图像复原实验报告

蒋颜丞，自动化(电气)1903，3190102563

1 实验内容和要求

附件文件图像是长度为30、逆时针方向角度为11、加有高斯白噪声的移动模糊图像。试用一种方法复原该图像。

2 实验原理

2.1 维纳滤波

维纳滤波是使得原始图像与复原图像之间的均方误差$e^2 = E\{|f - \hat{f}|^2\}$最小的复原方法。其中$E$是期望值操作符，$f$是原始图像，$\hat{f}$是复原图像。图像的最佳估计的频谱为：

$$\begin{aligned} F(u,v) &= \frac{1}{H(u,v)} \times \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)} G(u,v) \\ &= \frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)} G(u,v) \end{aligned}$$

其中，$H(u,v)$表示退化函数，$H^*(u,v)$表示$H(u,v)$的复共轭，$S_\eta(u,v) = |N(u,v)|^2$表示噪声的功率谱，$S_f(u,v) = |F(u,v)|^2$表示未退化图像的功率谱，比率$S_\eta(u,v)/S_f(u,v)$称为信噪功率比。

事实上，我们往往无法事先知道信噪功率比，因此，在实际情况中，往往取一常量K来作为信噪功率比的近似，则此时图像的最佳估计的频谱应为：

$$F(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K} G(u,v)$$

其中，$F(u,v), H(u,v), G(u,v)$分别是清晰复原图的频谱，模糊核的频谱以及模糊图的频谱，K是噪声抑制因子，与原始图和噪声功率谱之比有关。依据逆滤波基本原则，加入噪声抑制因子，适当调节K值，可在高噪声条件下取得良好的复原效果。在本例中，K取0.02可以得到不错的复原效果。

复原后的图像即为：

$$\hat{f}(x,y) = \mathcal{F}^{-1}[F(u,v)]$$

维纳滤波方法设计简单，计算量小，抗噪性能高，在图像处理领域应用广泛。

2.2 点扩展函数

进行维纳滤波的关键是要知道退化函数$H(u,v)$，而$H(u,v)$即为点扩展函数PSF的傅立叶变换。对于运动模糊图片来说，其点扩展函数是一条由图像中点出发，沿模糊方向有一定长度的线段。由于本例已经给出了运动模糊的长度和角度，因此能够很容易地写出点扩展函数，只需要在黑色(值为0)背景中，将位于点扩展函数线段上的点置为1，然后再做归一化即可。因此，对本例的模糊图像进行复原是容易的。

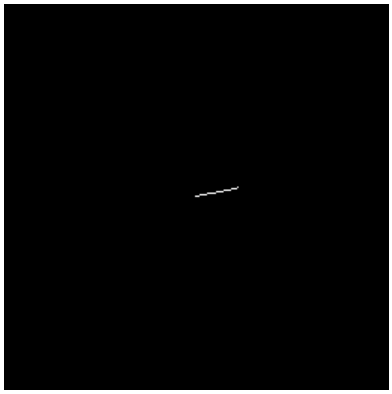
3 源代码

```python
import math
import numpy as np
import cv2

def cal_PSF(img_size, length, angle):
    '''
    计算点扩展函数
    输入：图像尺寸，运动长度，运动角度(弧度制)
    输出：点扩展函数
    '''
    PSF = np.zeros(img_size)  # 点扩展函数初始化
    h,w = img_size
    x_center = int((h - 1) / 2)
    y_center = int((w - 1) / 2) # 图像中心坐标

    # 将angle角度上length个点置成1
    for i in range(length):
        delta_x = round(math.sin(angle) * i)
        delta_y = round(math.cos(angle) * i)
        PSF[int(x_center - delta_x), int(y_center + delta_y)] = 1

    cv2.imwrite('PSF.png', PSF*255)
```

```
    PSF = PSF / np.sum(PSF) # 归一化
    return PSF

def wiener_filter(g, PSF, K=0.01):
    '''
    维纳滤波
    输入：模糊图像g，点扩展函数PSF，噪声抑制因子K
    输出：清晰复原图
    '''
    G = np.fft.fft2(g) # 傅里叶变换，计算模糊图频谱
    H = np.fft.fft2(PSF) # 傅里叶变换，计算模糊核频谱
    wiener_fft = np.conj(H) / (np.abs(H) ** 2 + K) # 维纳滤波器公式
    F = wiener_fft * G
    f = np.fft.ifftshift(np.fft.ifft2(F)) # 傅里叶逆变换得到复原后图像
    return f.real

if __name__ == '__main__':
    g = cv2.imread("origin_img.bmp",0) # 读取模糊图像
    length = 30 # PSF长度
    angle = 11 # PSF角度
    angle = angle * math.pi / 180 # 角度转弧度

    PSF = cal_PSF(g.shape, length, angle) # 计算点扩展函数
    f = wiener_filter(g, PSF, K = 0.02) # 维纳滤波
    f = f.astype(np.uint8) # 转换数据类型

    cv2.imwrite("restoreImage.png",f)
    cv2.waitKey(0)
```

4 实验结果与分析

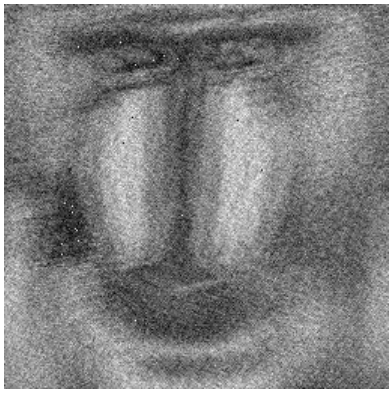
本例的点扩展函数图像为：



使用以上点扩展函数和噪声抑制因子K=0.02可以得到复原后的图像为：

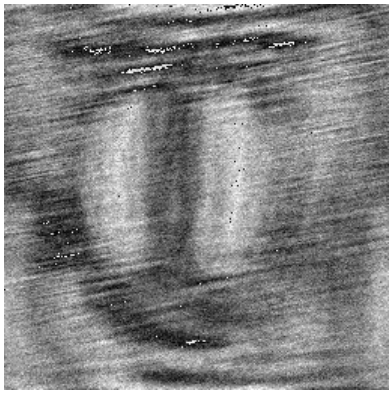
从图中可以看到，使用维纳滤波进行图像复原的效果非常不错，其原因在于点扩展函数是已知的，而在大多数情况中，点扩展函数是未知的，需要自行确定，复原效果就可能不如本例理想。

例如，将点扩展函数（PSF）的长度减小（设为15），进行复原，可以得到如下结果：



从图中可以看到，图像的模糊程度有所改善，但并未完全清晰，其原因就在于PSF的长度不够，复原不完全。

再将点扩展函数（PSF）的角度增大（设为30），进行复原，可以得到如下结果：



从图中可以看到，图像的模糊程度并无明显改善，且出现了十分明显的条状纹理，其原因就在于PSF的角度与运动模糊方向角度并不一致，复原并不有效。

此外，还可以注意到复原后的图像仍然包含一定噪声，其原因在于噪声抑制因子K可以抑制噪声，但不能完全消除噪声，可以在复原后采用滤波方法对图像进行进一步的处理，以提高图像复原的质量。