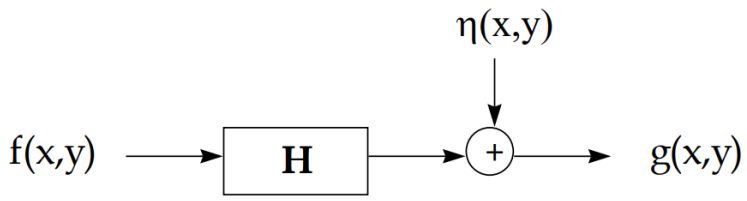
Report

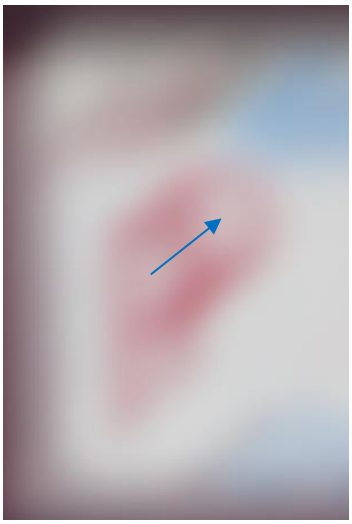
1. Blind image restoration

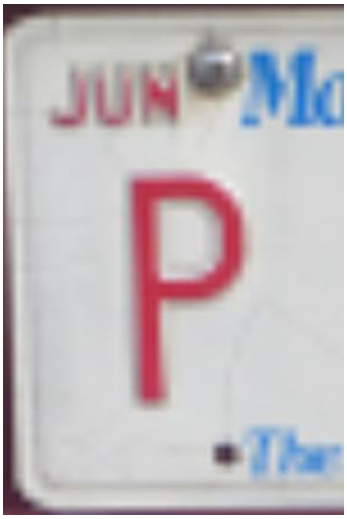
For the first task, we need to restore the input1.bmp without using the known clean image (input1_ori.bmp).

Frequency Domain: $G(u,v) = H(u,v)F(u,v) + N(u,v)$



From the above, we are given only $g(x, y)$. We don't know the degradation function (H) and the noise $\eta(x, y)$ of the process. But we can observe the blurred image by human eyes to estimate the degradation process.





From the above images, we can guess that the blurring process is following the $45°$ direction and roughly counting the difference between two center points of P is about 20 pixels. Therefore, I use the motion blur kernel with 21 as the kernel size and $45°$ as the blurring direction.

On the other hand, I try the Gaussian blur kernel with different sigma values to make the deblurring effect more significant.

PSNR: 78.065 dB

PSNR: 64.747 dB

```
for(int i=0; i<Nb; i++) {
    for(int j=0; j<Nb; j++) {
        if(i == j)
            MotionBlurKernel[i][j] = 1.0 / double(Nb);
        else
            MotionBlurKernel[i][j] = 0;
    }
}
```

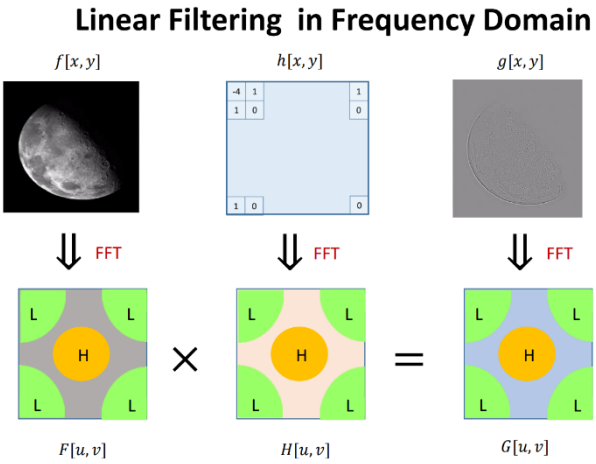
MotionBlurKernel with 45°

```
// generate Gaussian blur kernel
for(int x=-padding; x<padding+1; x++) {
    for(int y=-padding; y<padding+1; y++) {
        r = (x * x + y * y);
        GaussianBlurKernel[x+padding][y+padding] = (exp((-1) * r / dev)) / (M_PI * dev);
        sum += GaussianBlurKernel[x+padding][y+padding];
    }
}
// normalize kernel
for(int i=0; i<N; i++) {
    for(int j=0; j<N; j++) {
        GaussianBlurKernel[i][j] /= sum;
    }
}
```

After preparing those kernels, we want to compute them in frequency domain.

We need to place the kernel values following the rule like this ->

Do 2D-FFT to get the frequency response of the kernel. The H from the formula in this task is combining the Gaussian blur kernel and the Motion blur kernel, which is simply multiplying in frequency domain. However, when we want to restore the image to get $\hat{F}$, we cannot directly divide $G(u,v)$ by $H(u,v)$, this is because $H(u,v)$ may be close to zero at some point which is often happened at high frequency component. Hence, we need the K term to compensate the effect.



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

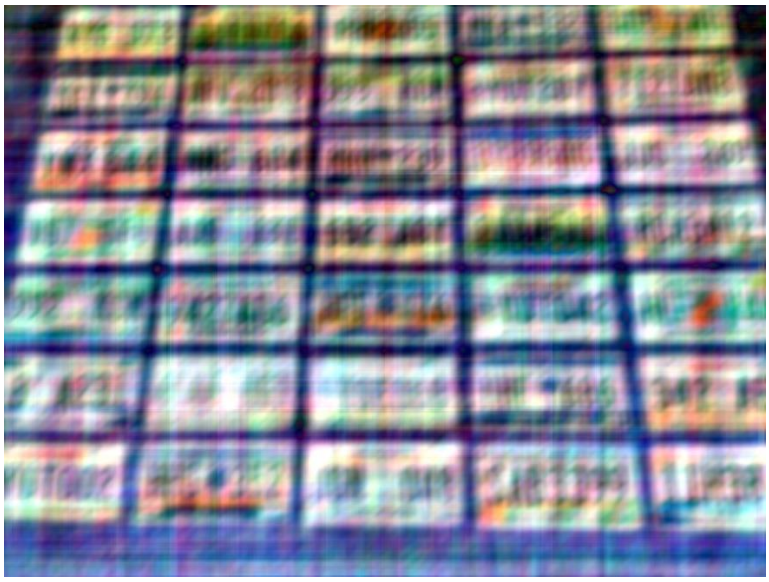
```
// do fft
fft2d(Bplane, 1);
fft2d(Gplane, 1);
fft2d(Rplane, 1);

fft2d(GaussianBlurKernel_F, 1);
fft2d(MotionBlurKernel_F, 1);

for(int y=0; y<pad_height; ++y) {
    for(int x=0; x<pad_width; ++x) {
        complex<double> H = GaussianBlurKernel_F[y][x] * MotionBlurKernel_F[y][x];
        complex<double> wf = (1.0 / H) * (pow(abs(H), 2) / (pow(abs(H), 2) + eps));
        Bplane[y][x] = Bplane[y][x] * wf;
        Gplane[y][x] = Gplane[y][x] * wf;
        Rplane[y][x] = Rplane[y][x] * wf;
    }
}

// do ifft
fft2d(Bplane, -1);
fft2d(Gplane, -1);
fft2d(Rplane, -1);
```

As for deblurring using both input1.bmp and input1_ori.bmp to estimate the degradation function. I fail on making good estimation to restore the image.



Input2.bmp using blind method

WYG 573	11FH756	PHP 2455	MKA 532	405 ZMU
MAY 794	AFV 2018	993 KCM	YUT207	7121AN8
YMX 644	MMG 604	MKM 239	378984K	JJS 269
V67 SFL	JJS 131	552 AOY	2AA4510	RCA3412
992 KCM	9427A06	HPR 476	YUT 042	HLF V4
8 A231	4144 AGH	YSE068	MHF 686	342 A8
YUT002	HHG352	JGN 048	SAB3399	11H38