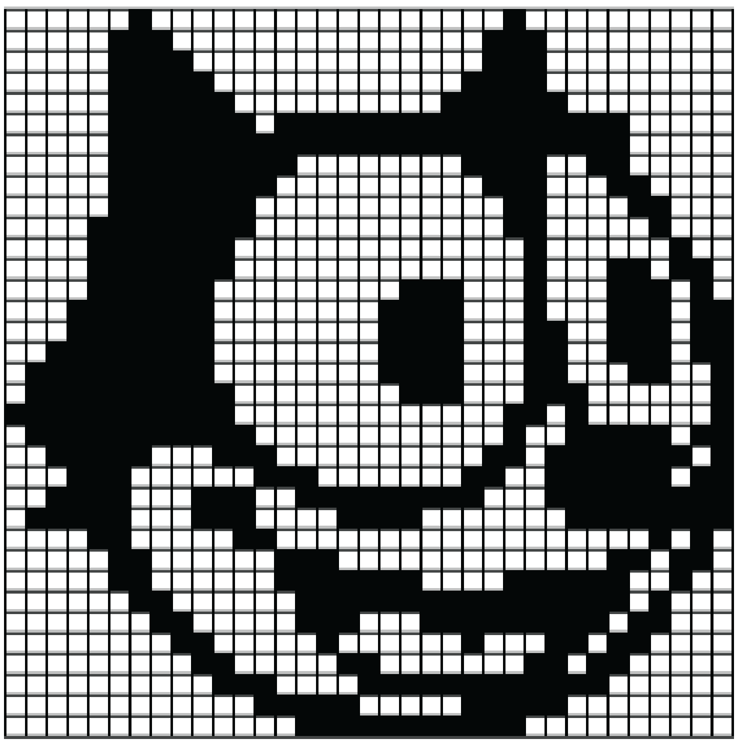
Chapter 3 – Binary Image Analysis

Visual Computing



$$\begin{bmatrix}
1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&0&0&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&0&0&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&0&0&1&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&1&1&1\\
1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&0&1&1\\
1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&0&1&0&0&1\\
1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&0&1&1&1&0&0&0&1&0&1\\
1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&1&1&1&0&0&0&1&0&0\\
1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&0\\
1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&0\\
1&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&0&1&1&1&0&0&1&1&0\\
1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&0&1&1&1&1&1&1&0\\
1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&0&0&0&0&0&1&0&0\\
1&1&0&0&0&0&0&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&1&0&0&0&0&0&0&0&1&0\\
1&1&1&0&0&0&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&0&0&1&1&0&0&0&0&0&0&1&0&0\\
1&1&0&0&0&0&1&1&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&1&1&1&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&0&1\\
1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&0&0&1\\
1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1&1&0&1&1\\
1&1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&1\\
1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&1&1&1&1&1&1&0&1&1&1&0&0&1&0&0&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&0&1&1&1&1&1&1&1&0&0&1&0&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

35x35

- Most common neighborhoods

	N	
W	*	E
	S	

Neighborhood N_4

NW	N	NE
W	*	E
SW	S	SE

Neighborhood N_8

- Use of masks

1	1	1
1	1	1
1	1	1

1	2	1
2	4	2
1	2	1

1
1
1
1
1

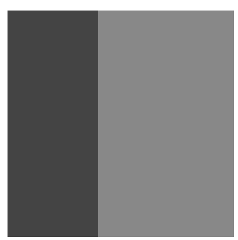
origin

  – Example:

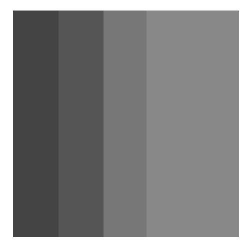
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80
40	50	70	80	80

input →



output →



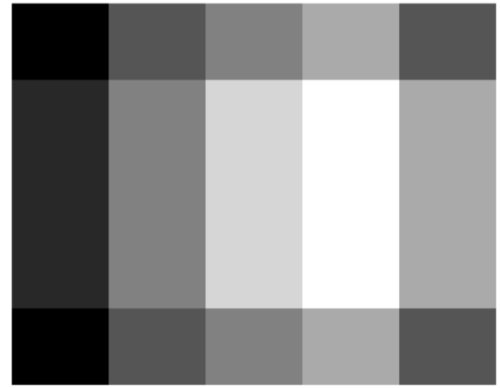
Example

40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80
40	40	80	80	80

input →

0	1	0
1	0	1
0	1	0

output →



80	160	200	240	160
120	200	280	320	240
120	200	280	320	240
120	200	280	320	240
80	160	200	240	160

and with the mask ?

0	0	0
0	1	0
0	0	0

- Algorithm
  - ***Hypothesis:*** An object is a set of connected pixels (connectivity 4) and without inner holes

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

Outer corners

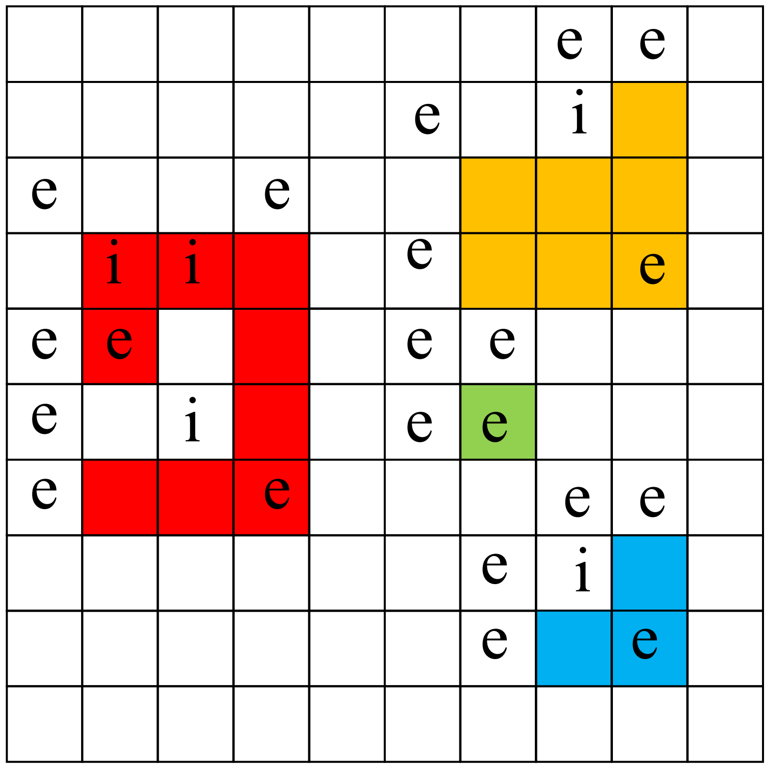
1	1	1	1	1	0	0	1
1	0	0	1	1	1	1	1

Inner corners

Compute the number of foreground objects of binary image B.
Objects are 4-connected and simply connected.
E is the number of external corners.
I is the number of internal corners.

```
procedure count_objects(B);
{
E := 0;
I := 0;
for L := 0 to MaxRow - 1
   for P := 0 to MaxCol - 1
      {
      if external_match(L, P) then E := E + 1;
      if internal_match(L, P) then I := I + 1;
      };
return((E - I) / 4);
}
```

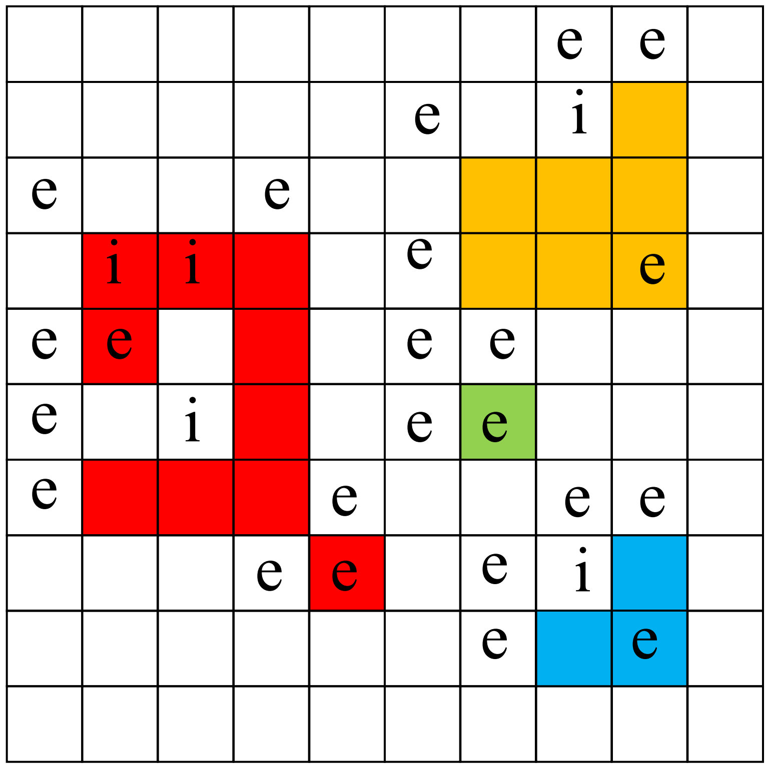
How many objects?



e - 21

i - 5

$$\# = \frac{21-5}{4} = \frac{16}{4} = 4$$



e - 23

i - 4

$$\# = \frac{23-4}{4} = \frac{19}{4} = ?$$

Connected Component Analysis – recursive algorithm

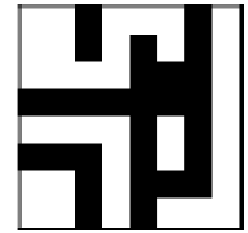
Compute the connected components of a binary image.
B is the original binary image.
LB will be the labeled connected component image.

```
procedure recursive_connected_components(B, LB);
{
LB := negate(B);
label := 0;
find_components(LB, label);
print(LB);
}

procedure find_components(LB, label);
{
for L := 0 to MaxRow
   for P := 0 to MaxCol
      if LB[L,P] == -1 then
         {
         label := label + 1;
         search(LB, label, L, P);
         }
}

procedure search(LB, label, L, P);
{
LB[L,P] := label;
Nset := neighbors(L, P);
for each (L',P') in Nset
   {
   if LB[L',P'] == -1
   then search(LB, label, L', P');
   }
}
```

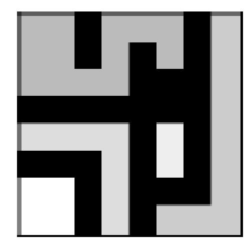
1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1



input

output

1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2



Recursive algorithm - Example

Step 1.

-1	-1	0	-1	-1	-1
-1	-1	0	-1	0	0
-1	-1	-1	-1	0	0

Step 2.

1	**-1**	0	-1	-1	-1
-1	-1	0	-1	0	0
-1	-1	-1	-1	0	0

Step 3.

1	1	0	-1	-1	-1
-1	-1	0	-1	0	0
-1	-1	-1	-1	0	0

Step 4.

1	1	0	-1	-1	-1
1	**-1**	0	-1	0	0
-1	-1	-1	-1	0	0

Step 5.

1	1	0	-1	-1	-1
1	1	0	-1	0	0
-1	-1	-1	-1	0	0

Neighbor 4

	1	
2	*	3
	4	

Neighbor 8

1	2	3
4	*	5
6	7	8

Union-Find structure

Construct the union of two sets.
X is the label of the first set.
Y is the label of the second set.
PARENT is the array containing the union-find data structure.

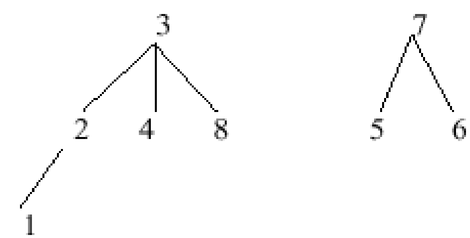
```
procedure union(X, Y, PARENT);
{
j := X;
k := Y;
while PARENT[j] <> 0
   j := PARENT[j];
while PARENT[k] <> 0
   k := PARENT[k];
if j <> k then PARENT[k] := j;
}
```

Find the parent label of a set.
X is a label of the set.
PARENT is the array containing the union-find data structure.

```
procedure find(X, PARENT);
{
j := X;
while PARENT[j] <> 0
   j := PARENT[j];
return(j);
}
```

PARENT

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
2	3	0	3	7	7	0	3



Classic CCA using Union-Find

Compute the connected components of a binary image.
B is the original binary image.
LB will be the labeled connected component image.

```
procedure classical_with_union-find(B,LB);
{
"Initialize structures."
initialize();
"Pass 1 assigns initial labels to each row L of the image."
for L := 0 to MaxRow
  {
  "Initialize all labels on line L to zero"
  for P := 0 to MaxCol
    LB[L,P] := 0;
  "Process line L."
  for P := 0 to MaxCol
    if B[L,P] == 1 then
      {
      A := prior_neighbors(L,P);
      if isempty(A)
      then { M := label; label := label + 1; };
      else M := min(labels(A));
      LB[L,P] := M;
      for X in labels(A) and X <> M
        union(M, X, PARENT);
      }
  }
"Pass 2 replaces Pass 1 labels with equivalence class labels."
for L := 0 to MaxRow
  for P := 0 to MaxCol
    if B[L,P] == 1
    then LB[L,P] := find(LB[L,P],PARENT);
};
```

input

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

1º step

1	1	0	2	2	2	0	3
1	1	0	2	0	2	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	7	7	3

PARENT

1	**2**	**3**	**4**	**5**	**6**	**7**
0	1	0	0	0	0	3

equiv. classes

2º step

1	1	0	1	1	1	0	3
1	1	0	1	0	1	0	3
1	1	1	1	0	0	0	3
0	0	0	0	0	0	0	3
4	4	4	4	0	5	0	3
0	0	0	4	0	5	0	3
6	6	0	4	0	0	0	3
6	6	0	4	0	3	3	3

- Apply the CCA algorithm considering connectivity 4

	1			2					
	1	1	1	1	1				
			1		1				
	3	3	1						
		3						4	
		3	3				5	4	
		3							

1	2	3	4	5
0	0 1	0 1	0	0 4

- What would be the result considering connectivity 8?

- Structuring elements

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

ones(3,5)

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

disk(5)

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

ring(5)

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

  - We need to define the center (i.e. origin)

- ***Definition***: A **dilation** of the binary image B by the structuring element S is defined as

$$B \oplus S = \bigcup_{b \in B} S_b \qquad S_b = \{s + b \mid s \in S\}$$

- ***Definition:*** A **erosion** of the binary image B by the structuring element S is defined as

$$B \ominus S = \{b \mid b + s \in B \, \forall s \in S\}$$

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				

a) Binary image B

1	1	1
1	**1**	1
1	1	1

b) Structuring Element S

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

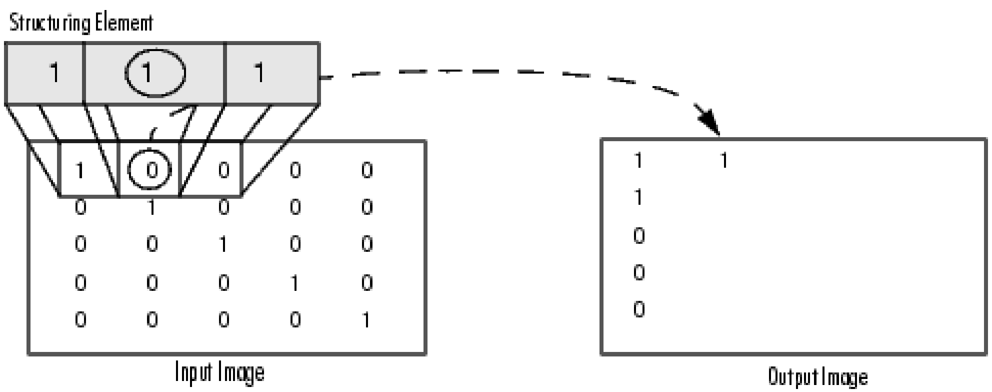
c) Dilation $B \oplus S$

				1	1		
				1	1		
				1	1		

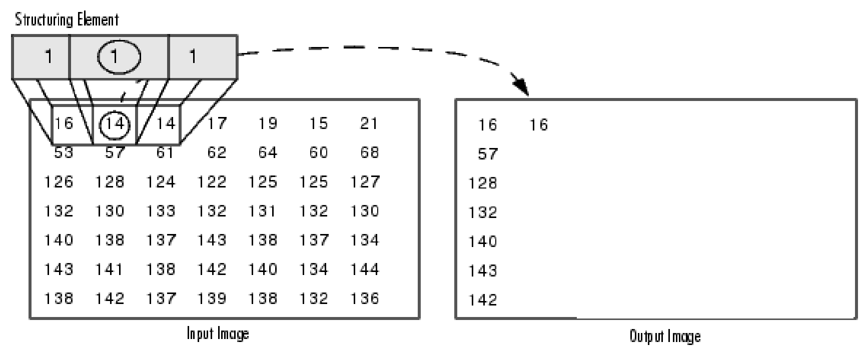
d) Erosion $B \ominus S$

Dilation and Erosion – Generalization for monochromatic images

Operação	Rule
Dilation	The output pixel value is the **maximum** value of all pixels in the neighborhood of the input pixel. It is assigned minimum value (0) to the other pixels
Erosion	The output pixel value is the **minimum** value of all pixels in the neighborhood of the input pixel. It is assigned a maximum value (1 or 255) to the other pixels



Dilation of the binary image



Dilation of a monochromatic image

- ***Definition:*** The **closing** of binary imge B by the structuring element S is defined by

$$B \bullet S = (B \oplus S) \ominus S$$

- ***Definition:*** A **opening** of a binary image B by the structuring element S is defined by

$$B \circ S = (B \ominus S) \oplus S$$

B

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				

S

1	1	1
1	1	1
1	1	1

$B \oplus S$

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

$B \bullet S$

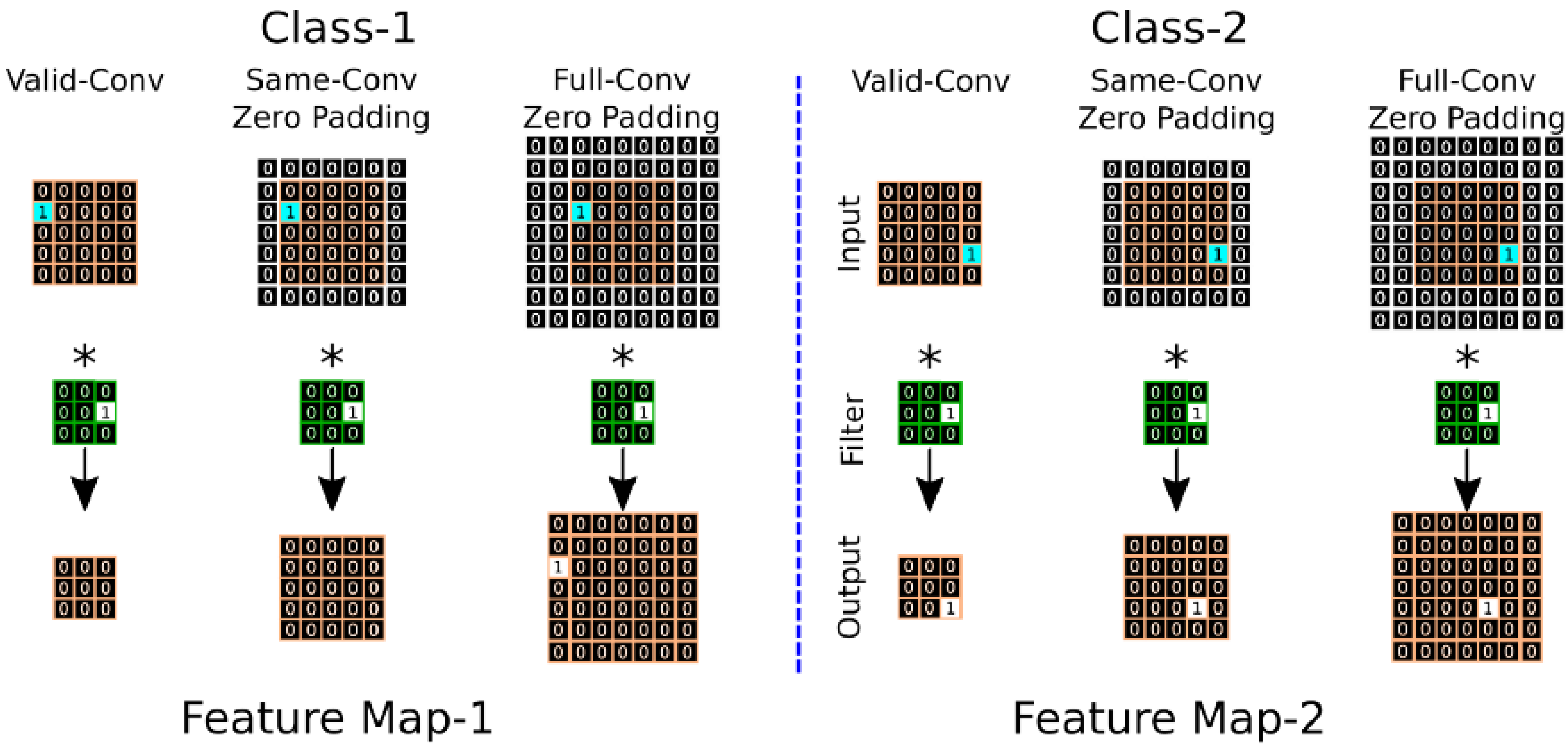
	1	1	1	1	1	1	
		1	1	1	1	1	
		1	1	1	1	1	
		1	1	1	1	1	
		1	1	1	1	1	
		1	1				

$B \ominus S$

				1	1		
				1	1		
				1	1		

$B \circ S$

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


Class-1
Class-2
Valid-Conv
Same-Conv
Zero Padding
Full-Conv
Zero Padding
Input
Filter
Output
Feature Map-1
Feature Map-2

Examples of morphology to extract shape primitives

- Medical applications (image resolution of 512x512)
  - **opening** with disk (13) followed by **closing** with disk (2)

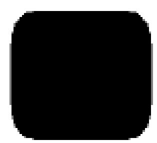
original

binarized

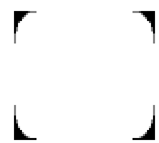
processed

- Extraction of shape primitives
  - Subtraction between the original image and the image obtained with the **opening** operator (using a disk as the structuring element)

original



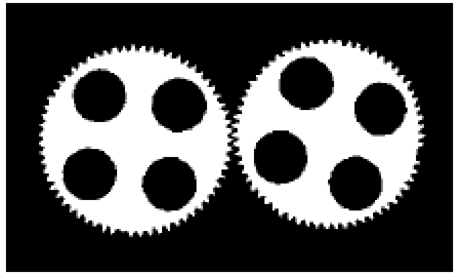
opening



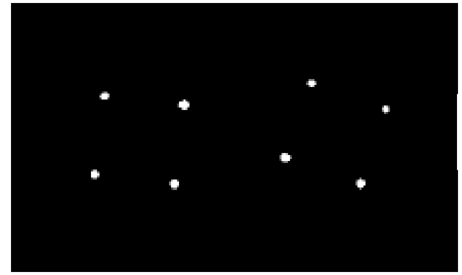
corners

input → B

a)



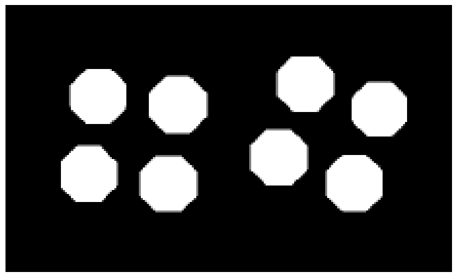
b)



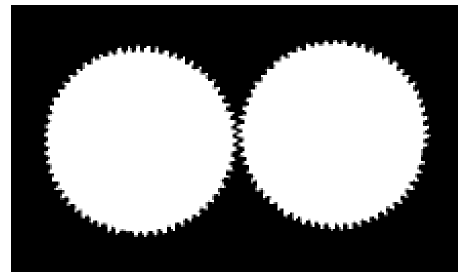
$B_1 = B \ominus \mathrm{HoleRing}$

c)

$B_2 = B_1 \oplus \mathrm{HoleMask}$

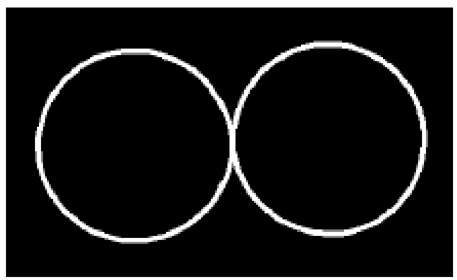


d)

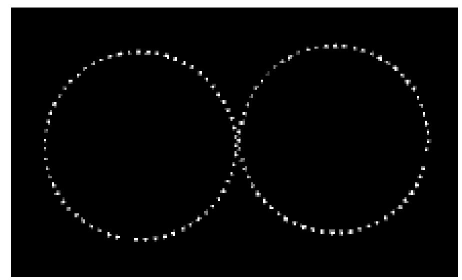


$B_3 = B \ \ \mathrm{OR} \ \ B_2$

e)



f)



$B_5 = B \ \mathrm{AND} \ B_4$

← output

Conditional Dilation

- ***Definition:*** Given two binary images, original *B*, and processed *C*, and the structuring element *S*, and let $C_0 = C$ e $C_n = \left(C_{n-1} \oplus S\right) \cap B$

The conditional dilation of *C* by *S* with respect to *B* is defined by

$$C \oplus\big|_B S = C_m$$

where *m* is the smallest integer satisfying $C_m = C_{m-1}$

B

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

$C = B \ominus V$

				1					
								1	
								1	

$C \oplus\big|_B S$

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

V

1
1
1

S

1	1	1
1	1	1
1	1	1

- Area $$A = \sum_{(r,c)\in R} 1$$

- Centroid $$\bar{r} = \tfrac{1}{A} \sum_{(r,c)\in R} r \qquad \bar{c} = \tfrac{1}{A} \sum_{(r,c)\in R} c$$

- Perimeter pixels $$P_4 = \{(r,c) \in R \mid N_8(r,c) - R \neq \emptyset\}$$ $$P_8 = \{(r,c) \in R \mid N_4(r,c) - R \neq \emptyset\}$$

  – Perimeter length $$|P| = \left|\{k \mid (r_{k+1}, c_{k+1}) \in N_4(r_k, c_k)\}\right| + \sqrt{2}\left|\{k \mid (r_{k+1}, c_{k+1}) \in N_8(r_k, c_k) - N_4(r_k, c_k)\}\right|$$

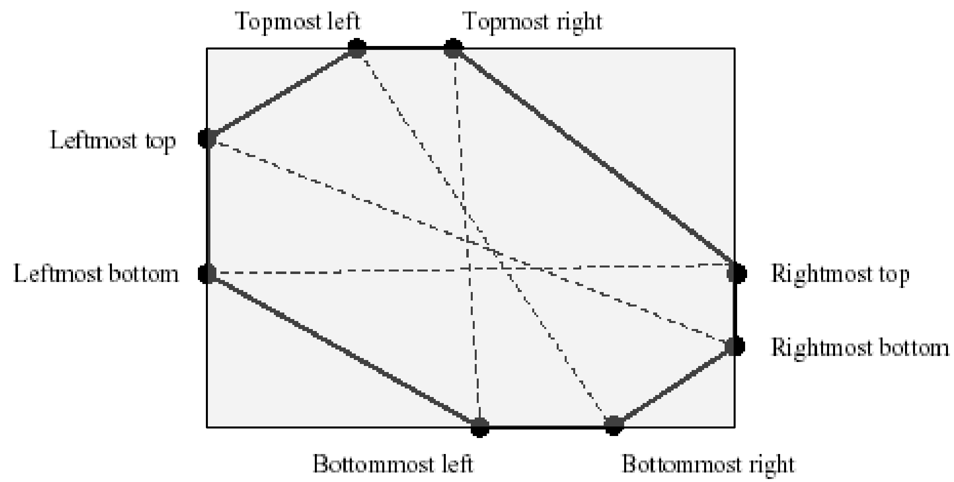
- Circularity (1) $$C_1 = \frac{|P|^2}{A}$$

- Circularity (2) $$C_2 = \frac{\mu_R}{\sigma_R}$$
  - Mean radial distance $$\mu_R = \frac{1}{K}\sum_{k=0}^{K-1}\left\|(r_k, c_k) - (\bar{r}, \bar{c})\right\|$$
  - Standard deviation of radial distance $$\sigma_R = \left(\frac{1}{K}\sum_{k=0}^{K-1}\left(\left\|(r_k, c_k) - (\bar{r}, \bar{c})\right\| - \mu_R\right)^2\right)^{1/2}$$

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

region num.	region area	row of center	col of center	perim. length	circu-larity$_1$	circu-larity$_2$	radius mean	radius var.
1	44	6	11.5	21.2	10.2	15.4	3.33	.05
2	48	9	1.5	28	16.3	2.5	3.80	2.28
3	9	13	7	8	7.1	5.8	1.2	0.04

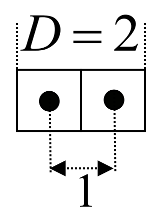
- Rectangle (and octagon) bounding box



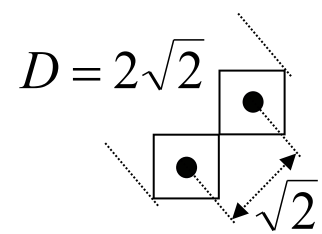
- Axis length

$$D = \sqrt{(r_2 - r_1)^2 + (c_2 - c_1)^2} + Q(\theta)$$

$$Q(\theta) = \begin{cases} \frac{1}{|\cos(\theta)|} & : \quad |\theta| < 45^{\circ} \\ \frac{1}{|\sin(\theta)|} & : \quad |\theta| > 45^{\circ} \end{cases}$$



$\theta = 0^{\circ}$

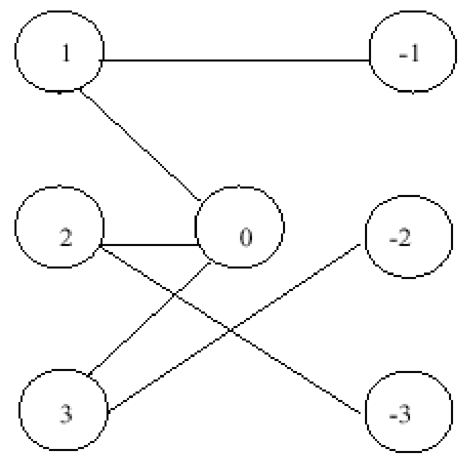


$\theta = 45^{\circ}$

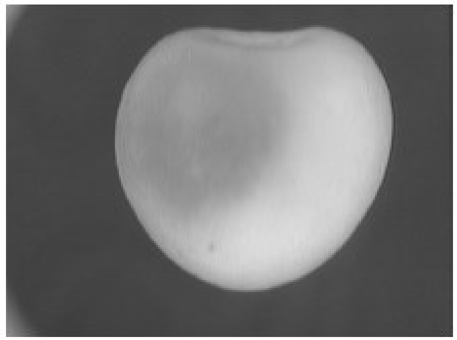
- ***Problem:*** regions that have inner holes (in the background)
- ***Solution:*** algorithm with 3 steps
  - Aplication of the CCA algorithm twice: (1) foreground pixels and (2) background pixels

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

  - (3) Building of graph relations



- ***Definition:*** o histogram h of the monochromatic image I is defined by $$h(m) = \left|\{(r,c) \mid I(r,c) = m\}\right|$$



Compute the histogram H of gray-tone image **I**.

```
procedure histogram(I,H);
{
"Initialize the bins of the histogram to zero."
for i := 0 to MaxVal
   H[i] := 0;
"Compute values by accumulation."
for L := 0 to MaxRow
   for P := 0 to MaxCol
      {
      grayval := I[r,c];
      H[grayval] := H[grayval] + 1;
      };
}
```

- Otsu Method
  - Let us assume that we have an image $I : \Omega \rightarrow R$, with $\Omega \subseteq R^2$ and $I(x, y)$ denotes the gray level intensity in the coordinates (x, y).

  - Also, assume that $M = \{1,2,3,\ldots,MaxVal\}$ represents the gray levels in the image I.
  - Defining n_i as the nº of pixels at a given level $i \in M$, the total nº of the pixels in the image is given by

$$n = \sum_{i=1}^{MaxVal} n_i$$

  - We can compute the weights of the pixels at the level i as

$$P(i) = n_i / n$$

  - We are going to assume that we have two classes of gray levels, $C_1 = \{1,2,\ldots,m\}$ $C2 = \{m+1, m+2,\ldots,MaxVal\}$ using a threshold m

Automatic computation of the Threshold (ii)

- Otsu Method
  - ***Idea:*** intra-class minimization variance $\sigma_W^2 = \mu_1(m)\sigma_1^2(m) + \mu_2(m)\sigma_2^2(m)$

$$w_1(m) = \sum_{i=1}^{m} P(i) \qquad w_2(m) = \sum_{i=m+1}^{MaxVal} P(i)$$

  - The weights allow to compute the mean of the gray levels of the two classes;

$$\mu_1(m) = \sum_{i=1}^{m} iP(i) / w_1(m) \qquad \mu_2(m) = \sum_{i=m+1}^{MaxVal} iP(i) / w_2(m)$$

  - Now, we can compute the variances, from the above means:

$$\sigma_1^2(m) = \sum_{i=i}^{m} \left(i - \mu_1(m)\right)^2 P(i) / w_1(m) \qquad \sigma_2^2(m) = \sum_{i=m+1}^{MaxVal} \left(i - \mu_2(m)\right)^2 P(i) / w_2(m)$$

Automatic computation of the Threshold (iii)

- Otsu then proposes the following "goodness" of the threshold

$$\lambda = \sigma_B^2 / \sigma_W^2$$

where

$$\sigma_B^2 = w_1(m)\left(1 - w_1(m)\right)\left(\mu_1(m) - \mu_2(m)\right)^2$$

$$= w_1(m)w_2(m)\left(\mu_1(m) - \mu_2(m)\right)^2$$

$$w_2(m) = 1 - w_1(m)$$

$$\sigma_W^2 = w_1\sigma_1^2 + w_2\sigma_2^2$$

σ_W^2 - within class (intra-class) variance

σ_B^2 - between class (inter-class) variance

- Synopsis of the Otsu algorithm to find the threshold t
  - Initializaion

$$P(i) = h(i)/|R\times C| \qquad w_1(0)=P(0)$$

$$\mu = \sum_{i=0}^{MaxVal} iP(i) \qquad \mu_1(0)=0$$

  - For $m := 0$ to $MaxVal$

$$w_1(m+1)=w_1(m)+P(m+1)$$

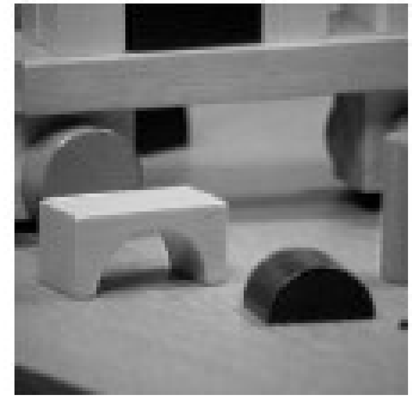
$$\mu_1(m+1)=\frac{w_1(m)\mu_1(m)+(m+1)P(m+1)}{w_1(m+1)}$$

$$\mu_2(m+1)=\frac{\mu-w_1(m+1)\mu_1(m+1)}{1-w_1(m+1)}$$

$$\sigma_B^2(m)=w_1(m)\big(1-w_1(m)\big)\big(\mu_1(m)-\mu_2(m)\big)^2$$

  - Limiar computation

$$\hat{m}=\arg\max_m \sigma_B^2(m)$$



Original ($MaxVal$=255)

$t = 93$

1 Compute histogram and probabilities of each intensity level
2 Set up initial \omega_i(0) and \mu_i(0)
3 Step through all possible thresholds t = 1 … maximum intensity
    3.1 Update \omega_i and \mu_i
    3.2 Compute \sigma^2_b(t)
4 Desired threshold corresponds to the maximum \sigma^2_b(t)