EE5175 - Lab 5

Space-variant Blurring

Nikilesh B
EE17B112

Space-variant Blurring :

Now we assume the blur to be space-variant, i.e. the standard deviation varies for each pixel.

We Consider the distribution of **σ** to be as follows :

$$\boldsymbol{\sigma}(m,n) = A \exp \frac{-\left(\left(m-\frac{N}{2}\right)^2+\left(n-\frac{N}{2}\right)^2\right)}{B}, \quad 0 \le m,n \le N-1$$

with

$$\boldsymbol{\sigma}\left(\frac{N}{2},\frac{N}{2}\right) = 2.0 \text{ and } \boldsymbol{\sigma}(0,0) = 0.01,$$

where N x N is the size of the image and pixel indices are in the range [0 , N -1] x [0, N -1]. Find A and B, and create the matrix **σ** . Perform Gaussian blurring on Globe.png using the values of **σ**(m,n).

The given Globe.png is attached below :

The **space variant function block** is as follows :

```
def varblur(src_img) :
      Nr,Nc = src_img.shape
      fin = np.zeros(src_img.shape)

      kmax = 13   #Go along rows and then along columns
      for i in range(kmax,Nr+kmax) :
      for j in range(kmax,Nc+kmax) :     #Obtain blur kernel-sigma values
            A = 2 ; B = (N*N/2)/(-np.log(0.01/A))
            ioff = i-kmax ; joff = j-kmax
            sig = A*np.exp(-((ioff-(N/2))**2+(joff-(N/2))**2)/B)
            kernel = generate_kernel(sig)   #Apply kernel on the image
            kext = len(kernel)//2

            img = np.zeros((int(Nr+2*kmax),int(Nc+2*kmax)))
            img[kmax:Nr+kmax,kmax:Nc+kmax] = src_img
            patch = np.zeros(kernel.shape)
            patch = img[i-kext:i+kext+1,j-kext:j+kext+1]
            patch = patch*kernel
            fin[i-kmax,j-kmax] = sum(sum(patch))
    return fin
```

The **function block for obtaining kernel for a given sigma** is attached below :

```
def generate_kernel(sig) :
      k = int(np.ceil(6*sig +1))
      if k%2 == 0 :
      k = k+1
      kernel = np.zeros((k,k)) ; mid = k//2
      for i in range(0,mid+1) :
      row = np.arange(mid+i,k)
      roweff = row-mid
      kernel[mid-i,row] = (1/(2*np.pi*sig*sig))*np.exp(-(roweff*roweff +
i*i)/(2*sig*sig))
      kernel[mid-roweff[1:],mid+i] = kernel[mid-i,row][1:]

      kernel[:mid+1,:mid] = np.fliplr(kernel[:mid+1,mid+1:])
      kernel[mid+1:,:] = np.flipud(kernel[:mid,:])
      kernel = kernel/sum(sum(kernel))
      return kernel
```

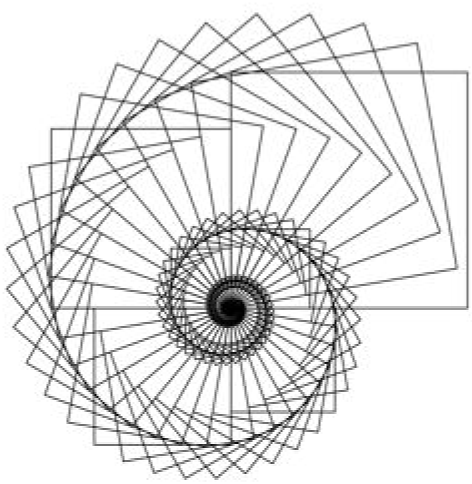
The obtained output is attached below:

2. Now for the second part we are going to blur Nautilus.png using

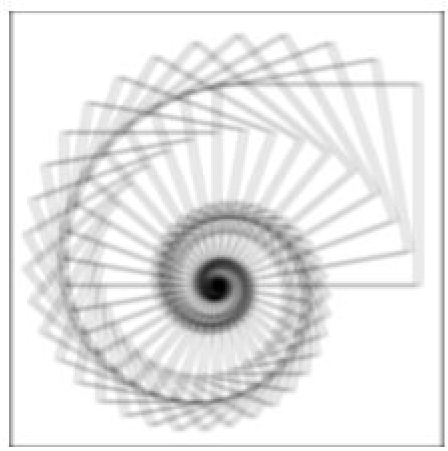
(a) space-invariant blur code of part 1 with $\boldsymbol{\sigma} = 1.0$, and
(b) space-variant blur code of part 2 with $\boldsymbol{\sigma}(m, n) = 1.0$ for $0 \le m, n \le N - 1$.

And finally we verify that the blurred images of the above two steps are the same.

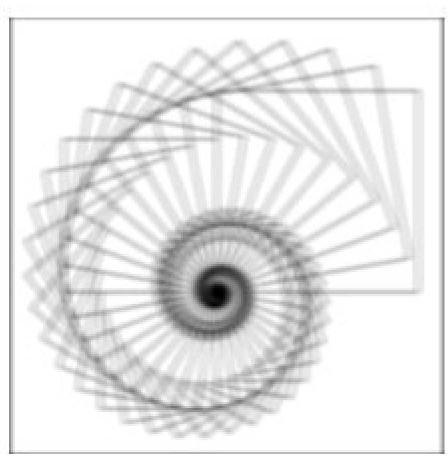
The given image is attached below :



The output image using **space-invariant blur** is attached below:



The second output image using **space-variant blur** is attached below:



We can thus verify that both the images obtained from step 1 and step 2 are the same .