Experiment Title: 7

Student Name: Pratibha **UID:18BCS6093**

Branch:AIML-1 **Section/Group:B**

Semester:5 **Date of Performance:**

Subject Name:DIP LAB **Subject Code: CSF-336**

1. Aim/Overview of the practical:

Write the python program which help us to understand the implementation and technicalities behind the image blurring and filtering techniques using the required libraries.

2. The task to be done:

- Blurring
- Filtering

3. Required libraries or software

import cv2

import numpy as np

4. Algorithm/Flowchart :

5. Theme/Interests definition(For creative domains):

6. Steps for experiment/practical:

```
# ## Image Blurring Techniques
# In[2]:
# importing cv2 module
import cv2
img = cv2.imread('aa.jpg')
# You can change the kernel size as you want
```

```
blurImg = cv2.blur(img,(10,10))
cv2.imshow('blurred image',blurImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
# ###  1.1Averaging
# In[2]:
# importing opencv CV2 module
import cv2
# bat.jpg is the batman image.
img = cv2.imread('aa.jpg')
# Averaging
# You can change the kernel size as you want
avging = cv2.blur(img,(10,10))
cv2.imshow('Averaging',avging)
cv2.waitKey(0)
# ### 1.2 Gaussian Blurring
# In[3]:
# Gaussian Blurring
gausBlur = cv2.GaussianBlur(img, (5,5),0)
cv2.imshow('Gaussian Blurring', gausBlur)
cv2.waitKey(0)
# ### 1.3 Median Blurring
# In[4]:
# Median blurring
medBlur = cv2.medianBlur(img,5)
cv2.imshow('Media Blurring', medBlur)
cv2.waitKey(0)
# ### 1.4 Bilateral Blurring
# In[7]:
# Bilateral Filtering
bilFilter = cv2.bilateralFilter(img,9,75,75)
cv2.imshow('Bilateral Filtering', bilFilter)
cv2.waitKey(0)
```

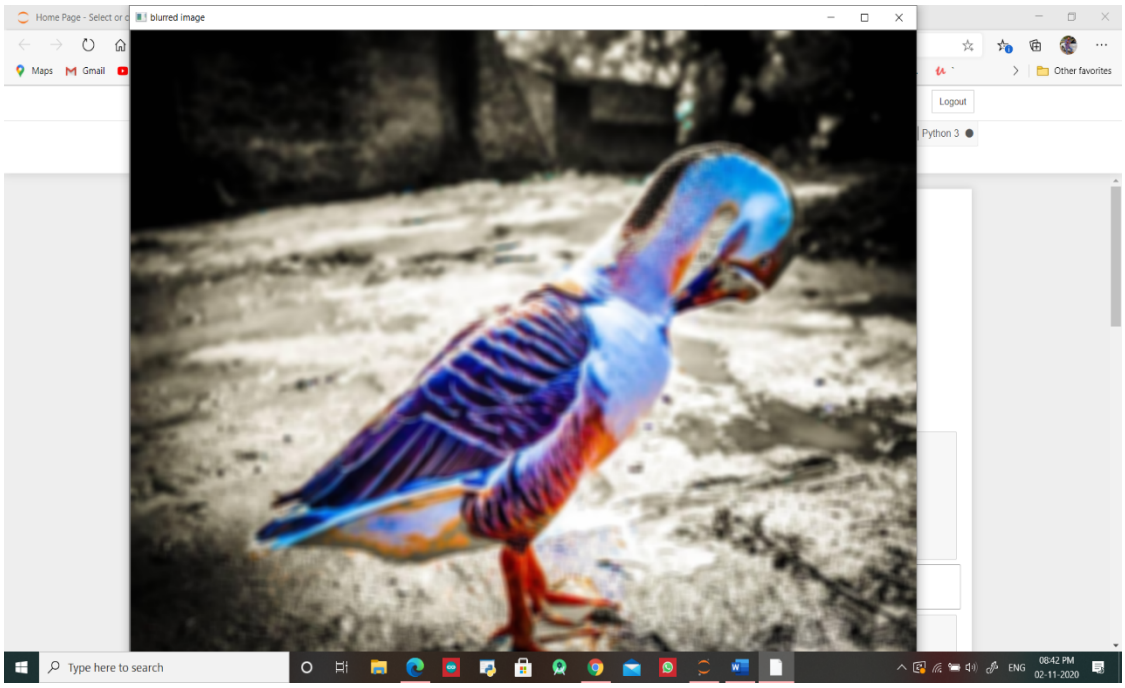
```
# ### 1.5 Motion Blur
# In[9]:
import cv2
import numpy as np
img = cv2.imread('aa.jpg')
# The greater the size, the more the motion.
kernel_size = 30
# Create the vertical kernel.
kernel_v = np.zeros((kernel_size, kernel_size))
# Create a copy of the same for creating the horizontal kernel.
kernel_h = np.copy(kernel_v)
# Fill the middle row with ones.
kernel_v[:, int((kernel_size - 1)/2)] = np.ones(kernel_size)
kernel_h[int((kernel_size - 1)/2), :] = np.ones(kernel_size)
# Normalize.
kernel_v /= kernel_size
kernel_h /= kernel_size
# Apply the vertical kernel.
vertical_mb = cv2.filter2D(img, -1, kernel_v)
# Apply the horizontal kernel.
horizonal_mb = cv2.filter2D(img, -1, kernel_h)
# Save the outputs.
cv2.imwrite('swan_vertical.jpg', vertical_mb)
cv2.imwrite('swan_horizontal.jpg', horizonal_mb)
# Display the image
cv2.imshow('swan_vertical', vertical_mb)
cv2.imshow('swan_horizontal', horizonal_mb)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

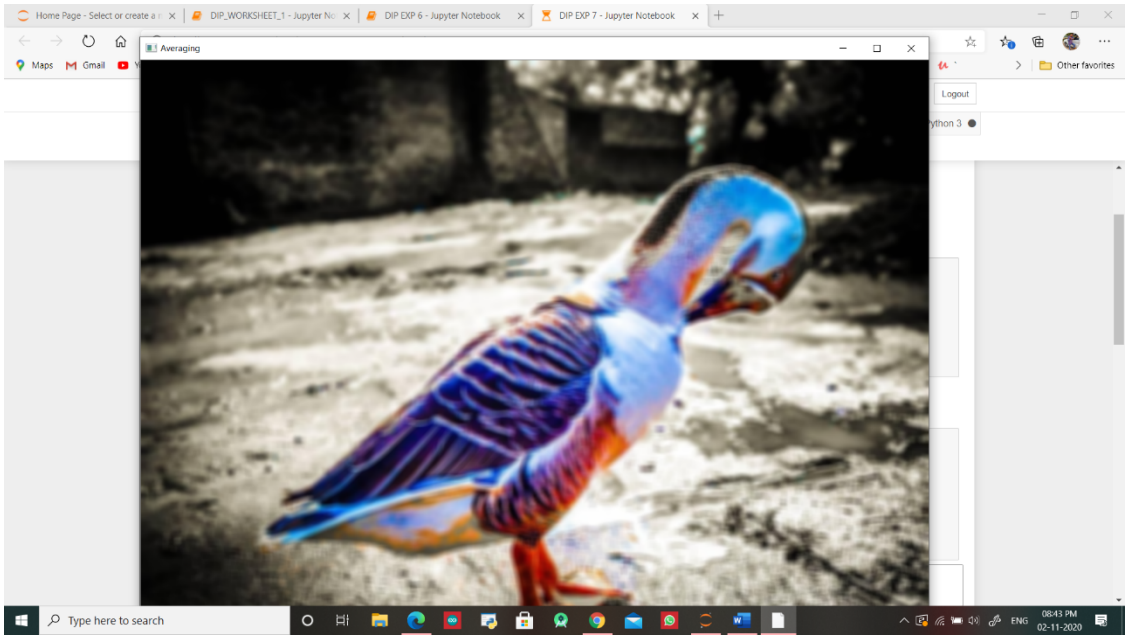
7. Observations/Discussions(For applied/experimental sciences/materials based labs):

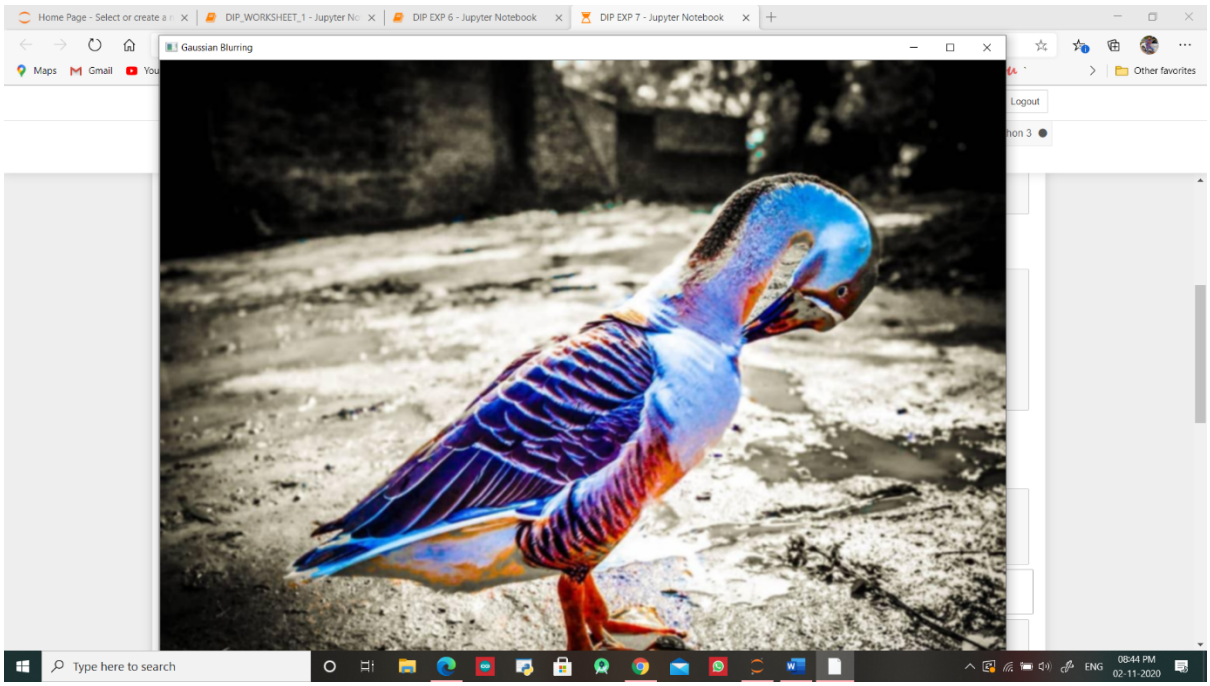
8. Percentage error (if any or applicable):

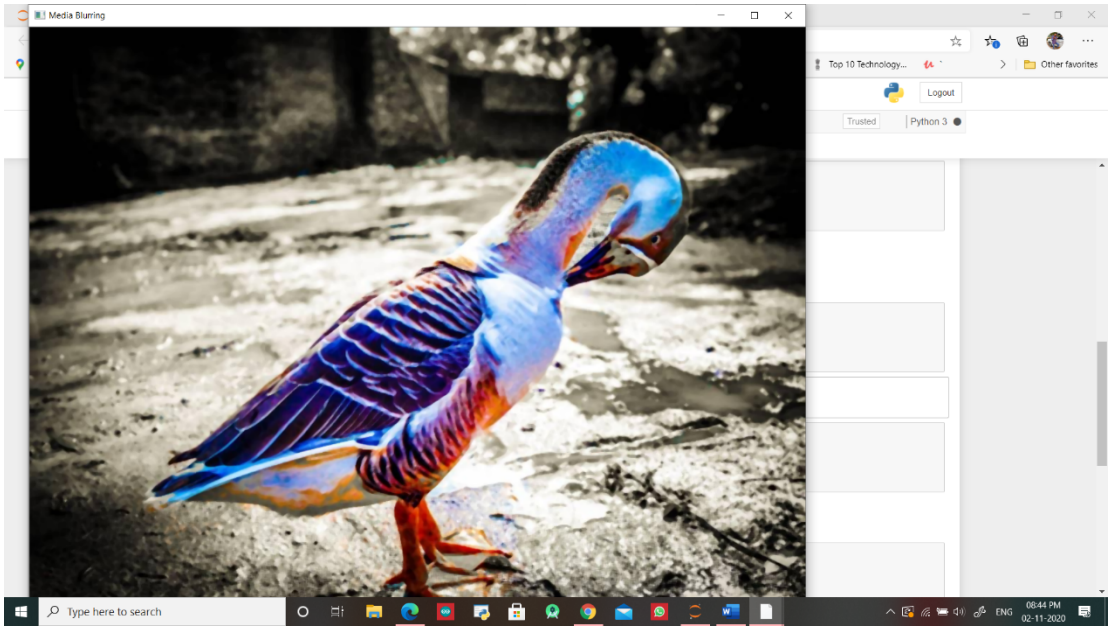
9. The command that we have learned today in the program :

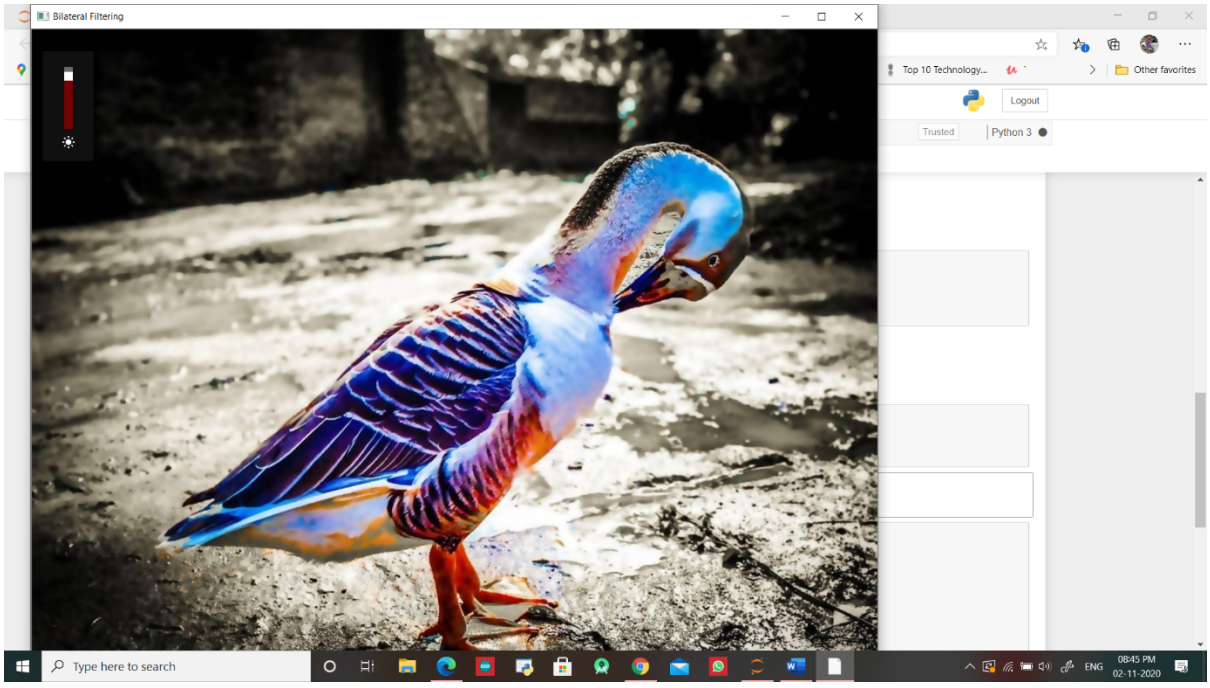
10. Result/Output/Writing Summary of the concept behind the experiment:

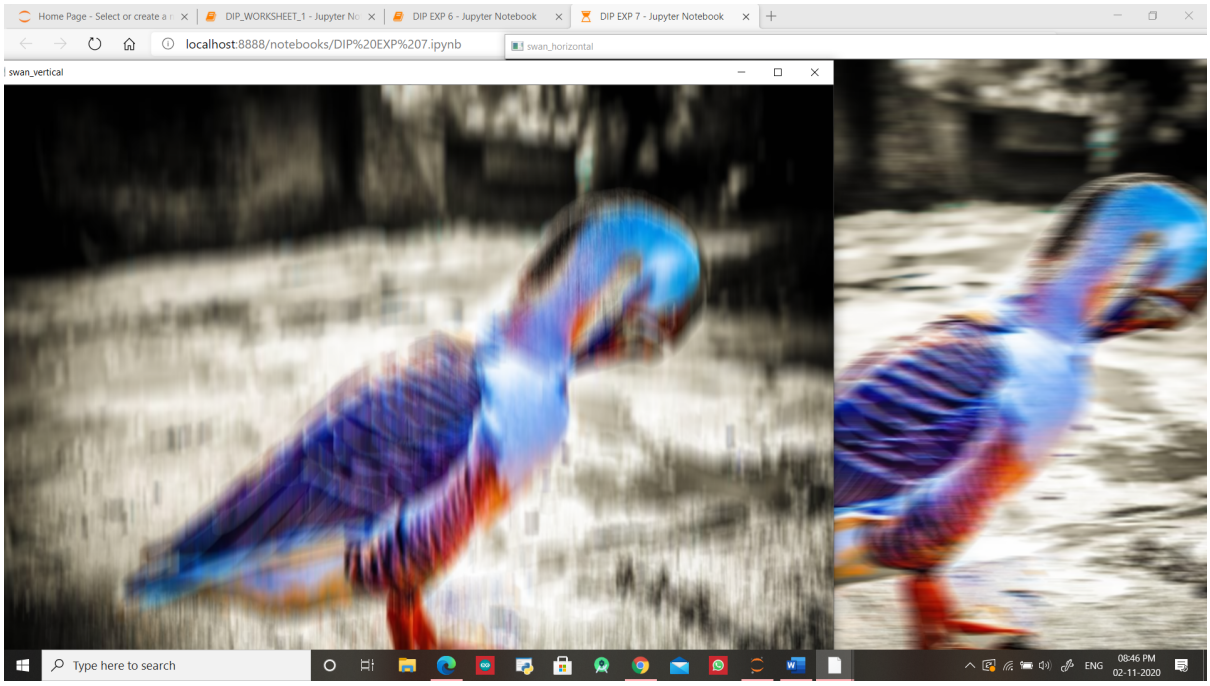



Averaging
Type here to search


Gaussian Blurring
Type here to search


Media Blurring
Top 10 Technology...
Other favorites
Logout
Trusted
Python 3
Type here to search
ENG
08:44 PM
02-11-2020


Bilateral Filtering
Top 10 Technology...
Other favorites
Logout
Trusted
Python 3
Type here to search
ENG
08:45 PM
02-11-2020



11. Graphs (If Any): Image /Soft copy of graph paper to be attached here