Customer-specific sensor models – Integration and Validation
Dr.-Ing. Christopher Wiegand, Product Manager Modeling & Scenarios, Munich, November 20, 2019
Integration of customer-specific sensor models

Simulation is the key to validate function for autonomous driving
Integration of customer-specific sensor models

Sensor Environment Model
Vehicle and Vehicle Dynamics Model

System under Test and Sensor Model

Sensor Frontend
Preprocessing
Raw data
Detection, Data Proc.
Target list
Object tracking
Object list
Application logic (Trajectory planning, Motion control)
Network Management
Sensor integration options

- **Technology independent options**
  - Ideal ground truth and probabilistic based information
  - Part of ASM (calculated on CPU)

- **Physics-based options**
  - More related to the measurement principle of a sensor
  - „Raw data injection“
  - Part of MotionDesk SensorSim (calculated on GPU)

- **Simulation over-the-air**
  - Test Bench with real Sensor ECU
Sensor integration options

- **Technology independent options**
  - Ideal ground truth and probabilistic based information
  - Part of ASM (calculated on CPU)
- **Physics-based options**
  - More related to the measurement principle of a sensor
  - “Raw data injection”
  - Part of MotionDesk SensorSim (calculated on GPU)
- **Simulation over-the-air**
  - Test Bench with real Sensor ECU

Customer-specific sensor models

System under Test and Sensor Model

- Preprocessing
- Raw data
- Detection, Data Proc.
- Target list
- Object tracking
- Application logic (Trajectory Planning, Motion Control)

Network Management

Restbus

Insert raw data
Insert target list
Insert Object list
Sensor integration options

- **Technology independent options**
  - Ideal ground truth and probabilistic based information
  - Part of ASM (calculated on CPU)

- **Physics-based options**
  - More related to the measurement principle of a sensor
  - „Raw data injection“
  - Part of MotionDesk SensorSim (calculated on GPU)

- **Simulation over-the-air**
  - Test Bench with real Sensor ECU
Sensor integration options

- **Technology independent options**
  - Ideal ground truth and probabilistic based information
  - Part of ASM (calculated on CPU)

- **Physics-based options**
  - More related to the measurement principle of a sensor
  - „Raw data injection“
  - Part of MotionDesk SensorSim (calculated on GPU)

- **Simulation over-the-air**
  - Test Bench with real Sensor ECU

System under Test and Sensor Model

---

**Insert raw data**

**Insert target list**

**Insert Object list**

**Restbus**
Integration of customer-specific sensor models

Sensor Environment Model

Vehicle and Vehicle Dynamics Model

RTMaps, NVIDIA Drive AGX, ROS, etc.

Customer-specific Sensor Model

System under Test

Sensor Frontend

Preprocessing

Raw data

Detection, Data Proc.

Target list

Object tracking

Application logic (Trajectory planning, Motion control)

Network Management

GPU: API for Customer Sensor Model Integration

CPU: Virtual Environment Sensor Interface API
Integration of customer-specific sensor models

Sensor Environment Model
Vehicle and Vehicle Dynamics Model

System under Test and Sensor Model

Sensor Frontend
- Preprocessing
- Raw data
- Detection, Data Proc.
- Object tracking
- Object list

Application logic (Trajectory planning, Motion control)

Network Management

GPU: API for Customer Sensor Model Integration
FPGA: Environment Sensor Interface
CPU: Virtual Environment Sensor Interface API
Integration of customer-specific sensor models

Sensor Environment Model

Vehicle and Vehicle Dynamics Model

System under Test and Sensor Model

Sensor Frontend

Preprocessing

Raw data

Detection, Data Proc.

Target list

Object tracking

Object list

Application logic (Trajectory planning, Motion control)

Network Management

Open Simulation Interface (OSI) – ASAM

FMU

GPU: API for Customer Sensor Model Integration

FPGA: Environment Sensor Interface

CPU: Virtual Environment Sensor Interface API
### Integration of customer-specific sensor models

- **OSI: Very short explanation**

  - **OSI::GroundTruth**
    - <Vehicle>
    - <MovingObject>
    - <TrafficSign>
    - <...>
  
  - **OSI::SensorView**
    - OSI::RadarSensorView
    - OSI::LidarSensorView
    - OSI::CameraSensorView
    - ...
  
  - **OSI::SensorData**
    - OSI::SensorView
    - OSI::FeatureData
    - <DetectedObject>
    - <...>
  
  - **OSI::SensorViewConfiguration**
    - <Identifier>
    - <MountingPosition>
    - <xxxViewConfiguration>
    - <...>

- Represents ideal world
- Contains all objects within the combined range of all sensors
- Earth coordinates / global coordinate system

- Derived from GroundTruth
- Contains only objects within the sensor field of view (FOV)
- Local sensor coordinate system
- May contain GT w.r.t. sensor FOV
- May contain low-level data, e.g. Radar reflection list

- Output of sensor models
- May contain detected objects, e.g. vehicle, traffic sign
- May contain feature data (low-level data before object hypothesis and tracking)

- Configuration message of sensor
- Allows automatic configuration
Integration of customer-specific sensor models – OSI::GroundTruth

- ASM supports OSI::GroundTruth (v3.0.1 and v3.1.2) for MiL, SiL, HiL (Scalexio)
- ASM OSI Interface currently used at different customers (MiL, SiL, HiL)
- GroundTruth Content:
  - Stationary Objects
  - Moving Objects
  - Traffic Signs
  - Traffic Lights
  - Lanes
  - Lane boundaries
  - Additional Information (Environmental Conditions, Time Stamp, Version, etc.)
GroundTruth Content – Moving Objects

- Moving Objects
  - Type
  - Position, Velocity, Acceleration
  - Orientation, Orientation-Rate
  - BoundingBox
  - Projected Contour
  - Light states
  - Driver_ID
  - Lane Mapping
  - ...
GroundTruth Content – Stationary Objects

- Stationary Objects
  - Type
  - Position
  - Orientation
  - BoundingBox
  - Projected Contour
  - Lane Mapping
  - ...

[Image of a 3D simulation of a street scene with stationary objects marked by bounding boxes]
GroundTruth Content – Lanes

- **Lanes**
  - Road-Lanes
  - Junction (each junctions is interpreted as one single lane)
  - Unique Lane ID
  - Successor / Predecessor IDs
  - Neighbor IDs
  - Lane boundary IDs (left, right, free)
  - *(Centerline (not clearly defined))*
Lane boundaries

(= Elements limiting Lanes)
- Lines, Guardrails, Edges, etc.
- Point Sequences (x, y, z)
- Consider Start and End position of Dashed-Line Elements
GroundTruth Content – Traffic Signs / Traffic Lights

- **Traffic Signs**
  - Main and Supplementary Signs
  - OSI Classification
  - Base Information (Position, Orientation, Box, Lane, etc.)

- **Traffic Lights**
  - Single Light-Elements
  - OSI Classification
  - Base Information (Position, Orientation, Box, Lane, etc.)
Integration of customer-specific sensor models – OSI::SensorView

Physics-based Sensor Models

- OSI Standard for SensorView is not fixed
- dSPACE supports the standard
Integration of customer-specific sensor models – VeloView

Customer-specific format and protocol

- Data can be processed directly on GPU
- Allows to emulate lidar devices
Integration of customer-specific sensor models

Sensor Environment Model

Vehicle and Vehicle Dynamics Model

Ideal Detection List

Channel Impulse Response

Customer-specific Radar Model

System under Test

Radar Frontend

Preprocessing

Raw data

Detection, Data Proc.

Target list

Object tracking

Object list

Application logic (Trajectory planning, Motion control)

Network Management

GPU: API for Customer Sensor Model Integration
dSPACE Ideal Detection-List

Detection-List from HELLA V-ECU
Summary

Integration

- dSPACE simulation environment
  - provides interfaces for customer-specific sensor models.
  - supports open standards, e.g. Open Simulation Interface (OSI)
How to validate customer-specific sensor models?

![Diagram](image-url)
How to validate customer-specific sensor models?

1. Measurements

2. Scenario Generation

3. Parameterization (e.g. Vehicle Dynamics, Environment, Sensors, etc.)

4. Simulation

5. Model Validation

Booths:
- Booth 12: Measurements
- Booth 13: Scenario Generation
- Booth 14: Parameterization
- Booth 14, 16, 18, 19: Simulation
- Booth 22
- 
- 
- 
- 
- 

Model validation as an essential part of the validation toolchain

- **Replay** of measurement data for simulation system verification
- **Automatic comparison** of data replay results with closed loop simulation of generated logic scenarios
- Validated scenarios as an input for scenario-based testing
Simulation parameter: Radius: 0.05 m, Frequency: 77 GHz, θ-step size: 0.1°, Triangles: 98.

Radar Model Validation

Trihedral Corner Reflector

Calculated RCS values by dSPACE software (red) agree well with data of full wave calculations (blue).

Reference plot by using full wave simulation (Multilevel Fast Multipole Algorithm (MLFMA)) of same geometry.
Calculated RCS values by dSPACE software (green) agree well with measured (full line) and simulated data (dashed line).

Simulation Parameter: side length = 0.2286 m, Frequency: 18 GHz, θ-step size: 0.1°, Triangles: 1536.

Radar Model Validation

Dihedral corner reflector

Calculated RCS values by dSPACE software (green) agree well with measured (blue) and via momentum method simulated data (red).

Summary

Sensor Model Integration
- dSPACE simulation environment
  - provides interfaces for customer-specific sensor models.
  - supports open standards, e.g. Open Simulation Interface (OSI)

Sensor Model Validation
- Iterative process along the development
- Based on state-of-the-art research results
- Real test drives (measurements) are needed
With us, autonomous driving gets more drive.