數位影像處理第四次作業

自動碩一 111618018 吳祐毅

NTUT, Institute of Automation Technology

Digital Image Processing

Instructor：Chen Chin-Sheng (陳金聖)

Homework Assignment #4 Due date: 2022/12/20

1. (50%) You have to write programs to respectively binarize "Lena image" and your own image at optimum thresholds to get a ***binary image***. Please do binary morphological operations for the above images. The black pixels would be processed (operating on black pixels).
   (1) Optimumizing the thresholds to get a ***binary image*** using observation and Otsu method.
   (2) Do binary morphological opening for the above images.
   (3) Do binary morphological closing for the above images.
   For (2) and (3), please use the the octogonal 3-5-5-5-3 structure element.

```
public void observation(IntPtr src, IntPtr src2, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);        // Get image width
    int Height = iImage.GetHeight(src);      // Get image height
    int Threshold = Convert.ToInt32(bin_bar.Value);
    lab_thr.Text = Convert.ToString(Threshold);
    byte[,] Graymatrix = new byte[Height, Width];
    err = iImage.iPointerFromiImage(src, ref Graymatrix[0, 0], Width, Height);
    print_error(err);
    // start sliding the matrix
    for (int i = 0; i < Height; i++)              // i index for cols ( 0~Hight-1)(Because that matrix index is start from 0)
    {
        for (int j = 0; j < Width; j++)           // j index for rows ( 0~Width-1)
        {
            if (Graymatrix[i, j] < Threshold)     // Compare if the image intensity of each pixel smaller than threshold
                Graymatrix[i, j] = 0;             // Set pixel to 0 (Black)
            else
                Graymatrix[i, j] = 255;           // Set pixel to 1 (White)
        }
    }

    IntPtr imgPtr = iImage.iVarPtr(ref Graymatrix[0, 0]);
    err2 = iImage.iPointerToiImage(src2, imgPtr, Width, Height);
    print_error(err2);
    show_img(src2, pictureBox);
}
```

Otsu’s 算法

```
public void Otsu(IntPtr src, IntPtr src2, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);        // Get image width
    int Height = iImage.GetHeight(src);      // Get image height
    int Threshold = Convert.ToInt32(bin_bar.Value);
    int best_Threshold = 0;
    byte[,] Graymatrix = new byte[Height, Width];
    int[] gray_total = new int[256];
    double[] gray_pdf = new double[256];
    double[] gray_p1 = new double[256];
    double[] gray_mean_k = new double[256];
    double gray_mean_g = 0;
    double[] gray_var = new double[256];
    err = iImage.iPointerFromiImage(Image, ref Graymatrix[0, 0], Width, Height);
    print_error(err);
    // start sliding the matrix
    for (int i = 0; i < Height; i++)              // i index for cols ( 0~Hight-1)(Because that matrix index is start from 0)
    {
        for (int j = 0; j < Width; j++)           // j index for rows ( 0~Width-1)
        {
            gray_total[Graymatrix[i, j]] = gray_total[Graymatrix[i, j]] + 1;
        }
    }

    for (int i = 0; i < 256; i++) //算出各個灰階值的 PDF
    {
        gray_pdf[i] = gray_total[i] / (double)(Width * Height);
    }
```

```
chart_his.ChartAreas[0].AxisX.Title = "rk";
chart_his.ChartAreas[0].AxisY.Title = "nk";
chart_his.Series.Clear();
chart_his.Series.Add("nk(rk)");
chart_his.Series["nk(rk)"].ChartType = SeriesChartType.Column;
chart_his.Series["nk(rk)"].XValueType = ChartValueType.Int32;
chart_his.Series["nk(rk)"].YValueType = ChartValueType.Double;
chart_his.ChartAreas[0].AxisX.Minimum = 0;
chart_his.ChartAreas[0].AxisX.Maximum = 256;

int[] gray_level = new int[256];
for (int i = 0; i < 256; i++)
{
    gray_level[i] = i;
}
chart_his.Series["nk(rk)"].Points.DataBindXY(gray_level, gray_pdf);

for (int i = 0; i < 256; i++) //將PDF累加求出 p1
{
    for (int j = 0; j <= i; j++)
    {
        gray_p1[i] = gray_p1[i] + gray_pdf[j];
    }
}
```

```
for (int i = 0; i < 256; i++) //累積均值m(k)
{
    for (int j = 0; j <= i; j++)
    {
        gray_mean_k[i] = gray_mean_k[i] + (double)(j * gray_pdf[j]);
    }
}

for (int i = 0; i < 256; i++) //整個圖像平均灰度
{
    gray_mean_g = gray_mean_g + (double)(i * gray_pdf[i]);
}

for(int i = 0; i < 256; i++) //計算方差
{
    gray_var[i] = (gray_mean_g * gray_p1[i] - gray_mean_k[i]) * (gray_mean_g * gray_p1[i] - gray_mean_k[i]) / (gray_p1[i] * (1 - gray_p1[i]));
}

double max_num = 0;
for (int i = 0; i < 256; i++)
{
    if(gray_var[i] > max_num)
    {
        best_Threshold = i;
        max_num = gray_var[i];
    }
}
```

```
lab_best_thr.Text = Convert.ToString(best_Threshold);
for (int i = 0; i < Height; i++)              // i index for cols ( 0~Hight-1)(Because that matrix index is start from 0)
{
    for (int j = 0; j < Width; j++)           // j index for rows ( 0~Width-1)
    {
        if (Graymatrix[i, j] < best_Threshold)      // Compare if the image intensity of each pixel smaller than threshold
            Graymatrix[i, j] = 0;             // Set pixel to 0 (Black)
        else
            Graymatrix[i, j] = 255;           // Set pixel to 1 (White)
    }
}

IntPtr imgPtr = iImage.iVarPtr(ref Graymatrix[0, 0]);
err = iImage.iPointerToiImage(src2, imgPtr, Width, Height);
print_error(err2);
show_img(src2, pictureBox);
```

開運算

```
int kernel_size = 5;
byte[,] stru = new byte[,]{
    {0, 255, 255, 255, 0 },
    {255, 255, 255, 255, 255 },
    {255, 255, 255, 255, 255 },
    {255, 255, 255, 255, 255 },
    {0, 255, 255, 255, 0 },
};
int flag;
for (int i = 2; i < Height - 2; i++)//侵蝕
{
    for (int j = 2; j < Width - 2; j++)
    {
        flag = 1;
        for (int m = -2; m < 2; m++)
        {
            for (int n = -2; n < 2; n++)
            {
                //自身及領域中若有一個為0
                //則將該點設為0
                if (Graymatrix[i + m, j + n] != stru[m + 2, n + 2] && stru[m + 2, n + 2] == 255)
                {
                    flag = 0;
                    break;
                }
            }
```

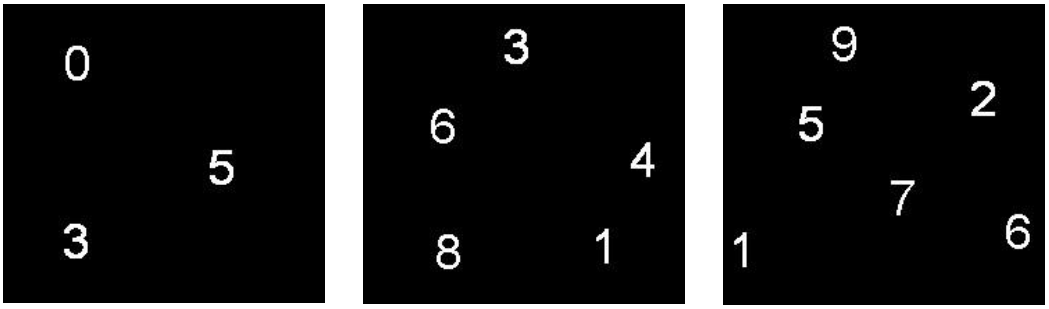
```
                if (flag == 0)
                {
                    break;
                }
            }
            if (flag == 0)
            {
                Graymatrix_ero[i, j] = (byte)0;
            }
            else
            {
                Graymatrix_ero[i, j] = (byte)255;
            }
        }
    }
for (int i = 2; i < Height - 2; i++)//膨脹
{
    for (int j = 2; j < Width - 2; j++)
    {
        flag = 1;
        for (int m = -2; m < 2; m++)
        {
            for (int n = -2; n < 2; n++)
            {
                //自身及領域中若有一個為0
                //則將該點設為0
                if (Graymatrix_ero[i + m, j + n] == stru[m + 2, n + 2] && stru[m + 2, n + 2] == 255)
                {
                    flag = 0;
                    break;
                }
            }
            if (flag == 0)
            {
                break;
            }
        }
```

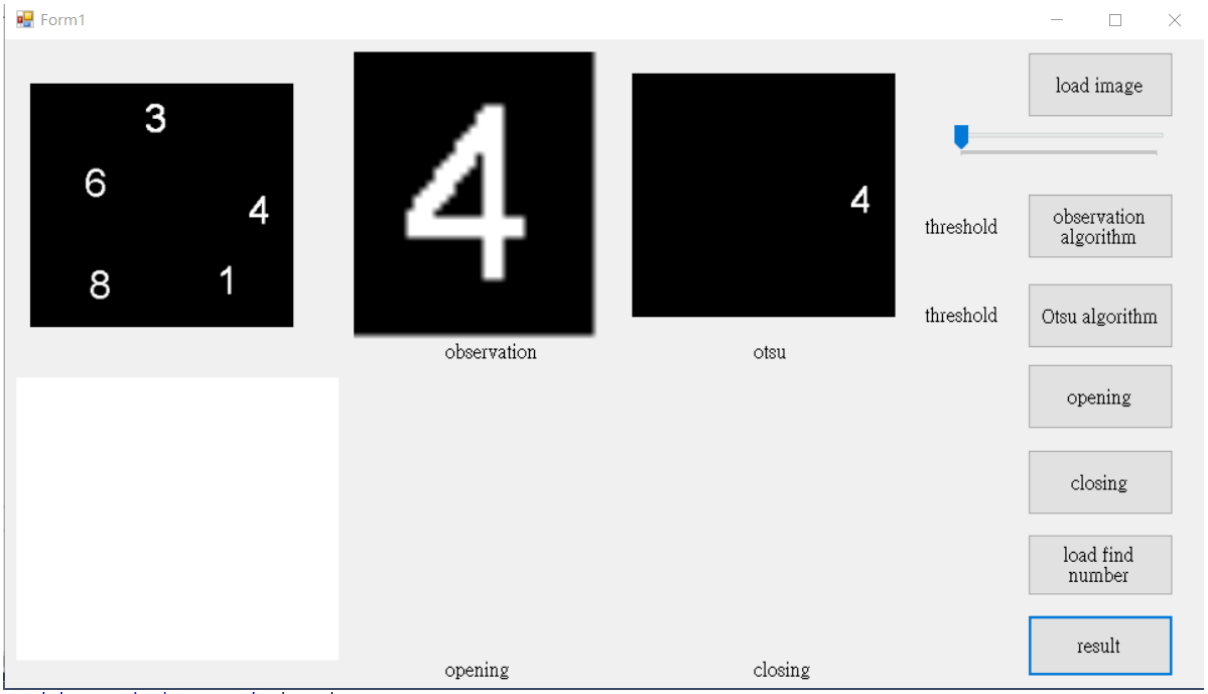
```
            if (flag == 0)
            {
                break;
            }
        }
        if (flag == 0)
        {
            Graymatrix_dil[i, j] = 255;
        }
        else
        {
            Graymatrix_dil[i, j] = 0;
        }
    }
}

IntPtr imgPtr = iImage.iVarPtr(ref Graymatrix_dil[0, 0]);
err2 = iImage.iPointerToiImage(src2, imgPtr, Width, Height);
print_error(err2);
show_img(src2, pictureBox);
```

閉運算與開運算侵蝕膨脹順序相反。

2. (50%) Hit-or-Miss Transform: please respectively recognize the characters present in the test images and displays it as output using morphological operations.






```
public static int find_pos_x;
public static int find_pos_y;
public void compare(IntPtr src, IntPtr src2, IntPtr src3, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);        // Get image width
    int Height = iImage.GetHeight(src);      // Get image height
    int Width2 = iImage.GetWidth(src3);        // Get image width
    int Height2 = iImage.GetHeight(src3);      // Get image height
    byte[,] image = new byte[Height, Width];
    byte[,] image_er = new byte[Height, Width];
    byte[,] image_in = new byte[Height, Width];
    byte[,] image_in_er = new byte[Height, Width];
    byte[,] number = new byte[Height2, Width2];
    byte[,] result = new byte[Height, Width];
    int[] pos = new int []{Width2, 0, Height2, 0};//左右上下

    err = iImage.iPointerFromiImage(src, ref image[0, 0], Width, Height);
    err3 = iImage.iPointerFromiImage(src3, ref number[0, 0], Width2, Height2);
    print_error(err);
    print_error(err3);
```

```
for(int i = 0; i < Height; i++)
{
    for(int j = 0; j < Width; j++)
    {
        image_in[i, j] = (byte)(255 - image[i, j]);
    }
}
for(int i = 0; i < Height2; i++)
{
    for(int j = 0; j < Width2; j++)
    {
        if(number[i, j] == 255)
        {
            if(j < pos[0])
            {
                pos[0] = j;
            }
            if(j > pos[1])
            {
                pos[1] = j;
            }
            if(i < pos[2])
            {
                pos[2] = i;
            }
            if(i > pos[3])
            {
                pos[3] = i;
            }
        }
    }
}
```

```
int Height_b = pos[3] - pos[2];
int Width_b = pos[1] - pos[0];
byte[,] b1 = new byte[Height_b, Width_b];
byte[,] b2 = new byte[Height_b, Width_b];

for(int i = 0; i < Height_b; i++)
{
    for(int j = 0; j < Width_b; j++)
    {
        b1[i, j] = number[i + pos[2], j + pos[0]];
    }
}

for (int i = 0; i < Height_b; i++)
{
    for (int j = 0; j < Width_b; j++)
    {
        b2[i, j] = (byte)(255 - b1[i, j]);
    }
}
```

```
int flag;
for (int i = (Height_b / 2); i < Height - (Height_b / 2); i++)//侵蝕
{
    for (int j = (Width_b / 2); j < Width - (Width_b / 2); j++)
    {
        flag = 1;
        for (int m = -1 * (Height_b / 2); m < (Height_b / 2); m++)
        {
            for (int n = -1 * (Width_b / 2); n < (Width_b / 2); n++)
            {
                //自身及領域中若有一個為0
                //則將該點設為0
                if (image[i + m, j + n] != b1[m + (Height_b / 2), n + (Width_b / 2)] && b1[m + (Height_b / 2), n + (Width_b / 2)] == 255)
                {
                    flag = 0;
                    break;
                }
            }
            if (flag == 0)
            {
                break;
            }
        }
```

```
        if (flag == 0)
        {
            image_er[i, j] = 0;
        }
        else
        {
            image_er[i, j] = 255;
        }
    }
}

for (int i = (Height_b / 2); i < Height - (Height_b / 2); i++)//侵蝕
{
    for (int j = (Width_b / 2); j < Width - (Width_b / 2); j++)
    {
        flag = 1;
        for (int m = -1 * (Height_b / 2); m < (Height_b / 2); m++)
        {
            for (int n = -1 * (Width_b / 2); n < (Width_b / 2); n++)
            {
                //自身及領域中若有一個為0
                //則將該點設為0
                if (image_in[i + m, j + n] != b2[m + (Height_b / 2), n + (Width_b / 2)] && b2[m + (Height_b / 2), n + (Width_b / 2)] == 255)
                {
                    flag = 0;
                    break;
                }
            }
        }
        if (flag == 0)
        {
            image_in_er[i, j] = 0;
        }
        else
        {
            image_in_er[i, j] = 255;
        }
    }
}


for (int i = 0; i < Height; i++)
{
    for (int j = 0; j < Width; j++)
    {
        if (image_er[i, j] == 255 && image_in_er[i, j] == 255)
        {
            find_pos_y = i;
            find_pos_x = j;
        }
    }
```

```
for (int i = -1 * (Height_b / 2); i < (Height_b / 2); i++)
{
    for (int j = -1 * (Width_b / 2); j < (Width_b / 2); j++)
    {
        result[find_pos_y + i, find_pos_x + j] = bl[i + (Height_b / 2), j + (Width_b / 2)];
    }
}

IntPtr imgPtr = iImage.iVarPtr(ref result[0, 0]);
err2 = iImage.iPointerToiImage(src2, imgPtr, Width, Height);
print_error(err2);
show_img(src2, pictureBox);
```