Midterm Phase 0 - Proposal —— Perception - Human Obstacle Detection

1 Group Members

- Vishnuu Appaya Dhanabalan Phase 1 : Navigator — Phase 2: Driver
- Vasista Ayyagari Phase 1: Driver — Phase 2: Navigator

2 Product Details and Final Deliverables

The ability of a robot to understand the environment is vital for autonomy. Our product is a Human Detection and Tracking software. The ability to detect and track humans opens up multiple opportunities for robots to imbibe reactive behaviours that can potentially lead to better Human-Robot Interaction, safety protocols, efficient path planning etc.

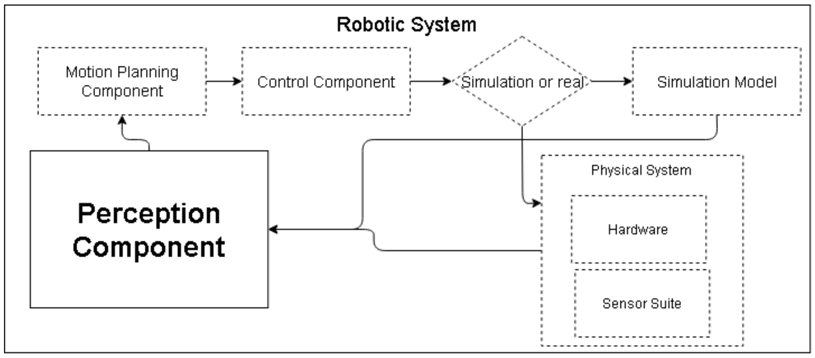

Figure 1: Robotic System with the Perception Component Highlighted

2.1 Deliverables

The final source file with code that will read raw images and compute a bounding box around each human in the image will be delivered. This can be used as API for another module or as a standalone module. As a standalone module, a visualisation will be provided to see the bounding boxes around said humans. We will also provide transformed detected human's coordinates in the robot frame either as a text output or output it on the screen. It will also be available using an API call and the latest 3D location is given. As an API, it can be used as a service in a service-client relationship of a larger robotic system.

3 System Design and Development Process

3.1 Modules

IO Handler, Pre-processing , Detection Module, Tracker Module, Depth Estimation Module, Post-processing Module, Visualisation Module, Coordinate Transformation Module.

3.2 Architecture and Data flow

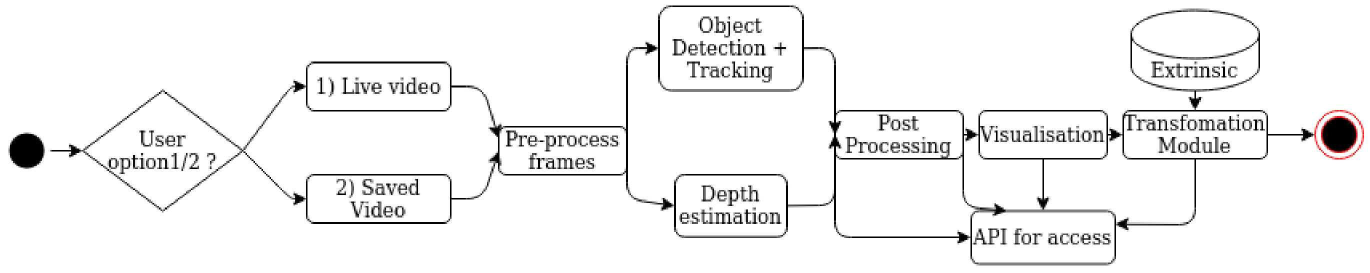

Figure 2: Activity Diagram

3.3 Tools and Processes Used

Language: C++11, Build System: CMake and Make, Cppcheck, Cpplint, CTest, Doxygen, Travis, Coveralls, OpenCV, Gtest, Gmock, Git, C++, Eclipse/VScode, Agile Iterative process, Pair Programming, Pytorch, TensorFlow.

3.4 Development Process

- Create UML diagrams for all classes and activities
- Follow Agile Iterative Process and Test Driven Development
- Develop Unit tests for each component
- Actualize each module mentioned in the previous section
- Develop an system pipeline that integrates all the modules together

4 Algorithms and Implementation Details

The following are potential options for the detection and tracking module

- **YOLO v4(detection) + Kalman Filter(tracking) + depth estimation using reference height of a person**. YOLO is a deep learning architecture which will take in the image as input and output a set of bounding boxes. The Kalman filter would help us keep track of objects across several frames. The depth estimation is planned to be done by assuming a person's height and comparing it with the number of pixels in our detection. Thereby we get the 3d Position of the person.
- **DeepSort(detection + tracking) + MonoDepth(depth estimation)**. DeepSort is a deep learning based method for object detection and tracking. It uses Yolo's architecture internally as well. Mono-depth is another unsupervised Deep Learning technique for estimating a depth map of entire scene. Using the bounding box detected, and the depth map, we can estimate the depth of object and thereby 3d position of the person.

5 Risks and Mitigation Strategies

This software uses Deep Learning tools. Hence, the output is an approximate not exact. We mitigate this by defining clear test cases and margin of error for these Deep Learning Algorithms.