Practical No.1

Aim: Write a program in python to read and display an image.

Softwares Used: Jupyter Notebook, OpenCV.

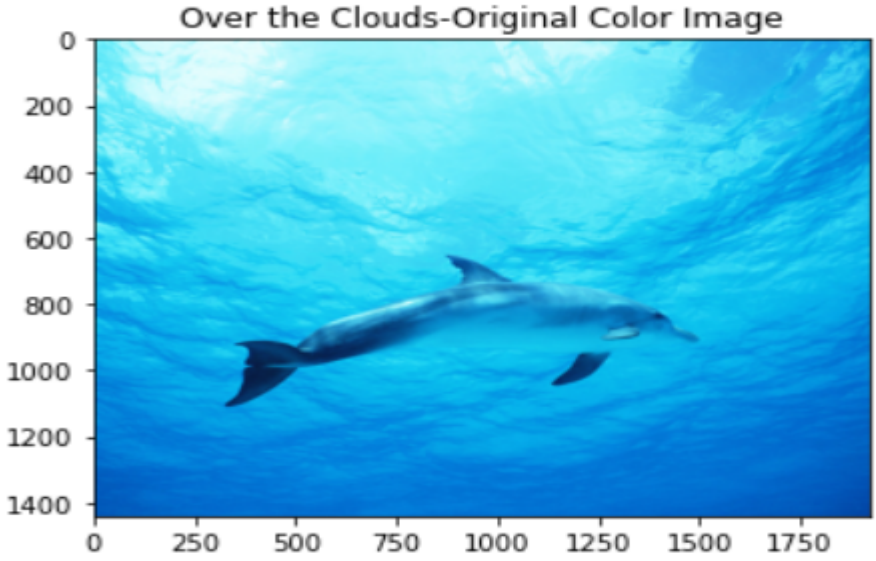
Theory: The practical demonstrates how to read and display an image using Python's OpenCV library and Jupyter Notebook. The program imports the necessary libraries and uses the cv2.imread() function to read an image file. The image's color space is then converted from BGR to RGB using cv2.cvtColor() function, and the image is displayed using matplotlib's imshow() function.

Code:

```
import cv2
from matplotlib import pyplot as plt
print(cv2.__version__)

img = cv2.imread("dolphin.jpg")
plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
plt.title("Over the Clouds-Original Color Image")
```

Output:

```
4.7.0

Text(0.5, 1.0, 'Over the Clouds-Original Color Image')
```



Conclusion: The Python program using OpenCV and Jupyter Notebook to read and display an image has been successfully implemented.

Practical No.2

Aim: Write a program in python to read an image and print a grayscale image.

Softwares Used: Jupyter Notebook, OpenCV

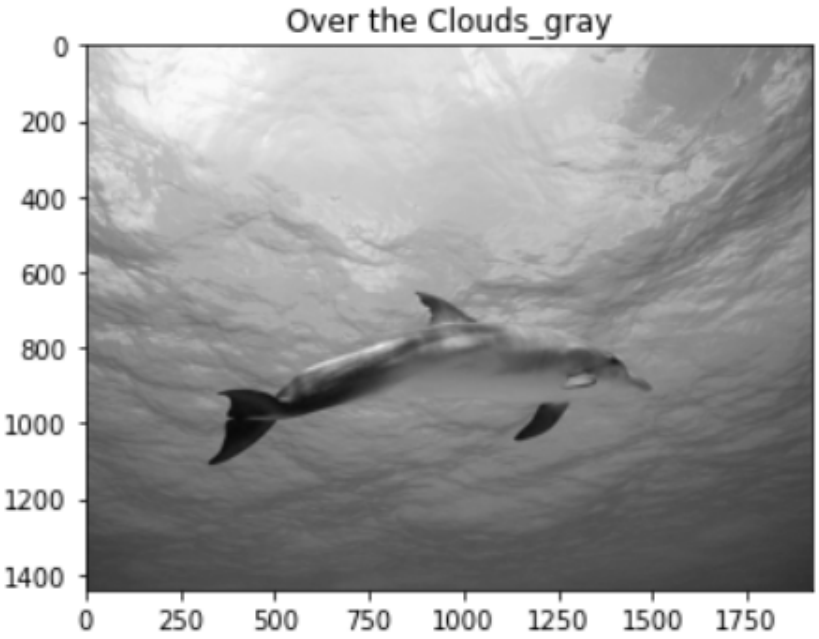
Theory: The practical demonstrates how to read an image and convert it to grayscale using Python's OpenCV library and Jupyter Notebook. The program imports the necessary libraries and reads an image file using the cv2.imread() function. The image is then converted to grayscale using cv2.cvtColor() function, and the grayscale image is displayed using matplotlib's imshow() function with a gray color map.

Code:

```
import cv2
from matplotlib import pyplot as plt
print(cv2.__version__)

img = cv2.imread("dolphin.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray')
plt.title("Over the Clouds_gray")
```

Output:

```
4.7.0

Text(0.5, 1.0, 'Over the Clouds_gray')
```



Conclusion: we have successfully studied practical which provided a basic understanding of how to convert an image to grayscale in Python and OpenCV.

Practical No.3

Aim: Write a program in python to scale an image.

Softwares Used: Jupyter Notebook, OpenCV

Theory: The practical demonstrates how to scale an image using Python's OpenCV library and Jupyter Notebook. The program imports the necessary libraries and reads an image file using the cv2.imread() function. The image's height and width are obtained using the img.shape property, and the image is resized using the cv2.resize() function, specifying the desired size and interpolation method. The scaled image is then written to a file using cv2.imwrite() function.

Code:

```
import cv2
import numpy as np

FILE_NAME = 'dolphin.jpg'
try:

# Read image from disk.
img = cv2.imread(FILE_NAME)

# Get the number of pixels horizontally and vertically.
(height, width) = img.shape[:2]

# Specify the size of image along with interpolation methods.
# cv2.INTER_AREA is used for shrinking, whereas cv2.INTER_CUBIC

# is used for zooming.
res = cv2.resize(img, (int(width / 2), int(height / 2)),interpolation =
cv2.INTER_CUBIC)

# Write the image back to disk.
cv2.imwrite('result.jpg', res)

except IOError:
print ('Error while reading files !!!')
```

Output:

Conclusion: We have successfully studied and implented a python program to scale an image.

Practical No.4

Aim: Write a program in python to rotate an image.

Softwares Used: Jupyter Notebook, OpenCV

Theory: The practical demonstrates how to rotate an image using Python's OpenCV library and Jupyter Notebook. The image's shape in terms of pixels is obtained using the img.shape property. The cv2.getRotationMatrix2D() function is used to create a matrix needed for transformation, and cv2.warpAffine() function is used to apply the transformation to the image. Finally, the rotated image is written to a file using cv2.imwrite() function.

Code:

```
import cv2
import numpy as np
FILE_NAME = 'dolphin.jpg'
try:

# Read images from the disk.
img = cv2.imread(FILE_NAME)

# Shape of image in terms of pixels.
(rows, cols) = img.shape[:2]

# getRotationMatrix2D creates a matrix needed for transformation.
# We want a matrix for rotation w.r.t center to 45 degree without scaling.
M = cv2.getRotationMatrix2D((cols / 2, rows / 2), 45, 1)
res = cv2.warpAffine(img, M, (cols, rows))

# Write the image back to disk.
cv2.imwrite('result.jpg', res)
except IOError:
 print ('Error while reading files !!!')
```

Output:

Conclusion: We have successfully studied and implemented the python program to rotate an image.

Practical No.5

Aim: Write a program in python to translate an image.

Softwares Used: Jupyter Notebook, OpenCV

Theory: The practical demonstrates how to translate an image using Python's OpenCV library and Jupyter Notebook. The program imports the necessary libraries and reads an image file using the cv2.imread() function. A translation matrix is created using the np.float32() function to specify the shift in pixels. The cv2.warpAffine() function is used to apply the translation to the image, and the result is written to a file using the cv2.imwrite() function.

Code:

```
import cv2
import numpy as np
FILE_NAME = 'dolphin.jpg'

# Create a translation matrix.
# If the shift is (x, y) then matrix would be
# M = [1 0 x]
# [0 1 y]
# Let's shift by (100, 50).
M = np.float32([[1, 0, 100], [0, 1, 50]])
try:

# Read image from disk.
img = cv2.imread(FILE_NAME)
(rows, cols) = img.shape[:2]

# warpAffine does appropriate shifting given the
# translation matrix.
res = cv2.warpAffine(img, M, (cols, rows))

# Write the image back to disk.
cv2.imwrite('result.jpg', res)
except IOError:
  print ('Error while reading files !!!')
```

Output:

Conclusion: We have successfully studied and implemented a python program to translate an image using OpenCV.

Practical No.6

Aim: Write a program in python for edge detection of an image.

Softwares Used: Jupyter Notebook, OpenCV

Theory: The practical demonstrates how to perform edge detection of an image using Python's OpenCV library and Jupyter Notebook. The program imports the necessary libraries and reads an image file using the cv2.imread() function. The cv2.Canny() function is then used to perform the edge detection on the image, using 100 and 200 as the threshold values. The result is written to a file using the cv2.imwrite() function.

Code:

```
import cv2
import numpy as np
FILE_NAME = 'dolphin.jpg'
try:

# Read image from disk.
img = cv2.imread(FILE_NAME)

# Canny edge detection.
edges = cv2.Canny(img, 100, 200)

# Write the image back to disk.
cv2.imwrite('result.jpg', edges)
except IOError:
        print ('Error while reading files !!!')
```

Output:

Conclusion: We have successfully studied and implemented a python program to perform edge detection on an image.

Practical No.7

Aim: Write a program in python for Point Processing in image.

Softwares Used: Jupyter Notebook, OpenCV

Theory: Point processing is a technique in digital image processing where the input pixel values are directly mapped to the output pixel values. In this program, we have demonstrated two point processing operations: image negative and thresholding. Image negative is obtained by subtracting the maximum intensity value of the image from each pixel intensity value. Thresholding is a process of converting a grayscale image into a binary image by setting a threshold value. If a pixel value is less than the threshold value, it is set to 0, else it is set to 255. The program demonstrates thresholding with and without background removal.

Code:

```
import cv2
import numpy as np

# Image negative
img = cv2.imread('Dolphin.jpeg',0)

# To ascertain total numbers of
# rows and columns of the image,
# size of the image
m,n = img.shape

# To find the maximum grey level
# value in the image
L = img.max()

# Maximum grey level value minus
# the original image gives the
# negative image
img_neg = L-img

# convert the np array img_neg to
# a png image
cv2.imwrite('Dolphin_Negative.png', img_neg)

# Thresholding without background
# Let threshold =T
# Let pixel value in the original be denoted by r
# Let pixel value in the new image be denoted by s
# If r<T, s= 0
# If r>T, s=255
T = 150
# create an array of zeros
```

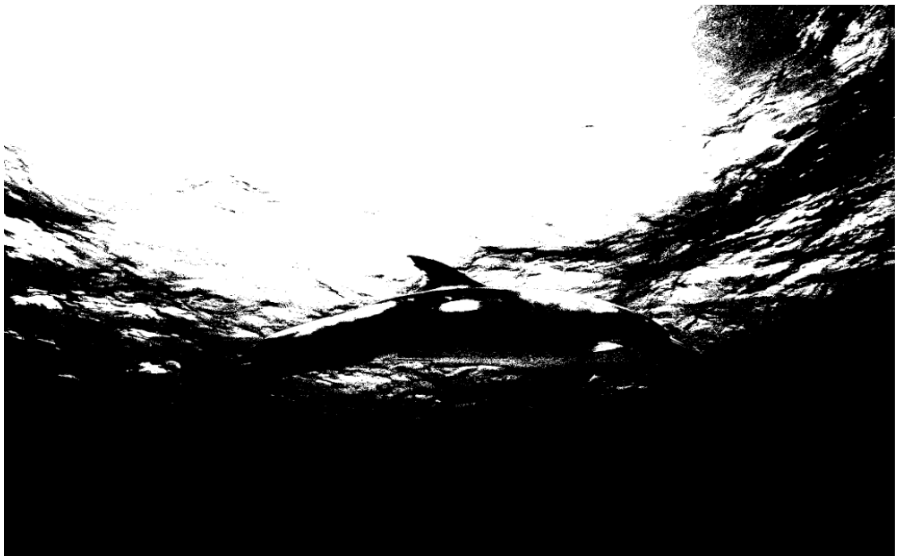
```
img_thresh = np.zeros((m,n), dtype = int)
for i in range(m):
        for j in range(n):
                if img[i,j] < T:
                        img_thresh[i,j]= 0
                Else:
                        img_thresh[i,j] = 255

# Convert array to png image
cv2.imwrite('Dolphin_Thresh.png', img_thresh)

# the lower threshold value
T1 = 100
# the upper threshold value
T2 = 180

# create an array of zeros
img_thresh_back = np.zeros((m,n), dtype = int)
for i in range(m):
        for j in range(n):
                if T1 < img[i,j] < T2:
                        img_thresh_back[i,j]= 255
                Else:
                        img_thresh_back[i,j] = img[i,j]
# Convert array to png image
cv2.imwrite('Dolphin_Thresh_Back.png', img_thresh_back)
```

Output:

1] Dolphin Negative

True

2] Dolphin_Thresh



3] Dolphin_Thresh_Back

Conclusion: we have successfully studied and implented a program in python for Point Processing in image.

Practical No.8

Aim: Write a program in python to insert a watermark in image.

Softwares Used: Jupyter Notebook, OpenCV

Theory: Watermarking is a technique used to protect the ownership and authenticity of digital content. It involves embedding a unique identifier into the content, such as an image, to make it traceable. In this program, we use the Python Imaging Library (PIL) to insert a text watermark into an image. We first open the image, create a copy of it, and then add the text as a watermark using the ImageDraw module.

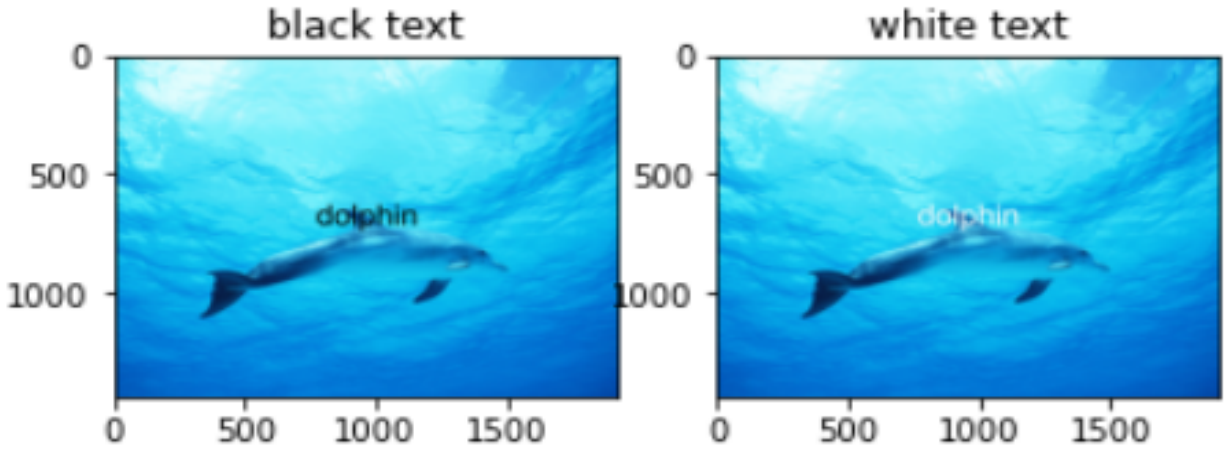
Code:

```
# import all the libraries
from PIL import Image
from PIL import ImageFont
from PIL import ImageDraw
import matplotlib.pyplot as plt
import numpy as np

# image opening
image = Image.open("dolphin.jpg")

# this open the photo viewer
image.show()
plt.imshow(image)
# text Watermark
watermark_image = image.copy()
draw = ImageDraw.Draw(watermark_image)

# ("font type",font size)
w, h = image.size
x, y = int(w / 2), int(h / 2)
if x > y:
font_size = y
elif y > x:
font_size = x
else:
font_size = x
font = ImageFont.truetype("arial.ttf", int(font_size/6))

# add Watermark
# (0,0,0)-black color text
draw.text((x, y), "dolphin", fill=(0, 0, 0), font=font, anchor='ms')
plt.subplot(1, 2, 1)
plt.title("black text")
plt.imshow(watermark_image)
```

```
# add Watermark
# (255,255,255)-White color text
draw.text((x, y), "dolphin", fill=(255, 255, 255), font=font, anchor='ms')
plt.subplot(1, 2, 2)
plt.title("white text")
plt.imshow(watermark_image)
```

Output:

```
<matplotlib.image.AxesImage at 0x261d89d2040>
```



Conclusion: we have successfully studied and implented a program in python to insert a watermark in image.