⇒ thresholding difficult but still possible

Good and Bad Histograms

image good 1 good 2 too low too high histogram

- Several thresholds are acceptable
  - near valley (G) in histogram
- Bad thresholds have different effects
  - too low threshold (L) tends to split lines
  - too high threshold (H) tends to merge lines

Maximum Separation



- Proposed by N.Otsu (Japan), 1978
- Consider a **candidate threshold** *t*
  - *t* defines two classes of grayvalues
- Define measure of **separation of classes**
  - distance between classes as function of *t*
- Find optimal threshold t_{opt} that **maximises separation**

Adaptive Thresholding

Mean and variance of **total** normalised histogram $P(i)$:

$$\mu = \sum_{i=0}^{G_{max}} iP(i) \qquad \sigma^2 = \sum_{i=0}^{G_{max}} (i-\mu)^2 P(i)$$

Threshold t splits $P(i)$ into **two classes** C_1, C_2 with

$$\mu_1(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} iP(i) \qquad \sigma_1^2(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} [i-\mu_1(t)]^2 P(i)$$

$$\mu_2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{G_{max}} iP(i) \qquad \sigma_2^2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{G_{max}} [i-\mu_2(t)]^2 P(i)$$

$$q_1(t) = \sum_{i=0}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{G_{max}} P(i) \qquad q_1(t) + q_2(t) = 1$$

Two Types of Variance

- Total variance σ^2 has two components
  - **within-class variance** for given t

  $\Rightarrow$ weighted sum of two class variances
  - **between-class variance** for given t

  $\Rightarrow$ distance between classes
- Within-class variance is

$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$\Rightarrow$ note that $\mu = q_1(t)\mu_1(t) + q_2(t)\mu_2(t)$

- Between-class variance is the rest of σ^2

$$\sigma_B^2(t) = \sigma^2 - \sigma_W^2(t)$$

Threshold selection via optimization

- It is easy to show that

$$\begin{aligned}\sigma_B^2(t) &= q_1(t)q_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2 \\ &= q_1(t)\left[1 - q_1(t)\right]\left[\mu_1(t) - \mu_2(t)\right]^2\end{aligned}$$

- Optimal threshold t_{opt} best separates the two classes
- $\sigma_W^2(t) + \sigma_B^2(t)$ is constant $\longrightarrow$ two equivalent options
  - *minimise* $\sigma_W^2(t)$ as *overlap of classes*
  - *maximise* $\sigma_B^2(t)$ as *distance between classes*

$\Rightarrow$ Use second option

Recursive Procedure

$$q_1(t+1) = q_1(t) + P(t+1) \quad \text{with} \;\; q_1(0) = P(0)$$

$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)} \quad \text{with} \;\; \mu_1(0) = 0 \quad (2)$$

$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}$$

Algorithm 1: Otsu threshold selection

1. Compute image histogram $P(i)$, calculate μ and σ
2. For each $0 < t < G_{max}$
   - recursively compute $q_1(t)$, $\mu_1(t)$ and $\mu_2(t)$ by eq.(2)
   - calculate $\sigma_B^2(t)$ by eq.(1)
3. Select threshold as $t_{opt} = \arg\max_t \sigma_B^2(t)$

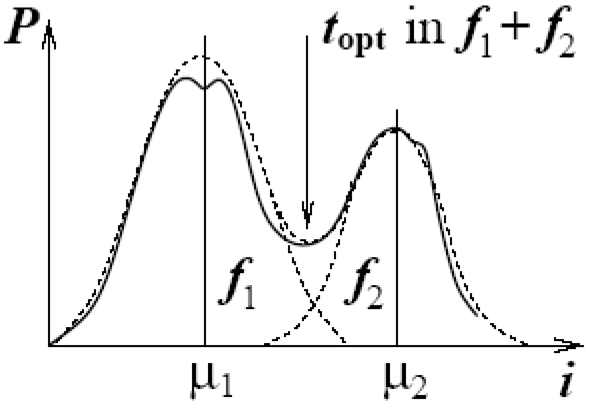
Properties

- **Advantages**
  - general: no specific histogram shape assumed
  - works well, stable
  - extension to *multilevel thresholding* possible

  $\Rightarrow$ for N thresholds and $M = G_{max} + 1$ grey levels, maximum search in array of M^N size
- **Drawbacks**
  - assumes that $\sigma_B^2(t)$ is unimodal: not always true
  - $\sigma_B^2(t)$ is often flat, false maxima may occur
  - tends to artificially enlarge small classes

  $\Rightarrow$ small classes may be merged and missed

Gaussian Mixture Modeling of Histograms



- Assume histogram $P(i)$ is mixture of **two Gaussian distributions**
- Fit this model to $P(i)$, estimate parameters of model
- Find optimal threshold **analytically** as valley in model function

Fitting Model Distribution

Model distribution is weighted sum of two Gaussians

$$f(i, \mathbf{p}) = q_1 f_1(i, \mathbf{p_1}) + q_2 f_2(i, \mathbf{p_2})$$
$$= \frac{q_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{i-\mu_1}{\sigma_1}\right)^2} + \frac{q_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{1}{2}\left(\frac{i-\mu_2}{\sigma_2}\right)^2} \quad (3)$$

- Parameter sets
  - function f: $\mathbf{p} = (q_1, q_2, \mu_1, \mu_2, \sigma_1, \sigma_2)$
  - functions $f_k, k = 1, 2$: $\mathbf{p_k} = (q_k, \mu_k, \sigma_k)$
- Weights q_1 and q_2 of partial distributions
  - $q_1 + q_2 = 1$

  $\Rightarrow$ **five free parameters** (degrees of freedom, dof)

  $\Rightarrow$ exclude q_2, denote $\mathbf{p}' = (q_1, \mu_1, \mu_2, \sigma_1, \sigma_2)$

Fitting Model Distribution - 2

- Fitting error function

$$C(\mathbf{p}') = \sum_{i=0}^{G_{max}} \left[f(i, \mathbf{p}') - P(i) \right]^2 \qquad (4)$$

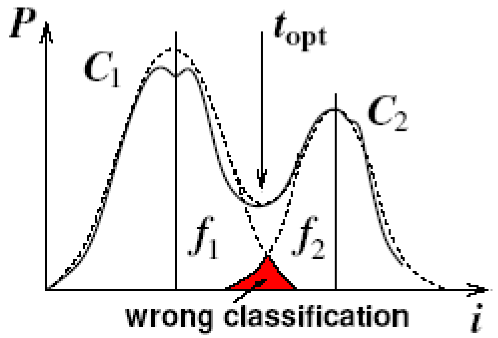
- To fit $f(i, \mathbf{p}')$ to $P(i)$, minimise $C(\mathbf{p}')$
  - $\Rightarrow$ estimate optimal parameters $\hat{\mathbf{p}}$
- Nonlinear minimisation with five variables
- A nonlinear minimisation algorithm can be used
  - $\Rightarrow$ Newton
  - $\Rightarrow$ Marquard-Levenberg
  - $\Rightarrow$ stochastic
- Iterative minimisation algorithms can *fail* to give any result
  - $\Rightarrow$ no solution for fitting, no threshold

Derivation of Optimal Threshold

- Assume model fitting has been done
  - $\Rightarrow$ optimal parameters obtained: $(\hat{q}_1, \hat{q}_2, \hat{\mu}_1, \hat{\mu}_2, \hat{\sigma}_1, \hat{\sigma}_2)$
- Now, hogy to calculate optimal threshold?

Minimise probability of wrong classification

$$E(t) = E_1(t) + E_2(t) = \int_{-\infty}^{t} f_2(i)di + \int_{t}^{\infty} f_1(i)di$$



- $E_1(t)$: pixel from C_1 classified as C_2
- $E_2(t)$: pixel from C_2 classified as C_1

Derivation of Optimal Threshold - 2

- Set $E'(t) = 0$, substitute f_1 and f_2 from eq.(3)

$\Rightarrow$ **Optimal threshold** t_{opt} is solution of

$$At^2 + Bt + C = 0, \qquad (5)$$

where

$$\begin{aligned} A &= \hat{\sigma}_1^2 - \hat{\sigma}_2^2 \\ B &= 2(\hat{\mu}_1\hat{\sigma}_2^2 - \hat{\mu}_2\hat{\sigma}_1^2) \\ C &= \hat{\sigma}_1^2\hat{\mu}_2^2 - \hat{\sigma}_2^2\hat{\mu}_1^2 + 2\hat{\sigma}_1^2\hat{\sigma}_2^2 \ln\left(\frac{\hat{\sigma}_2\hat{q}_1}{\hat{\sigma}_1\hat{q}_2}\right) \end{aligned}$$

Cases for Optimal Threshold

- If eq. (5) has **two real roots** $\in [0, 255]$
  - $\Rightarrow$ select root for which error $E(t)$ is smaller
- If eq. (5) has **no real root** $\in [0, 255]$
  - $\Rightarrow$ no optimal threshold available
- If $\sigma_1^2 = \sigma_2^2 = \sigma^2$
  - $\Rightarrow$ single optimal threshold exists

$$t_{opt} = \frac{\hat{\mu}_1 + \hat{\mu}_2}{2} + \frac{\hat{\sigma}^2}{\hat{\mu}_1 - \hat{\mu}_2} \ln\left(\frac{\hat{q}_1}{\hat{q}_2}\right)$$

Algorithm for Gaussian Threshold Detection

Algorithm 2: Gaussian threshold selection

1. Calculate normalised histogram $P(i)$
2. Minimise fitting error function $C(\mathbf{p}')$ defined by (4) and (3)
   $\Rightarrow$ obtain optimal parameter estimates $\hat{q}_1, \hat{q}_2, \hat{\mu}_1, \hat{\mu}_2, \hat{\sigma}_1, \hat{\sigma}_2$
3. Solve equation (5) for t, obtain two roots
4. Discard imaginary roots and real roots $\notin [0, 255]$
   - if single root t_s remains, set $t_{opt} = t_s$
   - if two roots remain, select root with smaller $E(t)$

Properties of Gaussian Mixture Approach

- **Advantages**
  - reasonably general histogram model
  - when model is valid, minimises classification error probability
  - may work for small-size classes
- **Drawbacks**
  - many histograms are not Gaussian mixtures

  ⇒ greyvalues are **finite** and **non-negative**

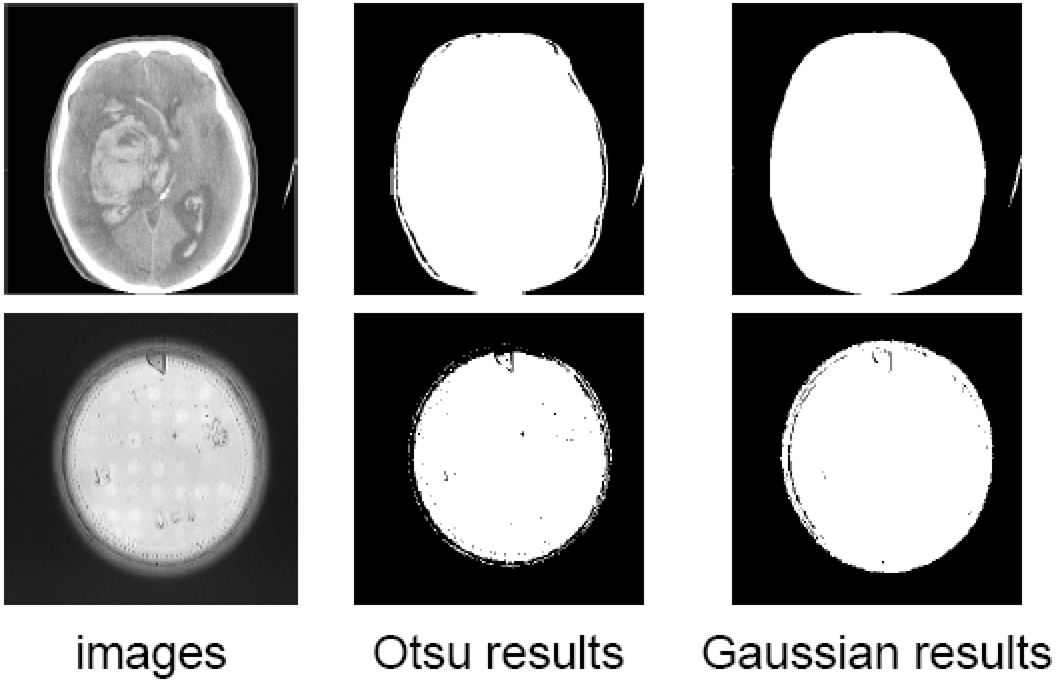
  ⇒ peak close to intenisity limit do not fit Gaussian

  - extension to multithresholding practically impossible

  ⇒ needs irrealistic simplification of model

  - difficult to detect near and flat modes of histogram

Examples



- Gaussian algorithm sets lower thresholds in both cases
  $\Rightarrow$ fits object contours better than Otsu

Otsu vs Gaussian Approach

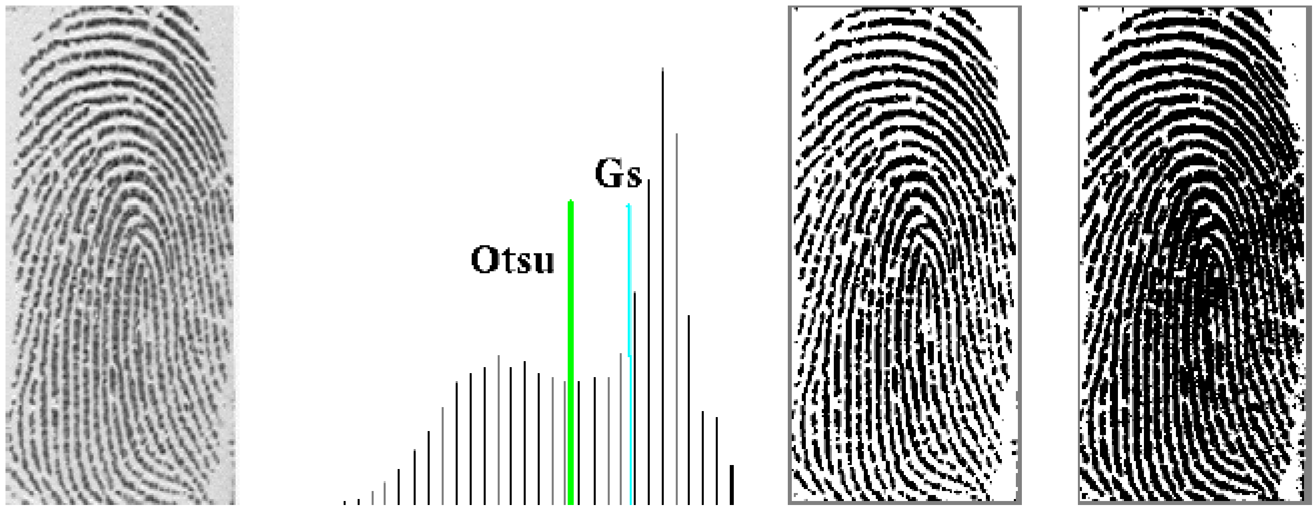


image histogram Otsu Gauss

- **Otsu** algorithm sets threshold $T = 158$ in valley
  - $\Rightarrow$ lines are well-separated
- **Gaussian** algorithm sets slightly high threshold $T = 199$
  - $\Rightarrow$ some lines touch

Gaussian Gives Poor Results

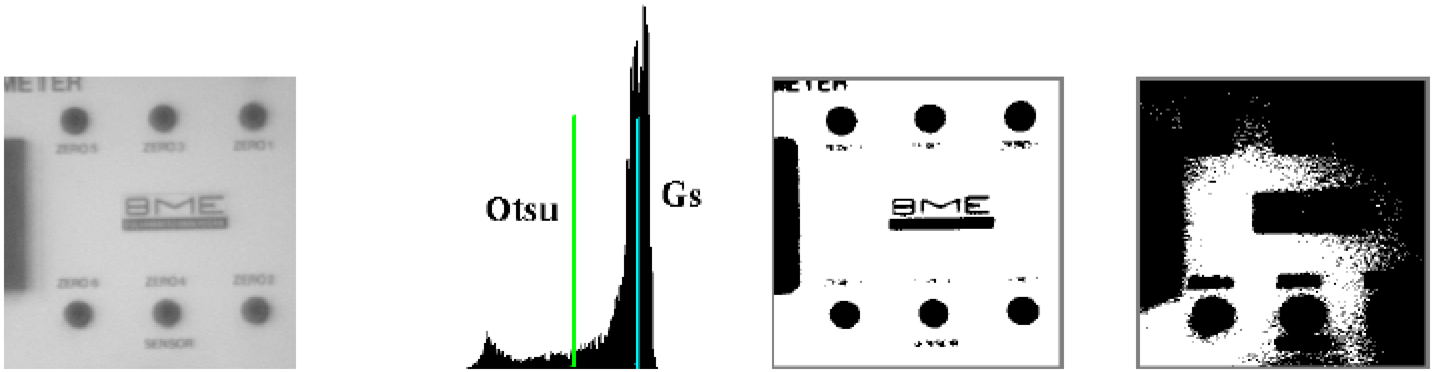
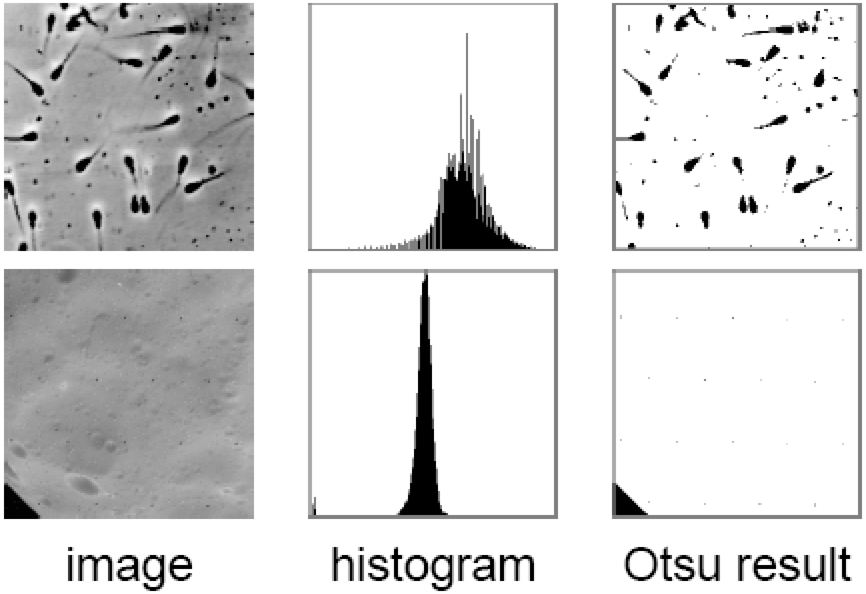


image | histogram | Otsu $T = 159$ | Gauss $T = 201$

- **Otsu algorithm** finds small class of pixels (dark discs)
- **Gaussian algorithm** tries to separate two high peaks formed by background

$\Rightarrow$ Selects noisy valley because true class is
  - too small
  - too far away

Gaussian Mixture – a Fail Case



- Only Otsu algorithm produces results
- Gaussian algorithm gives **no results** at all
  - upper row: unimodal histogram, model fitting failed
  - lower row: fitting done, threshold equation has no real root

Issues with Thresholding

- Histogram based thresholding is very effective
- Even with low noise, if one class is much smaller than the other we might still be in trouble.
- Remember also that both these images have the same histogram: