Autonomous Driving Duckiebot

GROUP V

Submitted By

Xinran Shen, Ruilai Ma, Weiyuan Du, Jing Cheng,
Yijie Gao, Yingtong Tian, Huixue Chen

Under the Guidance of

Prof. Dr. Amr Alanwar

August 2025

1 Abstract

This project presents the development of an autonomous driving system for the Duckiebot platform, integrating lane-keeping, obstacle avoidance, and traffic sign recognition to emulate essential self-driving behaviors. A vision-based lane detection module employs image preprocessing and color segmentation to identify lane markings, while a PID controller ensures the vehicle remains centered. Obstacle detection enhances safety by identifying objects along the trajectory and triggering slow-down or avoidance strategies. Road sign and traffic light recognition provide semantic context, enabling the vehicle to comply with stop commands, directional instructions, and traffic signals. These perception modules are fused in a decision-making layer that governs maneuver selection, while a control layer converts high-level decisions into low-level motor commands for steering and velocity regulation. The entire framework is implemented within the Robot Operating System (ROS), ensuring modularity, real-time communication, and extensibility. Through this project, we gain practical experience in computer vision, control theory, and robotics middleware, establishing a foundation for more advanced autonomous vehicle research.

2 Problem Statement

During the development of autonomous driving functions on Duckiebot, several practical challenges emerged. Lane markings were often partially missing or distorted under changing lighting, making stable lane keeping difficult. Obstacles such as small objects required fast and reliable detection to avoid collisions without overreacting. Traffic signs and lights, being small and visually similar to background elements, demanded precise recognition for correct decision-making. At the same time, all processing had to run in real time on limited hardware. Addressing these issues required robust computer vision pipelines, well-tuned PID control, and a modular ROS-based architecture capable of handling uncertainty and dynamic road scenarios.

3 Introduction

This project implements basic autonomous driving functions on the Duckiebot DB21J platform.

The main focus is on:

- Open Loop and PID Control
- Lane keeping and Obstacle Avoidance
- Traffic light recognition and decision-making.
- Emergency stop and live monitoring

These functions establish a foundation for more advanced autonomous behaviors, such as obstacle avoidance and road sign detection.

4 Hardware Setup

- **Platform**: Duckiebot DB21J
- **Processing Unit**: Jetson Nano
- **Sensors**:
  - Camera (front-facing, for lane and traffic light detection)
  - Wheel encoders (for velocity estimation)
  - IMU (for orientation feedback, optional)
  - 8MP camera (primary visual input)
  - Time-of-Flight sensor

5 Software Architecture

The software architecture of our Duckiebot project is built on top of **ROS (Robot Operating System)** and custom Python scripts. The design follows a modular approach, separating vision, control, and decision-making into distinct layers.

5.1 Lane Detection Module

- **Preprocessing**: Each camera frame is enhanced using Contrast Limited Adaptive Histogram Equalization (CLAHE) to handle varying lighting conditions. The image is then converted into both HSV and LAB color spaces.
- **Color Filtering**:
  - Yellow lines are detected using an adaptive HSV filter.
  - White lines are extracted using a combination of HSV brightness thresholds and LAB chroma filtering, restricted to the right-hand side of the image.
- **Geometric Processing**: Processed masks are divided into slices. Within each slice, Hough Line Transform and Canny edge detection estimate lane boundaries. From this, the lane center and lane width are computed.
- **Fallback Mechanism**: If the white line cannot be reliably detected, an edge-histogram peak method is applied to approximate its location.

5.2 Lane Following and PID Control

- The lateral error (deviation from the lane center) is fed into a PID controller.
- **PID Configuration**:
  - Proportional gain ensures immediate reaction.
  - Integral term corrects accumulated bias.
  - Derivative term damps oscillations.
- The steering command is smoothed using a low-pass filter to reduce jitter.
- The speed controller modulates forward velocity inversely proportional to steering magnitude, slowing down during sharp turns.

5.3 Turn Awareness

- If one lane marking (yellow or white) is missing for a set number of frames (`disappear_hysteresis`), the system enters **Turn Mode**.
  - Left Turn: If the white line disappears, the robot follows the yellow line with inward bias.
  - Right Turn: If the yellow line disappears, the robot follows the white line with inward bias.
- During Turn Mode, an additional steer bias and speed reduction are applied for stability.
- A ramp gain is used at the beginning of turns to strengthen steering, then gradually decays.

5.4 Recovery and Re-Acquisition

- If both lane markings disappear for too long (`lost_frames_to_search`), the system enters **Recovery Mode**.
- The robot first steers slightly towards the last known lane side.
- If still unsuccessful, it performs a sweeping search with alternating steering at low speed until lane lines are detected again.

5.5 Traffic Light Detection

- A Region of Interest (ROI) is defined in the upper central part of the camera frame.
- The image is converted to HSV color space, and thresholds are applied to detect red and green regions.
- Hough Circle Transform is applied to confirm circular light blobs.
- **Behavior**:
  - Red light $\rightarrow$ stop immediately (speed = 0.0).
  - Green light $\rightarrow$ resume lane following.

5.6 ROS Integration

- The main node `lane_keeping_pid_turnaware_node.py` wraps the controller into a ROS interface.
- **Subscriptions**: `/camera_node/image/raw` (camera input).
- **Publications**: `/wheels_driver_node/car_cmd` (motor command), `/lane_keeping/debug_image` (debug visualization).
- ROS parameters allow runtime configuration of PID gains, vision thresholds, and turn parameters.
- On shutdown, an emergency stop is triggered by publishing zero velocity to both wheels.

6 Experimental Results

- Lane following works reliably on straight roads and smooth curves.
- Turn awareness improves stability in intersections and sharp corners.
- Traffic light recognition successfully stops at red and resumes at green.
- Web GUI allows remote monitoring and emergency stop.

7 Challenges and Limitations

1. **Lighting Sensitivity** – Lane and traffic light detection are affected by shadows, reflections, and low light conditions.

2. **Turning Stability** – Turn mode works, but sharp turns can still cause temporary drift.

3. **Traffic Light Reliability** – Detection may fail under long distance or bright background.

4. **Computational Load** – Raspberry Pi struggles to maintain high FPS when running both vision and control.

8 Conclusion and Future Work

This project successfully demonstrates:

- Lane following with PID + turn awareness.
- Traffic light detection and decision-making.
- Safety integration with emergency stop.

Future work will extend the system to:

- More robust perception with deep learning methods.
- Multi-robot coordination in Duckietown.