CarMaker Driver Trajectory Extension

Proof of Concept

Date: May. 28th, 2020
Author: IPG Automotive GmbH

Table of Content

1 Overview

1.1 Introduction

The challenges of vehicles for autonomous driving have a high focus on interconnected control strategies with different hardware and software components. The overall strategy needs to cover a wide range including

- Sensing the environment and own vehicle state
- Motion planning (e.g. trajectory based)
- Vehicle control based on target trajectory and current vehicle state
- Actuator control for the manipulation of the vehicle state

While sensing and actuator control is more a generalized topic the motion planning and vehicle control itself have high dependency on the vehicle behaviour in lateral and longitudinal movement. Especially the motion planning is currently a challenging field in today’s vehicle development.

A Driver Trajectory extension (abbreviation in this document: DrvTrj) for CarMaker is currently being developed as proof of concept. The behaviours of the IPGDriver and the responses of the ego-vehicle towards the given target trajectory in a virtual test environment can be easily tested. This Driver Trajectory extension can be used in many fields, e.g. verification of motion planning algorithms of autonomous driving vehicles and testing of vehicle control in a cooperative driving assistance system.

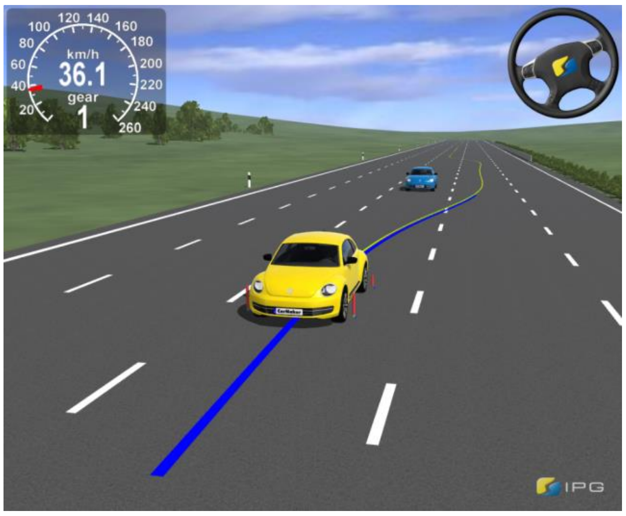


Figure 1-1: Visualization for Driver Trajectory extension in IPGMovie

1.2 System Requirements

General system requirements are linked to the requirements of Car-/TruckMaker (see CarMaker Release Notes).

Currently there is **only support** for **Car-/TruckMaker Office** for Linux and Windows.

Please contact IPG for additional information!

1.3 Concept

Many motion planning algorithms for autonomously driving vehicles generate trajectories for movement based on the information from environment perception and mapping localization. Before a target trajectory can be generated, a feasible path from start position to destination position needs to be computed. Then a trajectory planning module can find a most suitable trajectory (generally a path with dependency on time) while taking the vehicle kinematics/dynamics constraints, e.g. velocity/acceleration limits and many other factors into account.

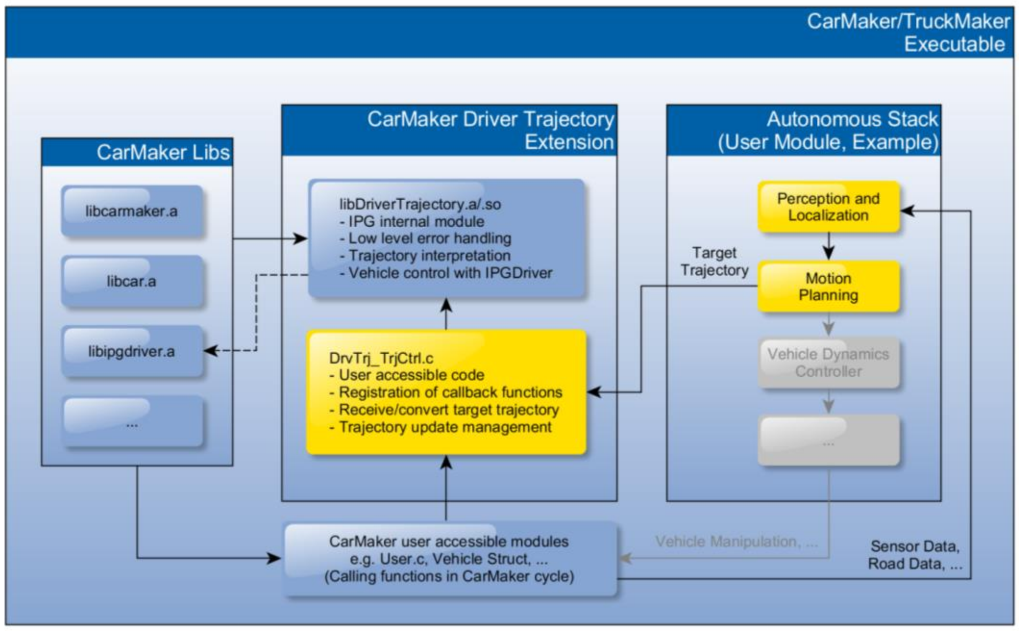

CarMaker/TruckMaker Executable
CarMaker Libs
libcarmaker.a
libcar.a
libipgdriver.a
...
CarMaker Driver Trajectory Extension
libDriverTrajectory.a/.so
- IPG internal module
- Low level error handling
- Trajectory interpretation
- Vehicle control with IPGDriver
DrvTrj_TrjCtrl.c
- User accessible code
- Registration of callback functions
- Receive/convert target trajectory
- Trajectory update management
Autonomous Stack (User Module, Example)
Perception and Localization
Target Trajectory
Motion Planning
Vehicle Dynamics Controller
...
CarMaker user accessible modules
e.g. User.c, Vehicle Struct, ...
(Calling functions in CarMaker cycle)
Vehicle Manipulation, ...
Sensor Data, Road Data, ...

Figure 1-2: Driver Trajectory extension and Autonomous Stack dependencies (Example)

Once a target trajectory is generated, it can be used for motion control within the CarMaker simulation environment (Figure 1-2). The Driver Trajectory extension code example (DrvTrj_TrjCtrl.c) is called directly and indirectly within the CarMaker calculation chain. It provides functions for common CarMaker hook points in “User.c” and shows basic mechanism how users can develop their own functions or integrate other already existing algorithms. The Driver Trajectory extension library works as a bridge between the user accessible code and CarMaker internal libraries that are necessary for trajectory based vehicle control. It requires information passed from the user accessible code, e.g. the target trajectory and includes several modules to manage trajectory requests and processes the input.

During the simulation the Driver Trajectory extension interacts with other CarMaker modules, e.g. IPGDriver. The simulation of control actions of a real driver is done by IPGDriver. The IPGDriver takes the interpreted target trajectory from the library as necessary inputs, then calculate a response and pass the appropriate output values back to the system.

Therefore, the CarMaker/TruckMaker Executable with Driver Trajectory extension has the ability to form a close loop simulation with some modules in autonomous vehicle stack, e.g. environment perception, localization and motion planning and make the development and testing much easier.

The basic mechanism how trajectories have to be provided to the CMDrvTrj Extension (Simulation) is shown in the following pictures. Once the module is activated the target trajectories are requested by an automatically called callback function where the user has the possibility to update/confirm the validity of the currently active trajectory or throw an error while simulation is running.

The provided trajectory (as point list for multiple time steps) need to cover some time/distance in the future and a bit in the past to correctly localize the current and upcoming target positions. An estimated time/distance in future is provided by the callback function. Currently the target trajectory is used as it is provided and no filtering, splining or equalizing is done. So the user is responsible for a plausible generation.

The position and orientation for each point of the target trajectory has to be provided in the CarMaker global coordinate system (Fr0) in the vehicle's Point of Interest (PoI, see CarMaker Reference Manual) and might need to be transformed based on the definition of the external trajectory format. Other quantities (vehicle velocity, acceleration, …) have to be described in the vehicle fixed coordinate system Fr1 in the vehicle's Point of Interest.

Chapter "2 Quick Start" describes how to build the CarMaker executable with the Driver Trajectory extension to run the example and details e.g. for parameterization can be found in chapter 3.

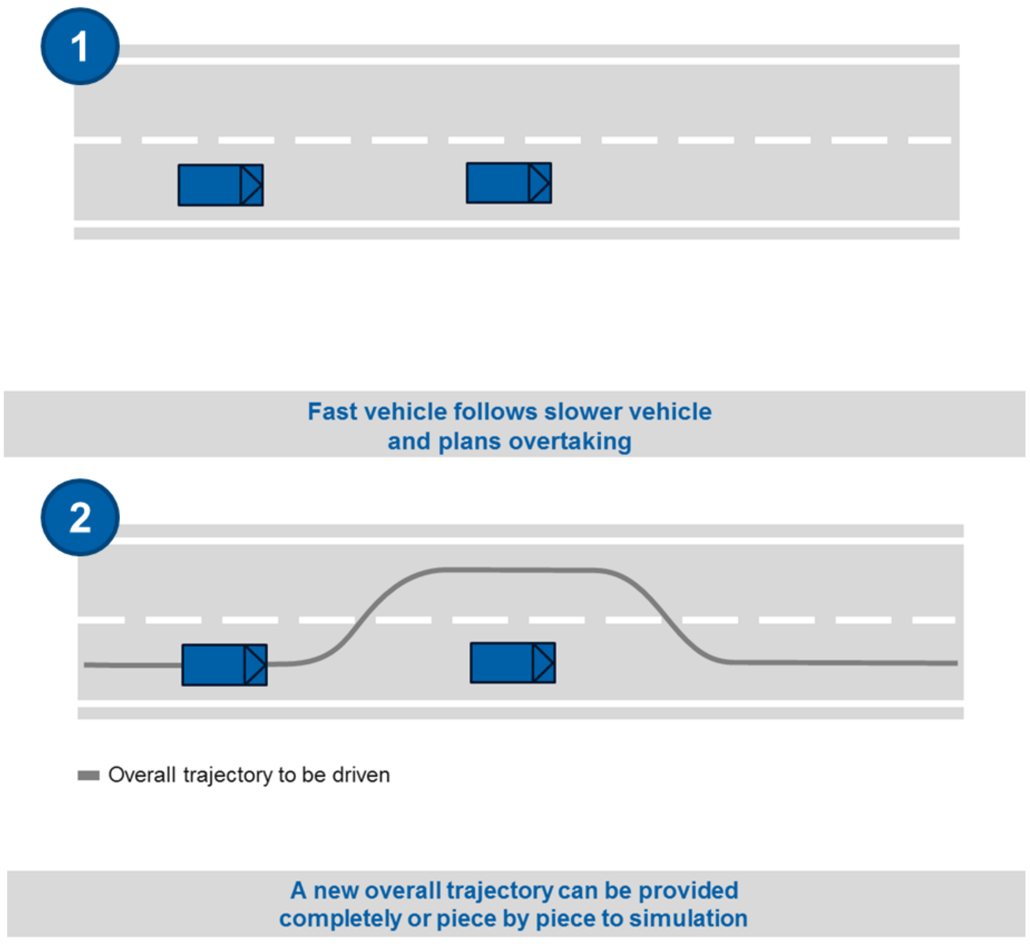
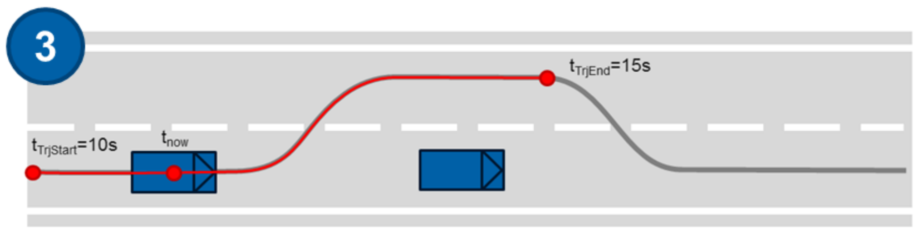
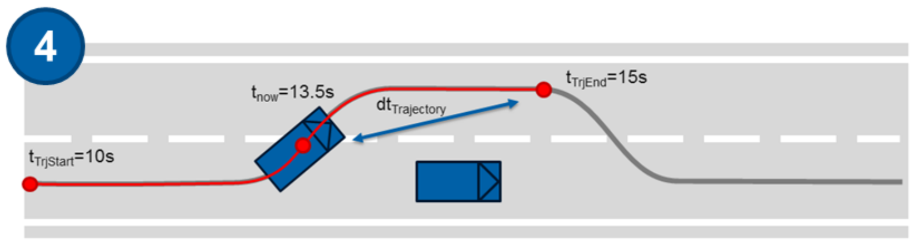

1
Fast vehicle follows slower vehicle and plans overtaking
2
Overall trajectory to be driven
A new overall trajectory can be provided completely or piece by piece to simulation

Figure 1-3: Exemplary procedure of trajectory generation (Part1)



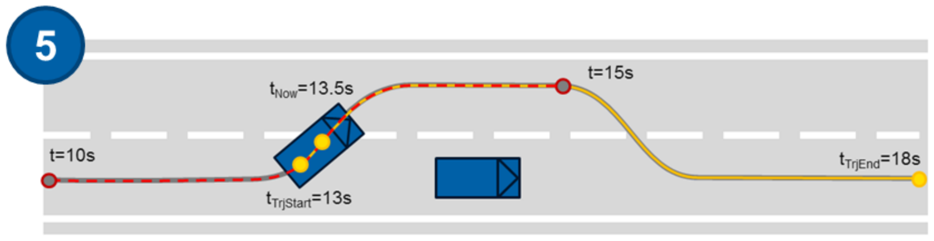
▬ Overall trajectory to be driven
🟥 1. Trajectory for simulation (at PoI in Fr0, point list, $t_{TrjStart}$=10s)

1. Trajectoy is provided to simulation in global coordinates at Vehicle‘s Point of Interest (e.g. duration = 5s)



▬ Overall trajectory to be driven
🟥 1. Trajectory for simulation (at PoI in Fr0, point list, $t_{TrjStart}$=10s)

Flexible trigger condition is reached
e.g. updated trajectoy or $dt_{Trajectory} < dt_{Limit}$



▬ Overall trajectory to be driven
🟥 1. Trajectory for simulation (at PoI in Fr0, point list, $t_{TrjStart}$=10s)
🟨 2. Trajectory for simulation (at PoI in Fr0, point list, $t_{TrjStart}$=13s)

Update the trajectory for simulation
$t_{TrjStart}$ must be a bit in the past

Figure 1-4: Exemplary procedure of trajectory generation (Part 2)

2 Quick Start

2.1 General Information

- This example is based on CarMaker 8.0.2, for other CarMaker versions there some updates needed, e.g. library version for other major/minor release. For bugfix releases (8.0.x) renaming the library might be sufficient. Please contact IPG for other library versions!
- Current CarMaker Driver Trajectory extension is a prototype and will be improved in future.

2.2 Preparation

Following the several steps below to install the CarMaker/Truck Executable with Driver Trajectory extension. Necessary files can be found in the delivered package.

- Install CarMaker /TruckMaker according to the CarMaker “InstallationGuide.pdf”.
- Install Driver Trajectory extension. Please check chapter 3.1 for detailed description.
- The delivered package with the Driver Trajectory extension is prepared for a specific CarMaker/TruckMaker version. See comment above!

2.3 Build CarMaker Executable

Build CarMaker executable to run examples.

- Ensure all the source/header files and libraries are in their proper directories.
- Make sure the source files and Makefile in CarMaker Project Directory are properly updated.
- Open a terminal and navigate to “<CMProjDir>/src” directory.
- Execute “make” to build CarMaker executable.
- Please check steps 3-5 in chapter 3.1 for detailed descriptions.

2.4 Run the Example

After the CarMaker executable is available you can run the delivered TestRun examples with Driver Trajectory extension in CarMaker.

- Start CarMaker GUI according to your operating system.
- Choose the newly built executable with Driver Trajectory extension from “<CMProjDir>/src”.
- Open the TestRun Examples: "CarMaker Main GUI -> File -> Open -> DrvTrj_Demo_Overtaking or DrvTrj_Demo_SinusCourse”
- Click “Start” and check the messages at “CarMaker Main GUI → Simulation → Session Log”. By successful implementation the log messages will be displayed in “Session Log” window.
- Open IPGMovie via “CM Main GUI -> File -> IPGMovie” to see data visualization of Driver Trajectory extension during the simulation.
- Two different user trajectory modes are available, they can switched via Infofile parameters. For details please see chapter 4.

3 Documentation

In this chapter a detailed description about the installation process for CarMaker executable with Driver Trajectory extension and its general parameterization is given.

3.1 Installation

Follow these steps to install the CMDrvTrj extension to an already existing CarMaker Project.

1. Create New CarMaker Project Directory
   - The CarMaker Driver Trajectory extension needs an already existing CarMaker Project directory with included sources/build environment ("src/" folder with User.c, CM_Main.c, …) to build the CarMaker executable. If the "src/" folder is missing please use "CM Main GUI -> File -> Project Folder -> Update Project…" with enabled component "Sources/Build Environment".
   - It is recommended to create a backup of your already existing CarMaker Project Directory.
   - The update procedure is identical for CarMaker and TruckMaker.
2. Extract provided Files
   - Copy the following files or directories from zip file into the corresponding folders into your already existing CarMaker Project Directory (or directly unzip inside).
     - Movie/DriverTrajectory.tclgeo
     - include/DriverTrajectory.h
     - lib/CM8.0.2 (Pay attention to the naming of the lib folder and the libraries, e.g. in CarMaker 7.1.2 version and Linux64 system the delivered should be: "<CMProjDir>/lib/CM7.1.2/libDriverTrajectory-linux64-CM7.1.2.so".)
     - src/DrvTrj_TrjCtrl.c, DrvTrj_TrjCtrl.h
   - In general files below need to be updated. We recommend using simple code compare tools, e.g. Meld or WinMerge, to compare source code.
3. Update "src/Makefile"
   - The Makefile rule to build the CarMaker executable needs to be extended so that the library for the CarMaker Driver Trajectory extension can be loaded properly. Please compare delivered "Makefile_DrvTrj" with current Makefile in CarMaker project.
4. Update "src/User.c"
   - Copy the lines marked with "WITH_DRVTRJ" from "User.c_CMDrvTrj" to your User.c".
5. Build CarMaker executable
   - Linux: Open terminal and navigate to the "<CMProjDir>/src" and execute "make" to build CarMaker executable with Driver Trajectory extension.
   - Windows: the MSYS development environment should be installed. Details see CarMaker Installation Guide 2.6 "Installing MSYS-2017". Then same as in Linux environment, open terminal and navigate to your own the "<CMProjDir>/src" and execute "make"
6. Open "CarMaker GUI -> Application -> Configuration / Status", choose the freshly built executable in "<CMProjDir>/src/" folder and click "Start & Connect". To make sure the correct executable is connected, check the information showed at the same window under "Application Status and Compile info".

3.2 Folder/File Overview

The paths below are relative to the CarMaker project directory and describe the structure and content needed for the Driver Trajectory extension (may vary based on OS/CM Version).

```
CM_DrvTrj_Demo
|-- Data
|   `-- TestRun
|       |-- DrvTrj_Demo_Overtaking
|       `-- DrvTrj_Demo_SinusCourse
|-- Movie
|   `-- DriverTrajectory.tclgeo
|-- SimInput
|   `-- DrvTrj_static_trajectory.info
|-- bin
|-- doc
|   `-- CMDrvTrj_UsersGuide.pdf
|-- include
|   `-- DriverTrajectory.h
|-- lib
|   `-- CM8.0.2
|       |-- libDriverTrajectory-linux64-CM8.0.2.so
|       `-- libDriverTrajectory-win64-CM8.0.2.a
`-- src
    |-- DrvTrj_TrjCtrl.c
    |-- DrvTrj_TrjCtrl.h
    |-- Makefile_DrvTrj
    `-- User.c_DrvTrj
```

Figure 3-1: Folder/File overview for the example

Description of folders/files in Figure 3-1:

- **Data/TestRun**
  - Example TestRuns provided with this package
- **SimInput/**
  - Infofile with static trajectory that is used in the static trajectory mode (see chapter 4.2).
- **Movie/**
  - Tclgeo script for visualization of the trajectory and other data in IPGMovie.
- **include/**
  - The header “DriverTrajectory.h” is located in this folder.
- **lib/**
  - Folder with DriverTrajectory extension libraries for different platforms and CM versions
- **src/**
  - **Makefile_DrvTrj**
    - CarMaker Makefile with additional flags/libraries for this extension
  - **User.c_DrvTrj**
    - Calling functions with defined flags in “DrvTrj_TrjCtrl.c”
  - **DrvTrj_TrjCtrl.c**
    - Functions in this source file called in “User.c”
    - User accessible code with trajectory generation and preparation for CMDrvTrj module

3.3 Activation of Trajectory Control

The Driver Trajectory extension can be activated via a Minimaneuver Command (e.g. Realtime Expression shown below) or a DVA command.

```
Eval first(Time > 5) ? DrvTrj.ModeReq = 1
```

For further information about Minimaneuver Commands and DVA please refer to CarMaker User's Guide. Driver Trajectory extension is activated if “DrvTrj.ModeReq” is set to “1”. By default, it is reset to “0” before each simulation start and therefore the extension is not activated.

It is necessary that **at least one valid route** is defined in the CarMaker TestRun. The module can be activated after IPGDriver has already followed a route. When the module is deactivated IPGDriver will continue following this initially defined route.

Backward driving is currently not supported. Please contact IPG for more information.

An example how to use the Driver Trajectory Extension can be found in chapter “4 Examples”.

3.4 Tuning the Vehicle Control

The Driver Trajectory extension uses IPGDriver for the vehicle control in lateral and longitudinal direction. Therefore it is possible to tune the control behaviour with the parameters available via “CM Main GUI -> Parameters -> Driver”.

For example limits for lateral acceleration, steering wheel angle/velocity/acceleration influence the lateral follow behaviour.

Additional tuning can be achieved with a manipulation of the Driver Knowledge in the TestRuns Infofile. This can be imported from other TestRuns or generated via Driver Adaption.

For example the Driver Knowledge parameter “Driver.Knowl.Lat.tPreview” can be used to tune the control behaviour for very dynamic trajectories. Smaller values make the control more aggressive. Take care with this parameter as too small values result in an unstable control. Please note that parameters with prefix “Knowl.” requires driver adaption performed or the set “Driver.Knowl.Learn.Remember = 1” inside the Infofile.

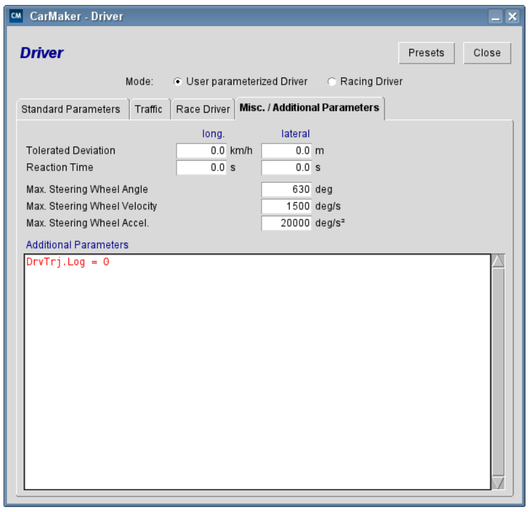
More information concerning Driver Adaption and tunable parameters can be found in CarMaker UsersGuide and Documentation for IPGDriver.

3.5 Parameterization

Following parameters can be used to parameterize the Driver Trajectory extension. Please note that parameters tuning the basic vehicle control behaviour are described in chapter "3.4 Tuning the Vehicle Control"

The parameters below allow a modification of internal mechanism, e.g. arguments provided to functions defined inside the extension. Some parameters are designed for debugging purposes.

The parameters are managed with a common prefix "DrvTrj.*". All these parameters can be applied under "CM Main GUI -> Parameters -> Driver -> Additional Parameters", as shown in Figure 3-2. The parameters are stored with the TestRun Infofile. If you want to edit them there, they can be found with the additional prefix "Driver." (e.g. Driver.DrvTrj.*).



Mode: [✓ User parameterized Driver] Racing Driver

	long.	lateral
Tolerated Deviation	0.0 km/h	0.0 m
Reaction Time	0.0 s	0.0 s
Max. Steering Wheel Angle		630 deg
Max. Steering Wheel Velocity		1500 deg/s
Max. Steering Wheel Accel.		20000 deg/s²

Additional Parameters

```
DrvTrj.Log = 0
```

Figure 3-2: Parameters in driver settings

DrvTrj.Log

This key is optional, default value is "1".

The parameters manages the basic logging. E.g. if the length of current trajectory is shorter than requested by IPGDriver, a message with detailed information will be shown in the CarMaker Session Log. By default, only first message will be displayed in the current trajectory during same simulation. By setting the value to "0" this mechanism is completely deactivated. A value of "2" means logging each circle. More detailed debugging logs can activated with a parameter below.

DrvTrj.TrjTrgt.nPntsMax

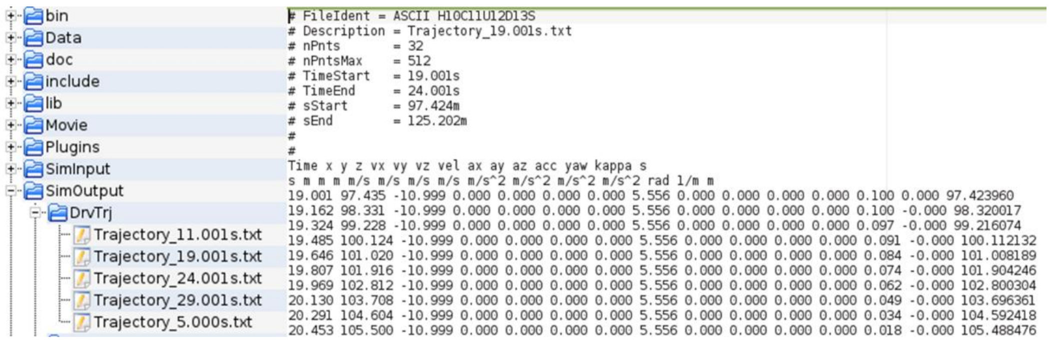
This key is optional, default value is "512".

It sets the maximum allowed data points of a single trajectory in the extension. Pay attention to the macro "USERTRJ_PNTS_USED" defined in the example code. This macro defines the current maximal points of a single trajectory from the user. "USERTRJ_PNTS_USED" should be smaller than "DrvTrj.TrjTrgt.nPntsMax". CarMaker executables has to be restarted after changing.

DrvTrj.Export

This key is optional, default value is “0”.

Once the user has complied a CarMaker executable with the Driver Trajectory extension, an additional directory (<CMProjDir>/SimOutput/DrvTrj/) is created where several information, e.g. the trajectories provided to IPGDriver during simulation is exported. “DrvTrj.Export = 1” activated the export. This only for debugging, and runs in CM Main Thread (problem with realtime condition)!



```
# FileIdent = ASCII H10C11U12D13S
# Description = Trajectory_19.001s.txt
# nPnts       = 32
# nPntsMax    = 512
# TimeStart   = 19.001s
# TimeEnd     = 24.001s
# sStart      = 97.424m
# sEnd        = 125.202m
#
#
Time x y z vx vy vz vel ax ay az acc yaw kappa s
s m m m m/s m/s m/s m/s m/s^2 m/s^2 m/s^2 m/s^2 rad 1/m m
19.001 97.435 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.100 0.000 97.423960
19.162 98.331 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.100 -0.000 98.320017
19.324 99.228 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.097 -0.000 99.216074
19.485 100.124 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.091 -0.000 100.112132
19.646 101.020 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.084 -0.000 101.008189
19.807 101.916 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.074 -0.000 101.904246
19.969 102.812 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.062 -0.000 102.800304
20.130 103.708 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.049 -0.000 103.696361
20.291 104.604 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.034 -0.000 104.592418
20.453 105.500 -10.999 0.000 0.000 0.000 0.000 5.556 0.000 0.000 0.000 0.000 0.018 -0.000 105.488476
```

Figure 3-3: Example of exported trajectory information

Figure 3-3 shows the export directory (“DrvTrj”) in a CarMaker project. Several text files about trajectories are exported in this example. In these files details about each trajectory are documented, e.g. time, arc length of trajectory etc. .

By default, the value of this key is set to “0”.

DrvTrj.Debug

This key is optional, default value is “0”.

This key allows user to have more information during simulation, such as data information about the first and last point of current trajectory. Set to value of this key to “1” to activate this function. Messages can be viewed in “CarMaker Main GUI → Simulation → Session Log”

4 Examples

The provided package already includes several examples how the Driver Trajectory extension can be used. The relevant code can be found in "<CMProjDir>src/DrvTrj_TrjCtrl.c". Currently two examples for different use cases exist. The uses cases can be selected via the optional parameter "DrvTrj.User.Mode" in "CM Main GUI -> Parameters -> Driver -> Additional Parameters", as described in chapter 3.5. Following values are currently supported

- "1" = "Dynamic": Generate trajectory dynamically while simulation is running
- "2" = "Static": Read static trajectory from file before simulation starts

By default, the mode "Dynamic" is used. The different modes are described below.

A single trajectory consists of several parameters, e.g. time, position etc. . Details about the trajectory structure and parameter information can be found in "DrvTrj_TrjCtrl.c" and "DriverTrajectory.h" of the delivered package.

4.1 Dynamic Trajectory generation at runtime

By setting the Infofile key "DrvTrj.User.Mode = 1" in the Drivers Additional Parameters the user trajectory mode will be set to dynamic mode, which means that the user trajectories are dynamically generated and provided to the Driver Trajectory extension during simulation. This mode shows the major use case and might be used as base to connect own modules, e.g. motion planning algorithms.

In the delivered TestRun examples several trajectory quantities are used to generate a simple sinus wave trajectory for demonstration (see "userTrj_Dynamic_Generator" function in "src/DrvTrj_TrjCtrl.c"). Time, vehicle positions with x/y coordinates, absolute velocity, yaw angle and curvature parameters are calculated with following formulas.

$$t_i = t_0 + i * dt; i \in [0\ 32) \tag{3-1}$$

$$x_i = x_0 + i * v_x * dt \tag{3-2}$$

$$y_i = y_0 + A * \sin(t_0 + i * dt) + y_{off\_set} \tag{3-3}$$

With these position and time information we're able to calculate the curvature of the trajectory, which is a very important parameter for the lateral control performance. For a plane curve given parametrically in Cartesian coordinates as $\gamma(t) = (x(t), y(t))$, the signed curvature is (Source: https://en.wikipedia.org/wiki/Curvature):

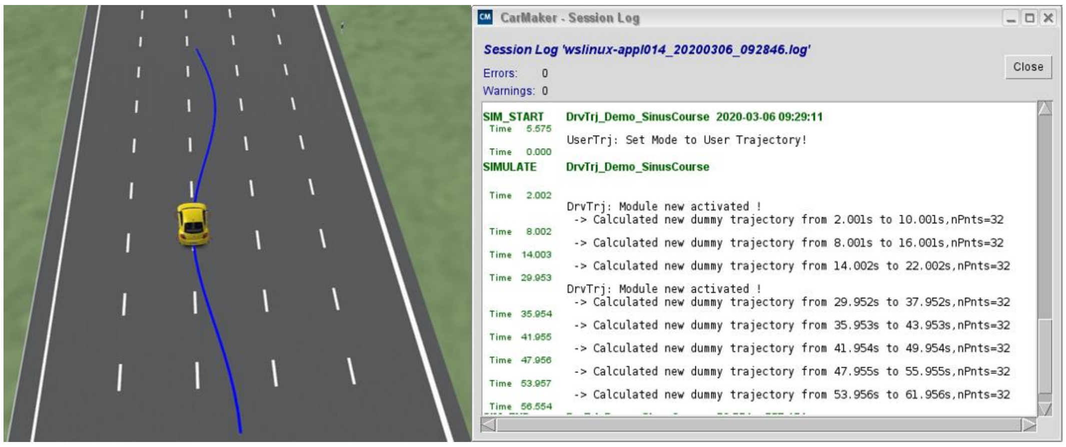
$$k = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{3-4}$$

The absolute velocity and the yaw angle of the vehicle are also parametrized in this extension in a simple way. If not parametrized, they would have a default value of 0.

$$vel_{abs} = |v_x| \tag{3-5}$$

$$yaw = 0 \tag{3-6}$$

It needs to be pointed out, that the trajectory is purely a simple trajectory for demonstration purpose. Thus, the quality of those trajectories is not perfect. The currently active trajectory is visualized in IPGMovie in blue (Figure 4-1).



CarMaker - Session Log

Session Log 'wslinux-appl014_20200306_092846.log'

Errors: 0
Warnings: 0

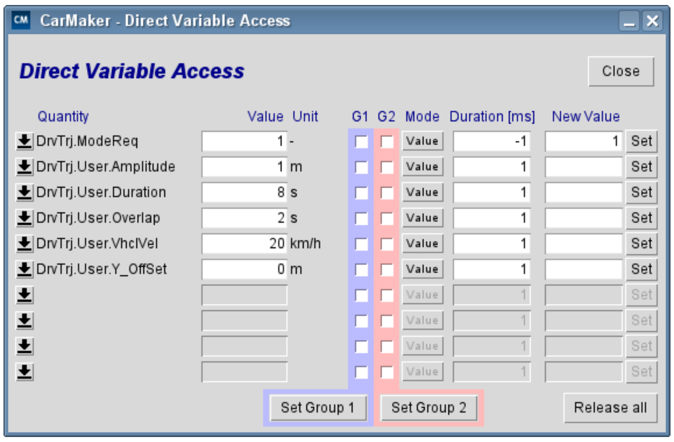
```
SIM_START    DrvTrj_Demo_SinusCourse  2020-03-06 09:29:11
 Time  5.575
             UserTrj: Set Mode to User Trajectory!
 Time  0.000
SIMULATE     DrvTrj_Demo_SinusCourse

 Time  2.002

             DrvTrj: Module new activated !
              -> Calculated new dummy trajectory from 2.001s to 10.001s,nPnts=32
 Time  8.002
              -> Calculated new dummy trajectory from 8.001s to 16.001s,nPnts=32
 Time  14.003
              -> Calculated new dummy trajectory from 14.002s to 22.002s,nPnts=32
 Time  29.953
             DrvTrj: Module new activated !
              -> Calculated new dummy trajectory from 29.952s to 37.952s,nPnts=32
 Time  35.954
              -> Calculated new dummy trajectory from 35.953s to 43.953s,nPnts=32
 Time  41.955
              -> Calculated new dummy trajectory from 41.954s to 49.954s,nPnts=32
 Time  47.956
              -> Calculated new dummy trajectory from 47.955s to 55.955s,nPnts=32
 Time  53.957
              -> Calculated new dummy trajectory from 53.956s to 61.956s,nPnts=32
 Time  56.554
```

Figure 4-1: Dynamically generated trajectory in IPGMovie and Session Log output

The configuration parameter, e.g. amplitude, are defined as CarMaker User Accesible Quantities and can be manipulated during the simulation, e.g. via “CarMaker GUI -> Application -> Direct Variable Access“ (Figure 4-2) or directly in the TestRun as Minimaneuver command. For example, changing the value of “DrvTrj.User.Amplitude” from 2 m (default) to 0 m during simulation will turn the sinus wave into a straight line.



CarMaker - Direct Variable Access

Direct Variable Access

Quantity	Value	Unit	G1	G2	Mode	Duration [ms]	New Value
DrvTrj.ModeReq	1	-	☐	☐	Value	-1	1
DrvTrj.User.Amplitude	1	m	☐	☐	Value	1	
DrvTrj.User.Duration	8	s	☐	☐	Value	1	
DrvTrj.User.Overlap	2	s	☐	☐	Value	1	
DrvTrj.User.VhclVel	20	km/h	☐	☐	Value	1	
DrvTrj.User.Y_OffSet	0	m	☐	☐	Value	1	
			☐	☐	Value	1	
			☐	☐	Value	1	
			☐	☐	Value	1	
			☐	☐	Value	1	

Figure 4-2: Manipulation of trajectory parameters for the dynamic mode via DVA

More details about using Direct Variable Access can be found in CarMaker User’s Guide in Chapter 12.3 “DVA: Online Manipulation of the Simulation”.

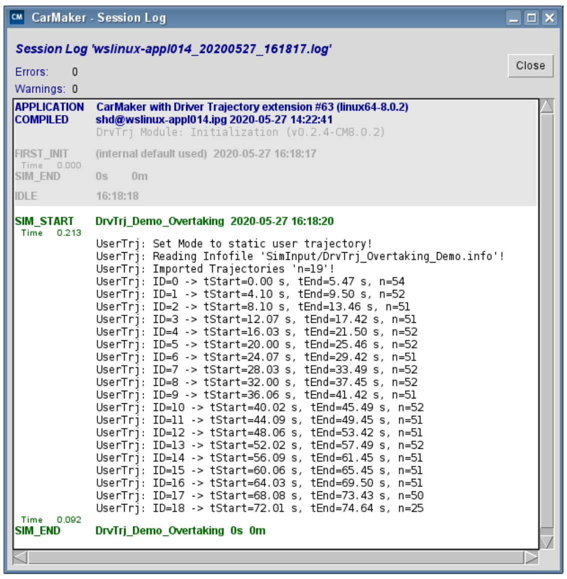
4.2 Static Trajectory from Infofile

By setting the Infofile key “DrvTrj.User.Mode = 2” the user trajectory mode will be set to static mode, which means the user trajectories are read into CarMaker at the beginning of simulation. An Infofile with example trajectories can be found in the “SimInput/DrvTrj_static_trajectory.info” of the delivered package. This Infofile was generated with a CarMaker Result file based on previous simulation. In order to generate this type of trajectory Infofile from erg-file please contact IPG.

The file path of trajectory Infofile can be parameterized via “CM Main GUI -> Parameters -> Driver -> Additional Parameters”, e.g.

```
DrvTrj.UserTrj.FPath = SimInput/<Name of Infofile>.info
```

The Infofile describes several trajectories that are read on the pre-processing phase of each simulation start. After successful initialization the trajectory information will also be displayed in Session Log (Figure 4-3).



Session Log 'wslinux-appl014_20200527_161817.log'

Errors: 0
Warnings: 0

```
APPLICATION   CarMaker with Driver Trajectory extension #63 (linux64-8.0.2)
COMPILED      shd@wslinux-appl014.ipg 2020-05-27 14:22:41
              DrvTrj Module: Initialization (v0.2.4-CM8.0.2)

FIRST_INIT    (internal default used)  2020-05-27 16:18:17
  Time  0.000
SIM_END       0s     0m

IDLE          16:18:18

SIM_START     DrvTrj_Demo_Overtaking  2020-05-27 16:18:20
  Time  0.213
              UserTrj: Set Mode to static user trajectory!
              UserTrj: Reading Infofile 'SimInput/DrvTrj_Overtaking_Demo.info'!
              UserTrj: Imported Trajectories 'n=19'!
              UserTrj: ID=0 -> tStart=0.00 s, tEnd=5.47 s, n=54
              UserTrj: ID=1 -> tStart=4.10 s, tEnd=9.50 s, n=52
              UserTrj: ID=2 -> tStart=8.10 s, tEnd=13.46 s, n=51
              UserTrj: ID=3 -> tStart=12.07 s, tEnd=17.42 s, n=51
              UserTrj: ID=4 -> tStart=16.03 s, tEnd=21.50 s, n=52
              UserTrj: ID=5 -> tStart=20.00 s, tEnd=25.46 s, n=52
              UserTrj: ID=6 -> tStart=24.07 s, tEnd=29.42 s, n=51
              UserTrj: ID=7 -> tStart=28.03 s, tEnd=33.49 s, n=52
              UserTrj: ID=8 -> tStart=32.00 s, tEnd=37.45 s, n=52
              UserTrj: ID=9 -> tStart=36.06 s, tEnd=41.42 s, n=51
              UserTrj: ID=10 -> tStart=40.02 s, tEnd=45.49 s, n=52
              UserTrj: ID=11 -> tStart=44.09 s, tEnd=49.45 s, n=51
              UserTrj: ID=12 -> tStart=48.06 s, tEnd=53.42 s, n=51
              UserTrj: ID=13 -> tStart=52.02 s, tEnd=57.49 s, n=52
              UserTrj: ID=14 -> tStart=56.09 s, tEnd=61.45 s, n=51
              UserTrj: ID=15 -> tStart=60.06 s, tEnd=65.45 s, n=51
              UserTrj: ID=16 -> tStart=64.03 s, tEnd=69.50 s, n=51
              UserTrj: ID=17 -> tStart=68.08 s, tEnd=73.43 s, n=50
              UserTrj: ID=18 -> tStart=72.01 s, tEnd=74.64 s, n=25
  Time  0.092
SIM_END       DrvTrj_Demo_Overtaking  0s  0m
```

Figure 4-3: Log information about static trajectories from trajectory Infofile

Pay attention to the activation time as described in chapter 3.3 while using this mode. The activation time of Driver Trajectory extension should be within the starting time of the first trajectory given by the Infofile. In this example the start time of the first trajectory is 0s, that means the Driver Trajectory should also be activated at beginning of the simulation e.g. via the Minimaneuver command “Eval first(Time > 0) ? DrvTrj.ModeReq = 1”

4.3 Visualize Trajectory in IPGMovie

By using *.tclgeo file in IPGMovie some dynamic animated objects can be displayed during the simulation. Quantities defined in C-Code of the CarMaker executable can be manually subscribed and used for individual visualization.

The corresponding “DriverTrajectory.tclgeo” file can be found at "<CMProjDir>/Movie/” and loaded via “CarMaker Main GUI → Parameter → Scenario / Road → 3D Preview → Visualization Parameters -> TclGeo File”.

More information about IPGMovie and tclgeo can be found at “CM Main GUI -> Help ->IPGMovie User’s Manual”.

In this example there is the possibility to enable different visualizations with some parameters and text information with different colours inside the TclGeo file. The visualization of certain lines can be deactivated by setting the value to 0.

When simulation is running it can be analysed how IPGDriver follows the target trajectory. The blue line represents the currently provided trajectory, while the yellow line shows the history up to the target point on trajectory where the PoI should be at current time. The green line visualizes the final movement of the vehicles PoI (Figure 4-4).

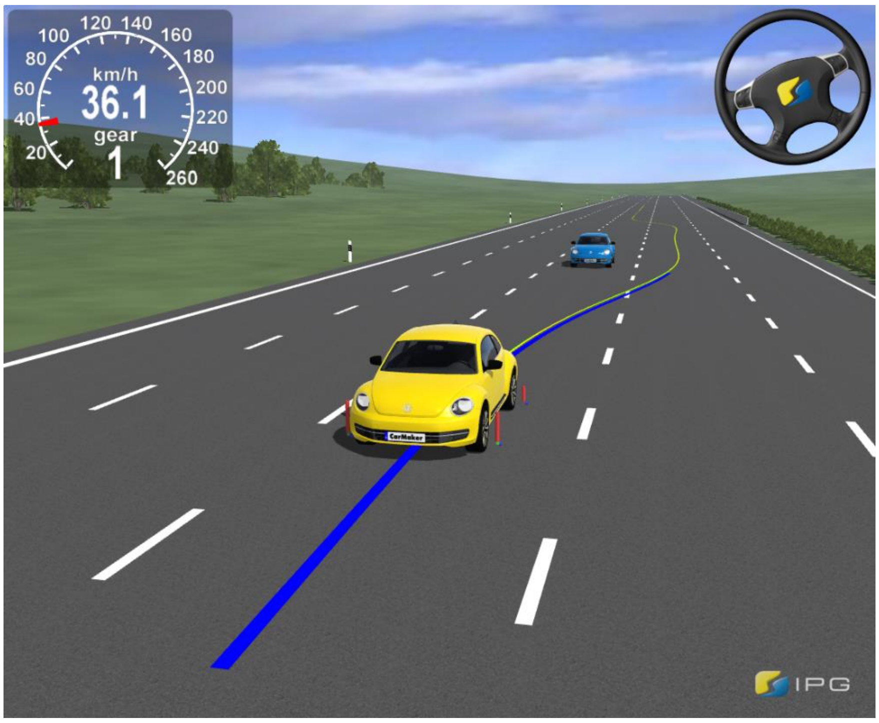


Figure 4-4: Visualization of Driver Trajectory extension in IPGMovie

5 Release History

5.1 Version 0.2.4

- Trajectories can now also be dynamically imported during simulation.
- Visualization in IPGMovie has been enhanced.
- Exported files are now located in < CMProjDir >/SimOutput/DrvTrj/.
- New Log user keys has been introduced.

5.2 Version 0.2.3

- Extern trajectories can be imported in CarMaker via Infofile mechanism.
- Visualization of vehicle position and target trajectory position for IPGDriver in IPGMovie.