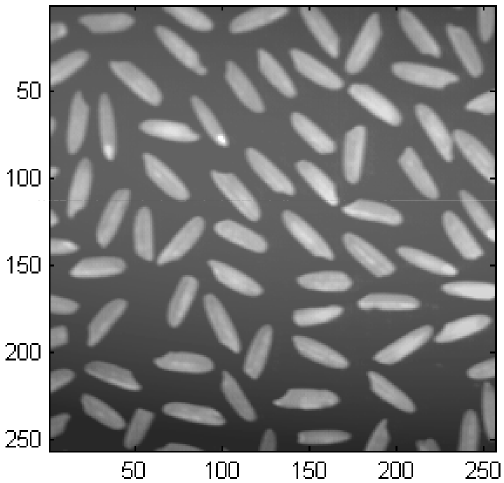
# Using Edge Representation

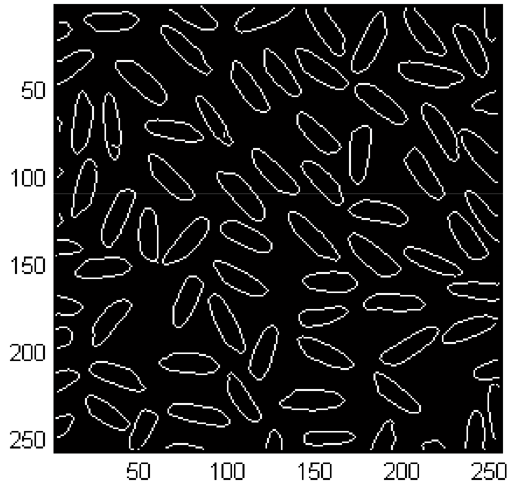
Lecture 13

## Using Edge Representation…

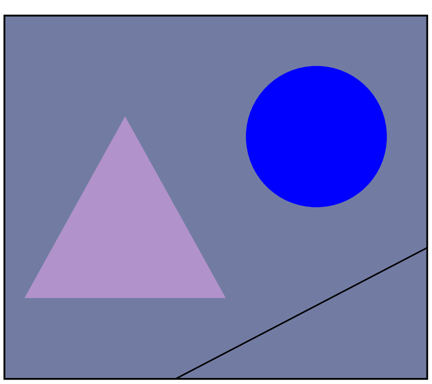
- Stereo matching problem
- Input:
  - Two images with disparity
  - Camera Calibration information
- Computation
  - Find corresponding features in two images
- Output
  - Disparity in corresponding features is related to depth
- Edges and corners help in finding correspondences

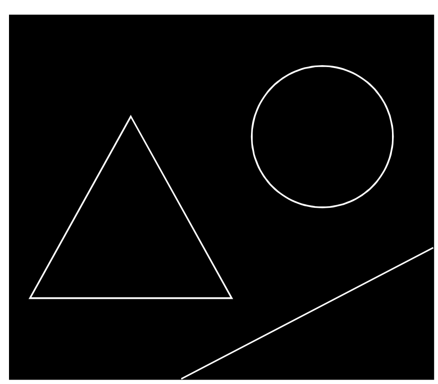
## Finding Shapes from Edges



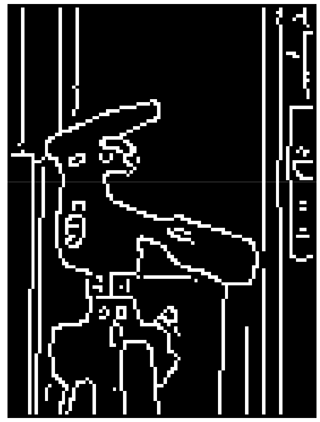

## Finding Shapes from Edges





## Edge Representation for Shape Analysis

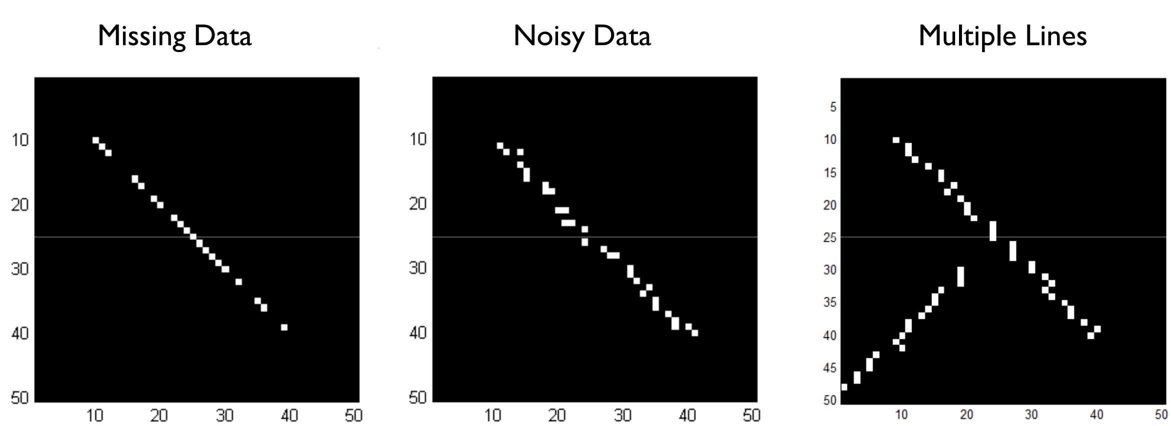
- What about noisy edges?



- Problem Def.: Find straight lines…

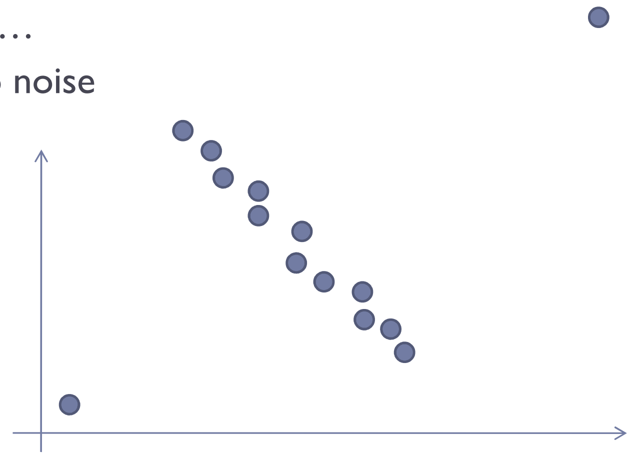
- Images from: http://www.cogs.susx.ac.uk/users/davidy/teachvision/vision4.html

## Problems in Finding Lines



## Least Squared Error Solution

- We have already looked at this solution in previous lectures
- Disadvantages?
  - Multiple Lines…
  - Not robust to noise
- Example



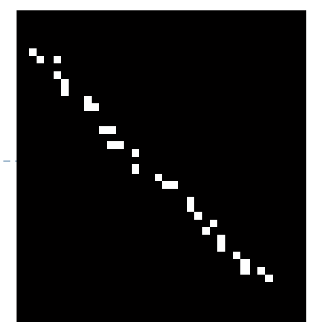
## Finding Lines

- Problem Definition:
- Given a binary image, find all *significant* lines
- Line: $y = \boldsymbol{m}x + \boldsymbol{c}$
- Estimate m,c parameters of all significant lines in presence of noise

## Hough Transform

- Method to find any type of shape that can be represented in **parametric** form
- E.g. lines, circles, parabolas, ellipses…
- Generalized Hough Transform
  - For arbitrary shapes

## Hough Transform for Lines

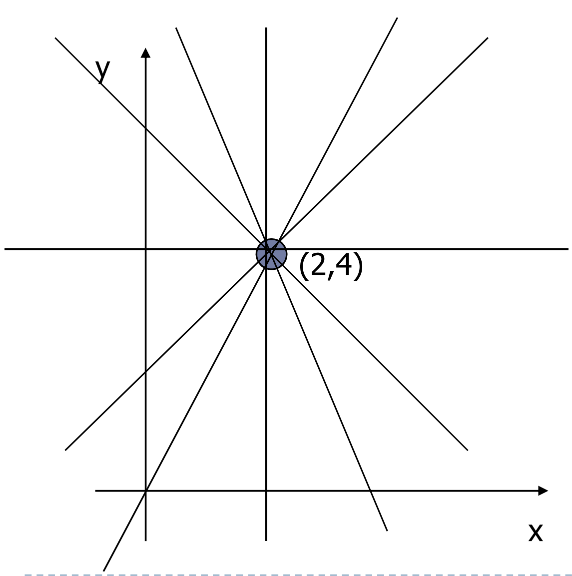


- General Idea:
  - Search for the best possible **m** and **c** parameters given the data
- Consider all possible lines in the image
- Consider all possible lines that can pass through a single point
  - Restriction of the statement above.
- A line that passes through 1 point gets one vote
- Find the line that gets most votes

## Hough Transform for Lines

- **Aim**: Create a mechanism for **voting**
  - A line should get as many votes as the points it passes through
- Equation of line is **y=mx+c**
  - **m** is slope, **c** is intercept
- Consider only one point **(x,y)**
  - For example (2,4)
- How many lines can pass through this point?

## Hough Transform for Lines



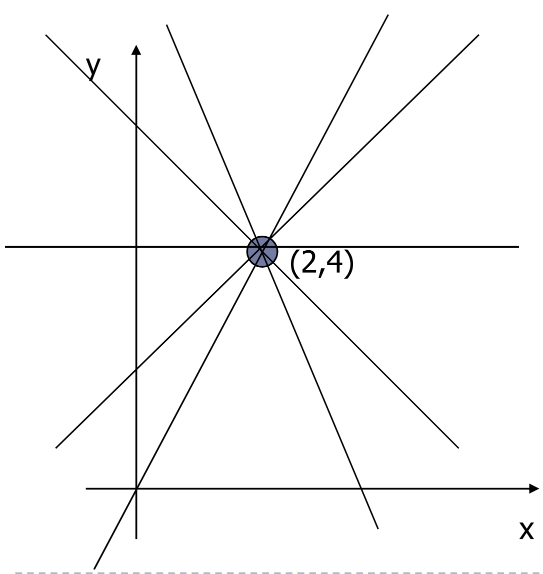

x=2
y=4
y=x+2
y=-x+6
y=2x
y=-2x+8

And so on… (infinite lines)

## Hough Transform for Lines

- Can we write the general expression for all the lines passing through (2,4)?
- All those lines will have a specific relationship between **m** and **c**
- Any arbitrary combination of **m** and **c** will not pass through the given point; only certain combinations will work

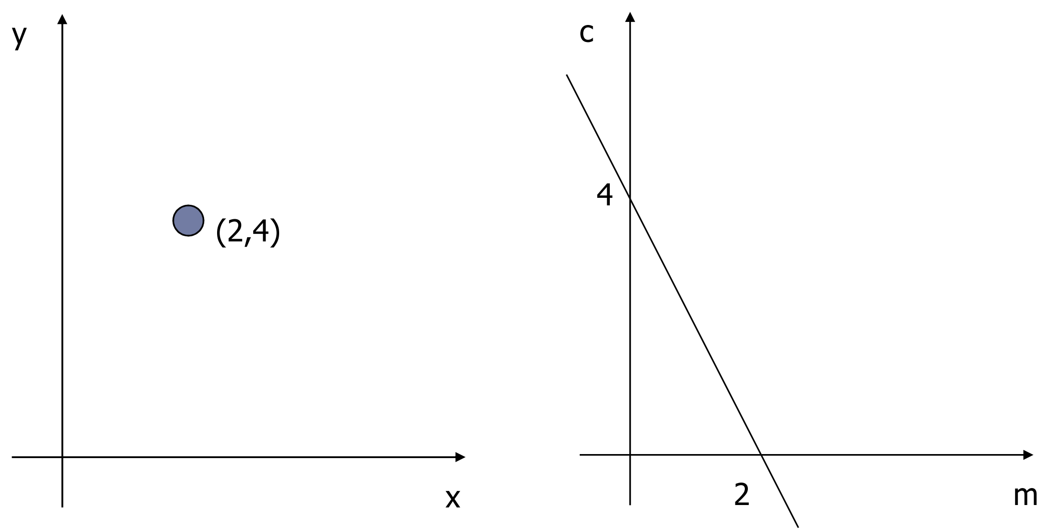
## Hough Transform for Lines




| | |
|---|---|
| $y=4$ | $m=0, c = 4$ |
| $y=x+2$ | $m=1, c = 2$ |
| $y=-x+6$ | $m=-1, c = 6$ |
| $y=2x$ | $m=2, c = 0$ |
| $y=-2x+8$ | $m=-2, c = 8$ |

What is the relationship between valid pairs of (m,c)?

## Hough Transform for Lines



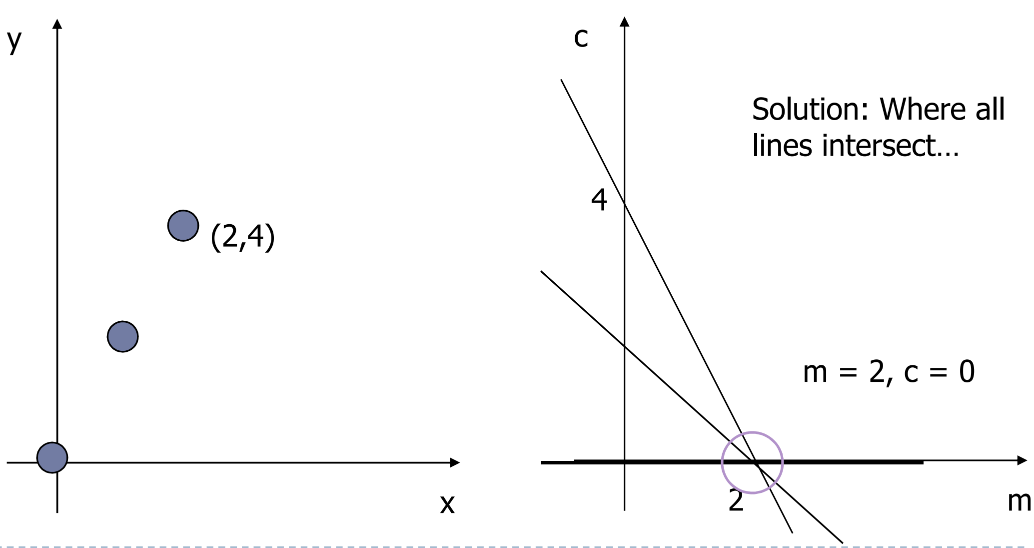

## Hough Transform for Lines

- Equation of line is **y=mx+c**
- We are given **(x,y)** [e.g. (2,4)]
- **(m,c)** are the unknowns
- Can be rewritten as **c = (-x)m +** y
- Consider **(x,y)** space: **y=mx+c** represents a line
- Consider transformed space **(m,c)**, then **c=(-x)m + y** is a line in this space
- **(-x)** is **gradient**, **y** is the **intercept**

## Interpretation

- Line in **(m,c)** space represents all possible lines that could pass through a **single point (x,y)**
- **Point** in **(x,y)** space is a **line** in **(m,c)** space
- **Point** in **(m,c)** space is a …
- **Line** in **(x,y)** space

## Finding Lines using Hough Transform



## Hough Transform for Lines

- Initialize **Accumulator** array, **A**, of two dimensions **(m, c)**
- For each point **(x,y)** in image, increment cells along line **c = -xm+y** by 1
- Find **maximum** point in accumulator array for solution

## Algorithm

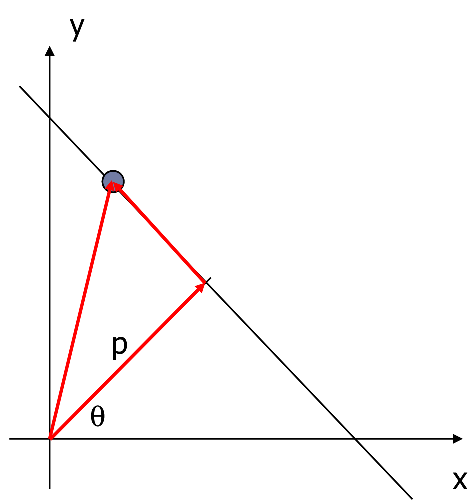
1. Quantize parameter space
   **A**[$c_{min}$, … , $c_{max}$, $m_{min}$, … , $m_{max}$]
2. For each edge point (x,y)
   For (m = $m_{min}$, m ≤ $m_{max}$, m++)
   c = (-x)m + y
   **A** [c,m] = **A** [c,m] + 1;
3. Find local maxima in **A**

## Hough Transform for Lines

- Problems with this procedure?
- What about the range of slope?
- m spans -∞ to ∞
- Solution?
- Use alternate parameterization of line

## Alternate Line parameterization

- $p = x \cos\theta + y \sin\theta$
- p is the perpendicular to the line
- $\theta$ is the angle p makes with the x-axis

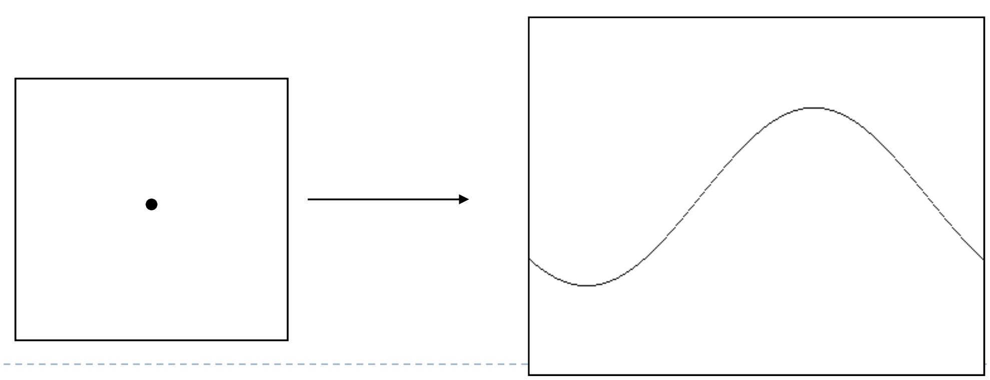


## Algorithm (polar form)

1. Quantize parameter space
   **A** [$\theta_{min}$, … , $\theta_{max}$, $p_{min}$, … , $p_{max}$]
2. For each edge point (x,y)
   For ($\theta = \theta_{min}, \theta \leq \theta_{max}, \theta$++)
   $p = x \cos\theta + y \sin\theta$
   **A** [θ, p] = **A** [θ, p] + 1;
3. Find local maxima in **A**

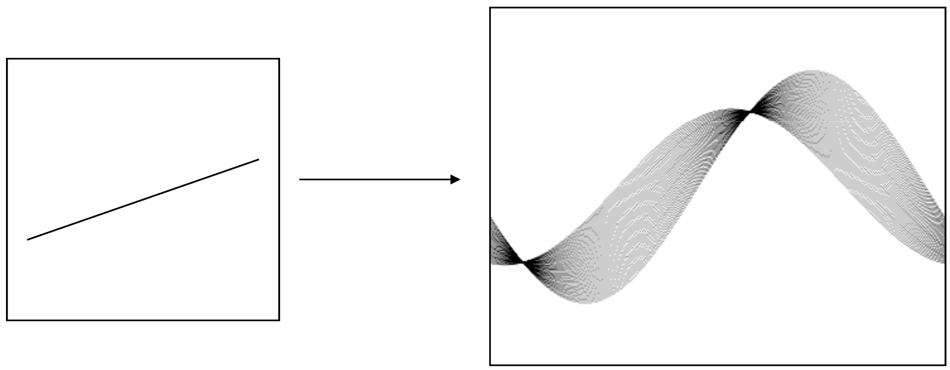
## HT for Lines (polar form)

- Point is **(x,y)** space represents _______ in the parameter space **(p, θ)**?
- Answer: Sinusoid curve

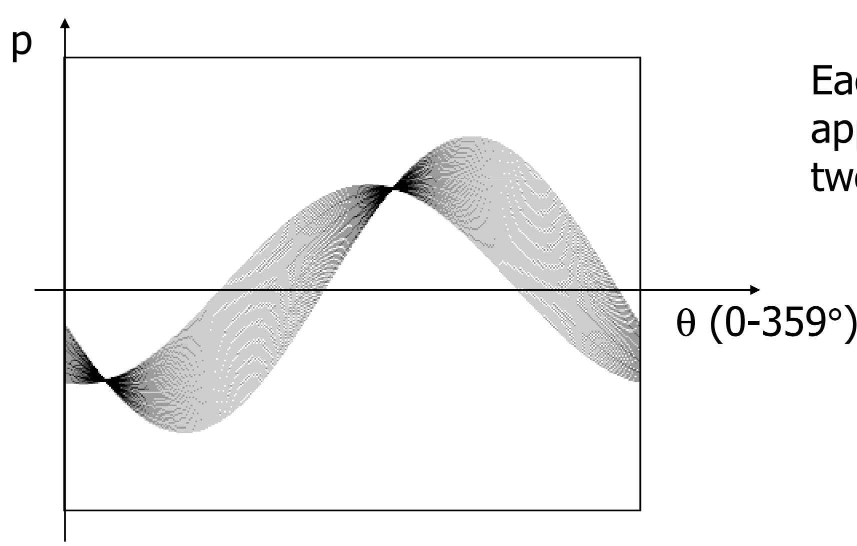


## HT for Lines (polar form)

- Line in **(x,y)** space represents ________ in **(p, θ)** space?



## HT for Lines (polar form)




Each line appears as two peaks

# Additional advantage of Polar Form

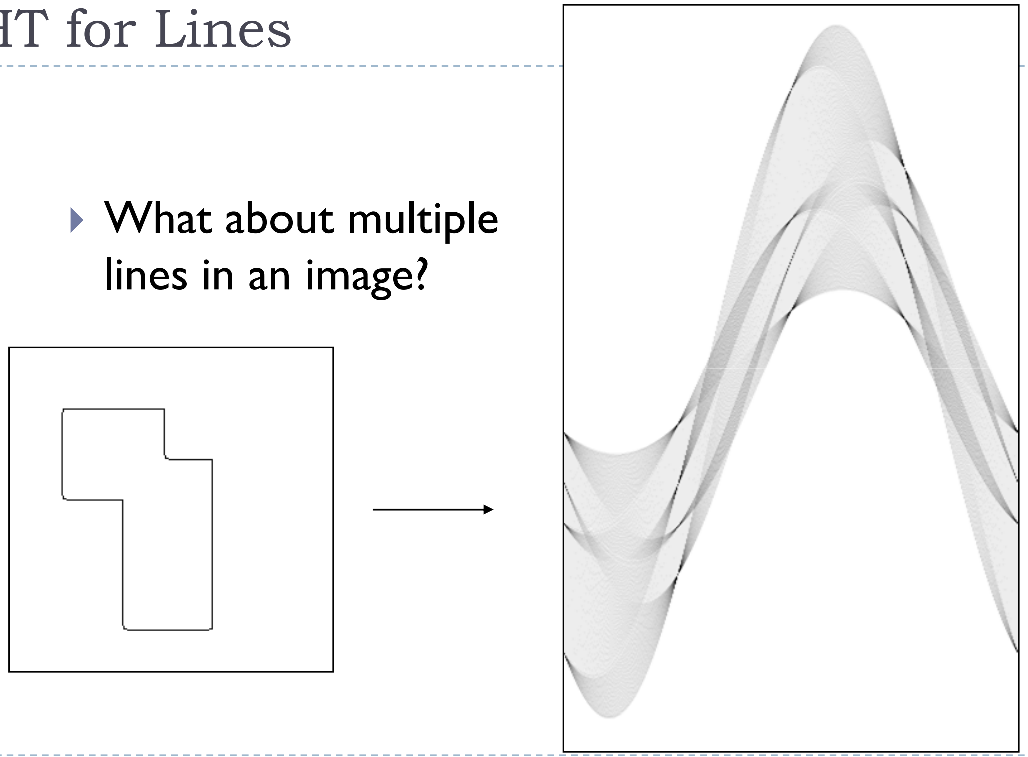
- Line which passes through **(x, y)** was assumed to have all possible values of $\theta$
- Gradient direction?
- $\theta$ can be computed from **gradient direction**

# Algorithm (polar form/improved)

1. Quantize parameter space
   **A** $[\theta_{min}, \dots, \theta_{max}, p_{min}, \dots, p_{max}]$
2. For each edge point (x,y)
   Compute $\theta$ from gradient direction
   $p = x \cos\theta + y \sin\theta$
   **A** $[\theta, p]$ = **A** $[\theta, p] + 1$;
3. Find local maxima in **A**

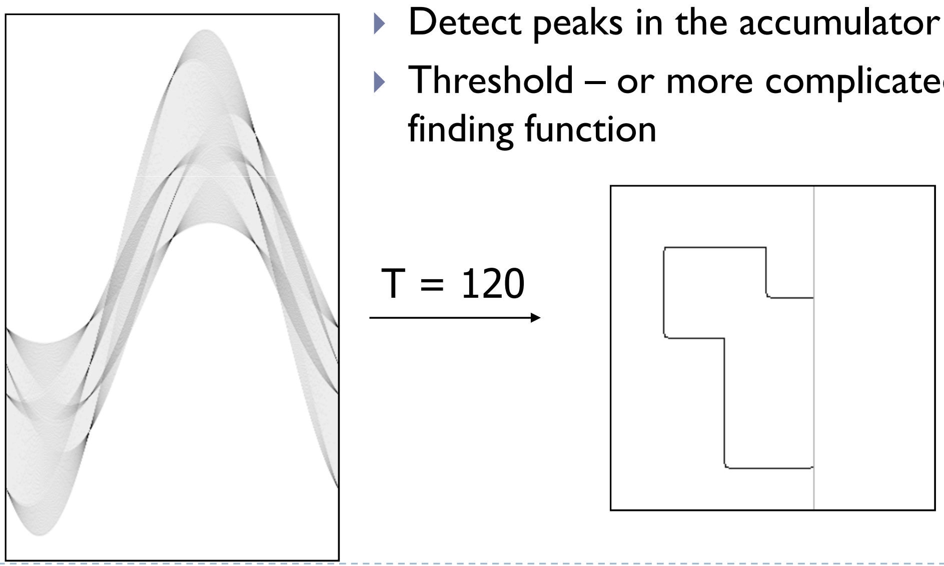
## HT for Lines

- What about multiple lines in an image?

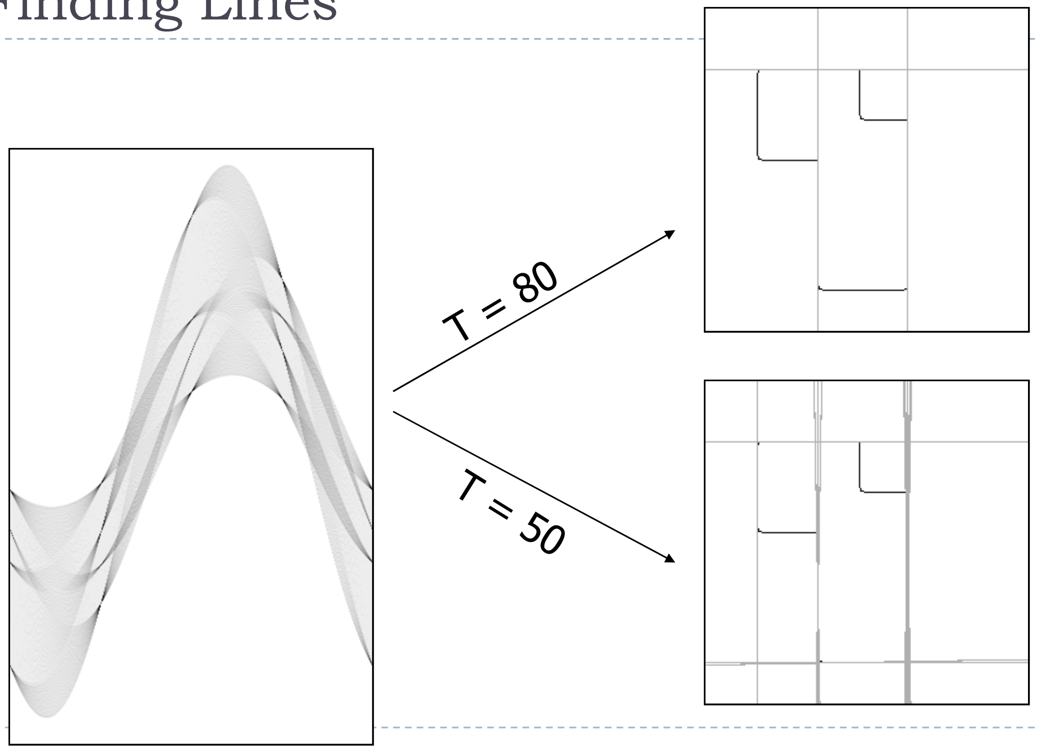


## Finding Lines

- Detect peaks in the accumulator array
- Threshold – or more complicated peak finding function

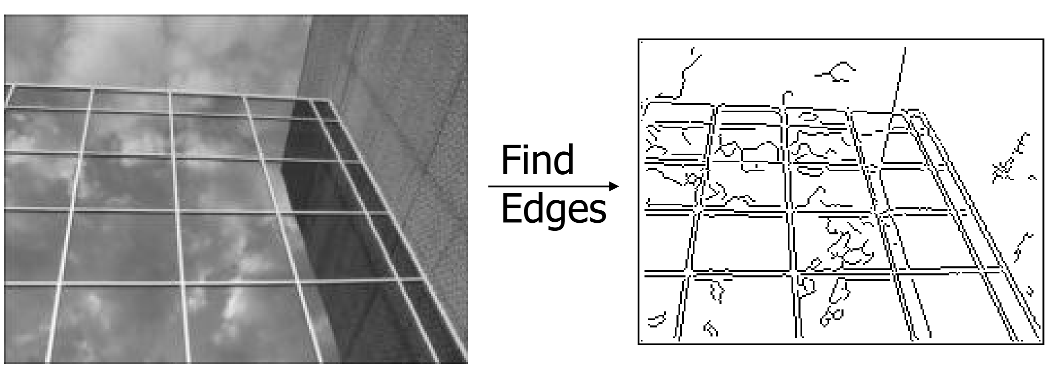


# Finding Lines

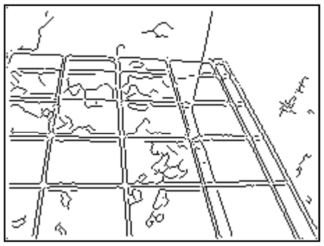



# HT for Lines

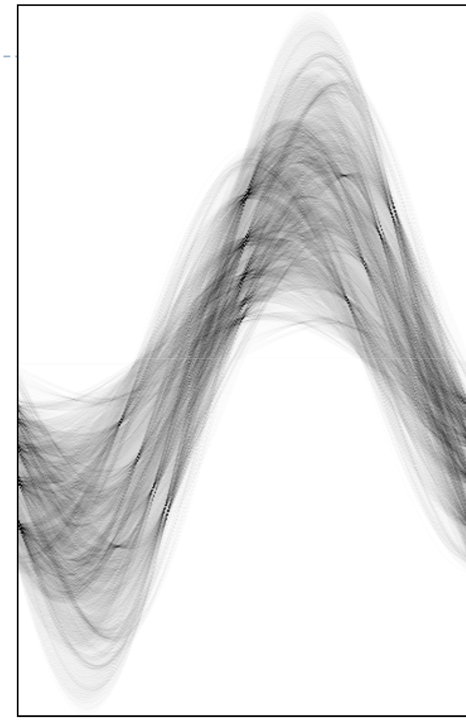
- Complicated Images?



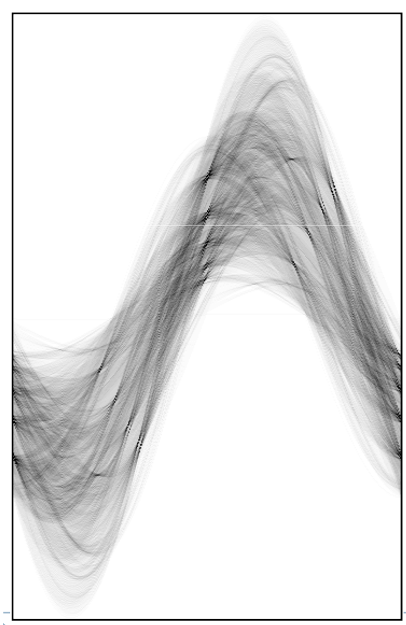
# HT for Lines

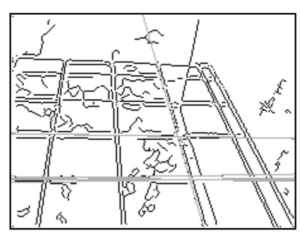


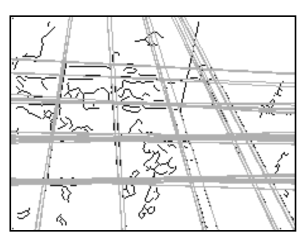
Accumulator Array?

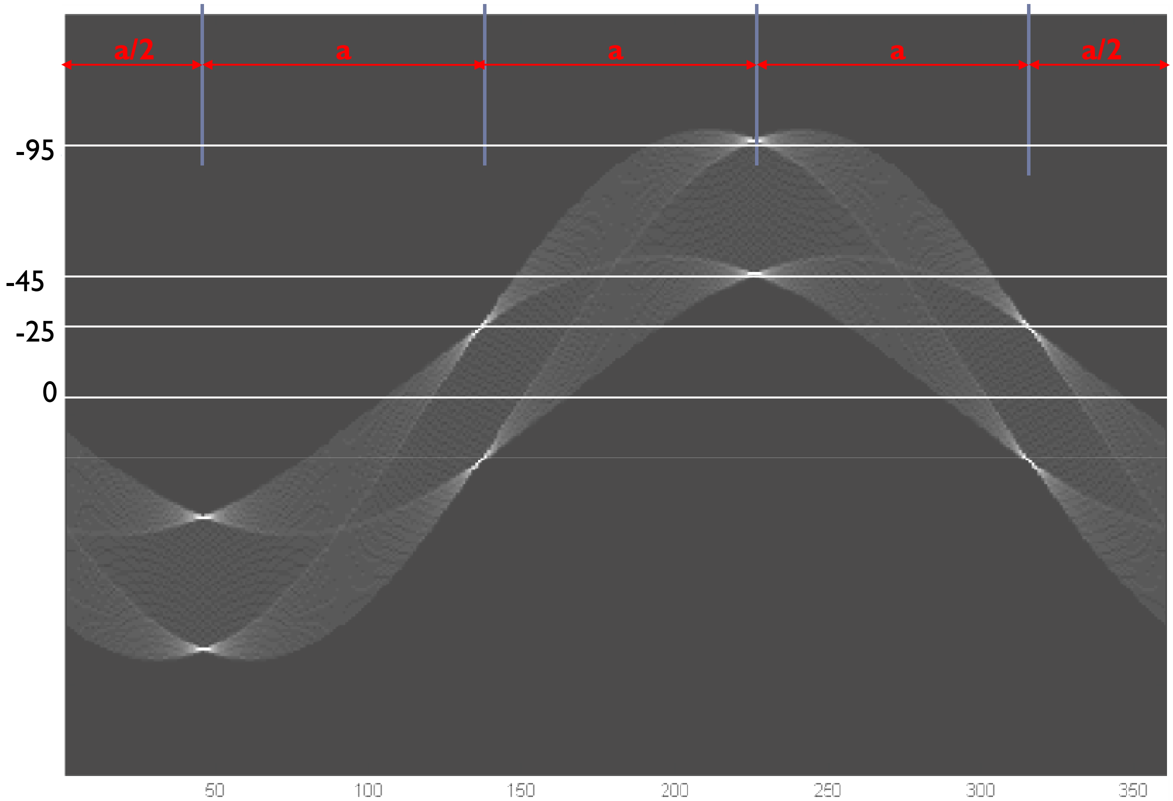


# HT for Lines

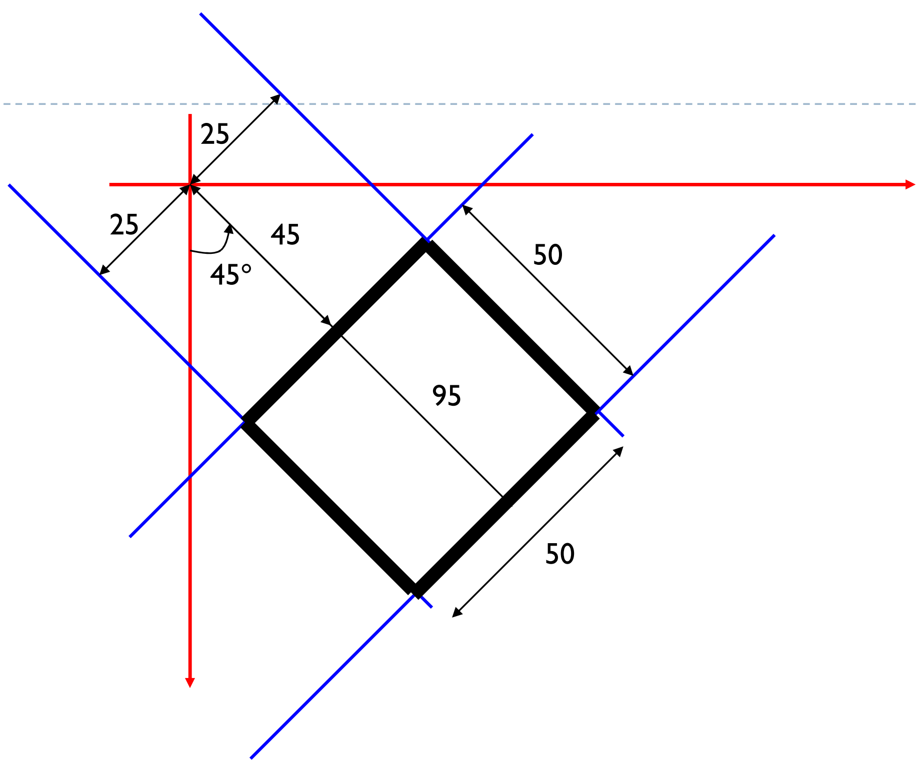






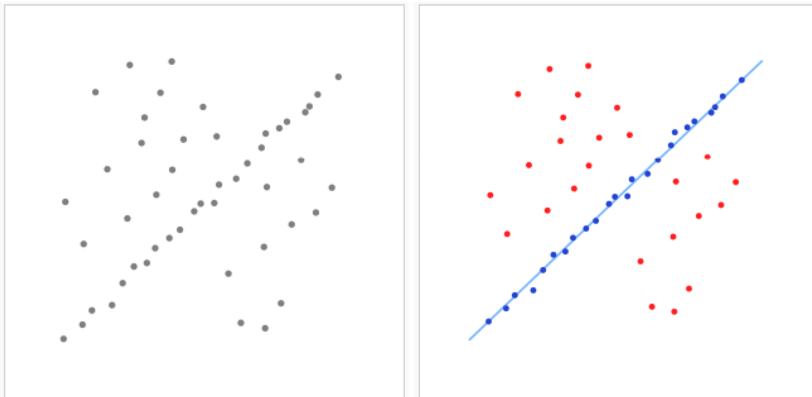



1: If all peaks are equal height, what can you tell about the original image?
2: What if it is known that the original image is a closed shape?



# Another Approach to Line Fitting

- RANSAC (RANdom SAmple Consensus)
- Very robust to outliers



A data set with many outliers for which a line has to be fitted.

Fitted line with RANSAC, outliers have no influence on the result.

- http://en.wikipedia.org/wiki/RANSAC

# RANSAC… General Approach

- Select minimum number of random points from data needed to estimate the model
- Estimate the model from selected random points
- Check how many other points are consistent with the fitted model (Consistent Set)
  - If consistent set is large enough, estimate model from all points in the consistent set
  - If error of fitted model is lower than previous best model, make the current model as the best model
- Repeat a number of times

## RANSAC… Pseudocode (from http://en.wikipedia.org/wiki/RANSAC )

```
input:
    data - a set of observations
    model - a model that can be fitted to data
    n - the minimum number of data required to fit the model
    k - the maximum number of iterations allowed in the algorithm
    t - a threshold value for determining when a datum fits a model
    d - the number of close data values required to assert that a model fits well to data
output:
    best_model - model parameters which best fit the data (or nil if no good model is found)
    best_consensus_set - data point from which this model has been estimated
    best_error - the error of this model relative to the data

iterations := 0
best_model := nil
best_consensus_set := nil
best_error := infinity
while iterations < k
    maybe_inliers := n randomly selected values from data
    maybe_model := model parameters fitted to maybe_inliers
    consensus_set := maybe_inliers

    for every point in data not in maybe_inliers
        if point fits maybe_model with an error smaller than t
            add point to consensus_set

    if the number of elements in consensus_set is > d
        (this implies that we may have found a good model,
        now test how good it is)
        better_model := model parameters fitted to all points in consensus_set
        this_error := a measure of how well better_model fits these points
        if this_error < best_error
            (we have found a model which is better than any of the previous ones,
            keep it until a better one is found)
            best_model := better_model
            best_consensus_set := consensus_set
            best_error := this_error

    increment iterations

return best_model, best_consensus_set, best_error
```