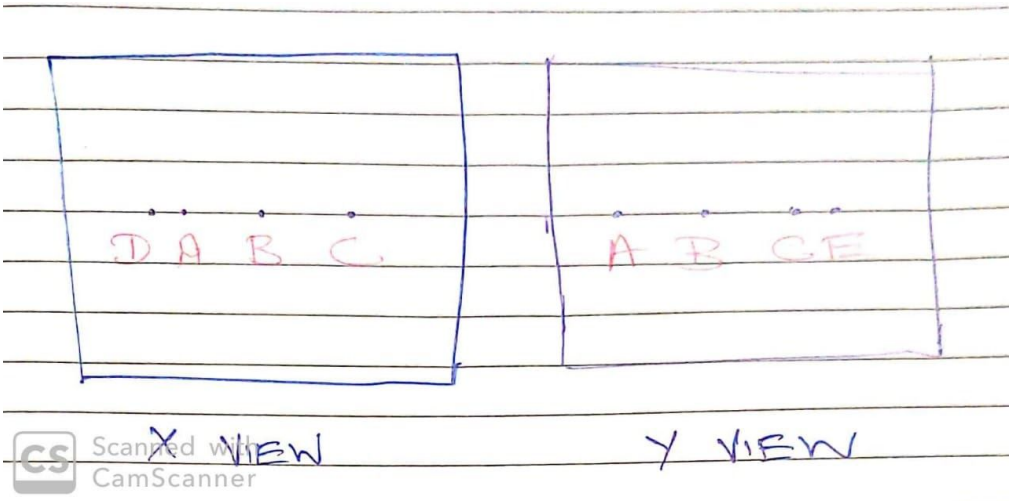
CS5100 HOMEWORK 7
08/04/2020
THEORY

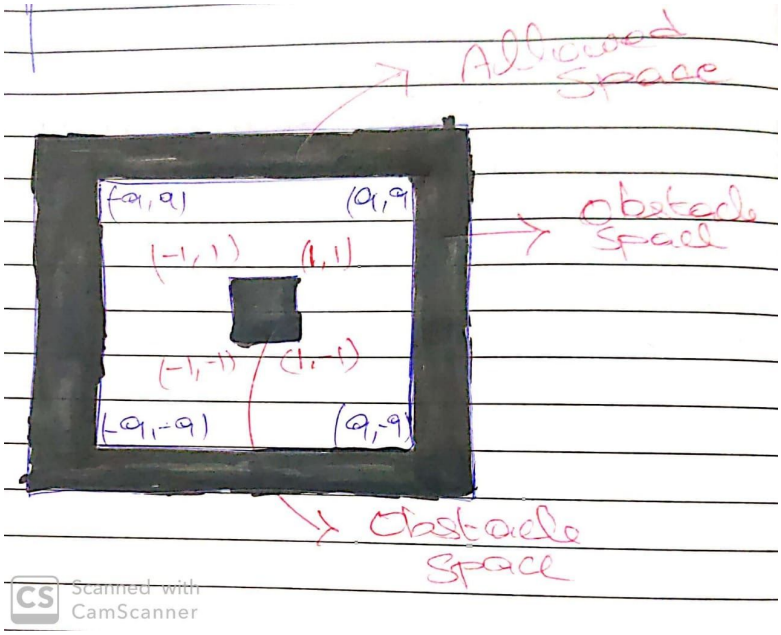
1. .
    a.
        i. Depth (in 3D is same as edges in 2d(separating boundaries))
            1. Height of the bowl (Depth of bowl)
            2. Length of the spoon (indicating the depth of the image)
            3. Food in Milk
        ii. Surface Orientation
            1. Orientation of bowl in table
            2. Corner of the table
        iii. Reflectance
            1. Shiny Spoon
            2. Shiny Milk
        iv. Illumination
            1. Shadow of the spoon projected on the table
            2. Shadow of bowl projected on the table
            3. Shadow of Food on the Milk

    b. A and C are equally separated from B (center). Since we have an opaque object D is not visible to Y and E is not visible to X and both D and E are are at greater depth compared to A, B, C

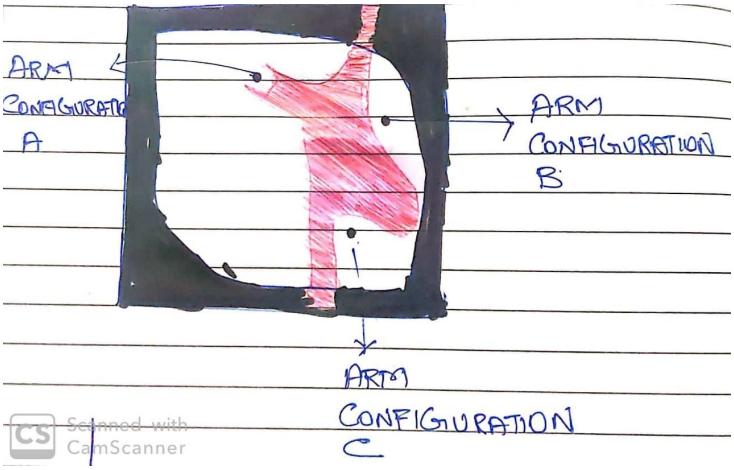


2. .

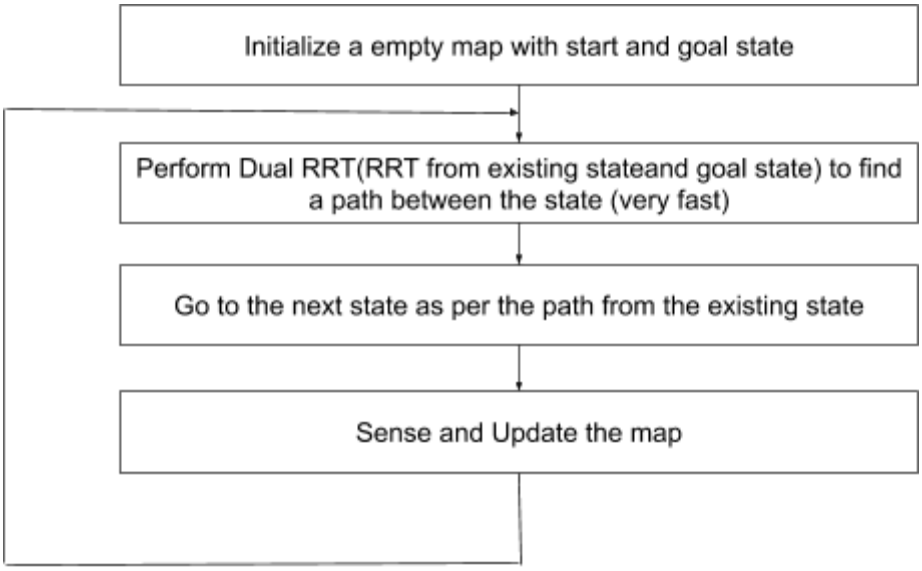
a. Since both blocks are side two we have to decrease the total length of x, y by 2 that is (-10,10) ⇒ (-9,9) to represent the manipulators as a point and since both these manipulators cannot be present at 0,0. A square of size 2 is drawn as an obstacle with 0,0 as center



b. *Hint:* Each arm configuration maps to a single point in configuration space



c. The design of the controller is given below

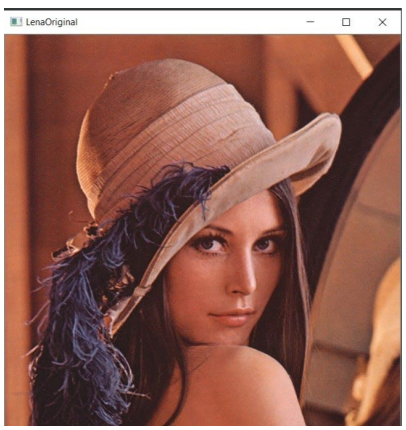


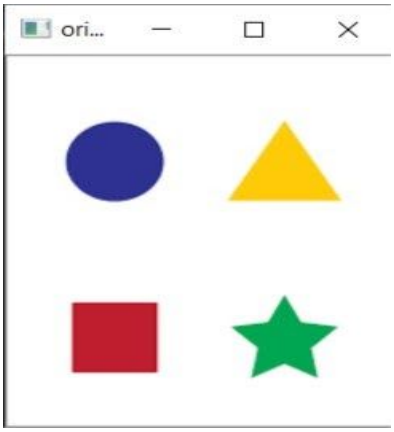
SInce RRT is probabilistically complete it will always give a path if one exists else will return there is no path existing

PROGRAMMING ASSIGNMENT

1. **Computer Vision - OpenCV**

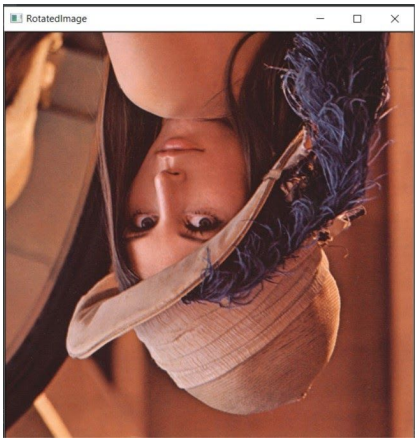
First Image is used for first 2 operations and 2nd image for remaining operations





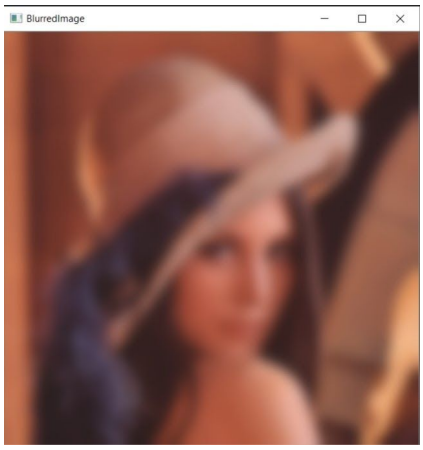
a. Rotating an image

Step1: Determine the center of the image so that we can rotate the image along the center
Step2: Now generate the rotation matrix from cv2.getRotationMatrix2D() 2X3 matrix which can be multiplied with the original image to get the rotated image
Step3 : cv2.warpAffine() is used to perform rotation of image maintaining the same size by multiplying the image with the rotation matrix generated.



b. Smoothing an image

(**Image Smoothing**) **Image** blurring is achieved by convolving the **image** with a low-pass filter kernel. It is useful for removing noise.
cv2.GaussianBlur() is used for blurring the image which accepts the kernel size, and image for blurring it.

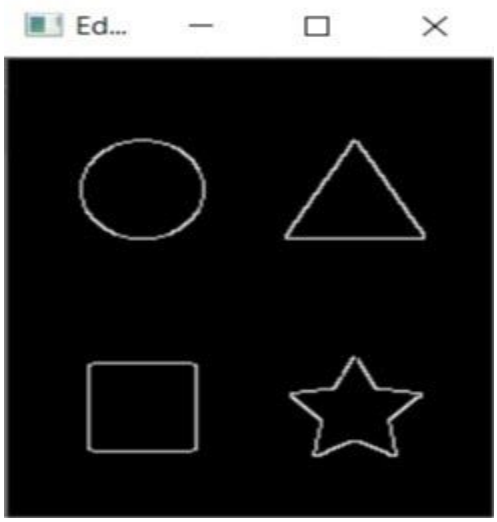


c. Converting an image to grayscale

Grayscale Image is 1Dimension whereas other images have multiple dimensions so we can convert images to grayscale for easier processing.
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) Converts RGB to Gray.

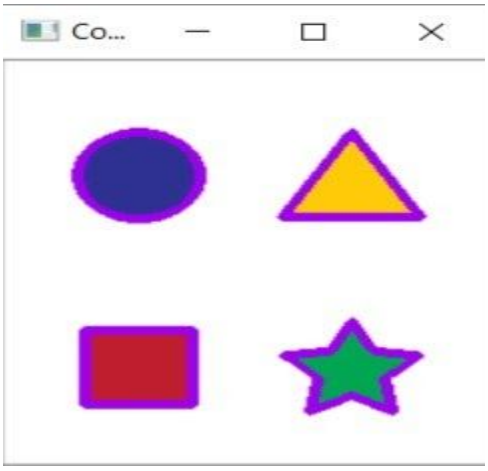
d. Edge detection

Edge detection is an image processing technique for finding the boundaries of objects within images.
Apply Canny Filter to detect the edges : cv2.Canny takes the gray scale image as input along with the threshold values(minimum and maxmum) to detect the edges.

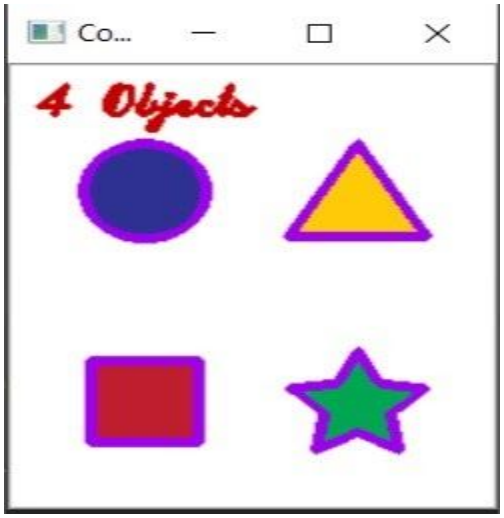


e. Detecting and Drawing Contours

Cnts=cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)is used to find the contours in the thresholded image(segmented image)[finding white from black]and imutils.grab_contours(cnts) stores the found contours in a variable and cv2.drawContours()draws the contour to each detected object.



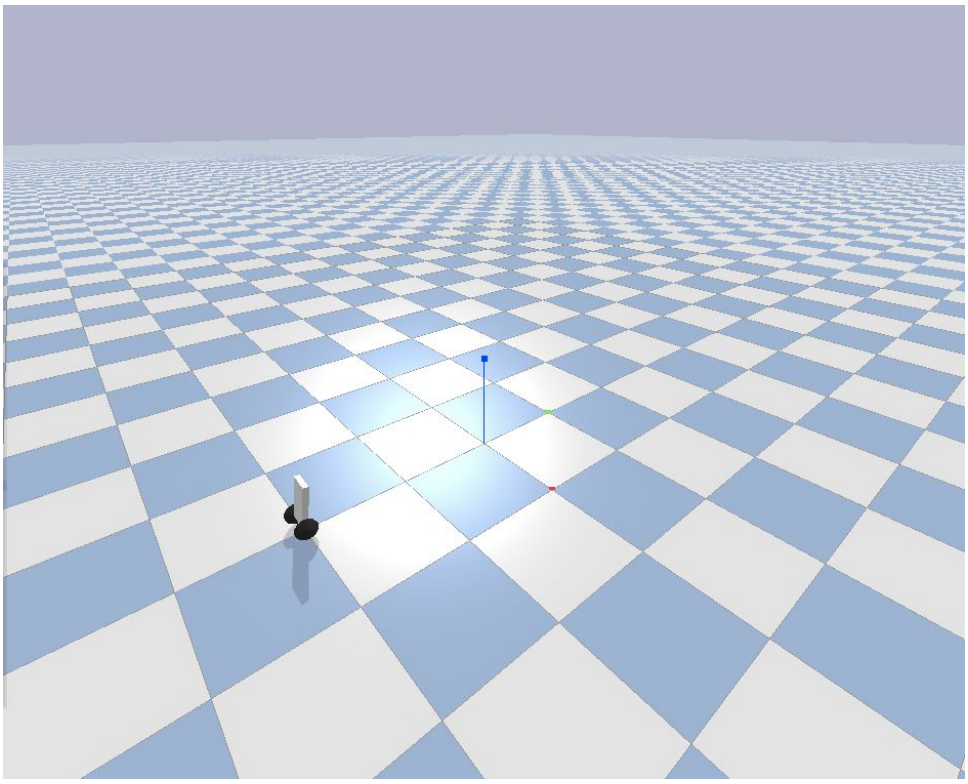
f. Counting Contours : count the detected contours



2. **Build a Balancing Bot (requires python 3.6.8)**

   Made the bot to randomly move in various velocity and fall or fall down immediately with weighted probability

   Increased the length of discrete observations from 9 to 13 and timesteps to 0.02



```
| % time spent exploring  | 2        |
| episodes                | 620      |
| mean 100 episode reward | 19.6     |
| steps                   | 43002    |
-------------------------------------
Saving model due to mean reward increase: 17.4 -> 24.1
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 630      |
| mean 100 episode reward | 30.7     |
| steps                   | 58012    |
-------------------------------------
Saving model due to mean reward increase: 24.1 -> 31.6
Saving model due to mean reward increase: 31.6 -> 37.8
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 640      |
| mean 100 episode reward | 40.7     |
| steps                   | 73022    |
-------------------------------------
Saving model due to mean reward increase: 37.8 -> 45.2
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 650      |
| mean 100 episode reward | 52.3     |
| steps                   | 88032    |
-------------------------------------
Saving model due to mean reward increase: 45.2 -> 53.4
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 660      |
| mean 100 episode reward | 54.8     |
| steps                   | 93950    |
-------------------------------------
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 670      |
| mean 100 episode reward | 55.2     |
| steps                   | 96540    |
-------------------------------------
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 680      |
| mean 100 episode reward | 55.5     |
| steps                   | 98659    |
-------------------------------------
-------------------------------------
| % time spent exploring  | 2        |
| episodes                | 690      |
| mean 100 episode reward | 55.5     |
| steps                   | 99829    |
-------------------------------------
Restored model with mean reward: 53.4
Saving model to balance.pkl
numActiveThreads = 0
stopping threads
Thread with taskId 0 exiting
Thread TERMINATED
destroy semaphore
semaphore destroyed
destroy main semaphore
main semaphore destroyed
(balance_bot) SUDHARSHANs-MacBook-Air:balance-bot sudharshan$
```