Software Requirements Specification

for

Mathworks (Sensor Fusion for Autonomous Systems)

Version 1.0

Prepared by Hagop Arabian, Daniel Gallegos, Roberto Garcia, Gerardo Ibarra, David Neilson, Patrick Emmanuel Sangalang, Jonathan Santos, Deepanker Seth, Angel Tinajero, Xiao Hang Wang

Mathworks

May 12th, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First Draft	12-9-2022	Initial Doc	1.0.0
Final Draft	5-12-2023	Final Doc	2.0.0

1. Introduction

1.1 Purpose

The purpose of this document is to describe the requirements for the execution and deployment of the Y.O.M.O. algorithm. Including, but not limited to an overall description of the algorithm's functional parts, hardware requirements and other necessary requirements for the algorithm's implementation.

1.2 Intended Audience and Reading Suggestions

The intended audience of the SRS document are developers, faculty advisors, testers, and project managers. This document contains pertinent information about the project and where the team is in accomplishing our goal. As we make updates to our software and design, testing will also be a major key in perfecting our project.

1.3 Product Scope

The scope of the product is limited to testing environments, in controlled settings within the confines of existing prerecorded datasets. The availability of synchronized lidar and camera data are necessary for the algorithm model to perform predictions. The Y.O.M.O. algorithm is designed to function in the scenario of a vehicle attempting to perform a successful lane change and merge. The operator will then reference the output of the algorithm to assist in determining the probability of a safe lane change.

1.4 Definitions, Acronyms, and Abbreviations

Contained herein, are the necessary definitions, acronyms and abbreviations that will be used in this document for the purpose of clarity. The list below contains the definition, acronyms, and abbreviations used for the SRS.

- **LIDAR:** Short for light detection and ranging, LIDAR is a type of sensor that is used to measure distances using laser pulses for the purpose of ranging objects. It works by sending laser pulses in a given direction and measuring the time it takes for the reflection of said laser to return. It is a method for determining ranges by targeting an object or surface with lasers.

- **Camera:** an image-based sensor that provides the driver with important visual information about the vehicle surroundings.

- **Y.O.M.O.** : short for "You Only Merge Once," it is the name of our algorithm solution to the problem of determining lane changing .

1.5 References

Python	Documentation: **https://www.python.org/doc/versions/**
OpenCV	Documentation: OpenCV
Sklearn	Documentation: Sklearn
KITT Vision Benchmark Suite	Setup: The KITTI Vision Benchmark Suite (cvlibs.net)
Flask	Documentation: Welcome to Flask — Flask Documentation (2.3.x) (palletsprojects.com)
React	Documentation: Built-in React Hooks – React

2. Overall Description

2.1 System Analysis

The overarching problem is that of the utilization of existing sensors to assist in the safe execution of lane changing. With the major goal of using Machine Learning algorithms in conjunction with multiple sensors to make a determination of safe and unsafe vehicle lane merging. By combining multiple sensors, also known as sensor fusion, under the umbrella of a Machine Learning algorithm, a clear picture can be extrapolated from that data collected.

Major technical hurdles include the need for large amounts of labeled data on which to train the Machine Learning Algorithm. Since the majority of data collected is that of collision-less driving, identifying scenarios that would lead to unfavorable results, are therefore limited in availability. Another issue is the vast amounts of data that is generated by the LiDAR sensor. Which can result in excessive amounts of unnecessary data.

One such solution for the issue is the topic of our solution. Some of the data sets were hand labeled to give the algorithm data on which to train. Another approach to work with the lidar data was to determine a general safe distance from all objects detected by the LiDAR. Since the focus of this project is lane changes, only the immediate distance is needed for a safe lane change.

2.2 Product Perspective

The product is a web application where the vehicle has a bird eye view of surrounding vehicles and objects to properly perform a successful lane change. This product highly relies on given sensor fusion data and with the help of OpenCV and Sklearn.

2.3 Product Functions

- The application will display a web page.
- The application will show a bird eye 2D LiDAR minimap of the vehicle.
- The application will collect various datasets to improve accuracy.

2.4 User Classes and Characteristics

A. Developer: The user will be able to modify and use features accessible in the application.

2.5 Operating Environment

This application will connect to a local backend server.

2.6 Design and Implementation Constraints

- Physical limitations of the sensors
- CPU computing performance
- Graphics performance
- Quality of the collected data

2.7 User Documentation

TBA

2.8 Assumptions and Dependencies

- Developers have the KITTI dataset
- Developers have the OpenCV Python library
- Developers have the Sklearn Python library

2.9 Apportioning of Requirements

The requirements were divided into three parts:

- Back-End
- Front-End
- Algorithm

3. External Interface Requirements

3.1 User Interfaces

The graphical interface will maintain the principle of minimalist design. Minimalism not only allows users to find the required information in a timely manner, but also reduces the cost of learning the software; in addition, the minimalist design will also result in lower hardware resource requirements, allowing more hardware resources to be allocated to the main machine learning algorithms and algorithm fusion, as well as increasing the likelihood that the project will be ported to more computing resource-constrained platforms. On this graphical interface, the right side is the bird eye view video derived from the algorithm, and the left side is the real-time camera video. Under the bird eye view video, if the machine learning algorithm gives a dangerous result, a warning will appear prominently on the left and a specific result on the right.

3.2 Hardware Interfaces

The data collection, aggregation, and pre-processing part of this project is done by the KITTI Vision Benchmark Suite, and the machine learning models training part will run on an x86 computer and an ARM MacBook.

- The KITTI Vision Benchmark Suite Setup
  - 1 Inertial Navigation System (GPS/IMU): OXTS RT 3003
  - 1 Laserscanner: Velodyne HDL-64E
  - 2 Grayscale cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M-C)
  - 2 Color cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
  - 4 Varifocal lenses, 4-8 mm: Edmund Optics NT59-917
- Aorus 15G
  - CPU: I7-10870h
  - GPU: RTX 3060 Laptop GPU
  - RAM: 16 GB DDR4
- Macbook Air
  - CPU: Apple M2 chip
  - RAM: 16GB unified memory

3.3 Software Interfaces

- Python 3.10
- Sklearn 1.1.3
- Pandas 1.5.2
- Numpy 1.23.5
- Jupyter notebook 5.2
- OpenCV 4.7.0-dev
- KITTI Vision Benchmark Suite 1.0
- Flask 2.3.2
- React.js 18.2.0

3.4 Communications Interfaces

The KITTI dataset was collected from a recording platform, it utilizes a Volkswagen Passat B6 that has been altered with actuators for the pedals (acceleration and brake) and the steering wheel. To record data, an eight-core i7 computer with a RAID system running Ubuntu Linux and a real-time database is used. The following sensors are employed. The video frames were passed to the back-end in the format of PNG, and the Lidar data were passed to the back-end in velodyne point cloud in the format of TXT.

4. Requirements Specification

4.1 Functional Requirements

- The application shall read sensor data from the KITTI dataset.
- The application will perform Machine Learning Algorithm.
- The application will have one model in training.

4.2 External Interfaces Requirements

The interface shall display the result of the algorithm and a real-time bird eye view in a web page.

4.3 Logical Database Requirements

The application does not require a logical database.

4.4 Design Constraints

- **Standard Limitation**
  - ○ Limited experience with Computer Vision Algorithms
  - ○ Limited experience with Machine Learning
- **Hardware Limitation**
  - ○ The Application must run on a stronger than decent computer.
  - ○ The accuracy of the sensor may not be sufficient to meet the algorithm requirements.
  - ○ Application may not run as fast as it should to get results in real time.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Requires synchronized LiDAR and Camera sensor data.

5.2 Safety Requirements

No safety requirements were found in this application.

5.3 Security Requirements

No security requirements were found.

5.4 Software Quality Attributes

- Adaptability: For desktop platforms, mobile platforms, and In-vehicle systems.
- Availability: Access through web server and web page.
- Reliability: Sensor data comes from the KITTI dataset.

5.5 Business Rules

All features are accessible to developers.

6. Legal and Ethical Considerations

No legal and ethical issues were identified in these applications.

Appendix A: Glossary

LiDAR	Light Detection and Ranging. Measures distances using lasers for the purpose of ranging objects.
Y.O.M.O	You Only Merge Once. Name of our algorithm that solves our problem regarding lane change safety.
Machine Learning	Computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyze and draw inferences from patterns in data.
Camera Sensors	Image-based sensors that provide the driver with important visual information.