Advanced Driver Assistance System Testing using OptiX

NVIDIA GTC 2012, San Jose
Advanced Driver Assistance System Testing using OptiX

Erwin Roth
Robotics and Embedded Systems, Technischen Universität München

Tugkan Calapoglu, Holger Helmich
Vires Simulationstechnologie GmbH, Bad Aibling

Kilian v. Neumann-Cosel
Automotive Safety Technologies GmbH, Ingolstadt

Marc-Oliver Fischer
Department for Vehicle Safety Sensor Algorithms, AUDI AG, Ingolstadt

Andreas Kern
Audi Electronics Venture GmbH, Ingolstadt

Andreas Heider, Alois Knoll
Robotics and Embedded Systems, Technische Universität München
Agenda

► Motivation, requirements and goals

► Integration of OptiX into the Vires Virtual Test Drive simulation software

► Advanced material and emitter data descriptions

► Example sensor model implementations

► Model validation and verification process

► Summary and Outlook
Motivation

Growing challenges for testing new Advanced Driver Assistance Systems (ADAS)

- Increased number of comfort-, energy-management- and safety-related functions
- Growing dependency of ADAS-functions on multiple perception sensors
- Difficulties to record reproducible sensor data for real world scenarios

Forecast: 300% increase in shipped ADAS units [Mio.]

Source: iSuppli
Main Objectives

- Support ADAS testing with computer simulations for **realistic multi-sensor data computation**
- Validated sensor models as parts of an integrated vehicle and environment simulation system
- Enable closed-loop simulations in Hardware- and Software-in-the-loop testbeds
- Reproducibility of test scenarios for a wide range of environment and traffic conditions

- Early evaluation of new sensor concepts and ADAS functions
- Increased test space coverage by combining real and virtual test drives
Multi-Sensor Simulation Environment Objectives

- **Simultaneous execution** of multiple perception sensor emulators with **realistic distortion effects**

- Share a common simulation infrastructure for sensor data consistency
  - Scenario description
  - Object, material and emitter (light) data
  - Communication
  - Configuration

---

E. Roth, T. Calapoglu, et al.
Sensor Emulator Requirements - Architecture

- Support the same data formats and interfaces as the real sensor

Simplified integration into the existing ADAS development and testing process
Sensor Emulator Requirements – Simulation Variants

- Support various simulation types in the existing ADAS test toolchain

Toolchain for the integrated development and testing of ADAS functions

- Software-in-the-loop
  - slower / faster than real-time

- Driver-in-the-loop
  - soft real-time

- Vehicle-in-the-loop
  - hard real-time

- Hardware-in-the-loop
  - hard real-time
Sensor Emulator Requirements – Level of Realism

- Extensible architecture regarding refined models for a higher level of realism

- Configurable approximation accuracy and distortion levels with a single consistent model

<table>
<thead>
<tr>
<th>Sensor Emulation Data Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ideal</strong></td>
</tr>
<tr>
<td>Algo. validation (SIL, faster than real-time)</td>
</tr>
</tbody>
</table>
Sensor Emulator Requirements – Physics

- Particle-, ray- and wave-based physical measurement methods shall be approximated

- Physics-oriented modeling of
  - sensor data acquisition process
  - related systematic and stochastic distortion effects
  - material, surface and emitter properties
Advanced Material and Emitter Description

Multi-modal sensor simulation requires extended material and light source (emitter) descriptions

- Each material is identified by a unique ID
- Covering also non-visible light spectrum, e.g. 300 – 1000nm
- Holding meta data for material/emitter classification, lookup, ...
- Storage of physical properties in form of scalars and textures
- Support for existing measurement techniques and material standards incl. accuracy information
- Material data records must be extensible
Integration of OptiX and VTD
Virtual Test Drive

- **VTD core** = simulation environment
  - 3D rendering (image generator)
  - traffic and scenario
  - sound
  - mockup interfaces
  - record/playback
  - event and data handling
  - content creation
  - management of custom modules

- **VTD dev** = development environment
  - interfaces for
    - run-time data (Run-time Data Bus – RDB)
    - event / control data (Simulation Control Protocol – SCP)
    - sensor development (using Image Generator v-IG)
  - module development via
    - library
    - C++ API
VTD – Content Creation Toolchain

Content Creation
- IOS (Project Edit Mode)
- Scenario Editor
- ROD (Road Designer)
- Project
- Scenario Description
- Road logic and Geometry
- Visual Database

Runtime
- IOS
- Traffic
- Vehicle Dynamics
- v-IG (3D renderer)

TaskControl

Content data
Real-time data
v-IG

- Open Scene Graph (and OpenGL) based 3D renderer
- Part of VTD but also available as a standalone renderer
- Provides an API
- Used in driving, train and flight simulators
- Used in sensor simulation applications

Sensor image for hardware-in-the-loop simulator (OpenGL)

Standard day scene (OpenGL)

HDR night scene with wet road (OpenGL)
OptiX plug-in

- Conversion of OSG scene to OptiX scene
  - Geometry, Materials
- Synchronizing OSG/OptiX scenes
  - Animations, LOD, Lights
- Post Processing
- Real-time data transfer

- C++ API provided for customization
  - New camera models with Cuda/C++
  - Different buffer formats, multiple output buffers
  - Custom light sources
  - Building post processing pipelines
- Adding/editing materials
Creating the OptiX node graph

- v-IG loads the scene and creates an OSG scene graph
- OptiX plug-in translates the OSG scene graph to an OptiX scene graph
- OptiX specific optimizations during translation
- Some objects (e.g. Vehicles) loaded and deleted in run-time
Material management

- XML material definitions
- Grouped according to wavelength and/or sensor type

• Associates materials with objects
• Grouped according to wavelength and/or sensor type

- Identified by textures or ID’s
- v-IG assigns the materials to OSG scene graph nodes
- OptiX plug-in creates OptiX materials and puts into the material buffer
Material Management

- Material definitions in XML

- Common materials for rasterizer and ray-tracer
  - Shader params will be GLSL uniforms in rasterizer and OptiX variables in ray-tracer
  - Rasterizer loads GLSL programs, OptiX ptx files

- New materials can be derived from existing ones

Sample material declaration

```xml
<Material name="Audi_PhantomBlack_RT">
  <GeneralParams ambient="0.01 0.01 0.01 1.0"
    diffuse="0.02 0.02 0.02 1.0"
    specular="1.0 1.0 1.0 1.0"
    emissive="0 0 0 1" shininess="100" />
  <ShaderParams>
    <Param type="vec4" name="u_genericConfig" value="0.2 0.5 1 1" />
  </ShaderParams>
  <FragmentShader file="../data/Shaders/vehicleBodyFrag.glsl" />
  <VertexShader file="../data/Shaders/vehicleBodyVert.glsl" />
  <OptiXHitProgram file="../data/Cuda/vehicleBody.ptx" />
</Material>
```

Copy and override material properties

```xml
<Material name="Default_Rim_RT" copy="Default_Rim">
  <OptiXHitProgram file="../data/Cuda/chrom.ptx" />
</Material>
```
Synchronization of scenes

- v-IG manipulates OSG scene graph nodes (e.g. DOF’s) for animations
- LOD nodes are automatically updated by OSG
- OptiX plug-in monitors function nodes and synchronizes their OptiX counterparts

**Simulation**

Animation routines

- Camera position
- Positions and orientation of objects
- Light positions
- Etc.

**OptiX node graph**

- LOD
- Group
- DOF
- Drawable
- GeometryGroup

OptiX plug-in synchronizes the graphs continuously

**OptiX scene graph**

v-IG manipulates the scene graph to animate the objects

v-IG

Animation data over network

Lights and emitters

- v-IG creates and manages a list of lights
  - Some lights are generated automatically with the information stored in the terrain and models (car headlights, street lamps)
  - Communication protocol allows for creating and controlling lights externally
- OptiX plug-in synchronizes OptiX light buffer with v-IG lights
- Lights can represent any type of emitter

Animation data over network
- Positions and orientation of objects
- Light positions

Simulation

v-IG

Update lights that are created and managed by external applications

Update lights that are parts of simulation entities (e.g. Car headlights)

v-IG synchronizes the lists continuously

OptiX plug-in

OptiX light buffer
Post processing

- Rendered buffers can be fed into v-IG post processing pipeline
  - Programmable through API
  - Post processing with GLSL shaders
  - 32 bit floating point

- Motivations
  - Bloom
  - Noise
  - Tone mapping
  - Etc.

OptiX plug-in

- OptiX output buffer
- 32-bit float OpenGL texture
- Post processing step 1
- ... 
- Post processing step n
Real-time data transfer

- Rendered buffers are made available to external applications
- Shared memory or network
- Producing data for hardware-in-the-loop and software-in-the-loop simulations

![Diagram of data transfer](image)

- RAM
  - v-IG + OptiX
- Shared Memory
- User application
- Other workstation(s) or sensor hardware
- Network
- Video RAM
  - 32-bit float OptiX output buffer
  - 32-bit OpenGL texture
  - 32-bit OpenGL texture
- Post processing
Sensor Model Examples

using OptiX + VTD
Photonic Mixing Device (PMD) Sensor Model

- The PMD-sensor uses the Time-of-Flight principle for measuring intensity and depth data of a 3D-scene with modulated infrared (IR) light

- Important systematic and stochastic distortion effects
  - non-ambiguity range
  - extraneous light sensitivity
  - “flying pixels”, motion blur

- Three-step sensor data simulation
PMD Sensor Model

- Approximation techniques in the current PMD sensor emulator
  - Multiple rays per pixel with stratified sampling
  - Simulate the angle-dependent emission characteristics of
    the modulated IR-light source based on Radiometry measurements
  - Phong reflection model in combination with
    measured IR-material reflection values
Ultrasonic Sensor Model

- Currently: ideal acoustic wave propagation model

- Modeling requirements
  - Computation of primary-, secondary and cross-echo
  - Efficient computation of up to 20 ultrasonic sensors on a single GPU
  - Consideration of target object material class (e.g. vegetation)

16 circularly-arranged ultrasonic sensor „depth maps“ simultaneously rendered with OptiX
Sensor Emulator Validation and Verification Stages

**Real Sensor System**
- Raw data Processing
  - Objectification Algorithms
- Sensor test bench
- Real test drive
- Issue recordings DB

**OptiX + ADTF Sensor Emulator**
- Raw data Generation
  - Objectification Algorithms
- Real-time capable
  - x-in-the-loop variants

**MATLAB + ADTF Sensor Emulator**
- Raw data Generation
  - Objectification Algorithms
- Non-real-time, high fidelity
  - MATLAB ray-tracer for raw data generation

**1. Validation**
Prototypical Testing

**2. Verification**
Testing with real sensor data

Comparison based on reference data & virtual scene generation

Comparison based on static + dyn. description of "analytical scenes" using XML

E. Roth, T. Calapoglu, et al.
Sensor Emulator Validation and Verification Toolchain

**MATLAB**
- offline data comparison
  - synthetic data
  - real data

**Simulink**
- Master Control Unit
- Closed-loop Model

**ADTF**
- Sensor Objectification Algorithms
- ADAS Algorithms
- Real sensor data recordings incl. reference- and scenario-data

**v-IG**
- Sensor-Plugin
- OptiX - API

**Vehicle Dynamics**
- Traffic
- ... (indicates more categories)

**Shared Memory or Network**

**TUM**

**VIRE**

**E. Roth, T. Calapoglu, et al.**
Summary

► We showed our approach for supporting ADAS algorithm and function testing by using Virtual Test Drive and OptiX for multi-sensor data simulation

► Related requirements and implemented concepts for realistic multi-sensor simulation
  ► Physics-oriented sensor modeling using OptiX
  ► A common sensor-model simulation infrastructure
  ► Advanced material and emitter specifications
  ► Validation and verification process

► Ray-tracing with OptiX seems to be a reasonable platform. However, we are just at the beginning ...
Outlook

Challenges to be tackled in the future regarding ...

- A standard for multi-spectral material and emitter specifications
  - Simulation software independent description and identification scheme
  - Physical property handling of materials and emitters, e.g. for wavelengths 300 – 1000nm
  - Support for different measurement data formats and standards

- OptiX
  - Support for large scenarios (1000x of objects, 100x materials, 10x sensor models, ...)
  - Improved multi-GPU scalability for 60 Hz and higher
  - Improved OptiX debugging, profiling and optimization tools
Thank you very much.

Get in touch with us, if you are also using OptiX for sensor simulation!

Contact:
Erwin Roth, Technische Universität München
erwin.roth@in.tum.de
Tugkan Calapoglu, Vires Simulationstechnologie GmbH
tugkan@vires.com
Why was NVIDIA OptiX selected?

- Since Intel’s Larrabee was never released ;-)

- Programmability and Flexibility of OptiX’s Ray Tracing pipeline
  - Customizable Ray Tracing pipeline
  - Focus on mathematical model rather than 3D programming
  - Many core, multi-GPU scalability
  - Availability for different platforms

- AUDI was already using the Virtual Test Drive (VTD) simulation system
  - We decided to extend the OpenSceneGraph-based 3D-renderer of VTD with an OptiX-plugin
  - Allows us to reuse most of the existing rendering and simulation infrastructure

[Source: NVIDIA]
XML-Scene Data Interchange Format

- File format for OptiX node graphs
  - XML based format
  - Easily readable and editable
  - C++ library for loading/saving
  - Can be integrated to other software
- Motivations
  - Sensor model validation
  - Debugging
Integrated Material Handling

**CAD System**
- Design of geometry
- Initial material assignment

**PDM System**
- Central data management
  - Geometry Data
  - + Global Unique Material ID (GUMID)

**Advanced Material Reference DB with GUMID**

**Material translator**

**Organization-wide Advanced Material + Emitter DB with GUMID**

**GUMID concept based on Gerd Sussner, RTT AG**

**Manufacturing**

**High End Visualization**
- RTTDeltagen, Maya, ...

**Simulation**
- VTD, FEM-crash, -thermo, ...

**Material Measurement Labs**

**Material translator**
Multi-Sensor Simulation Material pre-processing Pipeline

- Geometry data incl. geometry IDs
- Road-surface model
- High polygon vehicle models
- Optimized polygon vehicle meta models
- Test Scenario

Level: Sim. Environment
- MA assignment table

Level: Organization
- MA assignment table

Level: Test Scenario
- MA assignment table

Level: Sensor Model
- MA assignment table

Offline pre-processing stage
- Simulator Environment

Offline pre-processed material + emitter data -> optimized for simulation engine

Organization-wide
- Advanced Material + Emitter DB

Simulation
- Vendor Environment
- Advanced Material + Emitter DB

Advanced Material + Emitter Reference DB

- E. Roth, T. Calapoglu, et al.
Advanced Emitter Description

- **Simulating interference effects on sensors** requires models of ego- and extraneous-emitters

- Examples for emitters:
  - Vehicle headlights, traffic lights, street lamps
  - Emitters of active sensors (RADAR, Infrared light source, ...)
  - Car2X-Transmitters

- Related emission characteristics should be stored in physically measurable SI units using Radiometry in order to not cover the visible light spectrum only

- The specific emitter characteristics should be stored in an organization-wide database with a unique emitter ID