Python+OpenCV图像处理

图像梯度

sobel算子函数及其使用

李大羊

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dst	，	计算结果
src	，	原始图像
ddepth	，	处理结果图像深度
dx	，	x轴方向
dy	，	y轴方向
ksize	，	核大小

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dst ， 计算结果

src ， 原始图像

ddepth ， 处理结果图像深度

dx ， x轴方向

dy ， y轴方向

ksize ， 核大小

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dst ， 计算结果

src ， 原始图像

ddepth ， 处理结果图像深度

dx ， x轴方向

dy ， y轴方向

ksize ， 核大小

Sobel算子

```
dst = cv2.Sobel( src , ddepth , dx , dy , [ksize] )
```

dst ， 计算结果

src ， 原始图像

ddepth ， 处理结果图像深度

dx ， x轴方向

dy ， y轴方向

ksize ， 核大小

Sobel算子

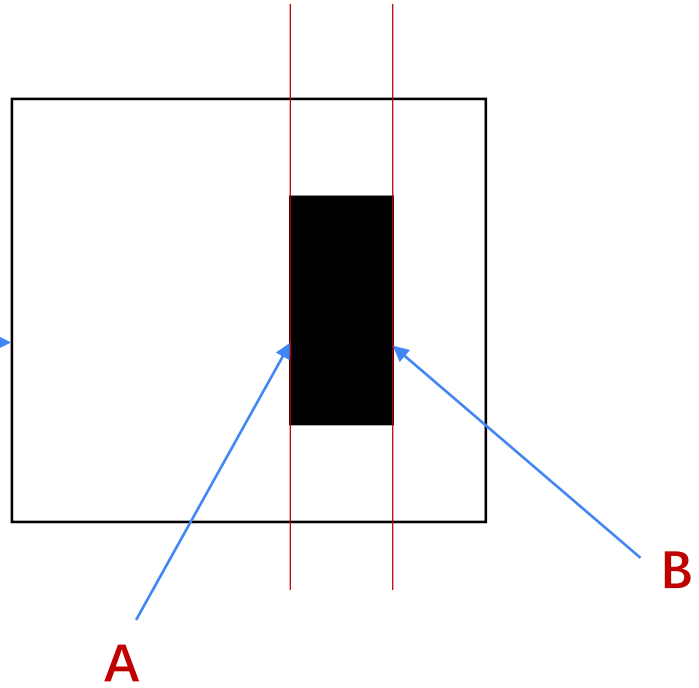
dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

ddepth ， 处理结果图像深度。

通常情况下，可以将该参数的值设置为**-1**，让处理结果与原始图像保持一致。

Sobel算子

示例图像



水平梯度、边界：

示例图像内，黑色块位置：

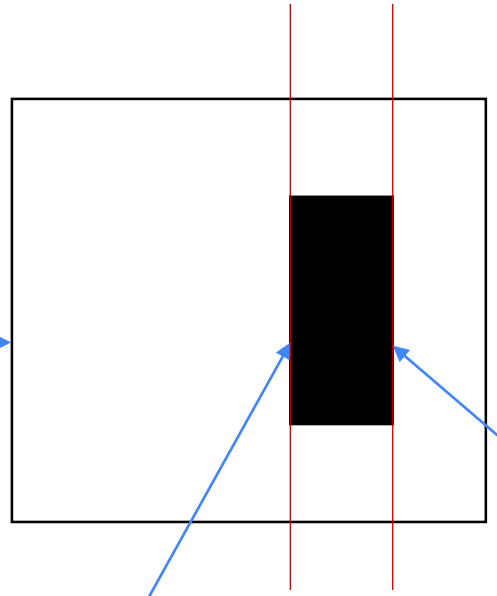
A列和B列，其右侧像素值与左侧像素值的差值不为零，是边界；

其余列，右侧像素值与左侧像素值的差值均为零，不是边界。

提示：256色位图中，白色点像素值255，黑色点像素值0.

Sobel算子

示例图像



水平梯度、边界：

针对A列，右侧减去左侧的值为负数。

针对B列，右侧减去左侧的值为正数。

针对A列处理结果：

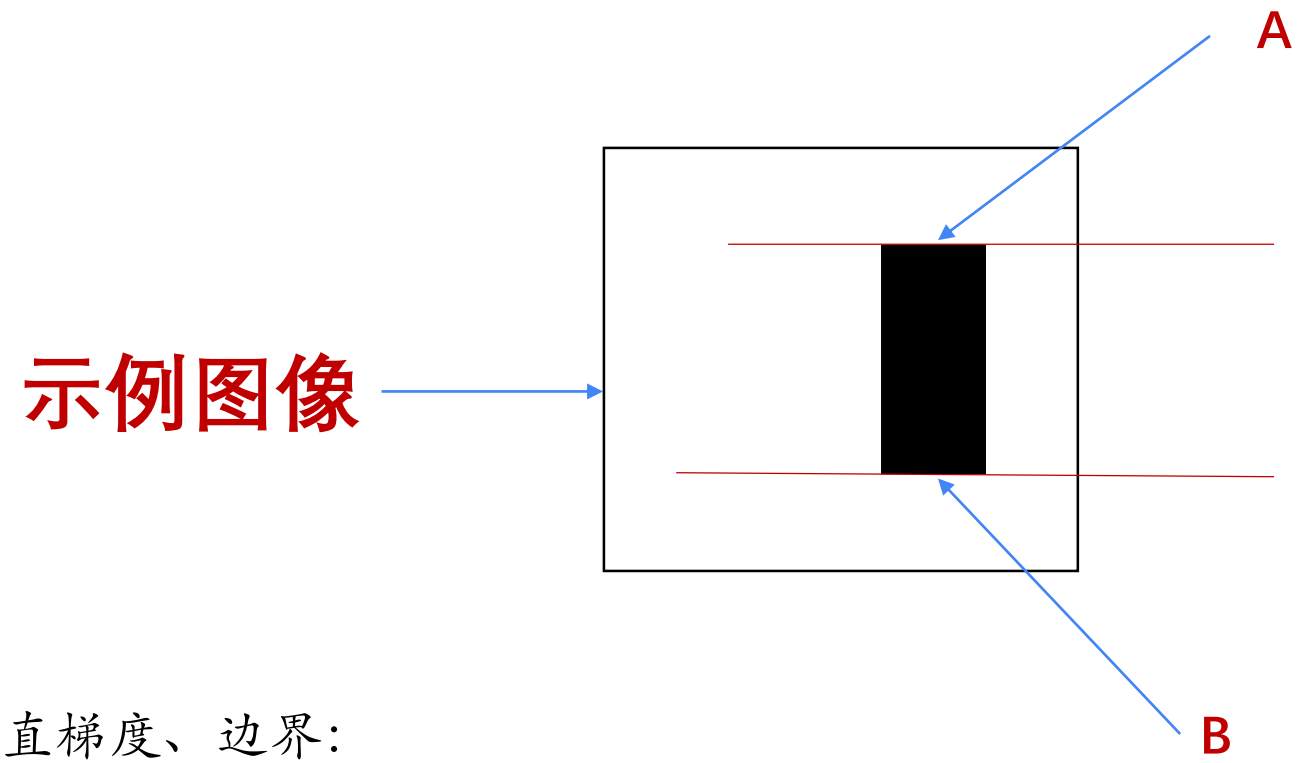
1）处理为0，

2）取绝对值。

直接计算

计算绝对值

Sobel算子



垂直梯度、边界：

示例图像内，黑色块位置：

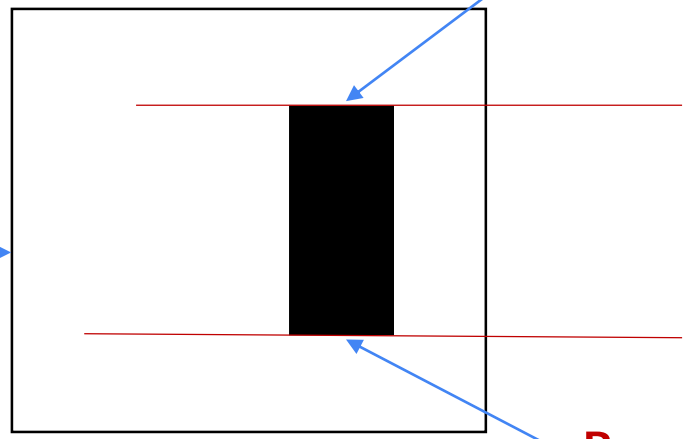
A行和B行，其下一行像素值与上一行侧像素值的差值不为零，是边界；

其余行，其下一行像素值与上一行像素值的差值均为零，不是边界。

提示：256色位图中，白色点像素值255，黑色点像素值0.

Sobel算子

示例图像



垂直梯度、边界：

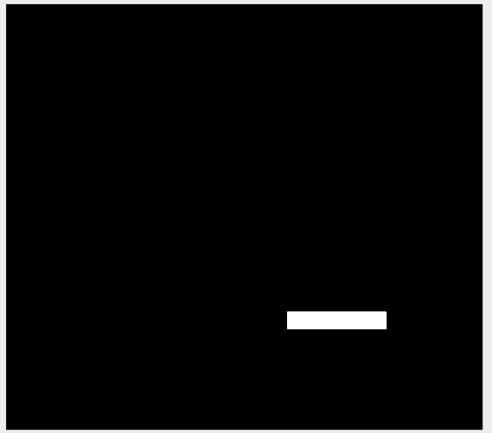
针对A行，下一行减去上一行的值为负数。

针对B行，下一行减去上一行的值为正数。

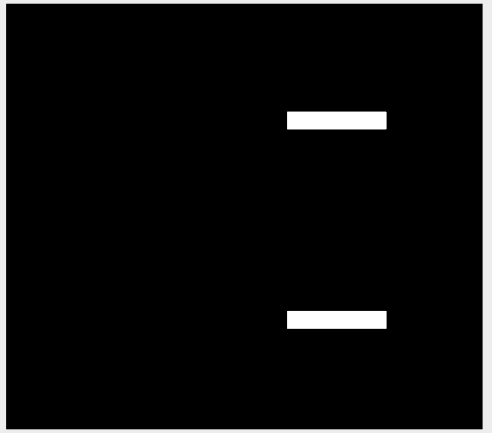
对A行的计算结果：

1）处理为0，

2）取绝对值。



直接计算



计算绝对值

Sobel算子

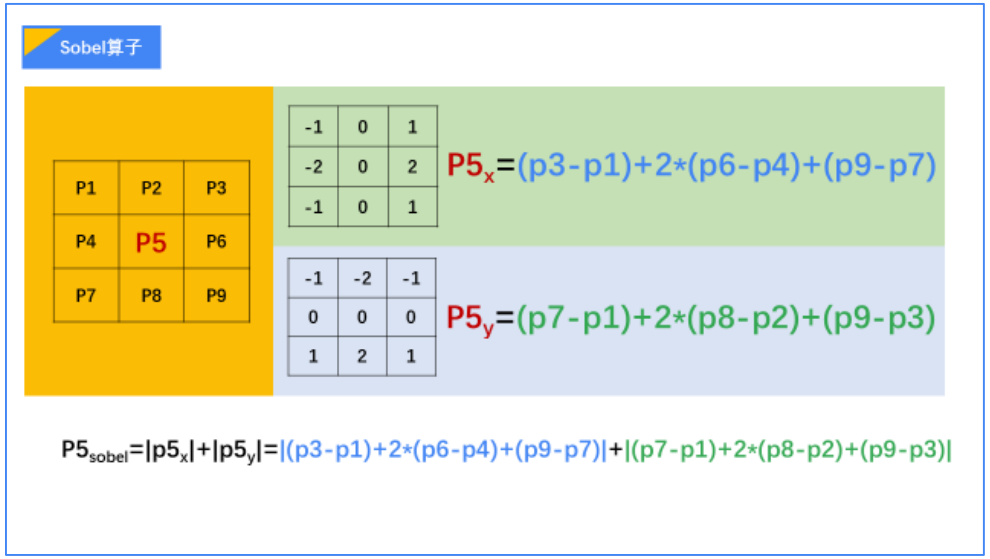
dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

ddepth　　，处理结果图像深度。

实际操作中，计算梯度值可能会出现负数。

通常处理的图像是np.uint8类型，如果结果也是该类型，所有负数会自动截断为0，发生信息丢失。

所以，通常计算时，使用更高的数据类型cv2.CV_64F，取绝对值后，再转换为np.uint8（cv2.CV_8U）类型。



Sobel算子

dst = cv2.convertScaleAbs(src [, alpha[, beta]])

作用：将原始图像src转换为256色位图。

Sobel算子

dst = cv2.convertScaleAbs(src [, alpha[, beta]])

公式:

目标图像=调整（原始图像*alpha+beta）

Sobel算子

dst = cv2.convertScaleAbs(src [, alpha[, beta]])

直接调整

目标图像= cv2.convertScaleAbs (原始图像)

Sobel算子

P1	P2	P3
P4	P5	P6
P7	P8	P9

-1	0	1
-2	0	2
-1	0	1

$$P5_x=(p3-p1)+2*(p6-p4)+(p9-p7)$$

-1	-2	-1
0	0	0
1	2	1

$$P5_y=(p7-p1)+2*(p8-p2)+(p9-p3)$$

$$P5_{sobel}=|p5_x|+|p5_y|=|(p3-p1)+2*(p6-p4)+(p9-p7)|+|(p7-p1)+2*(p8-p2)+(p9-p3)|$$

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dst	，	计算结果
src	，	原始图像
ddepth	，	处理结果图像深度
dx	，	x轴方向
dy	，	y轴方向
ksize	，	核大小

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dst ， 计算结果

src ， 原始图像

ddepth ， 处理结果图像深度

dx ， x轴方向

dy ， y轴方向

ksize ， 核大小

Sobel算子

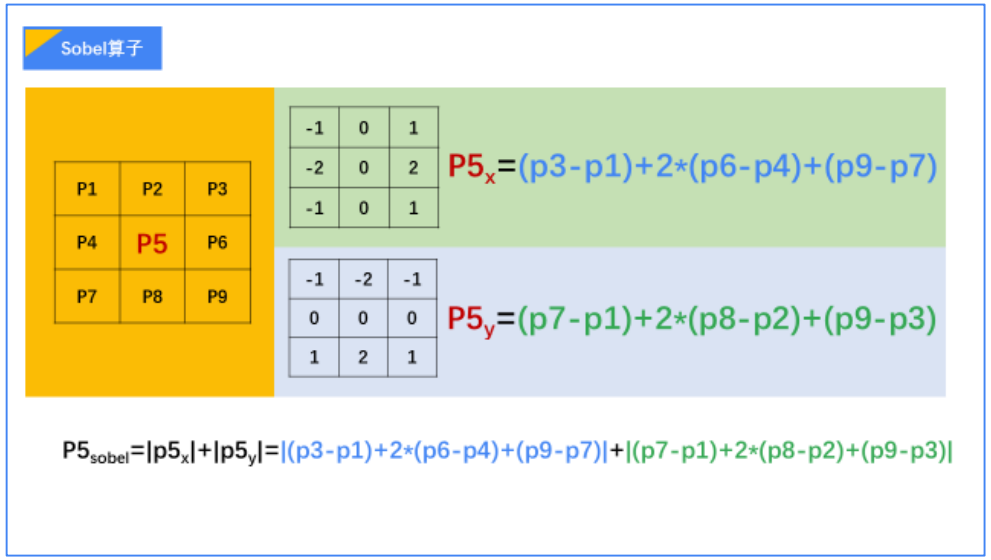
dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dx ， x轴方向

dy ， y轴方向

计算x方向梯度：【dx=1,dy=0】

计算y方向梯度：【dx=0,dy=1】



Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dx ， x轴方向

dy ， y轴方向

计算x方向梯度: 【dx=1,dy=0】

计算y方向梯度: 【dx=0,dy=1】

-1	0	1
-2	0	2
-1	0	1

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dx ， x轴方向

dy ， y轴方向

计算x方向梯度: 【dx=1,dy=0】

计算y方向梯度: 【dx=0,dy=1】

-1	-2	-1
0	0	0
1	2	1

Sobel算子

计算sobel结果

方式1

dx=1， dy=1

```
dst = cv2.Sobel( src , ddepth , 1 , 1 )
```

方式2

分别计算dx和dy后相加

```
dx= cv2.Sobel( src , ddepth , 1 , 0 )
dy= cv2.Sobel( src , ddepth , 0 , 1 )
dst= dx + dy
```

dst= dx * 系数1+ dy *系数2

Sobel算子

函数

dst=cv2.addWeighted(src1 , alpha , src2 , beta , gamma)

功能：计算两幅图像的权重和。

Sobel算子

函数

dst=cv2.addWeighted(src1 , alpha , src2 , beta , gamma)

dst ， 计算结果

src1 ， 源图像1

alpha ， 源图像1的系数

src2 ， 源图像2

beta ， 源图像2的系数

gamma ， 修正值

Sobel算子

函数

dst=cv2.addWeighted(src1 , alpha , src2 , beta , gamma)

关系

$$\texttt{dst}(I) = \texttt{saturate}(\texttt{src1}(I) * \texttt{alpha} + \texttt{src2}(I) * \texttt{beta} + \texttt{gamma})$$

函数

dst=cv2.addWeighted(src1 , alpha , src2 , beta , gamma)

示例　　dst=cv2.addWeighted(src1 , 0.5 , src2 , 0.5 , 0)

Sobel算子

dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

dst ， 计算结果

src ， 原始图像

ddepth ， 处理结果图像深度

dx ， x轴方向

dy ， y轴方向

ksize ， 核大小

Sobel算子

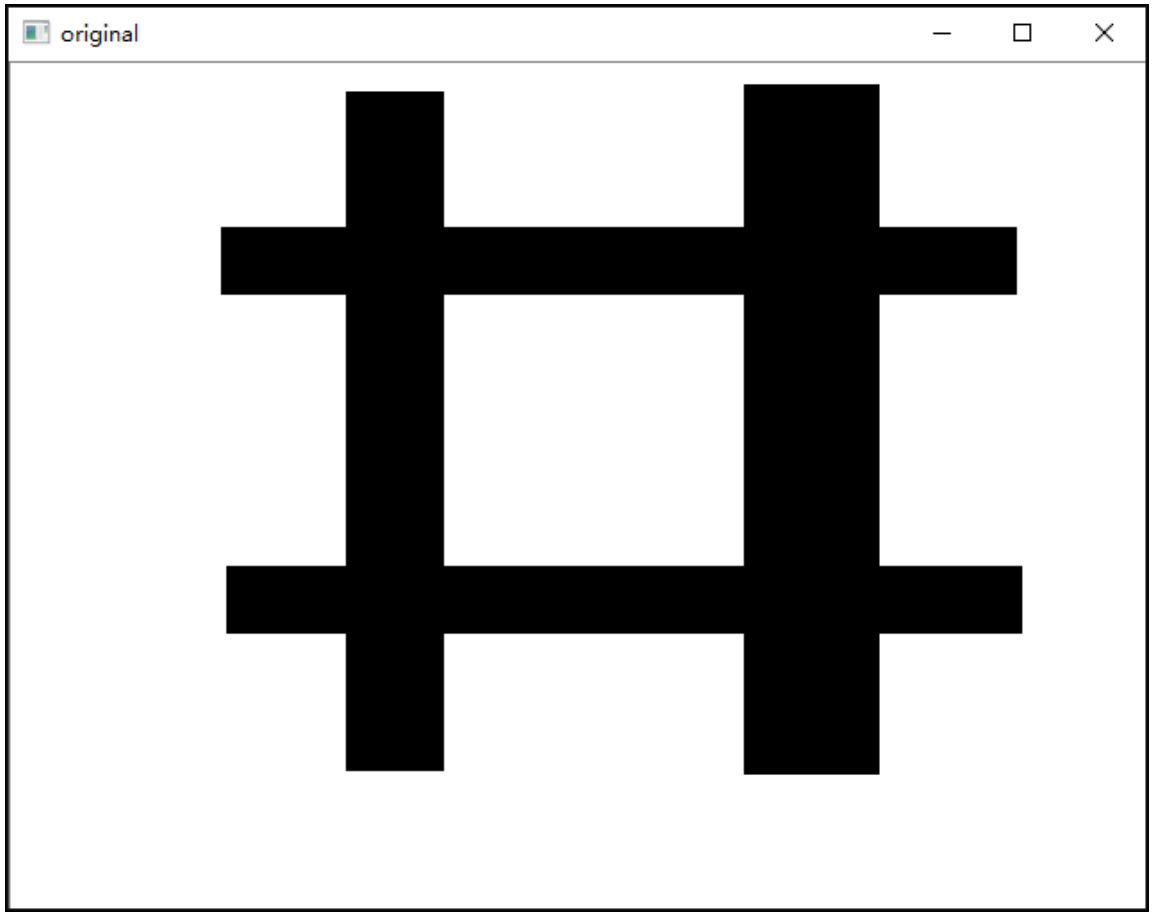
dst = cv2.Sobel(src , ddepth , dx , dy , [ksize])

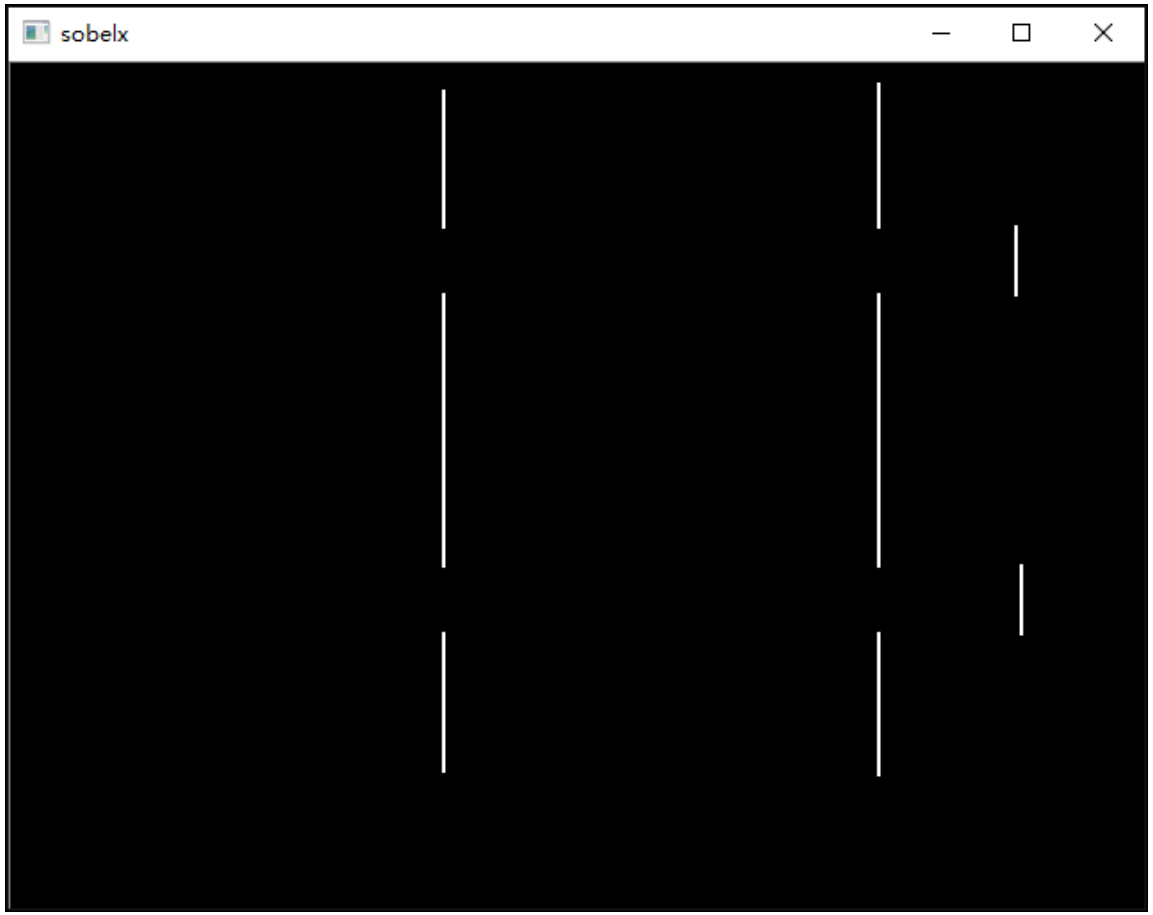
dst	，	计算结果
src	，	原始图像
ddepth	，	处理结果图像深度
dx	，	x轴方向
dy	，	y轴方向
ksize	，	核大小

Sobel算子

```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(o,-1,1,0)
cv2.imshow("original",o)
cv2.imshow("x",sobelx)
cv2.waitKey()
cv2.destroyAllWindows()
```

Sobel算子

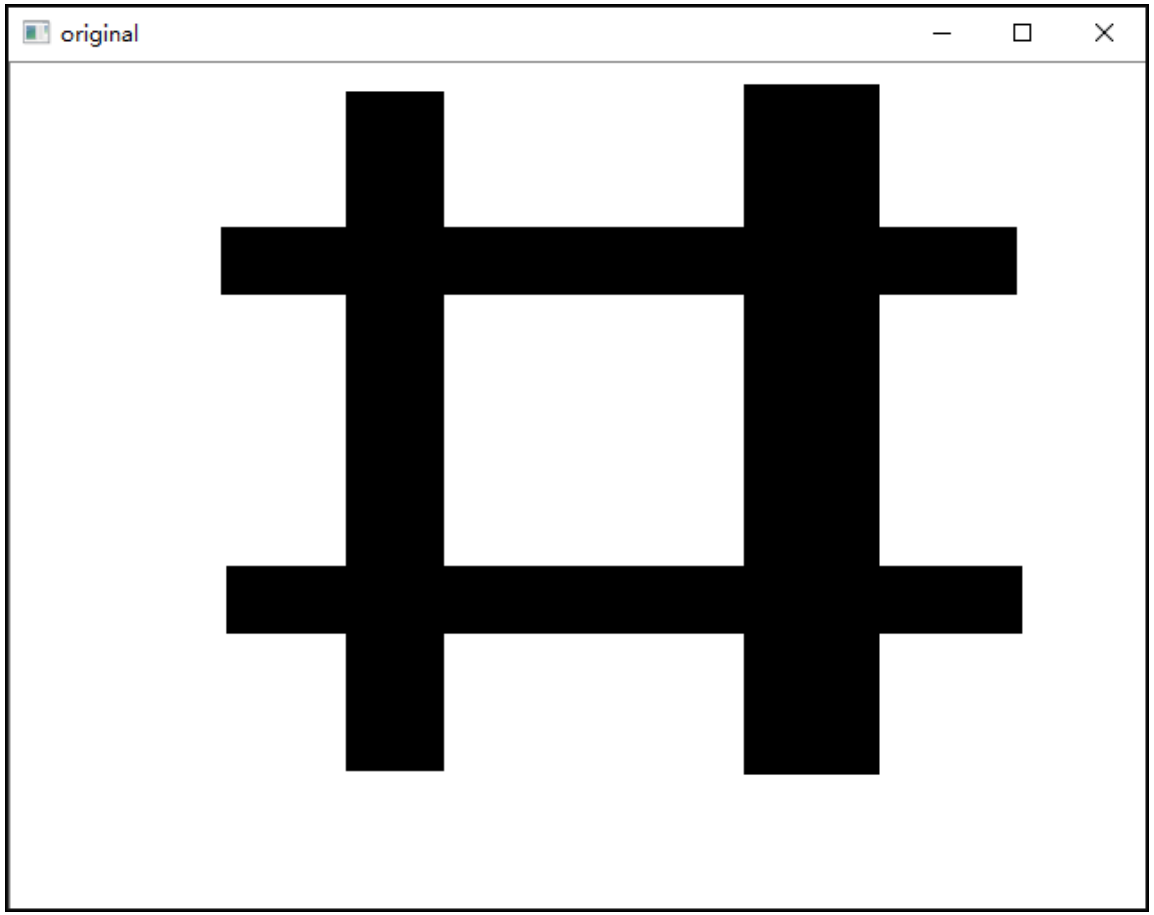


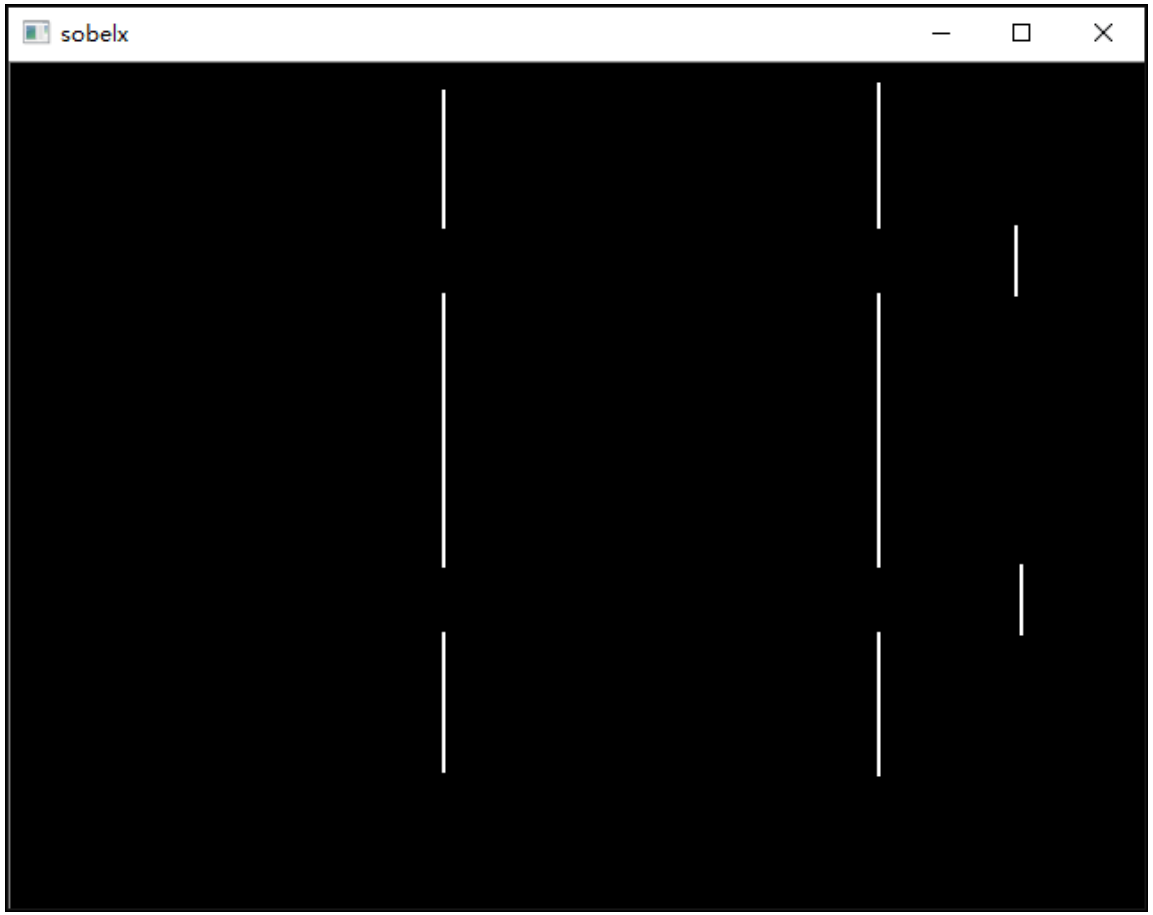


Sobel算子

```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(o,cv2.CV_64F,1,0)
cv2.imshow("original",o)
cv2.imshow("x",sobelx)
cv2.waitKey()
cv2.destroyAllWindows()
```

Sobel算子

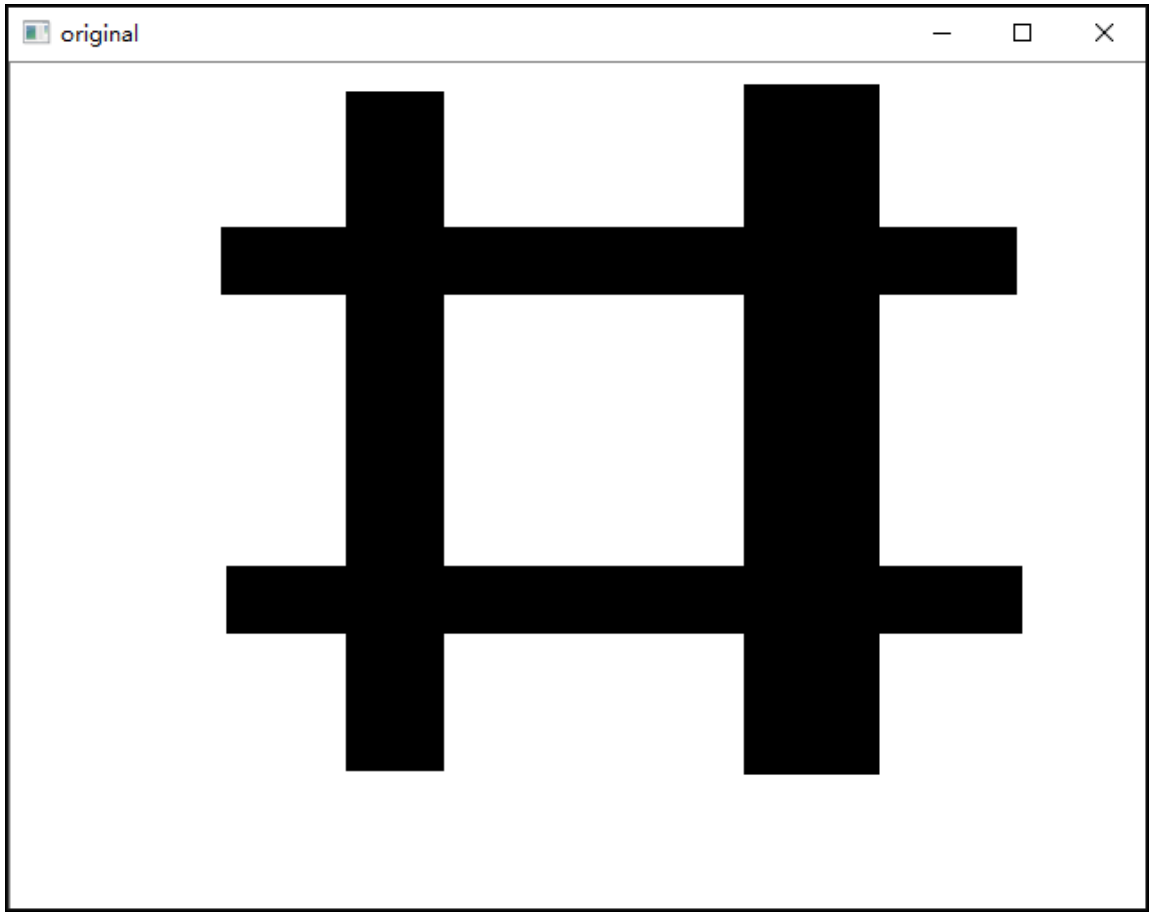


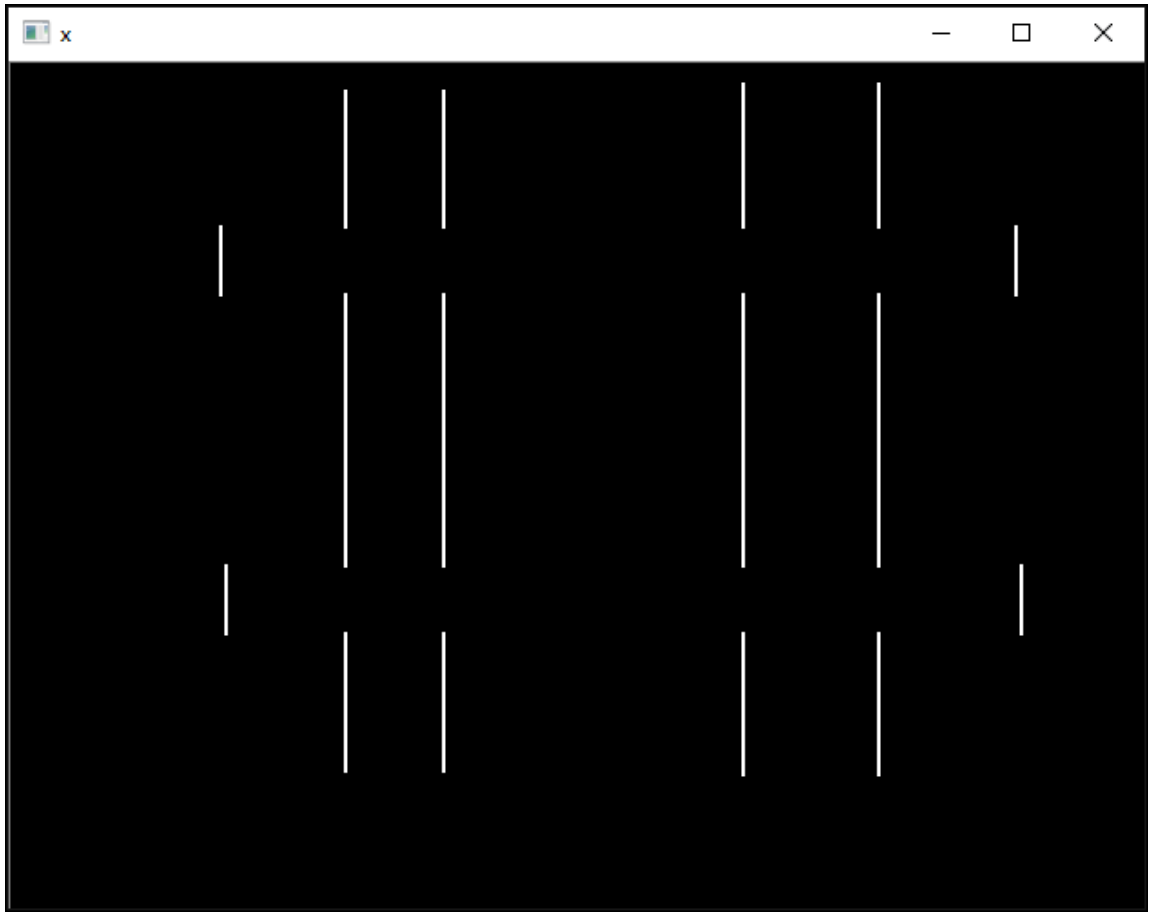


Sobel算子

```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(o,cv2.CV_64F,1,0)
sobelx = cv2.convertScaleAbs(sobelx)   # 转回uint8
cv2.imshow("original",o)
cv2.imshow("x",sobelx)
cv2.waitKey()
cv2.destroyAllWindows()
```

Sobel算子

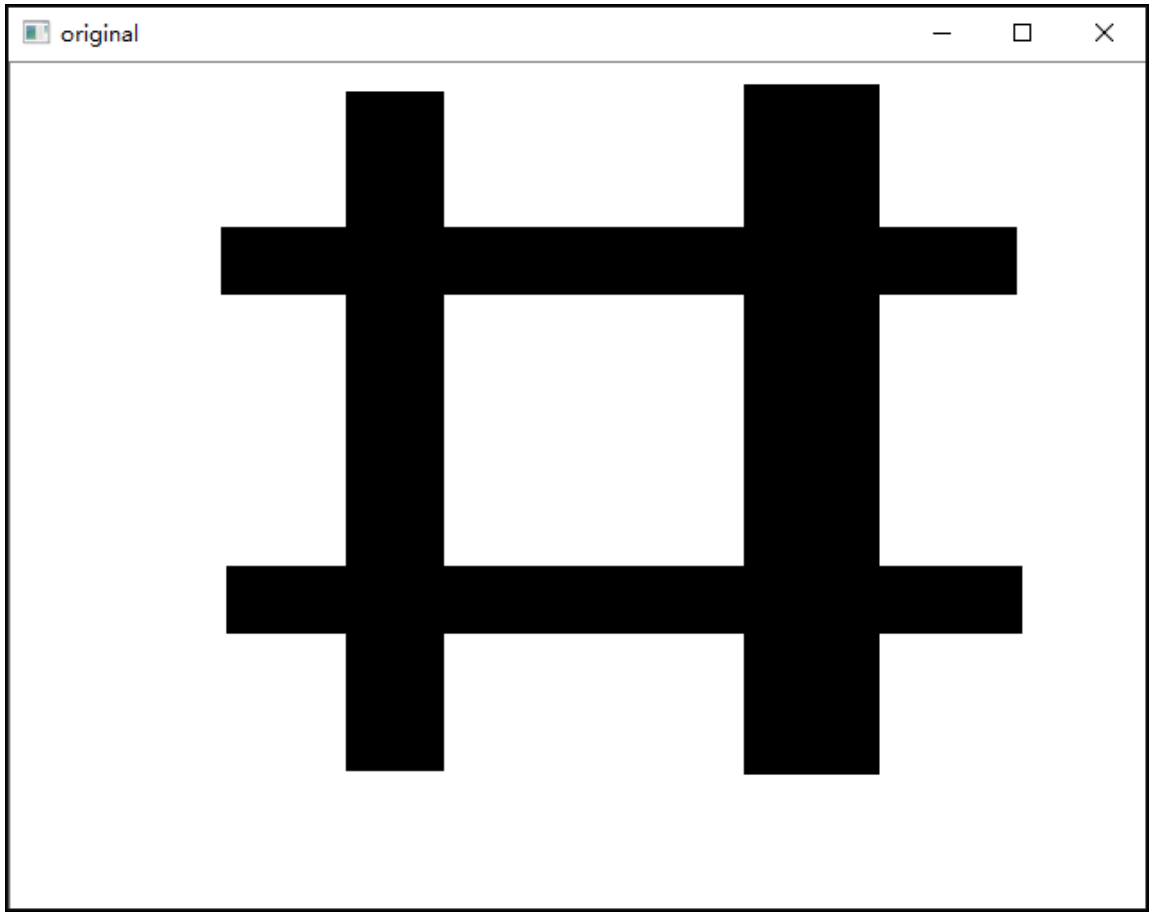


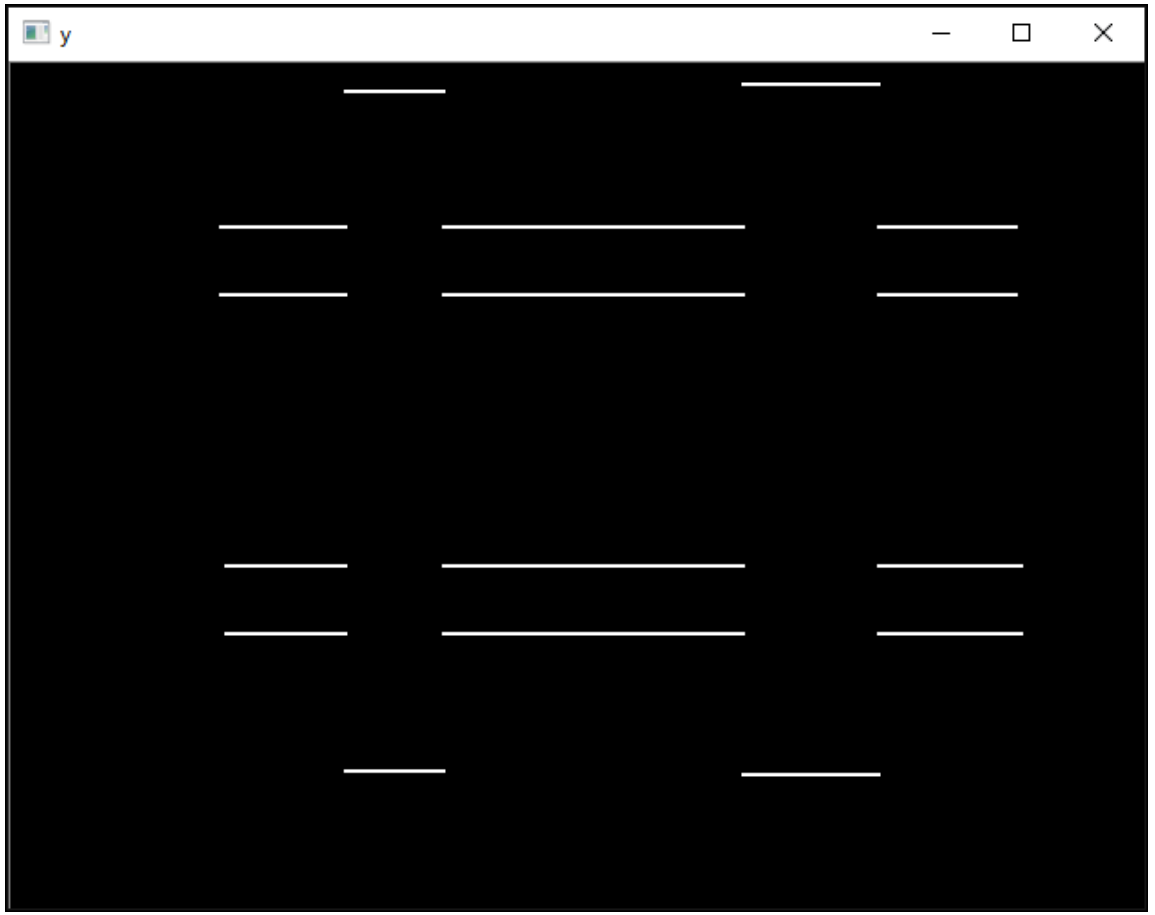


Sobel算子

```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobely = cv2.Sobel(o,cv2.CV_64F,0,1)
sobely = cv2.convertScaleAbs(sobely)
cv2.imshow("original",o)
cv2.imshow("y",sobely)
cv2.waitKey()
cv2.destroyAllWindows()
```

Sobel算子

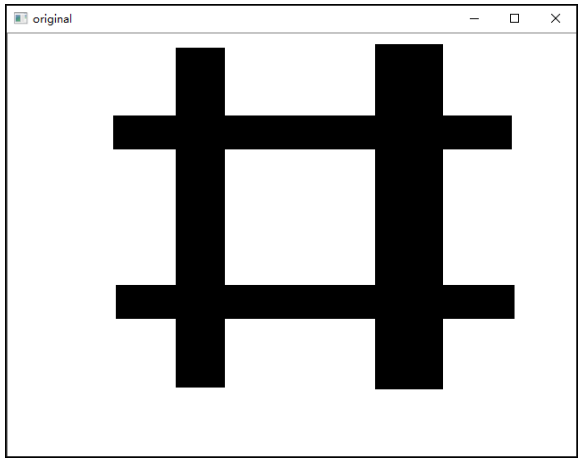


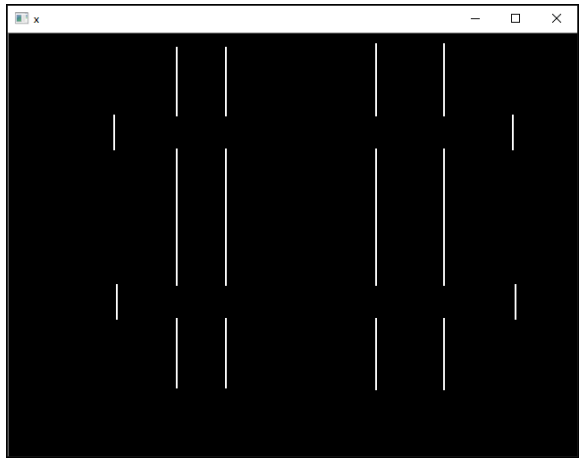


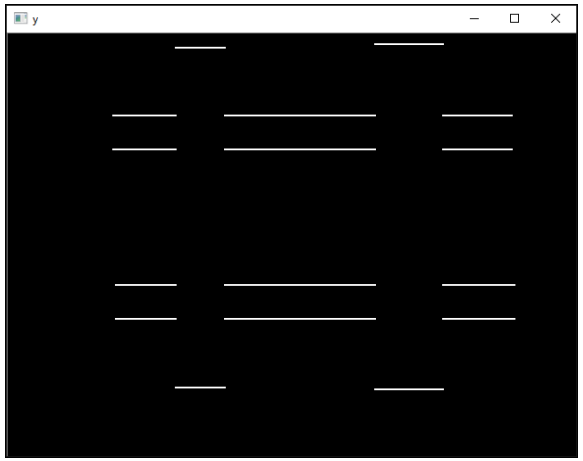
Sobel算子

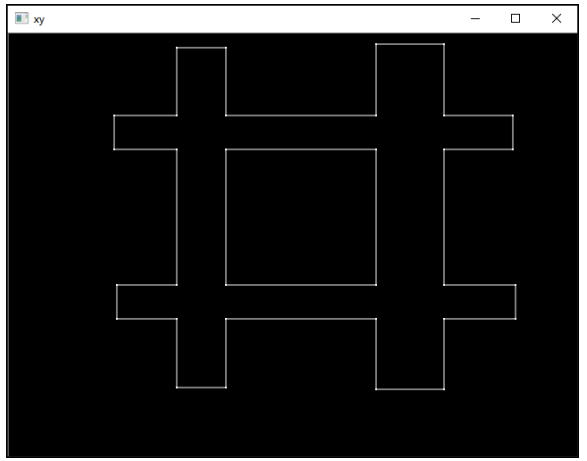
```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(o,cv2.CV_64F,1,0)
sobely = cv2.Sobel(o,cv2.CV_64F,0,1)
sobelx = cv2.convertScaleAbs(sobelx)   # 转回uint8
sobely = cv2.convertScaleAbs(sobely)
sobelxy =  cv2.addWeighted(sobelx,0.5,sobely,0.5,0)
cv2.imshow("original",o)
cv2.imshow("x",sobelx)
cv2.imshow("y",sobely)
cv2.imshow("xy",sobelxy)
cv2.waitKey()
cv2.destroyAllWindows()
```

Sobel算子









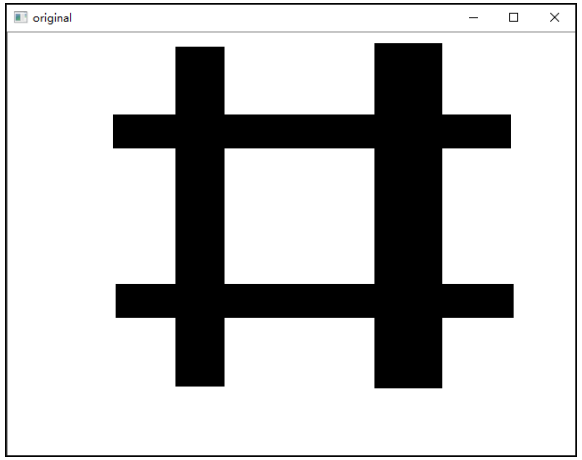
Sobel算子

```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(o,cv2.CV_64F,1,0)
sobely = cv2.Sobel(o,cv2.CV_64F,0,1)
sobelx = cv2.convertScaleAbs(sobelx)   # 转回uint8
sobely = cv2.convertScaleAbs(sobely)
sobelxy =  cv2.addWeighted(sobelx,0.5,sobely,0.5,0)
sobelxy11=cv2.Sobel(o,cv2.CV_64F,1,1)
cv2.imshow("original",o)
cv2.imshow("xy",sobelxy)
cv2.imshow("xy11",sobelxy11)
cv2.waitKey()
cv2.destroyAllWindows()
```

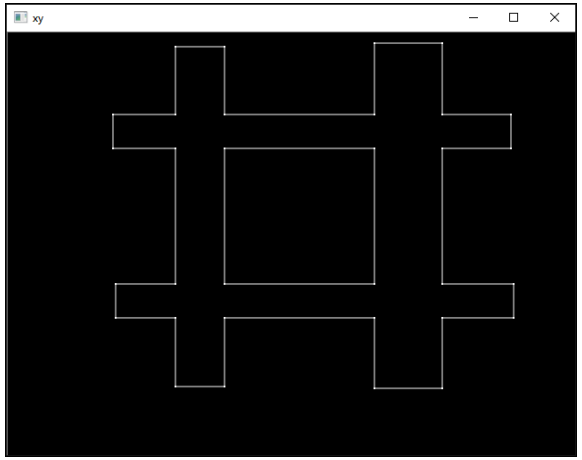
Sobel算子

```
import cv2
import numpy as np
o = cv2.imread('image\\sobel4.bmp',cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(o,cv2.CV_64F,1,0)
sobely = cv2.Sobel(o,cv2.CV_64F,0,1)
sobelx = cv2.convertScaleAbs(sobelx)   # 转回uint8
sobely = cv2.convertScaleAbs(sobely)
sobelxy =  cv2.addWeighted(sobelx,0.5,sobely,0.5,0)
sobelxy11=cv2.Sobel(o,cv2.CV_64F,1,1)
sobelxy11=cv2.convertScaleAbs(sobelxy11)
cv2.imshow("original",o)
cv2.imshow("xy",sobelxy)
cv2.imshow("xy11",sobelxy11)
cv2.waitKey()
cv2.destroyAllWindows()
```

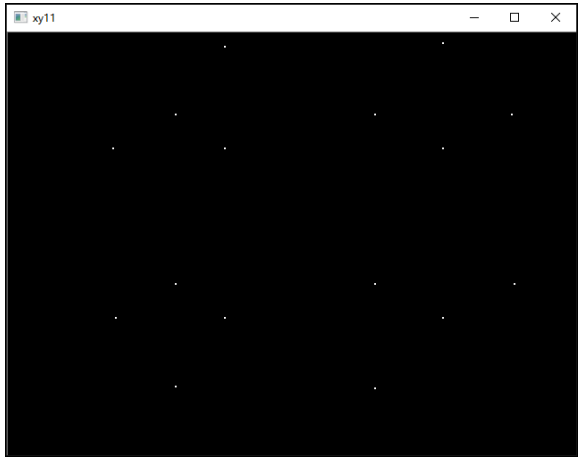
Sobel算子



原始图像



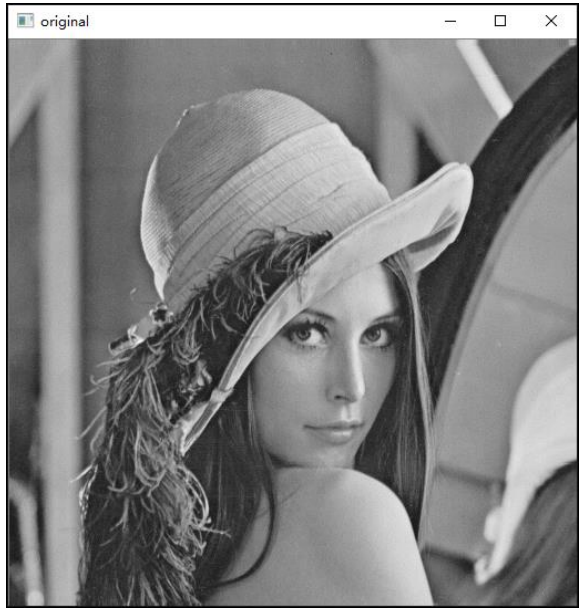
dx+dy



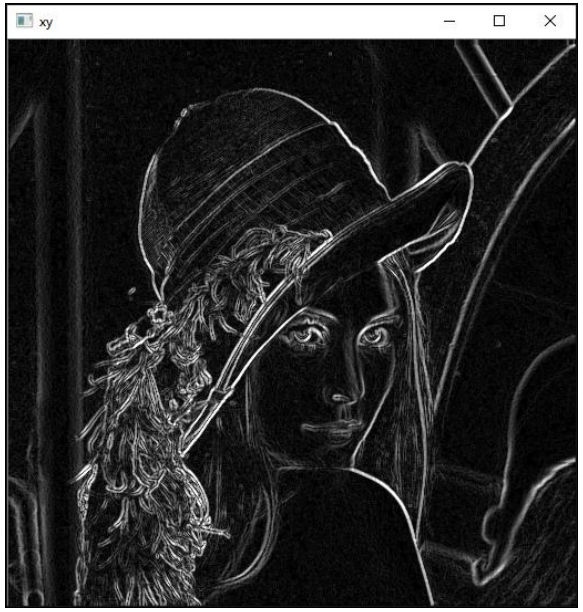
dx=1,dy=1

检测不到边的情况：dx=1，dy=1

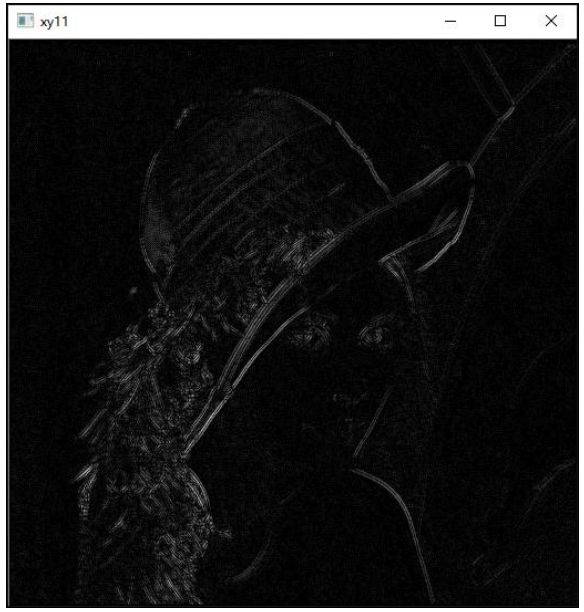
Sobel算子



原始图像



dx+dy



dx=1,dy=1

检测不到边的情况：dx=1，dy=1

Python+OpenCV图像处理

图像梯度

sobel算子函数及其使用

李大羊

lilizong@gmail.com