資工所碩一 R07922164 林群崴

本次作業採用 C++配合 OpenCV 完成

Thinning

步驟一：二值化，利用先前 HW2 的 function

```
//先進行二值化
Mat b_img = binary(img, THREASHOLD, widthLimit, heightLimit);
```

步驟二：用8 × 8的 blocks 進行 downsample

```
Mat lena_down(64, 64, CV_8U, Scalar(0));
for (int height = 0; height < 64; height++) {
    for (int width = 0; width < 64; width++) {
        lena_down.at<uchar>(height, width) = b_img.at<uchar>(8 * height, 8 *
width);
    }
}
```

步驟三：進行 Thinning

Main Function:

```
bool flag = true;

 while (flag) {

    flag = false;

    Mat old = img_thin.clone();

    //step 1: mark the interior/border pixels
    Mat s1 = Matmark_interior_border(img_thin, widthLimit, heightLimit);

    //step 2: pair relationship operator
    Mat s2 = mark_pair_relationship(s1, widthLimit, heightLimit);

    //step 3: check and delete the deletable pixels
    Mat s3 = step3(s1);

    for (int i = 0; i < heightLimit; i++) {
        for (int j = 0; j < widthLimit; j++) {
```

```
                    if (s3.at<uchar>(i, j) == 1 && s2.at<uchar>(i, j) == 1)
                         img_thin.at<uchar>(i, j) = 0;
               }
          }

          for (int i = 0; i < heightLimit; i++) {
               for (int j = 0; j < widthLimit; j++) {
                    if (img_thin.at<uchar>(i, j) != old.at<uchar>(i, j))
                         flag = true;
               }
          }

     }
```

marked-interior/border-pixel operator

```
Mat Matmark_interior_border(Mat img, int widthLimit, int heightLimit) {
     Mat result(heightLimit, widthLimit, CV_8U, Scalar(0));
     int x1, x2, x3, x4;
     for (int i = 0; i < heightLimit; i++) {
          for (int j = 0; j < widthLimit; j++) {
               if (img.at<uchar>(i, j) > 0) {
                    x1 = x2 = x3 = x4 = 0;
                    if (i == 0) {
                         if (j == 0) {
                              x1 = img.at<uchar>(i, j + 1);
                              x4 = img.at<uchar>(i + 1, j);
                         }
                         else if (j == widthLimit - 1) {
                              x3 = img.at<uchar>(i, j - 1);
                              x4 = img.at<uchar>(i + 1, j);
                         }
                         else {
                              x1 = img.at<uchar>(i, j + 1);
                              x3 = img.at<uchar>(i, j - 1);
                              x4 = img.at<uchar>(i + 1, j);
                         }
                    }
                    else if (i == heightLimit - 1) {
```

```
                    if (j == 0) {
                        x1 = img.at<uchar>(i, j + 1);
                        x2 = img.at<uchar>(i - 1, j);
                    }
                    else if (j == widthLimit - 1) {
                        x2 = img.at<uchar>(i - 1, j);
                        x3 = img.at<uchar>(i, j - 1);
                    }
                    else {
                        x1 = img.at<uchar>(i, j + 1);
                        x2 = img.at<uchar>(i - 1, j);
                        x3 = img.at<uchar>(i, j - 1);
                    }
                }
                else {
                    if (j == 0) {
                        x1 = img.at<uchar>(i, j + 1);
                        x2 = img.at<uchar>(i - 1, j);
                        x4 = img.at<uchar>(i + 1, j);
                    }
                    else if (j == widthLimit - 1) {
                        x2 = img.at<uchar>(i - 1, j);
                        x3 = img.at<uchar>(i, j - 1);
                        x4 = img.at<uchar>(i + 1, j);
                    }
                    else {
                        x1 = img.at<uchar>(i, j + 1);
                        x2 = img.at<uchar>(i - 1, j);
                        x3 = img.at<uchar>(i, j - 1);
                        x4 = img.at<uchar>(i + 1, j);
                    }

                    x1 /= 255;
                    x2 /= 255;
                    x3 /= 255;
                    x4 /= 255;

                    char a1 = h1('1', x1 + '0');
```

```
                    char a2 = h1(a1, x2 + '0');
                    char a3 = h1(a2, x3 + '0');
                    char a4 = h1(a3, x4 + '0');

                    if (a4 == 'b')
                        result.at<uchar>(i, j) = 2;
                    else
                        result.at<uchar>(i, j) = 1;
                }
            }
        }
    }
    return result;
}
```

說明：

$$h(c,d) = \begin{cases} c & if\ c = d \\ b & if\ c \neq d \end{cases}$$

$$f(c) = \begin{cases} b & if\ c = b \\ i & if\ c \neq b \end{cases}$$

pair relationship operator

```
Mat mark_pair_relationship(Mat img, int widthLimit, int heightLimit) {
    Mat pair(heightLimit, widthLimit, CV_8U, Scalar(0));
    int x1, x2, x3, x4;
    for (int i = 0; i < heightLimit; i++) {
        for (int j = 0; j < widthLimit; j++) {
            if (img.at<uchar>(i, j) > 0) {
                x1 = x2 = x3 = x4 = 0;
                if (i == 0) {
                    if (j == 0) {
                        x1 = img.at<uchar>(i, j + 1);
                        x4 = img.at<uchar>(i + 1, j);
                    }
                    else if (j == widthLimit - 1) {
                        x3 = img.at<uchar>(i, j - 1);
                        x4 = img.at<uchar>(i + 1, j);
                    }
                    else {
```

```
                    x1 = img.at<uchar>(i, j + 1);
                    x3 = img.at<uchar>(i, j - 1);
                    x4 = img.at<uchar>(i + 1, j);
               }
          }
          else if (i == heightLimit - 1) {
               if (j == 0) {
                    x1 = img.at<uchar>(i, j + 1);
                    x2 = img.at<uchar>(i - 1, j);
               }
               else if (j == widthLimit - 1) {
                    x2 = img.at<uchar>(i - 1, j);
                    x3 = img.at<uchar>(i, j - 1);
               }
               else {
                    x1 = img.at<uchar>(i, j + 1);
                    x2 = img.at<uchar>(i - 1, j);
                    x3 = img.at<uchar>(i, j - 1);
               }
          }
          else {
               if (j == 0) {
                    x1 = img.at<uchar>(i, j + 1);
                    x2 = img.at<uchar>(i - 1, j);
                    x4 = img.at<uchar>(i + 1, j);
               }
               else if (j == widthLimit - 1) {
                    x2 = img.at<uchar>(i - 1, j);
                    x3 = img.at<uchar>(i, j - 1);
                    x4 = img.at<uchar>(i + 1, j);
               }
               else {
                    x1 = img.at<uchar>(i, j + 1);
                    x2 = img.at<uchar>(i - 1, j);
                    x3 = img.at<uchar>(i, j - 1);
                    x4 = img.at<uchar>(i + 1, j);
               }
          }
```

```
                    if (h2(x1, 1) + h2(x2, 1) + h2(x3, 1) + h2(x4, 1) >= 1 &&
img.at<uchar>(i, j) == 2)
                        pair.at<uchar>(i, j) = 1;
                    else
                        pair.at<uchar>(i, j) = 2;
                }
            }
        }
        return pair;
}
```

說明：

$$h(a,m) = \begin{cases} 1 & if\ a = m \\ 0 & otherwise \end{cases}$$

$$採用4-connected：output = \begin{cases} q & if \sum_{n=1}^{4} h(x_n,m) < \theta \ \vee \ x_0 \neq l \\ p & if \sum_{n=1}^{4} h(x_n,m) \geq \theta \ \wedge \ x_0 = l \end{cases}$$

marked-pixel connected shrink operator 先做類似 Yokoi Connectivity Number 處理。

結果如下：（64x64 放大 300%）