Assignment 3 Perception for Self-Driving Cars

Due: 17.01.2024, 12:00AM

Introduction

Welcome to Assignment 3 of 'Techniques for Self-Driving Cars'! This file will lead you through the provided framework and present the four exercises.

Framework

For the assignments in this course, we provide you with a framework which manages the integration with the CARLA simulator and lets you focus on the implementation of a particular technique. The framework can be downloaded from eCampus as a `zip` file and has the following structure:

```
.
|-- ex3_planning.pdf
|-- params.yaml
|-- pyproject.toml
|-- setup.py
|-- scripts
|   |-- main_perception.py
+-- src
|   |-- pyilqr
|   |-- sdc_course
|       +-- control
|       |   |-- __init__.py
|       |   |-- controller.py
|       |   |-- decomposed.py
|       |   |-- pid.py
|       |   |-- stanley.py
|       +-- perception
|       |   |-- __init__.py
|       |   |-- perception.py
|       |   |-- utils.py
|       |   |-- traffic_sign_map.py
|       +-- planning
|       |   |-- __init__.py
|       |   |-- a_star.py
|       |   |-- behavior_planner.py
|       |   |-- global_route_planner_dao.py
|       |   |-- graph.py
|       |   |-- global_planner.py
|       |   |-- local_planner.py
|       +-- utils
|           |-- __init__.py
|           |-- car.py
|           |-- scenario1.yaml
|           |-- scenario2.yaml
|           |-- scenario3.yaml
|           |-- town03_map.png
|           |-- town03_signs.txt
|           |-- utility.py
|           |-- window.py
|           |-- world.py
+-- tests
    |-- test_perception.py
```

You can already see that it is similar to last assignments. We just added some files which are related to perception. As before we provide a solution to the previous tasks such that you can solve these assignments. You can of course also use your own implementation, just make sure to use our new `params.yaml` with some updated parameters for perception.

File you need to edit:

Files	Description
`perception/traffic_sign_map.py`	Representation of traffic signs and it category.
`perception/utils.py`	Main classes used by the perception module and utility methods for coordinate transformation.
`planning/global_planner.py`	Global planner that now must consider the traffic sign detection in the plan.
`scripts/main_perception.py`	Main loop that also triggers the GUI updates.

File you might have a look at:

Files	Description
`perception/perception.py`	Actual implementation of the simulated detector.
`params.yaml`	All the parameters for control and planning.

Please ensure that the tasks run with the provided scenario `scenario3.yaml`. We will verify your solution with this settings.

Attention!

You have to run the setup script again to get the updated files, i.e.,

~~~
```bash
$ cd assignment3
$ pip3 install -e .
```
~~~

Submission:

Once you have completed your assignment by editing the relevant scripts in the framework, upload your solution as a `zip` file via eCampus with the filename `LastName1_LastName2.zip`. We will check the files that you need to edit. Make sure that your solution is working, when we download a fresh framework and replace it with the abovementioned files.

Please make only one submission per team.

Academic Dishonesty:

We will check your code against other submissions in the class. We will easily know if you copy someone else's code and submit it with some minor changes. We trust that you will submit your own work only. Please don't let us down. Note that in the case of plagiarism, you will be expelled from the course. We are strict on that.

Task 1: Visualize Detections [25 Points]

Given the detections generated by the detector, it is a good idea to visualize the bounding boxes in the captured image and also output some infos about the detections.

Your task: Use the provided method `add_bounding_box` and `add_text` of the Window's pane to draw the detections and output some information (see the main loop in `scripts/main_perception.py`)

Task 2: Image to World Coordinates [25 Points]

Given the two-dimensional bounding boxes, we need to localize these in the three-dimensional world, which will be part of task 3. Only then we can ensure that we can associate information over time to the correct traffic signs. Here, we can exploit the depth image provided by the sensor to determine the world position from image coordinates.

Your task: Determine the correct depth from the given depth image in the `script/main_perception.py` and complete the method `image_to_world` in the file `perception/utils.py` that transforms a point in image coordinates into the corresponding world coordinates given the pixel coordinate and depth at the pixel.

Hint: Note that world coordinates are given in Unreal Engine's left-handed coordinate system, where x points forward, y points right, and z points upwards. The local coordinate system of the camera uses a right-handed coordinate system, where x points right, y points downwards, and z points forward.

Task 3: Traffic Sign Map [25 Points]

The world positions and the detected category (which is to some degree uncertain) can be now used to update our belief over the world. Here, we represent the locations of the traffic signs and the currently estimated category in the so-called traffic sign map.

Your task: Implement the method `update_map` of the TrafficSignMap (see `perception/traffic_sign_map.py`) that accumulates the detections of the traffic signs. See also the definition of `TrafficSign` in `perception/utils.py`.

Task 4: Updating the planner [25 Points]

This is the final task. In this task you will update the global planner, such that the car obeys the traffic signs and replans if some roads are forbidden. You need to modify the `integrate_traffic_sign` method of the `GlobalPlanner` in the `global_planner.py`.

Your task: Implement the method `integrate_traffic_sign` to integrate the speed limits for the current segment and the turn signs. A blue arrow means that you have to turn at the next junction according to the direction of the sign.