Software Requirements Specification

Autonomous Driving Without Lane Marks

Humayra Rashid, Daniel Gomez, Henry Xiong, Rodrigo Valdez

May 10, 2025

Revision History

Name	Date	Reason for Changes
Humayra Rashid, Daniel Gomez, Henry Xiong, Rodrigo Valdez	May 10, 2025	Initial Draft

1 Introduction

1.1 Purpose

This document outlines the software requirements for an autonomous driving system capable of navigating without clear lane markings. It includes all functional and non-functional requirements needed for development.

1.2 Intended Audience and Reading Suggestions

This document is intended for software engineers, project managers, and testers. Section 4 is most relevant to developers and testers, while sections 1 and 2 provide context for stakeholders.

1.3 Product Scope

The project involves designing and implementing a software system that enables autonomous vehicles to drive on roads with unclear or absent lane markings. The system will process real-time sensor input to determine navigation paths.

1.4 Definitions, Acronyms, and Abbreviations

- ADA - Autonomous Driving Algorithm
- PM - Processing Module
- RM - Response Module
- SIM - Simulator

1.5 References

No academic papers were cited. Project built on top of publicly available open-source driving simulators.

2 Overall Description

2.1 System Analysis

The system addresses the challenge of autonomous driving without lane marks. Major hurdles include accurate environmental perception and safe decision-making. Solutions involve a modular software pipeline with pre-processing and response logic.

2.2 Product Perspective

This system is a standalone software solution that can be integrated into toy-scale or full-scale vehicles equipped with sensors. It does not depend on any existing proprietary systems.

2.3 Product Functions

- Interpret visual data to identify drivable paths.
- Process sensor input to detect obstacles.
- Compute steering and speed commands.

2.4 User Classes and Characteristics

- Developers - configure modules and debug performance.
- Testers - evaluate system behavior in simulated and real-world conditions.

2.5 Operating Environment

The software will run on Raspberry Pi or Android systems in the prototype and on a desktop computer for simulation.

2.6 Design and Implementation Constraints

Must be lightweight enough to run on embedded systems. Should work without GPS or high-resolution maps.

2.7 User Documentation

A developer's guide and simulation test plan will be included.

2.8 Assumptions and Dependencies

Assumes availability of a camera, a motor controller, and basic Linux-based system.

2.9 Apportioning of Requirements

Advanced lane merging and multi-car coordination will be postponed for later versions.

3 External Interface Requirements

3.1 User Interfaces

CLI and logs only; no GUI. Possible voice input in later versions.

3.2 Hardware Interfaces

Interacts with sensors (camera, ultrasonic), motors, and microcontrollers.

3.3 Software Interfaces

Built on open-source libraries like OpenCV. Will communicate with a control daemon on the prototype.

3.4 Communications Interfaces

Uses Bluetooth for prototype communication and file-based interfaces in simulator.

4 Requirements Specification

4.1 Functional Requirements

SM - Sensor Module

- SM.1 The system shall collect visual data from the front-facing camera.
- SM.2 The system shall detect obstacles and send data to the controller.

CM - Controller Module

- CM.1 The system shall receive sensor input and pass it to the app module.
- CM.2 The system shall forward control commands to the motor.

AM - Application Module

- AM.1 The system shall manage the processing and response pipeline.
- AM.2 The system shall communicate results back to the controller.

4.2 External Interface Requirements

Described in Section 3.

4.3 Logical Database Requirements

No persistent data storage required. All data processed in real time.

4.4 Design Constraints

Must run in under 200MB RAM. Execution must remain under 100ms per loop.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

95% of commands must be generated in under 100ms.

5.2 Safety Requirements

Prototype must stop movement on sensor failure.

5.3 Security Requirements

Bluetooth pairing must be encrypted.

5.4 Software Quality Attributes

- Modularity
- Real-time responsiveness
- Portability to embedded devices

5.5 Business Rules

Only trusted developers can change the driving policy logic.

6 Legal and Ethical Considerations

Prototype testing will be done only in safe, closed environments. Code will be open-source for educational use only.