Midterm Proposal – Phase 0

Lowell Lobo (UID: 120095719)
Mayank Deshpande (UID: 120387333)
Kautilya Chappidi (UID: 120380204)

1 - Overview

1) Purpose

Perception is essential for object detection, environmental awareness, path planning and control; through perception, a system can truly be considered autonomous. Acme requires the development of a perception component in its autonomous car system, and a human obstacle detection and tracking module needs to be built.

2) Objective

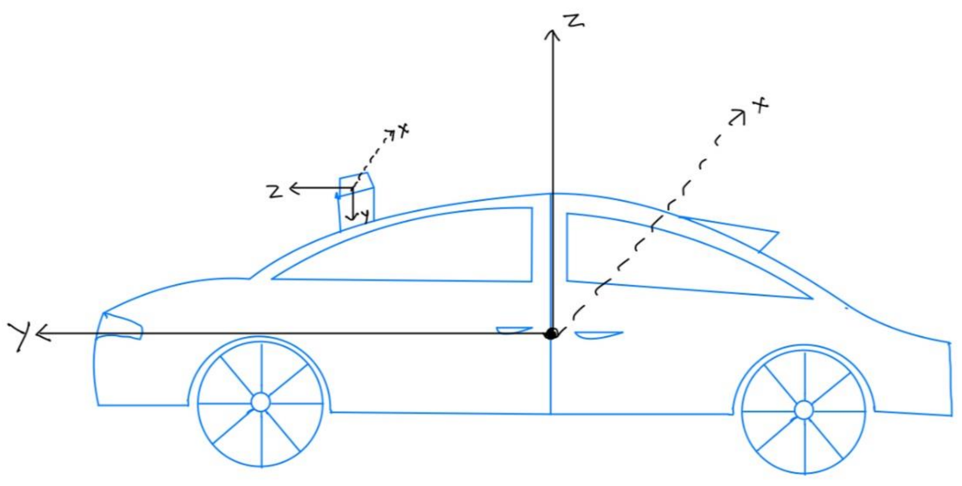
The aim is to build a perception module to detect human objects and track their location directly into the robot reference frame.

3) Scope

The system consists of a camera mounted on a car that will keep track of human motion within the camera's field of view, but the system will be unable to handle occlusion cases.

4) Assumption

The camera used for perception will be placed on the top of the car. The camera reference frame is taken such that the z-axis is pointing out of the lens, the x-axis is to the right, and the y-axis is pointing downward. The robot reference frame is at the centre of mass where the z-axis points upward, and the x-axis is to the right, and the y-axis points straight ahead. We assume the camera is monocular, and the user must input most of the configuration information.



5) Constraints

The module uses a monocular camera and is unable to handle occlusions. Also, the monocular camera cannot perform precise depth calculation and thus would require a deep learning model for accurate estimation. The runtime fps and memory management depend on the physical constraints of the system.

6) Deliverables

The end product will be a C++ library/API that enables human object detection and tracking, without considering occlusion, that provides obstacle location based on the robot reference frame.
UML diagrams for the implementation will be provided, along with developer-level documentation and a comprehensive report.
The module can be accessed via a GitHub Repository.

7) Evolution

The module can be enhanced to work with any system that requires perception components. Also, the module can be developed to use better cameras for precise depth and location calculation.

8) Summary

The project will be completed within two weeks following the AIP model and pair programming. The project uses a monocular camera to read a video stream and detect and track any humans in the frame.

2 - Reference Materials

Reference material consists of mainly C++, CMake, OpenCV and Eigen documentation.
Documentation for the project will be present as either Doxygen documentation or GitHub README and will be present in the GitHub Repository.

3 - Process

The project development will be executed using pair programming concepts. After a specific time frame, driver and navigator roles will be swapped. The project will follow AIP concepts and be performed using Pair Programming. The module works such that a video is fed into the system using the monocular camera. The program then converts the video stream into image frames and performs human obstacle detection. Once obstacles are found in the image frame, unique IDs are assigned to the objects, which will hold the calculated current location with respect to the robot reference frame. The process is repeated over time for tracking of obstacle movement.
Testing of components is performed using GoogleTest, and system testing will be performed every iteration for overall functionality verification.

4 - Organization

Pair Programmers and Design Keeper
Driver, Navigator and Design Keeper roles will switch daily, and Lowell will initially be the driver, Mayank, the navigator and Kautilya, the design keeper.

5 - Technologies

The project will be built in C++ using CMake build tools running on Ubuntu 22.04. OpenCV, an open-source licensed Apache2 library, and Eigen, an open-source licensed MPL2 library, will be used in the implementation. OpenCV's FaceRecogizer algorithm will used for human obstacle localization, and GitHub will be used for version control, code coverage reports and running unit tests.

6 - Management

The project will take two weeks to complete over two iterations, consisting of Phase-1 and Phase-2, where each iteration is for one week. Since the project consists of only two members, only product backlog, daily meetings and iteration meetings in the AIP model will be performed. Project monitoring will be done using timely git commits, and the navigator ensures code quality. Daily meetings will be held for error correction based on code coverage and GitHub CI reports.