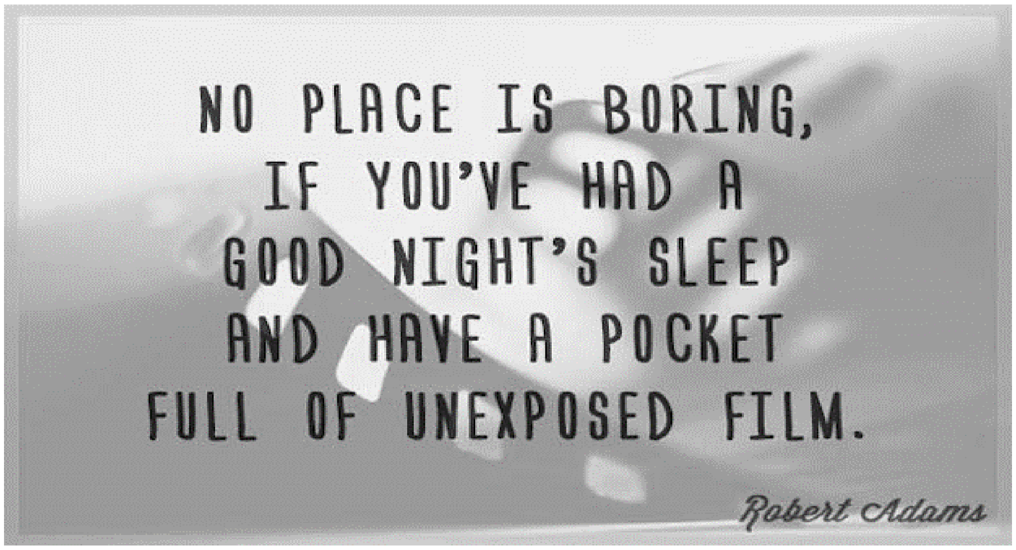
# Current progress in Deblurring :

*Our original image*



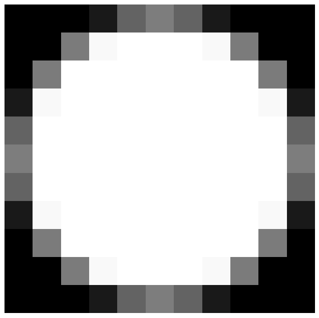
## A little information on Blurs

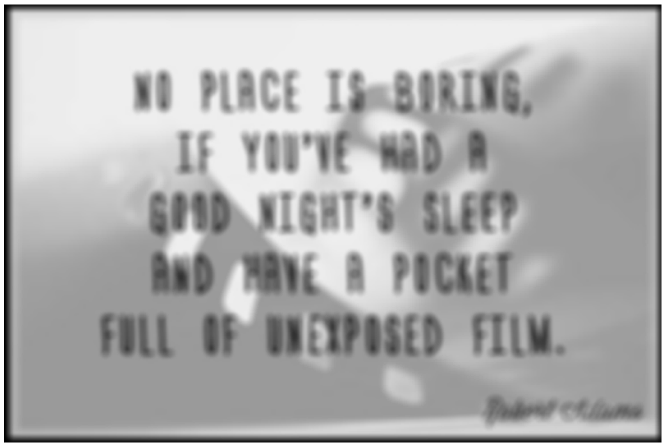
The blurs below are the important ones we need to consider.

The result image is attached for reference.

### 1)Disk blurr (Similar to bokeh) :

```
        0         0         0    0.0012    0.0050    0.0063    0.0050    0.0012         0         0         0
        0    0.0000    0.0062    0.0124    0.0127    0.0127    0.0127    0.0124    0.0062    0.0000         0
        0    0.0062    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0062         0
   0.0012    0.0124    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0124    0.0012
   0.0050    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0050
   0.0063    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0063
   0.0050    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0050
   0.0012    0.0124    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0124    0.0012
        0    0.0062    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0127    0.0062         0
        0    0.0000    0.0062    0.0124    0.0127    0.0127    0.0127    0.0124    0.0062    0.0000         0
        0         0         0    0.0012    0.0050    0.0063    0.0050    0.0012         0         0         0
```





PSF with disk radius 5

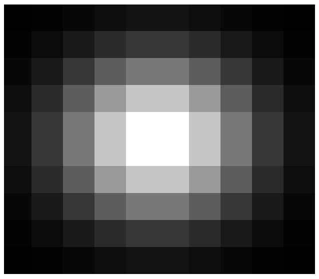
Blurred output

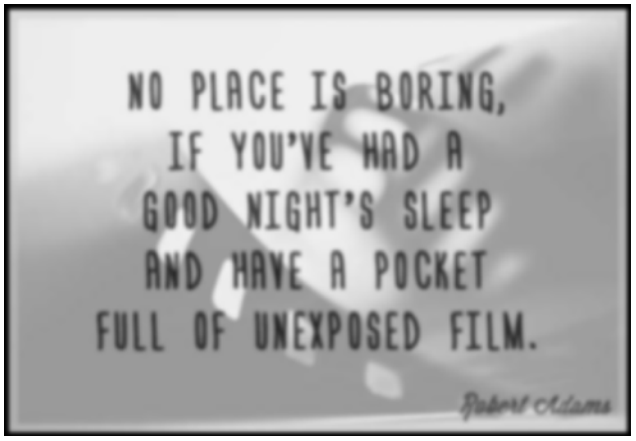
## 2)Gaussian blurr :

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Size 10 sigma 2

```
 0.0003    0.0007    0.0015    0.0024    0.0031    0.0031    0.0024    0.0015    0.0007    0.0003
 0.0007    0.0019    0.0040    0.0066    0.0085    0.0085    0.0066    0.0040    0.0019    0.0007
 0.0015    0.0040    0.0085    0.0141    0.0181    0.0181    0.0141    0.0085    0.0040    0.0015
 0.0024    0.0066    0.0141    0.0232    0.0298    0.0298    0.0232    0.0141    0.0066    0.0024
 0.0031    0.0085    0.0181    0.0298    0.0383    0.0383    0.0298    0.0181    0.0085    0.0031
 0.0031    0.0085    0.0181    0.0298    0.0383    0.0383    0.0298    0.0181    0.0085    0.0031
 0.0024    0.0066    0.0141    0.0232    0.0298    0.0298    0.0232    0.0141    0.0066    0.0024
 0.0015    0.0040    0.0085    0.0141    0.0181    0.0181    0.0141    0.0085    0.0040    0.0015
 0.0007    0.0019    0.0040    0.0066    0.0085    0.0085    0.0066    0.0040    0.0019    0.0007
 0.0003    0.0007    0.0015    0.0024    0.0031    0.0031    0.0024    0.0015    0.0007    0.0003
```





PSF

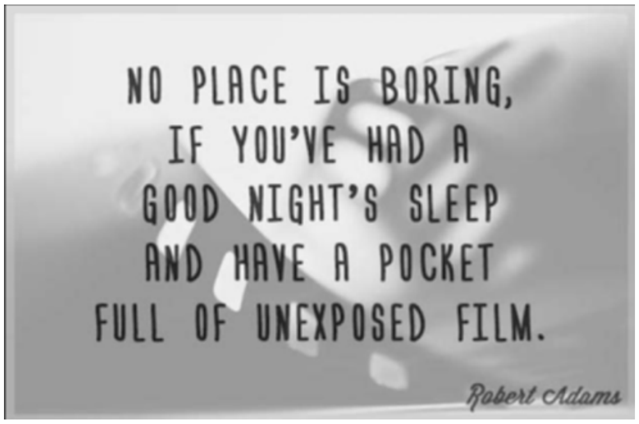
Average Blurr:

Matrix of size n
With elements 1
Divided by the weight $n^2$

n=3

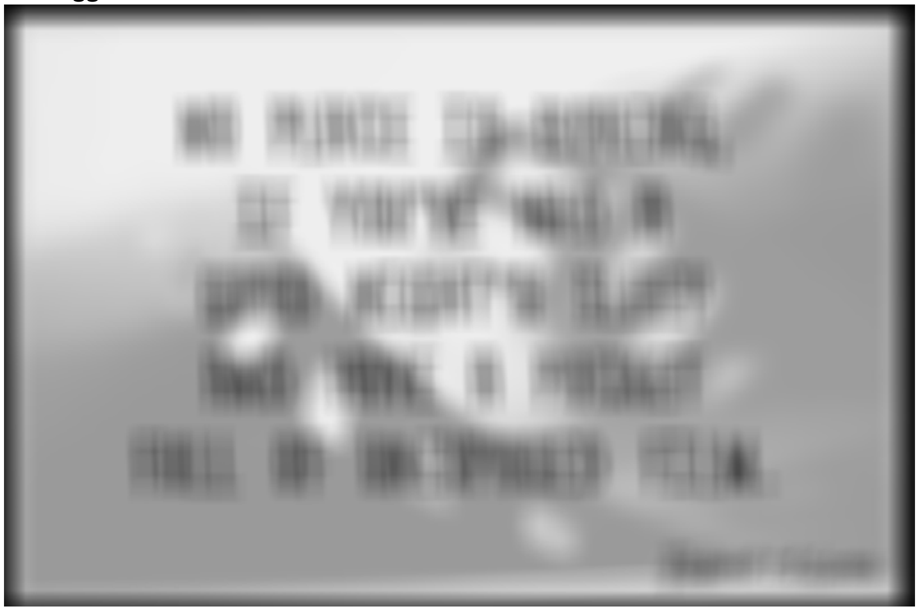
```
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
```

PSF



Result

n=20
For bigger size blurs

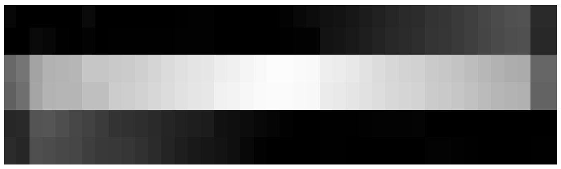


Result

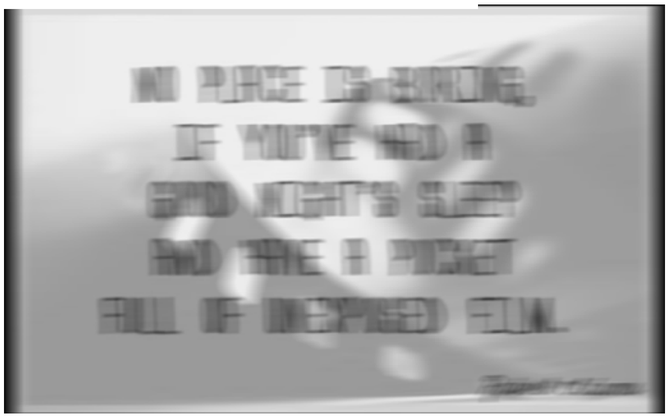
4)Motion blurr :

```
      0         0         0         0         0         0         0         0         0         0    0.0000
 0.0196    0.0341    0.0358    0.0375    0.0393    0.0410    0.0428    0.0445    0.0462    0.0480    0.0497
 0.0080    0.0156    0.0139    0.0122    0.0104    0.0087    0.0070    0.0052    0.0035    0.0018    0.0000

 0.0018    0.0035    0.0052    0.0070    0.0087    0.0104    0.0122    0.0139    0.0156    0.0080
 0.0480    0.0462    0.0445    0.0428    0.0410    0.0393    0.0375    0.0358    0.0341    0.0196
      0         0         0         0         0         0         0         0         0         0
```



PSF



Result

There are still other kernels available but this are the most significant and almost all can be traced back to the above.

## Methods of deblurring we employed

### 1)FFT method

Since the Images are convolved with the PSF's.

convolution( I, PSF ) = J

In frequency domain, it's just a basic multiplication

I x PSF = J

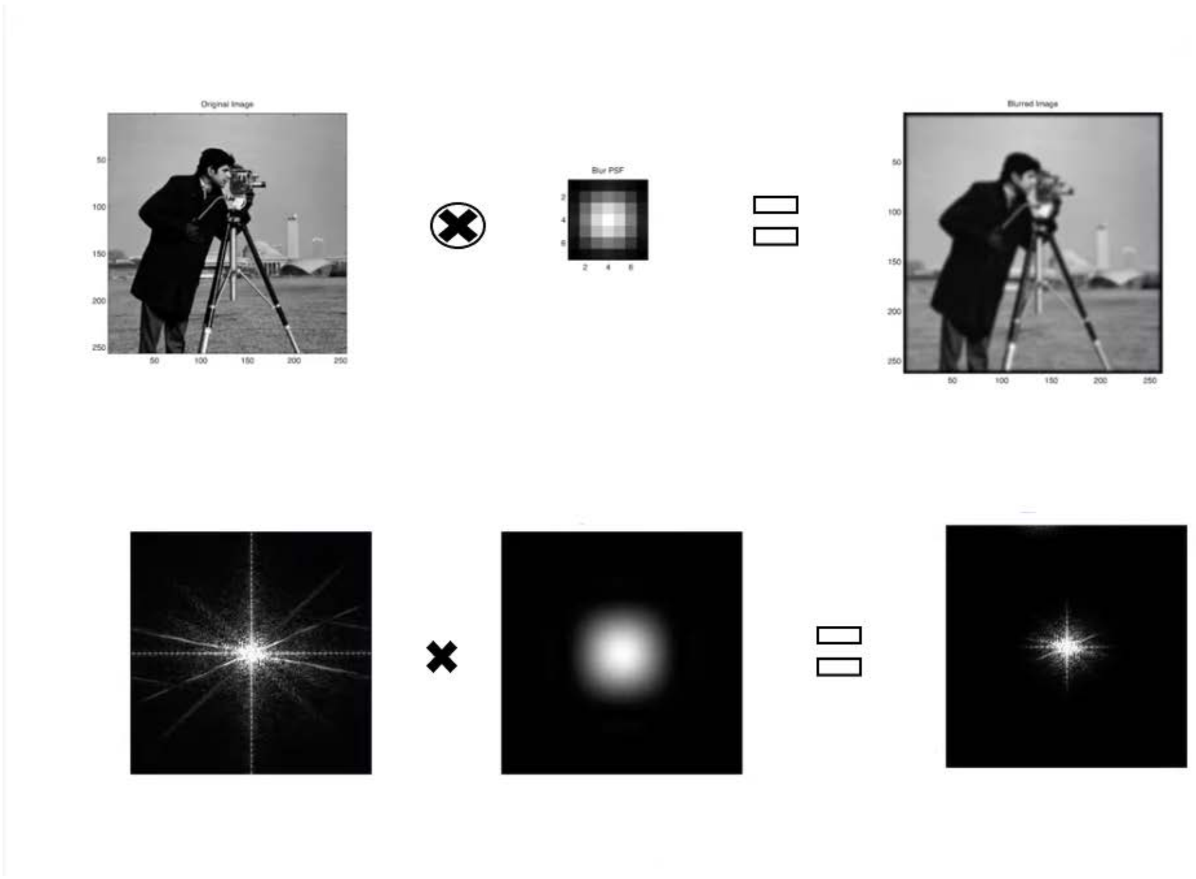
Therefore

I = J/PSF

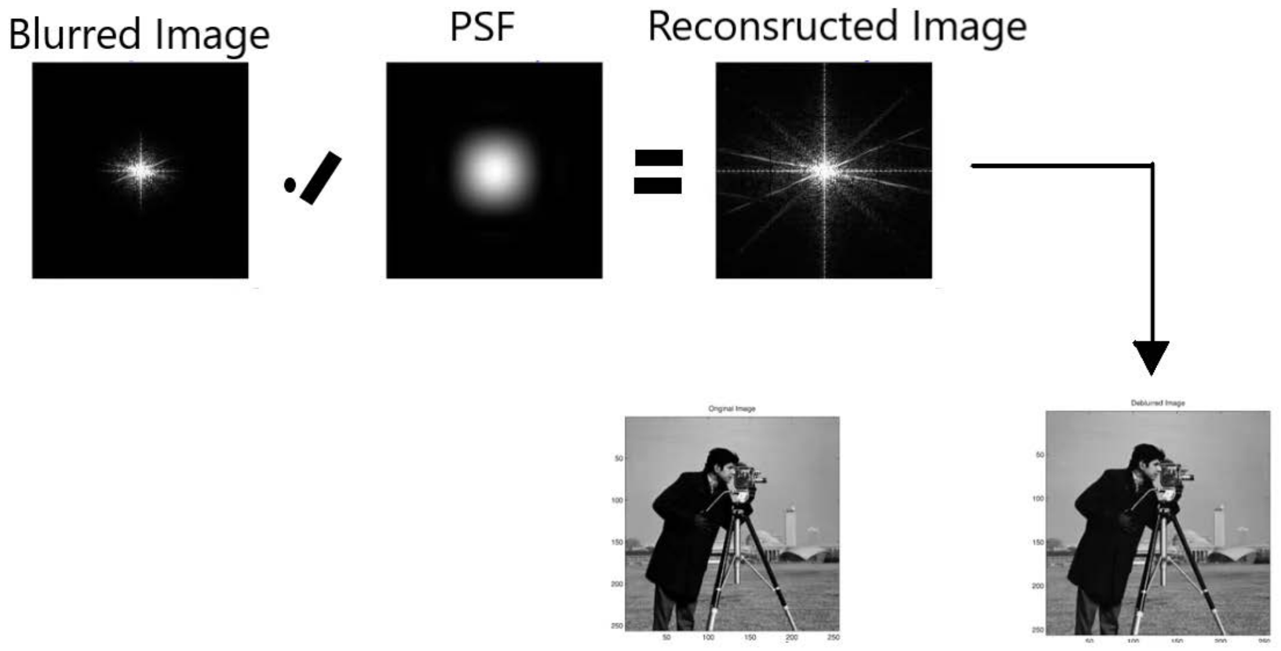
Taking Inverse gives us back our image

This method also works with multiple layers of blur , given that the user knows about the type of blur used. A Video is attached regarding the working of this approach .

Burring normally vs Blurring in frequency domain:



Deblurring :

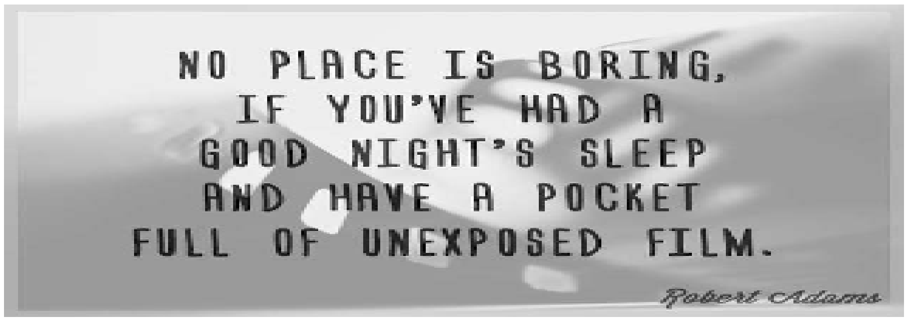


Although this method works exceptional well. There is still a problem about noise which greatly hampers the restoration ability.

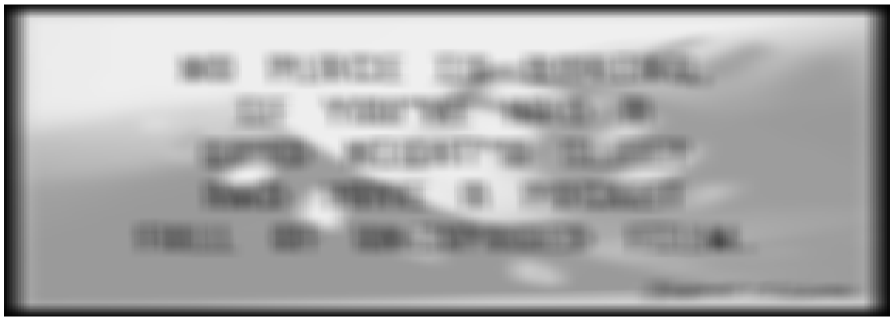
Our idea is to use this with our below function.

## 2)Blind deconvolution :

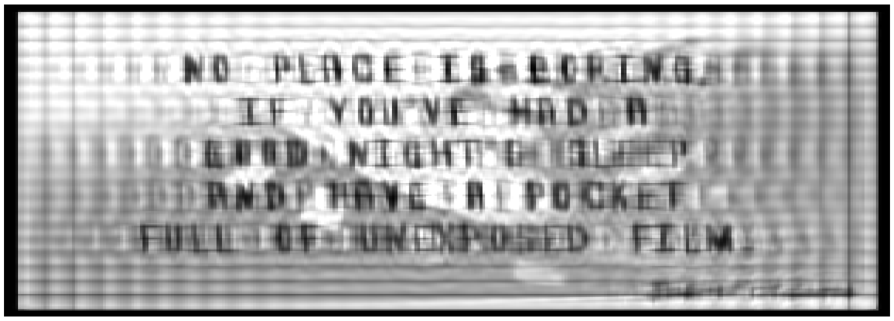
Original image



Blurred input



Restored image



The above were the results obtained using the program of blind deconvolution(2) .

Blind deconvolution does not necessarily require the exact knowledge of the PSF used .

The program was implemented using blind deconvolution by means of Lucy-Richardson algorithm .

Matlab's builtin function 'deconvblind' also uses this algorithm for implementation .

References for the program:

1)Blind deconvolution by means of the Richardson-Lucy algorithm –D.A Fish,A.MBrinicombe and E.R.Pike

2)Acceleration of iterative image restoration algorithm – Davis S.C. Biggs and Mark Andrews , April 1997

3)Matlab’s builtin function ‘deconvblind’.