

## Canny Edge Detection and Hough Transform

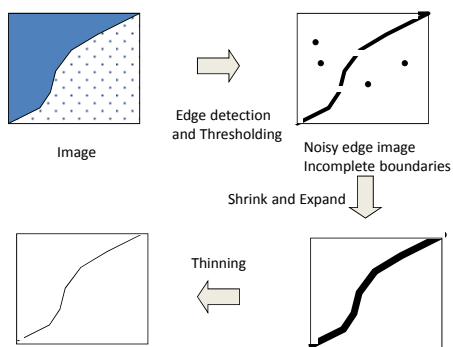
Chen Yu  
Indiana University

Adapted from Pietro Perona

### Optimal edge detection (Canny, 1986)

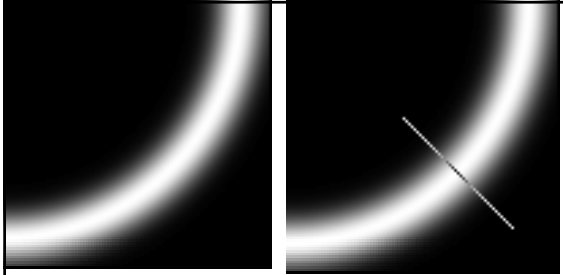
- Good Detection. Filter responds to edge, not noise.
- Good Localization: detected edge near true edge.
- Number of responses: One real edge should not result in more than one detected edge

### Edge Images



### Canny Edge Detector

- 1) **Smooth** image with a Gaussian
  - optimizes the trade-off between noise filtering and edge localization
- 2) Compute the Gradient **magnitude** using approximations of **partial derivatives**
- 3) Thin edges by applying **non-maxima suppression** to the gradient magnitude
- 4) Detect edges by **double thresholding**



**Non-maximum suppression:**  
Select the single maximum point across the width of an edge.

### Thinning

0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

### Non-Maxima Suppression


0	0	0	0	1	1	1	3
3	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	2
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

*false edges* (pointing to the left edge) and *gaps* (pointing to the right edge) are labeled with yellow arrows.

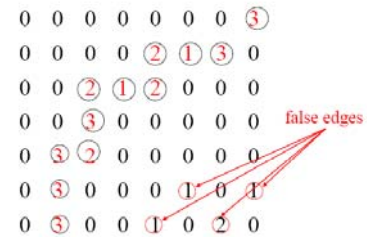
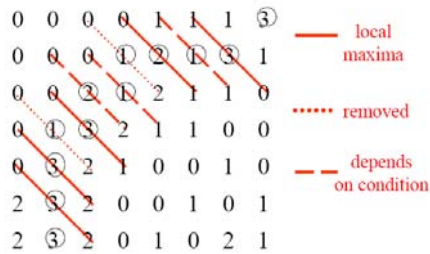
- Thin the broad ridges in  $M[i,j]$  into ridges that are **only one pixel wide**
- Find local maxima in  $M[i,j]$  by suppressing all values along the line of the Gradient that are not peak values of the ridge

### Gradient Orientation

- Reduce angle of Gradient  $\theta[i,j]$  to one of the 4 sectors
- Check the 3x3 region of each  $M[i,j]$
- If the value **at the center** is not greater than the 2 values along the gradient, then  $M[i,j]$  is set to 0



### Pixel Selection



- The suppressed magnitude image will contain many false edges caused by noise or fine texture

### Hysteresis Thresholding

- Two Issues with one threshold:
  - Allows noisy maxima to breakthrough if too low
  - Kills off weak edges if too high
- Solution: Exploit the fact that edge pixels are normally connected
  1. Set 2 thresholds,  $t_{low}$  and  $t_{high}$
  2. If a pixel  $I_N(i,j)$  is  $\geq t_{high}$ , mark it as an edge
  3. Check its neighbors along the direction  $\theta(i,j)$ . If they are greater than  $t_{low}$ , mark them as edges
- Allows a higher threshold to kill off noise, while retaining valid contours from the image

### Hysteresis Thresholding

- Idea: use a high threshold to **start** edge curves and a low threshold to **continue** them.
- **Start** with a high threshold (strong edge).
- Apply another threshold which is lower than the higher one.
- Look for the other edges which are higher than the lower threshold and **connected** to the strong edges
- Add them to the edge list

**Linking to the next edge point**

Assume the marked point is an edge point.

Take the normal to the gradient at that point and use this to predict continuation points (either r or s).

Very strong edge response. Let's start here

Weaker response but it is connected to a confirmed edge point. Let's keep it.

Continue....

Note: Darker squares illustrate stronger edge response (larger  $M$ )

**Example: Canny Edge Detection**

Original image

Strong + connected weak edges

Strong edges only

Weak edges

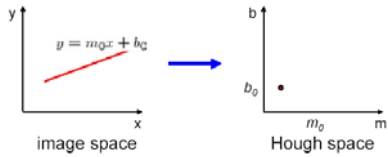
gap is gone

courtesy of G. Loy

**Hough Transform**

- Elegant method for edge detection and 2D object recognition
  - Edges need **not** be connected
  - Complete object need not be visible
  - Key Idea: Edges VOTE for the possible model

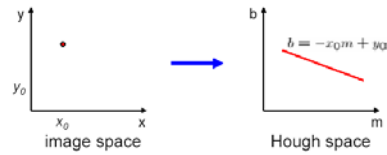
### Image space and Hough space



Connection between image (x,y) and Hough (m,b) spaces  
 • A line in the image corresponds to a point in Hough space

### Image Space and Hough Space

• What does a point  $(x_0, y_0)$  in the image space map to?



To go from image space to Hough space:  
 - given a set of points (x,y), find all (m,b) such that  $y = mx + b$

### Image and Parameter Spaces

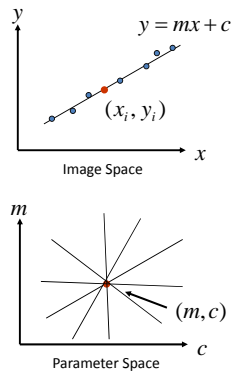
Equation of Line:  $y = mx + c$

Find:  $(m, c)$

Consider point:  $(x_i, y_i)$

$y_i = mx_i + c$  or  $c = -x_i m + y_i$

Parameter space also called Hough Space



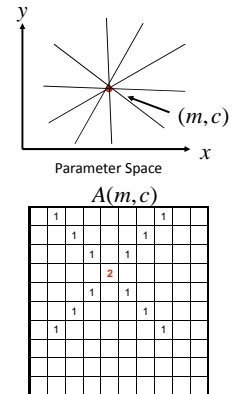
### Line Detection by Hough Transform

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  $(x_i, y_i)$  increment:  

$$A(m, c) = A(m, c) + 1$$
- If  $(m, c)$  lies on the line:  

$$c = -x_i m + y_i$$
- Find local maxima in  $A(m, c)$



### Better Parameterization

NOTE:  $-\infty \leq m \leq \infty$

Large Accumulator (e.g. vertical lines)

More memory and computations

Improvement: (Finite Accumulator Array Size)

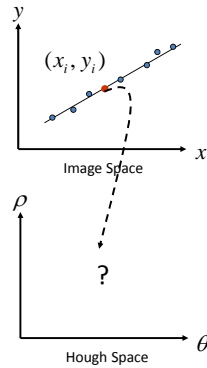
Line equation:  $\rho = x \cos \theta + y \sin \theta$

Here  $0 \leq \theta \leq 2\pi$

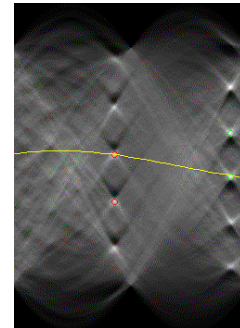
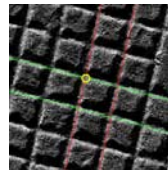
$0 \leq \rho \leq \rho_{\max}$

Given points  $(x_i, y_i)$  find  $(\rho, \theta)$

Hough Space Sinusoid

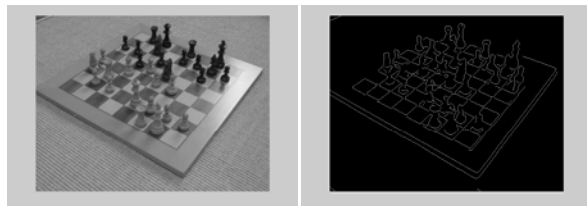


### Example




### Mechanics of the Hough transform


- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
  - How many lines?
    - Count the peaks in the Hough array
    - Treat adjacent peaks as a single peak



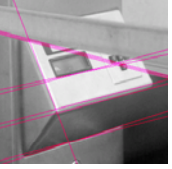
### Real World Example



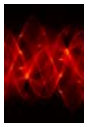
Original



Edge Detection



Found Lines



Parameter Space

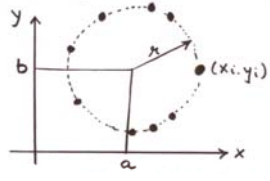
### Finding Circles by Hough Transform

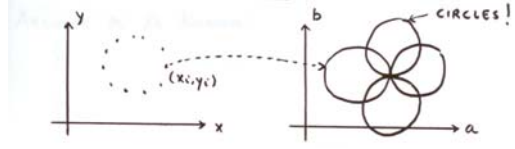
Equation of Circle:

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

If radius is known: (2D Hough Space)

Accumulator Array  $A(a, b)$





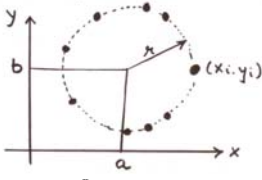
### Finding Circles by Hough Transform

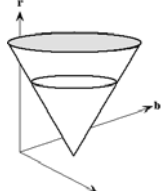
Equation of Circle:

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

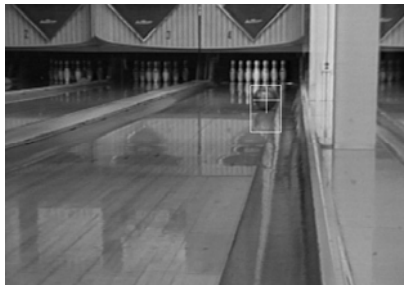
If radius is not known: 3D Hough Space!  
Use Accumulator array  $A(a, b, r)$

What is the surface in the Hough space?





### Real World Circle Examples



Crosshair indicates results of Hough transform.

