INF 4300 – Hough transform

Anne Solberg (anne@ifi.uio.no)

•This lecture goes more in detail than Gonzales and Woods 10.2.

•Introduction to Hough transform

• Using gradient information to detect lines

• Representing a line

• The [a,b]-representation

• The [ρ,θ]-representation

• Algorithm for detection of lines

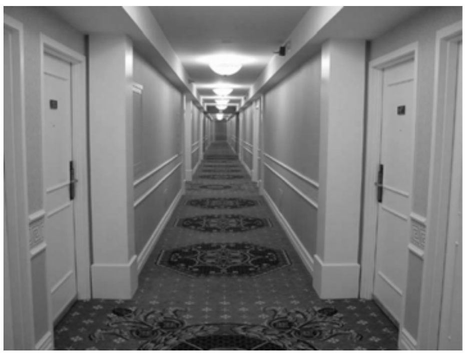
• Detecting circles

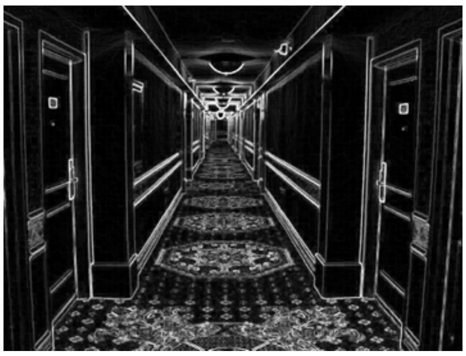
Introduction to Hough transform

- The Hough transform (HT) can be used to detect lines, circles or other parametric curves.
- It was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972).
- The goal is to find the location of lines in images.
- This problem could be solved by e.g. Morphology and a linear structuring element, or by correlation.
  - Then we would need to handle rotation, zoom, distortions etc.
- Hough transform can detect lines, circles and other structures if their parametric equation is known.
- It can give robust detection under noise and partial occlusion.

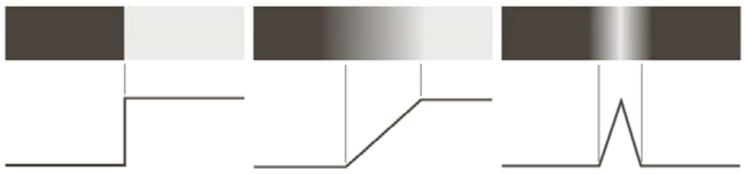
An image with linear structures

- Borders between the regions are straight lines.
- These lines separate regions with different grey levels.
- Edge detection is often used as preprocessing to Hough transform.





Remember edge types?



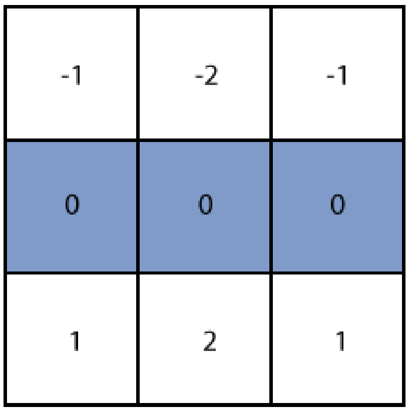
a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

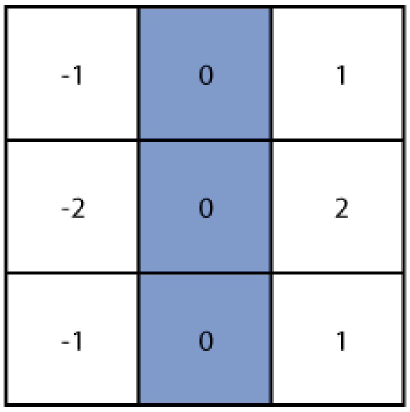
Step edge Ramp edge Roof edge

Hough-transform – the input

- The input image must be a thresholded edge image.
- The magnitude results computed by the Sobel operator can be thresholded and used as input.



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



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

Repetition - Basic edge detection

- A thresholded edge image is the starting point for Hough transform.
- What does a Sobel filter produce?
- Approximation to the image gradient:

$$\nabla f(x)$$

- ...which is a vector quantity given by:

$$\nabla \mathbf{f}(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Repetition – Edge magnitude

- The gradient is a measure of how the function f(x,y) changes as a function of changes in the arguments x and y.
- The gradient vector points in the direction of maximum change.
- The length of this vector indicates the size of the gradient:

$$\nabla f = |\nabla \mathbf{f}| = \sqrt{G_x^2 + G_y^2}$$

G_x, G_y and the gradient operator

- Horisontal edges:
  - Compute $g_x(x,y)=H_x*f(x,y)$
  - Convolve with the horisontal filter kernel H_x
- Vertical edges:
  - Compute $g_t(x,y)=H_y*f(x,y)$
- Compute the gradient operator as:

$$g(x,y)=\sqrt{g_x^2(x,y)+g_y^2(x,y)}$$ Gradient-magnitude (kant-styrke)

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

Repetition – Edge direction

- The direction of this vector is also an important quantity.
- If α(x,y) is the direction of f in the point (x,y) then:

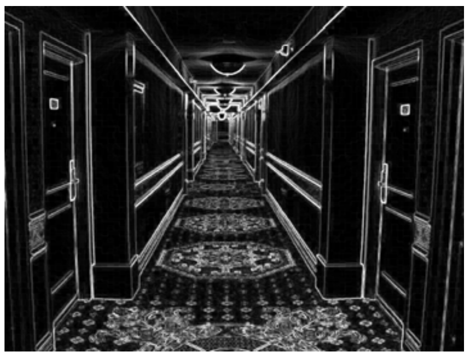
$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

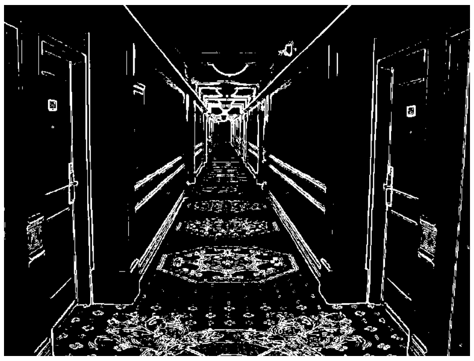
- Remember that α(x,y) will be the angle with respect to the x-axis
- Remember also that the direction of an edge will be perpendicular to the gradient in any given point

Input to Hough – thresholded edge image

Prior to applying Hough transform:

- Compute edge magnitude from input image.
- As always with edge detection, simple lowpass filtering can be applied first.
- Threshold the gradient magnitude image.



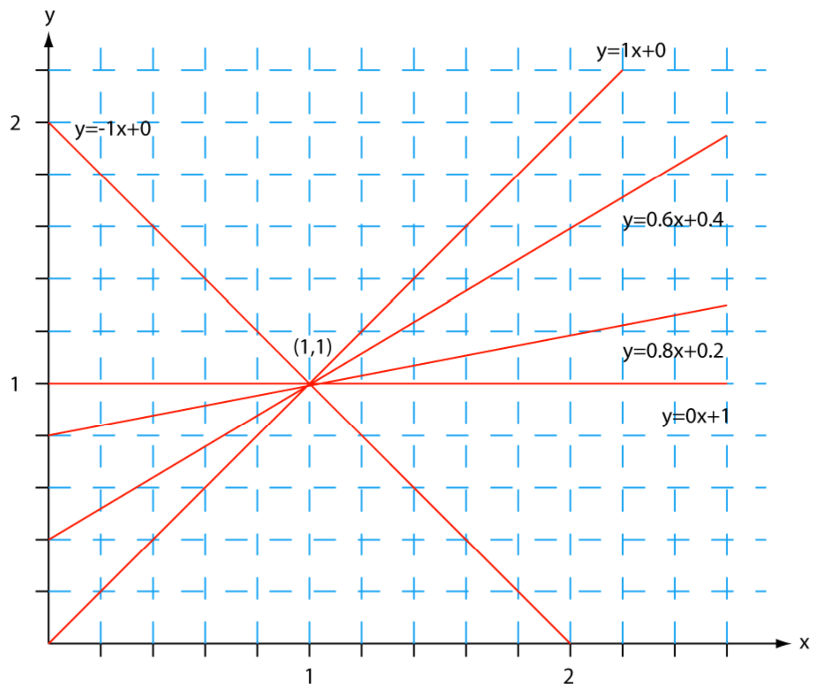


Hough-transform

- Assume that we have performed some edge detection, and a thresholding of the edge magnitude image.
- Thus, we have *n* pixels that may partially describe the boundary of some objects.
- We wish to find sets of pixels that make up straight lines.
- Regard a point $(x_i; y_i)$

  and a straight line $y_i = ax_i + b$
  - There are many lines passing through the point (x_i,y_i).
  - Common to them is that they satisfy the equation for some set of parameters (a,b).

Hough transform – basic idea



Hough transform – basic idea

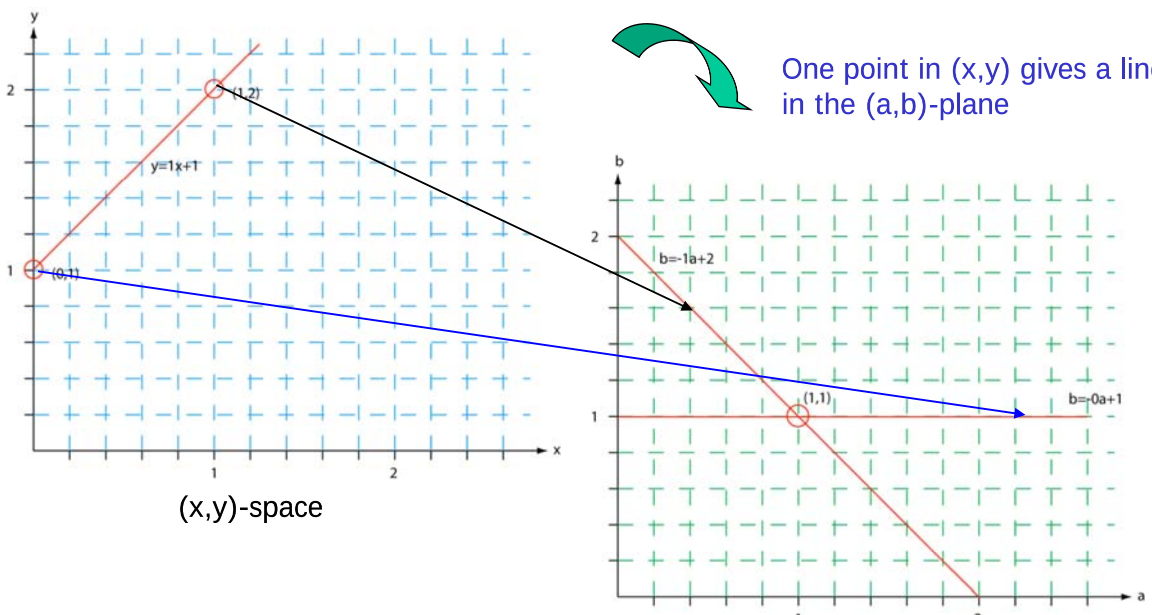
- This equation can obviously be rewritten as follows:

$$b = -xa + y$$

- We now consider *x* and *y* as parameters and *a* and *b* as variables.
- This is a line in (*a,b*) space parameterized by *x* and *y*.
  - So: a single point in xy-space gives a line in (a,b) space.
- Another point (*x,y*) will give rise to another line in (*a,b*) space.

Hough transform – basic idea

One point in (x,y) gives a line in the (a,b)-plane



(x,y)-space

(a,b)-space

Hough transform – basic idea

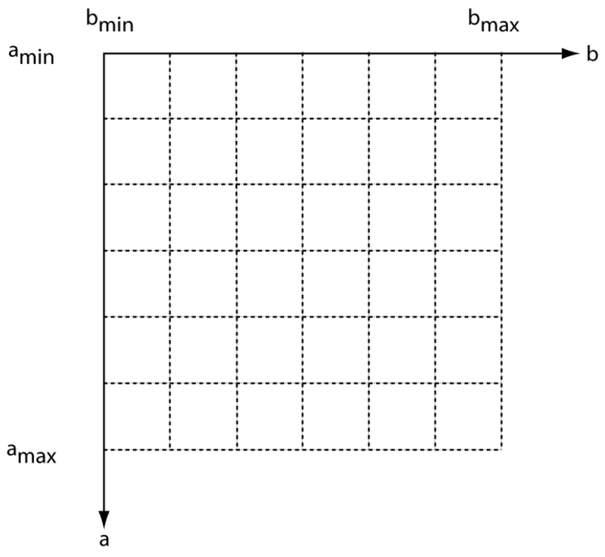
- Two points (x,y) and(z,k) define a line in the (x,y) plane.
- These two points give rise to two different lines in (a,b) space.
- In (a,b) space these lines will intersect in a point (a',b') where a' is the rise and b' the intersect of the line defined by (x,y) and (z,k) in (x,y) space.
- The fact is that all points on the line defined by (x,y) and (z,k) in (x,y) space will parameterize lines that intersect in (a',b') in (a,b) space.
- Points that lie on a line will form a "cluster of crossings" in the (a,b) space.

Hough transform – algorithm

- Quantize the parameter space (a,b), that is, divide it into cells.
- This quantized space is often referred to as the accumulator cells.
- In the figure in the next slide a_{min} is the minimal value of a etc.
- Count the number of times a line intersects a given cell.
  - For each point (x,y) with value 1 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'] point.
  - Then proceed with the next point in the image.

- Cells receiving a minimum number of "votes" are assumed to correspond to lines in (x,y) space.
  - Lines can be found as peaks in this accumulator space.

Hough transform - algorithm



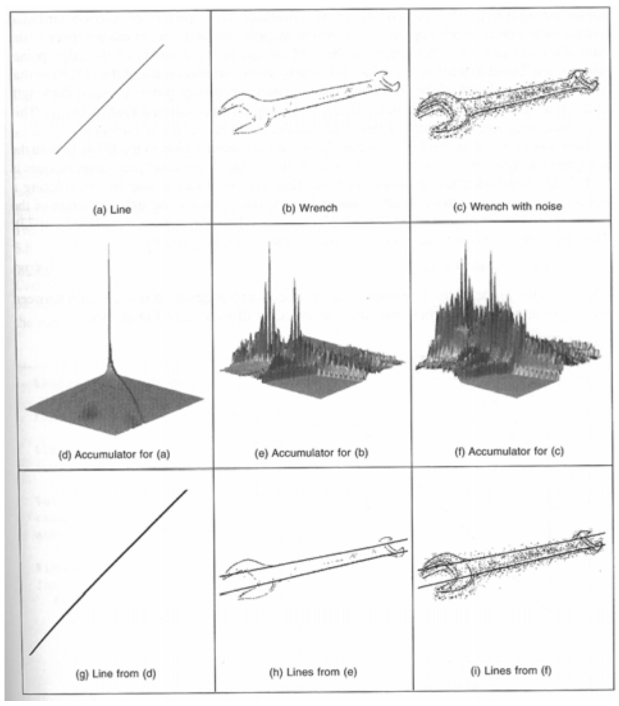
Hough accumulator cells

Example – images and accumulator space

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 this onto the edge image



Note how noise effects the accumulator. Still with noise, the largest peaks correspond to the major lines.

Hough transform – polar representation of lines

- In practical life we do not use the equation

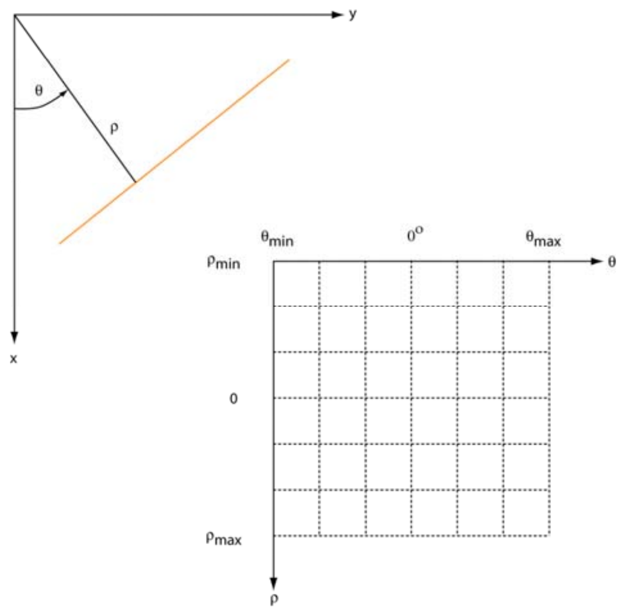
$$y = ax + b$$

in order to represent lines (why?)

- Rather, we use the polar representation of lines:

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

Hough transform – polar representation of lines



Polar representation of lines

Hough transform and the polar representation

- The polar (also called normal) representation of straight lines is

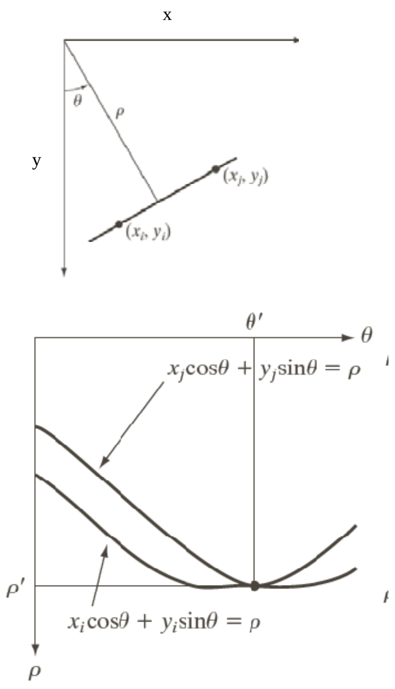
$$x\cos\theta + y\sin\theta = \rho$$

- Each point (x_i,y_i) in the xy-plane gives a sinusoid in the $\rho\theta$-plane.
- M colinear point lying on the line

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

will give M curves that intersect at (ρ_i,θ_j) in the parameter plane.
- Local maxima give significant lines.

- Each curve in the figure represents the familiy of lines that pass through a particular point (x_i,y_i) in the xy-plane.
- The intersection point (ρ',θ') corresponds to the lines that passes through two points (x_i,y_i) and (x_j,y_j)
- A horizontal line will have θ=0 and ρ equal to the intercept with the y-axis.
- A vertical line will have θ=90 and ρ equal to the intercept with the x-axis.



Matlab demo

- From (x,y) to (ρ_i,θ_j)

Hough transform - algorithm using polar representation of lines

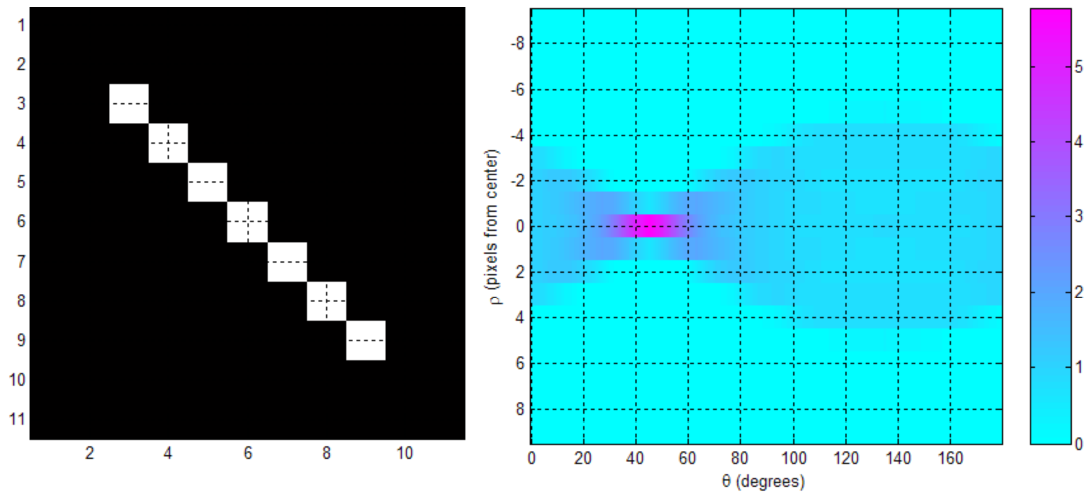
- Partition the $\rho\theta$-plane into accumulator cells $A[\rho,\theta]$, $\rho\in[\rho_{min}, \rho_{max}]$; $\theta\in[\theta_{min}, \theta_{max}]$
- The range of θ is $\pm90°$
  - Horizontal lines have $\theta=0°$, $\rho\geq0$
  - Vertical lines have $\theta=90°$, $\rho\geq0$
- The range of ρ is $\pm N\sqrt{2}$ if the image is of size NxN
- The discretization of θ and ρ must happen with values $\delta\theta$ and $\delta\rho$ giving acceptable precision and sizes of the paramter space.

Algorithm continued

- The cell (i,j) corresponds to the square associated with parameter values (θ_j, ρ_i).
- Initialize all cells with value 0.
- For each foreground point (x_k,y_k) in the thresholded edge image
  - Let θ_j equal all the possible θ-values
    - Solve for ρ using ρ=x cos θ_j +ysin θ_j
    - Round ρ to the closest cell value, ρ_q
    - Increment A(i,q) if the θ_j results in ρ_q
- After this procedure, A(i,j)=P means that P points in the xy-space lie on the line ρ_j=x cos θ_j +ysin θ_j
- Find line candiates where A(i,j) is above a suitable threshold value.

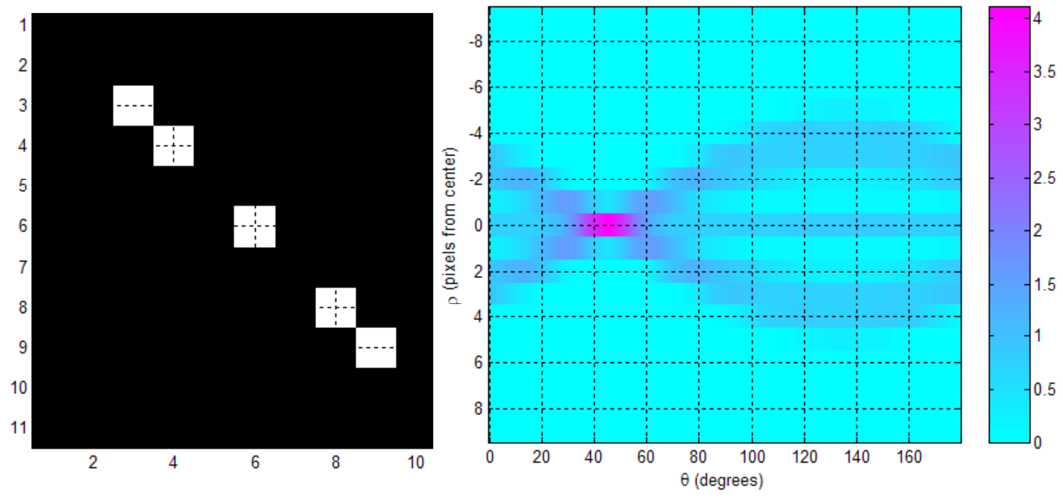
Hough transform – example 1

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



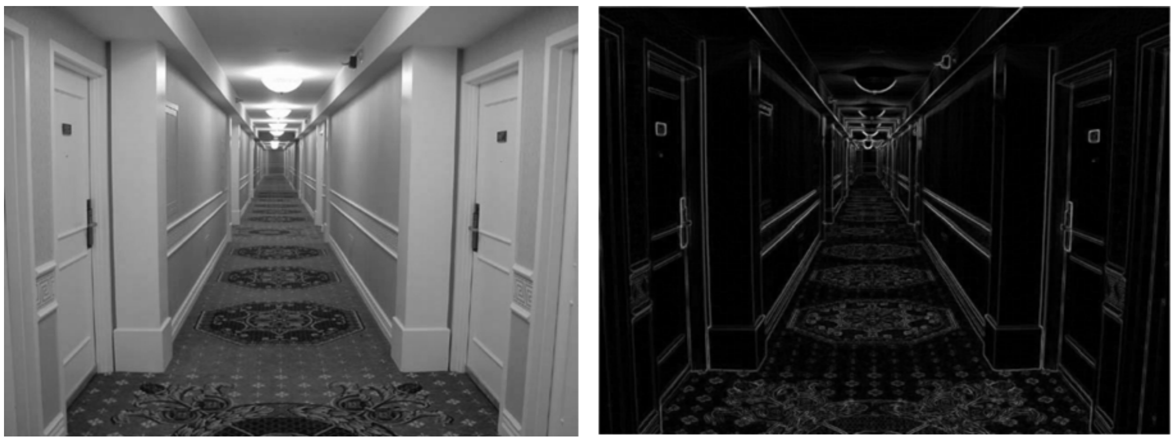
Hough transform – example 2

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



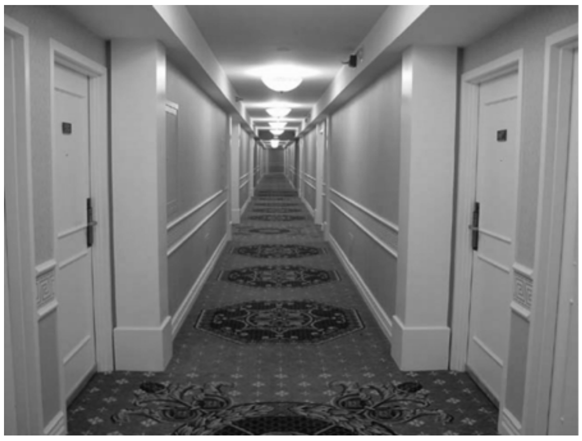
Hough transform – example 3

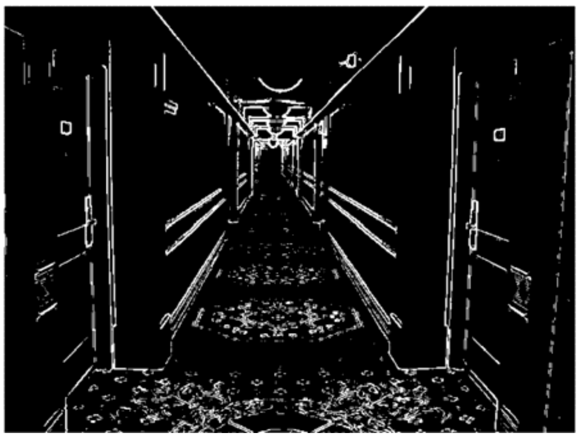
- Example 3: Natural scene and result of Sobel edge detection:



Hough transform – example 3

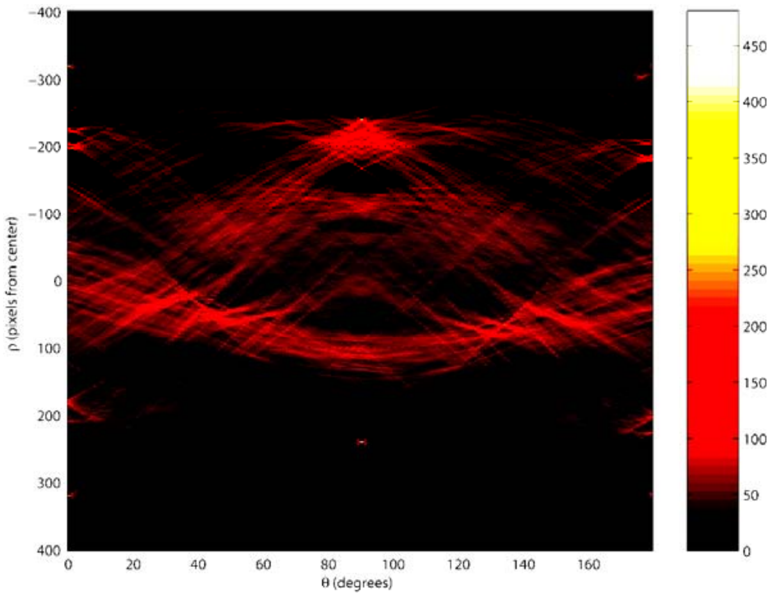
- Example 3: Natural scene and result of Sobel edge detection followed by thresholding:





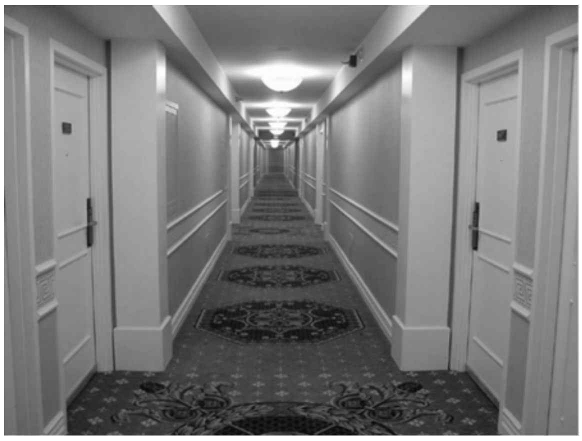
Hough transform – example 3

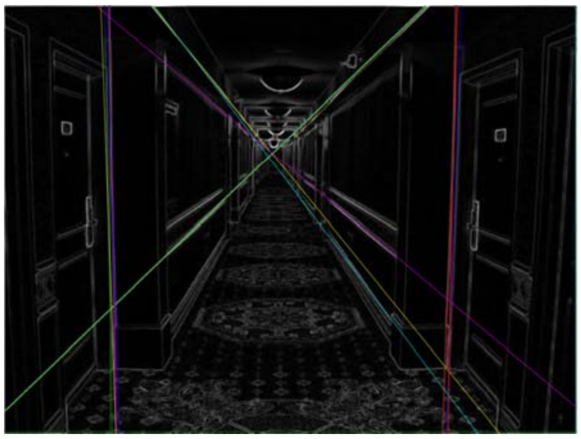
- Example 3: Accumulator matrix:



Hough transform – example 3

- Example 3: Original image and 20 most prominent lines:





Hough transform - advantages

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

Hough transform - disadvantages

- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object.
  - Can be “fooled” by “apparent lines”.
  - The length and the position of a line segment cannot be determined.
  - Co-linear line segments cannot be separated.

Hough-transform using the full gradient information – a variant

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

```
for all $g(x_i,y_i)>T$ do
    for all $\theta$ do
        $\rho = x_i \cos\theta + y_i \sin\theta$
        find corresponding indexes (m,n) and increment A(m,n)
```

- Better algorithm if we have both
  - The gradient magnitude g(x,y)
  - The gradient components g_x and g_y and can compute

$$\phi_g(x,y) = \arctan\left(\frac{g_x}{g_y}\right)$$

- Algorithm:

```
for all $g(x_i,y_i)>T$ do
    for all $\theta$ do
        $\rho = x_i \cos(\phi_g(x,y)) + y_i \sin(\phi_g(x,y))$
        find index m corresponding to $\rho$ and increment $A(m, \phi_g(x,y))$
```

Hough transform and edge linking

- How can we look for lines with a certain orientation?

1. Obtain a thresholded edge image
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.
   - Continuity here normally means distance between disconnected pixels. A gap in the line can be bridged if the length of the gap is less than a certain threshold.

Using edge linking

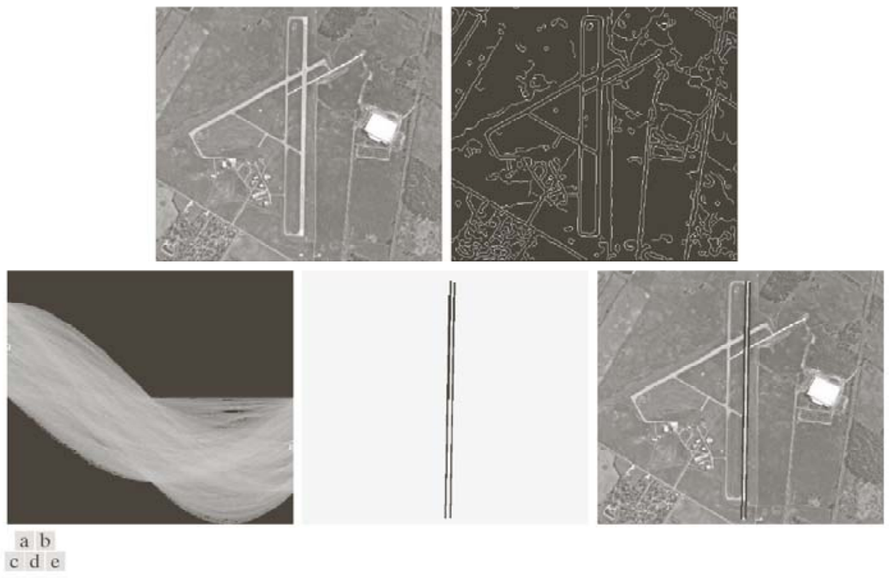


FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed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

Hough transform for circles

- A circle in the xy-plane is given by

$$(x-a)^2 + (y-b)^2 = c^2$$

- So we have a 3D parameter space.
- Simple procedure:

```
set all A[a,b,c]=0
for every (x,y) where g(x,y)>T
        for all a and b
            c=sqrt((x-a)^2+(y-b)^2);
            A[a,b,c] = A[a,b,c]+1;
```

Hough transform – exercise 1

- Familiarize with Matlab function for line detection:
  - Functions hough(), houghpeaks(), and houghlines()

- Next exercise:
  - Test Hough transform for equal size circles on the coins image.

Hough transform – exercise 2

- Next exercise: The randomized Hough transform.
  - Simple idea (line case): From the edge image, pick two points.
  - Find the ρ and θ corresponding to this set of points.
  - Increment the indicated (ρ,θ) cell.
  - Once a cell reaches a certain (low) count, assume that an edge is present in the image.
  - Verify this.
  - If truly present, erase this line from the image
  - Continue until no more points or until the number of iterations between two detections is to high.
  - Orders of magnitude faster than the ordinary transform.