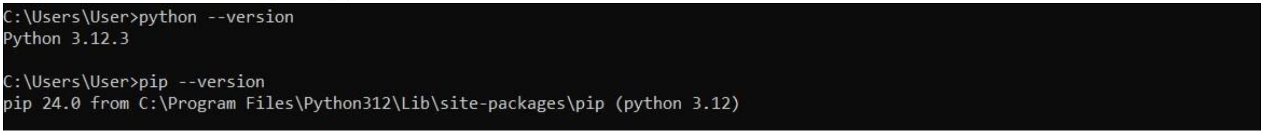
Note: Basic instruction to follow before or while performing practicals Step 1: Install latest Version of python.

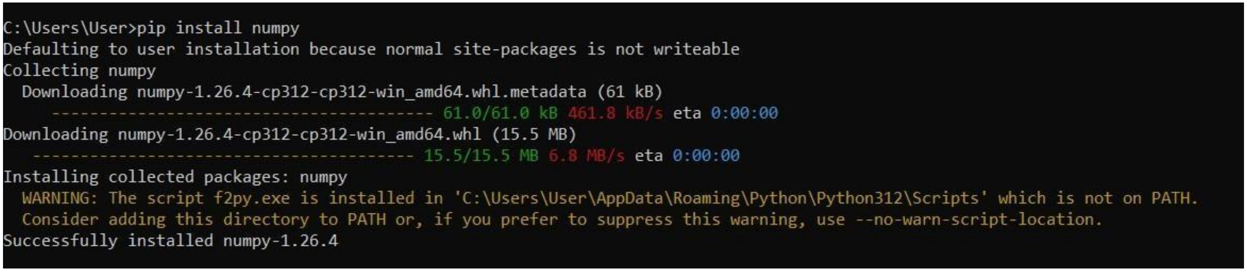


```
C:\Users\User>python --version
Python 3.12.3

C:\Users\User>pip --version
pip 24.0 from C:\Program Files\Python312\Lib\site-packages\pip (python 3.12)
```

Step 2: Set the environment variable path.

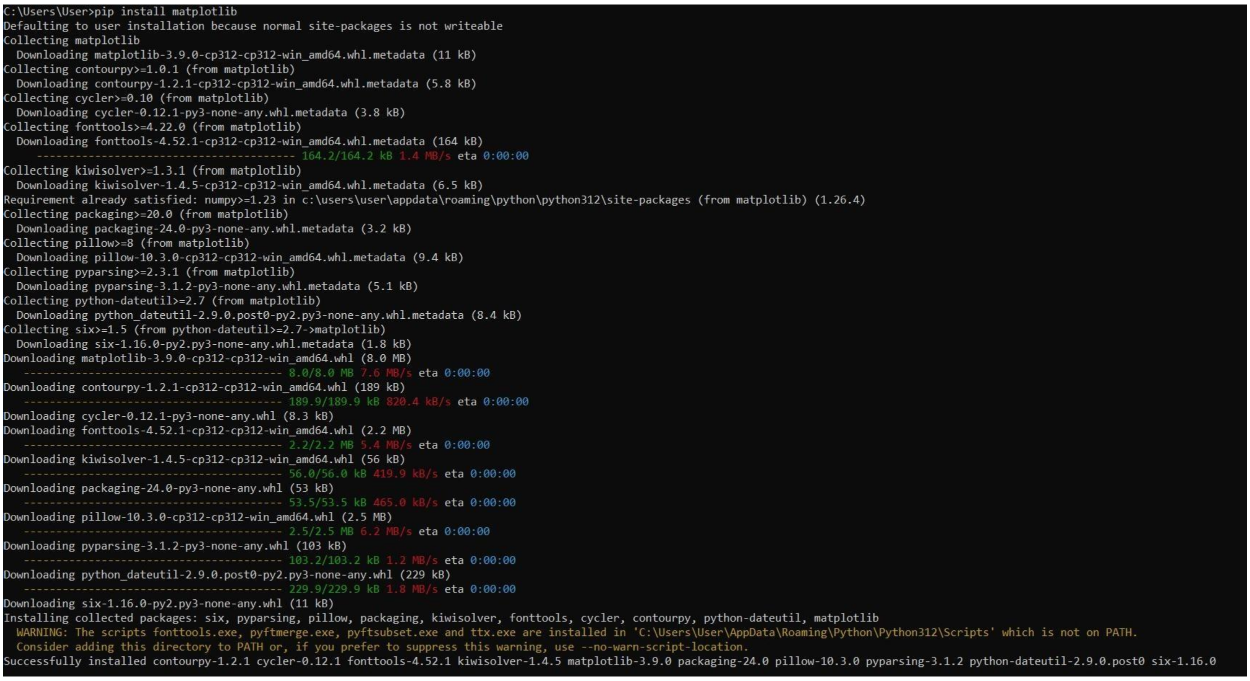
Step 3: Install packages as per required for particular practical.

i. Install numpy (cmd : pip install numpy)



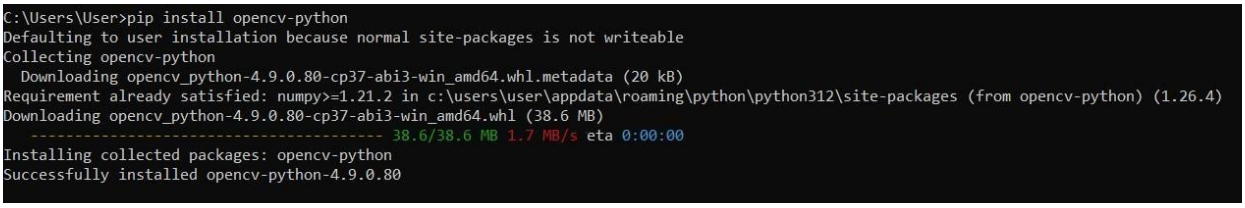
```
C:\Users\User>pip install numpy
Defaulting to user installation because normal site-packages is not writeable
Collecting numpy
  Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl.metadata (61 kB)
     ---------------------------------------- 61.0/61.0 kB 461.8 kB/s eta 0:00:00
Downloading numpy-1.26.4-cp312-cp312-win_amd64.whl (15.5 MB)
   ---------------------------------------- 15.5/15.5 MB 6.8 MB/s eta 0:00:00
Installing collected packages: numpy
  WARNING: The script f2py.exe is installed in 'C:\Users\User\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.26.4
```

ii. Install matplotlib (cmd : pip install matplotlib)



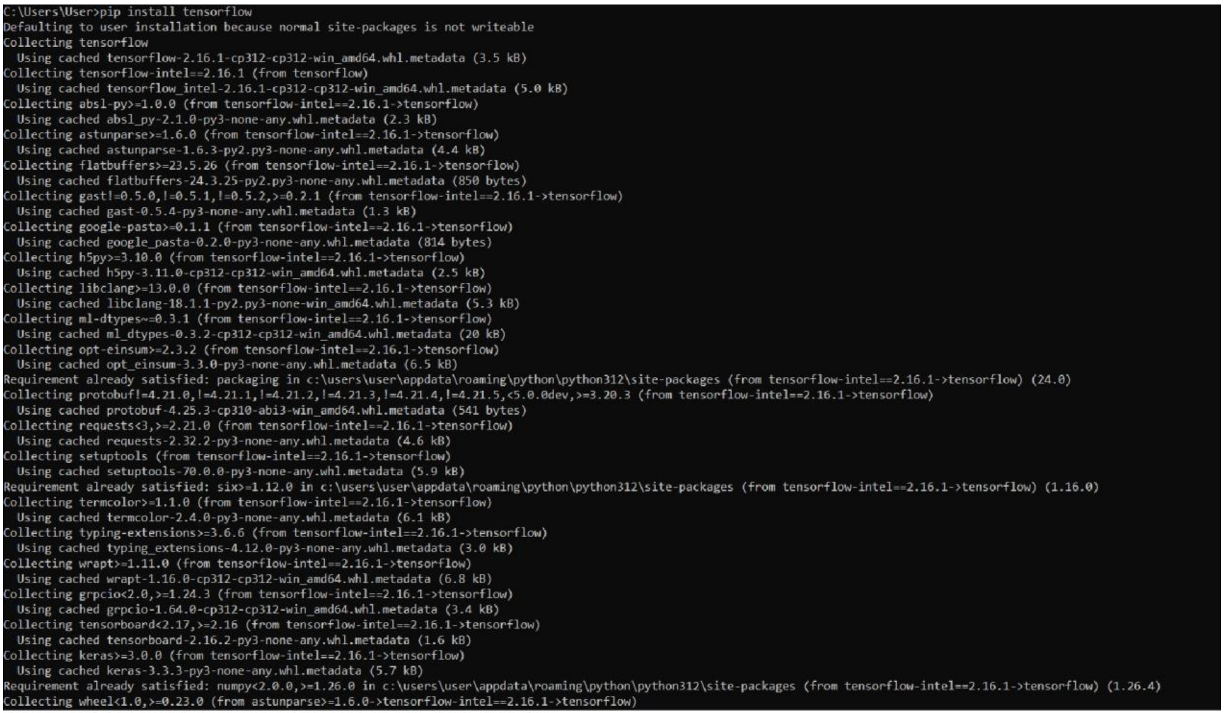
```
C:\Users\User>pip install matplotlib
Defaulting to user installation because normal site-packages is not writeable
Collecting matplotlib
  Downloading matplotlib-3.9.0-cp312-cp312-win_amd64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.2.1-cp312-cp312-win_amd64.whl.metadata (5.8 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.52.1-cp312-cp312-win_amd64.whl.metadata (164 kB)
     ---------------------------------------- 164.2/164.2 kB 1.4 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.5-cp312-cp312-win_amd64.whl.metadata (6.5 kB)
Requirement already satisfied: numpy>=1.23 in c:\users\user\appdata\roaming\python\python312\site-packages (from matplotlib) (1.26.4)
Collecting packaging>=20.0 (from matplotlib)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-10.3.0-cp312-cp312-win_amd64.whl.metadata (9.4 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.1.2-py3-none-any.whl.metadata (5.1 kB)
Collecting python-dateutil>=2.7 (from matplotlib)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.7->matplotlib)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Downloading matplotlib-3.9.0-cp312-cp312-win_amd64.whl (8.0 MB)
   ---------------------------------------- 8.0/8.0 MB 7.6 MB/s eta 0:00:00
Downloading contourpy-1.2.1-cp312-cp312-win_amd64.whl (189 kB)
   ---------------------------------------- 189.9/189.9 kB 820.4 kB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.52.1-cp312-cp312-win_amd64.whl (2.2 MB)
   ---------------------------------------- 2.2/2.2 MB 5.4 MB/s eta 0:00:00
Downloading kiwisolver-1.4.5-cp312-cp312-win_amd64.whl (56 kB)
   ---------------------------------------- 56.0/56.0 kB 419.9 kB/s eta 0:00:00
Downloading packaging-24.0-py3-none-any.whl (53 kB)
   ---------------------------------------- 53.5/53.5 kB 465.0 kB/s eta 0:00:00
Downloading pillow-10.3.0-cp312-cp312-win_amd64.whl (2.5 MB)
   ---------------------------------------- 2.5/2.5 MB 6.2 MB/s eta 0:00:00
Downloading pyparsing-3.1.2-py3-none-any.whl (103 kB)
   ---------------------------------------- 103.2/103.2 kB 1.2 MB/s eta 0:00:00
Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
   ---------------------------------------- 229.9/229.9 kB 1.8 MB/s eta 0:00:00
Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pyparsing, pillow, packaging, kiwisolver, fonttools, cycler, contourpy, python-dateutil, matplotlib
  WARNING: The scripts fonttools.exe, pyftmerge.exe, pyftsubset.exe and ttx.exe are installed in 'C:\Users\User\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed contourpy-1.2.1 cycler-0.12.1 fonttools-4.52.1 kiwisolver-1.4.5 matplotlib-3.9.0 packaging-24.0 pillow-10.3.0 pyparsing-3.1.2 python-dateutil-2.9.0.post0 six-1.16.0
```

iii.Install opencv (cmd : pip install opencv-python)

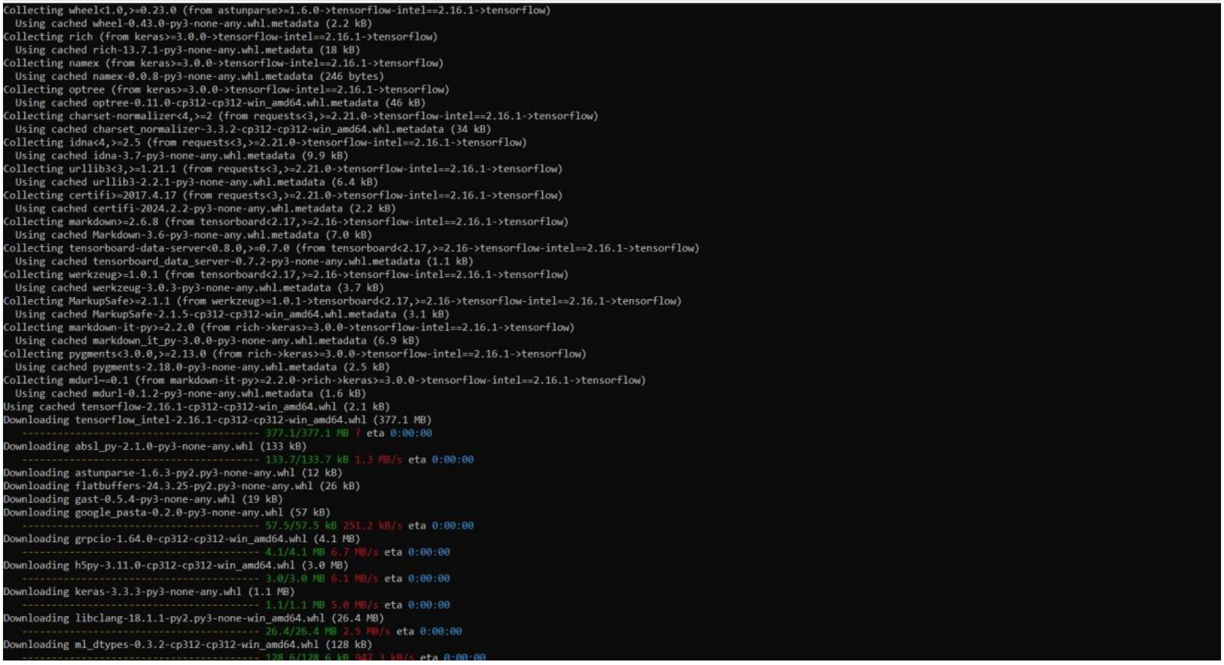


```
C:\Users\User>pip install opencv-python
Defaulting to user installation because normal site-packages is not writeable
Collecting opencv-python
  Downloading opencv_python-4.9.0.80-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\user\appdata\roaming\python\python312\site-packages (from opencv-python) (1.26.4)
Downloading opencv_python-4.9.0.80-cp37-abi3-win_amd64.whl (38.6 MB)
   ---------------------------------------- 38.6/38.6 MB 1.7 MB/s eta 0:00:00
Installing collected packages: opencv-python
Successfully installed opencv-python-4.9.0.80
```

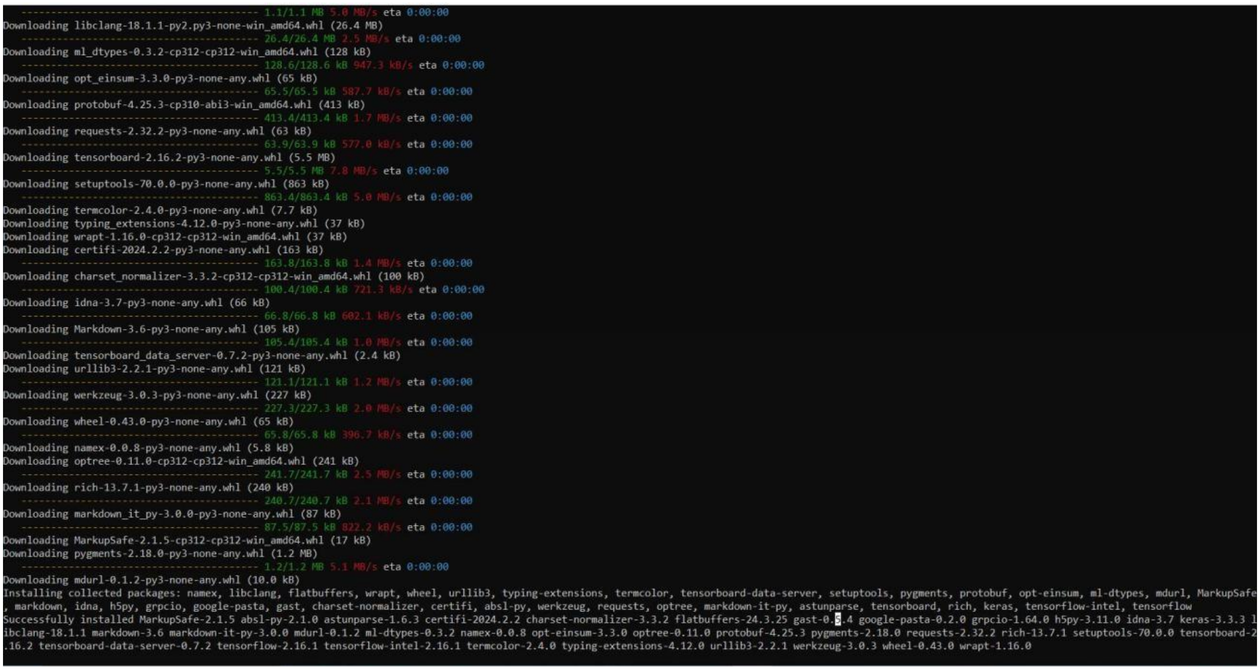
iv. Install tensorflow (cmd : pip install tensorflow)



```
C:\Users\User>pip install tensorflow
Defaulting to user installation because normal site-packages is not writeable
Collecting tensorflow
  Using cached tensorflow-2.16.1-cp312-cp312-win_amd64.whl.metadata (3.5 kB)
Collecting tensorflow-intel==2.16.1 (from tensorflow)
  Using cached tensorflow_intel-2.16.1-cp312-cp312-win_amd64.whl.metadata (5.0 kB)
Collecting absl-py>=1.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached flatbuffers-24.3.25-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached gast-0.5.4-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting h5py>=3.10.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached h5py-3.11.0-cp312-cp312-win_amd64.whl.metadata (2.5 kB)
Collecting libclang>=13.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached libclang-18.1.1-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Collecting ml-dtypes~=0.3.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached ml_dtypes-0.3.2-cp312-cp312-win_amd64.whl.metadata (20 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached opt_einsum-3.3.0-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: packaging in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.16.1->tensorflow) (24.0)
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached protobuf-4.25.3-cp310-abi3-win_amd64.whl.metadata (541 bytes)
Collecting requests<3,>=2.21.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached requests-2.32.2-py3-none-any.whl.metadata (4.6 kB)
Collecting setuptools (from tensorflow-intel==2.16.1->tensorflow)
  Using cached setuptools-70.0.0-py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: six>=1.12.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached termcolor-2.4.0-py3-none-any.whl.metadata (6.1 kB)
Collecting typing-extensions>=3.6.6 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached typing_extensions-4.12.0-py3-none-any.whl.metadata (3.0 kB)
Collecting wrapt>=1.11.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached wrapt-1.16.0-cp312-cp312-win_amd64.whl.metadata (6.8 kB)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached grpcio-1.64.0-cp312-cp312-win_amd64.whl.metadata (3.4 kB)
Collecting tensorboard<2.17,>=2.16 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached tensorboard-2.16.2-py3-none-any.whl.metadata (1.6 kB)
Collecting keras>=3.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached keras-3.3.3-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: numpy<2.0.0,>=1.26.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.26.4)
Collecting wheel<1.0,>=0.23.0 (from astunparse>=1.6.0->tensorflow-intel==2.16.1->tensorflow)
```

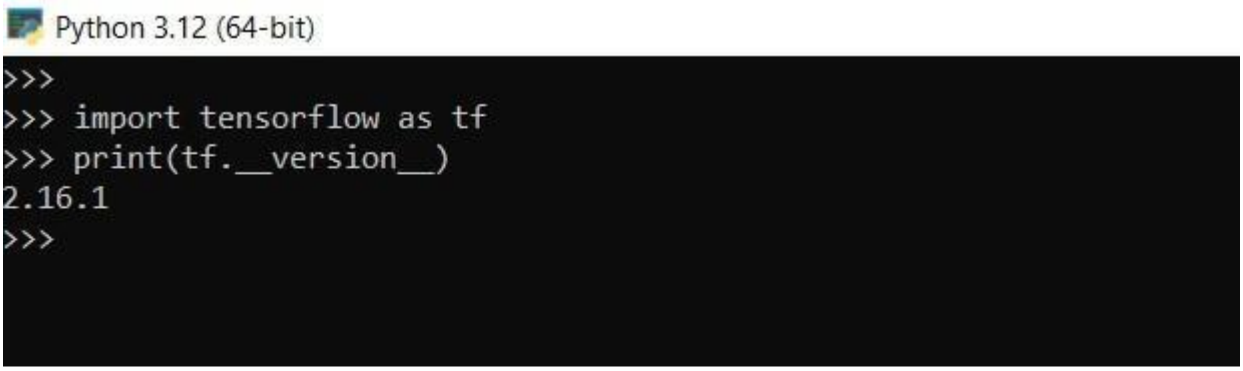


```
Collecting wheel<1.0,>=0.23.0 (from astunparse>=1.6.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached wheel-0.43.0-py3-none-any.whl.metadata (2.2 kB)
Collecting rich (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached rich-13.7.1-py3-none-any.whl.metadata (18 kB)
Collecting namex (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached namex-0.0.8-py3-none-any.whl.metadata (246 bytes)
Collecting optree (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached optree-0.11.0-cp312-cp312-win_amd64.whl.metadata (46 kB)
Collecting charset-normalizer<4,>=2 (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl.metadata (34 kB)
Collecting idna<4,>=2.5 (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Collecting urllib3<3,>=1.21.1 (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached urllib3-2.2.1-py3-none-any.whl.metadata (6.4 kB)
Collecting certifi>=2017.4.17 (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached certifi-2024.2.2-py3-none-any.whl.metadata (2.2 kB)
Collecting markdown>=2.6.8 (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
  Using cached Markdown-3.6-py3-none-any.whl.metadata (7.0 kB)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
  Using cached tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
  Using cached werkzeug-3.0.3-py3-none-any.whl.metadata (3.7 kB)
Collecting MarkupSafe>=2.1.1 (from werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
  Using cached MarkupSafe-2.1.5-cp312-cp312-win_amd64.whl.metadata (3.1 kB)
Collecting markdown-it-py>=2.2.0 (from rich->keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached markdown_it_py-3.0.0-py3-none-any.whl.metadata (6.9 kB)
Collecting pygments<3.0.0,>=2.13.0 (from rich->keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached pygments-2.18.0-py3-none-any.whl.metadata (2.5 kB)
Collecting mdurl~=0.1 (from markdown-it-py>=2.2.0->rich->keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached mdurl-0.1.2-py3-none-any.whl.metadata (1.6 kB)
Using cached tensorflow-2.16.1-cp312-cp312-win_amd64.whl (2.1 kB)
Downloading tensorflow_intel-2.16.1-cp312-cp312-win_amd64.whl (377.1 MB)
   ---------------------------------------- 377.1/377.1 MB ? eta 0:00:00
Downloading absl_py-2.1.0-py3-none-any.whl (133 kB)
   ---------------------------------------- 133.7/133.7 kB 1.3 MB/s eta 0:00:00
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-24.3.25-py2.py3-none-any.whl (26 kB)
Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
   ---------------------------------------- 57.5/57.5 kB 251.2 kB/s eta 0:00:00
Downloading grpcio-1.64.0-cp312-cp312-win_amd64.whl (4.1 MB)
   ---------------------------------------- 4.1/4.1 MB 6.7 MB/s eta 0:00:00
Downloading h5py-3.11.0-cp312-cp312-win_amd64.whl (3.0 MB)
   ---------------------------------------- 3.0/3.0 MB 6.1 MB/s eta 0:00:00
Downloading keras-3.3.3-py3-none-any.whl (1.1 MB)
   ---------------------------------------- 1.1/1.1 MB 5.0 MB/s eta 0:00:00
Downloading libclang-18.1.1-py2.py3-none-win_amd64.whl (26.4 MB)
   ---------------------------------------- 26.4/26.4 MB 2.5 MB/s eta 0:00:00
Downloading ml_dtypes-0.3.2-cp312-cp312-win_amd64.whl (128 kB)
   ---------------------------------------- 128.6/128.6 kB 947.3 kB/s eta 0:00:00
```



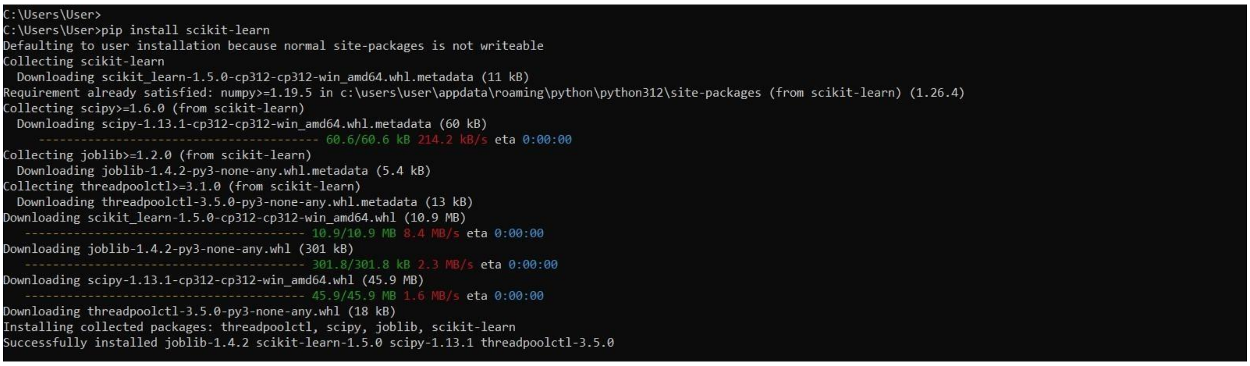
```
   ---------------------------------------- 1.1/1.1 MB 5.0 MB/s eta 0:00:00
Downloading libclang-18.1.1-py2.py3-none-win_amd64.whl (26.4 MB)
   ---------------------------------------- 26.4/26.4 MB 2.5 MB/s eta 0:00:00
Downloading ml_dtypes-0.3.2-cp312-cp312-win_amd64.whl (128 kB)
   ---------------------------------------- 128.6/128.6 kB 947.3 kB/s eta 0:00:00
Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
   ---------------------------------------- 65.5/65.5 kB 587.7 kB/s eta 0:00:00
Downloading protobuf-4.25.3-cp310-abi3-win_amd64.whl (413 kB)
   ---------------------------------------- 413.4/413.4 kB 1.7 MB/s eta 0:00:00
Downloading requests-2.32.2-py3-none-any.whl (63 kB)
   ---------------------------------------- 63.9/63.9 kB 577.0 kB/s eta 0:00:00
Downloading tensorboard-2.16.2-py3-none-any.whl (5.5 MB)
   ---------------------------------------- 5.5/5.5 MB 7.8 MB/s eta 0:00:00
Downloading setuptools-70.0.0-py3-none-any.whl (863 kB)
   ---------------------------------------- 863.4/863.4 kB 5.0 MB/s eta 0:00:00
Downloading termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Downloading typing_extensions-4.12.0-py3-none-any.whl (37 kB)
Downloading wrapt-1.16.0-cp312-cp312-win_amd64.whl (37 kB)
Downloading certifi-2024.2.2-py3-none-any.whl (163 kB)
   ---------------------------------------- 163.8/163.8 kB 1.4 MB/s eta 0:00:00
Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl (100 kB)
   ---------------------------------------- 100.4/100.4 kB 721.3 kB/s eta 0:00:00
Downloading idna-3.7-py3-none-any.whl (66 kB)
   ---------------------------------------- 66.8/66.8 kB 602.1 kB/s eta 0:00:00
Downloading Markdown-3.6-py3-none-any.whl (105 kB)
   ---------------------------------------- 105.4/105.4 kB 1.0 MB/s eta 0:00:00
Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Downloading urllib3-2.2.1-py3-none-any.whl (121 kB)
   ---------------------------------------- 121.1/121.1 kB 1.2 MB/s eta 0:00:00
Downloading werkzeug-3.0.3-py3-none-any.whl (227 kB)
   ---------------------------------------- 227.3/227.3 kB 2.0 MB/s eta 0:00:00
Downloading wheel-0.43.0-py3-none-any.whl (65 kB)
   ---------------------------------------- 65.8/65.8 kB 396.7 kB/s eta 0:00:00
Downloading namex-0.0.8-py3-none-any.whl (5.8 kB)
Downloading optree-0.11.0-cp312-cp312-win_amd64.whl (241 kB)
   ---------------------------------------- 241.7/241.7 kB 2.5 MB/s eta 0:00:00
Downloading rich-13.7.1-py3-none-any.whl (240 kB)
   ---------------------------------------- 240.7/240.7 kB 2.1 MB/s eta 0:00:00
Downloading markdown_it_py-3.0.0-py3-none-any.whl (87 kB)
   ---------------------------------------- 87.5/87.5 kB 822.2 kB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp312-cp312-win_amd64.whl (17 kB)
Downloading pygments-2.18.0-py3-none-any.whl (1.2 MB)
   ---------------------------------------- 1.2/1.2 MB 5.1 MB/s eta 0:00:00
Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: namex, libclang, flatbuffers, wrapt, wheel, urllib3, typing-extensions, termcolor, tensorboard-data-server, setuptools, pygments, protobuf, opt-einsum, ml-dtypes, mdurl, MarkupSafe
, markdown, idna, h5py, grpcio, google-pasta, gast, charset-normalizer, certifi, absl-py, werkzeug, requests, optree, markdown-it-py, astunparse, tensorboard, rich, keras, tensorflow-intel, tensorflow
Successfully installed MarkupSafe-2.1.5 absl-py-2.1.0 astunparse-1.6.3 certifi-2024.2.2 charset-normalizer-3.3.2 flatbuffers-24.3.25 gast-0.5.4 google-pasta-0.2.0 grpcio-1.64.0 h5py-3.11.0 idna-3.7 keras-3.3.3 l
ibclang-18.1.1 markdown-3.6 markdown-it-py-3.0.0 mdurl-0.1.2 ml-dtypes-0.3.2 namex-0.0.8 opt-einsum-3.3.0 optree-0.11.0 protobuf-4.25.3 pygments-2.18.0 requests-2.32.2 rich-13.7.1 setuptools-70.0.0 tensorboard-2
.16.2 tensorboard-data-server-0.7.2 tensorflow-2.16.1 tensorflow-intel-2.16.1 termcolor-2.4.0 typing-extensions-4.12.0 urllib3-2.2.1 werkzeug-3.0.3 wheel-0.43.0 wrapt-1.16.0
```

For checking Version of tensorflow



```
>>>
>>> import tensorflow as tf
>>> print(tf.__version__)
2.16.1
>>>
```

v. Install scikit-learn (cmd : pip install scikit-learn)



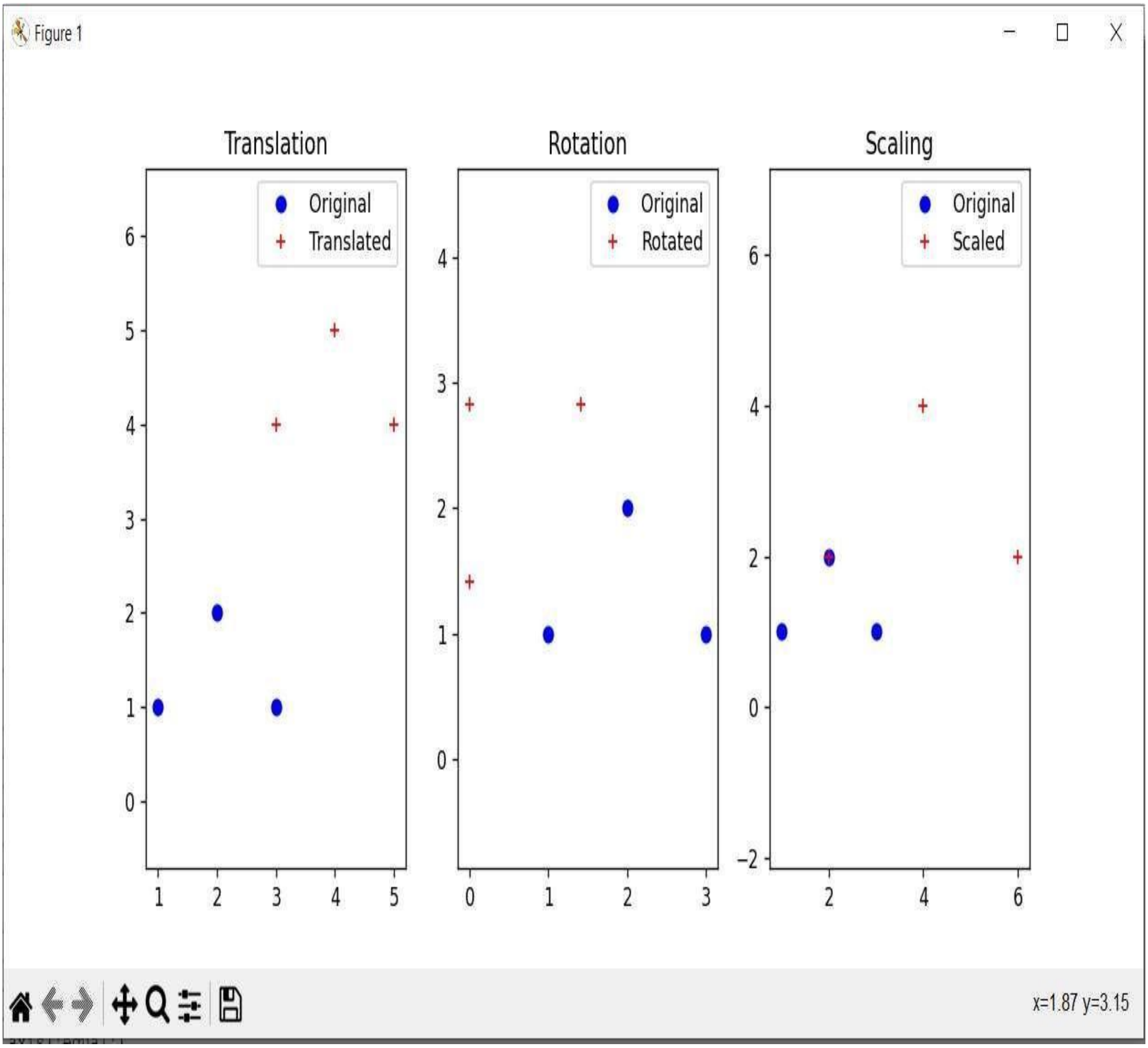
```
C:\Users\User>
C:\Users\User>pip install scikit-learn
Defaulting to user installation because normal site-packages is not writeable
Collecting scikit-learn
  Downloading scikit_learn-1.5.0-cp312-cp312-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: numpy>=1.19.5 in c:\users\user\appdata\roaming\python\python312\site-packages (from scikit-learn) (1.26.4)
Collecting scipy>=1.6.0 (from scikit-learn)
  Downloading scipy-1.13.1-cp312-cp312-win_amd64.whl.metadata (60 kB)
     ---------------------------------------- 60.6/60.6 kB 214.2 kB/s eta 0:00:00
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.5.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.5.0-cp312-cp312-win_amd64.whl (10.9 MB)
   ---------------------------------------- 10.9/10.9 MB 8.4 MB/s eta 0:00:00
Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
   ---------------------------------------- 301.8/301.8 kB 2.3 MB/s eta 0:00:00
Downloading scipy-1.13.1-cp312-cp312-win_amd64.whl (45.9 MB)
   ---------------------------------------- 45.9/45.9 MB 1.6 MB/s eta 0:00:00
Downloading threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.4.2 scikit-learn-1.5.0 scipy-1.13.1 threadpoolctl-3.5.0
```

Practical 1 Perform geometric transformation using python

```
import numpy as np import
matplotlib.pyplot as plt
import matplotlib.transforms as transforms
# Original points points = np.array([[1,
1], [2, 2], [3, 1]])
# Translation translation_matrix = np.array([[1, 0, 2], [0, 1, 3], [0, 0, 1]]) # Translation by (2, 3)
translated_points = np.dot(translation_matrix, np.hstack([points, np.ones((points.shape[0],
1))]).T).T[:, :2] # Rotation theta = np.pi / 4 # Rotation angle (45 degrees) rotation_matrix =
np.array([[np.cos(theta), -np.sin(theta), 0], [np.sin(theta), np.cos(theta), 0], [0, 0,
1]]) rotated_points = np.dot(rotation_matrix, np.hstack([points, np.ones((points.shape[0],
1))]).T).T[:, :2]
# Scaling scaling_matrix = np.array([[2, 0, 0], [0, 2, 0], [0, 0, 1]]) # Scaling by a factor of 2
scaled_points = np.dot(scaling_matrix, np.hstack([points, np.ones((points.shape[0], 1))]).T).T[:, :2]
# Plotting plt.figure(figsize=(10, 5)) plt.subplot(1, 3, 1) plt.title('Translation')
plt.plot(points[:, 0], points[:, 1], 'bo', label='Original')
plt.plot(translated_points[:, 0], translated_points[:, 1], 'r+', label='Translated')
plt.axis('equal') plt.legend() plt.subplot(1, 3, 2) plt.title('Rotation')
plt.plot(points[:, 0], points[:, 1], 'bo', label='Original')
plt.plot(rotated_points[:, 0], rotated_points[:, 1], 'r+', label='Rotated')
plt.axis('equal') plt.legend() plt.subplot(1, 3, 3) plt.title('Scaling')
plt.plot(points[:, 0], points[:, 1], 'bo', label='Original') plt.plot(scaled_points[:,
0], scaled_points[:, 1], 'r+', label='Scaled') plt.axis('equal') plt.legend()
plt.show()
```

Output:

Code:



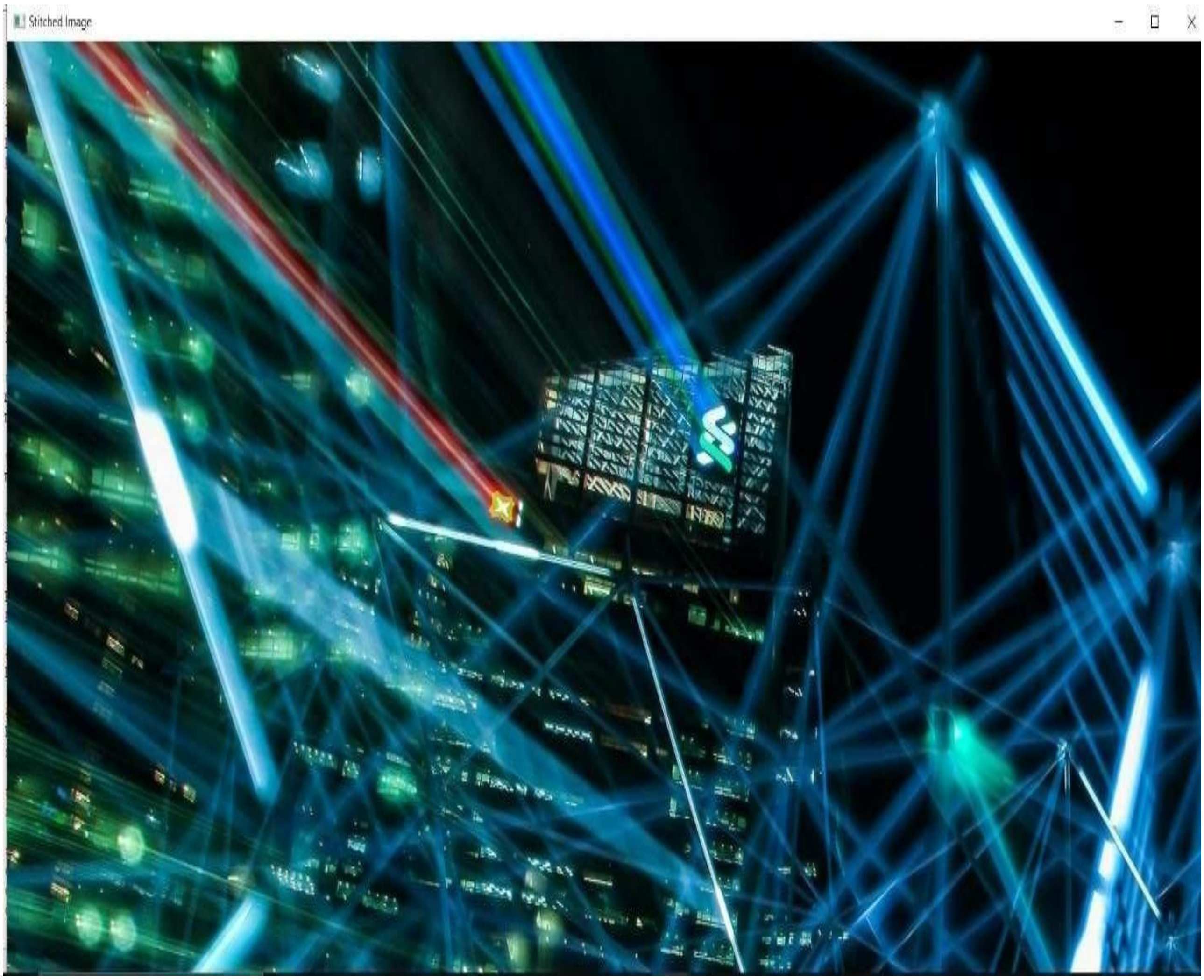
Practical No.2
Perform Image Stitching.

```
import cv2 import numpy as np # Load images image2 = cv2.imread("D:\Abhishek
CV\ComputerVisionPractical\imageFolder\pex1.jpg") image1 = cv2.imread("D:\Abhishek
CV\ComputerVisionPractical\imageFolder\pex.jpg") print("Image 1 shape:",
image1.shape) print("Image 2 shape:", image2.shape) # Convert images to grayscale gray1
= cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY) gray2 = cv2.cvtColor(image2,
cv2.COLOR_BGR2GRAY)
```

```
# Detect keypoints and compute descriptors sift =
cv2.SIFT_create() keypoints1, descriptors1 =
sift.detectAndCompute(gray1, None) keypoints2, descriptors2 =
sift.detectAndCompute(gray2, None)
# Match descriptors between the two images matcher
= cv2.BFMatcher() matches =
matcher.match(descriptors1, descriptors2)
# Sort matches by distance matches = sorted(matches,
key=lambda x: x.distance)
# Extract matched keypoints points1 = np.float32([keypoints1[match.queryIdx].pt for match
in matches]).reshape(-1, 1, 2) print("Number of points in points1:", len(points1)) points2 =
np.float32([keypoints2[match.trainIdx].pt for match in matches]).reshape(-1, 1, 2)
print("Number of points in points2:", len(points2))
# Find homography matrix homography, _ =
cv2.findHomography(points1, points2, cv2.RANSAC)
# Warp image1 to align with image2 height, width = gray2.shape
stitched_image = cv2.warpPerspective(image1, homography, (width, height))
# Combine the stitched image with image2 stitched_image[0:image2.shape[0],
0:image2.shape[1]] = image2
# Display the stitched image
cv2.imshow('Stitched Image',
stitched_image) cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:

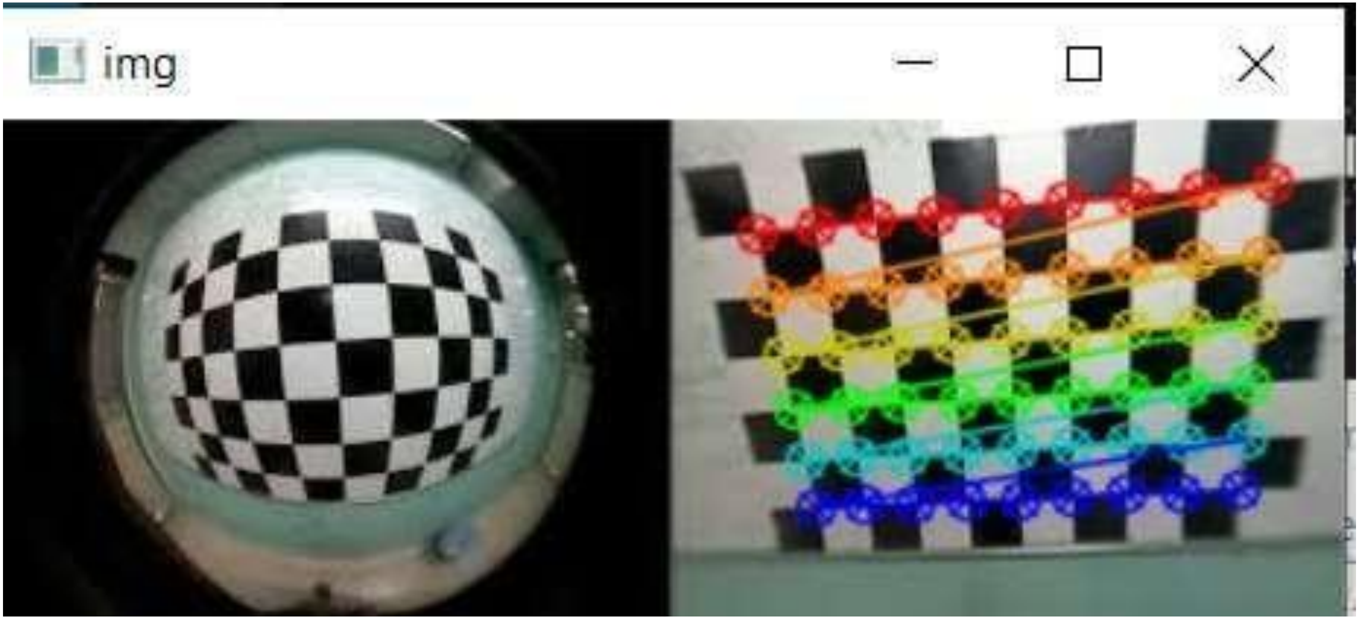
Code:



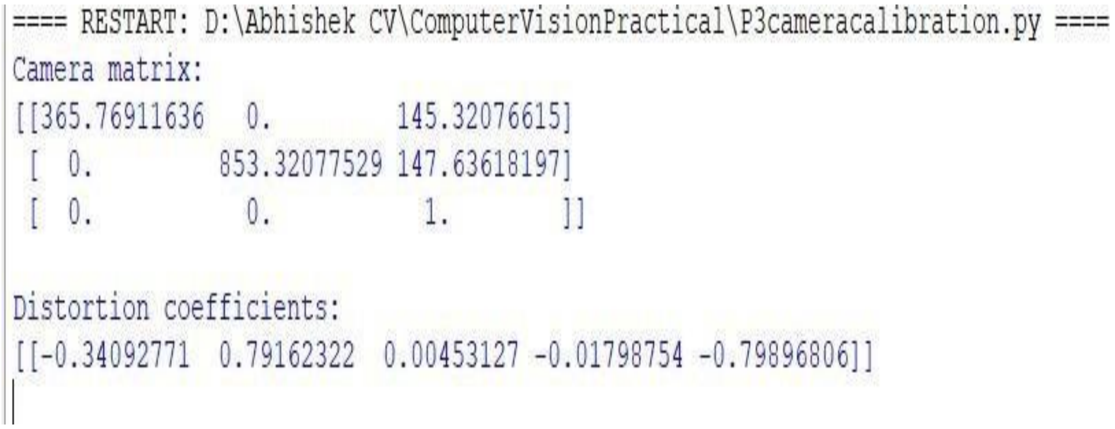
Practical No. 3 Camera Calibration

```
import numpy as np
import  cv2  import
glob
# Define the number of corners in the chessboard
num_corners_x = 9
num_corners_y = 6
# Prepare object points, like (0,0,0), (1,0,0), (2,0,0) .... ,(6,5,0) objp =
np.zeros((num_corners_x * num_corners_y, 3), np.float32) objp[:, :2] =
np.mgrid[0:num_corners_x, 0:num_corners_y].T.reshape(-1, 2) # Arrays to
store object points and image points from all the images objpoints = [] # 3d
point in real world space imgpoints = [] # 2d points in image plane.
```

```
# Load images images = glob.glob("D:\Abhishek
CV\calibration_images/*.jpg")
# Loop through images and find chessboard corners for
fname in images:
   img = cv2.imread(fname) gray = cv2.cvtColor(img,
   cv2.COLOR_BGR2GRAY)
   # Find the chessboard corners ret, corners = cv2.findChessboardCorners(gray,
   (num_corners_x, num_corners_y), None)
   # If found, add object points, image points (after refining them) if
   ret:
      objpoints.append(objp) corners2 = cv2.cornerSubPix(gray,
      corners, (11, 11), (-1, -1),
criteria=(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001))
      imgpoints.append(corners2) # Draw and display the corners img =
      cv2.drawChessboardCorners(img, (num_corners_x, num_corners_y), corners2, ret)
      cv2.imshow('img', img) cv2.waitKey(5000)
cv2.destroyAllWindows() #
Perform camera calibration
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)
# Save calibration results np.savez('calibration.npz', mtx=mtx, dist=dist,
rvecs=rvecs, tvecs=tvecs)
# Print calibration results
print("Camera matrix:")
print(mtx) print("\nDistortion
coefficients:") print(dist) Output:
```



Code:

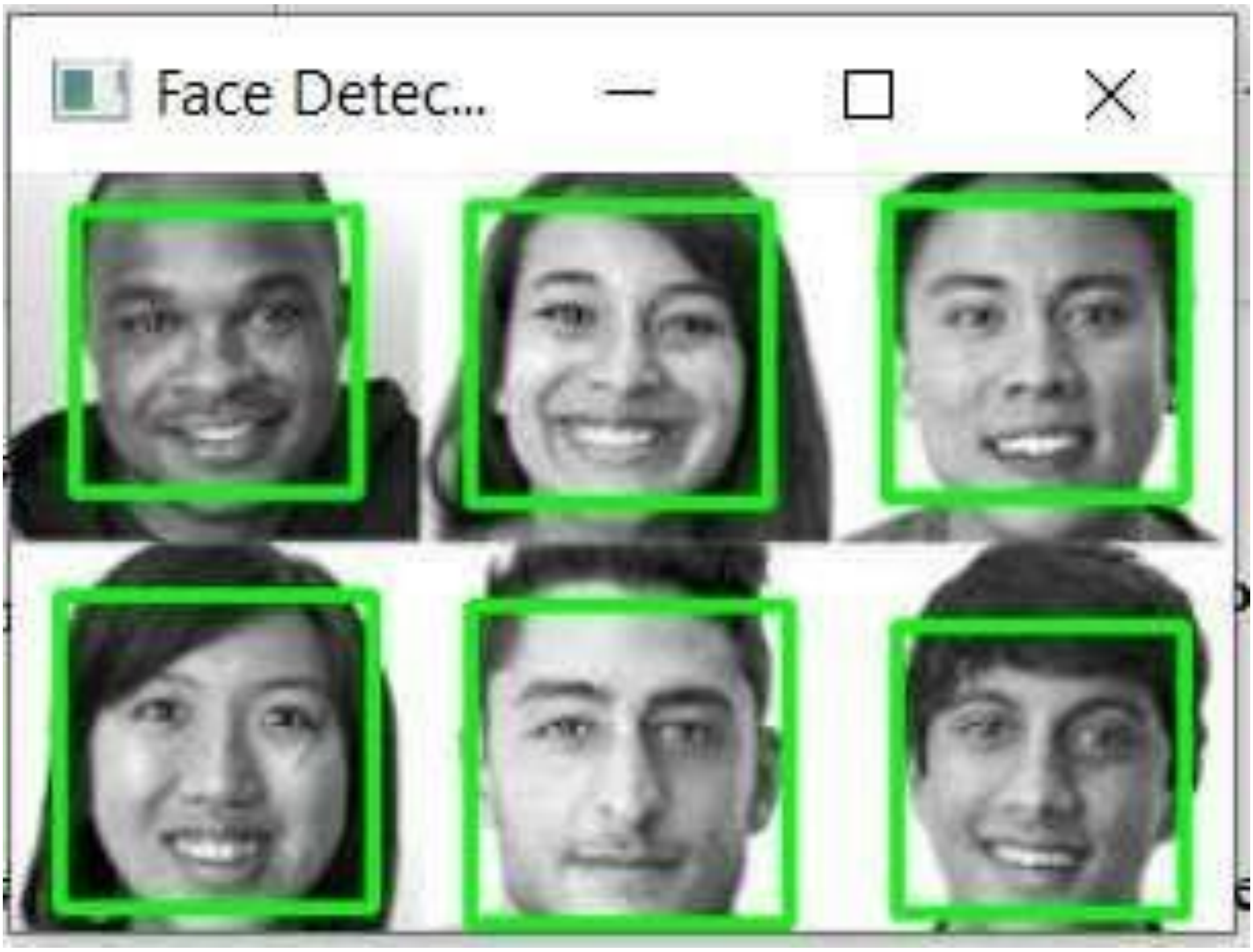


```
==== RESTART: D:\Abhishek CV\ComputerVisionPractical\P3cameracalibration.py ====
Camera matrix:
[[365.76911636   0.         145.32076615]
 [  0.         853.32077529 147.63618197]
 [  0.           0.           1.        ]]

Distortion coefficients:
[[-0.34092771  0.79162322  0.00453127 -0.01798754 -0.79896806]]
```

Practical No.4

Face detection Code:

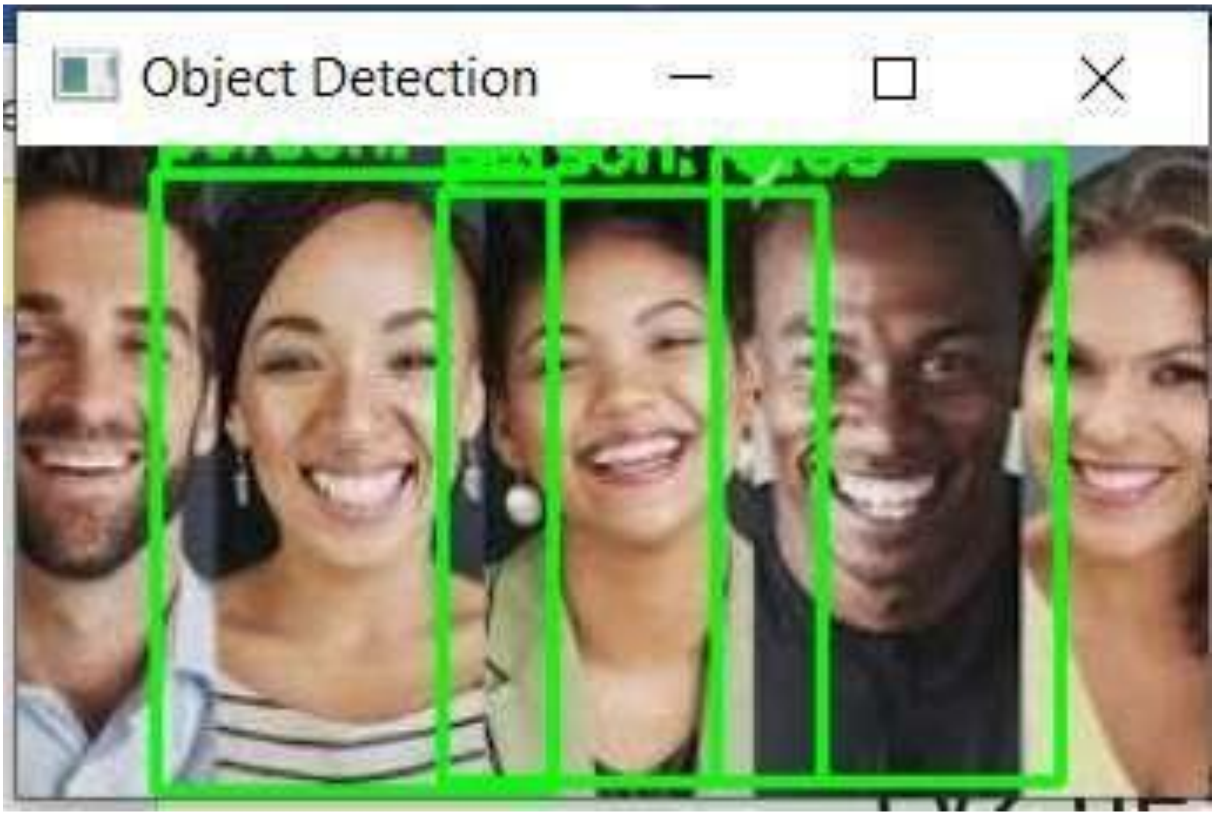
```
import cv2
# Load the pre-trained Haar Cascade face detector
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
# Load the image image = cv2.imread("D:\Abhishek
CV\ComputerVisionPractical\imageFolder\IMG1.jpg")
# Convert the image to grayscale (face detection works on grayscale images) gray
= cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Detect faces in the image faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))
# Draw rectangles around the faces for
(x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
# Display the result
cv2.imshow('Face Detection',
image) cv2.waitKey(0)
cv2.destroyAllWindows() Output:
```



Practical No.5
Object Detection

```
Code:   import   numpy
as np import os import
tensorflow as tf import
cv2

# Load the pre-trained model
MODEL_NAME = 'ssd_mobilenet_v2_coco_2018_03_29'
PATH_TO_CKPT          =          os.path.join(MODEL_NAME,          'frozen_inference_graph.pb')
NUM_CLASSES = 90

detection_graph = tf.Graph() with
detection_graph.as_default():
    od_graph_def    =    tf.compat.v1.GraphDef()    with
    tf.io.gfile.GFile(PATH_TO_CKPT,    'rb')    as    fid:
    serialized_graph                =                fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

# Load label map
```

```
#PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
PATH_TO_LABELS = "D:\Abhishek
CV\ComputerVisionPractical\imageFolder\mscoco_label_map (2).pbtxt"
category_index = {} with open(PATH_TO_LABELS, 'r') as f: lines =
f.readlines() for line in lines:
        if 'id:' in line:
            id_index = int(line.strip().split(':')[1])
        if 'display_name:' in line:
            name = line.strip().split(':')[1].strip().strip('"') category_index[id_index]
            = {'name': name}

# Function to perform object detection def
detect_objects(image):
    with detection_graph.as_default():
        with tf.compat.v1.Session(graph=detection_graph) as sess:
            # Expand dimensions since the model expects images to have shape: [1, None, None,
            3] image_expanded = np.expand_dims(image, axis=0) image_tensor =
            detection_graph.get_tensor_by_name('image_tensor:0')
            # Each box represents a part of the image where a particular object was detected.
            boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
            # Each score represents the level of confidence for each of the objects. #
            The score is shown on the result image, together with the class label.
            scores = detection_graph.get_tensor_by_name('detection_scores:0')
            classes = detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections =
            detection_graph.get_tensor_by_name('num_detections:0') # Actual
            detection.
            (boxes, scores, classes, num_detections) = sess.run([boxes, scores, classes,
num_detections],feed_dict={image_tensor: image_expanded}) # Visualization of
the results of a detection.
            for i in range(len(scores[0])):
                if scores[0][i] > 0.5: # Adjust confidence threshold as needed
                    class_id = int(classes[0][i]) class_name =
                    category_index[class_id]['name'] score =
                    float(scores[0][i])
                    ymin, xmin, ymax, xmax = boxes[0][i]
                    (left, right, top, bottom) = (xmin * image.shape[1], xmax * image.shape[1], ymin *
image.shape[0], ymax * image.shape[0]) cv2.rectangle(image, (int(left), int(top)), (int(right),
                    int(bottom)), (0, 255, 0), 2) cv2.putText(image, '{}: {:.2f}'.format(class_name,
                    score), (int(left), int(top - 5)),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2) return
    image

# Object detection on an image input_image = cv2.imread("D:\Abhishek
CV\ComputerVisionPractical\imageFolder\IMG.jpg") output_image =
detect_objects(input_image) cv2.imshow('Object Detection', output_image) cv2.waitKey(0)
cv2.destroyAllWindows() Output:
```



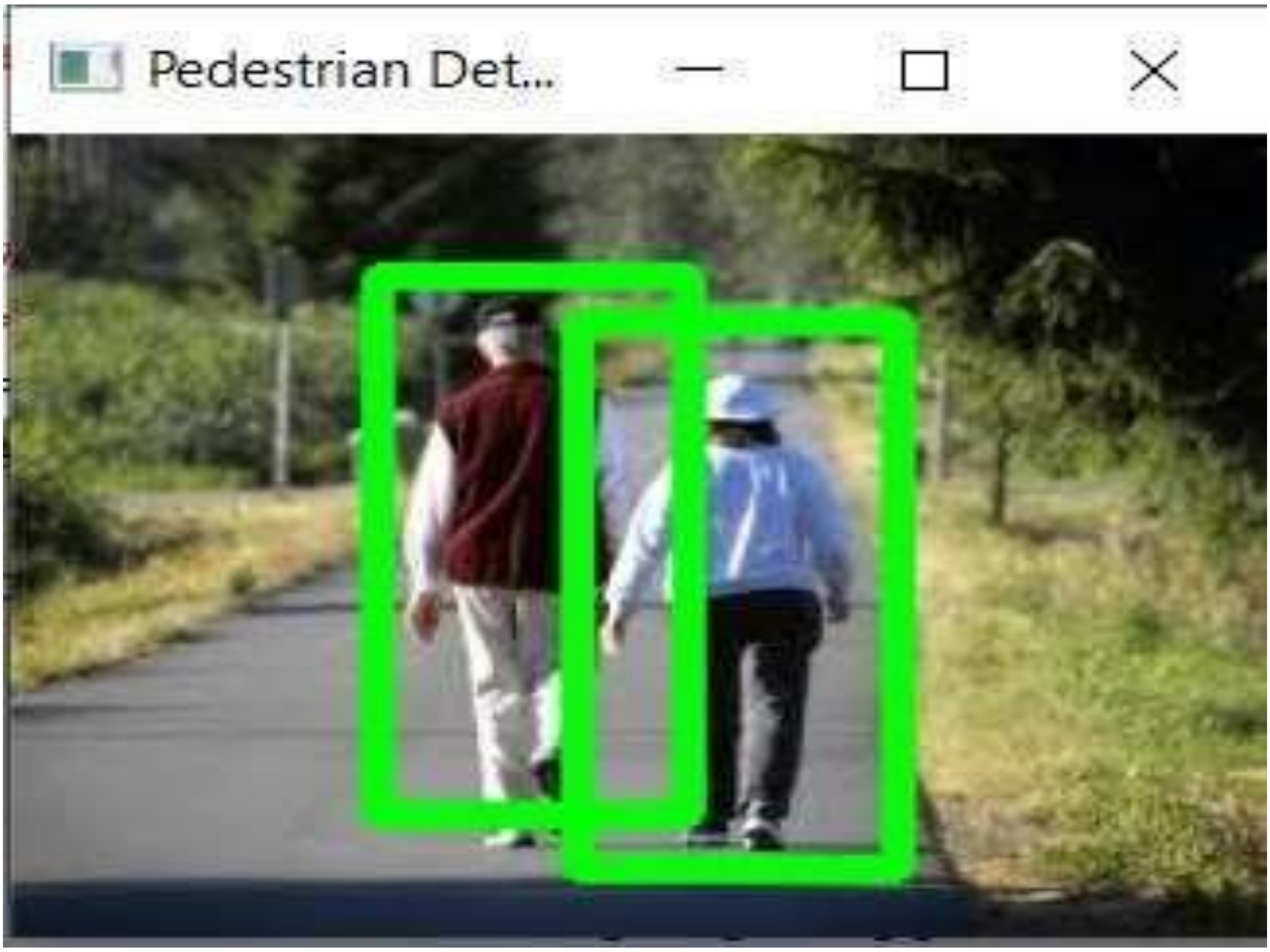
Practical No.6

Pedestrian detection

Code:

```
import cv2
# Load the pre-trained pedestrian detector pedestrian_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_fullbody.xml')
# Load the input image image = cv2.imread("D:\Abhishek
CV\ComputerVisionPractical\imageFolder\pedestrainimg.jpg")
# Convert the image to grayscale gray =
cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Detect pedestrians in the image pedestrians = pedestrian_cascade.detectMultiScale(gray,
scaleFactor=1.1, minNeighbors=1, minSize=(5, 5))
# Draw rectangles around the detected pedestrians for
(x, y, w, h) in pedestrians:
   cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 5)
# Display the image with pedestrian detections
cv2.imshow('Pedestrian Detection', image)
cv2.waitKey(0) cv2.destroyAllWindows()
```

Output:

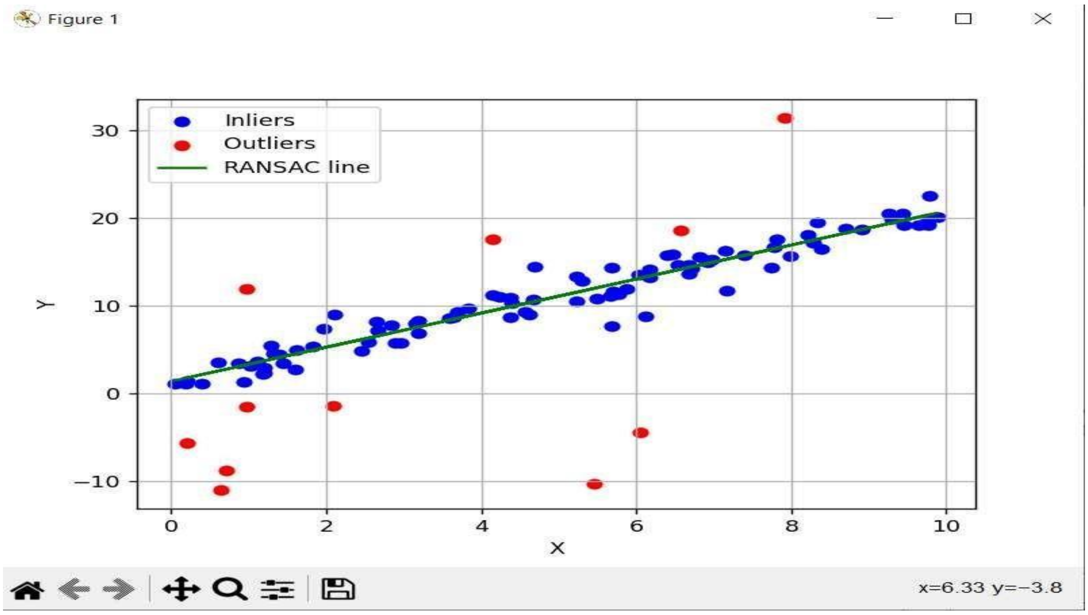


Practical No.7 Feature extraction using RANSAC Code:

```
import numpy as np from sklearn.linear_model import
RANSACRegressor import matplotlib.pyplot as plt #
Generate some noisy data points np.random.seed(0) x
= np.random.uniform(0, 10, 100) y = 2 * x + 1 +
np.random.normal(0, 1, 100)
# Add outliers outliers_index = np.random.choice(100, 20,
replace=False) y[outliers_index] += 10 *
np.random.normal(0, 1, 20)
# Stack the points for RANSAC data
= np.vstack((x, y)).T
# Initialize RANSAC model (using sklearn) ransac
= RANSACRegressor()
# Fit the model ransac.fit(data[:, 0].reshape(-
1, 1), data[:, 1])
# Extract the inliers and outliers inlier_mask
= ransac.inlier_mask_

outlier_mask = np.logical_not(inlier_mask)
```

```
# Extract the line parameters line_slope =
ransac.estimator_.coef_[0] line_intercept =
ransac.estimator_.intercept_
# Plot the data points plt.scatter(data[inlier_mask][:, 0], data[inlier_mask][:, 1],
c='b', label='Inliers') plt.scatter(data[outlier_mask][:, 0], data[outlier_mask][:, 1],
c='r', label='Outliers')
# Plot the fitted line plt.plot(x, line_slope * x + line_intercept, color='g',
label='RANSAC line') plt.xlabel('X') plt.ylabel('Y') plt.legend()
plt.grid(True)
plt.show()
```

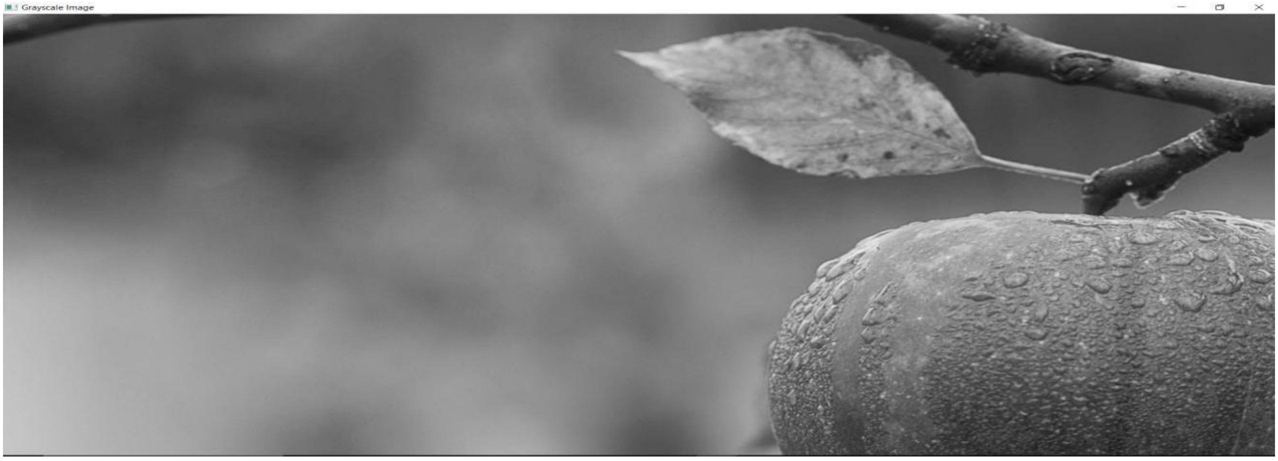
Output:

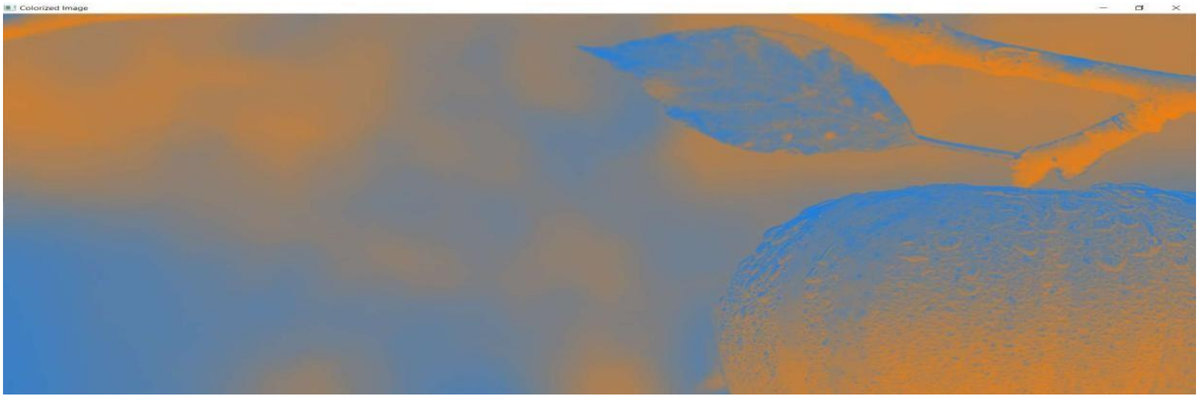


Practical No.8

Colorization of an image Code:

```
import cv2 import numpy as np # Load the grayscale image gray_image =
cv2.imread("D:\Abhishek CV\ComputerVisionPractical\imageFolder\grayimg.jpeg",
cv2.IMREAD_GRAYSCALE)
# Convert grayscale image to BGR (3-channel) for colorization color_image
= cv2.cvtColor(gray_image, cv2.COLOR_GRAY2BGR)
# Define a basic colorization lookup table
color_lookup_table = np.zeros((256, 1, 3), dtype=np.uint8)
for i in range(256):
  color_lookup_table[i, 0, 0] = i # Blue channel
color_lookup_table[i, 0, 1] = 127 # Green channel
color_lookup_table[i, 0, 2] = 255 - i # Red channel # Apply the
colorization lookup table to the grayscale image
colorized_image = cv2.LUT(color_image, color_lookup_table)
# Display the original grayscale image and the colorized image
cv2.imshow('Grayscale Image', gray_image)
cv2.imshow('Colorized Image', colorized_image)
cv2.waitKey(0) cv2.destroyAllWindows() Output:
```



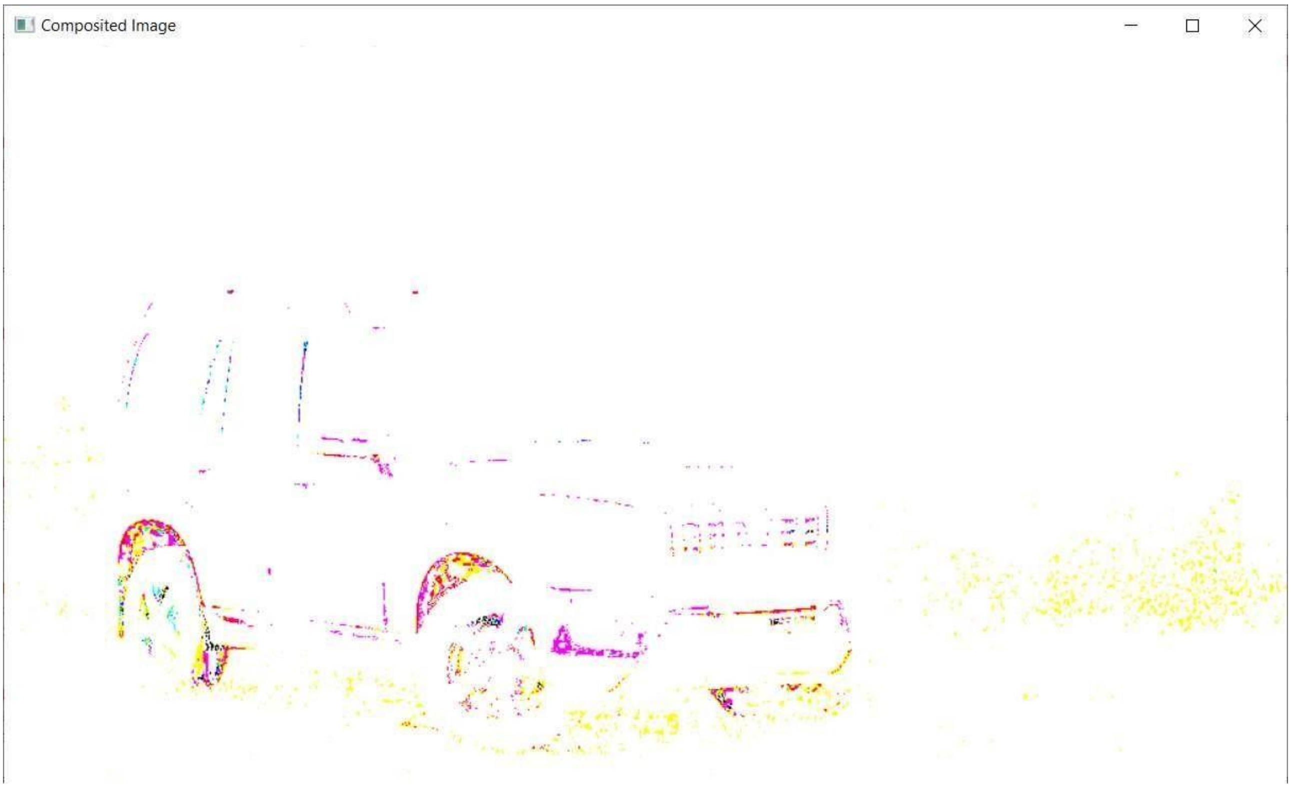


Practical No.9 Image matting and composting Image composting:

Code:

```
import cv2 import numpy as np
def estimate_alpha(image,
trimap):
   # Placeholder function, replace with your matting algorithm implementation
   # This example simply sets alpha values based on trimap (e.g., foreground = 1, background = 0,
unknown = interpolated) alpha = np.zeros_like(trimap, dtype=np.float32) alpha[trimap == 255] =
1.0 # Foreground alpha[trimap == 0] = 0.0 # Background alpha[(trimap > 0) & (trimap < 255)] =
0.5 # Interpolated return alpha

def image_matting(image, trimap): #
   Convert image and trimap to float32
   image = image.astype(np.float32) / 255.0
   trimap = trimap.astype(np.float32) / 255.0

   # Estimate alpha matte using a matting algorithm #
   Replace this with your desired matting algorithm
   alpha = estimate_alpha(image, trimap)

   # Clip alpha values to [0, 1] alpha
   = np.clip(alpha, 0, 1)
```

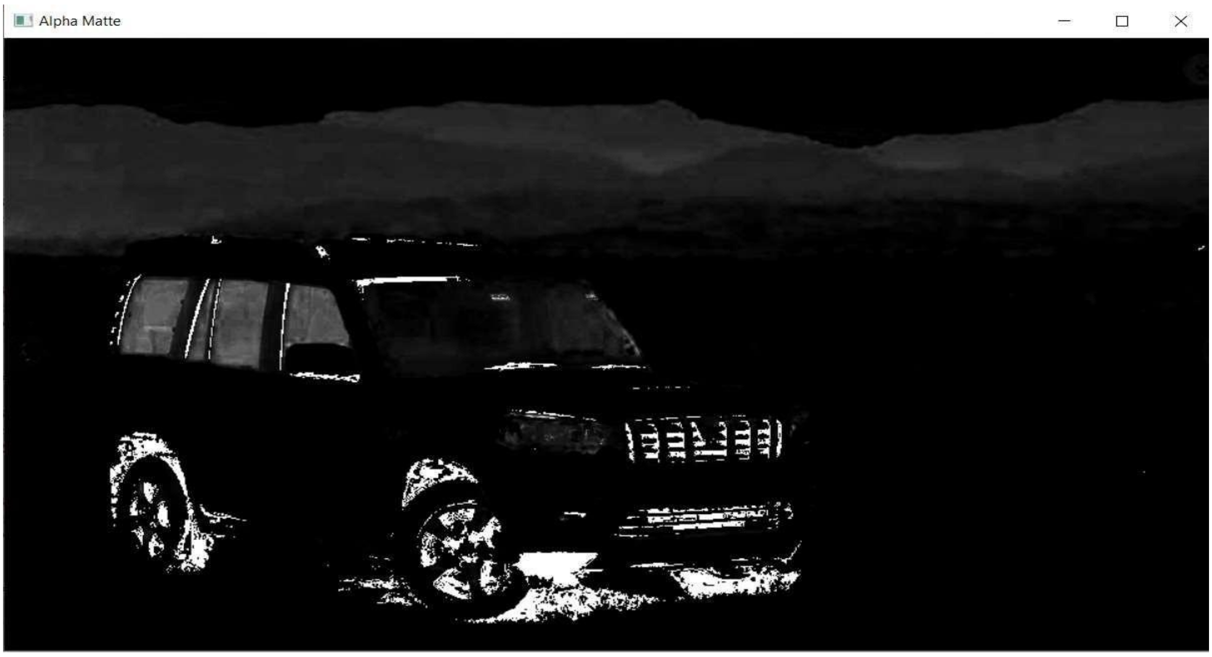
```
    return alpha
def composit_foreground_background(foreground, background, alpha):
    # Resize background to match the foreground size background =
    cv2.resize(background, (foreground.shape[1], foreground.shape[0]))

    # Convert alpha to 3 channels alpha =
    np.stack((alpha, alpha, alpha), axis=2)

    # Composite foreground and background using alpha matte

    composited_image = alpha * foreground + (1 - alpha) * background

    return composited_image

# Example usage if __name
____== "_main_":
    # Read foreground, background, and trimap images foreground = cv2.imread("D:\Abhishek
    CV\ComputerVisionPractical\imageFolder\model.jpg") background = cv2.imread("D:\Abhishek
    CV\ComputerVisionPractical\imageFolder\model.jpg") trimap = cv2.imread("D:\Abhishek
    CV\ComputerVisionPractical\imageFolder\model.jpg", cv2.IMREAD_GRAYSCALE)

    # Perform image matting alpha =
    image_matting(foreground, trimap)

    # Perform compositing composited_image =
    composit_foreground_background(foreground, background, alpha)

    # Display result cv2.imshow("Composited Image",
composited_image) cv2.waitKey(0)
cv2.destroyAllWindows() Output:
```



OR Image Matting

Code:

```
import cv2 import numpy as np def
estimate_alpha(image, trimap):
   # Convert to float image =
   image.astype(np.float32) / 255.0
   # Normalize trimap to [0, 1] trimap = trimap.astype(np.float32) / 255.0 # Compute alpha
   matte using Closed-Form matting foreground = np.where(trimap > 0.95, 1.0, 0.0) #
   Foreground mask alpha = np.where(trimap > 0.05, 1.0, 0.0) # Alpha initialization for _ in
   range(5): # Iterative refinement alpha = (image[:, :, 0] - image[:, :, 2] * alpha) / (1e-12 +
   foreground + (1.0 - trimap) * alpha) alpha = np.clip(alpha, 0, 1)
   return alpha
# Example usage
  if _name_____== "_main_":
  # Read image and trimap image = cv2.imread("D:\Abhishek
  CV\ComputerVisionPractical\imageFolder\model.jpg")
   trimap = cv2.imread("D:\Abhishek CV\ComputerVisionPractical\imageFolder\model.jpg",
cv2.IMREAD_GRAYSCALE)
   # Estimate alpha matte alpha =
estimate_alpha(image, trimap) # Save or
display alpha matte cv2.imshow("Alpha
Matte", alpha) cv2.waitKey(0)
cv2.destroyAllWindows() Output:
```


Alpha Matte