Practical no :01

Aim: Study and Implement NumPy ,array, attributes, functions, matrix

```
import numpy as np

a1=np.array([1,2,3,4,5,6,7,8,9,10])

a2=np.array([[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10]])

a3=np.array([[[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10],[
1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10]]])

print ("1D array is",a1)

print ("2D array is",a2)

print ("3D array is",a3)

print ("Shape of 1D array is",a1.shape)

print ("Shape of 2D array is",a2.shape)

print ("Shape of 3D array is",a3.shape)

print ("size of 1D array is",a1.size)

print ("size of 2D array is",a2.size)

print ("Size of 3D array is",a3.size)

print ("Dimension of 1D array is",a1.ndim)

print ("Dimension of 2D array is",a2.ndim)

print ("Dimension of 3D array is",a3.ndim)

print ("datatype of 1D array is",a1.dtype)

print ("datatype of 2D array is",a2.dtype)

print("datatype of 3D array is",a3.dtype)

print ("length of 1D array is",len(a1))

print ("length of 2D array is",len(a2))

print ("length of 3D array is",len(a3))

#.......functions on array(mean, sum, argmin, argmax & reshape)…….

import numpy as np
```

```
a2=np.array([[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10]])

print("sum of a:",a2.sum())

print("mean of a:",a2.mean())

print("argmin value of a:",a2.argmin())

print("argmax of a:",a2.argmax())

a21 = a2.reshape (30)

print("reshape of a2:",a21)

a4 = np. zeros ((3,3))

a4

a5 = np.ones ((3,3))

a5

a6 = np.identity (3)

a6
```

Practical No 02

Aim: Implementing various basic image processing operations in python-open cv.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread(r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg")
plt.imshow(img)
plt.show()
#---------- write an image -------------------
import numpy as np
import matplotlib.pyplot as plt
import cv2
path=r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg"
wpath=r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg"
vpath=r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg"
upath=r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg"
xpath=r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg"
bgr = cv2.imread(path,1)
rgb = cv2.cvtColor(bgr,cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(bgr,cv2.COLOR_BGR2GRAY)
hsv = cv2.cvtColor(bgr,cv2.COLOR_BGR2HSV)
if(cv2.imwrite(wpath,bgr)):
  print("BGR File Written Succesfully...")
else:
 print("BGR Writin File was unsuccesful...")
if(cv2.imwrite(vpath,rgb)):
 print("RGB File Written Succesfully...")
else:
 print("RGB Writin File was unsuccesful...")
if(cv2.imwrite(upath,gray)):
 print("Gray File Written Succesfully...")
else:
 print("GRAYWritin File was unsuccesful...")
if(cv2.imwrite(xpath,hsv)):
 print("HSV File Written Succesfully...")
else:
 print("HSV Writin File was unsuccesful...")
#------------------ basic operations ------------
import cv2
import numpy as np
import matplotlib.pyplot as plt
ragvinder = cv2.imread(r"C:\Users\Admin\OneDrive\Desktop\butterfly.jpeg")
plt.imshow(ragvinder)
print("size of image:",ragvinder.size)
print("dimension of image:",ragvinder.ndim)
print("datatype of image:",ragvinder.dtype)
print("shape of image:",ragvinder.shape)
#----- color conversion -------
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread(r'C:\Users\Admin\OneDrive\Desktop\flower.jpg')
plt.imshow(img)
plt.show
rgb=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(rgb)
plt.show()
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
plt.imshow(gray,cmap="gray")
plt.show()
hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
plt.imshow(hsv)
plt.show()
#--- display color channel -----
import numpy as np
import matplotlib.pyplot as plt
import cv2
path = r'C:\Users\Admin\OneDrive\Desktop\flower.jpg'
img = cv2.imread(path,1)
imgrgb = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.subplot(2,2,1)
plt.imshow(imgrgb)
plt.show()
r,g,b = cv2.split(imgrgb)
plt.subplot(2,2,2)
plt.imshow(r)
plt.show()
plt.subplot(2,2,3)
plt.imshow(g)
plt.show()
plt.subplot(2,2,4)
plt.imshow(b)
plt.show()
#------ annotation on image ------
import numpy as np
import matplotlib.pyplot as plt
import cv2
path = r"J:\BK Birla - MSC Part - Advanced IoT\Compu Vision-2023\Scenary.jpg"
img = cv2.imread(path,1)
imgrgb = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
cv2.putText(imgrgb,"Nature",(198,101),cv2.FONT_HERSHEY_SCRIPT_COMPLEX,1.6,(255,0,0),3)
cv2.putText(imgrgb,"Nature",(200,100),cv2.FONT_HERSHEY_SCRIPT_COMPLEX,1.5,(255,255,255),3)
plt.imshow(imgrgb)
plt.show()
```

Practical No 03

Aim: To Perform Arithmetic Operations and Bitwise Operations on Image

```
import numpy as np

import matplotlib.pyplot as plt

import cv2

# Paths to the images

path = r"C:\Users\PC0047\Downloads\Doremom.jpg"

path1 = r"C:\Users\PC0047\Downloads\dora.jpg"

# Read the images

img = cv2.imread(path, 1)

img1 = cv2.imread(path1, 1)

# Check if images are loaded successfully

if img is None:

   print("Error loading image 1. Check the file path.")

   exit()

if img1 is None:

   print("Error loading image 2. Check the file path.")

   exit()

# Print original sizes

print("Original Image 1 size =", img.shape)

print("Original Image 2 size =", img1.shape)

# Resize both images to a common size (e.g., the size of img)

img1_resized = cv2.resize(img1, (img.shape[1], img.shape[0]), cv2.INTER_AREA)

# Convert BGR to RGB

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

img1_resized = cv2.cvtColor(img1_resized, cv2.COLOR_BGR2RGB)
```

```
# Print resized sizes

print("Resized Image 1 size =", img.shape)

print("Resized Image 2 size =", img1_resized.shape)

# Perform bitwise operations

img_or = cv2.bitwise_or(img, img1_resized)

img_and = cv2.bitwise_and(img, img1_resized)

img_xor = cv2.bitwise_xor(img, img1_resized)

img_not = cv2.bitwise_not(img)

# Plotting the results

plt.figure(figsize=(10, 8))

plt.subplot(221)

plt.title("Image 1")

plt.imshow(img)

plt.axis('off')

plt.subplot(222)

plt.title("Image 2 (Resized)")

plt.imshow(img1_resized)

plt.axis('off')

plt.subplot(223)

plt.title("Bitwise OR")

plt.imshow(img_or)

plt.axis('off')

plt.subplot(224)

plt.title("Bitwise AND")

plt.imshow(img_and)

plt.axis('off')
plt.tight_layout()
plt.show()
# Optional: display the XOR and NOT images
plt.figure(figsize=(10, 4))
plt.subplot(121)
plt.title("Bitwise XOR")
plt.imshow(img_xor)
plt.axis('off')
plt.subplot(122)
plt.title("Bitwise NOT")
plt.imshow(img_not)
plt.axis('off')
plt.tight_layout()
plt.show()
```

Practical No 04

Aim: Implement Histogram processing and equalization.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Load the image
image = cv2.imread("C:/Users/PC0047/Downloads/image1.jpeg")  # Replace with your image path
# Convert the image to grayscale (for simplicity)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Function to plot histogram of an image
def plot_histogram(img, title="Histogram"):
    plt.figure(figsize=(6, 4))
    plt.hist(img.ravel(), bins=256, range=[0, 256], color='gray', alpha=0.7)
    plt.title(title)
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
    plt.show()
# Plot histogram of the original image
plot_histogram(gray_image, "Original Image Histogram")
# **Histogram Equalization**
# Apply histogram equalization to the grayscale image
equalized_image = cv2.equalizeHist(gray_image)
# Plot histogram of the equalized image
plot_histogram(equalized_image, "Equalized Image Histogram")
# Display the original and equalized images using OpenCV
cv2.imshow("Original Image", gray_image)
cv2.imshow("Equalized Image", equalized_image)
```

```
# Wait until any key is pressed and then close the windows

cv2.waitKey(0)

cv2.destroyAllWindows()

# Optionally save the equalized image

cv2.imwrite('equalized_image.jpg', equalized_image)
```

Practical no 05

Aim: Implement various low and high pass filtering mechanism

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread(r"C:\Users\PC0047\Downloads\image1.jpeg")
assert img is not None, "file could not be read, check with os.path.exists()"
blur = cv.blur(img,(5,5))
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
import cv2
import numpy as np
import matplotlib.pyplot as plt
# read image
img = cv2.imread(r"C:\Users\PC0047\Downloads\image1.jpeg")
# resize image
img = cv2.resize(img, (500, 450), interpolation=cv2.INTER_CUBIC)
# convert image to gray scale image
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# apply laplacian blur
laplacian = cv2.Laplacian(gray, cv2.CV_64F)
# sobel x filter where dx=1 and dy=0
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=7)
# sobel y filter where dx=0 and dy=1
```

```
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=7)
# combine sobel x and y
sobel = cv2.bitwise_and(sobelx, sobely)
# plot images
plt.subplot(2, 2, 1)
plt.xticks([])
plt.yticks([])
plt.imshow(laplacian, cmap='gray')
plt.title('Laplacian')
plt.subplot(2, 2, 2)
plt.xticks([])
plt.yticks([])
plt.imshow(sobelx, cmap='gray')
plt.title('SobelX')
plt.subplot(2, 2, 3)
plt.xticks([])
plt.yticks([])
plt.imshow(sobely, cmap='gray')
plt.title('SobelY')
plt.subplot(2, 2, 4)
plt.xticks([])
plt.yticks([])
plt.imshow(sobel, cmap='gray')
plt.title('Sobel')
plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Practical No 06

Aim: Write a python code for implementing image segmentation.

```
import cv2
import numpy as np
img = cv2.imread(r"C:\Users\PC0047\Downloads\image1.jpeg")
cv2.imshow('original image', img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 0, 255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
# Noise removal
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel,
iterations = 2)
# Sure background area
sure_bg = cv2.dilate(opening,kernel,iterations=3)
# Finding sure foreground area
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg =cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)
# Marker labelling
ret, markers = cv2.connectedComponents(sure_fg)
# Add one to all labels so that sure background is not 0, but 1
markers = markers+1
# Now, mark the region of unknown with zero
markers[unknown==255] = 0
```

```
markers = cv2.watershed(img,markers)

img[markers == -1] = [255,0,0]

cv2.imshow('segmented image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Practical No 07

Aim: Implement different Morphological operations on image.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Function to read and binarize the image
def read_and_binarize_image(image_path):
    img = cv2.imread(image_path, 0)
    _, binarized = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    return binarized
# Function to apply morphological operations and display the result
def apply_morphological_operation(image, operation, kernel_size=(3, 3), iterations=1, invert=False, title=""):
    if invert:
        image = cv2.bitwise_not(image)
    kernel = np.ones(kernel_size, np.uint8)
    result = cv2.morphologyEx(image, operation, kernel, iterations=iterations)
    plt.imshow(result, cmap='Blues')
    plt.title(title)
    plt.axis('off')  # Hide axes
    plt.show()
# Load and binarize the image
image_path = r"C:/Users/PC0047/Downloads/image1.jpeg"
binr = read_and_binarize_image(image_path)
# Erosion
apply_morphological_operation(binr, cv2.MORPH_ERODE, kernel_size=(5, 5), iterations=1, invert=True, title="Erosion")
# Dilation
```

```
apply_morphological_operation(binr, cv2.MORPH_DILATE, kernel_size=(3, 3), iterations=1,
invert=True, title="Dilation")

# Opening

apply_morphological_operation(binr, cv2.MORPH_OPEN, kernel_size=(3, 3), iterations=1,
invert=False, title="Opening")

# Closing

apply_morphological_operation(binr, cv2.MORPH_CLOSE, kernel_size=(3, 3), iterations=1,
invert=False, title="Closing")

# Morphological Gradient

apply_morphological_operation(binr, cv2.MORPH_GRADIENT, kernel_size=(3, 3), iterations=1,
invert=True, title="Morphological Gradient")

# Top Hat

apply_morphological_operation(binr, cv2.MORPH_TOPHAT, kernel_size=(13, 13), iterations=1,
invert=False, title="Top Hat")

# Black Hat

apply_morphological_operation(binr, cv2.MORPH_BLACKHAT, kernel_size=(5, 5), iterations=1,
invert=True, title="Black Hat")
```

Practical No 08

Aim: Writing and reading in Video using openCV

```
import cv2
# Create a video capture object, in this case we are reading the video from a file
vid_capture = cv2.VideoCapture("C:/Users/PC0047/Downloads/2103099-uhd_3840_2160_30fps.mp4")
if (vid_capture.isOpened() == False):
 print("Error opening the video file")
# Read fps and frame count
else:
 # Get frame rate information
 # You can replace 5 with CAP_PROP_FPS as well, they are enumerations
 fps = vid_capture.get(5)
 print('Frames per second : ', fps,'FPS')
 # Get frame count
 # You can replace 7 with CAP_PROP_FRAME_COUNT as well, they are enumerations
 frame_count = vid_capture.get(7)
 print('Frame count : ', frame_count)
while(vid_capture.isOpened()):
 # vid_capture.read() methods returns a tuple, first element is a bool
 # and the second is frame
 ret, frame = vid_capture.read()
 if ret == True:
  cv2.imshow('Frame',frame)
  # 20 is in milliseconds, try to increase the value, say 50 and observe
  key = cv2.waitKey(20)
  if key == ord('q'):
   break
 else:
  break
# Release the video capture object
vid_capture.release()
cv2.destroyAllWindows()
```

Practical No :09

Aim: Component Analysis

```
import cv2
import numpy as np
# Loading the image
img = cv2.imread(r"C:\Users\PC0047\Downloads\Doremon1.jpg")
# preprocess the image
gray_img = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)
# Applying 7x7 Gaussian Blur
blurred = cv2.GaussianBlur(gray_img, (7, 7), 0)
# Applying threshold
threshold = cv2.threshold(blurred, 0, 255,
        cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
# Apply the Component analysis function
analysis = cv2.connectedComponentsWithStats(threshold,4, cv2.CV_32S)
(totalLabels, label_ids, values, centroid) = analysis
# Initialize a new image to store
# all the output components
output = np.zeros(gray_img.shape, dtype="uint8")

# Loop through each component
for i in range(1, totalLabels):
        # Area of the component
        area = values[i, cv2.CC_STAT_AREA]
        if (area > 140) and (area < 400):
                componentMask = (label_ids == i).astype("uint8") * 255
                output = cv2.bitwise_or(output, componentMask)
```

```
cv2.imshow("Image", img)

cv2.imshow("Filtered Components", output)

cv2.waitKey(0)
```

Practical No 10

Aim: Performing all 5 Threshold operations on images

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread(r"C:\Users\PC0047\Downloads\image1.jpeg",cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.exists()"
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
titles = ['OriginalImage','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
 plt.subplot(2,3,i+1),plt.imshow(images[i],'gray',vmin=0,vmax=255)
plt.title(titles[i])
plt.xticks([]),plt.yticks([])
plt.show()
```