Chapter 3 – Binary Image Analysis
Pixels and Neighborhoods

• Most common neighborhoods

\[
\begin{array}{c|c|c}
N & W & E \\
\hline
W & * & E \\
S & & \\
\end{array}
\quad \begin{array}{c|c|c|c}
NW & N & NE \\
\hline
W & * & E \\
SW & S & SE \\
\end{array}
\]

Vizinhança N₄ Vizinhança N₈

• Use of masks

  – Example:

\[
\begin{array}{cccccc}
40 & 40 & 80 & 80 & 80 & 80 \\
40 & 40 & 80 & 80 & 80 & 80 \\
40 & 40 & 80 & 80 & 80 & 80 \\
40 & 40 & 80 & 80 & 80 & 80 \\
40 & 40 & 80 & 80 & 80 & 80 \\
\end{array}
\quad \begin{array}{cccccc}
40 & 50 & 70 & 80 & 80 & 80 \\
40 & 50 & 70 & 80 & 80 & 80 \\
40 & 50 & 70 & 80 & 80 & 80 \\
40 & 50 & 70 & 80 & 80 & 80 \\
40 & 50 & 70 & 80 & 80 & 80 \\
\end{array}
\]

input \rightarrow \quad \text{output} \rightarrow
Example

input →

and with the mask?

and with the mask?
Counting objects

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

  ![Pixel Grid](image)

  **Outer corners**
  **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);
  }
  ```

CVI - Binary Image Analysis
How many objects?

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>i</td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>i</td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{e} - 21 \]
\[ \text{i} - 5 \]
\[ \# = \frac{21 - 5}{4} = \frac{16}{4} = 4 \]

CVI - Binary Image Analysis
And now?

\[
e - 23
\]

\[
i - 4
\]

\[
# = \frac{23 - 4}{4} = \frac{19}{4} = ?
\]
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');
    }
}

CVI - Binary Image Analysis
## Recursive algorithm - Example

<table>
<thead>
<tr>
<th>Step 1.</th>
<th>Neighbor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 -1 0 -1 -1 -1</td>
<td>1</td>
</tr>
<tr>
<td>-1 -1 0 -1 0 0</td>
<td>2 * 3</td>
</tr>
<tr>
<td>-1 -1 -1 -1 0 0</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2.</th>
<th>Neighbor 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -1 0 -1 -1 -1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>-1 -1 0 -1 0 0</td>
<td>4 * 5</td>
</tr>
<tr>
<td>-1 -1 -1 -1 0 0</td>
<td>6 7 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 -1 -1 -1</td>
</tr>
<tr>
<td>-1 -1 0 -1 0 0</td>
</tr>
<tr>
<td>-1 -1 -1 -1 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 -1 -1 -1</td>
</tr>
<tr>
<td>1 -1 0 -1 0 0</td>
</tr>
<tr>
<td>-1 -1 -1 -1 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 5.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 -1 -1 -1</td>
</tr>
<tr>
<td>1 1 0 -1 0 0</td>
</tr>
<tr>
<td>-1 -1 -1 -1 0 0</td>
</tr>
</tbody>
</table>
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.

```c
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.

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

CVI - Binary Image Analysis
Exercise

• Apply the CCA algorithm considering connectivity 4

• What would be the result considering connectivity 8?
Morphological operators

- Structuring elements

\[
\begin{align*}
\text{ones(3,5)} & \quad \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 
\end{array} \\
\text{disk(5)} & \quad \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 
\end{array} \\
\text{ring(5)} & \quad \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 
\end{array}
\end{align*}
\]

- 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

\[
B \oplus S = \bigcup_{b \in B} S_b \quad \quad S_b = \{ s + b \mid s \in S \}
\]

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

\[
B \ominus S = \{ b \mid b + s \in B \forall s \in S \}
\]
Morphological operations - Examples

a) Binary image \( B \)

b) Structuring Element \( S \)

c) Dilation \( B \oplus S \)

d) Erosion \( B \ominus S \)
Dilation and erosion – Generalization for monochromatic images

<table>
<thead>
<tr>
<th>Operação</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dilation</td>
<td>The output pixel value is the <strong>maximum</strong> value of all pixels in the neighborhood of the input pixel. It is assigned minimum value (0) to the other pixels</td>
</tr>
<tr>
<td>Erosion</td>
<td>The output pixel value is the <strong>minimum</strong> value of all pixels in the neighborhood of the input pixel. It is assigned a maximum value (1 or 255) to the other pixels</td>
</tr>
</tbody>
</table>

**Dilation of the binary image**

**Dilation of a monochromatic image**
Morphological operations

• **Definition:** The **closing** of binary image $B$ 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

$$B \circ S = (B \ominus S) \oplus S$$
Examples of morphology to extract shape primitives

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

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

input $B$

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

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

$B_3 = B \lor B_2$

$B_5 = B \land B_4$

output
Conditional Dilation

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

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

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

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

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

- **Centroid**
  \[ \bar{r} = \frac{1}{A} \sum_{(r,c) \in R} r \quad \bar{c} = \frac{1}{A} \sum_{(r,c) \in R} c \]

- **Perimeter pixels**
  \[ P_4 = \{(r,c) \in R | N_8(r,c) - R \neq \emptyset \} \]
  \[ P_8 = \{(r,c) \in R | N_4(r,c) - R \neq \emptyset \} \]

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

- **Circularity (1)**
  \[ C_1 = \frac{|P|^2}{A} \]
Properties (cont.)

- **Circularity (2)**

  - Mean radial distance

  \[ C_2 = \frac{\mu_R}{\sigma_R} \]

  \[ \mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \| (r_k, c_k) - (\bar{r}, \bar{c}) \| \]

  - Standard deviation of radial distance

  \[ \sigma_R = \left( \frac{1}{K} \sum_{k=0}^{K-1} (\| (r_k, c_k) - (\bar{r}, \bar{c}) \| - \mu_R)^2 \right)^{1/2} \]

<table>
<thead>
<tr>
<th>region</th>
<th>region num.</th>
<th>row of center</th>
<th>col of center</th>
<th>perim. length</th>
<th>circu.</th>
<th>circu. mean</th>
<th>radius</th>
<th>radius var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>44</td>
<td>6</td>
<td>11.5</td>
<td>21.2</td>
<td>10.2</td>
<td>15.4</td>
<td>3.33</td>
<td>.05</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
<td>9</td>
<td>1.5</td>
<td>28</td>
<td>16.3</td>
<td>2.5</td>
<td>3.80</td>
<td>2.28</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>13</td>
<td>7</td>
<td>8</td>
<td>7.1</td>
<td>5.8</td>
<td>1.2</td>
<td>0.04</td>
</tr>
</tbody>
</table>
Properties – lengths and boundaries

- Rectangle (and octagon) bounding box

- Axis length

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

\[ Q(\theta) = \begin{cases} 
\frac{1}{|\cos(\theta)|} & : |\theta| < 45^\circ \\
\frac{1}{|\sin(\theta)|} & : |\theta| > 45^\circ 
\end{cases} \]
Properties – 2nd order moments

- Second order moments

\[
\mu_{rr} = \frac{1}{A} \sum_{(r,c) \in R} (r - \bar{r})^2 \\
\mu_{cc} = \frac{1}{A} \sum_{(r,c) \in R} (c - \bar{c})^2 \\
\mu_{rc} = \frac{1}{A} \sum_{(r,c) \in R} (r - \bar{r})(c - \bar{c})
\]

- Relation between moments and elliptic regions

\[
R = \{(r,c) | dr^2 + 2erc + fc^2 \leq 1\}
\]

\[
\begin{pmatrix} d & e \\ e & f \end{pmatrix} = \frac{1}{4(\mu_{rr} - \mu_{cc}^2)} \begin{pmatrix} \mu_{cc} & -\mu_{rc} \\ -\mu_{rc} & \mu_{rr} \end{pmatrix}
\]

- Axis with least second order moment
  - Formulation

\[
\mu_{r,c,\alpha} = \frac{1}{A} \sum_{(r,c) \in R} d^2
\]

\[
= \frac{1}{A} \sum_{(r,c) \in R} (\bar{V} \circ (\cos \beta, \sin \beta))^2
\]

  - Solution

\[
\tan(2\hat{\alpha}) = \frac{2\mu_{rc}}{\mu_{rr} - \mu_{cc}}
\]

\[
\beta = \alpha + 90
\]

\[
\beta = \alpha + \pi / 2
\]
Region Adjacency Graphs

• **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 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 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 1 1 1 1 1 0 2 2 0
```

– (3) Building of graph relations
**Definition:** the histogram $h$ of a monochromatic image $I$ is defined by

$$h(m) = |\{(r,c) | I(r,c) = m\}|$$

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

```plaintext
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;
            }
}
```
Automatic computation of the Threshold

• 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 \( M = \{1, 2, 3, ..., MaxVal\} \) 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

    \[
    P(i) = \frac{n_i}{n}
    \]

  – We are going to assume that we have two classes of gray levels, \( C_1 = \{1, 2, ..., m\} \) \( C_2 = \{m+1, m+2, ..., MaxVal\} \) using a threshold \( m \).
Automatic computation of the Threshold

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

\[
w_1(m) = \sum_{i=1}^{m} P(i) \quad w_2(m) = \sum_{i=m+1}^{\text{MaxVal}} P(i)
\]

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

\[
\mu_1(m) = \sum_{i=1}^{m} iP(i) / w_1(m) \quad \mu_2(m) = \sum_{i=m+1}^{\text{MaxVal}} iP(i) / w_2(m)
\]

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

\[
\sigma_1^2(m) = \sum_{i=1}^{m} (i - \mu_1(m))^2 P(i) / w_1(m) \quad \sigma_2^2(m) = \sum_{i=m+1}^{\text{MaxVal}} (i - \mu_2(m))^2 P(i) / w_2(m)
\]
Otsu then proposes the following “goodness” of the threshold

\[ \lambda = \frac{\sigma_B^2}{\sigma_W^2} \]

where

\[ \sigma_B^2 = w_1(m)(1 - w_1(m))(\mu_1(m) - \mu_2(m))^2 \]

\[ = w_1(m)w_2(m)(\mu_1(m) - \mu_2(m))^2 \]

\[ w_2(m) = 1 - w_1(m) \]

\[ \sigma_W^2 = w_1\sigma_1^2 + w_2\sigma_2^2 \]

\[ \sigma_W^2 \] - within class (intra-class) variance

\[ \sigma_B^2 \] - between class (inter-class) variance
Recursive Algorithm

• Synopsis of the Otsu algorithm to find the threshold $t$
  
  – Initialization
    
    $$P(i) = h(i) / |R \times C|$$
    
    $$w_1(0) = P(0)$$
    
    $$\mu = \sum_{i=0}^{MaxVal} iP(i)$$
    
    $$\mu_1(0) = 0$$
    
  – For $m := 0$ to $MaxVal$
    
    $$w_1(m+1) = w_1(m) + P(m+1)$$
    
    $$\mu_1(m+1) = \frac{w_1(m)\mu_1(m) + (m+1)P(m+1)}{w_1(m+1)}$$
    
    $$\mu_2(m+1) = \frac{\mu - w_1(m+1)\mu_1(m+1)}{1 - w_1(m+1)}$$
    
    $$\sigma_B^2(m) = w_1(m)(1 - w_1(m))(\mu_1(m) - \mu_2(m))^2$$
    
  – 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 $t = 1 \ldots$ maximum intensity
   3.1 Update $\omega_i$ and $\mu_i$
   3.2 Compute $\sigma^2_b(t)$
4 Desired threshold corresponds to the maximum $\sigma^2_b(t)$