Mid-Term Proposal

Software Development for Robotics

October 12, 2022

Students:
Shailesh Pranav Rajendran
UID: 118261997

Harika Pendli
UID: 117501421

Semester:
Fall 2022

Course code:
ENPM808X

1 Overview

Acme Robotics Inc. is a private company set to launch a 4-wheeled delivery robot used to deliver packages for an undisclosed multinational company next year. This robot 'Tubby' is set to debut early next year delivering packages inside the office. Scout moves in the corridor at a walking pace. The package is stored inside of the robot and drives itself to the customer. They have given us complete ownership of designing and developing this detector and tracker. Hence in this project, we propose a "human detection and tracking module" to be added to Tubby's perception software stack.

2 Design and Development Process

We will be making use of ESC methodology for the initial design process which is by extraction of significant concepts. We have identified the classes of the future program and their responsibilities. The class relations have been explained through UML class diagrams below.

Our software team will work on this project through iterative software evolution and service processes. Agile Development Process will be used in the development process with Test-Driven Development. Being a team of two programmers, we have decided to adopt pair programming and switch roles as and when necessary.

3 Process Model

3.1 Class Diagram

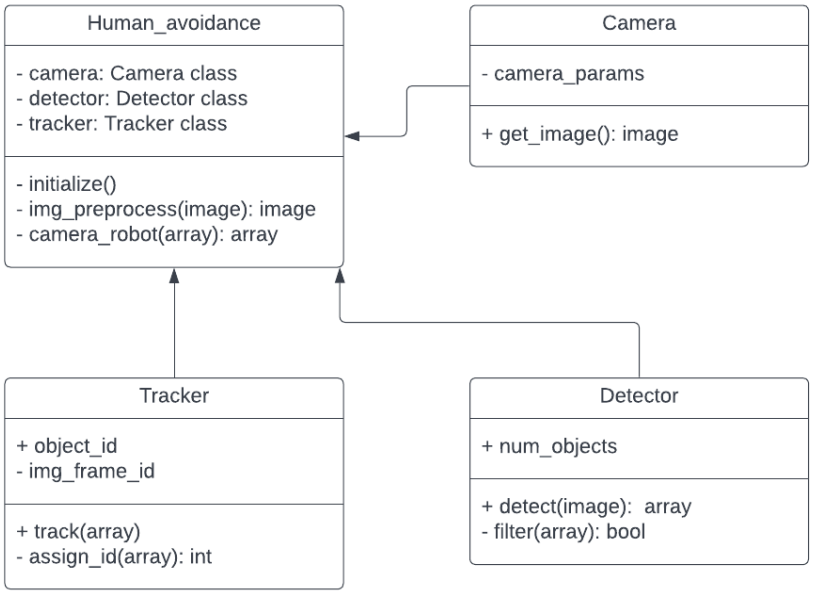

Figure 1: Proposed Class Diagram

3.2 Activity Diagram

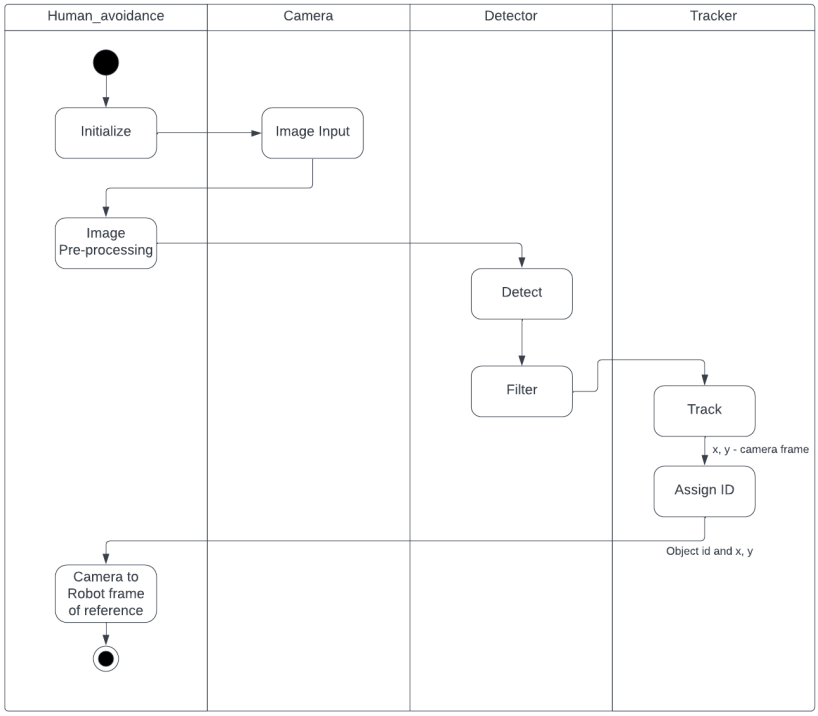

Figure 2: Activity Diagram

4 Algorithm and methodology

First, the class Human avoidance is initialized, which in turn initializes, the other three classes namely, the Camera class, the Detector class and the Tracker class. This gets the input image from the camera class which then undergoes a pre-processing stage. This is followed by the image being sent to a detection function of the detector class [1].

In the detector class, the frame is analyzed for the presence of humans. Then this information is passed through a filter for detecting false positives, if any. The updated coordinates along with the frame is then sent to the tracker function of the Tracker class [2].

Here the detected objects are assigned unique ids. This id along with the frame id as well as the coordinate of the object in the camera frame is saved in an array. This array is passed back to the Human avoidance class for the conversion of camera frame to robot frame. This process is repeated until all the image frames are completed.

5 Tools and Technologies

Our team will be using C++ 17 with Microsoft Visual studio code IDE configured with clangd, cppcheck, valgrind and cpplint. This scalable and self-contained API will be developed using modern software practices along with an Object Oriented approach.

Developer-level software documentation will be maintained using Doxygen. Git will be used to maintain all versions of software changes. All builds will be verified using TravisCI and Coveralls

with respective unit test cases using the Google Test framework.

Other dependencies and libraries used for development are:

- Ubuntu 20.04(LTS)
- CMake
- OpenCV
- Travis CI
- Coveralls
- Git
- Valgrind

6 Assumptions

- The lighting conditions are good.
- Images are already rectified, and there is no lens distortions.
- No occlusion's of objects are taken into account

7 Potential Risks and mitigation

- Missed detection: It may happen that the model sometimes misses the detection of human presence. In such a case, deploying more than one and different models can be more useful.
- False and duplicate detection: Can be reduced by training the model such that the parameters are well-tuned and the loss on the validation data set is minimum.
- Flickers in detection: Filter can be deployed for a better estimate.

8 Final Deliverables

- A module in C++ to detect and track humans using feed from a monocular camera.
- Overview of proposed work including timeline, risks, mitigations and UML diagrams.
- GitHub repository with README.
- Continuous integration and code coverage with TravisCI and Coveralls.
- Memory leaks check and profiling using Valgrind.
- Developer-level documentation.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. 2020.

[2] Addie Ira Borja Parico and Tofael Ahamed. Real time pear fruit detection and counting using yolov4 models and deep sort. *Sensors*, 21(14):4803, 2021.