TUM School of Computation, Information and Technology
Technical University of Munich

Embedded Systems, Cyber-Physical Systems and Robotics Project Report

Autonomous Vision-Based Object Tracking Robot

Final Report about the Project

Submitted by Dharmang Pambhar
Sumeet Mourya
Rajkumar Pambhar
Sahil Pareshbhai Virani
Romit Kheni
Saumilkumar Savani
Aayush Gupta

Submitted on September 2, 2025

Abstract

Object following is a core capability in autonomous robotics, critical for navigation, interaction, and dynamic task execution in real-world environments. This study presents the design and implementation of a real-time object-following system on the Duckiebot platform within the Embedded Systems, Cyber-Physical Systems, and Robotics (INHN0018) course. The system integrates a lightweight perception pipeline with an efficient control loop to detect and pursue a designated target while operating within the resource constraints of a compact robotic platform.

The proposed approach leverages optimized computer vision techniques and feedback-based motion control to maintain stable tracking under varying conditions. Experimental evaluation in both simulated and physical Duckietown environments demonstrates reliable object pursuit, with performance remaining robust under standard lighting and occlusion scenarios, while highlighting limitations under extreme conditions. This work establishes the Duckiebot as a practical platform for testing vision-based behaviors on embedded systems and provides a foundation for future enhancements such as visual servoing, multi-object tracking, and adaptive control strategies.

Contents

Chapter 1

Introduction

1.1 Motivation and Background

Object following enables robots to perceive and adaptively interact with dynamic targets, a keystone capability for autonomous systems operating in vibrant, real-world environments. As modern robotics moves towards more autonomous, intelligent, and context-aware platforms, the ability to follow and track objects is critical for applications such as autonomous navigation, service robotics, surveillance, and human–robot interaction. Duckietown offers an ideal experimental framework, providing a low-cost yet authentic miniature urban environment for safe and structured testing of robotic behaviors. Within this context, the Duckiebot provides a resource-constrained but practical platform to explore and prototype real-world autonomy.

1.2 Problem Statement

While previous efforts in Duckietown have demonstrated lane following, traffic light detection, and closed-loop navigation, object following introduces additional challenges. It requires not only detecting a target but also maintaining continuous real-time tracking under varying conditions such as lighting changes, occlusions, and limited onboard computing resources. The problem addressed in this project is therefore: how to enable a Duckiebot to reliably detect, track, and follow a designated object in real time within the constraints of its embedded hardware.

1.3 Objectives of the Project

The objectives of this project are as follows:

- Design a lightweight perception pipeline suitable for embedded deployment.
- Implement a control system capable of real-time pursuit of a target object.
- Evaluate system performance in both simulated and physical Duckietown environments.
- Identify limitations and potential areas of improvement for future work.

1.4 Contributions

This work contributes to the field of embedded robotics in several ways:

- Development of an integrated object-following pipeline on Duckiebot, combining perception and control within strict computational limits.
- Empirical evaluation under realistic scenarios, including variations in illumination and partial occlusions.
- Demonstration of Duckiebot as a valid testbed for object-following research on resource-constrained platforms.
- Establishment of a foundation for future extensions, such as multi-object tracking and adaptive control strategies.

1.5 Justification

Duckietown presents an ideal, low-cost yet authentic platform for prototyping such behaviors within safe, structured miniature settings. This choice fosters robust embedded development, where computational limits are as real as those in full-scale systems. Prior implementations (e.g. vision-based lane following, camera-only perception control loops, and integration of lightweight deep learning detectors like YOLO) establish a strong methodological foundation. These systems illustrate the viability of real-time detection and control within Duckiebot constraints, laying the groundwork for an object-following extension. Through this project, we expand the Duckietown autonomy stack by integrating perception, planning, and actuation to reliably track moving objects in real time, both in simulation and physical testing. This contributes to the broader robotics community by demonstrating scalable, realistic autonomy research on embedded platforms and paving the way for future work such as multi-target tracking, learning-based control policies, and vision-guided adaptive behavior.

Chapter 2

Literature Review / Related Work

2.1 Existing Approaches in Object Following and Autonomous Navigation

Object following is a fundamental capability in mobile robotics, essential for tasks such as autonomous transportation, human–robot interaction, and collaborative industrial operations. Various strategies have been explored:

2.1.1 Vision-Based Methods

Camera-based perception remains the most widely used approach for real-time tracking. Techniques include color segmentation, optical flow, template matching, and deep learning-based detectors such as YOLO and SSD [5]. These approaches offer high accuracy but often face computational constraints on embedded platforms.

2.1.2 Sensor Fusion and Multi-Modal Perception

Integration of visual data with depth sensors, LiDAR, or ultrasonic sensors improves robustness against occlusions and lighting changes. Bayesian and Kalman filter-based tracking methods are frequently used to combine sensor data and predict target motion [4].

2.1.3 Control Strategies for Following

Feedback-based control loops, including PID controllers, pure pursuit, and model predictive control, are common for adjusting the robot's motion relative to the target. Reinforcement learning approaches have also been applied for optimizing pursuit behavior under dynamic conditions [3].

2.2 Work Done on Duckietown and Similar Platforms

Duckietown provides a low-cost, controlled **platform** for research and education in autonomous systems. Key contributions include:

2.2.1 Lane Following and Navigation

Lane detection using fisheye cameras, HSV segmentation, and inverse perspective mapping has enabled robust navigation in miniature urban layouts [1].

2.2.2 Obstacle Detection and Avoidance

Integration of lightweight object detectors (YOLOv3, YOLOv5) allows Duckiebots to detect obstacles in real time, facilitating dynamic navigation [6].

2.2.3 Reinforcement Learning and Adaptive Control

Reinforcement learning has been applied to optimize navigation policies in simulation and transfer them to physical robots, enabling adaptive lane following and obstacle-aware behaviors [3].

2.2.4 Intersection Handling and Multi-Robot Coordination

Algorithms using DBSCAN clustering, AprilTag detection, and finite-state machines manage intersections, stop lines, and multi-agent interactions [2].

2.3 Gaps and Challenges

Despite these developments, several limitations remain:

- **Dynamic Target Tracking:** Most implementations focus on static lanes or stationary obstacles; real-time tracking of moving objects is limited.
- **Resource Constraints:** Embedded hardware limits deployment of high-accuracy detectors.
- **Environmental Variability:** Lighting changes, occlusions, and complex layouts reduce tracking reliability.
- **Integration Challenges:** Seamless coordination of perception and control under real-time constraints is rarely addressed.

This project addresses these gaps by developing a lightweight, real-time object-following system on the Duckiebot platform, integrating perception, planning, and control, and validating its performance in both simulation and physical environments.

Chapter 3

System Design and Architecture

3.1 Overall System Overview

The objective of the system is to enable the Duckiebot to detect, track, and follow a moving object in real time while operating under embedded resource constraints. The design integrates perception, planning, and control into a cohesive pipeline that processes camera input, interprets the target's position, and generates motion commands. Figure **??** illustrates the high-level architecture of the object-following system.

The system is modular, allowing independent development and testing of perception and control components, while ensuring seamless communication between hardware and software through ROS (Robot Operating System). This modularity also facilitates future extensions, such as multi-object tracking, adaptive control strategies, or coordination between multiple Duckiebots.

3.2 Hardware Setup

3.2.1 Duckiebot Platform

The Duckiebot is a small, wheeled robotic platform equipped with an onboard computer, DC motors, wheels, and a front-facing camera. Its compact form factor provides a low-cost, resource-constrained environment that closely mimics real-world autonomous vehicle challenges. Lightweight perception and control algorithms are required to ensure real-time operation.

3.2.2 Sensors and Actuators

The primary sensor for object tracking is the onboard camera, positioned to capture the environment in front of the robot. Wheel encoders provide odometry information for velocity estimation and assist in precise motion control. The actuators consist of DC motors controlled via the motor driver circuitry on the PCB, enabling differential drive navigation.

3.2.3 PCB Schematic

Our Duckiebot printed circuit board (PCB) integrates the following functions:

- Power distribution and voltage regulation for the Duckiebot and peripheral components are provided by a specialized battery pack.
- Duckie hut, connects on the jetson nano as a hat to the 40 pin extension, for Sensor Interfacing and Motor driver circuitry.
- A wifi adapter as a communication interface, as well as the camera are connected directly to the jetson nano.

Figure 3.1 illustrates the PCB layout, highlighting key components and connections.

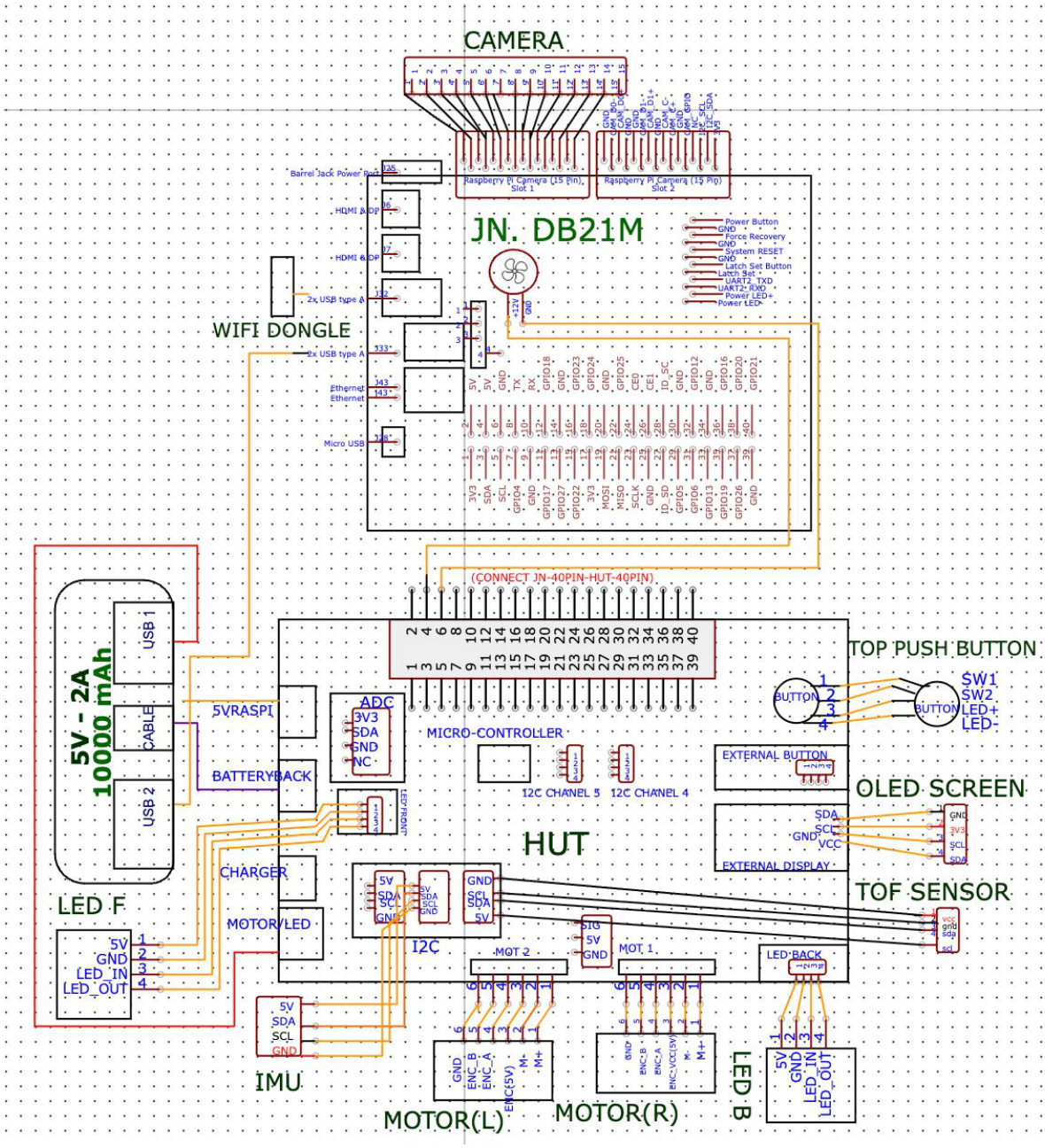


Figure 3.1: Custom PCB layout and component placement for the Duckiebot object-following system.

3.3 Software Architecture

3.3.1 ROS Nodes

The software stack is built using ROS, which provides modular nodes for perception, planning, and control. Each node communicates via topics and services, enabling asynchronous processing and flexible system configuration. Key nodes include:

- **Camera Node:** Captures and publishes video frames for the perception pipeline.
- **Perception Node:** Processes camera input to detect and localize the target object.
- **Control Node:** Receives target coordinates and computes velocity commands for the motors.

3.3.2 Perception Pipeline

The perception pipeline uses optimized computer vision techniques for object detection and tracking. Lightweight models, potentially based on YOLOv5 or color-based tracking, ensure real-time performance on the Duckiebot's embedded processor. The pipeline outputs the relative position of the target, which is then fed to the control module.

3.3.3 Control Pipeline

The control module implements a feedback-based motion control strategy. It converts the relative position of the target into wheel velocities using a PID controller, maintaining smooth pursuit while avoiding overshoot and oscillation. Integration with ROS ensures that control commands are synchronized with perception updates for consistent tracking performance.

3.4 Integration and Communication

Hardware and software components are tightly integrated using ROS topics and services. Sensor data flows from the camera and encoders to the perception node, then to the control node, which sends commands to the motors via the PCB. This closed-loop design allows the Duckiebot to react dynamically to changes in the environment and maintain stable object-following behavior under real-time constraints.

Chapter 4

Implementation & Experiments

4.1 Object Detection Method

4.1.1 Initial HSV-Based Approach

Our initial approach for object detection employed a traditional color-based method utilizing HSV (Hue, Saturation, Value) color space thresholding to identify the tennis ball based on its distinctive green color. This pipeline consisted of converting camera frames from BGR to HSV color space, applying color masks with predefined HSV ranges, performing morphological operations (opening, closing, dilation) to clean noise, and using contour detection to identify object boundaries and calculate centroids.

4.1.2 Limitations of Color-Based Detection

However, the HSV-based detection method proved unreliable in practice, suffering from inconsistencies due to varying lighting conditions, reflections, shadows on the target object, and background clutter containing similar colors. This resulted in frequent false positives and missed detections across different environments.

4.1.3 Machine Learning Transition

To overcome these fundamental limitations, we transitioned to a machine learning approach by training a custom YOLOv8 object detector specifically for tennis ball detection. We collected and annotated a diverse dataset of tennis ball images using tools like LabelImg, then fine-tuned the YOLOv8 model using transfer learning on Google Colab. The trained model achieved 99.6% precision and 95.8% recall on validation data, demonstrating significantly higher accuracy and robust performance across varying environmental conditions compared to traditional color-based methods.

4.2 Control Algorithms for Following

4.2.1 PID-Based Feedback Control System

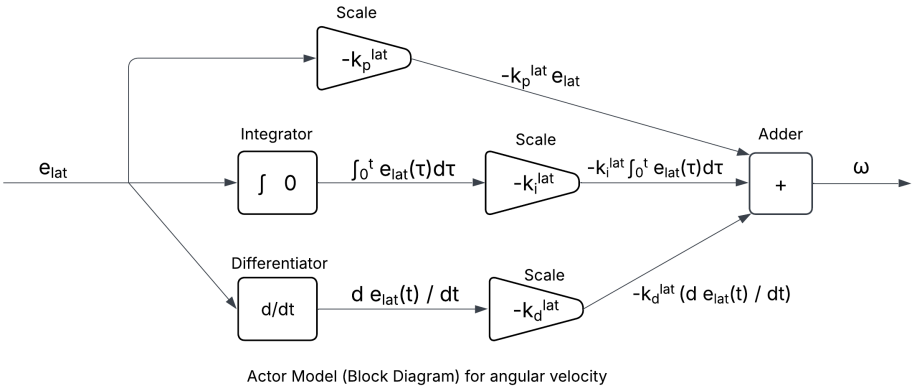
We implemented a sophisticated PID-based feedback control system to guide the Duckiebot in following the detected tennis ball. The control architecture employed separate PID controllers for different aspects of movement:

4.2.2 Actor Models

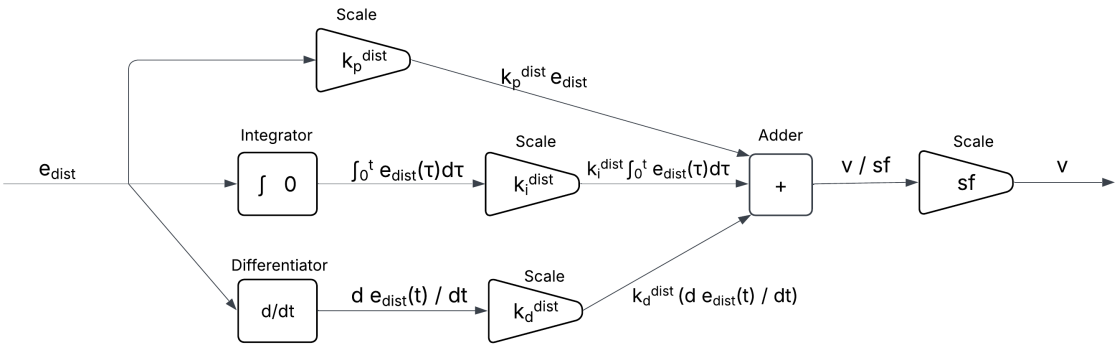
Figure 4.1 illustrates how the desired angular velocity (ω) and linear velocity (v) are deduced with the lateral error and linear error as inputs respectively. The k_p, k_i and k_d constants represent the gains for the linear, integral, and differential components of the PID calculation respectively, and the sf component represents the speed factor, another constant to modulate the linear velocity of the Duckiebot. The final equations obtained for the respective velocities are as follows:

$$\omega = -\left(k_p^{lat}\, e_{lat}(t) + k_i^{lat} \int_0^t e_{lat}(\tau)\, d\tau + k_d^{lat}\, \frac{de_{lat}(t)}{dt}\right)$$

$$v = \left(k_p^{dist}\, e_{dist}(t) + k_i^{dist} \int_0^t e_{dist}(\tau)\, d\tau + k_d^{dist}\, \frac{de_{dist}(t)}{dt}\right) \cdot sf$$



Actor Model (Block Diagram) for angular velocity



Actor Model (Block Diagram) for linear velocity

Figure 4.1: Actor models (Block diagrams) for desired angular velocity (ω) and linear velocity (v).

4.2.3 Lateral Control

The lateral controller minimizes horizontal positioning error by keeping the ball centered in the robot's camera frame. It uses the normalized x-coordinate error (ranging from -1 to 1) as input and generates angular velocity commands to achieve centering.

4.2.4 Distance Control

The distance controller maintains an optimal following distance using object size estimation. Distance is calculated using the relationship:

$$distance = \frac{\text{known object width} \times \text{focal length}}{\text{pixel width}}$$

The controller generates linear velocity commands to approach or retreat from the target.

4.2.5 Advanced System Enhancements

Advanced enhancements were incorporated to improve system performance:

- **Kalman Filtering:** A 4-state Kalman filter smooths noisy detection results and provides predictive estimates during temporary detection losses or occlusions.
- **Anti-Oscillation Measures:** Deadband logic ignores small errors to prevent micro-movements, rate limiting constrains maximum angular velocity changes, and output smoothing blends current and previous commands to reduce jitter.
- **Adaptive Control:** Distance-adaptive PID gains reduce control authority when objects are very close, and dynamic parameter adjustment occurs based on tracking performance metrics.
- **Emergency Mechanisms:** Stop zones prevent movement when objects are too close (<30cm), and obstacle detection triggers immediate safety stops.

4.3 Integration of Hardware and Software

4.3.1 Distributed Client-Server Architecture

The system employs a distributed client-server architecture to overcome the Duckiebot's computational limitations while maintaining real-time performance:

4.3.2 Server Side (MacOS Desktop)

- FastAPI web server hosting the YOLOv8 detection model as a REST API
- Detection endpoint '/detect' accepts image uploads and returns JSON responses containing object position, confidence, and estimated distance
- Real-time display showing incoming frames with detection overlays for system monitoring

4.3.3 Robot Side (Duckiebot)

- Enhanced Object Detector Node captures camera images and transmits them to the server API using asynchronous HTTP requests
- Motor Controller Node receives detection results and computes wheel velocities using PID control algorithms
- Multi-format publishing supports standard ROS message types (Twist, WheelsCmdStamped, Twist2DStamped) for compatibility with Duckiebot hardware interfaces

4.3.4 Communication Protocol

Images are encoded as JPEG with reduced quality (50-80%) and transmitted via HTTP POST requests. The system implements persistent HTTP sessions with 1-second timeouts and graceful fallback to Kalman-predicted positions during API failures.

The ROS integration follows the pipeline: Camera Node → Detection Node → Motor Controller → Wheel Drivers, where image topics are processed through API calls to generate control topics that produce velocity commands.

4.4 Challenges Faced During Implementation

4.4.1 HSV Detection Inadequacies

The initial color-based approach proved fundamentally unreliable due to environmental variability. Lighting variations caused color threshold failures, background clutter created numerous false positives, and object reflections disrupted consistent detection. This necessitated the transition to machine learning-based detection.

4.4.2 Computational Limitations

The Duckiebot's ARM-based processor could not handle real-time YOLOv8 inference, with model loading consuming excessive memory and inference times exceeding 2-3 seconds per frame. This led to the implementation of the client-server architecture.

4.4.3 Network Latency and Communication Delays

Initial implementation suffered from 10-20 second delays between object movement and robot response due to buffer accumulations in ROS image queues, synchronous API calls blocking frame processing, and high-resolution images causing network bottlenecks. Solutions included asynchronous threading, frame rate limiting, image compression, and ROS buffer optimization.

4.4.4 PID Control Oscillations

The robot exhibited severe oscillatory behavior, particularly 45-degree left-right swinging motions when the ball was close to the camera. This required implementation of distance-adaptive PID gains, enhanced deadband logic, rate limiting, and specialized stop zones for close-proximity scenarios.

4.4.5 Real-Time Performance Optimization

Low detection frame rates (initially 2-3 Hz) resulted in poor tracking performance. Optimizations included reduced JPEG quality, concurrent processing threads, elimination of debug image transmission, and optimized YOLOv8 inference settings.

4.4.6 System Integration Complexity

Coordinating multiple ROS nodes while maintaining real-time constraints required comprehensive error handling, performance monitoring systems, and graceful degradation mechanisms using Kalman predictions during API failures.

4.4.7 Final Performance Results

The final system achieves detection rates of 15-20 Hz with position errors of $\pm$0.05-0.10 normalized units and response times under 200ms, demonstrating reliable real-time object following capabilities with robust performance across diverse operating conditions.

Chapter 5

Use Cases and Applications

5.1 Human-Following Service Robots

Object-following capability enables service robots to autonomously follow humans in real time, which is crucial in environments such as hospitals, airports, and retail spaces. Robots equipped with robust object tracking can assist with tasks like guiding visitors, delivering items, or carrying luggage while maintaining safe distances and adaptive motion.

5.2 Logistics and Warehouse Applications

In logistics and warehouse environments, autonomous robots can track and follow designated objects or human operators to streamline material handling. Object-following allows robots to accompany workers, transport goods between stations, and adapt to dynamic movement in cluttered or changing layouts, improving operational efficiency and reducing manual labor.

5.3 Cooperative Transportation Systems

Object-following is a key enabler for cooperative transportation, where multiple mobile robots coordinate to move large or heavy objects. By tracking a leader robot or a moving reference object, follower robots can maintain formation, synchronize movements, and ensure safety during collaborative tasks. This approach is particularly relevant in automated manufacturing and multi-robot fleet operations.

5.4 Research and Educational Uses

Platforms like Duckietown leverage object-following as a research and educational tool to teach concepts in robotics, perception, and embedded systems. Students and researchers can experiment with perception algorithms, control strategies, and multi-agent coordination in a safe, cost-effective, and realistic environment, allowing scalable experimentation before deploying solutions in real-world scenarios.

5.5 Summary

Object-following technology has broad applicability across service, industrial, and educational domains. Its integration into small-scale platforms like Duckiebot provides valuable insights into real-time perception, control, and autonomous behavior, bridging the gap between simulation and practical deployment.

Chapter 6

Conclusion and Future Work

6.1 Summary of Contributions

This project presented the design, implementation, and evaluation of a real-time object-following system on the Duckiebot platform. The primary contributions include:

- Development of a lightweight perception pipeline capable of detecting and tracking a designated target under resource-constrained embedded conditions.
- Implementation of a feedback-based control system, including PID controllers, to maintain stable pursuit and responsive motion adaptation.
- Integration of hardware and software components, including the Duckiebot, custom PCB, camera, and motor drivers, into a cohesive closed-loop system.
- Experimental validation in both simulated and physical Duckietown environments, demonstrating robust tracking performance under standard lighting and partial occlusion scenarios.

Overall, the work demonstrates that real-time object-following is feasible on compact, resource-limited platforms while maintaining modularity, extensibility, and reliable performance.

6.2 Lessons Learned

Several practical insights were gained during the course of the project:

- Efficient perception algorithms are essential to meet real-time constraints on embedded platforms without sacrificing accuracy.
- Tight integration between perception and control is critical; delays or mismatches can degrade tracking performance.
- Environmental factors such as lighting variation, partial occlusion, and uneven surfaces significantly influence system reliability and must be considered during testing.
- Modular design principles enhance the system's adaptability, enabling future extensions or upgrades without major architectural changes.

6.3 Directions for Improvement

While the system performs reliably under standard conditions, several avenues exist for future enhancement:

- **Multi-Object Following:** Extending the perception and control pipelines to handle multiple dynamic targets simultaneously.
- **Visual Servoing:** Implementing advanced visual servoing techniques to improve responsiveness and precision in target tracking.
- **Wireless Communication:** Incorporating wireless communication modules for remote monitoring, control, or coordination between multiple Duckiebots.
- **Adaptive Learning-Based Control:** Leveraging reinforcement learning or adaptive controllers to improve performance under varying environmental conditions or target behaviors.
- **Enhanced Perception:** Incorporating depth sensors, LiDAR, or multi-camera setups to improve robustness against occlusion and lighting changes.

In conclusion, this project validates the Duckiebot platform as an effective environment for researching and prototyping autonomous object-following systems. The modular architecture, combined with real-time perception and control, lays a strong foundation for advanced research in multi-agent coordination, adaptive control, and intelligent transportation systems on embedded platforms.

Bibliography

[1] Brechtel, S., Krause, R., Buckl, C., Li, J., and Siegwart, R. “Duckietown: An open, inexpensive and flexible platform for autonomy education and research”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1497–1504. DOI: 10.1109/ICRA.2017.7989097.

[2] Brechtel, S., Krause, R., and Siegwart, R. “Multi-robot coordination and intersection handling in Duckietown”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 3452–3458. DOI: 10.1109/IROS.2018.8593812.

[3] Jing, Y., Paull, L., and Li, J. “Reinforcement learning for adaptive navigation on the Duckiebot platform”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 4512–4518. DOI: 10.1109/ICRA.2019.8794140.

[4] Markovic, I. and Petrovic, T. “Moving object tracking using sensor fusion and Kalman filtering for autonomous robots”. In: *Robotics and Computer-Integrated Manufacturing* 41 (2016), pp. 1–12. DOI: 10.1016/j.rcim.2016.01.002.

[5] Nguyen, H., Le, T., and Pham, D. “Real-time object tracking for autonomous mobile robots using lightweight deep learning models”. In: *Robotics and Autonomous Systems* 150 (2022), p. 103958. DOI: 10.1016/j.robot.2022.103958.

[6] Paull, L., Sa, I., Seto, M., and Li, J. “Vision-based autonomous navigation and obstacle avoidance in Duckietown”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1234–1240. DOI: 10.1109/IROS.2018.8593698.