EE5175 - Lab 4

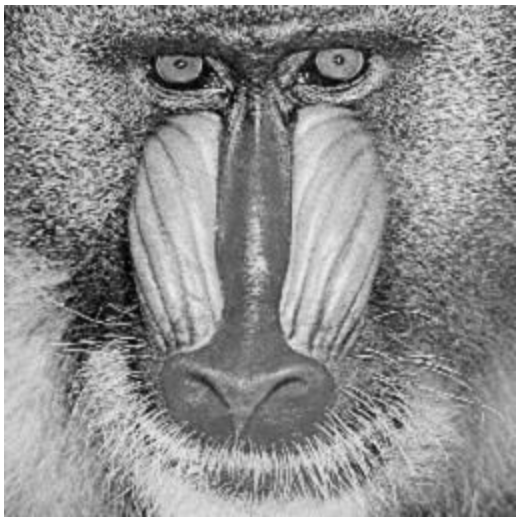
Space-invariant Blurring

Nikilesh B
EE17B112

Our aim in this assignment is to perform Gaussian blurring on the given image (Mandrill.png) with standard deviation .

We assume **space-invariant blur** and a kernel of size [6σ+1][6σ+1]. And we observe the outputs for the values of **σ = 1.6, 1.2, 1.0, 0.6, 0.3 ,0.0.**

The given image is attached below:



We import the image and set the sigma values as shown below :

```
src_img = cv2.imread("Mandrill.png",0)
sigma = np.array([1.6, 1.2, 1, 0.6, 0.3, 0])
```

The function block for space invariant blur is as follows :

```
def invblur(src_img,kernel) :

      kext = len(kernel)//2
      Nr,Nc = src_img.shape
      img = np.zeros((int(Nr+2*kext),int(Nc+2*kext)))
      fin = np.zeros(src_img.shape)
      img[kext:Nr+kext,kext:Nc+kext] = src_img
      patch = np.zeros(kernel.shape)

      #Go along rows and then along columns
      for i in range(kext,Nr+kext) :
      for j in range(kext,Nc+kext) :
            patch = img[i-kext:i+kext+1,j-kext:j+kext+1]
            patch = patch*kernel
            #print(sum(patch))
            fin[i-kext,j-kext] = sum(sum(patch))

      return fin
```

The function block for obtaining kernel for a given sigma is as follows:

```
def generate_kernel(sig) :
      k = int(np.ceil(6*sig +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(kernel)
      return kernel
```

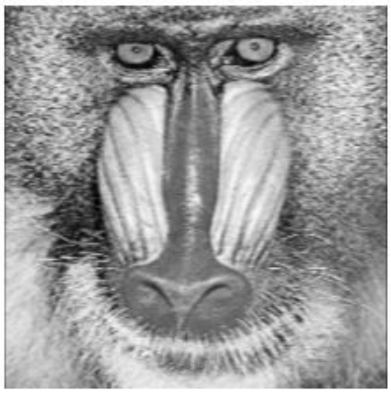
(i) for **σ = 1.6** we get the output image as follows :

(i) for **σ = 1.2** we get the output image as follows :

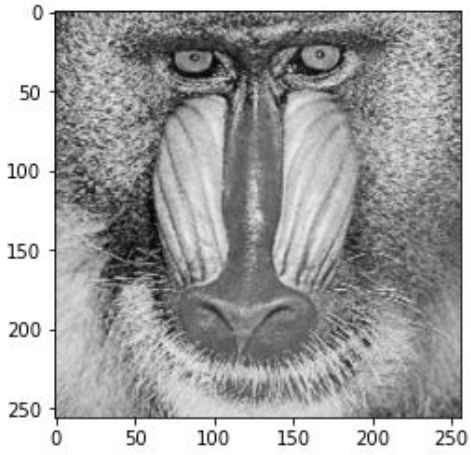
(i) for **σ = 1.0** we get the output image as follows :

(i) for **σ = 0.6** we get the output image as follows :

(i) for **σ = 0.3** we get the output image as follows :



(i) for **σ = 0** we get the output image as follows :



As expected we see the blurring increases as we increase the standard deviation from 0 to 1.6