PERIYAR UNIVERSITY

(NAAC 'A++' Grade with CGPA 3.61 (Cycle - 3)
State University - NIRF Rank 56 - State Public University Rank 25
SALEM - 636 011, Tamil Nadu, India.

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

MASTER OF COMPUTER APPLICATIONS
SEMESTER – II

Elective – II: COMPUTER VISION LAB
(Candidates admitted from 2024 onwards)

PERIYAR UNIVERSITY

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)
MCA 2024 admission onwards

Elective Course – II LAB
COMPUTER VISION LAB

Prepared by:
Centre for Distance and Online Education (CDOE)
Periyar University
Salem – 636011.

SYLLABUS

COMPUTER VISION LAB

COURSE OBJECTIVES:

- To get an idea of how to build a computer vision application with Python language.
- To learn the basic image handling and processing
- To get familiar with various Computer Vision fundamental algorithms and how to implement and apply.
- To get an idea of how to implement the image transforms.
- To understand various image segmentation algorithms.

IMPLEMENT THE FOLLOWING PROBLEMS USING PYTHON WITH OPENCV

LIST OF EXPERIMENTS

1. Image Loading, Exploring and displaying an Image.
2. Access and Manipulate of Image Pixels.
3. Image
   Transformations.
   i) Resizing
   ii) Rotation
4. Addition operation of Two Images.
5. Image filtering operations
   i) Mean Filtering
   ii) Gaussian Filtering
6. Image Binarization Using Simple Thresholding method.
7. Edge Detection operation using Sobel and Scharr Gradients.
8. Find Grayscale and RGB Histograms of an Image.
9. Segment an Image using K-means Clustering algorithm.
10. Write a program to classify an Image using KNN Classification algorithm.

COURSE OUTCOMES:

On the successful completion of the course, students will be able to:

CO1	To develop and implement the image loading and exploring	K1-K6
CO2	To Evaluate the image transforms	
CO3	To apply and analyze for image processing denoising algorithms	
CO4	To design and develop the Image Segmentation using Edge detection	
CO5	To apply and analyze image clustering and classification algorithms	

K1- Remember, K2 - Understand, K3 - Apply , K4 - Analyze, K5 - Evaluate, K6 -Create

MAPPING WITH PROGRAMME OUTCOMES:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10
CO1	H	L	M	L	L	L	M	M	M	H
CO2	H	M	L	M	M	L	H	L	H	L
CO3	H	H	H	M	M	L	M	L	M	L
CO4	H	H	H	M	M	L	M	L	M	L
CO5	H	H	H	M	M	L	H	L	H	L

H- High; M-Medium; L-Low

CONTENTS

1 IMAGE LOADING, EXPLORINGAND DISPLAYING AN IMAGE

AIM:

TO WRITE A PROGRAM FOR IMAGE LOADING, EXPLORING, AND DISPLAYING AN IMAGE

ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Read the Image**
- **Step 3: Convert the Image to Grayscale**
- **Step 4: Display the Grayscale Image**
- **Step 5: Save the Original Image to a New Location**

SOURCE CODE:

```
import cv2

img = cv2.imread("C:/Users/Dell/Downloads/modi.jpeg") imgGray
= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) cv2.imshow("Gray
Image", imgGray)
cv2.waitKey(0)
cv2.imwrite("E:/College Info/sem 2/lab/cv/modi.jpeg", img)
```

OUTPUT:

RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

2 ACCESS AND MANIPULATE OF IMAGE PIXELS

AIM:

TO WRITE A PROGRAM FOR ACCESSING AND MANIPULATING IMAGE PIXELS

ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Read the Image**
- **Step 3: Accessing the pixel value**
- **Step 4: modifying the pixel value**
- **Step 5: Save the Original Image to a New Location**
- **Step 6:Display the modified image**

SOURCE CODE:

```
import cv2

img = cv2.imread('C:/Users/Dell/Downloads/1.jpg', cv2.IMREAD_COLOR)
value = img[10, 10, :]
print("ACCESSING PIXEL VALUES :", value)

img[10, 10, 0] = 255 value
= img[10, 10, :]
print("MODIFYING PIXEL VALUES :", value)

cv2.imshow('Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

```
MODIFYING PIXEL VALUES FOR GRAYSCALE IMAGES
[ 28  98 246]
ACCESSING PIXEL VALUES FOR COLOR IMAGES
[255  98 246]
```

RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY.

3 IMAGE TRANSFORMATIONS
i) RESIZING
ii) ROTATION

AIM:

TO WRITE A PROGRAM FOR IMAGE TRANSFORMATIONS LIKE IMAGE RESIZING AND ROTATING

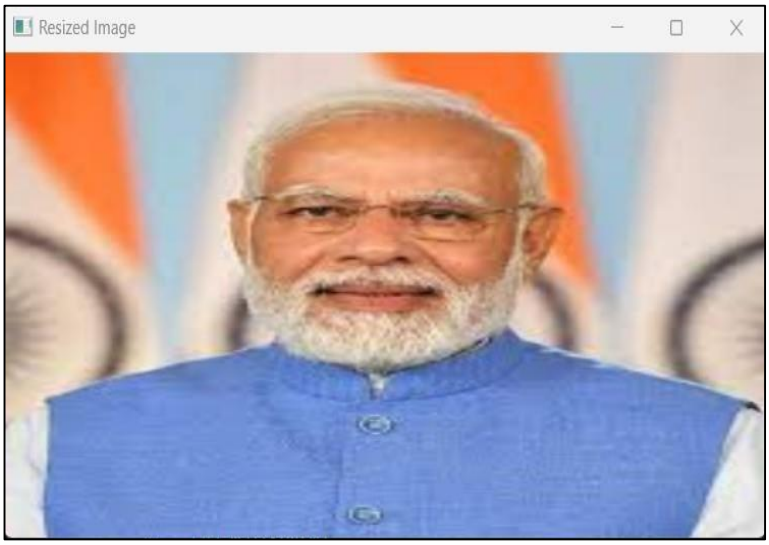
ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Read the Image**
- **Step 3: Resize the image using cv2,resize() function**
- **Step 4: Resize the image using cv2,getRotationMatrix2D**
- **Step 5:Display the modified image using** cv2.imshow

SOURCE CODE:

```
import cv2

img = cv2.imread("C:/Users/Dell/Downloads/modi.jpeg")

resized_img = cv2.resize(img, (600, 300))

rotation_matrix = cv2.getRotationMatrix2D((img.shape[1]/2,
img.shape[0]/2), 30, 1)

rotated_img = cv2.warpAffine(img, rotation_matrix, (img.shape[1], img.shape[0]))

cv2.imshow("Resized Image", resized_img) cv2.imshow("Rotated Image",
rotated_img) cv2.waitKey(0)
```

OUTPUT:



RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

4 ADDITION OPERATION OF TWO IMAGES

AIM:

TO WRITE A PROGRAM FOR ADDITION OPERATION OF TWO IMAGE

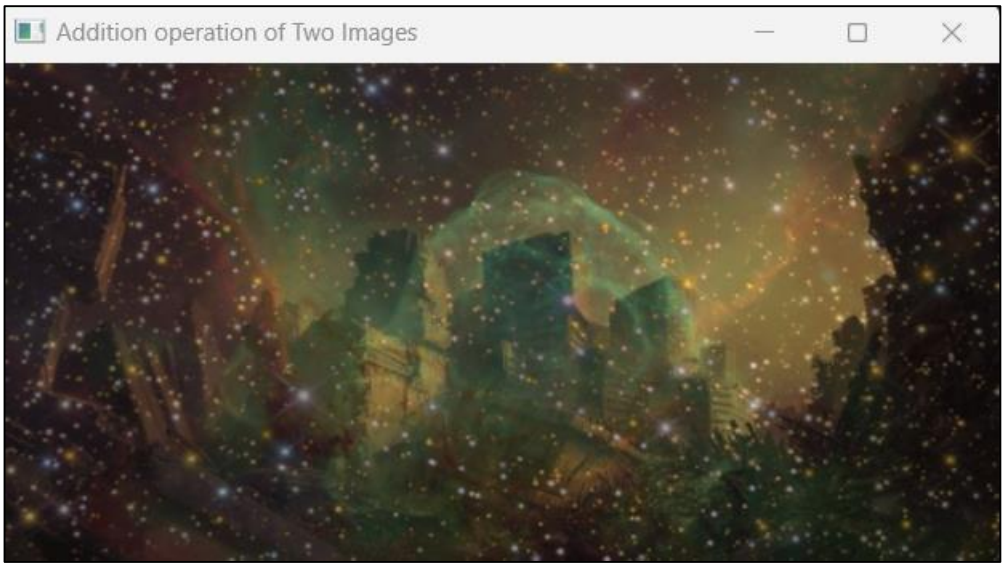
ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Read the Image 1**
- **Step 3: Read the Image 2**
- **Step 4: Add the two images 1 and 2 using cv2.addweighted() function**
- **Step 5:Display the modified image using cv2.imshow() functiom**

SOURCE CODE:

```
import cv2

image_one = cv2.imread("C:/Users/Dell/Downloads/img1.jpg") image_two =
cv2.imread("C:/Users/Dell/Downloads/img2.jpg") result_image =
cv2.addWeighted(image_one, 0.5, image_two, 0.5, 0) cv2.imshow('Addition
operation of Two Images', result_image) cv2.waitKey(0)
```

OUTPUT:



RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

5 IMAGE FILTERING OPERATIONS

i) MEAN FILTERING

ii) GAUSSIAN FILTERING

AIM:

TO WRITE A PROGRAM FOR IMAGE FILTERING OPERATIONS LIKE MEAN AND GAUSSIAN FILTERING

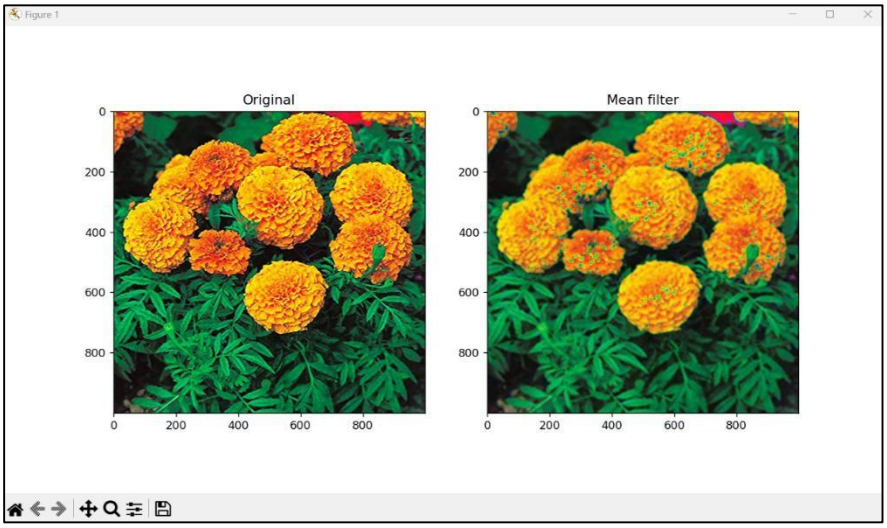
ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Import Numpy library**
- **Step 3: Import matplotlib library**
- **Step 4: Read the Image using cv2.imread()**
- **Step 5: Blur the image using cv2.blur() function and show the image using cv2.imshow() function**

SOURCE CODE:

```
import cv2

import numpy as np

from matplotlib import pyplot as plt


# Mean Filtering

image = cv2.imread('C:/Users/Dell/Downloads/71.jpg')

new_image = cv2.blur(image,(9, 9))

plt.subplot(121), plt.imshow(cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)),plt.title('Original')

plt.subplot(122), plt.imshow(cv2.cvtColor(new_image,
cv2.COLOR_BGR2RGB)),plt.title('Mean Filter')

plt.show()


# Gaussian Filtering

img = cv2.imread("C:/Users/Dell/Downloads/nature.jpeg")

dst = cv2.GaussianBlur(img,(9,9),cv2.BORDER_REFLECT_101)

cv2.imshow('Gaussian Blur Image', np.hstack((img, dst))) cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY.

6 IMAGE BINARIZATION USING SIMPLE THRESHOLDING METHOD

AIM:

TO WRITE A PROGRAM FOR IMAGE BINARIZATION USING SIMPLE THRESHOLD METHOD

ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Read the Image using cv2.imread()**
- **Step 3: Change the colour using cvtColor()**
- **Step 4: By changing the threshold of the image using cv2.threshold() function**
- **Step 5: show the image using cv2.imshow() function**

SOURCE CODE:

```
import cv2


im = cv2.imread("C:/Users/Dell/Downloads/im2.jpeg")
img = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
cv2.imshow('Binary Threshold', thresh1)
cv2.waitKey(0)
```

OUTPUT:

RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

7 FIND GRAYSCALE AND RGB HISTOGRAMS OF AN IMAGE

AIM:

TO WRITE A PROGRAM FOR FIND GRAYSCALE AND RGB HISTOGRAMS OF AN IMAGE

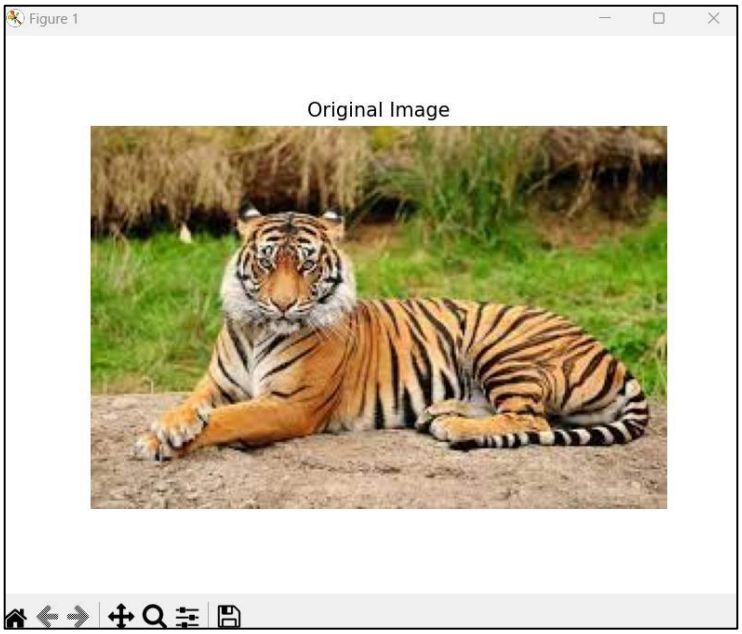
ALGORITHM:

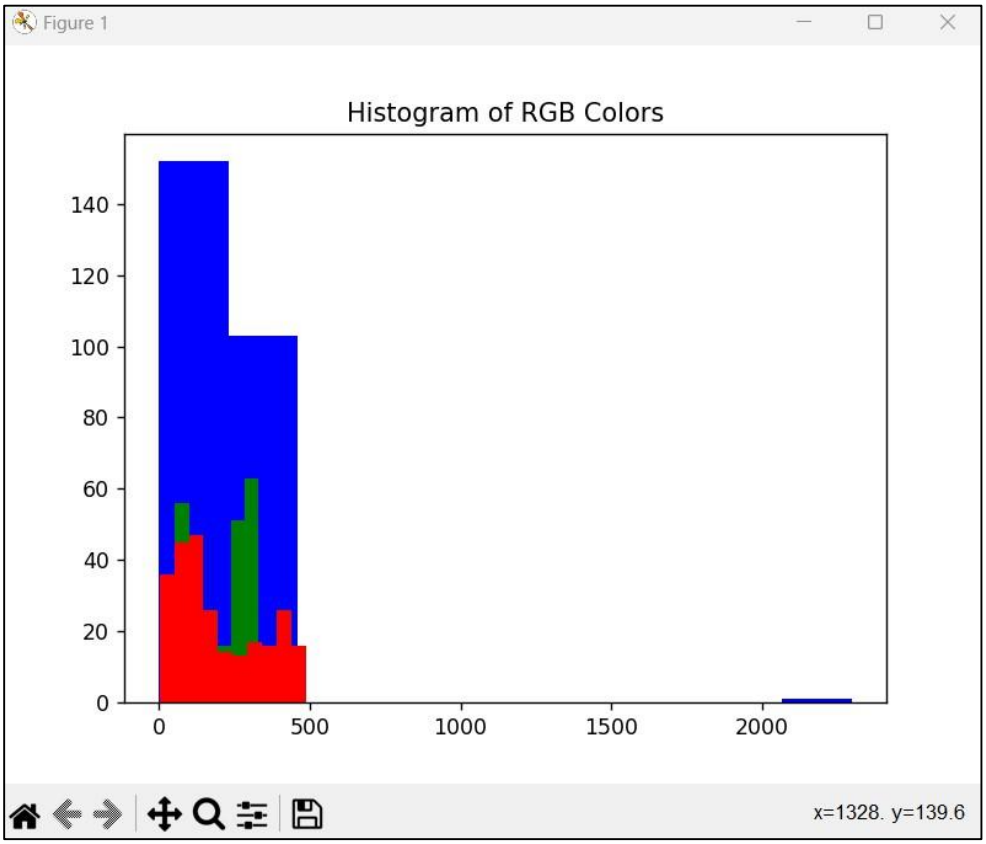
- **Step 1: Import OpenCV Library**
- **Step 2: Import matplotlib library**
- **Step 3: Read the Image using cv2.imread()**
- **Step 4: Giving the permissions and title for the graphical representation**
- **Step 5: by sets the colours respectively RED GREEN and BLUE with an function of display the histogram cv2.imshow()**

SOURCE CODE:

```
import cv2
import matplotlib.pyplot as plt

imageObj = cv2.imread('C:/Users/Dell/Downloads/im2.jpeg')
plt.axis("off")
plt.title("Original Image")
plt.imshow(cv2.cvtColor(imageObj, cv2.COLOR_BGR2RGB))
plt.show()

blue_color = cv2.calcHist([imageObj], [0], None, [256], [0, 256]) red_color =
cv2.calcHist([imageObj], [1], None, [256], [0, 256]) green_color =
cv2.calcHist([imageObj], [2], None, [256], [0, 256]) plt.title("Histogram of RGB
Colors")
plt.hist(blue_color, color="blue")
plt.hist(green_color, color="green")
plt.hist(red_color, color="red") plt.show()
```

OUTPUT:





RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

8 SEGMENT AN IMAGE USING K-MEANS CLUSTERING ALGORITHM

AIM:

TO WRITE A PROGRAM FOR SEGMENT AN IMAGE USING K-MEANS CLUSTERING ALGORITHM

ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Import Numpy library**
- **Step 3: Read the Image using cv2.imread()**
- **Step 4: set the properties for segmentation using numpy as res, float, label center, criteria**
- **Step 5:Display the modified image using cv2.imshow() function**

SOURCE CODE:

```
import numpy as np
import cv2 as cv

img = cv.imread('C:/Users/Dell/Downloads/ig2.jpg') Z =
img.reshape((-1,3))
Z = np.float32(Z)
criteria = (cv.TERM_CRITERIA_EPS +
cv.TERM_CRITERIA_MAX_ITER, 10, 1.0) K = 8
ret,label,center=cv.kmeans(Z,K,None,criteria,10,cv.KMEANS_RANDO M_CENTERS)
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape)) cv.imshow('Image using
K-means Cluster',res2) cv.waitKey(0)
cv.destroyAllWindows()
```

OUTPUT:

RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

9 EDGE DETECTION OPERATION USING SOBEL AND SCHARR GRADIENTS

AIM:

TO WRITE A PROGRAM FOR EDGE DETECTION OPERATION USING SOBEL AND SCHARR GRADIENTS

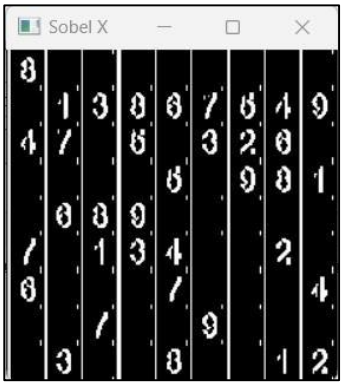
ALGORITHM:

- **Step 1: Import OpenCV Library**
- **Step 2: Read the Image using cv2.imread()**
- **Step 3: set the properties for sobel x and sobel y and scharr x and scharr y**
- **Step 4: Display the each image using cv2.imshow() function**

SOURCE CODE:

```
import cv2

img = cv2.imread('C:/Users/Dell/Downloads/sudo.png',
cv2.IMREAD_GRAYSCALE)

sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)

scharrx = cv2.Scharr(img, cv2.CV_64F, 1, 0)
scharry = cv2.Scharr(img, cv2.CV_64F, 0, 1)

cv2.imshow('Original', img)
cv2.imshow('Sobel X', sobelx)
cv2.imshow('Sobel Y', sobely)
cv2.imshow('Scharr X', scharrx)
cv2.imshow('Scharr Y', scharry)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:



Sobel

Scharr

RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

10 WRITE A PROGRAM TO CLASSIFY AN IMAGE USING KNN CLASSIFICATION ALGORITHM

AIM:

To WRITE A PROGRAM TO CLASSIFY AN IMAGE USING KNN CLASSIFICATION ALGORITHM

ALGORITHM:

- **Step 1: Import time Library**
- **Step 2: Import Numpy library**
- **Step 3: Import pandas library**
- **Step 4: Import matplotlib library**
- **Step 5: Import sklearn library**
- **Step 6: Import seaborn library**
- **Step 7: creating ojects instances test x , y and train x ,y , digits , predictions**
- **Step 8: Read the Image using cv2.imread()**
- **Step 9: figure the data using plt.figure() function and set the titles**
- **Step 10:Display the resulted data in command prompt**

SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
from sklearn.datasets import load_digits

digits = load_digits()
print(digits.keys())
print('Label Data Shape', digits.target.shape)

X = digits.images

from sklearn.metrics import accuracy_score, confusion_matrix from
sklearn.model_selection import train_test_split
from sklearn.multiclass import OneVsRestClassifier
from sklearn.neighbors import KNeighborsClassifier import
seaborn as sns

X = digits.data
y = digits.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
knn = OneVsRestClassifier(KNeighborsClassifier())
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
print('KNN Accuracy: %.3f' % accuracy_score(y_test, predictions))

cm = confusion_matrix(y_test, predictions)
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt='.3f', linewidths=.5, square=True,
cmap='Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title='Accuracy_Score:{0}'.format(accuracy_score(y_test,pre dictions))
plt.title(all_sample_title, size=15)
```

OUTPUT:

```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'])
Label Data Shape (1797,)
KNN Accuracy: 0.980
```

RESULT:

THUS THE ABOVE PROGRAM HAS BEEN EXECUTED SUCESSFULLY

Reference Books:

1. Richard Szeliski, “Computer Vision: Algorithms and Applications”, Springer-Verlag London Limited, 2011.
2. Richard Hartley and Andrew Zisserman, “Multiple View Geometry in Computer Vision”, 2nd Edition, Cambridge University Press, 2003.
3. Rajalingappaa Shanmygamani, “Deep Learning for Computer Vision”, Packt Publishing, 2018.
4. Chistopher M. Bishop, “Pattern Recognition and Machine Learning”, Springer Science + Business Media, LLC, 2006.
5. Adrian Kaehler and Gary Bradski, “Learning OpenCV3 Computer Vision in C++ with the OpenCV Library”, O’Reilly, 2017.

Website References:

1. https://opencv.org/university/free-opencv-course/?utm_source=opcv&utm_medium=menu&utm_campaign=obc
2. https://pyimagesearch.com/
3. https://www.kaggle.com/datasets
4. https://cs231n.stanford.edu/
5. https://paperswithcode.com/