Human Obstacle Detector : Software Management Plan

Abhishek Nalawade
University of Maryland
College Park, USA
abhi1793@umd.edu
117515589

Aditya Jadhav
University of Maryland
College Park, USA
amjadhav@umd.edu
117389616

I. Introduction

Obstacle detection has been widely studied in the field of Computer Vision. Obstacle detection is a crucial part of Self Driving Cars as well as Industrial Autonomous Systems. In the case of Self Driving Cars, a sub-task of Obstacle Detection can be Detection of Humans who become obvious and high risk obstacles for the vehicles. The real-time Human Obstacle Detection becomes increasingly difficult in noisy and chaotic environments and hence the algorithms deployed need to be highly efficient and robust to flimsy environmental variables. For example, color based approaches lack efficiency in unstable lighting environments. Also, Deep Learning based approaches offer high efficiency but are extremely expensive to implement.

- **Overview:** This project attempts to implement a Perception module for ACME Robotics using high-quality software engineering practices such as the Agile Iterative Process, Object Oriented Programming, Pair Programming and Test Driven Development. We will attempt to achieve robust human detection using Histogram of Oriented Gradients (HOG) feature descriptor combined with a Support Vector Machine Model.

The algorithm uses HOG features extracted from the input data to detect Humans and surround them with rectangular bounding boxes, the centers of which act as the pixel coordinates of the detected humans. These coordinates combined with the pose matrix of the homogeneous transformation between the monocular camera and the 3D world will be used to track the detected humans. The pixel coordinates will be used for determining the distance of the humans with respect to the Robot's Reference Frame. Support Vector Machine classifier will be used to train the extracted HOG features to train the training data.

- **Project Deliverables:** The algorithm should output the location of the humans with respect to the robot's reference frame. The input to the system can be a live camera feed or a prerecorded video stream.

- **Reference Materials:** Ubuntu 18.04 Operating System, C++ Programming Language, CMake Build System, OpenCV Third Party Library, Reference Papers 1 , 2 , 3.

- **Definitions and Acronyms:** AIP – Agile Development Process, TDD – Test Driven Development. HOG – Histogram of Oriented Gradients. SVM – Support Vector Machine. YOLO – You Only Look Once. PDF – Portable Document Format. OpenCV – Open Source Computer Vision Library.

II. Project Organization

- **Project Model:** In this project, we will be following the Test Driven Software Development model. This is an iterative process which involves creating unit tests first and then refactoring the code which helps the team practice optimized programming. TDD approach aids the Agile Development Process which heavily relies on feedback. In TDD, we will create precise tests for the proposed features or their stubs. To make the tests pass at each stage we will write code for these features. Once the tests pass, we try finding redundancies to optimize the code.

- **Organizational Structure:** We follow the Agile Iterative Process for this project which is a iterative development practice that breaks down large requirements into small tasks and uses feedback to deliver a consistent, stable and efficient product. We have a team of two programmers with the skills needed to deliver the product. We assign and switch between the driver and navigator roles though the entire AIP process.

- **Organizational Boundaries and Interfaces:** The project is initially designed to run on Ubuntu 18.04 operating system with support for the C++ programming and the OpenCV library. The algorithm will be designed to detect maximum two humans in an input frame.

- **Project Responsibilities:** As the project follows the Agile Iterative Process, we have designed our work be done in phases. Additionally we will be completing the work in these phases in two Sprints. We will be switching the roles of driver and navigator in these two sprints as required. We will also perform tasks simultaneously if the need arises.

III. Managerial Process

- **Management Objectives and Priorities:** As defined in the sections before, the Objective of the project will be to locate humans in the given input frames. And since we are following the AIP and TDD practices, our priority will be developing optimized code by writing efficient tests. The product backlog resonates with these practices initially containing unit tests for stub implementation and then improving those tests for testing the implementation code.

- **Assumptions, dependencies and constraints:** An important assumption is that the project will be run on a system which boasts similar dependencies as the ones used for this project. Another assumption for the project to work well would be a noise-free and clean input. The inputs for the HOG feature extractor or the SVM classifier would need to be images of specific dimensions. The SVM model would need to be provided with positive and negative set of images.

- **Risk Management:** This will be the team's first time implementing a human detection algorithm. And although we have modest (separate) experience with HOG and training of models, this would be our first integration of the two. In the events of a delay due to incorrect time estimates or any other unforeseen circumstance, the team would try to implement the YOLO algorithm in the remaining time period.

- **Monitoring and Controlling Mechanisms:** Git version control system will be used for this project to monitor and save the progress. A GitHub repository will be the place which will see all the timely phase changes of the project. Writing precise unit tests which do not contain redundancies will be a high priority to make sure the project gets delivered on time.

IV. Technical Process

- **Methods, tools and techniques:** We will be using static code analysis tools such as Cpplint and Cppcheck to conform to the practices of clean development. The project will have the unit tests written in accordance of the Google Test unit testing library. Data class can be tested if it outputs frames in the correct format and if the images provided to the model are in the specified dimensions. Train class can be tested if the weights and labels are generated in the required return type.

- **Software Documentation:** The documentation generator Doxygen is used to produce documentation of the project source code. The Class and Activity diagrams will be in PDF format. Agile process specific documents are in .ods format similar to .xls format. Other documents are a mix of Linux and Windows compatible format.

- **Project Support Functions:** We are using the Computer Vision third party library OpenCV which provides functions for HOG features, bounding boxes, pre-processing filters, etc. We will also use OpenCV SVM functions such as SVM::train and SVM::predict.

V. Work Packages

- **Work Packages and Dependencies:** We require OpenCV 3.0 or above versions in order to work with OpneCV SVM functions. We will need to work on Ubuntu 18.04 to avoid any dependency issues.

- **Resource Requirements:** We will need at least a quad core processor due to the data heavy functionalities of SVM and OpenCV. Testing can be a heavy operation if there are thousands of cases in a test suite. On top of that if a live feed is provided as input, then the processor needs to be able to handle the processing efficiently to provide desired outputs.

- **Schedule:** The Agile Process documentation contains Product Backlog and Work Log which provide estimation about the development schedule and resource allocation and consumption.