ISAL 2025
Article DOI: 10.26083/tuprints-00030906
16. International Symposium on Automotive Lighting
Darmstadt, 22. – 24. September 2025
DOI (proceedings): 10.26083/tuprints-00030825

3D Virtual Reality Testing-Tool for GFHB-Algorithm

A. Kneib[1], S. Ramanan[1], S. Poppe[1]
1: Stellantis N.V. / Opel Automobile GmbH, Bahnhofsplatz, 65423 Rüsselsheim

1. Abstract

This paper describes the product of a project, evaluating the feasibility to develop and validate algorithms and software for the automotive feature Glare-Free-High-Beam (GFHB). Aiming to evaluate the developed algorithms, a fully virtual testing platform was created to test, debug and visualize algorithm behaviour and performance in a 3D scene.
GFHB consumes data from a camera-based object-detection system in order to cut off specific high-beam segments, so that the affected traffic participant will not perceive a glaring from the ego-vehicle´s high-beam.
One major challenge in developing such algorithms is the availability of testing means and feasibility of testing activities in a controlled working environment, especially because real-world tests need to be performed during darkness / night time. Furthermore, the mentioned algorithms are complex in nature and contain a high amount of process steps, making it quite difficult to break down, isolate and analyse components the of the main software model, while performing a real-world test drive.
Therefore, a 3D virtual-reality testing tool has been developed, based on MATLAB/Simulink and the built-in Virtual Reality Toolbox, which enables the developers of GFHB algorithms to rapidly prototype and test several algorithmic approaches on calculating the so-called Glare-Free-Areas. The testing tool is able to use pre-recorded driving scenarios and it visualizes the outputs of the GFHB algorithms, while simulating these, based on the recorded vehicle data. The users of this testing tool are free to decide, whether to visualize only several specific process steps of the control algorithm, or the full end-to-end process chain (from object detection to LED control of the headlamps).

Keywords: Glare-Free-High-Beam, Virtual Testing, Automotive Software

2. Introduction

2.1 Umbrella Project

The hereby described testing-tool has been developed during a project to establish a baseline of algorithms for the Glare-Free-High-Beam feature suited for the new automotive electrical-electronic-architecture strategy of Stellantis and to assess their complexity and performance.
The task was to work out runnable and AUTOSAR-compliant proof-of-concept software-components, based on inhouse requirements of the Glare-Free-High-Beam feature. In fact, one key aspect was also to demonstrate the in-house virtual development capabilities and fast turn-around times including final demonstration on a prototype vehicle.

Since the Glare-Free-High-Beam feature is a quite complex feature, which relies on a multitude of geometrical calculations with a high number of consecutive process-steps, the development of such proof-of-concept algorithms becomes a challenge without a good visualization of the inputs and outputs.
Also, the usage of standard software-testing tools, like being used for unit-testing, is reaching its limits quite fast in terms of interpretability. The visualization capabilities of such tools are also limited in terms of graphical means or values.

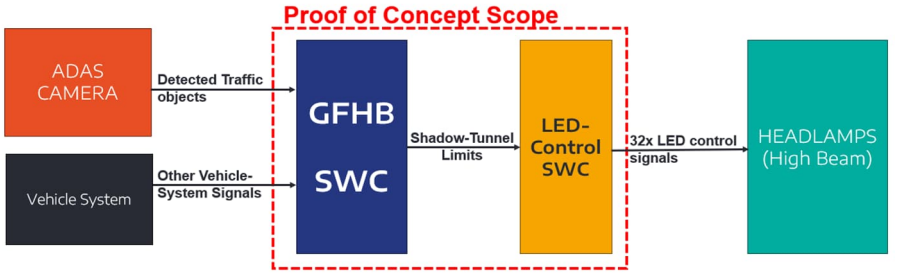

Figure 1: High-Level Architecture of the studied system

2.2 Glare-Free-High-Beam Feature

The examined feature “Glare-Free-High-Beam” (short: GFHB) is an advanced exterior lighting function, which makes use of camera-based object-detection and the presence of Headlamp-Technology, based on an array of multiple light-sources (LEDs), which can be controlled individually.

Once the object-detection algorithms are transmitting the parameters of the respectively perceived objects, based on bounding-boxes described by height, width, spatial location and other parameters, the GFHB algorithms are processing these information in order to turn off specific LEDs within the headlamp, so that the area in which the perceived object (e.g. another vehicle) has been detected, remains unlit.

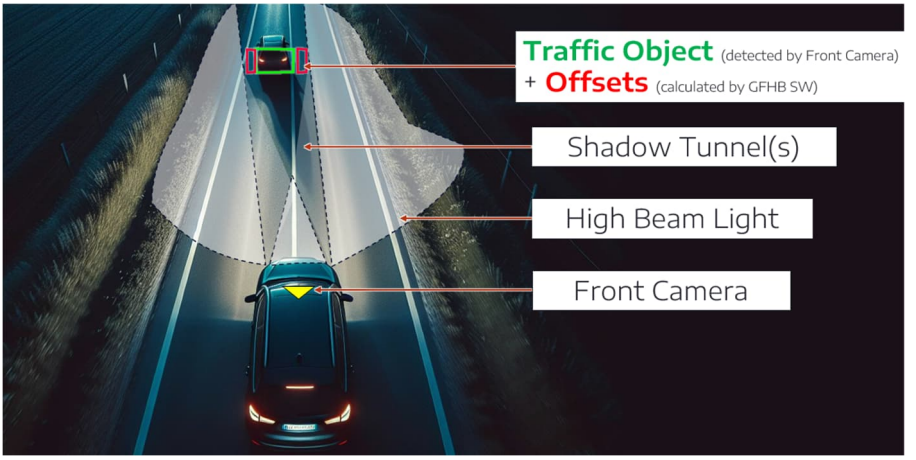

Figure 2: Illustration of fundamentals on GFHB-Feature

The unlit areas within the High-Beam are usually referred as “Glare-Free-Areas” or “Shadow-Tunnels”, in terms of the control algorithm.

The GFHB algorithm also calculates so called “offsets” in real-time, which are being added to the bounding boxes of the detected traffic object.

Furthermore, the GFHB feature is supposed to aggregate multiple traffic objects within a maximum of two Glare-Free-Areas (short: GFAs). This grouping approach ensures a high quality appearance of the remaining High-Beam, since fragmentation of the High-Beam and scattering of small, illuminated portions is being prevented.
Finally, the determined GFAs are being broadcasted from the GFHB feature to the headlamp control units. These are then mapping the GFAs to the array of LEDs intending to turn off the correct light sources and to obtain unlit areas at the region, where the traffic objects were detected.
This all happens periodically during one discrete time-step at system runtime.

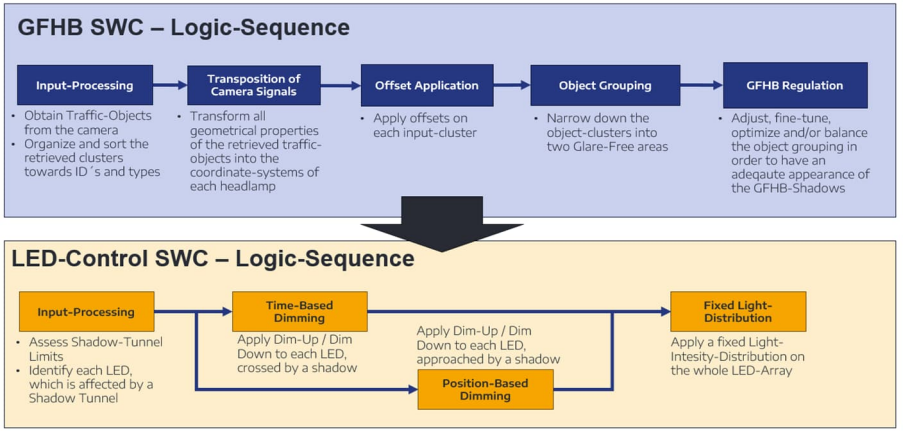

Figure 3: Overview on developed algorithms during the project

2.3 The challenge that created the need of 3D testing tool

Testing the own algorithms, while developing the same is a key to successful software development. Test driven development (TDD) is also an enabler for fast software development. But the available testing tools for automotive application software were showing several limitations, when dealing with a highly complex software component, like GFHB.

At present, the usual requirements-based testing approach in automotive application software development is based on unit-tests, which are performed by considering the standalone software component as the System Under Test (SUT). The test-case is composed of a sequence of pre-defined input signals in a spreadsheet-based document (also called “the stimuli vector”), as well as the expected output of the tested application. The application is simulated by feeding the stimuli-vector into the SUT during simulation and comparing the observed outputs to the expected outputs.

If both, expected and observed outputs match each other, the unit-tests are passed, otherwise failed.

	A	B	C	D	E	F	G	H	I	J
1	Format Version	1,4								
2	Architecture	Wrapper_MirrPosRvs								
3	Scope	MirrPosRvs/MirrPosRvs/MirrPosRvs_Main_sys								
4	Name	General_Test_Case_MirrPosRvs								
5	Mode	Inputs				Outputs				
6	Step	Rp_DriverMirrorVer	Rp_GearEngagedFo	Rp_PassengerMirro	Rp_Tilt_Mirr_Rev_E	Pp_DriverMirrorVer	Pp_DriverMirrorVer	Pp_PassengerMirro	Pp_PassengerMirro	Comment
7	0	45	13	45	0	0	45	0	45	Initialization: Neutral Gear & Feature deactivated
8	1	45	13	45	1	0	45	0	45	Feature activated
9	2	45	14	45	1	3	30	3	30	Gear changed to Reverse - Mirrors tilting down
10	3	45	13	45	1	3	45	3	45	Gear restored to Neutral - Mirrors tilting back to sav
11	4	45	14	45	0	0	45	0	45	Feature deactivated and Gear changed to reverse - M
12	5	45	14	45	1	3	30	3	30	Feature activated and Gear still in reverse - Mirrors t
13	6	45	13	45	1	3	45	3	45	Gear restored to Neutral - Mirrors tilting back to sav
14	7	127	13	127	1	0	255	0	255	SNA received for both mirrors - No Movement Requ
15	8	45	13	45	1	0	45	0	45	Valid value for both mirrors received - No movemen
16	9	45	14	45	1	3	30	3	30	Gear changed to reverse - Mirrors tilting down
17	10	45	13	45	1	3	45	3	45	Gear restored to Neutral - Mirrors tilting back to sav
18	11	10	13	10	1	0	10	0	10	Mirror position changed by customer to a value belo
19	12	10	14	10	1	3	0	3	0	Reverse Gear activated - Mirrors tilting down to pos
20	13	10	13	10	1	3	10	3	10	Gear restored to Neutral - Mirrors tilting back to sav

Figure 4: Example of test-case for state-of-the-art requirements based unit-test

This testing-approach is allowing a high-precision assessment of the software-quality and repeatable identification of defects.
But this approach is reaching its limits of usability when dealing with highly complex software-components, such as GFHB. The visualization of the observed outputs is also limited.

The GFHB feature is ingesting multi-dimensional data points and performing multiple sequential calculation steps, such as using trigonometrical functions, linear algebra and coordinate-system transformation and transforming data, originated from a highly dynamic three-dimensional space, to data, matching a two-dimensional space.
This would produce an overly high effort on creating the test-cases and interpreting the test-results, when using the described unit test approach.

As initially mentioned, the focus of this proof-of-concept was to rapidly prototype a vehicle application, for the purpose of better understanding the effort behind a full development of such a feature. Therefore, a tool was required to examine the behaviour of the developed algorithms visually, while working the overall component further out.

3. 3D Virtual Reality Testing Tool

3.1 Basic Concept and Architecture

The developed testing tool is fully based on MATLAB/Simulink and is purely using the default pre-installed libraries and toolboxes of MATLAB version 2023b.
Besides the default Simulink-Toolbox, the “Simulink 3D Animation” and the Computer Vision Toolbox are playing an integral part on forming the mentioned testing tool within Simulink.
The virtual testing tool is represented by a Simulink-Model, containing all signal-sources, signal-processors and output-visualizers.
The GFHB Software-Component and the LED-Control Software-Component, which have been developed during the proof-of-concept project, are embedded in the testing tool in the form of referenced Simulink models. These referenced models are executed, when the Simulink model of the testing tool is run in Simulink. There are also several referenced subsystems embedded in the testing tool, which contain the source-signals for the actual simulation.
The source signals are based on pre-recorded vehicle CAN network data, captured during several real-world drive sessions with the front camera system, as well as a corresponding dashcam video recording for better comprehension of the currently used source data. The captured signals are re-played during execution of the testing tool and fed through a gateway-component into the main GFHB Simulink-Model. The resulting outputs, which are produced by the GFHB- and LED-Control-Model are forwarded to several post-processing algorithms, which feed the 3D-Visualization functionality aiming to visualize the GFHB-outputs at simulation runtime.

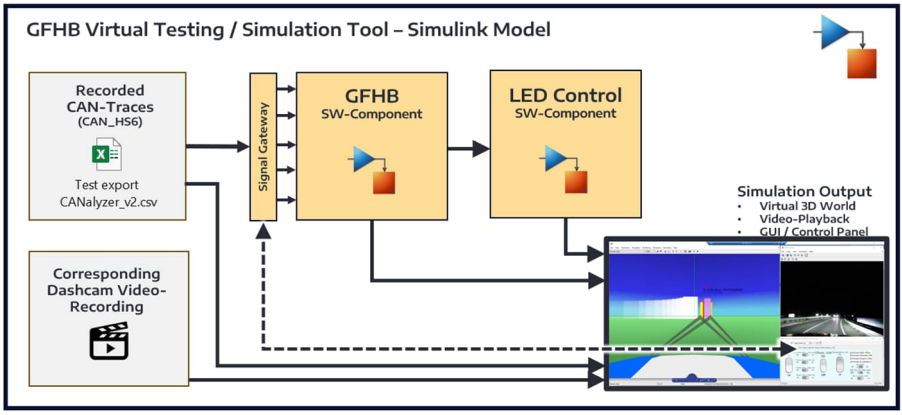

Figure 5: Architecture of developed virtual testing tool

The 3D-Visualization is realized through a pre-defined virtual world (with the Simulink block “VR-Sink”), which is containing all required graphical assets to render the visualization.
Additionally, the corresponding dashcam video is played back in a video-player, which is synchronized with the Simulink simulation.
A control-panel is enabling the user to turn on/off several graphic-elements of the visualization or read-out several parameters, while the simulation is running.

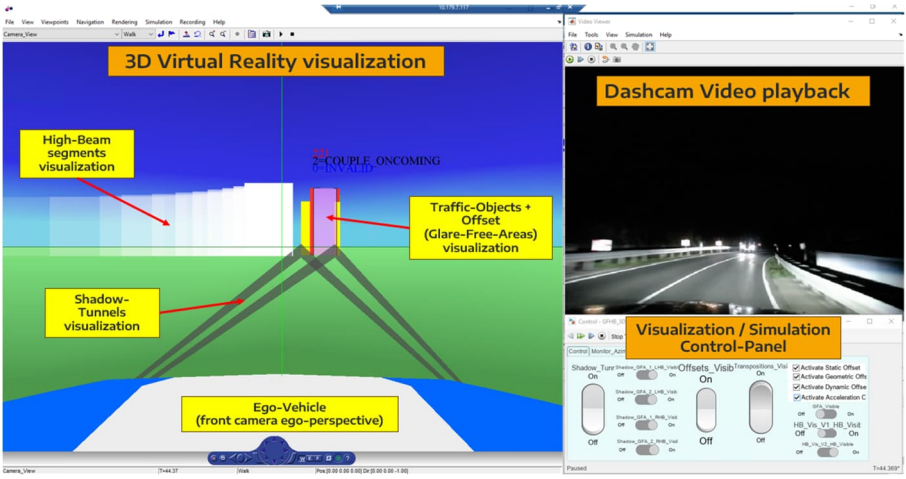

Figure 6: Description of GUI of developed virtual testing tool

3.2 Enhanced GFHB and LED-Control visualization tool

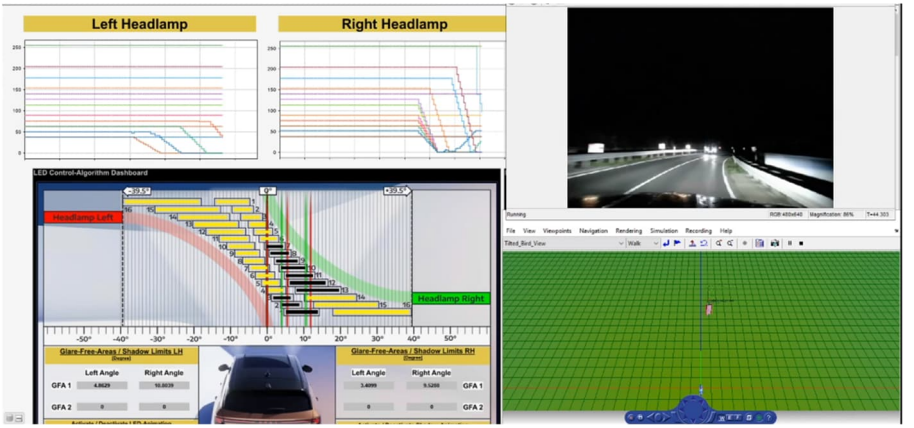

Figure 7: GUI of enhanced testing tool for LED-Control

The herein described virtual testing tool has been further advanced in order to be used while driving a prototype vehicle (mule vehicle). The corresponding Simulink model has been enhanced with a further GUI, depicting the current actuation of each LED segment in the vehicle headlamp versus the Glare-Free-Areas calculated by the GFHB component. This enhanced testing tool is running on a laptop, simulating the GFHB software component, ingesting signals coming from the mule vehicle CAN network and visualizing the LED-actuation commands being transmitted to the same vehicle CAN. Additionally, the enhanced testing tool visualizes the currently calculated GFHB artifacts in the already known virtual world (in sync with the current vehicle drive). The two scrolling plots at the upper left side are indicating the current dim-level of each LED-Segment, since the developed LED-control algorithms are also applying several dimming-mechanisms, when turning off the light at the glare-free-areas.

4. Conclusion

The herein described testing tool has been developed within a period of 7 months, while working on the actual umbrella-project and required much less than 1 software engineer full-time-equivalent. The development efforts have been made mostly virtual, with the real target-system (mule vehicle) demonstration at the very end.
Besides studying the feature-requirements for GFHB and LED-Control, working-out the algorithms and Simulink models for GFHB and LED-Control, integrating the software-models in a physical test-vehicle and interacting with all stakeholders, the development of the herein mentioned testing-tools was a major part of the project as well.

4.1 Key-Findings

The described testing-tools have been designed from scratch and further extended on a needs basis in parallel to all of the actual algorithm development activities mentioned above and contributed strongly to a streamlined and focused approach on understanding and improving performance of the main feature algorithms and models.
The tools also turned out to be helpful as a communication means regarding system and service requirements for GFHB and LED-Control.
The fact that it was possible to simulate the algorithms and to visualize their respective (intermediate) outputs in an interactive way also served the respective Service-Designers to better assess the quality and validity of their requirements and also enabled all parties to try out different approaches.

4.2 Limitations and Risks

As with every virtual testing tool, the usage of such tools needs to be carried out with caution and critical thinking, since there may be a risk of unidentified defects, which can not be reproduced in a virtual environment. Also, the capability of the virtual testing-tool needs to be ensured by validating its processes and outputs.
For this specific virtual testing tool, there is also the risk of loosing the access and maintainability, once MathWorks (the vendor of MATLAB/Simulink) is deciding to remove the herein used toolboxes, such as the Simulink 3D Animation toolbox, from further product-releases.

4.2 Potential Next Steps

The herein described testing tool(s) are highly dependent on input-signals, which are formed on real-world systems, such as a drivable mule-vehicle with an open CAN-Interface or pre-recorded CAN-Messages from prior real-world driving sessions.

In order to eliminate the need of a physical vehicle or pre-recorded CAN-Data, there was the deliberation to attach an advanced simulation tool like SCANeR Studio from AVSimulation, which is supposed to synthesize the required object-detection signals, usually coming from the vehicle front-camera.

4. Acknowledgements

The mentioned proof-of-concept project has been carried out by a highly collaborative and cross-functional team.

During the project, the mainly acting software-engineers have been highly supported on Automotive Lighting Technology and the respective service-specifications for Glare-Free-High-Beam and Headlamp LED-Control by the following experts:

- Carsten Neitzke – Lighting Service Designer
  – Stellantis N.V. / Opel Automobile GmbH
- Stefan Holderried – Lighting Subsystem Architect
  – Stellantis N.V. / Opel Automobile GmbH
- Dominic Weil – Lighting Validation Engineer
  – Stellantis N.V. / Opel Automobile GmbH
- Philipp Röckl – Group Leader Lighting Systems
  – Stellantis N.V. / Opel Automobile GmbH