实验二 边缘检测实验

1 实验简介和目的

边缘是图像亮度急剧变化的部分，检测边缘目的是捕捉重要事件和世界属性的变化。可以看出，在图像形成模型的一般假设下，图像亮度的不连续性可能对应于

1. 深度不连续，
2. 表面取向的不连续性
3. 材料特性的变化和场景照明的变化。

在理想情况下，将边缘检测器应用于图像的结果可能会导致一组连接的曲线，这些曲线指示对象的边界、表面标记的边界以及对应于表面方向不连续性的曲线。因此，将边缘检测算法应用于图像可以显着减少要处理的数据量，因此可以过滤掉可能被认为不太相关的信息，同时保留图像的重要结构属性。如果边缘检测步骤成功，则随后解释原始图像中的信息内容的任务可以因此大大简化.

2 算法模型和原理

在实验过程中采用了一阶和二阶微分算子检验边缘.

其中一阶算子：Roberts算子，sobel算子

Roberts算子：

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{和} \quad \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

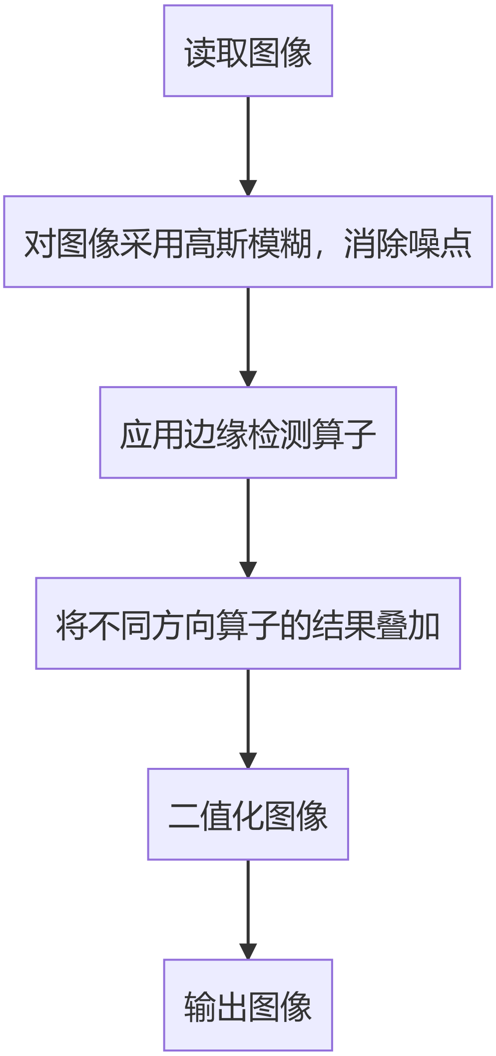
sobel算子：

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \text{和} \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

二阶算子：Laplace算子

$$\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix}$$

3 算法框图



4 算法设计

Roberts算子：

```python
import cv2 as cv
import numpy as np
name = 'scen.jpg'
img0 = cv.imread(name, 0)
img = img0
img = cv.GaussianBlur(img0, (9, 9), 0)
# flower2: 5, 5
# scen: 3, 5
# pipe organ:
# window: 7

#Roberts
kernelx = np.array([[-1,0],[0,1]], dtype=int)
kernely = np.array([[0,-1],[1,0]], dtype=int)
x = cv.filter2D(img, cv.CV_16S, kernelx)
y = cv.filter2D(img, cv.CV_16S, kernely)

absX = cv.convertScaleAbs(x)
absY = cv.convertScaleAbs(y)
Roberts = cv.addWeighted(absY,0.5,absY,0.5,0)

ret, thres = cv.threshold(Roberts,5,255,cv.THRESH_BINARY)
```

```

cv.imwrite(name[:-4] + ' Robert edge.jpg', thres)
cv.imwrite(name[:-4] + ' Robert edge raw.jpg', Roberts)
```

Sobel算子：

```
import cv2 as cv
import numpy as np
name = 'pipe organ.jpg'
img0 = cv.imread(name, 0)
img = img0
img = cv.GaussianBlur(img0, (7, 7), 0)

x = cv.Sobel(img,cv.CV_16S,1,0)
y = cv.Sobel(img,cv.CV_16S,0,1)

absX = cv.convertScaleAbs(x)   # 转回uint8
absY = cv.convertScaleAbs(y)

Sobel = cv.addWeighted(absX,0.5,absY,0.5,0)

ret, thres = cv.threshold(Sobel,20,255,cv.THRESH_BINARY)

cv.imwrite(name[:-4] + ' Sobel edge.jpg', thres)
cv.imwrite(name[:-4] + ' Sobel edge raw.jpg', Sobel)
```

Laplace算子

```
import cv2 as cv
import numpy as np
name = 'flower2.jpg'
img0 = cv.imread(name, 0)
img = img0
img = cv.GaussianBlur(img0, (7, 7), 0)

x = cv.Sobel(img,cv.CV_16S,1,0)
y = cv.Sobel(img,cv.CV_16S,0,1)

absX = cv.convertScaleAbs(x)   # 转回uint8
absY = cv.convertScaleAbs(y)

Sobel = cv.addWeighted(absX,0.5,absY,0.5,0)

ret, thres = cv.threshold(Sobel,20,255,cv.THRESH_BINARY)

cv.imwrite(name[:-4] + ' Laplace edge.jpg', thres)
cv.imwrite(name[:-4] + ' Laplace edge raw.jpg', Sobel)
```

5 实验分析

下面是采取不同算子边缘检测的结果

original

Roberts

Binarized Roberts

Sobel

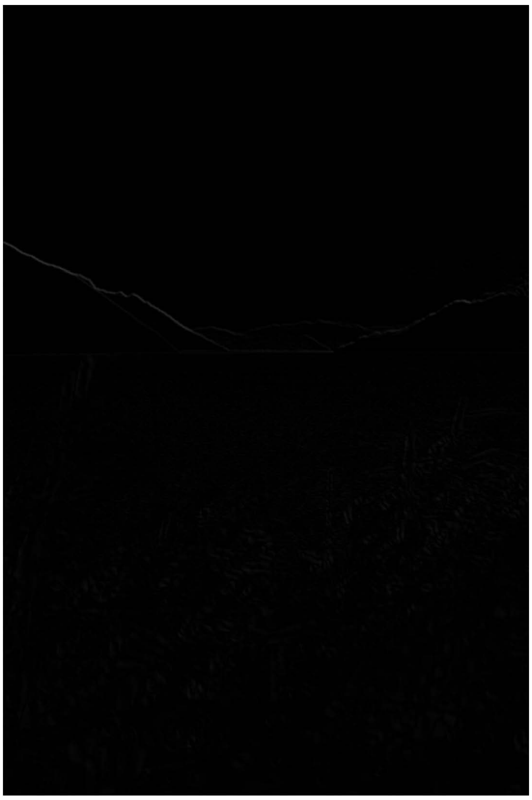
Binarized Sobel

Laplace

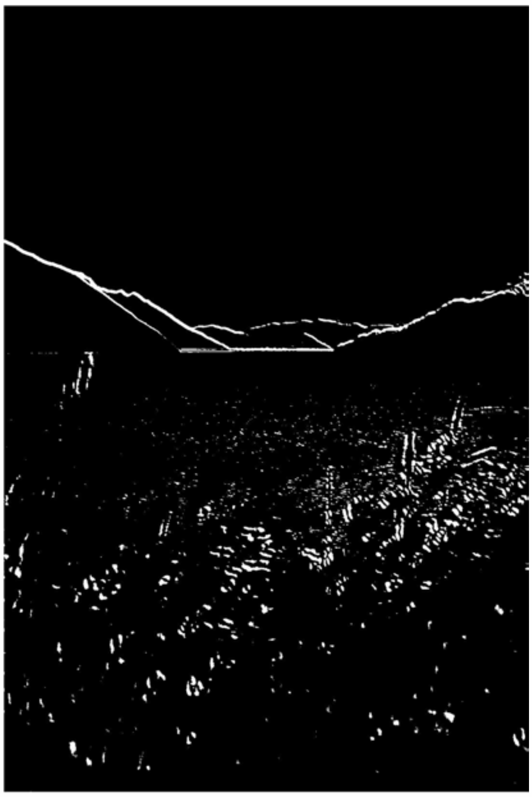
Binarized Laplace

original

Roberts

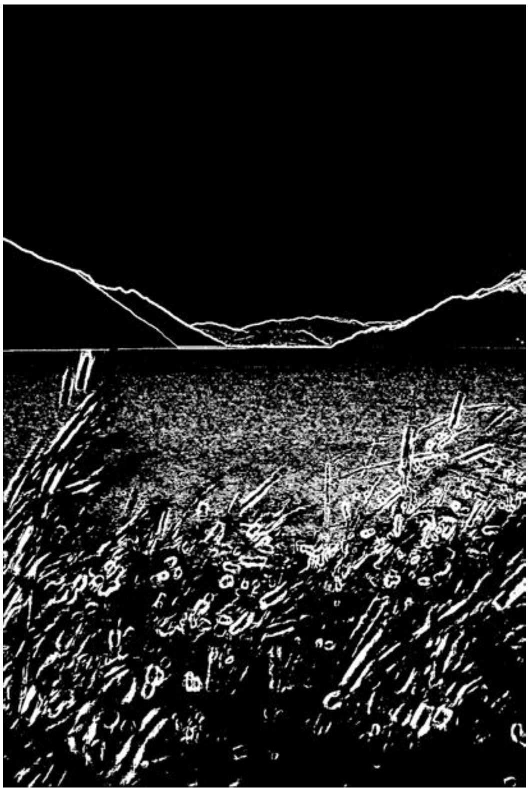


Roberts&binarized

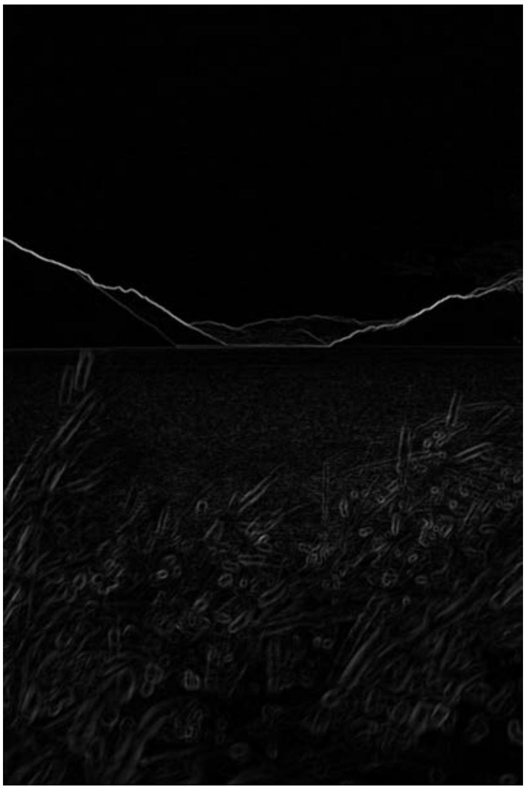


Sobel

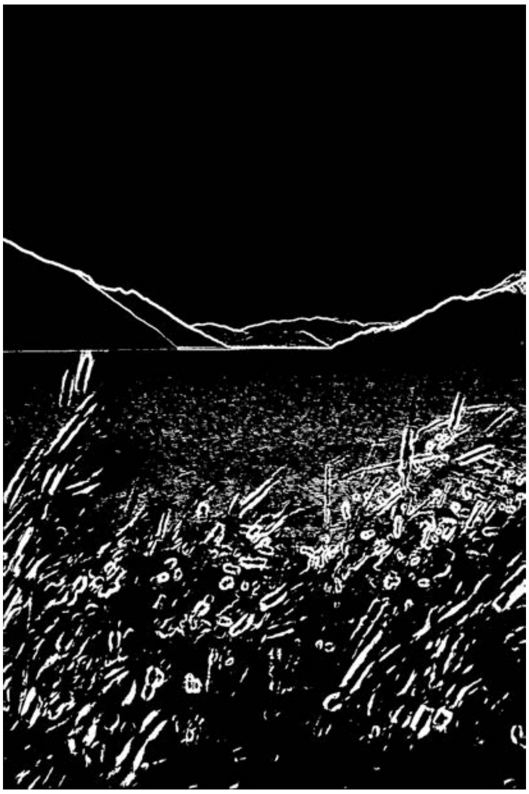
Sobel&binarized



Laplace

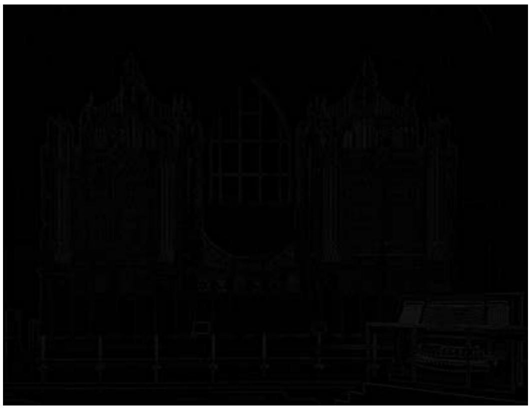


Laplace&binarized

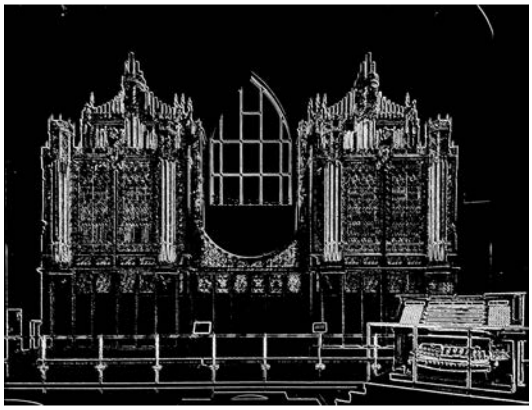


original

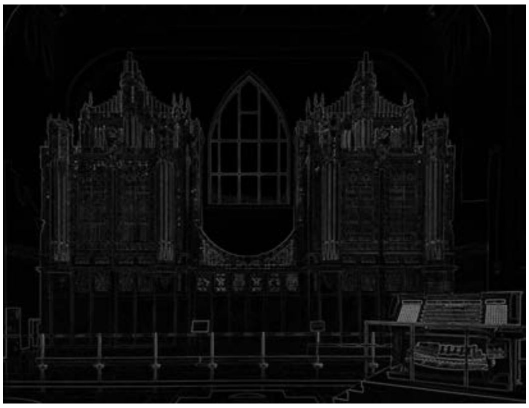
Roberts



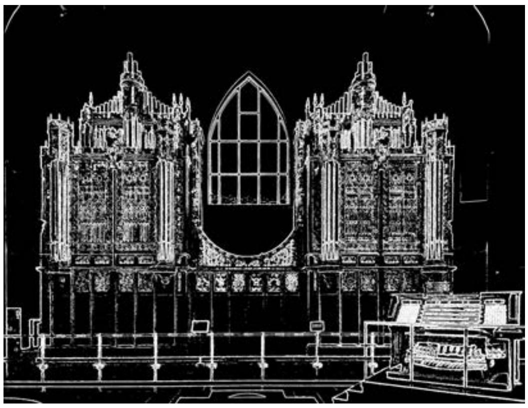
Roberts&binarized



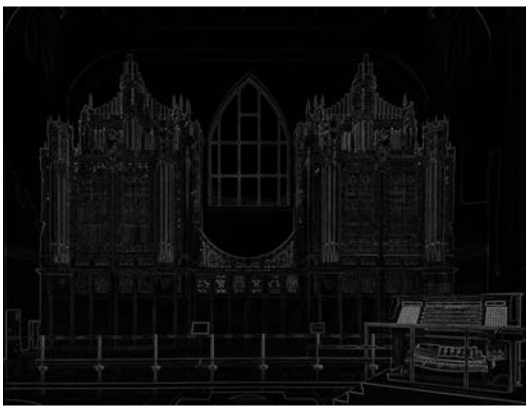
Sobel



Sobel&binarized



Laplace



Laplace&binarized