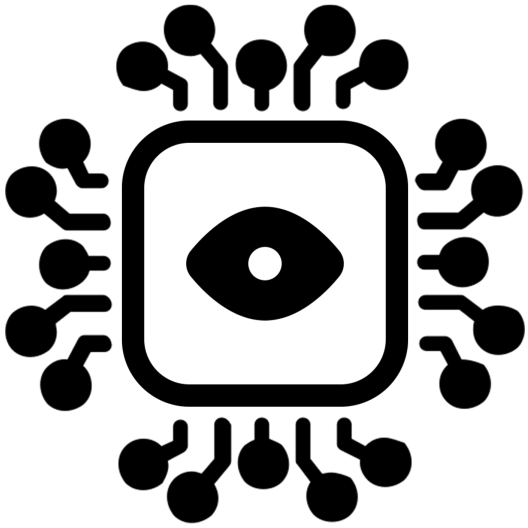
Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2024



IGVC Computer Vision Team
IGVC Computer Vision Module

Sameer Dayani
James Leo Caetano Jr.
Abu Talha Nayyar
Brandon Joel Bowles
William Periman

Revision History

Revision	Date	Author(s)	Description
0.1	2.08.2024	ATN	document creation
0.2	2.12.2024	SD, JLC, ATN, BJB, WP	complete draft

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

The Product is a Modular component built for the IGVC participating vehicle and serves the purpose of acting as its Computer vision solution. This solution should be capable of identifying obstacles, and road constraints of the IGVC competition course using onboard sensors (Lidar, Cameras) and be able to communicate this information to the path-planning aspect as well as the vehicle competing in the IGVC competition in 2024.

2 SYSTEM OVERVIEW

The Sensors layer will collect data based on their surroundings and will be sent to different modules for calculations and detection. The camera will process the surroundings and save images to be sent to the Computer Vision module. The lidar sensor collects distance measurements and generates a precise 3D map of the surrounding environment. Another component is the Gyroscope, which is a part of the Inertial Measurement Unit (IMU) and measures acceleration and angular velocity to provide information about the vehicle's motion and orientation.

The Computer Vision layer will recognize obstacles and detect lanes based on the images provided by the camera sensor to help navigate through the course. For obstacle detection, the image processed will be identified through the predefined model actively recognizing the object and outputted as a grid-based representation due to computational resources. Similarly, lane detection will operate the same as obstacle detection including the output. The Output layer takes inputs from other subsystems and provides outputs to the path planner team and the sensor mounts managing the stability of the sensors. The onboard computer manages the stream of changing parameters of the gyroscope to constantly keep the sensors in a stable state to ensure an ideal input recording environment. A dot matrix map is to be communicated to the Path Planner that is generated from the Computer Vision layer in the off-board computer and is to be packaged with measured distance information to be sent to the path planner at a rate of 10 times a second.

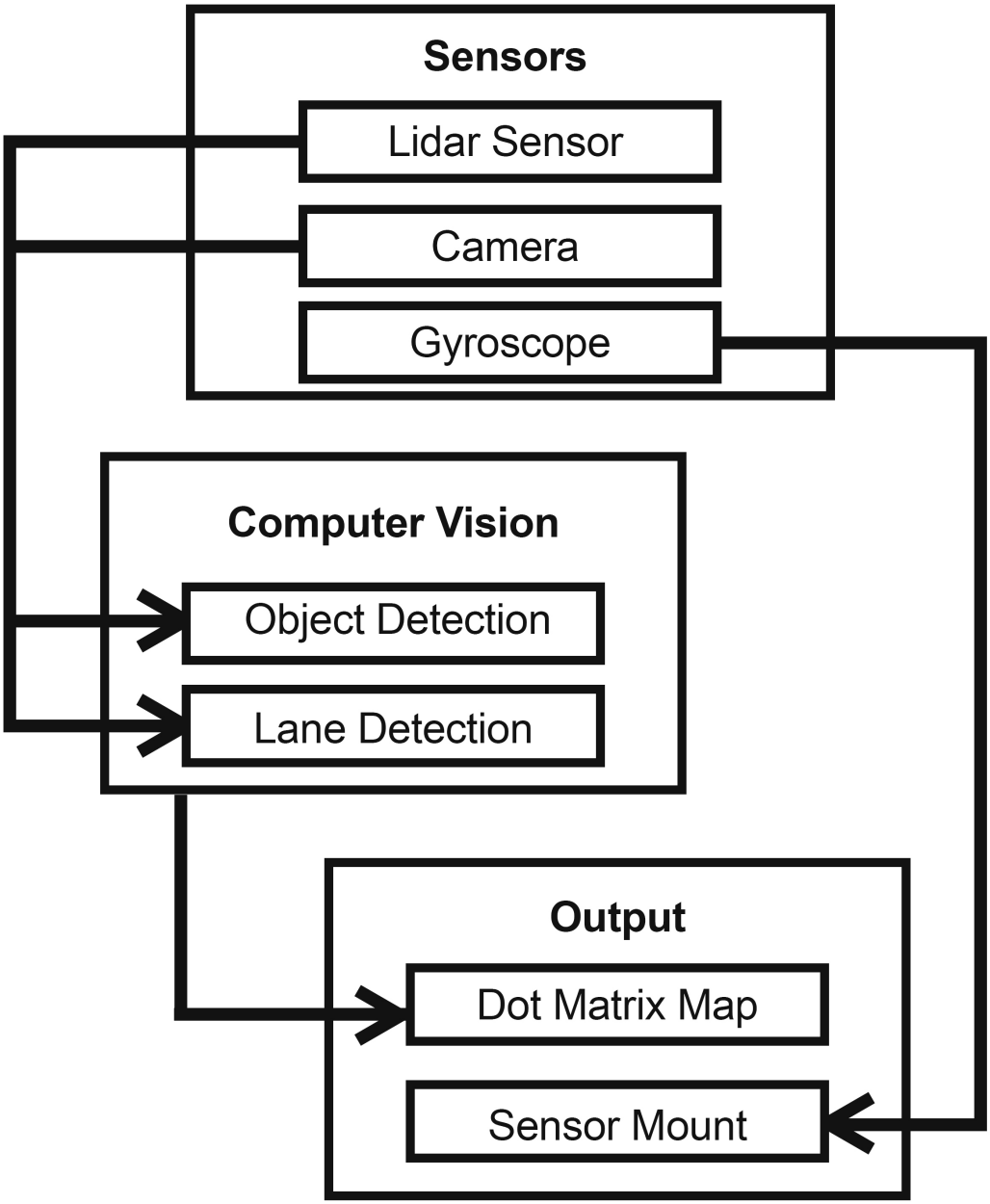


Figure 1: System architecture

3 Sensors Layer Subsystems

In this section, the layer is described in terms of the hardware and software design. Specific implementation details, such as hardware components, programming languages, software dependencies, op-

erating systems, etc. should be discussed. Any unnecessary items can be ommitted (for example, a pure software module without any specific hardware should not include a hardware subsection). The organization, titles, and content of the sections below can be modified as necessary for the project.

3.1 Layer Hardware

Connected to the Raspberry Pi 5 via USB port.

3.2 Layer Operating System

The sensor layer will be running on the Raspberry Pi Operating System. The Raspberry Pi OS is a Unix-like operating system based on the Debian GNU/Linux distribution for the Raspberry Pi family.

3.3 Layer Software Dependencies

- Python - programming language
- OpenCV - open source python library that allows for specialized methods specific to computer vision
- NumPy - Python library for scientific computing using multidimensional arrays and matrices for more optimized image processing
- Matplotlib - a comprehensive library for creating static, animated, and interactive visualizations in Python

3.4 Subsystem Lidar Sensor

The lidar sensor subsystem is a component that will be enclosed in the custom housing that will be mounted on the ground vehicle. The lidar sensor will emit pulsed light waves in order to generate an accurate, real-time 3D map of the environment, called a point cloud, that will help the ground vehicle navigate obstacles.

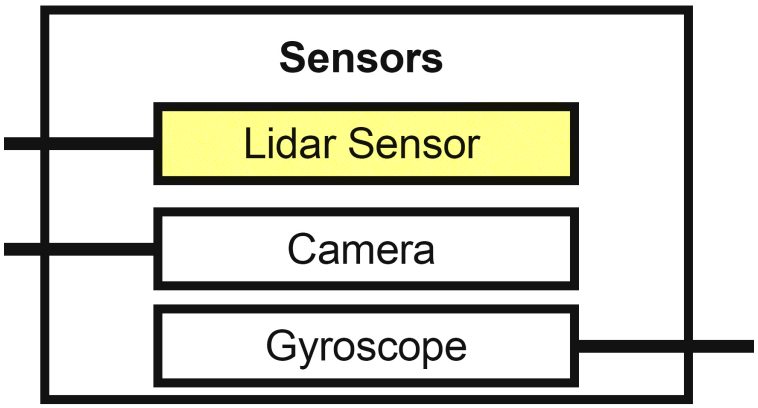


Figure 2: Lidar sensor subsystem description diagram

3.4.1 Subsystem Hardware

Connected to the Raspberry Pi 5 via USB port.

3.4.2 Subsystem Operating System

The lidar subsystem will be running on the Raspberry Pi Operating System. The Raspberry Pi OS is a Unix-like operating system based on the Debian GNU/Linux distribution for the Raspberry Pi family.

3.4.3 Subsystem Software Dependencies

- OpenCV - open source python library that allows for specialized methods specific to computer vision
- NumPy - Python library for scientific computing using multidimensional arrays and matrices for more optimized image processing
- Matplotlib - a comprehensive library for creating static, animated, and interactive visualizations in Python

3.4.4 Subsystem Programming Languages

Python, C++

3.4.5 Subsystem Data Structures

A description of any classes or other data structures that are worth discussing for the subsystem. For example, data being transmitted from a microcontroller to a PC via USB should be first be assembled into packets. What is the structure of the packets?

3.4.6 Subsystem Data Processing

YOLO v5 (You Only Look Once) algorithm - a state-of-the-art, real-time object detection system

3.5 Subsystem Camera

Camera hardware that captures visual information and feeds images for object recognition and lane detection.

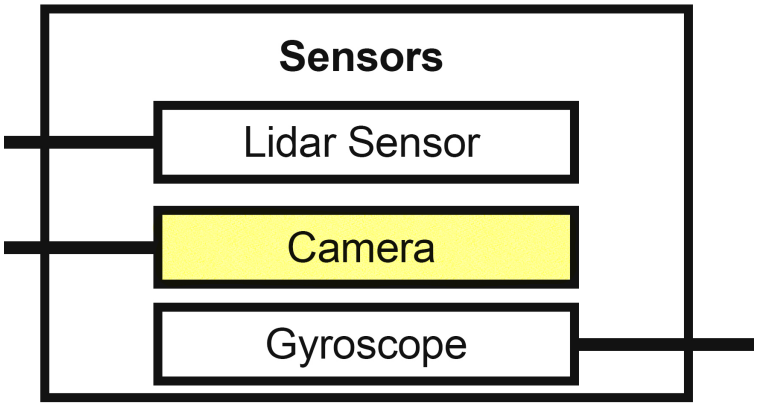


Figure 3: Camera subsystem description diagram

3.5.1 Subsystem Operating System

Camera calibration runs on ROS 2 where packages and tools utilize camera calibration and use images as input.

3.5.2 Subsystem Programming Languages

Written in Python or C++ (depending on the performance of the language)

3.5.3 Subsystem Data Structures

Images are sent from the camera to Raspberry Pi 5 via a USB port. Data structu

3.6 Gyroscope

Apart from the lidar sensor and cameras, one crucial device for this project would be a gyroscope, which would be integrated into an Inertial Measurement Unit (IMU). Itâs main function is to generate information about the vehicleâs angular velocity and the real time orientation. Without this part, it would not be possible for the other sensors to generate a correct data.

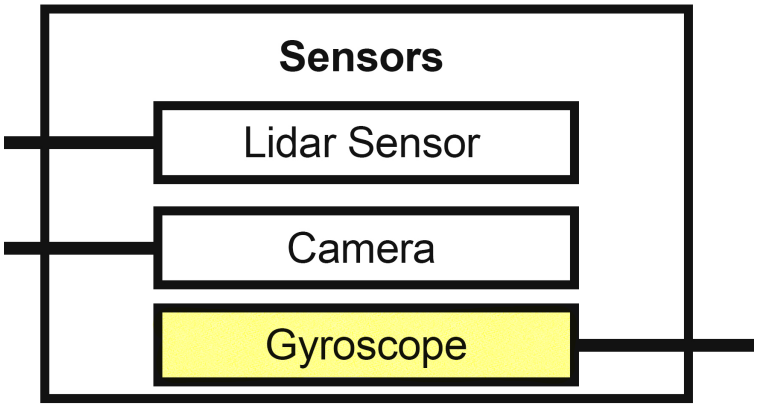


Figure 4: Gyroscope subsystem description diagram

3.6.1 Subsystem Hardware

This Gyroscope would be consisted of several sub-parts that will include the Sensor, Processor which will be installed on the vehicle by default, Power Supply (Voltage may vary), Mounting Structure to ensure accuracy in measurements. This gyroscope will also make use of various software components including Image Processing and Filtering Algorithms and the integration of gyroscope with a proper navigation system.

3.6.2 Subsystem Operating System

As for now, this entire model will be making use of RTOS(Real Time Operating System).Custom firmware's would also be used as part of integration between the main device and gyroscope. While integrating the firmware, certain features would be tested for the compatibility including software libraries. Meeting software requirements and familiarity with the OS concept is crucial for every team member.

3.6.3 Subsystem Software Dependencies

According to the overall software architecture, some of the software dependencies include:- 1. Gyroscope Library:- To allow the communication between microcontroller and sensor to perform read and write functions. 2. Firmware:- Complete firmware for the gyroscope to perform certain calibration and certain operations. 3. Image Processing and Filtering Algorithms:- Part of the computer vision requiring several algorithms to analyze the image of the objects encountered on the way. 4. Calibration Software:- Required for the accurate readings.

3.6.4 Subsystem Programming Languages

Linux, Python.

3.6.5 Subsystem Data Processing

1. Low-pass Filtering to reduce noise reductions. 2. Vibration Rejection 3. Adaptive Filtering 4. Continuous Monitoring and Error Handling

4 Computer Vision Layer Subsystems

The computer vision module will take the surrounding images to identify what the obstacles are and keep track of the lane. The camera will communicate to the computer vision module and give them an image as input and the module will output a grid-based map.

4.1 Layer Hardware

The computer vision module possesses a USB camera attached to a Raspberry Pi 5.

4.2 Layer Operating System

The computer vision module will run off of ROS 2.

4.3 Layer Software Dependencies

- OpenCV - open source python library that allows for specialized methods specific to computer vision
- NumPy - Python library for scientific computing using multidimensional arrays and matrices for more optimized image processingThe computer vision module will use a YOLOV8 program.

4.4 Subsystem Object Detection

The computer vision module will use the camera to view the area in front of the ground vehicle and feed this information into a Raspberry Pi 5. The computer will be running ROS 2, taking the data points in and classifying them.

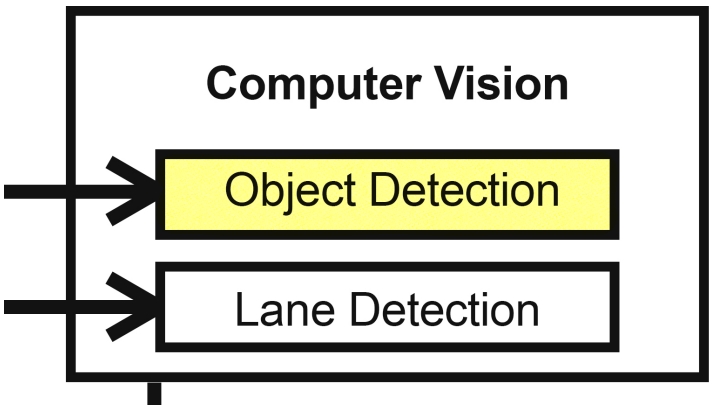


Figure 5: Object detection subsystem description diagram

4.4.1 Subsystem Programming Languages

The object detection subsystem will be programmed in Python or C++ (depending on performance).

4.4.2 Subsystem Data Processing

Utilizes YOLOv8 (You only look once), an object detection and recognition algorithm in real-time. The accuracy and speed of real-time object detection is fast in the environment of IGVC.

4.5 Subsystem Lane Detection

The lane detection subsystem will use data gathered through the camera and processed via ROS 2 and YOLOv8 to determine where the lanes the ground vehicle must stay between are, and feed this information to the path finding system.

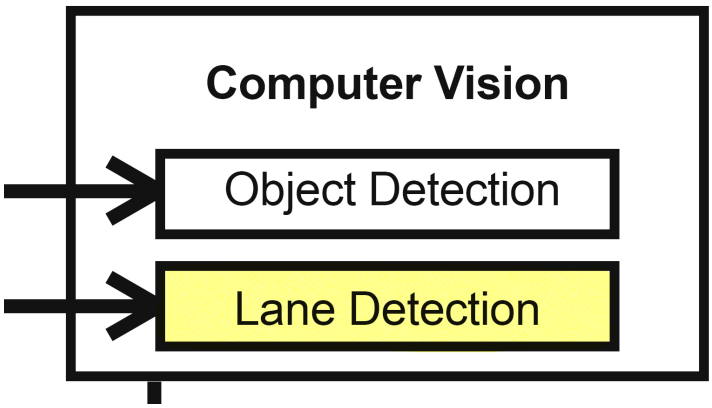


Figure 6: Lane detection subsystem description diagram

4.5.1 Subsystem Programming Languages

The lane detection subsystem will be programmed in Python or C++ (depending on performance) on ROS 2.

4.5.2 Subsystem Data Processing

The image received will be processed/transformed through different methods such as canny edge detection, which uses convolutional nerual networks (CNNs), or other methods that will improve land detection.

5 Output Layer Subsystems

The Output Subsystem controls the output that is sent to the Path Planner team as well as the stabilization instructions that are to be sent to the sensor mounts determined by processing the inputs from the Gyroscope sensor in order to stabilize the sensors.

5.1 Layer Hardware

No specific Hardware is involved in the output layer. other than perhaps a wifi module.

5.2 Layer Operating System

Packets of data being sent require an operating system that supports the networking and wireless data transfer protocols.

5.3 Layer Software Dependencies

JSON environment as well as wireless data transfer protocols.

5.4 Subsystem Dot Matrix Map

The output that is to be provided to the path planner team is going to be a dot matrix map of what has been detected by the computer vision model with marked detected obstacles, to be sent 10 times every second. A supplementary JSON file including measured distances from the obstacles might also be included in each sent package.

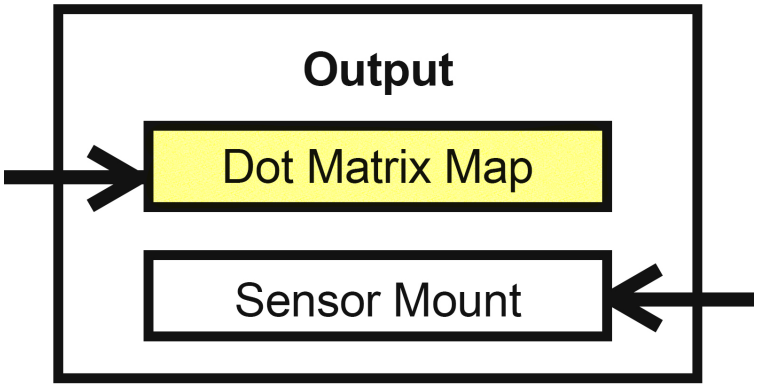


Figure 7: Dot matrix map subsystem description diagram

5.4.1 Subsystem Hardware

No hardware components are necessary for this subsystem.

5.4.2 Subsystem Operating System

No specific operating system is needed for this subsystem.

5.4.3 Subsystem Software Dependencies

Matplotlib, (more to be added)

5.4.4 Subsystem Programming Languages

Python and Java are needed for this subsystem.

5.4.5 Subsystem Data Structures

Not yet available.

5.4.6 Subsystem Data Processing

Not yet available.

5.5 Subsystem Sensor Mount

This is the subsystem that calculates the parameter adjustments necessary for the sensor mounts to be able to stabilize using inputs from the gyroscopes on the onboard computer.

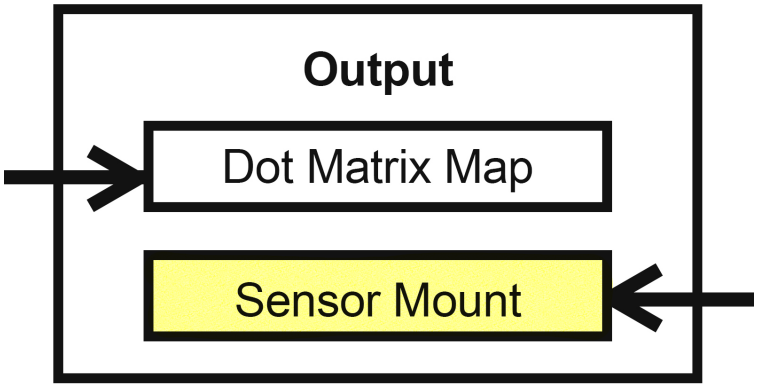


Figure 8: Sensor mount subsystem description diagram

5.5.1 Subsystem Hardware

Steppers and stepper controllers connected to a raspberry pi 5 automating gyroscopic features according to input provided by a gyroscopic sensor onboard the module.

5.5.2 Subsystem Operating System

ROS2, Raspian.

5.5.3 Subsystem Software Dependencies

Stepper Motor libraries, as well as Gryoscope libraries, more specifics will be added according to ordered parts.

5.5.4 Subsystem Programming Languages

C, C++ and Python.

5.5.5 Subsystem Data Structures

Not yet available.

5.5.6 Subsystem Data Processing

Not yet available.

6 APPENDIX A

Additional Documents will be added as they are used.

REFERENCES