Sobel算子边缘检测的CUDA实现

SC19023100

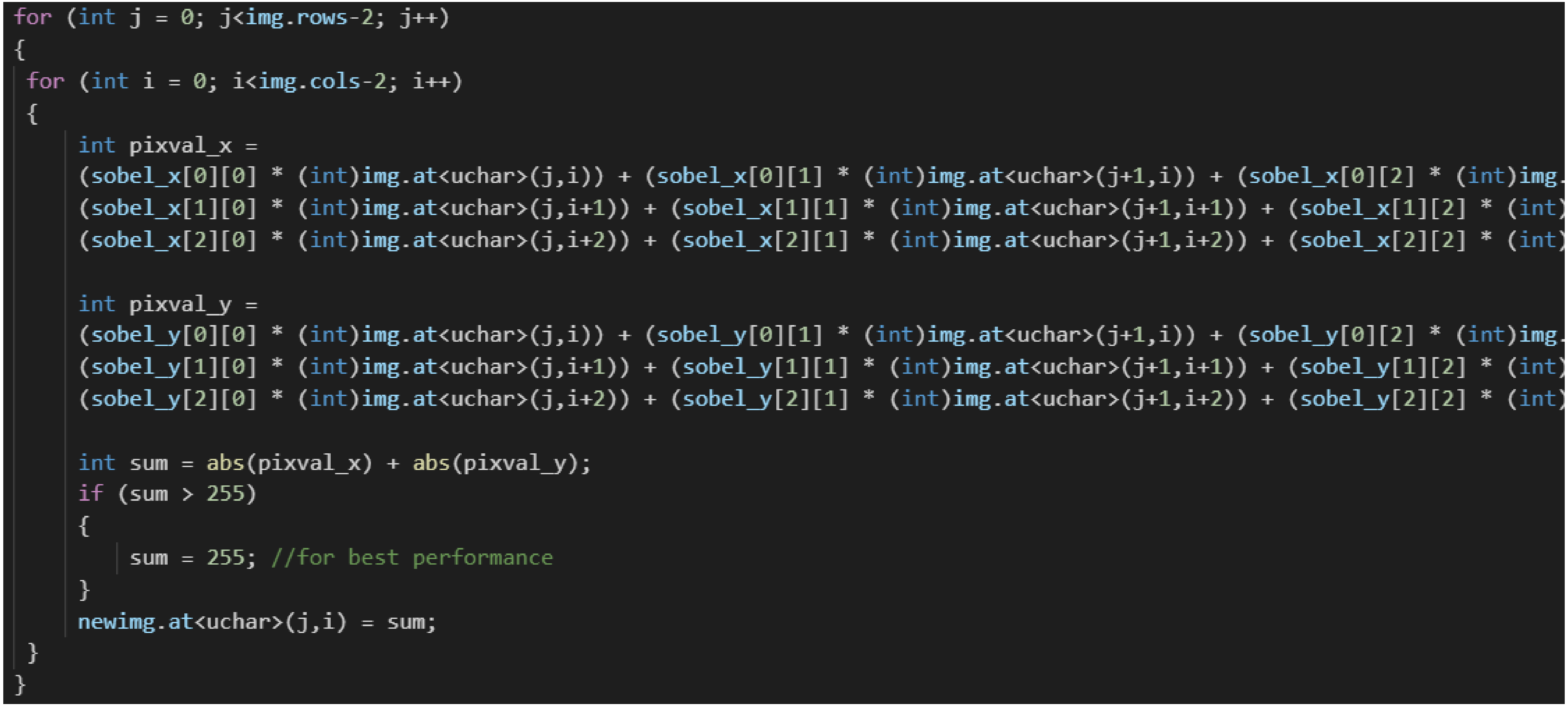
孙霖

1、Sobel原理

- 主要用于获得数字图像的一阶梯度，常见的应用和物理意义是边缘检测

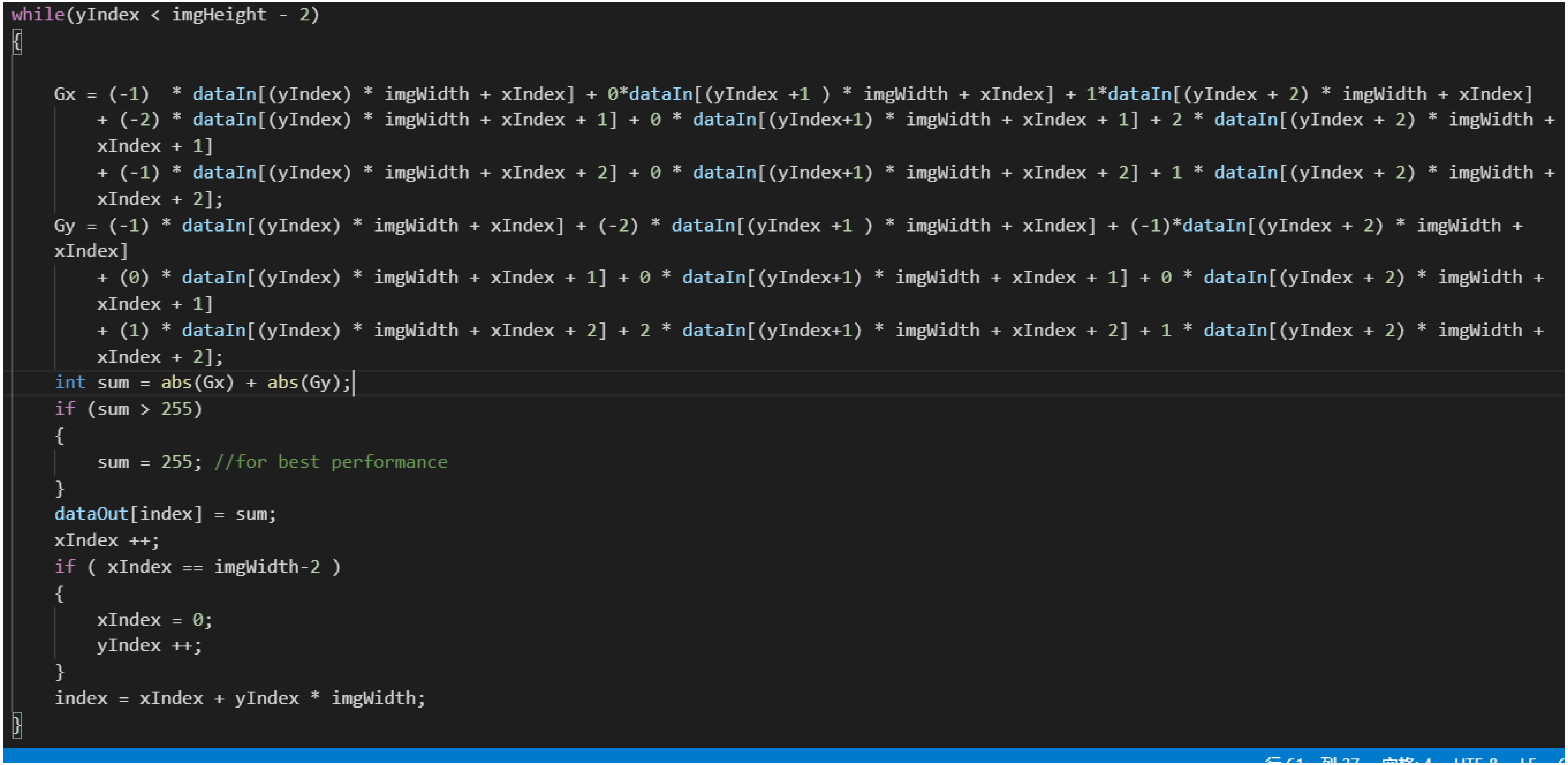
$$\mathrm{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \qquad \mathrm{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & 1 \end{bmatrix} * A$$

2、Sobel的CPU实现

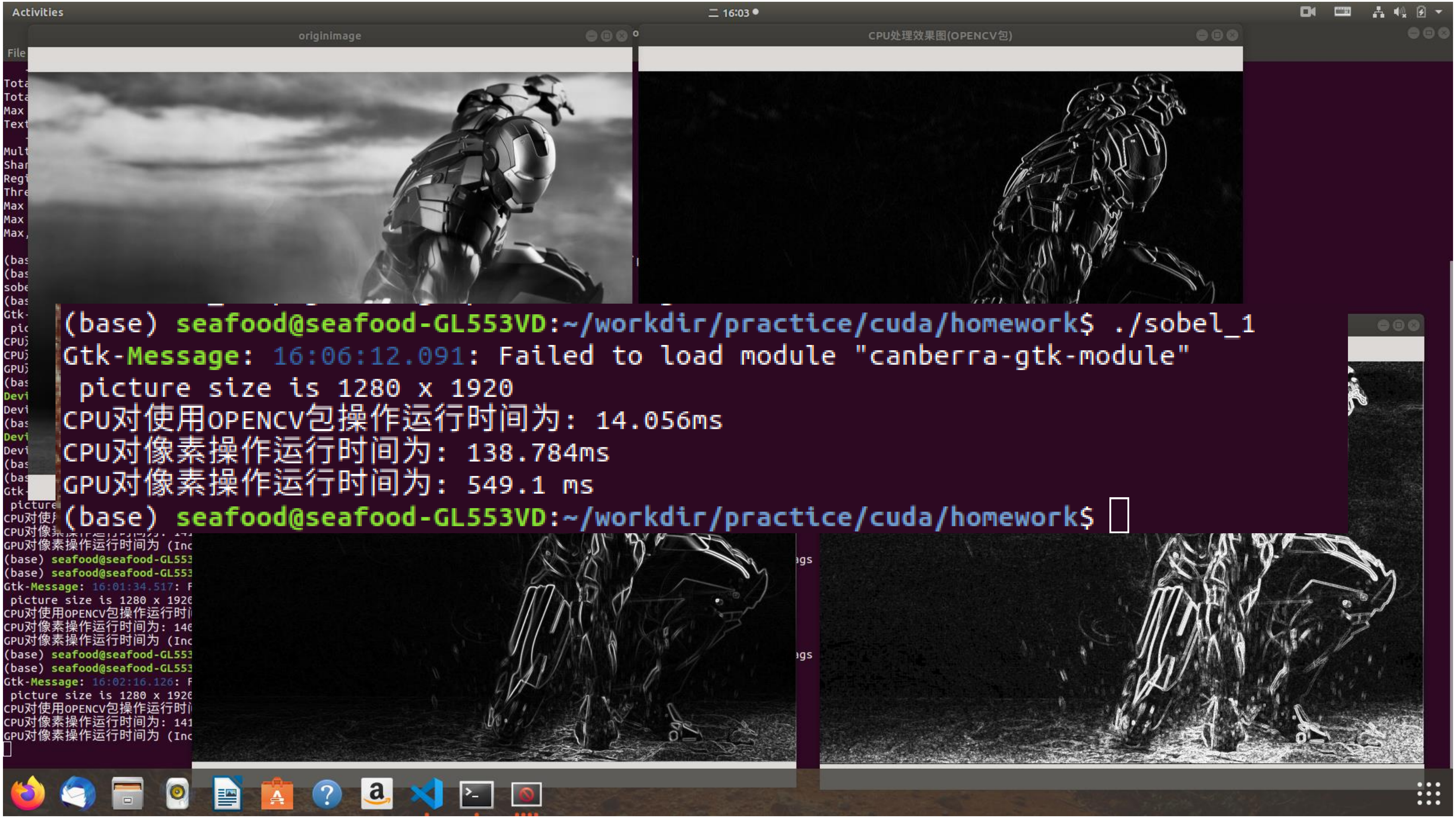


```
for (int j = 0; j<img.rows-2; j++)
{
 for (int i = 0; i<img.cols-2; i++)
 {
     int pixval_x =
     (sobel_x[0][0] * (int)img.at<uchar>(j,i)) + (sobel_x[0][1] * (int)img.at<uchar>(j+1,i)) + (sobel_x[0][2] * (int)img.
     (sobel_x[1][0] * (int)img.at<uchar>(j,i+1)) + (sobel_x[1][1] * (int)img.at<uchar>(j+1,i+1)) + (sobel_x[1][2] * (int)
     (sobel_x[2][0] * (int)img.at<uchar>(j,i+2)) + (sobel_x[2][1] * (int)img.at<uchar>(j+1,i+2)) + (sobel_x[2][2] * (int)

     int pixval_y =
     (sobel_y[0][0] * (int)img.at<uchar>(j,i)) + (sobel_y[0][1] * (int)img.at<uchar>(j+1,i)) + (sobel_y[0][2] * (int)img.
     (sobel_y[1][0] * (int)img.at<uchar>(j,i+1)) + (sobel_y[1][1] * (int)img.at<uchar>(j+1,i+1)) + (sobel_y[1][2] * (int)
     (sobel_y[2][0] * (int)img.at<uchar>(j,i+2)) + (sobel_y[2][1] * (int)img.at<uchar>(j+1,i+2)) + (sobel_y[2][2] * (int)

     int sum = abs(pixval_x) + abs(pixval_y);
     if (sum > 255)
     {
         sum = 255; //for best performance
     }
     newimg.at<uchar>(j,i) = sum;
 }
}
```

3、Sobel的GPU实现



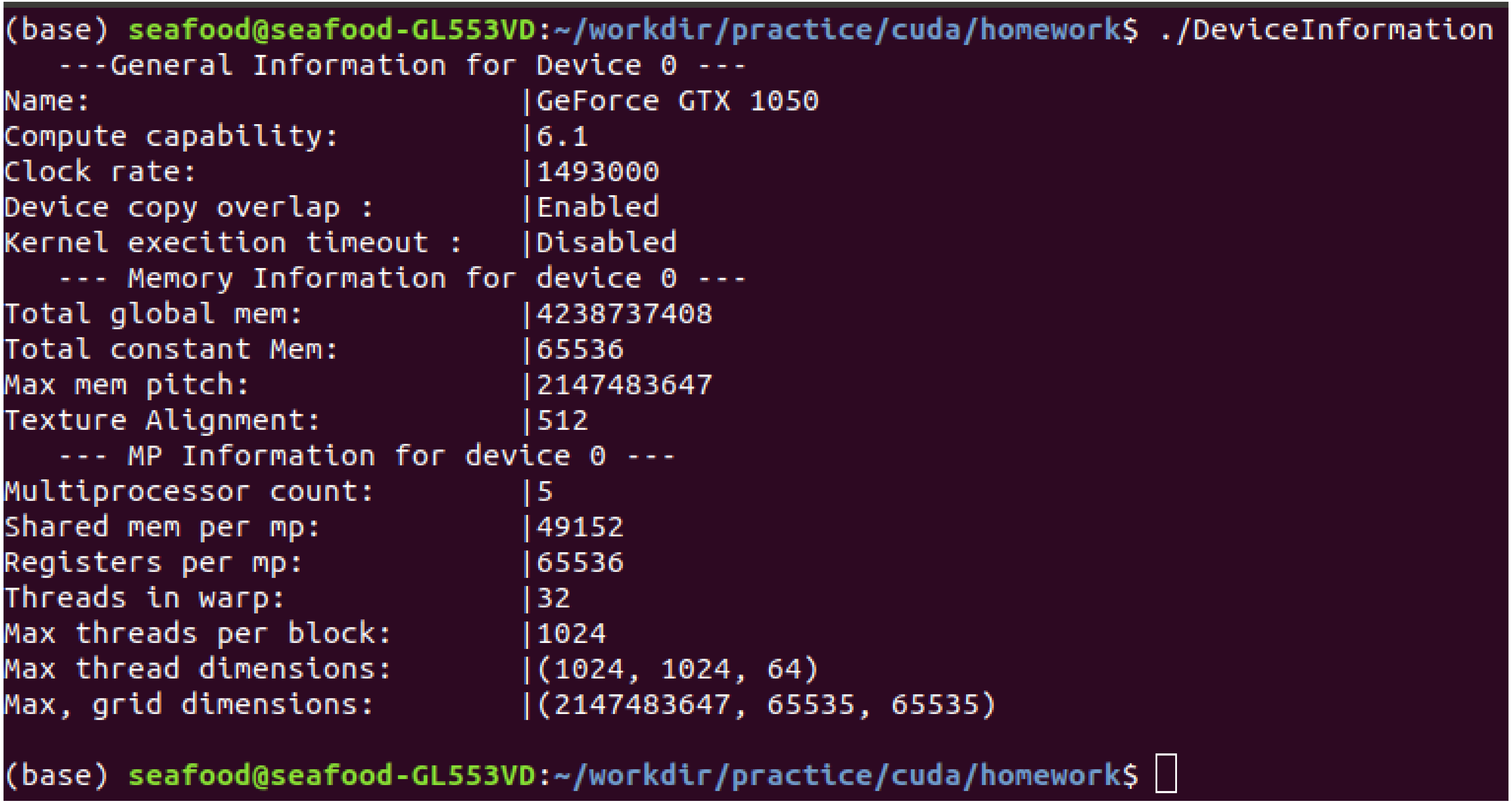
```
while(yIndex < imgHeight - 2)
{


    Gx = (-1)  * dataIn[(yIndex) * imgWidth + xIndex] + 0*dataIn[(yIndex +1 ) * imgWidth + xIndex] + 1*dataIn[(yIndex + 2) * imgWidth + xIndex]
        + (-2) * dataIn[(yIndex) * imgWidth + xIndex + 1] + 0 * dataIn[(yIndex+1) * imgWidth + xIndex + 1] + 2 * dataIn[(yIndex + 2) * imgWidth +
        xIndex + 1]
        + (-1) * dataIn[(yIndex) * imgWidth + xIndex + 2] + 0 * dataIn[(yIndex+1) * imgWidth + xIndex + 2] + 1 * dataIn[(yIndex + 2) * imgWidth +
        xIndex + 2];
    Gy = (-1) * dataIn[(yIndex) * imgWidth + xIndex] + (-2) * dataIn[(yIndex +1 ) * imgWidth + xIndex] + (-1)*dataIn[(yIndex + 2) * imgWidth +
    xIndex]
        + (0) * dataIn[(yIndex) * imgWidth + xIndex + 1] + 0 * dataIn[(yIndex+1) * imgWidth + xIndex + 1] + 0 * dataIn[(yIndex + 2) * imgWidth +
        xIndex + 1]
        + (1) * dataIn[(yIndex) * imgWidth + xIndex + 2] + 2 * dataIn[(yIndex+1) * imgWidth + xIndex + 2] + 1 * dataIn[(yIndex + 2) * imgWidth +
        xIndex + 2];
    int sum = abs(Gx) + abs(Gy);
    if (sum > 255)
    {
        sum = 255; //for best performance
    }
    dataOut[index] = sum;
    xIndex ++;
    if ( xIndex == imgWidth-2 )
    {
        xIndex = 0;
        yIndex ++;
    }
    index = xIndex + yIndex * imgWidth;
}
```



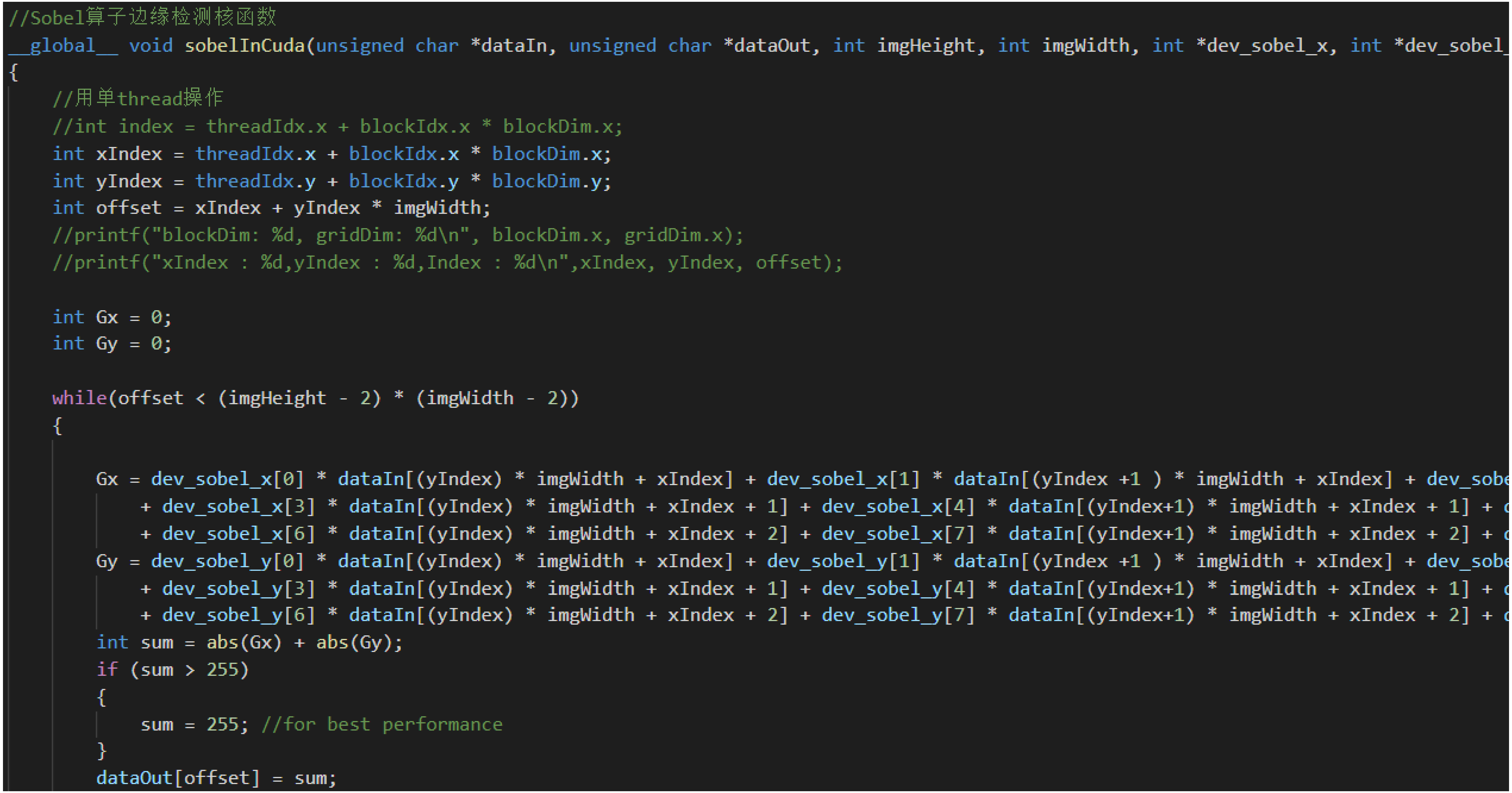
```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_1
Gtk-Message: 16:06:12.091: Failed to load module "canberra-gtk-module"
 picture size is 1280 x 1920
CPU对使用OPENCV包操作运行时间为: 14.056ms
CPU对像素操作运行时间为: 138.784ms
GPU对像素操作运行时间为: 549.1 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```

```
GPU对像素操作运行时间为 (Inc
(base) seafood@seafood-GL553
(base) seafood@seafood-GL553
Gtk-Message: 16:01:34.517: F
 picture size is 1280 x 1920
CPU对使用OPENCV包操作运行时间
CPU对像素操作运行时间为: 140
GPU对像素操作运行时间为 (Inc
(base) seafood@seafood-GL553
(base) seafood@seafood-GL553
Gtk-Message: 16:02:16.126: F
 picture size is 1280 x 1920
CPU对使用OPENCV包操作运行时间
CPU对像素操作运行时间为: 141
GPU对像素操作运行时间为 (Inc
```

5、程序优化-GPU配置

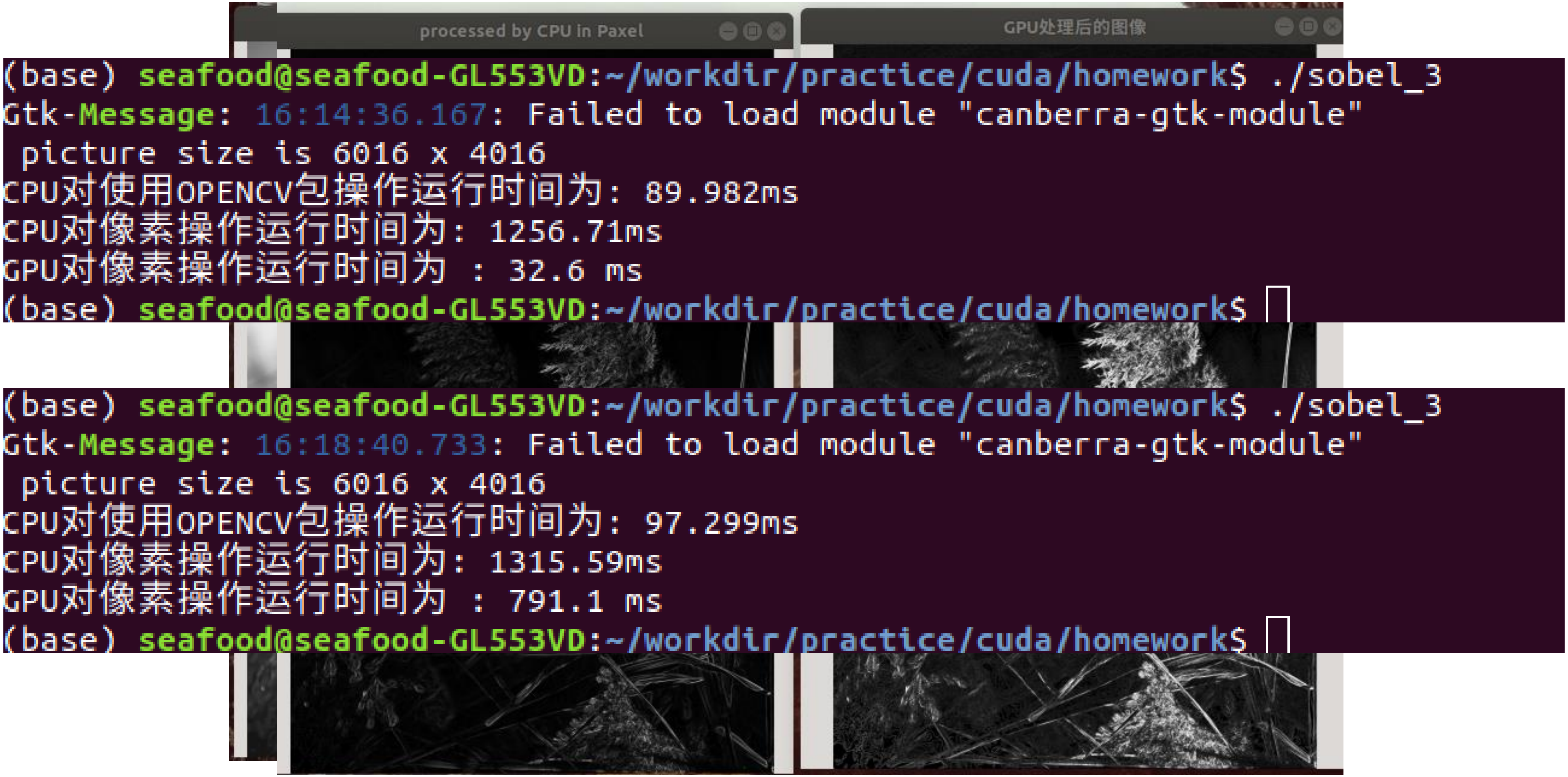


```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./DeviceInformation
   ---General Information for Device 0 ---
Name:                        |GeForce GTX 1050
Compute capability:          |6.1
Clock rate:                  |1493000
Device copy overlap :        |Enabled
Kernel excition timeout :    |Disabled
   --- Memory Information for device 0 ---
Total global mem:            |4238737408
Total constant Mem:          |65536
Max mem pitch:               |2147483647
Texture Alignment:           |512
   --- MP Information for device 0 ---
Multiprocessor count:        |5
Shared mem per mp:           |49152
Registers per mp:            |65536
Threads in warp:             |32
Max threads per block:       |1024
Max thread dimensions:       |(1024, 1024, 64)
Max, grid dimensions:        |(2147483647, 65535, 65535)

(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```



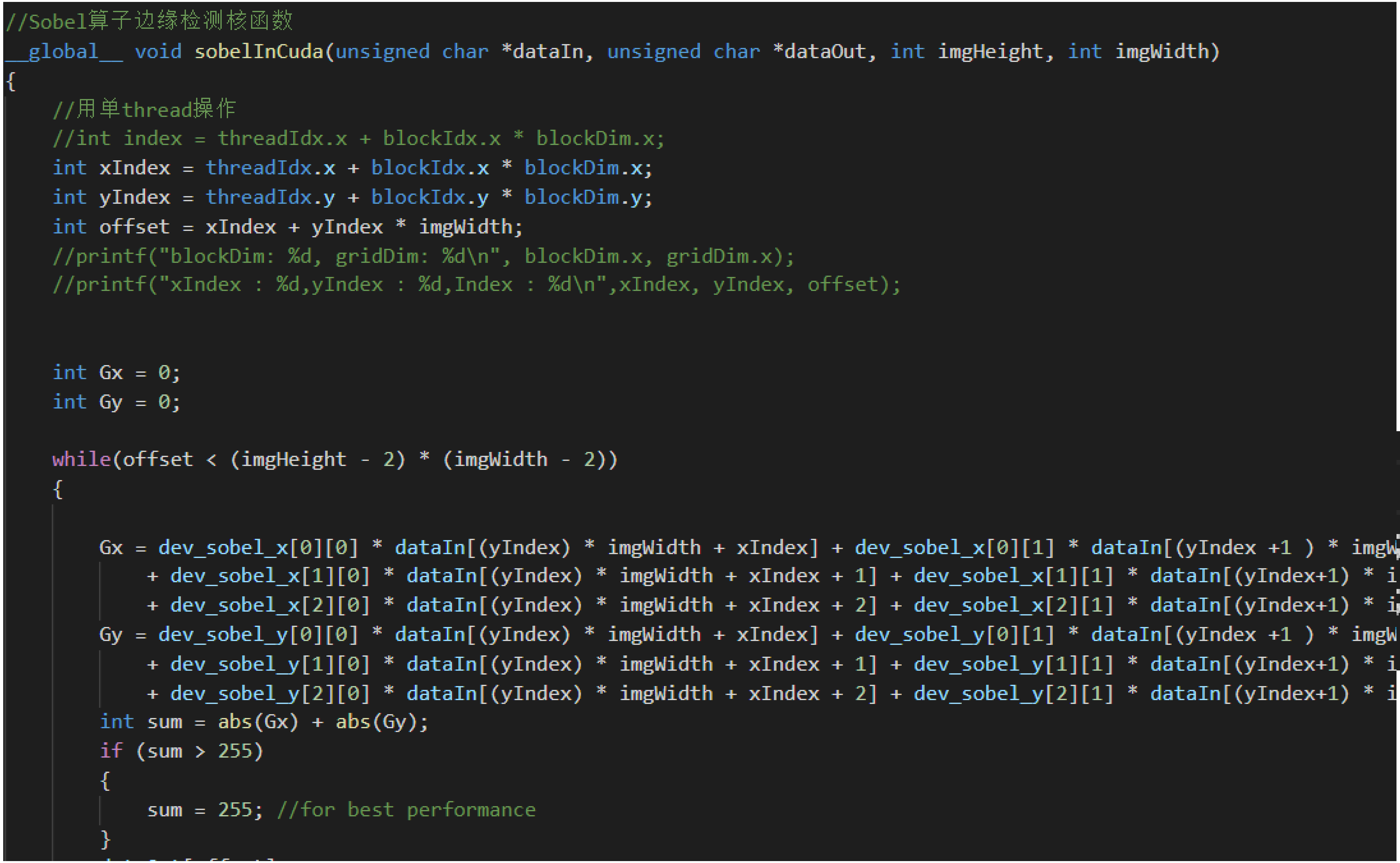
```
//Sobel算子边缘检测核函数
__global__ void sobelInCuda(unsigned char *dataIn, unsigned char *dataOut, int imgHeight, int imgWidth, int *dev_sobel_x, int *dev_sobel_
{
    //用单thread操作
    //int index = threadIdx.x + blockIdx.x * blockDim.x;
    int xIndex = threadIdx.x + blockIdx.x * blockDim.x;
    int yIndex = threadIdx.y + blockIdx.y * blockDim.y;
    int offset = xIndex + yIndex * imgWidth;
    //printf("blockDim: %d, gridDim: %d\n", blockDim.x, gridDim.x);
    //printf("xIndex : %d,yIndex : %d,Index : %d\n",xIndex, yIndex, offset);

    int Gx = 0;
    int Gy = 0;

    while(offset < (imgHeight - 2) * (imgWidth - 2))
    {

        Gx = dev_sobel_x[0] * dataIn[(yIndex) * imgWidth + xIndex] + dev_sobel_x[1] * dataIn[(yIndex +1 ) * imgWidth + xIndex] + dev_sobe
            + dev_sobel_x[3] * dataIn[(yIndex) * imgWidth + xIndex + 1] + dev_sobel_x[4] * dataIn[(yIndex+1) * imgWidth + xIndex + 1] + d
            + dev_sobel_x[6] * dataIn[(yIndex) * imgWidth + xIndex + 2] + dev_sobel_x[7] * dataIn[(yIndex+1) * imgWidth + xIndex + 2] + d
        Gy = dev_sobel_y[0] * dataIn[(yIndex) * imgWidth + xIndex] + dev_sobel_y[1] * dataIn[(yIndex +1 ) * imgWidth + xIndex] + dev_sobe
            + dev_sobel_y[3] * dataIn[(yIndex) * imgWidth + xIndex + 1] + dev_sobel_y[4] * dataIn[(yIndex+1) * imgWidth + xIndex + 1] + d
            + dev_sobel_y[6] * dataIn[(yIndex) * imgWidth + xIndex + 2] + dev_sobel_y[7] * dataIn[(yIndex+1) * imgWidth + xIndex + 2] + d
        int sum = abs(Gx) + abs(Gy);
        if (sum > 255)
        {
            sum = 255; //for best performance
        }
        dataOut[offset] = sum;
```

5.1 程序优化-二维Block二维Thread



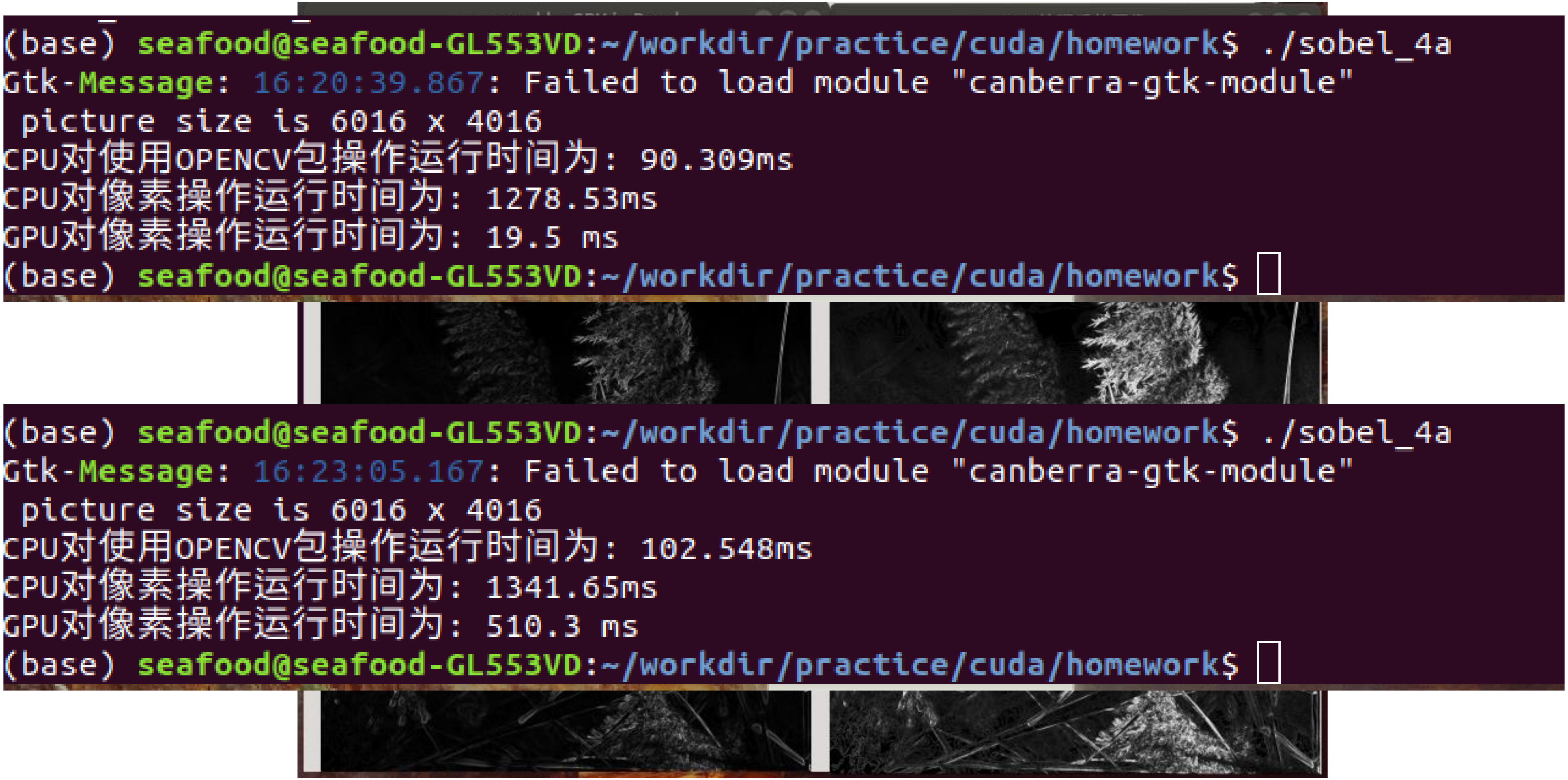
```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_3
Gtk-Message: 16:14:36.167: Failed to load module "canberra-gtk-module"
 picture size is 6016 x 4016
CPU对使用OPENCV包操作运行时间为: 89.982ms
CPU对像素操作运行时间为: 1256.71ms
GPU对像素操作运行时间为 : 32.6 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```

```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_3
Gtk-Message: 16:18:40.733: Failed to load module "canberra-gtk-module"
 picture size is 6016 x 4016
CPU对使用OPENCV包操作运行时间为: 97.299ms
CPU对像素操作运行时间为: 1315.59ms
GPU对像素操作运行时间为 : 791.1 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```



```
//Sobel算子边缘检测核函数
__global__ void sobelInCuda(unsigned char *dataIn, unsigned char *dataOut, int imgHeight, int imgWidth)
{
    //用单thread操作
    //int index = threadIdx.x + blockIdx.x * blockDim.x;
    int xIndex = threadIdx.x + blockIdx.x * blockDim.x;
    int yIndex = threadIdx.y + blockIdx.y * blockDim.y;
    int offset = xIndex + yIndex * imgWidth;
    //printf("blockDim: %d, gridDim: %d\n", blockDim.x, gridDim.x);
    //printf("xIndex : %d,yIndex : %d,Index : %d\n",xIndex, yIndex, offset);


    int Gx = 0;
    int Gy = 0;

    while(offset < (imgHeight - 2) * (imgWidth - 2))
    {

        Gx = dev_sobel_x[0][0] * dataIn[(yIndex) * imgWidth + xIndex] + dev_sobel_x[0][1] * dataIn[(yIndex +1 ) * imgW
            + dev_sobel_x[1][0] * dataIn[(yIndex) * imgWidth + xIndex + 1] + dev_sobel_x[1][1] * dataIn[(yIndex+1) * i
            + dev_sobel_x[2][0] * dataIn[(yIndex) * imgWidth + xIndex + 2] + dev_sobel_x[2][1] * dataIn[(yIndex+1) * i
        Gy = dev_sobel_y[0][0] * dataIn[(yIndex) * imgWidth + xIndex] + dev_sobel_y[0][1] * dataIn[(yIndex +1 ) * imgW
            + dev_sobel_y[1][0] * dataIn[(yIndex) * imgWidth + xIndex + 1] + dev_sobel_y[1][1] * dataIn[(yIndex+1) * i
            + dev_sobel_y[2][0] * dataIn[(yIndex) * imgWidth + xIndex + 2] + dev_sobel_y[2][1] * dataIn[(yIndex+1) * i
        int sum = abs(Gx) + abs(Gy);
        if (sum > 255)
        {
            sum = 255; //for best performance
        }
```

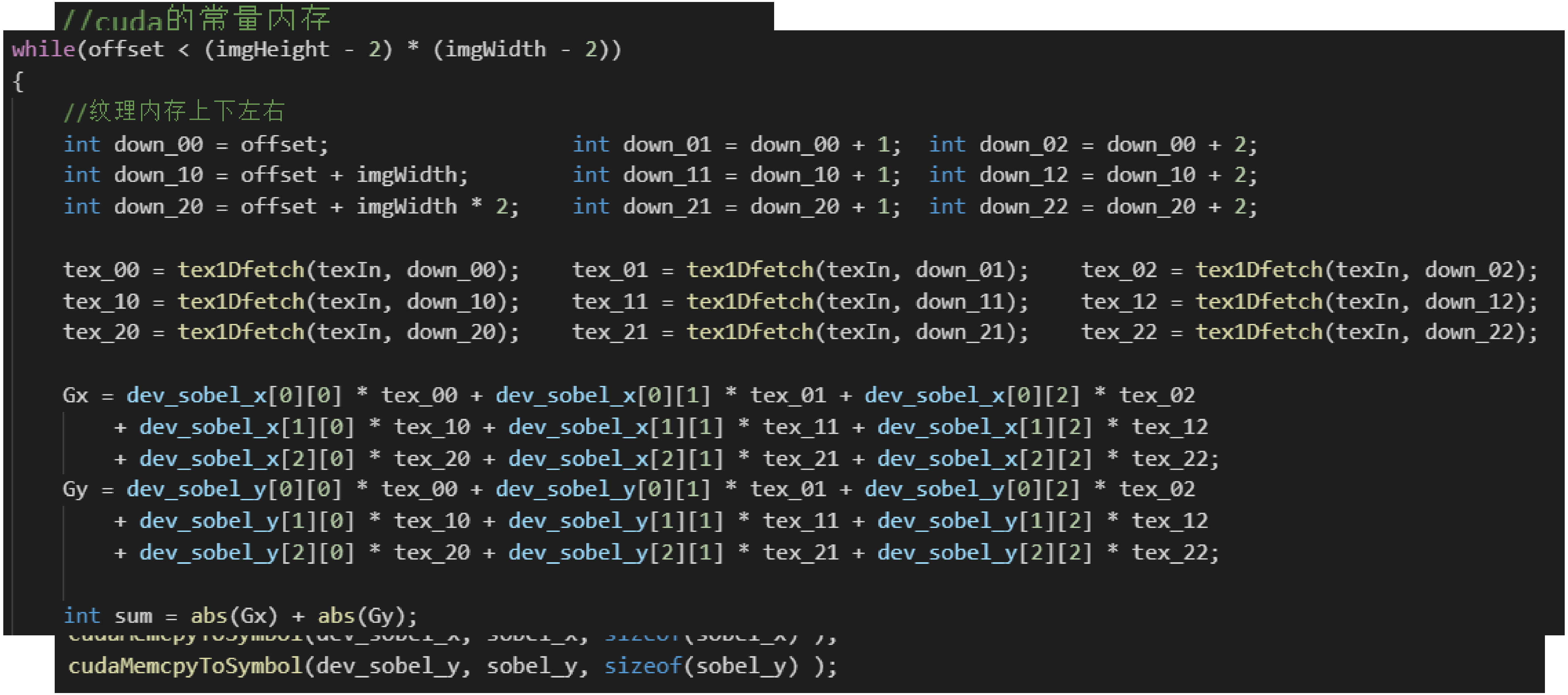
5.2 程序优化-常量内存



```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_4a
Gtk-Message: 16:20:39.867: Failed to load module "canberra-gtk-module"
 picture size is 6016 x 4016
CPU对使用OPENCV包操作运行时间为: 90.309ms
CPU对像素操作运行时间为: 1278.53ms
GPU对像素操作运行时间为: 19.5 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```

```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_4a
Gtk-Message: 16:23:05.167: Failed to load module "canberra-gtk-module"
 picture size is 6016 x 4016
CPU对使用OPENCV包操作运行时间为: 102.548ms
CPU对像素操作运行时间为: 1341.65ms
GPU对像素操作运行时间为: 510.3 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```

5.3 程序优化-纹理内存

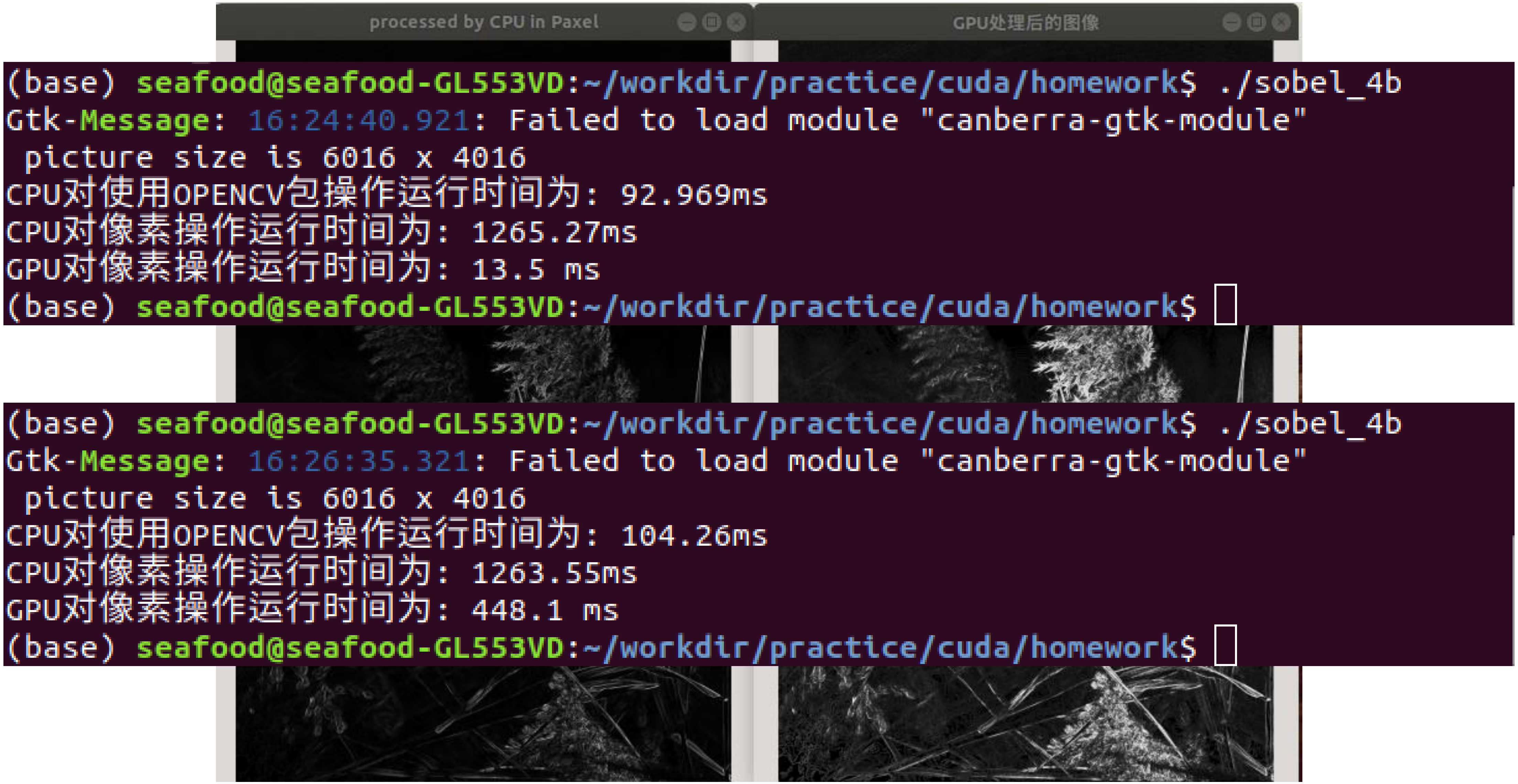


```
//cuda的常量内存
```

```
while(offset < (imgHeight - 2) * (imgWidth - 2))
{
    //纹理内存上下左右
    int down_00 = offset;                   int down_01 = down_00 + 1;  int down_02 = down_00 + 2;
    int down_10 = offset + imgWidth;        int down_11 = down_10 + 1;  int down_12 = down_10 + 2;
    int down_20 = offset + imgWidth * 2;    int down_21 = down_20 + 1;  int down_22 = down_20 + 2;

    tex_00 = tex1Dfetch(texIn, down_00);    tex_01 = tex1Dfetch(texIn, down_01);    tex_02 = tex1Dfetch(texIn, down_02);
    tex_10 = tex1Dfetch(texIn, down_10);    tex_11 = tex1Dfetch(texIn, down_11);    tex_12 = tex1Dfetch(texIn, down_12);
    tex_20 = tex1Dfetch(texIn, down_20);    tex_21 = tex1Dfetch(texIn, down_21);    tex_22 = tex1Dfetch(texIn, down_22);

    Gx = dev_sobel_x[0][0] * tex_00 + dev_sobel_x[0][1] * tex_01 + dev_sobel_x[0][2] * tex_02
        + dev_sobel_x[1][0] * tex_10 + dev_sobel_x[1][1] * tex_11 + dev_sobel_x[1][2] * tex_12
        + dev_sobel_x[2][0] * tex_20 + dev_sobel_x[2][1] * tex_21 + dev_sobel_x[2][2] * tex_22;
    Gy = dev_sobel_y[0][0] * tex_00 + dev_sobel_y[0][1] * tex_01 + dev_sobel_y[0][2] * tex_02
        + dev_sobel_y[1][0] * tex_10 + dev_sobel_y[1][1] * tex_11 + dev_sobel_y[1][2] * tex_12
        + dev_sobel_y[2][0] * tex_20 + dev_sobel_y[2][1] * tex_21 + dev_sobel_y[2][2] * tex_22;

    int sum = abs(Gx) + abs(Gy);
```

```
cudaMemcpyToSymbol(dev_sobel_x, sobel_x, sizeof(sobel_x) );
cudaMemcpyToSymbol(dev_sobel_y, sobel_y, sizeof(sobel_y) );
```

5.3 程序优化-纹理内存



```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_4b
Gtk-Message: 16:24:40.921: Failed to load module "canberra-gtk-module"
 picture size is 6016 x 4016
CPU对使用OPENCV包操作运行时间为: 92.969ms
CPU对像素操作运行时间为: 1265.27ms
GPU对像素操作运行时间为: 13.5 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```

```
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ ./sobel_4b
Gtk-Message: 16:26:35.321: Failed to load module "canberra-gtk-module"
 picture size is 6016 x 4016
CPU对使用OPENCV包操作运行时间为: 104.26ms
CPU对像素操作运行时间为: 1263.55ms
GPU对像素操作运行时间为: 448.1 ms
(base) seafood@seafood-GL553VD:~/workdir/practice/cuda/homework$ 
```

6. 优化结果(6016x4016)

优化	Opencv包运行	CPU对像素操作	GPU对像素操作（自定）	GPU对像素操作（4x4）
单Block单Thread	100ms	1260ms		
多Block多Thread	100ms	1260ms	32ms	791ms
常量内存	100ms	1260ms	20ms	510ms
纹理内存	100ms	1260ms	13ms	448ms

6. 优化结果

优化过程	相对于CPU对像素进行操作的优化程度	优化过程	相对于上一级优化程度（4x4）
多Block多Thread	**40倍**	多Block多Thread	**-**
常量内存	**60倍**	常量内存	**1.6倍**
纹理内存	**96倍**	纹理内存	**1.13倍**