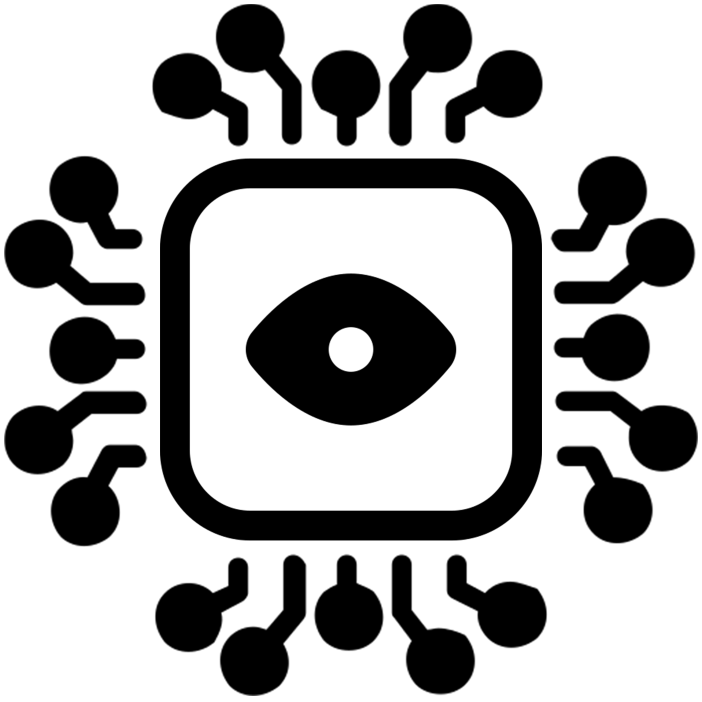
# Department of Computer Science & Engineering
# The University of Texas at Arlington

## 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 |
| 0.3 | 5.07.2024 | SD, JLC, ATN, BJB, WP | Updated document |
| 0.4 | 5.08.2024 | SD, JLC, ATN, BJB, WP | Finalized document |

# CONTENTS


**1 Introduction** **6**

**2 System Overview** **6**

**3 Sensors Layer Subsystems** **7**
- 3.1 Layer Hardware . . . 7
- 3.2 Layer Operating System . . . 7
- 3.3 Layer Software Dependencies . . . 7
- 3.4 Subsystem Camera . . . 8

**4 Computer Vision Layer Subsystems** **9**
- 4.1 Layer Hardware . . . 9
- 4.2 Layer Operating System . . . 9
- 4.3 Layer Software Dependencies . . . 9
- 4.4 Subsystem Object Detection . . . 9
- 4.5 Subsystem Lane Detection . . . 10

**5 Output Layer Subsystems** **11**
- 5.1 Layer Hardware . . . 11
- 5.2 Layer Operating System . . . 11
- 5.3 Layer Software Dependencies . . . 11
- 5.4 Subsystem Dot Matrix Map . . . 11

**6 Appendix A** **13**

## List of Figures

1 System architecture . . . . . 7
2 Camera subsystem description diagram . . . . . 8
3 Object detection subsystem description diagram . . . . . 9
4 Lane detection subsystem description diagram . . . . . 10
5 Dot matrix map subsystem description diagram . . . . . 11

## List of Tables

# 1 Introduction

The product is a modular component built for the Intelligent Ground Vehicle Competition (IGVC) vehicle and serves the purpose of acting as its computer vision solution. This solution should be capable of identifying obstacles, road constraints of the IGVC course using onboard sensors (cameras), communicating this information to the path-planning aspect, as well as the vehicle competing in the IGVC in 2024.

# 2 System Overview

The first layer will consist of the camera, which will collect data based on the 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 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. 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.

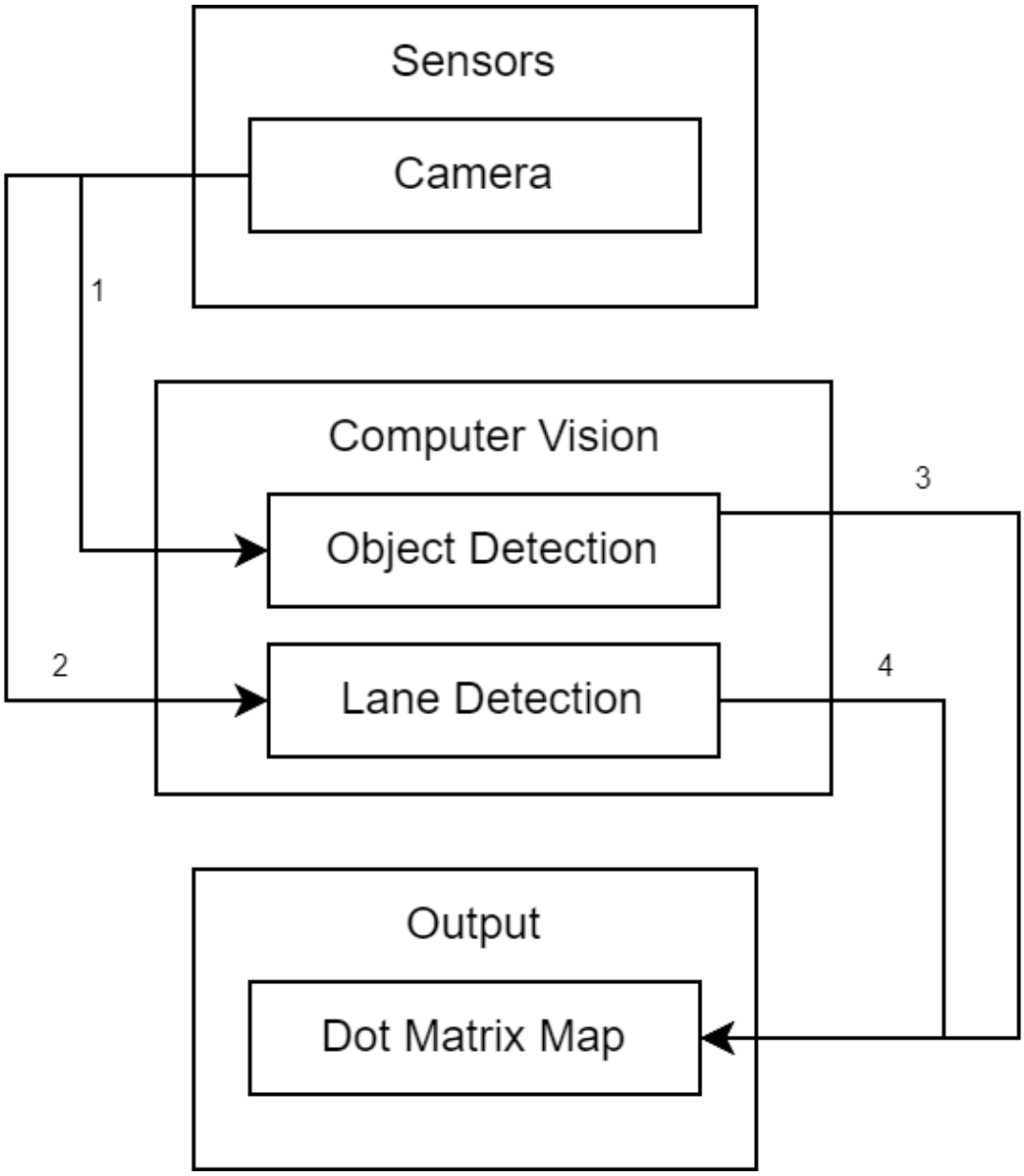



Figure 1: System architecture

# 3 Sensors Layer Subsystems

## 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 Camera

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

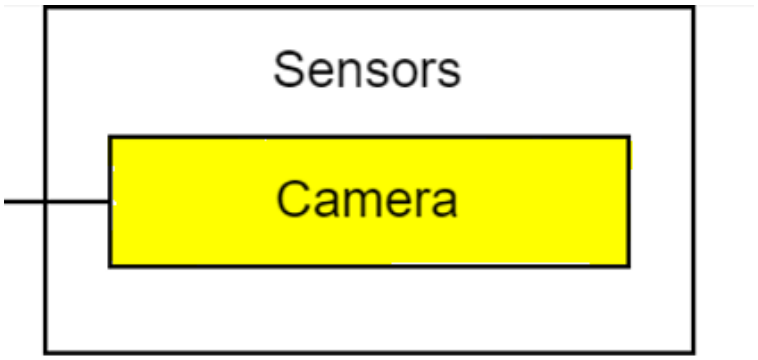



Figure 2: Camera subsystem description diagram

### 3.4.1 Subsystem Operating System

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

### 3.4.2 Subsystem Programming Languages

Written in Python using OpenCV and NumPy.

### 3.4.3 Subsystem Data Structures

Images are sent from the camera to Raspberry Pi or computer via a USB port.

# 4 Computer Vision Layer Subsystems

The computer vision module will take the real-time data and/or pre-recorded video's 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

A wireless camera/wired camera attached to a device running IDE such as Visual Studio code.

## 4.2 Layer Operating System

Linux OS

## 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 processing The computer vision module will use a YOLOV5 program.
- Pytorch - Python-based, fully featured framework for building deep learning models, which is a type of machine learning.
- YOLOV5 - Multi-scale feature map-based object detection machine learning algorithm.

## 4.4 Subsystem Object Detection

The computer vision module uses the camera to view the area in front of the ground vehicle and feed this information ideally being sent from the onboard computer to the off-board computer to perform object detection of the obstacles placed on the IGVC track. Currently it runs on the off-board computer taking in camera feed and detecting generic items that could be laid on the track as obstacles and traffic cones.

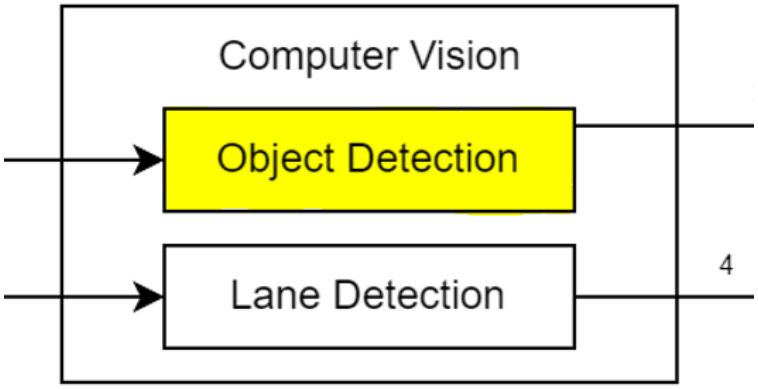



Figure 3: Object detection subsystem description diagram

### 4.4.1 Subsystem Programming Languages

The object detection subsystem will be programmed in Python.

### 4.4.2 Subsystem Data Processing

Utilizes YOLOv5 (You only look once), an object detection and recognition algorithm in real-time. The accuracy and speed of real-time object detection are fast in the environment of IGVC with high confidence.

## 4.5 Subsystem Lane Detection

The lane detection subsystem will use data gathered from the input captured by the camera on the vehicle. Using OpenCV and NumPy, various computer vision techniques such as Canny edge detection, Gaussian blur, erosion, dilation, and Hough transforms will be implemented. Using these methods, the input of the camera will be processed to identify and map the edges of the lane lines on the surface being traversed by the vehicle to help it stay within the lines and measure where the lines are located. This information will then be fed to the path-finding system.

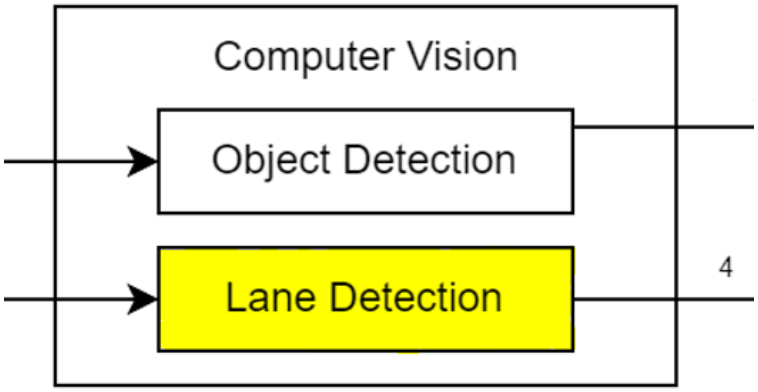



Figure 4: Lane detection subsystem description diagram

### 4.5.1 Subsystem Programming Languages

The lane detection subsystem will be programmed in Python using OpenCV and NumPy.

### 4.5.2 Subsystem Data Processing

The video images received will be processed by first converting the images to gray-scale, applying a Gaussian blur filter to smooth the edges of the image, Canny edge detection will then be used to identify edges within a predefined region of interest, and Hough transforms will be used to identify the position of the lane lines.

# 5 Output Layer Subsystems

The Output Subsystem controls the output that is sent to the Path Planner team.

## 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

Wireless data transfer protocols.

## 5.4 Subsystem Dot Matrix Map

The output will be formatted in a matrix of tuples, consisting of X/Y coordinates, distance, and a boolean if a line is detected. On each pass, the elements that contain the true boolean will provide their tuples to the output.

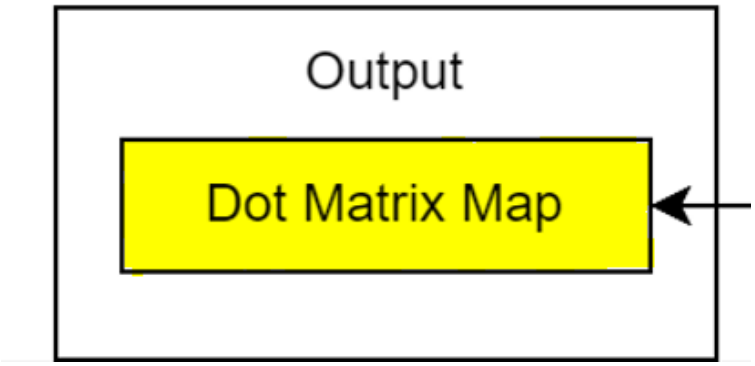



Figure 5: Dot matrix map subsystem description diagram

### 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

No specific software is needed outside of the OS.

### 5.4.4 Subsystem Programming Languages

Python, OpenCV, and NumPy are needed for this subsystem.

### 5.4.5 Subsystem Data Structures

A matrix consisting of tuples. x, y, d, Boolean X will be number of inches from reference point horizontally, Y will be number of inches from reference point vertically, and d is the distance.

### 5.4.6 Subsystem Data Processing

On each frame the Boolean value of the corresponding matrix elements will be updated to true. Only the true values are collected and prepared to be sent on as snapshot of the current detected line locations.

# 6 APPENDIX A

Additional Documents will be added as they are used.

## REFERENCES