Hough Transform

Dr. Elli Angelopoulou

Pattern Recognition Lab (Computer Science 5)

University of Erlangen-Nuremberg

Features

- We have seen so far different types of features that can be extracted directly from images:
  - Edges
  - Textons
  - Color
- All of these features are local.
- Textons and color can be used for more general scene analysis (e.g. segmentation, identification of type of reflectance).
- Edges typically convey geometric information.
- How do we generalize from edge-pixels to more abstract geometric shapes (lines, circles etc.)?

On Detecting Lines, Circles, …

- Goal: Detect a particular line L or curve C.
- Input:
  - The output image E of an edge detector (e.g. Sobel, Canny etc.) when applied on an image I.
  - The parameters that define a particular line L (curve C)
- Output:
  - The locations of all instances of the given line L (curve C) or parts of it in I.

Method 1: Template Matching

- Create a template (mask, mini-image) of an exemplary line (curve) that we are looking for.
- Convolve the image with that template, so that an exact match will give a high response.
- A separate template is needed for each orientation.
- Challenge: Creating an exemplary mask that is accurate, yet general enough.
- Advantage: Idea generalizes to arbitrary shapes (has been used in face detection).

Method 2: Hough Transform

- General method that can be applied for a variety of shapes.
- Idea: We may not have the analytic equation of the shape we are trying to detect, but we can measure a ***distinct characteristic/property*** of the shape.
- For example, a line has a *slope* and an *intercept*.
- If the shape has this characteristic/property, then it is highly probable that it is the correct shape.

Method 2: Hough Transform - continued

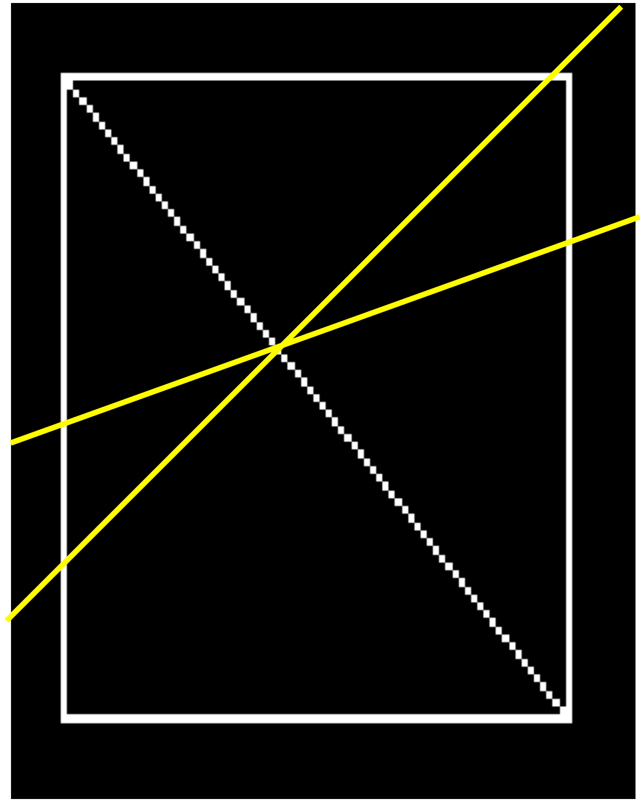
- Idea: We focus on a ***distinct characteristic/ property*** of the shape.
- Measure that characteristic/property at every pixel.
- If a pixel has that property, then it is a shape candidate.
- Count how many times this property occurred. If the count is high, then the object is detected.

Attributes of Hough Transform (HT)

- HT is an elegant method that maps a possibly difficult pattern detection problem in to a simple "peak" detection.
- One can think of HT as a fancy name for a voting scheme, since edges *vote* for the possible model.
- Advantages:
  - Edges need not be connected
  - The object (line, circle) may be only partially visible.
- It is a traditional way to detect lines and circles.

Key Idea of HT for Line Detection

- Each straight line in this image can be described by an equation.
- Each white pixel, if considered in isolation, could lie on an infinite number of straight lines.
- In the HT each pixel votes for every line it could be on.
- The line(s) with the most votes wins.



Step 1: Measure property

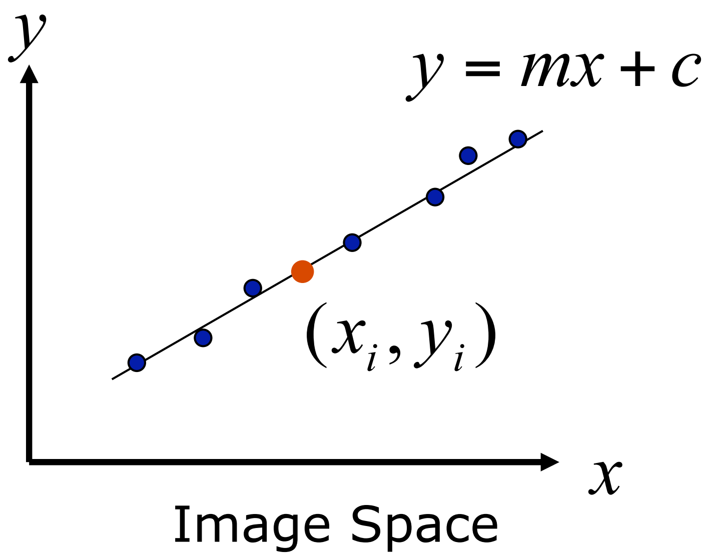
Equation of a line: $y = mx + c$

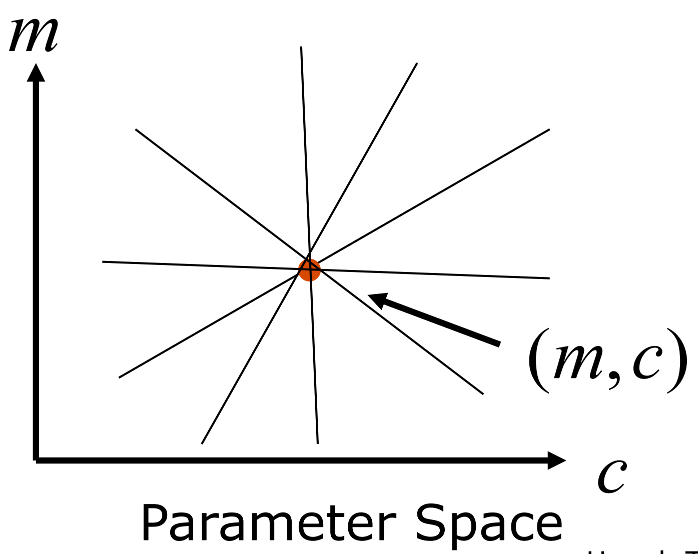
A line is uniquely identified by the parameter pair (m, c)

A line in image space I is represented by a single point in the parameter space (m, c)-space.

Consider a point: (x_i, y_i)
It can be seen as a point of either line $y_i = mx_i + c$
in image space
or a line $c = -x_i m + y_i$
in parameter space

The Parameter Space is also called the **Hough Space.**





Step 2: Count

Algorithm:

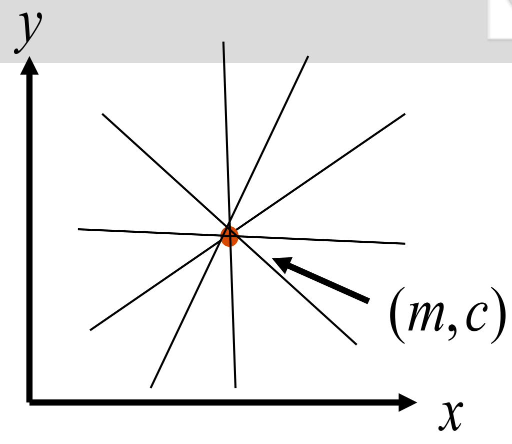
- Quantize Parameter Space (m, c)
- Create Accumulator Array $A(m, c)$
- Set $A(m, c) = 0 \quad \forall m, c$
- For each image edge (x_i, y_i)
  - For each *m*
    - Compute *c*, based on

$$c = -x_i m + y_i$$

    - Increment count

$$A(m, c) = A(m, c) + 1$$

- Find local maxima in $A(m, c)$



Parameter Space $A(m, c)$

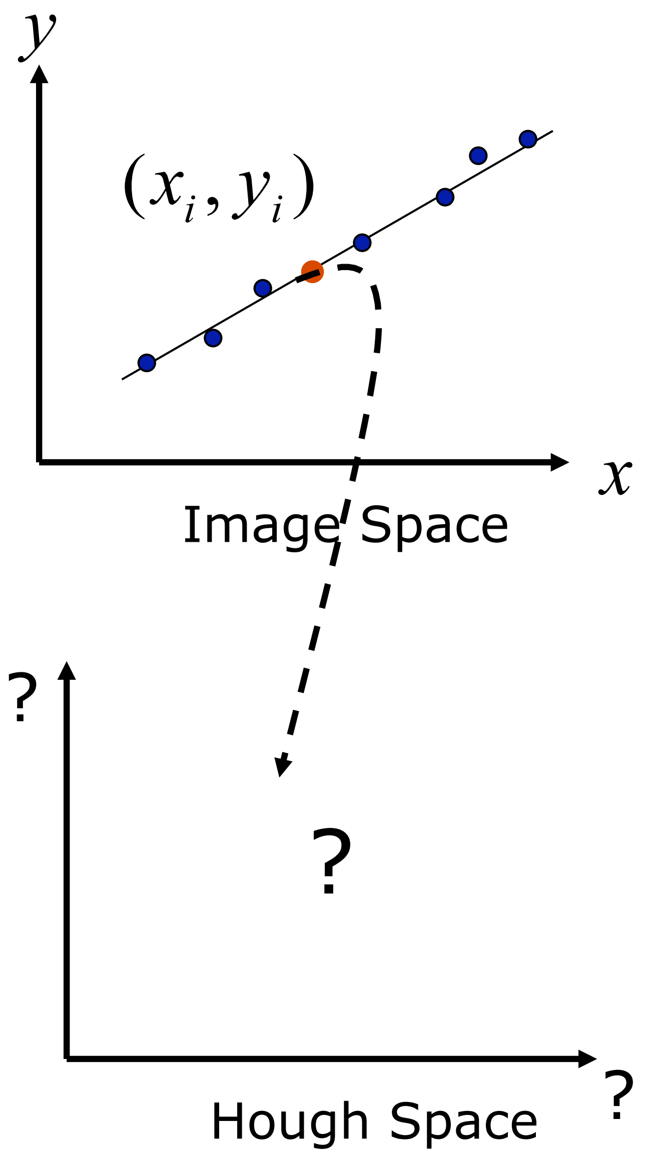
	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Remarks

- Due to quantization and noise, we may have missed some of the edge points that truly belong to the line.
- Still the cell $A(m,c)$ has the highest count.
- Problems:
  - We have a finite Accumulator Array size,
  - But an infinite range of slopes

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

  - How do we represent vertical lines?
- Can we find a different representation of the line?



Foot of the Normal Representation

- Use another form of the equation of the line:

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

- The parameter space is bounded.

$$0 \le \theta \le 2\pi$$

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

- Given a point (x_i, y_i) compute (ρ, θ)

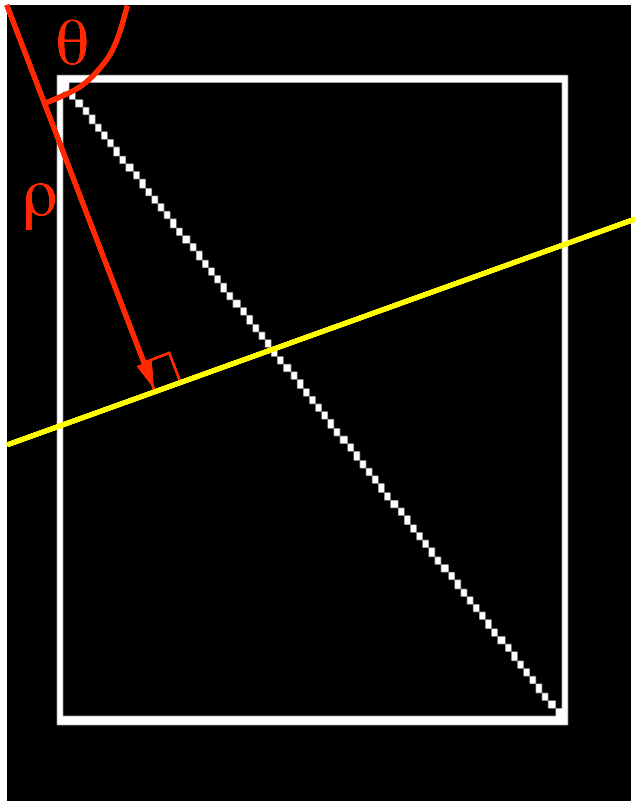


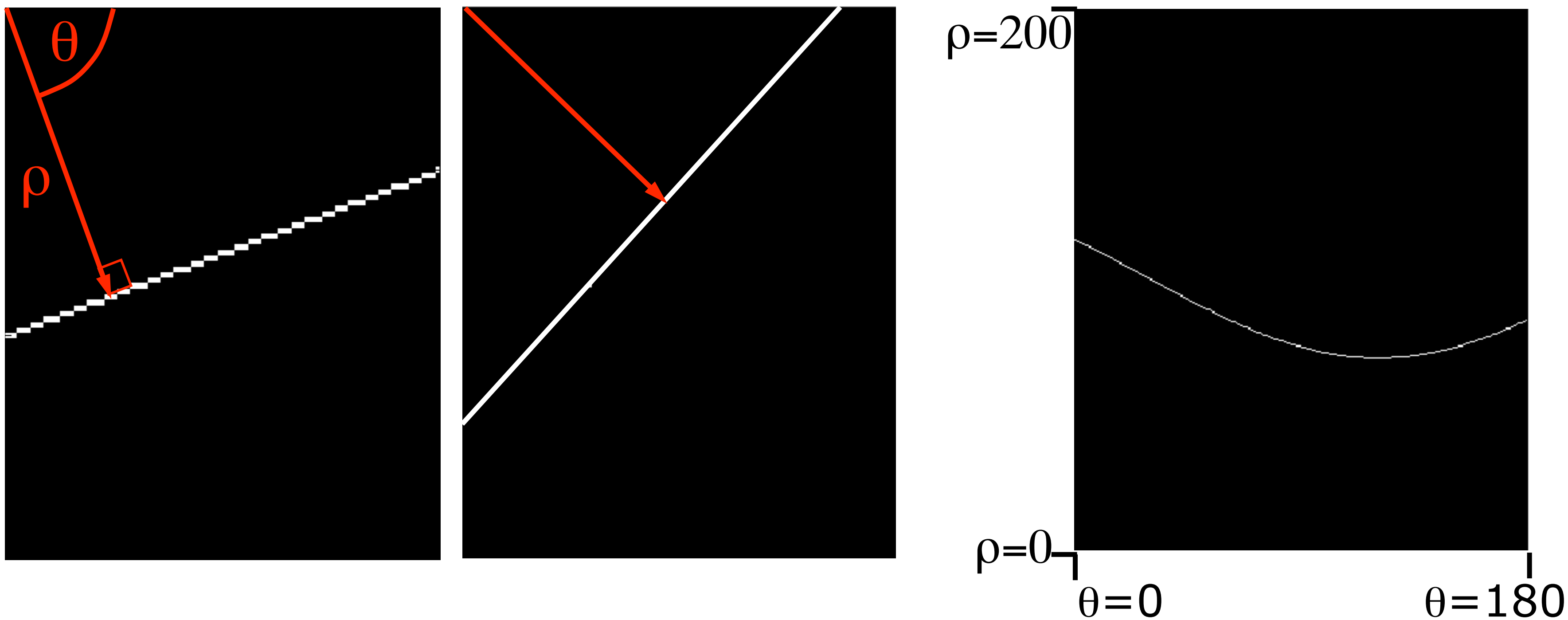
Image Space to Hough Space

- Given an edge point (x_i, y_i)

  for all $0 \leq \theta \leq 2\pi$

  compute $\rho = x_i \cos(\vartheta) + y_i \sin(\vartheta)$

- A point in image space corresponds to a sinusoidal curve in Hough space.



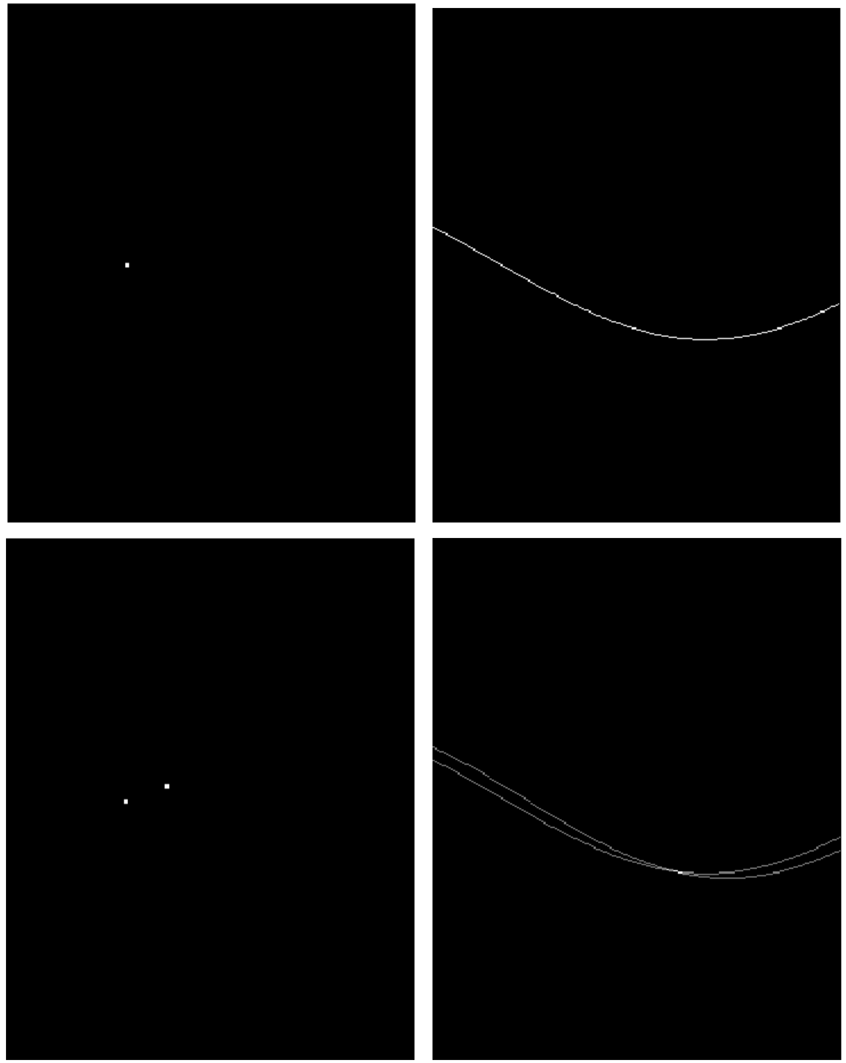
HT Voting - Points

One point in image space corresponds to a sinusoidal curve in image space

Two points correspond to two curves in Hough space

The intersection of those two curves has “two votes”.

This intersection represents the straight line in image space that passes through both points



HT Voting - Line

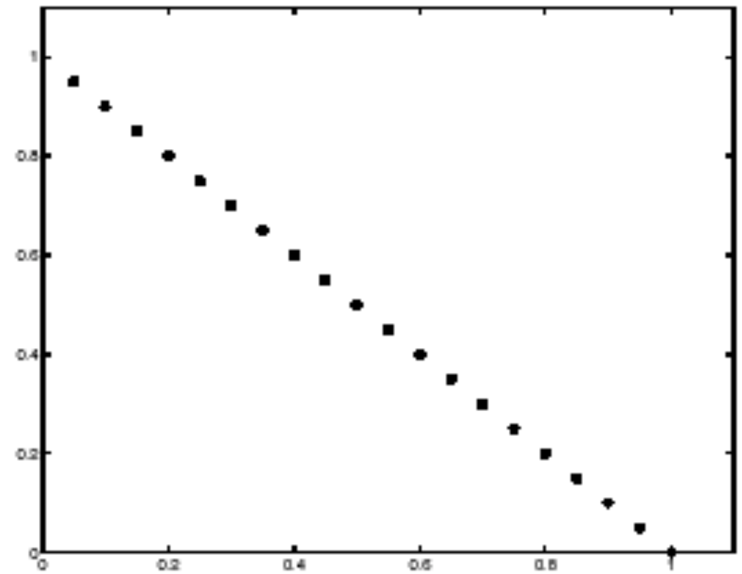
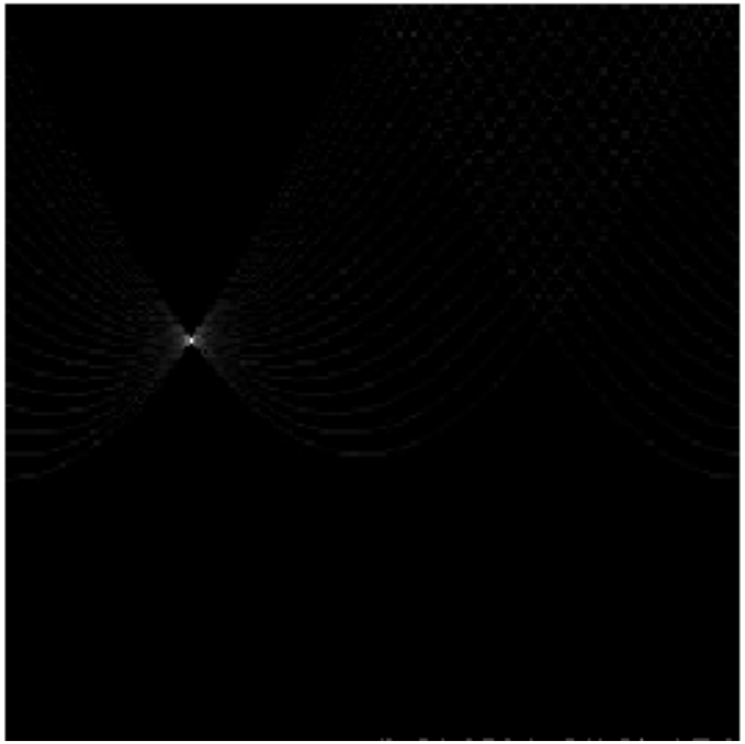


Image space



Votes

HT Voting – Noisy Line

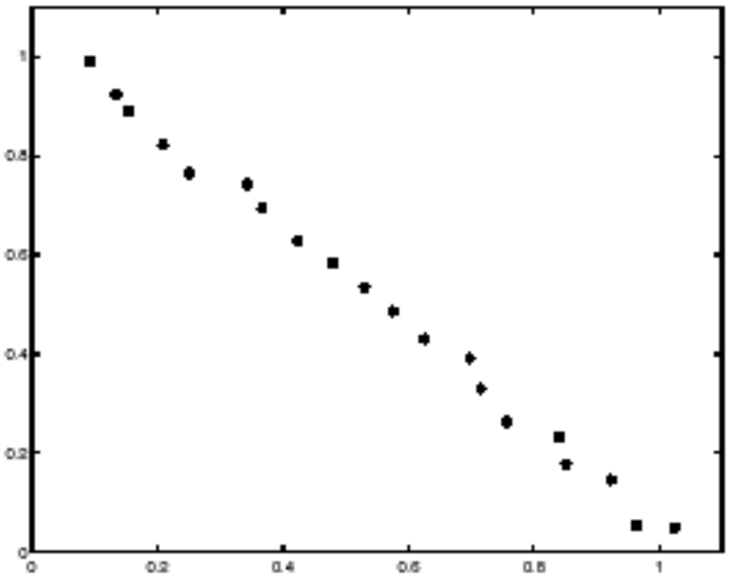
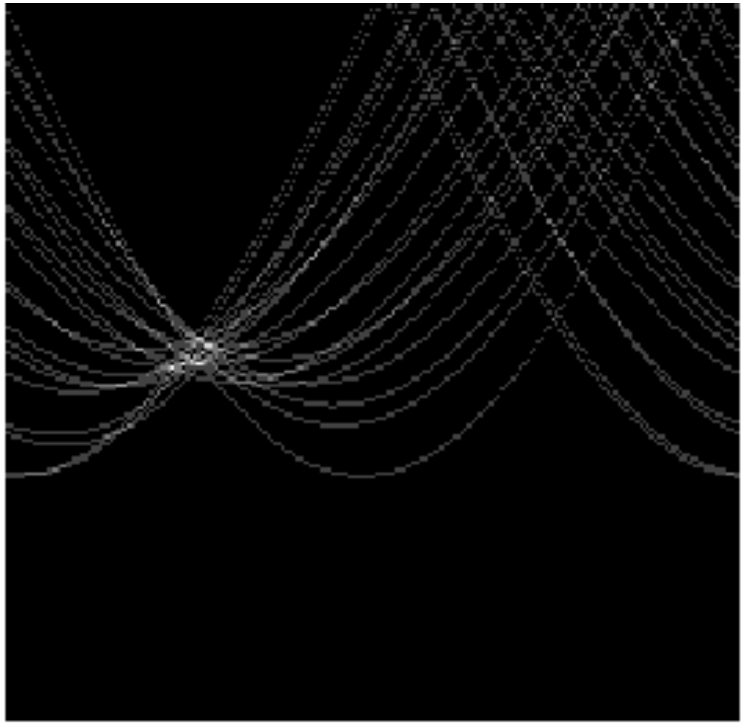


Image space



Votes

HT Voting – No Line

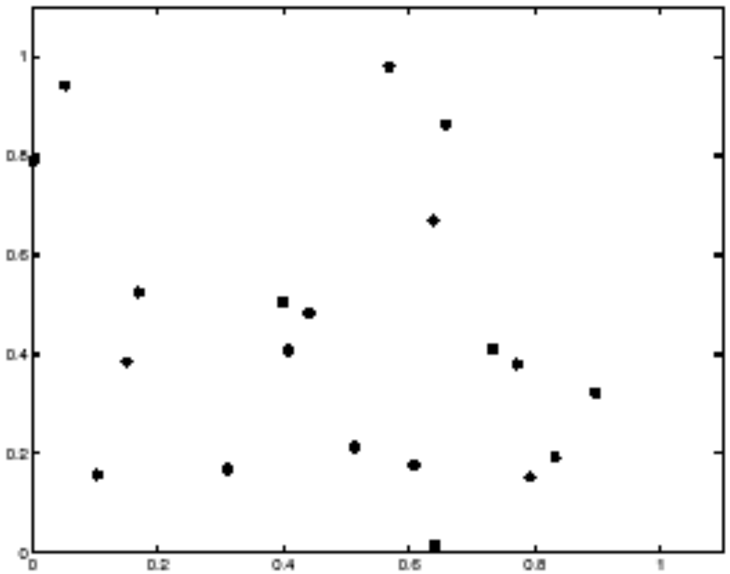
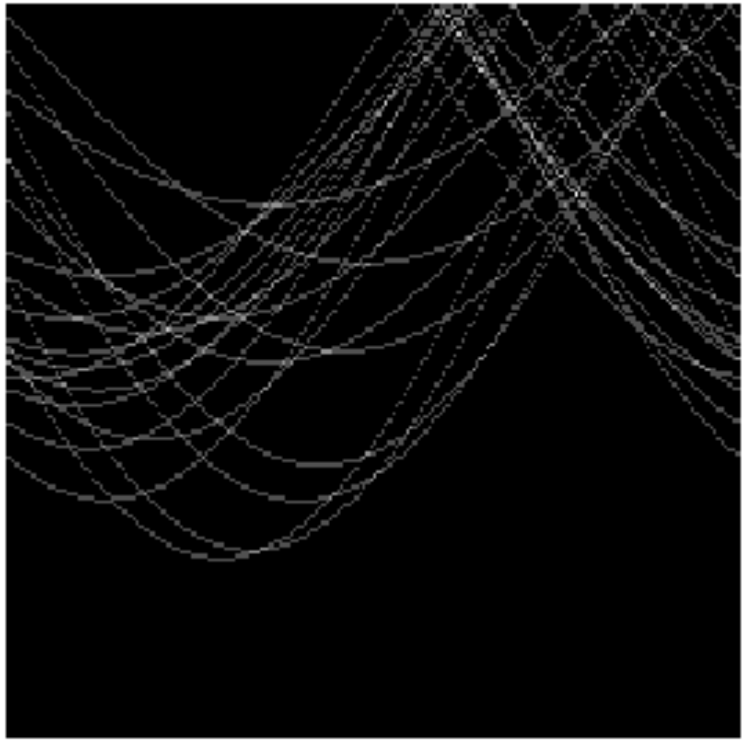
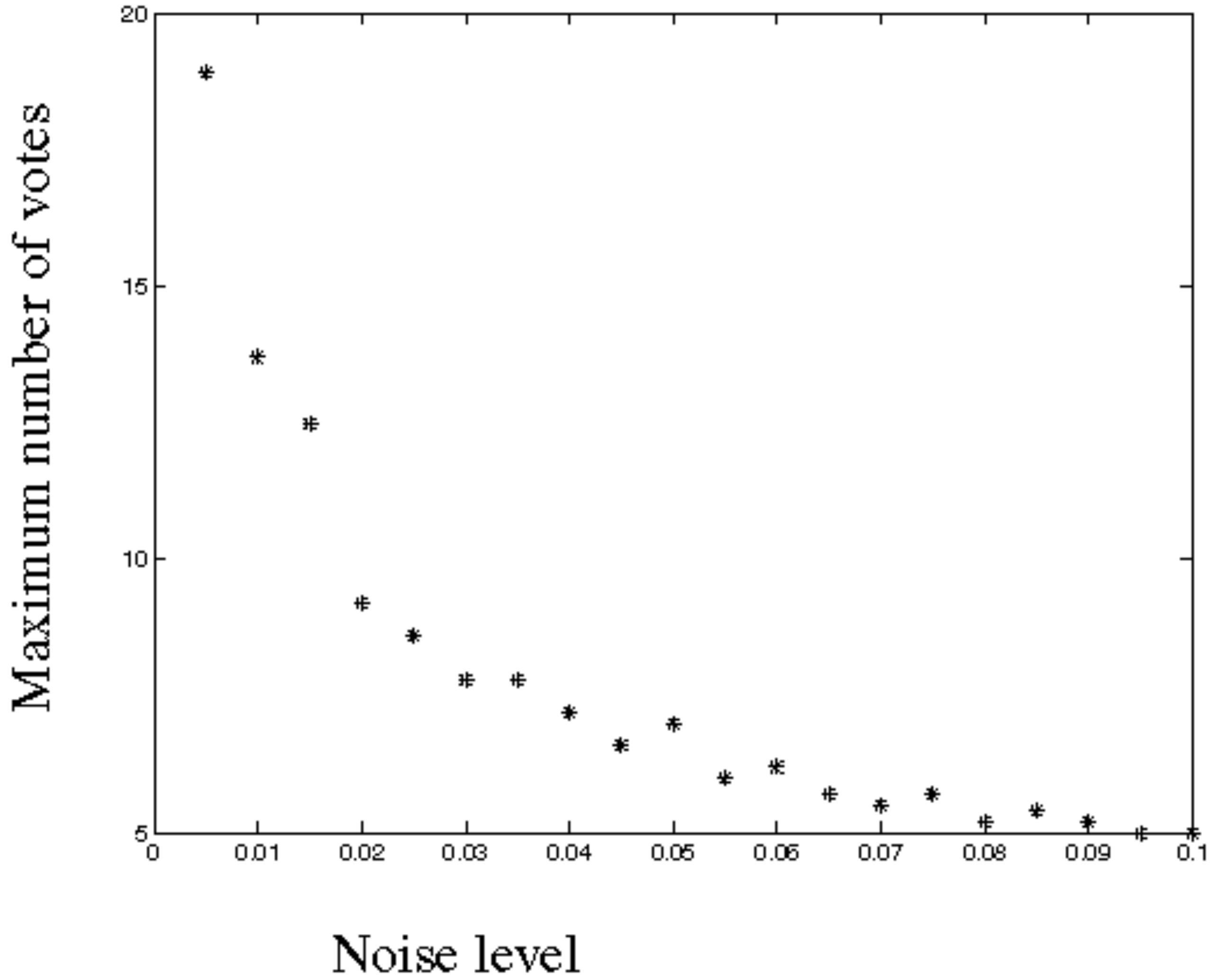


Image space



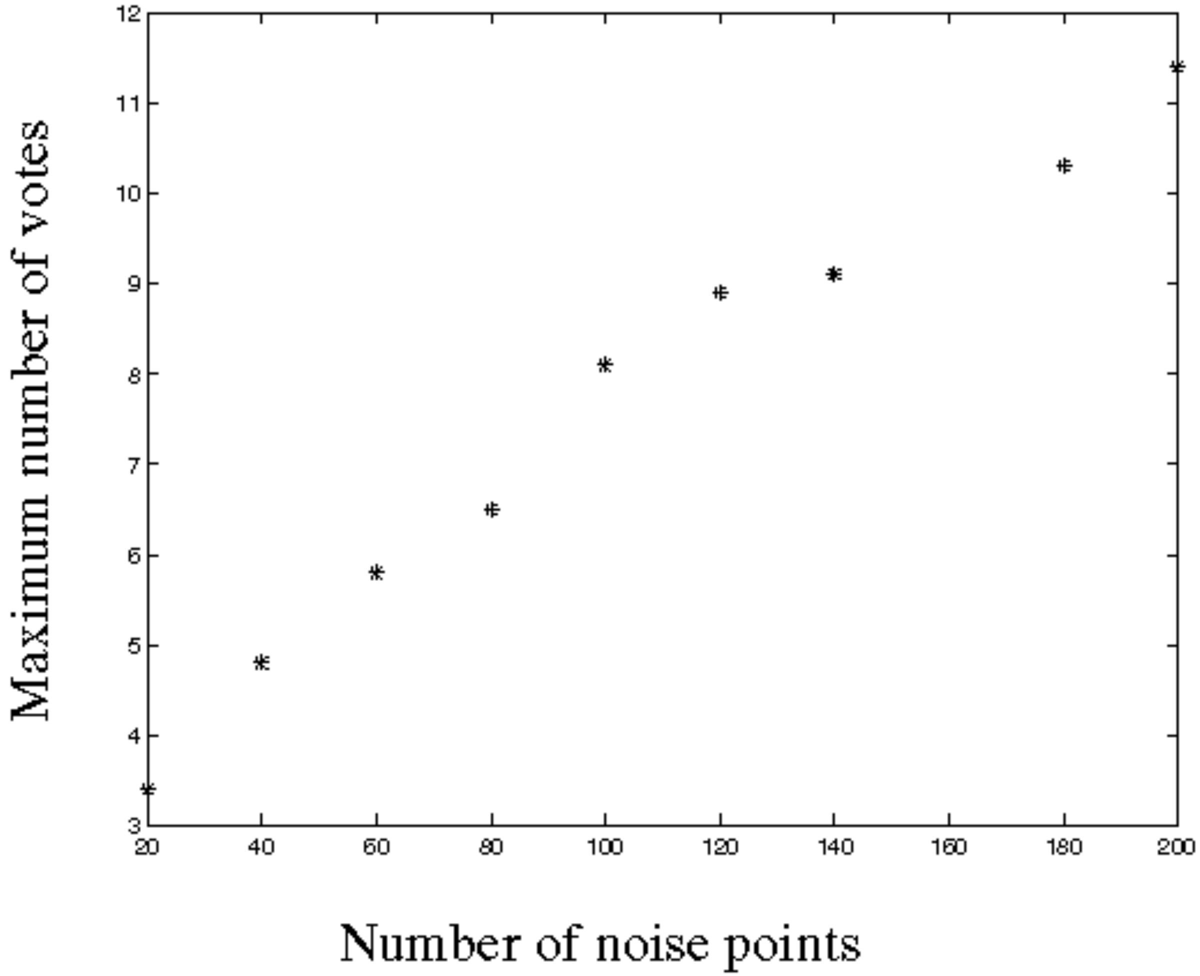
Votes

HT Voting and Noise



Fewer votes land in a single bin when noise increases.

HT Voting and Noise



Adding more clutter increases number of bins with false peaks.

Remarks on HT for Line Detection

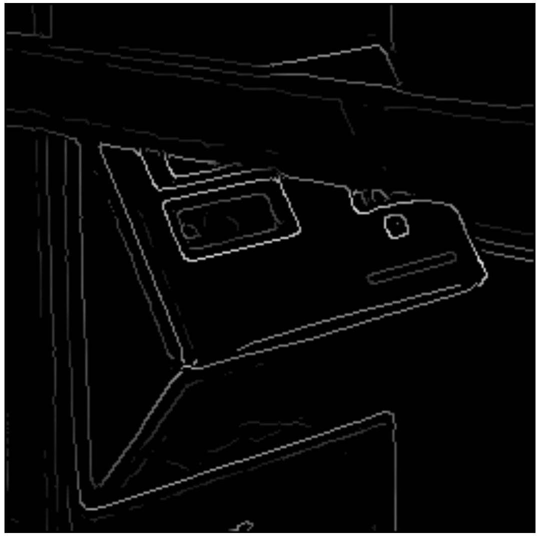
- How do we know how many lines we have in an image?
  - As many as the number of peaks.
  - Merge adjacent peaks together.
  - Use a threshold for peak detection.

- Which points belong to the line?
  - Search for points close to the peak of that line in Hough space.
  - Search for points close to the line in image space.
  - Solve for the specific line and iterate.

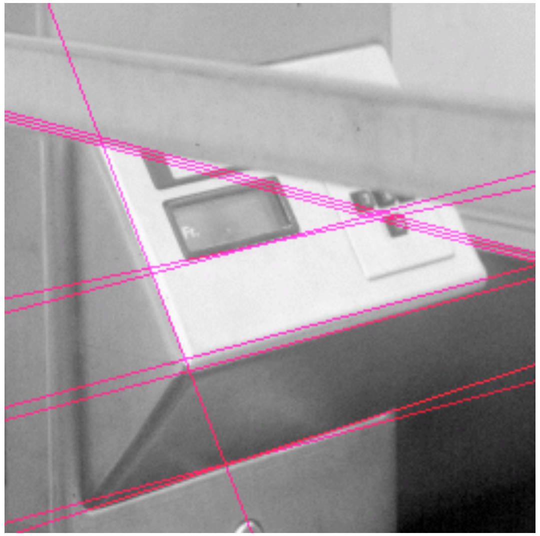
- Important issue: Size (number of) accumulator cells.
  - Too small => we may miss lines, especially in the presence of noise.
  - Too big => we may merge distinct lines together.

Real World Example

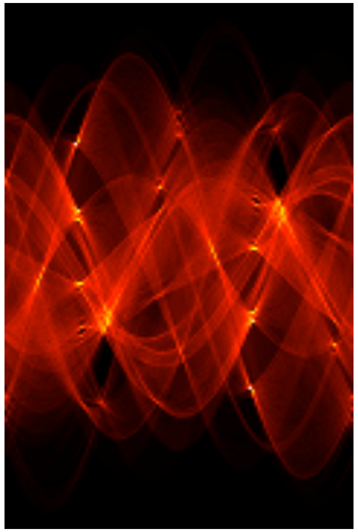
Original



Edge Detection



Detected Lines



Hough Space

HT for Circle Detection

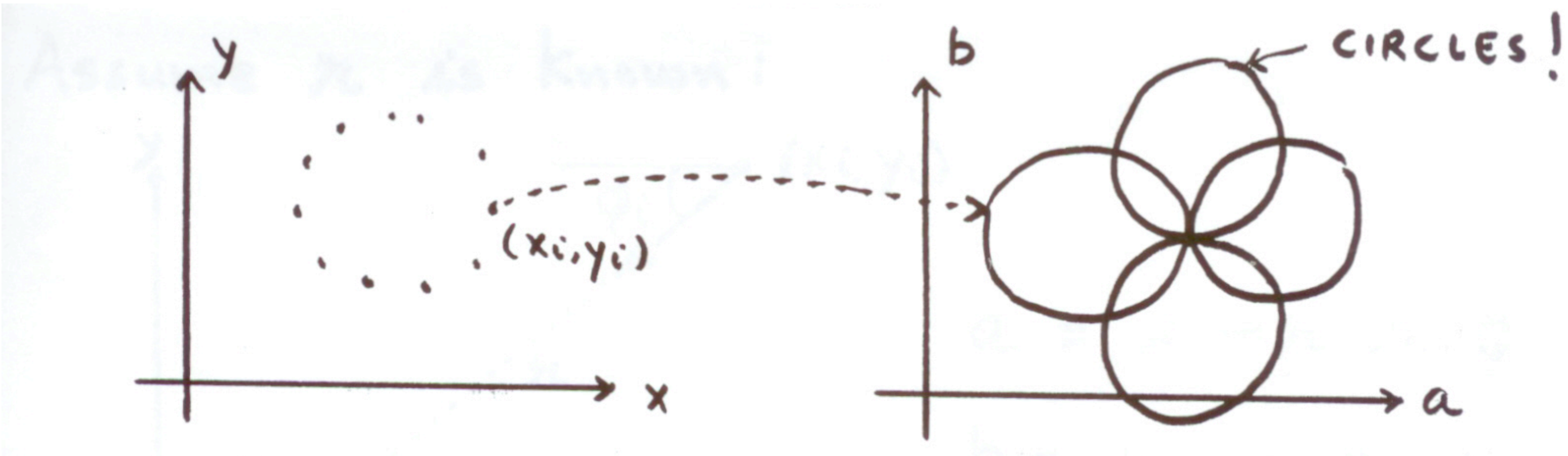
Equation of a circle: $(x_i - a)^2 + (y_i - b)^2 = r^2$

A circle of a **specific** radius *r* is uniquely identified by the location of its center, i.e. the parameter pair (a,b)

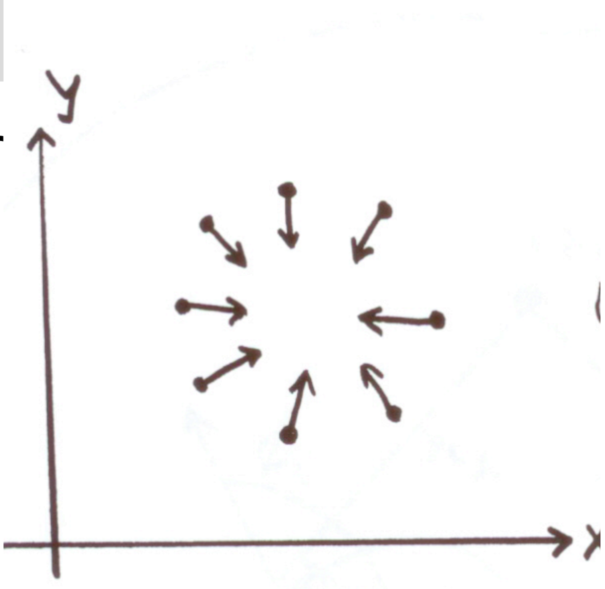
For circle detection of known radius *r*, a point in image space becomes a circle in Hough space.

If the radius is **known**, we have a 2D Hough space. The accumulator array is $A(a,b)$.

If the radius is **unknown**, we have a 3D Hough space. The accumulator array is $A(a,b,r)$.

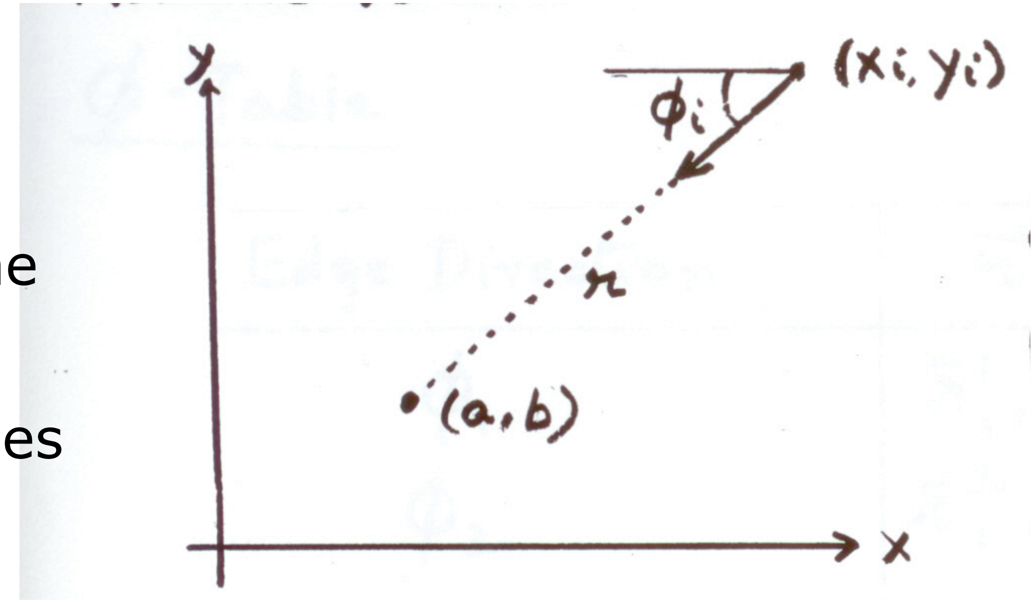


Using Gradient Information



- If we use a gradient-based edge detector, for each edge point (x_i, y_i) we also know the edge direction ϕ_i .
- For a specific radius *r*, we only need to move distance *r* in the direction of ϕ_i to obtain an estimate of the center of the circle.

$$a = x_i - r\cos\phi_i$$
$$b = y_i - r\sin\phi_i$$

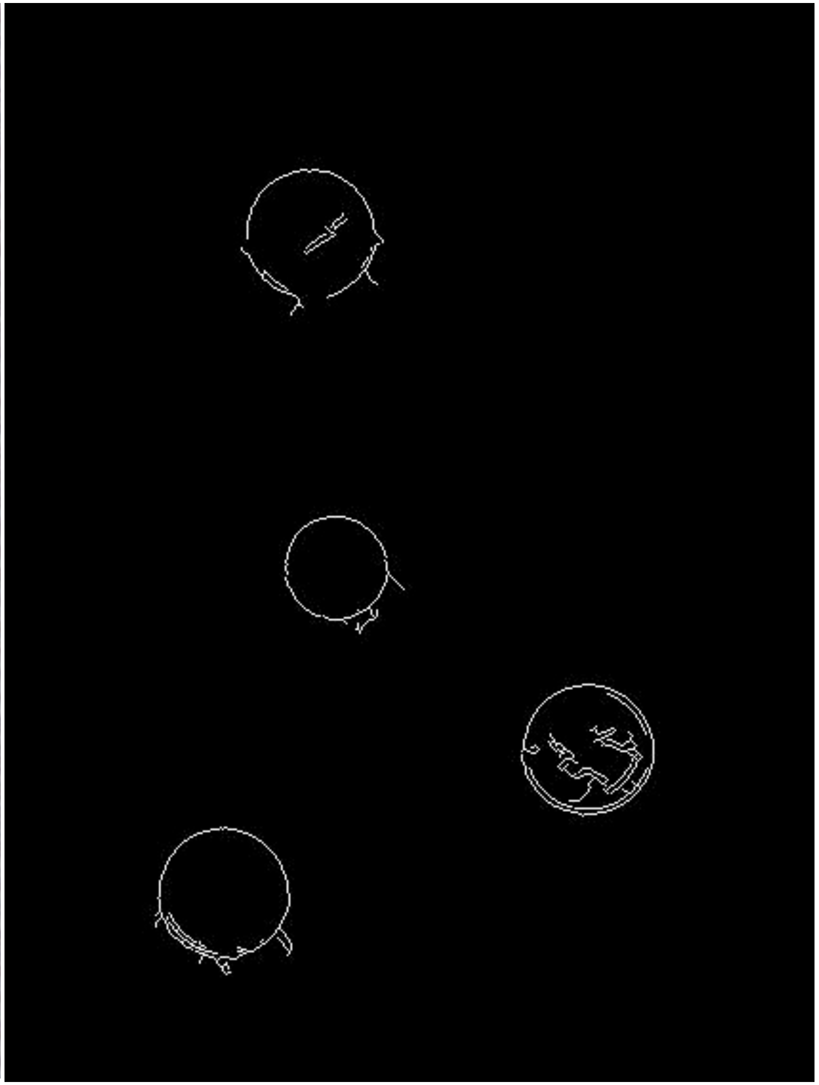


- We only need to increment one point in the accumulator.
- A point in image space becomes a **point** in Hough space.

Example – Finding Coins

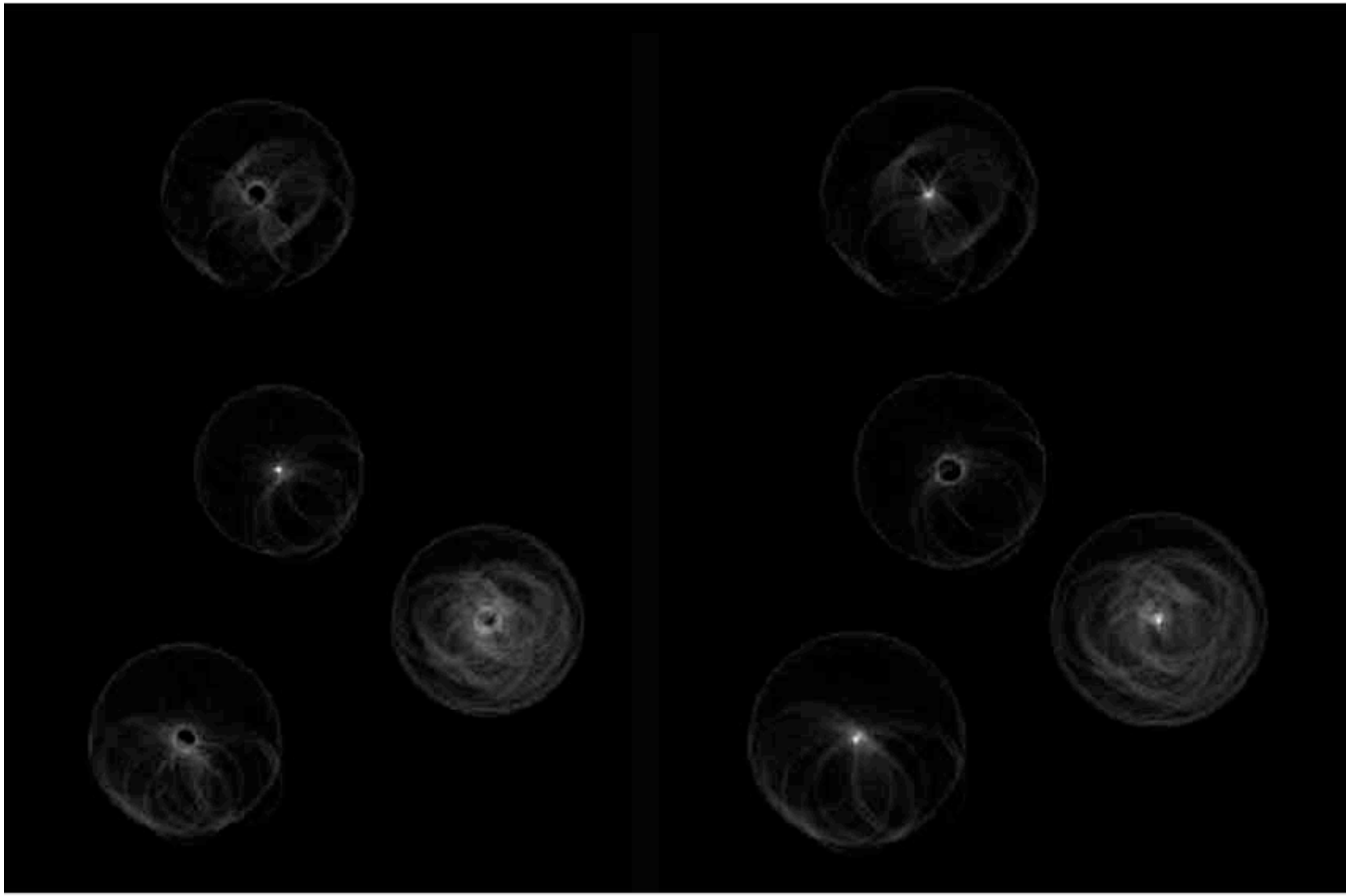
Original Image

Edges (note noise)

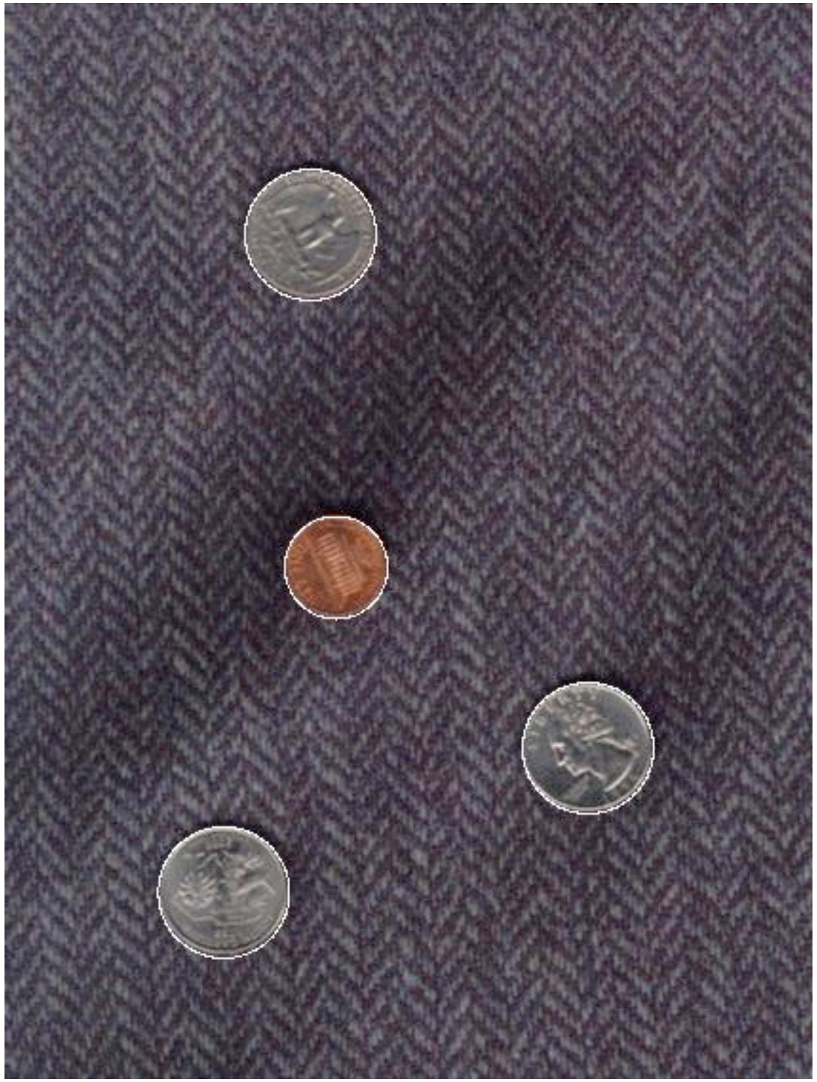


Hough Space of Coin Example

Pennies Quarters



Example – Final Result



Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

HT for Ellipse Detection

Equation of an ellipse:

$$\frac{(x_i \cos\vartheta + y_i \sin\vartheta - a)^2}{d_1^2} + \frac{(-x_i \sin\vartheta + y_i \cos\vartheta - b)^2}{d_2^2} = 1$$

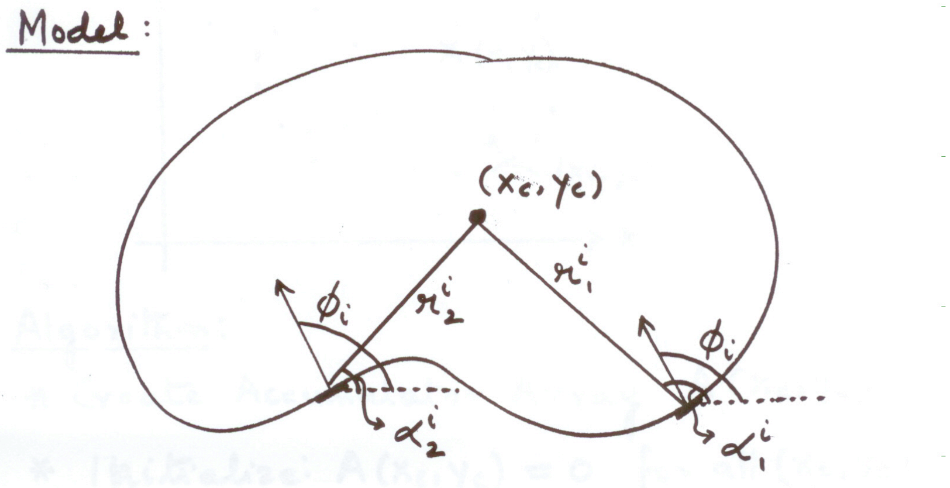
where (a,b) is the center of the ellipse, d_1 and d_2 are the lengths of the major and minor axes of the ellipse and ϑ is the angle between the major axis and the horizontal axis.

An ellipse is then uniquely identified by the location of its center, its size and its orientation, i.e. by the parameter tuple (a,b,d_1,d_2,ϑ)

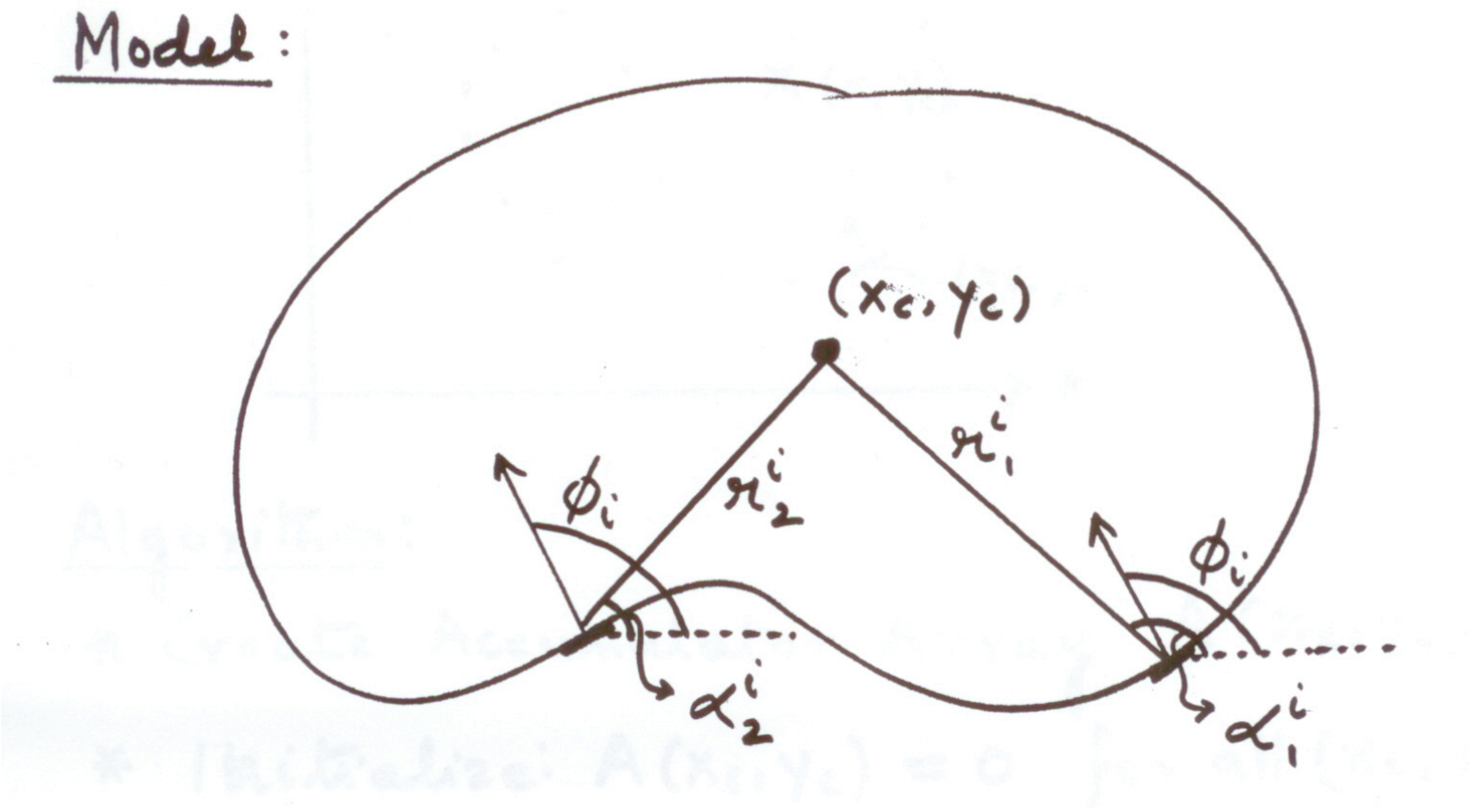
For arbitrary ellipse detection we have a 5D Hough space. The accumulator array is $A(a,b,d_1,d_2,\vartheta)$.

Generalized Hough Transform

- What characteristic/property can we use for an arbitrary shape?
- Generalize the circle idea.
- Preprocess a sample of the shape.
  - Run a gradient-based edge detector, so that for each border pixel there is an edge pixel and its associated orientation.
  - Choose an arbitrary point C as the center of the shape.
  - Compute for the given shape what distance and in which direction one needs to move from an edge pixel to reach the center point C.
- After preprocessing, there exists a representation that allows voting for the shape's center.



Generalized Hough Transform



GHT- Shape Description

ϕ - Table

Edge Direction	$\bar{r} = (r, \alpha)$
ϕ_1	$\bar{r}^1_1, \bar{r}^1_2, \bar{r}^1_3$
ϕ_2	$\bar{r}^2_1, \bar{r}^2_2$
$\vdots$	$\vdots$
ϕ_i	$\bar{r}^i_1 \; \vdots \; \bar{r}^i_2$
$\vdots$	$\vdots$
ϕ_n	$\bar{r}^n_1, \bar{r}^n_2$

GHT Algorithm

Find Object Center (x_c, y_c) given edges (x_i, y_i, ϕ_i)

Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$

For each edge point (x_i, y_i, ϕ_i)

For each entry r_k^i in the shape table, compute:

$$x_c = x_i + r_k^i \cos \alpha_k^i$$

$$y_c = y_i + r_k^i \sin \alpha_k^i$$

Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find Local Maxima in $A(x_c, y_c)$

HT Final Comments

- In order to handle different scales *s* a new dimension must be added in the Accumulator.
- In order to handle different rotations θ, a new dimension must be added in the Accumulator.
- HT is relatively insensitive to occlusion.
- HT can handle discontinuities.
- It is relatively insensitive to noise.
- HT is very effective on simple shapes (lines, circles, etc.)

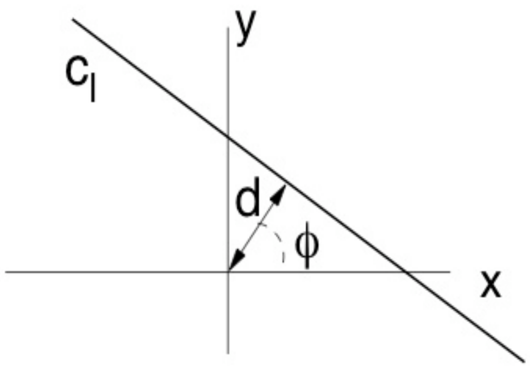
Hough Transform vs. Radon Transform

- One key characteristic of Hough Transforms is that shapes can be described as a set of parameters that are characteristic for a specific shape.
- Both the Radon and the Hough transform can be used for detecting parameterized shapes.
- In fact, the two transforms can be considered equivalent.
- Some regard the Radon Transform formulation:
  - more rigorous mathematically
  - more intuitive.

Hough Transform vs. Radon Transform

- The Radon transform is named after J. Radon who showed how to describe a function in terms of its (integral) projections.
- The mapping from the function onto the projections is the Radon transform.
- The inverse Radon transform corresponds to the reconstruction of the function from the projections.
- For a line, we compute the Radon transform along the projection line c_l as follows:

$$R(d,\varphi) = \int_{\Re} I(d\cos\varphi - s\sin\varphi, d\sin\varphi + s\sin\varphi)\,ds$$



Hough vs. Radon Transform

- The algorithms for looking for a shape using Radon vs. Hough can then be seen as:
- Radon: For each point ***p*** in parameter space, collect all the values of I(x,y), apply the template weights K(x,y; ***p***), and sum everything (integrate)
- Hough: Initialize the accumulator A(***p***) to zero. For each point (x,y) in the input image determine its contribution, weighted with K(x,y; ***p***), to each of the counters in A(***p***) and update A(***p***).

1. M. van Ginkel, C.L. Luengo Hendriks and L.J. van Vliet. "A Short Introduction to the Radon and Hough Transforms and how They Relate to Each Other." Delft University of Technology Quantitative Imaging Group Technical Report Number QI-2004-01.

Image Sources

1. A large portion of these slides have been adapted by the presentation of S. Narasimhan, http://ww.cs.cmu.edu/afs/cs/academic/class/15385-s06/lectures/ppts/lec-9.ppt
2. The data on slides 7,11,X X, are from the slides of J. Wyatt http://www.cs.bham.ac.uk/~jlw/vision/hough_transform.ppt
3. The coin example is courtesy of Vivek Kwatra.