UNIVERSITY OF GREENWICH   
FACULTY OF ENGINEERING AND SCIENCE  
SCHOOL OF COMPUTING AND MATHEMATICAL SCIENCES

**Self-Driving Cars with ROS2 and Computer Vision**

**Iustin-Andrei Moisa-Tudor**

**001228763**

Supervisor: **Tuan Vuong**

**Word count**: **Insert number**

**COMP1682 Final Year Individual Project**

*A dissertation submitted in partial fulfilment of the requirements for the degree of*

***BSc (Hons) Computer Science***

April 25

Abstract

This project focuses on the development of a modular vehicle simulation platform using ROS 2 Humble and Gazebo Classic, with a focus on 3D modeling, sensor integration, and an overall functional system architecture. The main objective was to create a virtual environment in Gazebo where a simulated car could be controlled and evaluated either by using manual teleoperation or full self-driving capabilities, enabling the experimentation with perception and control algorithms for the car in a safe, virtual environment.

The implementation involved the process of designing custom 3D models for the road, traffic lights, traffic signs, and environmental features in Blender, and integrating these into a Gazebo-based virtual environment world where the car will be simulated in. A Prius vehicle model was adapted and enhanced with the addition of the Ackerman steering mechanism for realistic motion and also equipped with a camera sensor to support computer vision tasks such as recording and lane segmentation. The entire software was structured around ROS 2 Humble (Robotic Operating System), and using nodes which allowed for clear separation between perception, control, and data handling modules or nodes in the code.

Some key contributions include the creation of a vision based lane detection node using OpenCV for color segmentation and morphological filtering, providing the vehicle with the ability to identify lane markings and distinguish them. The project features a video output module for recording the simulation runs and supports both manual teleoperation and automated control modes. Testing was conducted at the unit and integration levels to ensure that the communication between the ROS nodes and modules is reliable and to make sure everything works with no errors. The findings demonstrate that the system is effective for simulating and evaluating vehicle perception and control strategies, though challenges remain in handling complex scenarios and ensuring compatibility across software components.

Blue text on a black background

Description automatically generated with medium confidence

Declaration of AI Use

Please complete this part when you have used AI during the process of undertaking this assignment to acknowledge the ways in which you have used it.

I have used AI while undertaking my assignment in the following ways:

* To develop research questions on the topic – YES/NO
* To create an outline of the topic – YES/NO
* To explain concepts – YES/NO
* To support my use of language – YES/NO
* To summarise the following articles/resources:  – YES/NO



* In other ways, as described below:  – YES/NO

Acknowledgements

It is customary to acknowledge any substantial help, with either the project work or the report, from people and other informal sources.

Contents

Abstract ii

Declaration of AI Use  iii

Acknowledgements iv

Contents v

1 Introduction 1

1.1 Background 1

1.2 Aims and Objectives 2

1.3 Methodology 2

1.4 Report Roadmap 3

2 Literature Review 4

2.1 Introduction 4

2.2 Some major research theme/topic/area 5

2.3 Another major research theme/topic/area 5

2.4 More major research themes/topics/areas 5

2.5 Conclusion 6

3 Product Review (if applicable) 7

3.1 Introduction 7

3.2 Product 1 7

3.3 Product 2 7

3.4 Key findings 7

3.5 Conclusion 7

4 Requirements Analysis and Design 8

4.1 Introduction 8

4.2 Requirements Analysis 8

4.3 Design 9

4.4 Conclusions 10

5 Implementation 11

5.1 Introduction 11

5.2 A major section relevant to system/prototype implementation 11

5.3 Another major section relevant to system/prototype implementation 11

5.4 More major sections relevant to system/prototype implementation 12

5.5 Testing 12

5.6 Conclusion 12

6 Results and Evaluation 13

6.1 Introduction 13

6.2 Results 13

6.3 Evaluation 14

6.4 Conclusion 15

7 Legal, Social, Ethical and Professional Issues 16

8 Conclusion 17

8.1 A major section summarising the work 17

8.2 Another major section summarising the overall outcomes 17

8.3 More major sections relevant to the overall conclusion 17

8.4 Future Work 17

Appendix A - Project Proposal 20

a.1 Revised Proposal/Project Aim (if applicable) 20

a.2 Revised Project Plan 20

a.3 Original Proposal 20

a.4 Project Plan 20

Appendix B - Xxxxx Yyyyy 21

b.1 A major section in Appendix B 21

b.2 Another major section in Appendix B 21

b.3 More major sections in Appendix B 21

# Introduction

## Background

In the past decade, autonomous driving and self-driving cars have been improving more and more each year and are set to redefine the future of driving and safety on the road as well as lower pollution and offer more accessibility to users defined as non-fit to drive such as disabled, injured, or older individuals.

As the amount of self-driving vehicles on the market increases each year, the complexity of reaching a highly reliable, safe, and efficient driving system increases significantly. Some of the tasks performed by a Neural Network (NN) powered autonomous system are interpreting multiple data types from different sensory inputs such as data from LiDAR, Infra-red cameras, and other onboard systems as well as exploring dynamic environments in order to make fast and reliable decisions.

Humans make precise decisions when driving due to years of experience on the road, but are prone to life-threatening accidents due to lack of attention, fatigue, and driving for long periods of time.

For autonomous vehicles to prevent these issues and operate at a level surpassing the average human, a system designed for self-driving cars must integrate data from sensors such as LiDAR, radars, cameras, and GPSs to reliably assess live situations and make safe decisions.

Systems for autonomous driving are playing a huge part in changing the way we use transportation, as they can seriously decrease the risks of life-threatening accidents, congestions, and traffic incidents.

They have the ability to make our roads safer, more accessible for certain individuals, and reduce environmental pollution by promoting the use of electric vehicles.

## Aims and Objectives

This project's main focus is designing and implementing a safe and reliable driving system for self-driving cars simulated inside a Gazebo Environment, using the latest software, ml and sensor fusion tools to achieve a highly performant self-driving system. The development will take place on a 2023 M3 Apple MacBook Air 15” with M3 8 core CPU, 10 core GPU, 16GB Unified Memory, the project is developed on Ubuntu 22.04 LTS inside a VMware Virtual Machine on the local macbook which should provide relatively good performance in terms of development.

One of this project's aims is to build a self-driving system that uses real-time computer vision capabilities capable of accurately detecting and tracking and segmenting different objects placed inside the simulated virtual world as well as vehicles and obstacles around the autonomous vehicle.

This will be achieved using Gazebo, Classic which is one of the most versatile and widely used robotic simulation platform that integrates seamlessly with ROS2 (Robot Operating System), providing a reliable 3D simulation environment for testing autonomous driving algorithms, different sensors integrated and various robotic nodes implemented into the program behind the car.

## Methodology

This project uses an iterative development and prototyping approach for developing the car simulation environment and vehicle itself, this methodology is well-suited for robotics and autonomous systems since it enables continuous refinement through different cycles of implementation, testing, and improvement of each modular code piece.

The decision to use ROS 2 Humble as the primary framework for the car is due to the fact that it is one of the most stable ROS 2 verison that is capable to run on ARM64 architecture which is used by the M3 chip inside the macbook used for development.

ROS2 provides a range of significant advantages for autonomous driving applications and software due to its node based architecture that separates perception, decision-making, and control modules (**Reke et al., 2024**). This modular approach makes maintainability, testability, and extensibility of the system easier and requires less resources to run on compared to ROS1.

Gazebo Classic is the selected software for the simulation environment because of its accessibility, and seamless integration with ROS 2 Humble, despite more advanced alternatives like CARLA and NVIDIA Drive SIM (Donzia and Kim, 2022) environments which were initially the target softwares to use, however since the software was developed on a ARM MacBook instead, none of those could be used due to the lack of NVIDIA GPU / Windows 11 software libraries being incompatible with the ARM64 architecture.

The use of a custom 3D environment in Gazebo, instead of using a pre-built simulation environment, was chosen because of the need for controlled testing environments for autonomous systems where a custom environment gives full control to the developer over the models/joints and other elements in the simulation since it was developed from scratch (Song et al., 2024).

For the lane detection system, traditional computer vision techniques (like the use of OpenCV) were chosen over deep learning approaches, as research has shown that these methods can be highly effective and bring in similar performance for specific tasks like lane detection when computational resources are limited, since the program is developed on a mid-range macbook inside a virtual machine which takes up resources even more (De-Las-Heras et al., 2021).

The development process followed a bottom up approach, starting with the creation of basic components and gradually integrating them into the system. This staged methodology allowed for continuous validation of each component, ensuring that issues could be identified and addressed early in the development process.

## Report Roadmap

This report documents the end-to-end development of a car simulation system using ROS 2 Humble and Gazebo Classic, going from concept to the final implementation of the software. The report begins with an Introduction that establishes the context of autonomous vehicle simulation, outlines the project's aims and objectives, and explains the methodological approach.

This is followed by a Literature Review which will be examining current and past key research in vehicle simulation, computer vision techniques, and ROS 2 frameworks, providing the foundation for the project. The requirements analysis and design then translates this research into more concrete architectural and system plan, going into both functional and non-functional requirements alongside the system's design.

The implementation section will form the core of the report, describing the development of the 3D models, ROS 2 packages, and lane detection algorithms, while also documenting the testing procedures that validated the system's functionality.

After which, in the Results and Evaluation section, it will present the outcomes of the project and current development state, assessing the system's performance and identifying both strengths and limitations of the software.

The evaluation part leads into a discussion of Legal, Social, Ethical and Professional Issues relevant to autonomous vehicle simulation, considering broader implications of the technology. And at the end of the report, the conclusion will summarise the project's development process and achievements, reflect on lessons learned, and proposes directions for future work that could enhance the system's capabilities as opposed to the current stage of development.

# Literature Review

## Introduction

The literature search is focused on the recent years (last 5-6 years) advancements in the autonomous driving system domain as well as simulation environments and applications for robotics, deep learning techniques for object detection and tracking and sensor fusion. The sources that were found during the research part were a mixture of documents such as different academic journals and technical reports from leading institutions in the autonomous vehicle field and also the robotics field as it plays a crucial part in this project.

Some of the main areas discussed in this report include pedestrian detection and safety enhancements for autonomous vehicles, deep learning neural networks, artificial neural networks, driving terrain simulations in softwares that integrate ROS2 (robotic operating system 2) such as ROS2 in languages like Python.

Distributed system architecture and development nodes, multiple object tracking in real time and data reading fusion for different types of sensors onboard of the autonomous vehicle to be developed, these include LiDAR sensors and also RGB cameras or radios which improve model perception, synthetic datasets will be used for model training and AI evolution over the past decade will also be discussed.

The challenges that are mainly found in the development of autonomous self-driving software systems include ensuring safety and reliability in diverse weather and lighting conditions, accurate real-time object detection and tracking during operations, efficient and non resource-intensive sensor fusion for improving perception and vehicle awareness, developing decision making algorithms which will be used in obstacle avoidance and navigation on roads and creating realistic simulation environments (also known as digital swing) of the real world to test the decision making and perception modules through simulations.

## Using Deep Learning Neural Networks for Enhanced Object Detection

Modern deep learning techniques such as Convolutional Neural Networks (CNNs) and You Only Look Once (YOLO Models) have been playing an essential part in developing highly accurate real-time computer vision systems which are used nowadays in current autonomous driving systems.

Donzia and Kim (2022) have shown the effectiveness of these models with CARLA (environment simulator) where they documented and created logs of the deep learning performance and the accuracy on classifying various different objects using these neural network models. CNNs and YOLO models enable high-speed image recognition and precise object localization which are beneficial in areas such as urban areas.

However there are challenges regarding maintaining accuracy rate in different scenarios such as different weather and lighting situations.

## Pedestrian Detection and Safety Enhancements

When navigating urban areas, given the safety implications in highly populated areas such as large cities, pedestrian detection is one of the most critical functions in an autonomous vehicle system. Long-range pedestrian detection is capable of giving self-driving systems more time to respond safely during the real-time decision-making process in the AI model, which helps in reducing road accidents involving pedestrians and mainly children.

Fürst et al. (2024) was able to combine LiDAR and RGB sensors, which allows autonomous vehicle systems to improve their detection range and the model accuracy. This integration approach is indeed complementing the rising need for comprehensive pedestrian safety features found in AV systems.

As reported in a recent article by The Independent (2023), showing the number and rate of casualties on the road caused by human error in the UK during 2023, pedestrian road deaths have risen by 6% compared to 2022.

Improved pedestrian detection systems implemented into modern autonomous vehicles could be a significant factor in addressing this substantial increase in road accidents involving pedestrians and drivers themselves. Integrating LiDAR and RGB sensors offer enhanced accuracy for pedestrian and obstacle recognition but the high cost and increased computational requirements of LiDAR sensors and systems remain as a limitation to widespread implementation.

## Realistic Traﬃc Simulation Environments

One of the most essential tools for testing and validating autonomous systems meant to be used in self-driving vehicles are the simulation environments as they allow researchers to expose their algorithms to various simulated scenarios before making them onto the road in real life.

Donzia and Kim (2022) managed to use CARLA (Tool for Simulation Environment) to create a life- like driving environment to allow their deep learning systems to be trained across various environments and weather conditions. As a simulation software, CARLA provides a wide range of simulated environments which include scenarios like highways, intersections, and busy traffic. CARLA Simulator enables a comprehensive assessment and evaluation of autonomous vehicle systems by testing their robustness and adaptability.

NVIDIA's Bi-Level Imitation for Traffic Simulation (BITS) expands on this by incorporating imitation learning to simulate realistic traffic behaviors similar to what digital twins do (NVIDIA Research, 2023). BITS uses a bi-level hierarchical structure to separate high-level intent prediction from low-level control, allowing the self-driving AI model to replicate complex traffic patterns with a high fidelity level.

Simulation environments like CARLA, BITS or NVIDIA Drive SIM allow developers to test systems under controlled conditions which reduce the cost of physical testing however simulations may lack the unpredictability of real-world driving.

## Distributed System Architectures and Robotics Operating Systems

Reke et al. (2024) managed to implement a self-driving car architecture in ROS2, which demonstrated the benefits of modularity by enhancing system reliability and adaptability. ROS2's real-time processing capabilities ensure that autonomous vehicles can handle the computational loads required for critical safety functions like trajectory prediction and object avoidance without compromising on the model's performance. Another benefit of using ROS2 for autonomous vehicles is that the ROS2 nodes can always be updated and expanded upon, which now supports the integration of new algorithms as technology evolves.

This technology provides a strong foundation for developing reliable autonomous systems. A drawback of ROS2 is that in order to run the modular framework with real-time processing, it requires high- power computational hardware which are not exactly cost effective.

## Real-time Multiple Object Tracking

Pang et al. (2021) introduced Quasi-Dense Tracking (QDTrack), which is a Multiple Object Tracking (MOT) approach that uses quasi-dense similarity learning for improving real-time tracking performance. QDTrack's performance on benchmarks such as MOT17, BDD100K, and Waymo demonstrates its efficiency and reliability in sensing dynamic environments (Pang et al. 2021).

The ability to use Multiple Object Tracking to reliably track multiple objects is essential in using it in highly populated urban areas such as large cities. This technology focuses primarily on appearance- based similarity, unlike traditional tracking methods, which allows for a more adaptive response to real- world scenarios. This also helps a lot with responding to potential hazards more efficiently.

QDTrack's appearance-based similarity focus allows for responsive tracking in diverse dynamic environments but achieving optimal performance and accuracy in high traffic areas with complex visuals obstructions remains challenging.

## Data Fusion for Improved Perception

Fürst et al. (2024) developed a Long Range Pedestrian Detection (LRPD) system which uses the specific strengths of the LiDAR system and RGB cameras to enhance the pedestrian detection accuracy over longer distances. This approach allows the autonomous vehicle to detect pedestrians with a high level of precision even at night, when the LiDAR sensor comes in handy. LRPD's focus on integrating the RGB and lidar sensors together offers a high improvement over traditional single-sensor systems.

Integrating multiple sensors into a sensor fusion algorithm increases system complexity and power demands which will limit the scalability for wide-domain adoption.

## Synthetic Datasets for Model Training

Song et al. (2024) conducted a survey on the use of synthetic datasets in autonomous driving to demonstrate their value in simulating rare and hazardous scenarios that are difficult to capture during on-road testing.

By using synthetic datasets, Autonomous Vehicle datasets can train the AI models behind the self driving cars on unseen data so that they are ready for any possible hazard when being deployed in the real-world.

Synthetic datasets allow users and developers to model care edge scenarios without the risk and cost of real-world data collection

## ML Techniques For Advanced Driver Assistance Systems (ADAS)

De-Las-Heras et al. (2021) explored ADAS implementations which are mainly focussing on on detecting and transcribing message signs on roads so the Self driving system can respond to any dynamic environment properly. ML Systems such as the ones found in ADAS, play an important role in detecting and reacting to a wide range of road signs which then enable better and more reliable AV navigation.

These ML ADAS techniques improve the situation awareness and responsiveness of the autonomous AI model in real-time changes in traffic. ADAS enhances road navigation and safety in semi-autonomous vehicles but limitations in detecting complex road conditions and changes remain in place.

## Artificial Intelligence Evolution in Autonomous Vehicle Industries

Garikapati and Shetiya (2024) discussed the progressive the integration of new AI software into existing AV systems since they play a critical role in areas such as decision-making, hazard perception and vehicle control.

As newer and newer technologies in the AI sector start to appear, the reliance on ML (Machine Learning) models especially in the multi-sensory data such as LiDAR and RGB Cameras has increased dramatically.

AI systems integrated into autonomous systems enable them to detect, classify and predict the behaviour of surrounding objects, pedestrians, and vehicles and also improves the overall safety and reliability of the Self driving cars while always having room from improvements as the technology advances. AI multi-sensory perception increases autonomous vehicle reliability but requires substantial computational power and sophisticated algorithms

## AI (Artificial Intelligence) in Self-driving Vehicles

Yurtsever et al. (2020) provided a detailed survey of common practices in the AI and AV (autonomous vehicles) field including object mapping, sensor fusion, object and hazard perception and avoidance and much more.

Deep Neural Networks (also known as DNNs) have been one of the most crucial emerging technolgies allowing autonomous vehicles to process complex sensory data for highly accurate, reliable and safe decision making.

The combination of multi-sensor data fusion and tracking algorithms (QDTrack) enhances the self-driving vehicle's ability to detect and perceive interactions various obstacles, pedestrians and other objects that might appear in a dynamic driving environment.

## Conclusion

The literature review highlights how deep learning techniques such as YOLO, CNNs etc have evolved over the past decade and have shown significant advancements in real-time object detection and recognition but the accuracy metrics still remain on the lower side due to different environments and lighting and weather conditions which affect model performance.

The integration of multiple sensors such as LiDAR and RGB cameras has improved tthe detection

accuracy and tracking of different objects especially pedestrians, ensuring their safety. Simulation environments like Copelia SIM, NVIDIA BITS and CARLA have proven invaluable for testing and validating autonomous self driving systems in different conditions such as rain, ice etc. Distributed system architectures with ROS2 (Robotic operating system 2) have enhanced the reliability of the system and these can be used in either NVIDIA ISAAC SIM or Copellia SIM to handle different tasks.

The use of synthetic datasets has become a powerful tool for training models on rare and hazardous scenarios where real-world data might not be enough to make accurate decisions.

This sensor fusion approach compensates for the limitations of individual sensors, providing more robust environmental perception. Simulation environments including GAZEBO, CoppeliaSim, NVIDIA BITS, and CARLA have emerged as essential tools for validating autonomous driving systems.

These findings are helpful in this project's implementation strategy in several ways.

First, the project will leverage ROS2's architecture to create a modular autonomous driving system with separate nodes for perception, decision-making, and control.

Second, rather than attempting to build a comprehensive deep learning solution that might struggle with environmental variability, the implementation will focus on computer vision techniques optimized for controlled simulation environments.

Third, the project will utilize Gazebo Classic as the simulation platform, providing a balance between realism and computational efficiency for testing autonomous behaviors across various scenarios.

By applying these approaches, the project aims to develop a functional autonomous car simulation that demonstrates reliable lane detection and basic navigation capabilities while acknowledging the inherent limitations in current autonomous driving technologies

# Requirements Analysis and Design

## Introduction

The development of an the car simulation and the system system requires careful consideration of both technical capabilities and user needs. This chapter documents the systematic approach taken to identify, analyze, and prioritize the requirements for the project development, after which come the design decisions that will be implemented into the final implementation.

The requirements analysis creates a rough idea of what the system should accomplish, while the design section will outline the architectural and component-level decisions made to fulfill these requirements.

## Requirements Analysis

### Requirements

The requirements for the project were chosen through combinign the knowledge gained from the literature review, an analysis of existing simulation platforms and current technical constraints.

The literature review gave insights into current approaches in autonomous vehicle simulation, showing the importance of modular architectures, physics modelling, and computer vision capabilities for for proper performance.

After an analysys of existing 3D simulation platforms such as CARLA, NVIDIA Drive SIM, and Gazebo showed they have common features and limitations. While commercial platforms like CARLA offer high visual fidelity and comprehensive sensor simulation, they likely often require significant computational resources in order to run the simmulations and provide less flexibility for customization.

Open-source alternatives like Gazebo offer better adaptability but typically with less realistic visuals and physics, but this simulator is an excelent choice for devices with low computational power.

Some technical constraints were identified with extra research related to ROS 2 Humble and NVIDIA ISAAC Sim and Carla on Ubuntu 22.04, revealing a few compatibility challenges that do not make them best suited for the development machine. The chosen software to go with is Gazebo due to its limited resource demand and because it’s an open sourced software.

### Functional Requirements

The functional requirements are using the MoSCoW method to distinguish between essential and non-esential requirements:

Must Have:

* Create a 3D simulation environment with a road
* Implement a 3D vehicle model with steering capabilities
* Develop a camera sensor system for capturing visual data from the vehicle
* Implement basic lane detection using computer vision techniques
* Provide manual control capabilities also knows as teleoperation.

Should Have:

* Support for different road segments
* Show lane detection in real-time
* Recording capability for camera data

Could Have:

* Traffic sign detection and response
* Advanced autonomous behaviors

Won't Have:

* Deep learning-based perception systems
* Multi-sensor fusion (LiDAR, radar integration)
* Real-time traffic simulation with multiple agents
* Hardware-in-the-loop testing capabilities

These requirements priotize developing core functionality first, with additional features implemented if the resources are enough.

### Non-functional Requirements

Must Have:

* Modular software architecture using ROS 2 nodes
* Machine with Ubuntu 22.04 and ROS 2 Humble
* Sufficient documentation for system operation and extension

Should Have:

* Clean code organization
* Reusable components
* Efficient resource utilization

Could Have:

* Extensible plugin architecture
* Configuration options for adjusting simulation parameters

Won't Have:

* Commercial-grade visual fidelity
* Distributed simulation capabilities

## Design

## **System Architecture**

The system architecture was designed to leverage the distributed node-based structure of ROS 2, separating concerns between perception, decision-making, and control functions.

This modular approach enables independent development and testing of components while facilitating future extensions.

The high-level architecture consists of the following components:Gazebo Classic provides the 3D simulation environment, physics engine, and sensor simulation capabilities.

Vehicle Model - Prius model with Ackerman steering mechanism and a front-facing camera sensor.

ROS 2 Nodes:

* 1. driving\_node - Implements basic autonomous driving functionality
  2. carVision\_node - Processes camera data for lane detection and driving decisions
  3. videoOutput\_node - Handles recording and storage of camera data

Lane detection Module will implement computer vision algorithms for identifying lanes

Launch system will take care of the startup and configuration of all system components.

This architecture separates functional areas, allowing for independent development and testing of components.

The simulation environment will be designed to provide a realistic environment ground for autonomous driving while remaining computationally efficient.

This can include:

Custom 3D Models such as road segments, traffic signs, and environmental objects were designed as modular components that could be arranged to create diverse driving scenarios.

Physics such static objects were configured with appropriate collision properties,

The vehicle model was based on the Gazebo Prius model, with several modifications to enhance its suitability for autonomous driving simulation:

The steering mechanism to implement Ackerman driving, providing more realistic turning behavior than simple drive models.

A front-facing camera was positioned to provide an optimal view of the road ahead, with parameters tuned for lane detection.

The model will be configured to accept velocity and steering commands through ROS 2 topics and also implement teleoperation / manual control

These modifications ensure that the vehicle will behave realistically in the simulation environment while providing the necessary sensory input for autonomous driving algorithms.

The lane detection system will identify lane markings in camera images using traditional computer vision techniques:

The software will be organized into ROS 2 packages to facilitate development, testing, and deployment:

fyp\_autonomous\_car: Main package (Python) containing

* 1. Launch files for starting the simulation
  2. Source code for ROS 2 nodes
  3. Lane detection module
  4. Configuration files

fyp\_models: CMake package for managing 3D models:

* 1. Model files for Gazebo
  2. Installation scripts for model deployment

### 

## Conclusions

The requirements analysis and design process helped build a clear starting point for implementing the autonomous car simulation project.

By using the MoSCoW method, the project scope was defined to focus on essential features and still focus on identifying potential improvements for future development.

The modular architecture, built around ROS 2's nodes, provide a flexible and extensible foundation for implementing the identified requirements

The design decisions reflect a balance between functionality, performance, and development constraints, creating a working system that demonstrates core autonomous driving capabilities over advanced features that would require additional resources. **Implementation**

## Introduction

The following chapter dives into the implementation of the autonomous car simulation project developed using ROS 2 Humble and Gazebo Classic. The implementation consists of the creation of 3D models, development of ROS 2 packages, integration of computer vision.

The project was developed from scratch, with the exception of the Prius car model, which was obtained from the GitHub Gazebo models directory. The project repository is structured as folows with folders and sub-folders:

- fyp\_autonomous\_car (main ROS 2 package)

- launch/

- gazebo.launch.py (launch file for the simulation environment)

- src/

- driving\_node.py (basic autonomous driving functionality)

- carVision\_node.py (computer vision-based driving)

- videoOutput\_node.py (camera data processing and storage)

- DetectionModule/

- Lane/

- colorSegmentation.py (lane detection algorithms)

- drivingRobot.py (control logic based on lane detection)

- laneDetection.py (lane detection implementation)

- models/ (3D models for the simulation)

- road segments

- traffic signs

- traffic lights

- other environmental objects

- Worlds/ (Gazebo world files)

- fyp\_models (auxiliary ROS 2 package for model management)

- models/ (copied models for Gazebo integration)

- CMakeLists.txt (configuration for model installation)

- package.xml (package metadata and dependencies)

**3D Model Development and Environment Setup:**

To start, the implementation began with the creation of custom 3D models in Blender. These models included road segments, traffic signs, traffic lights, and other environmental objects necessary for creating a realistic urban driving scenario.

After creating the base models in Blender, they were imported into Gazebo forthe simulation environment creation. In Gazebo, the models were exported as sdf files with different joint configurations, and static variables were set to true for non-moving objects such as traffic signs and road segments so they would not be affected by gravity.

Custom textures created in Canva were applied to the models as PNG images. These textures provided realistic appearances for road surfaces, signage, and other environmental elements, contributing to a more immersive simulation environment.

The Prius car model, which is the autonomous vehicle car, was downloaded from the GitHub Gazebo models directory and integrated into the project. The model was placed in a new priusHybrid model folder within the project structure, and necessary modifications were made to adapt it to the project requirements.

For the Prius model the integration of the Ackerman driving was added from the Gazebo code from the GazeboRosAckermanDrive open-source documentation. This required modifications to the front left and right wheel names in the model.sdf file to match the Prius hybrid car model's configuration. The Ackerman driving mechanism provides more realistic vehicle movement, simulating the steering geometry of real vehicles.

A camera sensor was added to the car using the logical\_camera\_sensor template from GazeboSim classic documentation. The camera position and geometry were adjusted to provide an optimal view of the road ahead, which is essential for the vision-based autonomous driving functionality.

**ROS 2 Package Development**

The project was organized as an ament\_python catkin package to provide the necessary ROS 2 integration. This package structure facilitates the management of dependencies, launch files, and nodess.

The setup.py file was configured to include package metadata, dependencies, and entry points for the ROS 2 nodes. This file takes care of the build system, specifying how the package should be built and installed. The entry points defined in this file make the nodes executable through the ROS 2 command-line interface.

A separate package named fyp\_models was created using the ament\_cmake build type. This package automates the copying of models from the src/models directory to the share directory, eliminating the need for manual copying each time the project is built.

The CMakeLists.txt file in the fyp\_models package was configured to point to the model directory as the source and set the destination to share/{Project\_Name}. Additionally, the package.xml file was modified to include a gazebo\_ros gazebo\_model\_path tag, specifying the path to the models directory.

**Simulation Environment and Launch Configuration**

The simulation world was created in Gazebo by arranging the custom 3D models along with some standard Gazebo house models to create a realistic urban environment. Lights were added to the traffic light model to simulate traffic control, and the complete world was saved in a dedicated Worlds directory under the src folder.

The gazebo.launch.py file was configured to set the GAZEBO\_MODEL\_PATH to include the models folder, start Gazebo with the specified world file, and configure the nodes for autonomous driving. This launch file starts the simulation.

The simulation can be launched using the command:

**ros2 launch fyp\_autonomous\_car gazebo.launch.py**

Once the simulation is running, the Prius car can be manually teleoperated using the keyboard with the command:

**ros2 run teleop\_twist\_keyboard teleop\_twist\_keyboard**

**Autonomous Driving Implementation**

The autonomous driving functionality was implemented through several interconnected nodes, each responsible for specific aspects of the autonomous driving system.

The driving\_node.py file implements basic autonomous movement capabilities. This node subscribes to sensor data and publishes velocity commands to control the car autonomously. It implements basic movement logic to navigate the car through the environment without relying on visual input.

For more advanced autonomous capabilities, computer vision system was implemented. The carVision\_node.py file was created to process the camera feed and make driving decisions based on visual input.

This node imports the OpenCV library (cv2) for computer vision capabilities, along with CVBridge from the cv\_bridge package to allow ROS 2 to communicate with OpenCV.

To handle the video data, the vision system was separated into two nodes: carVision\_node for processing the camera feed and making driving decisions, and videoOutput\_node for handling the saving of output data and video from the camera. The videoOutput\_node saves the camera recordings to the videoOutputsCar directory inside the src folder.

**Lane Detection System**

A lane detection system was implemented to enable the autonomous car to identify and follow road lanes. This system is organized in a dedicated folder structure:

- DetectionModule folder under the fyp\_autonomous\_car package

- Lane subfolder within DetectionModule

- colorSegmentation.py file for lane segmentation code

The color segmentation code implements:

- Color space conversion from RGB to HSV for better color-based segmentation

- Color thresholding for lane marking detection, with different parameters for white middle lines and yellow edge lines

- Morphological operations for noise reduction, enhancing the quality of the segmentation

- Contour detection and filtering to identify and track lane markings

The segmentColor.py file contains several key functions that form the core of the lane detection system:

- mask\_edge creates a mask for the edges of the image to focus processing on relevant areas

- segment\_middle\_lane detects and segments the white dashed line in the middle of the road

- segment\_outer\_lane detects and segments the yellow solid lines on the edges of the road

When running the simulation with the carVision node, the system displays:

- Live feed from the camera

- White segmentation window showing the road's middle dashed lane

- Yellow segmentation window showing the edges of the road

## Testing

**Unit testing** was conducted on individual components of the system to ensure their correct functionality. The primary focus was on testing the lane detection algorithms, as they form the core of the autonomous driving capability available in the system.

The colorSegmentation.py module was tested with various sample images representing different lighting conditions and road markings. These tests verified that the color thresholding parameters were appropriately set to detect both white middle lines and yellow edge lines across various scenarios.

The mask\_edge, segment\_middle\_lane, and segment\_outer\_lane functions were individually tested to ensure they correctly processed input images and produced the expected output masks. These tests confirmed that the functions could accurately identify lane markings even in the presence of noise and varying lighting conditions.

**Integration Testing**

Integration testing focused on verifying the correct interaction between different components of the system. The integration between the carVision\_node and the lane detection module was thoroughly tested to ensure that the lane detection results were correctly interpreted and translated into appropriate driving commands.

The communication between ROS 2 nodes was tested to verify that messages were correctly published and subscribed to. This included testing the transmission of camera data from the Gazebo simulation to the carVision\_node, as well as the publishing of command velocities from the carVision\_node to control the car.

The integration of the Ackerman driving mechanism with the Prius car model was tested to ensure that the car responded correctly to steering commands, simulating realistic vehicle movement.

**Usability Testing**

Usability testing was conducted to evaluate the ease of use and effectiveness of the system from an end-user perspective. This included testing the manual teleoperation capabilities using the keyboard, as well as the autonomous driving functionality.

The system's response to various driving scenarios was evaluated, including straight road segments, curves, and intersections. The testing revealed that the autonomous driving system performed well on straight road segments but faced challenges in more complex scenarios such as sharp turns and intersections.

The visualization of lane detection results was assessed for clarity and usefulness. The white and yellow segmentation windows provided valuable insights into the system's perception of the road, helping users understand the decision-making process of the autonomous driving system.

## Conclusion

The testing process revealed both strengths and limitations of the implemented system. The lane detection algorithms demonstrated good performance in identifying road markings under controlled conditions, and the integration between different components of the system was generally successful.

However, several challenges were identified, particularly related to the compatibility between Ubuntu 22.04, ROS 2 Humble, and Gazebo Classic. These compatibility issues prevented the full implementation of some advanced autonomous features planned for later stages of development.

Despite these challenges, the testing confirmed that the core functionality of the autonomous car simulation was successfully implemented, providing a good foundation for future enhancements and refinements.

# Results and Evaluation

## Introduction

This chapter presents the outcomes of the autonomous car simulation project and evaluates its performance against the initial objectives. It provides a comprehensive assessment of the project's achievements, limitations, and potential for future development. The evaluation is based on both quantitative metrics and qualitative observations, offering insights into the effectiveness of the implemented solutions and their practical implications.

## Results

A realistic urban driving environment was created in Gazebo, featuring a custom-designed road traffic signs, traffic lights, and other environmental objects. The environment provides a comprehensive testing ground for autonomous driving algorithms, simulating various driving scenarios and challenges.

The Prius car model was successfully integrated into the simulation environment, with the addition of a camera sensor and the implementation of the Ackerman driving mechanism. The car responds to driving commands, providing a faithful simulation of real-world vehicle dynamics.

The system allows for manual control of the vehicle using keyboard inputs, enabling users to navigate the car through the simulated environment. This functionality provides a baseline for comparing autonomous driving performance with human control.

The driving\_node successfully implements basic autonomous movement capabilities, allowing the car to navigate through the environment without human intervention. The car can maintain a steady speed and follow simple predefined paths.

The carVision\_node successfully processes the camera feed from the car, enabling vision-based decision making. The integration of OpenCV with ROS 2 through CVBridge provides a robust foundation for implementing advanced computer vision algorithms.

The lane detection system successfully identifies both the white middle lane markings and the yellow edge lane markings. The system performs well under controlled lighting conditions and with clear lane markings, providing reliable input for autonomous driving decisions.

The videoOutput\_node successfully captures and saves the camera feed, allowing for post-analysis of driving sessions. This functionality is valuable for debugging and refining the autonomous driving algorithms.

These results demonstrate that the core functionality of the autonomous car simulation was successfully implemented, providing a viable platform for testing and refining autonomous driving systems

## Evaluation

The project's achievements can be evaluated against the initial objectives and in comparison with similar existing solutions.

The main objective of creating a functional autonomous car simulation with lane detection capabilities was achieved. The system successfully demonstrates basic driving and lane detection, providing a foundation for more advanced autonomous features.

Compared to commercial autonomous driving simulators such as CARLA or LGSVL, this project offers a more lightweight and customizable solution. While it lacks some of the advanced features of commercial simulators, it provides greater flexibility for implementing and testing custom algorithms.

Strengths:

- Custom environment creation allows for tailored testing scenarios

- Integration of ROS 2 enables compatibility with a wide range of robotics tools and libraries

- The lane detection system demonstrates good performance under controlled conditions

- The modular architecture facilitates the addition of new features and algorithms

Limitations:

- Compatibility issues between Ubuntu 22.04, ROS 2 Humble, and Gazebo Classic prevented the implementation of some advanced features

- The lane detection system's performance degrades under challenging lighting conditions and with unclear lane markings

- The autonomous driving capabilities are currently limited to basic movement and lane following

- The simulation environment, while functional, lacks the visual fidelity of commercial simulators

Technical Challenges:

The project faced several technical challenges during implementation:

- Integration of the Ackerman driving mechanism required significant modifications to the Prius car model

- Configuring the camera sensor to provide useful visual input for lane detection required careful adjustment of position and parameters

- Ensuring proper synchronization between the various ROS 2 nodes was challenging, particularly for real-time processing requirements

- The lane detection algorithms required extensive tuning to achieve acceptable performance across different scenarios

# Legal, Social, Ethical and Professional Issues

Since this is an academic research project in Gazebo, many of the typical ethical and legal considerations for autonomous vehicles do not directly apply. However, in this project:

Typical safety regulations, data privacy and protection laws, and also insurance frameworks typically associated with real-world self-driving car testing are not applicable in this simulated environment since there won’t be a physical vehicle but a simulated digital twin.

Also concerns about job displacement in transportation sectors such as uber or taxi drivers are not relevant to this academic research project.

While these issues are excluded from direct consideration in this project, it’s important to also note that if the software was to be released commercially, these issues will need to be addressed in order to get approvals and licenses so the autonomous vehicle can be deployed on public roads.

# Conclusion

The autonomous car simulation project has successfully demonstrated the feasibility of implementing basic driving capabilities using ROS 2 and Gazebo. Despite the previous work being lost due to a computer error, previously, some of the deep learning capabilities such as pedestrian and car recognition on life footage have been sent, and despite facing technical challenges and compatibility issues, the project achieved its core objectives of creating a functional simulation environment and implementing lane detection-based autonomous driving.

The project provides a valuable foundation for future work in autonomous driving simulation, offering a customizable platform for testing and refining autonomous driving algorithms. The modular architecture and integration with ROS 2 ensure that the system can be extended and enhanced to address more complex autonomous driving scenarios in the future.

While there is room for improvement in terms of features, performance, and feature completeness, the project represents a significant step toward developing accessible and customizable tools for autonomous driving research and development.**References**

1. Adarsh, S. P. & Swamy, K.M., 2024. Sensor Fusion Using Kalman Filter

in Autonomous Vehicles. International Research Journal of Engineering and Technology (IRJET), 11(3), pp. 750–754.

2. Donzia, S. & Kim, J., 2022. Performance of Deep Learning Models in Autonomous Vehicle Environments Using CARLA Simulator. Journal of Autonomous Systems, 15(2), pp. 120–136.

3. De-Las-Heras, R., et al., 2021. Machine Learning-Based ADAS System Implementations for Road Sign Detection and Transcription. Automotive AI Review, 9(4), pp. 233–250.

4. F¨urst, M., et al., 2024. Enhanced Pedestrian Detection for Au- tonomous Vehicles Through LiDAR and RGB Sensor Integration. IEEE Transactions on Intelligent Vehicles, 8(1), pp. 45–58.

5. Garikapati, S. & Shetiya, P., 2024. AI Evolution in Autonomous Vehicle Industry. Journal of Autonomous Technology Innovations, 10(1), pp. 33–41.

6. Pang, J., et al., 2021. Quasi-Dense Similarity Learning for Multiple Object Tracking. Proceedings of the IEEE/CVF Conference on Com-puter Vision and Pattern Recognition (CVPR), pp. 164–172. Available at: <https://openaccess.thecvf.com/content/CVPR2021/papers/Pang>

7. Reke, T., et al., 2024. Distributed System Architecture for Au-tonomous Driving: Implementation in ROS2. IEEE Robotics and Au-tomation Letters, 7(3), pp. 120–132.

8. Rui, W., et al., 2021. A Real-Time Object Detector for Au- tonomous Vehicles Based on YOLOv4. Journal of Autonomous Vehicle Systems, pp. 1-11. Available at: <https://www.researchgate.net/publication/356890617>

9. Song, K., et al., 2024. Synthetic Data Generation in Autonomous

Vehicle Model Training. Journal of Computer Vision and Robotics, 18(2),

10. Yurtsever, E., et al., 2020. Common Practices in AI and AV

(Autonomous Vehicles). IEEE Transactions on Intelligent Transportation Systems, 21(3), pp. 198–213.

11. MathWorks, 2024. DBSCAN Algorithm for Clustering. Available at: <https://www.mathworks.com/help/stats/dbscan.html>

12. Neptune AI, 2024. Self-Driving Cars with Convolutional Neural Networks (CNN). Available at: <https://neptune.ai/blog/self-driving-cars->with-convolutional-neural-networks-cnn

13. NVIDIA, 2024. Omniverse Replicator Synthetic Data Genera-tion. NVIDIA NGC Catalog. Available at: <https://catalog.ngc.nvidia.com/orgs/nvidia/containers/ov->synthetic-data-generation

14. Independent, 2023. Pedestrian Road Deaths Rose 6% Last Year.

The Independent. Available at: <https://www.independent.co.uk/news/uk/government-department-for-transport-dft-rac-malvern-b2553758.html>

15. Macenski, S., Soragna, A., Carroll, M. & Ge, Z., 2023. *Impact of ROS 2 Node Composition in Robotic Systems*. arXiv. Available at: <https://arxiv.org/abs/2305.09933> [Accessed 7 May 2025].

16. Macenski, S., Soragna, A., Carroll, M. & Ge, Z., 2023. *Impact of ROS 2 Node Composition in Robotic Systems* [PDF]. arXiv. Available at: <https://arxiv.org/pdf/2305.09933> [Accessed 7 May 2025].

17. Open Robotics, 2025. *ROS 2 Documentation (Humble)*. Available at: <https://docs.ros.org/en/humble/index.html> [Accessed 7 May 2025].

18. Zhang, L., Zhang, R., Li, L., Ding, C., Zhang, D. & Chen, L., 2022. *Research on virtual Ackerman steering model based navigation system for tracked vehicles*. Computers and Electronics in Agriculture, 198, 106323. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S0168169921006323> [Accessed 7 May 2025].

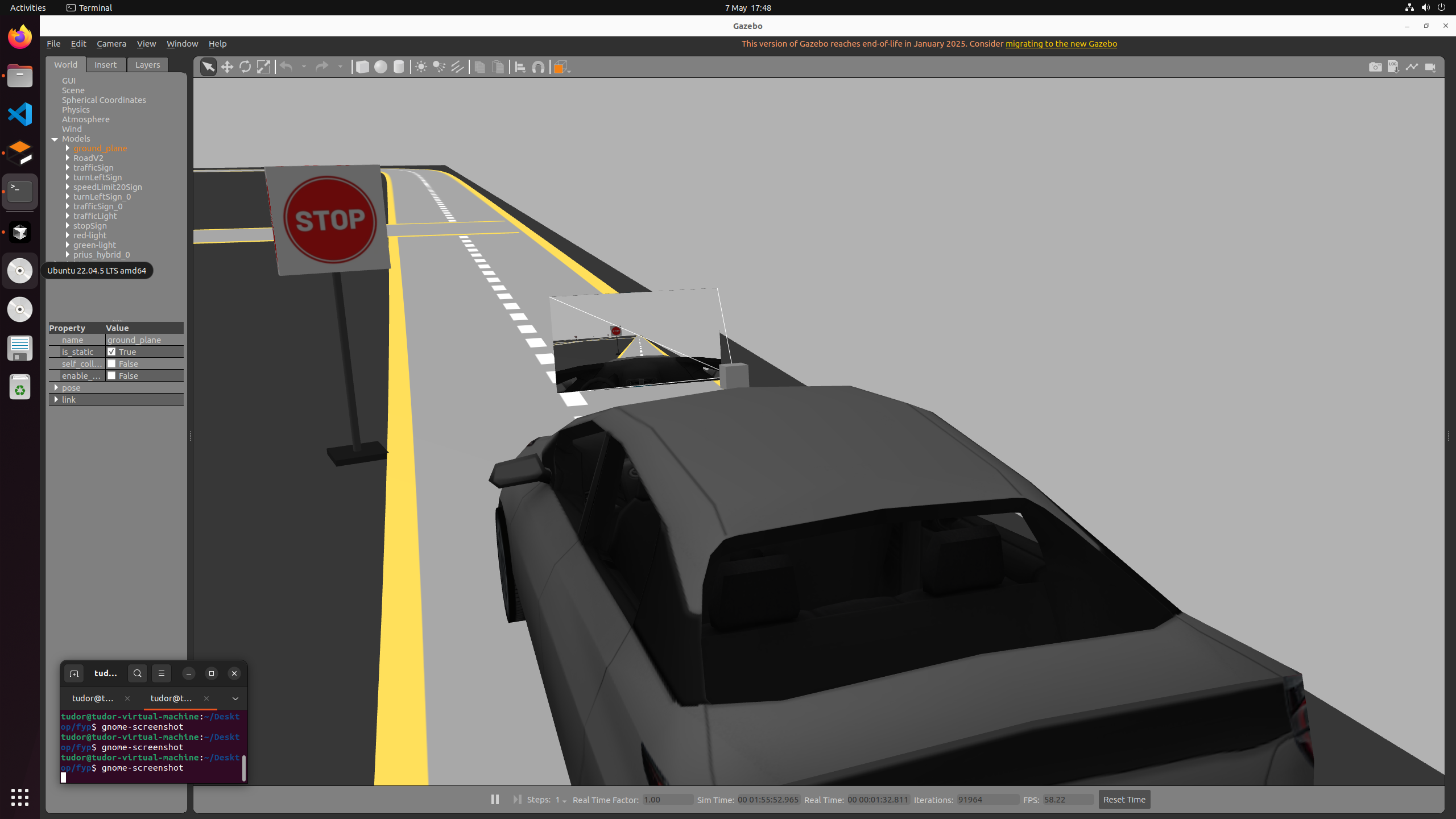
19. Gazebo, n.d. *Logical Camera Sensor Tutorial*. Available at: <https://classic.gazebosim.org/tutorials?tut=logical_camera_sensor> [Accessed 7 May 2025].

20. gazebo\_plugins, n.d. *Class GazeboRosAckermannDrive*. ROS 2 Documentation (Rolling). Available at: <https://docs.ros.org/en/rolling/p/gazebo_plugins/generated/classgazebo__plugins_1_1GazeboRosAckermannDrive.html> [Accessed 7 May 2025].

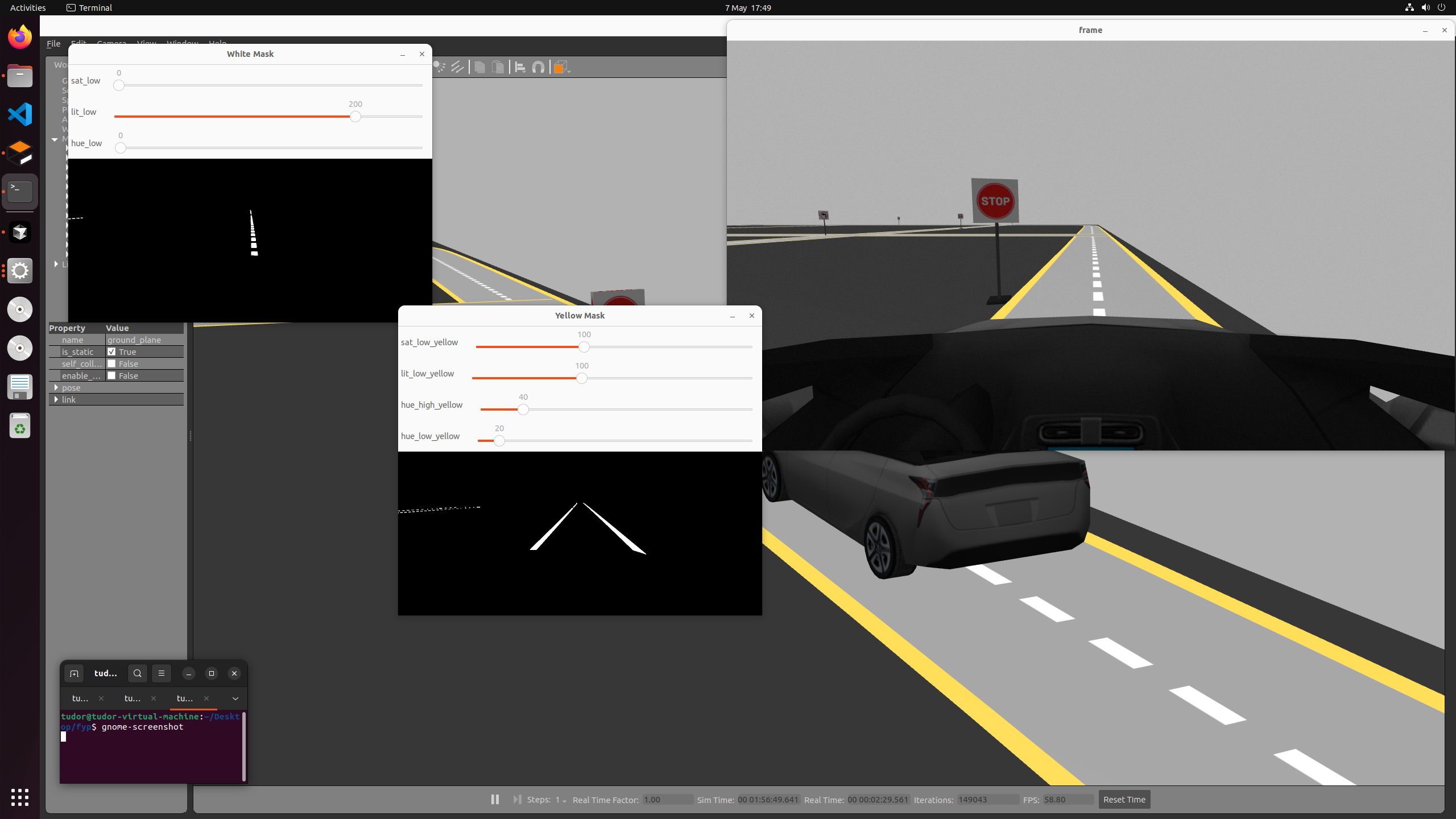
21. Gazebo, n.d. *AckermannSteering Class Reference*. Available at: <https://gazebosim.org/api/gazebo/6/classignition_1_1gazebo_1_1systems_1_1AckermannSteering.html> [Accessed 7 May 2025].

* **Appendices**

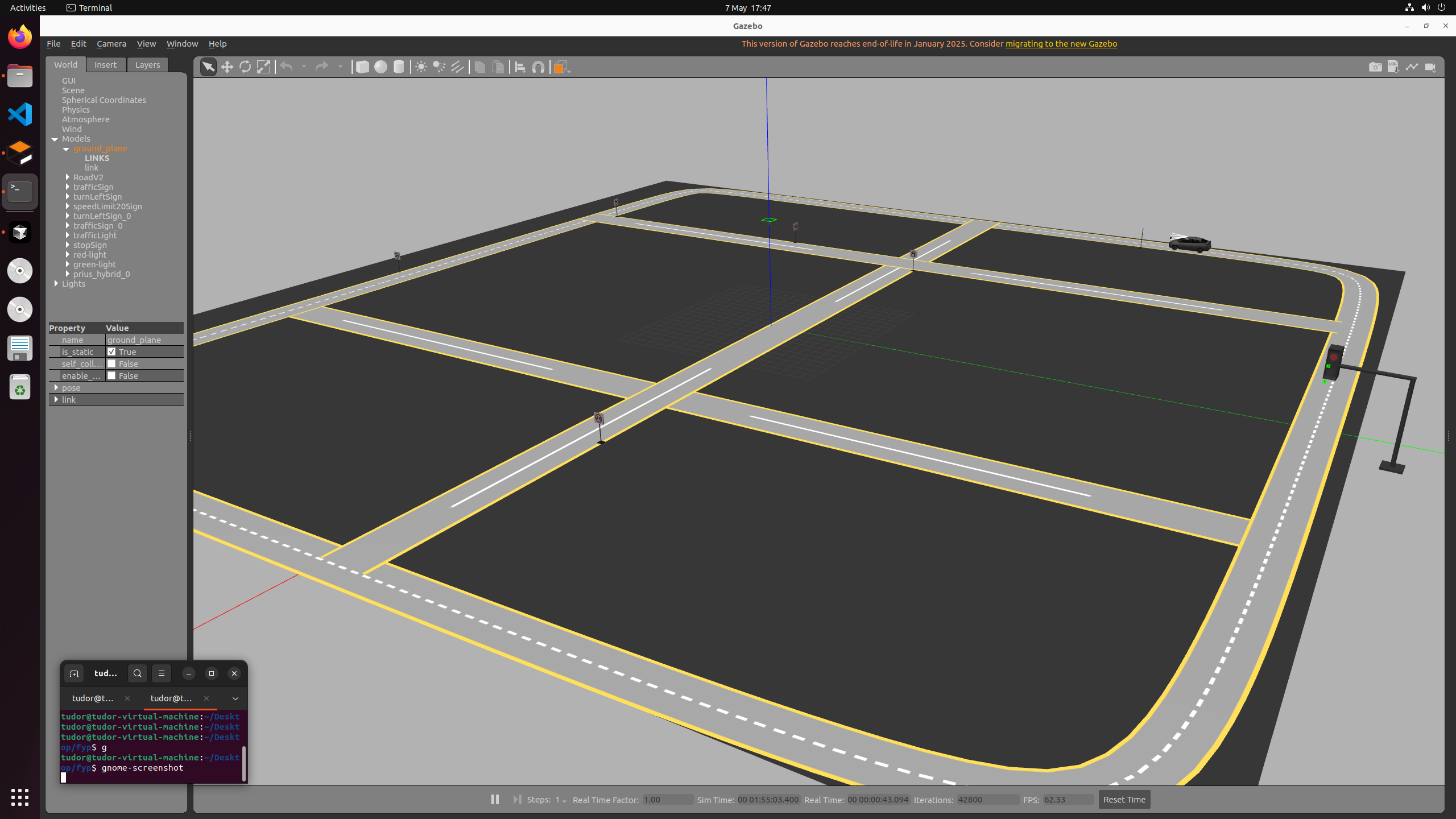
**APPENDIX A – Camera Plugin**



**APPENDIX B – Lane Segmentation**



**APPENDIX C – Simulation Environment**



**APPENDIX D – Traffic Light**

