2021/11/25 陳嘉政 r10922171 HW7

程式碼

Step1: 將原圖做 **binarize**，接著做 **downsampling**。

Step2: 進入遞迴前先計算 **image** 的 **sum**，用來後續與結果做比對。

```
while (1) {
    int value = Sum(downsampled_image);
```

```
int Sum(Mat input)
{
    int total = 0;
    for (int i = 0; i < input.rows; i++) {
        for (int j = 0; j < input.cols; j++) {
            if (input.at<uint8_t>(i, j) == 255)
                total += 1;
        }
    }
    return total;
}
```

Step3: 將 **downsampled image** 做 **Yokoi operator**，函式與作業 **6** 相同就沒貼了。

```
/*Yokoi operator*/
Mat Yokoi_image = Mat::zeros(64, 64, CV_8UC1);
for (int i = 0; i < downsampled_image.rows; i++) {
    for (int j = 0; j < downsampled_image.cols; j++) {
        if (downsampled_image.at<uint8_t>(i, j) == 255) {
            int startRow = i > 0 ? 1 : 0;
            int startCol = j > 0 ? 1 : 0;
            int sizeRow = startRow == 1 ? 3 : 2;
            int sizeCol = startCol == 1 ? 3 : 2;
            sizeRow = i == 63 ? 2 : sizeRow;
            sizeCol = j == 63 ? 2 : sizeCol;
            Mat temp = downsampled_image(cv::Rect(j - startCol, i - startRow, sizeCol, sizeRow));
            Yokoi_image.at<uint8_t>(i, j) = Yokoi(temp, i, j);
        }
    }
}
```

Step4: 將 **Yokoi** 後的 **image** 拿來做 **pair relationship operator**，**input** 分別為 **Yokoi image**，**row**，**column** 及此 **pixel** 的 **value**。若 **value** 不為 **1** 直接 **pair image** 中此點 **assign** 為 **q**，否則計算此點的四連通鄰居，若有 **pixel** 為 **1** 則 **count** 加 **1**，最後如果 **count >= 1** 則此點

assign 為 p，否則 assign 為 q。

```
char Pair(Mat input, int i, int j, int value)
{

    if (value != 1)
        return 'q';
    else {
        int di[] = { 0, -1, 0, 1 };
        int dj[] = { 1, 0, -1, 0 };
        int count = 0;
        for (int n = 0; n < 4; n++) {
            if (i + di[n] < 0) continue;
            else if (i + di[n] > 63) continue;
            else if (j + dj[n] < 0) continue;
            else if (j + dj[n] > 63) continue;
            else {
                if (input.at<uint8_t>(i + di[n], j + dj[n]) == 1)
                    count++;
            }

        }
        if (count == 0)
            return 'q';
        else
            return 'p';
    }
}
```

Step5: 宣告 **Connected shrinking image**，將 **downsampled image(**原圖**)** 複製到 **connected shrinking image**，接著每個點做，如果此點對應到的 **Pair image** 的點值為 **p** 以及此點值為 **255**，代表我們要考慮此點要不要刪。接著與 **Yokoi** 的做法相同，以此點為中心畫出適當大小的 **temp** 矩陣，以及 **row**，**column** 一起當 **Yokoi(**作業 **6** 的函式**)**的 **input**，因為 **Yokoi** 的回傳值為 **0-5** 的整數，若回傳值為 **1** 則將此點的值設成 **0(**刪掉**)**。

```
/*Connected shriking operator*/
Mat Connected_shrinking_image;
downsampled_image.copyTo(Connected_shrinking_image);

for (int i = 0; i < Connected_shrinking_image.rows; i++) {
    for (int j = 0; j < Connected_shrinking_image.cols; j++) {
        if (Pair_image.at<uint8_t>(i, j) == 'p' && Connected_shrinking_image.at<uint8_t>(i, j) == 255) {
            int startRow = i > 0 ? 1 : 0;
            int startCol = j > 0 ? 1 : 0;
            int sizeRow = startRow == 1 ? 3 : 2;
            int sizeCol = startCol == 1 ? 3 : 2;
            sizeRow = i == 63 ? 2 : sizeRow;
            sizeCol = j == 63 ? 2 : sizeCol;
            Mat temp = Connected_shrinking_image(cv::Rect(j - startCol, i - startRow, sizeCol, sizeRow));
            if (Yokoi(temp, i, j) == 1)
                Connected_shrinking_image.at<uint8_t>(i, j) = 0;
        }
    }
}
```

Step6: 跑完所有點後，若結果的 **sum** 不等於原圖的 **sum**，代表圖仍有變動，所以將結果圖當成原圖，重複做 **Step2,3,4,5**，直到結果與原圖的 **sum** 相同就跳離迴圈，表示圖已經不變。

```
if (Sum(Connected_shrinking_image) == value) {
    Connected_shrinking_image.copyTo(result);
    break;
}
Connected_shrinking_image.copyTo(downsampled_image);
```

結果

為放大過後的 **64x64** 圖