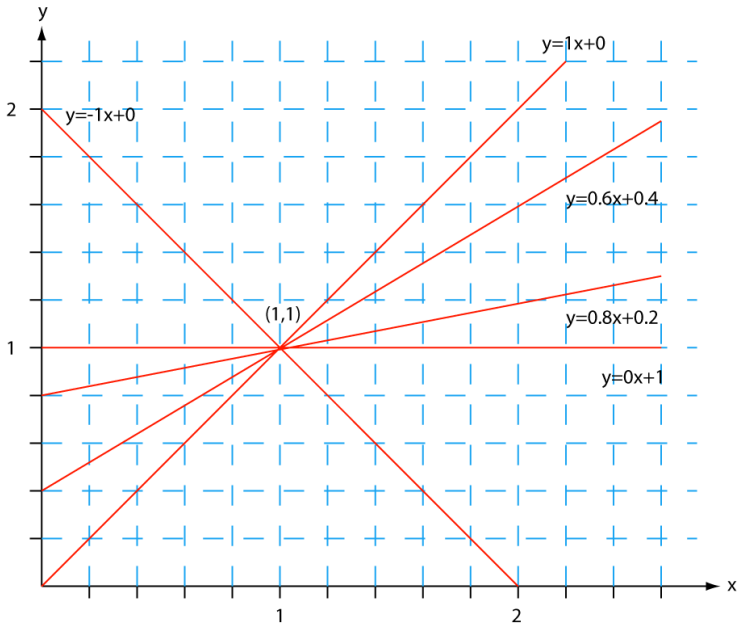
IN5520 – Digital Image Analysis

HOUGH TRANSFORM



Andreas Kleppe 08.09.2022
Modification of slides by Fritz Albregtsen

Curriculum

- G&W, "DIP4E", Ch. 10 (pages 737-742): Last section of 10.2: "Global Processing Using the Hough Transform"
- Full lecture available online (video + slides).
- Today's lecture will focus on the main points.

Plan for today

- Before the break, lecture and Q&A about:
  - Detecting lines using Hough transform (including preliminaries)
- After the break, lecture and Q&A about:
  - Detecting other parametric curves using Hough transform

Motivation

- Assume we want to find the location of line segments in an image.

- Could we simply search for it (e.g. using morphology)?
  - Need a linear structuring element for each direction.
  - Issues with discrepancies from a perfect line.
  - Will probably be both slow and provide a suboptimal solution.

- Could we use regression?
  - Works poorly if based on pixels from multiple lines.
  - Works nicely if we know which pixels belongs to each line, but to know this we need to find the line.

- Hough transform can give robust detection despite discrepancies (e.g. noise and partial occlusion) from a perfect line.

Preprocessing: Edges detection

- Edge detection is often used as preprocessing to Hough transform.
- From intro lecture (and IN2070):
  1. Find vertical and horisontal edges response by convolving the image with e.g. the Sobel operator:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

$g_x(x,y)=H_x*f(x,y)$
$g_y(x,y)=H_y*f(x,y)$

  2. Calculate the gradient magnitude:

$$G(x,y)=\sqrt{g_x^2(x,y)+g_y^2(x,y)}$$

Example →





Preprocessing: Edges detection

- Edge detection is often used as preprocessing to Hough transform.
- From intro lecture (and IN2070):
  3. Threshold the gradient magnitude image using e.g. Otsu's method.
     - Can be tricky, e.g. if there are relatively few foreground pixels.



Preprocessing: Edges detection

- Other approaches for edge detection includes:
- The Canny edge detector (see G&W, "DIP4E", Ch 10, p 729-735)
  1. Lowpass filtering, & finding gradient magnitude and direction (Sobel).
  2. Thinning of gradient magnitude $G(x,y)$ orthogonal to gradient direction $\theta(x,y)$

$$\theta(x,y) = \tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$$

  3. Hysteresis thresholding (two thresholds, T_h and T_l):
     a. Mark all pixels where $g(x,y) \geq T_h$
     b. For all pixels where $g(x,y) \in [T_l, T_h)$:
        - If (4 or 8)-neighbour of marked pixel, then mark this pixel too.
     c. Repeat from 3b until convergence.
- Laplacian of Gaussian (see intro lecture and/or G&W, "DIP4E", Ch 10, p 724-729)
  – Find zero-crossings after convolving image with a LoG filter:

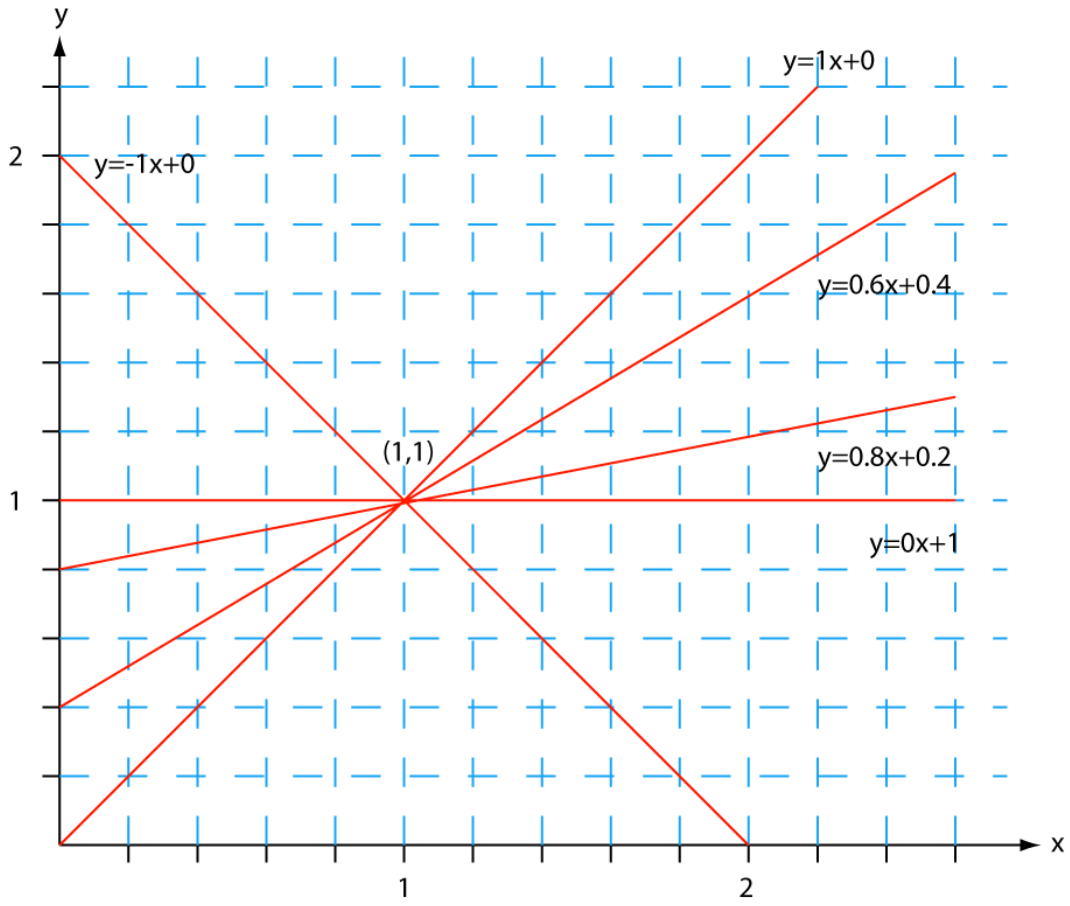
$$-\nabla^2 G = \frac{1}{2\pi\sigma^4}\left(2 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}}$$

Hough transform – basic idea

- Assume we want to find straight lines in a binary image.
- For each foreground pixel (x_i ; y_i), there are multiple lines passing through it.
- They all satisfy the equation:

  $$y_i = ax_i + b$$

  for varying values of the parameters (a,b)



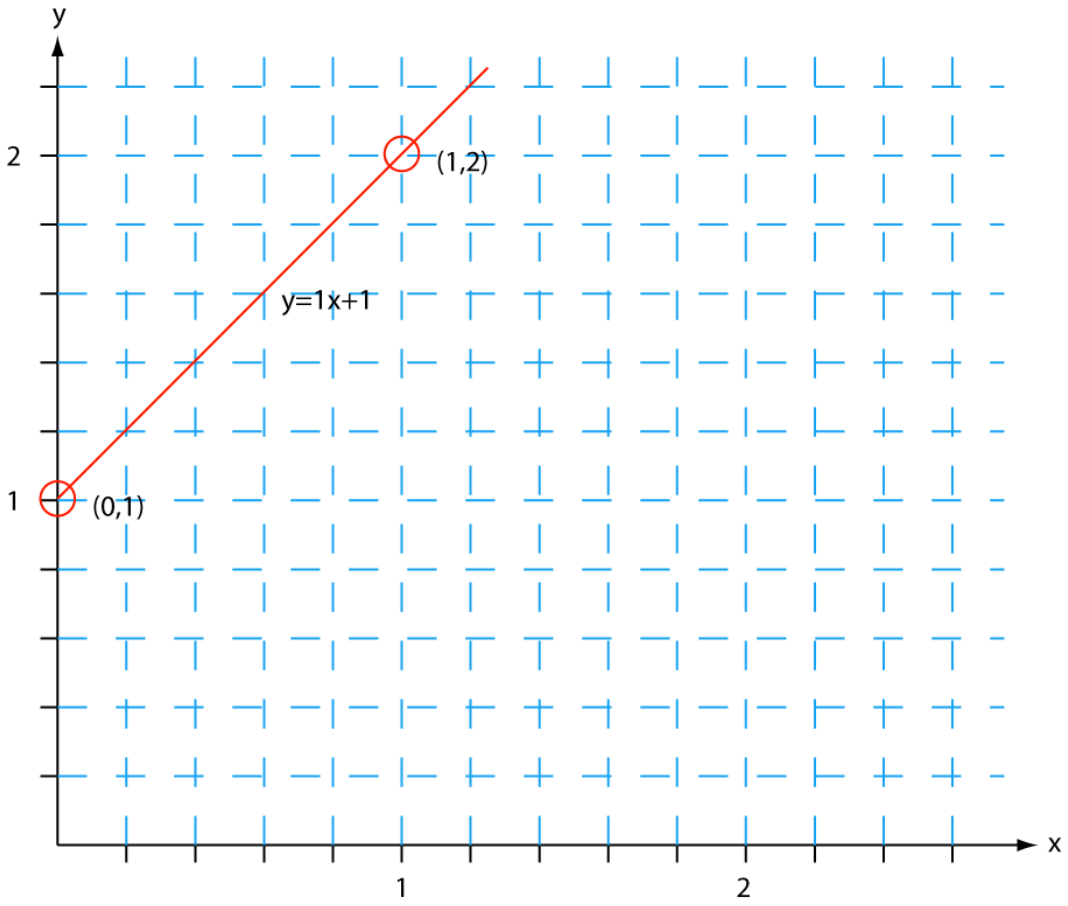
Hough transform – basic idea

- Consider two foreground pixels, $(x_1 ; y_1)$ and $(x_2 ; y_2)$
- Multiple lines pass through each of them:

$$y_1 = ax_1 + b$$
$$y_2 = ax_2 + b$$

for varying values of the parameters (a,b)

- One and only one of the lines are equal!
  - That is, same (a,b)



Hough transform – basic idea

- A line in (x,y) space with parameters (a,b) is given by:

$$y = ax + b$$

Recall: a = slope, b = intercept

- This equation can be rewritten to:

$$b = -xa + y$$

- The line in (x,y) space is simply a point in (a,b) space.

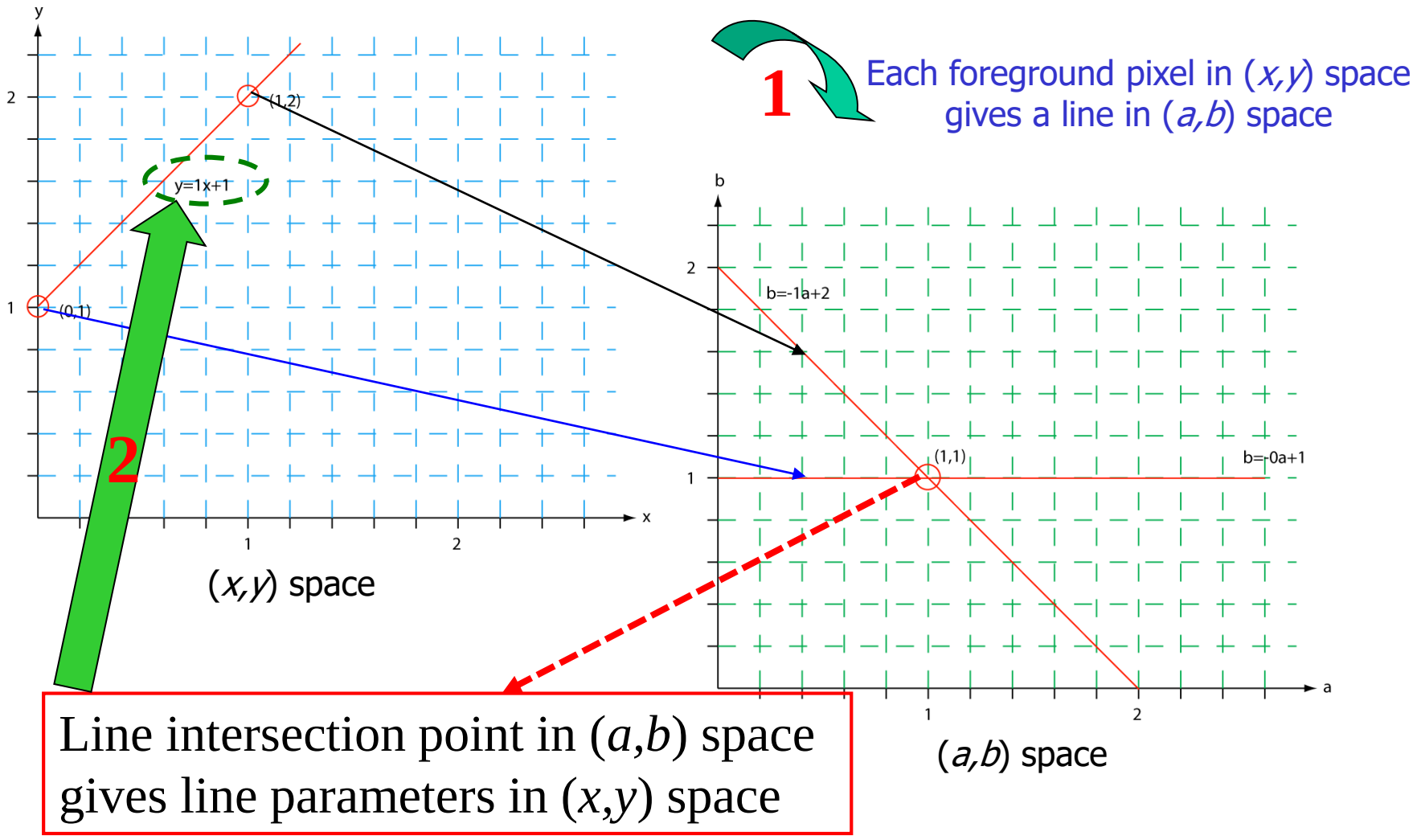
- Consider (x,y) as parameters and (a,b) as variables in:

$$b = -xa + y$$

Then: $-x$ = slope, y = intercept

- A point (x_1,y_1) in (x,y) space gives a line in (a,b) space.
- Another point (x_2,y_2) on the same line $y = a'x + b'$ in (x,y) space will give rise to another line in (a,b) space and the two lines in (a,b) space will intersect in (a',b').

Hough transform – basic idea



Implementing basic idea

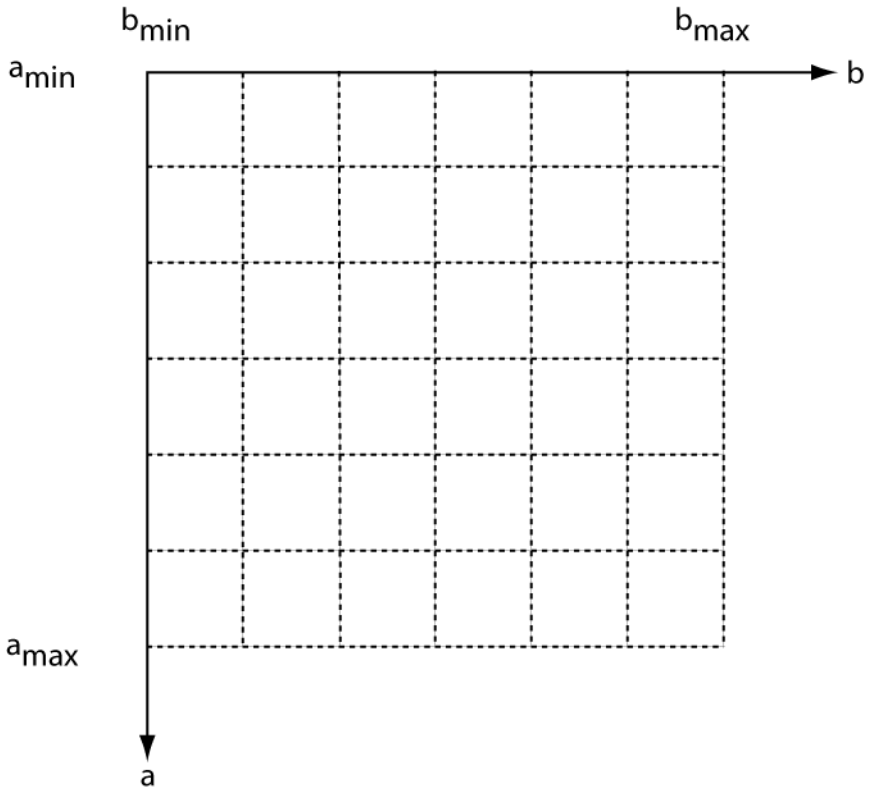
- Quantize the parameter space (*a,b*), i.e., divide it into cells.
  - Also need to select some minimum and maximum for *a* and *b*
- This quantized space is often referred to as the accumulator cells.
- Count the number of times a line intersects a given cell.

  => For each foreground pixel (*x,y*) in the binary image,
  find the values of (*a,b*) in the range $[[a_{min},a_{max}],[b_{min},b_{max}]]$
  defining the line corresponding to this point.
  - Increase the value of the accumulator for these (*a,b*) points.
  - Then proceed with the next point in the image.

- Cells receiving more than a given number of "votes"
  are assumed to correspond to lines in (*x,y*) space.

  => Lines in image space are detected as peaks in this accumulator space.

Implementing basic idea



Hough accumulator cells

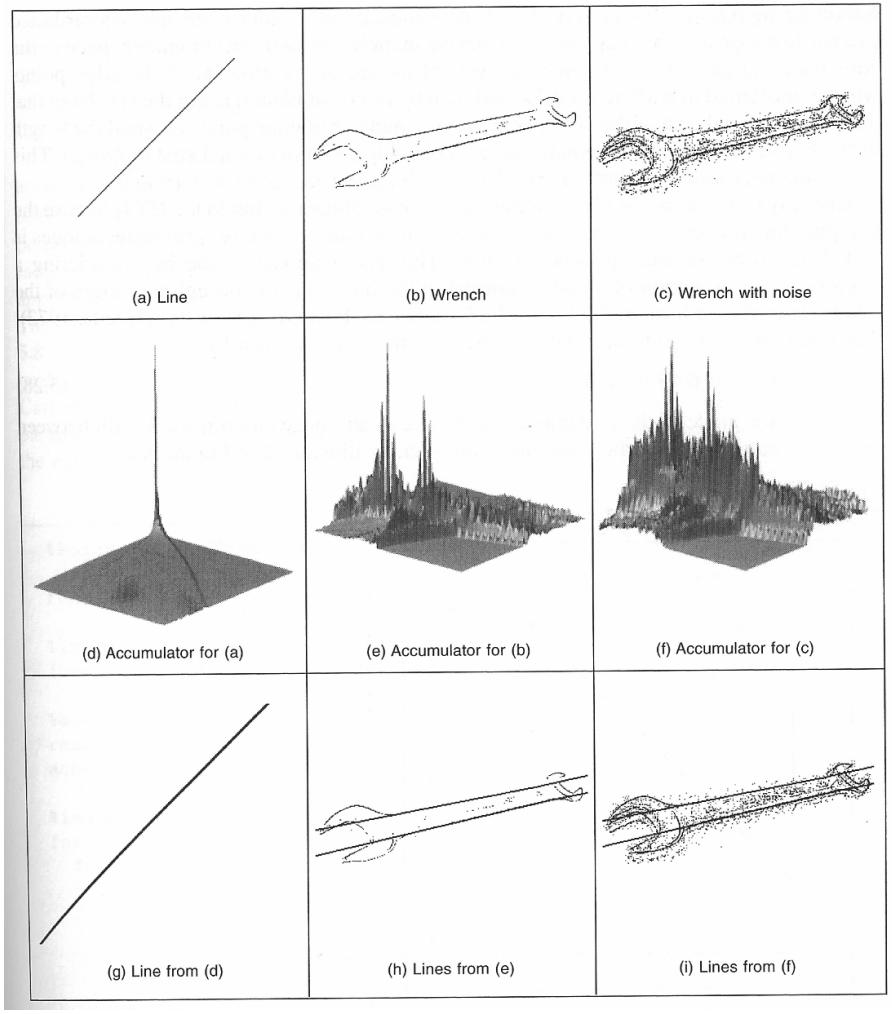
Example using basic idea

Thresholded edge images

Visualizing the accumulator space

The height of the peak will be defined by the number of pixels in the line.

Thresholding the accumulator space and superimposing corresponding lines onto the edge image



Note how noise affects the accumulator. Even with noise, the largest peaks correspond to the major lines.

Also note that points that lie approximately on a line in the (x,y) space will form a "cluster of crossings" in the (a,b) space.

Polar representation of lines

- In practice, we do not use the equation

$$y = ax + b$$

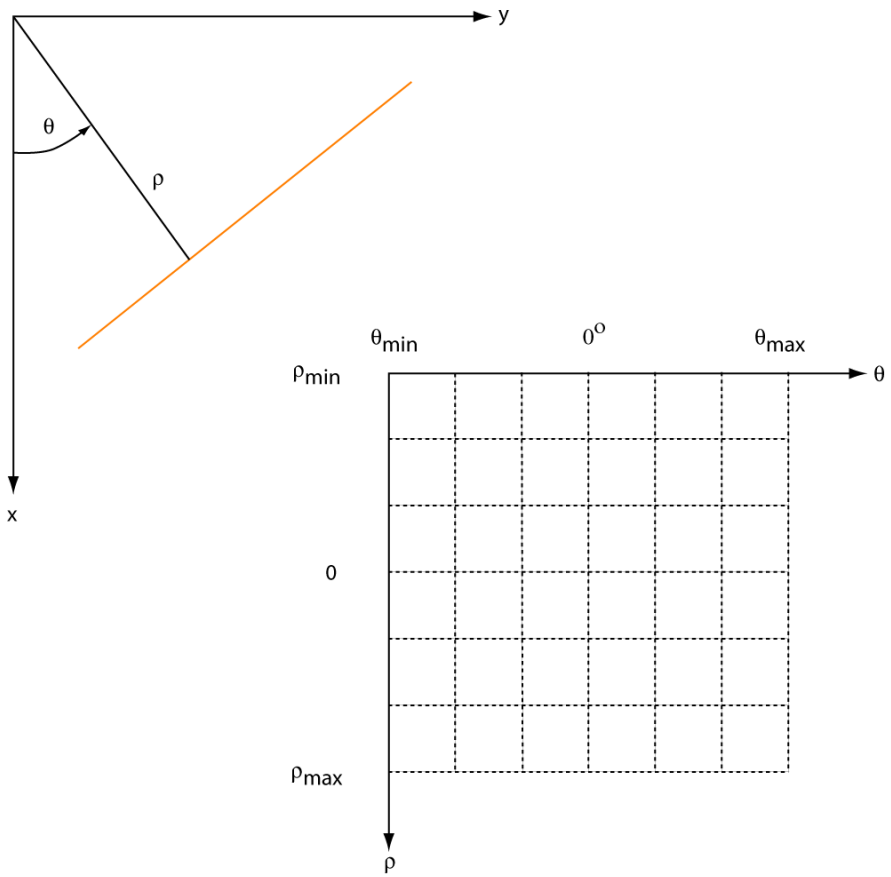
in order to represent lines (why?)

- Rather, we use the polar (also called normal) representation of lines:

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

  - where ρ is the length of the normal from origin to the line, and θ is the angle between this normal and the x axis.

HT and polar representation of lines



Polar representation of lines

Two points on a line

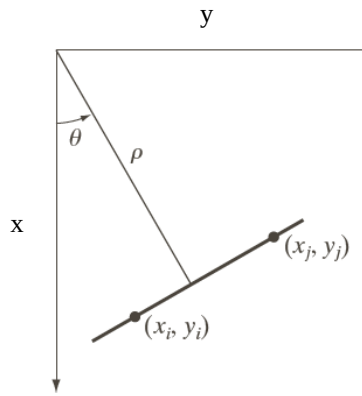
- Each point (x_i, y_i) in the (x,y) space gives a sinusoid in the (ρ,θ) space

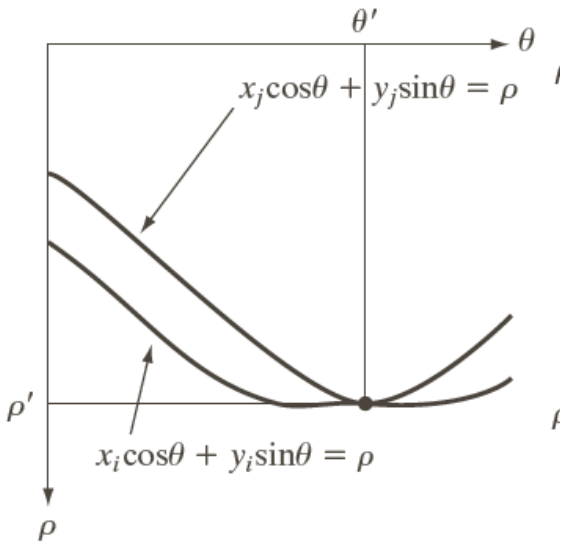
$$x_i \cos\theta + y_i \sin\theta = \rho$$

- Two points (x_i, y_i) and (x_j, y_j) on the same line in (x,y) space will intersect at the point (ρ', θ') in (ρ,θ) space which corresponds to the line satisfying both equations:

$$x_i \cos\theta' + y_i \sin\theta' = \rho'$$
$$x_j \cos\theta' + y_j \sin\theta' = \rho'$$

- Horizontal lines have θ =0° and ρ equal to the intercept with x axis.
- Vertical lines have θ =90° and ρ equal to the intercept with y axis.





Implementing line detector

- Partition the (ρ,θ) space into accumulator cells
  $A[\rho,\theta]$, $\rho \in [\rho_{min}, \rho_{max}]$; $\theta \in [\theta_{min}, \theta_{max}]$
- The range of θ is $\pm 90°$
  - Horizontal lines have $\theta=0°$, $\rho \geq 0$
  - Vertical lines have $\theta=90°$, $\rho \geq 0$
- The range of ρ is $\pm$ sqrt(M^2+N^2)
  if the image is of size MxN
- The discretization of θ and ρ must give acceptable precision and size of the parameter space.

Algorithm for detecting lines

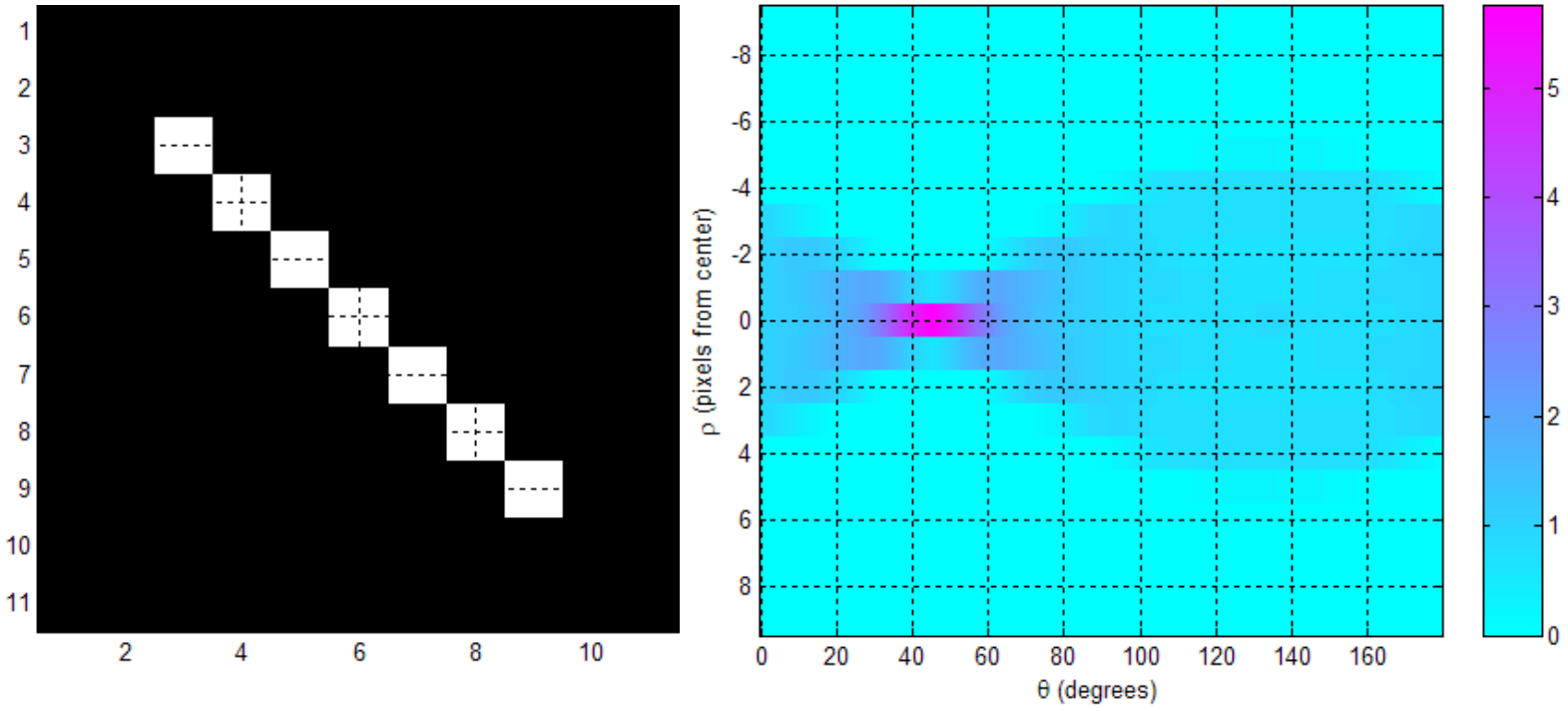
- Initialize an accumulator matrix A with zeros.
- For each foreground point (x_i,y_i) in the binary image:
  - For each possible θ value θ_j (e.g. -89, -88, …, 89, 90):
    - Calculate ρ using $\rho = x_i \cos \theta_j + y_i \sin \theta_j$
    - Round ρ to the closest cell value, ρ_q
    - Increment A(q,j), the element corresponding to parameters (ρ_q,θ_j)

- A(q,j)=P means that P points in (x,y) space lie on the line
$$\rho_q = x \cos \theta_j + y \sin \theta_j$$

- Find line candidates where A(q,j) peaks above a threshold.

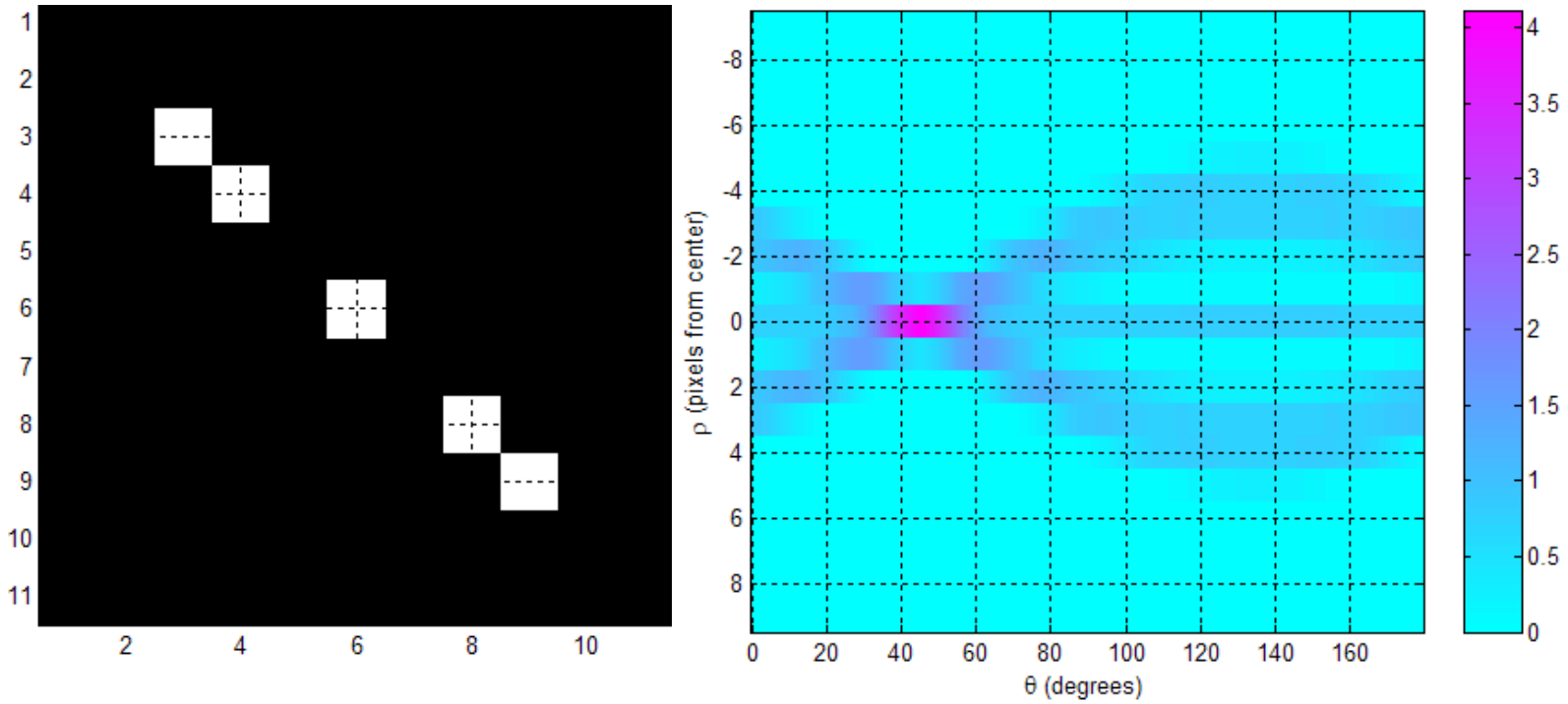
Line detection example 1

- Example 1: 11x11 image and its Hough transform:



Line detection example 2

- Example 2: 11x11 image and its Hough transform:



Line detection example 3

- Example 3: Natural scene & result of Sobel magnitude:





Line detection example 3

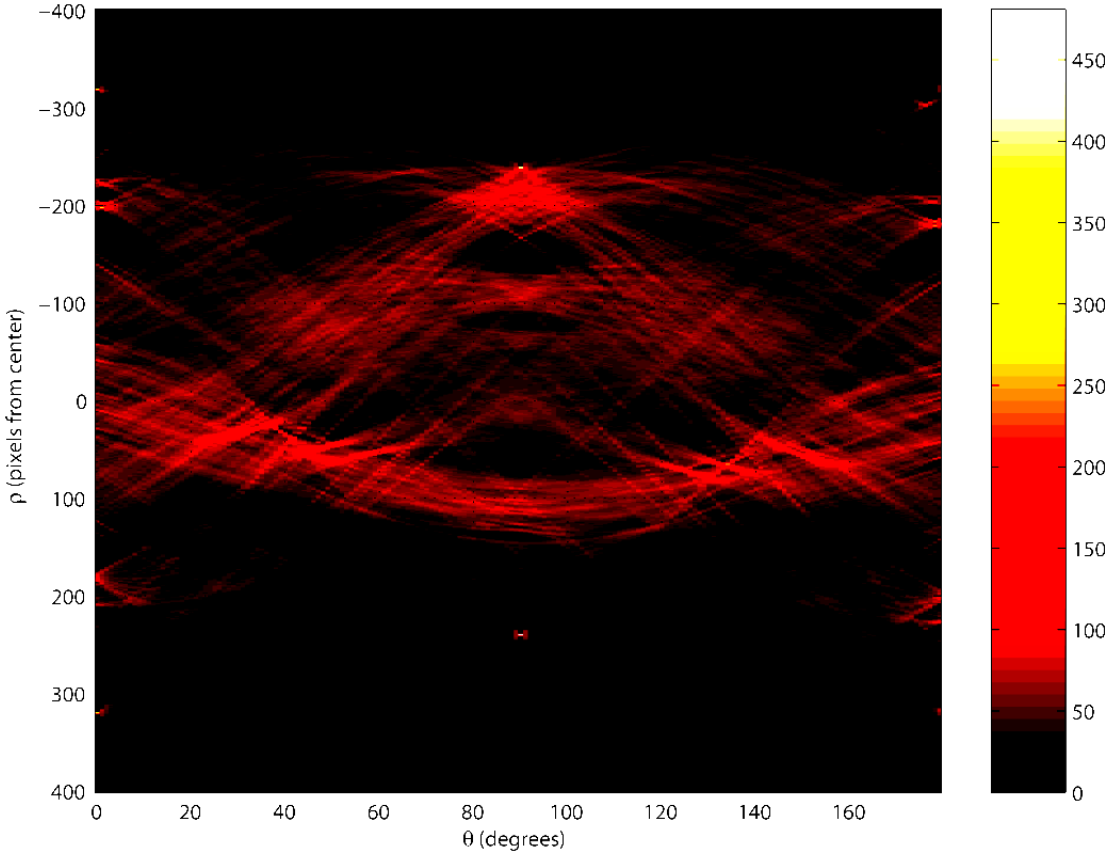
- Example 3: Natural scene and result of Sobel gradient magnitude followed by thresholding:





Line detection example 3

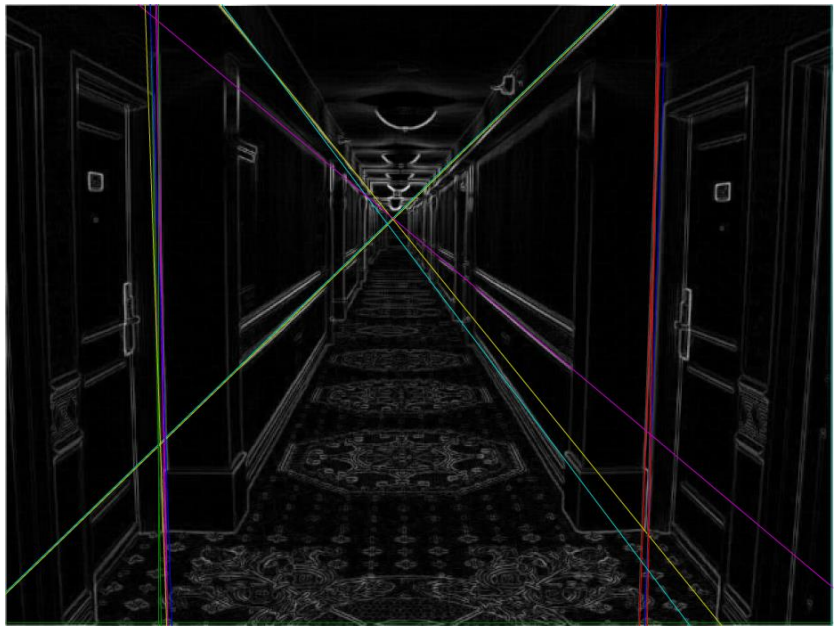
- Example 3: Accumulator matrix:



Line detection example 3

- Example 3: Original image (left) and 20 most prominent lines superimposed on gradient magnitude image (right):





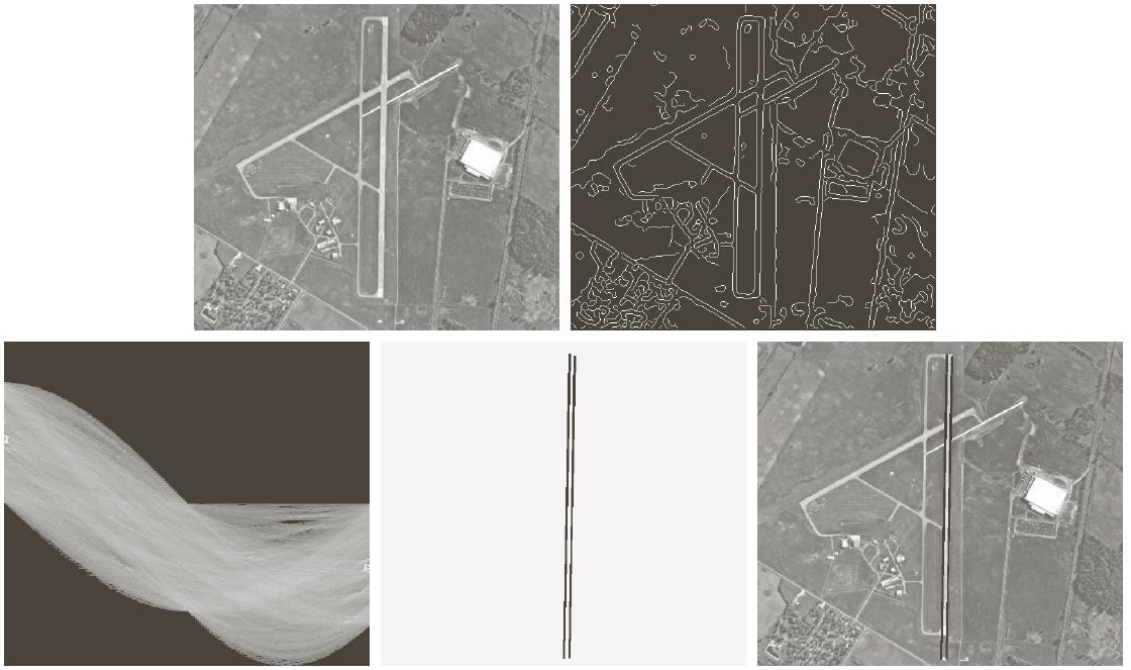
Pros and cons

- Advantages:
  - Conceptually simple.
  - Easy implementation.
  - Handles missing and occluded data very gracefully.
  - Can be adapted to several types of shapes, not just lines.

- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object.
  - Length and position of a line segment is not given directly.
    - Post-processing needed (example on next slide).
  - Co-linear line segments are not separated directly.
    - Can be "fooled" by "apparent lines".

Line segment detection example

1. Detect lines using Hough transform.
   - Peaks in the accumulator matrix above a threshold provides line candidates.
2. Examine pixels in the image for each line candidate:
   - Bridge gaps if sufficiently short.
   - Interpret large gaps as different line segments.
   - Remove small segments likely to be noise or other structures.



a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

HT using the full gradient information

- Given a gradient magnitude image g(x,y) containing a line.
- Standard algorithm:

  for all $g(x_i,y_i)>T$ do

    for all θ_j do

      $\rho = x_i \cos\theta_j + y_i \sin\theta_j$

      Round ρ to the closest cell value, ρ_q

      Increment A(q,j) where indexes (q,j) corresponding to (ρ_q, θ_j)

- In addition to the gradient magnitude g(x,y), we can normally compute the gradient direction: $\theta(x,y) = \tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$
- New algorithm:

  for all $g(x_i,y_i)>T$ do

    $\rho = x_i \cos(\theta(x,y)) + y_i \sin(\theta(x,y))$

    Round ρ to the closest cell value, ρ_q

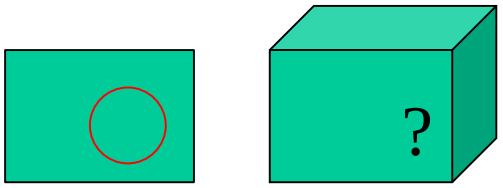
    Round $\theta(x,y)$ to the closest cell value, θ_j

    Increment A(q,j) where indexes (q,j) corresponding to (ρ_q, θ_j)

Hough transform for circles

- A circle in the (*x,y*) space is given by

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$



- So we have a 3D parameter space. What size, what resolution?

- **A simple 3D accumulation procedure:**

  set all $A[x_c,y_c,r]=0$;

  for every (x,y) where g(x,y)>T

  for all x_c

  for all y_c

  $r = \text{round}(\text{sqrt}((x-x_c)^2+(y-y_c)^2))$;

  $A[x_c,y_c,r] = A[x_c,y_c,r]+1$;

- **Better procedure(s)? 1 circle? Several circles?**

Hough transform for ellipses

- A general ellipse in the (x,y) space has 5 parameters:
  - Position of center (x_c, y_c), semi-axes (a,b), and orientation (θ).
- Thus, we have a 5D parameter space.
- For large images and full parameter resolution,
  straight forward HT may easily overwhelm your computer!

- **Reducing accumulator dimensionality (1):**
  - Pick pixel pairs with opposite gradient directions
  - Accumulate 2D histogram of mid-points of such pairs
  - Peak histogram locations are center candidates.
  - Reduces HT-accumulator from 5D to 3D.

  - Disadvantage: Fails if there are lots of occlusions.

Hough transform for ellipses

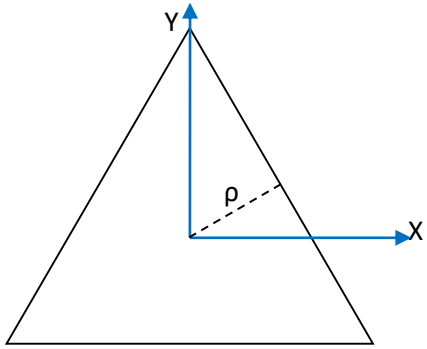
- **Reducing accumulator dimensionality (2):**
  - Pick pixel pair (P,Q) with non-parallel tangents.
  - Tangents intersect at a point T.
  - Let M be mid-point of PQ.
  - Line TM goes through ellipse center.
  - Accumulate intersection point of such lines to get (x_c, y_c).
  - Reduces HT-accumulator from 5D to 3D.

- **Reducing accumulator dimensionality (3):**
  - Pick pixel pair (P,Q) with orthogonal tangents.
  - Tangents intersect at «director circle» of ellipse, giving (x_c, y_c).

Hough transform of an equilateral triangle

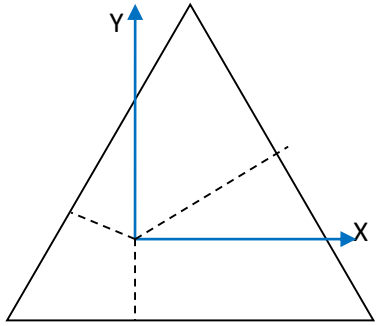
- The three HT peaks will be π/3 apart.
- The peaks will have the same height.

equilateral = equal side lengths



- If positioned so that the origin coincides with the center of mass of the triangle, the absolute value of ρ is also the same.

- If positioned so that the origin does not coincide with the center of mass of the triangle, the absolute value of ρ will differ (but their sum is the same as when the triangle was positioned so that the origin coincides with its center of mass).



- Similar is valid for all regular polygons.

Detecting a convex polygon using HT

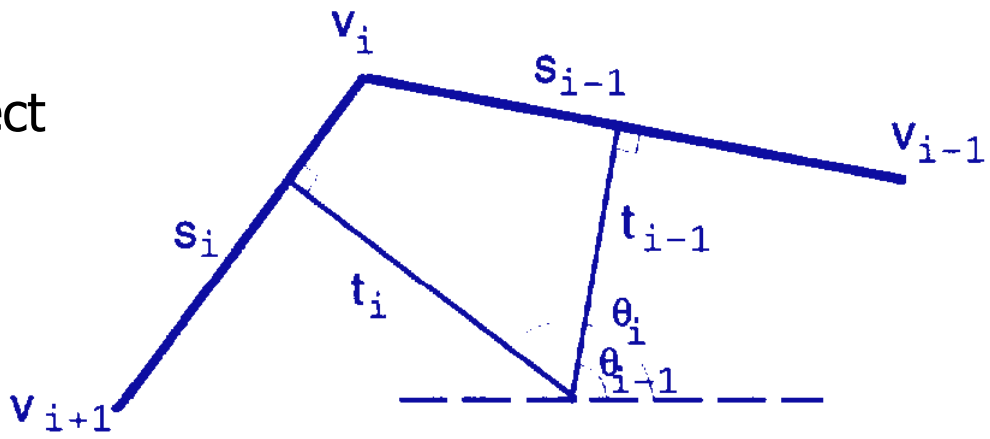
- First detect lines using HT.
- Find each vertex as the intersect between two lines s_{i-1} and s_i
  - All (x,y) on s_{i-1} satisfy:

    $t_{i-1} = x \cos(\theta_{i-1}) + y \sin(\theta_{i-1})$
  - All (x,y) on s_i satisfy:

    $t_i = x \cos(\theta_i) + y \sin(\theta_i)$
  - This can be written as:

$$\begin{bmatrix} t_{i-1} \\ t_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_{i-1}) & \sin(\theta_{i-1}) \\ \cos(\theta_i) & \sin(\theta_i) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

  - Solving this set of linear equations gives the vertex $v_i = [x_i, y_i]^T$
- Given $0 \le \theta_1 < \theta_2 \ldots < \theta_N \le 2\pi$,
  we can find every vertex and thereby reconstruct the polygon.



Learning goals

- Understand the basic idea of the Hough transform.
- Understand how to detect lines using HT.
  - Understand why polar representation is more suitable than the line representation with slope and intercept.
- Be able to implement line detection using HT.
  - Understand how this may be improved by gradient direction information.
- Be able to implement circle detection using HT.
  - Understand how this may be improved.
- Understand how to detect ellipses using HT.
  - Understand how this may be improved.