实验报告

16340282 袁之浩

算法描述

1.运动模糊

$$H(u,v)=\frac{T}{\pi(ua+vb)}\sin[\pi(ua+vb)]e^{-j\pi(ua+vb)} \qquad (5.6\text{-}11)$$

使用书上给的退化函数构造频域滤波器，对图像和滤波器都要进行中心化。注意分母可能为0，这时直接将函数值赋为1。

2.高斯噪声

在原图像矩阵上加上一个噪声矩阵。产生噪声矩阵的方法是，产生一个和原图像等大的、值在0到1之间的随机矩阵，作为概率矩阵。对每一个概率值在高斯分布上逆向求得对应该概率值的横坐标，该横坐标就是噪声矩阵对应位置上的值。这样产生的噪声矩阵就会服从以0为均值，500为方差的高斯分布。

3.逆滤波

逆滤波是最简单的复原方法。我们用退化函数除以退化图像的傅里叶变换G(u, v)来计算原始图像傅里叶变换的估计F(x, y)，即

$$\hat{F}(u,v)=\frac{G(u,v)}{H(u,v)} \qquad (5.7\text{-}1)$$

在没有噪声时，逆滤波能取得不错的效果，但有噪声时，会形成明显干扰。

4.维纳滤波

维纳滤波也叫最小均方差滤波，该方法建立在图像和噪声都是随机变量的基础上，目标是找出未污染图像f的一个估计，使得它们之间的均方误差最小。这里用到了一个知识：一个复数量与其共轭的乘积等于该复数量幅度的平方。

$$\hat{F}(u,v)=\left[\frac{H^*(u,v)S_f(u,v)}{S_f(u,v)|H(u,v)|^2+S_\eta(u,v)}\right]G(u,v)=\left[\frac{H^*(u,v)}{|H(u,v)|^2+S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$$
$$=\left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2+S_\eta(u,v)/S_f(u,v)}\right]G(u,v) \quad (5.8\text{-}2)$$

H(u, v)是退化函数的变换，G(u, v)是退化图像的变换，Sn(u,v) / Sf(u,v)是噪声功率谱和原图像功率谱的比值，由于该值较难确定，所以实验过程中用多个常数来替代该值，直到找到使恢复效果较优的值。我们通常使用下面的表达式来近似

$$\hat{F}(u,v)=\left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2+K}\right]G(u,v) \quad (5.8\text{-}6)$$

关键代码

运动模糊退化函数

```
function H=myFilter(height,width,T,a,b)
for u=1:height
    for v=1:width
        x=u-height/2;
        y=v-width/2;
        c=pi*(x*a+y*b);
        if(c==0)
            H(u,v)=T;
        else
            H(u,v)=T/c*sin(c)*exp(-1j*c);
        end
    end
end
end
```

产生高斯噪声

```
function gn = GaussianNoise(g, mean, var)
    [height, width] = size(g);
    noise = norminv(rand(height, width), mean,
sqrt(var));
    gn = noise + double(normalize(g));
end
```

逆滤波器

```
function f_inv = InverseFilter(g, H)
    [height, width] = size(g);
```

```
    G = fft2(Centralize(g));
    B = BLPF(50, 2, height, width);

    for u = 1 : height
        for v = 1 : width
            if abs(H(u, v)) < 0.001
                tmp = 1;
            else
                tmp = H(u, v);
            end
            F(u, v) = G(u, v) * B(u, v) / tmp;
        end
    end

    f_inv = ifft2(F);
    f_inv = real(f_inv);
    f_inv = Centralize(f_inv);
end
```

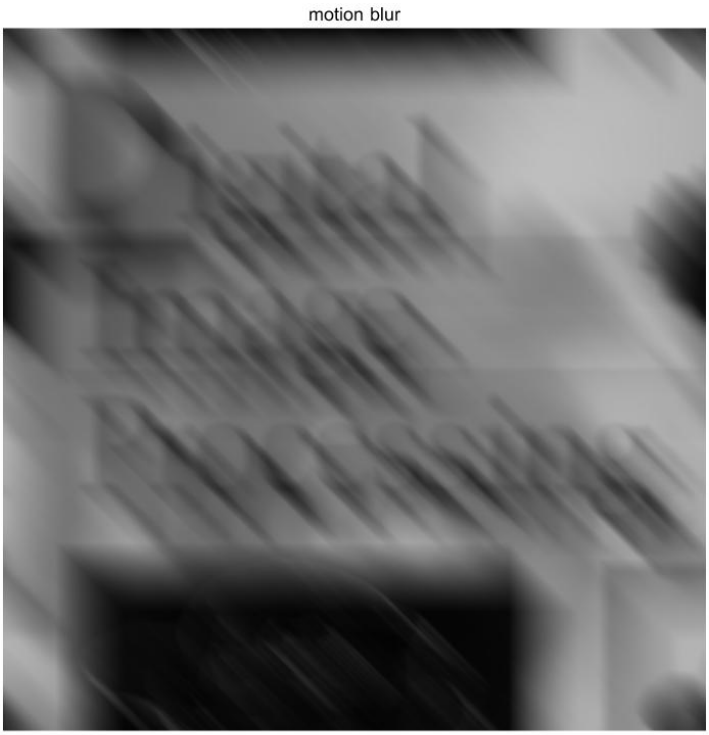
维纳滤波

```
function f = WienerFilter(g, H, k)
    G = fft2(Centralize(g));
    H2 = H .* conj(H);
    F = (G .* H2) ./ (H .* (H2 + k));
    f = ifft2(F);
    f = real(f);
    f = Centralize(f);
end
```

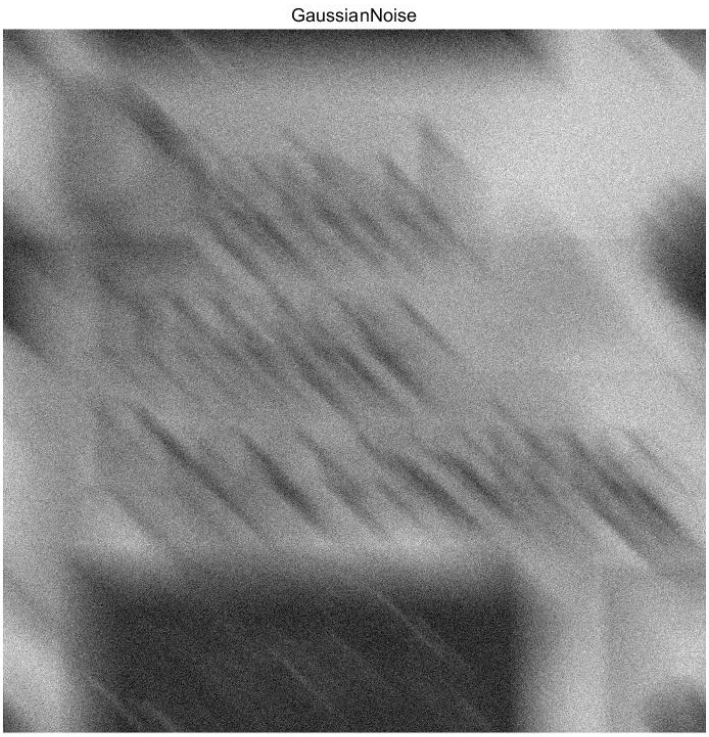
完整代码见code文件夹

实验结果

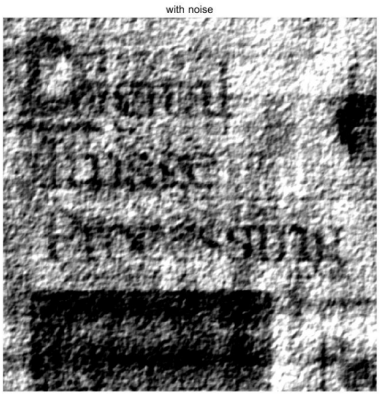
运动模糊



添加高斯噪声



使用逆滤波器恢复，左边是对有噪声图像恢复，右边是对无噪声图像恢复，可以看到右边的效果要好很多。



使用维纳滤波器对有噪声图像进行恢复，调整K的值，并与逆滤波器的结果进行比较，可以看到维纳滤波器在K=0.01左右的效果比较好。