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


## Detailed Design Specification
## CSE 4317: Senior Design II
## Spring 2023


## Team Mercury
## ARGOOSE - Drone Detection System


James Grumbles
Sanyogita Piya
Nirdesh Sakh
Mahin Roddur
Augustine Nguyen

## Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 2.06.2023 | AN | document creation |
| 0.2 | 2.10.2023 | AN, JG, SP, NS, MR | complete draft |

# CONTENTS

## List of Figures



## List of Tables

# 1 Introduction

ARGOOSE is a system that performs drone detection via sensors. Users of ARGOOSE will be able to set up a perimeter detection zone and receive information on detected drones within the area of operation. The ARGOOSE system detects drones and sends back information on detected drone to users. System is designed to perform in an open space with 360 camera detection. The system is designed to address general drone safety concerns. By providing feedback on drones located in the area, customers can be informed of drone behavior around sensitive locations (Page 5 of 22 in ADS). The system is designed to address general drone safety concerns. By providing feedback on drones located in the area, customers can be informed of drone behavior around sensitive locations. General customers are defined as anyone in need of this system's capabilities (Page 7 of 18 in SRS).

# 2 System Overview

A live video feed is captured by an external camera hooked up to the system, acting as the system's Computer Vision. The captured frame is then dewarped and processed for compatibility. Once the image is compatible, it is processed by an object-tracking library called YOLOv7; YOLOv7 will then draw on frame lines with labels onto identified objects. The image with frame lines will then be sent to the application and then the results will be displayed.

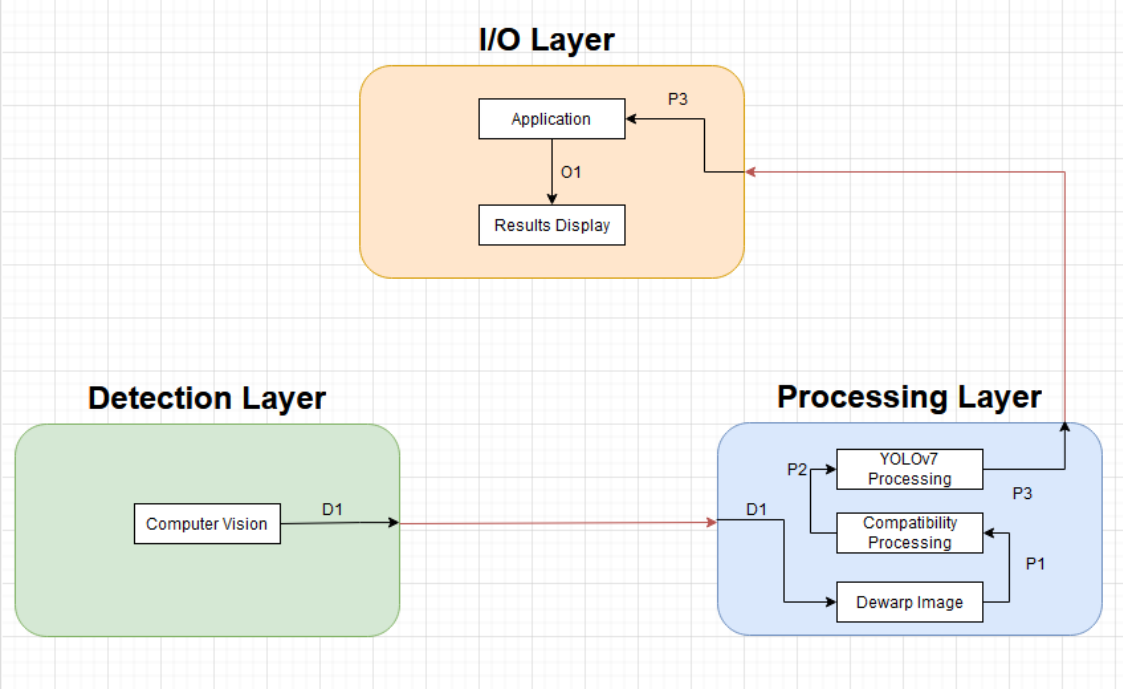


Figure 1: System Architecture

# 3 Detection Layer

## 3.1 Layer Hardware

There is no specific overarching hardware for this layer.

## 3.2 Layer Operating System

The detection layer requires a basic operating system such as Windows or Linux. This is necessary to handle all data inputs as they proceed to the other layers.

## 3.3 Layer Software Dependencies

This layer is dependent on webcam operating software that is intrinsic to the Windows operating system. It can alternatively be managed through manufacturer specific camera software.

## 3.4 Computer Vision

The computer vision hardware consists of a 360 degree view web camera that has transmission via WiFi or hard-line. Image quality must be high definition and it must include the ability to dewarp via cube map or other methodology.

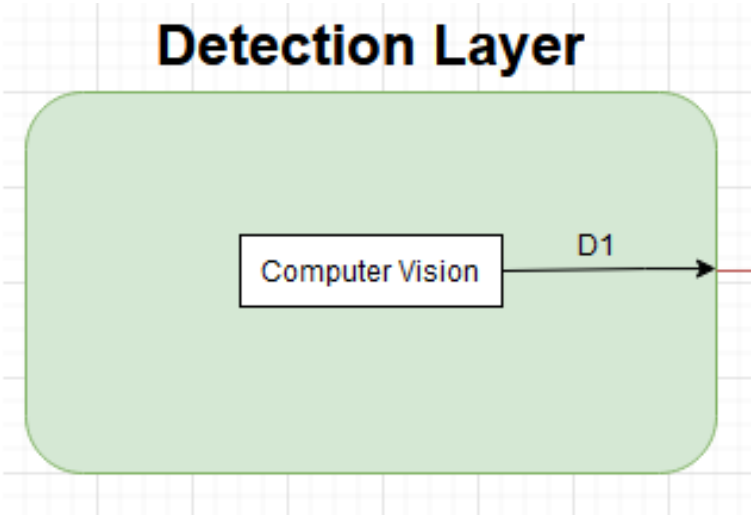


Figure 2: Detection Layer - Computer Vision

### 3.4.1 Subsystem Hardware

Hardware for this subsystem will consist of a web camera and connecting hard-line cable.

### 3.4.2 Subsystem Operating System

Operating system will be off of basic Windows or Linux.

### 3.4.3 Subsystem Software Dependencies

This subsystem is dependent on webcam software, whether intrinsic or installed on the operating system. Webcam drivers as well as display protocols are required.

### 3.4.4 Subsystem Programming Languages

The web camera does not utilize a programming language.

### 3.4.5 Subsystem Data Structures

Data from the webcam is transmitted via USB cable utilizing webcam and/or OS standard protocols. All data structure manipulation is handled in the processing layer.

### 3.4.6 Subsystem Data Processing

There are no data processing or algorithmic manipulations at this layer.

# 4 Processing Layer

## 4.1 Layer Hardware

Intel Nuc Mini Desktop System

## 4.2 Layer Operating System

Linux or Windows

## 4.3 Layer Software Dependencies

PyTorch library of Python

## 4.4 Image Dewarping

After the webcam has taken in a frame or image, the information is passed on to the dewarping algorithm. This algorithm will take the image data and transform it from a 360 degree fish eye view into a linear one that can be used by YOLOv7 to attempt to detect drones.

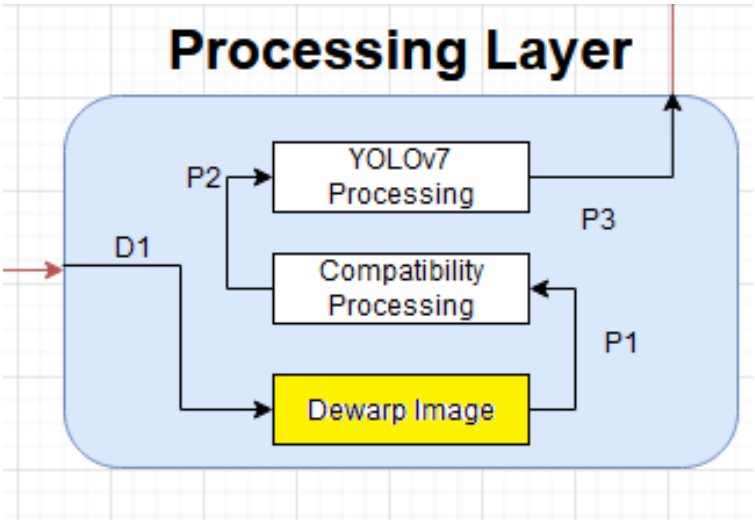


Figure 3: Processing Layer - Image Dewarping

### 4.4.1 Subsystem Hardware

There is no hardware for this process.

### 4.4.2 Subsystem Operating System

Dewarping software requires an operating system of Windows or Linux.

### 4.4.3 Subsystem Software Dependencies

Dewarping necessitates a software dependency on the video and image processing from the webcam software. This can be provided intrinsically by the OS or through driver software from the manufacturer.

### 4.4.4 Subsystem Programming Languages

Dewarping will be processed through software designed in the Python programming language.

#### 4.4.5 Subsystem Data Structures

Data structures necessary for the dewarping procedure consist of the live stream or image data from the webcam transmitted over USB cable or WiFi. This image data will be received by the dewarping software to process and pass further along the pipeline in the processing layer.

#### 4.4.6 Subsystem Data Processing

The dewarping procedure will follow either Equirectangular processing or Cube mapping. Both of these methods deliver slightly different results and it has yet to be determined which is more effective for drone detection.

### 4.5 Compatibility Processing

This pre-processing step might be necessary before passing the image or video stream to YOLO v7. Essentially, it's the final check on captured data before it gets tested by the machine learning model in YOLOv7.

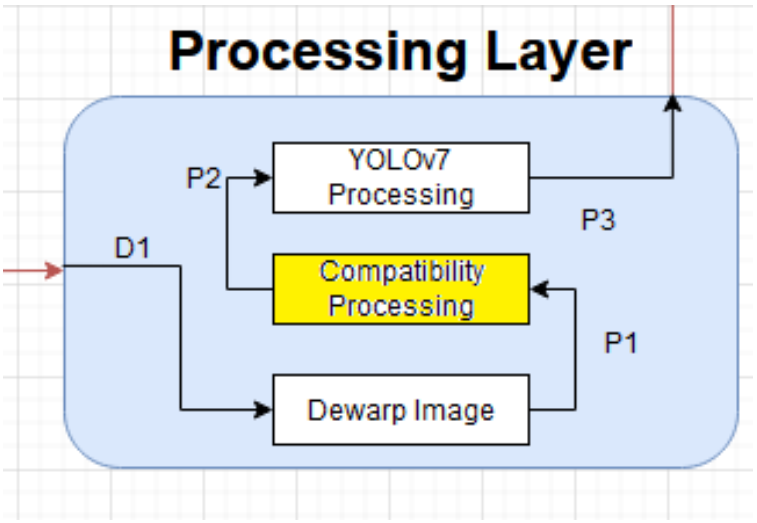


Figure 4: Processing Layer - Compatibility Processing

#### 4.5.1 Subsystem Hardware

No hardware is involved in this subsystem

#### 4.5.2 Subsystem Operating System

Windows or Linux

#### 4.5.3 Subsystem Software Dependencies

The subsystem might require packages pertaining to OpenCV or PyTorch for image processing.

#### 4.5.4 Subsystem Programming Languages

Python

#### 4.5.5 Subsystem Data Structures

Data structures depend on the type of computer vision library used but generally, they involve image or video capture.

### 4.5.6 Subsystem Data Processing

Pre-processing checks would involve the use of basic image transformations and filters involved with color spaces that would accentuate the edges and features of the object. Format conversions might be necessary.

## 4.6 YOLO V7 Processing

YOLOv7 (You Only Look Once version 7) is an object detection machine learning algorithm used in computer vision. It uses deep learning to identify objects in an image. It's designed to work quickly and efficiently, making it ideal for real-time applications such as video surveillance. In our case we use it for drone detection and tracking system.

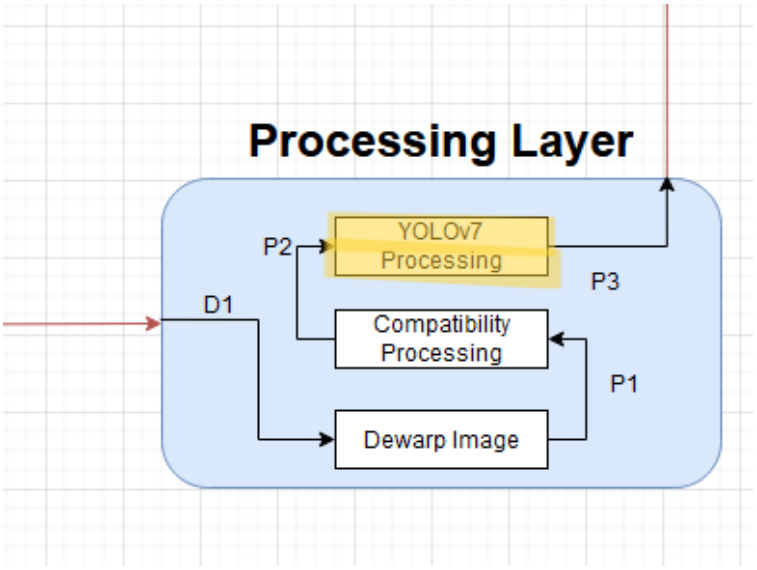


Figure 5: Processing Layer - YOLOv7 Processing

### 4.6.1 Subsystem Hardware

No hardware is involved in this subsystem

### 4.6.2 Subsystem Operating System

Windows or Linux

### 4.6.3 Subsystem Software Dependencies

The subsystem might requires bunch of packages that needs to be configured in order for YOLOv7 to work properly. Some are listed below:

- CUDA: If YOLOv7 needs to run in NVIDIA GPUs.
- OpenCV: OpenCV is a computer vision library that is often used with YOLOv7 to provide tools for image and video processing.
- PyTorch: YOLOv7 can be implemented using either TensorFlow or PyTorch, we are using PyTorch for the deep learning framework
- Python libraries: Numpy, Matplotlib, imutils,Pillow, Scikit-learn

### 4.6.4 Subsystem Programming Languages

Python

### 4.6.5 Subsystem Data Structures

It depends on the implementation of the framework library but typically the following data structures will be used:

- Image arrays: In our application, YOLOv7 operates on image data, so images are often represented as arrays of pixel values in memory.
- OpenCV: OpenCV is a computer vision library that is often used with YOLOv7 to provide tools for image and video processing.
- Bounding boxes: Bounding boxes are used to represent the location and size of objects in the image.
- Anchor boxes: Anchor boxes are predefined bounding boxes that are used to anchor the predictions to the objects in the image.
- Tensors: Tensors are multi-dimensional arrays that are used to represent data in deep learning frameworks like PyTorch.
- Classes: Classes are used to represent the drone and non drone objects that YOLOv7 can detect.

### 4.6.6 Subsystem Data Processing

Some Data Preprocessing to be used are as follows:

- Image resizing: YOLOv7 operates on a fixed-size input, so it's important to resize the images to the correct size before feeding them into the algorithm.
- Image normalization: Normalizing the pixel values in the image can help improve the performance of the network.
- Label encoding: The labels for the objects in the images should be encoded as integers, with each class having a unique integer label.
- Anchor boxes: Anchor boxes are predefined bounding boxes that are used to anchor the predictions to the objects in the image.
- Bounding box encoding: The bounding boxes for the objects in the images should be encoded using a format that is compatible with YOLOv7.

# 5 Input/Output Layer

## 5.1 Layer Hardware

Intel Nuc Mini Desktop

## 5.2 Layer Operating System

x86 Windows

## 5.3 Layer Software Dependencies

The application is mainly python.

## 5.4 Application

The application subsystem is a software application that runs live feed from an external visual sensor and analyzes it for the presence of drones frame by frame. It mainly houses yolov7, an object detection algorithm that uses machine learning.

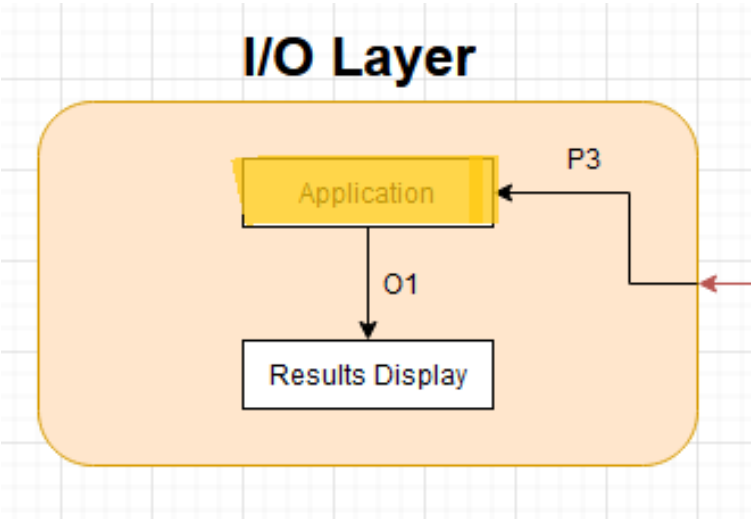


Figure 6: I/O Layer - Application Subsystem

### 5.4.1 Subsystem Hardware

Webcam connected to the Intel Nuc Mini Desktop system.

### 5.4.2 Subsystem Operating System

Any x86 system.

### 5.4.3 Subsystem Software Dependencies

Mainly yolov7 and python3+. Other software dependencies might be included as child processes running off of the main python program.

### 5.4.4 Subsystem Programming Languages

Python. Other Languages may be used as child processes.

### 5.4.5 Subsystem Data Structures

N/A

### 5.4.6 Subsystem Data Processing

Dewarping process for images.

## 5.5 Results Display

The results display falls under the application subsystem which mainly focuses on how the final output is displayed to the user.

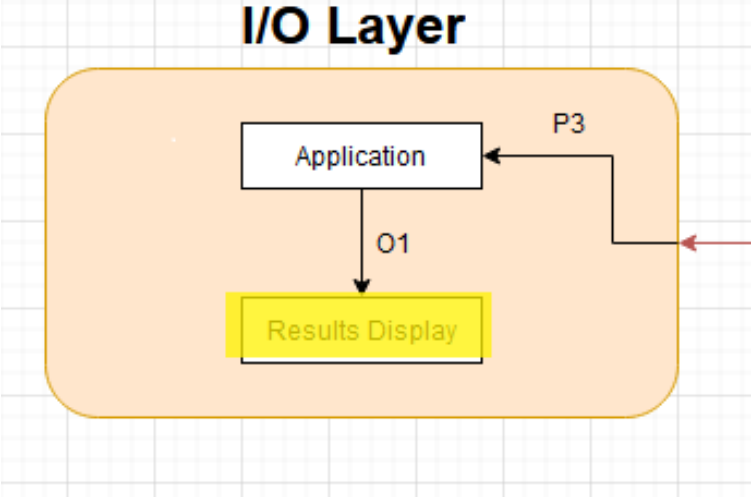


Figure 7: I/O Layer - Results Display

### 5.5.1 Subsystem Hardware

Intel Nuc Mini Desktop system. It is a mini desktop computer that provides high performance and versatility in a compact form factor. The NUC system consists of a small motherboard with integrated CPU, memory, storage, and other components that fit into a compact, customizable chassis. The Intel NUC is also highly customizable. It supports a wide range of peripheral devices, including the 360 angle camera we are using for the project.

### 5.5.2 Subsystem Operating System

As the Intel NUC mini computer does not come with an operating system pre-installed, we are responsible for installing the OS of our choice. For now, both windows and linux are our options with both 32 bit and 64 bit x-86 architecture.

### 5.5.3 Subsystem Software Dependencies

Mainly YOLOv7 and python. As YOLOv7 is an implementation of a deep learning model written in python, it further relies on a number of software dependencies such as OpenCV for image processing and computer vision, Tensorflow/PyTorch for training and running deep learning models, Darknet for training and running YOLOv7.

### 5.5.4 Subsystem Programming Languages

Python and JavaScript are used for this subsystem to view results of whether the detected object in the dewarped image is a drone or not a drone. This subsystem only shows the user whether the object is a drone or not by drawing a rectangular box around the object with the likely probability that the detected object is a drone.

### 5.5.5 Subsystem Data Structures

Arrays, Strings, Lists in both Javascript and python.

### 5.5.6 Subsystem Data Processing

Conversion of JavaScript object containing de-warped images from the camera to OpenCV BGR image. Another is conversion of OpenCV Rectangle bounding box image to base64 byte string. Another would be creation of a live video stream where the JavaScript code creates a live video stream from the camera and displays it on the web page as a HTML element.

# 6 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

## REFERENCES