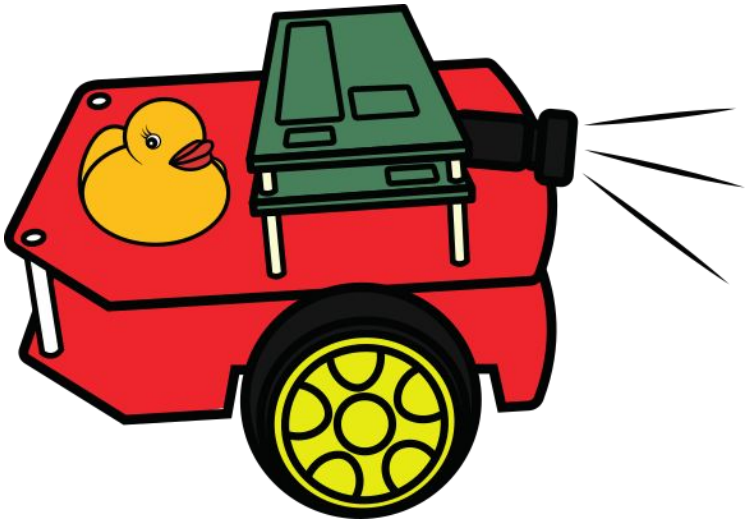
# RIS Project Group 4

## Ball tracking  gesture recognition

By:Haseeb Ahmed,Saad El Jadouri,Purshotam
Group 4

# Our Bot



DB19-

Yadavbot

# Ball Tracking

Ball tracking refers to the process of detecting and following the trajectory of a ball in a given video or image sequence, some of most used techniques to complete this task include: colorbase tracking, feature based tracking…

Ball tracking entails following and updating a detected object's position throughout time continuously. By using tracking, Duckiebots can keep track of an object's trajectory and direction, ensuring that it remains in the robot's field of view.

We aim to develop algorithms for detecting and tracking a ball using the robot's sensors and control systems. The Duckiebot/Yadavbot will be able to move towards the ball, follow it.

# Gesture recognition

Gesture recognition is an exciting area of research that enables machines to interpret and respond to human gestures. With a Duckiebot, we will develop algorithms for recognizing simple hand gestures such as waving, pointing, or fist bumping. These gestures will serve as commands for controlling the Duckiebot's movement and performing various tasks.

This technology has many applications, including human-robot interaction, virtual reality, and gaming. There are different types of gestures, including hand gestures, body gestures, and facial expressions. Hand gestures are the most commonly recognized, and they can be used to control robots, drones, and other machine

# Our implementation:

# Overview

- For our RIS Project using the duckietown we plan to implement using computer vision and ROS knowledge simple features. For our choice, we think that the human robot interaction is an interesting area probably one of the most interesting and fun to watch, which many of are easy concepts in theory

# Background

ROS:The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools.

Computer vision knowledge is a key in our project since we will use the camera sensor for our features implementation

Docker is a containerization platform that provides a lightweight and portable environment for running applications. In the context of Duckietown, Docker is used extensively to package and distribute software for running on Duckiebots.

# Prerequisites

- Operating system Ubuntu 20.04
- Dependencies:Python 3, Git,Curl,Docker
- Account setup:GIT & Duckietown DockerHub
- Perform Calibration
- Opencv rospy (python dependencies)...

# Ball Tracking

In order to do that, we went via

- Object detection
- And Object/Ball Tracking

# Steps  for our Ball detection:

- Acquisition of Sensor Data:
- Preprocessing
- Object detection Algorithm:
- Object Localization:
- Ball Tracking Initialization:
- Motion Estimation and Occlusion handling:

# Methodology

Computer Vision:

- Gaussian Blur: We used a gaussian blur to filter our images.

- Canny Edge detector comes to affect and provide contours based on their area to find ball.

ROS Subscribers & Publishers:

- Subscribes to the our camera topic and receives the camera image data.

- Publishes in the motor topic depending on the balls position.

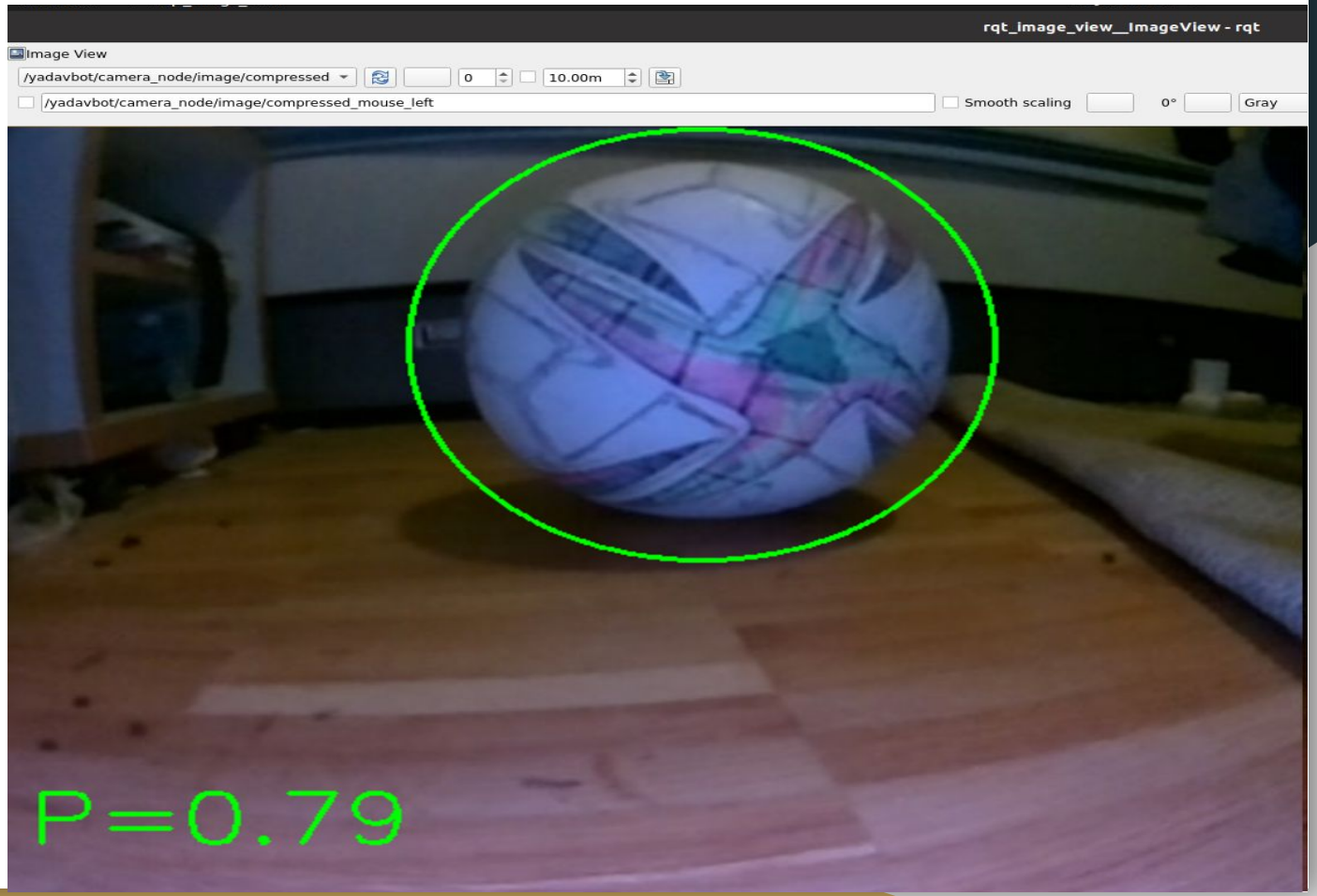
# Code snippets:

```
ball_publisher = rospy.Publisher('/yadavbot/ball_position', Point, queue_size=10)

# Create a publisher for motor commands
motor_pub = rospy.Publisher('/yadavbot/motor_control/cmd_vel', Twist2DStamped, queue_size=1)

# Set the control rate
rate = rospy.Rate(10)

while not rospy.is_shutdown():
    # Create Twist message for motor control
    motor_cmd = Twist2DStamped()

    # Calculate motor control based on ball position and gesture count
    if ball_x > 0:
        # Adjust the turn based on the ball's x position
        if ball_x < 320:
            motor_cmd.v = FORWARD_SPEED
            motor_cmd.omega = -TURN_SPEED
        else:
            motor_cmd.v = FORWARD_SPEED
            motor_cmd.omega = TURN_SPEED

        # Perform additional actions based on the gesture count
        if gesture_count > 0:
            # Do something based on the detected gesture
            pass

    # Publish the motor command
    motor_pub.publish(motor_cmd)

    # Sleep to maintain control rate
    rate.sleep()

if __name__ == '__main__':
    try:
        motor_control()
    except rospy.ROSInterruptException:
        pass
```

```
^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos    M-U
^X Exit        ^R Read File    ^\ Replace     ^U Paste Text  ^T To Spell   ^_ Go To Line M-E
```

```
# Publish the ball position
if len(filtered_contours) > 0:
    M = cv2.moments(filtered_contours[0])
    if M['m00'] > 0:
        ball_x = int(M['m10'] / M['m00'])
        ball_y = int(M['m01'] / M['m00'])
        ball_position = Point(x=ball_x, y=ball_y, z=0)
        ball_publisher.publish(ball_position)

# Display the image
cv2.imshow("Gesture and Ball Detection", cv_image)
cv2.waitKey(1)

motor_control():
global ball_x, ball_y, gesture_count

# Initialize the ROS node
rospy.init_node('motor_control_node')

# Subscribe to the camera image topic
rospy.Subscriber('/yadavbot/camera_node/image/compressed', Image, image_callback)

# Create a publisher for ball position
ball_publisher = rospy.Publisher('/yadavbot/ball_position', Point, queue_size=10)

# Create a publisher for motor commands
motor_pub = rospy.Publisher('/yadavbot/motor_control/cmd_vel', Twist2DStamped, queue_size=1)
```

# Results:


rqt_image_view__ImageView - rqt
Image View
/yadavbot/camera_node/image/compressed
0
10.00m
/yadavbot/camera_node/image/compressed_mouse_left
Smooth scaling
0°
Gray
P=0.79

# Gesture recognition

We have used Handlandmarks Mediapipe Algorithm for Gesture Recognition:

Handlandmarks mediapipe: MediaPipe is an open-source framework developed by Google that provides a wide range of machine learning solutions for various computer vision tasks. One of its notable features is the hand tracking module, which enables hand landmark detection and tracking.

# Methodology for Gesture Recognition

- Duckiebot performs gesture detection using the `HandLandmarks.detect` method. It updates the gesture count and draws landmarks on the image.
- The depth of each defect point from the HandLandmarks is calculated, and if the depth exceeds a certain threshold, it is considered as a gesture.
- Gesture count is published using the gesture_publisher object. And ball position is calculated by finding the centroid of the largest contour (assumed to be the ball) and publishing it using the ball_publisher object.
- The image with the detected ball and gestures is displayed. The main function initializes the ROS node, subscribes to the camera image topic, creates publishers for the ball position and gesture count, and starts the ROS loop.

# Code snippet:

```
# Initialize the variables for ball position and gesture count
ball_x = 0
ball_y = 0
gesture_count = 0

def image_callback(msg):
    global ball_x, ball_y, gesture_count

    # Convert the ROS Image message to OpenCV format
    bridge = CvBridge()
    cv_image = bridge.imgmsg_to_cv2(msg, "bgr8")

    # Perform gesture detection
    landmarks = HandLandmarks.detect(cv_image)
    if landmarks is not None:
        gesture_count = HandLandmarks.get_gesture_count(landmarks)
        HandLandmarks.draw_landmarks(cv_image, landmarks)
```

rqt_image_view__ImageView - rqt

Image View

/yadavbot/camera_node/image/compressed 0 10.00m

/yadavbot/camera_node/image/compressed_mouse_left Smooth scaling 0° Gray

# Challenges:

During our project we encountered many challenges of which can push your search and learning to a better level, from these challenges we cite:

-Setup: For our setup, we encountered a bit difficulties, especially within the sd card and network.

-Docker: To build and run our docker images we encountered architecture compatibility problems that took long to troubleshoot

- RaspberryPi: Having often issue with a raspberry either in network or also just crashing and needing boot which was time consuming.

# Future work:

For future work, we can say that many other features can be developed using duckietown.

The handland marks offer a large set of inputs.Yet, the idea is to create more embedded features that can connect between functionalities we are trying to implement, allowing us  to develop more complex robot behavior (like for our example using a gesture to detect and track the ball and so on..)

Trajectory planning, distance computation are also a good ideas as a future work for our bot that aims to

Interact with humans using different sensors (camera..)

# Summary

In summary, this project aims to explore the practical applications of robotics and artificial intelligence by focusing on ball tracking and gesture recognition using a Duckiebot. By the end of this project, wehave gained valuable knowledge and skills in robotics, machine learning, and computer vision.

Throughout this project, we tried to gain valuable knowledge and skills in robotics, machine learning, and computer vision. By developing and implementing algorithms for ball tracking and gesture recognition, we can enable the Duckiebot to perform more complex tasks and interact with humans in a more intuitive and natural way. As it is already shown previously, we tried to detect the Ball initially and then movement. Similarly, In Gesture recognition , Handlamarks mediapipe was totally new to us. We adjust ourselves and tried our code to rotate the Duckiebot namely Yadavbot to rotate around.