**CoInCar-Sim :**

Focusses on Co-operative Motion planning (Open Source Multi Vehicle Simulation Framework)

1. Background :

* Data Sets Used:

KITTI and Cityscapes (these data sets also provide ground truth)

* The framework focuses on the interaction of traffic participants by enabling the simultaneous simulation of several instances of one or more automated vehicles.
* Existing Solutions : IPG CarMaker, TASS PreScan [5], VIRES VirtualTestDrive, dspace ASM, SiVIC - but these don’t support multi vehicle simulation, they just focus on Vehicle Dynamics.
* Carla serves the purpose but its based on proprietary “Unreal Engine 4”.

2. High level Framework design:

* multiple instances of the same behavior generation algorithm alongside each other is a basic requirement.
* And as next step, different algorithms and human drivers will share the road. To test these situations,

common interfaces, including the frequency with which they provide information should be defined.

* These interfaces should support both implicit and V2X communication.
* Framework should allow human drivers to control a vehicle in simulation frame. For testing multiple confidential algorithms, the interconnection of multiple vehicles over network (e.g. TCP/IP) should be facilitated.
* **Replicability**: The co-simulation of different vehicles/Algorithms should be recordable in order to allow downstream in-depth analysis of corner cases.

3. Concept of Framework :

* ROS as underlying middleware. It provides an interface definition language, it allows for communication via network, and it supports recording all data sent via the interfaces. Also it provides usability and facilitate visualization.
* The simulation framework must model all relevant interaction between the system under test and its environment.

3.1 Environmental Model :

* A environment model should support wrapper for all the interfaces to communicate to the environment.
* In order to both maximize the performance and minimize the maintenance effort, the environment model is kept as lean as possible. Wiz. all the interfaces directly impacting the behaviour of the motion planning should be modelled and the rest should be mocked up.
* The time management provides the simulation time, which can be real time, slower or faster.
* The localisation management provides an array of object states, each containing the current position of the object.

3.2 Interfaces (Ros messages) :

* The two major messages of the localisation management are the ObjectState, respectively an array of those, and the DeltaTrajectory.
* The delta trajectory depicts the future motion of an object, It consists of delta poses that each describe the delta between the current pose and the desired pose, along with the time difference in which the desired pose is to be reached.
* We defined an additional message called AdditionalHeaderForCommunication. It contains information about who sent the message, and from which pose the message was sent.

3.3 Wrapper

* sensor and actuator characteristics are modelled in the wrapper, which allows to simulate the effects of those characteristics on the system under test.
* The characteristics that can be modelled include for example uncertainty and sensor range, but also reduced friction in the motion model.

3.4 Scenario Definition

* Co-operative scenarios cannot be comprehensively defined, since they are largely affected by the objects’ interaction.
* The use of lanelet maps(map specification as well as an editing tool chain) for behaviour generation and planning is used.
* define a place and map for the simulation as starting point.
* common map projection must be utilised when using real places and maps of Cartesian coordinates. We decided to use Universal Transverse Mercator (UTM).

3.5 Usage

* The Java OpenStreetMap Editor (JOSM)7 is used to determine a path of an object created by different ROS nodes. An initialisation module, provided with the framework, transforms this path and a desired initial velocity into a trajectory.

3.6 Visualisation

* For visualising the scenarios, we make use of the 3D visualizer RViz that is provided with ROS. RViz can be extended by plugins, visualising custom ROS messages.
* Plugins for the lanelet map, the object states, as well as the desired motion, depicted by the delta trajectory has been provided.

3.6 Use case and Conclusion

The motion planner generates cooperative trajectories for a single vehicle in mixed traffic, without explicit communication is simulated. Similarly with multiple vehicles.

Plug and Play : . It facilitates to instantiate several objects with SAE4+ capable behavior generation algorithms, allowing to replace widely-used reactive driver models that do not allow for development and evaluation of cooperative planning algorithms.

Different objects can either use different behavior generation algorithms or different instances of the same algorithm. Perception and control algorithms as well as V2X communication algorithms are not in-the-loop, but their effects on the vehicles’ behavior can be evaluated. As each object has an individual wrapper, different perception, control and communication setups can be tested simultaneously

**Robot Operating System: A Modular Software Framework for Automated Driving**

Background : This paper analyses the requirements for software frameworks

**1. Design Requirements :**

* **Modularity and Extensibility** : In order to achieve modularity, one needs to define common interfaces, which the user can use from a high level of abstraction. Given a modular SWF with well-defined interfaces, functionality can be extended and exchanged easily.
* **Performance, Simulation and Debugging, Fault Tolerance and Security, Usability and Support.**

**2. AUTOMOTIVE SOFTWARE FRAMEWORKS**

**2.1 Real-time Database for Cognitive Vehicles**

* The Real-time Database for Cognitive Automobiles (KogMo-RTDB) was used as a data exchange provider within our team’s research vehicles.
* KogMo-RTDB is accompanied by tools to record and playback data objects.
* The proprietary EB Assist Automotive Data and TimeTriggered Framework (ADTF) is the mostly used development and testing environment for Advanced Driver Assistance Systems (ADASs).
* It is designed as a real-time system with distributed process chains and supports synchronous and asynchronous data processing.

3 **Node Pipelining**

* For certain tasks in automated vehicles, computed output data at a constant, predefined rate is beneficial. For example, in vision-based localisation with input images recorded at 15Hz, the position updates shall generally be available at 15Hz as well.
* To meet the desired update frequency of the vehicle motion estimations, the individual steps are split into separate nodes and chained such that vehicle motion results are available at 15Hz with a dead time equal to the depths of the processing pipeline.

4. Simulation : For simulation, the modular messaging structure of ROS allows to interchange the source of incoming data.

* If user input is needed within the simulation, the tool rviz4 provides a set of interactive markers that can be used for positioning and orienting obstacles, for example.
* For more advanced, whole system simulations or even regression testing, ROS integrates seamlessly with Gazebo. Gazebo is a powerful 3D simulation engine with complete dynamic and kinematic physics.
* **Gazebo itself is a standalone software, but happens to use the same messaging interface that ROS is build upon, since it originates from the same developers.**

5. **System Diagnostics and Monitoring**

* ROS provides an easy to use and integrated infrastructure for sending diagnostic messages to a central diagnostic manager that displays the system health in a traffic light color scheme to immediately pinpoint critical components or if data integrity is at risk.

6. **Coordinate Systems and Transformations**

* Inherent to multi-sensor systems is the definition of multiple coordinate systems, e.g. camera, vehicle or global coordinate system, and the frequent need to transform data from one coordinate system into another one.
* To reduce hard-to-debug transformation bugs, ROS provides a time-based transformation service that keeps track of extrinsic transformations between coordinate systems tagged by a timestamp stating the transformation’s validity.
* To associate message data with coordinate systems, most messages embed the coordinate system name into its payload by using a standardized ROS message header.