

## Modeling and trajectory control of a virtual autonomous vehicle on ROS-Gazebo softwares

Tibério D. A. R. Ferreira (IC), Olmer García Bedoya (PG), Janito Vaqueiro Ferreira (PG).

### Abstract

This research project aims to introduce new functionality to a GUI which serves as a control center and development platform for UNICAMP's Intelligent Vehicle of the Laboratory of Autonomous Mobility (VILMA) car. It focuses mainly on recording, treating and following trajectories. Additionally, since the GUI is supposed to serve as a platform to other project it has also been converted into a modular design.

*Key words: GUI, QT, Autonomous Vehicle.*

### Introduction

The focus of this project is to improve the development platform created using the QT<sup>1</sup> framework used in UNICAMP's autonomous car project VILMA. In a previous project a 3D car model was created with physical properties close to the real life model. The model was also integrated into a 3D physics engine which allowed it to have dynamic behavior like a car. A GUI to control the vehicle and test algorithms was also developed. This project aims to expand the GUI functionality and usability.

In order to do so, many autonomous functions for the car were added, many sensors implemented and the simulation was ported to a new engine (from Gazebo to MORSE<sup>2</sup>). The GUI was also rewritten in a new modular design in order to make maintenance easier and also provide other researchers a way to reuse the code.

The added functions were: record the car's trajectory, plot the recorded trajectory and the current car's trajectory, smooth the trajectory (get rid of hard turns), keep current speed, get to a desired speed, generate more trajectory points (by linear interpolation).

### Results and Discussion

The final GUI design can be viewed in Figure 1. All the objectives were achieved. The GUI was adapted with buttons to trigger the desired events. The project had to be ported to a new simulator engine (from Gazebo to MORSE) in order to obtain new functionalities (such as having a suspension model and a slip model) and facilitate joint working with partner universities such as USP. The GUI had to be adapted to the new simulator. In addition, the code used in the GUI was also modularized in classes. This way other projects can easily build on top of the work

developed here by importing only the necessary c++ classes and using the api defined by them without needing to understand the internal implementation of them. For example, if one wants to receive car information from the simulator it suffices to import the class which encapsulates the communication with the simulator and utilize its API.

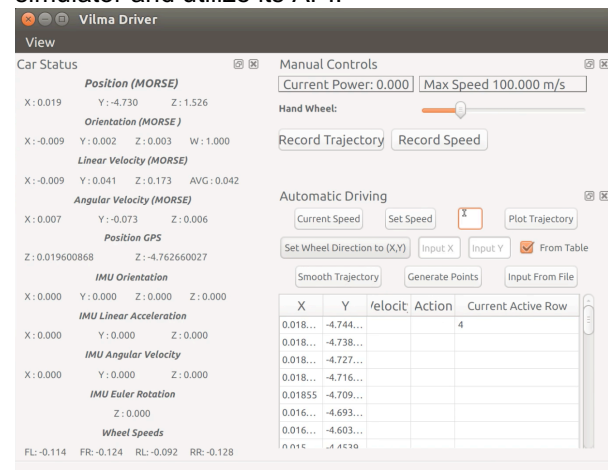


Figure 1 – Final GUI design

### Conclusions

All the objectives were completed and even extended with the modularization of the code into classes and the adaptation of the model and GUI to the new simulator.

### Acknowledgement

The work developed here wouldn't be possible without the collaboration of USP and access to their CaRINA simulation code in which most of the work here described is based on.

<sup>1</sup> Digia. QT Project @ONLINE. url: <http://qt-project.org>.

<sup>2</sup> Laboratoire d'analyse et d'architecture des systèmes (LAAS-CNRS). Modular OpenRobots Simulation Engine @ONLINE. url: <https://www.openrobots.org/wiki/morse>.