Human Detector & Tracker – Proposal

Introduction

We propose to develop a robust human obstacle detector and tracker using a monocular camera, directly usable in a robot's reference frame. The deliverables will include but not be limited to, a new module for detection & tracking in C++ using high-quality software engineering practices, class and activity diagrams, an up-to-date GitHub repository with complete documentation and unit tests integrated with Travis CI and Coveralls.

Project Organization

Product development shall be done using the Agile Iterative Process where tasks will be tracked using a backlog table. All the tasks will be outlined in the form of Epics, User Stories & Tasks in Microsoft Azure DevOps. Team members shall follow the pair programming method for development, where the roles of navigator & driver will be interchanged every sprint.

The release model will follow a rapid prototype development process and the testing model to be used will be unit testing where several unit tests will be designed for complete code coverage.

Managerial Process

The aim of the management is to create a quality deliverable with no bugs. Risk mitigation for the human detector will be performed by testing it on curated data from the COCO dataset. For the tracker, a centroidal Euclidean drift technique will be used, for which a fallback plan of running the tracker every N number of frames, is in place. Additionally, a fallback plan of using a Viola Jones face detector and dilating the output box in order to hopefully capture the human, is in place.

Technical Process

The project will be developed using C++. Additionally, for image operations OpenCV will be used. A YOLOv3 neural network trained on the COCO dataset will be used as a human detector and a KCF tracker will be used to track the detected objects.

The camera is considered to be at a fixed position on the robot. A transformation matrix ${}^{R}T_{C}$ relates the body frame of the robot to the camera frame [1]. The X coordinates of the object to the camera will be calculated by using an area of bounding box to distance constant. This constant will be calculated during calibration of the camera during which the pixel to meter conversion factor is also noted. The bounding box outputs from the YOLOv3 model will be converted to SI units by using the constant of calibration. An instance of a KCF multi-object tracker will be initialized with the bounding box outputs from the model. Each subsequent frame will then be passed to the KCF tracker which will generate bounding boxes for all the detected objects. In each frame, the centroids of the tracker boxes will be compared with the centroids

of the original detection to calculate the drift. If the drift is beyond a certain threshold, then the detection model will be run again and the tracker will be re-initialized with the new bounding box co-ordinates. The resulting detections will be multiplied with the ${}^{C}T_{R}$ to output co-ordinates in the robot's body frame.

Complete documentation will be maintained for each component, with proper code commenting. Doxygen will be used to generate relevant documentation and cpplint and cppcheck will be used for style sheet adherence and static code analysis.

Current Backlog Table

ID	Work Item Type	Title	State	Effort (person hours)
1	Task	Select a pretrained human detection model	New	3
2	Task	Select a multi-object tracker for tracking	New	2
3	Task	Define Robot frame of reference	New	2
4	Task	Define transformation matrix from robot base to camera frame of reference	New	2
5	Task	Script for distance estimation from bounding box area - camera calibration	New	5
6	Task	Script for estimating pixel dimensions to SI units - camera calibration	New	4
7	Task	Shell script for downloading model files & weights	New	1
8	Task	Create Robot class and define constructors and methods	New	5
9	Task	Implement constructors & DetectHuman method in Robot class	New	20
10	Task	Implement helper methods in Robot class	New	6
11	Task	Create Human Detector class and define constructors and methods	New	3
12	Task	Implements constructors of Human Detector Class	New	1
13	Task	Implement Detect method of Human Detector class	New	4
14	Task	Implement LoadNet method of Human Detector class	New	2
15	Task	Create Tracker class and define constructors and methods	New	2
16	Task	Implement method for tracker initialization of Tracker class	New	1
17	Task	Implement method for tracking in Tracker class	New	3
18	Task	Create script to download subset of COCO dataset with only human labels for testing	New	4
19	Test Case	Test transform method in Robot class to see if it returns the correct transformed co-ordinates	Design	3
20	Test Case	Test CalcCentroidDistance to see if it returns an accurate Euclidean distance	Design	2
21	Test Case	Test Detect method of HumanDetector class	Design	3
22	Test Case	Test LoadNet method from class HumanDetector	Design	1
23	Test Case	Test track function of Tracker class	Design	3
24	Test Case	Test tracker initialization method of Tracker class	Design	2
			Total	**84**

Appendix

References

[1] Ma, Lu & Ghafarianzadeh, Mahsa & Coleman, Dave & Correll, Nikolaus & Sibley, Gabe. (2014). Simultaneous Localization, Mapping, and Manipulation for Unsupervised Object Discovery. Proceedings - IEEE International Conference on Robotics and Automation. 2015. 10.1109/ICRA.2015.7139365.