Autonomous Driving

Miladin Momčilović
Faculty of Technical Sciences, Novi Sad

Problem and motivation

Concept of autonomous cars were always fascinating to me, so this was opportunity to see how it works in a nutshell.

My main goal was to build system for autonomous driving. Initial idea was to implement whole system that would be able to work on it’s own on both simulation and RPi car.

I divided image processing into two modules:

- Traffic sign detection
- Lane detection

ROS/Gazebo simulator

Gazebo is an open-source 3D robotics simulator. It is a powerful physics-based simulator that provides a realistic environment for simulating robot behaviors and interactions.

ROS is an open-source robotics middleware suite. It is not operating system but rather a flexible set of tools, libraries and conventions that facilitate the development of robot applications.

Gazebo and ROS can be integrated to facilitate communication and control between the simulated robot and external software components. This integration is achieved through the use of ROS plugins, which allow for data exchange and interaction with the simulated environment.

Data collection

I acquired data for training from BFMC-Sign-detection dataset from Kaggle. Data in this dataset contains images from Gazebo simulation and from real life camera input on polygon.
Input data for predictions is gathered from camera located on front of the car. It is processed in both modules separately depending on usage.

Brain module

At the core of this autonomous driving system, the brain module serves as the intelligent decision-making center. It integrates data from a multitude of sensors, including cameras, GPS and traffic light. It combines those inputs and make real-time decisions for vehicle control.

This module constantly follows GPS location, to be able to follow predefined route and make decision on intersections and parking in the right spot.

It is subscribed to traffic sign and lane detection modules and makes decision according to those information, whether it should keep moving or stop or turn.

Initial idea was that this module use agent such as imitation learning or reinforcement learning, but in the end it is rule-based, with switch case approach.

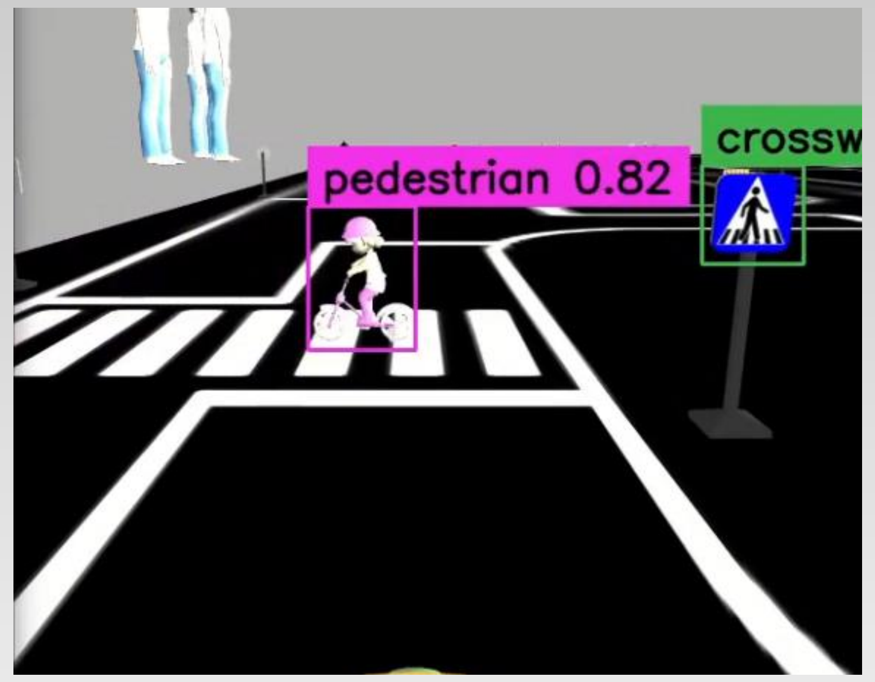
Traffic sign detection module

On of the most complex problems in autonomous driving is following road rules and for that it needs sign detection, so it can determine what to do on turns and crossroads.

The idea behind this module is to predict in real time and with high confidence. That was accomplished using the YOLOv8 the latest version of the real-time object detection model, fine-tuned for our image input. It offers information about type of sign and location of it.

Model achieved a 72.5 mAP-50 on average for all objects. The model was trained on around 5000 images.

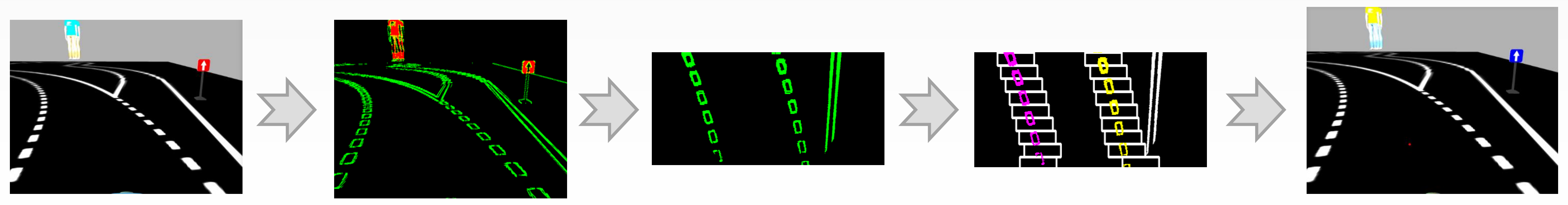
I tried few different formats in the end PyTorch gave the best prediction time. But overall the was around 50-100ms for prediction with some delay in execution because of ROS node communication between two modules.



Lane detection module

Vehicle position and orientation on the road is an important information for driving. As I used only camera input, I had to come up with a solution that separates lane path from other objects on image.

To ensure precise lane-keeping and safe navigation, I implemented sophisticated lane detection module using computer vision techniques, primarily leveraging OpenCV. This module analyzes real-time camera input, detection and tracking lane markings to determine the vehicle’s position within the lane.



Results

In the end results are pretty good. Model performed well in simulation. It succeeded in completing most of the tasks, like taking a turn, stopping at traffic light and stop sign, when pedestrian walks on crosswalk, overtaking the car, parking etc.

However there are a lot of thins that needs to be improved to make it full autonomous.

It need smarter brain module, that does not contains hardcoded path, but more like path some sort of path following with given tasks.

It needs better accuracy with sign detection, although this model was trained on 6 core/12 threads CPU, with only 7 epochs. And better dataset is needed.

It needs better parameter adaption for real life camera input, this is only optimized for simulation testing.

And whole thing needs to be optimized to be able to work on the weaker hardware, like Raspberry Pi, because this code had some problems with processing data and delay in execution.

One of potential solution could be to change sign detection from YOLO to OpenCV detection.

References

- ROS (https://www.ros.org/)
- Gazebo (https://gazebosim.org/home)
- YOLOv8 (https://github.com/ultralytics/ultralytics)
- Lane detection (https://www.hackster.io/kemfic/curved-lane-detection-34f771)
- Dataset (https://www.kaggle.com/code/arpitvaghela9210/sign-detection-for-bosch-future-mobility-challenge)
- BFMC (https://boschfuturemobility.com/)