Final Year Project

AuxVision – Assisted Vision for the Visually Impaired

Mid Evaluation – Report

FYP Team

Ali Asghar	(18I-0475)
Affan Arif	(18I-0484)
Buraq Khan	(18I-0800)

Supervised by

Dr. Zohaib Iqbal

FAST School of Computing

National University of Computer and Emerging Sciences

Islamabad, Pakistan

2021

Anti-Plagiarism Declaration

This is to declare that the above FYP report produced under the:

Title: ____

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as **Plagiarism.** All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: ________________ Student 1

Name: _____________________

Signature: ___________________

Student 2

Name: _____________________

Signature: ___________________

Student 3

Name: _____________________

Signature: ___________________

Supervisor (Faculty)

Name: _____________________

Signature: ___________________

Executive Summary

The ability to see and perceive everything around us is a blessing from God. Unfortunately, a section of our society suffers from vision impairment. Being unable to perform day to day tasks independently is a major hindrance in the lives of those who are blind. They are dependent on other human beings, special tools (walking sticks) and even ‘guide dogs’ that help with navigation.

This report aims to firstly present the problem to the reader, define the project’s scope, identify key stakeholders, define constraints/limitations, and to present expected outcomes. Afterwards, we will explore the different methods used to derive the solution along with an iterative plan that highlights what we aim to do in each iteration. For the reader’s convenience, the documentation encompasses various diagrams such as:

- Use Case Diagram.
- Activity Diagram.
- Domain Model.
- System Sequence Diagrams.
- Sequence Diagrams.
- Class Diagram.
- Architecture Diagram.

Our project’s iterative plan can be divided into four major iterations as follows:

1. Perception of the Environment.
2. Providing feedback to the user through TTS (text-to-speech).
3. Extensive testing of previous modules in different environments.
4. Integration of all modules into a single wearable device.

In the end, we present our implementation’s details to make the reader understand how we have combined hardware and software to present our solution.

Table of Contents

Introduction 7

Project Vision 7

Problem Statement 7

Business Opportunity 7

Objectives 8

Project Scope 8

Constraints 9

Stakeholders Description 9

Stakeholders Summary 9

Key High-Level Goals and Problems of Stakeholders 10

Software Requirements Specifications 10

List of Features 10

Functional Requirements 11

Quality Attributes 11

Non-Functional Requirements 11

High Level Use Cases 12

Use Case Diagram 13

Iteration Plan 14

Iteration 1 14

Expanded Use Cases 14

Activity Diagram 16

Domain Model 18

System Sequence Diagrams 19

Sequence Diagrams 21

Class Diagram 23

Architecture Diagram 24

Iteration 2 24

Expanded Use Cases 24

Activity Diagrams 26

Domain Model 27

System Sequence Diagrams 28

Sequence Diagrams 29

Class Diagram 30

Architecture Diagram 31

Package Diagram 32

Deployment Diagram 33

Implementation Details 34

Arduino: 34

SONAR Sensor: 35

Stereo Camera: 36

Camera Calibration 36

Stereo Rectification: 37

Disparity Calculation: 38

Distance Estimation: 39

References 41

1. Introduction

"AuxVision – Assisted Vision for the Visually Impaired" aims to change the lives of the visually impaired by enabling them to navigate autonomously in indoor environments. To accomplish this task, we will rely on hardware components such as different sensors to perceive the environment. All this will be combined to make a wearable device. We will scan the environment around the individual, identify the type of objects in the vicinity and at what distance they are to the person. All this information will then be converted from textual format to speech for the person to interpret. The user can then make an informed decision as to what they want to do.

2. Project Vision

2.1. Problem Statement

The visually impaired community has faced incredible difficulty since the start of time in accomplishing even day-to-day tasks. Finding their way around, locating objects of interest and colliding with obstacles are just some of the challenges they face. The need for independence in their lives holds utmost importance and we plan on providing a novel solution to this problem.

2.2. Business Opportunity

According to a study, in 2015 there were an estimated 253 million people with visual impairment worldwide, and of these about 36 million were completely blind and a further 217 million had moderate to severe visual impairment (Ackland et al., 2017). Since we have an aging population and that the risk of most eye conditions increases with age, the prevalence of blindness and moderate to severe visual impairment is expected to increase in the upcoming years. With so many people at risk, visual assistance for the visually impaired is a need of the hour, hence it is a good business opportunity which is potentially life-changing.

2.3. Objectives

1. Conveying the locations of objects of interest and at what distance they are to the person.

2. Providing the user with constructive feedback based on perception of the environment for decision making using text to speech translation.

3. Collision avoidance for safe and effective navigation.

4. Make an easy to carry and hassle-free device, for example; in the form of a backpack.

5. Create value in the life of the user by allowing them to navigate independently without any external assistance so that they can carry out day to day tasks with ease.

2.4. Project Scope

We have split our project into three primary modules:

- Gathering information from the environment as input, which encompasses the object detection module and distance calculation/estimation module. For this purpose, we will rely on sensors such as LiDAR, Stereo Camera and SONAR based on whichever are most suitable for our needs. We will be using object detection models such as MobileNet. The types of objects that we will be catering are as follows:

    1. Common household furniture (chairs, bed, couch etc.)
    2. Everyday use appliances (Mobile, microwave, laptop, TV etc.)

3. General items (Vase, cup, bottle, bag, toothbrush, fork, spoon etc.)
4. Food items (Fruits and vegetables, cakes etc.)
5. Humans and animals.
6. Vehicles.

- Communicating feedback to the user in the form of text to speech, taking into consideration the perception of the environment. This information will be communicated using an audio channel such as a wireless headphone with Bluetooth.
- A wearable device such as a backpack which will be equipped with sensors, a computing device in the form of either a laptop or Arduino/Raspberry Pi and a battery pack to power the Arduino/Raspberry Pi if required.

All of these features will be implemented considering the convenience of the user.

2.5. Constraints

- Appropriate lighting conditions.
- Insufficient computing power.
- Overheating of the computing device placed in the backpack.
- Stereo camera stability.
- Complex deep learning modules are not feasible for real-time object detection.
- Poor performance in outdoor environments.

2.6. Stakeholders Description

2.6.1. Stakeholders Summary

Visually Impaired Person: The person who will be using the device to navigate autonomously.

Emergency Handler: The person who in case any of the sensors stops working or gets unplugged, will fix the problem.

2.6.2. Key High-Level Goals and Problems of Stakeholders

The visually impaired person will be able to achieve the following high-level goals:

1. Using sensors to perceive the environment by detecting objects and estimating how far they are to the user.
2. Receiving feedback in the form of TTS (text-to-speech).
3. Make an informed decision after receiving feedback from the device.

The following problems may be encountered:

1. Failure of hardware like any one of the sensors gets disconnected or damaged.

3. Software Requirements Specifications

3.1. List of Features

1. Perceive the environment through input from sensors by object detection and distance estimation.
2. Receive feedback from the sensors with regards to perception of the environment.
3. Collision avoidance to prevent the user from falling or colliding with objects.
4. Be able to navigate indoor environments without external assistance.

3.2. Functional Requirements

- The user should be able to initiate the device.
- The user should be able to terminate the device.
- The device should be able to perceive the environment around the user.
- The device should be able to provide feedback to the user by identifying objects of interest.
- The device should be able to prevent collision with objects by providing feedback to the user.

3.3. Quality Attributes

- Performance:
  The system provides results in real-time while running components such as sensors and the object detection model.
- Reliability.
  As long as all components are connected and the system's battery doesn't finish it will operate normally.
- Availability.
  In case any component gets disconnected, it will be reconnected within two to three minutes and the system will be back online.
- Testability:
  System will be tested in different environments to test the working of the system in different scenarios.
- Usability:

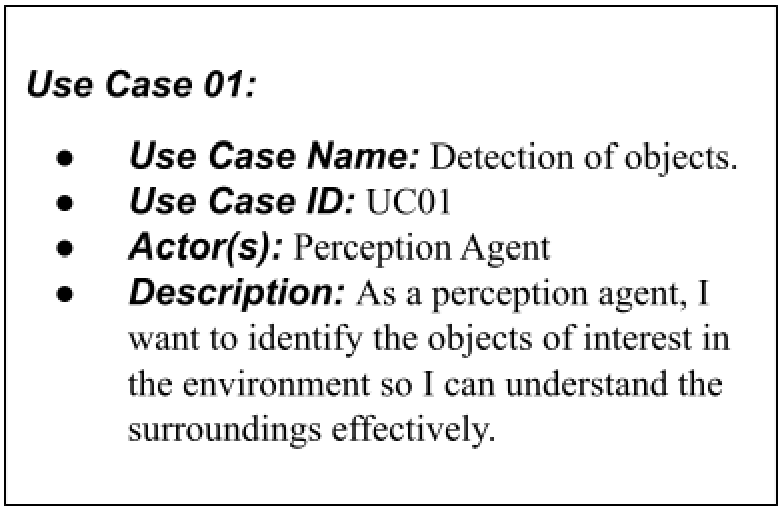
  Once the system is online, it doesn't require the user's input continuously hence it is easy to use.

3.4. Non-Functional Requirements

- System shall be able to detect objects and estimate their distance in real-time, i.e., be able to do both of these things within 5 seconds.

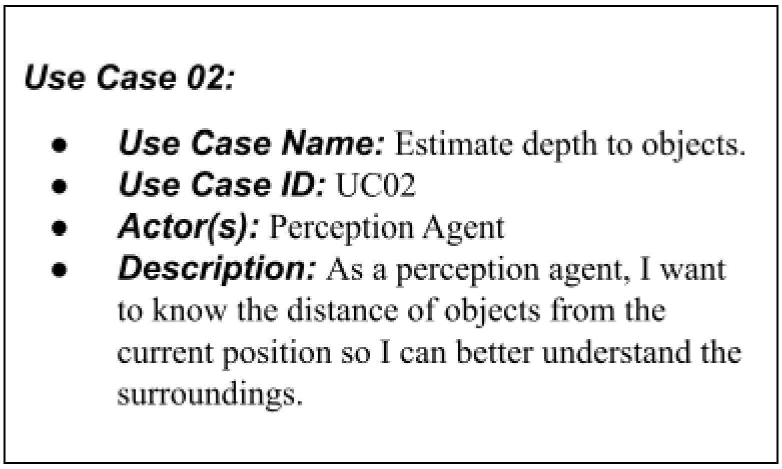
- Feedback about surroundings shall be provided to the user within 5 seconds when required.
- In case the user is about to collide with an object, the system shall provide warning through feedback within 3 seconds.
- System shall be easy to use for the user.
- System will remain online as long as the battery lasts.

4. High Level Use Cases



Use Case 01:

- ***Use Case Name:*** Detection of objects.
- ***Use Case ID:*** UC01
- ***Actor(s):*** Perception Agent
- ***Description:*** As a perception agent, I want to identify the objects of interest in the environment so I can understand the surroundings effectively.



Use Case 02:

- ***Use Case Name:*** Estimate depth to objects.
- ***Use Case ID:*** UC02
- ***Actor(s):*** Perception Agent
- ***Description:*** As a perception agent, I want to know the distance of objects from the current position so I can better understand the surroundings.

Use Case 03:

- ***Use Case Name:*** Collision avoidance.
- ***Use Case ID:*** UC03
- ***Actor(s):*** Perception Agent
- ***Description:*** As a perception agent, I want to generate an alert in case there is about to be a collision with any obstacle so that I can understand the surroundings better.

4.1. Use Case Diagram

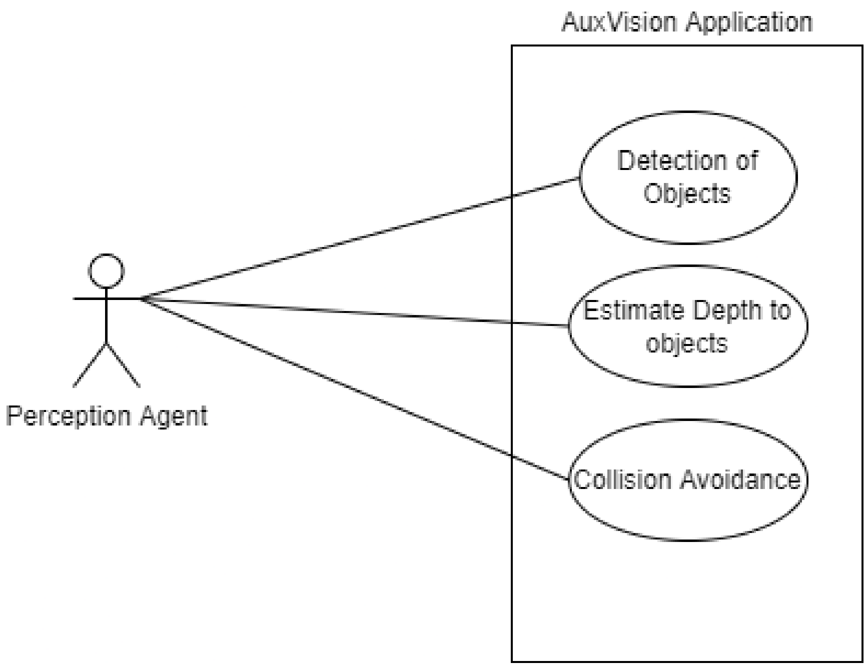


Figure 1: Use Case Diagram

5. Iteration Plan

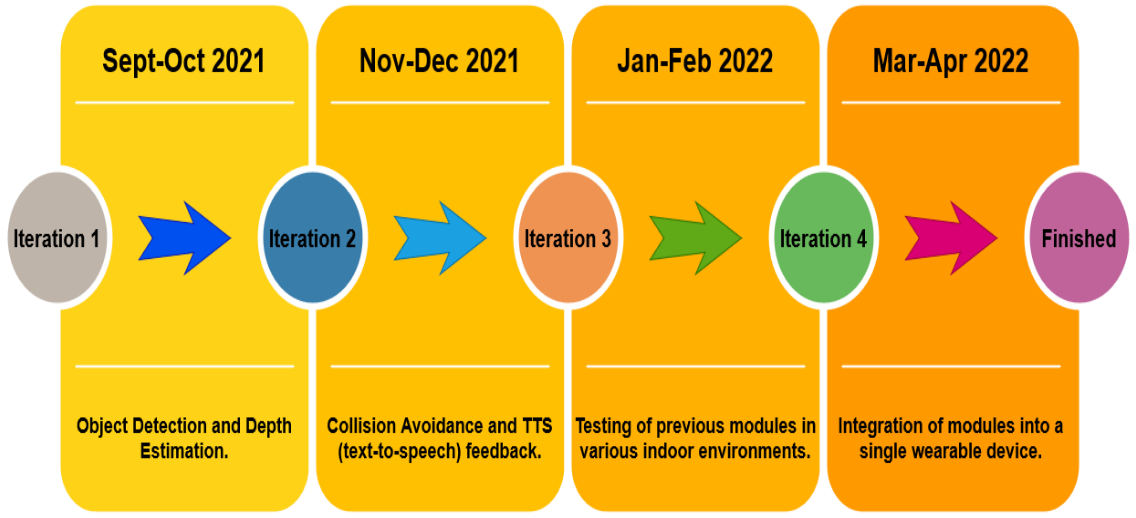

Sept-Oct 2021
Nov-Dec 2021
Jan-Feb 2022
Mar-Apr 2022
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Finished
Object Detection and Depth Estimation.
Collision Avoidance and TTS (text-to-speech) feedback.
Testing of previous modules in various indoor environments.
Integration of modules into a single wearable device.

Figure 2: Iteration Plan

6. Iteration 1

6.1. Expanded Use Cases

Use Case: UC01 – Detection of objects

Primary Actor: Perception Agent.

Preconditions: System is powered on.

Actors Action	System Response
1. Perception agent initiates the object detection module.	
	2. System identifies objects placed in the vicinity which are in the field of view through the camera.
	3. System shows the labels of the identified objects on the computation device.

	4. System prepares for providing feedback.

Use Case: UC02 – Estimate depth to objects

Primary Actor: Perception Agent.

Preconditions: System is powered on.

Actors Action	System Response
1. Perception agent initiates the object depth estimation module.	
	2. System identifies the distance to the objects in its vicinity through SONAR and Stereo Camera sensors.
	3. System shows the distance of the objects on the computation device.
	4. System prepares for providing feedback.

6.2. Activity Diagram

UC01 – Detection of objects:

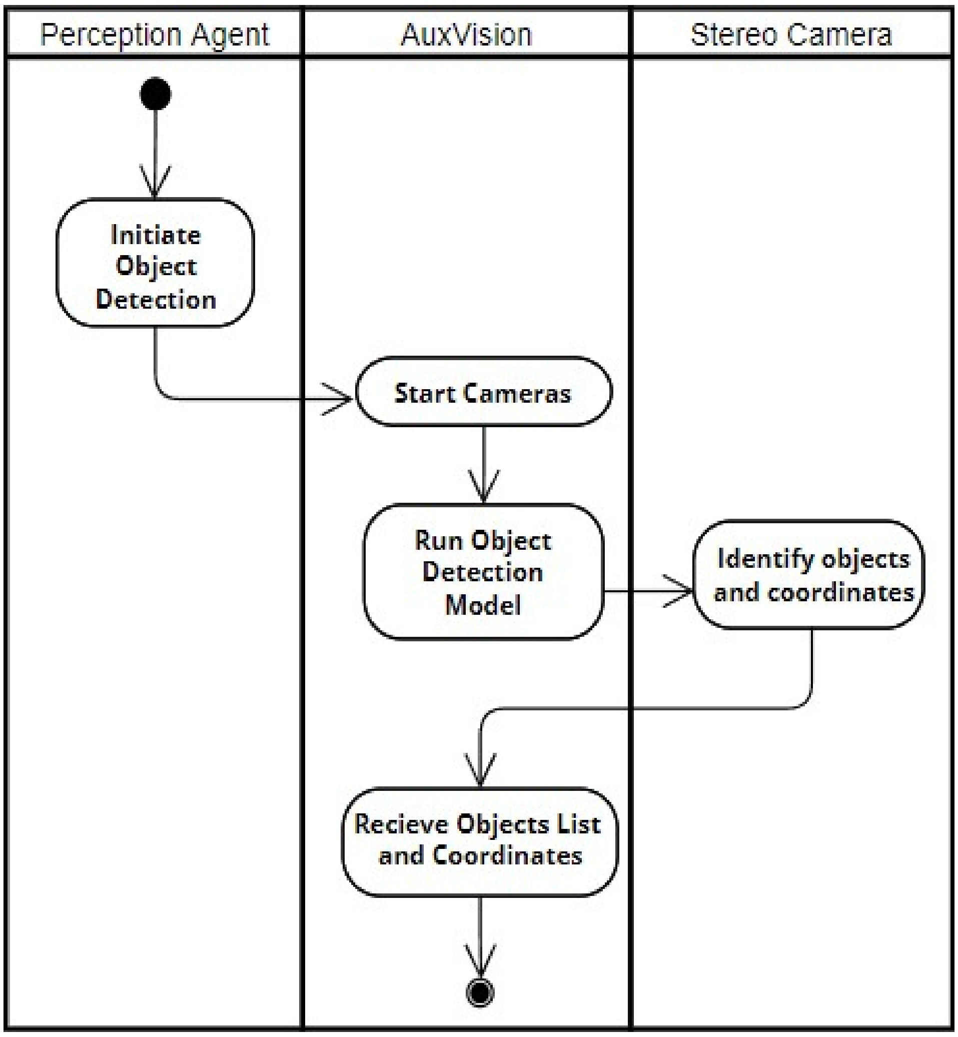

Perception Agent
AuxVision
Stereo Camera
Initiate Object Detection
Start Cameras
Run Object Detection Model
Identify objects and coordinates
Recieve Objects List and Coordinates

Figure 3: Activity Diagram - Detection of objects

UC02 – Estimate depth to objects:

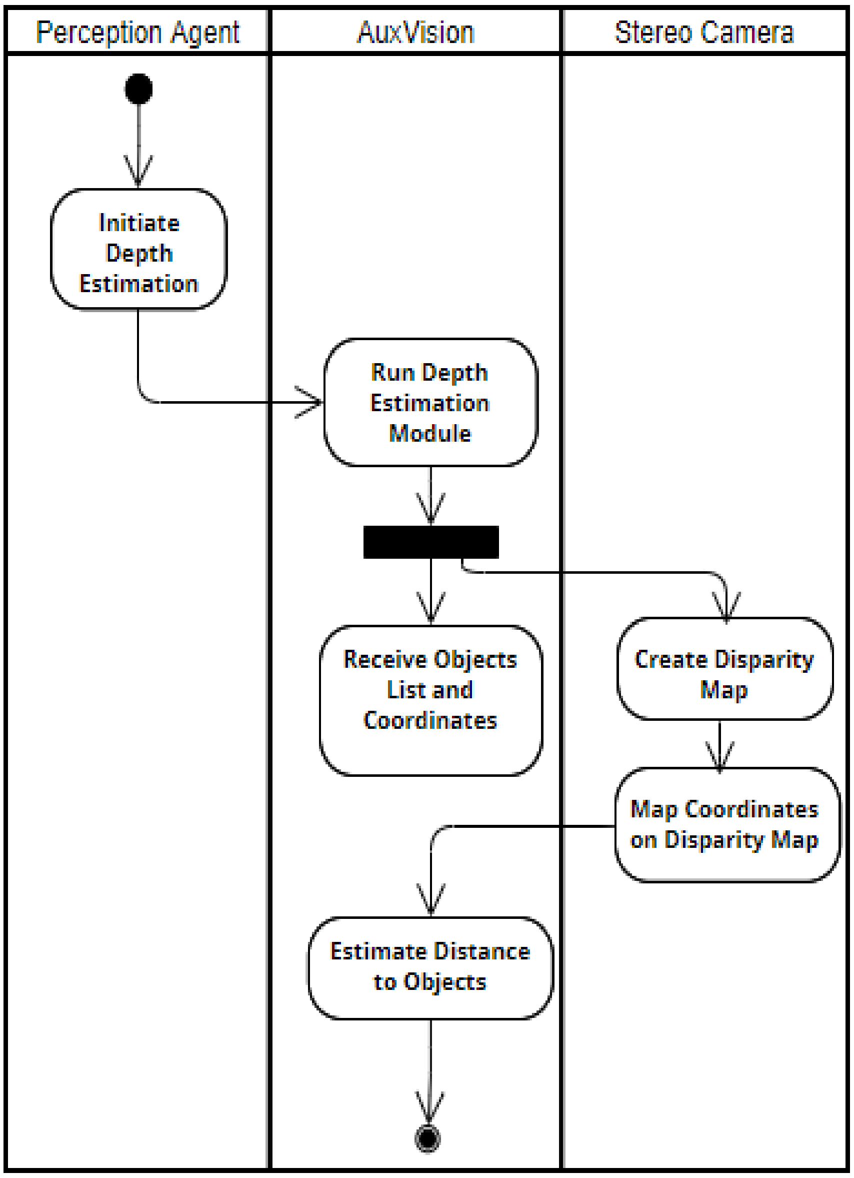

Perception Agent
AuxVision
Stereo Camera
Initiate Depth Estimation
Run Depth Estimation Module
Receive Objects List and Coordinates
Create Disparity Map
Map Coordinates on Disparity Map
Estimate Distance to Objects

Figure 4: Activity Diagram - Estimate depth to objects

6.3. Domain Model

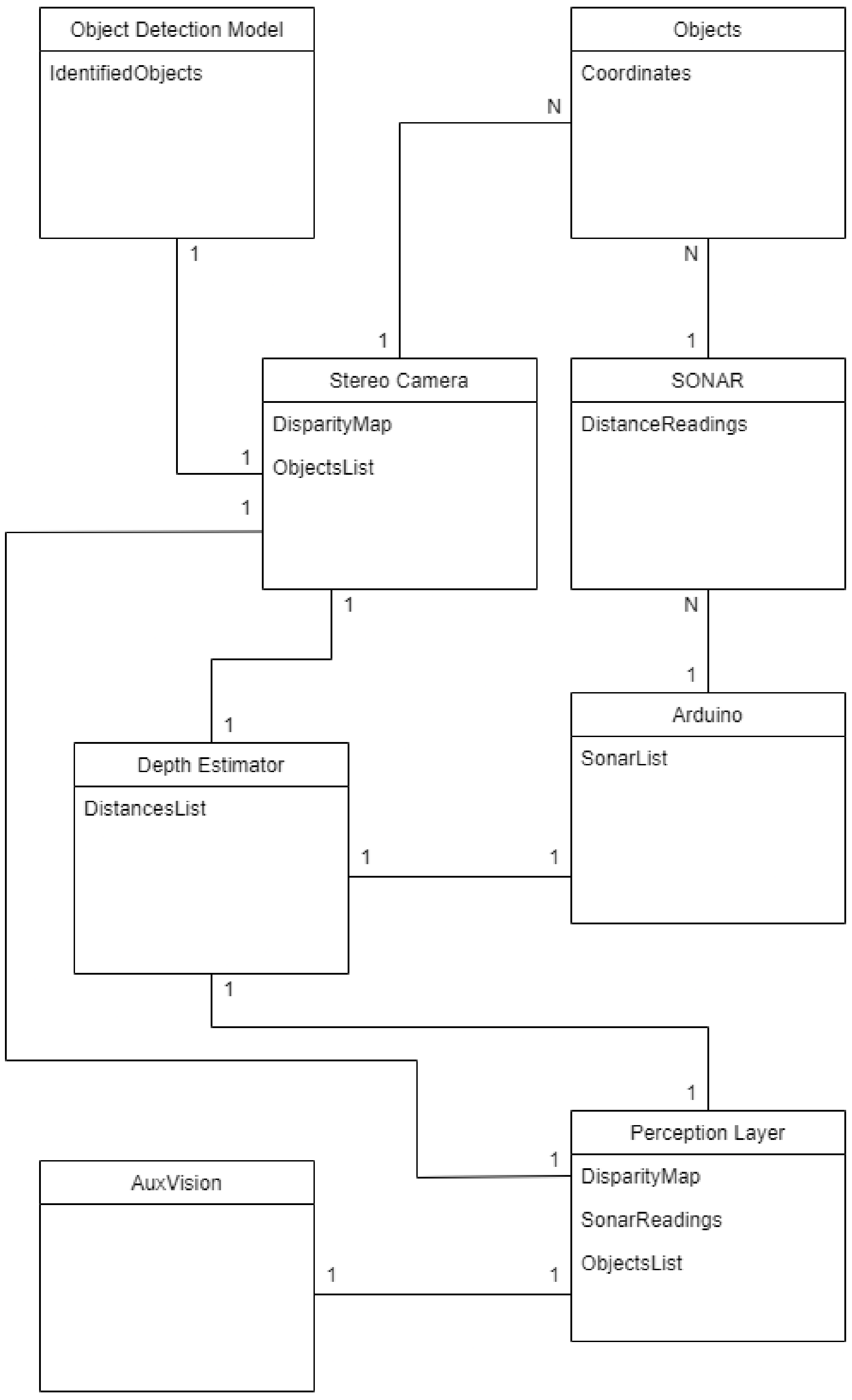


Figure 5: Domain Model

6.4. System Sequence Diagrams

Use Case: UC01 – Detection of objects

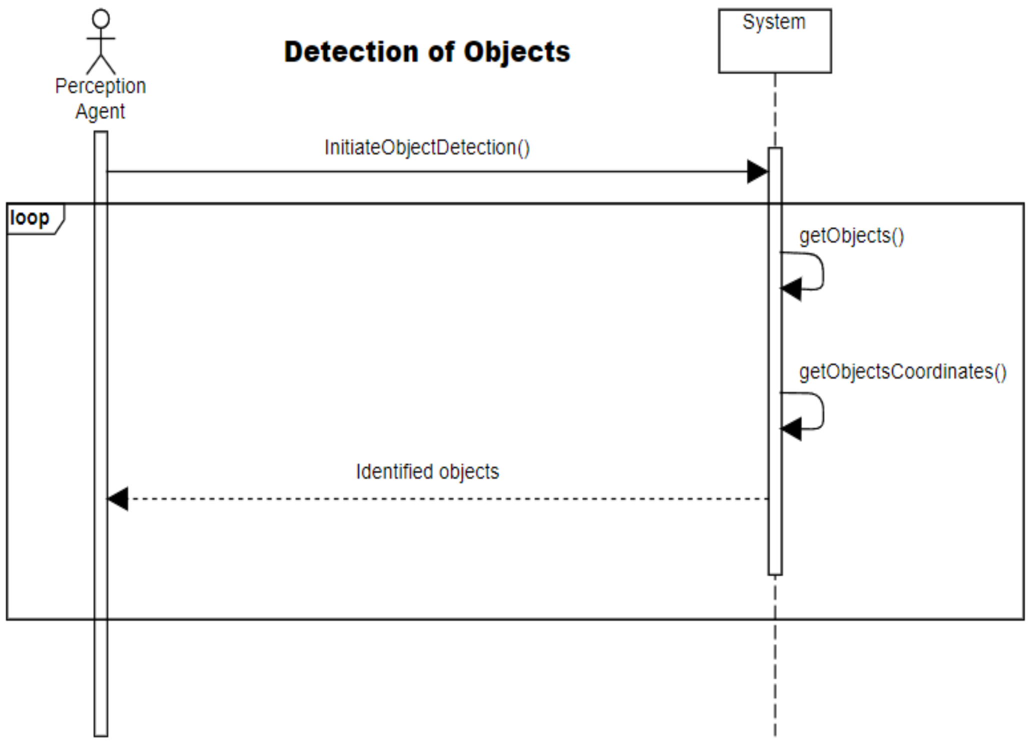


Figure 6: System Sequence Diagram - Detection of objects

Use Case: UC02 – Estimate depth to objects

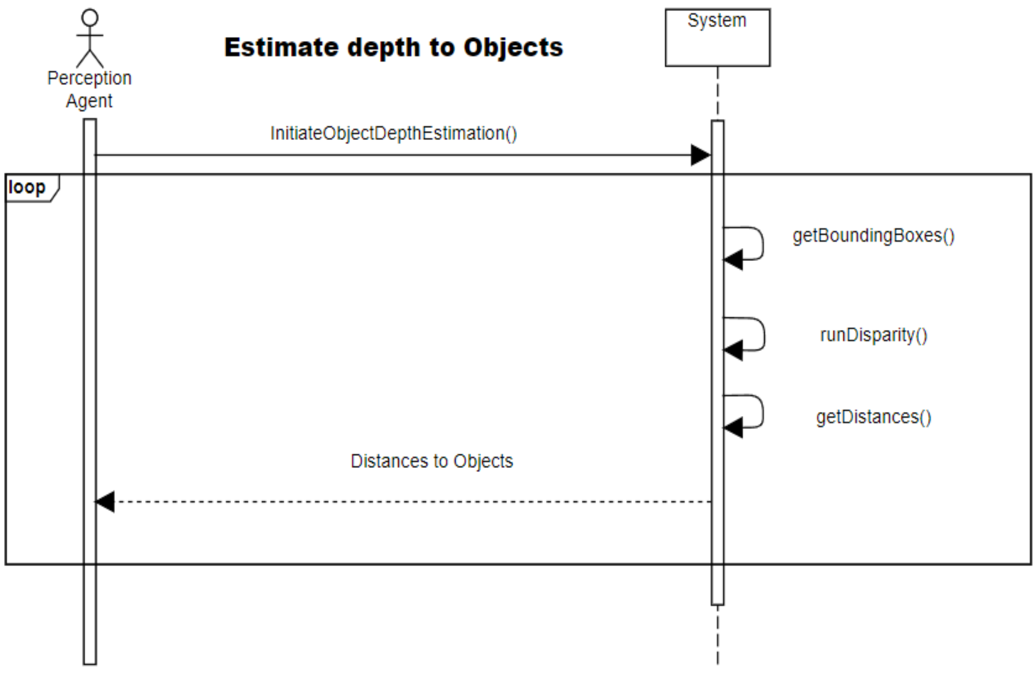


Figure 7: System Sequence Diagram - Detection of objects

6.5. Sequence Diagrams

System Event: Detection of Objects

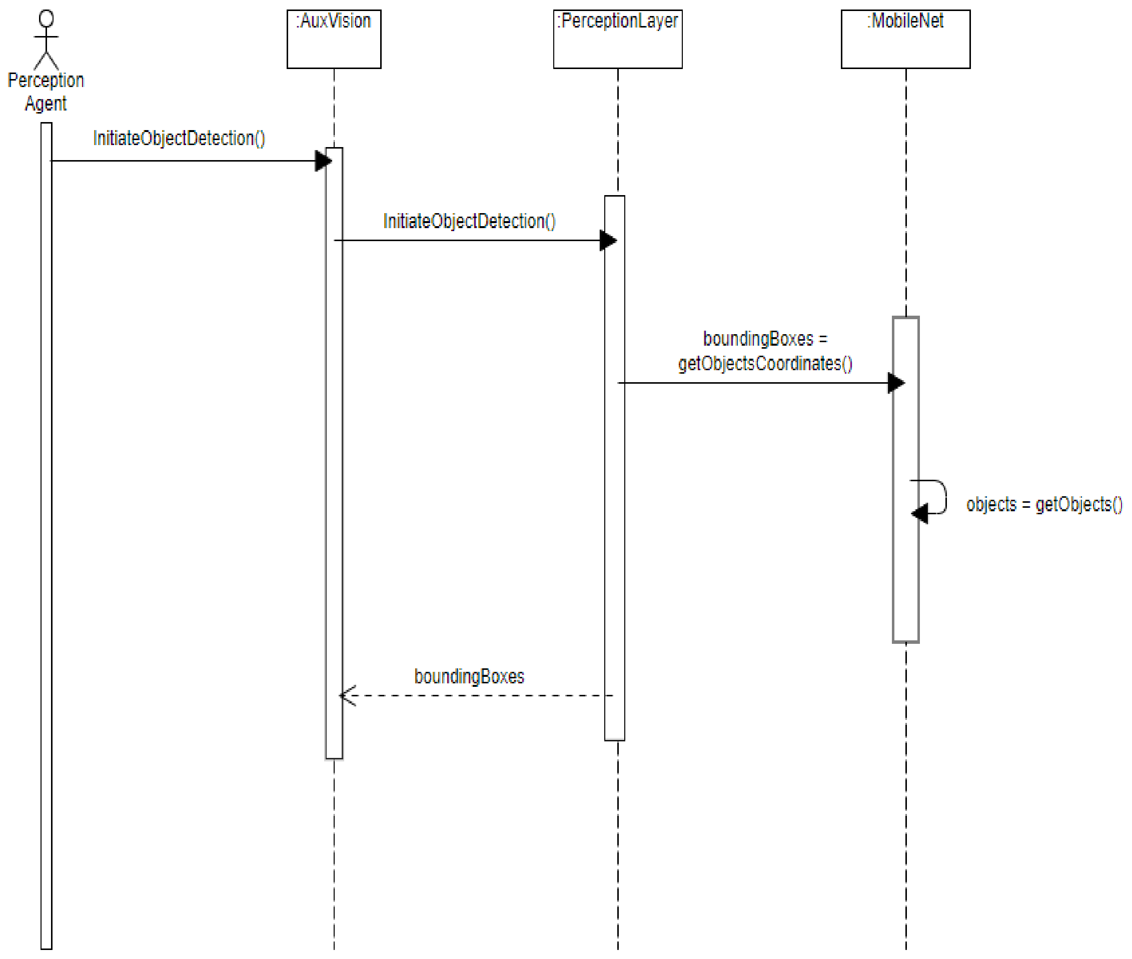

Perception Agent
:AuxVision
:PerceptionLayer
:MobileNet
InitiateObjectDetection()
InitiateObjectDetection()
boundingBoxes = getObjectsCoordinates()
objects = getObjects()
boundingBoxes

Figure 8: Sequence Diagram - Detection of objects

System Event: Estimate Depth to Objects

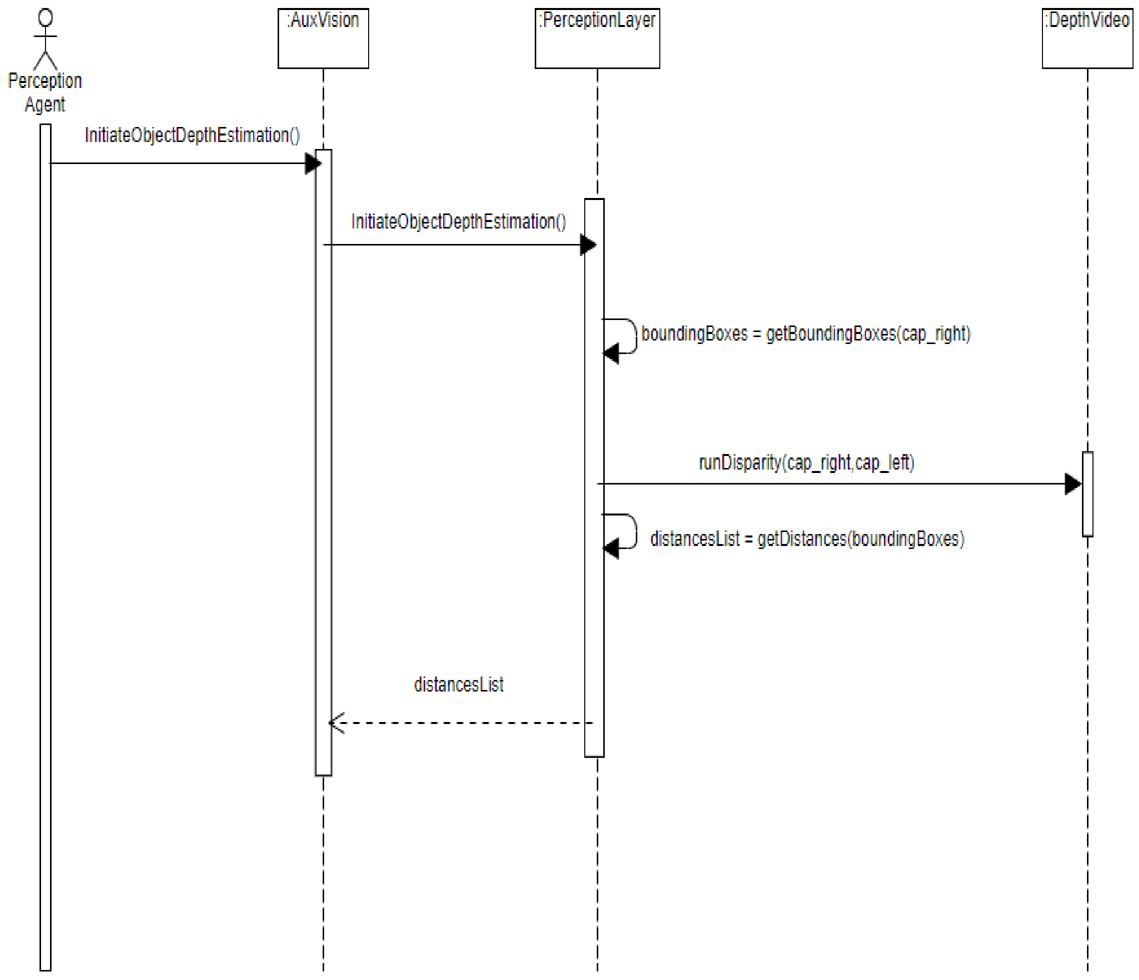
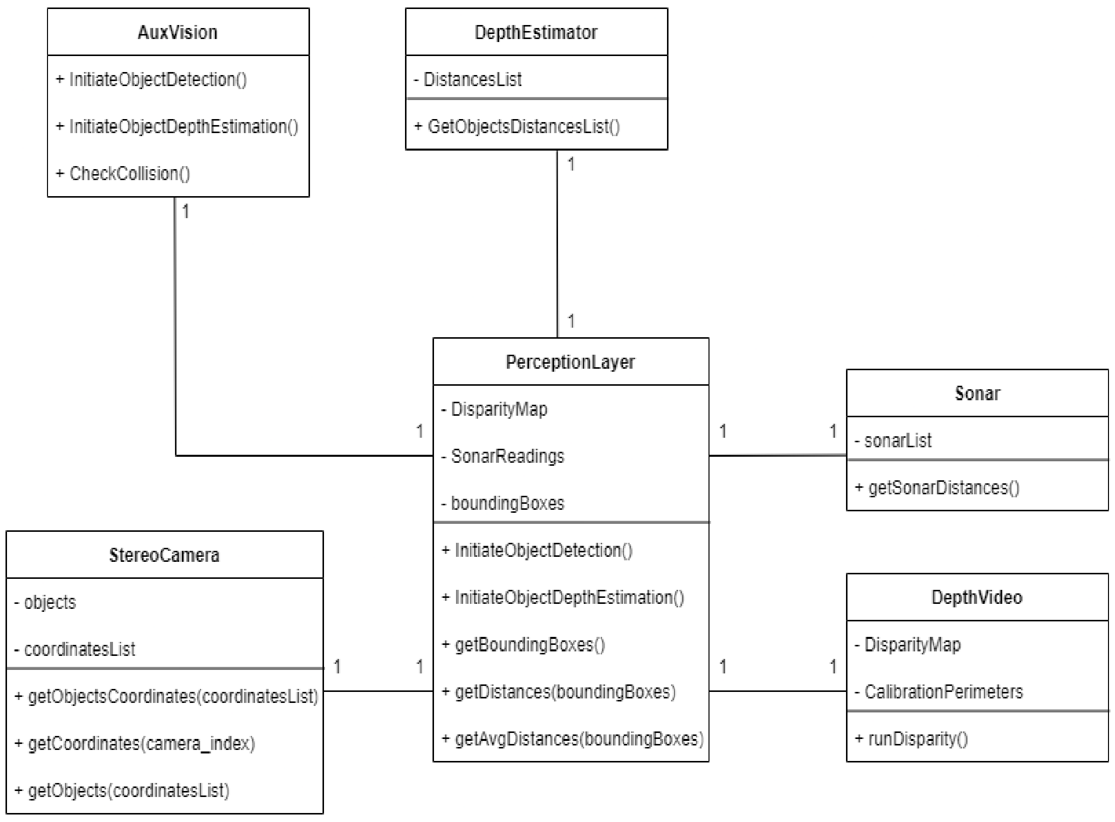

Perception Agent
:AuxVision
:PerceptionLayer
:DepthVideo
InitiateObjectDepthEstimation()
InitiateObjectDepthEstimation()
boundingBoxes = getBoundingBoxes(cap_right)
runDisparity(cap_right,cap_left)
distancesList = getDistances(boundingBoxes)
distancesList

Figure 9: Sequence Diagram - Estimate Depth to objects

6.6. Class Diagram



AuxVision
+ InitiateObjectDetection()
+ InitiateObjectDepthEstimation()
+ CheckCollision()
DepthEstimator
- DistancesList
+ GetObjectsDistancesList()
PerceptionLayer
- DisparityMap
- SonarReadings
- boundingBoxes
+ InitiateObjectDetection()
+ InitiateObjectDepthEstimation()
+ getBoundingBoxes()
+ getDistances(boundingBoxes)
+ getAvgDistances(boundingBoxes)
Sonar
- sonarList
+ getSonarDistances()
StereoCamera
- objects
- coordinatesList
+ getObjectsCoordinates(coordinatesList)
+ getCoordinates(camera_index)
+ getObjects(coordinatesList)
DepthVideo
- DisparityMap
- CalibrationPerimeters
+ runDisparity()
1
1
1
1
1
1
1
1
1
1
1
1

Figure 10: Class Diagram

6.7. Architecture Diagram

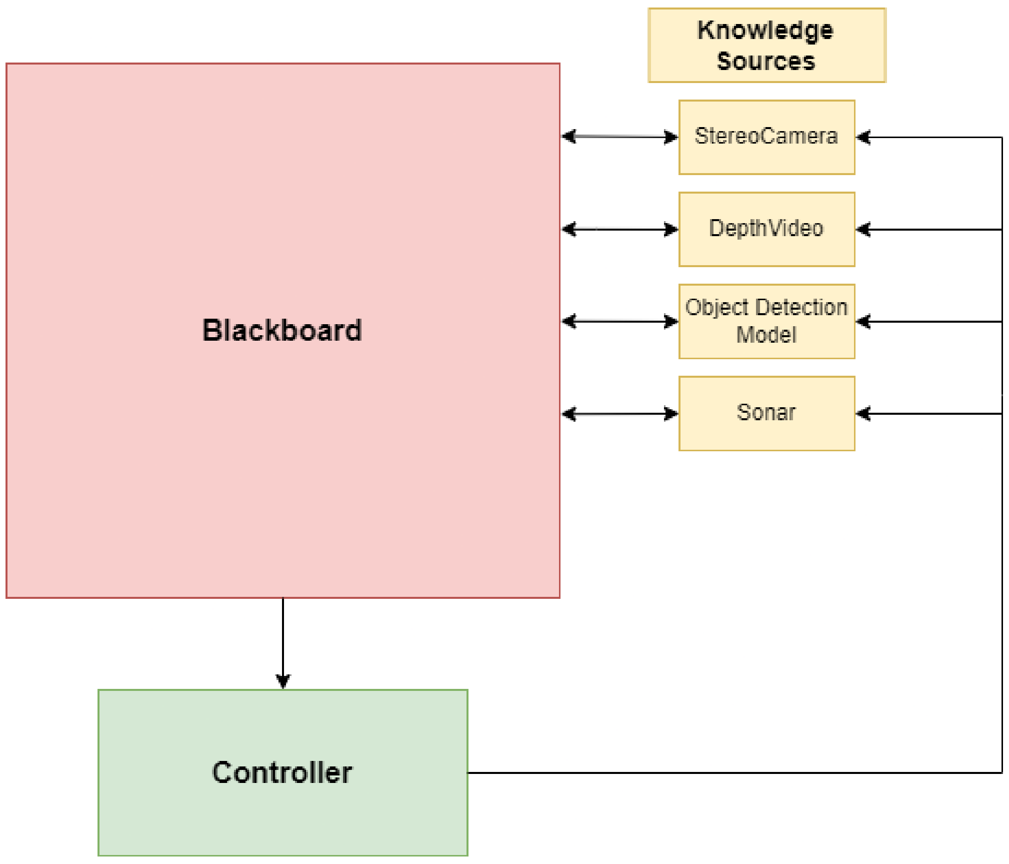


Figure 11: Architecture Diagram

7. Iteration 2

7.1 Expanded Use Cases

Use Case: UC03 – Collision Avoidance

Primary Actor: Perception agent.

Preconditions: System is powered on.

Actors Action	System Response
1. Perception agent initiates collision avoidance module..	
	2. System identifies obstacles in the vicinity through sensors.

	3. System generates alerts of potential collision with obstacles through Text To Speech (TTS) feedback via earphones.

7.2 Activity Diagrams

UC03 – Collision Avoidance:

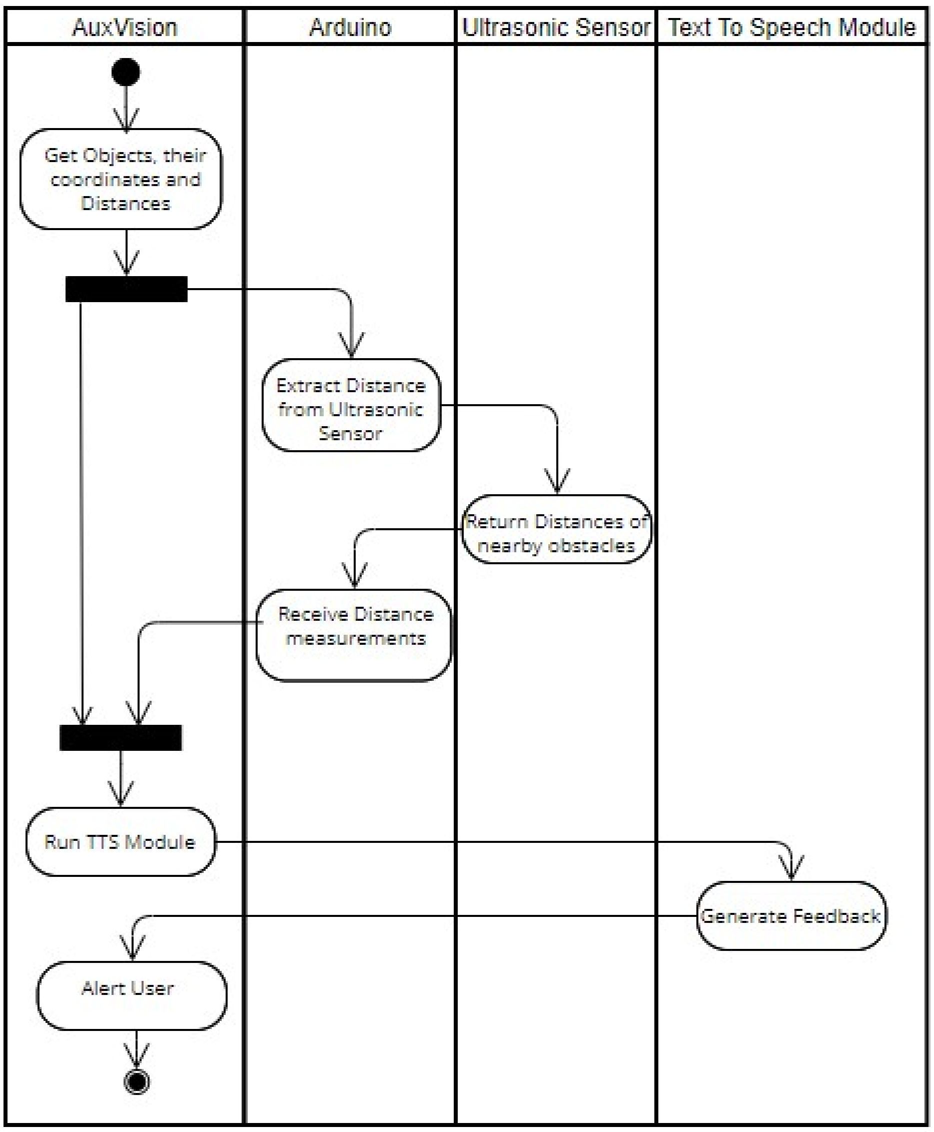

AuxVision
Arduino
Ultrasonic Sensor
Text To Speech Module
Get Objects, their coordinates and Distances
Extract Distance from Ultrasonic Sensor
Return Distances of nearby obstacles
Receive Distance measurements
Run TTS Module
Generate Feedback
Alert User

Figure 12: Activity Diagram - Collision Avoidance

7.3 Domain Model

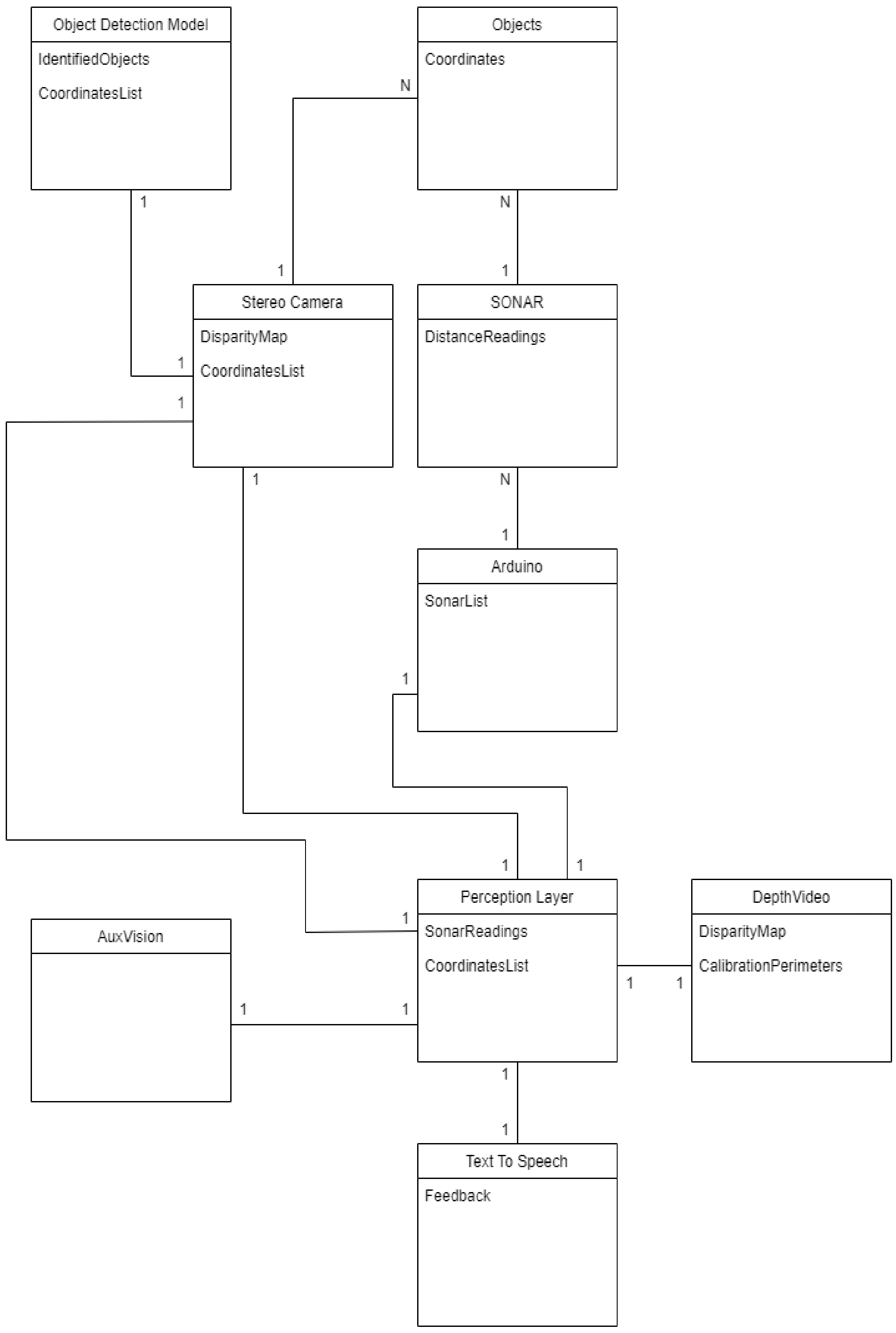


Figure 13: Domain Model

7.4 System Sequence Diagrams

Use Case: UC03 – Collision Avoidance

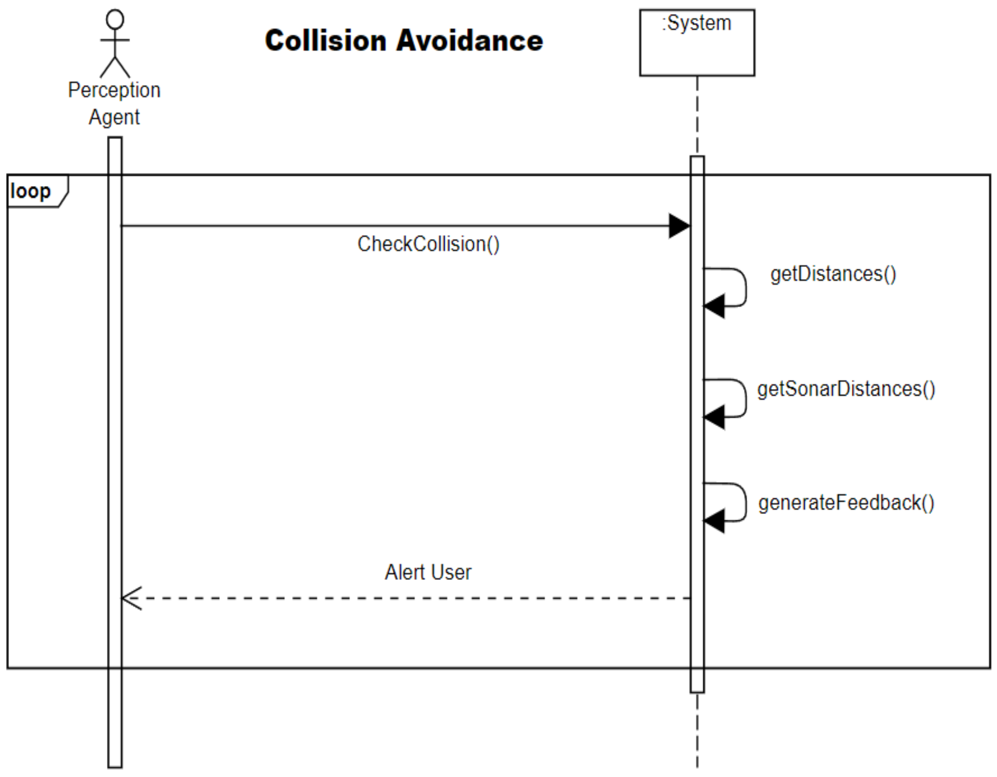


Figure 14: System Sequence Diagram- Collision Avoidance

7.5 Sequence Diagrams

System Event: Collision Avoidance

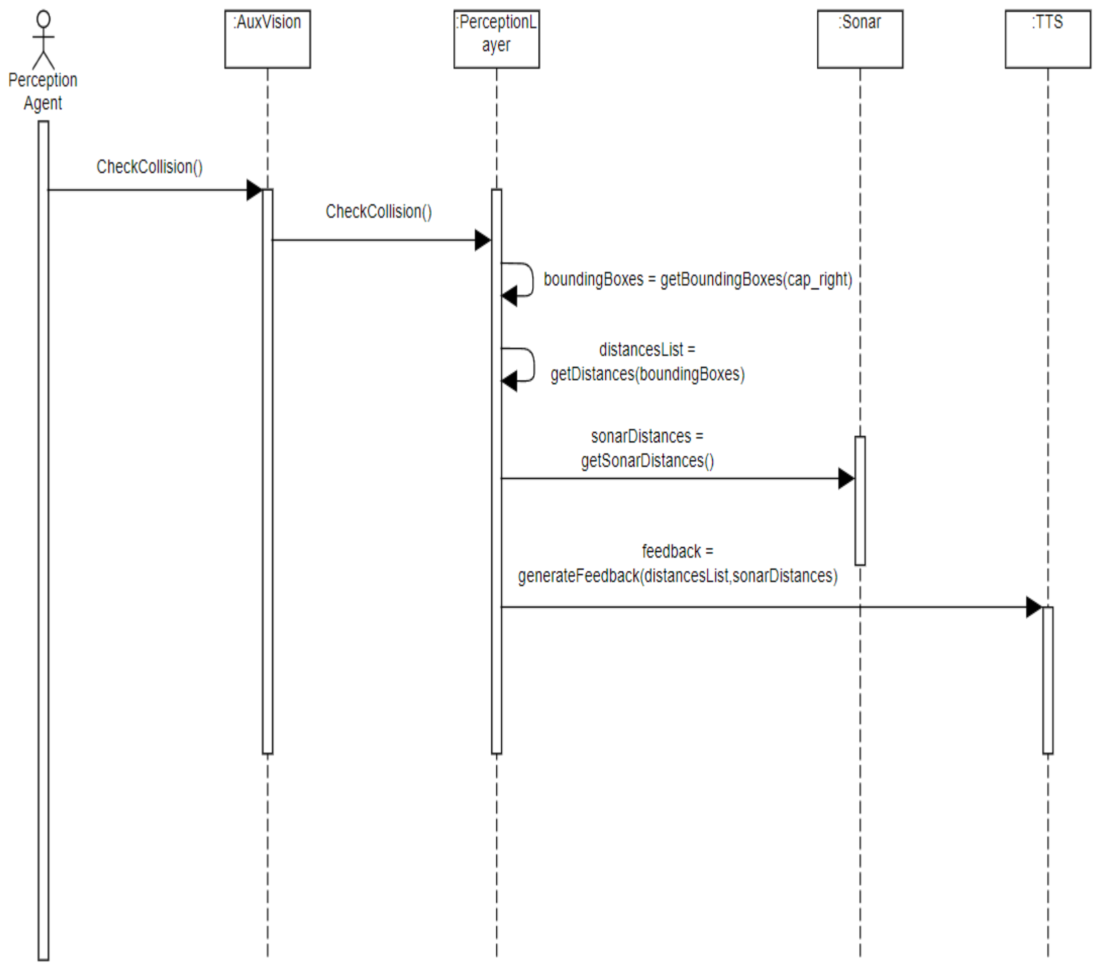

Perception Agent
:AuxVision
:PerceptionLayer
:Sonar
:TTS
CheckCollision()
CheckCollision()
boundingBoxes = getBoundingBoxes(cap_right)
distancesList = getDistances(boundingBoxes)
sonarDistances = getSonarDistances()
feedback = generateFeedback(distancesList,sonarDistances)

Figure 15: Sequence Diagram- Collision Avoidance

7.6 Class Diagram

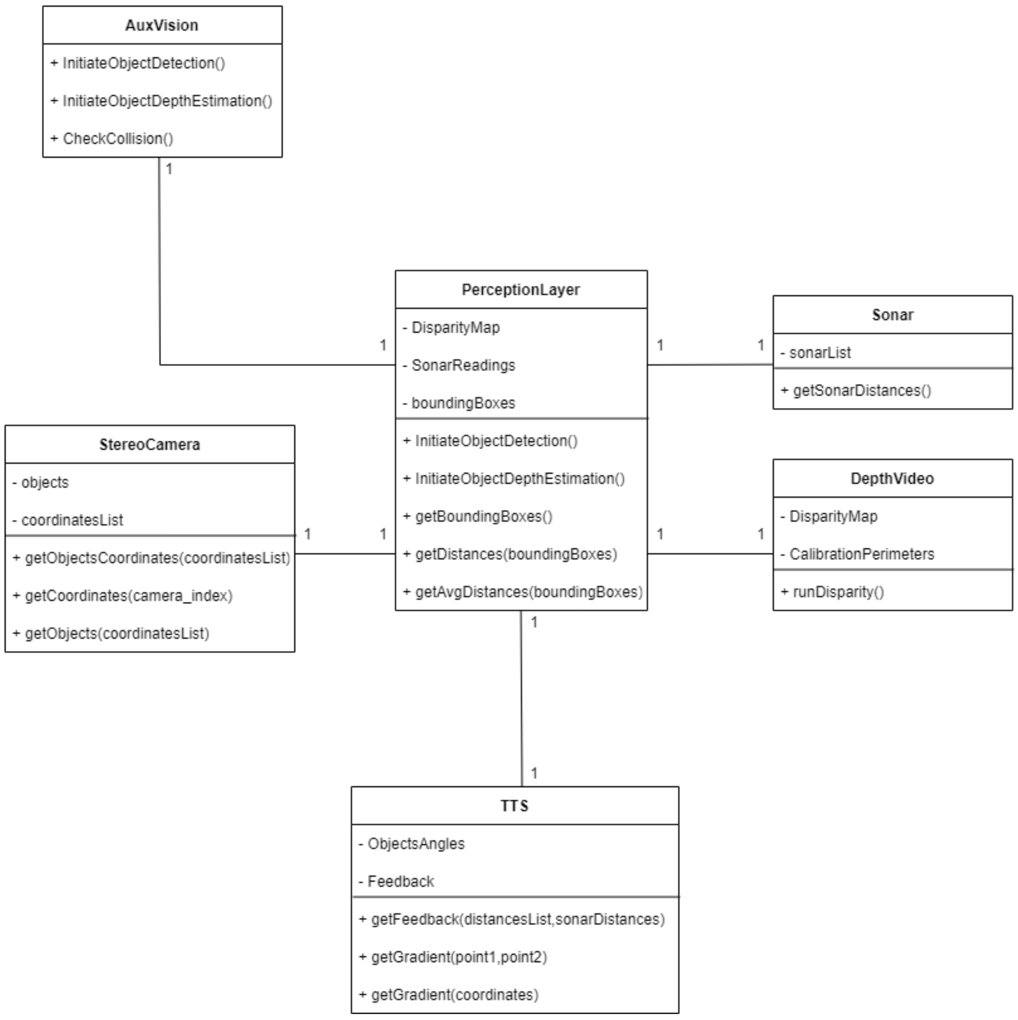

AuxVision
+ InitiateObjectDetection()
+ InitiateObjectDepthEstimation()
+ CheckCollision()
PerceptionLayer
- DisparityMap
- SonarReadings
- boundingBoxes
+ InitiateObjectDetection()
+ InitiateObjectDepthEstimation()
+ getBoundingBoxes()
+ getDistances(boundingBoxes)
+ getAvgDistances(boundingBoxes)
Sonar
- sonarList
+ getSonarDistances()
StereoCamera
- objects
- coordinatesList
+ getObjectsCoordinates(coordinatesList)
+ getCoordinates(camera_index)
+ getObjects(coordinatesList)
DepthVideo
- DisparityMap
- CalibrationPerimeters
+ runDisparity()
TTS
- ObjectsAngles
- Feedback
+ getFeedback(distancesList,sonarDistances)
+ getGradient(point1,point2)
+ getGradient(coordinates)
1

Figure 16: Class Diagram

7.7 Architecture Diagram

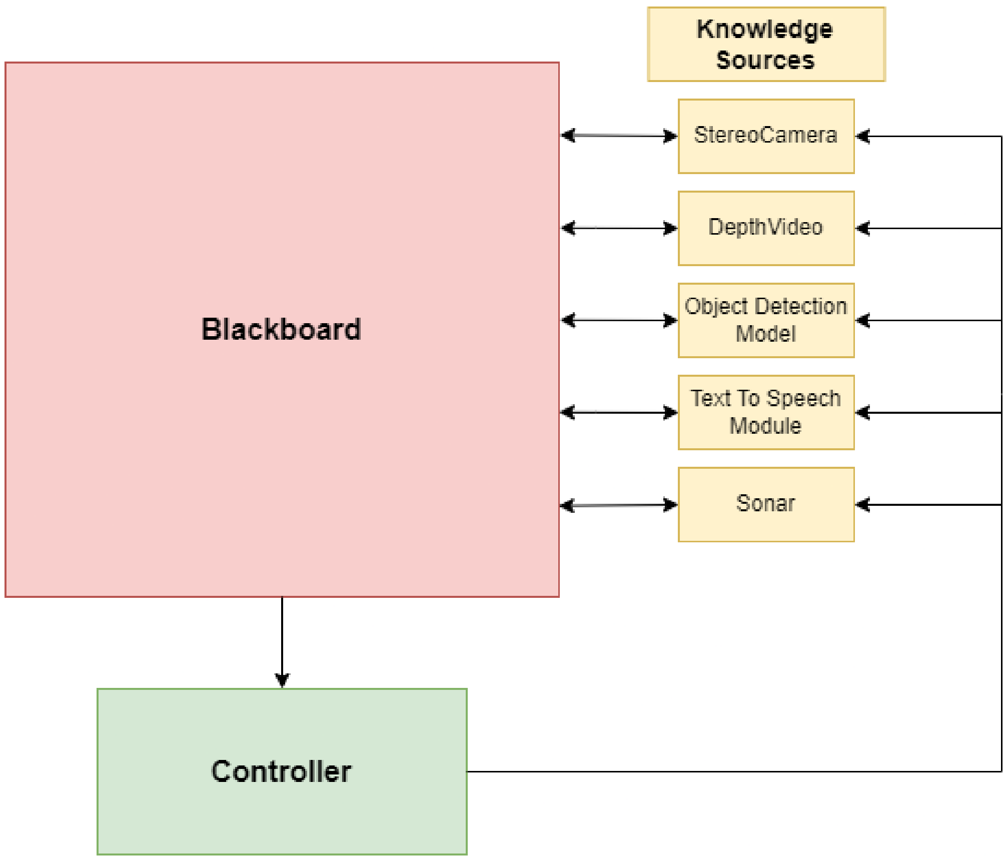
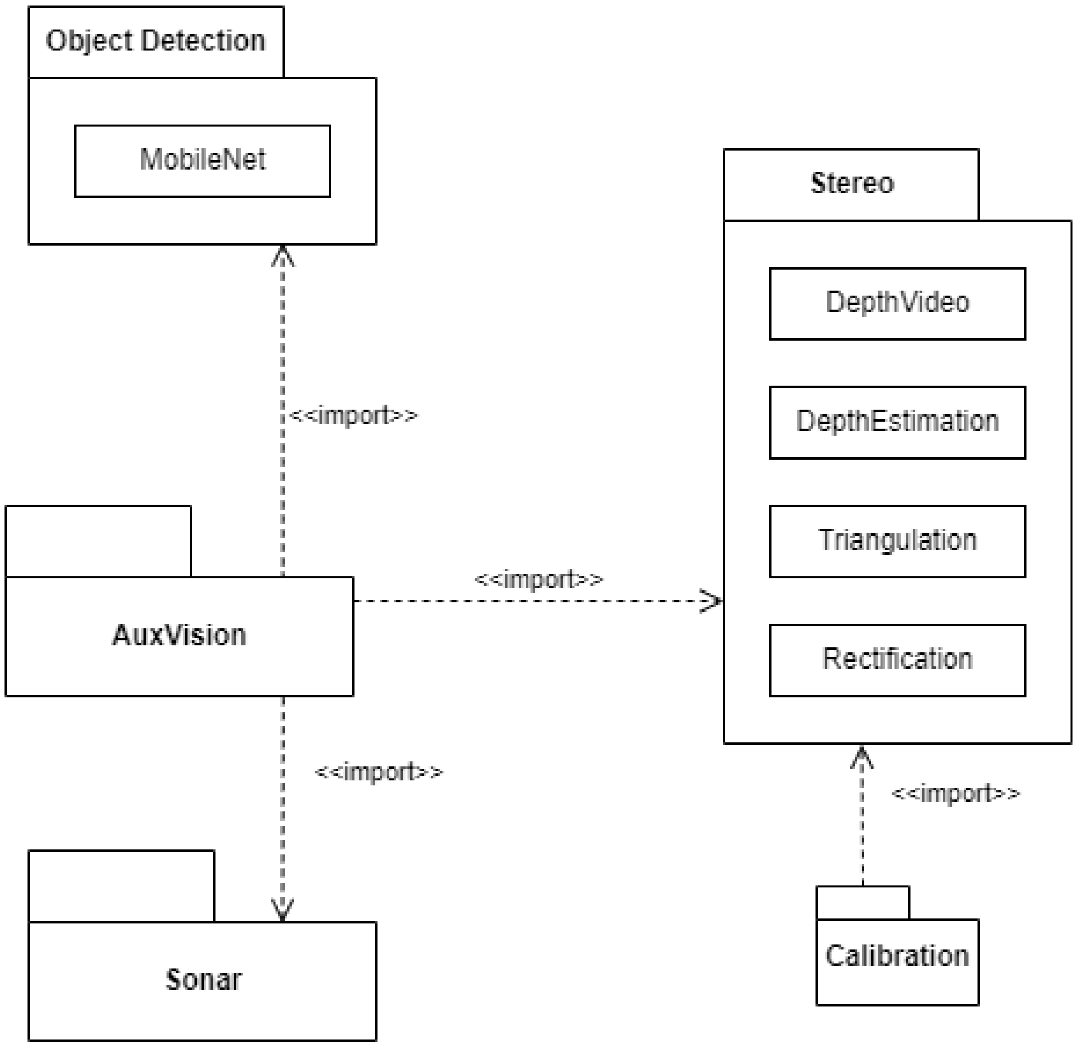
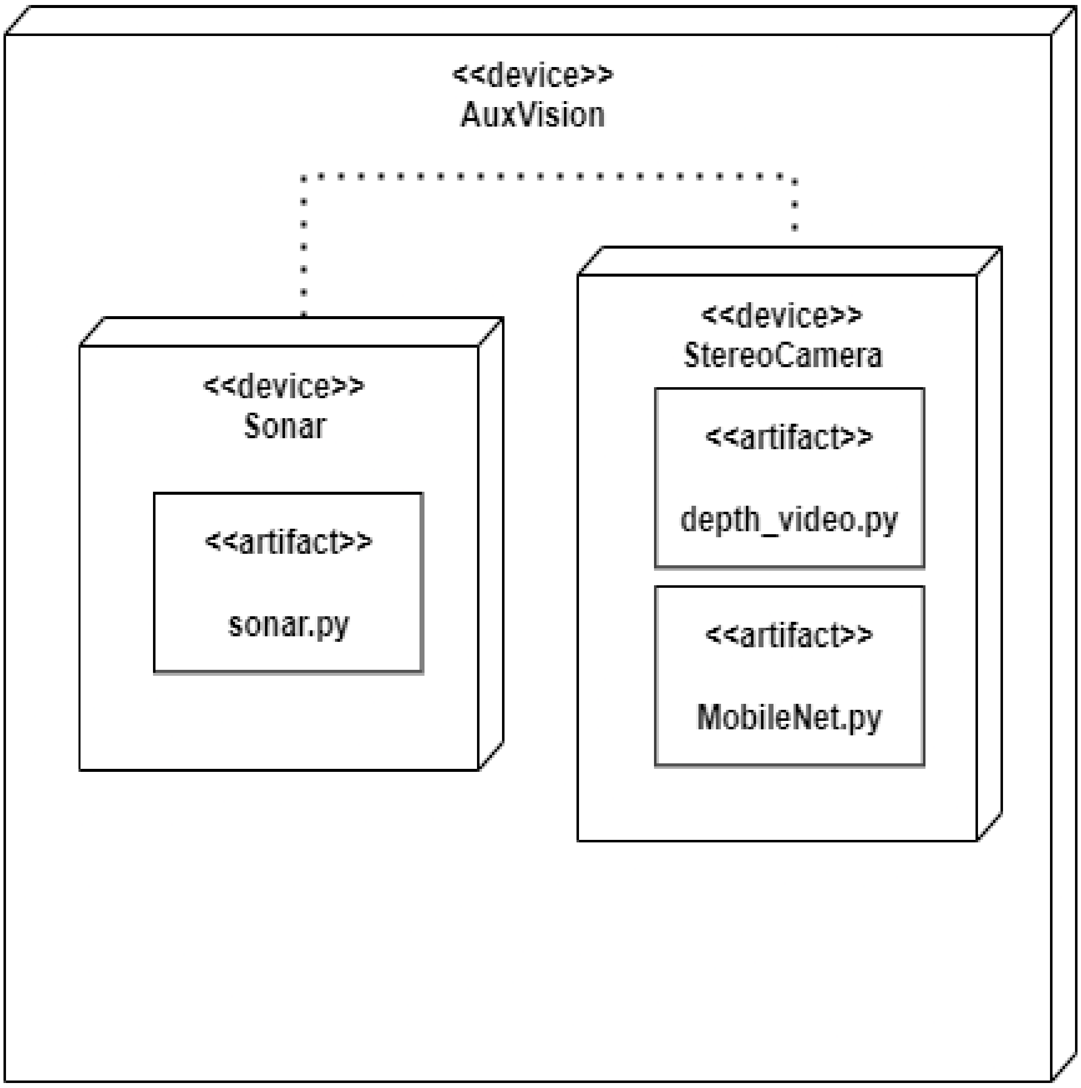


Figure 17: Architecture Diagram

8. Package Diagram



9. Deployment Diagram



10. Implementation Details

Arduino:

We have used an Arduino Nano V3.0 to take distance measurements from our SONAR sensors in the following setup below:

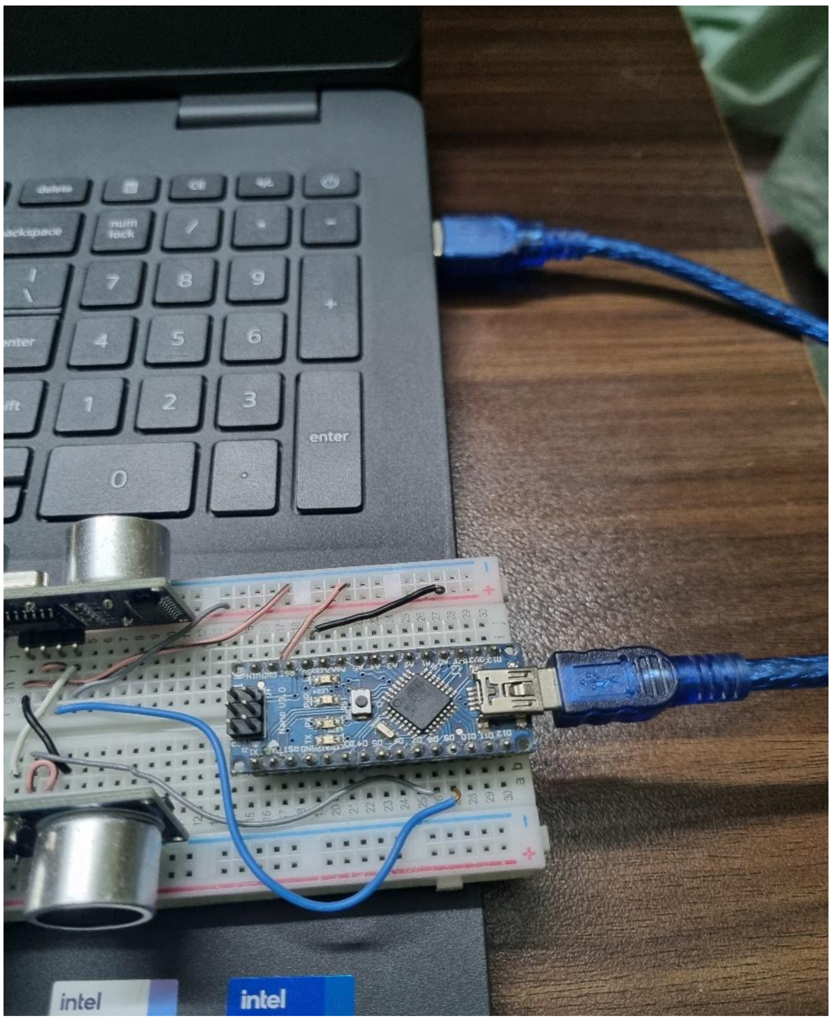
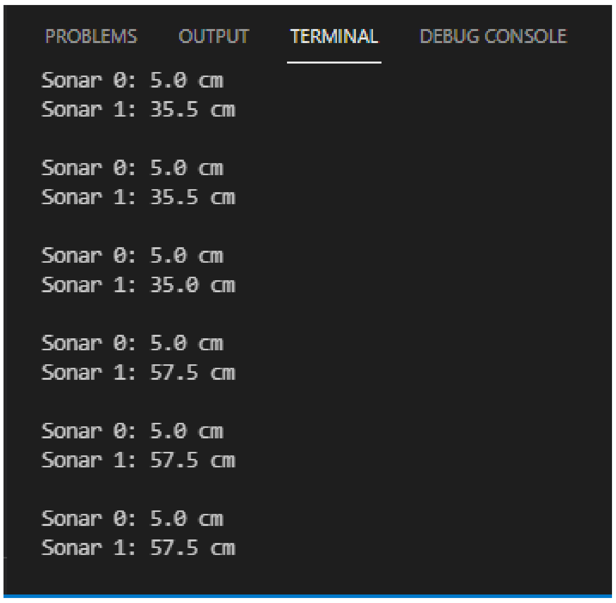


Figure 18: Arduino Nano and SONAR setup

SONAR Sensor:

The SONAR sensor is a key part of our perception layer. The primary purpose of incorporating the HC-04 SONAR sensor is to prevent collision with objects that are in close proximity to the user. For this iteration, we have connected two SONAR sensors to our Arduino and take distance measurements from them at a regular fixed interval.



```
Sonar 0: 5.0 cm
Sonar 1: 35.5 cm

Sonar 0: 5.0 cm
Sonar 1: 35.5 cm

Sonar 0: 5.0 cm
Sonar 1: 35.0 cm

Sonar 0: 5.0 cm
Sonar 1: 57.5 cm

Sonar 0: 5.0 cm
Sonar 1: 57.5 cm

Sonar 0: 5.0 cm
Sonar 1: 57.5 cm
```

Figure 19: Distance measurements from SONAR

The setup with Arduino connected is shown below:

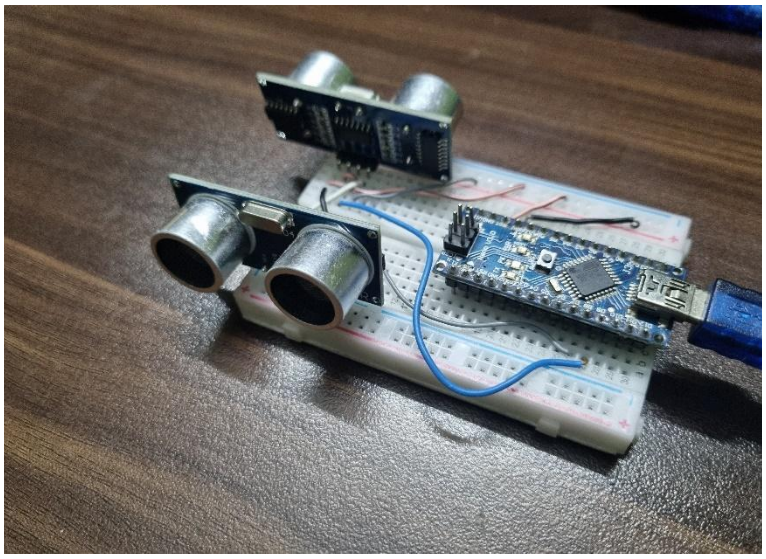


Figure 20: Arduino Nano and SONAR setup

Stereo Camera:

Camera Calibration

In order for the two cameras used to detect depth, it is essential that they know the relative position of each other both in the real world as well as compared to each other. For this we use camera calibration techniques implemented in OpenCV that uses a chessboard and detects their corners in different orientations to make precise guesses of the camera's locations. We obtain different forms of matrices that relate to the intrinsic and extrinsic parameters of the cameras.

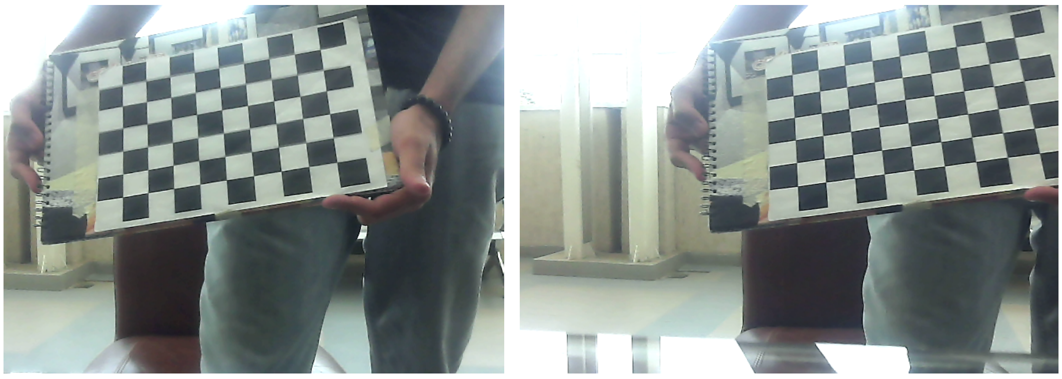


Figure 21: Chess Boards used for calibration of Stereo Cameras

Stereo Rectification:

After we obtain the calibration measurements and make sure the errors in their readings are low enough for the setup to work properly, we move on to the stereo rectification step. In order for the cameras to work together and get accurate depth information from the environment we need to make sure the y-axis and the z-axis of the two cameras are aligned perfectly, and any sort of lens distortion is removed from the input of the camera. This is important because Stereo-matching algorithms are designed to work on a single axis not only to provide better results but also to lighten the computation needed. For this we remap our image inputs to align, using OpenCV.

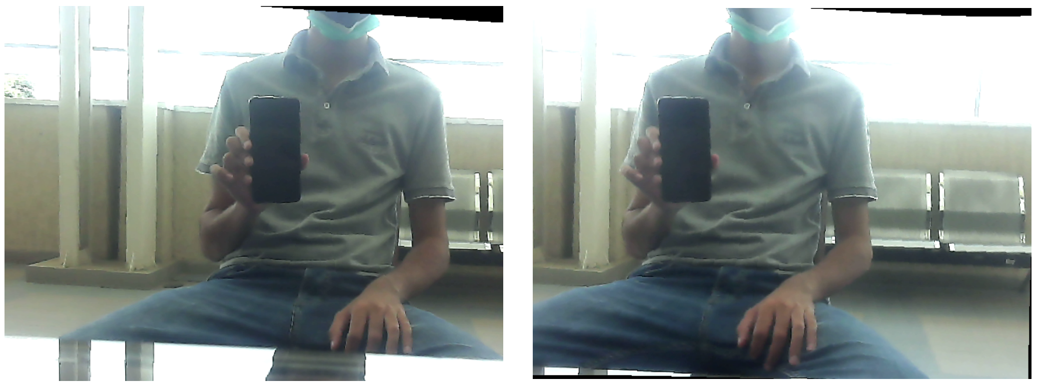


Figure 22: Rectifying images after calibration

Disparity Calculation:

After we're done with the rectification, we need to get the disparity in the images obtained from the two cameras. The disparity is key to finding the depth and we use the stereoSGBM module from OpenCV which is based on the modified H. Hirschmuller algorithm.

Once we get the accurate disparities, we use principles of Triangulation to calculate the depth of the scene captured from our cameras.

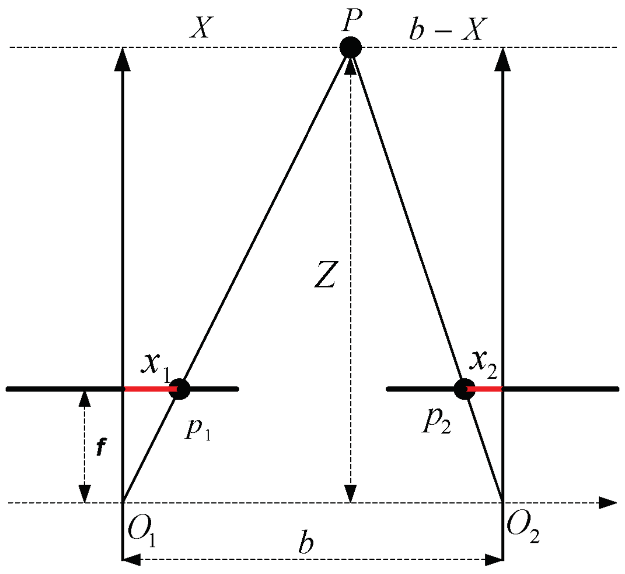


Figure 23: Triangulation principle

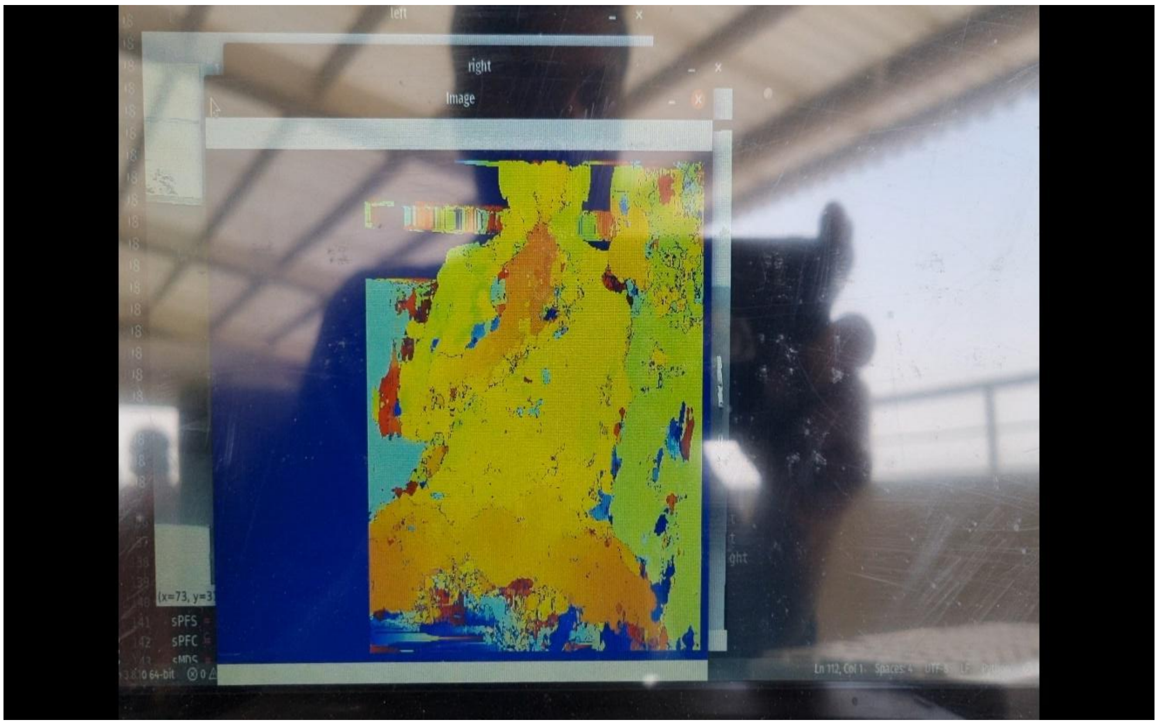
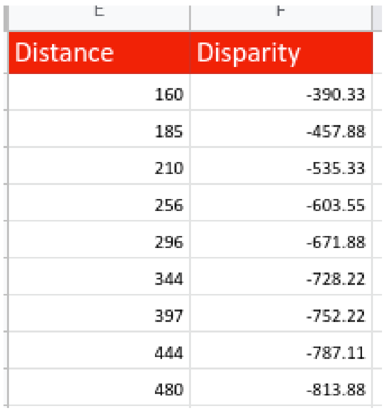


Figure 24: Disparity Map

Distance Estimation:

Once done calculating the accurate disparity values we move onto the distance estimation. We map real world distances against the disparity values our system provides, with increasing distances every step, this returns us a polynomial equation. The equation then helps us find the distances of the real-world objects our system identifies.



Distance	Disparity
160	-390.33
185	-457.88
210	-535.33
256	-603.55
296	-671.88
344	-728.22
397	-752.22
444	-787.11
480	-813.88

Figure 25: Distance - Disparity table

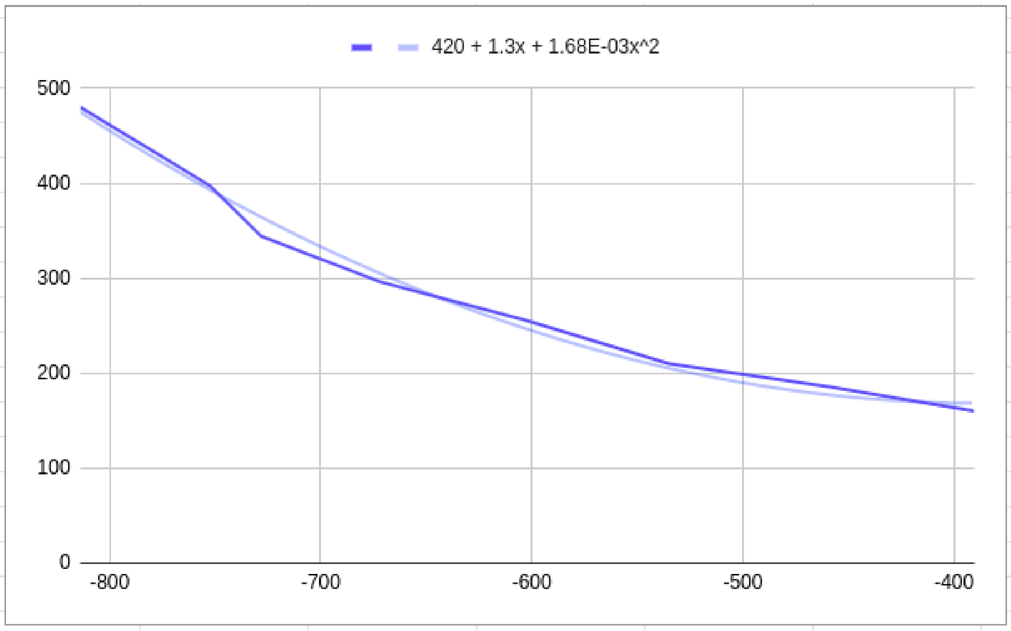


Figure 26: Distance equation

11. References

Ackland, P., Resnikoff, S., & Bourne, R. (2017). World blindness and visual impairment: despite many successes, the problem is growing. *Community Eye Health*, *30*(100), 71. /pmc/articles/PMC5820628/