Advantages of Physically Based Rendering for Autonomous Driving Validation

Johannes Günther
johannes.guenther@intel.com

Oliver Grau
oliver.grau@intel.com

Isha Sharma
isha.sharma@intel.com

Björn Brücher
bjoern.bruecher@intel.com

Intel Deutschland GmbH
Munich, Germany

Figure 1: Photorealistic rendering of a driving scenario generated with Intel® OSPRay's path tracer, demonstrating complex materials, soft-shadows from the illuminating sun and depth of field effects by the camera.

ABSTRACT

To gain the trust of governments and customers in the safety of autonomous driving systems it is crucial to demonstrate that they have been developed and successfully validated using a broad range of difficult scenarios. Because it is virtually impossible to capture all representative variants in the real world, researchers have already employed the simulations of virtual scenarios, e.g., using the CARLA [1] system.

However, often the sensor (including camera) and lighting simulation of those system is quite approximative, many real world effects are neglected. While today it is unclear how accurate such simulations need to be, we like to provide several examples where approximations are conspicuous and where intuitively those shortcomings will affect the validation coverage. Moreover, this position paper shows the limitations of rendering systems currently used in simulation, specifically using game engines, as CARLA [1] does. The main concern regarding such game engines is that they are built – and optimized – toward fast generation of visual appealing images at real-time frame rates. That makes a physical correct simulation of other sensor classes, which operate not directly in the visual spectrum hard or impossible. This is in particular the case for sensors like LIDAR, which operates in the infrared light spectrum, and radar, which again works on different principles.

Furthermore, we demonstrate that physically based rendering, as implemented in the open source rendering library Intel® OSPRay [2], can close the gap between simulation and the real world dramatically. We finish with a brief discussion on how these concepts can help validation of autonomous driving systems.

KEYWORDS

Autonomous Driving, Machine Learning, Global Illumination, Ray Tracing

ACM CSCS 2019, October 8, 2019, Kaiserslautern, Germany
.

1 ADVANTAGES OF PHYSICALLY BASED RENDERING

In the following we will detail some real world scenarios which are difficult to simulate with traditional rasterization-based renderers, but are well supported by ray tracing and physically based systems.

1.1 Physical Sensor Simulation

The accurate simulation of optical sensors such as cameras or LIDAR is well suited for ray tracing based systems. Lens effects like depth of field, chromatic aberration, distortions, and vignetting differ from camera model to camera model and can be computed. A ray tracer can even realistically simulate dominating artifacts such as glare or lens flares, which are caused by multiple internal reflection in the optical system.

Arguably more important are time dependent effects. The pixel-sensors of camera are not queried and read-out at the same time, resulting in a so-called "rolling shutter" effect. Similarly, one LIDAR scan of the surrounding takes some amount of time. Thus, when objects are in motion, or when the sensor itself is in motion, the shapes of captures objects will be blurred ("motion blur" effect) and distorted.

1.2 Complex Materials

Road markings and street signs have retroreflective properties to maximize visibility, in particular during twilight and night, i.e., they reflect light from the headlights primarily back in the direction toward the car.

Furthermore, especially when simulating LIDAR, it is important to support spectral rendering, because the reflection properties at infrared wavelength can be quite different to those in the visible light spectrum. Neglecting that will probably result in too idealistic LIDAR responses, whereas accurately modeling the infrared light propagation correctly results in either missing or in ghosting responses.

1.3 Shadows, Reflections and Illumination

Shadows from bright light sources like the sun, or headlights of cars create high-contrast regions. Furthermore, rasterization based techniques such as shadow maps cannot render contact shadows created near small geometric details, whereas shadows computed with ray tracing support this effect. Such details result in considerably different images and can be quite important for creating scenarios for training and validation for autonomous driving systems.

Correctly generated reflection are similarly important. Narrow crossings, where it is not possible to see around sharp turns, have mirror installations to aid the determination whether the crossing is free. Another example are glass facades of office buildings, where the reflections of cars can lead to detection ghost cars. Computing reflections is what ray tracers are famous for.

Two example situations where computing direct illumination with high dynamic range and with absolute physical quantities are: A low sun shining at a wet street, resulting in glare and blending. And the visibility of traffic lights with the sun shining from behind.

Indirect illumination including color bleeding could also play a (minor) role in accuracy, because it lowers the contrast of object contour, which could make segmentation and detection fail more often, if not incorporated in training.

1.4 Different Weather Conditions

Accounting for all different weather conditions is important for training the autonomous system for the real world. Physically based engines like OSPRay can support a physically based atmosphere and sky model, to not only render at sunny conditions, but also simulate the ambient diffuse lighting on an overcast day.

Many weather phenomena require volume rendering to render them accurately. This includes not only the obvious fog (which also scatters light from tail lamps), or clouds. Notably, also rain or snow can only be efficiently simulated by volumetric techniques, because it is prohibitive expensive to simulate myriads of rain drops or snowflakes individually.

2 THE OSPRAY RENDERING LIBRARY

OSPRay is an open source, scalable, and portable ray tracing engine for high-performance, high-fidelity visualization. OSPRay is part of the Intel oneAPI Rendering Toolkit and is released under the permissive Apache 2.0 license. The purpose of OSPRay is to provide an open, powerful, and easy-to-use rendering library that allows one to easily build applications that use ray tracing based rendering for interactive applications (including both surface- and volume-based visualizations). OSPRay scales well and runs on anything from laptops, to workstations, to compute nodes in HPC systems.

OSPRay also implements a physically based renderer which is based on path tracing. It supports complex materials, advanced cameras and realistic light sources, see Figure1. OSPRay is also designed in a modular way, making it easy to customize, extend or replace almost any part of the rendering pipeline, from geometry or volume objects over materials to cameras and light sources, or even the renderer core itself.

3 DISCUSSION AND OUTLOOK

The physical based rendering library OSPRay provides an excellent basis to simulate scenes, sensors, material and light more realistic than it would be possible with tools based on video game engines. These tools provide a basis to evaluate the importance of sensor and scene effects for the validation of complete autonomous driving systems or components of it. We will implement and test a physical based rendering engine in the BMWi project "KI-Absicherung" and evaluate it for the use in the validation of AI-based perception for automated driving.

REFERENCES

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16. http://carla.org/

[2] Ingo Wald, Greg P. Johnson, Jefferson Amstutz, Carson Brownlee, Aaron Knoll, Jim Jeffers, Johannes Günther, and Paul Navrátil. 2017. OSPRay - A CPU Ray Tracing Framework for Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics* (2017). https://www.ospray.org/