

Binary Image Analysis

Binary image analysis

- consists of a set of image analysis operations that are used to produce or process binary images, usually images of 0's and 1's.

0 represents the background
1 represents the foreground

```
00010010001000
00011110001000
00010010001000
```

1

Binary Image Analysis

- is used in a number of practical applications
 - part inspection
 - riveting
 - fish counting
 - document processing

2

What kinds of operations?

- Separate objects from background and from one another
- Aggregate pixels for each object
- Compute features for each object

3

Example: red blood cell image



- Many blood cells are separate objects
- Many touch – bad!
- Salt and pepper noise from thresholding
- How useable is this data?

4

Results of analysis

- 63 separate objects detected
- Single cells have area about 50
- Noise spots
- Gobs of cells

Object	Area	Centroid	Bounding Box	
1	383	(8.8 , 20)	[1 22 1 39]	
2	83	(5.8 , 50)	[1 11 42 55]	
3	11	(1.5 , 57)	[1 2 55 60]	
4	1	(1 , 62)	[1 1 62 62]	
5	1048	(19 , 75)	[1 40 35 100]	gobs
32	45	(43 , 32)	[40 46 28 35]	cell
33	11	(44 , 1e+02)	[41 47 98 100]	
34	52	(45 , 87)	[42 48 83 91]	cell
35	54	(48 , 53)	[44 52 49 57]	cell
60	44	(88 , 78)	[85 90 74 82]	
61	1	(85 , 94)	[85 85 94 94]	
62	8	(90 , 2.5)	[89 90 1 4]	
63	1	(90 , 6)	[90 90 6 6]	

5

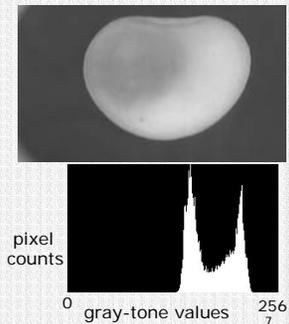
Useful Operations

1. Thresholding a gray-tone image
2. Determining good thresholds
3. Connected components analysis
4. Binary mathematical morphology
5. All sorts of feature extractors (area, centroid, circularity, ...)

6

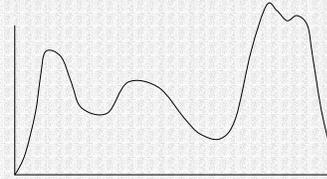
Thresholding

- Background is black
- Healthy cherry is bright
- Bruise is medium dark
- Histogram shows two cherry regions (black background has been removed)



Histogram-Directed Thresholding

How can we use a histogram to separate an image into 2 (or several) different regions?

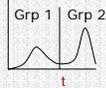


Is there a single clear threshold? 2? 3?

8

Automatic Thresholding: Otsu's Method

Assumption: the histogram is bimodal

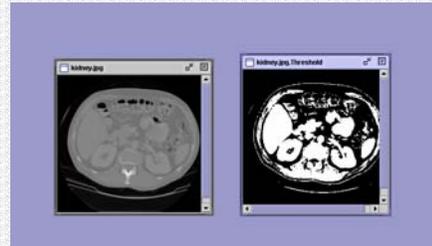


Method: find the threshold t that minimizes the **weighted sum of within-group variances** for the two groups that result from separating the gray tones at value t .

See text (at end of Chapter 3) for the recurrence relations; in practice, this operator works very well for true bimodal distributions and not too badly for others.

9

Thresholding Example

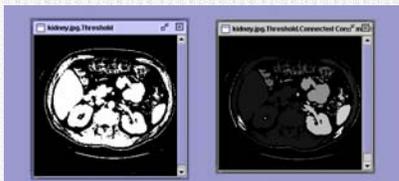


10

Connected Components Labeling

Once you have a binary image, you can identify and then analyze each **connected set of pixels**.

The connected components operation takes in a binary image and produces a **labeled image** in which each pixel has the integer label of either the background (0) or a component.



11

Methods for CC Analysis

1. Recursive Tracking (almost never used)
2. Parallel Growing (needs parallel hardware)
3. Row-by-Row (most common)
 - Classical Algorithm (see text)
 - Efficient Run-Length Algorithm (developed for speed in real industrial applications)

12

Equivalent Labels

Original Binary Image

```

0001110000111100001
0001111000111100011
0001111100111100111
0001111110111100111
0001111111111100111
0001111111111110011
0001111111111111111
0001111111111111111
0001111110000011111
    
```

13

Equivalent Labels

The Labeling Process

```

0001110000222200003
0001111000222200033
0001111100222200333
0001111110222200333
0001111111111100333
00011111111111100333
00011111111111110333
0001111111111111111
0001111110000011111
    
```

$1 \equiv 2$
 $1 \equiv 3$

14

Run-Length Data Structure

	0 1 2 3 4		
0	1 1 1 1	Binary Image	row scol ecol label
1	1 1 1 1		0 UNUSED 0
2	1 1 1 1		1 0 3 4 0
3	1 1 1 1		2 0 0 1 0
4	1 1 1 1		3 1 0 1 0
	Rstart Rend		4 1 4 4 0
0	1 2		5 2 0 2 0
1	3 4	Row Index	6 2 4 4 0
2	5 6		7 4 1 4 0
3	0 0		
4	7 7		

15

Run-Length Algorithm

```

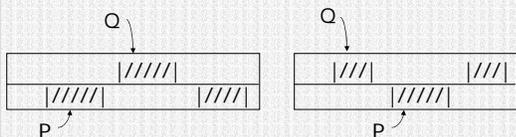
Procedure run_length_classical
{
  initialize Run-Length and Union-Find data structures
  count <- 0

  /* Pass 1 (by rows) */

  for each current row and its previous row
  {
    move pointer P along the runs of current row
    move pointer Q along the runs of previous row
  }
}
    
```

16

Case 1: No Overlap



```

/* new label */
count <- count + 1
label(P) <- count
P <- P + 1
    
```

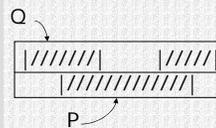
```

/* check Q's next run */
Q <- Q + 1
    
```

17

Case 2: Overlap

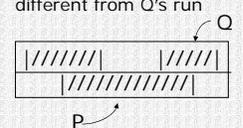
Subcase 1:
P's run has no label yet



```

label(P) <- label(Q)
move pointer(s)
    
```

Subcase 2:
P's run has a label that is different from Q's run



```

union(label(P), label(Q))
move pointer(s)
    
```

```

}
    
```

18

Pass 2 (by runs)

```

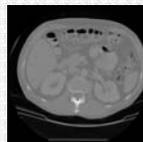
/* Relabel each run with the name of the
   equivalence class of its label */
For each run M
{
  label(M) <- find(label(M))
}

```

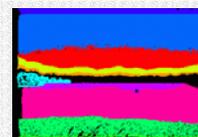
where union and find refer to the operations of the Union-Find data structure, which keeps track of sets of equivalent labels.

19

Labeling shown as Pseudo-Color



connected components of 1's from thresholded image



connected components of cluster labels

20

Mathematical Morphology

Binary mathematical morphology consists of two basic operations

dilation and erosion

and several composite relations

closing and opening
conditional dilation

...

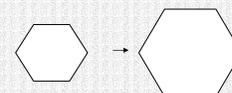
21

Dilation

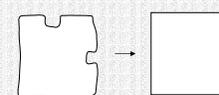
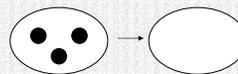
Dilation **expands** the connected sets of 1s of a binary image.

It can be used for

1. growing features



2. filling holes and gaps



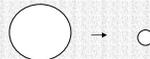
22

Erosion

Erosion **shrinks** the connected sets of 1s of a binary image.

It can be used for

1. shrinking features



2. Removing bridges, branches and small protrusions



23

Structuring Elements

A **structuring element** is a shape mask used in the basic morphological operations.

They can be any shape and size that is digitally representable, and each has an **origin**.



box



hexagon



disk



something

box(length,width)

disk(diameter)

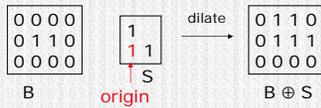
24

Dilation with Structuring Elements

The arguments to dilation and erosion are

1. a binary image B
2. a structuring element S

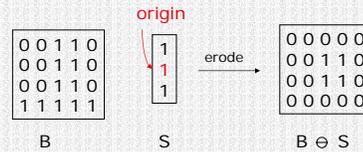
$dilate(B, S)$ takes binary image B, places the origin of structuring element S over each 1-pixel, and ORs the structuring element S into the output image at the corresponding position.



25

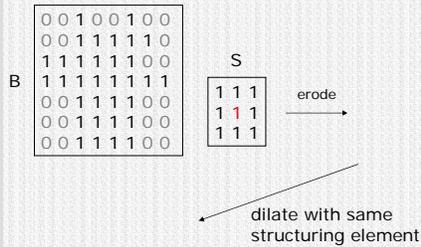
Erosion with Structuring Elements

$erode(B, S)$ takes a binary image B, places the origin of structuring element S over every pixel position, and ORs a binary 1 into that position of the output image only if every position of S (with a 1) covers a 1 in B.



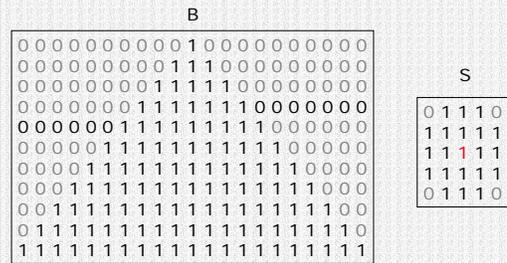
26

Example 1 to Try



27

Example 2 to Try



First erode and then dilate with the same S.

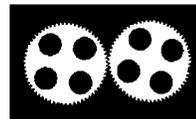
28

Opening and Closing

- **Closing** is the compound operation of dilation followed by erosion (with the same structuring element)
- **Opening** is the compound operation of erosion followed by dilation (with the same structuring element)

29

Gear Tooth Inspection



original binary image

How did they do it?



detected defects

30

Region Properties

Properties of the regions can be used to recognize objects.

- **geometric properties (Ch 3)**
- **gray-tone properties**
- **color properties**
- **texture properties**
- **shape properties (a few in Ch 3)**
- **motion properties**
- **relationship properties (1 in Ch 3)**

31

Geometric and Shape Properties

- area
- centroid
- perimeter
- perimeter length
- circularity
- elongation
- mean and standard deviation of radial distance
- bounding box
- extremal axis length from bounding box
- second order moments (row, column, mixed)
- lengths and orientations of axes of best-fit ellipse

Which are statistical? Which are structural?

32

Region Adjacency Graph

A **region adjacency graph** (RAG) is a graph in which each node represents a region of the image and an edge connects two nodes if the regions are adjacent.

This is jumping ahead a little bit.

We'll consider this further for structural image analysis.

33