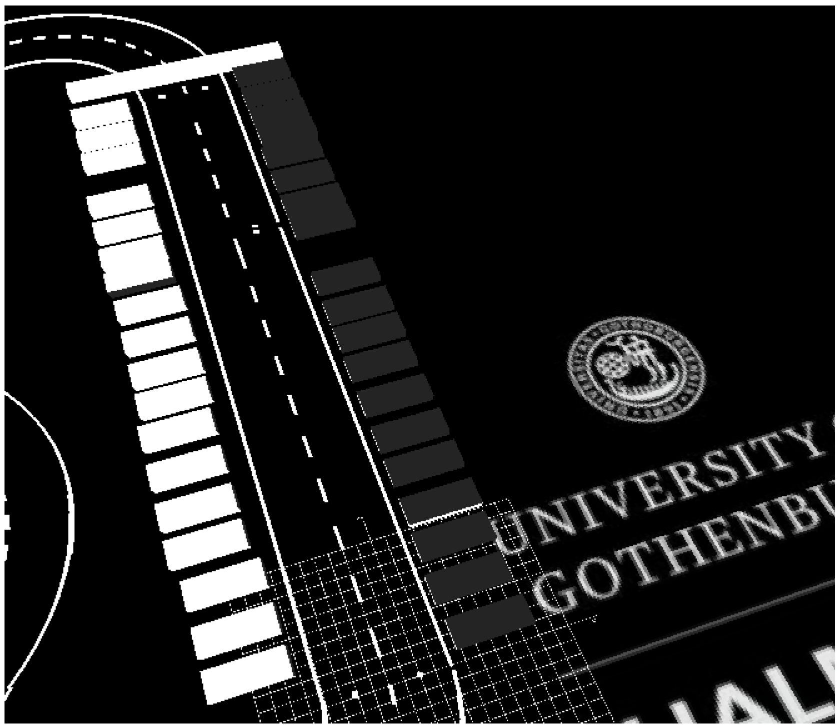
Revision history:

Today's vehicles are equipped with many embedded systems to provide comfort and safety functions to the driver, passengers, and other traffic participants. The development of these vehicle functions requires nowadays the combined knowledge and teamwork of different disciplines like computer science, electrical engineering, and mechanical engineering. Moreover, these functionalities shall be developed in tighter schedules with the same or even better quality for several vehicle families.
The purpose of this course is to familiarize the student with contemporary challenges and technologies for developing self-driving vehicles. But instead of working with real vehicles, this course is working with a simulation environment and miniature cars to allow the students to develop autonomous lane following behavior, safe handling of intersections, passing other vehicles, and to park vehicles on sideways.

This document defines the re-examination. It extends and refines the official CoursePM for this purpose.

The re-examination is organized as follows:

The student who is participating in the re-examination needs to realize an automated parking functionality for a parking situation in the simulation environment. Available parking spots can be available on both sides of the road and the goal is to pick a suitable parking spot automatically. In addition, the road can be blocked at the very end; the autonomous vehicle is then supposed to make a 180° turn (so-called U-turn or K-turn) to continue its search for a free parking spot on the other side of the road. An example scenario is depicted in the following figure:



Please note that there might be no free parking spot on the right hand side of the road but only on the left hand side; i.e., your algorithm must be able to reach out to the end of the road where the blockage is appearing, perform a turn, and continue to search on the other side of the road.

As a starting point, a basic running example for a parking algorithm is provided here: https://github.com/se-research/OpenDaVINCI/tree/master/automotive/miniature/boxparker. The student is advised to make himself/herself familiar with the provided example.

The goal for the re-examination is to reason, modify, and extend the basic example so that the algorithm can park in a parking lot. An example scenario for this task is available here: http://www.cse.chalmers.se/~bergerc/dit168/ParkingStreet.scnx. This file needs to be downloaded and referenced from the `configuration` file for odsupercomponent.

The student has access to all supportive documents here http://code.opendavinci.org and http://docs.opendavinci.org.

The simulation environment is provided pre-arranged in Docker images as described here: https://github.com/se-research/OpenDaVINCI/blob/master/automotive/miniature/boxparker/README.md. The student is advised to install the latest version of Docker (cf. https://docs.docker.com/engine/installation/ubuntulinux/) to make use of the tools. Alternatively, you can run your parking algorithm in simulation without Docker. A tutorial of the boxparker example without Docker can be found at the bottom of the tutorial page: http://opendavinci.readthedocs.io/en/latest/tutorials.html (in the section "Running the boxparker example in simulation"). Further background material about algorithms for self-driving vehicles is available here: http://arxiv.org/abs/1406.7768.

The student needs to use at least the following tools from the simulation:

- odsupercomponent – providing the configuration data
- odsimirus – providing measured distances from virtual infrared and ultrasonic devices
- odsimvehicle – simulation of vehicle movements

The following tool is useful during the development:

- odcockpit – visualization of the vehicle surroundings

It is suggested that the student implements the parking functionality in the component `BoxParker.cpp` based on the basic example, which is provided here: https://github.com/se-research/OpenDaVINCI/blob/master/automotive/miniature/boxparker/src/BoxParker.cpp. Therefore, the student needs to fork from the existing official GitHub repository to his/her own user-repository to document the changes that he/she is applying to the template code to realize the functionality (use the button fork on the GitHub website for that).

uled date/time for the oral examination will be communicated by the course responsible. You