HAW HAMBURG
Fakultät TI, Department Informatik
Intelligente Sensorik
Prof. Dr. Tim Tiedemann, Max De Muirier

Übungsblatt 4
zur Veranstaltung „AFR – Autonomes Fahren und Robotik”

4. Programmierung in Carla

In der Übung 4 geht es um die Programmierung und Auseinandersetzung des PythonAPIs in Kombination mit der ROS Bridge.

Zur Bearbeitung der Übungsaufgaben: Bearbeiten Sie auf jeden Fall *alle* Übungsaufgaben. Ausgenommen hiervon sind lediglich die mit „optional” gekennzeichneten Textstellen. Lesen Sie sich die Aufgaben gut durch. Sollten Sie eine Aufgabe nicht lösen können, so beschreiben Sie zumindest, wie weit Sie gekommen sind und auf welche Weise Sie vorgegangen sind. Aufgaben mit der Randbemerkung *“Protokoll!”* sind im Protokoll zu beantworten. Als Lösung der Aufgaben ist (je nach Aufgabe) Programmcode abzugeben (kommentiert) oder ein kurzer Text.

4.1. Vorbereitungsaufgaben

1. Setzen Sie sich mit dem Beispielcode aus dem letztem Praktikum auseinander. Wie genau wird hier die Umgebung sowie das Fahrzeug erzeugt und Steuerungsbefehle generiert?

4.2. Praktikumsaufgaben

1. Bearbeiten Sie die ersten Schritte in **Getting started** (Anhang A).
2. Erläutern Sie, wie der Beispielcode *carla_ros_node* aufgebaut ist? Wie werden wird die aktuelle Fahrzeuggeschwindigkeit erhalten? Wann ist das Programm beendet? *Protokoll!*
3. Wie wird auf die einzelnen *topics* zugegriffen? Was wird dafür alles benöigt? *Protokoll!*
4. Welche Nachrichtentypen finden Anwendung? Und welche werden für eine Geschwindigkeitsregelung benötigt? *Protokoll!*
5. Wie lässt sich hier eine Regelung für die Fahrzeugsteuerung realisieren? Welche entscheidende Komponente ist dafür essentiell und macht den Unterschied zu einer einfachen Regelung? *Protokoll!*
6. Implementieren Sie eine einfache Fahrtregelung auf Basis eines PID-Reglers. Wie wird die Zeitschritt für den I- und D-Anteil festgelegt? *Protokoll!*
7. Setzen Sie die Zielgeschwindigkeit auf *100 km/h* mit den Verstärkungen für *P=1.8*, *I=0.5* und *D=0.5*. Schauen Sie sich nun den generierten Plot der Geschwindigkeiten. Welches Regelverhalten zeichnet sich hier ab? *Protokoll!*
8. Finden Sie nun passendere Werte für die Regelung. *Protokoll!*

9. Erweitern Sie das Programm um ein weiteres *topic*, dass *subscribed* wird. Geben Sie den Inhalt in einem festen Intervall aus. *Protokoll!*

A. Getting started

In this session we will work with the Carla ROS Bridge. All required files for this workshop can be found in the ROS package *carla_ros_node*. Just download them from **Teams** and move it into the *source* directory of your *catkin workspace*.

Install the **ROS** dependencies:

```
1 mkdir -p ~/carla_ws/src
2 cd ~/carla_ws/src
3 git clone https://github.com/demuma/carla_ros_node
4 python -m pip install -r ros-bridge/requirements.txt
5 cd ..
6 sudo apt-get update
7 rosdep update
8 rosdep install --from-paths src --ignore-src --rosdistro noetic -y
9 catkin_make
```

Let's start **Carla** and the **Carla ROS Node**.

```
10 /opt/carla-simulator/CarlaUE4.sh &
11 source ~/carla_ws/devel/setup.bash
12 roslaunch carla_ros_node start_carla_ros_node.launch
```

Our vehicle named *ego_vehicle* has spawned.

For the vehicle to drive, we first need to implement a **PID** controller. The basic Python script can be found in:

```
13 gedit ~/carla_ws/src/carla_ros_node/src/carla_ros_node.py
```

After implementing the PID controller, you can start the Python script with:

```
14 rosrun carla_ros_node carla_ros_node.py
```

A live plot will display the vehicle's current velocity over time.