INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

VII Semester B.Tech in Information Technology

Report

HUMANOID ROBOTICS GROUP ASSIGNMENT (compulsory) (C2)

IMPLEMENTATION OF COMPUTER VISION IN ROBOTICS

By :-(Individual submission) .*No group

Chinmay Shravanbal Tayade (IIT2018138)

Table of contents

Abstract: This project is based on robotics vision using computer vision in Python. Generally, visuality/vision of a robot consists of hardware and some well trained machine learning, deep learning algorithms which allow the robots to utilize and process the existing visual data. For example we could consider like , if we want to command our robot to detect an object among different objects, so we will need to feed a accurate template of that object so that our robot can recognise that object easily. We can say that computer vision is just like an organ for visualisation for robots.

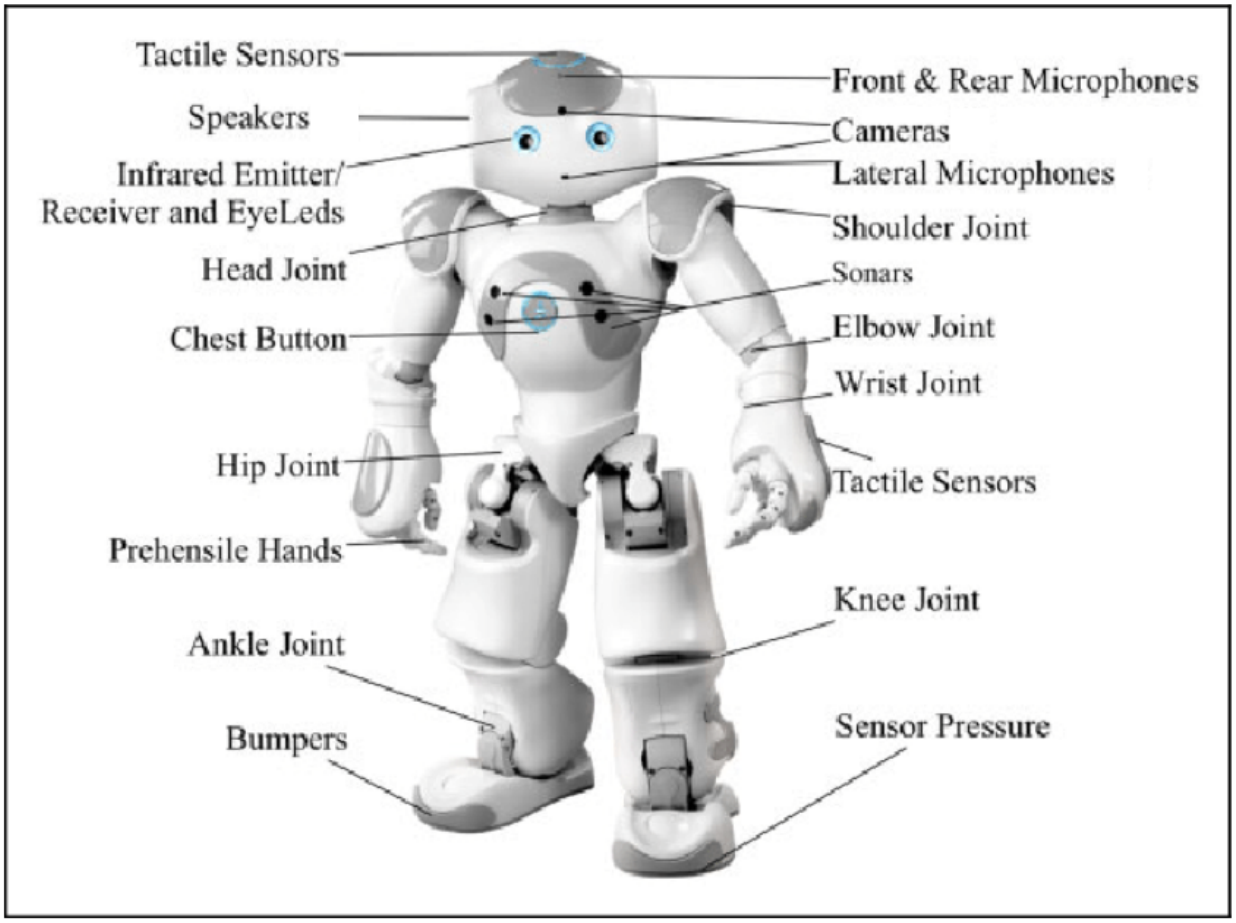
1. Introduction

While many people associate the word "robot" with Hollywood humanoid characters, robots are mainly mechanical devices programmed to perform specific repetitive tasks. People use them routinely to perform boring, dirty, or dangerous tasks that they don't want to do. It’s possible to program the robots to perform some tasks that would be too complicated for a human

to complete. In general, these robots are industrial in nature and are used for a number of purposes ranging from welding parts on auto assembly lines to interacting with humans in service areas. When you use the self-checkout machines at the grocery store, purchasing tickets from computerized ticketing systems, it might not seem like you're dealing with a robot. Obviously, robots have an impact on everyday life in the service sector.

DETAILED DIAGRAM OF A HUMANIOD ROBOT

This is NAO



-

2. Problem statement

Project-A

1.Take a Photo of any random scene containing some objects.
Use Template Matching and feature based template matching to detect objects.

2.Take a photo of any random scene containing objects. Detects bounding boxes for the objects in the scene using any object detection technique.
Use the midpoint of the bounding box to measure the angular orientation in vertical and horizontal direction of each object from the centerline of the camera.
Hint: Use Field of View values for reference.

3.Make a video of 15 seconds. Split it into different frames by taking frequencies of 10, 15, 20, 25 frames per second.

4.Make a video of 15 seconds with a moving camera while tracking an object.
Detect that object in each frame and draw the location of the center of the bounding box.

-

3. Language ,tools and model

- Python (3.8.3)notebook Using Conda environment system as well as Jupyter server on Google colab
- Open CV is a helpful library for the study of computer vision.
- Matplotlib.pylab is a module which provides a matlab type namespace in python
- Glob helps to return files paths related to a specific pattern in code.
- Numpy this module helps to deal with arrays and matrices easily.
- Jupyter Lab

4. Implementation and Results

Q1.

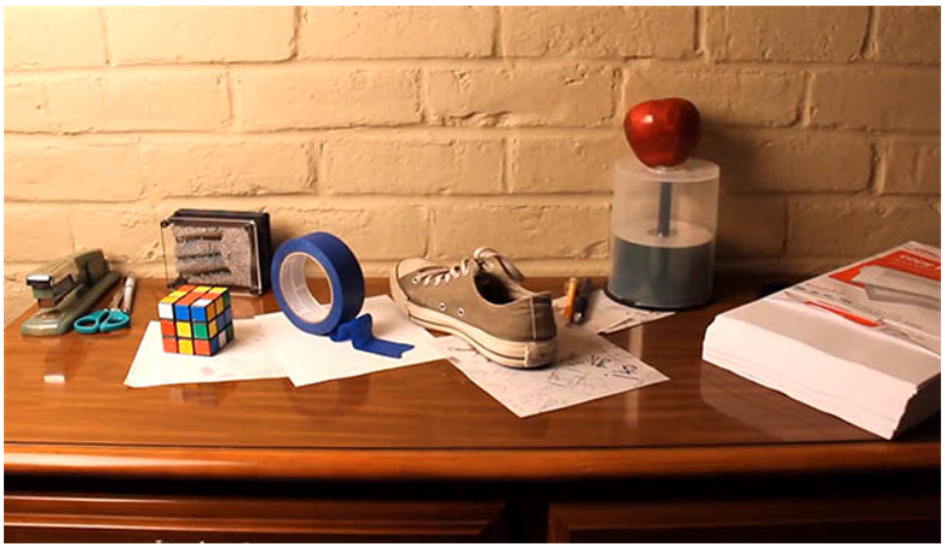
- Take a photo of any random scene containing some objects. Use Template Matching and feature based template matching to detect objects.
- So in this question we would like to implement simple template matching, so in the first part I have used the NUMPY library and CV 2 library.
- First I fed the test image and the template image to the computer then
- Converted the test image to grey
- Then I compared the test image with the template image and the output result was compared with the set threshold value (I used 0.6 as the threshold value)
- The detected output has a value greater than the threshold value it gives output as detected.
- In this program we used the CV2 match template function
- `Cv2.matchTemplate(grey_image, test_template, cv2.TM_CCOEFF_NORMED)` Inputs are - grey image of test image , template image , cv2.TM_CCOEFF_NORMED method was used to match the template image.

```
#Implementation of Template Matching
import numpy #this module helps to deal with arrays and matrices easily.

import cv2 #importing opencv library

#reading the testing image through open cv
colored_image = cv2.imread('test_image_f.jpg')
#converting the image to BGR to grey
grey_image = cv2.cvtColor(colored_image, cv2.COLOR_BGR2GRAY)

#reading testing template image
test_template = cv2.imread('test_template_f.jpg', 0)

width, height = test_template.shape[::-1]

resolution = cv2.matchTemplate(grey_image, test_template, cv2.TM_CCOEFF_NORMED)

# setting

threshold = 0.6

localization = numpy.where(resolution >= threshold)

for alpha in zip(*localization[::-1]):

    cv2.rectangle(colored_image, alpha, (alpha[0] + width, alpha[1] + height), (0, 253, 0), 1)

cv2.imshow('TEMPLATE DETECTED', colored_image),cv2.waitKey(0); cv2.destroyAllWindows(); cv2.waitKey(1)
```

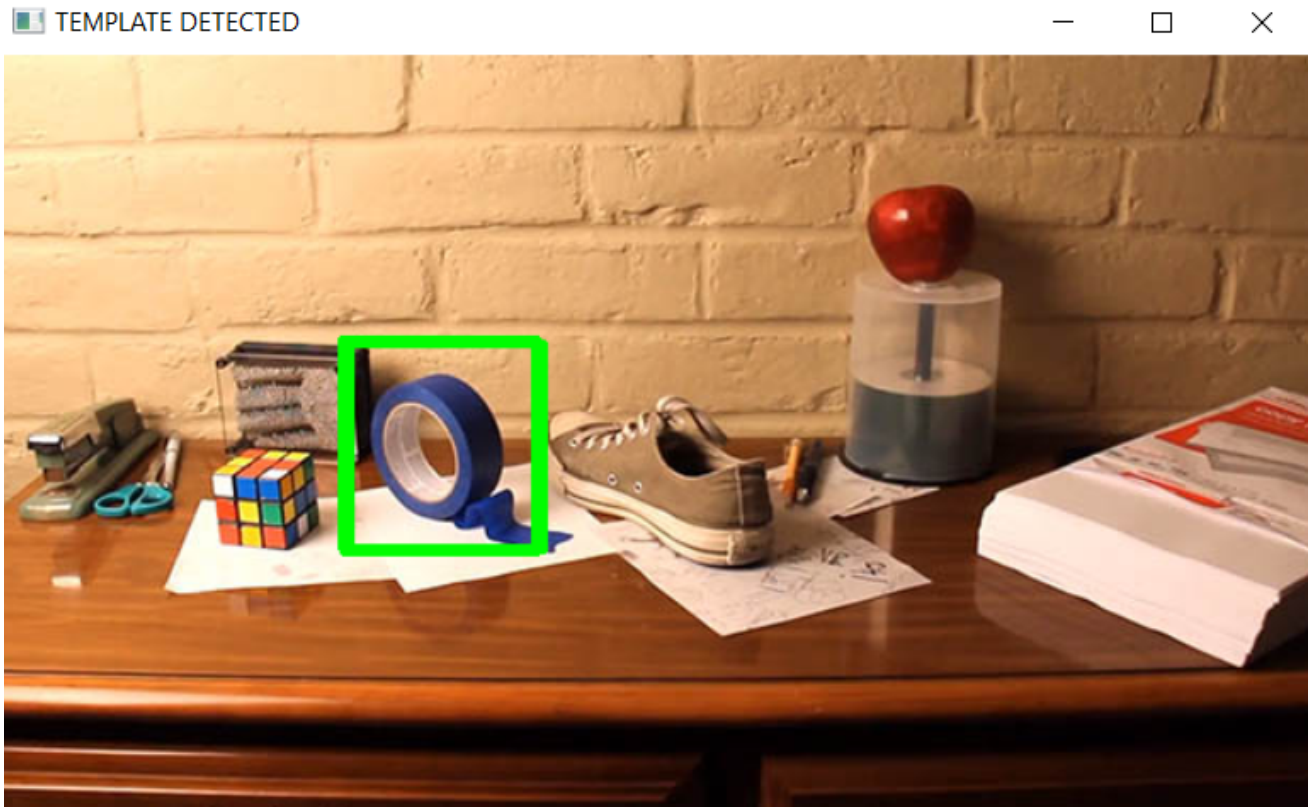
✓ 9.7s

1.Template

2.Test Image



-



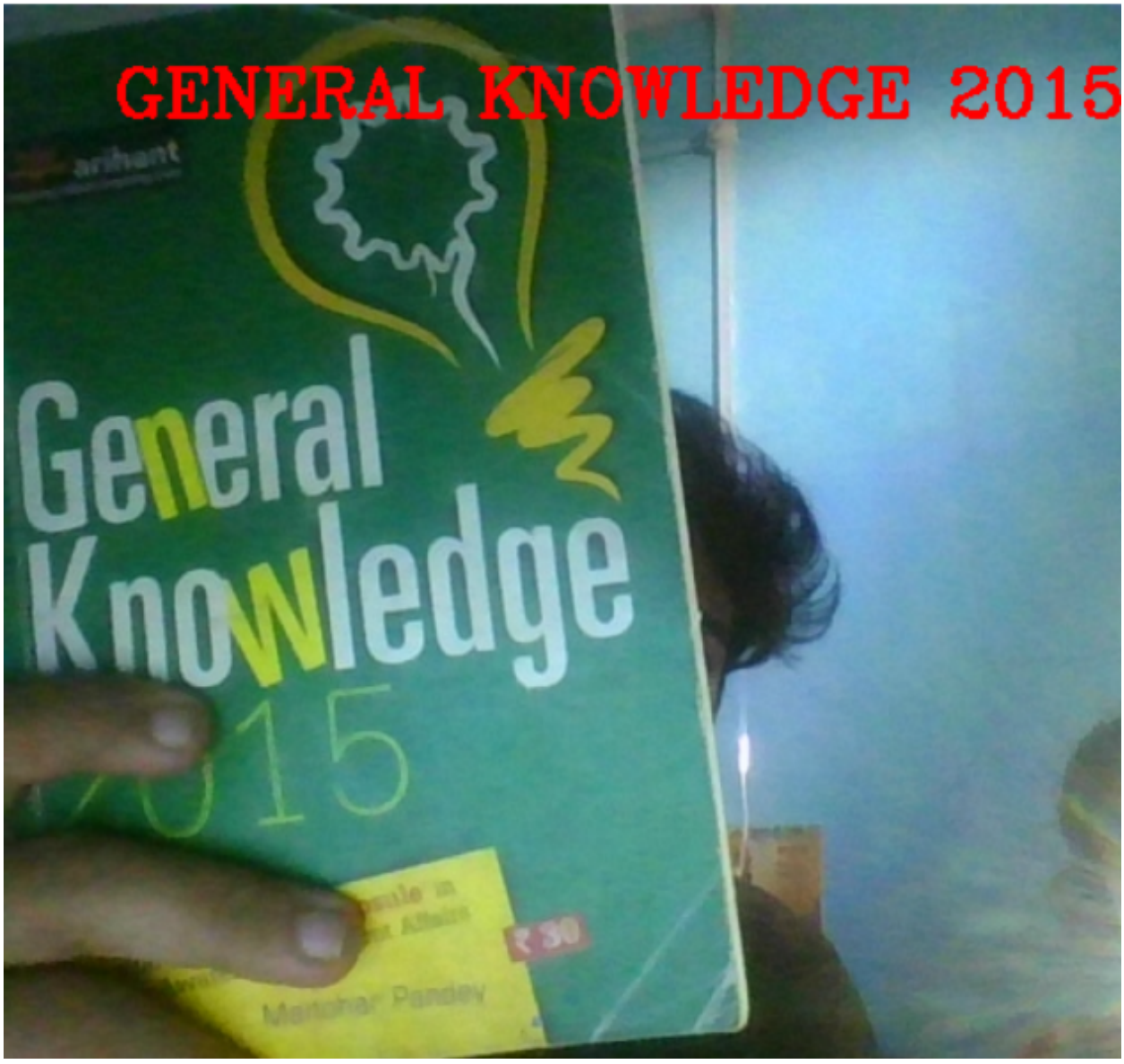
- (b). Implemented feature matching method
- In feature matching we compare two images that can have different light angles and rotation will be compared.
- I used the CV2 and NUMPY libraries
- As an example i used the following images

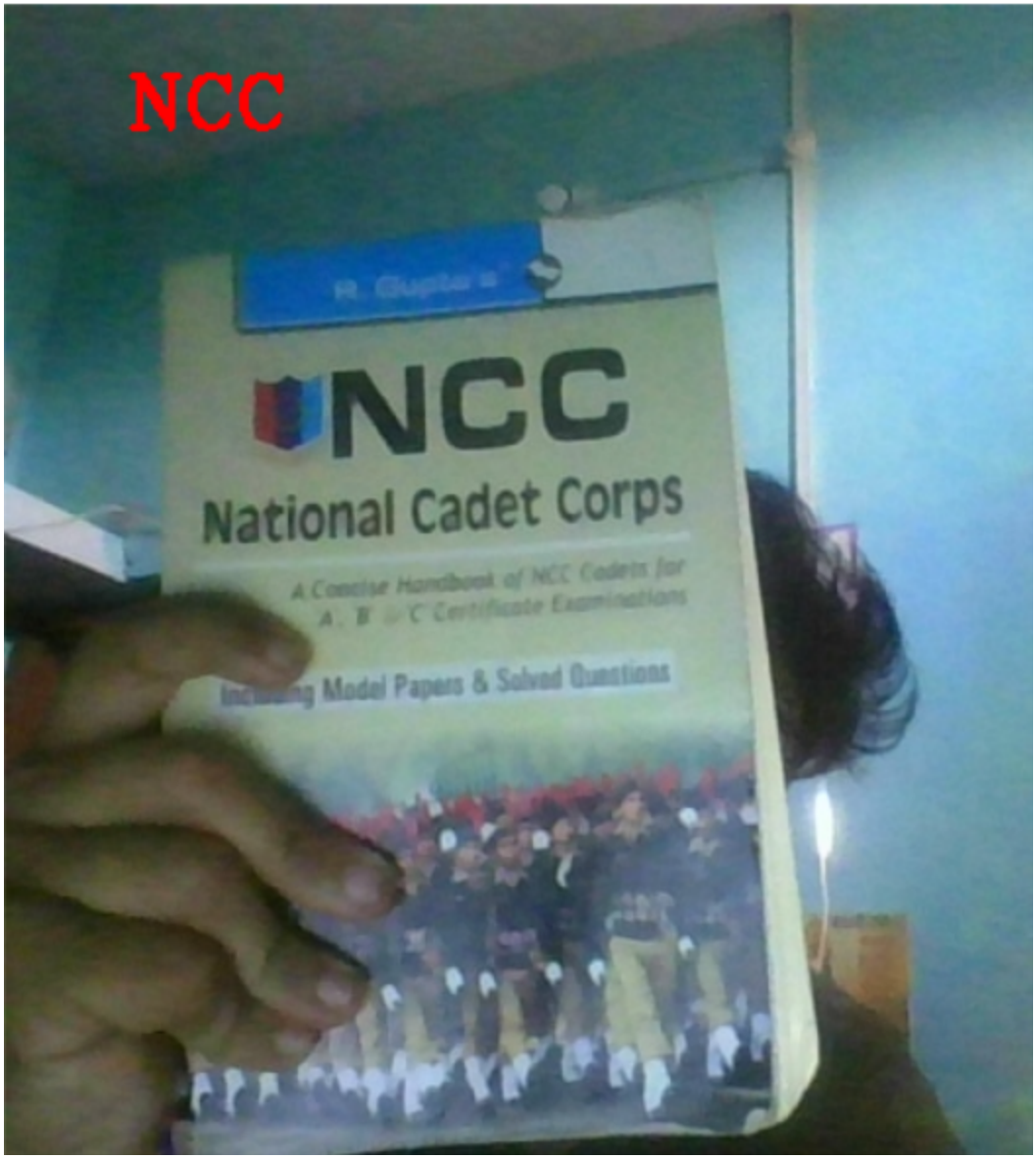
1.image 1

2. image 2

- In this matching we will use brute force matching we will do all the possible matches and sort out the best possible match.
- In this we will use ORB ORB. It is a combination of fast key point detector and brief descriptor including some extra features to enhance performance.
- FAST - Fast is an feature from accelerated segment test it is used in detection of image's features.
- BRIEF - Brief helps in the calculation of orientation and descriptives.
- ORB rotates the brief according to the keypoints.(ORB has by default 500 features)
- In this algorithm I took a test image and transformed it into a grey image image. Then I initiated the ORB and detected the keypoints of test image and sample image.
- Then calculated the descriptor belonging to both images to match the keypoints using brute force.
- And finally displayed the matched image
- 

- (c) So finally I have implemented feature-based template matching real time via video capture from a web cam as well as from existing video files.
- I had a list of playlists images of books , those images are stored in folder Q1 ,
- In this algorithm I merged the file names of all images existing in the Q1 folder into an array of lists.of books.
- Counted the no, of existing images and using feature based template matching I detected the file or playlist images from my webcam using ORB .
- Sample output :- AJDUST THE THRESHOLD VALUE TO AVOID ERRORS

- file_mv
- 
GENERAL KNOWLEDGE 2015
arihant
General
Knowledge
2015
₹ 30
Manohar Pandey

- file_mv
- 

Q2...Take a photo of any random scene containing objects. Detects bounding boxes for the objects in the scene using any object detection technique..Use the midpoint of the bounding box to measure the angular orientation in vertical and horizontal direction of each object from the centerline of the camera.

Hint: Use Field of View values for reference.

- Library is used cv2 and numpy
- Fed the image file consists of various objects in a frame using cv2.imread() .

- Turn the image file of objects to grey and then to binary
- Calculated the no, of contours in the image
- Find the area of all existing contours
- Restricting the area to be considered to avoid really small and huge objects
- Getting the parameters of the contours
- Finally displaying the output image enclosed within their bounding boxes and printing their calculated angular orientation with respect to the centre of object using cv.putText and saving the output file using cv,imwrite.
- OUTPUT FILE:-

Q3.Make a video of 15 seconds. Split it into different frames by taking frequencies of 10, 15, 20, 25 frames per second.

```
import numpy as np
import cv2

cap = cv2.VideoCapture('car-overhead-1.avi')

cv2.namedWindow('Display_frame',0)

cv2.resizeWindow('Display_frame',300,300)

fourcc = cv2.VideoWriter_fourcc(*'XVID')

out1 = cv2.VideoWriter('output1.avi', fourcc, 10.0, (300,300))

out2 = cv2.VideoWriter('output2.avi', fourcc, 15.0, (300,300))

out3 = cv2.VideoWriter('output3.avi', fourcc, 20.0, (300,300))

out4 = cv2.VideoWriter('output4.avi', fourcc, 25.0, (300,300))

while(cap.isOpened()):

    val_return, Display_frame = cap.read()

    if val_return:
        vidout=cv2.resize(Display_frame,(300,300))


        out1.write(vidout)

        out2.write(vidout)

        out3.write(vidout)

        out4.write(vidout)

        cv2.imshow('Display_frame',Display_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()

out1.release()

out2.release()

out3.release()

out4.release()

cv2.destroyAllWindows()
```

- Imported the required library
- Fed the sample video
- Adjusted the frame rates of video using cv2.videoWriter

-

- Saved the modified output as an AVI video file

Q4.Make a video of 15 seconds with a moving camera while tracking an object.

Detect that object in each frame and draw the location of the center of the bounding box.

- Captured the video file using the video capture , we can also use webcam for real time object tracking
- In this, we used the background substraction method from the open CV library
- Background sub. Helps us to remove existing background from the image which helps us to get the moving foreground easily from the background.
- Next we used numpy.ones() to squeeze or modify the existing dimensions and return a new array
- Tracked each frame of video
- Deduced the foreground from the background
- Reduce the noise
- Detected the index of the largest moving object (contour) and enclosing it inside the bounding boxes.
- Retrieve the centre of a moving object and the coordinates of its centre
- Finally, released the output file.
-