Northeastern University

CS 5100   Foundations of Artificial Intelligence

Homework and PA 7

NAME: **PARSHVA TIMBADIA**

EMAIL: timbadia.p@northeastern.edu

NUID: **001091783**

Q1>

Now, we can provide the example for the **Depth as follows:**
1. Between the bowl and the spoon.
2. Between the bowl and the table(greater depth).
3. Between the spoon and the table.

Now, we can provide the example for the **Surface Orientation as follows:**

1. We can clearly differentiate the curvature of the spoon when considered using different orientation.
2. The top layer of the fruits also shows this property.

Now, we can provide the example for the **Reflectance as follows:**

1. The continuity of the spoon is broken due to the light falling on it and hence reflectance.
2. Also, similar effect can be seen between the table and other elements since the continuity is broken by them.

Now, we can provide the example for the **Illumination as follows:**

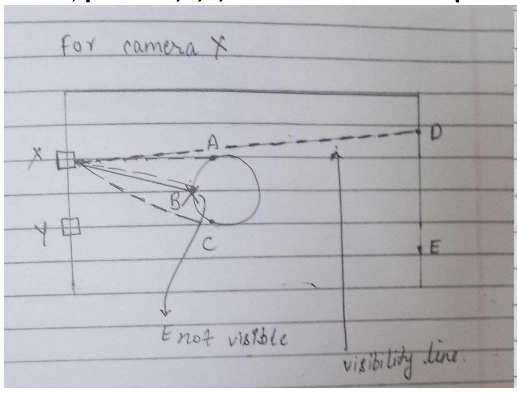
1. The shadow of the bowl represented on the spoon.
2. The shadow of the spoon on the table.
3. The shadow of the fruits which are elevated on the top at the background layer, ie other fruits.

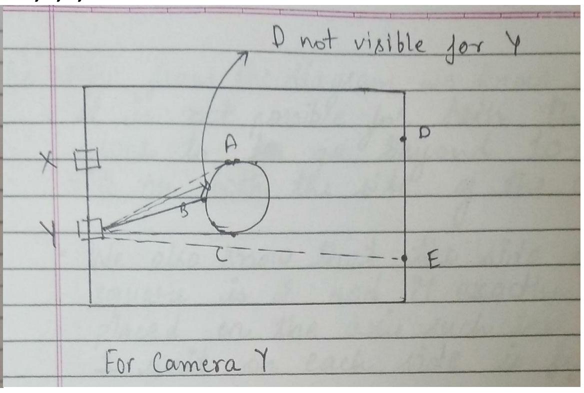
B>

Now when the view is considered as Stereo we get the chance to even measure the depth of the points located. For both the cameras all the points A,B,C can be seen. We can also determine that the point B is the nearest of all and A and C are at the equal distant from point B.

It is also interesting to note that the points D and E can not be covered by both the cameras. So the camera X would only be able to cover D , the vision of X is being limited by the circular object in between and same goes with the camera Y, where only point E is visible. Apart from all of the above, we can also conclude that the D is more closer to camera X rather than camera Y and reverse is also true.

Hence, **points A,B,C,D visible from X and points A,B,C,E visible from Y.**



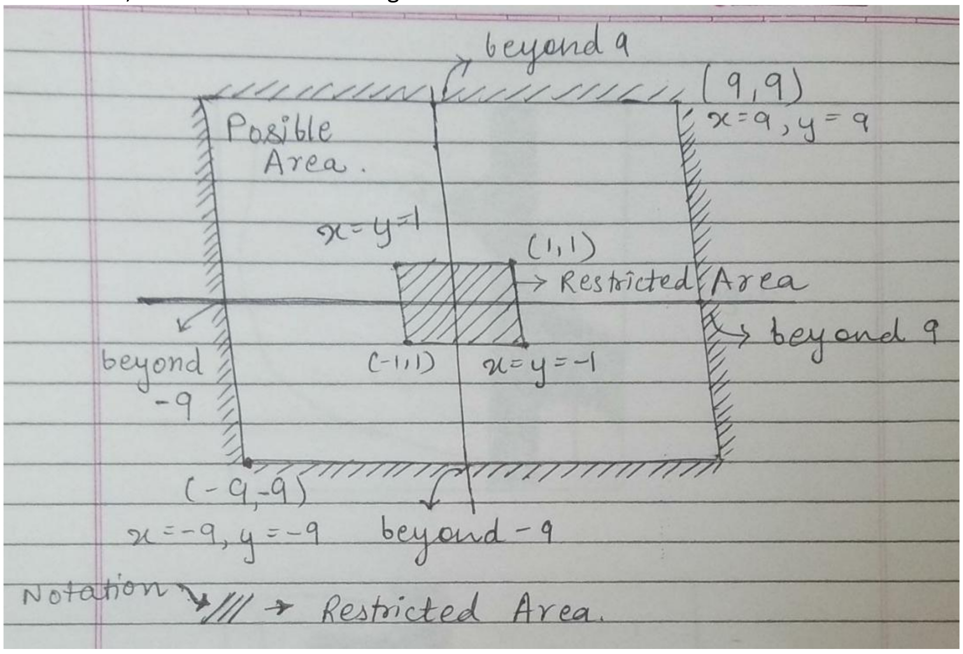


C>

Now, from this given diagram we know that it is not possible for both of the blocks to go beyond 10 and -10 on either side of the axis. We are also given that the side of the square is 2 and it is exactly placed on he axis such that only 1 unit if left on the either side.

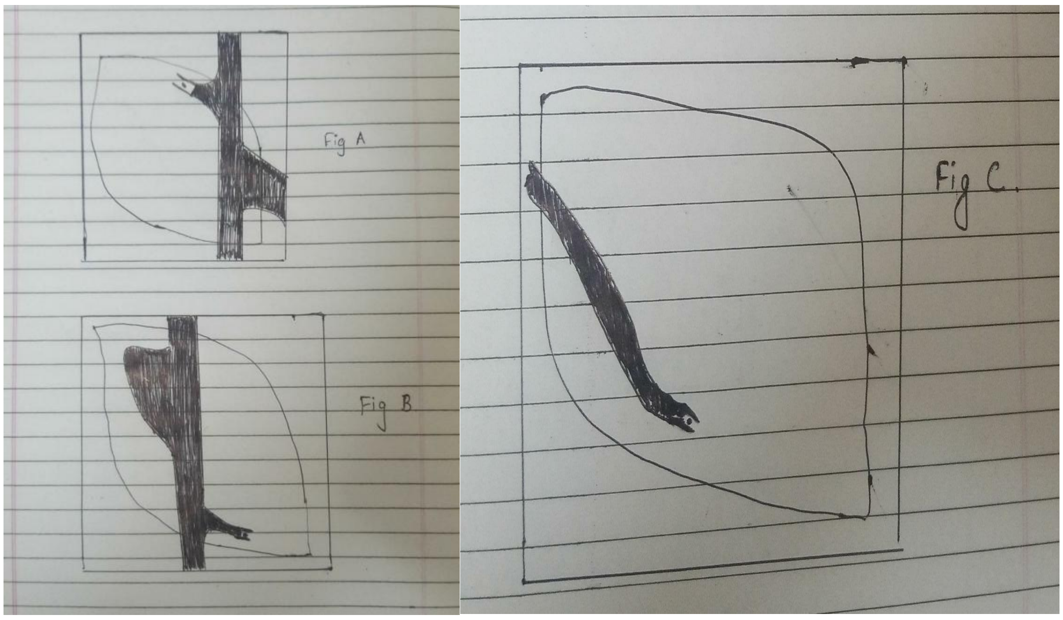
Hence, the squares **cannot be at +1 and -1 position together,** else they would collide.

Below here, I have attached the diagram with the restricted zone.



D>

Below are the attached diagram for the given question.

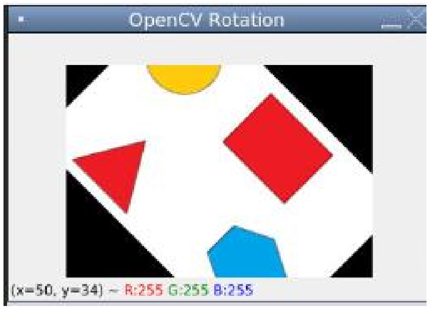


E>
To begin with, lets consider a robot's map an empty map and all the states reachable from the current state. Now we will find the closest path from the current position to the goal position using A* search with admissible heuristic and execute the steps. Now whenever the robot face some obstacle it is keep it in the memory of map and make changes to the map accordingly. The robot will showcase the result successful, if it was able to reach goal state, else failure ie no path exists.

PA1:

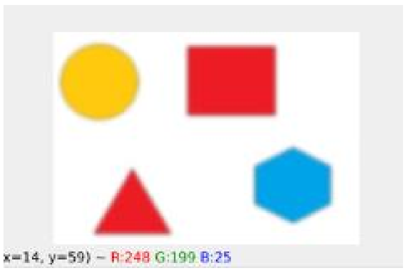
NOTE: Below are the attached images of the code.

Rotating Image :



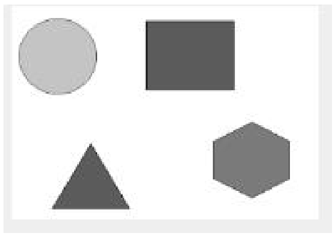
Well first we are fixing the center and then the image is tilted by certain degree in the clockwise direction, without moving the center.

Smoothing Image:



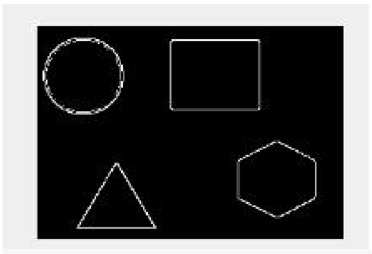
Every image contains noise and it is imperative to remove it for getting the better results. This is because after the noise is removed the algorithm develops a better understanding about the useful contents. This can be done using OpenCv using a function called Gaussian Blur. Here, there is a kernel where we have to provide value for the pixels near to it are provided more weight compared to those who are away from it and the average is calculated which get reflected back on the screen as a new value.

Conversion to Gray Scale:



Processing colored images is more difficult as compared to the singular colored object, this is because when we convert the image to gray scale less information is passed on. THis can be done using this module cvtColor, which helps to convert an image in a singular color format.

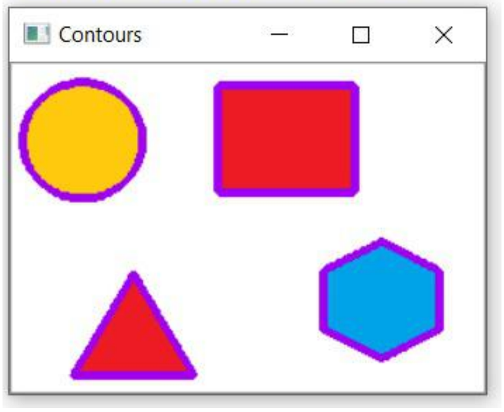
Edge Detection:



Well there occur many events where we need to segregate the objects from the other objects and hence we need this particular module. This helps in demarcating the objects. This is done using the module canny, multistage algorithm and works by reducing the effect of the noise. We need to pass gray scaled image and minimum Threshold and max Threshold values.

Detecting and Drawing Contours:

```
width=269, height=187, depth=3
```



Now the contours in the image can be detected by using cv2.findcontours. The algorithm help is differentiating all the pixels from the background and there by helping to identify the contours, which is done by passing thresh.copy() argument.
In the end we use cv2.drawcontours to help up provide the shape for the contours.