

Notes for lab 3 (Image Segmentation and Connected Object Recognition)

Part1: Thresholding Segmentation

What is thresholding segmentation?

Thresholding segmentation is a method, which separates an image into two meaningful regions: foreground and background, through a selected threshold value T . If the image is a grey image, T is an integer in the range of $[0..K]$, where K is the maximum intensity value. For example, if the image is an 8-bit gray image, K takes the value of 255 and T is in the range of $[0..255]$.

Whenever the value of T is decided, the segmentation procedure is indicated by the following equation:

$$G_B(x, y) = \begin{cases} 1, & \text{if } G(x, y) > T \\ 0, & \text{if } G(x, y) \leq T \end{cases} \quad (1)$$

In equation (1), $G(x,y)$ indicates the intensity value of pixel (x,y) in the grey image G . G_B is the segmentation result. Actually it forms a binary image, in which each value of $G_B(x,y)$ gives the category (foreground or background) that the corresponding pixel belongs to. If $G_B(x,y) = 1$, then pixel (x,y) in the image G is classified as a foreground pixel, otherwise it is classified as a background pixel.

Equation (1) is formulated under the assumption that foreground pixels in the image G have relatively high intensity values and background pixels take low intensity values. Of course you can reverse the equation when you need to set the low intensity region as the foreground.

How to select the value of T

The major problem of thresholding segmentation is how to select the optimal value of threshold T . Usually in order to get the optimal value of T , we need to statically analyze the so-called “histogram” (or “intensity histogram”) of the input gray image G . Before talking about the algorithm, first we list all the notions and statistic definitions that relate to histogram as follows.

Basic notations

$G(x,y)$: The input gray image that we want to segment.

$G_B(x,y)$: The segmentation result of G . It is a binary image, the value of $G_B(x,y)$ is either 0 or 1 indicating the corresponding pixel (x,y) in G belongs to background or foreground respectively.

K : The maximum possible intensity value defined by G . If G is an 8-bit gray image, then K takes the value 255.

T : The thresholding value. It is an integer within the range $[0..K]$.

N : The total number of pixels in G . If G has *width* = w , and *height* = h , then of course $N = w \times h$.

Definition of histogram and normalized histogram

H_G is the intensity histogram of image G and it maps from each intensity value to an integer. The value of $H_G(i)$ indicates the number of pixels in G that takes the intensity value, i , where $i \in [0..K]$, K is the maximum intensity value as mentioned above (for example $K = 255$ for 8-bit grey images). Obviously $H_G(i)$ is an integer value within range $[0..N]$, N is the total number of pixels in G as mentioned above. Based on the definition of histogram H_G , we can get the so-called normalized histogram P_G . It is defined as follows:

$$P_G(i) = H_G(i) / N \quad (2)$$

In equation (2), the value of each $P_G(i)$ indicates the percentage of pixels in G that takes the intensity value i . Clearly $P_G(i)$ is a real value within the range $[0..1]$. The main reason of introducing the normalized

histogram is that sometime we need to compare two histogram from two images that contains different number of pixels. And the definition of normalized histogram makes this kind of comparison meaningful.

Statistic analysis of normalized histogram

Assume the current thresholding value is T , which separates the input image G into two regions: foreground and background according the equation (1). The frequency of background, $\omega_B(T)$ and the frequency of foreground, $\omega_F(T)$ are defined as follows:

$$\omega_B(T) = \sum_{i=0}^T P_G(i), \quad \omega_F(T) = \sum_{i=T+1}^K P_G(i) \quad (3)$$

Of course the frequency of the entire image G is calculated as $\omega = \omega_B(T) + \omega_F(T) = 1$, no matter what value T takes. The mean intensity values of background and foreground, $\mu_B(T)$ and $\mu_F(T)$ are calculated as:

$$\mu_B(T) = \left(\sum_{i=0}^T i \cdot P_G(i) \right) / \omega_B, \quad \mu_F(T) = \left(\sum_{i=T+1}^K i \cdot P_G(i) \right) / \omega_F \quad (4)$$

The mean intensity value of the entire image μ can be calculated as: $\mu = \mu_B(T) \omega_B(T) + \mu_F(T) \omega_F(T)$. Clearly no matter what value T takes, μ keeps the same. The intensity variances of background and foreground, $\sigma_B^2(T)$ and $\sigma_F^2(T)$ are defined as:

$$\sigma_B^2(T) = \sum_{i=0}^T \left(\frac{(i - \mu_B(T))^2 \cdot P(i)}{\omega_B(T)} \right), \quad \sigma_F^2(T) = \sum_{i=T+1}^K \left(\frac{(i - \mu_F(T))^2 \cdot P(i)}{\omega_F(T)} \right) \quad (5)$$

Having the definition of variances of the background and the foreground, it is the time to define the so-called “within-class” variance, σ_{within}^2 :

$$\sigma_{within}^2(T) = \omega_B(T) \cdot \sigma_B^2(T) + \omega_F(T) \cdot \sigma_F^2(T) \quad (6)$$

Also we can define the so-called “between-class” variance, $\sigma_{between}^2$:

$$\sigma_{between}^2(T) = \sigma^2 - \sigma_{within}^2(T) = \omega_B(T) \cdot \omega_F(T) \cdot [\mu_B(T) - \mu_F(T)]^2 \quad (7)$$

In the above equation, σ^2 indicates the intensity variance of the entire image G and it is calculated as:

$$\sigma^2 = \sum_{i=0}^K \left(\frac{(i - \mu)^2 \cdot P(i)}{\omega} \right) \quad (8)$$

Obviously given the image G , σ^2 is a constant value independent to the selection of the value of T .

Otsu's algorithm

The Otsu's algorithm is simple. We let T try all the intensity values from 0 to K and choose the one that gives the minimum “within-class” variance σ_{within}^2 as the optimal thresholding value. Formally speaking:

$$\text{Optimal value of } T = T_{Opt} \text{ where } \sigma_{within}^2(T_{Opt}) = \min_{0 \leq T \leq K} [\sigma_{within}^2(T)] \quad (9)$$

As we said before, $\sigma^2 = \sigma_{within}^2(T) + \sigma_{between}^2(T)$, and σ^2 is independent of the selection of T , therefore, minimization of σ_{within}^2 means maximization of $\sigma_{between}^2$. So the optimal value of T can also be taken as:

$$\text{Optimal value of } T = T_{Opt} \text{ where } \sigma_{between}^2(T_{Opt}) = \max_{0 \leq T \leq K} [\sigma_{between}^2(T)] \quad (10)$$

In fact equation (10) is the usual way that we use to find the optimal thresholding value. It is because that for each T , the calculation of $\sigma_{between}^2$ only needs the calculations of ω_B , ω_F , μ_B and μ_F according to equation (7). And these values can be updated iteratively:

Initially, $T = 0$:

Calculate the mean intensity of the entire image, μ ;

$\omega_B(0) = P_G(0)$; $\omega_F(0) = 1 - \omega_B(0) = 1 - P_G(0)$;

$\mu_B(0) = 0$; $\mu_F(0) = \mu / \omega_F(0)$, be careful if $\omega_F(0) = 0$, then $\mu_F(0) = 0$.

Iteratively, $T = T+1$:

$\omega_B(T+1) = \omega_B(T) + P_G(T+1)$; $\omega_F(T+1) = 1 - \omega_B(T+1)$;
If $\omega_B(T+1) = 0$, then $\mu_B(T+1) = 0$;
ELSE $\mu_B(T+1) = [\omega_B(T) \times \mu_B(T) + (T+1) \times P_G(T+1)] / \omega_B(T+1)$;
If $\omega_F(T+1) = 0$; then $\mu_F(T+1) = 0$;
ELSE $\mu_F(T+1) = [\mu - \omega_B(T+1) \times \mu_B(T+1)] / \omega_F(T+1)$;

Part2: Connect Object Recognition

After obtaining the binary image G_B , it is the time to count the number of connected objects in the foreground region (or in the background region if you want). Assume here we interest in the foreground (for background region, you just need to reverse the equation 1). The way we used to count the connected objects in the foreground region is based on the warshell's algorithm. The entire algorithm of connect object counting is described as follows:

Step 1:

Create a label image G_L with the same size (width and height) as G_B , and initially set each $G_L(x,y) = -1$. Create a variable, *current_label*, for recording the current available label and initially *current_label* = 0.

Step 2:

Scan the binary image G_B sequentially and update the label image G_L as follows:

```
FOR each y FROM 0 TO height - 1
  FOR each x FROM 0 TO width - 1
    IF pixel (x,y) is a foreground, which means  $G_B(x,y) = 1$ ,
      THEN
        Check the 8 neighborhood pixels around the pixel (x,y);
        FOR each neighborhood pixel (x',y')
          IF it has a nonnegative label in the label image  $G_L$ , which means  $G_L(x',y') \geq 0$ ,
            THEN
              SET  $G_L(x,y) = G_L(x',y')$ ;
            ELSE
              SET  $G_L(x,y) = \text{current\_label}$ ;
              SET  $\text{current\_label} = \text{current\_label} + 1$ ;
            END IF
          END FOR
        END FOR
      END IF
    END FOR
  END FOR
END FOR
```

Step 3:

Build the 2D transit matrix M_T . Initially create an empty matrix:

$M_T[0..\text{current_label}-1][0..\text{current_label}-1]$.

Clearly it is an 2D array with $\text{size} = (\text{current_label}) \times (\text{current_label})$, and each element in the matrix is first set to 0. Then assign 1 to each element of the matrix that locates at the diagonal. It means: set $M_T[i][i] = 1$, where i is from 0 to $\text{current_label}-1$.

After the above initialization, we need to update the matrix M_T according to the label image G_L and let M_T become the transit matrix of G_L . The update procedure is given as follows:

```
FOR each y FROM 0 TO height - 1
  FOR each x FROM 0 TO width - 1
    IF pixel (x,y) is a foreground pixel, which means  $G_B(x,y) = 1$ ,
      THEN
        Check the 8 neighborhood pixels around pixel (x,y).
```

```

FOR each neighborhood pixel (x',y')
  IF its label is nonnegative, which means  $G_L(x',y') \geq 0$ 
  THEN
    SET  $M\_T[G_L(x,y)][G_L(x',y')] = 1$ ;
    SET  $M\_T[G_L(x',y')][G_L(x,y)] = 1$ ;
  END IF
END FOR
END IF
END FOR
END FOR

```

After the above procedure, you get the updated matrix M_T , which represents the transit relationship in G_L .

Step 4:

Calculate the transit closure matrix, M_TC , of M_T . This calculation of transit closure matrix is based on the warshell's algorithm which is an iteration procedure described as follows:

- (1) Initially create two temporary matrixes M_0 and M_1 , which have the same size of the matrix M_T . Copy each value in M_T into the corresponding position in M_0 and M_1 , which is:

```

FOR i FROM 0 TO current_label-1
  FOR j FROM 0 TO current_label-1
    SET  $M\_0[i][j] = M\_1[i][j] = M\_T[i][j]$ ;
  END FOR
END FOR

```

- (2) Update the matrix M_1 using the transitivity law, which is:

```

FOR i FROM 0 TO current_label-1
  FOR j FROM 0 TO current_label-1
    FOR k FROM 0 TO current_label-1
      IF  $M\_1[i][j] = 1$  AND  $M\_1[j][k] = 1$ 
      THEN
        SET  $M\_1[i][k] = 1$ ;
        SET  $M\_1[k][i] = 1$ ;
      END IF
    END FOR
  END FOR
END FOR

```

- (3) Compare M_1 and M_0 to see if they are exactly the same, which means each element from M_1 has the same value as the corresponding element from M_0 . If so, set $M_TC = M_1$, which means we got the transit closure matrix. If not, copy M_1 to M_0 and go back to (2).

Step 5:

Count the number of connected objects (or the number of equivalent classes) in the transit closure matrix M_TC . It is not hard to see that the number connected objects equals to the number of distinct rows (or columns) in matrix M_TC . Assume $M_TC[i]$ and $M_TC[j]$ are the two rows of matrix M_TC , where $i \neq j$. We say these two rows are distinct if and only if there exists at least one $k \in [0.. current_label-1]$ that $M_TC[i][k] \neq M_TC[j][k]$.

Step 6:

Output the number of connected objects (or the number of distinct rows of M_TC) and return.

One example of transit closure calculation (step 4)

Assume after step 3, we got the transit matrix M_T . In step 4 (1), we initially set two temporary matrixes M_0 and M_1 , and copy M_T to these two matrixes, as shown in Figure 1. Then we do some operations on the matrix M_1 as described in the Step 4 (2), we get an updated M_1 as shown in Figure 2:

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

Figure 1: Copy M_T to M_0 and M_1 .

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

Figure 2: Updated M_1 by applying Step 4 (2).

Then according to Step 4 (3), we compare matrix M_1 (Figure 2) with matrix M_0 (Figure 1). We find M_1 is different from M_0 . Therefore we copy M_1 to M_0 , as shown in Figure 3. Then go back to (2) do the same operations on M_1 again, and get M_1 updated again, as shown in Figure 4.

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

Figure 3: Copy M_1 to M_0 .

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

Figure 4: M_1 is updated again.

Then we compare M_1 (Figure 4) and M_0 (Figure 3). Again we find that they are different. So we need to copy M_1 to M_0 (as shown in Figure 5) and go back to (2) to get M_1 updated again (as shown in Figure 6).

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

Figure 5: Copy M_1 to M_0 .

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

Figure 6: M_1 is updated again.

This time we find that M_1 did not change, which means M_0 (Figure 5) and M_1 (Figure 6) are identical. So we say the current M_1 is the transit closure matrix that we want, and set $M_{TC} = M_1$. Furthermore we discover there are two distinct rows in the transit closure matrix: (1111100) and (0000011) . It means that there are two connected objects.