UNIVERSIDAD PANAMERICANA

Mecatrónica y Control de Sistemas

Segmentation

Introduction

What is segmentation?

It consists of dividing an image in regions or objects. It is based on two concepts:

-Discontinuities: Partition of an image based in abrupt changes of intensity (example: contours)

-Similarities: Partition of an image into regions that meet a pre-established criterion

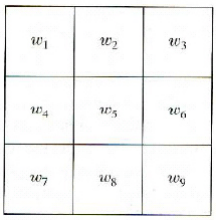
Types of discontinuities

-Points

-Lines

-Borders

Use of masks →



$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

Detection of points and lines

Points

−1	−1	−1
−1	8	−1
−1	−1	−1

$|R| \geq T$

↓

Threshold

Idea: To detect a point by checking the differences with its neighbors. If the pixel is very different, then it is an isolated point.

Lines

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

−1	−1	2
−1	2	−1
2	−1	−1

+45°

2	−1	−1
−1	2	−1
−1	−1	2

−45°

Contour detection

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

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}$$

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

Simplifying:

$$\nabla f \approx G_x^2 + G_y^2$$

$$\nabla f \approx |G_x| + |G_y|$$

Masks: Gx Gy

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

−1	0
0	1

0	−1
1	0

Roberts

Contour detection

Gx

0	−1	0
−1	4	−1
0	−1	0

Gy

−1	−1	−1
−1	8	−1
−1	−1	−1

Laplacian

Masks: Diagonals

0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

Canny: The most powerful for line detection

1. Gaussian filter for noise reduction
2. Calculates the gradient and direction
3. Threshold for eliminating weak pixels
4. Retains the pixels in 8-A

Contour (edge) detection

```
[g, t] = edge(f, 'method', parameters)
```

Output

Threshold used (optional)

Input

Sobel
Prewitt
Roberts
Laplacian of a Gaussian (LoG)
Zero crossings
Canny

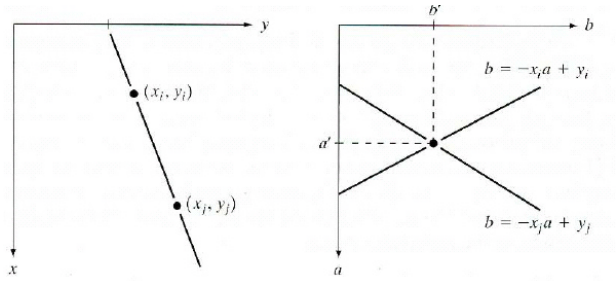
Horizontal
Vertical
Both (default)

Hough Transform

Problem of the methods aforementioned: Sensitive to noise, lines cut, discontinuities due to illumination, etc.

Hough Transform: line detection + robustness

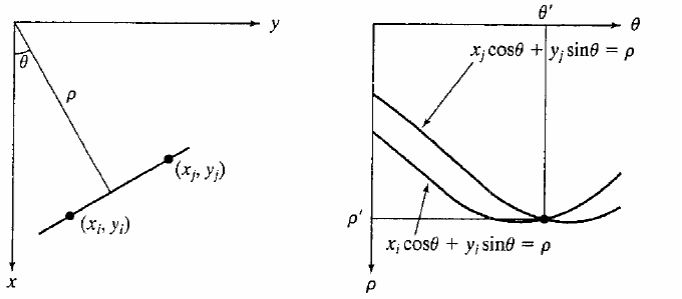
Principle



2 planes: *xy* plane
ab plane (plane of parameters)

Hough Transform

If a → ∞ $x\cos\theta + y\sin\theta = \rho.$



If θ=0, p>0 (intercepts x)
If θ=90, p>0 (intercepts y)
If θ=-90, p<0 (intercepts -y)

Matlab: [H,theta,rho]=hough(f,dtheta,drho);

Spacing

Hough Transform-Application

1. Line detection using any method
2. Calculate the Hough Transform
3. Detect the most important intersections (peaks)

```
[r,c]=houghpeaks(H,numpeaks);
```

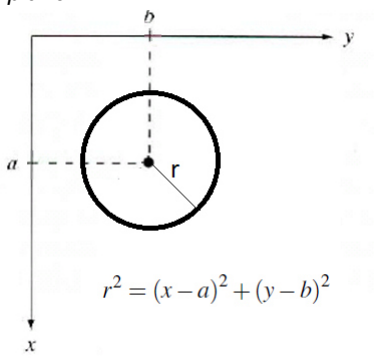
4. Reconstruct the lines from the peaks

```
lines=houghlines(f,theta,rho,r,c);
```

Circular Hough Transform

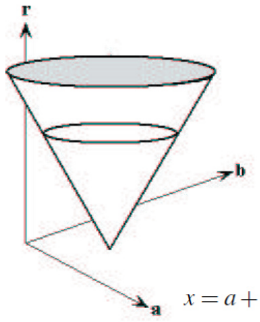
Objective: Circle detection with robustness

xy plane



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

ab plane

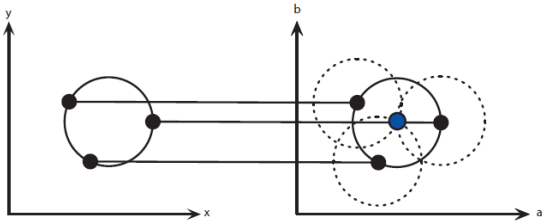


$$x = a + r\cos(\theta)$$
$$y = b + r\sin(\theta)$$

4 parameters ($\mathbf{R}^3$) → r constant

Circular Hough Transform

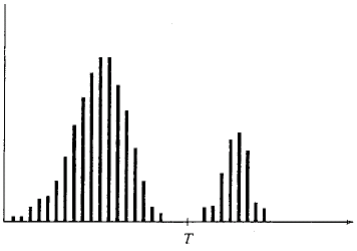
Principle



Application:

1. Determine the radius (range or unique value) of the circles to be found
2. Contour (edge) detection by any method
3. Obtain the Circle Hough Transform (*circle_hough*)
4. Detect the most important intersections or peaks (*circle_houghpeaks*)
5. Reconstruct the circles from the Circular Hough Transform peaks (*circlepoints*)

Binarization by Threshold Selection



$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{if } f(x,y) < T \end{cases}$$

Object

Background

T=graythresh(f);

1- Method insensitive to noise (except for salt & pepper)

2.- Sensitive to illumination

Region segmentation

T1			Tn
			Tm

Segmentation – Watershed Transform

In geography, *watershed* is the line that divides two land areas with water
Base: distance transform

1	1	0	0	0	0.00	0.00	1.00	2.00	3.00
1	1	0	0	0	0.00	0.00	1.00	2.00	3.00
0	0	0	0	0	1.00	1.00	1.41	2.00	2.24
0	0	0	0	0	1.41	1.00	1.00	1.00	1.41
0	1	1	1	0	1.00	0.00	0.00	0.00	1.00

```
D = bwdist(f)
```

Distance of a pixel to the closest non-zero pixel

Algorithm:

-Binarization: Black object in white background

-Obtain the distance transform

-Obtain the watershed (from the negative of the distance transform)

L=watershed(~D)

-Post-processing