User Guide

Editor: Julius Kokko Ekholm

Version 0.2

Status

Reviewed		
Approved		

PROJECT IDENTITY

2018/HT, Cascar Group
Linköping University, Dept. of Electrical Engineering (ISY)

Project Group Members

Name	**Responsibility**	**Phone**	**Email**
Andreas Lundgren	Project Leader (PL)	070-022 56 44	andlu901@student.liu.se
Julius Kokko Ekholm	Documentation (DOC)	070-304 21 42	julek017@student.liu.se
Adam Ahlgren	Software (SW)	070-796 83 71	adaah731@student.liu.se
Karl Gudmundson	Testing (TST)	070-578 84 82	kargu357@student.liu.se
Anton Kullberg	Software (SW)	073-959 97 71	antku588@student.liu.se
Jonathan Hanses	Design (DES)	076-260 77 47	jonha594@student.liu.se
Andreas Larsson	Information (INF)	076-815 58 03	andla514@student.liu.se
Sten Magnusson	Hardware (HW)	073-766 99 43	stema100@student.liu.se

Email list for the whole group: cascargroup@gmail.com
Website: *under development*

Customer: Lars Nielsen, lars.nielsen@liu.se
Customer contact: Erik Frisk, erik.frisk@liu.se
Course leader: Daniel Axehill, daniel.axehill@liu.se
Supervisor: Pavel Anistratov, pavel.anistratov@liu.se

Contents

Document history

Version	Date	Changes	Sign	Reviewed
0.2	2018-12-	Second draft. Minor adjustments.		
0.1	2018-12-07	First draft.		

List of Figures

List of Tables

1 Introduction

This user guide is a part of the project course TSRT10 CDIO Project (Automatic Control) at Linköping University, Sweden. Final year engineering students apply for a specific project related to automatic control and sensor informatics. In general, the projects of the course are closely linked to research in the fields or companies operating within these areas.

This user guide presents how the system described in the design [1] and requirement specification [2] is set up and used.

1.1 Definitions

Abbreviations and their definitions that are used throughout this document can be seen in Table 1.

Table 1: Abbreviations and definitions.

Abbreviation	Definition
ROS	Robot Operating System
RP	Raspberry Pi
VPS	Visionen Positioning System
SLAM	Simultaneous Localization and Mapping
GUI	Graphical User Interface
RC	Radio-Controlled
LIDAR	Light Detection and Ranging sensor
BDM	Behavioural Decision Making
MC	Mission Control
VP	Vehicle Perception
BDM	Behavioral Decision Making
MP	Motion Planning
VC&M	Vehicle Control & Modelling
SIM	Simulator

There are also some specific terms used throughout the document defined in Table 2.

Table 2: Terms and their definitions.

Term	Description
Route	A route is a way which is to be travelled. It can either be determined by the user manually or by using computational algorithms. It is defined by two or more predefined waypoints. Also, a route is determined by the coordinates (x,y).
Path	A path is a way which is to be travelled. It differs from route in the sense that it considers the movements constraints of the vehicles. Therefore, no sharp turn or similar unfeasible kinematic will be part of the path.
Trajectory	A trajectory is a calculated path which a body travels along e.g. a route. It is hence determined by a specific route. However, it differs because it includes the time as well, i.e. at what time point the body is at the coordinates (x,y). More explicitly, a trajectory is defined by (x(t),y(t)).
Agent	An entity capable of making it's own decisions interacting with the environment as well as other agents. In this specific case an agent is an autonomous vehicle.
Action	An action is something an agent does after it has made a decision. It is some kind of movement in the physical world, e.g. change lane, turn left.
Static object	Anything that does not move is considered a static object.
Static environment	An environment where every object, apart from the agent, is static.
Dynamic object	Anything that moves is considered a dynamic object.
Dynamic environment	An environment where at least one object, other than the agent, is dynamic.

2 Overview of the System

This section presents the general overview of the system.

2.1 System Architecture

The product consists of multiple autonomous vehicles that can handle complex scenarios in a road network using route planning, behavioural planning, motion planning, control theory and communication. Subsystems that have implemented these fields are

called: **Mission Control**, **Vehicle Perception**, **Behavioural Decision Making**, **Motion Planning**, **Vehicle Control & Modeling**, **Simulator** and **GUI**. The vehicles are able to handle these scenarios in a safe and efficient manner, making sure collisions are avoided at all costs. Most scenarios are commonly encountered in a standard road network, such as overtakes, roundabouts and intersections, but the vehicles have the ability to drive in a convoy, using communication to limit the distance between them. See Figure 1 for an overview of the general system architecture of one car.

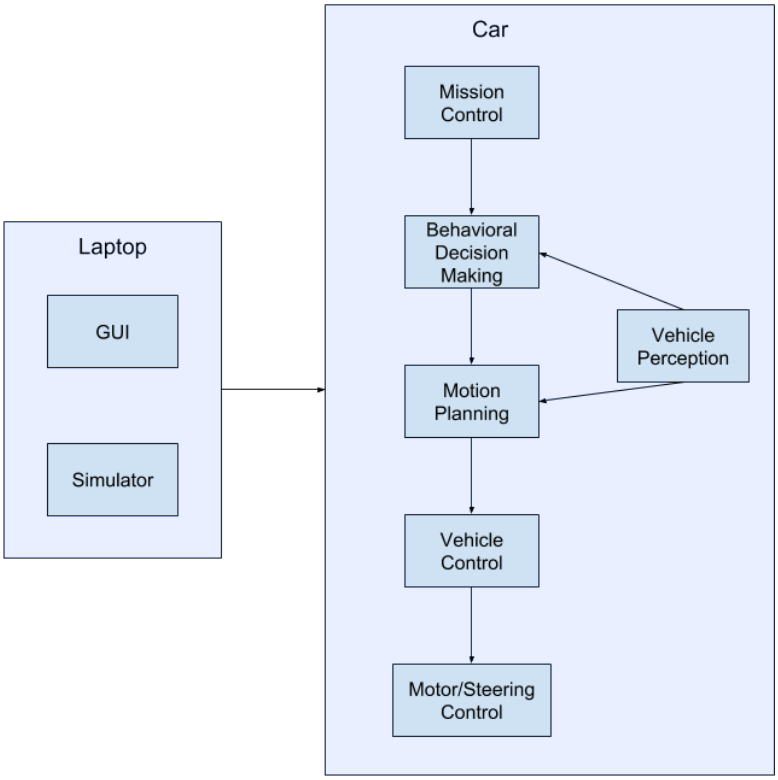

Figure 1: System architecture of one car.

2.2 Vehicle

The complete system consists of three cars with autonomous driving ability. Each car consists of a custom RC-car fitted with an RP, LIDAR, two odometers, an Arduino and a camera. Two of the cars are also equipped with IMUs connected to the RP. Figure 2 shows an overview of a single car and what hardware it consists of.

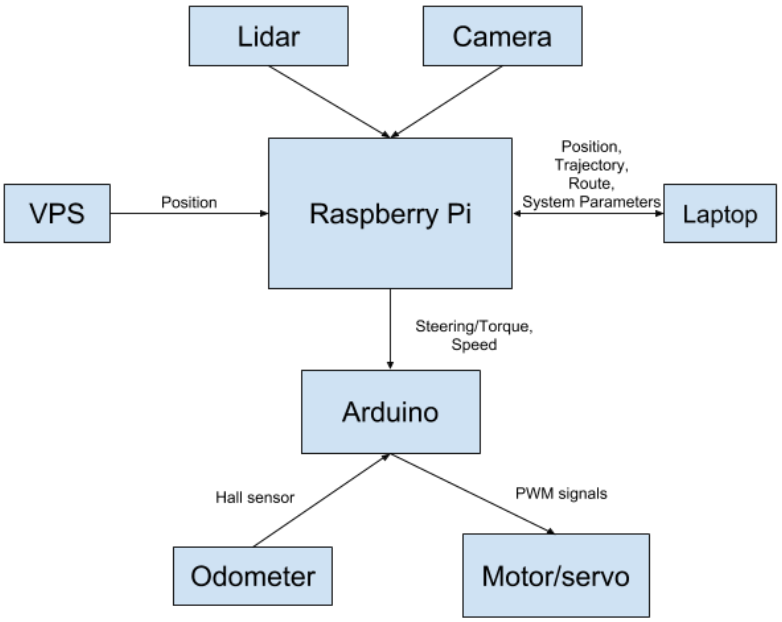

Figure 2: Overview of one separate vehicle and its communication.

The dimensions of each individual vehicle can be seen in Table 3.

Table 3: Dimensions of a single car

Dimension	Value (mm)
Length	415
Width	255
Height	142
Wheelbase	275

3 Operational Requirements

In this section, the operational requirements of the system are presented. In order to have sufficient operability conditions, these requirements must be fulfilled.

3.1 Software

In this subsection, the software requirements of the system are displayed.

3.1.1 Operating System

The operating system that has been used during development is Ubuntu 16.04. It is not guaranteed to work on any other system.

3.1.2 ROS

The different subsystems are implemented in ROS Lunar 1.13.7 [3]. The communication between the different described subsystems, see Figure 1, is broadcasted through ROS topics and services. The node sending a message publishes on a specific ROS topic and the receiving node subscribes on the same ROS topic.

Most of the subsystems and ROS nodes are run on the cars' Raspberry Pi except the GUI and simulator subsystems which run on a laptop. Each car runs its own set of ROS nodes for each subsystem. Thereafter, the cars communicate with each other through their own Vehicle Perception subsystem via ROS topics.

Forward compatibility with new ROS versions is not guaranteed but the Simulator has been tested on both ROS Melodic Morenia and Lunar.

3.1.3 Qualisys Motion Capture System

For positioning of the cars, Qualisys Motion Capture System is used. This capture system is installed at Linköping University and is here referred to as VPS, Visionen Positioning System. It makes use of twelve infrared cameras which detect reflective balls on physical objects to determine its position. These reflective balls are mounted on the cars in a specific pattern to make them distinguishable from one another.

3.2 Hardware

To be able to drive a specific scenario and have interaction between the cars, at least two cars equipped with the hardware shown in Figure 2 are required. Additionally, each car needs to have a charged 7.2 V battery to run properly.

3.2.1 Vehicle Range of Operation

The cars are able to complete different scenarios in a road network. Some limitations on the layout of the roads have been made since the cars have limited turning radius and a track width of about 25 cm. The roads are able to have a lane width of 50 cm and a minimum turning radius of 100 cm, see Figure 3.

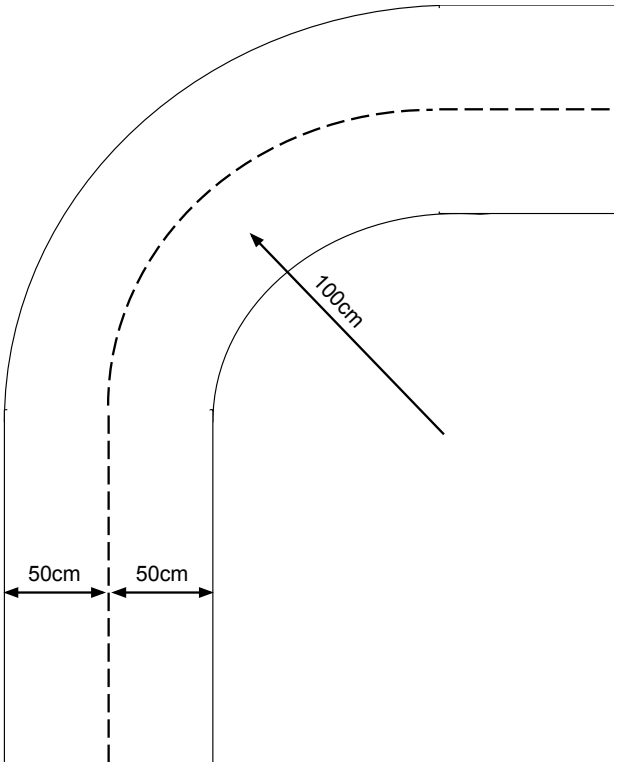

Figure 3: Road used in the project with minimum turning radius and lane width.

The cars can also drive in roundabouts. The roundabouts are limited to the same constraints as the rest of the road network: a lane width of 50 cm and a turning radius of 100 cm. A schematic overview of a roundabout can be seen in Figure 4. The lane width of the roundabout is set to be 50 cm, and the inner diameter to 250 cm.

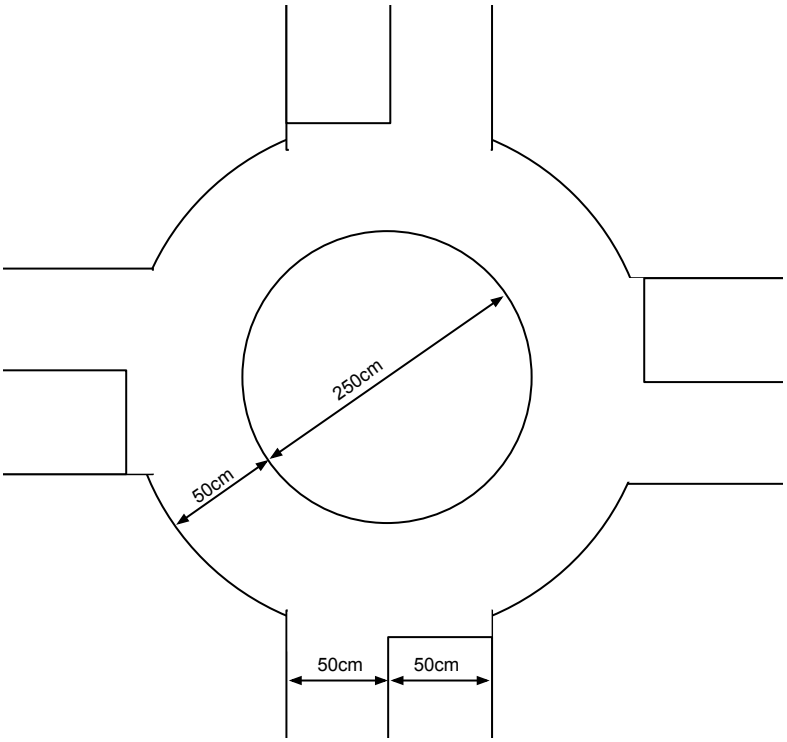

Figure 4: Overview and dimensions of the roundabout.

Additionally, the cars differ from one another since they are mechanically non-identical. Thus, their range of operation are unequal. Table 4 presents the range of operation for each car. Worth mentioning is that the electric motors of the black and red car are more equal in terms of behaviour than the one of the orange. The motors of those cars experience a poor performance at lower speeds, which is likely due to stronger magnets within the motor. This phenomenon is connected to the stated minimum speed in Table 4, which is basically the lower boundary for acceptable behaviour of a car. The maximum speed is set to make the cars operate similarly when maximum throttle is requested.

Table 4: Range of operation of the cars.

Car	Min. speed [m/s]	Max. speed [m/s]
Orange	-	≈ 1.3
Red	≈ 0.4	≈ 1.5
Black	≈ 0.4	≈ 1.5

4 Installation

The installation of the cars are described in this chapter. This is only needed if either a computer or car stops being functional and have to be thoroughly reset.

4.1 Laptop

The laptop used to run the cars should use Ubuntu, version number 16.04 has been tested, and ROS Lunar 1.13.7 has to be installed as well. The laptop will act as the rosmaster and must be identified as such before attempting to run the cars. The easiest way to do this is by following the file `cascar/README.md` and add the `cws` command to allow for easy configuration. When adding the `cws` command for the laptop it is important to change `setrosclient.sh` to `setrosmaster.sh` instead. You also need to have the `rosmasterip.service` running for it to function properly which is described in `cascar/develop/conf_scripts.md`

For convenience, VNC should be installed on the laptop. The VNC used is called RealVNC. Connecting to the cars is described in `cascar/README.md` in the git repository.

4.2 Installation of Cars

The installation process of a car is quite substantial and all parts will not be described fully in this section. However, the relative structure of what needs to be done will be described.

The Rasberry Pi needs to be installed on the car. There are several guides for how this can be done so it will not be described here in detail. A short instruction can be found under `cascar/doc/rasberry_install.md` in the git repository.

The communication of the cars is done by using ROS, which has to be installed on the cars for them to able to function. Furthermore, the workspace where the ROS code is run needs to be installed. Thereafter, the code for the cars needs to be cloned into the correct file. The code for the Lidar and qualisys also need to be added. How this is all done can be found under `cascar/doc/installation.md` in the git repository. **Note**: the Udev rules described in the last part of the file must be followed for the joystick to be functional.

To have VNC installed on the cars is practical. How this is done properly is described at `cascar/README.md` in the git repository.

Communication between the cars requires that the cars are on the same network and that the IPs are correctly set by ROS. To make sure this is done correctly in ROS please follow the instructions located at `cascar/doc/conf_scripts.md`. Furthermore, a script to automatically set the Wi-Fi the cars are connected to is described there. **Note**: the cars are ROS-clients and the project laptop is ROS-master.

For practicality a function called `cws` is added to the car. How this is done be seen in the file "`cascar/README.md`" located in the git repository.

4.3 Installation GUI

If you have installed ROS lunar with either **Desktop-Full** or **Desktop** you should already have rqt installed. If you however have not or want to install QTDesigner, to be able to create or change the UI, you can follow the instructions in `rqt_mypkg/doc/INSTALL.md`.

4.4 Installation Simulator

For the simulator to run, Gazebo and gazebo_ros_pkgs need to be installed. Instructions are found in: `cascar_sim/doc/INSTALL.md`

4.5 Required Python Packages

Some extra Python packages need to be installed for all scripts to work.

Table 5: Extra Python packages required.

Package	Platform	How to install
SciPy	Laptop	`sudo apt-get install python-scipy`

5 Setup

The setup of the cars describes how to get the whole system up and running. For both the real user case and the simulated case this process will be thoroughly described.

5.1 Safety Concerns

- When connecting the car to a screen, make sure that its wheels are off the ground to prevent damage if any unpredictable motions occur
- Follow the safety regulations of Visionen
- Handle batteries with caution.

5.2 Map Creation

The map creation tool exists to make it easier for the user to create new maps without having to do it by hand, which can be highly time consuming. The creation tool is a Textbased User Interface (TUI) with a few repeating questions.

5.2.1 Structure

The map creation tool has two different building blocks: road segments and roundabouts. Road segments can be seen as standard roads, and are created by choosing three points that defines the curvature of the segment. Roundabouts are pre-made roundabouts that are connected to the last road segment made.

5.2.2 Startup

To use the map creation tool, start by navigating to `cascar/scripts/MC`. While in this directory, call the map creation tool script by typing `python map_creation_tool.py`. The terminal should then present the following text:

```
Let's create a map!
Continue creating map?  [y/n]
```

If the user presses `y`, the map creation process is initiated.

5.2.3 Adding an Initial Road Segment

After the startup, the user will be presented with the question:

```
Create road segment or roundabout?  [rs/r]
```

In this scenario, directly after startup, the only option is to create a road segment `rs`, since there are no road segments to connect the roundabout to. A blank figure will then appear, with x- and y-values between -5.4 and 5.4. With the mouse, the user can choose three points, along which the road segment will go. Please be aware that the first point clicked is expected to be the first point in the right lane, independently if the user chooses to create the road from left to right or right to left. The left lane will be created automatically next to the right lane. When the points have been chosen, the figure will update and present the road segment that has been created. When exiting the figure, the program goes to its main loop, see section 5.2.4.

5.2.4 Main Loop

The main loop of the map creation tool starts with asking if the user wants to continue creating the map. If the user is satisfied with the map in its current state, the answer should be no (`n`) and the program prepares to exit, see section 5.2.5. If the answer is yes (`y`), three additional questions will be asked. Below, the questions and possible answers along with consequences of each answer is described.

Question 1

```
Do you want to use a previous node as the starting point?
[y/n]
```

If the user wants to start the next road segment or roundabout on a previously defined road segment or roundabout, the answer to this question should be yes (`y`). When the intention is to create a roundabout, the answer to this question must be yes. If the user answers yes and instead wants to create a new road segment, the first point of the new road segment must be close to an end of a previous road segment or roundabout. The map creation tool will then see the end of the previous road segment as the starting point of the new one.

If the user answers no (`n`), the next road segment can be started at any point in the figure.

Question 2

```
Do you want to connect the last node to another road segment
[y/n]?
```

This is essentially the same question as the previous one, except this regards the last node in the new road segment instead of the first. If the answer to this question is yes (`y`), the user should place the last point of the new road segment close to the start of a previous one. If the answer is no (`n`), this requirement can simply be ignored. If the

user is creating a roundabout, the answer to this question should be no.

Question 3

`Create road segment or roundabout? [rs/r]`
To create a road segment, the answer should be `rs`, while the answer `r` creates a roundabout. If a roundabout is chosen to be created, the user does not have to specify anything more, since a roundabout automatically will be created, starting at the last road segment made. If the user instead wants to create a road segment, the same procedure as described in section 5.2.3 needs to be done, possibly with some additional requirements depending on the answers to Question 1 and 2. When creating the road segment, all previous nodes made will be shown in the figure to make it easier to place out the new road segment.

When a new road segment or roundabout has been created, it is shown in the figure. Upon exit of this figure, the main loop starts over again.

5.2.5 Saving and Exiting

When the user is satisfied with the map, and therefore has answered no to continue creating the map, the map is shown one last time. When exiting the figure, the user is asked to write a name to save the map in. The map is saved as a .txt-file, but the .txt-ending is not necessary in the name. The map will be saved in the current directory, but to be able to use the file in the GUI later on it needs to be manually moved to `cascar/maps`.

5.2.6 Manual Editing

It is also possible to edit the map manually after it has been saved. To do this, got to `cascar/scripts/MC` and run `python show_map` with the map name as argument, without the .txt-ending, to show the map. Also open the map .txt-file in an editor of preference. This way the user can look at the map and change the coordinates of nodes that are not in satisfactory positions. Please be aware that removing nodes or changing other arguments, such as road_type or id, of nodes may cause troubles later on when a path is to be created and is therefore not recommended.

5.2.7 Known Limitations and Errors

There are some known limitations and errors present in the map creation tool. This section will try to present the most common ones. The biggest limitation is the fact that

there is not any revert functionality. This means that if an error (unless minor) is made, the user needs to remake the whole map. If the error indeed is minor, there might be a possibility to fix the error manually later on.

Wrongsided Opposing Lane

As stated earlier, the opposing lane is created automatically by the program when the user makes a road segment. However, in some cases the opposing lane is created on the wrong side of the manually created lane. This scenario is especially common when the sign of the derivative of the curve changes. Often the opposing lane starts on the correct side, but switches side when the derivative changes sign. This problem seems to be more common when making a curve primarily in the x-direction rather than the y-direction.

Road Segment Too Straight

When the derivative between two points is exactly 0, the program tries to divide by zero and throws an error. This error is however not that common.

Too Sharp Turns in the Roundabout

The in- and outgoing lanes to and from the roundabout are often too sharp for the car to handle. Therefore it is almost mandatory to change the coordinates of these nodes manually after the map is created. These nodes are easy to spot in the map txt-file, since they have special road_types (1 and 2).

5.3 Cascar Setup

To get the cars running three separate systems have to be initialized, the cars, VPS and the laptop containing the GUI.

5.3.1 Laptop

The project laptop needs to be connected to the Visionen network. Once it is connected the project laptop needs to be made ROS-master. This is done by using **cws** function in the terminal. Note that it needs to be done for every terminal window being used.

5.3.2 Vehicle

Before the setup of the car can commence, make sure that the cars all have fully charged batteries.

To be able to get the cars running they have to be connected to a computer screen (separate display or laptop). This can be done using a HDMI cable and connecting it to a screen or through a Ethernet cable connected to a laptop having the program VNC installed (the project laptop has the program installed). By using VNC and a Ethernet cable the project computer can connect to the car and therefore reducing the need to have an extra screen used specifically for the cars, which can be practical when experimenting in Visionen. For further information see the file, `cascar/README.md` located in the git repository. All the cars needs to be connected to the Visionen network, which is because Qualisys publishes the location and orientation of the cars on a topic on that network. Once the cars and the laptop are all located on the same network the car can be accessed using ssh. However, the IP of the cars must be known before they can be accessed through ssh. By firstly having run cws on the project laptop and then doing it on the cars the IP of the cars can be seen. The following command allows you to access the car using ssh:

```
> ssh cascar@192.168.xx.xx
```

Where xx.xx are different depending on the car being used. Important to note is that the cars can be accessed from any computer which is on the same network as the cars, as long as the IP addresses of the cars are known.

Once the car has been accessed remotely, it can be started. How this is done can be seen in chapter 6.

5.3.3 Visionen Positioning System

The process of setting up VPS is not trivial, especially if the system needs to be calibrated. As earlier mentioned, VPS is a positioning system called Qualisys which makes use of infrared cameras to detect reflective balls on physical objects. It is of high importance that the reflective balls placed on the cars have a unique pattern so the cars becomes distinguishable from each other. When starting the software a base project has to be chosen. On the computer used in Visionen there is a base project located under `A:/Documents/2018-03Base Base Settings-do not change/Settings`, copy this project to your workspace and use that as your default project when opening Qualisys. The process of starting and adding cars to Qualisys is typically:

- Make sure the blinds are down so no sun distorts the measurements
- Start the computer and Qualisys with the default project

- Change Euler angle settings by going to:
  - Chose "Settings" under the "Tools" tab
  - Go to "Euler angles"
  - Change from "Default" to "Qualisys Standard"
  - Make sure to apply your changes
- Check the calibration
  - If calibration is sufficient, then:
    - Try using the current calibration
  - If calibration seems insufficient, then:
    - Follow instructions to re-calibrate Qualisys:
      - Assemble calibration stick
      - Place the other L-shaped calibration stick in where you want your coordinate systems origin to be located, it should be in the middle of the rooms since most cameras can see clearly in the middle. The long end of the L-shaped stick defines the x-axis and the short end the y-axis
      - Choose new in the software, otherwise calibration is not possible
      - Set a time for calibration, reasonably around 4-5 minutes
      - When the calibration starts, walk around the room slowly waving the stick up and down to calibrate the system
- Define cars that are to be used in Qualisys. Follow these instructions to define a body in Qualisys. Note that the objects that are defined needs to have a few reflection balls positioned on them
  - Place the object you want to define close to the origin and point it towards the x-axis.
  - Start a measurement, make sure the reflective balls can be seen clearly
  - Stop the measurement, you need to do this since you can't define an object once the system has started measuring
  - Press down shift and use the mouse to pick out all the reflection balls that belongs to one object
  - Right click on one of the balls and choose "Define rigid body"
  - Name the object. The name of the cars should be orange, red or black depending on the color of the car being used otherwise the cars cannot read their respective position

  - Make sure the coordinate system of the cars matches the physical cars, that is x pointing in the direction of travel and z point upwards. If it does not it can be changed under "Settings"

- Start measuring
- Once Qualisys is measuring, the node to publish the actual data can be started. It is started by the following command: (The Qualisys package has to be installed on the computer launching, the computer also has to be connected to the Visionen network)
  - `roslaunch qualisys qualisys.launch`
- Use the following function to check that data is being published: (Make sure you have used the `cws` function in the terminal that examines what is being published)
  - `rostopic list`
  - You should see that a message with the following structure is published:
    - /qualisys/[your given name of the object]

A typical problem that occurs when starting Qualisys is that no data comes. The terminal running the "qualisys.launch" will show "no more data". To solve this stop the measurement and then restart it, it usually solves the problem.

It should be noted that the quality of the measurement from Qualisys differs strongly between calibrations and the system in general is very unpredictable.

5.4 Projector and Map Displaying Setup

In the VPS, there are two projectors in the ceiling. These can be used to project a map to the floor, making it easier to see how the cars perform in real life. To be able to display a map, there are a few steps to pass. These steps assumes that a map has been saved as a png-file, with x- and y-axis going from -5.4 to 5.4 m. Everything except the actual plot e.g. axes, needs to be cropped out of this image.

In the VPS control room there is a computer next to the computer used by the VPS. Log in to this computer and make sure that the map exists locally. Turn on the projectors by using the projector software found on the desktop. The projectors' display should now be visible as an additional screen, meaning it should be possible to just drag any window to them (often to the right of the initial screen). Open **Paint** and open the map image. Move the Paint window to be mostly in the projectors' screen, but not completely. Then

choose the option **Full screen** under the menu **View** and the map should be displayed in the VPS.

Use the functions in **Paint** to rescale, rotate and move the image until the map has the correct proportions and position. A first initial tip about the scaling is to at least double the size of the image, but ultimately this comes down to testing what works.

5.5 Simulator Setup

Since no physical cars are used in the simulation the setup becomes easier.

For the car model to display properly the path to the 3D model has to be set correctly. This is done by altering the uri in:

`/cascar_ws/src/cascar_sim/vehicle_models/car/model.sdf`

The current path to Car.dae should look like

```
<uri>/home/name_of_user/cascar_ws/src/cascar_sim/...
vehicle_models/car/Car.dae</uri>
```

Where the *name_of_user* needs to match the system. For the project laptop it is *tsrt10* at the time of writing this. Note that this has to be altered for each different computer/user. If this path is incorrect the model will spawn without warning but will be invisible in Gazebo.

5.5.1 Adding and Displaying Maps

For a map to show properly in Gazebo there are some steps that need to be completed.

1. For the existing maps to be displayed properly, locate the folder: `~/cascar_ws/src/cascar_sim/worlds` The folders *materials* and *track* will need to be copied to the hidden folder: `~/.gazebo/models`
2. The *.world files located in: `~/cascar_ws/src/cascar_sim/worlds` need to be copied to: `/usr/share/gazebo-X/worlds` where x is the version of Gazebo that you are running.

Now the maps can be launched using the virtual_world.launch file.

The easiest way to create a new map, called *my_track*, is as follows:

1. Add the picture. The image should be saved as my_track.png with both x- and y-axis from -5 to 5 m and everything except the actual plot cropped out (e.g., axes), and then copied to: `~/.gazebo/models/materials/textures/`

2. Locate the file:
   `~/.gazebo/models/materials/scripts/template.material`
   Copy it and name the copy *my_track.material*.
   Open it and change the first line to: material MyTrack/Image.
   The texture line should also be changed to: texture my_track.png

3. Locate and copy the file:
   `~/cascar_ws/src/cascar_sim/worlds/template.world`
   Open the copy and change the line:
   <name>Template/Image</name> to: <name>MyTrack/Image</name>
   Save the copy as *my_track.world* and copy it to the folder:
   `/usr/share/gazebo-X/worlds`
   where X is your version of Gazebo

4. You should now be able to show your map using:
   ```
   > roslaunch cascar_sim virtual_world.launch ...
   ...  world:='my_track'
   ```

6 Usage

This chapter discussed how the actual usage of the car will occur.

6.1 Usage for Cascar

Describes how to use the actual cars. Important to note is that everything has to be properly setup for the system to be fully functional.

6.2 Start Cascar

Once the instructions for the setup have been properly followed the cars can be started. By using the terminal connected to the car and using the following command the car will be started:

```
> roslaunch cascar cascar.launch my_ns:=name man:=0
```

where name is either orange, red or black and `man:=` is equal to 0 or 1 controlling the mode the car will start in. Every car must be started using this launch file. The launch file itself starts all the different subsystem that controls the car. It is very important that the `my_ns` argument is set since the communication between the GUI and the car is dependent on one of the three names is being used. The `man` argument is set to 1 on

autonomous cars without paths that should still attempt to platoon. The man argument is therefore set to 1 on the car that you want to follow a car that is driven manually.

If the car wants to be driven manually this can be set up quickly. The following command, when run on any laptop connected to the Visionen network, starts a node that enables the user to drive the chosen car manually:

```
> roslaunch cascar joy_control.launch my_ns:=name
```

where name is the car you want to control.

6.3 GUI

The GUI can be started by:

```
> roslaunch rqt_mypkg rqt_mypkg.launch
```

An example of how the GUI will look, after you have selected a map and the red car is active can be seen in Figure 5.

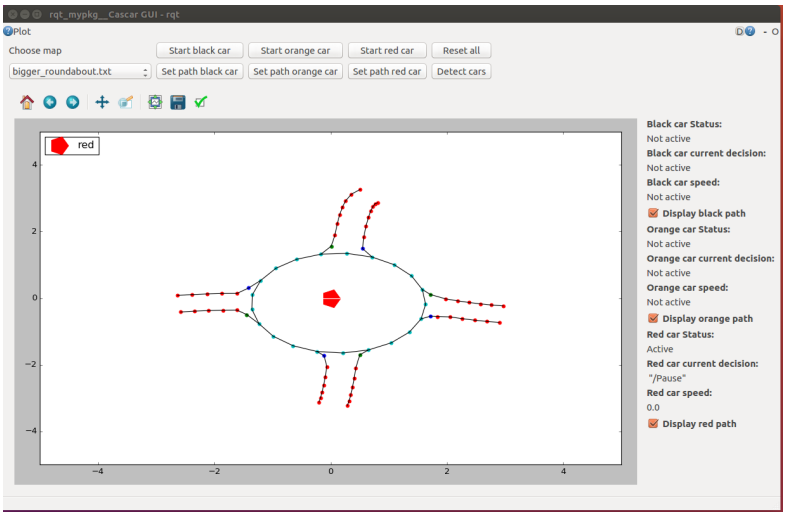

Figure 5: GUI window with red car active.

6.3.1 Map Selection

A map can be selected by viewing the dropdown menu in the top left corner. Once a map is selected it will be shown in the GUI and its name will be sent to the active cars.

6.3.2 Detect Cars

If you start the GUI before the cars, you need to press the **Detect cars** button. It will find all active cars and display them on the map.

6.3.3 Selecting Path

By pressing for example **Set path black car** you will be able to select way points for the black car in the map with pressing the left mouse button. Once you are satisfied you need to press the right mouse button to receive the path. The path will be shown on the map if you have the box **Display black path** checked on the right side of the map.

If you only select one way point at the same location as the car, the car will try to find a loop which it can drive through repeatedly.

6.3.4 Start Car

You simply start the car by pressing its own **Start car** button. If the car does not have a path, nothing will happened.

6.3.5 Troubleshooting

If you experience a malfunction with the GUI, you can try and press the **Reset all** button. It will reset most of the variables and unsubscribe from all cars and then search for active cars.

If this does not work, try restarting the GUI.

6.4 Simulator

To start the simulator run:

```
> roslaunch cascar_sim virtual_world.launch
```

This should only be ran once launches Gazebo and the virtual Qualisys node. For information about the *world* argument and how to create a new world, see Section 5.5.1

For each car you wish to launch, run:

```
> roslaunch cascar_sim virtual_car.launch
```

This file may take the following arguments:

- *x_pos* & *y_pos*: Initial position [m]. Default {0, 0}
- *theta*: Initial orientation [rad]. Default 0
- *model*: Name of the car [string]. Default *orange*.

There is no theoretical limit to how many cars that can run at once. The only requirement is that *model* is a unique string.

From here on the system is used just as for the physical cars.

7 Troubleshooting

Some know errors that have not yet been discussed will be presented in this chapter.

7.1 The Vehicle Behaves Unpredictably

Most often caused by loss of Qualisys position data. This problem can be solved by:

- Performing a refined calibration or an entirely new one in Qualisys
- If you already have a well defined calibration, it might be certain dead spots in the VPS which cause the problem. A known dead spot is the space within 1 m of the walls in Visionen. Hence, avoidance of driving in this region is recommended
- If the previous actions does not resolve the problem, it might be that the Qualisys standard angles are not used. This can be fixed in the Project Options menu under "Euler Angles"
- Another possible solution would be to change the battery to a newly charged one.

8 FAQ

Why do the cars' coordinate systems get mixed up in Qualisys?

Probably because a car has been taken out of VPS whilst running a capture. This can not be done, since Qualisys tries to fit the car being taken out to the other available cars. Alternatively, the pattern of the reflective balls of the cars are too similar.

Why does the calibration not pass?

The reason can be found on the dialog window after a calibration has been performed. Most often, it is due to a high wand velocity or insufficient data points for each camera (500 are required).

9 Further Help

If there is a problem that can not be solved by using this user guide, please visit the relevant website at first hand for further help. If the problem is still unresolved, contact Prof. Erik Frisk or PhD student Pavel Anistratov at Linköping University.

Websites

Gazebo Simulation Toolbox

ROS Lunar

Qualisys Track Manager - User Manual

Contact Information

Erik Frisk

Professor
Vehicular Systems
Dept. of Electrical Engineering
Linköping University
erik.frisk@liu.se

Pavel Anistratov

PhD student
Vehicular Systems
Dept. of Electrical Engineering
Linköping University
pavel.anistratov@liu.se

References

[1] A. Ahlgren, J. Kokko Ekholm, K. Gudmundson, et al. *Design Specification Version 1.0.* 2018.

[2] A. Ahlgren, J. Kokko Ekholm, K. Gudmundson, et al. *Requirement Specification Version 1.1.* 2018.

[3] ROS Wiki. *ROS Lunar Loggerhead*. Last edited 2018-01-08 by Tully Foote. URL: `http://wiki.ros.org/lunar`.