電腦視覺作業七

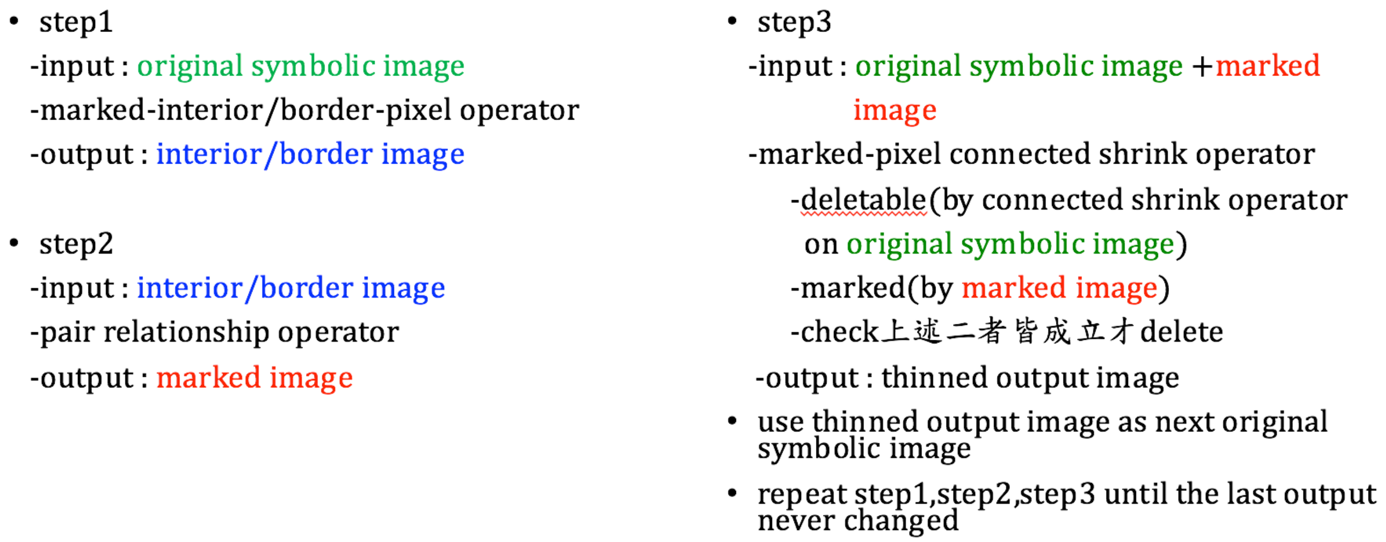
Computer Vision HW7

R04525092 工科碩二 鄭力文

https://github.com/Mike-Zheng/NTU-Computer-Vision-I-

使用語言 C++ with openCV

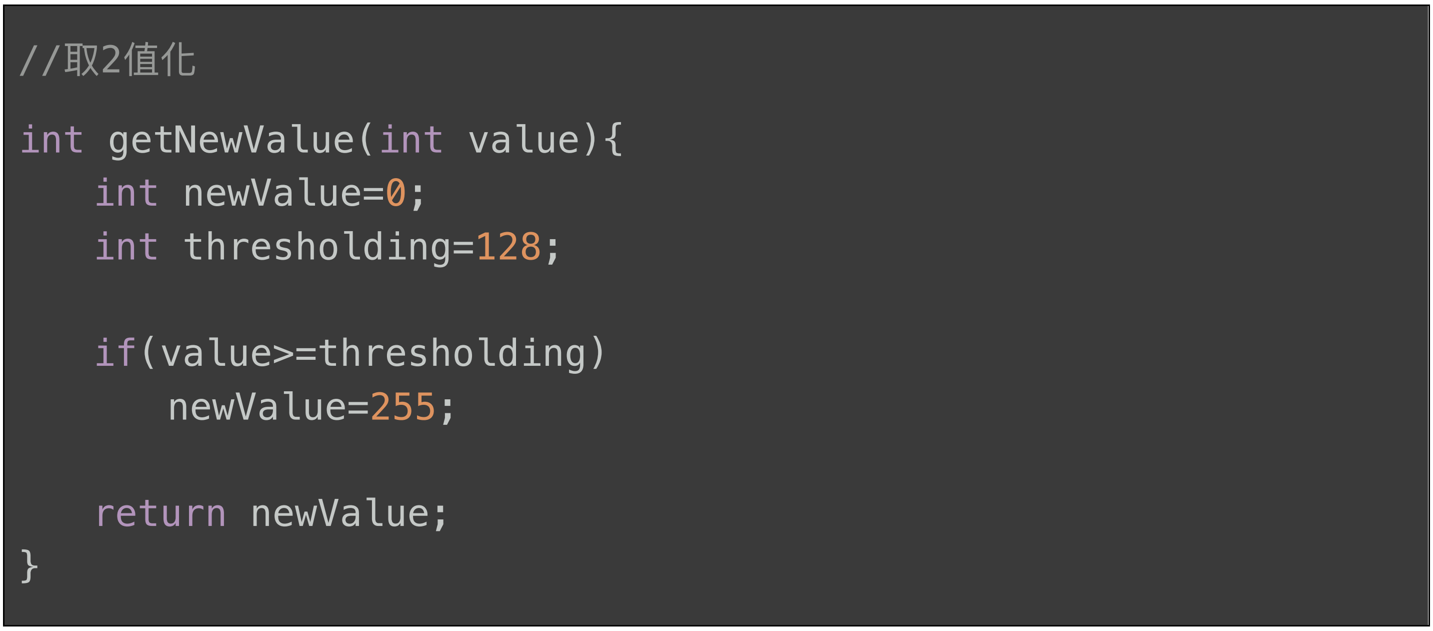
Write a program to generate thinned image.
Down sample lena.bmp from 512*512 to 64*64 first.
Sample pixels positions at each 8*8 top-left corner, so everyone will get the same answer .



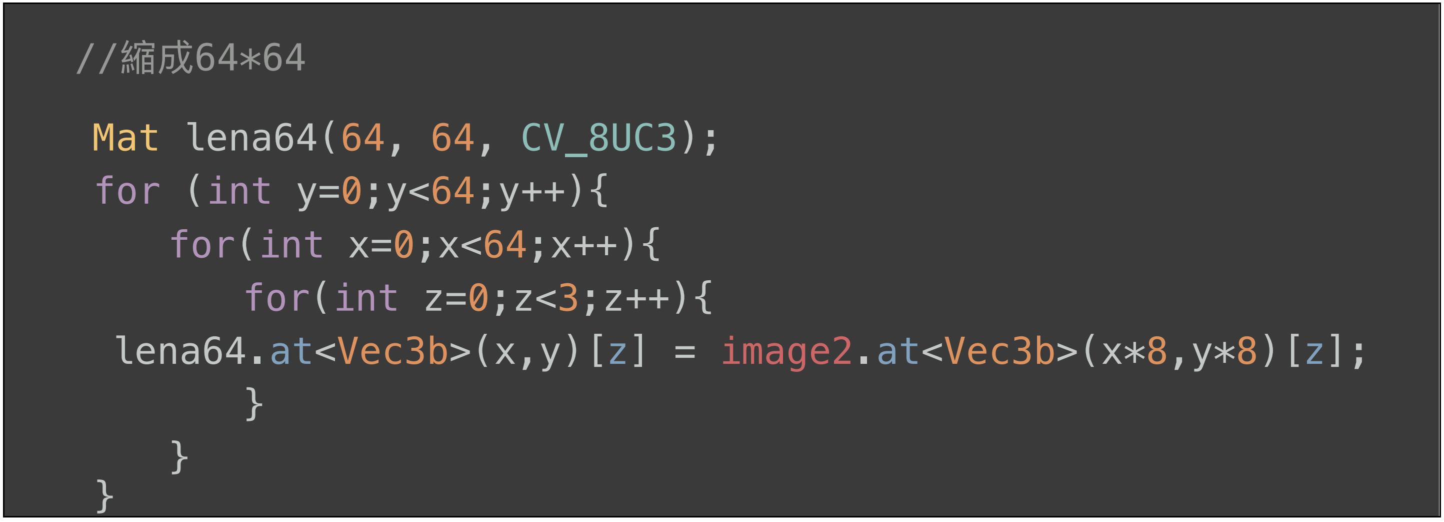
- step1
  - -input : original symbolic image
  - -marked-interior/border-pixel operator
  - -output : interior/border image
- step2
  - -input : interior/border image
  - -pair relationship operator
  - -output : marked image
- step3
  - -input : original symbolic image +marked image
  - -marked-pixel connected shrink operator
    - -deletable(by connected shrink operator on original symbolic image)
    - -marked(by marked image)
    - -check上述二者皆成立才delete
  - -output : thinned output image
- use thinned output image as next original symbolic image
- repeat step1,step2,step3 until the last output never changed

1. Binary & Down sample lena.bmp from 512*512 to64*64

膨脹將每一點透過 kernel 進行運算，該點中心吻合 kernel 的周邊為 255。



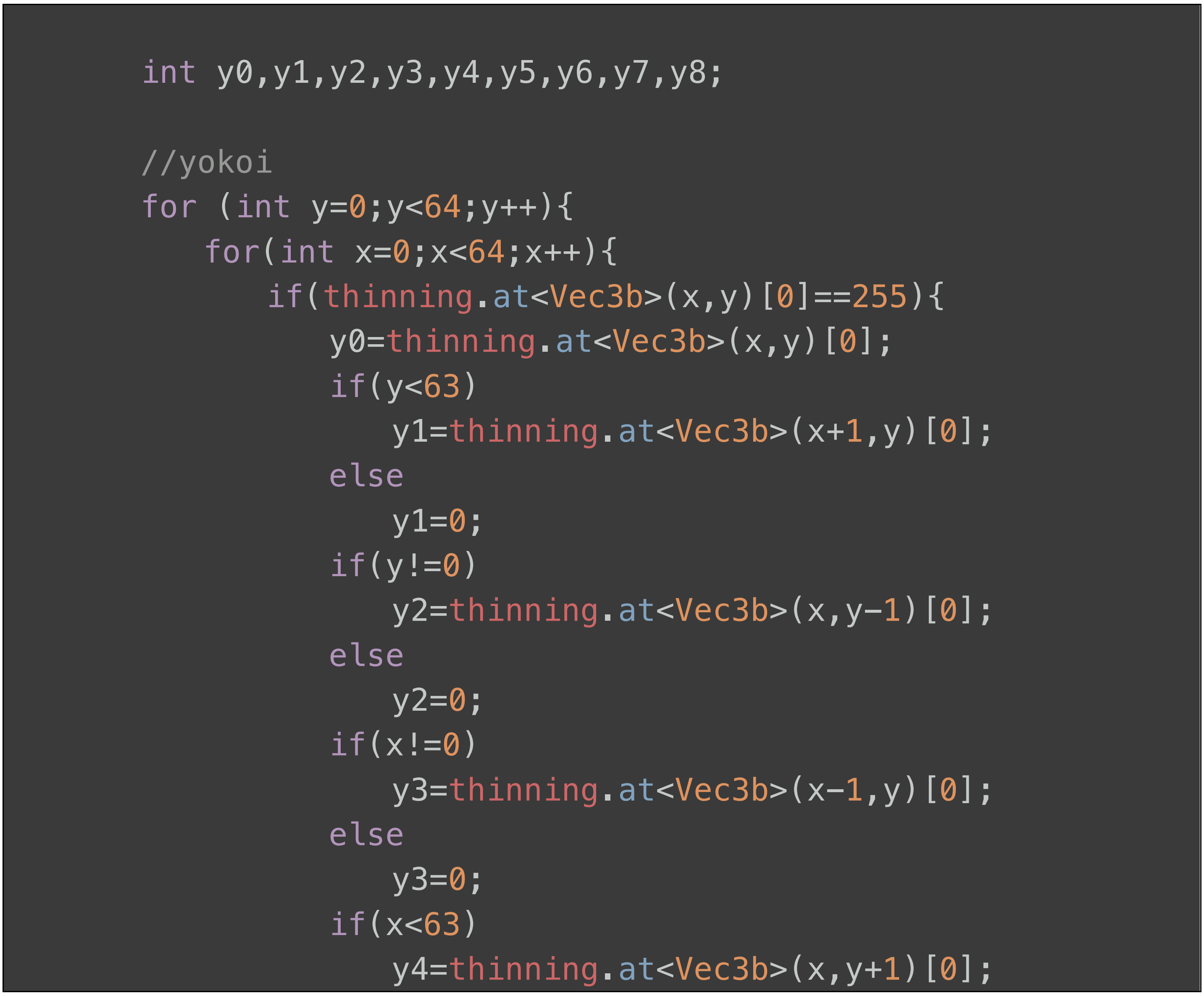
```
//取2值化

int getNewValue(int value){
    int newValue=0;
    int thresholding=128;

    if(value>=thresholding)
        newValue=255;

    return newValue;
}
```



```
//縮成64*64

Mat lena64(64, 64, CV_8UC3);
for (int y=0;y<64;y++){
    for(int x=0;x<64;x++){
        for(int z=0;z<3;z++){
 lena64.at<Vec3b>(x,y)[z] = image2.at<Vec3b>(x*8,y*8)[z];
        }
    }
}
```

2. Mark-Interior/Border-Pixel

透過 yokoi 做出結果得出 5=interior 並以四聯通進行 mark。

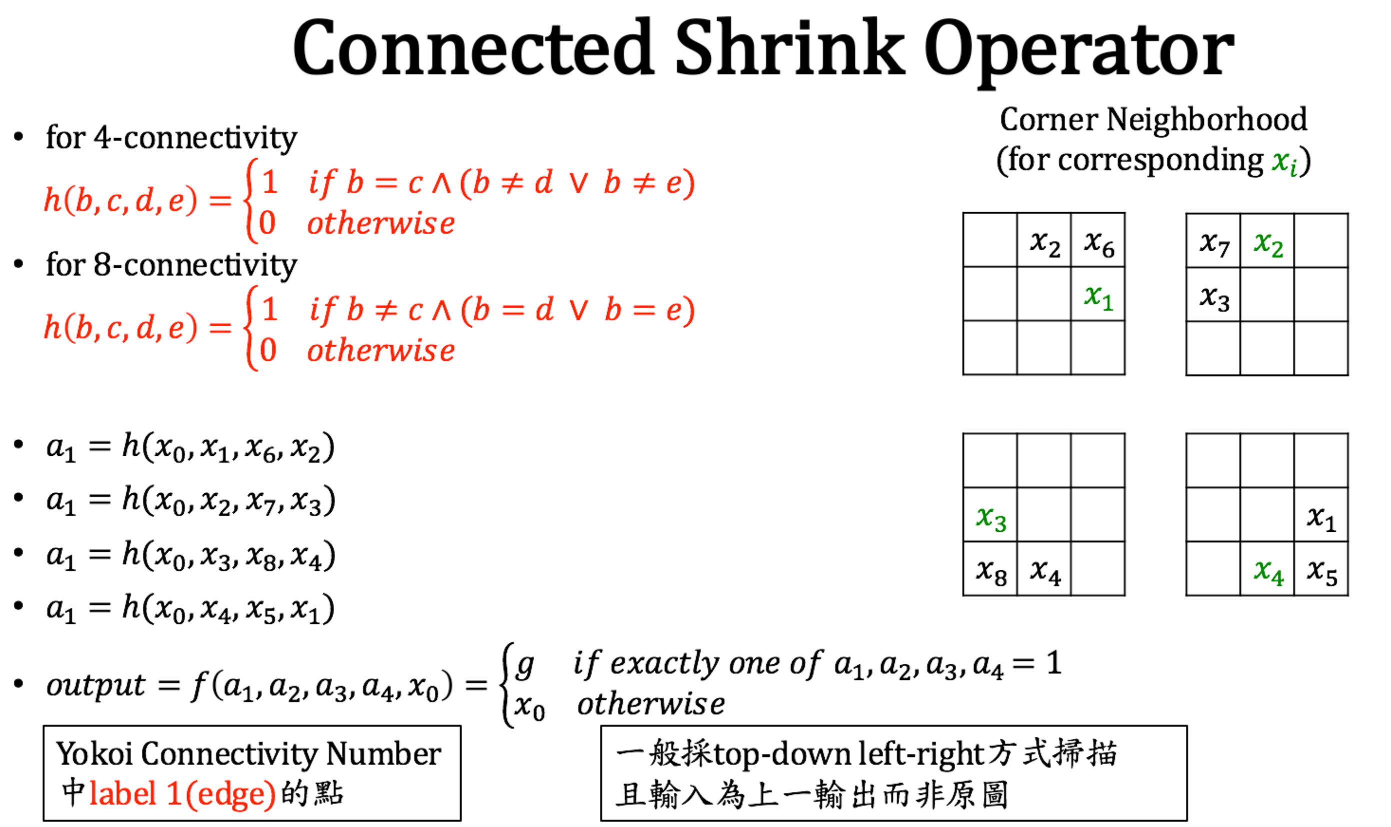


```
int y0,y1,y2,y3,y4,y5,y6,y7,y8;

//yokoi
for (int y=0;y<64;y++){
    for(int x=0;x<64;x++){
        if(thinning.at<Vec3b>(x,y)[0]==255){
            y0=thinning.at<Vec3b>(x,y)[0];
            if(y<63)
                y1=thinning.at<Vec3b>(x+1,y)[0];
            else
                y1=0;
            if(y!=0)
                y2=thinning.at<Vec3b>(x,y-1)[0];
            else
                y2=0;
            if(x!=0)
                y3=thinning.at<Vec3b>(x-1,y)[0];
            else
                y3=0;
            if(x<63)
                y4=thinning.at<Vec3b>(x,y+1)[0];
```

```
                    else
                        y4=0;
                    if(x<63&&y<63)
                        y5=thinning.at<Vec3b>(x+1,y+1)[0];
                    else
                        y5=0;
                    if(y!=0&&x<63)
                        y6=thinning.at<Vec3b>(x+1,y-1)[0];
                    else
                        y6=0;
                    if(y!=0&&x!=0)
                        y7=thinning.at<Vec3b>(x-1,y-1)[0];
                    else
                        y7=0;
                    if(x!=0&&y<63)
                        y8=thinning.at<Vec3b>(x-1,y+1)[0];
                    else
                        y8=0;

                    char a1=yokoi_h(y0,y1,y6,y2);
                    char a2=yokoi_h(y0,y2,y7,y3);
                    char a3=yokoi_h(y0,y3,y8,y4);
                    char a4=yokoi_h(y0,y4,y5,y1);
                    yokoiArray[x][y]=yokoi_f(a1,a2,a3,a4);
                }
                else{
                    yokoiArray[x][y]=8;
                }
            }
        }
```

3.Connected shrink

4 連通進行比較。



Connected Shrink Operator

- for 4-connectivity

$$h(b,c,d,e)=\begin{cases}1 & if\ b=c\wedge(b\neq d\vee b\neq e)\\ 0 & otherwise\end{cases}$$

- for 8-connectivity

$$h(b,c,d,e)=\begin{cases}1 & if\ b\neq c\wedge(b=d\vee b=e)\\ 0 & otherwise\end{cases}$$

- $a_1=h(x_0,x_1,x_6,x_2)$
- $a_1=h(x_0,x_2,x_7,x_3)$
- $a_1=h(x_0,x_3,x_8,x_4)$
- $a_1=h(x_0,x_4,x_5,x_1)$
- $output=f(a_1,a_2,a_3,a_4,x_0)=\begin{cases}g & if\ exactly\ one\ of\ a_1,a_2,a_3,a_4=1\\ x_0 & otherwise\end{cases}$

Yokoi Connectivity Number 中label 1(edge)的點

一般採top-down left-right方式掃描且輸入為上一輸出而非原圖

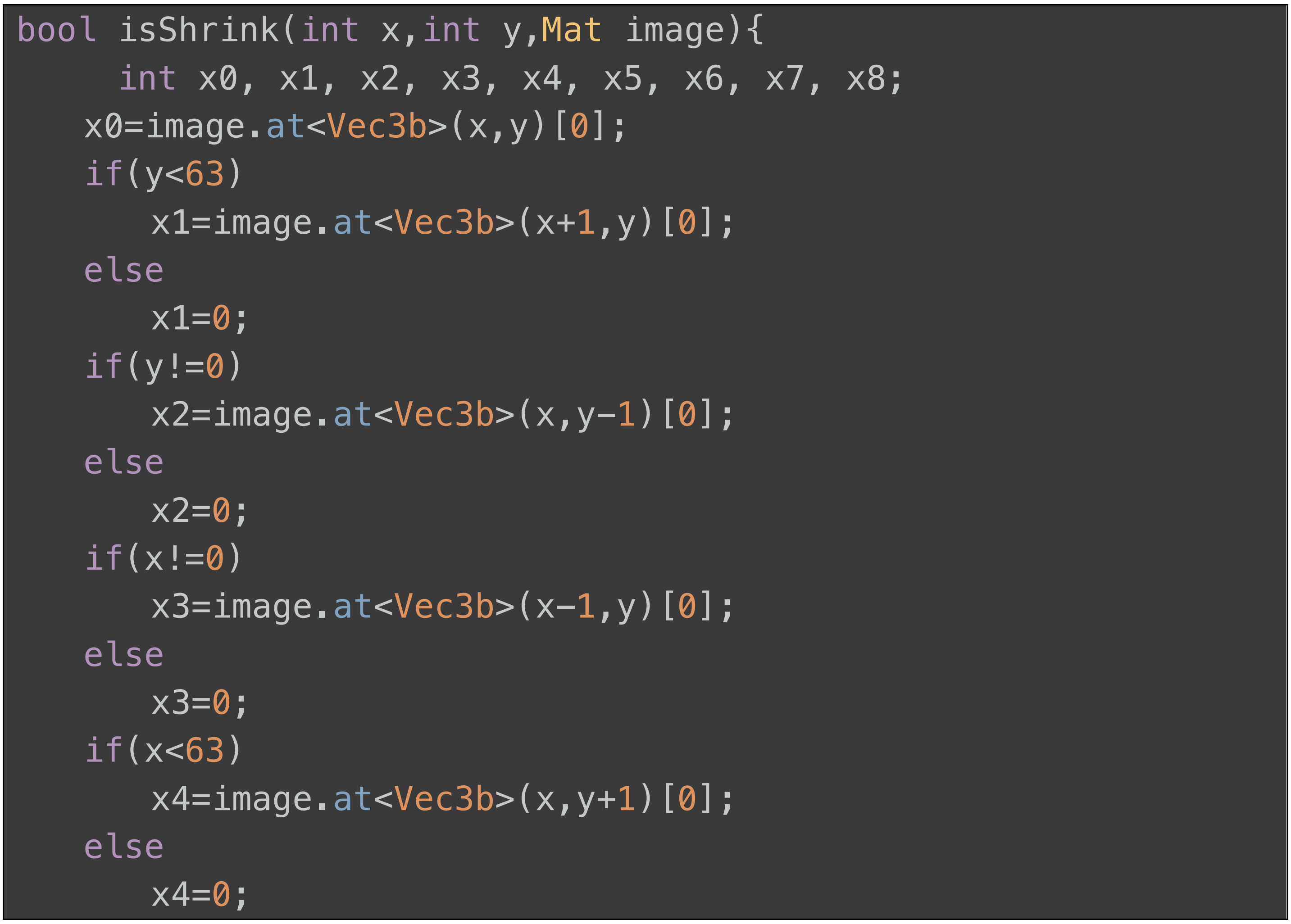
Corner Neighborhood (for corresponding x_i)

	x_2	x_6
		x_1

x_7	x_2	
x_3		

x_3		
x_8	x_4	

		x_1
	x_4	x_5



```
bool isShrink(int x,int y,Mat image){
     int x0, x1, x2, x3, x4, x5, x6, x7, x8;
   x0=image.at<Vec3b>(x,y)[0];
   if(y<63)
      x1=image.at<Vec3b>(x+1,y)[0];
   else
      x1=0;
   if(y!=0)
      x2=image.at<Vec3b>(x,y-1)[0];
   else
      x2=0;
   if(x!=0)
      x3=image.at<Vec3b>(x-1,y)[0];
   else
      x3=0;
   if(x<63)
      x4=image.at<Vec3b>(x,y+1)[0];
   else
      x4=0;
```

```
    if(x<63&&y<63)
        x5=image.at<Vec3b>(x+1,y+1)[0];
    else
        x5=0;
    if(y!=0&&x<63)
        x6=image.at<Vec3b>(x+1,y-1)[0];
    else
        x6=0;
    if(y!=0&&x!=0)
        x7=image.at<Vec3b>(x-1,y-1)[0];
    else
        x7=0;
    if(x!=0&&y<63)
        x8=image.at<Vec3b>(x-1,y+1)[0];
    else
        x8=0;

    char a1 = h(x0, x1, x6, x2);
    char a2 = h(x0, x2, x7, x3);
    char a3 = h(x0, x3, x8, x4);
    char a4 = h(x0, x4, x5, x1);
    if(f(a1, a2, a3, a4, x0)=='s')
        return true;
    else
        return false;

}
```

結果