# **Binary Machine Vision**

## Introduction

- binary value 1: considered part of object
- binary value 0: background pixel
- binary machine vision: generation and analysis of binary image



# Thresholding

- B(r,c) = 1 if  $I(r,c) \ge T$
- B(r,c) = 0 if I(r,c) < T
- *r*: row, *c*: column
- *I*: grayscale intensity, *B*: binary intensity
- *T*: intensity threshold

# Thresholding



#### how to choose T?

#### Histogram Analysis for Image Thresholding



#### Histogram Analysis for Image Thresholding



# Thresholding

- Histogram shape can be useful in locating the threshold.
  - However it is not reliable for <u>threshold selection</u> when peaks are not clearly resolved.
- Choosing a threshold in the valley between two overlapping peaks, and inevitably some pixels will be incorrectly classified by the thresholding.





The determination of peaks and valleys is a non-trivial problem

# Sensitivity to object size



# Otsu Thresholding

Select a threshold that minimize the within-class variance







# Otsu Thresholding

• define the *within-class* variance as the weighted sum of the variances of each cluster:

$$\sigma^2_{\text{Within}}(T) \neq n_B(T)\sigma^2_B(T) + n_O(T)\sigma^2_O(T)$$

T = 1

Minimize

$$n_B(T) = \sum_{i=0}^{N-1} p(i)$$
$$n_O(T) = \sum_{i=T}^{N-1} p(i)$$

- $\sigma_B^2(T) =$  the variance of the pixels in the background (below threshold)
- $\sigma_O^2(T) =$  the variance of the pixels in the foreground (above threshold)



$$\sigma_{\text{Between}}^2(T) = n_B(T)n_O(T)[\mu_B(T) - \mu_O(T)]^2$$
Maximize

# Algorithm of Otsu Thresholding

Goal: select a *T* that maximizes  $n_B(T)n_O(T)[\mu_B(T) - \mu_O(T)]^2$ 

- For each potential threshold T,
  - 1. Separate the pixels into two clusters according to the threshold.
  - 2. Find the mean of each cluster.
  - 3. Square the difference between the means.
  - 4. Multiply by the number of pixels in one cluster times the number in the other.

# An Computational Issue

- The computations aren't independent as we change from one threshold to another.
- We can incrementally update with the following equations:

$$n_B(T+1) = n_B(T) + n_T$$
$$n_O(T+1) = n_O(T) - n_T$$
$$\mu_B(T+1) = \frac{\mu_B(T)n_B(T) + n_T T}{n_B(T+1)}$$
$$\mu_O(T+1) = \frac{\mu_O(T)n_O(T) - n_T T}{n_O(T+1)}$$

#### Connected components analysis:

- connected components labeling of the binary-1 pixels
- · property measurement of the component regions
- decision making





#### Connected components analysis:

- connected components labeling of the binary-1 pixels
- · property measurement of the component regions
- decision making





 All pixels that have value binary-1 and are connected to each other by a path of pixels all with value binary-1 are given the same identifying label.

- All pixels that have value binary-1 and are connected to each other by a path of pixels all with value binary-1 are given the same identifying label.
- label: unique name or index of the region
- label: identifier for a potential object region

- All pixels that have value binary-1 and are connected to each other by a path of pixels all with value binary-1 are given the same identifying label.
- label: unique name or index of the region
- label: identifier for a potential object region
- connected components labeling: a grouping operation

- All pixels that have value binary-1 and are connected to each other by a path of pixels all with value binary-1 are given the same identifying label.
- label: unique name or index of the region
- label: identifier for a potential object region
- connected components labeling: a grouping operation
- pixel property: position, gray level or brightness level
- region property: shape, bounding box, position, intensity statistics

- Definition of Connected Component
  - Two pixels p and q belong to the same connected component C if there is a sequence of 1-pixels  $(p_0, p_1, \dots p_n)$ , where

• 
$$p_0 = p$$

- $p_n = q$
- $p_{i-1}, p_i : i = 1, ..., n$  are neighbor

- · 4-connected: north, south, east, west
- 8-connected: north, south, east, west, northeast, northwest, southeast, southwest



Border: subset of 1-pixels also adjacent to 0-pixels

- · 4-connected: north, south, east, west
- 8-connected: north, south, east, west, northeast, northwest, southeast, southwest



Border: subset of 1-pixels also adjacent to 0-pixels

0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0

Original Image

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

**Connected Components** 

#### Connected Components Algorithms

- Common Features
  - Process a row of image at a time
  - Assign a new labels to the first pixel of each component.
  - Propagate the label of a pixel to its neighbors to the right or below it.

0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	0	1	1	1	0	1
0	0	1	1	1	1	1

0	0	1	0	0	0	2
0	0	1	1	0	0	2
0	0	1	1	1	0	2
0	0	1	1	1	1	Α

#### Connected Components Algorithms

Common Features

0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	0	1	1	1	0	1
0	0	1	1	1	1	1

0	0	1	0	0	0	2
0	0	1	1	0	0	2
0	0	1	1	1	0	2
0	0	1	1	1	1	Α

- What label should be assigned to A
- How does the algorithm keep track of the equivalence of two labels
- How does the algorithm use the equivalence information to complete the processing

## Algorithm 1: Iterative Algorithm

• Use no auxiliary storage

Algorithm Steps

- Computational Expensive
- Step1 (Initialization): Assign an unique label to each pixel.
- Step2 (Iteration) : Perform a sequence of top-down and bottom-up label propagation.



	1	2		3	4	
1	5	6		7	8	
	9	10	11	12	13	

1	1	Γ	3	3	
1	1		3	3	
1	1	1	1	1	

1	1		1	1	
1	1		1	1	
1	1	1	1	1	

- Two-Pass Algorithm
  - Pass 1:
    - Perform label assignment and label propagation
    - Construct the equivalence relations between labels when two different labels propagate to the same pixel.
    - Apply resolve function to find the transitive closure of all equivalence relations.
  - Pass 2:
    - Perform label translation.

• Example:





												1	1	1	1
															1
						2	2	2	2						1
	3					2	2								1
	1		1	1	1	1	1								1
1	1				1	1	1								1
1															1
1															1
1															1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1



{2=4} {3=5} {1=5}

- Example:
  - Resolve Function

 $\{2=4\} \ \{3=5\} \ \{1=5\} \implies \{2=4\} \ \{1=3=5\}$ 

### Computational EfficiencyNeed a lot of space to store equivalence

- Example:
  - Resolve Function

 $\{2=4\} \ \{3=5\} \ \{1=5\} \implies \{2=4\} \ \{1=3=5\}$ 

### Computational EfficiencyNeed a lot of space to store equivalence

## **Computer Vision**

Kuan-Wen Chen 2018/3/15