TX9 Mobile Robot Perception Module

Pratik Acharya
117513615

Sumedh Koppula
117386066

Introduction

Acme Robotics is building a next generation mobile robot to be used on sports fields to detect and track players and provide analytics data about them to be utilized for broadcasting. A swarm of robots will be deployed on the field.

The mobile robot will have a monocular camera placed on it to receive live video feed of the field. A perception module on the robot will receive the video feed and will label each player and store their location with respect to the robot. The output of the module will be the labels and location of the players. This will be fed to a path planner, so that the mobile robot can track a player to ensure that the player is in its field of vision.

We plan to use Machine Learning, specifically Convolution Neural Network Algorithms to detect, classify and track objects from a video stream. In order to perform deep neural computation we are using open source computer vision libraries such as openCV and YOLO with C++. For computation we assume that Nvidia Jetson Nano 2GB chipset for deep learning computation is used. The targets are assumed to be 180cm in height, which will be used to calculate distance from the robot in the forward direction. Assuming MX219-160 Camera with 3280 × 2464 Resolution, 8 Megapixels and 160° FOV for video feed.

Assumptions

- For deep learning computation we assume that Nvidia Jetson Nano 2GB chipset is used as a processing unit.
- The targets are assumed to be 180cm in height, which will be used to calculate distance from the robot in the forward direction
- Assuming MX219-160 Camera with 3280 × 2464 Resolution, 8 Megapixels and 160° FOV for video feed.

Technologies

- Programing language: C++
- Build system: cmake
- Testing Library: Google Test, Google Mock
- Continuous Integration: Travis CI, Coverall

Algorithms

- YOLOv4 under YOLO LICENSE
- OpenCV version 4.2, under Apache 2 License

YOLO Functionality

The YOLO algorithm uses convolutional neural networks (CNN) to detect multi objects in real-time. This algorithm uses a single forward

propagation through a neural network to detect objects in a single run. Here the role of CNN is to simultaneously predict various class probabilities and bounding boxes. Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images

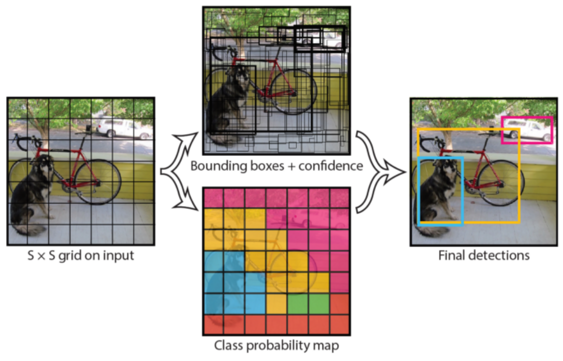

Figure: input video being processed by YOLO vision algorithms outputting boxed video as image detection. [Source : https://blog.netcetera.com/object-detection-and-tracking-in-2020-f10fb6ff9af3]

OpenCV Functionality

An open-source library for computer vision, machine learning, and image processing. Some of the header files which we would like to access in our project are below:

```
#include <opencv2/dnn.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/opencv.hpp>
```

Design and Development Process

We are following Agile iterative process methodologies for high quality product delivery with a burndown chart(AIP chart link). The programming is completed using the Test Driven Development process and pair programming will be utilized. The driver and navigator for the process will be switched after completion of each task mentioned in the burndown chart starting with Sumedh R Koppula as the Driver and Pratik Acharya as Navigator for the first task.

Below is the brief description about the Activity diagram and UML.

Activity Diagram Description (Activity Diagram link)

The live stream of the video data path from the robot's monocular camera is fed to a C++ parser to interface it with openCv libraries. Using OpenCV we capture each Frame of the video and convert it into a 4D Blob. This blob is fed to a convolutional neural network to detect, classify and track the object. Human detection and tracking is done through bounding box regression and a probabilistic estimation of confidence score. If the confidence score is less than the threshold we are eliminating the detected box. Similarly if the confidence score matches the expectation, we are displaying the detected object.

UML (UML Diagram Link)

We are using 5 classes for human detection namely:

1. **HumanDetectionAndTracking**
2. **StreamParser**
3. **YOLOConfig**
4. **Human**
5. **Utils**

The class diagram is explained via an activity diagram and its brief description.

Risk and Mitigations

Risk 1: The created software fails to detect humans.
Solution: OpenCv will be used to detect humans based on the jersey color and the dimensions of the bounding box. Utilizing the dimensions of the bounding box will ensure that false positives are minimalized. The average height assumed will be modified to include only the size of an average human torso.

Risk 2: The software fails to track the detected humans
Solution: The final product will be delivered only for human detection and the tracking update will be provided in a future release.

Testing

Google's Open Images Dataset V6+ will be used for testing and quality assurance. The dataset contains labelled images of humans with the information about bounding boxes provided in it.

For real time testing, a laptop and webcam will be used and the output will be monitored to ensure proper functioning.

Technical Paper referred to

Traffic-Net: 3D Traffic Monitoring Using a Single Camera

Final Deliverables and Timeline

- Perception module which outputs detected labels of humans and its location from robots frame of reference.
- Quality Assurance of the module with report.
- **Timeline:** The estimated time to deliver perception module is within 16 days i.e on or before 22nd Oct 2021