In [1]:
```
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
import cv2

#importing packages
```

In [2]:
```
photo = Image.open("task1_3.jpg")
photo.size


#Open the image and save it as "photo"
```

Out[2]: (564, 846)

In [9]:
```
photo = photo.resize([180,270]) #resize因为原始图片比较大不适合展示
grey = photo.convert("L")
grey
#convert the photo to greyscale ie 黑白only
```

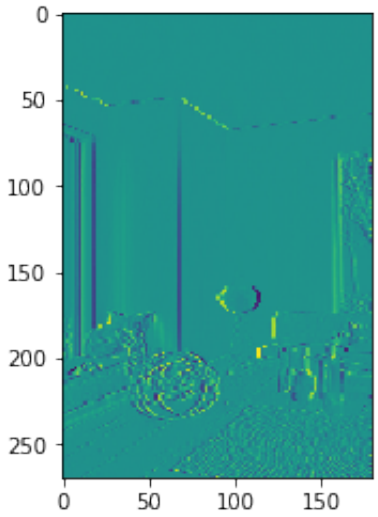
Out[9]:

In [10]:
```
grey= np.array(grey)/255.0
grey.shape
# 将jpg图片变成矩阵，并用0-1表示每个channel的深度
#看一下像素
```

Out[10]: (270, 180)

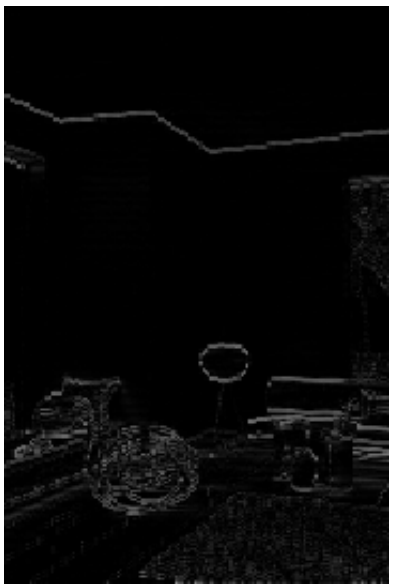
In [11]:
```
x,y= np.gradient(grey)
y.shape
plt.imshow(y)
#x，y分别储存了横向与纵向的一阶导 ie边缘
```

Out[11]: <matplotlib.image.AxesImage at 0x123fcd3c8>



In [12]:
```
Im2=np.abs(x)*255 #从0-1变回0-255 因为下面的image要以unit8的形式储存
newIm=Image.fromarray(Im2.astype("uint8"))
newIm
#从x中画出之前grey图片的横向边缘，背景色为黑
```

Out[12]:



In [13]:
```
Im3=(1-np.abs(x))*255
newIm=Image.fromarray(Im3.astype('uint8'))
newIm
#从x中画出之前grey图片的横向边缘，背景色为白（从黑图中减去黑色的部分）
```

Out[13]:

In [14]:
```
Im4=(1-np.abs(x)**0.25)*255 #想要保留黑色背景，只减去很小一部分的x
newIm=Image.fromarray(Im4.astype('uint8'))
newIm
```

Out[14]:

可以看到效果并不是很好，有大量留白很粗糙。也许一般的代数运算无法解决这个问题?

In [ ]:

In [ ]: