SENSOR BASED ROBOTICS

COURSEWORK PROJECT REPORT

Title: **AUTONOMUS OBJECT PICKER**

Name: Dheenu Kasinathan
N-ID: N-10498050
Mail-Id: dk3250@nyu.edu
Instructor: Dr. Prashanth Krishnamurthy

SUMMARY:

The advent of Control System and Robotics has resulted in the phenomenal transformation of our lifestyle. Its adaptive nature has resulted in increased accuracy, precision, reliability and reduced the errors and time. This project "Autonomous Object Picker" deals with the simulation of a robot which automatically detects a desired object and picks it from an environment. This Robot will play an important role in the manufacturing sectors when there is a need to separate the components based on our requirements. In Industries these Robots will replace various human domains in near future.

INTRODUCTION:

This Project "Autonomous Object Picker" involves building a robot which automatically picks up an object using V-Rep/MATLAB. The main aim of the project is to pick a coloured object decided by the user in the V-Rep scene. I have used V-Rep for simulation and MATLAB for processing the desired object and used Remote-Api to communicate between the server (V-Rep) and the client (MATLAB). It is done is several phases. The project is done by scanning the environment and get the desired object's coordinates by Matlab and sending the data back to the V-Rep to control the Robot.

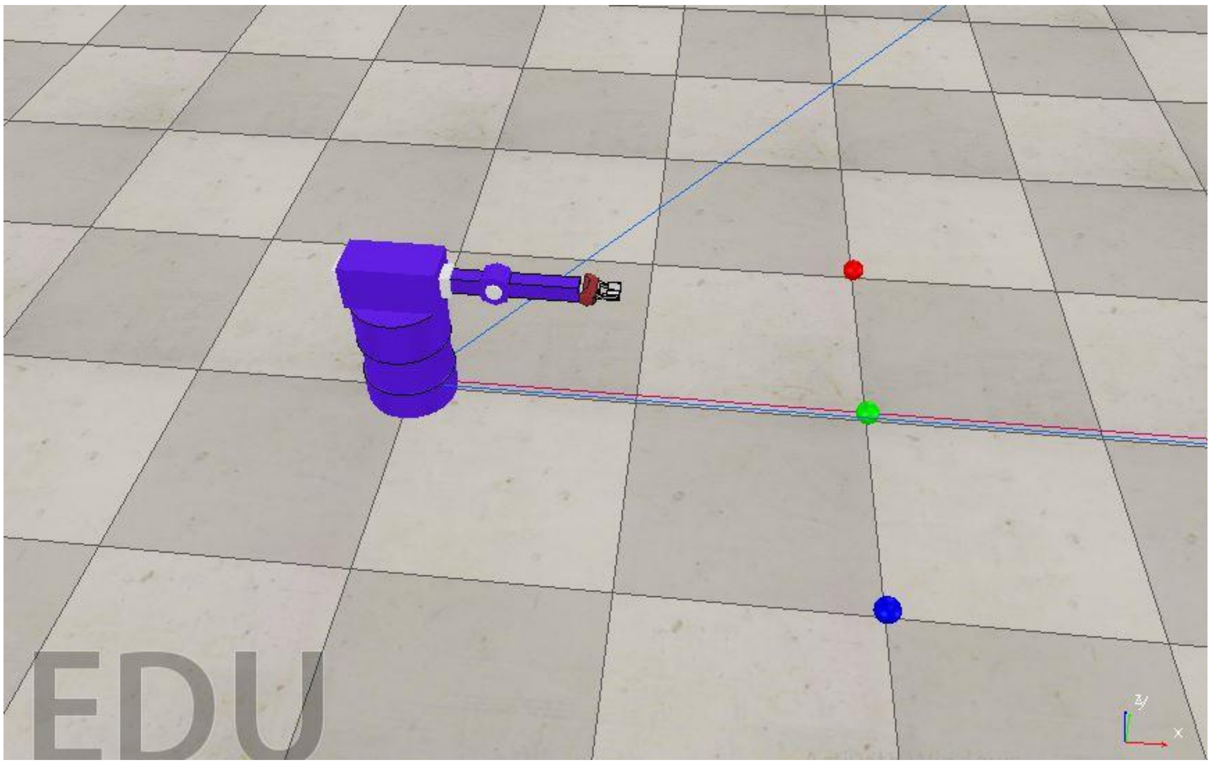

Figure 1. Automomous Object Picker Simulation in V-Rep

SIMULATION AND EXPERIMENTAL SETUP:

The Simulation of Autonomous Object Picker in V-Rep is shown in the figure previously.

The Project can be split into several parts. Initially, the environment in v-rep is scanned by the vision sensor and it is image processed in Matlab to find the desired object. Then find the distance between the object and the robot using a distance sensor (ultrasonic proximity sensor). By knowing these values we will be able to find the actuating angle and position for the Revolute and Prismatic joints in the robot to reach that desired object. Finally the object is picked by using a gripper.

The Vision Sensor are similar to camera objects. They will render the objects that are in their field of view and trigger detection if specified thresholds are over- or under-shot. The vision sensor's content can be viewed, processed and accessed through API. The resolution of the vision sensors can be varied depending on our requirement. Higher the resolution higher the time to communicate between the client and the server. Two types of vision sensors are available in V-Rep - 1.Orthograpic type 2.Prespective type. The vision sensors are added to the scene with menu bar -> Add -> Vision sensor -> type. This project uses Perspective type vision sensor with a resolution of 512x512. The vision sensor is attached to the base of the robot. The vision sensor content from the scene can be seen through floating view as shown.

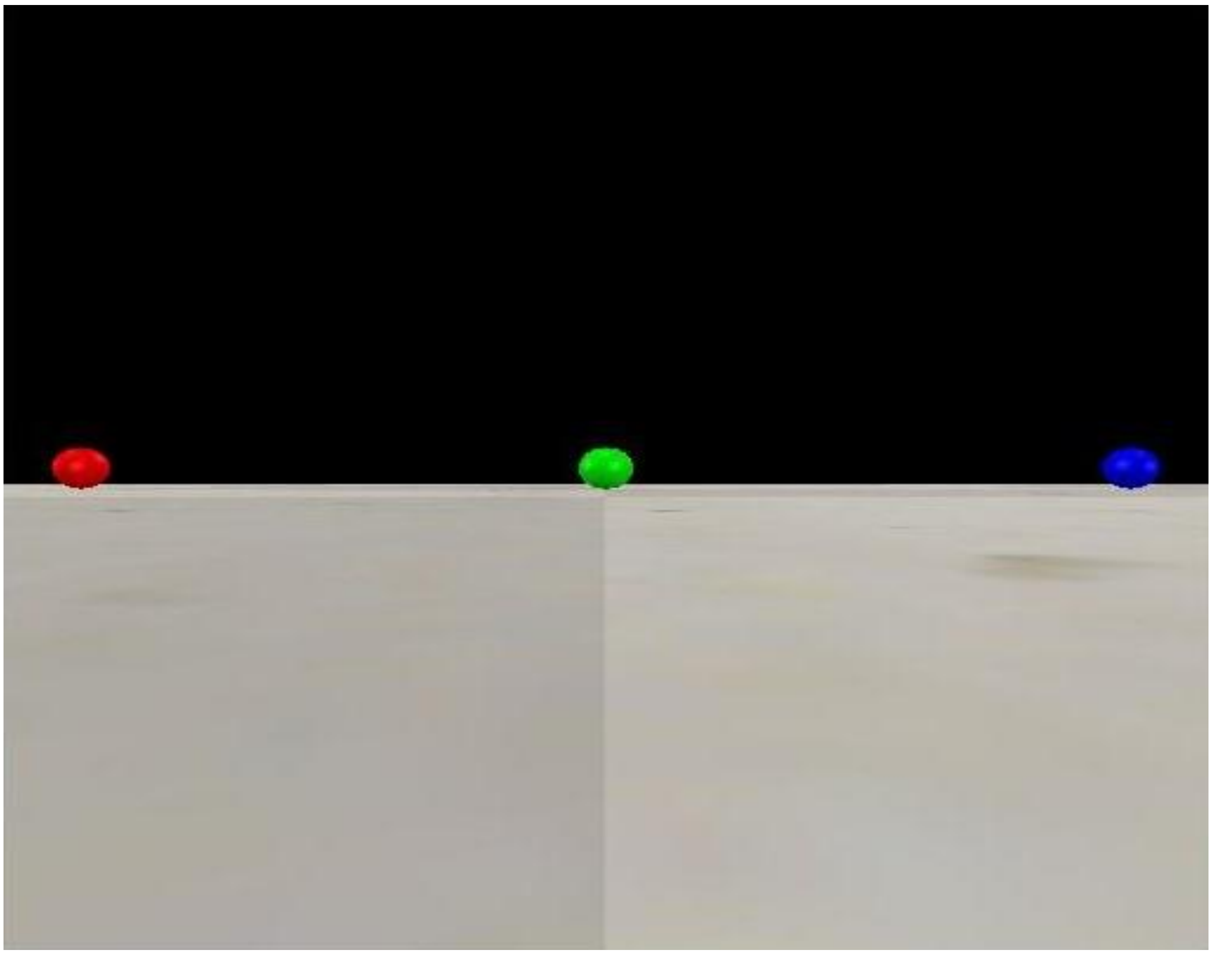

Figure 2. Floating View of the Vision sensor

The object to be picked is decided by the user based on the colours Red, Green, Blue. The input is got from the Matlab.

Then, the content from the vision sensor is sent to matlab by remote api framework. It is done by using ‘simxGetImageSensorImage2’ function. Then this is image processed in Matlab. The image is first converted into R or G or B image format depending on the user input. Then with the help of the Matlab function ‘imfindcircles’ we can find the desired object by knowing its radii of the object. This will give the centre of the object with respect to the matlab figure. Then there will be a need to convert the coordinates in matlab to the coordinates in the scene with some calculations. The formula used here to calculate it was given below

Y coordinate in vrep scene =

(centre of the matlab fig – desired coordinate in matlab fig) / (% change btw matlab & v-rep)

This will gives you the desired y coordinate in the V-rep scene.

The different images converted into R or G or B is as shown

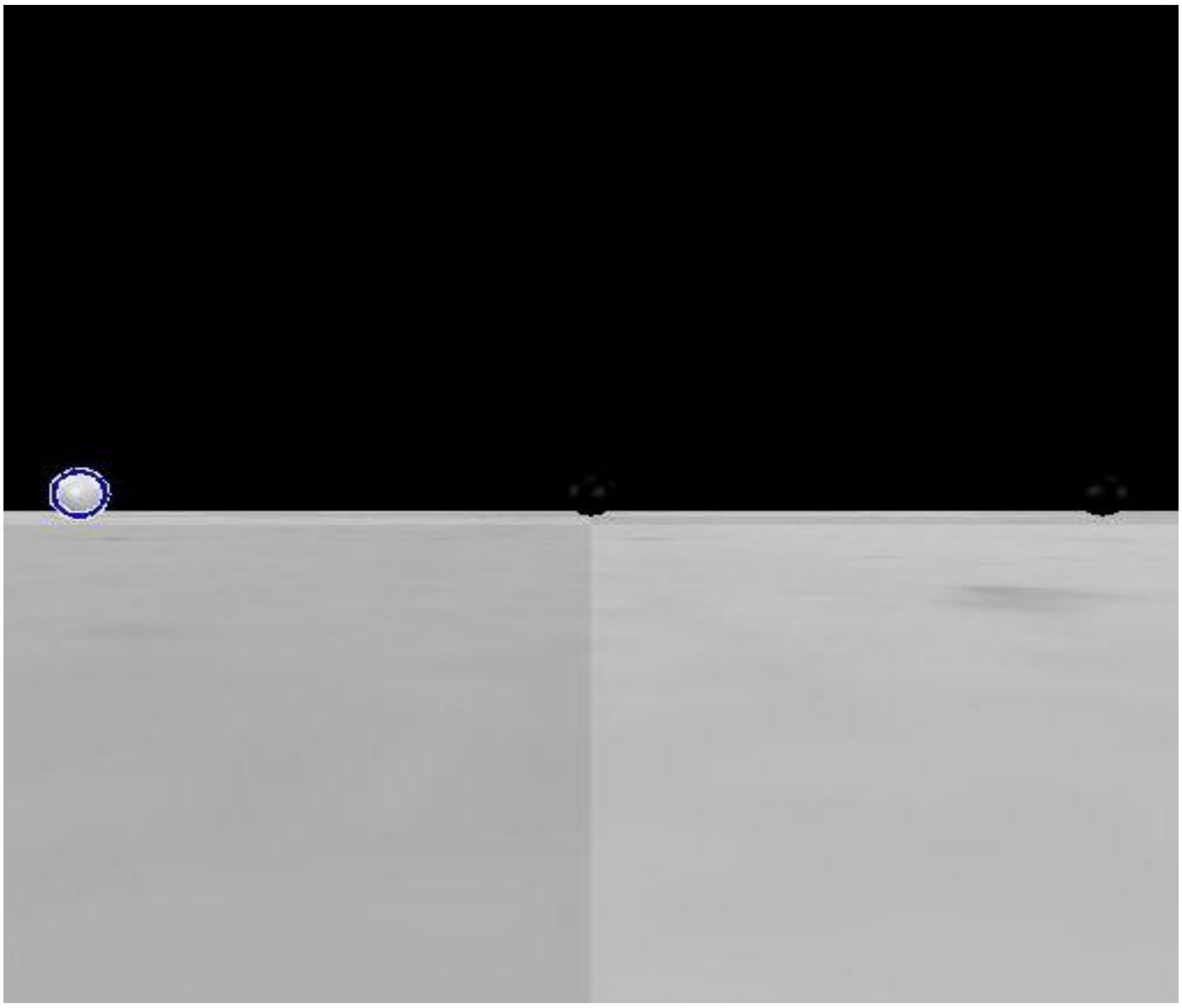

Figure 3. Red ball after processing

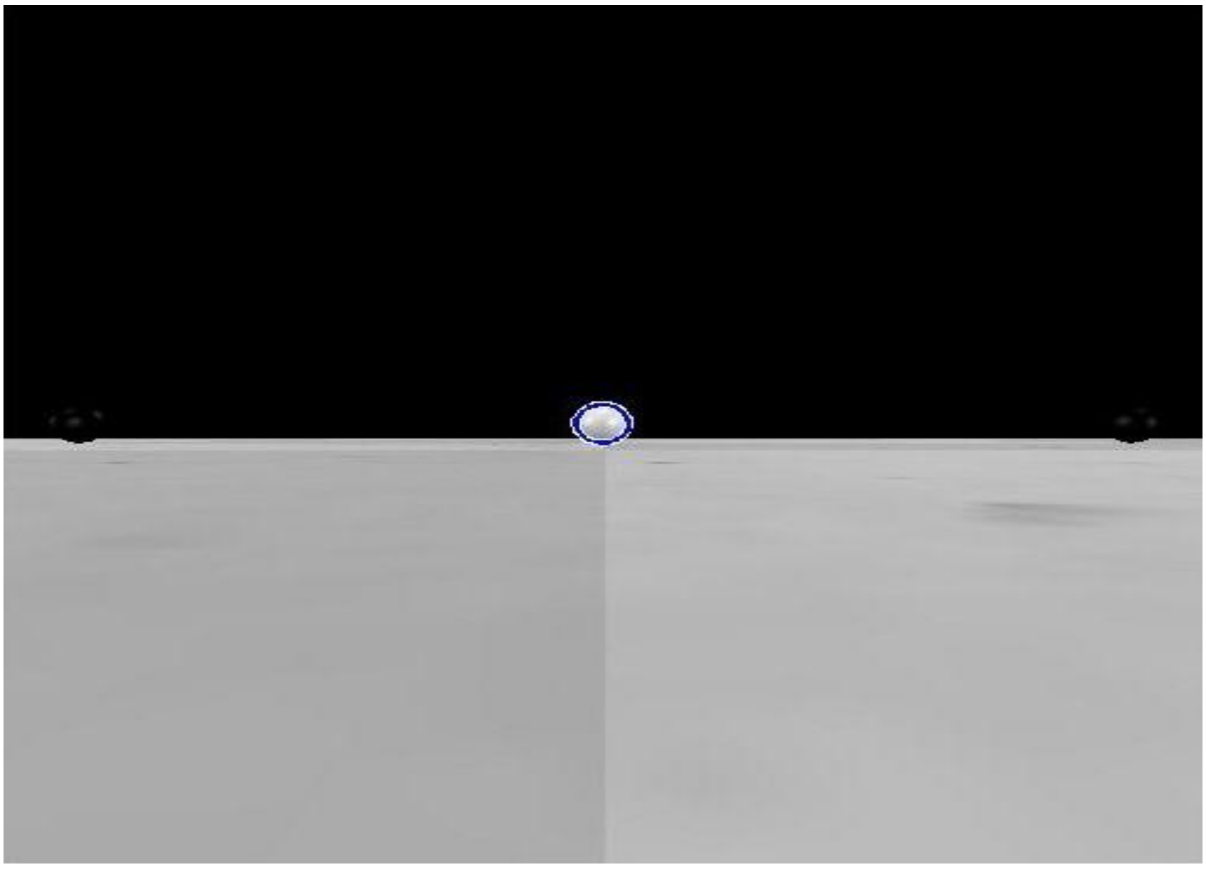
Figure 4. Green ball after processing

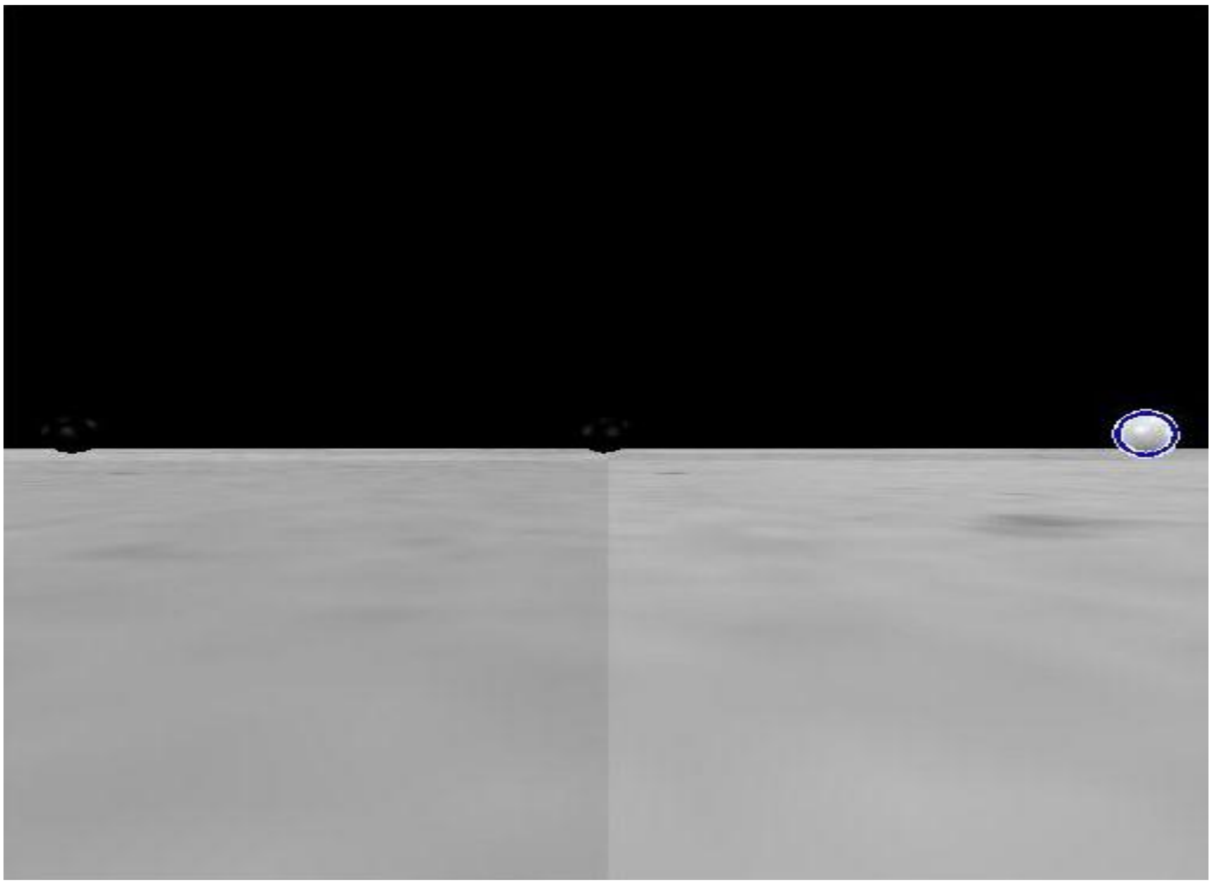
Figure 5. Blue ball after processing

Then the distance between the object and the robot is found by using a distance sensor. Some of the distance sensor commonly used are Proximity Sensors, LASER sensor, Hokuyo laser sensor. Therefore with this we got the coordinates of x and y of the desired object in the v-rep scene. This project uses Ultrasonic proximity sensor to the distance. The distance is obtained by using the remote api function ‘simxReadProximitySensor’ in the matlab.

Then the angle for the for the revolute joint cant be found by using the formula

Theta=asin(Y coordinate/X coordinate)

Thus the desired angle and distance is obtained.

Now the commands should be sent to the robot. So the remote api function ‘simxSetJointPosition’ is used to set the desired joint angles for revolute and distance for prismatic joints. Another Revolute joint is used to move the Gripper to the desired position. BaxterGripper is used in this project. We can change the gripper’s opening and closing length depending on our requirement. To send the commands to the gripper ‘simxSetIntegerSignal’ is used. If the signal name and value is equal to ‘BaxterGripper_close,1’ the gripper will close and open if the signal value is 0.

Then we can return the robotic arm to the desired position by sending the joint angle and position of revolute and prismatic accordingly.

Some of the importing things while doing the simulation are balancing the weight of several elements in robot, grippers should be able to hold the object perfectly, conversion of coordinates and sensitivity of the camera to find desired object should be carefully adjusted in order to eliminate the other unwanted objects in the screen.

The Matlab code for image processing is shown below

```
vrep.simxGetVisionSensorImage2(clientID,Vision_sensor,0,vrep.simx_opmode _streaming);
[ret,resolution,Image]=vrep.simxGetVisionSensorImage2(clientID,Vision_sensor,0,vrep.sim
x_opmode_oneshot_wait);
a=Image;
redball=a(:,:,1)
imshow(redball)
Rmin = 8;
Rmax = 13;
[centersBright, radiiBright,metricBright] =
imfindcircles(redball,[RminRmax],'ObjectPolarity','bright','Sensitivity',0.95,'EdgeThreshold',
0.1);
hBright = viscircles(centersBright, radiiBright,'EdgeColor','b');
y11= centersBright(1);
```

This will give the coordinates of the desired object in matlab.

The angles and position of the revolute can be set by using the following code

```
%Initializing the coordinates
Y1=(257.3-y22)/(443); %conversion of coordinated
X1=1;
d11=(sqrt((x2^2)+(y2^2)));
d1=d11-0.19;
o11=y1/d11;
theta1=asin(o11);


%Set the desired Joint values
loop=1;
while loop<2%(clientID~=-1)
ret = vrep.simxSetJointTargetPosition(clientID, Revolute, theta1,
vrep.simx_opmode_oneshot_wait);
pause(1)

[ret,handle]=vrep.simxGetObjectHandle(clientID,'Proximity_sensor',vrep.simx_opmode_blo
cking);

[ret,detectionState,detectedPoint,detectedObjectHandle,detectedSurfaceNormalVector]=vrep.
simxReadProximitySensor(clientID,handle,vrep.simx_opmode_oneshot_wait);
d111=detectedPoint(3);
      if d111 ~= d1
          ret =
vrep.simxSetJointTargetPosition(clientID,Prismatic,d2,vrep.simx_opmode_oneshot_wait);
       else
          display('Error')
       end
       loop=loop+1;
   end

%Get the new joint angle
[ret,o1]=vrep.simxGetJointPosition(clientID,Revolute,vrep.simx_opmode_buffer);
[ret,d1]=vrep.simxGetJointPosition(clientID,Prismatic,vrep.simx_opmode_buffer);

%to controlling the BaxterGripper
[ret,BaxterVaccumCup]        =        vrep.simxGetObjectHandle(clientID,'BaxterVaccumCup'
,vrep.simx_opmode_blocking);
vrep.simxSetIntegerSignal(clientID,'BaxterVaccumCup_active',
1,vrep.simx_opmode_oneshot);
```

CONCLUSION:

Thus the “Autonomous Object Picker” can pick any object in the scene using the colours. Thus sending the commands from the server (V-Rep) and client (Matlab) through remote Api made this possible. I believe that this project will find its place in real-time scenarios in industries and in many applications.