EXPERIMENT -01 :

PERFORMING OPERATION OF READING AND DISPLAYING IMAGE AND VIDEO IN OPEN CV

AIM:

To read and display image and video file using open-cv in python

PROCEDURE/STEPS:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for reading and displaying the image and video file

Step 4: Execute code

Python code:

```
import cv2 as cv

import numpy as np
```

Reading an image

```
img = cv.imread("korala.jpg")

img
```

Display an image

```
cv.imshow("korala",img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Reading a video

```
# Create a VideoCapture object and read from input file
# If the input is the camera, pass 0 instead of the video file name
cap = cv2.VideoCapture('jellyfish.mp4')
 # Check if camera opened successfully
if (cap.isOpened()== False):
  print("Error opening video stream or file")
 # Read until video is completed
while(cap.isOpened()):
  # Capture frame-by-frame
  ret, frame = cap.read()
  if ret == True:
```

Display a video

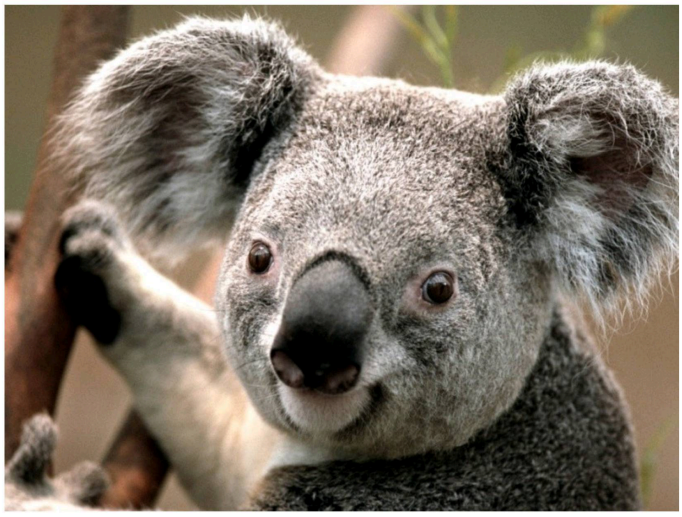
```
# Display the resulting frame
  cv2.imshow('Frame',frame)
  # Press Q on keyboard to  exit
  if cv2.waitKey(25) & 0xFF == ord('q'):
     break
```

```
  # Break the loop
  else:
    break

# When everything done, release the video capture object
cap.release()

# Closes all the frames
cv2.destroyAllWindows()
```

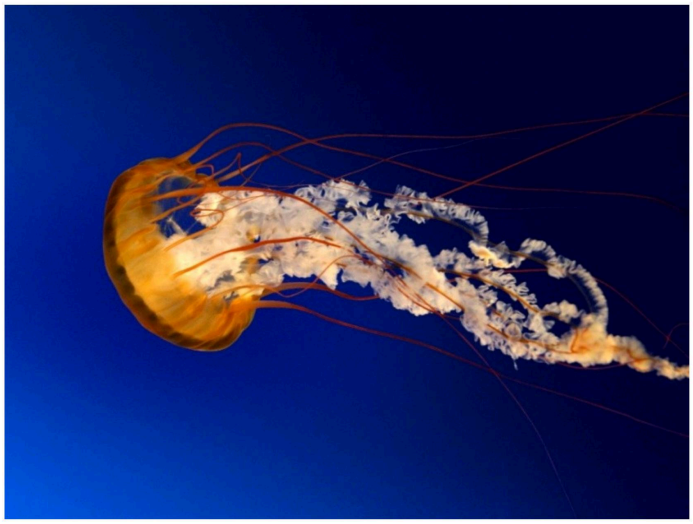
Result :

Thus the program has been executed and verified successfully using opencv in python.

OUTPUT:



IMAGE



VIDEO SCREENSHOT

EXPERIMENT -02 :

PERFORMING BASIC OPERATIONS ON AN IMAGE

Resize ,Color Shape ,Blending an image , Subtraction , Pixel manipulation

AIM:

To performing basic operations on an image using Opencv in python

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing basic operations on an image

Step 4: Execute code

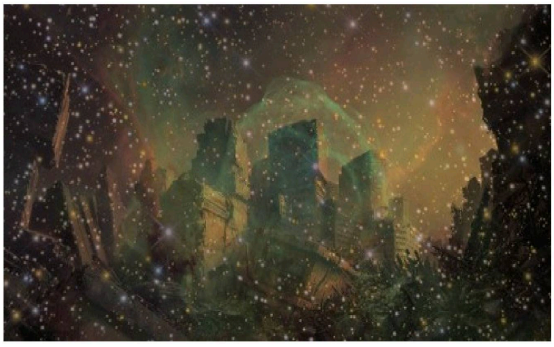
Python code:

```
import cv2
import numpy as np

Add or Blending an image  :

# path to input images are specified and
# images are loaded with imread command
image1 = cv2.imread('input1.jpg')
image2 = cv2.imread('input2.jpg')

# cv2.addWeighted is applied over the
# image inputs with applied parameters
weightedSum = cv2.addWeighted(image1, 0.5, image2, 0.4, 0)

# the window showing output image
# with the weighted sum
cv2.imshow('Weighted Image', weightedSum)


# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

INPUT IMAGE 1

INPUT IMAGE 2

OUTPUT :

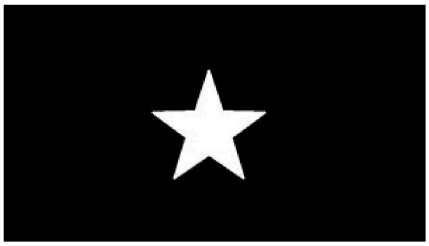


Subtraction:

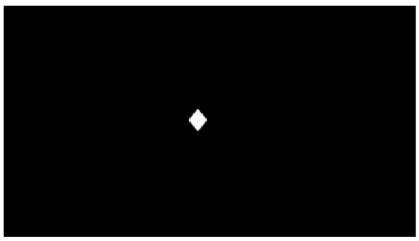
Syntax: `cv2.subtract(src1, src2)`

```
import cv2
import numpy as np

# path to input images are specified and
# images are loaded with imread command
image1 = cv2.imread('input1.jpg')
image2 = cv2.imread('input2.jpg')

# cv2.subtract is applied over the
# image inputs with applied parameters
sub = cv2.subtract(image1, image2)

# the window showing output image
# with the subtracted image
cv2.imshow('Subtracted Image', sub)

# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

INPUT IMAGES:

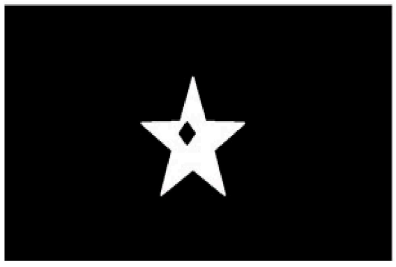


INPUT IMAGE 1



INPUT IMAGE 2

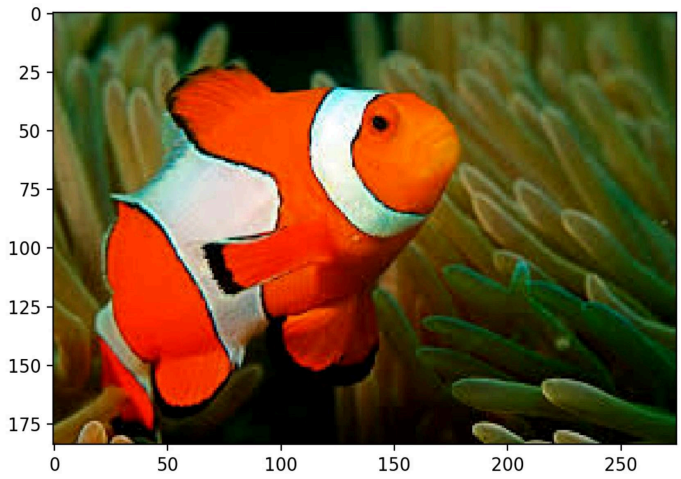
OUTPUT (SUBTRACTION):



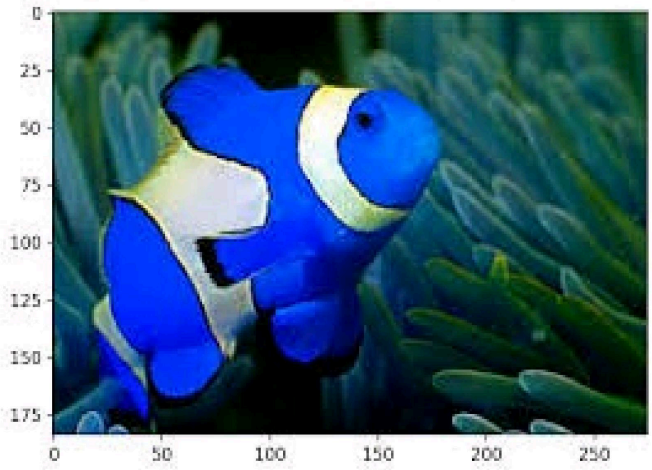
COLOR SHAPE:

```
import cv2
 import matplotlib.pyplot as plt
 import numpy as np
 nemo = cv2.imread('./images/nemo0.jpg')
nemo =cv2.cvtColor(nemo, cv2.COLOR_BGR2RGB)
 plt.imshow(nemo)
 plt.show()
```

Input image:



Output image:



PIXEL MANIPULATION:

```
# import cv2 module

import cv2
```

Python code 1 :

```
# read the image

img = cv2.imread('image.png')

# shape prints the tuple (height,weight,channels)

print(img.shape)

# img will be a numpy array of the above shape

print(img)
```

Python code 2:

```
# read the image
img = cv2.imread('image.png')

# this is pixel of 0th row and 0th column
print(img[0][0])
```

python code 3:

```
# read the image
img = cv2.imread('image.png')
for i, row in enumerate(img):

  # get the pixel values by iterating
 for j, pixel in enumerate(img):
 if(i == j or i+j == img.shape[0]):

 # update the pixel value to black
 img[i][j] = [0, 0, 0]

# display image
cv2.imshow("output", img)
cv2.imwrite("output.png", img)
```

INPUT:

OUTPUTS :

PYTHON CODE 1:

```
(225, 225, 3)
[[[ 87 157  14]
 [ 87 157  14]
 [ 87 157  14]
 ...
 [ 87 157  14]
 [ 87 157  14]
 [ 87 157  14]]

[[ 87 157  14]
 [ 87 157  14]
 [ 87 157  14]
 ...
 [ 87 157  14]
 [ 87 157  14]
 [ 87 157  14]]

...

[[ 72 133   9]
 [ 72 133   9]
 [ 72 133   9]
 ...
 [ 87 157  14]
```

```
 [ 87 157  14]
 [ 87 157  14]]


[[ 72 133   9]
 [ 72 133   9]
 [ 72 133   9]
 ...
 [ 87 157  14]
 [ 87 157  14]
 [ 87 157  14]]]
```

PYTHON CODE 2:

```
[  87  157   14]
```

PYTHON CODE 3:

EXPERIMENT -03:

PERFORMING GEOMETRIC OPERATIONS ON IMAGE

AIM:

To performing a geometric operations on image

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing geometric operations on an image

Step 4: Execute code

PYTHON CODE:

```
import cv2
import numpy as np
```

ROTATION:

```
im1=cv2.imread("C:/Users/SN56.CSE2-25/Downloads/galaxy.jpg")
shp=im1.shape
print (shp)
height,width=im1.shape[:2]
print(height)
print(width)
cent=(height/2,width/2)
cent
rotation_matrix=cv2.getRotationMatrix2D(center=cent,angle=25,scale=0.3)
print(rotation_matrix)
rotated_im=cv2.warpAffine(src=im1,M=rotation_matrix,dsize=(width,height))
cv2.imshow('Orginal image',im1)
cv2.waitKey(0)
cv2.imshow('Rotated image',rotated_im)
cv2.waitKey(0)
```

cv2.destroyAllWindows()

TRANSLATION MATRIX:

```
tx,ty=width/2,height/2
tx
ty
```

Translation_matrix=np.array([[2,0,tx],[1,5,ty]])

Translation_matrix

Trans_im=cv2.warpAffine(src=im1,M=rotation_matrix,dsize=(width,height))

```
cv2.imshow('Orginal image',im1)
cv2.waitKey(0)
cv2.imshow('Transimage',Trans_im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

ROTATION➔

```
(1280, 980, 3)

1280

980
```

(640.0, 490.0)

```
[[ 2.71892336e-01  1.26785479e-01  4.03864020e+02]
 [-1.26785479e-01  2.71892336e-01  4.37915462e+02]]
```

IMAGE OUTPUT:

TRANSLATIONAL→

```
490.0

640.0

array([[  2.,   0., 490.],
       [  1.,   5., 640.]])
```

IMAGE OUTPUT:

EXPERIMENT -04:

PERFORMING DIFFERENT IMAGE THRESHOLDING TECHNIQUE IN OPEN-CV

AIM:

To performing different image thresholding technique in open-cv

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing different image thresholding technique in open-cv

Step 4: Execute code

PYTHON CODE:

```
import cv2
```

PYTHON CODE 1 :

```
# Loading the image named test.jpg
img = cv2.imread(r"ex2.jpg")
# Converting color mode to Grayscale
# as thresholding requires a single channeled image
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('Image', img)
```

PYTHON CODE 2 :(**Binary Thresholding)**

```
# Thresholding the image placing 127 intensity level as threshold
# Pixel values below 127 would be changed to Black
# Pixel values above 127 would be changed to White (255)
ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# Displaying the output image
cv2.imshow('Binary Threshold', thresh)
```

PYTHON CODE 3:(**Binary – Inverse Thresholding)**

```
ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)

# Displaying the output image

cv2.imshow('Binary Inverse Threshold', thresh)
```

PYTHON CODE 4:(**Truncate Thresholding)**

```
ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_TRUNC)

# Displaying the output image

cv2.imshow('Truncate Threshold', thresh)
```

PYTHON CODE 5:(**Zero Thresholding)**

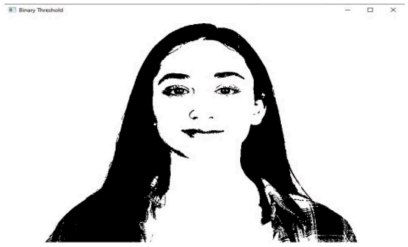
```
ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO)

# Displaying the output image

cv2.imshow('Truncate Threshold', thresh)
```

PYTHON CODE 6:(**Inverse Thresholding to Zero)**

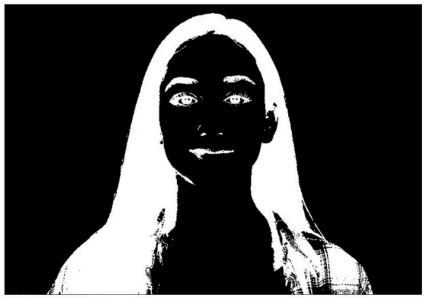
```
ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO_INV)

# Displaying the output image

cv2.imshow('Truncate Threshold', thresh)
```

INPUT:

OUTPUT:



BINARY THRESHOLD



BINARY-INVERSE THRESHOLD

TRUNCATE THRESHOLDING

ZERO THRESHOLDING

INVERSE THRESHOLDING TO ZERO

EXPERIMENT -05:

APPLICATION OF IMAGE FILTERS IN OPEN CV

AIM:

To performing an application of image filters in open cv

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing an application of image filters in open cv

Step 4: Execute code

PYTHON CODE:

AVERAGING FILTER:

Syntax of cv2.blur()

cv2.blur(image, ksize)

```
import cv2
import numpy as np

# image path
path = r'salad.jpg'

# using imread()
img = cv2.imread(path)

im1 = cv2.blur(img,(5,5))
im2 = cv2.boxFilter(img, -1, (2, 2), normalize=True)

cv2.imshow('image', np.hstack((im1, im2)))
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

GAUSSIAN FILTER:

Syntax of Gaussian Filter

cv2.GaussianBlur(src, ksize, sigma_x, dst, sigma_y, border_type)

```
import cv2
import numpy

# image path
path = r'cat.jpg'
```

```
# using imread()
img = cv2.imread(path)

dst = cv2.GaussianBlur(img,(5,5),cv2.BORDER_DEFAULT)

cv2.imshow('image', numpy.hstack((img, dst)))
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

MEDIAN FILTERING:

Syntax of Median Filter

cv2.medianBlur(image, ksize)

```
import cv2
import numpy

# image path
path = r'cat.jpg'
# using imread()
img = cv2.imread(path)
dst = cv2.medianBlur(img,7)

cv2.imshow('image', numpy.hstack((img, dst)))
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

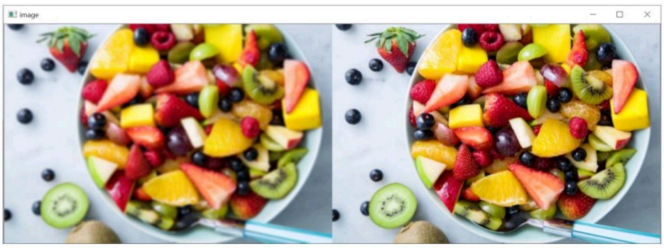
BILATERAL FILTER:

Syntax of Bilateral Filter

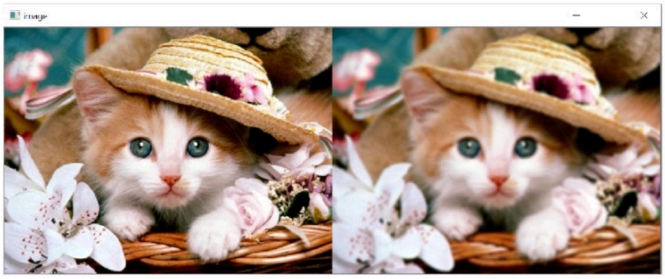
cv2.bilateralFilter(image, dst, d, sigmaColor, sigmaSpace)

```
import cv2
import numpy

# image path
path = r'salad.jpg'

# using imread()
img = cv2.imread(path)
dst = cv2.bilateralFilter(img, -1, 5, 5)

cv2.imshow('image', numpy.hstack((img, dst)))
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

OUTPUT:

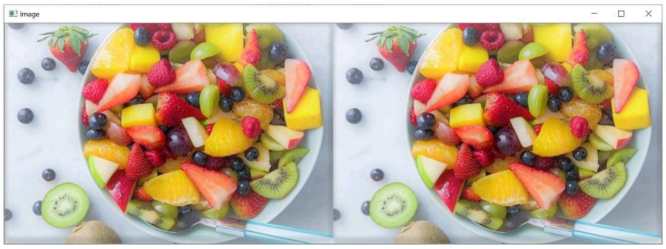


AVERAGING FILTER



GAUSSIAN FILTER

MEDIAN FILTERING



‘BILATERAL FILTER

EXPERIMENT -06:

PERFORMING DIFFERENT MORPHOLOGICAL OPERATION USING OPEN-CV

AIM:

To performing different morphological operation using open-cv

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing different different morphological operation using open-cv

Step 4: Execute code

PYTHON CODE:

EROSION:

```
# import the necessary packages
import cv2
import numpy as np
import matplotlib.pyplot as plt
# read the image
img = cv2.imread(r"Downloads\test (2).png", 0)
# binarize the image
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
# define the kernel
kernel = np.ones((5, 5), np.uint8)
# invert the image
invert = cv2.bitwise_not(binr)
# erode the image
erosion = cv2.erode(invert, kernel, iterations=1)
# print the output
plt.imshow(erosion, cmap='gray')
```

DILATION:

```
import cv2

# read the image

img = cv2.imread(r"path to image", 0)

# binarize the image

binr = cv2.threshold(img, 0, 255, cv.THRESH_BINARY+cv.THRESH_OTSU)[1]

# define the kernel

kernel = np.ones((3, 3), np.uint8)

# invert the image

invert = cv2.bitwise_not(binr)

# dilate the image

dilation = cv2.dilate(invert, kernel, iterations=1)

# print the output

plt.imshow(dilation, cmap='gray')
```

OPENING:

```
# import the necessary packages

import cv2

# read the image

img = cv2.imread(r"\noise.png", 0)

# binarize the image

binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel

kernel = np.ones((3, 3), np.uint8)

# opening the image

opening = cv2.morphologyEx(binr, cv2.MORPH_OPEN, kernel, iterations=1)
```

```
# print the output

plt.imshow(opening, cmap='gray')
```

CLOSING:

```
# import the necessary packages

import cv2

# read the image

img = cv2.imread(r"\Images\noise.png", 0)

# binarize the image

binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]


# define the kernel

kernel = np.ones((3, 3), np.uint8)

# opening the image

closing = cv2.morphologyEx(binr, cv2.MORPH_CLOSE, kernel, iterations=1)

# print the output

plt.imshow(closing, cmap='gray')
```

MORPHOLOGICAL GRADIENT:

```
# import the necessary packages

import cv2

# read the image

img = cv2.imread(r"path to your image", 0)

# binarize the image

binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel

kernel = np.ones((3, 3), np.uint8)

# invert the image
```

```
invert = cv2.bitwise_not(binr)

# use morph gradient

morph_gradient = cv2.morphologyEx(invert, cv2.MORPH_GRADIENT, kernel)

# print the output

plt.imshow(morph_gradient, cmap='gray')
```

TOP HAT:

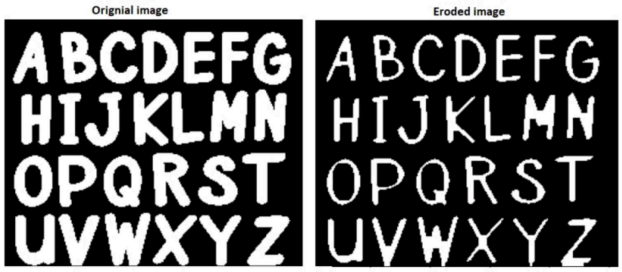
```
# import the necessary packages

import cv2

# read the image

img = cv2.imread("your image path", 0)

# binarize the image

binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel

kernel = np.ones((13, 13), np.uint8)

# use morph gradient

morph_gradient = cv2.morphologyEx(binr, cv2.MORPH_TOPHAT, kernel)

# print the output

plt.imshow(morph_gradient, cmap='gray')
```

BLACK HAT:

```
# import the necessary packages

import cv2

# read the image

img = cv2.imread("your image path", 0)

# binarize the image

binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
```

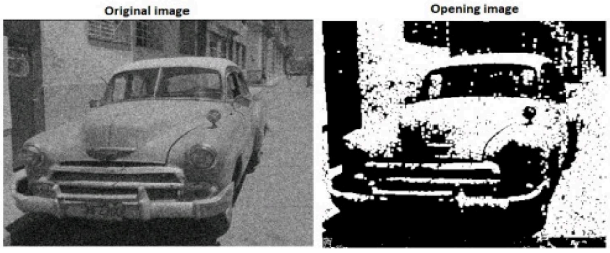
```
# define the kernel

kernel = np.ones((5, 5), np.uint8)

# invert the image

invert = cv2.bitwise_not(binr)

# use morph gradient

morph_gradient = cv2.morphologyEx(invert, cv2.MORPH_BLACKHAT, kernel)

# print the output

plt.imshow(morph_gradient, cmap='gray')
```

OUTPUT:

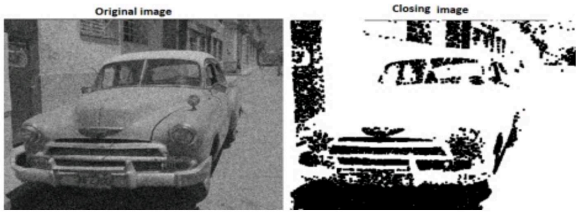


EROSION

DILATION

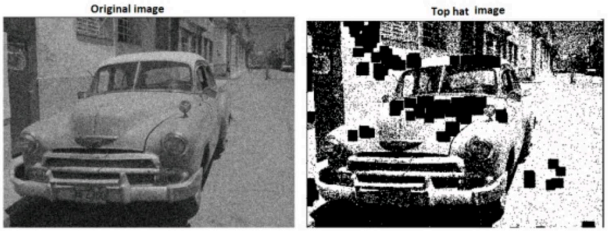


OPENING

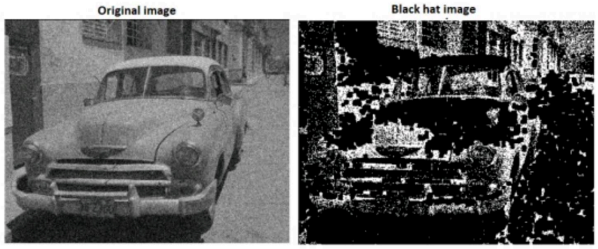


CLOSING

MORPHOLOGICAL GRADIENT



TOP HAT



BLACK HAT

EXPERIMENT -07:

IMAGE SEGMENTATION USING WATER SHED METHOD

AIM:

To performing image segmentation using watershed method

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing image segmentation using watershed method

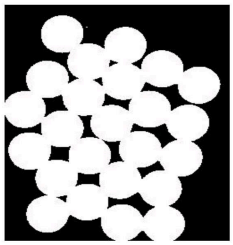
Step 4: Execute code

PYTHON CODE:

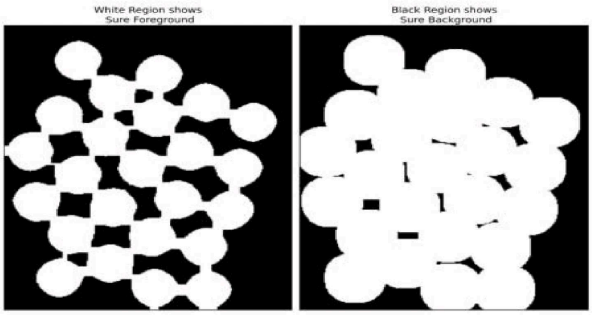
```
   import numpy as np
 import cv2 as cv
 from matplotlib import pyplot as plt

 img = cv.imread('coins.png')
 assert img is not None, "file could not be read, check with os.path.exists()"
 gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
ret, thresh = cv.threshold(gray,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)

# noise removal
kernel = np.ones((3,3),np.uint8)
opening = cv.morphologyEx(thresh,cv.MORPH_OPEN,kernel, iterations = 2)

# sure background area
sure_bg = cv.dilate(opening,kernel,iterations=3)

# Finding sure foreground area
dist_transform = cv.distanceTransform(opening,cv.DIST_L2,5)
ret, sure_fg = cv.threshold(dist_transform,0.7*dist_transform.max(),255,0)
# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv.subtract(sure_bg,sure_fg)
# Marker labelling
ret, markers = cv.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 = cv.watershed(img,markers)
img[markers == -1] = [255,0,0]
```

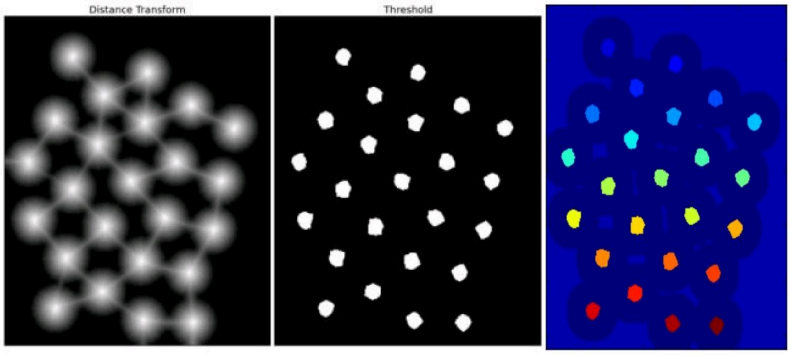
OUTPUT:

ORIGINAL IMAGE

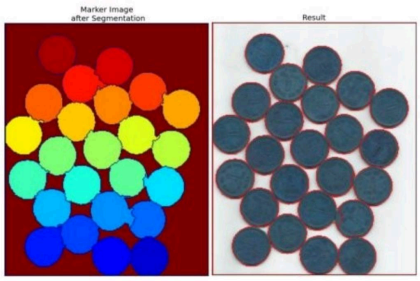


THRESH BINARY





MARKERS



MARKER IMAGE AFTER SEGMENTATION

EXPERIMENT -08:

IMAGE SEGMENTATION USING MEAN SHIFT METHOD

AIM:

To performing image segmentation using mean shift method

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing image segmentation using mean shift method

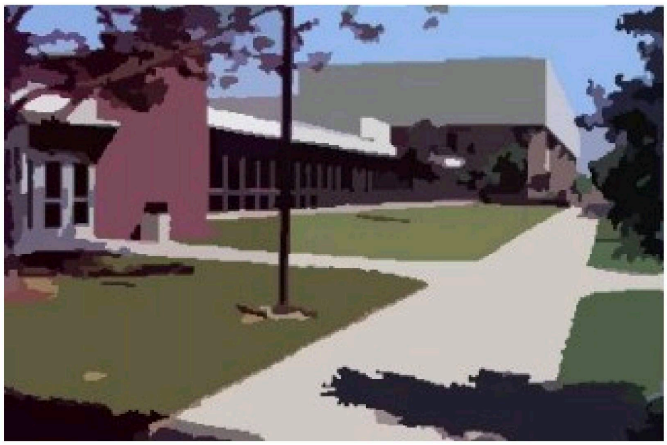
Step 4: Execute code

PYTHON CODE:

```
import numpy as np
import cv2 as cv
from sklearn.cluster import MeanShift, estimate_bandwidth
img = cv.imread("C:/Users/SN56.CSE2-25/Downloads/nature.jpg")

# filter to reduce noise

img = cv.medianBlur(img, 3)

# flatten the image
flat_image = img.reshape((-1,3))
flat_image = np.float32(flat_image)
# meanshift
bandwidth = estimate_bandwidth(flat_image, quantile=.06, n_samples=3000)
ms = MeanShift(bandwidth, max_iter=800, bin_seeding=True)
ms.fit(flat_image)
labeled=ms.labels_
# get number of segments
segments = np.unique(labeled)
print('Number of segments: ', segments.shape[0])
# get the average color of each segment
total = np.zeros((segments.shape[0], 3), dtype=float)
count = np.zeros(total.shape, dtype=float)
for i, label in enumerate(labeled):
    total[label] = total[label] + flat_image[i]
    count[label] += 1
avg = total/count
avg = np.uint8(avg)
# cast the labeled image into the corresponding average color
res = avg[labeled]
result = res.reshape((img.shape))
# show the result
cv.imshow('result',result)
cv.waitKey(0)
cv.destroyAllWindows()
```

INPUT:

OUTPUT:



EXPERIMENT -09:

IMAGE SEGMENTATION USING CLUSTERING METHOD

AIM:

To performing image segmentation using clustering method

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing image segmentation using clustering method

Step 4: Execute code

PYTHON CODE:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline
```

PYTHON CODE1:

```
# Read in the image
image = cv2.imread('images/monarch.jpg')
# Change color to RGB (from BGR)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)
```

PYTHON CODE 2:

```
# Reshaping the image into a 2D array of pixels and 3 color values (RGB)
pixel_vals = image.reshape((-1,3))
# Convert to float type
pixel_vals = np.float32(pixel_vals)
```

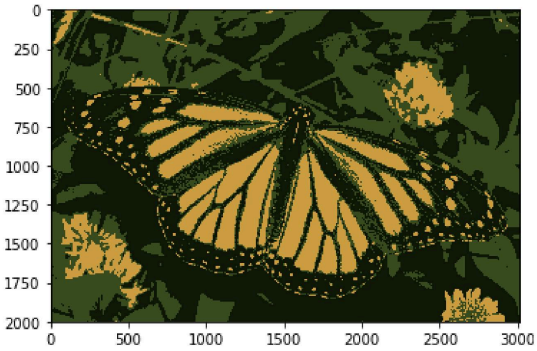
PYTHON CODE 3:

```
#the below line of code defines the criteria for the algorithm to stop running,
#which will happen is 100 iterations are run or the epsilon (which is the required accuracy)
#becomes 85%
```

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.85)

# then perform k-means clustering with number of clusters defined as 3

#also random centres are initially choosed for k-means clustering

k = 3

retval, labels, centers = cv2.kmeans(pixel_vals, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

# convert data into 8-bit values

centers = np.uint8(centers)

segmented_data = centers[labels.flatten()]

# reshape data into the original image dimensions

segmented_image = segmented_data.reshape((image.shape))

plt.imshow(segmented_image)
```

OUTPUT:

ORIGINAL IMAGE



CLUSTERED IMAGE

EXPERIMENT -10:

EDGE DETECTION IN OPEN – CV

AIM:

To performing edge detection in open- cv

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

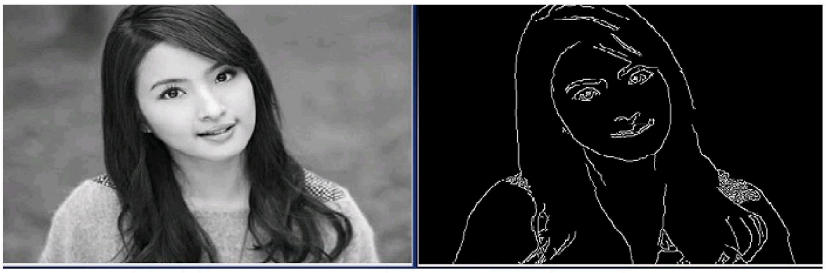
Step 3: Write the code for performing edge detection in open- cv

Step 4: Execute code

PYTHON CODE:

```
import cv2
def main():
 # read image
 gray_img = cv2.imread("./lenna.jpg", 0)
 cv2.imshow("img",gray_img)# Try masks of different sizes
 for n in range(1, 4):
  # use sobel operator
  kernel_size = 1+(n*2)
  x = cv2.Sobel(gray_img, cv2.CV_16S, 1, 0, ksize=kernel_size)
  y = cv2.Sobel(gray_img, cv2.CV_16S, 0, 1, ksize=kernel_size)  # Convert image format to uint8
  absX = cv2.convertScaleAbs(x)
  absY = cv2.convertScaleAbs(y)  # Add the results from both directions to form a complete contour
  dst = cv2.addWeighted(absX, 0.5, absY,0.5,0)  # display image
  cv2.imshow(f"{1+n*2}_x",absX)
  cv2.imshow(f"{1+n*2}_y",absY)
  cv2.imshow(f"{1+n*2}_x+y",dst)
  cv2.waitKey(0)
  cv2.destroyAllWindows()
```

OUTPUT:



EXPERIMENT -11:

FEATURE EXTRACTION USING HOG

AIM:

To performing feature extraction using HOG

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing feature extraction using HOG

Step 4: Execute code

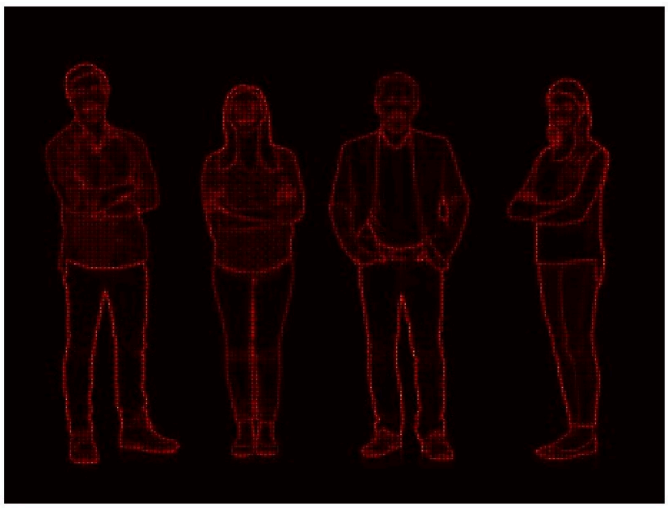
PYTHON CODE:

```
pip install scikit-image

from skimage.io import imread

from skimage.transform import resize

from skimage.feature import hog

import matplotlib.pyplot as plt

img=imread("C:/Users/SN56.CSE2-25/Downloads/gpp.jpg")

plt.axis("off")

plt.imshow(img)

print("img.shape")

resized_img=resize(img,(128*10,64*10))

plt.axis("off")

plt.imshow(resized_img)

plt.show()

print(resized_img.shape)

fd,hog_image
=hog(img,orientations=10,pixels_per_cell=(2,2),cells_per_block=(4,4),visualize=True,channel_axis=2)
```

```
#show image

print(fd.shape)

print(hog_image.shape)

plt.axis("off")

plt.imshow(hog_image, cmap='hot')

plt.shogrey
```

OUTPUT:

```
(1280, 640, 3)
```



EXPERIMENT -12:

FEATURE EXTRACTION USING SIFT IN OPEN-CV

AIM:

To performing feature extraction using SIFT in open-cv

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing feature extraction using SIFT in open-cv

Step 4: Execute code

PYTHON CODE:

```
import cv2

# Loading the image

img = cv2.imread('geeks.jpg')

# Converting image to grayscale

gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

# Applying SIFT detector

sift = cv.SIFT_create()

kp = sift.detect(gray, None)

# Marking the keypoint on the image using circles

img=cv2.drawKeypoints(gray ,kp ,img ,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imwrite('image-with-keypoints.jpg', img)
```

OUTPUT:

EXPERIMENT -13:

FEATURE EXTRACTION USING OPTICAL FLOW

AIM:

To performing feature extraction using optical flow

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing feature extraction using optical flow

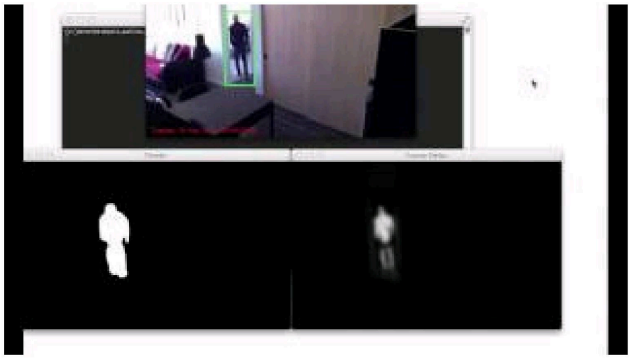
Step 4: Execute code

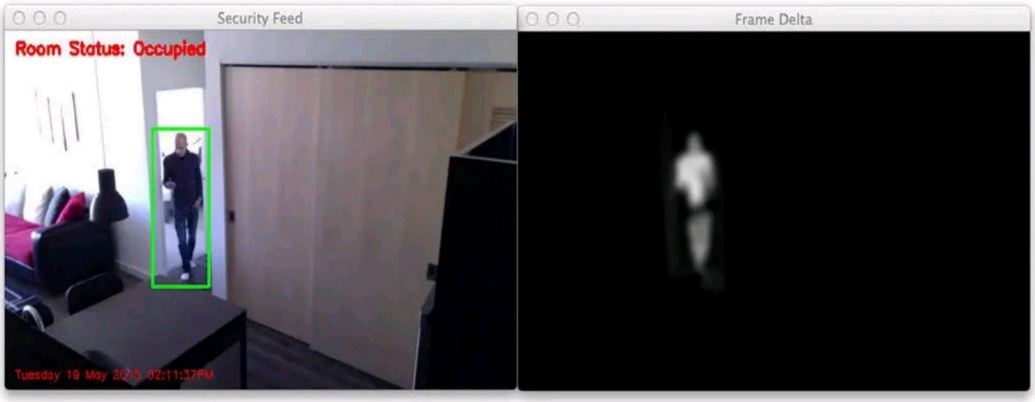
PYTHON CODE:

```
import cv2

# Initialize video capture

cap = cv2.VideoCapture(0)

# Initialize background subtractor

fgbg = cv2.createBackgroundSubtractorMOG2()

# Loop through frames

while True:

   # Capture frame-by-frame

   ret, frame = cap.read()

    # Apply background subtraction

   fgmask = fgbg.apply(frame)

    # Apply thresholding to remove noise

   th = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)[1]

    # Find contours of objects

   contours, hierarchy = cv2.findContours(th, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
# Loop through detected objects
    for contour in contours:
 # Calculate area of object
        area = cv2.contourArea(contour)
  # Filter out small objects
        if area > 100:
  # Get bounding box coordinates
            x, y, w, h = cv2.boundingRect(contour)
   # Draw bounding box around object
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
 # Display the resulting frame
    cv2.imshow('frame', frame)
 # Exit loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release video capture and destroy windows
cap.release()
cv2.destroyAllWindows()
```

OUTPUT:




Security Feed
Room Status: Occupied
Tuesday 19 May 2015 02:11:37PM
Frame Delta

EXPERIMENT -14:

FACE RECOGINITION USING HAARCASCADE METHOD

AIM:

To performing face recognition using Haarcascade method

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing face recognition using Haarcascade method

Step 4: Execute code

PYTHON CODE:

```
import cv2
import numpy as np
fd = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
ed= cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

#cap = cv2.VideoCapture(0)
img = cv2.imread('cricket.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = fd.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=1,minSize=(30, 10))
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
eyes = ed.detectMultiScale(gray,)
for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(img,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
cap = cv2.VideoCapture(0)
while(1):
    ret, fra = cap.read()
    gray2= cv2.cvtColor(fra,cv2.COLOR_BGR2GRAY)
    faces = fd.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=1,minSize=(30, 10))
    for (x,y,w,h) in faces:
            cv2.rectangle(fra,(x,y),(x+w,y+h),(0,255,0),2)
            eyes = ed.detectMultiScale(gray,)
    for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(fra,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)
    cv2.imshow('fra',fra)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT:

EXPERIMENT -15:

OBJECT RECOGNITION USING TEMPLATE MATCHING

AIM:

To performing object recgonitiion using template matching

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing object recognition using Template matching

Step 4: Execute code

PYTHON CODE:

```
import cv2
import numpy as np
cv2.__version__
img = cv2.imread('cat.jpg')
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
temp = cv2.imread('cat- Copy.jpg',0)
cv2.imshow('Source',img)
cv2.imshow('Template',temp)

if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
w, h = temp.shape[::-1]
mat = cv2.matchTemplate(img_gray,temp,cv2.TM_CCOEFF_NORMED)
threshold = 0.7
location = np.where( mat >=threshold )
for pt in zip(*location[::-1]):
    cv2.rectangle(img, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
cv2.imshow('Detected',img)
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

OUTPUT:

INPUT IMAGE

TEMPLATED IMAGE

MATCHED IMAGE

EXPERIMENT -14:

OBJECT TRACKING USING FEATURE DETECTION IN OPEN CV

AIM:

To performing object tracking using feature detection in open cv

ALGORITHM/PROCEDURE:

Step 1: Open new python file in jupyter notebook

Step 2: Install Open cv using pip command

Step 3: Write the code for performing object tracking using feature detection in open cv

Step 4: Execute code

PYTHON CODE:

```
import cv2
cap = cv2.VideoCapture("C:/Users/SALMA/Documents/E&I B1/Data/VID-20150928-WA0000.mp4")
def drawBox(img,bbox):
    x, y, w, h  = int(bbox[0]),int(bbox[1]),int(bbox[2]),int(bbox[3])
    cv2.rectangle(img,(x,y),((x+w),(y+h)),(255,0,255),1)
    cv2.putText(img,"PDS",(75,75),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,255,0),2)
#tracker = cv2.legacy.TrackerMedianFlow_create()
#tracker = cv2.legacy.TrackerMOSSE_create()
tracker = cv2.TrackerCSRT_create()
sucess, img = cap.read()
bbox = cv2.selectROI("PDS",img,False)
tracker.init(img,bbox)

while True:
    timer = cv2.getTickCount()
    sucess, img = cap.read()

    sucess, bbox = tracker.update(img)

    if sucess:
       drawBox(img,bbox)
```

```
    else:
        cv2.putText(img,"Lost",(75,75),cv2.FONT_HERSHEY_SIMPLEX,0.5,
(0,0,255),2)


    fps = cv2.getTickFrequency()/(cv2.getTickCount()-timer)
    cv2.putText(img,str(int(fps)),(75,50),cv2.FONT_HERSHEY_COMPLEX,0.7,
(0,0,255),2)


    cv2.imshow("Tracking",img)


    if cv2.waitKey(1) & 0xff == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

OUTPUT: