姓名：洪子敬

学号：2022155033

课堂程序练习书面作业

课堂程序题：基于 Sobel 算子的边缘检测和锐化程序

问题求解思路及程序实现过程：

思路：Sobel 算组是一阶导数算子，通过计算图像在**某个方向**的光亮变化来确定边缘，一般使用 3*3 的卷积核，常见的 Sobel 算子如下所示：

$$Sobel_x = [\,[-1,0,1],[-2,0,2],[-1,0,1]\,]$$

$$Sobel_y = [\,[-1,-2,-1],[0,0,0],[1,2,1]\,]$$

$$Sobel_u = [\,[-2,-1,0],[-1,0,1],[0,1,2]\,]$$

$$Sobel_v = [\,[0,1,2],[-1,0,1],[-2,-1,0]\,]$$

其中$Sobel_x$主要针对水平方向的边缘检测，$Sobel_y$主要针对竖直方向的边缘检测，$Sobel_u$主要针对次对角方向的边缘检测，$Sobel_v$主要针对主对角方向的边缘检测。

为了简化过程，下面以单个 Sobel 算子用于边缘检测为例介绍大体流程：

（1）将图片以灰度图片读入；

（2）用 3*3 的 Sobel 算子对该灰度图片进行卷积操作，编写成函数如下：

（边缘处这里为了方便直接赋值为 0，当然也可以取最近的像素值进行赋值）

```
def apply_one_sobel(image, sobel):
    # 获取图像的维度
    height, width = image.shape
    # 创建一个空的图像用于存储结果
    new_image = np.zeros( shape: (height, width), dtype=np.float32)
    # 进行卷积操作
    for i in range(1, height - 1):
        for j in range(1, width - 1):
            g = 0
            for k in range(3):
                for l in range(3):
                    g += image[i + k - 1][j + l - 1] * sobel[k][l]
            new_image[i][j] = g
    return new_image                                    HZJ
```

注意：这里进行运算时都要是浮点数，即 sobel 算子和 image 都要是 float32 类型。

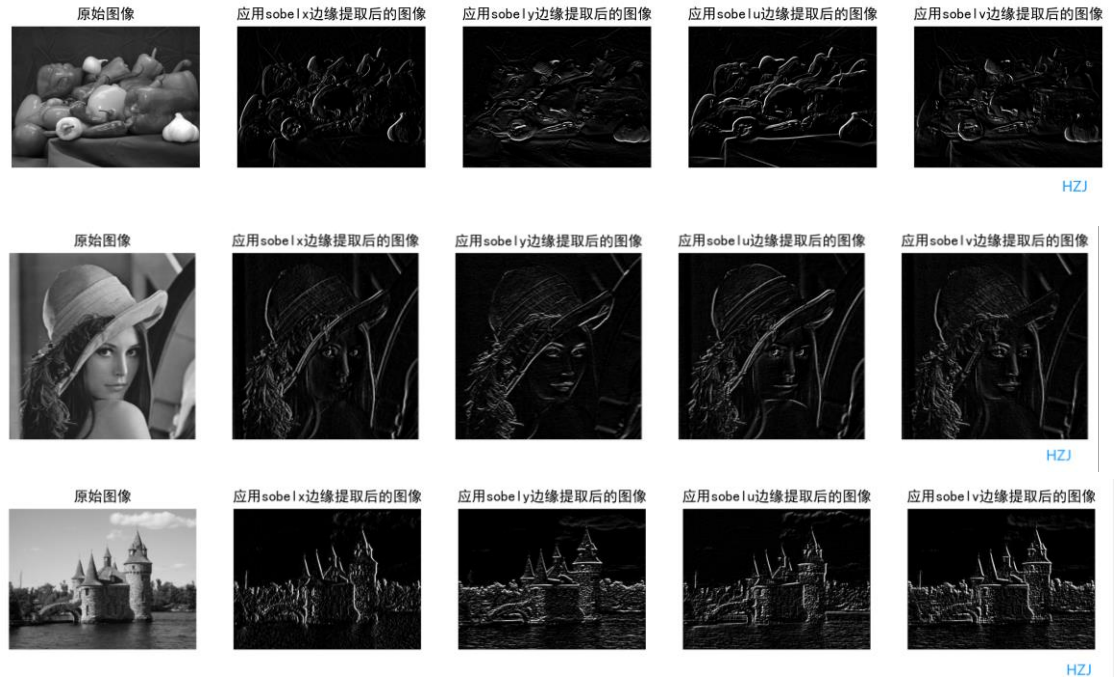
（3）最后将边缘图像加上原图，按照 1：1 的比例即可得到锐化后的图像，函数如下：

```
def sharpen_one_image(image, sobel):
    # 先应用Sobel算子
    edge_image = apply_one_sobel(image, sobel)
    height, width = image.shape
    sharpened_image = np.zeros( shape: (height, width), dtype=np.uint8)
    for i in range(height):
        for j in range(width):
            # 通过加上边缘图像来锐化原图
            sharpened_image[i][j] = (np.clip(image[i][j] * 0 + edge_image[i][j] * 1,  a_min: 0,  a_max: 255)).astype(np.uint8)
    return sharpened_image
                                                        HZJ
```

此处同样要注意：先把像素限制在 0-255 之间，再进行类型转换，防止精度损失。

结果展示：几张测试图像**锐化效果**如下：

同样的，如果不加上原图，只对**边缘进行提取**，可以得到以下效果：



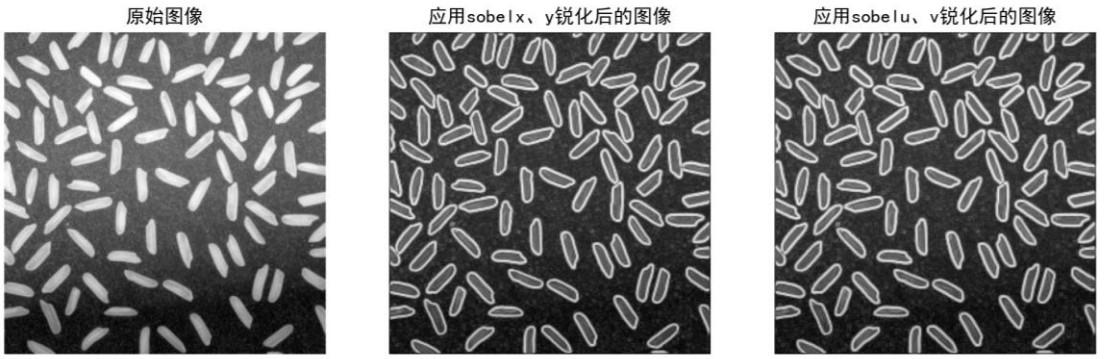
从结果可以明显看到各个方向 Sobel 算子的边缘提取和锐化效果，印证了各个 Sobel 算子的作用，实验成功。（**主要是精度要把握好**）

最后也是采用两个 Sobel 算子同时进行锐化，此时多了一步操作就是，怎么将两个锐化结果进行合并，这里采用计算梯度幅值的方法，即计算各自幅值的算术平方根，代码如下：

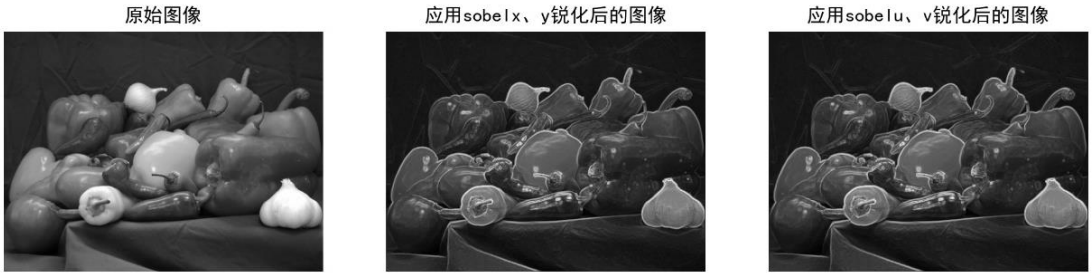
```python
def apply_sobel(image, sobel_x, sobel_y):
    # 获取图像的维度
    height, width = image.shape
    # 创建一个空的图像用于存储结果
    new_image = np.zeros( shape: (height, width), dtype=np.float32)
    # 进行卷积操作
    for i in range(1, height - 1):
        for j in range(1, width - 1):
            gx = 0
            gy = 0
            # 计算Gx和Gy
            for k in range(3):
                for l in range(3):
                    gx += image[i + k - 1][j + l - 1] * sobel_x[k][l]
                    gy += image[i + k - 1][j + l - 1] * sobel_y[k][l]
            # 计算梯度幅值 限制范围在0-255
            g = min(255, max(0, ((gx ** 2 + gy ** 2) ** 0.5)))
            new_image[i][j] = g
    return new_image
```

HZJ

将水平 x 方向和竖直 y 方向的 Sobel 算子一起使用，两个对角线方向 u、v 算子一起使用，同样应用于几张测试图片，锐化效果如下：



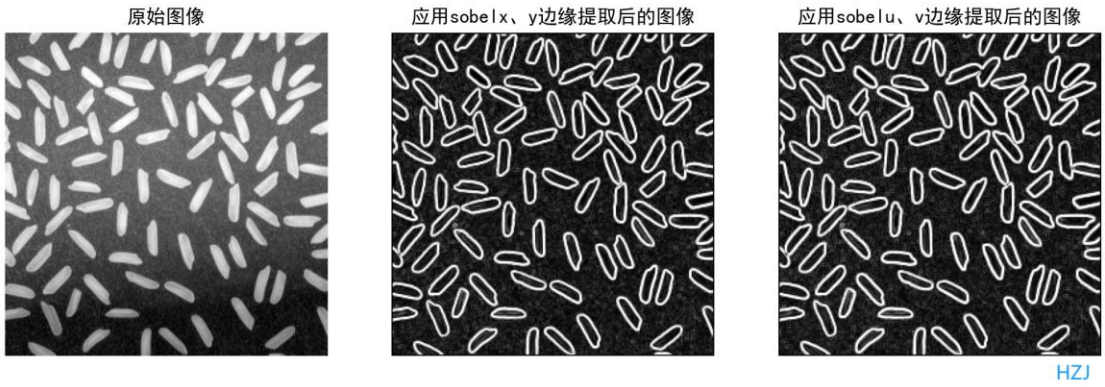
HZJ

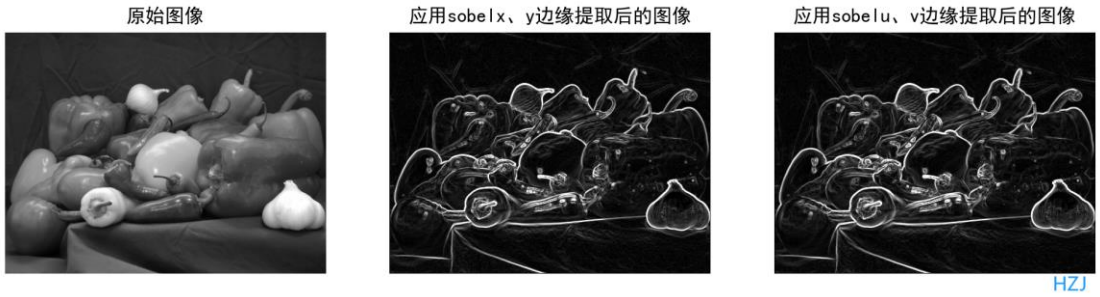


HZJ

HZJ

同样的边缘提取效果如下所示：（不加原图）





从结果可以看到，效果相比单个 Sobel 算子要好很多，同时我们也可以清晰的看到目标的轮廓，说明边缘检测的程序的编写较为成功，