OBJECT RECOGNITION BOT

AN INTERNAL FUNDED PROJECT REPORT

Submitted by

GAJESH S	**312215104086 – III Year**
GOKUL H	**312215105039 - III Year**
GOMATHY V	**312215105041 - III Year**

ACADEMIC YEAR 2017-18

COMPUTER SCIENCE AND ENGINEERING

SSN COLLEGE OF ENGINEERING

KALAVAKKAM 603110

JAN 2019

BONAFIDE CERTIFICATE

Certified that this project report titled **OBJECT RECOGNITION BOT** is the bonafide work of **GAJESH S (312215104086 – IV Year), GOKUL H (312215105039 – IV Year) GOMATHY V (312215105041 - IV Year)** , who carried out the project work under our supervision as an internal funded project in Department of CSE during the Academic year 2017-2018.

Dr.D.Thenmozhi
Supervisor
Assistant Professor,
Department of CSE,
SSNCE

Dr. Chitra Babu
Head of the Department
Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Place:
Date:

ACKNOWLEDGEMENTS

I thank GOD, the Almighty for giving me strength and knowledge to do this project.

I would like to thank, with a deep sense of gratitude, our guide **Dr.D.Thenmozhi**, Associate Professor, Department of Computer Science and Engineering, for his valuable advice and suggestions as well as his continued guidance, patience and support that helped me to shape and refine my work.

My sincere thanks to **Dr.CHITRA BABU**, Head of the Department of Computer Science and Engineering, for her words of advice and encouragement. I would like to thank our project **Coordinator Dr. T. T. MIRNALINEE**, Professor, Department of Computer Science and Engineering for her valuable suggestions throughout the first phase of our project.

I express my deep respect to the founder **Dr. SHIV NADAR**, Chairman, SSN Institutions. I also express my appreciation to **our Dr. S. SALIVAHANAN**, Principal, for all the help he has rendered during this course of study.

I would like to extend my sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of my project work. Finally, I would like to thank my parents and friends for their patience, cooperation and moral support throughout my life.

Gajesh S **Gokul H** **Gomathy V**

ABSTRACT

The project intends to design a bot with the ability to map a given space or a room, distinctly find an object with the help of cameras and position itself near it, so that the mounted robotic arm can execute its task. Instructions are given as voice commands. An aim of this project is to allow mounting of a robotic arm which accurately tracks an object, picks it up and delivers it to the user. By developing the technology of the project with the implementation of intelligent systems which provides the ability to perform a sequence of operations and by providing a user-friendly interactive experience, the project can act as an aid for the physically disabled, as a personal assistant. Object Recognizing Bot is a collaborative student's project between Computer Science department (Gajesh S) and Electrical and Electronics Department (Gokul H, Gomathy V)

CHAPTER 1

INTRODUCTION

The main objective of this project is to overcome that shortcoming by designing a high torque terrestrial robot capable of recognizing objects based on user instruction and loads simple objects to navigate around.

ORB is capable of recognizing and moving near the desired object so that the arm can pick it up. It is installed with clamp brackets capable of mounting and unmounting the robotic arm and planning optimal routes for faster response time.

OBJECTIVES

a. To study various concepts of kinematics, computer vision, control systems and acquire proper knowledge on the same
b. To implement the above concepts in a physical system by building ORB and provide instructions to solve a given task.
c. To facilitate the linking of a robotic arm to a robotic arm to work coordinately

DEFINITION OF PROBLEMS

i. Receiving and understanding of voice commands
ii. Mapping of the room and distinct recognition of an object
iii. Positioning the bot in the desired position, avoiding obstacles in its trajectory

The entire project revolves around the idea of building a convenient base robot, which understands and recognizes any particular robot's requirements and aids the robot, in reaching this objective. The robot revolves around communication pathways, and motion planning, to reach where required. The object's structural details given as voice commands are analyzed by the speech recognition process and the interpreted instruction is fetched to a microcontroller's processing unit. Cameras are used to take multiple pictures of the area under mapping and using computer vision, it distinctly recognizes the object. A Raspberry Pi module will act as this processing unit which sends the signal to control the DC motors which in turn provides mobility to the bot and reach the nearest accessible position to the object.

CHAPTER 2

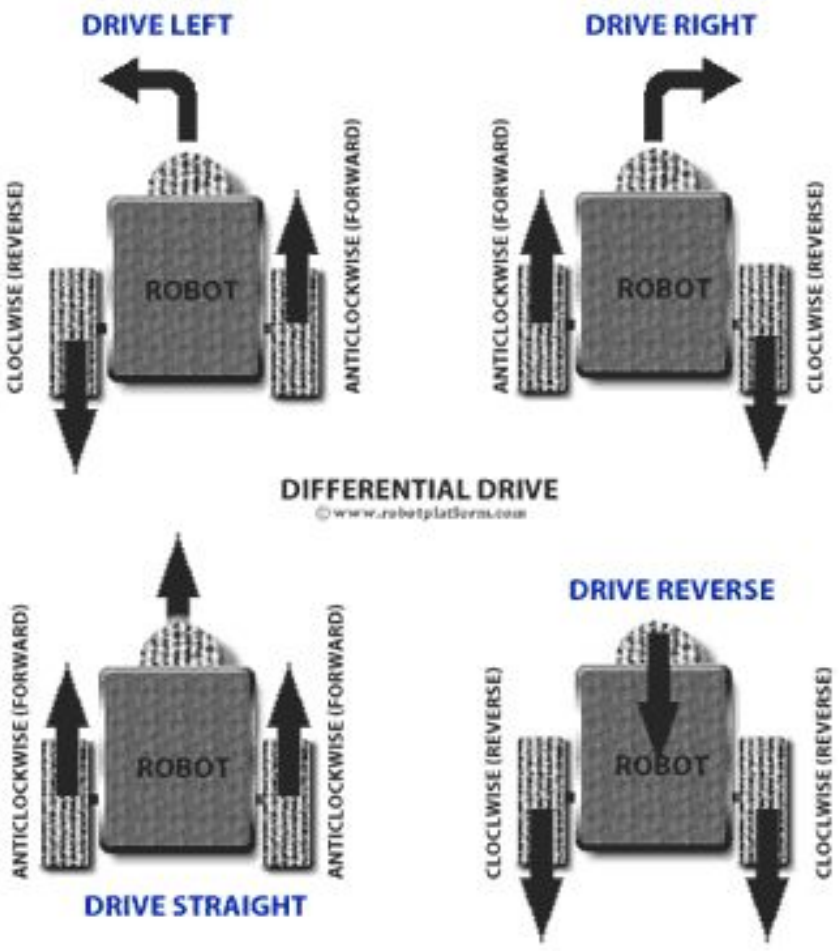
MECHANICAL SETUP

Wheel selection and Arrangement:

We used standard/ fixed wheels from the 4 different types of wheels, orientable, Omni and ball wheel because we only required two degrees of freedom- front and back. The center of the wheel is fixed to the robot chassis. The angle between the robot chassis and wheel plane is constant.

Differential Drive:

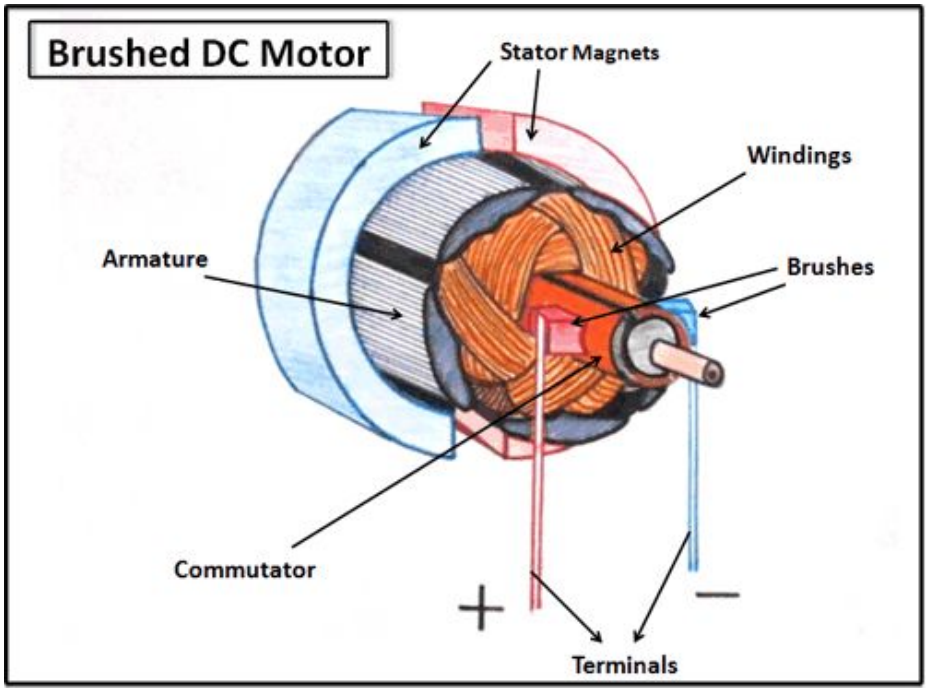
The velocity difference between the two motors drives the robot in any required path and direction. Differential wheeled robot can have two independently driven wheels fixed on a common horizontal axis or three wheels where two independently driven wheels. One of the major disadvantages of this control is that the robot does not drive as expected. It neither drives along a straight line nor turns exactly at expected angles, especially when we use DC motors. This is due to difference in the number of rotations of each wheel in a given amount of time. To handle this problem, we need to add a correction factor to the motor speed. For example, if you intend to drive your robot in a linear path and feel that the robot is turning towards one side, then a correction factor can be added to reduce the speed of the other wheel.



Actuators:

We used a brushed DC motor ie, a motor has a rotor and a stator with one of them being a permanent magnet. In a brushed DC motor, the rotor has a permanent magnet and the stator has electromagnets. Since the motor needs a way to detect the rotor's orientation, it uses brushes as a commutator which is a piece of rotor touching the shaft. When the rotor rotates (in turn the brush rotates), it detects the

change in orientation and flips the current. We used a DC motor with 10kg pulling capacity as the ARM will contribute the more weight.

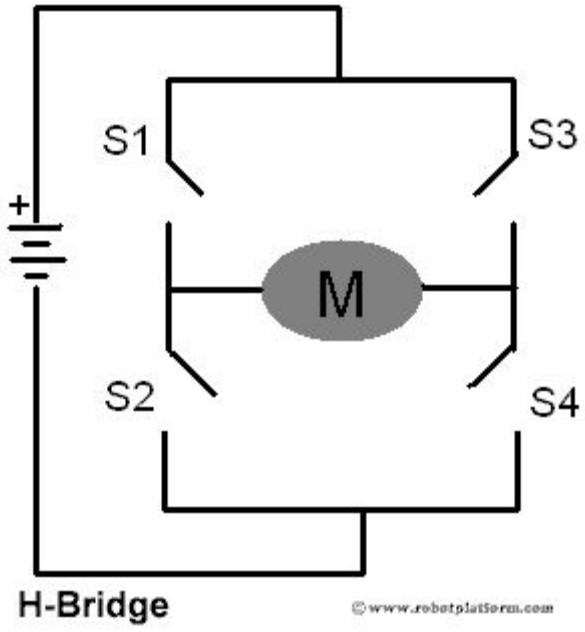


Speed Control:

We used L293D which is a typical Motor driver or Motor Driver IC which allows DC motor to drive in either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC.

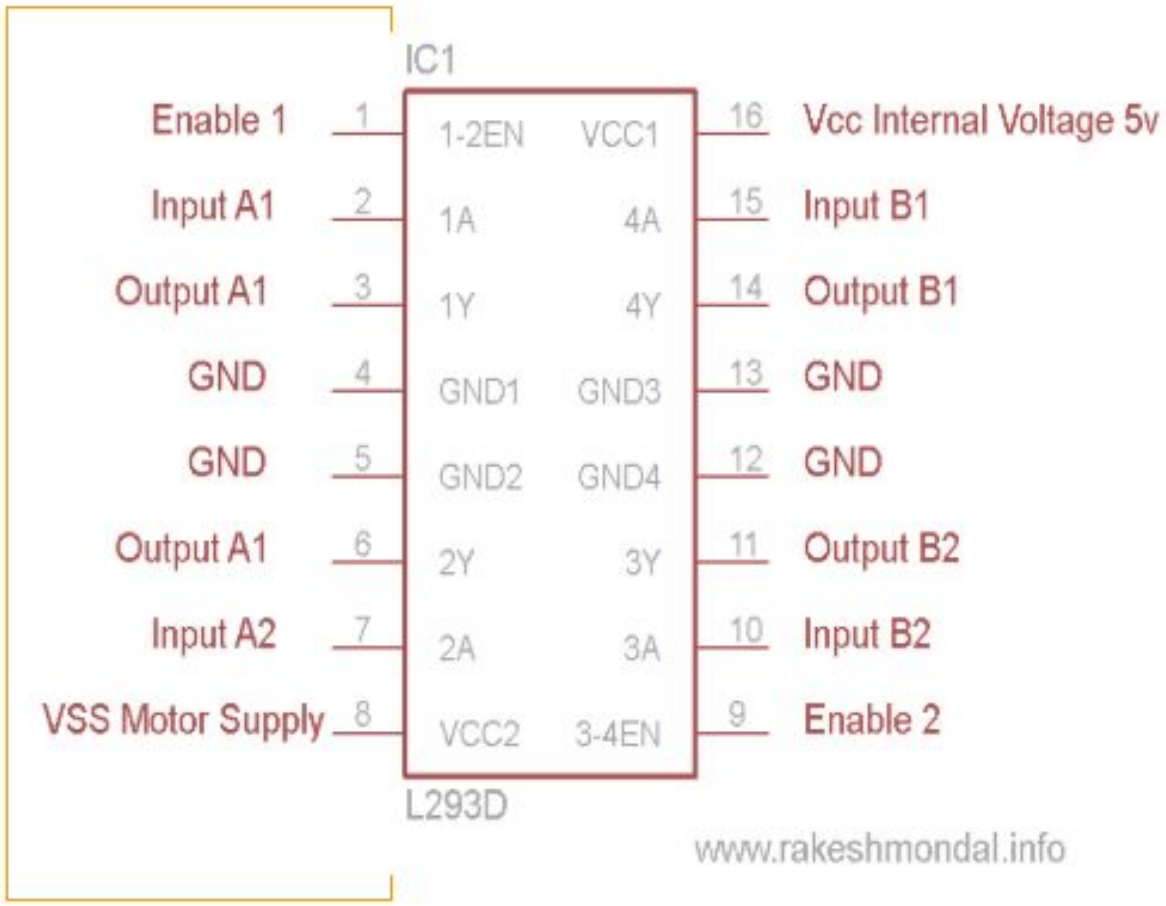
It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know, the voltage needs to change its

direction for being able to rotate the motor in a clockwise or anticlockwise direction, Hence H-bridge IC is ideal for driving a DC motor.



In a single L293D chip there are two h-Bridge circuits inside the IC which can rotate two dc motors independently. Due to its size, it is very much used in robotic applications for controlling DC motors.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It’s like a switch.



There are 4 input pins for l293d, pin 2,7 on the left and pin 15, 10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across the left side and right input for the motor on the right-hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.

- **Pin 2** = **Logic 1** and **Pin 7** = **Logic 0** | Clockwise Direction
- **Pin 2** = **Logic 0** and **Pin 7** = **Logic 1** | Anticlockwise Direction
- **Pin 2** = **Logic 0** and **Pin 7** = **Logic 0** | Idle [No rotation] [Hi-Impedance state]
- **Pin 2** = **Logic 1** and **Pin 7** = **Logic 1** | Idle [No rotation]

CHAPTER 3

SENSOR SELECTION

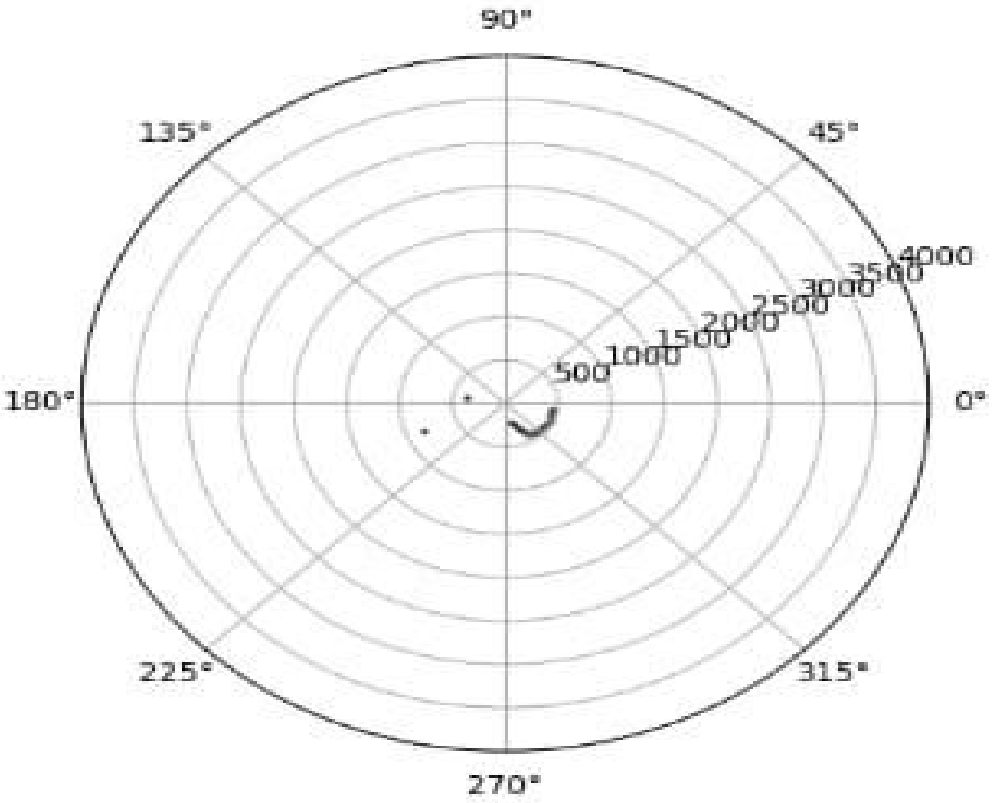
LIDAR:

LIDAR (Light Detection and Ranging) is an optical remote sensing system which can measure the distance of a target by illuminating it with light. LIDAR technology is being used in Robotics for the perception of the environment as well as object classification. The ability of LIDAR technology to provide 2D elevation maps of the terrain, high precision distance to the ground and approach velocity can enable safe landing of robotic and manned vehicles with a high degree of precision.

LIDAR consists of a transmitter which illuminates a target with a laser beam, and a receiver capable of detecting the component of light which is essentially coaxial with the transmitted beam. Receiver sensors calculate a distance, based on the time needed for the light to reach the target and return. The sensor measures the phase shift between the transmitted and reflected signals. The lidar plot is given below.

However, it provides us with a huge set of data to process and map which cannot be independently done by Raspberry Pi. It needs a machine which can store and work with huge sets of data, which is not feasible in our case as it is a navigation

bot. The costs of the LIDAR module is also high, thus explored other options were explored.



Time of flight Sensor:

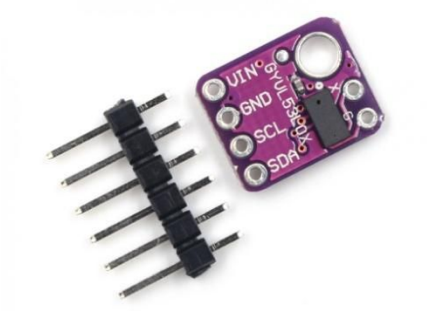
Time-of-flight sensors that measure depth by estimating the time delay from light emission to light detection. There are two TOF principles: (i) modulated-light and (ii) pulsed-light.

However the following problems were faced,

- Did not possess a broad range of detection in laser source.

- Installing the sensor on top of a servo did not work as serial data transmission was not robust.
- Laser source gets damaged easily and prone to highly reflective

surfaces and light sources



Sharp IR Sensor:

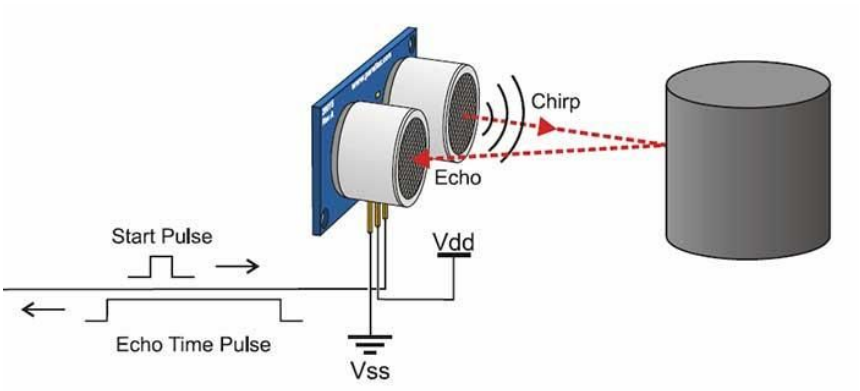
For measuring the distance to an object there are optical sensors using triangulation measuring method. Company “Sharp” produces the most common infra-red (IR) wavelength using distance sensors which have analog voltage output. The sensors made by “Sharp” have IR LED equipped with a lens, which emits a narrow light beam. After reflecting from the object, the beam will be directed through the second lens on a position-sensitive photodetector (PSD). The conductivity of this PSD depends on the position where the beam falls. The conductivity is converted to voltage and if the voltage is digitalized by using an analog-digital converter, the distance can be calculated.

This sensor did possess a border range of detection since IR beams were used. However, it did not provide sufficient data for mapping and obstacle avoidance. It was also not robust susceptible to damage but too expensive to be replaced.

Ultrasonic Sensor:

Ultrasonic sensors measure distance by using ultrasonic waves.
The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception.

However, it had low proximity reach despite possessing a decent range in space. Also, the rate of sensor data available was not sufficient to map. It was also highly prone to noise



All these sensors were tested, various filtering algorithms were implemented and still were unable to make substantial trade-offs to replace a LIDAR.

CAMERAS :

Instead of installing other proximity sensors, the problem route was altered to design a bot with one camera that could possibly detect the target object among other obstacles using image processing techniques.

Advantages of using a camera :

- Ease of installing due to compatibility with microprocessor board.
- Lesser issues due to latency and data transfer rate.
- Usability of computer vision based image processing techniques due to readily available libraries.
- Ease of analysis and hardware debugging.
- Reduced noise issues
- Better suitability with any environment.

CHAPTER 4

METHODOLOGY

Control Module:

1) Make the Connections as provided in the diagram. The values for GPIO pin numbers and frequency are provided below. Start the GPIO pins during the setup.

```
L_pwm=12, R_pwm=13, frequency = 100 (Hz), fast = (16,14), slow = (12,10).
Fast and slow are tuples to account for mechanical differences between
the motors used. May vary for other systems.
```

2) Define the functions for different motor actuation and bot movements as needed. We have specified our prototype to have slow and fast movements in the forward, left and right direction, as well as to stop. These definitions are done by providing the PWM values and direction values for each action. We have defined the reverse movement of the left motor as given below.

```
def slow_left_reverse():
        L.ChangeDutyCycle(slow[0])
        GPIO.output(L_dir, GPIO.LOW)
```

3) Define the image module as given below:
    a) read and store an image (test environment and pi)
        - OpenCV cannot directly read from the PiCam. You need to use the PiCamera module to do so. The image can then be converted to the required BGR format.
    b) object detection algorithm (prototype defined for 3 shapes and colors)
        - Filter by color followed by shape to remove some background noise.
    c) decoding direction from the target position based on centroid.

Note: We are building a standalone application on a raspberry Pi3, which cannot process computationally heavy algorithms such as CNN, Inception Network, and

YOLO within the tolerable latencies. Therefore, we have used basic image processing.

4) Integrate the modules and test them.
    a) Place all the lower level functions in a file lib.py which can act as a library for all the modules. This improves the readability of the code.
    b) Allow the image module to read and process the camera feed. It will provide the direction to bot has to move.

ALGORITHM FOR TARGET DETECTION:

Case 1: Target is a ball in the scene (Initial Case)

1) Read the BGR image using the camera.
2) convert to grayscale using cvtColor function of cv2 library.
3) use the HoughCircles algorithm of cv2 to detect circles in the image (2D version of the ball is a circle). This returns a list of circles. The first entry is what we are looking for.
4) Return the centroid (y,x) of the object detected.

Case 2: 3 Different colors and shapes the target can come in

1) Read the BGR image using the camera.
2) convert to HSV colour format using cvtColor function of cv2 library.
3) create a colour based image mask using cv2.inRange function.
4) filter the image based on the mask
5) if the shape is a sphere, look for circles using houghcircles and return as in previous case

6) else if it is a box or a cylinder, threshold the image and detect the contours of objects. We can differentiate between the two using the shape of the contour.
7) return the detected object as the centroid (y,x)

HoughCircles:

1. For each A[a,b,r] = 0;
2. Process the filtering algorithm on image Gaussian Blurring, convert the image to grayscale (grayScaling), make Canny operator, The Canny operator gives the edges on image.
3. Vote the all possible circles in accumulator.
4. The local maximum voted circles of Accumulator A gives the circle Hough space.
5. The maximum voted circle of Accumulator gives the circle.

Voting Algorithm for HoughCircles:

```
For each pixel(x,y)
   For each radius r = 10 to r = 60 // the possible radius
     For each theta t = 0 to 360  // the possible  theta 0 to 360
       a = x – r * cos(t * PI / 180); //polar coordinate for center
       b = y – r * sin(t * PI / 180);  //polar coordinate for center
       A[a,b,r] +=1; //voting
     end
   end
 end
```

Contour Approximation Algorithm - Douglas Pucker Algorithm:

```
function DouglasPeucker(PointList[], epsilon)
   // Find the point with the maximum distance
   dmax = 0
```

```
    index = 0
    end = length(PointList)
    for i = 2 to ( end - 1) {
        d = perpendicularDistance(PointList[i], Line(PointList[1], PointList[end]))
        if ( d > dmax ) {
            index = i
            dmax = d
        }
    }

    ResultList[] = empty;
// If max distance is greater than epsilon, recursively simplify
    if ( dmax > epsilon ) {
        // Recursive call
        recResults1[] = DouglasPeucker(PointList[1...index], epsilon)
        recResults2[] = DouglasPeucker(PointList[index...end], epsilon)

        // Build the result list
        ResultList[] = {recResults1[1...length(recResults1)-1], recResults2[1...length(recResults2)]}
    } else {
        ResultList[] = {PointList[1], PointList[end]}
    }
    // Return the result
    return ResultList[]
end
```

CHAPTER 5

RESULTS

The bot was first attempted to work with Raspberry Pi Zero which has the following specifications:

- 1GHz, Single-core **CPU**.
- 512MB **RAM**.

However, Raspberry Pi 3B+ with

- SoC: Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz.
- GPU: Broadcom Videocore-IV.
- **RAM**: 1GB LPDDR2 SDRAM.

was found to show better and smoother operation with fewer latencies by making a significant price tradeoff of additional 2000 INR and the same has been installed.

The Object detection section of the bot can effectively detect objects based on shapes and colors specified within its dictionary of objects. Further research can improve this dictionary with more shapes, colors, featuristic attributes and even 3D objects.

The Navigation section of the bot can steer with the help of differential drive arrangement and moves the bot towards the object whose position is constantly being detected.

The complete code and a output video for the bot can be found at the following GitHub URL: https://github.com/GajeshS/ObjectRecognitionBot .