# Real-Time Simulation and Control of High-Definition Matrix Headlights for Automated Driving

A thesis approved for the academic degree of

Doktor der Ingenieurswissenschaften (Dr.-Ing.)

at the

Faculty of Electrical Engineering and Information Technology

TU Dortmund University

by

Mirko Waldner, M.Sc.

| | |
|---|---|
| Supervisor: | Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram, TU Dortmund University |
| Co-Advisor: | Univ.-Prof. Dr. sc. nat. habil. Christoph Schierz, TU Ilmenau |

Day of Oral Examination: 24.11.2025

# Acknowledgement

Without the support and insightful discussions with many individuals, groups, and organizations over the past years, I would not have been able to complete this dissertation. I want to express my sincere gratitude to all of you. To prevent this acknowledgment from becoming as overly lengthy as the dissertation itself, the following special acknowledgments represent only a selection, presented without ranking. Please do not be disappointed if you are not explicitly mentioned in this list.

I want to express my special thanks to my doctoral supervisor, Prof. Torsten Bertram, for his support, scientific discussions, and valuable advice, and above all for giving me the freedom to explore my ideas. Without this scientific freedom and the trust he placed in me, my work would not have been possible.
I would also like to thank my co-supervisor, Prof. Christoph Schierz, for his guidance and for reviewing my dissertation. His insightful comments have improved this work.

The entire RST team deserves my deepest gratitude for the fantastic years we shared. All current and former scientific, technical, and administrative RST members have made the institute a truly positive and special place. I have always felt supported and thoroughly enjoyed my time as a research assistant. Perhaps the team's excellence explains why my work took so long.
I would also like to thank the project partners of the publicly funded research projects SHT and AHEAD. These collaborations and discussions enabled the incorporation of practical experience and expert feedback. This exchange has genuinely broadened my horizons. Special thanks go to Forvia-Hella GmbH, which enabled me to conduct experiments with real matrix headlights and test my algorithms in actual traffic situations.

Last but not least, I would like to thank my proofreaders, Nathalie Müller and Heiko Renz, for reading an early draft of my work and correcting my written inconsistencies. Their help and support have substantially improved the quality of this dissertation.
Finally, I would like to thank my family and friends for their unwavering support and encouragement. They have always been there for me, even when I was stressed, in a bad mood and had little time for them. I am deeply grateful for their understanding and patience. Without their support, I would not have been able to complete this work. Thank you all very much!

# Abstract

Modern matrix headlights with up to 1.3 million controllable light sources called pixellights enable precise illumination control and sophisticated lighting functions for automated driving. However, their development, verification and validation present substantial challenges that require real-time headlight simulation and control capabilities to be mastered efficiently in terms of time and cost.
This thesis at hand addresses these challenges through three primary contributions: a novel real-time matrix headlight simulation for commercial rendering engines, an intuitive and flexible real-time matrix headlight control algorithm and a comprehensive matrix headlight rapid prototyping and development tool. The developed methods have been extensively validated by testing with real matrix headlights.

The first significant contribution is a physically accurate, real-time capable simulation approach that combines standard rendering engine spotlights with Sparse Matrix-Vector Multiplication (SpMV) to be lighting technology-independent. The SpMV method achieves real-time performance while maintaining physical accuracy by leveraging established sparse matrix formats and parallel computing algorithms.
Experimental verification shows the ability to simulate a matrix headlight with 1.51 million pixellights at a mean computation time of 0.47 ms on a Nvidia 2080 Ti graphics card.

The second significant contribution is the SuperSampling Control (SSC) algorithm for real-time matrix headlight control of arbitrary lighting functions and beam patterns. SSC synthesizes ray tracing, Supersampling Anti-Aliasing and the MapReduce parallel processing method into an intuitive yet computationally efficient approach for rapidly prototyping arbitrary dynamic lighting functions.
Experimental evaluation shows SSC can control a matrix headlight with 1.05 million pixellights at a median computation time of 1.79 ms while maintaining high illumination quality on a Nvidia 2080 Ti graphics card.

The third significant contribution is developing the See the Optimal Lighting (SOL) matrix headlight development tool, which integrates the proposed real-time simulation and control methods. SOL supports hybrid development approaches through virtual rapid prototyping and hardware-in-the-loop testing with real matrix headlights.
Practical applications are energy-efficient, environment-optimal illumination and observer-specific distortion-free symbol projection with closed-loop feedback control.

# Contents

# Nomenclature

**Roman Symbols**

| | |
|---|---|
| $A$ | Illuminated area |
| $a_{\mathrm{ac,ssc}}$ | Adjustment of SSC pixel activation |
| $a_{\mathrm{d,ssc}}$ | Adjustment of SSC edge darkening |
| $a_{\mathrm{x,ssc}}$ | Adjustment of $I_{\mathrm{v}}$ threshold of SSC value excluding |
| $\boldsymbol{A}_{\mathrm{h}}$ | Headlight matrix |
| $a_{\mathrm{m,ssc}}$ | Adjustment of equality of SSC mapping |
| $a_{\mathrm{r,ssc}}$ | Adjustment of SSC reduction |
| $\boldsymbol{b}(l_{\mathrm{b}})$ | Bézier curve of length $l$ |
| $\mathcal{C}_{\mathrm{p,gz},i}$ | Set of pixel indices with $I_{\mathrm{v}} > 0\,\mathrm{cd}$ of $i$-th texel |
| $d$ | Distance |
| $d_{\mathrm{cu}}$ | Width of a curve or distance threshold |
| $\boldsymbol{d}_i$ | Direction vector of $i$-th ray |
| $\boldsymbol{d}_{\mathrm{p},1}$ | Primitive first direction vector |
| $\boldsymbol{d}_{\mathrm{p},2}$ | Primitive second direction vector |
| $e_{\mathrm{dssim}}$ | Structural dissimilarity error |
| $e_{\mathrm{mae}}$ | Mean absolute error |
| $e_{\mathrm{mare}}$ | Mean absolute relative error |
| $e_{\mathrm{rmse}}$ | Root mean squared error |
| $e_{\mathrm{sse}}$ | Sum of squared errors |
| $e_{\mathrm{ssim}}$ | Structural similarity index measure error |
| $e_{\mathrm{sy,pi}}$ | Symbol projection error of the pitch rotation |
| $e_{\mathrm{sy,ro}}$ | Symbol projection error of the roll rotation |
| $e_{\mathrm{sy,x}}$ | Symbol projection error of $x$-coordinate |
| $e_{\mathrm{sy,y}}$ | Symbol projection error of $y$-coordinate |
| $E_{\mathrm{v}}$ | Illuminance |
| $E_{\mathrm{v,s}}$ | Illuminance setpoint |
| $f_{\mathrm{bd}}(l_{\mathrm{b}})$ | Function of the squared distance of Bézier curve to a point |
| $\boldsymbol{I}$ | Grayscale image |
| $\boldsymbol{I}_{\mathrm{s}}$ | Desired symbol image |
| $I_{\mathrm{v}}$ | Luminous intensity |
| $I_{\mathrm{v,av}}$ | Average of luminous intensities |
| $\boldsymbol{I}_{\mathrm{v,h}}$ | Headlight luminous intensity texture |
| $\boldsymbol{i}_{\mathrm{v,h}}$ | Headlight luminous intensity vector |
| $\boldsymbol{I}_{\mathrm{v,h,s}}$ | Headlight luminous intensity setpoint texture |
| $\boldsymbol{i}_{\mathrm{v,h,s}}$ | Headlight luminous intensity setpoint vector |
| $I_{\mathrm{v},i}$ | Luminous intensity of $i$-th texel |
| $I_{\mathrm{v,a},i,j}$ | Additional luminous intensity of $i$-th texel by $j$-th pixel |

| | |
|---|---|
| $I_{\mathrm{v,m},i,j}$ | Maximal luminous intensity of $i$-th texel by $j$-th pixel |
| $I_{\mathrm{v,s},i,j}$ | Mapped luminous intensity setpoint of $i$-th texel to $j$-th pixel |
| $I_{\mathrm{v,m},i}$ | Maximal luminous intensity of $i$-th texel by all pixels |
| $I_{\mathrm{v,m},i,\mathrm{sum}}$ | Sum of maximal luminous intensities of $i$-th texel of all its pixel |
| $I_{\mathrm{v,s},i}$ | Luminous intensity setpoint of $i$-th texel |
| $I_{\mathrm{v,m},j}$ | Maximal luminous intensity of $j$-th pixel |
| $I_{\mathrm{v,lb}}$ | Luminous intensity lower bound |
| $I_{\mathrm{v,m}}$ | Maximal luminous intensity |
| $I_{\mathrm{v,m,s}}$ | Maximal luminous intensity of all setpoints |
| $\boldsymbol{I}_{\mathrm{v,p},j}$ | Luminous intensity texture of $j$-th pixel |
| $\boldsymbol{i}_{\mathrm{v,p},j}$ | Luminous intensity vector of $j$-th pixel |
| $\boldsymbol{i}_{\mathrm{v,p},j,\mathrm{cp}}$ | Compressed luminous intensity vector of $j$-th pixel |
| $\boldsymbol{i}_{\mathrm{v,p,s},j}$ | Luminous intensity setpoint vector of $j$-th pixel |
| $I_{\mathrm{v,s}}$ | Luminous intensity setpoint |
| $I_{\mathrm{v,sum}}$ | Sum of luminous intensities |
| $I(v,u)$ | Single element of a grayscale image at position $u, v$ |
| $I_{\mathrm{v,ub}}$ | Luminous intensity upper bound |
| $l$ | Length of a ray or curve |
| $l_{\mathrm{b},k}$ | Length of a Bézier curve at step $k$ |
| $l_{\mathrm{b}}$ | Length of a Bézier curve |
| $L_{\mathrm{v}}$ | Luminance |
| $m_{p,q}$ | Raw image moment of order $(p+q)$ |
| $\boldsymbol{n}$ | Normal vector |
| $n_{\mathrm{bez}}$ | Degree of Bézier curve |
| $n_{\mathrm{col}}$ | Number of matrix columns |
| $n_{\mathrm{p}}$ | Number of pixels |
| $n_{\mathrm{p,av}}$ | Average illuminating pixels per texel and overlaps |
| $n_{\mathrm{p},i}$ | Number of illuminating pixels of $i$-th texel |
| $n_{\mathrm{row}}$ | Number of matrix rows |
| $n_{\mathrm{t}}$ | Number of light texture texels |
| $n_{\mathrm{t,av}}$ | Average number of light texture texels of all pixels |
| $n_{\mathrm{t,gz}}$ | Number of light texture texels illuminated by at least one pixel |
| $n_{\mathrm{t,gz},k}$ | Number of currently illuminated light texture texels |
| $n_{\mathrm{t,gz,s}}$ | Number of light texture texels with a setpoint greater zero |
| $n_{\mathrm{t},j}$ | Number of light texture texels of $j$-th pixel |
| $\boldsymbol{p}_{\mathrm{bez},i}$ | $i$-th Bézier curve control point |
| $p_{\mathrm{cog,bl}}$ | Position of bottom-left centroid of a symbol |
| $p_{\mathrm{cog,bl,s}}$ | Position of bottom-left centroid of setpoint image |
| $p_{\mathrm{cog,br}}$ | Position of bottom-right centroid of a symbol |
| $p_{\mathrm{cog,br,s}}$ | Position of bottom-right centroid of setpoint image |
| $p_{\mathrm{cog,cen}}$ | Position of central centroid of an image |
| $p_{\mathrm{cog,cen}}(k)$ | $k$-th element/coordinate of the position of central centroid |
| $p_{\mathrm{cog,cen,s}}$ | Position of central centroid of setpoint image |
| $p_{\mathrm{cog,cen,s}}(k)$ | $k$-th element/coordinate of the position of the central centroid of setpoint image |

| | |
|---|---|
| $p_{\text{cog,tl}}$ | Position of top-left centroid of a symbol |
| $p_{\text{cog,tl,s}}$ | Position of top-left centroid of setpoint image |
| $p_{\text{cog,tr}}$ | Position of top-right centroid of a symbol |
| $p_{\text{cog,tr,s}}$ | Position of top-right centroid of setpoint image |
| $\boldsymbol{p}_i$ | $i$-th 3D world point |
| $\boldsymbol{p}_{\text{ray},i}(l)$ | World point of $i$-th ray at length $l$ |
| $\boldsymbol{p}_{\text{o,e}}$ | Ellipsoid origin point |
| $\boldsymbol{p}_{\text{o},i}$ | Origin point of $i$-th ray |
| $\boldsymbol{p}_{\text{o,p}}$ | Primitive origin point |
| $\mathcal{P}$ | Set of lighting primitive vectors and additional control parameters. |
| $\boldsymbol{p}_{\text{t},i}$ | $i$-th 2D texel position |
| $r$ | Radius |
| $r_{\text{el}}$ | Radius of ellipsoid |
| $r_{\text{s},k,j}$ | Row and texel index of $k$-th setpoint of $j$-th pixel |
| $\boldsymbol{s}_{\text{el}}$ | Ellipsoid size vector |
| $\boldsymbol{T}_{\text{hl,wo}}$ | Homogenous transformation matrix from headlight to world coordinates |
| $\boldsymbol{T}_{\text{wo,pers}}$ | Perspective homogenous transformation matrix of world coordinates |
| $u$ | 2D image and texture $u$-coordinate |
| $u_{\text{p,a}}$ | Absolute pixel utilization $\in \mathbb{R}_{\geq 0 \wedge \leq 1}$ |
| $u_{\text{p,def},j}$ | Default pixel utilization $\in \mathbb{R}_{\geq 0 \wedge \leq 1}$ of $j$-th pixel |
| $u_{\text{p},j}$ | Pixel utilization $\in \mathbb{R}_{\geq 0 \wedge \leq 1}$ of $j$-th pixel |
| $\boldsymbol{u}_{\text{p,def}}$ | Default pixel utilization vector $\in \mathbb{R}^{n_{\text{pix}}}_{\geq 0 \wedge \leq 1}$ |
| $\boldsymbol{u}_{\text{p}}$ | Pixel utilization vector $\in \mathbb{R}^{n_{\text{pix}}}_{\geq 0 \wedge \leq 1}$ |
| $u_{\text{p,r}}$ | Relative pixel utilization $\in \mathbb{R}_{\geq 0 \wedge \leq 1}$ |
| $v$ | 2D image and texture $v$-coordinate |
| $V(\lambda)$ | Luminous efficiency function |
| $w$ | Weight of a setpoint |
| $w_{i,j}$ | Weight of $i$-th $I_{\text{v}}$ setpoint of $j$-th pixel |
| $\boldsymbol{w}_j$ | Weight vector of $I_{\text{v}}$ setpoint of $j$-th pixel |
| $\boldsymbol{W}$ | Weight matrix |
| $x$ | Right-handed 3D Cartesian $x$-coordinate |
| $x_{\text{h}}$ | $x$-coordinate of mounting or origin position of the headlamp |
| $y$ | Right-handed 3D Cartesian $y$-coordinate |
| $y_{\text{h}}$ | $y$-coordinate of mounting or origin position of the headlamp |
| $z_{\text{h}}$ | $z$-coordinate of mounting or origin position of the headlamp |
| $z$ | Right-handed 3D Cartesian $z$-coordinate |

**Greek Symbols**

| | |
|---|---|
| $\epsilon_{\text{add}}$ | Additional infinitesimal positive value |
| $\lambda$ | Wavelength |
| $\Omega$ | Solid angle |
| $\phi$ | Horizontal angle of the headlight |
| $\phi_{\text{fov}}$ | Horizontal field of view and opening angle |
| $\phi_{\text{IES}}$ | Azimuth or horizontal angle |

| | |
|---|---|
| $\Phi_v$ | Luminous flux |
| $\psi$ | Vertical angle of the headlight |
| $\psi_{fov}$ | The vertical field of view and opening angle |
| $\psi_{IES}$ | The pitch or vertical angle |
| $\theta$ | Angle between the surface normal and light ray |

**Italics for Arrays (Computer Data Type)**

| | |
|---|---|
| *ColumnIndices* | Column indices of non-zero entries |
| *RowIndices* | Row indices of non-zero entries |
| *RowOffsets* | Starting indices of each row data block |
| *SliceOffsets* | Starting index of each Ellpack slice |
| *srcIDBuffer* | Indices of values |
| *srcRGBYBuffer* | RGB-color and luminous intensity values |
| *trgBuffer* | Coordinates, memory offset and number of data values |
| *Values* | Matrix values |

**Abbreviations and Acronyms**

| | |
|---|---|
| 2D | 2-Dimensional |
| 3D | 3-Dimensional |
| ADB | Adaptive Driving Beam |
| AFS | Adaptive Front-lighting System |
| bl | Bottom-Left |
| br | Bottom-Right |
| COG | Center of Gravity |
| COO | Coordinate |
| CPU | Central Processing Unit |
| CSR | Compressed Sparse Row |
| CUDA | Compute Unified Device Architecture |
| DMD | Digital Micromirror Device |
| DoF | Degrees of Freedom |
| DR | Dynamic Range |
| DSSIM | Structural Dissimilarity |
| ELL | Ellpack |
| FoV | Field of View |
| GFHB | Glare-Free High Beam |
| GPU | Graphics Processing Unit |
| GUI | Graphical User Interface |
| HD | High-Definition |
| HiL | Hardware-in-the-Loop |
| HiRes | High Resolution |
| HLS | Headlamp Leveling System |
| HRC | Headlight Range Control |
| HSPR | Headlight Safety Performance Rating |
| IES | Illuminating Engineering Society |
| JPO | Joint Pixel Optimization |
| LED | Light-Emitting Diode |

| | |
|---|---|
| LGC | Light Group Combination |
| LID | Luminous Intensity Distribution |
| LoRes | Low Resolution |
| MAE | Mean Absolute Error |
| MARE | Mean Absolute Relative Error |
| MBL | Material-Based Lighting |
| MRPC | MapReduce Pixel Control |
| MSE | Mean Squared Error |
| OBL | Object-Based Lighting |
| PBLD | Primitive-Based Lighting Design |
| PBR | Physically Based Rendering |
| PID | Proportional Integral Derivative |
| pixel | picture-element |
| PSNR | Peak Signal-to-Noise Ratio |
| RGB | Red-Green-Blue |
| RGSS | Rotated Grid SuperSampling |
| RMSE | Root Mean Squared Error |
| RTHC | Ray Tracing Headlight Control |
| SELL | Sliced Ellpack |
| SOL | See the Optimal Lighting |
| SpMV | Sparse Matrix-Vector Multiplication |
| SSAA | Supersampling Anti-Aliasing |
| SSC | SuperSampling Control |
| SSIM | Structural Similarity Index Measure |
| SSL\|HD | Solid State Lighting High Definition |
| StD | Standard Deviation |
| texel | texture-element |
| tl | Top-Left |
| tr | Top-Right |

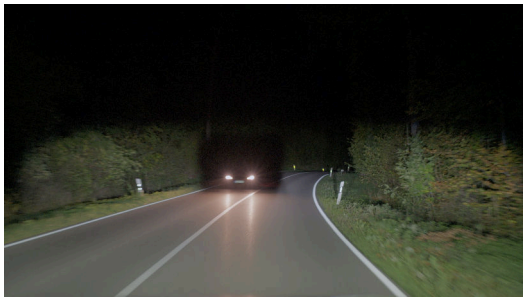
**Indication of usage of generative AI models to create the sections or content**

| | |
|---|---|
| $‡_c$ | Content Generation: Generating entire sections |
| $‡_t$ | Media Generation: Creating entire passages from given content |

# 1 Introduction

Over the past decades, automotive lighting has evolved from simple bulbs to sophisticated matrix headlights as an integral component of driver assistance systems and automated driving [Erk+24]. Modern matrix headlights address the dilemma between maximizing visibility for the driver and minimizing glare for other road users by creating precise, selective illumination or masking traffic objects while maintaining bright illumination elsewhere. The precise lighting control creates new possibilities for enhancing road safety and vehicle-to-human communication.

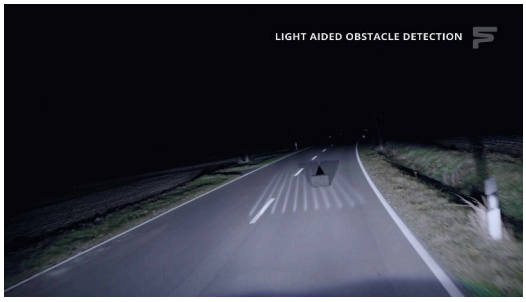
Fig. 1.1 shows adaptive lighting functions like Glare-Free High Beam (GFHB) to adjust beam patterns to prevent dazzling other road users by creating dark tunnels [Hum10]. Marker light highlights potential hazards or points of interest and guides the drivers' attention [Sch11]. Symbol projection by matrix headlights communicates intentions or warnings to other traffic participants [Riv+15]. Additionally, matrix headlights can project structured light for environmental sensing, turning the headlight into a part of an active sensor for automated driving [Kub+14; Wal+22a].



(a) GFHB [Por22].

(b) Marker light onto a person [Mer21].

(c) Warning and guiding symbol projection [Aud22a].




(d) Structured light projection [Kub+14; For23a].

Figure 1.1.: Adaptive lighting functions to support the driver.

## 1.1. Simulation-based Validation of Lighting Functions[‡c]

Matrix headlights consist of several thousand up to 1.3 million individually controllable light sources called pixellights arranged as a matrix grid [Ros18; Erk+24]. Using only physical prototyping and real-world testing, while valuable and indispensable, have significant limitations and economic considerations. Besides the general benefits of simulation-based development, verification and validation, such as cost reduction, time savings, early error detection, save failure and dangerous case testing, reproducibility, regression testing and increased flexibility, matrix headlights present specific challenges that make simulation particularly valuable [Jun23; Kau+21b].
The high number of pixellights creates a vast lighting design space. Simulations enable rapid exploration of the design possibilities, allowing developers to evaluate numerous beam patterns and control strategies efficiently.
Physical prototyping of matrix headlights is expensive and time-consuming. These economic costs multiply when considering the need for multiple iterations and variants during development. Moreover, physical testing requires dedicated facilities, test vehicles and personnel, increasing development expenses. Simulation reduces these costs by enabling virtual testing and optimizing multiple configurations without additional hardware costs before committing to physical prototypes.
Physical testing of automotive components requires specific environmental conditions that severely constrain development schedules. The night driving tests of vehicle lighting systems can typically only be carried out during a 4-6 hour window each day, weather permitting, extending development timelines. Additionally, varying atmospheric conditions and changing seasons affect the required illumination conditions, making it challenging to maintain consistent and required testing environments. Simulation eliminates these temporal constraints, enabling continuous development regardless of the time of day or weather conditions [Jan+19; Kau+21b].

However, realizing these benefits requires simulations that accurately simulate complex matrix headlights while maintaining real-time performance. Additionally, the simulation must support rapid prototyping and optimization of headlight control algorithms to enable the efficient development of sophisticated lighting functions. These challenges motivate the development of improved simulation and control methodologies to enable rapid prototyping and evaluation of lighting functions while maintaining real-time performance, which form the core contribution of this thesis.

## 1.2. Differentiation from the State-of-the-Art

Current approaches of real-time matrix headlight simulation employ one of three distinct methodologies: masking [Syn23; Syn21a; Syn21b; AVS22a; Thy22; AVS22b], scaling [Bor+22] and additive superposition [SN18; Rüd+19a; Rüd22] of beam patterns.
The masking approach creates user-definable shadow masks as overlays in headlamp beam patterns without considering physical feasibility [Syn23; Rüd22]. Scaling uses a bitmap of dimming values to modify the luminous intensity curve of a full matrix headlight. This method offers flexible adjustment of headlamp resolution, but it neglects pixellight

illumination overlaps [Bor+22]. While these two methods offer computational efficiency and are real-time capable, they sacrifice physical accuracy by neglecting complex optical interactions like overlapping pixellight illuminations.
Conversely, additive superposition provides physically accurate results by multiplying utilizations for each pixellights with its Luminous Intensity Distribution (LID), followed by cumulative addition [SN18]. However, this approach faces significant computational challenges as the number of pixellights increases. The calculation time is scaled proportionally to the pixellight count. Parallel and independent to the development of the Sparse Matrix-Vector Multiplication (SpMV) solution of this thesis [WB20], Rüddenklau and Weise [RW20] developed the Light Group Combination (LGC) format [Rüd22]. Both methods use the fact that each pixellight illuminates a specific direction within a limited solid angle. Both approaches avoid accessing and processing data elements with zero value to increase the computational efficiency of additive pixellight illumination superposition.

A significant differentiation from previous work [Rüd22] lies in the headlight modeling. While Rüddenklau [Rüd22] uses virtual light source models with spherical beam propagation, this thesis uses a conventional pyramid-shaped light model [Lec+99; LS02; Ber05; Epi23a] using Projective Texture Mapping [Eve01; Seg+92]. The beam propagation is similar to a video projector, which offers computational advantages by avoiding expensive trigonometric calculations. Using a pyramid-shaped beam propagation maintains compatibility with standard rendering engine spotlights from Unreal [Epi23b] and Unity [Uni23].
Furthermore, this choice incorporates shadow casting capabilities, which are notably absent in the simulator of Rüddenklau [Rüd22] and many commercial matrix headlight simulators as illustrated in Sec. A.1 in Fig. A.1 [Ans23; Syn23; Rfp22b; AVS22a; VIR23]. Shadow casting enhances simulation realism, particularly in foliage illumination scenarios, obstacle detection (see Fig. 1.1d) and pedestrian detection systems where shadow information can serve as additional detection features.

After solving the challenge of real-time simulation of matrix headlights with commercial rendering engines, the novel matrix headlight simulator is used to develop efficient control and optimization approaches of dynamic lighting functions.
The real-time control of matrix headlights presents a similar challenge to simulation, particularly for High-Definition (HD) systems incorporating many pixellights with arbitrary shapes. Current control methods frequently rely on simplified pixellight illumination approximations [Mic14; SB16; Rüd22] that fail to capture the full complexity of pixellight illumination interactions, or employ computationally intensive Joint Pixel Optimization (JPO) approaches [Dan+21; Bau+21; RT21; Rüd+22] that prove unsuitable for real-time applications. The increasing sophistication of lighting functions, combined with the requirements of automated driving, necessitates new control strategies that balance computational efficiency with illumination quality.
For real-time matrix headlight control Michenfelder [Mic14] approximates the matrix headlights as ideal video projectors and Saralajew and Benderman [SB16] approximate the pixellight illumination as homogeneous rectangles or centroid points. These simplifications neglect the complex pixellight interactions and overlaps critical for accurate lighting control and do not consider anti-aliasing requirements for appealing symbol projection [GI19]. Real-time symbol projection is possible with the known image point transformation of

Rivero et al. [Riv+15] and the Moore-Penrose inverse of Hummel et al. [Hum+18], but this approach is limited to minimizing the unconstrained Mean Squared Error (MSE) between the desired and the real overall illumination, which may not be optimal for all applications and lighting functions.
To the best of the author's knowledge, no general real-time capable matrix headlight control approach in the literature can be used for any number of pixellights, arbitrary shapes, homogeneity of illumination, or arbitrary lighting functions. However, such an approach is essential for developing advanced lighting functions for automated driving to rapidly prototype novel lighting functions and allow developers to explore the large lighting design space efficiently.

## 1.3. Research Objective and Outline

Due to the limitations of existing simulation and control methods, this thesis aims to develop novel real-time simulation and control methodologies for matrix headlight development that combine physical accuracy with computational efficiency to enable real-time rapid prototyping of advanced lighting functions for automated driving. Therefore, the three main research objectives and contributions of this thesis are:

1. Development of a physically accurate, real-time capable simulation for matrix headlights that scales efficiently with increasing pixellight number while maintaining high fidelity in optical characteristics and spatial LID.
2. Creation of a control algorithm that enables real-time control of arbitrary dynamic HD lighting functions while considering pixellight illumination overlap effects and arbitrary shapes, maintaining computational efficiency and supporting anti-aliasing.
3. Implementation of the simulation and control methods within a novel development tool that supports real-time rapid prototyping of arbitrary dynamic HD lighting functions and enables integration in real test vehicles.

The primary novel contributions for real-time matrix headlight simulation and development tools are:

- Connection of real-time matrix headlight simulation to SpMV calculations on the Graphics Processing Unit (GPU).
- Method for using standard rendering light sources [Epi23a; Uni24] of rendering engines [Uni23; Epi23b] for real-time matrix headlight simulation.
- Formulation of the simulation method independent of lighting technology. The pixellight photometric data must only be available in an IES-file-like [IES20] data format.
- Extension of SpMV to incrementally update beam patterns during simulation to reduce calculation time.
- Allowing the simulation to be adapted and validated by reproducing the real headlamp illumination like a digital twin.
- Enabling rapid prototyping of dynamic lighting functions in simulation and reality.

The novel real-time matrix headlight control algorithm is called SuperSampling Control (SSC). The primary novel contributions of SSC are:

- Enabling the intuitive and efficient desired lighting design of arbitrary lighting functions by parameterizing rectangular 3-Dimensional (3D) target objects, so-called primitives.
- Combination of computer graphics ray tracing [Ake+18; HA19; AW21] and MapReduce techniques [ZP09] for real-time control of matrix headlights with arbitrary beam patterns and lighting functions.
- Enabling the identification of efficient headlight control strategies for specific lighting functions through SSC parameter optimization.
- Creating a feedback control loop to adjust symbol projection on uneven surfaces.

The proposed real-time simulation method offers physical accuracy and computational efficiency and the real-time SSC algorithm enables new possibilities for advanced lighting functions. The See the Optimal Lighting (SOL) development tool integrates these methods into a comprehensive framework for automotive lighting development by allowing real-time rapid prototyping of novel lighting functions for automated driving. As SOL can control real headlights, a lighting function developed in SOL can be directly validated with a real test vehicle. The proposed methods have been validated by testing with real matrix headlights, demonstrating their practical applicability in developing automotive lighting. Fig. 1.2 shows a visualization of lighting functions in SOL, including dynamic shadows and volumetric fog.

Figure 1.2.: SOL virtual environment impressions. Active lighting functions are selective lane and sidewalk illumination. In addition, the headlights project reversing arrows on the oncoming lane, a warning symbol in front of speed bumps and the planned route for turning right. A fog-like volume illumination simulation makes the beam spread visible [Wal+23b].

This thesis is structured as follows: Ch. 2 reviews related work in matrix headlight simulation and Ch. 3 reviews related work in headlight control. Ch. 4 introduces the novel SOL headlight development tool, detailing its architecture and capabilities. The proposed simulation method's theoretical foundations and implementation details are presented in Ch. 5 and Ch. 6 evaluates the simulation method. Ch. 7 shows details of SSC and Ch. 8 provides control evaluation results, including quantitative performance metrics and qualitative assessments of illumination quality. Ch. 9 evaluates the usability of SOL and shows possible application scenarios. Finally, Ch. 10 concludes with a summary of contributions and an outlook for future research directions.

# 2

# Fundamentals for Real-Time Matrix Headlight Simulation‡c‡t

This chapter presents the related work of real-time simulation of matrix headlights. The fundamental photometric quantities and terminology of matrix headlights, methods for visualizing and rendering lighting in real-time and a reference approach for simulating matrix headlights are explained. In addition, an overview of SpMV on GPUs is given, which is used to simulate matrix headlights efficiently.

## 2.1. Basic Photometric Quantities and Formulas

Visible light is a subset of the electromagnetic spectrum perceptible to the human eye. The photometric quantities are derived from radiometric measurements weighted by the human visual system's spectral sensitivity [WW00; RA00; Rye97; RR14; Kob19].

The luminous flux $\Phi_\mathrm{v} \in \mathbb{R}_{\geq 0}$ in lumens (lm) is the total power of a light source as perceived by the human eye [RR14]. $\Phi_\mathrm{v}$ is calculated by integrating the spectral radiant flux over all wavelengths $\lambda \in \mathbb{R}_{\geq 0}$, with each wavelength contribution weighted by the luminous efficiency function $V(\lambda)$. This integration translates the physical power of electromagnetic radiation into a measure of its visual effectiveness. $V(\lambda)$ varies depending on differences in human visual perception under various conditions, such as photopic (daylight) and scotopic (low-light) vision and age.

The solid angle $\Omega \in \mathbb{R}_{\geq 0}$ in steradians (sr) quantifies the proportion of the full spherical Field of View (FoV) occupied by an object or illuminated area [RA00; RR14]. $\Omega$ is defined as the ratio of the illuminated area $A \in \mathbb{R}_{\geq 0}$ on a spherical surface to the square of the distance $d \in \mathbb{R}_{\geq 0}$ from the origin of this sphere to the area as

$$\Omega = \frac{A}{d^2}. \tag{2.1.1}$$

The luminous intensity $I_\mathrm{v} \in \mathbb{R}_{\geq 0}$ in candelas (cd) quantifies the emitted light power in a particular direction [RR14]. $I_\mathrm{v}$ is defined as the luminous flux per solid angle in a specified direction as

$$I_\mathrm{v} = \frac{\Phi_\mathrm{v}}{\Omega}. \tag{2.1.2}$$

Spherical coordinates around the photometric center describe the spatial LID. The horizontal angle $\phi \in \mathbb{R}$ and the vertical angle $\psi \in \mathbb{R}$ are measured clockwise concerning the photometric beam axis, adhering to the left-hand rule convention.

The illuminance $E_\mathrm{v} \in \mathbb{R}_{\geq 0}$ in lux (lx) indicates the intensity of incident light on a surface [RR14]. $E_\mathrm{v}$ is defined as the luminous flux incident on an illuminated object divided by the area of that object as

$$E_\mathrm{v} = \frac{\Phi_\mathrm{v}}{A}. \tag{2.1.3}$$

When the light emitter can be approximated as a punctiform light source of infinitesimal size, which is often the case in real-time lighting simulations [Uni24; Epi23a], the relationship between $E_\mathrm{v}$, $I_\mathrm{v}$ and distance is governed by the photometric inverse-square law [RA00]. This law can be derived by combining (2.1.1) and (2.1.2) with (2.1.3) to

$$E_\mathrm{v} = \frac{\Phi_\mathrm{v}}{A} = \frac{I_\mathrm{v}\,\Omega}{A} = \frac{I_\mathrm{v}\frac{A}{d^2}}{A} = \frac{I_\mathrm{v}}{d^2}. \tag{2.1.4}$$

When the receiving surface is not perpendicular to the incident light direction, the illuminance must be adjusted by the cosine of the impact angle $\theta \in \mathbb{R}$ between the surface normal vector and the light ray. This adjustment accounts for the change in the effective area on the spherical surface [RA00] as

$$E_\mathrm{v} = \frac{I_\mathrm{v}}{d^2}\cos(\theta). \tag{2.1.5}$$

The distance at which an arbitrary light emitter can be approximated with a negligible error as a punctiform light source is known as the limiting photometric distance [RR14]. According to Ryer [Rye97], this distance should be "greater than five times the largest dimension of the [emitting area of the light] source" to ensure the correctness of the inverse-square law approximation.

Lastly, the luminance $L_\mathrm{v} \in \mathbb{R}_{\geq 0}$ in candelas per square meter quantifies the perceived brightness of an object's surface and relates to the brightness perceived by the human eye [RR14; Rye97]. $L_\mathrm{v}$ is defined as $I_\mathrm{v}$ per surface $A$ in a specific direction and within a specific $\Omega$ containing the light in that direction as

$$L_\mathrm{v} = \frac{\Phi_\mathrm{v}}{A\,\cos(\theta)\,\Omega} = \frac{I_\mathrm{v}}{A\,\cos(\theta)}. \tag{2.1.6}$$

## 2.2. Fundamental Matrix Headlight Terminology

Various basic terms and lighting functions are first defined to avoid confusion and ambiguity regarding the vehicle lighting terms. A headlamp is a light-emitting device that actively illuminates the environment. A headlight denotes the beam pattern or LID distributed by the headlamp.

The structural composition of a headlamp typically involves multiple light modules. Their beam patterns are superposed to create the illumination in front of the vehicle [Ros+18; Gro+18; Url+21; Por22; Won+24; Klä+18]. This modular approach allows precise and efficient control over the overall beam pattern. Each module can comprise one or a group of individual light sources positioned behind a single outer lens. When these light sources are

arranged in a grid-like configuration of rows and columns, they form a matrix headlamp module.

Matrix headlamps comprise several thousand to 1.3 million individual light sources per module [Ros18; Gro+18; Erk+24]. Individual light sources are called pixellights within the matrix headlamp. However, the term picture-element (pixel) is commonly used in industry parlance. The term pixel describes the unique beam pattern or LID produced by a single pixellight, analogous to the relationship between headlamp and headlight. The pixel utilization $u_{\mathrm{p},j} \in \mathbb{R}_{\geq 0 \wedge \leq 1}$ quantifies the relative brightness of the $j$-th pixel compared to its maximum potential. To avoid ambiguity, the term texture-element (texel) describes an element of a digital image or texture, distinguishing it from the automotive lighting context of pixels. This terminology is borrowed from computer graphics, where textures are images applied to virtual object surfaces to define their visual characteristics. Fig. 2.1 shows the foundational definitions and their relationships.

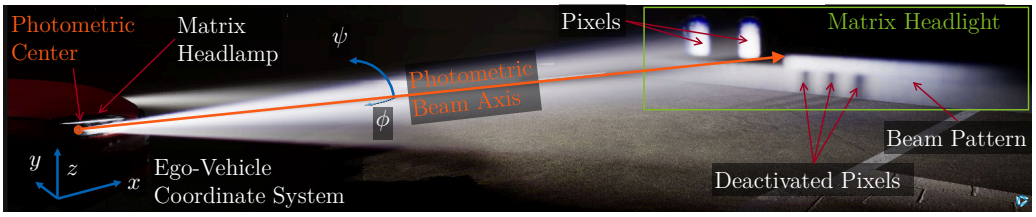
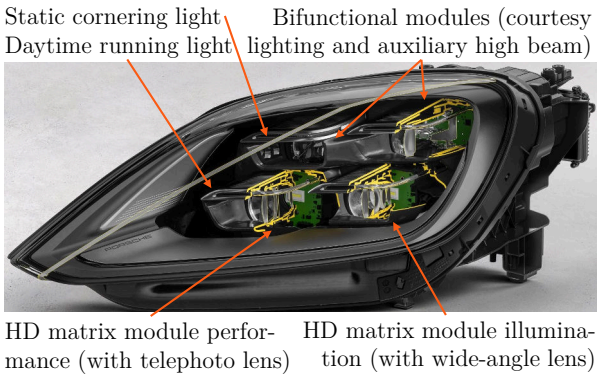


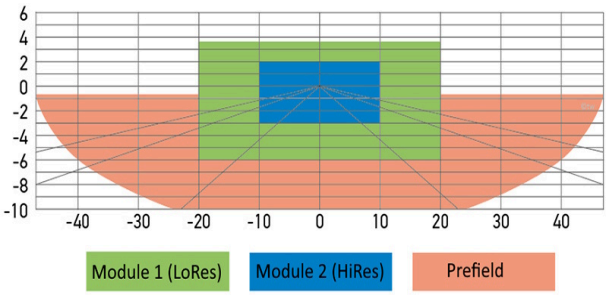

Figure 2.1.: Fundamental matrix headlight definitions and terms.

Implementing pixellights in modern matrix headlights uses diverse lighting technologies [Moi15; BK17; Gro+18; Knö+19]. These include, but are not limited to Light-Emitting Diodes (LEDs) [Hum10; For23a; Tro+19], liquid crystal displays [Hes15; DF17] and Digital Micromirror Devices (DMDs) [SA18; Aud22a; MB24].
This thesis at hand focuses on beam patterns of arbitrary lighting technologies. Hence, details of the lighting technologies are irrelevant and will not be described here. The proposed novel control and simulation methods can be applied as long as the pixellight's LID can be described by Illuminating Engineering Society (IES)-file-like formats.

Fig. 2.2 illustrates a real LED matrix headlamp, representing one of the two real matrix headlights used in the evaluation. This Solid State Lighting High Definition (SSL|HD) headlamp [Kle+22] comprises five main modules, including two lower High Resolution (HiRes), two Low Resolution (LoRes) HD matrix modules and one static cornering light module. The HiRes and LoRes HD matrix modules use a SSL|HD chip by Forvia-Hella GmbH [For23b] with each $256 \times 64 = 16{,}384$ pixellights and an aspect ratio of 4:1. The lighting from the modules is further refined through a multi-lens optical system to achieve the desired LID in front of the vehicle. Fig. 2.2b shows that the HiRes module has half the horizontal and vertical FoV of the LoRes module, which results in a higher angular resolution of 0.078° of the HiRes module compared to 0.156° of the LoRes module. The two upper modules with 6 LEDs of SSL|HD matrix headlamp create the courtesy lighting and auxiliary high beam and the two lower HiRes and LoRes LED SSL|HD matrix modules with 16,384 pixellights realize the HD main beam [Por22; Por23a].




(a) Matrix headlamp with one cornering and four main modules [Por23b; Por23a].




(b) Module illumination areas [For24]. The two lower HD matrix modules are HiRes and LoRes.

Figure 2.2.: SSL|HD matrix headlamp with two upper modules with 6 LEDs and two lower HiRes and LoRes LED matrix modules with each 16,384 pixellights [Por22; Por23a].

## 2.3. Real-Time Lighting Calculation for Rendering

The rendering process transforms in computer graphics 3D scene definitions into 2-Dimensional (2D) images [Kaj86; MH16]. 3D object geometric data, camera viewpoint specifications, surface textures, lighting configurations and shading parameters define a 3D scene. The rendering pipeline consists of several computational phases and uses computer programs called shaders [Ups89; Ros+09]. The geometry processing phase transforms 3D mesh vertices (junction points of object mesh edges) onto a 2D image plane. The rasterization converts geometric primitives, which typically are mesh triangles, into the image texel grid and the shading determines the visual properties of each image texel based on material properties, lighting and textures [Ake+18].

A fundamental principle underlying lighting calculations during rendering is the intercept theorem [Zac30; AF14; Ake+18]. The intercept theorem states that the resulting segments are divided proportionally when a pair of parallel lines intersect two crossed lines. This ratio determines how objects at different distances from the viewpoint should be scaled in the rendered image. This scaling is used for perspective transformations and pinhole camera models [Bus03; Leh+23].
The intercept theorem can be formulated with two rays emanating from a common point $\boldsymbol{p}_1 \in \mathbb{R}^3$ intersected by a pair of parallel lines. $\boldsymbol{p}_2, \boldsymbol{p}_3 \in \mathbb{R}^3$ are points on the first ray and $\boldsymbol{p}_4, \boldsymbol{p}_5 \in \mathbb{R}^3$ are points on the second ray, so that $\boldsymbol{p}_2$ and $\boldsymbol{p}_4$ lie on one parallel line, while $\boldsymbol{p}_3$ and $\boldsymbol{p}_5$ lie on the other. Fig. 2.3 shows the rays, lines and points. The intercept theorem states that the following ratios of distances are equal as

$$\frac{|\boldsymbol{p}_2 - \boldsymbol{p}_1|}{|\boldsymbol{p}_3 - \boldsymbol{p}_1|} = \frac{|\boldsymbol{p}_4 - \boldsymbol{p}_1|}{|\boldsymbol{p}_5 - \boldsymbol{p}_1|} = \frac{|\boldsymbol{p}_4 - \boldsymbol{p}_2|}{|\boldsymbol{p}_5 - \boldsymbol{p}_3|}. \tag{2.3.1}$$

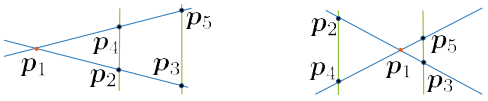



Figure 2.3.: Intersection of parallel lines with two rays from the same origin $\boldsymbol{p}_1$.

The intercept theorem is used in computer graphics for projective texture mapping to simulate a non-uniform LID [Seg+92; Eve01]. This method projects a light texture onto a

scene, analogous to how a slide projector casts an image. The projected texture is the light source LID, called gobo or cookie, similar to physical cutouts and masks used in theater and film lighting [Ake+18]. For light sources confined to a frustum shape with a FoV below 90°, projective texture mapping directly modulates the emitted $I_\mathrm{v}$ at a world point according to the corresponding texture value. This modulation enables the creation of arbitrarily shaped LIDs and video projector effects. Fig. 2.4 from Lecocq et al. [Lec+99] shows a simple headlight simulation with a projected light texture.

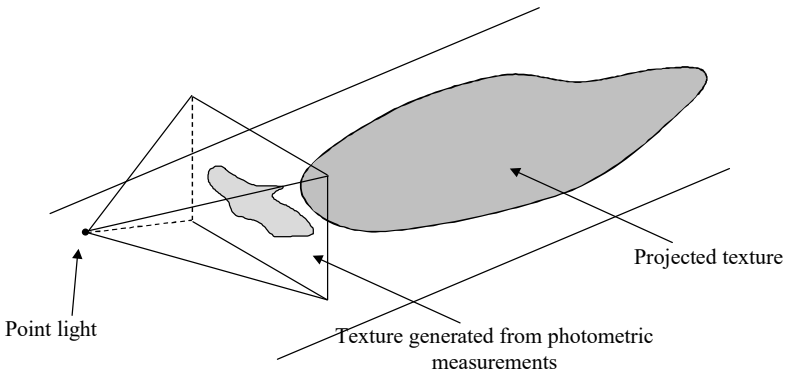



Figure 2.4.: Projective texture mapping of headlight light texture [Lec+99].

Projective texture mapping uses a perspective transformation of world points $\boldsymbol{p}_i \in \mathbb{R}^3$, typically mesh vertices, into the texture space of the projector as 2D texture $u, v$ coordinates. When the projector represents a light source, the texture color at the projected texture $u, v$ coordinates is used to modulate the light source's maximal luminous intensity $I_{\mathrm{v,m}} \in \mathbb{R}_{\geq 0}$, determining the resulting $I_\mathrm{v}$ emitted to the world point. The perspective transformation uses homogeneous coordinates and a perspective projection matrix $\boldsymbol{T}_{\mathrm{wo,pers}} \in \mathbb{R}^{4\times 4}$ that encapsulates the projector's properties [Ake+18; Leh+23; Ber+03; Ber05]. $\boldsymbol{T}_{\mathrm{wo,pers}}$ is derived from both intrinsic and extrinsic parameters of the projector, including its position, orientation, field of view and aspect ratio [Ake+18].
An example of perspective transformation is transforming a world point $\boldsymbol{p}_i = [x\ y\ z]^\top$ onto a plane at a distance $d$ in the negative $x$-direction from the origin [Ake+18]. The $x$-axis is the photometric beam axis and is oriented away from the camera like a depth. The transformation process mirrors a pinhole camera model with the pinhole at the origin as

$$\boldsymbol{T}_{\mathrm{wo,pers}} \left[\boldsymbol{p}_i\ 1\right]^\top = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1/d & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -x/d \end{bmatrix} \xRightarrow[\text{fourth component}]{\text{Division by}} \begin{bmatrix} -d \\ -y\,d/x \\ -z\,d/x \\ 1 \end{bmatrix}. \tag{2.3.2}$$

The resulting texture $u, v$ coordinates are $u = -y\,d/x$ and $v = -z\,d/x$, which are negative due to the plane's placement at a negative $x$-position. The multiplication of world coordinates by $d/x$ in (2.3.2) is analogous to the application of the intercept theorem from (2.3.1), where $d$ and $x$ lie on one ray, while $y$ and $z$ are on parallel lines. The distance $d$ of the projection plane can be calculated from the maximum of the horizontal FoV $\phi_\mathrm{fov} \in \mathbb{R}_{\geq 0}$ and the vertical FoV $\psi_\mathrm{fov} \in \mathbb{R}_{\geq 0}$ of the projector or light source as

$$d = \frac{1}{\tan\left(\max\left\{\phi_\mathrm{fov}, \psi_\mathrm{fov}\right\}/2\right)}. \tag{2.3.3}$$

This formulation for textures with a square aspect ratio can be adapted for arbitrary aspect ratios by treating horizontal and vertical components independently [Leh+23].

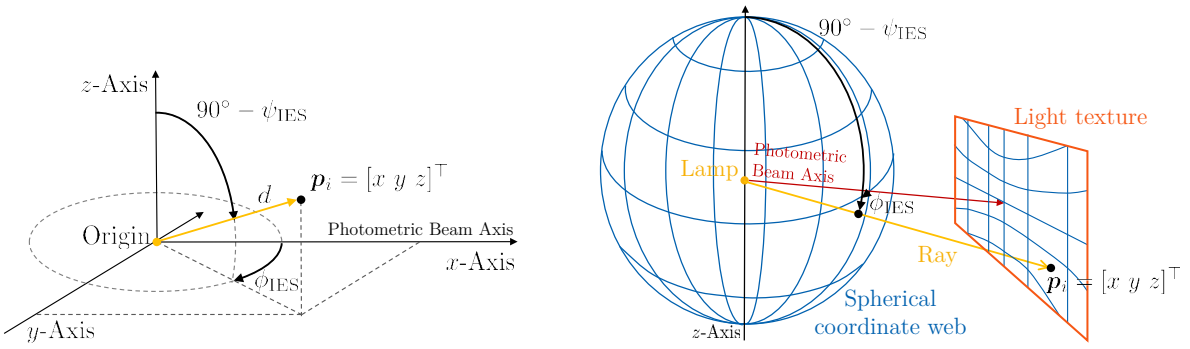
The LIDs of headlights are typically measured using goniophotometers, which capture the emitted $I_v$ at various angles. The resulting spherical LID is often stored in IES photometric data files [IES20], which describe the LID as a function of the horizontal $\phi_{\text{IES}}$ and vertical $\psi_{\text{IES}}$ angle in discrete spherical coordinates, with the origin at the photometric center of the headlight. In the IES file standard for a type A lamp [IES20], the beam angle $\phi_{\text{IES}}$ is counted clockwise and $\psi_{\text{IES}}$ upwards from the photometric beam axis.
A coordinate transformation between Cartesian and spherical coordinates is necessary to use IES data in rendering with projective texture mapping, which requires a planar texture. The transformation from spherical to Cartesian coordinates [Doe61] is

$$\begin{aligned} x &= d \sin(90^\circ - \psi_{\text{IES}}) \cos(\phi_{\text{IES}}) \\ y &= -d \sin(90^\circ - \psi_{\text{IES}}) \sin(\phi_{\text{IES}}) \\ z &= d \cos(90^\circ - \psi_{\text{IES}}), \end{aligned} \tag{2.3.4}$$

with the distance $d$ of the target point $\boldsymbol{p}_i = [x\ y\ z]^\top$ to the photometric center of the headlight. From Cartesian to spherical coordinates, the transaction [Doe61] is

$$\begin{aligned} d &= \sqrt{x^2 + y^2 + z^2} \\ \psi_{\text{IES}} &= 90^\circ - \arccos\left(z/d\right) \\ \phi_{\text{IES}} &= \arctan\left(-y/x\right). \end{aligned} \tag{2.3.5}$$

Fig. 2.5a shows the point $\boldsymbol{p}_i$ in a spherical coordinate system to visualize the quantities in (2.3.4) and (2.3.5). Fig. 2.5b shows the projection of the spherical coordinate headlamp measuring system [UNE23a] onto a screen to visualize the connection between Cartesian and spherical coordinates.




(a) Point $\boldsymbol{p}_i$ in spherical coordinate system. (b) Headlamp measuring system [UNE23a].

Figure 2.5.: Connection between Cartesian and spherical coordinates.

The simulation of automotive headlights can be achieved by projective texture mapping from a punctiform light source [Usó+99; Lec+99; LS02; Ber05; Ber+03]. This approach needs a generated plane headlight light texture or LID texture from photometric measurement data. Berssenbrügge [Ber05] create the light texture by projecting each $I_v$ point from the spherical photometric data format onto a plane where the target light texture is placed (see Fig. 2.5b).

First, the process sets the texel size of the light texture to correspond with the minor angle step in the photometric data. Consequently, the number of texels of the light texture and its overall size are determined by the maximum angle of the photometric data.
The light texture is then populated with photometric data by projecting the spherical coordinates onto the nearest texel in the light texture. The projection is governed by the transformation from spherical to Cartesian coordinates of (2.3.4) and illustrated in Fig. 2.5. To ensure continuity, empty texels between projected data points are filled using curve fitting like linear interpolation. Subsequently, the maximum luminous intensity $I_{v,m}$ in the texture is determined and set as the virtual light source $I_v$. Finally, all texels in the texture are normalized by dividing by $I_{v,m}$ to obtain the final relative brightness values.
In the simulation phase, the illuminance at a given point is calculated by multiplying the relative brightness of a texel with the light source $I_{v,m}$ and an attenuation factor depending on the distance between the photometric center and the target point. For a punctiform light source, which is the typical light type in computer graphics, the attenuation factor is calculated using the photometric inverse-square law [RA00] in (2.1.4). Fig. 2.6 from Berssenbrügge et al. [Ber+03] shows a lighting simulation on a vertex using a light texture as a projection slide.

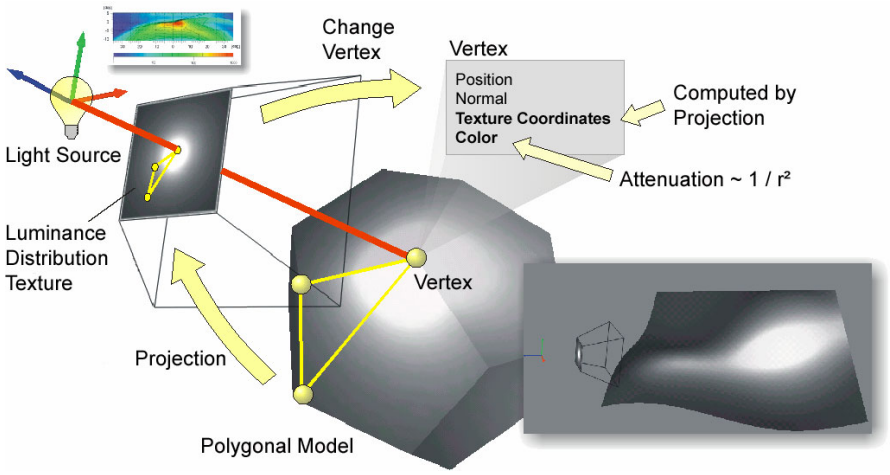



Figure 2.6.: Lighting from a LID texture onto a vertex. The vertex is the radius $r \in \mathbb{R}_{\geq 0}$ away from the light origin [Ber+03].

## 2.4. Linear Superposition of Matrix Headlamp Pixellights

The human visual system's subjective perception of brightness exhibits a nonlinear relationship with the incident light on the eye [GW17]. However, the total $E_v$ at a given point follows the principle of linear superposition and is the arithmetic sum of the $E_v$ contributions from each light source [Jia+20]. Therefore, combining the LIDs of individual light sources in a matrix headlight can be achieved through the addition of the individual pixel maximal bright LIDs weighted by their current utilization [SN18; Rüd+19a]

The simulation of matrix headlights with dynamic LID involves a three-step process [LS02; Ber05; Rüd+19a; Rüd22]. The initial offline preprocessing step transforms the photometric data of all pixels into 2D light textures of every pixel, where each texel represents an $I_v$

value at a specific angular coordinate (see Sec. 2.3). When headlight color information is available, the light texture has four channels for Red-Green-Blue (RGB) and $I_v$ values [Rüd22]. A single-channel grayscale texture stores $I_v$ with no color information.

The LID combination step realizes the dynamic lighting of matrix headlights under varying driving conditions. At runtime, the simulation combines the LID of all $n_p \in \mathbb{N}$ individual pixellights as a weighted superposition by their respective utilization [SN18; Rüd+19a]. Utilization is the relative operating $I_v$ of the pixellight, which is dynamically adjusted by the lighting controller based on the current traffic situation. The combination result is a single, cumulative LID of the aggregate lighting of all pixels.
For the $i$-th texel in the cumulative light texture, the combining step multiplies the maximum possible $I_v$ from the $j$-th pixellight light texture corresponding to that texel with the $j$-th pixel utilization. The weighted $I_v$ is added to the current value of the cumulative light texture of the $i$-th texel. The weighted $I_v$ sum of $n_p$ pixels is calculated as

$$I_{v,i} = \sum_{j=0}^{n_p-1} I_{v,m,i,j}\, u_{p,j}, \tag{2.4.1}$$

where $n_p$ is the pixel count of the matrix headlight, $I_{v,i} \in \mathbb{R}_{\geq 0}$ is $I_v$ of the $i$-th texel in the cumulative LID, $I_{v,m,i,j} \in \mathbb{R}_{\geq 0}$ is the maximum possible $I_v$ of the $j$-th pixellight on the $i$-th texel and $u_{p,j} \in \mathbb{R}_{\geq 0}$ is the utilization of the $j$-th pixellight. Fig. 2.7 from Rüddenklau et al. [Rüd+19a] illustrates the creation of low-beam and high-beam LIDs of the HD84 84-pixel matrix headlight [Kal+16] through the scaled addition of pixels.

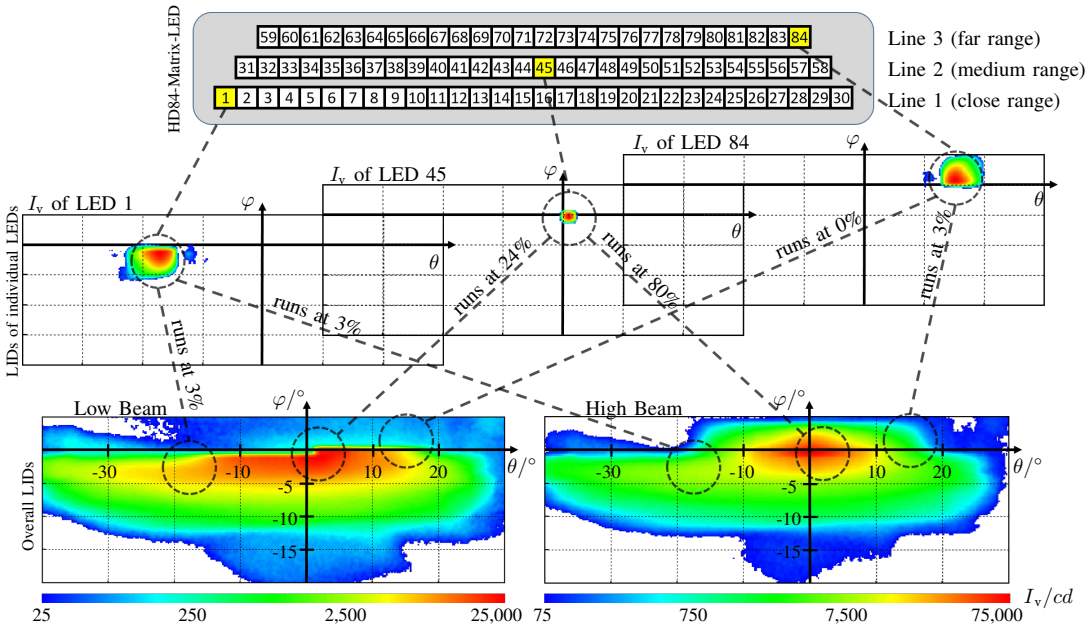
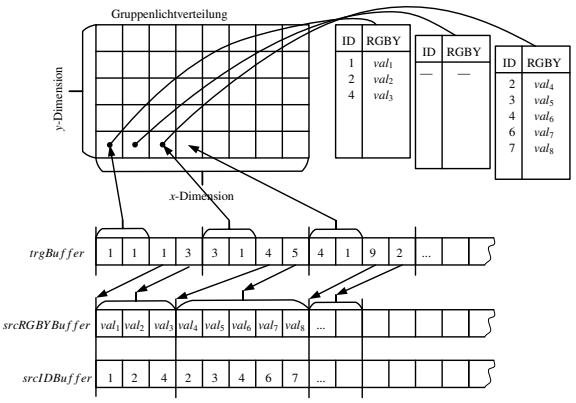



Figure 2.7.: Superposition of pixellight LIDs of a 84-pixel matrix headlight. A low-beam LID is shown at the bottom left and high-beam LID at the bottom right [Rüd+19a].

The combination step for each texel in (2.4.1) has a computational complexity of $\mathcal{O}(n_p)$ [Rüd22]. Therefore, the overall execution time is proportional to $n_p$ and the number of texels $n_t \in \mathbb{N}$. To reduce computational time, Rüddenklau [Rüd22] introduced a data structure called the LGC format. LGC eliminates unnecessary multiplications with zero values in (2.4.1) and reduces the complexity of the combination step to $\mathcal{O}(n_{p,i})$ for each

texel with the number of pixels $n_{\mathrm{p},i} \in \mathbb{N}$ that contribute non-zero illumination $I_{\mathrm{v,m},i,j} > 0\,\mathrm{cd}$ to the texel.
LGC consists of three primary array data structures, called *trgBuffer*, *srcRGBYBuffer* and *srcIDBuffer* [RW20; Rüd22]. LGC assumes that the LID cumulative textures have two dimensions as $u$ and $v$, so the *trgBuffer* has for each texel four entries, namely the $u$- and $v$-coordinates of the texel, an offset to the first entry in the *srcRGBYBuffer* for this texel and the number of pixels $n_{\mathrm{p},i}$ that illuminate the texel. The *srcRGBYBuffer* contains $I_{\mathrm{v}}$ and RGB-color values of the pixels that illuminate each texel and the *srcIDBuffer* contains the indices of the pixels that illuminate each texel.
The LGC algorithm for calculating $I_{\mathrm{v},i}$ retrieves for each texel in the cumulative light texture the start offset and $n_{\mathrm{p},i}$ from the *trgBuffer* for the current texel. For each of the $n_{\mathrm{p},i}$ pixels, LGC retrieves RGB and $I_{\mathrm{v}}$ values from the *srcRGBYBuffer* and obtains the pixel index from the *srcIDBuffer* to access the corresponding utilization factor. The pixel $I_{\mathrm{v}}$ is multiplied by the utilization factor and accumulated to obtain the final value of the texel [RW20; Rüd22]. Fig. 2.8 from Rüddenklau [Rüd22] shows the data structure and algorithm of the LGC format.




(a) LGC data structure [Rüd22].

**require:** $instID \in \mathbb{N}$, $dimValBuffer \in [0,1]^K$, $trgBuffer$ as Buffer of $\mathbb{N}^4$, $srcRGBYBuffer$ as Buffer of $\mathbb{R}^4$, $srcIDBuffer$ as Buffer of $\mathbb{N}$, $groupLightDist \in (\mathbb{R}^4)^{M_g \times N_g}$
**function** COMBINEGROUP($instID, dimValBuffer$)
**local variables:** $trgLoc \in \mathbb{N}^2$, $srcBufferOffset, srcBufferCount \in \mathbb{N}$, $srcVal, tmpVal \in \mathbb{R}^4$, $srcID \in \mathbb{N}$
$trgLoc.m \leftarrow trgBuffer[instID][1]$
$trgLoc.n \leftarrow trgBuffer[instID][2]$
$srcBufferOffset \leftarrow trgBuffer[instID][3]$
$srcBufferCount \leftarrow trgBuffer[instID][4]$
$tmpVal \leftarrow (0,0,0,0)$
**for** $i = srcBufferOffset$ to $srcBufferOffset + srcBufferCount - 1$ **do**
$srcVal \leftarrow srcRGBYBuffer[i]$
$srcID \leftarrow srcIDBuffer[i]$
$tmpVal \leftarrow tmpVal + dimValBuffer[srcID] \cdot srcVal$
**end for**
$groupLightDist[trgLoc] \leftarrow tmpVal$
**end function**

(b) Group combiner shader pseudocode [Rüd22].

Figure 2.8.: Data structure and algorithm of the LGC format [RW20].

In the final simulation step, the rendering pipeline uses the combined LID to calculate the illumination of the virtual world. This step applies the combined LID to the virtual environment by loading the combined LID into a virtual light source and optional adjusting the light source parameters [LS02; Ber05; Rüd+19a].

## 2.5. Sparse Matrix-Vector Multiplication

This thesis uses for the SpMV implementation for matrix headlight simulation the Compute Unified Device Architecture (CUDA) parallel computing platform [Nic+08; Nvi23c] on a Nvidia GPU. CUDA uses a hierarchical organization of GPU threads, blocks and grids, each with associated memory spaces, as shown by Fig. 2.9. At the finest granularity, a GPU thread is the basic execution unit, with local memory and registers for rapid data access. A CUDA kernel is a GPU program and is executed parallelized by all assigned GPU threads similar to a shader [Nvi23b]. Threads are aggregated into blocks, facilitating cooperation and sharing data among threads through a fast shared memory space. At the

highest level, blocks are organized into grids connected to the device's slow global memory [Nvi23b]. CUDA cores are individual processing units that simultaneously execute threads' operations in a warp, which are thread groups of fixed size [LG18; Nvi23b].
A coalesced memory access reduces memory transaction overhead and bandwidth using warp combined operations [Har13a; Nvi23b]. For a coalesced memory access, the GPU threads in a warp access consecutive or aligned memory addresses in global memory simultaneously. This way, the GPU combines the requests of multiple threads into fewer memory transactions and does not transfer unnecessary data, decreasing the memory bandwidth consumption.

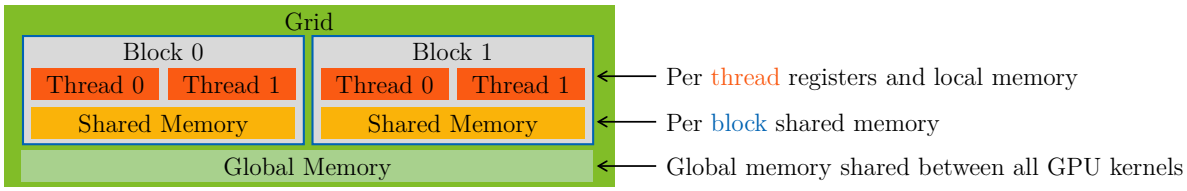



Figure 2.9.: CUDA's hierarchical memory organization [Nvi23b].

A matrix-vector multiplication is computationally intensive for large matrices. By avoiding the multiplications with zeros as a SpMV, the required memory for storing the sparse matrix and the computation time can be reduced. A matrix is called sparse if most elements are equal to zero, but there is no general threshold value. In a dense matrix, most of the elements are non-zero.
An efficient SpMV requires a suitable storage format for the sparse matrix and parallel row vector multiplication algorithm. There are many possible formats for sparse matrices known, each with its advantages and disadvantages for different matrix sparsity patterns [Im00; Saa03; BG09; BG08; BB09; Dal+15; Ste+16; Fil+17; Li+20; Xia+23].

The Coordinate (COO) format [Saa03; BG08; BG09] stores as shown in Fig. 2.10 a matrix using three arrays of equal length of the number of matrix non-zero entries: *RowIndices*, *ColumnIndices* and *Values*. *RowIndices* stores the row indices of the non-zero entries, *ColumnIndices* stores their column indices and *Values* the entry values.
The sparsity pattern has little influence on COOs performance [BG09]. However, COO is not memory efficient because it requires memory of three times the number of non-zero entries.

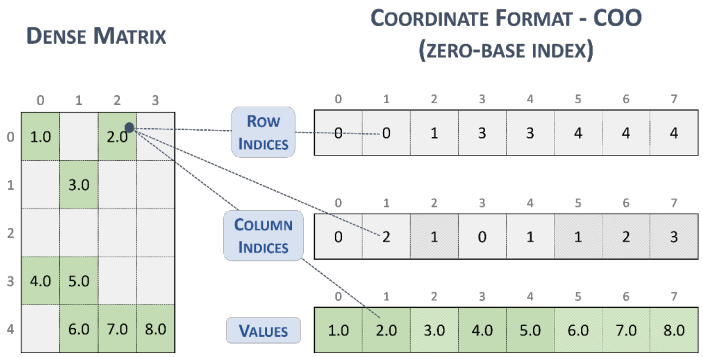



Figure 2.10.: COO format of a $5 \times 4$ matrix [Nvi23d].

The Compressed Sparse Row (CSR) format [Saa03; BG08; BG09] is more memory-efficient than COO and consists as shown in Fig. 2.11 of three arrays: *RowOffsets*, *ColumnIndices*

and *Values*. *RowOffsets* has the length of the number of rows plus one and *ColumnIndices* and *Values* have the length of the number of non-zero entries. *RowOffsets* stores the starting index of each row data block in *ColumnIndices* and *Values* and as an additional last element the total number of non-zero entries. Therefore, the length of a row is the difference between the current and the following element of *RowOffsets*. The index to access the *ColumnIndices* and *Values* arrays for a row is the current element of *RowOffsets* plus an offset, which goes from 0 to the length of the current row minus one. *ColumnIndices* stores the column indices of the non-zero entries in each row and *Values* the values.
CSR reduces memory usage by eliminating the row array compared to COO, and it allows for coalesced memory access with a vector CUDA kernel [BG08]. As a disadvantage, CSR is not designed for irregular sparsity patterns like large variations in the number of non-zero entries per row. A row length difference can lead to thread load imbalance and poor coalescing memory operations, as the threads in a warp do not operate on neighboring memory locations.

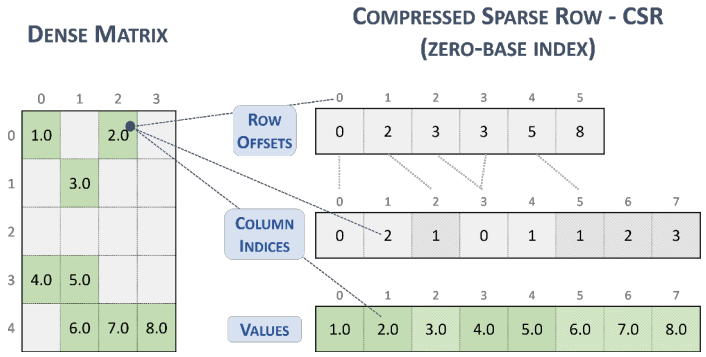



Figure 2.11.: CSR format of a $5 \times 4$ matrix [Nvi23d].

The CSR format can be used in two CUDA kernel implementations: scalar and vector. Fig. 2.12 presents CUDA source code for both CSR scalar and vector CUDA kernels. The scalar kernel assigns one GPU thread per matrix row. In contrast, the vector kernel assigns multiple GPU threads per block to a row and each thread of the block computes a partial sum of the dot product between the row and the vector. Then, the threads in the block use a reduction technique to obtain the final result. Using multiple threads achieves higher parallelism and improved memory coalescing at the cost of increased shared memory usage and synchronization overhead [BG08].

The Ellpack (ELL) format [RB85; Kin+89; Saa03; BG08; BG09] pads rows to a uniform length. ELL consists of two arrays: *ColumnIndices* and *Values*, each of the size of the number of rows times the maximum number of non-zero entries per row. *ColumnIndices* stores the column indices and *Values* the values.
ELL pads the rows with less than the maximum number of non-zero entries with zeros to enable efficient coalesced memory access. Moreover, no GPU thread divergence or imbalance exists. The disadvantage is that ELL wastes memory and computation for irregular sparsity patterns since it pads the rows to match the longest row.
The Sliced Ellpack (SELL) format [Kre+14] slices the matrix into smaller sub-matrices of equal height and a different number of columns and applying the ELL format to each sub-matrix. SELL consists as shown in Fig. 2.13 of three arrays: *SliceOffsets*, *ColumnIndices* and *Values*, where *SliceOffsets* has the length of the number of slices plus

```cuda
__global__ void
spmv_csr_scalar_kernel(const int num_rows,
                       const int   * ptr,
                       const int   * indices,
                       const float * data,
                       const float * x,
                             float * y)
{
    int row = blockDim.x * blockIdx.x + threadIdx.x;

    if(row < num_rows){
        float dot = 0;

        int row_start = ptr[row];
        int row_end   = ptr[row+1];

        for (int jj = row_start; jj < row_end; jj++)
            dot += data[jj] * x[indices[jj]];

        y[row] += dot;
    }
}
```

(a) CSR scalar CUDA kernel.

```cuda
__global__ void
spmv_csr_vector_kernel(const int num_rows,
                       const int   * ptr,
                       const int   * indices,
                       const float * data,
                       const float * x,
                             float * y)
{
    __shared__ float vals[];

    int thread_id = blockDim.x * blockIdx.x + threadIdx.x; // global thread index
    int warp_id   = thread_id / 32;                        // global warp index
    int lane      = thread_id & (32 - 1);                  // thread index within the warp

    // one warp per row
    int row  = warp_id;

    if (row < num_rows){
        int row_start = ptr[row];
        int row_end   = ptr[row+1];

        // compute running sum per thread
        vals[threadIdx.x] = 0;
        for(int jj = row_start + lane; jj < row_end; jj += 32)
            vals[threadIdx.x] += data[jj] * x[indices[jj]];

        // parallel reduction in shared memory
        if (lane < 16) vals[threadIdx.x] += vals[threadIdx.x + 16];
        if (lane <  8) vals[threadIdx.x] += vals[threadIdx.x +  8];
        if (lane <  4) vals[threadIdx.x] += vals[threadIdx.x +  4];
        if (lane <  2) vals[threadIdx.x] += vals[threadIdx.x +  2];
        if (lane <  1) vals[threadIdx.x] += vals[threadIdx.x +  1];

        // first thread writes the result
        if (lane == 0)
            y[row] += vals[threadIdx.x];
    }
}
```

(b) CSR vector CUDA kernel.

Figure 2.12.: CUDA C++ code for CSR scalar and vector CUDA kernels [BG08].

one and *ColumnIndices* and *Values* have the length of the sum over all slices of its number of rows times the maximum number of non-zero entries per row of this slice. *SliceOffsets* stores the starting index of each slice in the *ColumnIndices* and *Values* and as the additional last element the total number of elements in SELL. *ColumnIndices* stores the column indices of the entries in each slice and *Values* the values. The number of columns of a slice can be calculated by dividing the difference between the end and start slice offset through the slice height.

By adapting to the sparsity pattern with sub-matrices, SELL reduces the memory usage compared to ELL for irregular sparsity while preserving performance benefits like coalesced memory access.

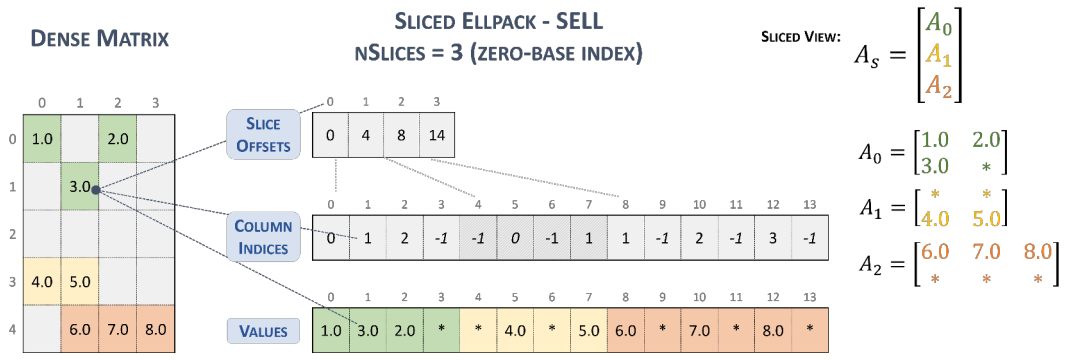



Figure 2.13.: SELL format of a $5 \times 4$ matrix with three slices [Nvi23d].

# 3

# Fundamentals for Real-Time Matrix Headlight Control‡c‡t

This chapter presents the related work of matrix headlight control. First, the foundational methodologies for the proposed novel control algorithm are explained: ray tracing, Supersampling Anti-Aliasing (SSAA) and the MapReduce data processing paradigm. Subsequently, an overview of lighting functions and established matrix headlight control approaches is given.

## 3.1. Ray Casting and Ray Tracing in Computer Graphics

Ray casting and ray tracing are techniques in the field of computer graphics for generating photorealistic images of 3D scenes [Ake+18; HA19; AW21]. The variants of ray tracing differ in the modeling of light-surface interactions in order to achieve different degrees of photorealism.

### 3.1.1. Ray Casting

Ray casting operates by projecting rays from an origin point, typically an eye or camera, into a virtual scene to determine object visibility and facilitate shading calculations. This method of a simple cast does not inherently account for complex optical phenomena such as reflections, refractions, or global lighting effects and is restricted to simulating direct lighting and rudimentary shading effects [Ake+18; HA19; AW21].
Ray casting projects multiple rays from the projector's origin point into the scene and calculates their interactions. This method is similar to the pinhole camera model and lighting simulation in computer graphics (see Fig. 2.6). The $i$-th ray has a projector origin point $\boldsymbol{p}_{\mathrm{o},i} \in \mathbb{R}^3$. It goes through the $i$-th corresponding texel of the projector, which defines its direction vector $\boldsymbol{d}_i \in \mathbb{R}^3$. The rays are typically cast in a grid pattern across all texels to sample the scene and generate a 2D image. The direction vector $\boldsymbol{d}_i$ is calculated based on the position of the texel on the projector plane relative to the origin and is normalized to unit magnitude. The position $\boldsymbol{p}_{\mathrm{ray},i}(l) \in \mathbb{R}^3$ along the ray at a distance $l \in \mathbb{R}$ from the ray origin is

$$\boldsymbol{p}_{\mathrm{ray},i}(l) = \boldsymbol{p}_{\mathrm{o},i} + l\,\boldsymbol{d}_i. \tag{3.1.1}$$

For each ray, intersection and hit tests are conducted with geometric primitive objects populating the scene to identify the nearest intersecting surface. Shading calculations are executed upon determining this intersection and hit, considering surface properties and lighting information. Hit tests typically involve solving equations specific to the geometric primitives encountered by the ray (3.1.1) [Ake+18]. For instance, the intersection of a ray with a spherical object or an ellipsoid with equal scaling is determined by

$$|\boldsymbol{p}_{\text{ray},i}(l) - \boldsymbol{p}_{\text{o,e}}|^2 = r_{\text{el}}^2 \tag{3.1.2}$$

where $\boldsymbol{p}_{\text{o,e}} \in \mathbb{R}^3$ is the center of the sphere and $r_{\text{el}} \in \mathbb{R}_{\geq 0}$ denotes its radius. $l$ gets negative when the spherical object is behind the origin, so the ray does not hit the ellipsoid.

### 3.1.2. Ray Tracing

Recursive ray tracing extends ray casting by introducing a recursive mechanism for tracing rays after they intersect with objects in the scene as secondary new casts [Com06; Ake+18; HA19; AW21]. This extension to multiple casts per texel enables the simulation of complex optical phenomena, including reflections, refractions and shadows.
The recursive ray tracing algorithm begins with casting primary rays and performing hit tests similar to ray casting. Secondary rays are generated recursively at each hit point to simulate various optical effects. Reflection rays are traced in the direction of mirror reflection from the surface and refraction rays are projected to simulate the entry or exit of light through transparent objects. Shadow rays are cast in the direction of each light source to ascertain whether the intersection point is obscured and thus in shadow. The final shading process aggregates contributions from direct lighting and all secondary rays, including their recursive offspring. This recursive process continues until a maximum recursion depth is reached or a termination condition is satisfied. Fig. 3.1 shows the different ray types of recursive ray tracing like shadow, reflection and refraction rays.

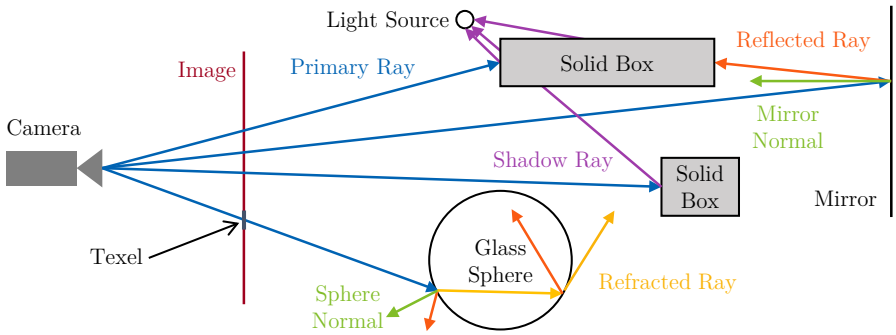



Figure 3.1.: Recursive ray tracing using different ray types. Four primary rays are traced from the camera into the scene, which creates secondary rays [HA19].

The calculation of reflected or refracted ray directions uses the surface normal vector $\boldsymbol{n} \in \mathbb{R}^3$ at the point of intersection of the primary and secondary ray and the incident ray direction $\boldsymbol{d}_i$ [De 06; Tka12; AW21; Tka12; Com06]. The surface normal vector, a unit vector perpendicular to the surface at the reflection point, represents the local orientation of the surface and facilitates the avoidance of trigonometric functions in reflection and refraction calculations.

The direction vector $\boldsymbol{d}_{i+1}$ of the reflected ray, projecting away from the surface, is computed for an ideal mirror reflection by reflecting the incident ray direction vector $\boldsymbol{d}_i$ about the surface normal $\boldsymbol{n}$ according to the law of reflection as

$$\boldsymbol{d}_{i+1} = \boldsymbol{d}_i - 2(\boldsymbol{d}_i \cdot \boldsymbol{n})\boldsymbol{n}, \tag{3.1.3}$$

where $\cdot$ denotes the dot product between vectors and calculates the scalar projection of $\boldsymbol{d}_i$ onto the unit vector $\boldsymbol{n}$. The dot product of two vectors is the product of their magnitudes and the cosine of the angle between them.
The direction of a refracted ray through a transparent surface can also be calculated using $\boldsymbol{d}_i$ and $\boldsymbol{n}$ without trigonometric functions, in conjunction with Snell's law and the ratio of refractive indices [De 06; AW21]. Snell's law stipulates that the ratio of the sines of the angles of incidence and refraction is equal to the inverse ratio of the refractive indices of the media involved.

## 3.2. Supersampling Anti-Aliasing

In computer graphics, anti-aliasing mitigates visual artifacts that arise when rendering scenes at a low resolution [Cro77; She+08; Ake+18]. The aliasing artifacts manifest as jagged edges, stair-stepping effects and moiré patterns (overlapping lines create a new pattern). The cause of aliasing lies in the discrete grid of digital images and the inherent limitations imposed by finite image texel resolutions.
SSAA addresses these aliasing issues using a higher sampling rate than the image resolution. SSAA samples the source at multiple points within each texel and then combines these samples to produce a final, anti-aliased texel value. This process increases the spatial resolution of the sampling, allowing for a more accurate representation of the original continuous signal before it is quantized to the discrete image texel grid [She+08; Ake+18]. The efficacy of SSAA depends on the distribution pattern of the sample points within each texel. Various SSAA patterns have been developed, each with trade-offs regarding visual quality and computational complexity [Ake03; She+08; Ake+18].

One approach for SSAA is raster patterns, such as uniform grid, rotated grid, N-rooks and Quincunx [Wik24; Ake+18]. Fig. 3.2 illustrates these grid patterns.
The most straightforward raster pattern is uniform grid sampling, subdividing each texel into a regular grid and sampling at the center of each image sub-texel. While uniform grid sampling ensures even coverage of the texel area, it can inadvertently introduce regular aliasing patterns due to its uniform distribution, particularly along edges that align with the sampling grid.
The rotated raster pattern rotates the sampling grid by a specific angle, typically 45°. The 45° Rotated Grid SuperSampling (RGSS) pattern reduces directional aliasing by distributing samples evenly across different orientations and breaking up the regular grid alignment, thereby mitigating aliasing artifacts along diagonal edges.
The N-rooks pattern distributes the samples such that no two samples lie in the same row or column of the image texel grid. This arrangement guarantees that each row and column is sampled at least once, balancing coverage and randomness.
The Quincunx pattern places samples at the center and four corners of each texel, forming an arrangement reminiscent of the five on a dice face. Quincunx uses sample sharing,

where each corner sample value is distributed to its four neighboring texels. This sharing mechanism results in an average of only two samples per texel, yet yields results that are considerably superior to conventional two-sample SSAA methods [Nvi01]. Quincunx uses a non-uniform weighting scheme, with the center sample given a weight of $\frac{1}{2}$ and each corner sample assigned a weight of $\frac{1}{8}$.


Figure 3.2.: SSAA grid patterns [Wik24; Ake+18]. The blue texel is sampled at the red locations.

## 3.3. MapReduce for Parallel Data Processing

MapReduce is a method for processing large datasets in parallel across distributed systems [DG04b; DG04a; Ran+07; ZP09]. This method is adequate for tasks that can be divided into independent units of work, which can then be processed in parallel.
The MapReduce method is built upon two primary functions: Map and Reduce. The Map function processes input data and generates a set of intermediate key-value pairs, while the Reduce function aggregates and processes these intermediate results with the same intermediate key to produce a final output.
To illustrate the MapReduce model, an example task is counting the occurrences of each word across a collection of documents [ZP09]. The map function processes each word in the input documents, emitting a key-value pair with the word as the key and the integer one as the value. The reduce function sums all the values for each unique word, producing the total count of occurrences for that word across the entire dataset.

The typical execution of a MapReduce job is illustrated in Fig. 3.3. First, the input dataset is divided into manageable input splits, processed independently by a map worker, allowing for parallel execution. In the map phase, the map workers apply the user-defined map function to each assigned input split to generate intermediate key-value pairs. Afterward, the intermediate key-value pairs with the same key are grouped and transferred to the appropriate reduce worker. In the reduce phase, the reduce workers process the intermediate value data by applying the user-defined reduce function to each unique key and its associated array of values. The reduction computes the final single output per key.

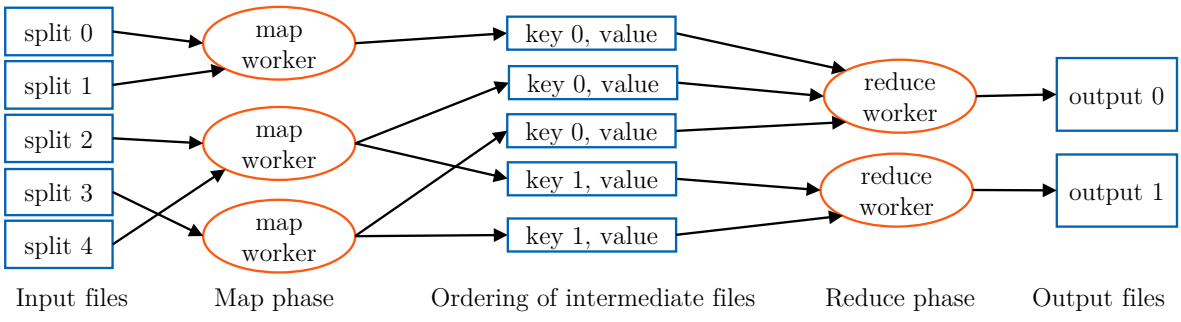
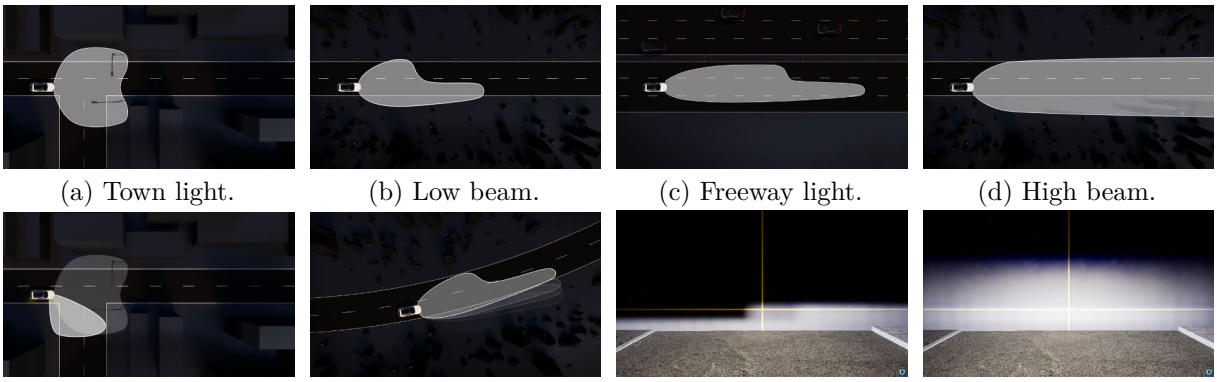



Figure 3.3.: Execution overview of MapReduce [DG04a; ZP09].

## 3.4. Adaptive Front-Lighting Systems

After the definitions of fundamental matrix headlight terminology in Sec. 2.2, lighting functions are divided into the two Adaptive Front-lighting System (AFS) and Adaptive Driving Beam (ADB) categories. An AFS is a lighting device that generates illumination with different properties for automatic adaptation to changing (environmental) conditions [GRE19; UNE23d; UNE23b]. ADB functions extends AFS to dynamic adaption to traffic objects [GRE19].
AFS automatically activates different beam patterns optimized for specific driving scenarios based on various environmental and vehicular parameters, ensuring optimal lighting for the current driving conditions [Wör+07; Rei11; RR14; Win+15; Übl21; Kal+16; Evr+16]. These control parameters include the vehicle speed, presence of street lighting, road surface conditions (e.g., wetness) and the driving environment (urban, rural, highway) [UNE23d].

Some static AFS beam patterns are low and high beam, town light, adverse weather light and freeway light, shown in Fig. 3.4. These patterns are characterized by distinct illumination ranges, beam angles and brightness levels tailored to specific driving scenarios. For example, the town light features a reduced illumination range and broader beam angle to accommodate lower vehicle speeds in urban environments and illuminate sidewalks. The adverse weather light reduces brightness in the forefield to mitigate glare from increased reflections on wet road surfaces. The freeway light provides an extended illumination range and a more focused beam pattern for high-speed driving scenarios.



(a) Town light. (b) Low beam. (c) Freeway light. (d) High beam.

(e) Cornering light. (f) Bend light. (g) Low beam on wall. (h) High beam on wall.

Figure 3.4.: Different AFS beam patterns and functions. Fig. a-f are schematic Birdseye views from Forvia-Hella GmbH (Ed.) [For23a]. Fig. g-h are simulated views of an illuminated projection wall, where the yellow lines are the vehicle angles' zero horizontal and vertical axis.

Beyond static beam pattern selection, AFS incorporate dynamic lighting functions that adjust the orientation of the beam pattern for optimal road illumination. These dynamic functions can be broadly categorized into horizontal and vertical beam control [Wör+07; Rei11; RR14; Win+15; Sha+23].
Vertical dynamic beam control, known as Headlamp Leveling System (HLS) and Headlight Range Control (HRC), involves the vertical adjustment of the beam pattern to compensate for changes in vehicle body pitch This function aims to maintain a consistent illumination height and range, regardless of variations in vehicle loading, road unevenness or speed-induced pitch changes. The vertical adjustment can be achieved through the physical

movement of the lighting module or, in the case of matrix headlamps, through digital shifting of pixel utilization patterns.

Horizontal dynamic beam control, called bend lighting [Zim07; For23a], involves the lateral adjustment of the beam pattern to illuminate the anticipated path of the vehicle through curves and corners. This function can be implemented through mechanical swiveling by physically rotating the lighting module or digital beam steering in matrix headlamps. The digital beam steering approach selectively activates and modulates individual pixellights to achieve the desired bending effect. Furthermore, additional modules can be activated to provide cornering lighting, enhancing visibility and safety during turning maneuvers. The control algorithm for bend lighting typically relies on driver input, such as steering angle, to estimate the vehicle's projected path.

In the case of digital bend lighting, Saralajew and Benderman [SB16] approximate each pixel as its centroid (is explained in Sec. 3.5.1) and establish a relationship between pixel brightness and the horizontal angle of the centroid. This connection creates an angle-intensity function for each matrix headlight row. When a horizontal angle swivels the beam pattern, the centroid angles of all pixels are adjusted accordingly. The new brightness values are calculated using a morphing function, such as linear interpolation, based on the brightness of the neighboring, non-shifted old centroids. For example, when the new centroid horizontal angle is between two non-shifted old ones, the new brightness is the average of the brightness of the two non-shifted centroids when a linear interpolation is used.

An alternative approach involves shifting the pixel utilization matrix toward the target swiveling angle [KP18; PK18; Kan+22]. This method treats the pixel utilization values as a grayscale image. It applies an image translation transformation to achieve the desired bending effect.

## 3.5. Adaptive Driving Beam

The ADB system is designed to optimize visibility for drivers while minimizing glare for other road users. ADB is an extension to AFS, which dynamically adjusts the headlamp beam pattern in response to the presence of oncoming and preceding vehicles, thereby enhancing long-range visibility for the driver without causing discomfort, distraction, or glare to other road users [UNE23c; GRE19].

This thesis at hand proposes an extended definition of ADB systems, encompassing a broader range of traffic objects beyond vehicles. This expanded scope includes pedestrians, traffic signs, streetlamps and animals, leading to the development of Object-Based Lighting (OBL) and Material-Based Lighting (MBL) functions. In OBL, the dynamic adaptation of the LID is contingent upon detecting and classifying various traffic objects within the illumination area [Bar+15; Alm+21]. MBL does not use the object classes, but their surface reflection properties for lighting adaptation [Mül+24c]. Consequently, an AFS equipped with OBL capabilities is considered an ADB system.

### 3.5.1. Glare-Free High Beam

The GFHB function enables the continuous use of high beam without causing glare to other road users by selectively deactivating specific segments of the beam pattern that would otherwise illuminate and potentially dazzle oncoming and preceding vehicles [Göt+07; Wör+07; Sch+09; Hum10; RR14; Win+15; Sha+23; MB19]. This approach maximizes visibility for the driver while ensuring the safety and comfort of other road users.

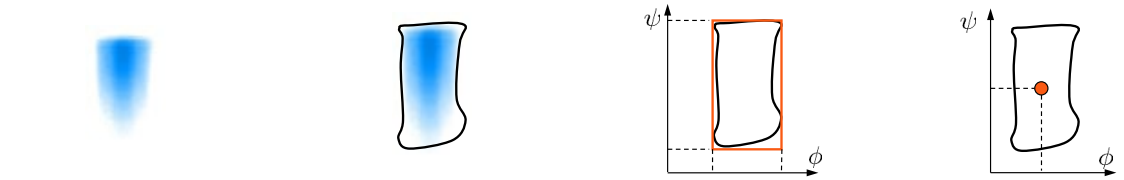
GFHB of matrix headlamps controls primarily software-based pixel utilization, eliminating the need for mechanical components such as shutters or rollers. The GFHB operation can be delineated into several steps [Hum10].
The first step is detecting relevant traffic objects, such as oncoming and preceding vehicles, pedestrians and cyclists [Hum10; GD21]. The system then calculates the bounding box for each detected object, defining the area that must be shielded from direct lighting to prevent glare. To account for potential calculation or sensor measurement errors, Hummel [Hum10] proposes the implementation of a dynamic safety area, which enlarges the bounding box. This safety margin can be adaptively adjusted based on factors such as vehicle speed, road conditions and the predicted trajectory of the detected object [Mim19].
Based on the bounding box, the system determines the minimal and maximal beam angles encompassing the protected area for each headlamp, establishing the critical angle range that requires pixel deactivation. Based on the calculated angles, the system selectively dims or deactivates individual pixels within the matrix headlamp that would otherwise illuminate the protected area, creating a "dark tunnel" around the detected object [Hum10; SB16]. Fig. 3.5 provides visual representations of the GFHB concept, illustrating both schematic and real-world implementations of the technology.
Walter and Hummel [WH20] proposes a GFHB control approach where a pixel control image is generated, marking objects to be masked in the vehicle's forward field of view. Pixel control is then based on this image, with additional deactivation of neighboring pixels governed by predefined control rules. This method allows for more nuanced and context-aware LID.
To mitigate potential visual discomfort caused by abrupt changes in LID, smooth illumination morphing functions like a sigmoid of pixel brightnesses can be used [Hüs+13; Lef+16]. These functions smooth the transition process between different lighting states, potentially incorporating factors such as time elapsed or the position of the traffic object.


(a) Schematic illustration [Mer16]. (b) Reality [Por22]. (c) Reality [Val21].

Figure 3.5.: GFHB creates dark tunnels around detected vehicles by selective pixel deactivation.

Real-world pixels in low-resolution matrix headlights often deviate from ideal rectangular shapes [Roe18; Ler+24; KR23; Per+19], which makes intersection tests computational complex [SB16; Str22]. To address the challenges posed by non-ideal pixel shapes, Saralajew

and Benderman [SB16] propose a simplification method for pixel shape approximation and intersection testing.
First, the LID of each pixel is approximated by a closed isoline and a minimal enclosing bounding rectangular box is calculated around this isoline, along with its centroid. The critical area, defined by four angle values, is represented as a quadrilateral in spherical coordinates. Intersection testing is then performed using either polygon intersection algorithms [BL14] for the rectangular pixel representation or point-in-polygon algorithms [HA01] for centroid-based representations. Fig. 3.6 illustrates this pixel simplification process, demonstrating the transformation from the actual pixellight illumination to its bounding rectangle and centroid.
Rüddenklau [Rüd22] extends the bounding rectangle pixel simplification to arbitrary illuminance control. In this approach, each pixel is treated as a homogeneous light source. The resulting illumination is calculated by dividing the target illuminance by the mean illuminance of the pixel within its bounding rectangle.




(a) True pixel shape. (b) Isoline. (c) Bounding rectangle. (d) Centroid.

Figure 3.6.: Pixel simplification by bounding rectangular box or centroid [SB16].

Marker light can be considered an inversion of GFHB, selectively increasing illumination on potential hazards or points of interest [Wör+07; Win+15; Sha+23; Kro+23]. By highlighting specific areas or objects, marker light guides the driver's attention to critical elements in the environment. Advanced implementations may employ dynamic blinking animations to accentuate the highlighted area further [Por22] or track the driver's eyes to dynamically increase the illumination of objects when the driver looks at them [Fun+16]. Fig. 3.7 presents visual examples of marker lighting functionality, demonstrating its application in real-world scenarios.

(a) Schematic illustration [Vol18] (b) Reality [Por22] (c) Reality [Mer16].

Figure 3.7.: Marker light highlights potential hazards or points of interest to guide driver attention.

Dynamic traffic sign dimming of ADB systems aims to improve the readability of traffic signs by selectively reducing their illumination, thereby mitigating glare caused by sign reflections [Hof09]. By optimizing the contrast and brightness of traffic signs, this feature enhances the driver's ability to perceive and interpret crucial road information.

### 3.5.2. Symbol Projection

Symbol projection by HD matrix headlights enables the projection of arbitrary non-uniform patterns, images, or symbols onto the road surface or other surfaces in front of the vehicle [Gro+18; Sha+23; Ham+22; Fun23; Aus+23]. The primary objective of this feature is to enhance driver assistance by guiding his attention, similar to a head-up display and improve communication with other road users in various traffic scenarios [UNE23d]. The selection and implementation of appropriate symbol projections encompass environmental and vehicle factors [Riv+15; Pot+18; Dan+21], like surrounding topography and detected traffic objects.

The applications of symbol projection are diverse and tailored to specific driving conditions [UNE23d; Mer18; For23a; Roe22; Kro+23; Mai+23; Bau+22; Aud22b]. For instance, the system can project a snowflake symbol on the road to alert the driver of potentially slippery conditions or display a vehicle icon with an exclamation mark to warn of an impending collision. Other use cases include projecting reverse arrows to indicate excessive speed, horizontal arrows to warn of vehicles in blind spots and vertical lines to delineate lane boundaries or vehicle width for improved spatial awareness as an optical lane assistant. Additionally, the projection of horizontal lines can indicate safe following distances from preceding vehicles. Fig. 3.8 illustrates real-world examples of symbol projection and a collection of potential symbols that can be used in various driving scenarios.
Symbol projection technology also finds application in non-safety-critical scenarios, such as welcome or farewell animations projected on the ground when the driver approaches or leaves the vehicle. While not directly related to driving safety, these features enhance the overall user experience and brand identity [Por22; Mai+23].

The technology's potential extends beyond human driver assistance [Böh19]. For instance, automated vehicles could project their intended path to communicate their planned trajectory to other road users [Ber+14]. One further potential of symbol projection is the improvement of the environment-sensing capabilities of automated vehicles and their sensor systems [Cla+21; Kau+22]. By projecting structured light patterns [Kub+14; NA16; Neu16; Sei20; Yar23b; Kri+23; Kau+22] or seemingly random patterns [Wol+23], the system can enhance object detection and road condition assessment. This approach uses computer vision algorithms to compare the projected ideal symbol (e.g., a straight line) with the captured image of the actual projection (e.g., a kinked line). The discrepancies between these images are used to reconstruct the shape of the road and objects in the vehicle's path [TM83; Yar23a].

(a) Various symbols [Mer18]. (b) Warning and guiding [Aud22a]. (c) Lane orientation [Por22].

Figure 3.8.: Symbol projection examples of real-world warning and guiding scenarios.

The implementation of symbol projection in automotive lighting systems presents several technical challenges. One key consideration is that headlight pixels often deviate from ideal rectangular shapes and may not exhibit homogeneous LID. Consequently, a direct transformation of a symbol image into a grayscale image as a utilization matrix, like for a video projector, may yield suboptimal results, potentially leading to unappealing or ineffective illuminations. To address this issue, an error metric between the desired optimal illumination from the symbol image and the best achievable illumination can be defined, which then minimizes this error by adjusting the pixel utilization [Hum+18; HS18]. Fig. 3.9 shows the differences between the desired and the best possible illumination with a matrix headlamp.


(a) Desired LID. (b) Best possible LID.

Figure 3.9.: Resulting possible LID from desired illumination [Hum+18]. As this Porsche patent [Hum+18] is from 2018, Fig. 3.9b seems to show the $3 \times 28 = 84$ pixel HD84 headlight.

To optimize the illumination by minimizing the error, the LID of a matrix headlamp can be represented as a 2D $I_\mathrm{v}$ image or light texture $\boldsymbol{I}_\mathrm{v,h} \in \mathbb{R}_{\geq 0}^{n_\mathrm{row} \times n_\mathrm{col}}$ with $n_\mathrm{row} \in \mathbb{N}$ rows and $n_\mathrm{col} \in \mathbb{N}$ columns as a discrete grid of $I_\mathrm{v}$ values. This image comprises $n_\mathrm{t} = n_\mathrm{row}\, n_\mathrm{col}$ discrete elements called texels, each storing $I_\mathrm{v}$ values analogous to an IES-file format. The texels of $\boldsymbol{I}_\mathrm{v,h}$ can be reorganized or reshaped into a one-dimensional luminous intensity vector $\boldsymbol{i}_\mathrm{v,h} \in \mathbb{R}_{\geq 0}^{n_\mathrm{t}}$ because the computation hardware stores the data of the 2D matrix $\boldsymbol{I}_\mathrm{v,h}$ with a single-dimension memory address, so the matrix $\boldsymbol{I}_\mathrm{v,h}$ is also a vector $\boldsymbol{i}_\mathrm{v,h}$. The utilization of the $n_\mathrm{p}$ pixels of the headlight creates a utilization vector $\boldsymbol{u}_\mathrm{p} \in \mathbb{R}_{\geq 0 \wedge \leq 1}^{n_\mathrm{p}}$.
As the LID of the headlamp results from the additive superposition of all pixel $I_\mathrm{v}$, the LID calculation is a matrix-vector multiplication as

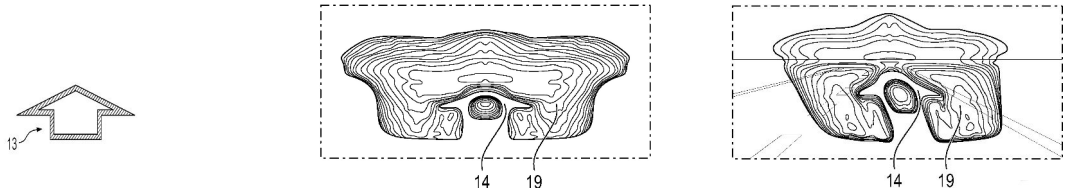
$$\boldsymbol{i}_\mathrm{v,h} = \boldsymbol{A}_\mathrm{h}\, \boldsymbol{u}_\mathrm{p}, \tag{3.5.1}$$

where $\boldsymbol{A}_\mathrm{h} \in \mathbb{R}_{\geq 0}^{n_\mathrm{t} \times n_\mathrm{p}}$ is the headlight matrix. Each column of $\boldsymbol{A}_\mathrm{h}$ is the LID of a pixel and each row is the emitted $I_\mathrm{v}$ onto a texel in a specific direction. In principle, the columns of $\boldsymbol{A}_\mathrm{h}$ are one-dimensional IES-files. The elements of $\boldsymbol{A}_\mathrm{h}$ are the maximum luminous intensity $I_{\mathrm{v,m},i,j} \in \mathbb{R}_{\geq 0}$ of the $i$-th texel by the $j$-th pixel at 100 % utilization, which weighted contribute to the overall brightness of the corresponding texel. As a texel represents a beam direction, then $I_{\mathrm{v,m},i,j}$ is the influence of a pixel on the lighting in this direction.

To determine the optimal pixel utilizations, Hummel et al. [Hum+18] propose using the least squares method, minimizing the MSE between the desired and achievable illumination for all texels as a JPO. The JPO accounts for individual pixels' overlapping, heterogeneous or irregular influence. Illumination optimization allows for adapting LIDs to different headlamps and handling different pixellights. The optimization problem is a linear equation system with $n_\mathrm{t}$ equations and $n_\mathrm{p}$ unknowns and can be formulated as

$$\boldsymbol{u}_\mathrm{p} = \left(\boldsymbol{A}_\mathrm{h}^\top \boldsymbol{A}_\mathrm{h}\right)^{-1} \boldsymbol{A}_\mathrm{h}^\top \boldsymbol{i}_\mathrm{v,h} = \boldsymbol{A}_\mathrm{h}^+ \boldsymbol{i}_\mathrm{v,h}, \tag{3.5.2}$$

where $\boldsymbol{A}_{\mathrm{h}}^{+}$ denotes the Moore-Penrose inverse [Pen55]. The Moore-Penrose inverse minimizes the MSE. It provides the optimal pixel utilizations for controlling the matrix headlight, described in Sec. B.1. The JPO can be done for other cost functions than the MSE and constraints [Dan+21; Rüd22], where the actual illumination is generated by adjusting the available pixels to minimize the deviation from the desired illumination on a projection surface. Fig. 3.10 compares a target symbol and its best possible projection on a vertical wall and a roadway.




(a) Target Symbol(13) with switched off lighting.

(b) Beam pattern(19) with symbol(14) on wall.

(c) Beam pattern(19) with symbol(14) on roadway.

Figure 3.10.: Target symbol and optimal beam pattern on a vertical wall and roadway [Dan+21].

To formulate the optimization process, Baumann et al. [Bau+21] converts the pixels into a mask dataset. Each mask describes the position, shape and characteristics of a pixel. The algorithm compares the desired image and the existing mask set and decides whether a pixel of the headlight is to be used for generation for each area of the setpoint LID.
The LID can be calculated for different threshold values of a pixel, specifying the output level from which a pixel generates the target LID. Choosing the threshold value represents a compromise between resolution and brightness. Higher threshold values result in sharper LIDs but with reduced overall illumination. In comparison, lower threshold values provide more light at the expense of increased blurring.

Rüddenklau [Rüd20] presents an iterative illumination optimization algorithm for calculating pixel utilizations. This algorithm first identifies a spatial selection area where the illuminance needs to be adjusted. It then determines a group of headlight pixels whose brightness should be modified based on the identified area. The algorithm iteratively changes the individual illuminance of the pixels in the group, either increasing or decreasing, based on the prescribed illuminance in the spatial selection area [Rüd+19b]. If the illumination requirements are unmet, the adjustment process is repeated, or the pixel group selection is adjusted.

To achieve the desired distortion-free appearance of the target symbol on the projection surface seen from the observer's perspective, the desired image must be pre-distorted in the headlamp beam pattern [Kub+17; RL18; Bau+21]. Rivero et al. [Riv+15] propose a multi-step image transformation method to calculate the setpoint LID from a desired image into a beam pattern. This approach considers the observer's position and the specific projection surface in front of the vehicle [KW18; Dan+21]. Fig. 3.11 shows an example of an observer, the desired image, the projection surface and the headlamp.
Based on the pinhole camera model with homogeneous coordinates, the symbol projection algorithm for calculating the desired LID traces the path light back from the desired image to what the given observer should perceive to what the headlight should project [Riv+15]. First, the desired image is binarized and relevant points to be illuminated are extracted. The relevant points are then scaled and projected from the desired image onto the projection

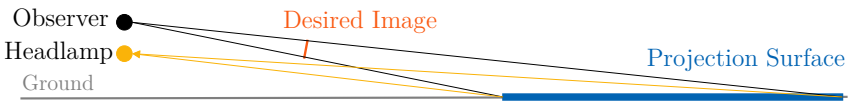
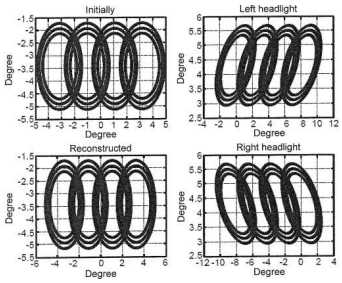
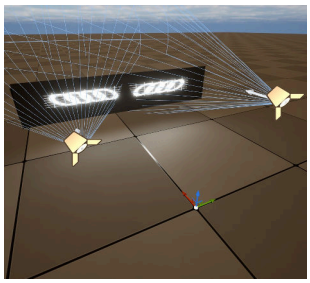



Figure 3.11.: Symbol observer, desired image, projection surface and headlamp [Riv+15].

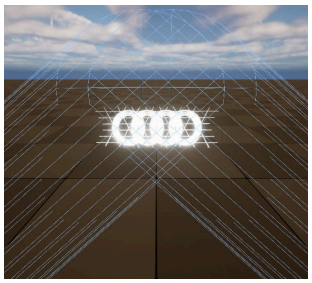
surface defined by the topography, using the observer's position as the origin of this transformation. The projection surface can be separated into subsurfaces, each illuminated exclusively by a specific headlight.
The algorithm then calculates a 2D projection of relevant points on the target projection surface for each headlight based on the headlamps' position and their line of vision. Finally, the two-times projected relevant point distribution is converted into the headlight beam angular values and luminous intensities.
Fig. 3.12 shows the results of this algorithm for both left and right headlamps, projecting the desired image in front of the ego-vehicle. The desired image appears distorted in the headlight pattern. However, it is perceived without distortion from the observer's perspective on the roadway.




(a) Desired and received observer image (left) and headlamp illuminations (right) [Riv+15].



(b) Unreal 5 simulation with a vertical wall.



(c) Unreal 5 simulation of observer view.

Figure 3.12.: The initial symbol at the top left of Fig. 3.12a is projected for the observer located between the two headlamps [Riv+15]. The symbol is individually distorted since the headlamps are at different positions. The beam patterns with the grid of Fig. 3.12a are loaded into Unreal 5 spotlights with inverted vertical angles and projected on a vertical wall and the road.

### 3.5.3. Rasterization Rendering Control

Rasterization rendering from computer graphics [Ake+18] can be used to control HD matrix headlights because of their similarity to video projectors [Win20; WH20; Sei+21; RT21; Rüd+22]. The principle of rasterization rendering control is generating a complex LID analogous to creating an image on a virtual screen by 3D modeling of the relevant road environment and using matrix headlights as virtual cameras. This method assumes that the virtual cameras observe the surroundings from the perspective corresponding to their actual mounting position on the vehicle. The resulting grayscale images by these virtual cameras serve as the headlight's setpoint LID or utilization matrix. For HD matrix headlights, these camera images can be directly interpreted as utilization matrices for individual headlight pixels.

As described by Michenfelder [Mic14], the rendering control process begins with a desired primary LID serving as the scene background. The rendering algorithm then calculates

camera images for each matrix headlight based on this background and any additional light objects in the scene. To accurately reproduce a desired LID using matrix headlights, the control software projects the setpoint LID onto a plane at a considerable distance from the light source. This projected LID is an $I_v$ texture on a virtual plane of appropriate dimensions within the 3D environment. This plane acts as the scene background, which is observed and captured by the virtual camera representing the matrix headlight.
Target light objects are placed in front of this background plane to implement lighting functions such as GFHB and marker light [Bar+15]. The rasterization rendering process incorporates geometric transformations, including rotation, scaling and translation of these target light objects. For instance, a conventional low beam LID can serve as a background plane. At the same time, additional objects representing the blanking or marking of road users can be modeled as cuboids. Additionally, the rendering allows for the modulation of the basic LID by transforming the background texture to realize dynamic lighting functions like adaptive bend lighting [MN14]. Fig. 3.13 shows a schematic example of the rendering control algorithm for two overlapping headlights.
Rüddenklau and Trächtler [RT21] further extended the rendering control algorithm to incorporate environmental objects such as buildings and road surfaces as part of a OBL function. In this enhanced model, each object class of the mesh is assigned a target illuminance value, which is then rendered into the virtual image plane of the headlight to generate a setpoint LID.

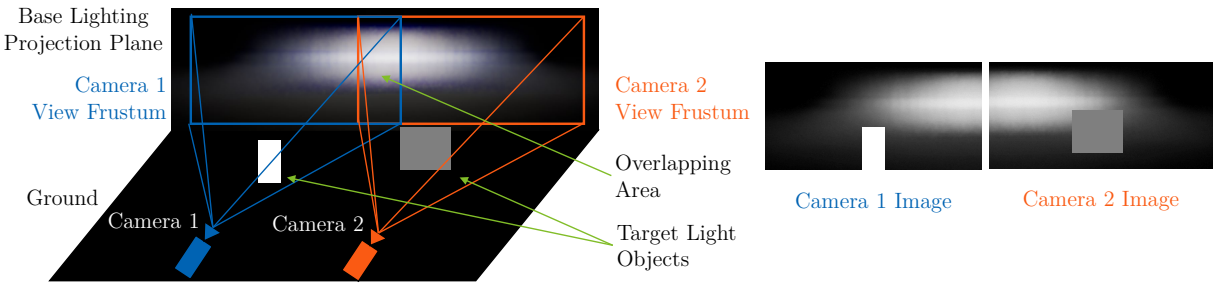



Figure 3.13.: Rendering control algorithm for two headlights. The virtual cameras capture the basic LID projected on the background plane and the target light objects in the foreground.

The rendering control algorithm can distribute a single setpoint LID across multiple headlights by observing the virtual plane by multiple virtual cameras [Mic+13; Mic14]. Given that each headlight and its corresponding virtual camera may have a distinct angular range of LID, there is often an overlap in the angular ranges covered by at least two virtual cameras. This overlap can potentially lead to the formation of visible edges in the resulting LID where the contributions from different headlights meet.
To ensure a homogeneous LID across the desired angular areas, a smoothing filter can blend the transitions between different headlight contributions [Mic14]. Another method uses a scaling factor to reduce the $I_v$ by an offline calculated percentage in the overlap regions [Mic+13]. This scaling factor is determined by comparing the $I_v$ values in the hotspot of the created LID with those of the desired LID.

# 4

# Real-Time Matrix Headlight Simulator

Before novel headlight simulation and control methods are explained and evaluated in the following chapters, the novel real-time matrix headlight simulator SOL is presented in this chapter. SOL integrates headlight simulation and control methods into a real-time rapid prototyping and development tool for dynamic HD lighting functions in both virtual and real-world scenarios. SOL generates inputs and processes outputs from the simulation and control algorithms. This chapter is based primarily on the own publications [Wal+20; WB21a; WB22; Wal+23b; Wal+25a] and provides an overview of SOL's software architecture.

## 4.1. Software Architecture and Modules[‡†]

The emergence of dynamic HD lighting functions for automated driving (see Sec. 3.5) has brought complex new headlight development challenges. These challenges include defining dynamic, sustainable beam patterns for human and computer vision that maximize safety, minimize energy consumption and increase consumer appeal. The matrix headlight simulator SOL is designed to overcome these multi-layered design challenges with its modular approach [Wal+23b].

SOL is implemented using the rendering engine Unreal 5 [Epi23b] and CUDA [Nic+08; Nvi23c] and is characterized by a modular, object-oriented software architecture that provides flexibility and scalability to different matrix headlight types. The architectural design of SOL is built upon three primary software modules: the real-time 3D rendering Unreal module, the matrix headlight simulation and control module written in CUDA and the real hardware communication module.

1. The real-time 3D rendering engine software module creates the high-fidelity visualization of headlamp LIDs in virtual environments. SOL uses for rendering the commercial engine Unreal 5 [Epi23b] by Epic Games. Unreal was chosen over Unity [Uni23] because of the native support for C++, facilitating seamless integration with existing scientific codebases and libraries that use C and C++ interfaces, which enhances the system's extensibility and interoperability. Additionally, SOL gets compatible with other automotive simulations, such as Carla [Dos+17], which also uses Unreal. The standard capabilities of commercial game engines like Unity [Uni23] and Unreal [Epi23b] do not allow the dynamic simulation of matrix headlights.

2. The matrix headlights software module is responsible for the real-time headlight simulation and control to extend the rendering engine. It uses Nvidia's CUDA [Nvi23c] to execute computations in parallel on the GPU, enhancing computational efficiency. Using a GPU confers advantages over a Central Processing Unit (CPU) because headlight simulation and control algorithms benefit from the highly parallelized processing. The CUDA software choice imposes a hardware constraint, limiting the compatibility to Nvidia GPUs.

   The matrix headlight simulation uses parallelized SpMV (see Sec. 2.5) to generate beam patterns in real-time [WB20] (is explained in Sec. 5). The resultant LID is then loaded into the engine's standard spotlight as a light texture [Uni24; Epi23a], an approach that is compatible with both Unity [Uni23] and Unreal [Epi23b].

   The headlight controller consists of a lighting planner and the underlying pixel controller with the SSC algorithm. The planner defines the optimal lighting for the current environment similar to known criteria and lighting functions (see Sec. 3.5). The SSC pixel controller adjusts the pixel utilization to match the plan in real-time. SSC uses abstract 3D objects called primitives for high-level lighting definition (is explained in Sec. 7). For example, for GFHB, the planner defines the masked bounding box and the illumination smoothing area around it and SSC transfers this headlight independent target description into the pixel utilization for the current headlight.

3. The real hardware communication module facilitates data transfer and synchronization between the simulation and physical hardware via network protocols. This module's primary function is to transmit headlight control data like pixel utilization and module orientation from the simulation to the real hardware.

   This bridging module allows for replicating virtual system tests in reality with minimal additional effort for efficient rapid prototyping, verification and validation of new lighting functions. By facilitating simultaneous visualization of both real hardware output and simulated results using identical input parameters, it also enables the calibration, verification and validation of the simulation against reality through direct comparison of visual performance. The comparison enables the investigation of perceptual disparities between simulated and real-world lighting conditions, as shown in Ch. 9.

Fig. 4.1 provides an overview of these three primary modules, illustrating their most significant connections and data flows. Sec. A.2 and Fig. A.2 provide an overview of the complete system architecture and all core components.

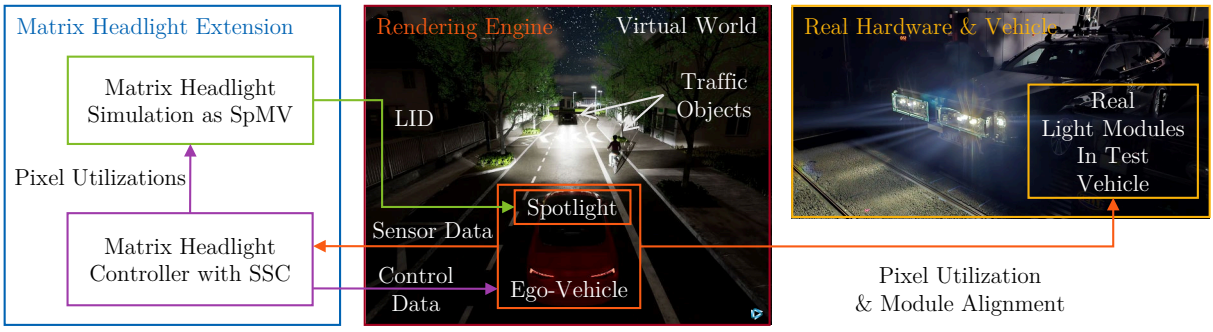



Figure 4.1.: Three primary software modules constituting SOL [Wal+25b; Wal+25a].

The SOL architecture has a data and command exchange function designed to simplify configuring test cases and evaluating results independently of the rendering engine and C++ programming environment. Using a shared-memory and mapped-memory approach on a RamDisk [FM99] for inter-process communication, high-level scripting languages such as MATLAB and Python can be used for simplified result analysis and experiment configuration. Additionally, the data exchange allows the usage of SOL by human operators and automated algorithms like machine learning or optimizers.
SOL allows dual-mode headlight optimization for different levels of beam pattern design details, providing a flexible and intuitive interface for creating and refining lighting functions. The lighting can be controlled using abstract 3D objects and SSC for high-level beam pattern definition and low-level pixel utilizations for fine-grained control. Fig. 4.2 illustrates this dual-mode optimization capability, featuring a headlight simulation input switch that allows the user (human or machine algorithm) to either employ the integrated control method or directly configure pixel utilizations.

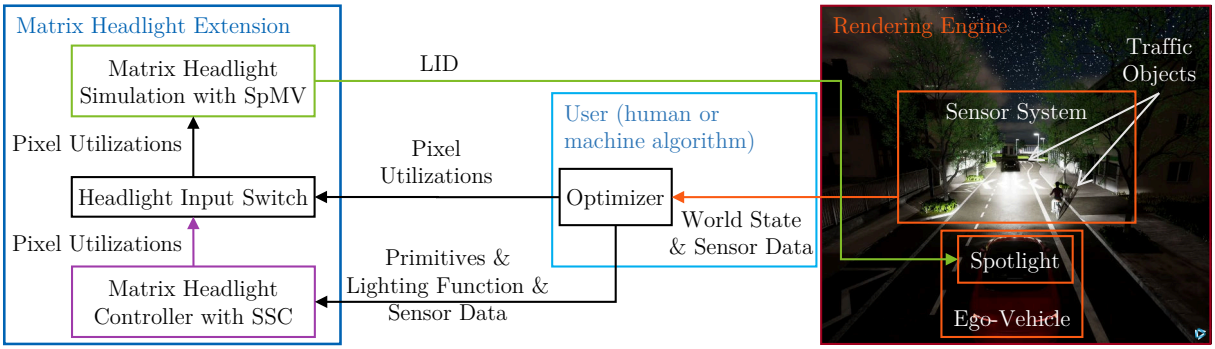



Figure 4.2.: Dual-mode headlight optimization in SOL. The optimizer can be a human user or an automated algorithm.

## 4.2. Simulator Usability with Increasing Number of Pixels[‡t]

SOL is designed as a user-friendly and scalable simulation to cope with the rapidly increasing resolutions of modern headlamps, real-world test vehicle deployment, progress in computer graphics and associated computational requirements [Wal+23b]. The resulting modular architecture of SOL ensures that the system can be adapted to different matrix headlight types and resolutions, providing a flexible, valuable and scalable platform for developing and validating advanced lighting functions.

### 4.2.1. Decoupling and Scalability

SOL uses a modular, object-oriented software architecture that decouples the headlight simulation, the control logic and the rendering pipeline from each other [Wal+23b]. This separation allows rapid integration of new software components and updating or replacing existing ones, including the rendering engine. By decoupling the headlight calculation module from the rendering engine, SOL can optimize the update frequencies for each component to ensure an optimal user experience. For example, updating the headlamp

beam patterns could be calculated at a lower frequency than the rendered image to ensure a smooth driving experience.

Technological progress requires the scalability of SOL of calculating headlights with more pixels and modules for real-time simulation. Therefore, SOL is designed to handle a theoretically infinite number of pixels and modules if the hardware has enough CPU and GPU threads. The architecture of SOL is designed to theoretically handle an arbitrarily large number of pixels and modules in parallel on suitable hardware. SOL uses a dual-layer parallelization strategy for real-time simulation and control that decomposes the lighting computation into two distinct module and ray levels, illustrated in Fig. 4.3 [Wal+23b].
The first parallelization layer occurs at the headlight module level on the CPU. Each module, representing an individual light device within the headlamp (see Sec. 2.2 and Fig. 2.2), is processed independently in parallel on different CPU threads.
The second parallelization layer is implemented within each module at the ray level on the GPU. Each ray, representing a discrete light path emitting from the module (see Sec. 3.1), is computed in parallel by a different GPU thread.
This dual-layer parallelization strategy can efficiently handle varying numbers of heterogeneous modules with varying pixel counts and allows for optimal utilization of available computational resources by allowing for processing as many modules and rays in parallel as the hardware is capable of, e.g., modules with a low pixel number are processed entirely in parallel.

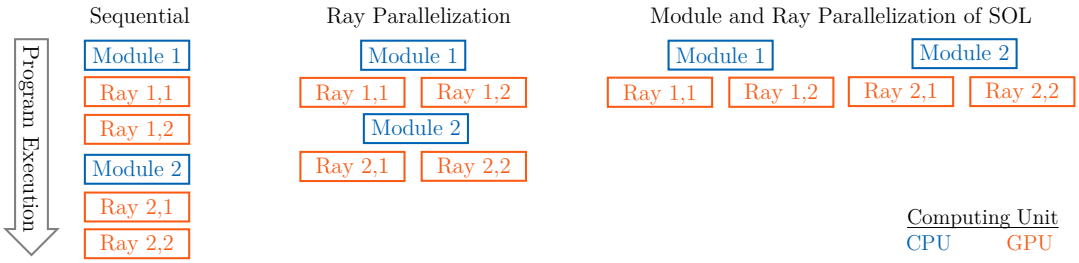



Figure 4.3.: Dual-layer parallelization strategy of two models (blue) with each two rays (red).

### 4.2.2. Intuitive Graphical User Interface

Developing an intuitive and efficient Graphical User Interface (GUI) is crucial for the practical application and user acceptance of SOL. The GUI underwent several iterations, each addressing challenges associated with the increasing design complexity of matrix headlights and the expanding capabilities of lighting functions [Wal+23b].

The initial GUI design adopted a button- and textbox-based approach, where discrete buttons and text fields represented individual pixels of the matrix [Wal+23b]. This interface allowed users to directly input and read pixel utilization values, with dynamic color updates of the field reflecting the brightness of each pixel. The spatial arrangement of these text fields mirrored the physical layout of the headlamp matrix, providing an intuitive visual representation of the LID. Fig. 4.4 illustrates this textbox-based GUI design, showcasing the text boxes for pixel utilization and sliders for module movement.
While this text-based interface offered a straightforward and easily comprehensible interaction model for systems with low pixel counts, it quickly became unusable as the resolution

of matrix headlights increased. The primary limitations of this approach are reduced readability and limited manipulation without zooming function. As pixel counts increased, the size of individual buttons' text fields had to be reduced, or some zooming had to be used. Additionally, using textboxes allows the modification of only one pixel at a time, making creating complex beam patterns time-consuming and cumbersome.

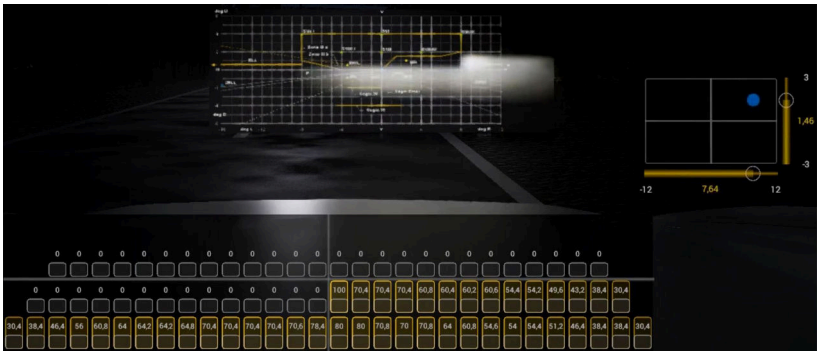


Figure 4.4.: Textbox GUI for pixel utilization and sliders for module orientation control [Wal+23b].

The second iteration of the GUI conceptualized the utilization matrix of matrix headlight pixels as grayscale images to address the limitations of the button- and textbox-based approach [Wal+23b]. This paradigm shift allowed users to manipulate multiple pixel utilization simultaneously using various brush tools with adjustable thicknesses, akin to digital image editing software. Hummel [Hum18] presents a similar approach, where the user does not paint the utilization matrix but the desired ideal illumination. The system optimizes the current LID to match the painted setpoints best [Hum+18].
Two significant limitations of the image-based GUI approach are that once beam patterns are created as cohesive units, they cannot be easily moved or modified and that the image-based approach makes it difficult to programmatically create or modify beam patterns based on algorithmic inputs or environmental conditions.
Rüddenklau et al. [Rüd+19b] present a light design method for GFHB, where the user defines a selection area on the screen and the edge points of the screen selection area are first projected into the scene and then transformed into the headlight similar to [Riv+15]. This approach is intuitive, but it is challenging to handle flying target objects above the ground because the selection area points are projected onto the ground. The screen space area method can be extended to a world painting approach, where the human user controls the headlight by clicking or selecting a world surface area with the mouse and the position of the click in the world coordinate space is transformed into the headlight image coordinate space. A lookup database is used to find the corresponding pixels for the image coordinates so the pixels can be identified and controlled, illuminating the area the human user clicked on. Fig. 4.5 shows the image-based GUI design, showcasing the image painting representation with sample letters drawn using the world painting function.

To overcome the presented limitations and provide a more flexible and powerful interface for both human designers and algorithmic control, SOL introduces with SSC a 3D primitive-based GUI based on abstract 3D objects similar to vector graphics editing software [Wal+23b]. This innovative approach uses 3D light objects, called primitives, placed within the world to define optimal beam patterns. The primitives are rectangles and act


Figure 4.5.: Image GUI for pixel utilization control [Wal+23b].

like free-floating desired images in the 3D space. Each primitive carries information about its setpoint LID as an image that defines the desired LID within its boundaries, combining the intuitiveness of image-based editing with spatial control. Multiple primitives can be combined to create complex, multi-layered lighting designs like cuboids, allowing lighting functions to adapt to complex road geometries, environments and traffic objects. Fig. 4.6 illustrates the primitive-based GUI design, showcasing the abstract 3D primitives used for pixel utilization control, along with a simulation of volumetric fog to visualize beam propagation.
Key features and advantages of the 3D primitive-based interface are spatial and image manipulation, layered design and algorithmic compatibility. The 3D primitives can be freely moved, rotated and scaled in the environment, providing intuitive control over beam patterns in 3D space. Using 3D primitives allows users to directly manipulate beam patterns in 3D space, aligning them intuitively with environmental features or traffic objects. The 3D primitive representation is compatible with algorithmic generation and modification of beam patterns based on sensor data or other inputs. Finally, the 3D primitive remains effective and intuitive regardless of the underlying pixel resolution of the headlight.

Figure 4.6.: Primitive GUI based on 3D primitives (colored boxes) for pixel utilization control, featuring volumetric fog simulation for beam propagation visualization [Wal+23b].

SOL use a two-step approach for intuitive lighting design. The hybrid approach of SSC combines the ease of use of the image-based interface for basic pattern creation with the flexibility of the 3D primitive system for advanced control and algorithmic integration.
Users first define default beam patterns and the desired projected symbols with the image-based approach, leveraging its intuitive painting metaphor for quick lighting design.

The 3D primitive-based method is then used to adjust the initial beam pattern based on specific environmental conditions. This step allows for precise control and adaptation of the beam pattern to dynamic scenarios.

## 4.3. Hardware-in-the-Loop Test of the Matrix Headlight

To be able to evaluate a complete real matrix headlamp at any time of day, a matrix headlamp Hardware-in-the-Loop (HiL) test bench digitizes the dynamic beam pattern in real-time as feedback into the simulation [Jan+19]. This section based on the own publications [Wal+19a; Wal+19b; WB19a; Wal+19a; WB19b; Wal+20; WB21a; Kau+21a; Kau+21b] and provides an overview of the novel real-time digitization and its technical challenges. The digitization was implemented in the SOL predecessor in Unity [Wal+19a; Wal+19b; WB19a]. As the digitization and the matrix headlight simulation result in the same representation of the beam pattern, it is possible to use a beam pattern switch, so a light module could either be simulated or digitized. The equal format allowed a hybrid operation, where, e.g., the left module is digitized and the right one is purely simulated.

Real-time digitization's objective is to evaluate complex lighting effects that are challenging or impractical to model or simulate accurately at any daytime [Jan+19]. The real-time digitization approach enables the assessment of the complete real matrix headlight within virtual scenarios, serving as a HiL test by placing the real hardware on a motion platform, e.g., a robotic arm within a specialized HiL test stand.
The real headlight illuminates a projection screen while the mechanical actuator replicates the dynamic positioning of the headlight as it would be moved in a real vehicle by the vehicle dynamics [Wal+20]. Concurrently, the headlamp receives input signals identical to those it would encounter in real-world conditions, ensuring its control and movement accurately mimic actual operational parameters [Wal+19a].
A camera captures the illumination on the projection screen, indirectly measuring the headlamp's LID like screen photometry techniques [Kat+22; Ins24; Tec24]. The captured color image undergoes real-time processing to generate a light texture, which is subsequently loaded into the engine's spotlight light source to recreate the headlight in the simulation [Wal+19a; WB21a]. This approach allows the virtual environment to be illuminated to reproduce the real headlight [WB19b].
Fig. 4.7 provides an overview of the HiL test stand, illustrating the integration of the matrix headlight with the mechanical actuator. Additionally, Fig. A.2 delineates the possible real-time digitization data flow as a further extension to SOL, represented by the gray pathway.

Implementing real-time digitization presents several technical challenges and financial considerations despite its potential benefits. Challenges are accurate real-time processing of camera images to LIDs and precise synchronization between the physical hardware and the virtual environment.
To achieve real-time capability, the GPU-accelerated computation of the light texture based on an adaptation of a microfacet-based Physically Based Rendering (PBR) material model [Bur12; MH16; Mar24], which serves as a simplified representation of the bidirectional reflectance distribution function of the projection screen [WB21a]. The data processing

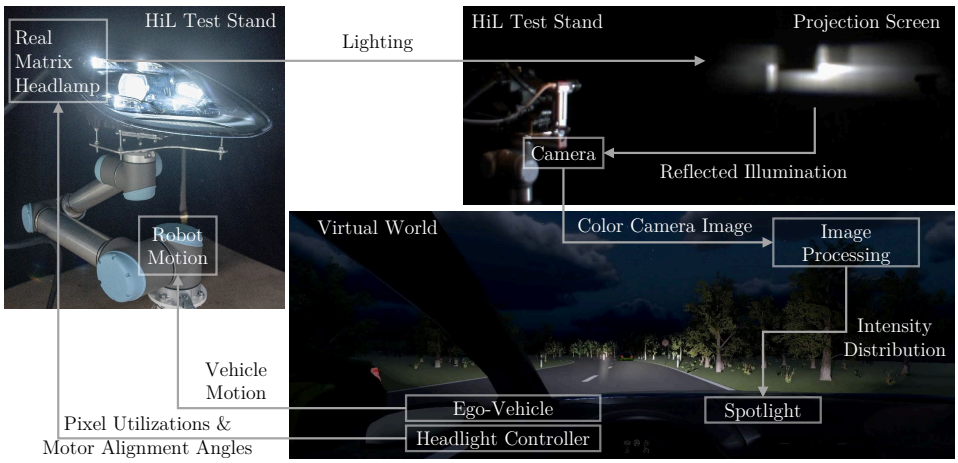



Figure 4.7.: HiL test stand configuration for a matrix headlight, featuring a mechanical actuator for dynamic positioning [Kau+21a; Kau+21b].

introduces a noticeable delay in the system response. It leads to a mismatch between headlight response and vehicle movement. A solution to reduce the effect's visibility is to delay the movement of the vehicle body in the simulation independently of the vehicle dynamics simulation. This latency poses challenges for evaluating dynamic lighting functions, such as HLS and GFHB systems, where real-time responsiveness is crucial.
The positioning of the camera within the test setup presents another technical consideration. One option is mounting the camera on the motion platform alongside the headlight or fixing it in a static position on the ceiling or structural supports of the test stand. The latter configuration offers advantages regarding camera protection and reduced exposure to dynamic mechanical stresses. However, it necessitates exact synchronization between the motion platform inputs and the headlight control signals to reproduce real-world conditions accurately [Wal+20].
Additionally, the operation of the HiL test stand imposes substantial requirements in terms of physical space, primarily due to the need for adequate measuring distances of the headlight to ensure it can be approximated as a punctiform light source. Furthermore, it demands photometric and geometric calibrations to accurately identify the reflection parameters of the PBR material model and ensure the digitization process's overall fidelity.
Moreover, real-time digitization does not offer unique advantages over alternative testing approaches. For instance, for evaluating headlight control algorithms, a conventional HiL simulation of system inputs and outputs often proves sufficient for assessing the quality of control signals. In such scenarios, a digitized LID does not necessarily provide substantial benefits over a well-calibrated simulation, provided that the simulation can accurately reproduce all relevant aspects of the lighting system's behavior.
Similarly, the real-time HiL method may be superfluous for testing the complete headlamp assembly. Compliance testing with legal standards and regulations typically does not mandate real-time capabilities. Functional evaluation can often be accomplished through direct observation of the LID on the projection screen, comparing it with the desired illumination profile geometrically transformed onto the screen surface [Ten+22].

# 5

# Real-Time Matrix Headlight Simulation

Following the overview of SOL, this chapter presents the novel methods for matrix headlight simulation. The novel approach combines a rendering engine's standard pyramid-shaped punctiform light source with a SpMV to create a dynamic virtual matrix headlight. The simulation foundations are explained in Ch. 2. This chapter is based primarily on the own publications [WB20; Wal+22c] and explains the simulation of matrix headlights.

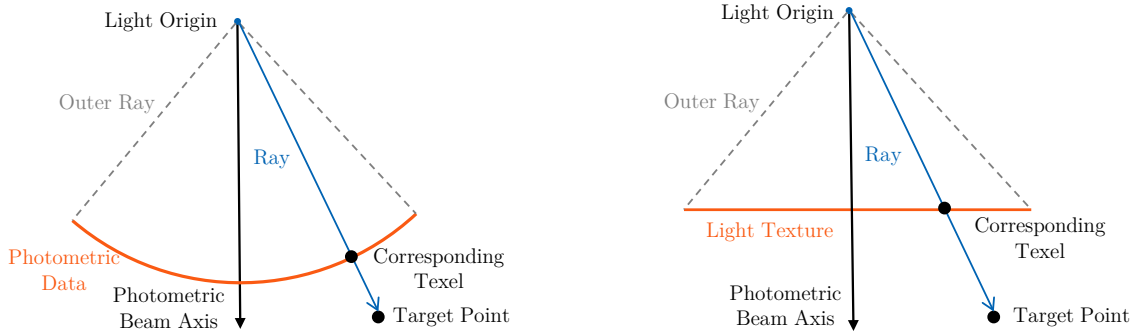
## 5.1. Selection of the Light Source Model

Established software development paradigms [McC04] recommend using existing software modules and libraries if they meet all the necessary criteria. Therefore, the default rendering engine light source is preferred as a virtual matrix headlight to minimize development effort. Given that Rüddenklau [Rüd22] developed a custom light source rather than using Unity's default spotlight [Uni24], this section will elucidate the rationale behind the decision to employ the engine's default spotlight.

Matrix LED light sources are typically configured in a rectangular grid arrangement [For24], forming an area light source. Therefore, the primary consideration is whether a virtual punctiform light source adequately represents a real matrix module and matrix headlight. When the distance to the headlamp exceeds the limiting photometric distance, it behaves in its far-field region as a punctiform light source (see Sec. 2.1). However, approximating the headlamp as a punctiform source within the near-field region introduces errors in $I_\mathrm{v}$ falloff, illumination shape and shadow formation. The limiting photometric distance and the illumination errors depend on the specific matrix headlight under consideration. The magnitude of this error increases inversely with distance [Rye97; RR14]. Vehicular lighting regulations stipulate that headlight measurements must be conducted at a far-field distance of 25 m [Yeo21; UNE23a]. Notably, measurements taken at a distance of approximately 10 m demonstrate a similarity with the 25 m far-field results [Yeo21; RR14]. Consequently, when using a punctiform light source, a small error is anticipated for distances below 25 m, with a more substantial error expected for distances under 10 m.
However, due to the vehicle's hood, the area less than 10 m in front of the vehicle is typically obscured from the driver's perspective. Furthermore, HD modules are engineered to illuminate the road from 10 m onwards [Ros+18] and symbols are projected typically $\approx 18$ m in front of the headlight onto the road [Riv+15; Ham+22]. The SSL|HD HiRes module

[For24] (see Fig. 2.2b) road illumination begins at a distance of $0.75\,\mathrm{m}\tan(90° - 3°) = 15.3\,\mathrm{m}$, assuming a mounting height of 0.75 m. Thus, the large error below 10 m is only pertinent in certain use cases. According to subjective assessment, the small error for distances greater than 10 m can be neglected in most applications.
A punctiform light source offers computational advantages over an area light model due to its singular photometric center, as it allows projective texture mapping for light simulation (see Sec. 2.3). Moreover, headlight photometric data is typically provided in a punctiform light source format, such as an IES-file (see Sec. 2.3), so area source data is typically unavailable. Punctiform light sources are also used in numerous headlight simulators [Lec+99; LS02; Sar+16; Ber05] and therefore usable in automotive simulation applications. In scenarios where a punctiform light source is deemed unsuitable, Löwenau and Strobl [LS02] propose applying ray tracing techniques through the headlight reflectors and lenses. Due to the real-time constraints and the significant model error below 10 m is often irrelevant, the punctiform light source is the preferred choice for the simulation and control of matrix headlights.

After deciding on a punctiform light source, the subsequent consideration is whether to employ a pyramid-shaped or spherical light model for beam propagation from the punctiform origin and photometric center. While the resulting illumination is identical, the data management, preprocessing and lighting calculations differ. The Unity [Uni24] and Unreal [Epi23a] engines, as well as other headlight simulators [Lec+99; Ber05], use a pyramid-shaped model. In contrast, the system proposed by Rüddenklau [Rüd22] uses a spherical model. Fig. 5.1 presents a Birdseye view of the spherical and pyramid-shaped light models.




(a) Spherical model. (b) Pyramid-shaped model.

Figure 5.1.: Birdseye view of the spherical and pyramid-shaped light models.

Rüddenklau [Rüd22] argues that conventional pyramid-shaped light source models, such as those used by Unity, are inadequate for the virtualization of vehicle headlights. The primary concern is the loss of spatial accuracy due to a congruence between the pyramidal geometry of the light model and the spherical coordinate system typically used to record automotive LID like an IES-file. In a pyramid-shaped light model, the light textures texel size and length resolution remain constant across the pyramid base, resulting in a non-uniform angular resolution from a spherical perspective. The solid angles of the texels increase with decreasing distance from the photometric beam axis and the light texture $\boldsymbol{I}_{\mathrm{v,h}}$ becomes less accurate about the solid angle [Rüd22].
Berssenbrügge [Ber05] notes a similar resolution issue and that to prevent loss of detail in $\boldsymbol{I}_{\mathrm{v,h}}$ of a pyramid-shaped model due to the distortion of the planar projected LID data

values, it is necessary to adjust the texel size. This adjustment is made such that the area of a projected discrete LID value on $\boldsymbol{I}_{\mathrm{v,h}}$ corresponds to the size of a texel in the region of the highest density of projected spherical LID values, which is typically around the photometric beam axis. The texel size adjustment solves the previous issue of the inaccuracy in the central area [Rüd22] but introduces a new memory requirement issue. As the area of a projected spherical LID value increases towards the borders of $\boldsymbol{I}_{\mathrm{v,h}}$, a projected LID value may cover several texels in these regions. This results in superfluous texels and an unnecessarily large light texture size, suboptimal for real-time rendering.

The second issue raised by Rüddenklau [Rüd22] concerns the limitations in representing wide-angle LIDs of headlamps with a horizontal angle range of $\pm 90°$, which may extend beyond the maximum reasonable angle of a pyramid-shaped light model and engine default spotlight.

To address the angle issue, Berssenbrügge [Ber05] proposes limiting the angles of the light model to the maximum beam angle as defined by legal requirements, typically 80° for horizontal angles and 15° for vertical angles. According to Berssenbrügge [Ber05], using the angle limitation and a measurement step size of 0.1°, the resulting number of texels $n_{\mathrm{t}}$ of $\boldsymbol{I}_{\mathrm{v,h}}$ is not computationally critical, as the correspondingly large textures were processed efficiently by graphics hardware even in 2005. An alternative solution for simulating wide-angle LIDs with a pyramid-shaped model is using a point light source and a light cube map to represent the LID by multiple light textures [Ake+18].

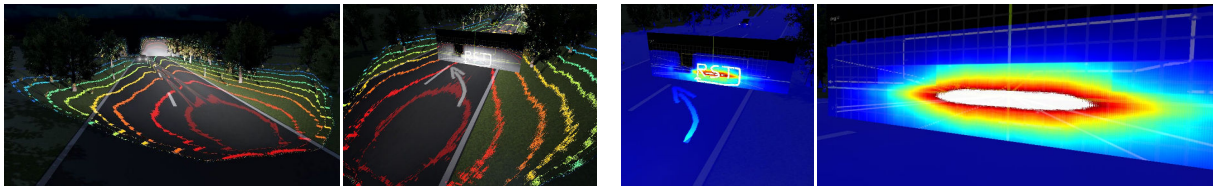
As Fig. 2.2b shows, HD matrix headlight modules typically have maximum beam angles excluding stray light much lower than 25° [Kle+19], so the previously mentioned issues are practically not that relevant for the majority of matrix headlights. The technical solution to reduce the influence of changing angular resolution on the illumination quality is implementing $I_{\mathrm{v}}$ interpolation in the light texture and during rendering.

Per default, Unity and Unreal interpolate during rendering between discrete texel values, resulting in a continuous illumination where individual discrete areas are imperceptible. Consequently, a changing or reduced angular resolution does not pose a significant problem, as it may not be discernible to the human observer. Additionally, photometric data image scaling techniques can adjust the light texture during the preprocessing. The scaling allows an adjustment of $n_{\mathrm{t}}$ to meet computational constraints without losing much visualization quality (is explained in Sec. 5.2).

Barring a potentially higher number of texels $n_{\mathrm{t}}$ of $\boldsymbol{I}_{\mathrm{v,h}}$, the pyramid-shaped light model appears to have no substantial disadvantages compared to the spherical model for the proposed use case. Moreover, it offers the advantage of faster computation during runtime, as it uses projective texture mapping [Seg+92] and the intercept theorem of (2.3.1), which results in one multiplication and division per dimension to obtain the texel coordinate with (2.3.2) (see Sec. 2.3). A spherical model requires using trigonometric functions [Rüd22] for the transformation of Cartesian coordinates to spherical coordinates in (2.3.5). Given that the pyramid-shaped light model is used for simulation and control, its computational efficiency is one primary rationale for selecting this approach.

The final decision lies between using the built-in light types of the rendering engine or developing a custom type. Rüddenklau [Rüd22] advocates for a custom light source, arguing that it allows for free manipulation of all underlying lighting functions. Rüddenklau

[Rüd22] uses retroreflection from road signs or reflector posts as an example. However, this can also be achieved with default light sources.
Other examples include analysis views for lighting engineers, such as Isolux lines and false color views. These analysis views can also be realized with built-in light sources as post-processing effects, functioning as an overlay image of the rendering engine result. The inputs to the post-processing effect include the $\boldsymbol{I}_{\mathrm{v,h}}$ of each headlight, the position and rotation of the light sources, the camera position and the screen depth, enabling the calculation of light falloff, the final illumination values and the resulting screen color following the results. Fig. 5.2 shows Isolux lines and false colors as a self-developed post-processing effect in Unity. When the rendering engine supports it, a more straightforward solution is to read the illumination data from the buffers of the rendering pipeline.
Given the absence of compelling reasons to develop a custom light source, the built-in light source of Unity and Unreal is used for matrix headlight simulation.



(a) Isolux lines. (b) False colors.

Figure 5.2.: Isolux lines and false colors in Unity [WB20].

A pyramid-shaped light model and the default engine spotlight [Uni24; Epi23a] can simulate uniform or varying color beam patterns. An example of changing inhomogeneous color is the blue edges of white LED headlights, a phenomenon caused by chromatic aberration [KR23; HG23]. To visualize changing colors of the beam pattern, the light texture incorporates not only $I_{\mathrm{v}}$ values but also RGB color information, typically storing three color values per direction instead of a single $I_{\mathrm{v}}$ value. The preprocessing, superposition and lighting calculations must be performed separately for each color channel, similar to single $I_{\mathrm{v}}$ values. However, the color values have to be adjusted to represent the $I_{\mathrm{v}}$ of the headlight by modifying their grayscale representation. When the rendering engine supports a colored RGB light texture, a single engine spotlight represents the headlight. Otherwise, three spotlights, one for each RGB color channel, with individual light textures, are required.
For the sake of simplicity, the light simulation is primarily described for the uniform color case. However, the extension to RGB colors is straightforward unless otherwise stated. Consequently, the terms "light texture" and "luminous intensity texture" are used synonymously in the following sections, with "light texture" being the more generalized term, including color data.

## 5.2. Preprocessing of Matrix Headlight Photometric Data

The preprocessing of headlight photometric data converts the measured or simulated LID from an IES-file-like format into a light texture $\boldsymbol{I}_{\mathrm{v,h}}$ for virtual pyramid-shaped punctiform light sources. This process is based on the work of Berssenbrügge [Ber05] and is modified to allow an adjustment of the light texture resolution to meet computational constraints

by using image scaling techniques like linear interpolation and multi-sampling [Fol+97; Mar+09].
To create $\boldsymbol{I}_{\mathrm{v,h}}$, it is necessary to transform and project the initial spherical photometric data onto a plane. Berssenbrügge [Ber05] proposes treating this transformation as a projection onto a planar plane positioned in front of the light source. This projection plane becomes the $\boldsymbol{I}_{\mathrm{v,h}}$ and the known process is described in Sec. 2.3.

While the known method preserves all information from the original photometric data, it may result in an unnecessarily high texel number $n_{\mathrm{t}}$, which can be computationally expensive. To reduce $n_{\mathrm{t}}$, a modified scaling approach allows the user to prioritize computational efficiency while maintaining acceptable accuracy.
In the novel scaling method, $n_{\mathrm{t}}$ is predetermined based on computational constraints and independent of the photometric data. The light texture's size, horizontal and vertical FoV, is also chosen. The preprocessing algorithm then iterates over all texels with (2.3.5) and not through the photometric data like Berssenbrügge [Ber05], using the texel centers as texture coordinates.
Two scenarios are considered when assigning values to each texel, which depends on the ratio of the texel area to the area of a projected angle step of the photometric data onto the plane. Fig. 5.3 shows a visualization of the two cases while projecting the data into the light texture.
When the texel area exceeds the area of the angle step in the photometric data, the texel value is calculated as the area-weighted average of all photometric data values within the texel area. The average is similar to image down-scaling techniques and SSAA with a regular grid (see Sec. 3.2).
When the texel area is smaller than the angle step area, the texel value is determined through distance-dependent interpolation of the surrounding nearest photometric data points around the texel position, with closer points given higher weight in the interpolation. Using nearest points is similar to image up-scaling techniques like bilinear or bicubic interpolation [Fol+97].
The scaling approach allows for adjustment of $n_{\mathrm{t}}$ to meet specific computation time requirements and hardware capabilities. Some information loss may occur depending on the chosen $n_{\mathrm{t}}$, but rendering engines typically employ interpolation techniques on the $\boldsymbol{I}_{\mathrm{v,h}}$ during rendering, which can mitigate perceptible resolution loss by the user.

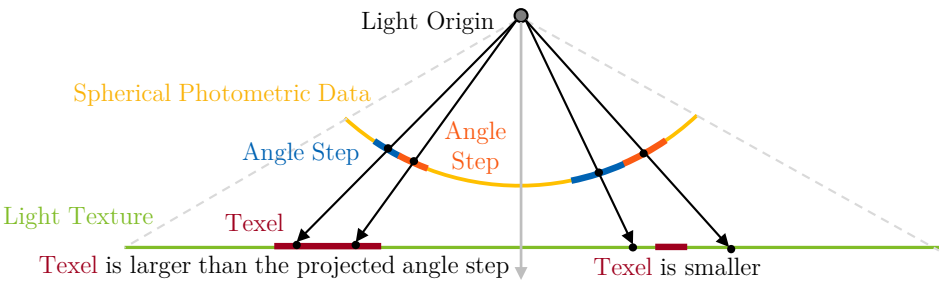



Figure 5.3.: Projection of spherical photometric data onto texels of a light texture.

Another method for optimizing computation time involves representing a matrix headlamp using multiple pyramid-shaped light sources instead of a single source. This approach uses one spotlight per module, increasing the number of light sources to be computed during

rendering. Rendering techniques such as deferred shading [Ake+18], which decouples scene geometry calculations from lighting computations, allow for rendering numerous light sources without significant performance degradation. Therefore, a multi-source approach can be beneficial when the matrix headlamp comprises multiple individual modules with varying resolutions.
A multi-source simulation allows for adaptive resolution, where $n_\mathrm{t}$ can be tailored to the photometric data of individual headlight modules. High-resolution modules require high-resolution light textures, while low-resolution modules can use lower resolutions. This division can reduce the total number of texels in multi-module headlights [Ros+18; For24], where high-resolution modules cover small central areas and low-resolution modules cover larger peripheral areas (see Fig. 2.2b) [Ros+18].
Additionally, using one light source per module simplifies the implementation of physical bend lighting and HLS through simple rotation of individual modules.

The following step is to prepare the data for the SpMV for headlight simulation. The preprocessing of the photometric data generates for the $j$-th pixel an individual light texture $\boldsymbol{I}_{\mathrm{v,p},j} \in \mathbb{R}_{\geq 0}^{n_\mathrm{row} \times n_\mathrm{col}}$. The $\boldsymbol{I}_{\mathrm{v,p},j}$ of every pixel of the headlight are combined to create the sparse headlight matrix $\boldsymbol{A}_\mathrm{h}$. Therefore, $\boldsymbol{I}_{\mathrm{v,p},j}$ of every pixel of $\boldsymbol{A}_\mathrm{h}$ have the same $n_\mathrm{row}$ and $n_\mathrm{col}$.
As the computation hardware stores the data of the 2D matrix $\boldsymbol{I}_\mathrm{v,h}$ with a single-dimension memory address, the matrix $\boldsymbol{I}_\mathrm{v,h}$ is also a vector $\boldsymbol{i}_\mathrm{v,h}$. In the same way, $\boldsymbol{I}_{\mathrm{v,p},j}$ is also a pixel light vector $\boldsymbol{i}_{\mathrm{v,p},j} \in \mathbb{R}_{\geq 0}^{n_\mathrm{row}\, n_\mathrm{col}}$ of the $j$-th pixel. This dual representation of the light texture allows the application of SpMV without a conversation on the GPU hardware. This thesis uses it for all light textures of the matrix headlight.
To create $\boldsymbol{A}_\mathrm{h}$, the pixel light vectors $\boldsymbol{i}_{\mathrm{v,p},j}$ are then assembled as columns in $\boldsymbol{A}_\mathrm{h}$, which has dimensions $n_\mathrm{t} \times n_\mathrm{p}$, where $n_\mathrm{t} = n_\mathrm{row}\, n_\mathrm{col}$ is the total number of texels in the $\boldsymbol{I}_\mathrm{v,h}$. The sparsity of $\boldsymbol{A}_\mathrm{h}$ results from the typical LID of pixelated headlights, where each pixel illuminates only a small portion of the total beam pattern. This characteristic produces $\boldsymbol{I}_{\mathrm{v,p},j}$ predominantly of zero values. $\boldsymbol{A}_\mathrm{h}$ and $\boldsymbol{u}_\mathrm{p}$ can represent lighting data from an arbitrary number of modules with the same photometric center, which means $\boldsymbol{A}_\mathrm{h}$ can be used for the complete matrix headlamp or an arbitrary number of modules. Fig. 5.4 shows an example representation of a beam pattern with three pixels as the product of $\boldsymbol{A}_\mathrm{h}$ and utilization vector $\boldsymbol{u}_\mathrm{p}$ with individual elements $u_{\mathrm{p},j}$.
As the final step, $\boldsymbol{A}_\mathrm{h}$ is converted into a suitable sparse matrix format like CSR or ELL (see Sec. 2.5) to conclude the preprocessing of the headlight photometric data [WB20].

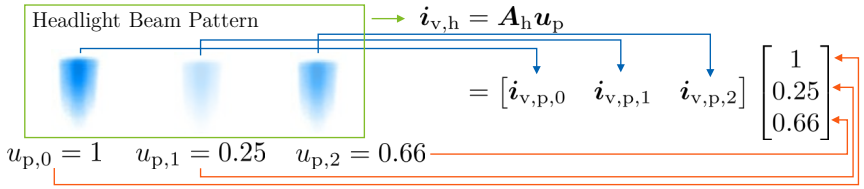



Figure 5.4.: Calculation of a beam pattern with three pixels as the product of $\boldsymbol{A}_\mathrm{h}$ and $\boldsymbol{u}_\mathrm{p}$. $\boldsymbol{u}_\mathrm{p}$ consists of the example values $[1\ 0.25\ 0.66]^\top$, which define the strongness of the illumination of the pixels.

After the creation of $\boldsymbol{A}_\mathrm{h}$, the corresponding virtual engine spotlight is parametrized according to Sec. 2.3 to represent the headlight, which means, e.g., similar FoV, position,

orientation and maximal possible brightness. To replicate a real headlight configuration in SOL, the LID and chromatic data of all pixels are loaded by the described preprocessing pipeline. As long the photometric data of all pixels can be provided in an IES-file-like format, SOL supports every lighting technology like LED or DMD (see Sec. 2.2). The virtual spotlights are positioned with spatial and orientational correspondence to the test vehicle's physical light modules. The loaded photometric data is connected to the corresponding spotlights.

In scenarios where computational resources are constrained, additional preprocessing steps may be necessary to optimize the headlight computation by reducing the number of pixels illuminating one texel. One approach involves setting $I_{\mathrm{v,m},i,j}$ values below an upper luminous intensity bound $I_{\mathrm{v,ub}} \in \mathbb{R}_{\geq 0}$ to zero, excluding them from the SpMV computation [Rüd22]. While this method reduces computational load, it can result in an overall darkening of the headlight projection, particularly noticeable in headlights with a high number of overlapping pixels, such as the SSL|HD.
A novel dimension reduction approach to reduce the number of overlapping pixels per row while preserving the overall $I_{\mathrm{v}}$ amount is presented to mitigate these issues. The user first selects a desired number of overlapping and illuminating pixels $n_{\mathrm{p},i}$ for the $i$-th row for the adjusted $\boldsymbol{A}_{\mathrm{h}}$. The choice of $n_{\mathrm{p},i}$ depends on the real-time requirements, as the SpMV can be calculated typically faster with decreasing $n_{\mathrm{p},i}$. The process determines $I_{\mathrm{v,ub}}$ such that after setting all $I_{\mathrm{v,m},i,j} < I_{\mathrm{v,ub}}$ to zero, $n_{\mathrm{p},i}$ non-zero values remain in the $i$-th row. The set of pixel and column indices of the $i$-th row with $I_{\mathrm{v,m},i,j} > 0\,\mathrm{cd}$ is $\mathcal{C}_{\mathrm{p,gz},i}$ and $|\mathcal{C}_{\mathrm{p,gz},i}| = n_{\mathrm{p},i}$. Similar to (2.4.1), $I_{\mathrm{v}}$ of the $i$-th row is

$$I_{\mathrm{v},i} = \sum_{j \in \mathcal{C}_{\mathrm{p,gz},i}} \left(I_{\mathrm{v,m},i,j} + I_{\mathrm{v,a},i,j}\right) u_{\mathrm{p},j}, \tag{5.2.1}$$

where $I_{\mathrm{v,a},i,j} \in \mathbb{R}_{\geq 0}$ is an additional $I_{\mathrm{v}}$ term for the $j$-th pixel. Using $I_{\mathrm{v,a},i,j}$ should minimize the error introduced by dimension reduction.
One approach for determining optimal values for $I_{\mathrm{v,a},i,j}$ is to minimize a quantifying error like the MSE between the original and reduced system outputs across relevant use cases, such as all AFS base lighting classes and maximum pixel utilization scenarios. An alternative method involves distributing the total lost luminous intensity $I_{\mathrm{v,sum}} \in \mathbb{R}_{\geq 0}$ resulting from thresholding across the remaining pixels in the row, weighted by their brightness as

$$I_{\mathrm{v,a},i,j} = \frac{I_{\mathrm{v,m},i,j}}{\sum_{j \in \mathcal{C}_{\mathrm{p,gz},i}} I_{\mathrm{v,m},i,j}} I_{\mathrm{v,sum}}. \tag{5.2.2}$$

This novel dimension reduction approach with a distribution of the lost $I_{\mathrm{v}}$ offers a more nuanced solution compared to simple thresholding, allowing for a reduction in SpMV computation time while preserving photometric data quality for relevant beam patterns.

## 5.3. Pixel Superposition by Sparse Matrix-Vector Multiplication

This thesis proposes a novel approach to matrix headlight simulation by interpreting the dynamic superposition of headlight pixel illuminations as a SpMV operation. This method uses the headlight matrix $\boldsymbol{A}_{\mathrm{h}}$ and a pixel utilization vector $\boldsymbol{u}_{\mathrm{p}}$, using well-established data formats like CSR (see Sec. 2.5) for efficient real-time computation of the dynamic

illumination on the GPU. The SpMV with known data formats approach stands in contrast to the work of Rüddenklau [Rüd22], who developed a proprietary data format called LGC (see Sec. 2.4) without explicitly referencing SpMV techniques.

The linear superposition of pixels in (2.4.1) can be reformulated as a SpMV operation, which eliminates the need to process elements where $I_{\mathrm{v,m},i,j} = 0\,\mathrm{cd}$ [WB20]. This optimization is achieved by storing the indices of pixels with non-zero contributions ($I_{\mathrm{v,m},i,j} > 0\,\mathrm{cd}$) for the $i$-th row in the index column set $\mathcal{C}_{\mathrm{p,gz},i}$. Consequently, (2.4.1) can be rewritten as

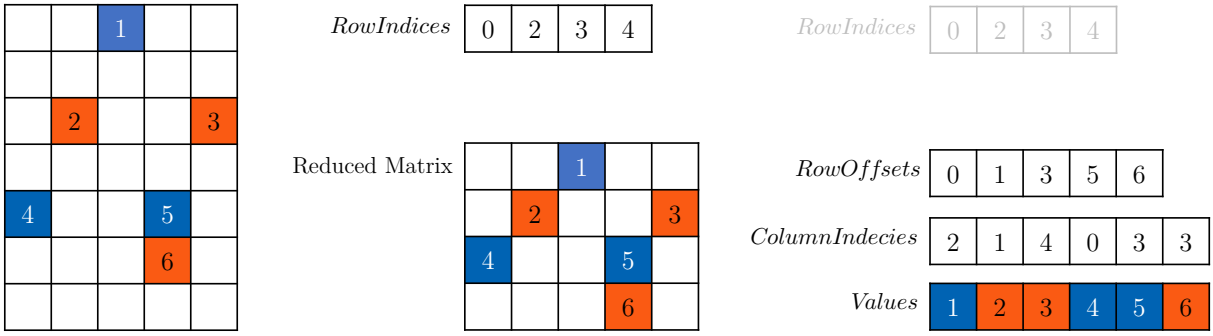
$$I_{\mathrm{v},i} = \sum_{j \in \mathcal{C}_{\mathrm{p,gz},i}} I_{\mathrm{v,m},i,j}\, u_{\mathrm{p},j}. \tag{5.3.1}$$

(2.4.1) and (5.3.1) can be parallelized similarly for all texels, allowing for the parallel computation of as many $I_{\mathrm{v},i}$ values as there are GPU threads available. After calculating $I_{\mathrm{v},i}$ for every row, the resulting $\boldsymbol{I}_{\mathrm{v,h}}$ is loaded into the $\boldsymbol{A}_{\mathrm{h}}$ corresponding engine spotlight for rendering. The virtual spotlight is moved and rotated according to the vehicle dynamics and the real headlamp's position and orientation to simulate the dynamic illumination of the matrix headlamp.
The SpMV calculation time of (5.3.1) is not determined by $n_{\mathrm{p}}$ like (2.4.1) but by the number of overlapping and illuminating pixels $n_{\mathrm{p},i} = |\mathcal{C}_{\mathrm{p,gz},i}|$ per row, which increases the computational efficiency similar to LGC of Sec. 2.4. Therefore, the complexity of a SpMV is $\mathcal{O}(n_{\mathrm{t}}\, n_{\mathrm{p,av}})$, where $n_{\mathrm{p,av}} \in \mathbb{R}_{\geq 0}$ is the average number of overlapping pixels per texture element texel. The SpMV complexity is lower than $\mathcal{O}(n_{\mathrm{t}}\, n_{\mathrm{p}})$ of (2.4.1), because for matrix headlights in general, $n_{\mathrm{p,av}} \ll n_{\mathrm{p}}$, as each pixel ideally illuminates a specific direction exclusively, analogous to a video projector.

As CSR stores the row indices of not illuminated pixels in *RowOffsets*, it may be memory-inefficient for $\boldsymbol{I}_{\mathrm{v,h}}$ with a high number of zero-only rows. For instance, when using Unreals' default spotlight [Epi23a; Wal+22c] with equal horizontal and vertical FoV, $\boldsymbol{I}_{\mathrm{v,h}}$ is squared with large dark areas at the top and bottom caused by the wide horizontal FoV of a typical matrix headlight.
To not store large dark areas at the top and bottom in CSR, a global memory offset for row indices to the first illuminated texel is introduced. This modified CSR format stores all texels from the first illuminated texel to the last illuminated continuous per rows as a wide sub-texture. The format requires a target memory offset and row indexes equal to the number of texels between the outermost illuminated ones plus three. The global memory offset is added to the CSR row index to obtain the correct row index of the complete $\boldsymbol{I}_{\mathrm{v,h}}$ for setting the sub-texture inside the original one.

An alternative modification uses a row processing control system [LS15; LS18], which assigns row indices to GPU threads for row computation. In a static, fixed assignment, the preprocessing identifies all potentially illuminated texels and stores the $n_{\mathrm{t,gz}} \in \mathbb{N}$ indices of potential illuminated texels in a row-index array *RowIndices*. Avoiding zero-only rows creates a reduced sub-matrix by storing only the illuminated texel. The sub-matrix is stored in CSR format, so *RowIndices* consists of $n_{\mathrm{t,gz}}$ target row indices and *RowOffsets* $n_{\mathrm{t,gz}} + 1$ indices of the sub-matrix.
This sub-matrix approach is a hybrid format of COO (see Sec. 2.5) and CSR and results in total memory consumption for the indices of $2n_{\mathrm{t,gz}} + 1$. The order of indices in *RowIndices* can be further optimized to process rows in a specific sequence, potentially improving

hardware utilization [Kre+14]. Fig. 5.5 provides an illustrative example of storing a sparse matrix with rows consisting entirely of zeros in the modified CSR format.




(a) Original matrix. (b) Reduced matrix with row indices array. (c) CSR of reduced matrix.

Figure 5.5.: Storing a sparse matrix with zero-only rows in a modified CSR format.

## 5.4. Incremental Computation of Beam Patterns

To enhance the computational efficiency of SpMV for matrix headlight simulation using incremental computation [RR93; Mil+94] is proposed to update $\boldsymbol{I}_{\mathrm{v,h}}$. The novel incremental beam pattern update avoids the complete recalculation of $\boldsymbol{I}_{\mathrm{v,h}}$ at each simulation time step by only updating those portions of $\boldsymbol{I}_{\mathrm{v,h}}$ corresponding to changes in $u_{\mathrm{p},j}$. Because using high-resolution light textures where a pixel covers a small space is required for simulating HD matrix headlights, an incremental update method will be required to efficiently simulate further matrix headlights like a DMD headlight [Mer18].

The incremental update requires an additional computational step prior to the SpMV to dynamically identify the changing texels within $\boldsymbol{I}_{\mathrm{v,h}}$. The incremental update's efficiency compared to a full update depends on the number of changing texels relative to the total number of texels and the computation time of the additional identification step. The threshold of the texel ratio depends on the current matrix headlight, e.g., the number of overlaps, the software implementation and the hardware capabilities. The additional identification step finds the indices of all changing texels at the next update and thus creates a texel index array *RowIndices* without duplicates, only with indices of the changed texels. The identification uses for each pixel a set of the indices of all texels illuminated by the pixel and when $u_{\mathrm{p},j}$ of the $j$-th pixel changes, all indices of the pixel set are added to *RowIndices* by avoiding duplicates. After the identification step, the SpMV operation is conducted only for the changing texels in *RowIndices*. The CUDA C++ source code of the incremental update is provided in Sec. A.4.

# 6

# Evaluation of the Headlight Simulation

This chapter evaluates the real-time matrix headlight simulation with SpMV and engine default spotlights. As the lighting simulation results in an image of the perceived illumination, objectively evaluating the simulation quality is challenging. This chapter is based primarily on the own publications [WB20; WB21a; Wal+22c; Wal+23b] and evaluates the real-time matrix headlight simulation.

## 6.1. Accuracy, Memory Usage and Computation Time

The reasons for selecting the pyramid-shaped light source model and its assessment were discussed in Sec. 5.1. The following evaluation assesses the generation of the light texture with SpMV. As SpMV avoids the computation of zero values, it does not influence the accuracy of the simulation, which depends on the quality of the photometric data and the preprocessing step.
The assessment of the accuracy of the preprocessing step compares the generated light texture and the actual photometric data of the headlight by using an error metric such as the Root Mean Squared Error (RMSE). As established in Sec. 5.2, the used light textures by the SpMV are a perspective projection of the reference photometric data onto a plane. Consequently, when the resolution of the light texture is high enough to store all photometric data points without compression and no reduction in pixel overlap is applied in the preprocessing, the error should converge to zero because all data points are kept. Conversely, a lower texture resolution or using the pixel overlap reduction will increase the error. The sensitivity of this error depends on the selected error metric, e.g., Mean Absolute Error (MAE) or RMSE, as well as the specific headlight [WB21a].
The threshold at which the error is deemed acceptable for the user varies according to the simulator's application and subjective user preferences. For instance, a functional test of light-related features might tolerate a more significant error margin than tests focused on headlight lenses' optical design, which demands higher accuracy. However, sensitivity analyses are not included due to the confidentiality surrounding the matrix headlights' optical design and photometric data.

CSR is chosen as the sparse matrix format for SpMV, as it uses memory more efficiently compared to ELL and SELL, which can introduce superfluous null elements (see Sec. 2.5). The selection of CSR is advantageous for the HD84 headlight [Kal+16; Rüd22],

which exhibited a higher concentration of overlapping pixellight illumination areas near its photometric beam axis and a sparser distribution towards the periphery. This non-uniform distribution of illumination areas aligns well with CSR's ability to store irregularly structured sparse matrices efficiently. Therefore, CSR is more memory efficient than ELL and SELL for the HD84.
However, CSR may not be the best choice for every matrix headlight. Alternative storage formats to CSR like SELL, show particular promise for headlights characterized by uniform pixel overlap distributions like the SSL|HD [Por22; Kle+22; For24], especially when combined with an overlap reduction and equalization approach over all rows (see Sec. 5.2).

An evaluation of the computation time for SpMV with CSR with varying numbers of headlight pixels and texture texels is presented in Sec. A.3. Experiments conducted on a Nvidia 2080 Ti demonstrated that a headlight with $1.51 \cdot 10^6$ pixels, 25 % overlap between adjacent pixels and $2.09 \cdot 10^6$ light texture texels could be simulated using CSR with a scalar kernel and full update in a mean time of 0.47 ms. These results indicate that the SpMV approach with scalar CSR is capable of real-time performance because there is no subjective visible lag in the dynamic lighting. This thesis uses the CSR format (see Fig. 2.11) because of its memory efficiency for headlight with changes in $n_{\mathrm{p},i}$.
As mentioned in Sec. 2.5, the computation time of SpMV is influenced by the computational capabilities of the hardware, the chosen sparse matrix format and the sparsity of specific headlights. The evaluation and optimization of sparse matrix formats have been studied in numerous works [Im00; BB09; Dal+15; Ste+16; Fil+17; Li+20; Xia+23]. As the photometric data for the matrix headlights is not freely available to the public, SpMV computation time behavior can not be evaluated. A possible approach to identify the optimal sparse format for a headlight is letting the preprocessing stage empirically determine the most efficient sparse format for the current combination of hardware and headlight. Empirical measurement is the only option for selecting optimal algorithms and data structures when not all hardware properties are publicly accessible [BG09; Nvi23a].

To quantitatively evaluate the efficacy of the incremental beam pattern update of Sec. 5.4, the computation time of both complete updates and incremental updates using CSR with a scalar kernel is compared for a random utilization change of 10 % of the headlight pixels. The used matrix headlight [Wal+22c] has an aspect ratio of 4:1 and no overlapping pixels, so $n_{\mathrm{p,av}} \approx 1$. The overall $n_{\mathrm{t}}$ is four times the horizontal count and is constant, so increasing $n_{\mathrm{p}}$ decreases the average number of texels illuminated by a pixel $n_{\mathrm{t,av}} \in \mathbb{R}_{\geq 0}$. A Nvidia 4090 GPU [Nvi24] is used and the CUDA C++ source code is provided in Sec. A.4. Fig. 6.1 presents the results of this analysis, depicting the relative median computation time of incremental updates compared to complete updates across 1,500 iterations for varying numbers of headlight pixels $n_{\mathrm{p}}$ and texture elements $n_{\mathrm{t}}$.
For headlights with small $n_{\mathrm{t}}$ and large $n_{\mathrm{t,av}}$, i.e., low $n_{\mathrm{p}}$ with fixed $n_{\mathrm{t}}$, complete updates have a higher efficiency than incremental updates. The incremental update is more efficient for large $n_{\mathrm{t}}$ and small $n_{\mathrm{t,av}}$.
One possible explanation for the dependency to $n_{\mathrm{p}}$ is that on real GPU hardware, only a limited number of threads and operations can be executed simultaneously [Nvi23b], while the rest are kept inactive. So when $n_{\mathrm{p}}$ is higher than the number of possible simultaneous

threads, the runtime of the additional identification step may not decrease, even when $n_{\mathrm{t,av}}$ gets lower. This hypothesis is supported by the fact that the Nvidia 4090 [Nvi24] has 16,384 CUDA cores (parallel processing units) and that the inflection points of the curves in Fig. 6.1 are at similar pixel numbers.
As the start of a CUDA kernel also incurs computing costs, one possible explanation for the dependency on $n_{\mathrm{t}}$ is that the additional identification step has fixed computing cost independent of $n_{\mathrm{t}}$. As the overall runtime increases, the relative influence of the additional identification step decreases and the incremental update becomes more efficient.

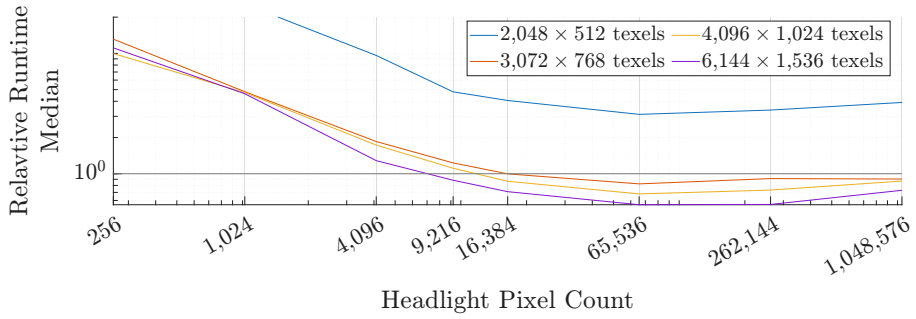



Figure 6.1.: Relative runtime median of 1,500 CSR SpMVs of the incremental update to a full update by a Nvidia 4090. Values below one mean the incremental update is faster. For every run, the utilization of randomly 10 % of the pixels are changed. The matrix headlight has no overlapping pixels. For every colored curve, the overall $n_{\mathrm{t}}$ is constant, so increasing $n_{\mathrm{p}}$ decreases $n_{\mathrm{t,av}}$.

## 6.2. Comparison of Sparse Matrix Formats

As Rüddenklau [Rüd22] developed a custom LGC data format for matrix headlights, LGC should be compared with a standard SpMV format such as the chosen CSR. This comparison is based on three primary criteria: calculation time, memory usage and accuracy. Sec. 2.4 explained the LGC format and Fig. 2.8 shows the LGC pseudocode. Conversely, Sec. 2.5 explained CSR and Fig. 2.12 shows the CSR CUDA code.

In terms of accuracy, both LGC and CSR formats provide exact calculations of $I_{\mathrm{v},i}$ without approximation, rendering them equivalent in this aspect. Regarding memory usage, both formats store column indices (*ColumnIndices* and *srcIDBuffer*) and matrix values (*Values* and *srcRGBYBuffer*) similarly. The primary difference lies in the storage of row indices, represented by *RowOffsets* of CSR and *trgBuffer* of LGC.
CSR uses *RowOffsets* to store value offsets for each texel of $\boldsymbol{I}_{\mathrm{v,h}}$, resulting in $n_{\mathrm{t}}+1$ indices. In contrast, LGC uses *trgBuffer* to store, for each texel illuminated by at least one pixel, the offset to its data, the number of illuminating pixels $n_{\mathrm{p},i}$ and horizontal and vertical indices of $\boldsymbol{I}_{\mathrm{v,h}}$. LGC results in $4n_{\mathrm{t,gz}}$ indices, where $n_{\mathrm{t,gz}} \in \mathbb{N}$ is the maximal number of possible illuminated texels by the headlight. Consequently, memory usage's relative efficiency depends on the headlight configuration and the $n_{\mathrm{t,gz}}$ ratio to $n_{\mathrm{t}}$.

To facilitate a fair comparison of computation time and memory usage, several assumptions are necessary to ensure a fair evaluation. The evaluation is only done theoretically because the shader code of LGC is not publicly accessible.

Firstly, it is assumed that both methods operate on single-dimension hardware memory like CUDA does, simplifying LGC to use a single index instead of two indices proposed by Rüddenklau [Rüd22].
Secondly, the effects of global memory coalescing access [Har13a] and shared or cached memory [Har13b] are disregarded, as their impact is hardware-dependent. With its original 2D indices, it is worth noting that LGC does not inherently follow memory order coalescing of the matrix headlight data, potentially impacting thread operations on non-neighboring data. However, the four indices could be coalescing accessed. In contrast, CSR adheres to memory order, allowing for coalescing writing and reading of column indexes and values, albeit with some computational overhead by the vector implementation. LGC initially uses four unique indices for each thread, so caching is useless, but with CSR, two threads share one-row offset so that caching could convert one slow global memory access into one fast cache read.
The third assumption posits that reading row indices has a negligible impact on computation time, as it occurs only once outside the main processing loop.

Under these assumptions, the core calculations of both methods appear to be similar, suggesting comparable computation times. However, the memory consumption calculations and comparison focus on the indices in *RowOffsets* and *trgBuffer*, as the storage of column indices and matrix values is identical in both approaches. The memory efficiency of CSR relative to LGC depends on the number of texel illuminated by at least one pixel $n_{\mathrm{t,gz}}$ about the total texel number $n_{\mathrm{t}}$. CSR is more efficient than LGC when

$$3\,n_{\mathrm{t,gz}} > n_{\mathrm{t}} + 1 \leftrightarrow n_{\mathrm{t,gz}} > \frac{n_{\mathrm{t}} + 1}{3}. \tag{6.2.1}$$

This inequality indicates that CSR is more memory-efficient than LGC when the headlight illuminates more than approximately one-third of all texels.
Analysis of real matrix headlights reveals that this condition is met in practice. The ratio of texels with possible $I_{\mathrm{v}} > 380\,\mathrm{cd}$ to all texels of the minimal enclosing rectangle is 81.73 % for the HD84 matrix module, 65.27 % for the complete HD84 headlamp including auxiliary light and 83.47 % for the LoRes SSL|HD matrix module. These ratios exceed the threshold of one-third, indicating that CSR is more memory-efficient than LGC for these real matrix headlights.

In some cases, like using Unreals' default spotlight, CSR is less memory-efficient than LGC because LGC does not store the indices of the large dark areas at the top and bottom of the squared $\boldsymbol{I}_{\mathrm{v,h}}$. Using the modified sub-matrix CSR format (see Sec. 5.3 and Fig. 5.5) can mitigate this issue. The modified CSR is more memory-efficient than LGC when $n_{\mathrm{t,gz}} > 1$, as it requires $2n_{\mathrm{t,gz}} + 1$ indices, which is less than the $3n_{\mathrm{t,gz}}$ indices of LGC.

## 6.3. Illumination Visualization Quality

The third aspect of the simulation evaluation involves the assessment of LID visualization within a virtual environment rendered by the engine. A general visualization verification process should involve multiple techniques [Sar92]. The visualization evaluation compares the simulated output to real-world observations. However, this thesis focuses on the simulation of the matrix headlight with rendering engines' default light sources and not

the real-time lighting simulation in general, so there will be no evaluation and comparison of the visual quality of the rendering engines. To ensure lighting simulation accuracy, the recommended simulation verification strategy is:

- Physical correctness verification of light propagation calculations.
- Quantitative photometric accuracy assessment using standardized error metrics.
- Subjective comparison of rendered images with reality by humans.
- Image-based similarity analysis with reality using quantitative error metrics.
- Functional verification through downstream system performance analysis like object detection neural networks and headlight control algorithms.

The first two verification steps have been discussed in Sec. 5.1 and 6.1. The remaining steps are discussed in the following paragraphs.

The procedure for subjective evaluation of the entire simulation is to replicate a real scenario within the simulation as closely as possible and compare the results from identical perspectives. Although technically demanding and time-consuming, this approach enables a complete evaluation of all simulation aspects, which are thus evaluated in combination. This final visual fidelity evaluation is a combined assessment of the visual fidelity provided by the rendering engine, including the PBR material models [Bur12; MH16; Mar24], the environment model and the underlying lighting calculations.
This aspect is challenging to substantiate in written form and with static images, as the evaluation primarily relies on subjective visual perception of dynamic scenes. Static images do not have the same visual impression as dynamic illuminations on computer monitors. Fig. 6.2 shows a comparative analysis of a real image and a simulated view of the Hellinghäuser Weg near the German city Lippstadt.

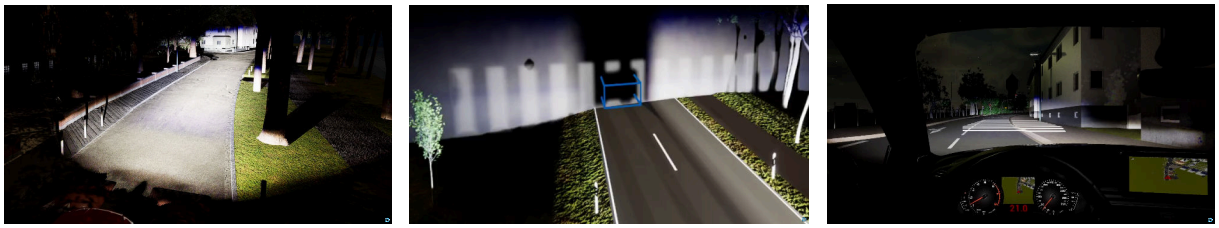
(a) Real image. (b) Simulated image by Unreal 5.2.

Figure 6.2.: Real image and simulated image of the Hellinghäuser Weg (GPS 51°40'13.6"N 8°19'29.1"E) [Wal+23b]. 3D Mapping Solutions GmbH built the environment model.

Fig. 6.3 enables a subjective comparative analysis between a simulated HD84 matrix headlamp illumination derived from photometric data in Unity and an actual image of the headlamp illumination. Another way to compare simulation and reality is to digitize the real LID using image processing (see Sec. 4.3) to be used as a virtual headlight in an identical environment and then compare the resulting virtual illuminations. Chromatic aberrations manifest as blue edges and yellow hues. They are visible in subjective equivalent regions in both simulated and digitized headlights in Fig. 6.3. In addition, key features such as base color, color gradients and the sharpness of the LID show a subjectively high degree of similarity. Fig. 6.4 shows further visual impressions from Unreal to demonstrate that the methods presented can be applied to different rendering engines and headlights.

(a) Simulation. (b) Digitized. (c) Real photo.

Figure 6.3.: Comprehension of the visual quality in Unity 2019 of the simulated headlight [WB20].



(a) City scenario [Wal+23b]. (b) Vertical wall [Wal+23b]. (c) Driver perspective [GW24].

Figure 6.4.: Simulation of matrix headlights in Unreal 5.2.

In addition to subjective human evaluation, objective quantitative methods for simulation evaluation can also be used. This thesis does not evaluate rendering engines' visual quality, so the recommended objectified methods are only presented and not carried out. The objective evaluation uses a numerical error metric for automatically comparing, adapting and optimizing different simulation configurations, such as material parameters.
One possible technique for measuring the error is to compare a simulated image with a real image, assessing their similarity and quantifying their differences as a metric like MAE, RMSE or Structural Similarity Index Measure (SSIM) similar to image quality assessments [Wan+04; Pal17; Liu+07]. A challenge in comparing final images is the sensitivity of the error metric to discrepancies in the position and orientation of the real and virtual cameras. Image registration techniques should be used prior to similarity calculations to mitigate these issues. The registration involves geometric transformations like translation or shear adjustments based on identifying corresponding points in both images and minimizing their positional differences, ensuring the images are as closely aligned as possible [Bro92]. Ultimately, determining an acceptable error threshold remains subjective, depending on the simulator's application and users.

Another objective evaluation method avoids directly comparing visual outputs by analyzing the behavior of downstream systems, algorithms or human users that use these outputs as inputs. The simulation is verified for that particular application when the performance and behavior of a downstream system or human users remains similar in simulated and real environments [Sin+22; Sun24].
Sundermeier [Sun24] uses a neural network for object detection as the downstream system, comparing the network's object detection accuracy between simulation and reality. A notable limitation of this verification method is the opaque nature of neural networks, which operate as black boxes, i.e., it is unclear how specific visual properties, such as physically accurate reflections, affect detection accuracy. Furthermore, there are differences between human and machine vision, so visual outputs that are not similar in human perception still provide similar results for machine vision algorithms (see Sec. A.5). Therefore, verification based on a neural network is limited to the specific network and conditions evaluated and cannot simply be generalized to human visual perception.

# 7

# Real-Time Supersampling Control of Matrix Headlights

This chapter presents the novel SSC algorithm for real-time matrix headlight control. SSC is a novel synthesis of ray tracing, SSAA and the MapReduce parallel processing method, resulting in an intuitive yet computationally efficient approach for rapidly prototyping dynamic lighting functions. The foundations of SSC are detailed in Ch. 3. This chapter is based primarily on the own publications [WB21b; WB21c; WB22; Wal+23b] and explains the SSC algorithm and the creation of dynamic lighting functions.

## 7.1. Combining Ray Tracing, Supersampling and MapReduce

The fundamental objective of any matrix headlight control system is to determine the pixel utilization vector $\boldsymbol{u}_\mathrm{p} \in \mathbb{R}^{n_\mathrm{p}}_{\geq 0 \wedge \leq 1}$, which generates the ideal LID for the current traffic and environment scenario in front of the ego-vehicle. This optimization challenge comprises three distinct but interconnected subproblems.
The first problem is the specification of a situation-optimal LID, which would be the best lighting for the current situation. This ideal LID assumes an ideal matrix headlight with an unlimited number of pixels and is independent of the current matrix headlight.
The second task is the calculation of a LID setpoint vector $\boldsymbol{i}_\mathrm{v,h,s} \in \mathbb{R}^{n_\mathrm{t}}_{\geq 0}$ for every matrix headlight from the ideal LID. $\boldsymbol{i}_\mathrm{v,h,s}$ is also independent of the current matrix headlight and can be used as a setpoint distribution for every headlight.
The third is the determination of the pixel utilizations $\boldsymbol{u}_\mathrm{p}$ that produces the closest approximation of $\boldsymbol{i}_\mathrm{v,h,s}$ by the actual illumination within the operational constraints of the current matrix headlight configuration.
Similar to the matrix headlight model for the headlight simulation in Sec. 5.2, a headlight for the controller can represent an arbitrary number of real modules with the same photometric center.

The SSC algorithm solves the situation-optimal lighting design with the novel Primitive-Based Lighting Design (PBLD) method and real-time pixel utilization calculation with the novel Ray Tracing Headlight Control (RTHC) and MapReduce Pixel Control (MRPC) methods. The SSC approach contains PBLD, RTHC and MRPC and combines ray tracing (see Sec. 3.1), SSAA (see Sec. 3.2) and MapReduce (see Sec. 3.3) to achieve flexible

real-time control of matrix headlights. This approach draws inspiration from real-time ray tracing rendering in computer graphics and is real-time capable on GPUs.

The real-time performance of the matrix headlight control is crucial for dynamic lighting adaptation to changing traffic conditions and user inputs, such as manual driving of the ego-vehicle. Contemporary research suggests that the maximum permissible system latency should not exceed 220 ms to prevent inadvertent dazzling of other road users [Kos+20; Sch+23], necessitating shorter optimization computation times. However, for a subjectively uniform lighting behavior without delays or abrupt changes, even shorter system cycles are required, whereby the exact times depend on the user group, similar to video games. The micro mirrors of a DMD matrix headlight "can be tilted up to 5,000 times per second" [Tro+19; Aud17] and LED headlights have an "update rate of up to 400 Hz" [Sch+23]. A DMD matrix headlight can react to inputs with a "latency of 1 to 2.5 ms" [Tam+14]. The SSC algorithm is designed to meet these real-time requirements by combining the RTHC and MRPC for parallelized calculations.

The process of designing optimal headlamp LIDs begins with the consideration of fundamental lighting requirements like maximal allowed brightness or environment landmarks to be marked (see Sec. 3.5.2). These fundamental design decisions (see Sec. 3.4 and 3.5) have to be made previous to using SSC.

In the first PBLD phase, the user defines the ideal LID with a 3D arrangement and parametrization of target 3D rectangles, the so-called primitives and optional the creation of default beam patterns for all modules (see Sec. 4.2.2).

In the second RTHC phase, SSC uses ray tracing for the calculation of module $I_v$ setpoints derived from the 3D configuration of the primitives. Therefore, SSC super-samples the pixel area in the regular grid of $\boldsymbol{I}_{v,h}$ and does not use pixel shape approximations. RTHC tests the cast ray against the primitives to calculate the module $I_v$ setpoints. Additionally, RTHC optional morphs the default beam pattern to create a digital rotation of the beam pattern for HLS and bend lighting.

In the third MRPC phase, SSC uses MapReduce by mapping the module $I_v$ setpoints to the pixels of the module and then reducing the pixel $I_v$ setpoints into optimal pixel utilizations incorporating SSAA by combining all pixel $I_v$ setpoints for smooth LID. This SSAA inspired approach enables anti-aliasing capabilities as requested by Gonçalves and Issoufou [GI19] for symbol projection and facilitates processing heterogeneous pixels with arbitrary shapes without pixel approximation techniques. Fig. 7.1 shows an example of SSC, where pixel utilizations are calculated by the ratio of rays hitting the bounding box.

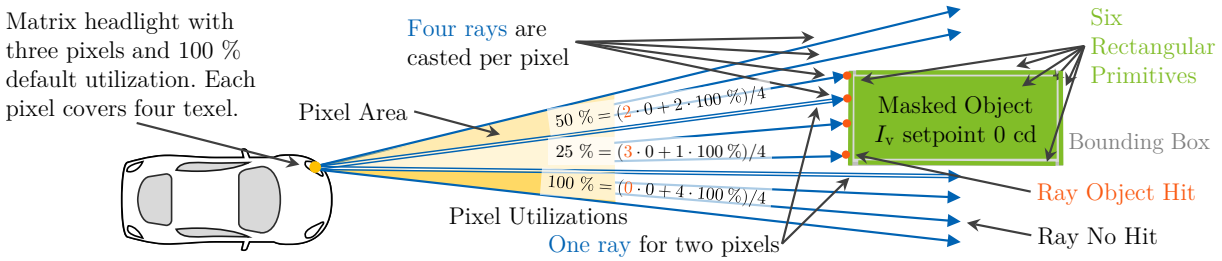



Figure 7.1.: Birdseye view of SSC controlling a three-pixel headlight. SSC casts four rays per pixel to every of its texel and tests them against a bounding box comprised of multiple primitives. Pixel utilizations are computed based on the ratio of rays hitting the bounding box [WB21b].

SSC begins with the PBLD method, which is based on three fundamental principles to allow lighting designers to define optimal lighting. PBLD uses the OBL paradigm [Bar+15] that the conceptualization of optimal lighting is primarily object-dependent, the realization that all known lighting functions are in principle symbol projections into 3D space and controlling a matrix headlight is in principle analogous to rendering operations in computer graphics for arbitrary computer monitor pixels.
PBLD combines the definition of default beam patterns like digital image editing and the parametrization of primitives (see Sec. 4.2.2). The lighting designer defines primitives, which are geometrically simple 3D objects serving as a definition of optimal desired lighting in the 3D world space for the pixel controller. Given that every lighting function is treated as a symbol projection, a primitive is a planar rectangular surface flying the 3D space containing a desired symbol image $\boldsymbol{I}_{\mathrm{s}} \in \mathbb{R}_{\geq 0 \wedge \leq 1}^{n_{\mathrm{row}} \times n_{\mathrm{col}}}$, augmented with lighting control parameters like target brightness or importance weighting. The primitive rectangle in the 3D space has in parametric form an origin point $\boldsymbol{p}_{\mathrm{o,p}} \in \mathbb{R}^3$, an optional normal vector $\boldsymbol{n} \in \mathbb{R}^3$ and two independent direction vectors $\boldsymbol{d}_{\mathrm{p},1} \in \mathbb{R}^3, \boldsymbol{d}_{\mathrm{p},2} \in \mathbb{R}^3$ as shown in Fig. 7.2. Thus, a primitive can be formally represented as the set $\mathcal{P}$ encompassing all these vectors and lighting control parameters.
The selection of rectangles in the 3D space as fundamental primitive objects for SSC, as opposed to the triangles for computer graphics, is primarily motivated by their ability to efficiently construct bounding boxes and their inherent similarity to rectangular images for symbol projection. Like a mesh of triangles in computer graphics, primitives can be combined to form complex lighting functions like masking cuboids for GFHB, which consists of six primitives.

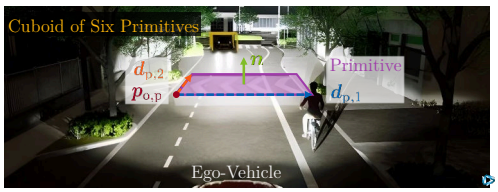



Figure 7.2.: Primitive rectangle and vectors in the 3D space.

The transformation from world-space primitives to a headlight setpoint LID is accomplished through a ray tracing rendering (see Sec. 3.1) with the RTHC method, where all primitives in principle are perspective projected onto a headlight luminous intensity setpoint texture $\boldsymbol{I}_{\mathrm{v,h,s}} \in \mathbb{R}_{\geq 0}^{n_{\mathrm{row}} \times n_{\mathrm{col}}}$, with the headlight conceptualized as a virtual pinhole camera and $\boldsymbol{I}_{\mathrm{v,h,s}}$ representing the resultant camera image [WB21b].
The headlight setpoint LID is initialized with a default beam pattern, e.g., low beam and is adjusted by the primitives. To realize HLS and bend lighting (see Sec. 3.4), the default beam pattern can be transformed by a morphing function such as a pattern shift (is explained in Sec. 7.3.2).
Following, parallelized ray tracing processes the primitives, wherein multiple rays are cast per headlight pixel from the photometric center through the texel of $\boldsymbol{I}_{\mathrm{v,h,s}}$ and tested against all primitives. The primitive properties at the hit points of these rays determine $\boldsymbol{I}_{\mathrm{v,h,s}}$. This method differs from conventional symbol projection techniques (see Sec. 3.5.2) because the ray tracing originates from the light source rather than the viewer, reversing the projection direction and processing all image points rather than a selected subset.

While the ray tracing approach may be computationally more intensive than conventional symbol projection techniques and rasterization rendering (see Sec. 3.5.3), it offers several significant advantages. It enables individual control of transparency and occlusion of setpoint LIDs based on the primitive parameters, facilitates the inclusion of reflection and refraction effects for complex lighting functions and allows future handling of overlapping beam patterns from multiple modules (is explained in Sec. 7.5.1). The propagation of the rays through the 3D space is calculated in parallel on the GPU for all rays of each headlight pixel, ensuring real-time capability.
The synthesis of primitive-based representation and ray tracing techniques provides lighting designers with flexibility in defining arbitrary lighting functions. By adjusting the parameters of the primitives, SSC enables the real-time implementation of arbitrary lighting functions like GFHB, symbol and video projection, variable lighting dimming and marker lighting functionality. Fig. 7.3 shows an illumination created by four modules and multiple primitives to give an impression of the flexibility of SSC.

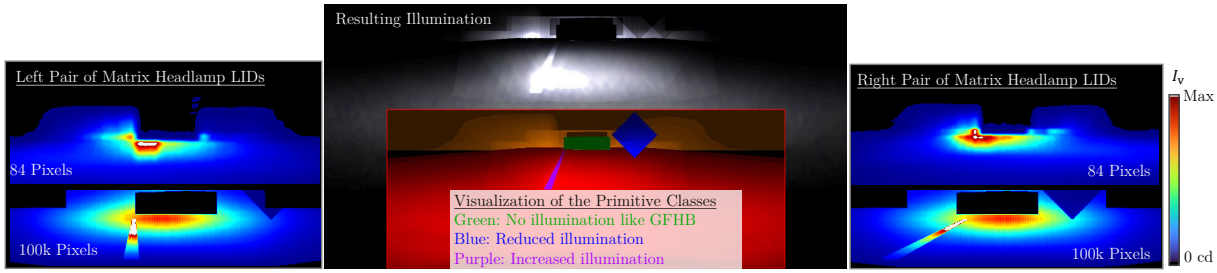



Figure 7.3.: Four headlight modules controlled by multiple primitives. The module's LIDs are shown in false colors at the sides and the primitive classes at the bottom [WB21b].

The subsequent pixel utilization control MRPC method after RTHC determines the optimal pixel utilization vector that generates the closest approximation to $\boldsymbol{I}_{\mathrm{v,h,s}}$ within the technological constraints of the current matrix headlight. Finding the best utilizations is a constrained non-linear optimization problem with $n_{\mathrm{t}}$ objective values from $\boldsymbol{I}_{\mathrm{v,h,s}}$, $n_{\mathrm{p}}$ Degrees of Freedom (DoF) and various optional non-linear constraints such as maximum energy consumption and thermal limitations.
While traditional approaches using a single non-linear JPO [PK18; Bau+21; Dan+21; Rüd+22] to handle pixel overlap effects and neighboring pixel influences, the JPO may become computationally intractable in real-time applications with large $n_{\mathrm{t}}$ or $n_{\mathrm{p}}$ values due to high dimensionality and non-linearity.
To be real-time capable, the proposed MRPC method uses MapReduce (see Sec. 3.3) to decompose the optimization problem into $n_{\mathrm{p}}$ independent subproblems that can be solved in parallel [WB21c]. Individual pixel $I_{\mathrm{v}}$ setpoints are mapped from $\boldsymbol{I}_{\mathrm{v,h,s}}$ to all pixels contributing to a respective lighting direction. The collective pixel setpoints are reduced to final utilization values through appropriate reduction functions. MRPC includes a limitation of pixel utilizations to ensure safety, thermal and energy consumption constraints compliance. Fig. 7.4 shows a program flow diagram of PBLD, RTHC and MRPC.

A distinguishing characteristic of SSC is its direct mapping of $I_{\mathrm{v}}$ setpoints to pixels without generating the intermediate $\boldsymbol{I}_{\mathrm{v,h,s}}$ while accounting for neighboring pixel effects indirectly through its mapping and reduction strategy [WB21c]. If MRPC is used without RTHC to calculate the pixel utilization, $\boldsymbol{I}_{\mathrm{v,h,s}}$ is the input of the MRPC algorithm.

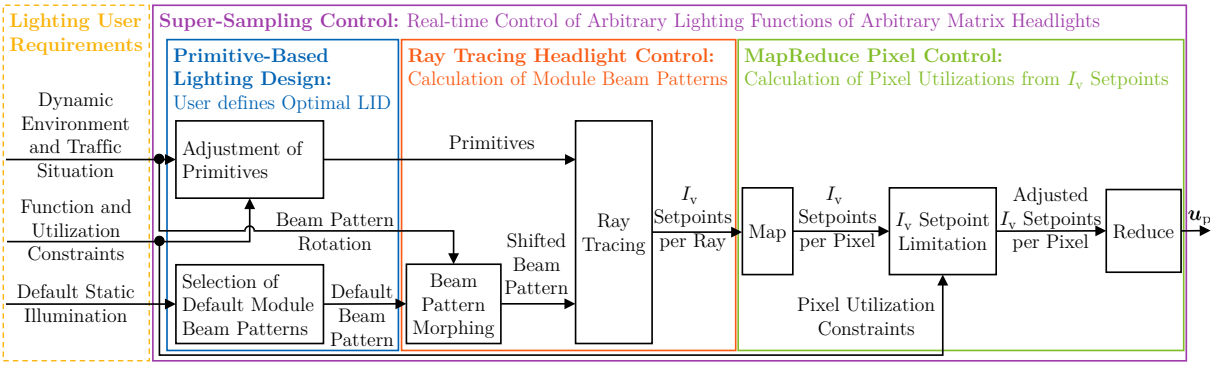



Figure 7.4.: SSC algorithm flow diagram with PBLD, RTHC and MRPC.

The mapping without a full intermediate setpoint LID differentiates SSC from existing headlight control algorithms like least-squares JPO (see Sec. 3.5.2). SSC does not use rectangular pixel masks or other approximations per default and is no iterative approach because ray tracing and MapReduce are only processed once per control step. SSC always uses the entire headlight area to calculate the pixel utilizations and not only a spatially limited selection area [Rüd+19b; Rüd20]. Additionally, SSC can be seen as a generalization of conventional control techniques such as bounding box pixel simplification (see Sec. 3.5.1) and rasterization rendering (see Sec. 3.5.3).

Through appropriate parameter adjustments, SSC can emulate these traditional methods while offering enhanced capabilities. SSC's utility extends beyond real-time control applications to identify optimal control strategies during pre-series development, enabling objective selection of the most efficient control algorithms for series production.

The quality of SSC-generated LIDs depends on the selected mapping and reduction strategies in MRPC. While the original SSC conception used influence-weighted mapping of $I_v$ setpoints to pixels, which means the possible brightest pixel has to contribute the most to a setpoint and a pixel setpoint RMSE minimization for reduction [WB21b; WB21c], alternative approaches may be better.

Fig. 7.5 shows SSC with the influence-weighted assignment of utilization values to the pixels. The $I_v$ setpoint is weighted by the ratio of the pixel $I_v$ to the sum of $I_v$ of all pixels contributing to the direction of the ray to calculate the mapped value. The mapped $I_v$ setpoints are divided by the pixel $I_v$ in its direction to determine the pixel utilizations, which are reduced by an average operation to the final value. Another possibility is the assignment of $I_v$ setpoints to the pixels with influence weighting and a reduction by the Moore-Penrose inverse of the pixel $I_v$ vector to minimize the quadratic error. Both approaches lead to different illuminations and choosing one method depends on lighting requirements and subjective preferences.

Comparative analysis between SSC solutions and those obtained through single nonlinear JPO reveals that while SSC achieves superior computational efficiency through $n_p$ parallelization, the resultant LIDs may exhibit suboptimal characteristics like slightly more blurred edges due to the absence of global optimization in the mapping process. The difference in quality depends on the specific lighting requirements and matrix headlight. Although the quality of SSC solutions could be enhanced through refined mapping and

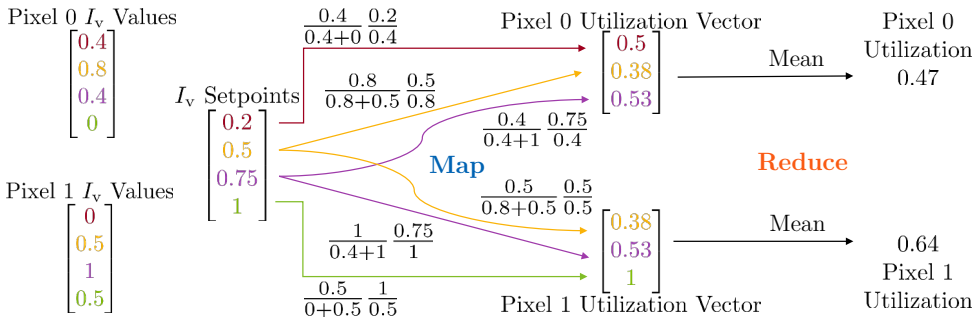



Figure 7.5.: SSC processing for overlapping dual-pixel configuration with four $I_v$ setpoints. SSC uses influence-weighted mapping of $I_v$ setpoints into pixel utilizations, followed by averaging-based reduction to determine pixel utilizations.

reduction functions tailored to specific matrix headlights, such improvements would require manual calibration and cannot guarantee equivalence with globally optimal solutions.
The justification for the SSC approach lies in the psychophysical characteristics of human visual perception and the limitations of machine vision systems. Given that minor variations in LIDs may be imperceptible to human observers and automated systems, absolute optimization of illumination quality becomes less critical and minor deviations from optimal patterns may be practically insignificant. The acceptable error threshold in this context is inherently subjective, depending on the human users' specific requirements and the particular evaluation scenario under consideration.

## 7.2. Primitive-Based Lighting Design

After the presentation of SSC, this section describes the PBLD of SSC of the 3D headlight-independent ideal LID. The fundamental principle underlying PBLD involves the strategic definition of the optimal spatial LIDs through the positioning of geometric primitives within the 3D world space by the user. Creating a default module beam pattern is optional because the primitives allow the definition of a complete LID.

### 7.2.1. Primitive Parametrization

While SSC processes $I_v$ values, the system architecture accommodates diverse setpoint quantities, which undergo systematic conversion to $I_v$ values during the ray tracing of RTHC. These target quantities are encoded within $\mathcal{P}$ of each geometric primitive. The principal setpoint quantities encompass absolute luminous intensity $I_{v,s} \in \mathbb{R}_{\geq 0}$, illuminance $E_{v,s} \in \mathbb{R}_{\geq 0}$, absolute pixel utilization $u_{p,a} \in \mathbb{R}_{\geq 0 \wedge \leq 1}$ and relative pixel utilization $u_{p,r} \in \mathbb{R}_{\geq 0}$ setpoints.
The absolute pixel utilization defines $I_{v,s}$ as $I_{v,s} = I_{v,m,i,j}\, u_{p,a}$, where $I_{v,m,i,j}$ denotes the maximum achievable $I_v$ for the pixel at the specified point, accessible through the headlight model of Ch. 5.
The relative pixel utilization setpoint $u_{p,r}$ is a multiplicative modification of the current or default pixel $I_v$, expressed as $I_{v,s} = I_{v,m,i,j}\, u_{p,r}\, u_{p,def,j}$, where $u_{p,def,j} \in \mathbb{R}_{\geq 0 \wedge \leq 1}$ denotes default utilization of the $j$-th pixel. $u_{p,r} = 0$ indicates pixel deactivation, while $u_{p,r} > 1$ signifies $I_v$ amplification. The MRPC phase ensures compliance with the limits so that $u_{p,j}$ remains in $[0, 1]$ after the reduction.

The conversion into $I_{\mathrm{v,s}}$ varies according to the selected setpoint quantity, which is done before the mapping of MRPC. Therefore, the mapping always uses $I_{\mathrm{v}}$ setpoints.
SSC transforms the absolute and relative pixel utilization to $I_{\mathrm{v}}$ by multiplying the setpoint with $I_{\mathrm{v,m},i,j}$ of the pixel.
Direct assignment as setpoints suffices without transformation for $I_{\mathrm{v}}$ values.
The conversion of $E_{\mathrm{v,s}}$ to $I_{\mathrm{v,s}}$ uses the photometric inverse-square law defined in (2.1.4). The distance $d$ is derived from the length $l$ of the ray casting specified by (3.1.1), measuring from the headlight's photometric center to the hit point with the primitive. When primitive orientation must be considered, the cosine of the angle of incidence $\cos(\theta)$ is computed via the dot product of the primitive's surface normal $\boldsymbol{n}$ and the negated ray direction vector $-\boldsymbol{d}_i$. Given normalized vectors, $I_{\mathrm{v}}$ is calculated by inverting (2.1.4) to

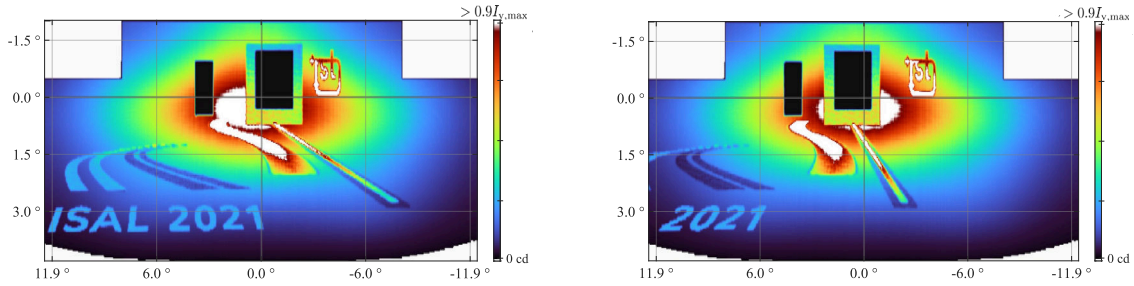
$$I_{\mathrm{v,s}} = E_{\mathrm{v}} \frac{d^2}{\boldsymbol{n} \cdot (-\boldsymbol{d}_i)}. \tag{7.2.1}$$

PBLD of SSC allows flexible primitive manipulation, enabling lighting design in Cartesian world coordinates in the environment rather than traditional LID spherical coordinate systems in the beam pattern. SSC eliminates the necessity for explicit smoothing regions around masked objects in GFHB applications, as it is necessary for Michenfelder [Mic14]. Instead, PBLD simplifies the creation of smooth GFHB with the possibility of nested lighting zones, which allows the positioning of larger primitives with partial lighting $0 < u_{\mathrm{p,r}} < 1$ encompassing smaller, fully masked primitives $u_{\mathrm{p,a}} = 0$ with forced ray hit properties. The custom ray tracing of RTHC supports customizable occlusion handling through hit priorities stored in $\mathcal{P}$. A forced ray hit designation ensures primitive detection regardless of intervening geometry, in contrast to the default behavior, which selects the nearest primitive intersection similar to physical correct ray propagation. The encapsulated large primitives realize the smoothing of the illumination without affecting the ray hits of the smaller inner ones.
Fig. 7.6 shows the resulting illumination of multiple overlapping primitives with varied setpoints. Fig. 7.6b shows the placement of the primitives for Fig. 7.6a, where the marker light comprises a small, forced hit, high-intensity line primitive ($u_{\mathrm{p,r}} > 1$) encapsulated by a more significant, reduced-intensity primitive ($u_{\mathrm{p,r}} < 1$). Fig. 7.7 shows the beam patterns of a matrix headlight with 100,000 pixels and non-overlapping elements for the scenario.


(a) Simulated fogy beam propagation. (b) Primitives.

Figure 7.6.: SSC controls two matrix headlights with multiple primitives [WB22].

SSC also incorporates primitive weighting through a weight $w \in \mathbb{R}_{\geq 0}$, stored in $\mathcal{P}$, which quantifies the setpoint's influence during reduction operations. A zero weight ($w = 0$) excludes the primitive from reduction calculations. In contrast, infinite weight ($w = \infty$)




(a) Left matrix headlight. (b) Right matrix headlight.

Figure 7.7.: Headlamp beam patterns with 100k pixels and no overlaps in false colors used in Fig. 7.6a. The color scale is limited to 90 % of the headlight maximal luminous intensity $I_{v,m}$ [WB22].

establishes absolute setpoint precedence. This functionality proves particularly valuable in safety-critical GFHB applications, where combinations of infinite weight and forced ray hits ensure consistent masking behavior under all operational conditions.

SSC supports primitive-specific projection source assignment by its custom ray tracing, enabling precise control over multi-module headlamps by defining the relevant headlights for a primitive in PBLD. Irrelevant primitives are not considered for the hit test of the ray tracing in RTHC.
This selectivity is valuable when integrating matrix headlamps incorporating high- and low-resolution modules (see Fig. 2.2), exclusively allowing selective symbol projection through high-resolution components. Additionally, this feature allows the implementation of separated projection spaces for left and right headlights, as proposed by Rivero et al. [Riv+15], through side-specific source assignments.

SSC supports the reproduction of arbitrary LIDs by multiple modules with overlapping illuminations through ray tracing. Similar to Michenfelder [Mic14], setpoint LIDs can be implemented as textures mapped onto primitives positioned vertically at a sufficient distance from the ego-vehicle. As the beam intersection points of multiple headlights vary with the projection distance, the resulting LID is only available in the desired form at the target distance. Therefore, separate setpoint LIDs are recommended for the left and right headlight to maintain projection accuracy, as this reduces the photometric center-to-center distance of the generating modules. To avoid unnecessary computations of not changing base beam pattern and unlike Michenfelder [Mic14], SSC loads the basic beam pattern for all modules during initialization from memory and does not calculate it every step.

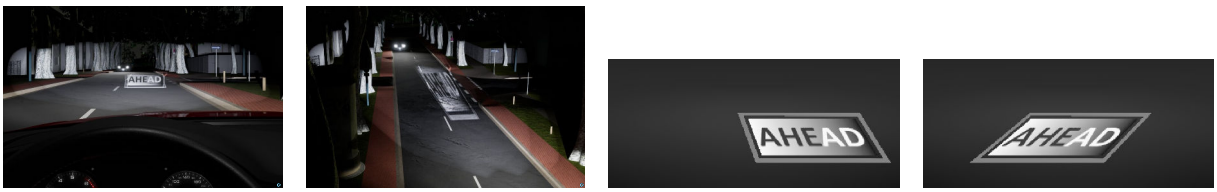
### 7.2.2. Creating Dynamic Lighting Functions

The most straightforward application of SSC involves the projection of homogeneous rectangular light patterns for GFHB (see Fig. 3.5) and marker lights (see Fig. 3.7b). For SSC GFHB, marker light and sign dimming functionalities are conceptualized as fundamental homogeneous symbol projection operations. A homogeneous LID is realized through an arrangement of primitives, incorporating a symbol image characterized by a singular constant setpoint. The geometric properties of the emitted lighting are determined by the primitive's position, orientation and size.
While a marker light implementation necessitates only a single primitive, the GFHB functionality with SSC requires six primitives to construct a fully enclosing cuboid bounding

box around the target traffic object, like the typical GFHB algorithm (see Sec. 3.5.1). The six primitives are positioned with normal vectors oriented outward to emulate the faces of a bounding box, as seen in Fig. 7.6b in dark teal. To achieve reliable traffic object masking, all bounding box primitives must have $I_\mathrm{v}$ setpoints of zero with an infinite weighting to ensure that a single ray hit is sufficient to darken the object completely. Baumann et al. [Bau+21] suggests an alternative approach in which a single plane is positioned as a masked area in front of the ego vehicle. This method reduces the number of primitives to be evaluated. However, additional calculations are required to ensure appropriate dimensioning for object masking from all directions. The masking cannot be adapted precisely to the target object.
Marker light is achieved through a single primitive positioned on the road surface, oriented toward the target object with an upward-facing normal vector as seen in Fig. 7.6b in purple and blue. Traffic sign lighting control also uses a single primitive matching the sign's surface, with normal vectors oriented toward the ego-vehicle. The primitive's setpoint is configured to augment pixel $I_\mathrm{v}$ and the weight can be modulated to optimize anti-aliasing and edge smoothness. The visual quality of GFHB mask transitions or marker line contrast can be enhanced by using supplementary primitives incorporating gradient setpoints for illumination reduction, as seen in Fig. 7.6b in red and purple.

The implementation of inhomogeneous marker lights (see Fig. 3.7c) and complex lighting functions (see Fig. 3.8) necessitates a generalized symbol projection method. A grayscale symbol image $\boldsymbol{I}_\mathrm{s}$ defines the desired projected symbol instead of a single value, where the maximum value of $\boldsymbol{I}_\mathrm{s}$ corresponds to the maximal $I_\mathrm{v}$ setpoint. The $\boldsymbol{I}_\mathrm{s}$ undergoes normalization to the interval $[0, 1]$ and modulates the maximal $I_\mathrm{v}$ according to the texel value intersected by the ray.
The symbol image's spatial configuration is determined by its primitive, enabling arbitrary symbol projection lighting functions through the design of $\boldsymbol{I}_\mathrm{s}$ and primitive positioning. The $\boldsymbol{d}_{\mathrm{p},1}$,$\boldsymbol{d}_{\mathrm{p},2}$ vectors of the rectangle are the 3D basis vectors for the light texture $u \in [0, n_\mathrm{col}-1]$ and $v \in [0, n_\mathrm{col} - 1]$ coordinates, which allows a transformation of ray hit position to texture coordinates. The primitive weight parameter facilitates fine-tuning anti-aliasing characteristics and projected symbol edge quality.

An application case for SSC and ray tracing involves distortion-free symbol projection for a specified observer (see Sec. 3.5.2). Fig. 7.8 shows the resulting symbol projected by two headlights for the desired driver observer and parts of the utilization matrices of the headlights. Similar to Fig. 3.12 and Fig. 7.7, the symbol is distorted in the utilization matrices to appear correctly for the desired observer.




(a) Desired driver view perspective. (b) Different view perspective. (c) Left grayscale pixel utilization matrix. (d) Right grayscale pixel utilization matrix.

Figure 7.8.: Projection of a symbol for the driver by the left and right headlight. The pixel utilization matrices of both matrix headlights are equally cropped and gray-scaled.

Implementing observer-specific symbol projection with SSC requires both a primary primitive representing the projection surface and an auxiliary rectangle flying in the 3D space representing the desired symbol image similarly. As seen in Fig. 3.11, the desired image is the imaginary appearance of the symbol for the intended observer by the reflected light from the projection surface. The auxiliary rectangle is used for a secondary ray casting from the primary primitive to the observer position to determine the symbol image's projected appearance on the projection surface, as seen in Fig. 7.9.
The origin and direction vectors of the desired symbol rectangle and the observer position are used for primitive dimensioning calculations. The primitive calculation is similar to the general symbol projection approach from Sec. 3.5.2, initially establishing projection surface orientation and position, followed by perspective projection of desired symbol rectangle vertices from the observer's viewpoint onto the surface. The resulting four-point configuration determines minimal primitive dimensions, ensuring the complete intersection of observer-directed rays with the desired symbol rectangle. Using the properties of the desired image to define the primitive ensures that all rays reflected from the projection surface to the observer that hit the symbol rectangle must first hit the primitive. The primitive is thus a filter for the cast rays to avoid unnecessary secondary ray casting that occurs only when a ray hits the primitive.
Fig. 7.9 shows an example of the projection of the desired symbol rectangle to the projection surface and the resulting primitive. This projection approach shares similarities with Baumann et al. [Bau+21], differing primarily in its projection of edge points rather than complete image and does not require any pre-distortion of the symbols, as the ray tracing tracks the observer's rays of vision. The ray tracing allows the dynamic handling of shadows and occlusions between the observer and the surface.

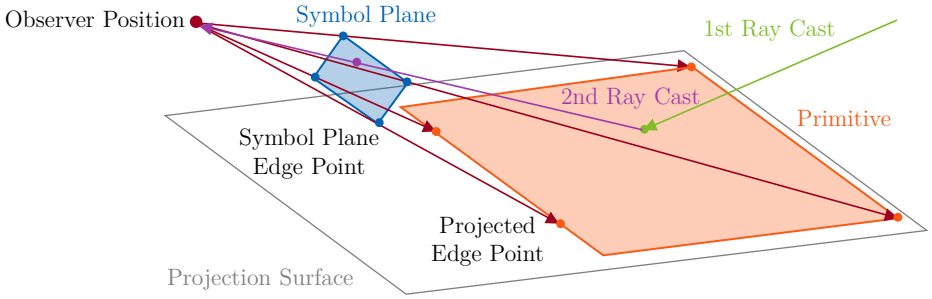



Figure 7.9.: Projection of edge points of the symbol plane to the projection surface to create the primitive. The projected edge points form a trapezoid on the projection surface like the symbol projection shown in Fig. 7.8b, so the primitive is the smallest rectangle enclosing the trapezoid.

Dynamic symbol animations like welcome light and vehicle path highlighting (see Sec. 3.5) are achieved through dynamic updating of the symbol images of the primitives. A frame-by-frame video stream can update the symbols [Win20; Sei+21].
An alternative is real-time vector graphics generation based on image parameters like splines. This is analogous to curved line rendering [Fol+97], using the primitive symbol image as a planar drawing surface. The projected symbols are distorted along defined curves, such as curved chevron arrows communicating vehicle turning intentions or navigation instructions for the driver as seen in Fig. 4.6 on the road turning right. Sec. B.2 details the path projection process by Bézier curves [Far01], which are selected for path definition because of their intuitive control point manipulation.

SSC supports dynamic LID animation by modification of the primitives and dynamic $\boldsymbol{I}_\mathrm{s}$ updating. Both methods can lead to identical-looking results like projecting moving lines, either with moving individual primitives for each line or using one large primitive with a dynamic line symbol. As SSC creates static lighting distributions, realizing dynamic functions like masking a moving vehicle requires connecting SSC to a high-level lighting planning algorithm (see Sec. 4.1), which defines the primitives according to the current traffic situation. Fig. 7.10 shows conditional dynamic lighting animations, where the illumination changes according to the environment state.

(a) Yellow target object away. (b) Yellow target object near.

Figure 7.10.: Conditional dynamic lighting animations. When the yellow-boxed vehicle is far away (a), the matrix headlight projects a snowflake symbol defined by blue and orange primitives. When the yellow-boxed oncoming vehicle is near (b), a blinking chevron arrow symbol is projected for an evasive maneuver of the oncoming vehicle defined by the purple primitive [GW24].

## 7.3. Ray Tracing Headlight Control

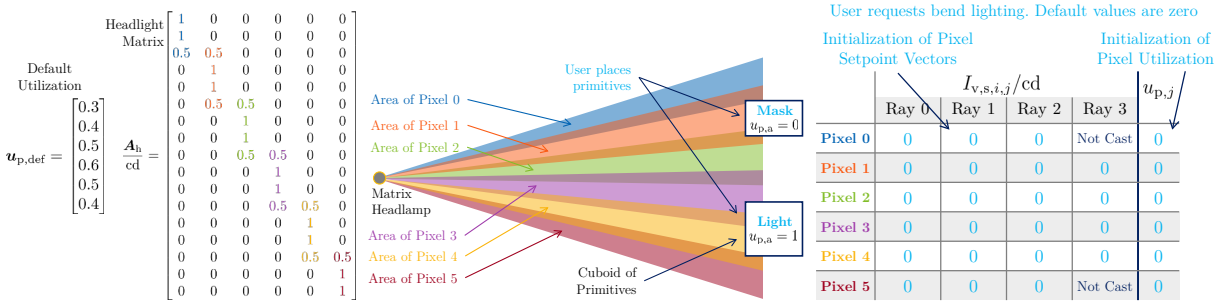
After the user defines the primitives, the symbol images and optional selects the default beam patterns, RTHC calculates the desired LID of every module of the matrix headlight. Humans or high-level lighting planning algorithms can do the primitive setup and activate the light morphing phase. SSC is real-time capable because of the ray parallel calculation of the ray tracing in RTHC and pixel parallel processing in MRPC.

### 7.3.1. Initialization Process and Data Structures

The initialization of SSC for each control step begins with transferring all primitives and default illuminations to the GPU memory. SSC uses the same headlight model as the simulation (see Sec. 5.3) for knowledge of the headlight characteristics.
Two vector data structures for each pixel are the intermediate data for MapReduce, similar to Fig. 3.3. These vectors of the $j$-th pixel are the luminous intensity setpoint vector $\boldsymbol{i}_{\mathrm{v,p,s},j} \in \mathbb{R}_{\geq 0}^{n_{\mathrm{t},j}}$ and the weight vector $\boldsymbol{w}_j \in \mathbb{R}_{\geq 0}^{n_{\mathrm{t},j}}$, where $n_{\mathrm{t},j} \in \mathbb{N}$ is the number of light texture elements and associated ray projections for the $j$-th pixel. Each element $I_{\mathrm{v,s},i,j} \in \mathbb{R}_{\geq 0}$ within $\boldsymbol{i}_{\mathrm{v,p,s},j}$ is an individual mapped $I_\mathrm{v}$ setpoint to the $j$-th pixel from the $i$-th ray of this pixel and $w_{i,j} \in \mathbb{R}_{\geq 0}$ within $\boldsymbol{w}_j$ the weight of the setpoint of the $i$-th ray. $\boldsymbol{i}_{\mathrm{v,p,s},j}$ and $\boldsymbol{w}_j$ are populated during the mapping phase and accessed during the reduction phase.

The initialization of $\boldsymbol{w}_j$ is a standard weight assignment at the beginning of every iteration. However, the initialization method for $\boldsymbol{i}_{\mathrm{v,p,s},j}$ depends on whether the digital transformation of the base LID, such as HLS or bend lighting, is required. SSC operates assuming that

each module maintains a predefined beam pattern, e.g., town light, with a default pixel utilization vector $\boldsymbol{u}_{\mathrm{p,def}} \in \mathbb{R}^{n_{\mathrm{pix}}}_{\geq 0 \wedge \leq 1}$, which undergoes dynamic modification during the ray tracing. In scenarios where digital transformation is not user-specified, $\boldsymbol{i}_{\mathrm{v,p,s},j}$ and $\boldsymbol{u}_{\mathrm{p}}$ are initialized with predefined $I_{\mathrm{v}}$ values and utilizations from $\boldsymbol{u}_{\mathrm{p,def}}$ that generate the user-selected default beam pattern. Conversely, when digital transformation is required, $\boldsymbol{i}_{\mathrm{v,p,s},j}$ and $\boldsymbol{u}_{\mathrm{p}}$ are initialized to zero values, as SSC uses a pixel update database to optimize computational efficiency by executing the reduction process exclusively for pixels that could deviate from the default state. For no digital transformation, the default state is the default beam pattern, but the default state is deactivated with transformation. Fig. 7.11 shows a SSC initialization example for a bend lighting and two cuboids of primitives.



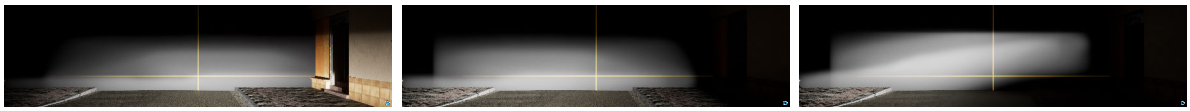

(a) $\boldsymbol{u}_{\mathrm{p,def}}$ and $\boldsymbol{A}_{\mathrm{h}}$. (b) Birdseye view. (c) Table of $\boldsymbol{i}_{\mathrm{v,p,s},j}$ and $\boldsymbol{u}_{\mathrm{p}}$.

Figure 7.11.: SSC initialization example. The matrix headlight has six pixels. The outer consists of three texels and the inner of four. The pixels overlap in one texel. The user places two cuboids of primitives and requests bend lighting to the left.

### 7.3.2. Digital Beam Pattern Morphing

The beam pattern morphing algorithm of SSC for digital movement represents a synthesis of the rotation angle-dependent pixel $I_{\mathrm{v}}$ transformation proposed by Saralajew and Benderman [SB16] (see Sec. 3.4) and the illumination image projection method developed by Michenfelder [Mic14] (see Sec. 3.5.3). The SSC morphing algorithm loads the desired beam pattern into a symbol image $\boldsymbol{I}_{\mathrm{s}}$ of a primitive, which is then positioned in front of the headlight as a projection screen. Adjusting the projection screen with the setpoint LID has eight DoF, which are the position, rotation and size of the rectangular primitive. The eight DoF enable SSC to address not only HLS and bend lighting requirements but also to compensate for vehicle body roll dynamics.

For HLS and bend lighting, the primitive undergoes position, pitch and yaw transformations that emulate the changing lighting directions of a physically articulating module. The primitive represents the lighting of the adjusted module on a projection screen. The transformation of the primitive is done by multiplying $\boldsymbol{p}_{\mathrm{o,p}}$ and $\boldsymbol{d}_{\mathrm{p},1}$, $\boldsymbol{d}_{\mathrm{p},2}$ with a transformation matrix, which is parameterized according to the desired lighting movement. Fig. 7.12 shows resulting images of moved beam patterns in the headlight simulation.
The morphing algorithm uses ray casting like the primitive processing in Sec. 7.3.3, so when the user requests morphing, SSC uses the same ray tracing algorithm twice. For morphing, the target is only one primitive with the desired lighting. The ray casting process initiates from the headlight's photometric center through all texel of $\boldsymbol{I}_{\mathrm{v,h}}$ toward



(a) Beam patterns of two modules without morphing. (b) Digital bend lighting to the left side. (c) Combined digital bend, pitch and roll transformation.

Figure 7.12.: Digital bending and pitching of beam patterns on a wall. Yellow lines indicate the headlamp position. The sharp edges are the limits of the module's illumination area.

the primitives to calculate the $I_\mathrm{v}$ setpoints based on the hits. Fig. 7.13 shows a schematic example of the digital bend lighting to the left side.

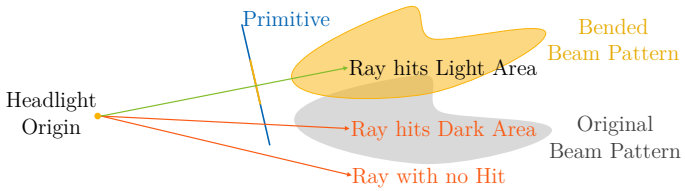



Figure 7.13.: Digital morphing of base lighting for lateral displacement as bend lighting. The orange rays intersect regions where $I_\mathrm{v} = 0$ and do not trigger pixel utilization updates. The green ray hitting an illuminated region initiates the reduction process.

Upon ray intersection with a primitive and when the derived $I_\mathrm{v}$ setpoint from $\boldsymbol{I}_\mathrm{s}$ exceeds zero, the setpoint is mapped to all pixels along the ray path and stored in $\boldsymbol{i}_{\mathrm{v,p,s},j}$. The mapping is explained in Sec. 7.4.1 and is used twice during SSC, which allows using an optimal separate mapping strategy for processing the morphed beam pattern in this phase and the primitives of the following phase. The ray casting and mapping method for digital beam pattern morphing eliminates the requirement for pixel approximation while enabling parallel computation across all ray casts. To optimize computational efficiency, rays that either miss the primitive or yield zero $I_\mathrm{v}$ setpoints do not trigger the reduction process. Fig. 7.14 shows the bend lighting with ray casting against a primitive following Fig. 7.11.

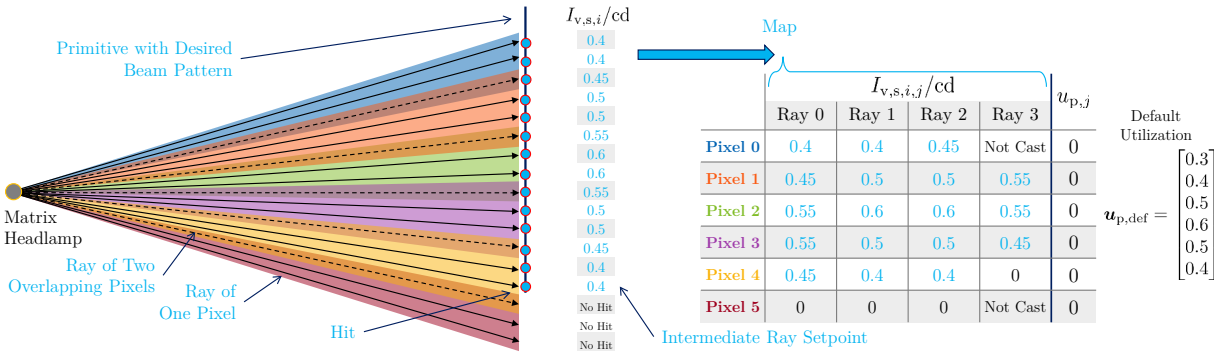



Figure 7.14.: SSC bend lighting example following Fig. 7.11. The desired illumination is switched three texels to the left. After a hit, the setpoints are mapped into $\boldsymbol{i}_{\mathrm{v,p,s},j}$.

### 7.3.3. Ray Tracing for Primitive Analysis

SSC uses the same pyramid-shaped headlight model as the simulation in Sec. 5.1 for primitive processing. The model sharing eliminates the transformation of primitive Cartesian positions to headlight angles as required for GFHB (see Sec. 3.5.1) to reduce

computation effort. The transformation of setpoints to pixels is accomplished through column indices derived from the SpMV format, e.g., the *ColumnIndices* array of CSR.

Because of the usage of a pyramid-shaped punctiform headlight model, the ray tracing algorithm first casts rays from the origin of the $i$-th ray $\boldsymbol{p}_{\mathrm{o},i} = [x_\mathrm{h}\ y_\mathrm{h}\ z_\mathrm{h}]^\top$, which is the headlight photometric center, through the centroid of each texel within the light texture $\boldsymbol{I}_{\mathrm{v,h}}$, imaginary scaled with $\phi$, $\psi$ and positioned at $d$ according to (2.3.3). This positioning ensures beam pattern FoV compliance in front of the headlamp, which is treated as a pinhole camera. Ray casting has similarities with the lighting simulation shown in Fig. 2.6. Hence, the ray tracing headlight control conceptually resembles the rendering control method (see Sec. 3.5.3).
For a vehicle body maintaining a neutral orientation, the ray direction vector $\boldsymbol{d}_i$ depends on the $u, v$ texel image coordinates as

$$\boldsymbol{d}_i = \begin{bmatrix} d & \frac{2u}{\max\{n_\mathrm{col}, n_\mathrm{row}\}} - 1 & \frac{2v}{\max\{n_\mathrm{col}, n_\mathrm{row}\}} - 1 \end{bmatrix}^\top . \tag{7.3.1}$$

The vector $\boldsymbol{d}_i$ undergoes normalization as $\boldsymbol{d}_i / ||\boldsymbol{d}_i||$ to unit length in world space coordinates before being used for ray casting.
The ray casting algorithm evaluates each ray against all primitives to determine the hit point. The primitive-ray intersection test is a line-plane intersection problem in 3D space [Gla89; Ake+18]. The ray parameters define the line in (3.1.1). The plane is characterized by the primitive's origin $\boldsymbol{p}_{\mathrm{o,p}}$ and direction vectors $\boldsymbol{d}_{\mathrm{p},1}$, $\boldsymbol{d}_{\mathrm{p},2}$ (see Sec. 7.1). The hit point of the $i$-th ray with the plane is computed by solving the line-plane intersection equation in parametric form, incorporating (3.1.1) of the ray propagation as

$$\boldsymbol{p}_{\mathrm{o},i} + \boldsymbol{d}_i\, l = \boldsymbol{p}_{\mathrm{o,p}} + u\, \boldsymbol{d}_{\mathrm{p},1} + v\, \boldsymbol{d}_{\mathrm{p},2}. \tag{7.3.2}$$

An intersection satisfies the conditions $0 \leq l$, $0 \leq u \leq 1$ and $0 \leq v \leq 1$ of the unknowns. The $u, v$ values are symbol image coordinates. The vector equation can be reformulated as a linear system in matrix notation to

$$\begin{bmatrix} \boldsymbol{d}_i & -\boldsymbol{d}_{\mathrm{p},1} & -\boldsymbol{d}_{\mathrm{p},2} \end{bmatrix} \begin{bmatrix} l \\ u \\ v \end{bmatrix} = \boldsymbol{p}_{\mathrm{o,p}} - \boldsymbol{p}_{\mathrm{o},i}. \tag{7.3.3}$$

The solution is obtained through the inversion of a $3 \times 3$ matrix [Ake+18] when the matrix is non-singular for a non-parallel ray-plane orientation. The resulting $u, v$ coordinates are used to retrieve the $\boldsymbol{I}_\mathrm{s}$ texel value, while the ray length determines the distance from the hit point to the headlight photometric center.
To prevent back-face intersections, SSC uses the primitive's $\boldsymbol{n}$ vector to calculate the cosine of the impact angle $\theta$ as $\boldsymbol{n} \cdot (-\boldsymbol{d}_i)$ prior to solving (7.3.3). Intersections are disregarded when the cosine is negative and $|\theta| > 90°$, indicating a back-face hit. Therefore, the orientation of $\boldsymbol{n}$ of the primitives during the lighting design is important (see Sec. 7.2).
When the hit point is proper, the luminous intensity setpoint $I_{\mathrm{v,s},i} \in \mathbb{R}_{\geq 0}$ of the $i$-th ray is calculated from the multiplication $\boldsymbol{I}_\mathrm{s}$ texel value with the primitive's target brightness. Then $I_{\mathrm{v,s},i}$ is mapped to the corresponding pixels of this ray. When the mapped setpoint differs from the default value, the reduction process of this pixel is initiated. Fig. 7.15 shows the ray casting for two modules through their respective light textures.
Given the high computational complexity of solving (7.3.3), SSC incorporates a single simple bounding volume around primitive groups like the six planes used to form a bounding

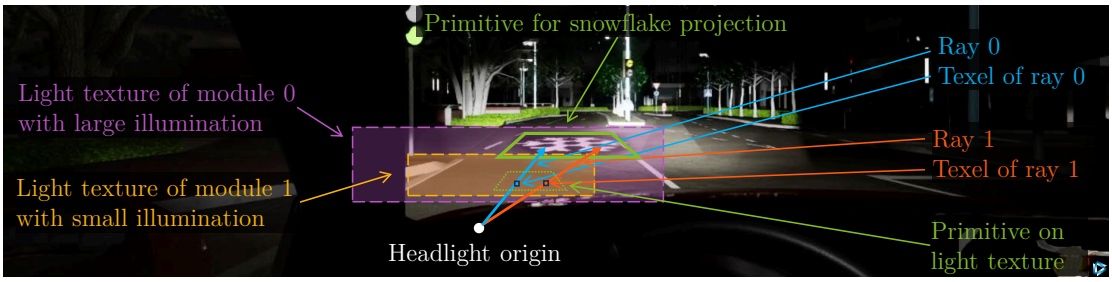



Figure 7.15.: Ray casting from two modules through their light texture against a primitive.

box for GFHB. The ray is first tested against the bounding volume. If the ray intersects the bounding volume, it is tested against the primitives inside [Gla89; Wal+07; Mei+21]. One efficient bounding volume is a minimum-volume ellipsoid [Gla89] encompassing all primitives and tangent to their edge points. For an ellipsoid with a size vector $\boldsymbol{s}_{\mathrm{el}} \in \mathbb{R}^3_{\geq 0}$ of expansion in every direction, a general radius $r_{\mathrm{el}}$ and origin $\boldsymbol{p}_{\mathrm{o,e}} \in \mathbb{R}^3$, the intersection condition is inserting the ray equation (3.1.1) into the ellipsoid equation similar to the sphere intersection of (3.1.2) and solving for $l$ as

$$|| \left(\boldsymbol{p}_{\mathrm{ray},i}(l) - \boldsymbol{p}_{\mathrm{o,e}}\right) \cdot \boldsymbol{s}_{\mathrm{el}} ||^2 = r_{\mathrm{el}}^2 \tag{7.3.4}$$

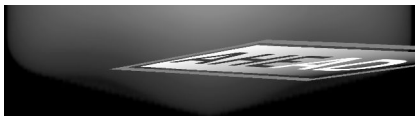
This equation reduces to a single quadratic equation of $l$, solvable via the quadratic formula. An intersection exists when one or two real solutions satisfy $l \geq 0$. Upon confirming an ellipsoid intersection, SSC proceeds to test against the contained primitives.
The ray tracing handles primitive transformations and headlight module adjustments through homogeneous transformation matrices. When a module is rotated or translated, the ray direction and origin are updated accordingly before solving (7.3.3).
Therefore, SSC facilitates symbol projection stabilization and vehicle body movement compensation without additional modifications and uses the reduction phase for anti-aliasing. However, rather than using illumination shifts in the beam pattern like Yargeldi [Yar23a] to counter vehicle movement, SSC modifies the homogeneous transformation matrix $\boldsymbol{T}_{\mathrm{hl,wo}} \in \mathbb{R}^{4\times4}$ mapping headlight space to world coordinates according to vehicle body position and orientation. The ray origin is computed as $[\boldsymbol{p}_{\mathrm{o},i}^\top \; 1]^\top = \boldsymbol{T}_{\mathrm{hl,wo}} \, [0 \; 0 \; 0 \; 1]^\top$ and the ray direction $\boldsymbol{d}_i$ undergoing identical rotation with $\boldsymbol{T}_{\mathrm{hl,wo}} \, [\boldsymbol{d}_i^\top \; 0]^\top$ to match actual vehicle body orientation. Vehicle body parameters are estimated using onboard sensor data [Yar23b].

The ray tracing control method shares many similarities with rasterization rendering control (see Sec. 3.5.3) but introduces advantages in handling shadows, reflections and complex lighting functions, albeit at increased computational cost. While some advanced features have yet to be fully implemented, the following theoretical concepts are intended to justify the chosen ray tracing approach over rasterization rendering.
The key innovation of the ray tracing control approach is the ability to perform secondary ray casts from primitive hit points to various targets, including observers, supplementary light sources, or additional surfaces. This capability enables rapid prototyping of dynamic environment-adaptive lighting functions like observer-specific projections or road lighting optimal dynamic town light (is explained in Sec. 7.5.1).
One implemented secondary ray cast application, described in Sec. 7.2.2, enables distortion-free symbol projection for the desired observer by tracking the observer's vision rays without

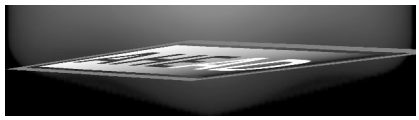
requiring symbol pre-distortion on the primitive. The second ray is cast from the initial hit point of (7.3.3) to the observer position and tested against the desired symbol image plane. When the first ray does not hit the corresponding primitive, the second ray cast is not performed. Fig. 7.16 illustrates a symbol projection by two headlights for an observer on the bicycle path and corresponding lower headlight utilization matrices.


(a) Desired observer perspective.

(b) Driver perspective.



(c) Left grayscale pixel utilization matrix.



(d) Right grayscale pixel utilization matrix.

Figure 7.16.: Symbol projection by two headlights for an observer positioned on the left cycle path at 8 m distance. The symbol appears correctly oriented only from the intended observer position, achieved through calculated distortion in both headlight utilization matrices. The pixel utilization matrices of both matrix headlights are equally cropped and gray-scaled.

## 7.4. MapReduce Pixel Control

After RTHC calculates the desired LID of every module of the matrix headlight, MRPC adjusts the pixel utilizations to match the setpoints best. The pixel utilization adjustment is a parallelized process across all pixels and texels, which is computationally intensive but real-time capable due to the pixel parallel processing.

### 7.4.1. Mapping

The MapReduce method (see Sec. 3.3) allows the calculation of the pixel utilizations from the ray $I_\mathrm{v}$ setpoint $I_{\mathrm{v,s},i}$ and weights in real-time. The ray tracing computation is parallelized across all rays. MapReduce enables parallel processing across all $I_{\mathrm{v,s},i}$ elements and headlight pixels. In contrast to alternative control approaches discussed in Sec. 3.5.2, SSC directly maps the setpoints to the pixels, eliminating the necessity for intermediate illumination setpoint image generation. However, MRPC can be applied to target $\boldsymbol{I}_{\mathrm{v,h,s}}$ and weights to calculate the pixel utilizations.

Following ray-primitive hit detection of RTHC, MRPC maps parallelized for all texels the $I_{\mathrm{v,s},i}$ of the primitives to the pixels $\boldsymbol{i}_{\mathrm{v,p,s},j}$, while simultaneously storing corresponding weights in $\boldsymbol{w}_j$. The weights remain unchanged and are directly stored in $\boldsymbol{w}_j$ at the index position of the corresponding ray. The parallelization of the mapping in MRPC is possible because each ray and texel has a fixed index for storing the values in $\boldsymbol{i}_{\mathrm{v,p,s},j}$ and $\boldsymbol{w}_j$, which are defined in the pre-processing phase of the headlight model.
SSC uses the identical headlight model as the simulation in Sec. 5.3 to know the individual pixel influences on ray directionality for the $I_\mathrm{v}$ mapping. Following mapping completion,

the mapped $I_\mathrm{v}$ setpoints are written to the $\boldsymbol{i}_{\mathrm{v,p,s},j}$ at the corresponding ray index position. When they differ from the default values, the pixel utilization updates start through the reduction process.

While multiple variants of $I_\mathrm{v}$ mapping functions are theoretically possible, this thesis concentrates on direct mapping (i.e., direct assignment without mapping) and potentiated influence mapping. The subjective requirements for the resulting illumination predominantly influence the selection of mapping functions.
The direct mapping function transfers $I_{\mathrm{v,s},i}$ values to pixels without modification. This approach is recommended for target absolute and relative pixel utilization setpoints $u_\mathrm{p,a}$ and $u_\mathrm{p,r}$ because of its unmodified value transfer to fulfill the targets. For direct mapping of $u_\mathrm{p,a}$ or $u_\mathrm{p,r}$, the utilization setpoints undergo conversion via $I_{\mathrm{v,m},i,j}$ into $I_\mathrm{v}$ values, which are subsequently stored in $\boldsymbol{i}_{\mathrm{v,p,s},j}$ at the corresponding ray's index position.
The potentiated influence mapping function incorporates pixel influence weighting relative to the cumulative influence of all pixels on the cast ray. This method ensures that pixels that reach the maximum $I_\mathrm{v}$ in the direction of the ray contribute the proportionally largest share to the $I_\mathrm{v}$ setpoint, optimizing energy efficiency when power consumption correlates linearly with pixel emitted $I_\mathrm{v}$. The mapping of $I_{\mathrm{v,s},i}$ to the pixels of the same module uses only $I_{\mathrm{v,m},i,j}$ for influence calculations, as the $d$ is the same for all pixels along a given ray. The potentiated influence mapping converts each $I_{\mathrm{v,s},i}$ of the $i$-th ray into a mapped pixel luminous intensity $I_{\mathrm{v,s},i,j}$ for the $j$-th pixel by weighting $I_{\mathrm{v,s},i}$ with the potentiated influence $I_{\mathrm{v,m},i,j}$ of each pixel normalized by the sum of all potentiated pixel influences on the target. This energy-efficient mapping function is defined as

$$I_{\mathrm{v,s},i,j} = \frac{I_{\mathrm{v,m},i,j}^{a_\mathrm{m,ssc}}}{\sum_{j \in \mathcal{C}_{\mathrm{p,gz},i}} I_{\mathrm{v,m},i,j}^{a_\mathrm{m,ssc}}} I_{\mathrm{v,s},i}, \tag{7.4.1}$$

where $a_\mathrm{m,ssc} \in \mathbb{R}_{\geq 0}$ is the mapping equality control parameter. For increasing $a_\mathrm{m,ssc}$, pixels with high $I_\mathrm{v}$ capability will contribute proportionally more to the $I_\mathrm{v}$ setpoint because higher exponentiation gives greater weight to large values. For simplicity of (7.4.1), $0^0 := 0$ for this thesis, so setting $a_\mathrm{m,ssc} = 0$ results in uniform $I_{\mathrm{v,s},i}$ mapping across all pixels capable of lighting in the ray direction. Fig. 7.17 continues Fig. 7.14 with the SSC ray tracing and mapping process for two cuboids of primitives.

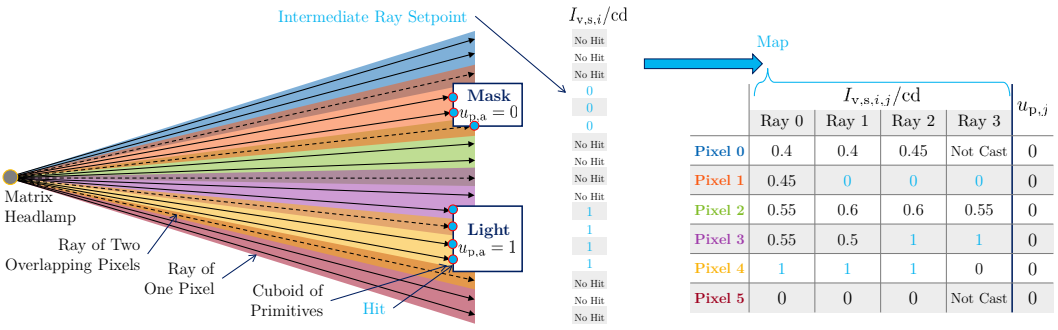



Figure 7.17.: SSC ray tracing and mapping example following Fig. 7.14. SSC tests the cast rays against the primitives and maps the resulting setpoints to $\boldsymbol{i}_{\mathrm{v,p,s},j}$ with $a_\mathrm{m,ssc} = 1$ in (7.4.1).

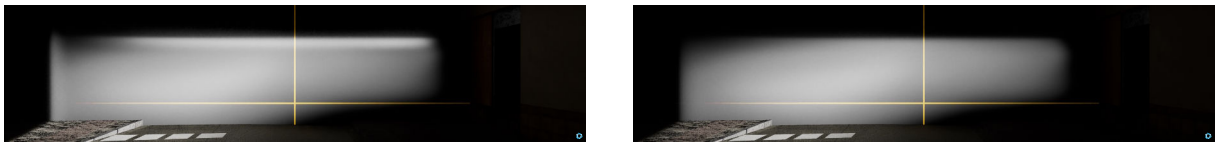
Post-mapping, $I_{\mathrm{v,s},i,j}$ values undergo constraint application to ensure compliance with lower luminous intensity bounds $I_\mathrm{v,lb} \in \mathbb{R}_{\geq 0}$ and upper bounds $I_\mathrm{v,lb} \leq I_{\mathrm{v,s},i,j} \leq I_\mathrm{v,ub}$, which

accommodate thermal and material limitations specified by the user. Additionally, $I_{\mathrm{v,s},i,j}$ is constrained not to extend the maximum pixel luminous intensity $I_{\mathrm{v,m},i,j}$ in the specified direction as $0 \leq \frac{I_{\mathrm{v,s},i,j}}{I_{\mathrm{v,m},i,j}} \leq 1$.

During digital beam movement (see Sec. 7.3.2) or primitive placement, peripheral regions of the headlamp's possible illumination area, which primarily comprises low-intensity stray light, may receive $I_{\mathrm{v}}$ setpoints exceeding their maximum capabilities. For example, digital beam movement shifts central regions with high $I_{\mathrm{v}}$ to the sides, which can result in unintended bright bands at illumination edges, as seen in Fig. 7.18a. While $I_{\mathrm{v,s},i,j}$ remains capped at $I_{\mathrm{v,m},i,j}$, the regions with peripheral low possible $I_{\mathrm{v}}$ receive utilization setpoints near one, which can lead to unintended increased illumination in these regions, resulting in visible bright bands.
To mitigate this effect, $I_{\mathrm{v,s},i,j}$ values can undergo edge darkening after the mapping, controlled by a user-defined parameter $a_{\mathrm{d,ssc}} \in \mathbb{R}_{>0}$. The darkening operation of $I_{\mathrm{v,s},i,j}$, based on the ratio of the cumulative luminous intensity $I_{\mathrm{v,m},i,\mathrm{sum}}$ of all pixel illuminating in the ray direction of the $i$-th texel to the mean maximum possible luminous intensity $I_{\mathrm{v,av}} \in \mathbb{R}_{\geq 0}$ of the headlamp illumination area. When the maximal possible $I_{\mathrm{v}}$ in a direction is below the mean, the setpoints are reduced as

$$I_{\mathrm{v,s},i,j} \min\left\{\frac{I_{\mathrm{v,m},i,\mathrm{sum}}}{I_{\mathrm{v,av}}\, a_{\mathrm{d,ssc}}}, 1\right\}. \tag{7.4.2}$$

This darkening function maintains independence from ray to headlight photometric center distance using $I_{\mathrm{v,av}}$, ensuring applicability across diverse beam patterns without adjustments. Fig. 7.18 shows the effectiveness of edge darkening for digital movement applications, evidenced by the elimination of bright bands. The images in Fig. 7.12 are captured with active edge darkening enabled.



(a) No darkening. (b) With darkening.

Figure 7.18.: Influence of edge darkening on the digital movement of the base illumination.

### 7.4.2. Reduction

The reduction phase of SSC combines all mapped setpoints of a pixel to calculate its final utilization and handles anti-aliasing operations [WB21c]. The reduction is done in parallel for all pixels to achieve real-time capability and concludes SSC. Afterward, the final pixel utilization values are transferred to the matrix headlight to illuminate the road.

As elucidated by Baumann et al. [Bau+21], selective exclusion of certain $I_{\mathrm{v,s},i,j}$ values from the utilization calculations may be necessary to achieve the desired illumination. This filtering process allows the enhancement of edge definition in the LID by suppressing parasitic stray light contributions. SSC changes the weighting coefficients in $\boldsymbol{w}_j$ to selectively ignore setpoints, whereby excluded values are assigned zero weighting before the reduction operation.

The exclusion of $I_\mathrm{v}$ setpoints uses a pixel adaptive $I_\mathrm{v}$ threshold to exclude $I_{\mathrm{v,s},i,j}$ values that fall below a fraction of the maximum luminous intensity $I_{\mathrm{v,m},j} \in \mathbb{R}_{\geq 0}$ for the $j$-th pixel. This approach enhances granularity compared to Baumann et al. [Bau+21] by enabling selective exclusion of individual setpoints rather than being constrained to entire pixel-level exclusions. The weight-adjusting operation uses a normalized intensity threshold parameter $a_\mathrm{x,ssc} \in \mathbb{R}_{>0 \wedge \leq 1}$ that determines the cutoff value to set the weighting to zero as

$$w_{i,j} = 0 \mid 0 \leq i < n_{\mathrm{t},j} \wedge I_{\mathrm{v,s},i,j} \leq a_\mathrm{x,ssc}\, I_{\mathrm{v,m},j}. \tag{7.4.3}$$

The reduction of mapped setpoints can be realized using various functions like an arithmetic mean, which calculates the pixel load as the average of all mapped $I_\mathrm{v}$ setpoints. The selection of the reduction function depends on the user's subjective preferences. The reduction process determines the pixel utilization $u_{\mathrm{p},j}$ for the $j$-th pixel by solving a system of weighted linear equations to minimize the deviation between the mapped $I_\mathrm{v}$ setpoints and the pixellight illumination.
The diagonal weight matrix $\boldsymbol{W} \in \mathbb{R}_{\geq 0}^{n_{\mathrm{t},j} \times n_{\mathrm{t},j}}$ [IM11] of setpoint weights $w_{i,j}$ is $\boldsymbol{W} = \mathrm{diag}(w_{0,j}, w_{1,j}, \ldots, w_{n_{\mathrm{t},j}-1,j})$. The equation system in matrix form is

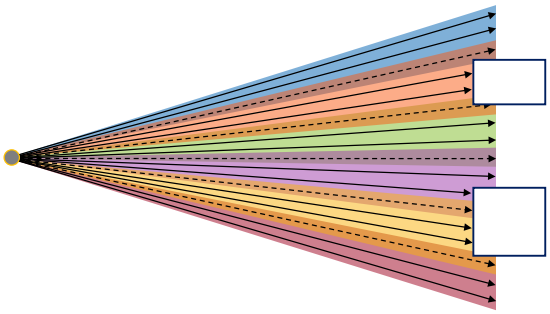
$$\boldsymbol{W} \begin{bmatrix} I_{\mathrm{v,s},0,j} \\ I_{\mathrm{v,s},1,j} \\ \vdots \\ I_{\mathrm{v,s},n_{\mathrm{t},j}-1,j} \end{bmatrix} = \boldsymbol{W} \begin{bmatrix} I_{\mathrm{v,m},r_{\mathrm{s},0,j},j} \\ I_{\mathrm{v,m},r_{\mathrm{s},1,j},j} \\ \vdots \\ I_{\mathrm{v,m},r_{\mathrm{s},n_{\mathrm{t},j}-1,j},j} \end{bmatrix} u_{\mathrm{p},j} \Leftrightarrow \boldsymbol{W}\, \boldsymbol{i}_{\mathrm{v,p,s},j} = \boldsymbol{W}\, \boldsymbol{i}_{\mathrm{v,p},j,\mathrm{cp}}\, u_{\mathrm{p},j}, \tag{7.4.4}$$

where $\boldsymbol{i}_{\mathrm{v,p},j,\mathrm{cp}} \in \mathbb{R}_{\geq 0}^{n_{\mathrm{t},j}}$ is the compressed $\boldsymbol{i}_{\mathrm{v,p},j}$ vector with zeros removed and $r_{\mathrm{s},k,j} \in \mathbb{N}$ denotes the row index corresponding to the $k$-th mapped value of the $j$-th pixel.

The pixel utilization $u_{\mathrm{p},j}$ is computed by a reduction function as the potentiated influence weighted summation of individual pixel utilizations as

$$u_{\mathrm{p},j} = \sum_{i=0}^{n_{\mathrm{t},j}-1} \frac{w_{i,j}\, I_{\mathrm{v,m},r_{\mathrm{t},i,j},j}^{a_\mathrm{r,ssc}}}{\sum_{i_2=0}^{n_{\mathrm{t},j}-1} w_{i_2,j}\, I_{\mathrm{v,m},r_{\mathrm{s},i_2,j},j}^{a_\mathrm{r,ssc}}} \frac{I_{\mathrm{v,s},i,j}}{I_{\mathrm{v,m},r_{\mathrm{t},i,j},j}}. \tag{7.4.5}$$

The controlling factor $a_\mathrm{r,ssc} \in \mathbb{R}_{\geq 0}$ modulates the relative importance of $I_{\mathrm{v,s},i,j}$ values for higher-magnitude $I_{\mathrm{v,m},i,j}$ and their associated pixel utilizations. The selection of $a_\mathrm{r,ssc}$ in (7.4.5) profoundly influences the reduction step's behavior and enables several different operating modes.
When $a_\mathrm{r,ssc} = 0$, the reduction function of (7.4.5) is a weighted arithmetic mean of individual pixel utilizations, independent of the pixel's $I_\mathrm{v}$. The reduction degenerates to a simple arithmetic mean where $w_{i,j}$ maintains uniformity across all $n_{\mathrm{t},j}$ texels.
Setting $a_\mathrm{r,ssc} = 1$ in (7.4.5) normalizes the sum of mapped $I_\mathrm{v}$ setpoints by the aggregate pixel luminous intensities, implicitly assuming a homogeneous distribution of pixel intensity.
When $a_\mathrm{r,ssc} = 2$ and all weights $w_{i,j} = 1$ in (7.4.5), the reduction corresponds to using the Moore-Penrose inverse $\boldsymbol{i}_{\mathrm{v,p},j,\mathrm{cp}}^{+}$ as $\boldsymbol{i}_{\mathrm{v,p},j,\mathrm{cp}} \boldsymbol{i}_{\mathrm{v,p,s},j}$, yielding the optimal least squares solution for the pixel (see Sec. 3.5.2). For non-uniform $w_{i,j}$, setting $a_\mathrm{r,ssc} = 2$ solves (7.4.4) with a weighted least squares solution [IM11]. Fig. 7.19 follows Fig. 7.17 with the reduction of $\boldsymbol{i}_{\mathrm{v,p,s},j}$ to calculate the final pixel utilizations.
Fig. 7.20 illustrates a scenario analogous to Fig. 7.5 but uses an equal division of $I_\mathrm{v}$ setpoints to pixels and a reduction strategy using the Moore-Penrose-Inverse $\boldsymbol{i}_{\mathrm{v,p},j}^{+}$ to



Reduce

| | $I_{\mathrm{v,s},i,j}$/cd | | | | $u_{\mathrm{p},j}$ |
|---|---|---|---|---|---|
| | Ray 0 | Ray 1 | Ray 2 | Ray 3 | |
| Pixel 0 | 0.4 | 0.4 | 0.45 | Not Cast | 0.42 |
| Pixel 1 | 0.45 | 0 | 0 | 0 | 0.11 |
| Pixel 2 | 0.55 | 0.6 | 0.6 | 0.55 | 0.44 |
| Pixel 3 | 0.55 | 0.5 | 1 | 1 | 0.76 |
| Pixel 4 | 1 | 1 | 1 | 0 | 0.75 |
| Pixel 5 | 0 | 0 | 0 | Not Cast | 0 |

No Reduce Operation, as no ray hit

Figure 7.19.: SSC reduction example following Fig. 7.17. SSC reduces all $I_{\mathrm{v,s},i,j}$ of $\boldsymbol{i}_{\mathrm{v,p,s},j}$ with $a_{\mathrm{r,ssc}} = 0$ in (7.4.5) to calculate the final pixel utilization $u_{\mathrm{p},j}$. All $\boldsymbol{w}_j$ values are one.

minimize the squared $I_{\mathrm{v}}$ error. The contrasting approaches depicted in Fig. 7.5 ($a_{\mathrm{m,ssc}} = 1$, $a_{\mathrm{r,ssc}} = 0$) and Fig. 7.20 ($a_{\mathrm{m,ssc}} = 0$, $a_{\mathrm{r,ssc}} = 2$) demonstrate the impact of mapping and reduction parameters on the resulting pixel utilizations as $\boldsymbol{u}_{\mathrm{p}} = [0.47\ 0.64]^{\top}$ and $\boldsymbol{u}_{\mathrm{p}} = [0.41\ 0.5]^{\top}$. The decision to employ a specific mapping and reduction function depends on the desired illumination and the user's subjective preferences.

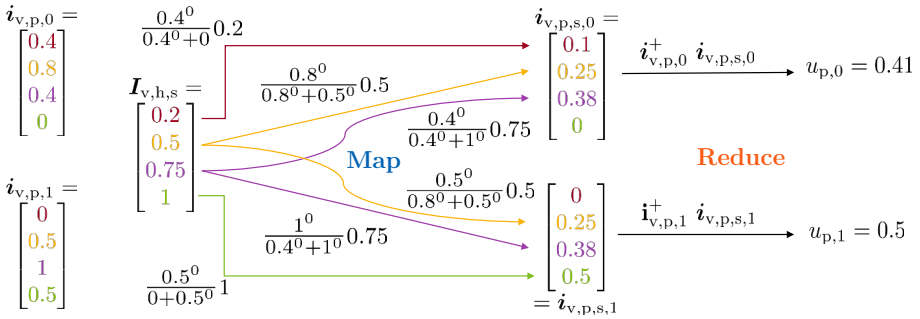



Figure 7.20.: SSC controls two pixels by four $I_{\mathrm{v}}$ setpoints same as Fig. 7.5. $\boldsymbol{i}_{\mathrm{v,h,s}}$ is equally divided to the pixels with $a_{\mathrm{m,ssc}} = 0$. The resulting $\boldsymbol{i}_{\mathrm{v,p,s},j}$ are reduced by the Moore-Penrose-Inverse $\boldsymbol{i}^{+}_{\mathrm{v,p},j}$ with $a_{\mathrm{r,ssc}} = 2$ to obtain the pixel utilizations.

## 7.5. Extensions of the Matrix Headlight Control

SSC provides, with its ray tracing and MapReduce method, a foundation for a wide range of advanced lighting control applications. This section outlines potential extensions for SSC to enhance its capabilities and address emerging challenges in automotive lighting.

### 7.5.1. Ray Tracing Lighting Functions

The ray tracing of SSC (see Sec. 7.3.3) offers a versatile foundation for advanced lighting functions, including dynamic environment-adaptive lighting, cooperative symbol projection and energy-efficient illumination. To realize this function, SSC casts secondary rays from primitive hit points to various targets like observers, supplementary light sources or additional surfaces.

A possible secondary ray cast application involves interaction with other headlamp modules of the ego-vehicle. The second ray is cast toward the module position and evaluated against the module's $\boldsymbol{I}_{\mathrm{v,h}}$ to determine the potential influence of the other module at the hit point.

Knowing the influence enables cooperative symbol projection and beam pattern generation across multiple modules and headlights with smooth transition zones. The ray casting approach eliminates the need for post-processing illumination smoothing filters [Mic14] and provides precise module-specific lighting influence information at each hit point.

The influence of additional modules functions is a negative offset to $I_{\mathrm{v,s},i}$ for the module of the first ray, distributing the desired illumination across multiple pixellights. The module influence on hit points follows the inverse square law depending on Euclidean distance $d$ from the module punctiform origin to the hit point (see Sec. 2.1) as $I_{\mathrm{v,m},i,\mathrm{sum}}/d^2$. An example of the division is that one module illuminates a hit point with three times more significant $E_\mathrm{v}$ than the other. $I_{\mathrm{v,s},i}$ of the darker module is then multiplied by 0.25 and that of the lighter module by 0.75 so that the resulting value at the hit point remains the same due to the superimposition of the multiple lightings.

SSC supports inter-vehicle cooperative symbol projection in principle, potentially offering reduced per-vehicle energy consumption [Dan+21]. This method mirrors the internal module casting process. However, inter-vehicle communication protocols are required to share primitives and headlight model data through vehicle-to-vehicle communication.

Additionally, SSC can account for ambient light sources such as street lighting, addressing the findings of Vogel [Vog23] regarding the impact of combined lighting on object visibility and road safety [Vog23]. The matrix headlight can selectively adjust brightness in street-lit areas with SSC while maintaining overall illumination range for optimal adaption to environmental conditions.

The second ray is cast from the hit point to the streetlight photometric center and the resulting $I_\mathrm{v}$ is used to adjust the $I_{\mathrm{v,s},i}$ of the hit point as a negative offset to the $I_\mathrm{v}$ setpoint. The consideration of streetlights requires a precise detection of the light source's origin and knowledge of its radiation characteristics.

The ray tracing method keeps, in contrast to a dynamic illumination range reduction based on street lighting [Vog23], the overall illumination range constant and only reduces the illumination in the streetlight area.

Future development opportunities include integration with high-fidelity 3D environment models incorporating material properties sourced from either advanced HD mapping or real-time 3D sensor data. Ray tracing allows for the calculation of the influences of the reflections in real-time, which can be used for dynamic environment-adaptive lighting functions.

Potential applications could include dynamic adverse weather adaptation through material-dependent secondary ray casting, particularly for wet surface reflectance and puddle management. As fog reflects light diffusely in multiple directions [Kri24], using secondary ray casts for reflection calculation does not offer a benefit for fog lighting. A standard primitive is sufficient to adjust the headlight brightness.

Another application is an environment-optimized illumination following Potter et al. [Pot+18], modulating output based on surface reflectivity, like reducing the illumination on strongly reflective objects like white house walls and increasing on dark ones. The tracing of reflections could be extended to a multi-bounce lighting analysis through recursive ray tracking.

The secondary ray cast allows shadow analysis similar to shadow rays (see Sec. 3.1) for adaptive lighting in occluded areas. The occlusion analysis by shadow rays allows the detection of blocked lighting paths by light-blocking objects like trees before streetlights and dynamically adjusting the lighting accordingly. For example, when one headlight is blocked, the other headlight can increase its lighting to compensate for the blocked source. Another application to save energy is deactivating symbol projection when the desired symbol is not seen by the observer, which can be done by casting a shadow ray from the hit point to the observer. Fig. 7.21 shows potential dynamic environment-adaptive lighting functions enabled by the SSC ray tracing method.

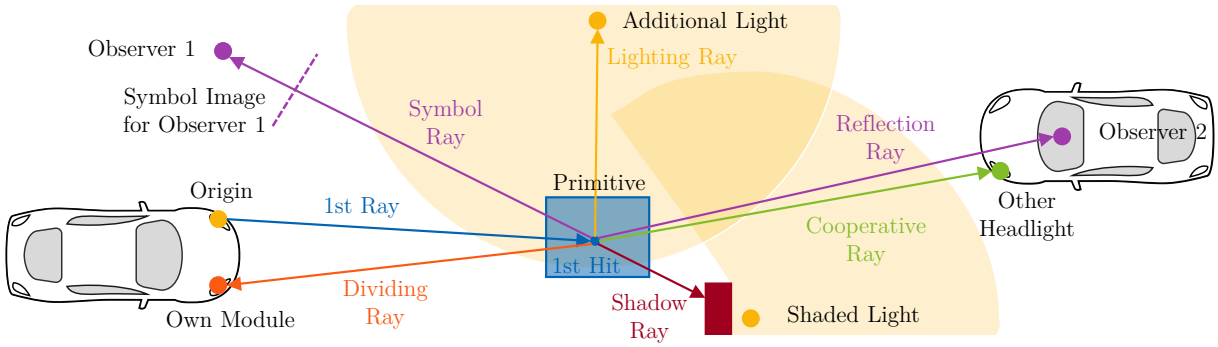



Figure 7.21.: Dynamic environment-adaptive lighting capabilities of SSC by secondary ray casts. The primary ray emanates from the headlight and hits the primitive. The secondary rays analyze the paths from the hit primitive to observers, additional modules, light sources and surfaces.

### 7.5.2. Adaptive Mapping and Reduction

Given that this thesis presents only a singular mapping strategy through (7.4.1) and reduction function via (7.4.5) without comparative analysis of alternative approaches, the selected strategies may not be optimal for all use cases. The evaluation in Sec. 8.4 will show that SSC with the described mapping and reduction functions creates too large illumination areas and produces blurry edges compared to a JPO.

One promising avenue for enhancing illumination areas involves a multi-factorial weighting scheme in the mapping phase. The modified weighting scheme considers pixel influence and granularity as the inverse relationship to pixel size as a factor, preferentially assigning higher $I_\mathrm{v}$ setpoints to smaller pixels due to their superior adaptability to setpoint LIDs. The reduction phase could also be improved by using a perceptual pixel activation threshold $a_\mathrm{ac,ssc} \in \mathbb{R}_{\geq 0 \wedge \leq 1}$ of the number of non-zero setpoints with $I_{\mathrm{v,s},i,j} > 0\,\mathrm{cd}$ required to activate a pixel. When the setpoints of $\boldsymbol{i}_{\mathrm{v,p,s},j}$ contain predominantly zero elements below the $a_\mathrm{ac,ssc}$ relative value, the pixel is deactivated to avoid unnecessary illuminations of large areas. To improve the illumination edges, the mapping algorithm could incorporate contextual information from surrounding setpoints, necessitating the generation of intermediate setpoint LIDs similar to the state-of-the-art in Sec. 3.5.2. To enhance the sharpness of edges created by SSC, a 2D smoothing convolution operation could be applied to the inverted setpoint LID. The result is masked and added to the original illumination to increase only the edge brightness and compensate for the undershot in the SSC output. However, this approach would entail additional memory requirements.

# 8

# Evaluation of the Matrix Headlight Control

The evaluation of SSC uses efficiency and accuracy criteria similar to the simulation evaluation in Ch. 6. This chapter is based primarily on the own publications [WB22; Wal+23b; Wal+24a] and evaluates SSC for real-time matrix headlight control.

To the author's knowledge, there is no generally recognized method for evaluating matrix headlight control algorithms independent of the matrix headlight design in the current scientific literature. Rüddenklau [Rüd22] uses the RMSE of the controlled headlamp LID to the reference LID as an accuracy criterion, but this method is not headlight independent. The Headlight Safety Performance Rating (HSPR) [GTB23; Erk+22; FR22] is a headlight rating method that considers Low Beam, High Beam and GFHB with multiple oncoming vehicle distances. The HSPR combines the multiple ratings of the individual scenarios weighted into a single final rating.

This thesis's novel matrix headlight control evaluation method is inspired by the HSPR. It uses objective computational performance, memory consumption evaluation and subjective and objectified illumination accuracy assessment. A thoroughly objectified assessment of illumination quality is challenging due to the subjective nature of human visual perception [Kle03; Kie12; Goc13; Wal+22c].
Additionally, the accuracy of a matrix headlight control algorithm exhibits dependencies on the correlation between the illumination capabilities of the matrix headlight and the specified illumination setpoints. An example of this relationship is the projection quality of the edges of symbols. When the illumination setpoints' geometric boundaries match the matrix headlight's pixel grid, a higher accuracy is automatically achieved than in scenarios with miss-matches, which require anti-aliasing compensation for misaligned boundaries.
The quality evaluation uses various LID setpoints and two real matrix headlights: the 84-pixel HD84 matrix headlight [Kal+16; Rüd22] and 16,384-pixel SSL|HD matrix headlight [Por22; Kle+22]. The qualitative and quantitative accuracy of SSC depends on the subjective proper adjustment of the chosen mapping and reduction function.

## 8.1. Computational Efficiency Evaluation

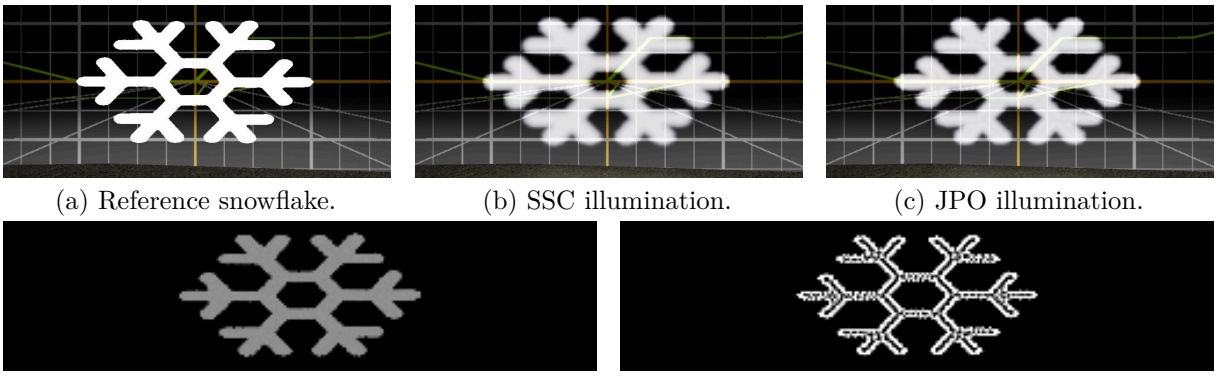
The SSC implementation uses a GPU for parallel computing to be real-time capable. However, the implementation maintains substantial optimization potential in its computational processes, data handling mechanisms and storage architectures.

For instance, global memory coalescing access patterns are not considered. The ray tracing performs hit tests for all bounding volumes since RTHC does not use a nested bounding volume hierarchy [Wal+07; Mei+21]. The optimizations to reduce computing time being implemented are the utilization of shared memory within thread blocks to minimize global memory access latency, early thread termination for rays that fail to intersect with a primitive to skip the mapping and conditional execution of the pixel setpoint reductions, which are only triggered if requested by the previous mapping phase. Therefore, the current memory consumption and computational performance should not represent the algorithm's ultimate capabilities.

The computational times of SSC, analogous to the simulation methods, depend on multiple variables like headlight configuration, activated lighting functions and the number of primitives. Computational performance analyses are documented in Sec. B.3. SSC is capable of real-time control of a HD headlight comprising $1.05 \cdot 10^6$ pixels, achieving a median computation time of 1.79 ms for the evaluation scenario seen in Fig. 7.6a, 7.6b and 7.7, on a Nvidia GTX 2080 Ti GPU. The JPO by Rüddenklau et al. [Rüd+22] of 6,000 pixels takes 30 s.
SSC can generate complex LIDs while offering flexibility for designing and implementing advanced lighting functions. As shown in Sec. B.3, SSC has computation times below 5 ms for over one million pixels, which is sufficient for real-time control of matrix headlights and subjective smooth dynamic lighting transitions. In contrast to the JPO, SSC can be used for fast-changing scenarios like a manual drive of the ego-vehicle or projecting of dynamic symbols. A dynamic SSC use case of smooth symbol projection stabilization is presented in Sec. 9.4, which is not possible with a JPO because of the long computation time.

## 8.2. Subjective Illumination Quality Evaluation

The subjective quality evaluation incorporates human observation and qualitative assessment of the resultant illuminations similar to the simulation evaluation in Sec. 6. Fig. 8.1 presents a comparative side-by-side view of a snowflake projected by a matrix headlight with $64 \times 256 = 16{,}384$ pixellights with blurred illuminations similar to the SSL|HD matrix headlight. The LIDs are generated using both MRPC of SSC and a single non-linear JPO (see Sec. 3.5.2), with corresponding utilization matrices displayed for analysis of the control strategy. The JPO has $n_\mathrm{p}$ DoF and controls all individual pixel utilizations independently by minimizing an error metric between target and actual illumination, not in real-time. In contrast, SSC uses MRPC to control the pixel utilization in real-time.
The visual differentiation of the resulting headlamp illuminations in Fig. 8.1b and 8.1c can be challenging due to the snowflakes' subtle nature of edge sharpness variations. The projected snowflake symbol with MRPC of SSC subjectively appears blurry with reduced edge sharpness compared to the results of the JPO for the whole headlight. However, examining the utilization matrices in Fig. 8.1d and 8.1e shows apparent differences in algorithmic behavior. JPO results in selective pixel activation patterns, concentrating utilization along symbol boundaries while deactivating interior pixels to enhance edge definition. MRPC generates with the mapping algorithm of (7.4.1) relatively homogeneous utilization matrices, lacking the capability to modulate pixel utilization near edges for



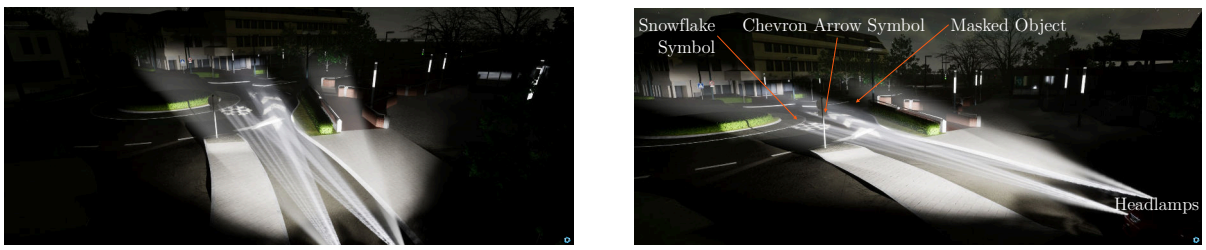
(a) Reference snowflake. (b) SSC illumination. (c) JPO illumination.

(d) SSC grayscale pixel utilization matrix. (e) JPO grayscale pixel utilization matrix.

Figure 8.1.: Projection of a snowflake with MRPC of SSC and JPO on a wall and the resulting pixel utilization matrices.

enhanced symbol definition selectively. When the energy consumption is directly proportional to pixel utilization, the JPO has a marginal efficiency advantage of 2.04 % lower energy consumption than SSC.
In many applications for HD matrix headlights like the SSL|HD, the difference in illumination between SSC and JPO is subjectively tricky for the human vision to detect. The quality difference is subjectively noticeable for low-resolution matrix headlights like the HD84. The difference generally becomes less noticeable as the number of pixels increases, as the illumination areas of the pixellights decrease (e.g., in Sec. 8.4). Therefore, small local non-optimal pixel brightnesses become subjectively less visible to the human vision.

Fig. 8.2 shows a subjective evaluation of beam propagation characteristics and lighting function flexibility of SSC, incorporating projected symbols like the snowflake and object masking. All figures of this thesis show that SSC can be used for a wide range of applications and dynamic lighting functions.




(a) View from above. (b) Side view.

Figure 8.2.: Volumetric fog simulation of beam propagation for SSC-controlled projection of snowflake symbol (left), chevron directional indicators (central) and masked object (right).

## 8.3. Definition of Objectified Illumination Quality Criteria

The objectified quantitative assessment of illumination quality has two critical aspects. The first one is the establishment of appropriate quality metrics that accurately reflect human and machine vision. The second is the determination of scientifically valid thresholds correlating with human or machine visual capabilities for deciding whether the user requirements are met [Vog23].

To the author's knowledge, there is no generally recognized objective quality criterion for headlight control algorithms in the literature. Thus, an objectified illumination quality criterion is defined for the evaluation.

A quantitative evaluation compares the controller output illumination $\boldsymbol{I}_{\mathrm{v,h}}$ with the desired illumination $\boldsymbol{I}_{\mathrm{v,h,s}}$. This comparison is performed across all texels, where the deviation between the setpoint and actual illumination is calculated. The single quality criterion is derived from a statistical aggregation of these deviations. A pixel should consists of multiple texels for precise illumination control with smooth transitions.
An arithmetic mean-based approach of all differences is selected over median-based methods to ensure the incorporation of outlier effects that could affect the visual perception of the entire illumination.

Selecting a normalization factor for the average calculations presents several considerations. The normalization by the total number of texels $n_{\mathrm{t}}$ of $\boldsymbol{I}_{\mathrm{v,h,s}}$ exhibits a limitation wherein the error becomes directly proportional to the illumination setpoint area when the controller maintains consistent absolute errors across all target illuminated texels ($I_{\mathrm{v,s},i} > 0\,\mathrm{cd}$). As an example, when a pixel controller has an error of 0 cd for texels with $I_{\mathrm{v,s},i} = 0\,\mathrm{cd}$ and fixed 10 cd for texels with $I_{\mathrm{v,s},i} > 0\,\mathrm{cd}$, the average error is $\frac{n_{\mathrm{t,gz,s}}}{n_{\mathrm{t}}} 10\,\mathrm{cd}$, which is proportional to the number of positive illumination setpoints $n_{\mathrm{t,gz,s}} \in \mathbb{N}$. This proportionality makes it challenging to compare different evaluation scenarios because more miniature setpoint LIDs will result in more minor average errors.
An alternative normalization using the number of currently illuminated texels $n_{\mathrm{t,gz},k} \in \mathbb{N}$ addresses the area dependency issue but introduces a potential optimization artifact where the error can be artificially reduced through the illumination of additional texels at low-intensity levels. With a $n_{\mathrm{t,gz},k}$ normalization, an optimizer of pixel utilization will minimally illuminate areas, which should be dark, to reduce the average error.
Another approach involves normalization by the number of texels with positive illumination setpoints $n_{\mathrm{t,gz,s}} \in \mathbb{N}$. This method overemphasizes regions where $I_{\mathrm{v,s},i} = 0\,\mathrm{cd}$, which makes the error unintuitive to interpret and misleading, as it is not the average of illuminated texels. However, the practical impact may be minimal due to the generally low illumination levels outside the setpoint regions. After a subjective consideration of these trade-offs, $n_{\mathrm{t,gz,s}}$ is selected as the normalization factor.

For the quantitative assessment of illumination quality, this thesis uses four primary metrics, which are MAE, RMSE, Mean Absolute Relative Error (MARE) and SSIM. Both the MAE and RMSE are normalized relative to the maximum luminous intensity across all setpoints $I_{\mathrm{v,m,s}} \in \mathbb{R}_{\geq 0}$.
The MAE $e_{\mathrm{mae}} \in \mathbb{R}_{\geq 0}$ between the illumination setpoint $I_{\mathrm{v,s},i}$ and the actual illumination $I_{\mathrm{v},i}$ for the $i$-th texel across $n_{\mathrm{t,gz,s}}$ setpoints is

$$e_{\mathrm{mae}} = \frac{1}{n_{\mathrm{t,gz,s}}\, I_{\mathrm{v,m,s}}} \sum_{i=0}^{n_{\mathrm{t}}-1} |I_{\mathrm{v},i} - I_{\mathrm{v,s},i}|. \tag{8.3.1}$$

The Peak Signal-to-Noise Ratio (PSNR) is a standard image quality criterion and is the logarithmic ratio of the maximum possible value $I_{\mathrm{v,m}}$ to the RMSE [HZ10]. The maximization of PSNR for constant $I_{\mathrm{v,m}}$ is equivalent to minimizing the RMSE. As $I_{\mathrm{v,m}}$ is

defined by the currently used headlight and is therefore constant for comparing control algorithms, the RMSE $e_{\mathrm{rmse}} \in \mathbb{R}_{\geq 0}$ is used as the error metric, which is

$$e_{\mathrm{rmse}} = \sqrt{\frac{1}{n_{\mathrm{t,gz,s}}\, I_{\mathrm{v,m,s}}^2} \sum_{i=0}^{n_{\mathrm{t}}-1} \left(I_{\mathrm{v},i} - I_{\mathrm{v,s},i}\right)^2}. \tag{8.3.2}$$

The MARE, analogous to the Weber contrast [RA00], has a definition gap for $I_{\mathrm{v,s},i} = 0\,\mathrm{cd}$ and goes to infinity as $I_{\mathrm{v},i} \to 0$. To address these singularities, the MARE $e_{\mathrm{mare}} \in \mathbb{R}_{\geq 0}$ is formulated with an infinitesimal (i.e., $\ll 10^{-3}$) positive offset $\epsilon_{\mathrm{add}} \in \mathbb{R}_{\geq 0}$ as

$$e_{\mathrm{mare}} = \frac{1}{n_{\mathrm{t,gz,s}}} \sum_{i=0}^{n_{\mathrm{t}}-1} \frac{\left|I_{\mathrm{v},i} - I_{\mathrm{v,s},i}\right|}{I_{\mathrm{v,s},i} + \epsilon_{\mathrm{add}}}. \tag{8.3.3}$$

An alternative to $\epsilon_{\mathrm{add}}$ is a maximal error for texels with $I_{\mathrm{v,s},i} = 0\,\mathrm{cd}$. However, then not all pixel utilizations change the error. With a constant divisor, $e_{\mathrm{mare}}$ is similar to $e_{\mathrm{mae}}$.

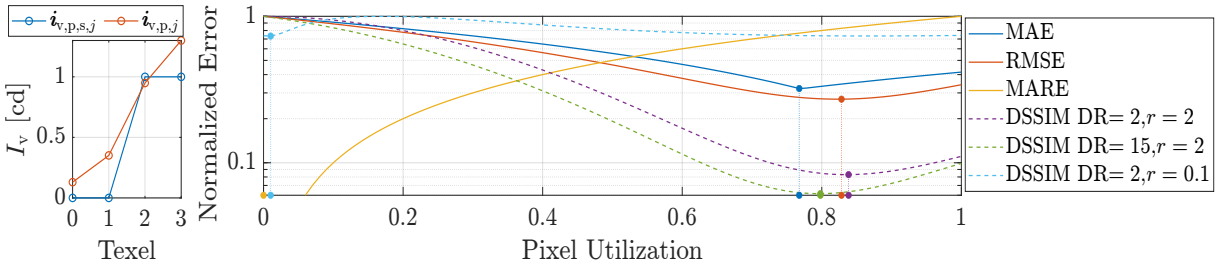
The SSIM considers human visual perception characteristics [Wan+04]. This metric incorporates luminance, contrast and structural similarity [Wan+04; Pal17] and acknowledges the spatial dependencies between adjacent image regions. The SSIM also considers perceptual phenomena such as luminance and contrast masking, which makes disturbances in bright or high-contrast areas less noticeable.
The SSIM depends on the Dynamic Range (DR) of texel values and the analysis window radius. Therefore, the SSIM can not be considered an objective error metric. The DR and window radius are subjectively empirically determined as the headlight's maximum $I_{\mathrm{v}}$ and twice the mean expansion of the target LID.
As the SSIM $e_{\mathrm{ssim}} \in [-1, 1]$ increases with similarity, it is converted to the Structural Dissimilarity (DSSIM) $e_{\mathrm{dssim}} \in \mathbb{R}_{\geq 0 \wedge \leq 1}$ error metric $e_{\mathrm{dssim}} = (1 - e_{\mathrm{ssim}})/2$ [Loz+06].

The various error metrics exhibit distinct sensitivities to illumination deviations, resulting in different optimal solutions when used as optimization objectives.
The MARE is extremely sensitive to errors in dark regions ($I_{\mathrm{v,s},i} = 0\,\mathrm{cd}$), often leading to solutions that altogether avoid illumination outside target areas.
Conversely, the RMSE, due to its quadratic error term, emphasizes larger deviations and typically results in complete illumination of bright target areas, even at the cost of large illuminations in dark regions. The typical frequent occurrence of larger absolute deviations in areas with high setpoints causes this brightening of dark areas.
The SSIM's behavior is parametrically dependent, with smaller window radii producing MARE-like sensitivity and larger radii approximating the RMSE. The behavior of the RMSE and SSIM can be closer than expected [DY11; HZ10].
The MAE represents a balanced compromise between RMSE and MARE, offering linear error sensitivity. Table 8.1 subjectively summarizes the sensitivity of the error metrics to lighting deviations in a simplified subjective manner.

Table 8.1.: Illumination error metric sensitivity (# high, + medium, | low, ∘ no).

| | MAE | RMSE | MARE | SSIM, Small Window | SSIM, Large Window |
|---|---|---|---|---|---|
| Small LID error | + | \| | + | \| | \| |
| Large LID error | + | # | + | # | # |
| LID error at $I_{\mathrm{v,s},i} = 0\,\mathrm{cd}$ | ∘ | ∘ | # | # | + |
| LID error at $I_{\mathrm{v,s},i} \gg 0\,\mathrm{cd}$ | ∘ | ∘ | \| | \| | + |

To give an impression of the sensitivity of the error metrics, the illumination shown in Fig. 8.1b using SSC yields $e_{\text{mae}} = 0.261$, $e_{\text{rmse}} = 0.278$, $e_{\text{dssim}} = 0.016$ and $e_{\text{mare}} = 3.77 \cdot 10^{12}$. The JPO results shown in Fig. 8.1c achieve improved metrics with $e_{\text{mae}} = 0.213$, $e_{\text{rmse}} = 0.238$, $e_{\text{dssim}} = 0.0115$ and $e_{\text{mare}} = 2.42 \cdot 10^{12}$. Both the SSC and JPO optimization procedures use $e_{\text{mae}}$ as their primary cost function.
Fig. 8.3 provides a comparative visualization of error metric behavior for a simplified edge illumination scenario using a blurred pixel. The analysis shows similar optimal utilization levels of 76.7 % for MAE and 82.8 % for RMSE, while MARE favors zero utilization due to its sensitivity to small setpoint values. The DSSIM response varies with parameter selection, with small window radii favoring a minimal 1 % utilization and larger radii suggesting optimal utilization between 79.8 % and 83.8 %, depending on the DR setting.




(a) $\boldsymbol{i}_{\text{v,p,s},j}$ and $\boldsymbol{i}_{\text{v,p},j}$.

(b) MAE, RMSE, MARE and DSSIM progression. Errors are normalized to their maximal values. DSSIM uses window radii of 0.01 and 2 and a DR of 2 cd and 15 cd.

Figure 8.3.: Error criteria behavior for a simplified edge illumination scenario of one pixel. The setpoint LID is $\boldsymbol{i}_{\text{v,p,s},j} = [0\ 0\ 1\ 1]$cd and the pixel is $\boldsymbol{i}_{\text{v,p},j} = [0.13\ 0.35\ 0.95\ 1.3]$cd.

Selecting an appropriate error metric involves subjective considerations regarding the relative importance of different error types.
The MARE is discarded because of its excessive sensitivity to dark region errors and the potential of a complete headlight deactivation as an optimal solution.
The SSIM is excluded due to its parameter sensitivity and non-intuitive behavior, making it difficult to interpret the results.
The choice between MAE and RMSE depends on the relative importance assigned to single significant errors versus distributed more minor errors. When using large, blurry pixels, the MAE might result in complete non-illumination of small bright areas. In contrast, the RMSE might illuminate both the small target area and its large surroundings. Significant errors typically occur in bright target areas less noticeable by the Weber-Fechner law [Fec60] due to Weber contrast masking, so large errors are subjectively not considered more critical than minor errors. Therefore, squaring the errors in the RMSE is unnecessary and the MAE is selected as the primary optimization metric.

## 8.4. Objectified Illumination Quality Evaluation

After the selection of MAE as the quantitative quality criteria, this section presents a comparative analysis between SSC and JPO for matrix headlight control. Assuming the desired LID is defined on flat surfaces, the setpoint distribution can be error-free recreated with PBLD of SSC. On non-flat surfaces, PBLD will introduce an error.
RTHC of SSC transforms the $I_\mathrm{v}$ setpoints on straight lines from 3D world space to 2D $\boldsymbol{I}_\mathrm{v,h,s}$ matrices space, which does not alter the $I_\mathrm{v}$ values. Similar to the headlight model in Sec. 5.2, a changing texel number $n_\mathrm{t}$ of $\boldsymbol{A}_\mathrm{h}$ will introduce an error, but SSC and JPO share the same headlight model and $n_\mathrm{t}$ for this evaluation. Therefore, the influence of the model resolution error on SSC and JPO should be identical.
As PBLD and RTHC introduce no difference or only a negligible one between SSC and JPO, the evaluation focuses on the mapping and reduction functions of MRPC of SSC and its pixel utilization calculation.

The difference in optimization complexity between MRPC of SSC and JPO stems from their DoF. While JPO has $n_\mathrm{p}$ DoF, which are the individual pixels, MRPC has only the three $a_\mathrm{m,ssc}$, $a_\mathrm{x,ssc}$ and $a_\mathrm{r,ssc}$ optimization parameters.
The MRPC parameter optimization uses the gradient-free particle swarm optimization algorithm [KE95; MC11; Ped10; Mat24d] because no assumptions about the behavior of the error function can be made. The optimizer only adjusts the three design parameters of MRPC and MRPC calculates the pixel utilizations.
The high number of DoF of the JPO necessitates a warm-start strategy. Therefore, the optimization process for JPO of all pixel utilizations follows a two-stage approach to reduce computation time. Initially, a constrained linear least-squares solver [Mat24c; Mat24a] determines promising initialization points. As shown in Fig. 8.3, the $e_\mathrm{rmse}$ and $e_\mathrm{mae}$ optima are typically similar, so the RMSE optima are used for the warm-start. Subsequently, the warm-started gradient-free particle swarm MAE optimization is used for $e_\mathrm{mae}$ minimization. The JPO optimizer directly adjusts the pixel utilizations.
Upon termination of the particle swarm optimization for both cases, typically due to iteration constraints, the best candidate solution is further refined using a gradient-based solver for constrained nonlinear multivariable functions [Mat24b; Mat24a]. The particle swarm optimization is limited to 75, while the warm-start is limited to 10 iterations. Empirical samples showed that longer iterations did not lead to different results.
The optimization constraints for MRPC are subjectively chosen parameter bounds as $a_\mathrm{m,ssc} \in [0, 2]$, $a_\mathrm{x,ssc} \in [0, 1]$ and $a_\mathrm{r,ssc} \in [0, 3]$. The JPO constrains all pixel utilizations to $[0, 1]$, ensuring physically realizable solutions.

The experimental evaluation uses two real matrix headlights manufactured by Forvia-Hella GmbH. The HD84 matrix headlight featuring a $3 \times 28 = 84$ LED pixel matrix [Kal+16; Rüd22] (see Fig. 2.7) and the SSL|HD LoRes matrix headlight incorporates a high-resolution $64 \times 256 = 16{,}384$ LED pixel matrix [Por22; Kle+22; For24]. Both systems have similar horizontal beam angles of $\pm 20°$ and vertical angles of $\pm 5°$.
The SSL|HD has reduced pixellight illumination areas with a more rectangular pixel shape than the HD84. Additionally, the SSL|HD has a higher spatial resolution, which allows for

more precise illumination and its maximal possible LID is more homogeneous. However, the SSL|HD has increased overlap between adjacent pixels compared to the HD84.

The evaluation uses 1,215 setpoint LIDs to assess algorithm quality across various operational scenarios. While the setpoint LID parameters allow modifications in the central position, width, height, rotation, brightness, homogeneity and shape, certain DoF are constrained to facilitate meaningful data visualization and analysis. Specifically, the test scenario maintains a homogeneous setpoint of $I_{\mathrm{v,s}} = 10{,}000\,\mathrm{cd}$ over the whole target area, uses either $I_{\mathrm{v,s}}$ or 0 cd as the setpoint for sharp illumination edges and uses rectangular LIDs in spherical coordinates. The setpoint LID is defined relative to the headlight photometric beam axis, incorporating variable horizontal and vertical angular positions, size parameters and rotational orientation. The resulting five DoFs are shown in Fig. 8.4. The same setpoint LID is presented to the MRPC and JPO and their parameters are optimized for minimal $e_{\mathrm{mae}}$.

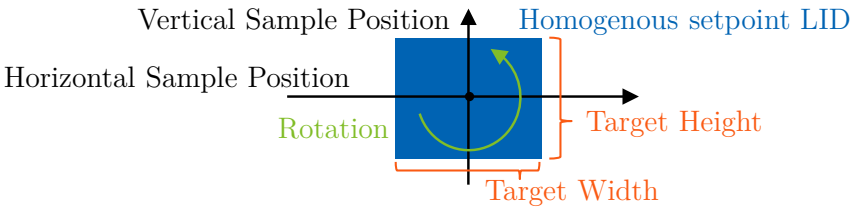



Figure 8.4.: Five DoF of the homogeneous setpoint LID for the evaluation.

The first evaluation examines the correlation between headlight design and illumination quality. Fig. 8.5 and 8.6 present the $e_{\mathrm{mae}}$ distribution resulting from JPO across various setpoint LIDs for both headlights. The color map shows the size and the markers of the rotation of the rectangular setpoint LID.
The HD84 headlight achieves a mean $e_{\mathrm{mae}}$ of 0.81. In contrast, the SSL|HD has a lower mean $e_{\mathrm{mae}}$ of 0.19 caused by its higher resolution. The HD84 exhibits position-dependent error distribution characteristics, particularly evident near the vertical extremities of the illumination field, attributable to the non-uniform pixel geometry across rows. In contrast, the SSL|HD maintains uniform error distribution across its operational range.

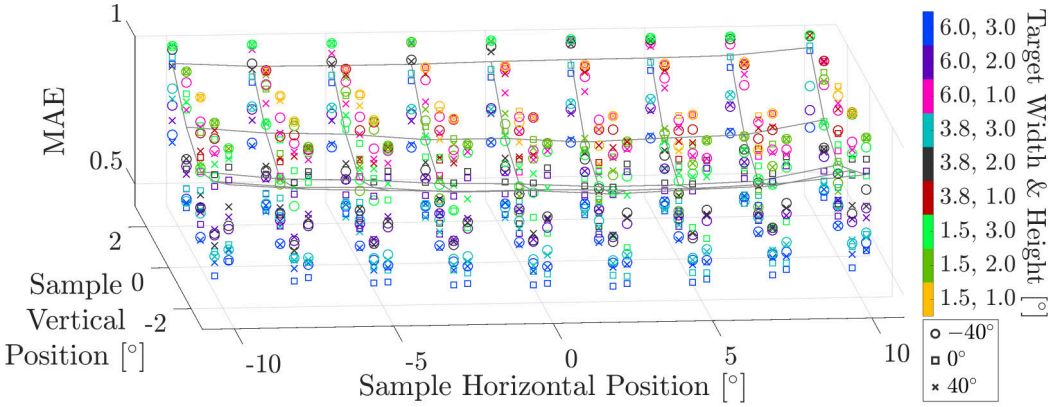



Figure 8.5.: JPO $e_{\mathrm{mae}}$ of the HD84 matrix headlight (mean 0.81, median 0.82). The MAE after (8.3.1) is normalized to $I_{\mathrm{v,s}}$. A gray grid shows the average of all samples at a position.

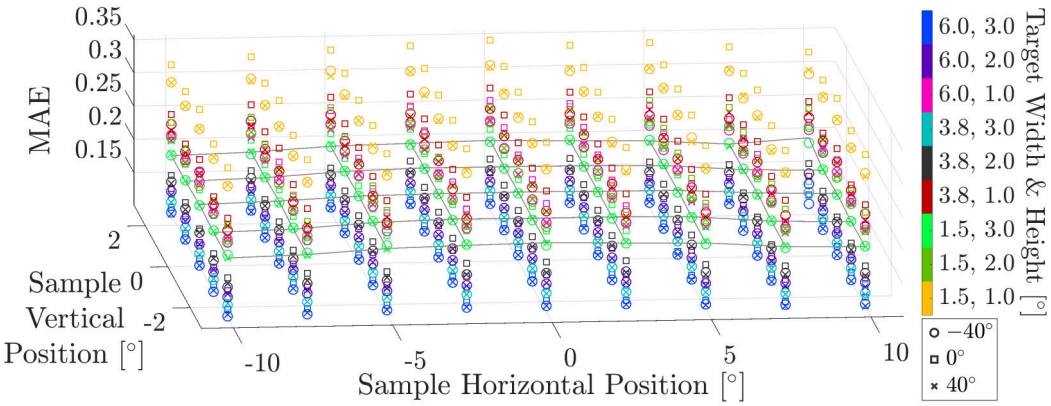



Figure 8.6.: JPO $e_{\text{mae}}$ of the SSL|HD matrix headlight (mean 0.19, median 0.20).

The second evaluation quantifies the illumination difference between the real-time capable MRPC and slow JPO. Fig. 8.7 and 8.8 illustrate the relative ratio as the MRPC $e_{\text{mae}}$ divided by the JPO $e_{\text{mae}}$. The relative ratio becomes independent of the headlight design by dividing the errors.

The error ratio maintains values $\geq 1$ across all tests. Thus, the real-time capable MRPC is never better than a conventional non-real-time JPO. However, the impression of the illumination of MRPC is usually subjectively close to that of JPO.

The HD84 headlight shows an average deterioration of 8.1 % when using MRPC, while SSL|HD shows a more significant loss of 31 %. The HD84 displays asymmetric errors across horizontal angles, potentially attributable to slight asymmetries in the pixel arrangement, with a notably increased relative error at vertical 0°. Conversely, the SSL|HD maintains consistent relative error distribution throughout its operational envelope.

However, the SSL|HD has slightly increasing pixel areas for increasing absolute horizontal angles in Fig. 8.8, which may explain the higher MAE in the outer horizontal regions.

Similar to the HSPR, the following calculation should be an importance-weighted combination of the ratios to give a better impression of the overall performance, but this weighting would be subjective. Therefore, this thesis does not calculate this combination and only presents raw data in Fig. 8.7 and 8.8. A method for objectified selection of samples and importance weighting is subjected to further research.

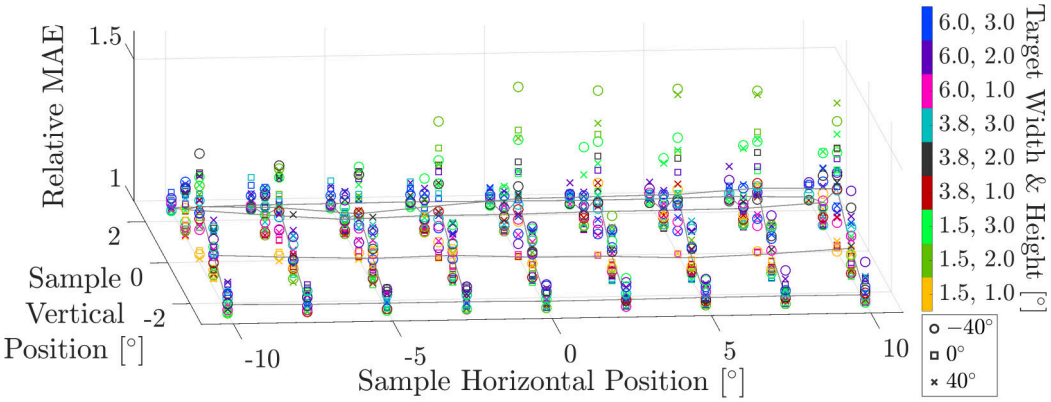



Figure 8.7.: MRPC $e_{\text{mae}}$ divided by JPO of the HD84 headlight (mean 1.081, median 1.045).

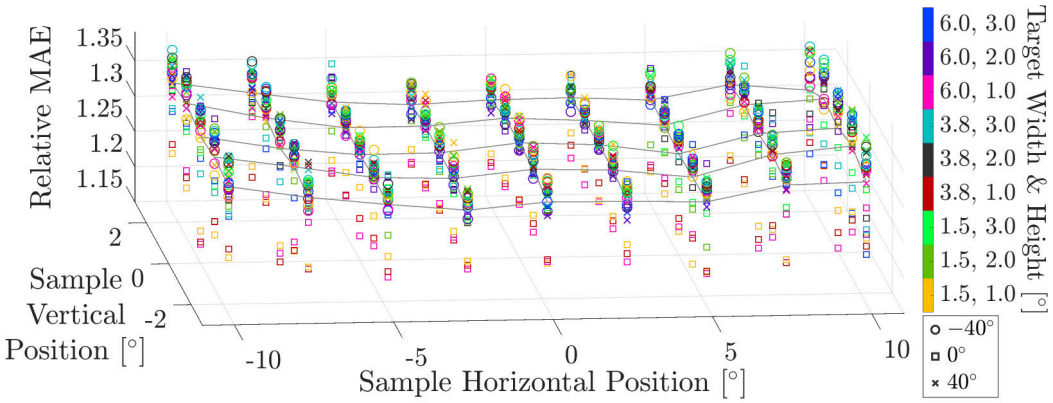



Figure 8.8.: MRPC $e_{\text{mae}}$ relative to JPO of the SSL|HD headlight (mean 1.31, median 1.32).

The comparison of the real-time capable MRPC with a typically non-real-time JPO shows a limitation of MRPC, which manifests in the substantial relative error observed at the horizontal line of the HD84 headlight in Fig. 8.7.

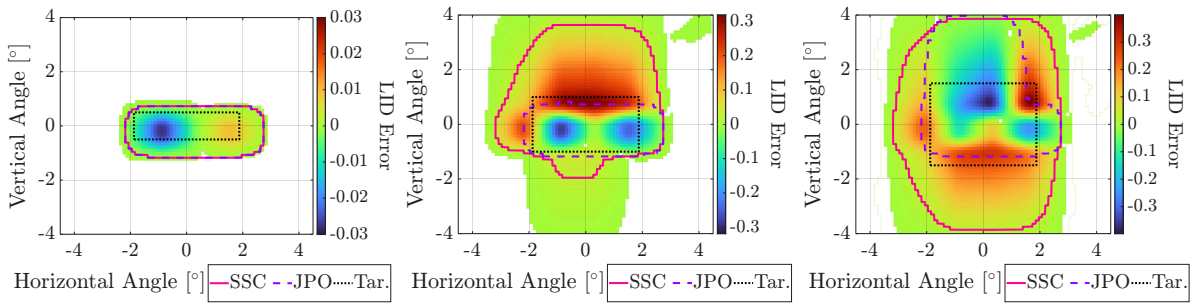
This phenomenon stems from the fundamental architectural approach of how MRPC processes spatial information and considers neighboring pixels. All ray tracing operations and subsequent pixel utilization calculations are executed parallelized independently, with pixel interconnectivity being evaluated locally along individual ray paths rather than globally across neighboring areas and the entire beam pattern. The global spatial relationships are addressed indirectly through specific mapping strategies like (7.4.1) and reductions like (7.4.5). The decoupled controlling of MRPC frequently creates LIDs that demonstrate subjective similarity to JPO results, as evidenced in Fig. 8.1.

However, significant divergences become apparent when examining the different results at the horizontal line of the HD84 headlight shown in Fig. 8.9. For instance, when evaluating a rectangular LID setpoint centered at 0° with horizontal extent of 3.75°, the resulting illumination of MRPC and JPO show strong correlation at 1° vertical extent. However, notable differences emerge at vertical target extents of 2° and 3°, where MRPC activates the larger pixels in the first and third rows of the HD84 headlight with greater frequency than the JPO approach. Fig. 2.7 shows individual pixels of the HD84.

The mapping strategy of MRPC defined by (7.4.1) implements a setpoint distribution across all emitted pixels with $I_{\text{v,m},i,j} > 0$ of a ray, resulting in the activation of any pixel capable of illuminating the target region. JPO achieves a lower $e_{\text{mae}}$ than MRPC by selectively deactivating pixels that predominantly illuminate areas designated for darkness while compensating through increased brightness in remaining pixels. The behavioral difference manifests as a substantial disparity at vertical 0°, as seen in Fig. 8.7.

It is noteworthy that when the JPO algorithm uses the RMSE as its optimization metric and uses the Moore-Penrose-Inverse [Hum+18] from (3.5.2) to calculate the optimal illumination, the resultant LIDs are more similar to the MRPC output, underscoring the significant influence of cost function selection on evaluation results.

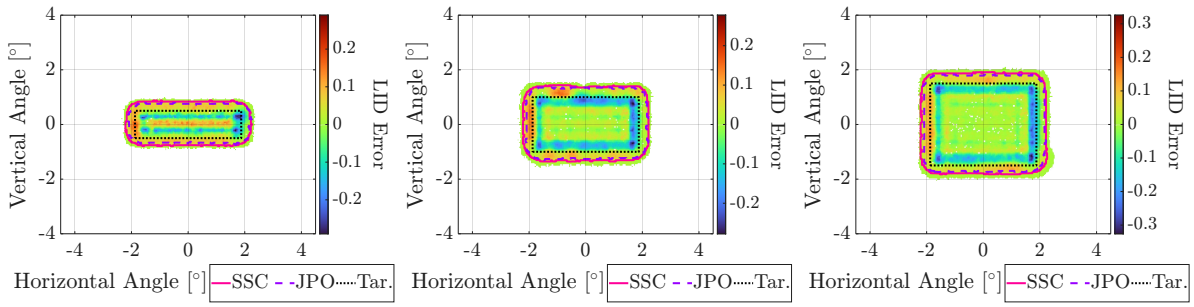
When using the pixel activation threshold $a_{\text{ac,ssc}}$ of non-zero setpoint (see Sec. 7.5.2), $a_{\text{ac,ssc}} = 28\%$ results for the scenario of Fig. 8.9 in a similar illumination shape of MRPC and JPO, but MRPC can not increase the illumination of the pixels in the second row, because the reduction phase does not know the current state of the other pixels.




(a) 1° target height. (b) 2° target height. (c) 3° target height.

Figure 8.9.: Difference between MRPC and JPO LIDs generated by the HD84 matrix headlight. The LID error is normalized to the $I_v$ setpoint. The contours of the resulting MRPC and JPO LIDs are gray and that of the rectangular target lighting is black.

The evaluation of the SSL|HD headlight reveals another characteristic of MRPC, which is also caused by the MapReduce parallel pixel processing. Fig. 8.10 illustrates the differences between MRPC and JPO in edge formation for identical sharp setpoint LIDs.
The uniform setpoint mapping by (7.4.1) of sharp edges results in homogeneous pixel utilization patterns, which produce smooth graduated illumination transitions when applied to pixels with diffuse boundaries like the SSL|HD. In contrast, JPO achieves enhanced edge definition with blurry pixels by a selective brightness modulation, increasing $I_v$ for pixels proximate to edges while reducing $I_v$ from pixels at greater distances.
The pixel utilization matrices with the described characteristic and resultant LIDs are shown in Fig. 8.1 for a target snowflake symbol. In Fig. 8.1, both methods achieve a comparable average pixel utilization and thus a difference in average energy consumption of less than ±1 % of the MRPC to the JPO. A notable distinction exists in peak pixel utilization, where JPO has a maximum pixel utilization four times greater than MRPC.




(a) 1° target height. (b) 2° target height. (c) 3° target height.

Figure 8.10.: Difference between MRPC and JPO LIDs generated by the SSL|HD headlight.

A novel comprehensive analysis of the illumination differences across all regions of the beam pattern is shown in Fig. 8.9 and 8.10. This method aggregates the absolute errors between the MRPC and JPO illuminations for all cases in Fig. 8.7 and 8.8, generating spatially-resolved absolute error distributions. The error naturally decreases outside the region of evaluation sample LIDs, which is bounded by the maximum horizontal LID position of ±10° and maximum width of 6°. The absolute error per texel is divided by the sample number per texel to get a MAE per texel.
The resulting MAE distribution depends on two factors outside the controller's influence, which are the shape and precision of pixellight illuminations in specific angular directions

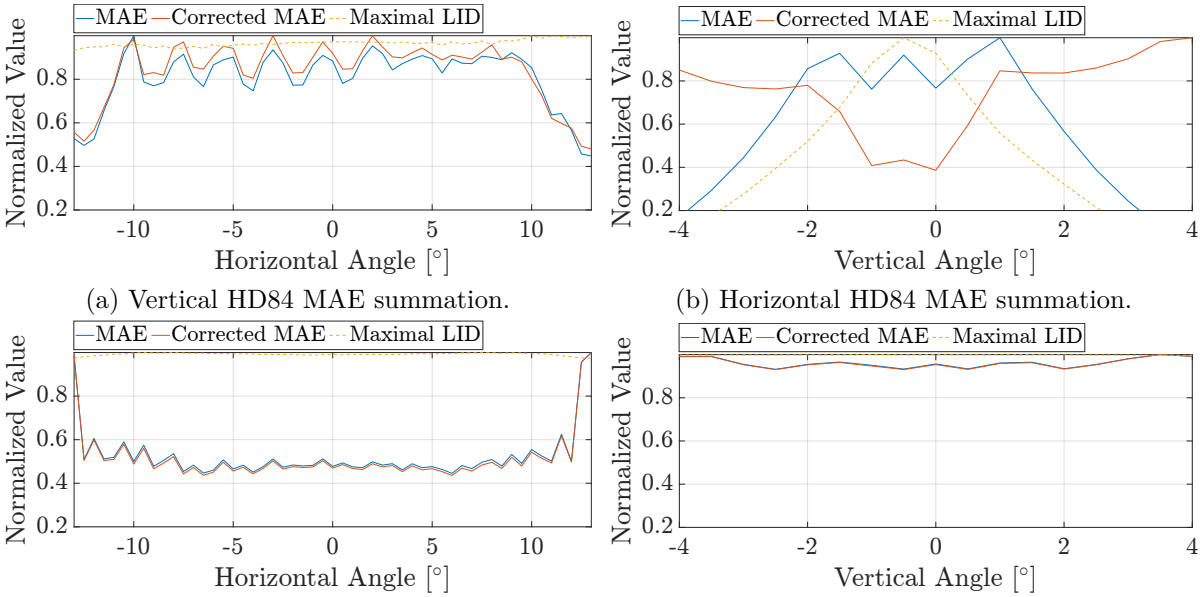
and the homogeneity of the maximal bright LID of the matrix headlight. Identical utilization differences manifest differently in the absolute error depending on the possible pixel brightness. This results in more minor absolute errors for dark pixels and more significant absolute errors for bright pixels. To correct this error bias, the MAE distribution is divided by the maximum possible $I_v$ in each angular direction.

Fig. 8.11 shows a summation of the spatial MAE distribution between MRPC and JPO illuminations on the horizontal and vertical axis for the HD84 and SSL|HD matrix headlights. Additionally, the maximal LID is summed up.

In all scenarios, MRPC never achieves a similar MAE than JPO. Fig. 8.11b reveals systematically higher errors in the regions corresponding to large pixels in the first and third rows of the HD84 matrix compared to the second-row region. This pattern shows the recurring phenomenon of Fig. 8.9 over the whole beam pattern where MRPC unnecessarily activates large pixels.

The SSL|HD headlight has different error characteristics because of the smaller pixel areas and a more uniform beam pattern. Fig. 8.11c and 8.11d show the superior homogeneity of the SSL|HD error distribution compared to the HD84. When comparing Fig. 8.11a and 8.11c, the SSL|HD error has half as large relative MAE fluctuations in the central ±8° range. Additionally, the SSL|HD has increasing pixel areas for increasing absolute horizontal angles, which results in a higher MAE in the outer regions.

A significant error appears at ±13° horizontal angles in Fig. 8.11c for the SSL|HD, where the diverging edge treatment strategies of MRPC and JPO become apparent. The errors originate exclusively from edge effects in these edge regions since edge-related errors are not averaged with illumination errors from the target pattern's interior.




(a) Vertical HD84 MAE summation.

(b) Horizontal HD84 MAE summation.

(c) Vertical SSL|HD MAE summation.

(d) Horizontal SSL|HD MAE summation.

Figure 8.11.: Summation of spatial MRPC and JPO MAE distribution on the horizontal and vertical axis for the HD84 and SSL|HD headlight. All curves are individually normalized to $[0, 1]$.

## 8.5. Evaluation of the Adjustability and Sensitivity[‡†]

The generalized formulation of SSC, especially the MRPC mapping and reduction functions in (7.4.1) and (7.4.5) enable the emulation and enhancement of existing matrix headlight control methods through control parameter optimization. This capability positions SSC as a rapid prototyping tool for matrix headlight control algorithms, enabling the identification of optimal control strategies for diverse operational scenarios [Wal+24a].

The bounding rectangle pixel simplification (see Sec. 3.5.1) approximates each pixel with homogeneous rectangular LID. To recreate this headlight control approach with MRPC, $w_{i,j}$ must be adjusted to unity weights for texels within the rectangular boundary and zero weights for external texels. When a pixel does not illuminate ($I_{\mathrm{v,m},i,j} = 0\,\mathrm{cd}$) a texel inside the bounding rectangle, the headlight model requires modification to ensure that $I_{\mathrm{v,m},i,j} > 0\,\mathrm{cd}$ for all internal pixels, with compensatory $I_\mathrm{v}$ reductions applied to other texels to maintain the total $I_\mathrm{v}$ sum. Homogeneous illumination across the rectangular region can be approximated by setting $a_{\mathrm{r,ssc}} = 1$ in (7.4.5).

The simplified control assumption of non-overlapping pixels cannot be recreated by adjusting $a_{\mathrm{m,ssc}}$ in (7.4.1). This limitation necessitates the implementation of a conditional logic statement that bypasses the mapping operation and directly assigns $I_{\mathrm{v,s},i,j} = I_{\mathrm{v,s},i}$.

A complete maximal influence mapping [Rüd+19b], where pixels with maximum potential luminous intensity $I_{\mathrm{v,m},i} \in \mathbb{R}_{\geq 0}$ on the $i$-th ray receive $I_{\mathrm{v,s},i}$ and the other pixels gets $I_{\mathrm{v,m},i,j}/I_{\mathrm{v,m},i}\, I_{\mathrm{v,s},i}$, also cannot be recreated with (7.4.1). This mapping approach violates the conservation of setpoints since the sum of all $I_{\mathrm{v,s},i,j}$ is not equal to $I_{\mathrm{v,s},i}$.

The rasterization rendering control method (see Sec. 3.5.3) can be recreated by setting $a_{\mathrm{x,ssc}} = 1$. This configuration ensures that only the highest $I_\mathrm{v}$ texel of each pixel contributes to the utilization calculation and deactivates anti-aliasing. When setting $a_{\mathrm{x,ssc}} < 1$, various regular grid SSAA sampling patterns (see Sec. 3.2) can be reproduced by modification of the weighting coefficients $w_{i,j}$ to incorporate the specific sample weights.

The first adjustability and sensitivity evaluation of MRPC followed the scenario of Rüddenklau [Rüd22]. The objective is generating with the 84-pixel HD84 headlight [Kal+16; Rüd22] (see Fig. 2.7) a uniform LID of 10,000 cd with a horizontal angular range from $-1.6°$ to $-10.5°$ and the vertical range from $-4.1°$ to $3.6°$ [Wal+24a]. As Sec. 8 shows, the quality of different control algorithms depends significantly on the specific matrix headlight and the LID setpoint. Rüddenklau [Rüd22] uses only a single scenario, necessitating careful interpretation of the results and allowing only one comparison.

The reference pixel bounding rectangle control method [Rüd22] uses an $I_\mathrm{v}$ threshold at 50 % of $I_{\mathrm{v,m},j}$ without $I_\mathrm{v}$ setpoint mapping. The evaluation uses the identical 84-pixel HD84 headlight and the reference utilizations from Fig. 1-14 of Rüddenklau [Rüd22]. The average illuminance of the generated LID with 96.58 lx closely matched the original study's results of 96.2 lx, which verifies data consistency,

The parameter optimization of MRPC is conducted using a particle swarm optimization [Mat24d] targeting minimal $e_\mathrm{mae}$. Tab. 8.2 shows the results of only optimizing the excluding threshold, using an additional mapping, using arbitrary pixel shape instead of bounding rectangle pixel simplification and optimizing the additional reduction.

For this particular scenario, mapping is recommended to reduce bright spots in the beam pattern. Unfortunately, mapping in combination with a lower $a_{\mathrm{x,ssc}}$ increases the illumination area, so $e_{\mathrm{mare}}$ degrades due to sensitivity to dark areas. The bounding rectangle pixel simplification can be used for this target distribution without significant loss of quality, possibly because its edges are similar to those of the pixels.

Table 8.2.: Comparative analysis of MRPC configurations versus the reference method [Rüd22] using bounding rectangle pixel simplification, no mapping and 50 % exclusion threshold for the HD84 headlight with 84 pixels. Negative values indicate improvement over the reference method.

| Mapping | $a_{\mathrm{m,ssc}}$ | $a_{\mathrm{x,ssc}}$ | $a_{\mathrm{r,ssc}}$ | Shape | $e_{\mathrm{mae}}$ | $e_{\mathrm{rmse}}$ | $e_{\mathrm{dssim}}$ | $e_{\mathrm{mare}}$ |
|---|---|---|---|---|---|---|---|---|
| No | - | 0.99 | 1 | Rectangle | −9.69 % | −5.04 % | −0.52 % | −21.85 % |
| Yes | 1.41 | 0.89 | 1 | Rectangle | −10.60 % | −4.02 % | 5.03 % | −34.80 % |
| Yes | 0.86 | 0.30 | 1 | Arbitrary | −10.74 % | −7.67 % | −5.54 % | 19.38 % |
| Yes | 0.89 | 0.08 | 2.48 | Arbitrary | −11.26 % | −7.35 % | −4.13 % | 10.20 % |

To analyze the sensitivity of MRPC-controlled illuminations with varying control parameters, Fig. 8.12, 8.13, 8.14 and 8.15 show the behavior of $e_{\mathrm{mae}}$, $e_{\mathrm{rmse}}$ and $e_{\mathrm{dssim}}$ over different combinations of $a_{\mathrm{m,ssc}}$, $a_{\mathrm{x,ssc}}$ and $a_{\mathrm{r,ssc}}$ as well as the comparison of bounding rectangle or arbitrary pixel shape [Wal+24a]. The setpoint LID is a 10,000 cd uniform LID within a $8.9° \times 7.7°$ angular region.
Negative exclusion threshold values on the axis indicate bounding rectangle pixel simplifications in the error metric behavior figures. In contrast, positive values denote free-form approaches with $a_{\mathrm{x,ssc}}$ maintained as positive for all cases. Similarly, negative mapping values signify the absence of mapping operations, defaulting to $I_{\mathrm{v,s},i,j} = I_{\mathrm{v,s},i}$. The error metrics are individually normalized to $[0, 1]$, with actual error ranges and minima location as $(a_{\mathrm{m,ssc}}, a_{\mathrm{x,ssc}}, a_{\mathrm{r,ssc}})$ indicated in the figure legends. The minimum locations are projected onto the parameter axis for each error metric.
Optimal values for minimizing $e_{\mathrm{mae}}$, $e_{\mathrm{rmse}}$ and $e_{\mathrm{dssim}}$ exhibit distinct variations across different $a_{\mathrm{r,ssc}}$ values (0, 1 and 2).
In specific scenarios, rectangular pixel shape approximation achieves superior results compared to free-form shape approaches, particularly for $a_{\mathrm{r,ssc}} = 2$ and the $e_{\mathrm{rmse}}$ as shown in Fig. 8.14. This phenomenon can be attributed to the Gaussian curve-like LID of the pixels of the HD84 headlight, where the circular shape patterns after thresholding may create suboptimal inter-pixel spacing compared to bounding rectangle pixel simplifications. The bounding rectangle pixel simplification's superior spatial coverage potentially enhances precision in these cases. However, detailed visualization of this effect is restricted by confidentiality limitations.
Implementing the $I_{\mathrm{v}}$ mapping generally has consistent benefits across all scenarios, as evidenced by elevated error metrics in its absence. However, since the deviation for the other cases is slight, the general advice is complex. Hence, selecting additional parameters and approximation techniques depends on subjective application-specific requirements.

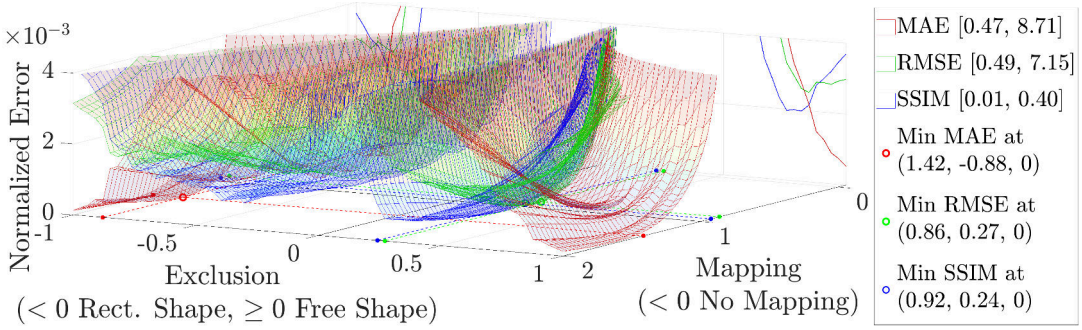



Figure 8.12.: Error metric variations for the HD84 headlight (84 pixels) with $a_{\mathrm{r,ssc}} = 0$.

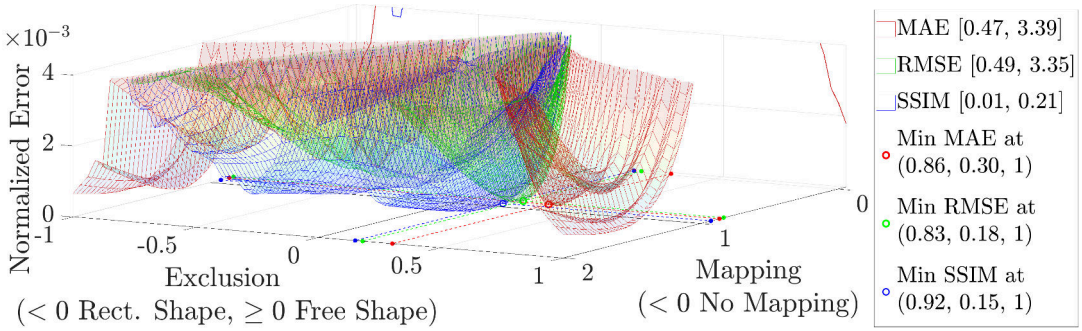



Figure 8.13.: Error metric variations for the HD84 headlight (84 pixels) with $a_{\mathrm{r,ssc}} = 1$.

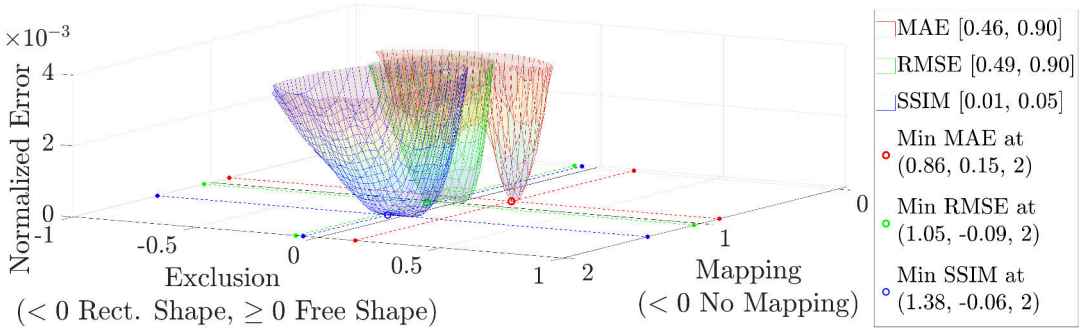



Figure 8.14.: Error metric variations for the HD84 headlight (84 pixels) with $a_{\mathrm{r,ssc}} = 2$.

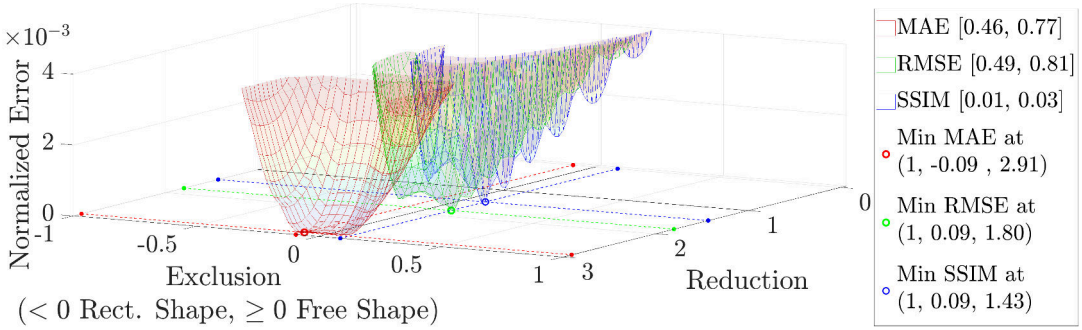



Figure 8.15.: Error metric variations for the HD84 headlight (84 pixels) with $a_{\mathrm{m,ssc}} = 1$.

The evaluation of the SSL|HD 16,384-pixel LoRes module [For24] revealed distinct characteristics compared to the 84-pixel HD84 [Wal+24a]. The SSL|HD has a more precisely defined rectangular pixel geometry, increasing the overlap between adjacent pixels.
With the SSL|HD, the previous HD84 experiments are replicated. The baseline is MRPC with no mapping of setpoints, a 50 % threshold exclusion parameter, $a_{\mathrm{r,ssc}} = 1$ and bounding rectangle pixel simplification. The experimental procedure is similar to Table 8.2 and the quantitative results are presented in Table 8.3.
The SSL|HD illumination quality exhibits robustness to variations in control parameters, with $I_{\mathrm{v}}$ mapping emerging as the sole critical factor for achieving optimal performance metrics. The comparison of rectangular and free-form pixel shapes results in similar performances, which is caused by the optical design of the individual pixels as diffuse rectangular elements.

Table 8.3.: Comprehensive analysis of MRPC against a reference using bounding rectangle pixel simplification, no intensity mapping, 50 % exclusion threshold and $a_{\mathrm{r,ssc}} = 1$ for the 16,384-pixel SSL|HD matrix headlight. Negative values indicate improvement over the reference method.

| Mapping | $a_{\mathrm{m,ssc}}$ | $a_{\mathrm{x,ssc}}$ | $a_{\mathrm{r,ssc}}$ | Shape | $e_{\mathrm{mae}}$ | $e_{\mathrm{rmse}}$ | $e_{\mathrm{dssim}}$ | $e_{\mathrm{mare}}$ |
|---|---|---|---|---|---|---|---|---|
| No | - | 0.99 | 1 | Rectangle | −7.52 % | −5.04 % | −1.74 % | −56.82 % |
| Yes | 2 | 0.04 | 1 | Rectangle | −97.65 % | −95.74 % | −98.49 % | −85.74 % |
| Yes | 1 | 0.97 | 1 | Arbitrary | −97.91 % | −95.77 % | −98.48 % | −90.98 % |
| Yes | 1 | 0.97 | 0 | Arbitrary | −97.93 % | −95.79 % | −98.52 % | −90.91 % |

Similar to the evaluation of the HD84 headlight, the SSL|HD response characteristics to parametric variations in the control algorithm are illustrated in Fig. 8.16, 8.17, 8.18 and 8.19 [Wal+24a]. The setpoint LID is identical to all previous scenarios, with a uniform distribution of 10,000 cd within a $8.9° \times 7.7°$ angular region.
Fig. 8.16, 8.17 and 8.18 reveal a symmetry in the error curves around the zero thresholds for varying $a_{\mathrm{r,ssc}}$ values, indicating that the selection between free-form and bounding rectangle pixel simplification has negligible impact on the error metrics. Furthermore, the variation of $a_{\mathrm{r,ssc}}$ between values of 1 and 2 has minimal influence on the location of global minima in the error space. The observation that error minimization correlates with elevated $a_{\mathrm{x,ssc}}$ values suggests that a rendering control method based on a single central pixel sample could serve as an efficient control paradigm for the SSL|HD. This observation aligns with the design philosophy of HD matrix headlights, which typically employ projection-based optical architectures.
However, as the errors without mapping are above the axis interval, even with this HD system, an $I_{\mathrm{v}}$ setpoint mapping consistently shows significant improvements in lighting quality, which underlines the crucial importance of considering the contributions of the non-primary pixels for all headlight types.

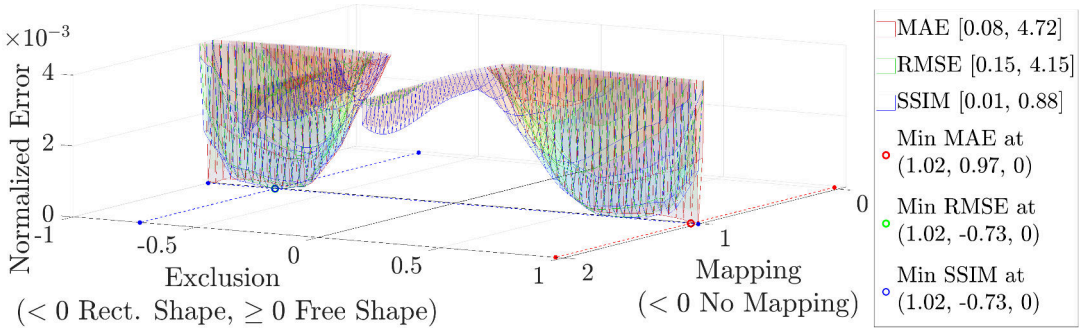



Figure 8.16.: Error metric variations for the SSL|HD headlight (16,384 pixels) with $a_{\mathrm{r,ssc}} = 0$.

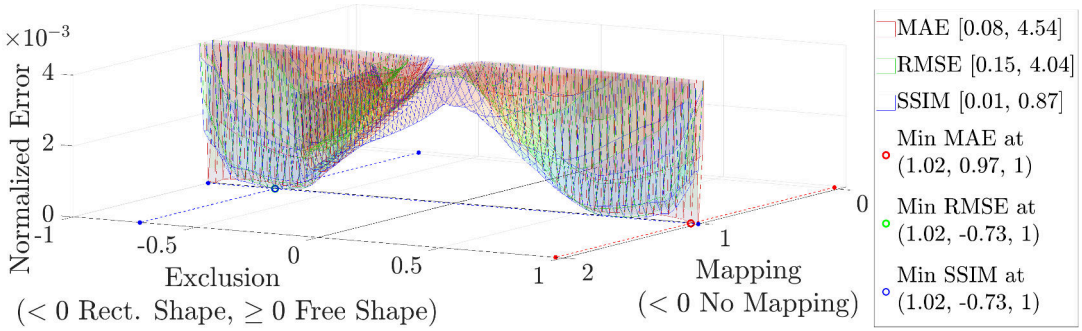



Figure 8.17.: Error metric variations for the SSL|HD headlight (16,384 pixels) with $a_{\mathrm{r,ssc}} = 1$.

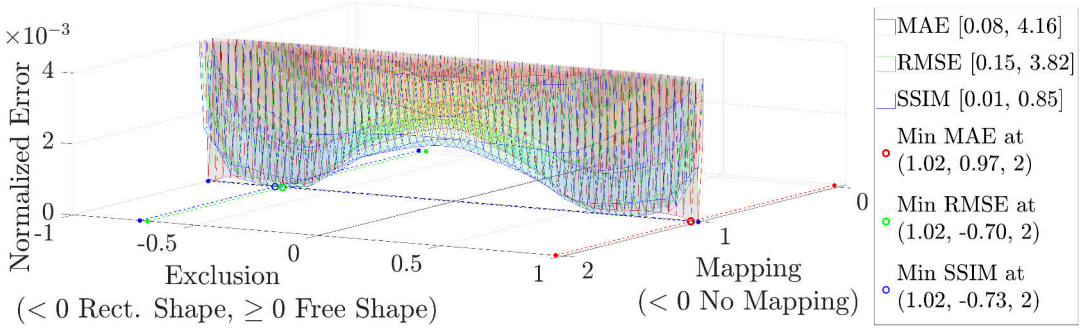



Figure 8.18.: Error metric variations for the SSL|HD headlight (16,384 pixels) with $a_{\mathrm{r,ssc}} = 2$.

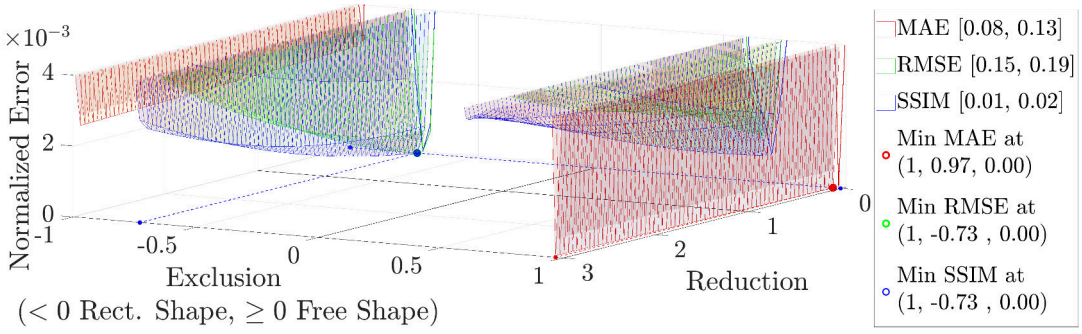



Figure 8.19.: Error metric variations for the SSL|HD headlight (16,384 pixels) with $a_{\mathrm{m,ssc}} = 1.015$.

# 9

# Evaluation of the Usability of the Matrix Headlight Simulator

This chapter evaluates and validates SOL, focusing on its application efficacy and usability for developing novel dynamic lighting functions. While the simulation and SSC have been evaluated in Ch. 6 and 8, this chapter examines the holistic system performance and its practical usability in research scenarios. It is important to note that usability assessment presents inherent challenges due to its subjective nature and task dependency. This chapter is based primarily on the own publications as primary and secondary author [Wal+22b; Kau+22; Wal+23a; Wal+24b; Wal+24c; Kau+23; Kau+24; Wal+25b; Wal+25a] and evaluates the usability of SOL in various research applications, including hybrid rapid prototyping, headlight optimization for human and computer vision, energy-efficient object-based lighting and feedback control of symbol projection.

There exist many other matrix headlight simulators like AVxcelerate Headlamp by Ansys Inc [Ans23; Ben22], Helios respectively Nightdriver by Forvia-Hella GmbH [Ber+03; Rüd22], Hyperion in the Unity engine by University Paderborn [Rüd22], LucidDrive by Synopsys Inc [Syn23], rFpro [Rfp22b], SCANeR Headlights by AVSimulation [AVS22a], simulators in the Unreal engine by University Hannover [Sun24] and by University Darmstadt [Sin+22; Ler+24] and Virtual Test Drive by Hexagon AB [VIR23; ET 21].
Due to the limited public and free availability of technical specifications for the other headlight simulations, a fair and objective scientific comparative analysis is not feasible in this thesis. However, some headlight simulators are compared in the doctoral dissertations of Rüddenklau [Rüd22] and Sundermeier [Sun24]. Therefore, this evaluation chapter emphasizes only empirical validation by showing research applications of SOL, providing quantitative and qualitative evidence of SOL's capabilities for matrix headlight development.

## 9.1. Hybrid Rapid Prototyping of Lighting Functions

Singer et al. [Sin+22] presents a general verification of the Unreal engine for headlight simulation. By using Unreal, SOL can realize dynamic lighting function in simulation and reality for hybrid rapid prototyping. The hardware communication module (see Sec. 4.1) allows pixel utilization input sharing between virtual headlights and physical lighting

systems of real test vehicles like SSL|HD modules. This functionality facilitates three critical objectives, which are the calibration of simulation parameters like the PBR material properties [Bur12; MH16; Mar24], the verification and validation of the simulation and the efficient and fast replication of virtual test scenarios in physical environments for efficient hybrid rapid prototyping, allowing seamless transition from simulation to physical implementation when integrated with test vehicles [Wal+25b; Wal+25a].

To evaluate the hybrid rapid prototyping, SOL is integrated into a field test vehicle from Forvia-Hella GmbH to test the behavior of real matrix headlights in dynamic scenarios virtually and in reality at the city Lippstadt and in the Forvia-Hella light tunnel [Kau+23; Kau+24], which is an over 140 m headlamp test hall [For20; BMW23; Det17]. The headlight configuration of a real test vehicle is recreated in SOL to control the physical real headlights (see Sec. 5.2). Through the communication module, SOL can control six SSL|HD modules in real-time [Kau+23].
Fig. 9.1 shows the hybrid rapid prototyping by lighting tests in a virtual environment of the Lippstadt station square and real light tunnel [Wal+25b; Wal+25a]. The 3D Lippstadt environmental model is created by 3D Mapping Solutions GmbH after precision environment scanning incorporates realistic road conditions, including surface irregularities, curbs, speed reduction elements and varying gradients [3D 24]. Various beam patterns and lighting functions created by SSL|HD headlights in the light tunnel are shown in Sec. B.5 to prove the flexibility of SOL and SSC for creating various lighting functions.

(a) Virtual model of Lippstadt station square.

(b) Real test vehicle.

(c) Real light tunnel.

Figure 9.1.: Similar illuminations in simulation and reality [Wal+25b; Wal+25a].

The alignment of virtual and real headlights and the communication module's input synchronization enables quantitative comparison of illuminations for simulation verification. The illumination comparison allows simulation parameter optimization so that virtual and real environments exhibit comparable characteristics and visual impressions [Mül+24d]. The optimization uses HiL configurations with physical headlights and real-world environments, incorporating either human observational feedback or computer vision algorithmic analysis to adjust the simulation. Adjustable simulation properties encompass PBR material definitions [Bur12; MH16; Mar24], including road surface roughness coefficients and post-processing parameters such as virtual camera characteristics.
Fig. 9.2 presents a qualitative comparison between the physical light tunnel and its simulated counterpart following iterative empirical parameter optimization [Mül+24d; Wal+25b; Wal+25a]. While the current implementation demonstrates high fidelity in main beam characteristics like spatial distribution and intensity falloff, specific limitations exist

regarding the precise modeling of headlight stray light phenomena and SSL|HD pixel color gradients, primarily due to proprietary constraints in accessing complete manufacturer specifications. Additionally, the delineator retroreflectors are not accurately parameterized and simulated. However, as evidenced in Fig. 6.3 and 6.4, SOL has the full capability for color gradient visualization when provided with appropriate input parameters.

(a) Unreal 5.2 simulation of the light tunnel.

(b) Real image of the Forvia-Hella light tunnel.

Figure 9.2.: Comparison of simulated and real illuminations [Mül+24d; Wal+25b; Wal+25a].

## 9.2. Headlight Optimization for Human and Computer Vision

After showing the adjustability and flexibility of SOL, this section focuses on headlight control without using SSC. The data processing (see Ch. 4) allows bidirectional data exchange between SOL and external programs like virtual camera data, traffic object representations, depth information and pixel utilization metrics [Mül+22; Mül+24a].
This interoperability enables complex computational tasks, such as neural network-based object detection, to be executed within optimal development frameworks like Python for neural networks. Therefore, SOL can be efficiently used for research tasks like optimizing dynamic beam patterns for human visual perception and computer vision [Mül+25c; Mül+25b]. The usability of a simulation for beam pattern optimizations relies fundamentally on its capacity to precisely calibrate simulation parameters, ensuring high fidelity between virtual and physical experimental outcomes.

An application domain of SOL lies in optimizing headlamp beam patterns to enhance neural network-based object detection performance. This optimization uses the low-level pixel utilization option of the dual-mode optimization capabilities (see Sec. 4.2).
One approach for quantifying detection performance uses neural network outputs in conjunction with simulation-derived ground truth data, incorporating metrics such as classification accuracy, generalized intersection over union [Rez+19] of predicted bounding boxes and network prediction confidence scores [Mül+23a].

An alternative to network outputs are image-centric metrics [Wal+23a] like statistical variance of colors, DR, Weber contrast computations and gradient analysis for edge definition and spatial homogeneity assessment [Kle03; Wör+07]. When using image-based analysis, it is imperative to apply consistent evaluation criteria to both the camera-captured image and the headlight pattern representation, which is interpreted as a grayscale $I_v$ image. The comparative analysis of the result and input facilitates formulating an optimization cost function that promotes efficient illumination [Wal+23a]. The cost function can also incorporate secondary optimization objectives, including energy consumption minimization, illumination uniformity maximization and glare reduction [Mül+22]. Fig. 9.3 illustrates the optimization feedback loop for neural network-based object detection enhancement.

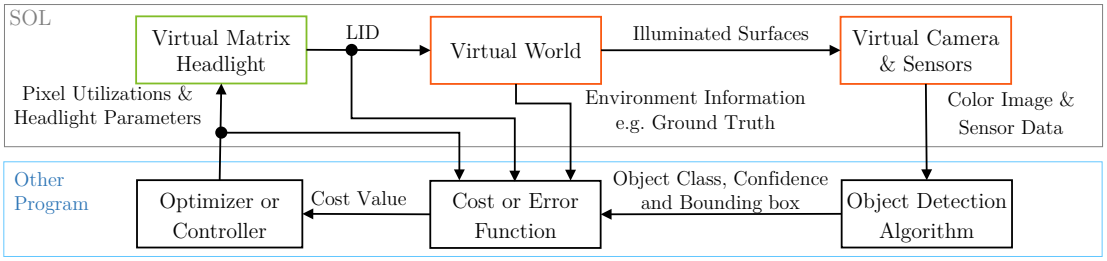
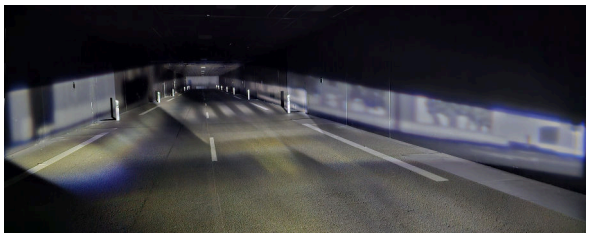


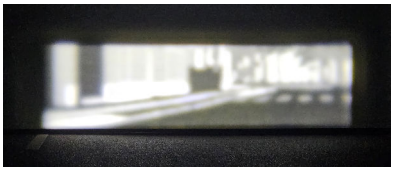

Figure 9.3.: Headlight optimization to enhance neural network object detection. The visualization uses a color scheme consistent with Fig. 4.2.

The material-based matrix headlight control is an approach for achieving energy efficiency and environmental optimization [Mül+23b; Mül+23a]. MBL segments environmental elements and objects, e.g., buildings, into distinct material classifications such as wood, glass, wet asphalt and concrete [Mül+24b]. Each material category receives illumination with a specifically chosen $I_v$, accounting for its unique optical reflection properties [Mül+24e; Mül+24c].
Implementation of MBL within SOL requires the synchronous transmission of environment material data, the camera image and ego-vehicle state parameters (including positional data) to an external headlight control system. This controller computes optimal pixel utilization patterns, which are returned to SOL for visualization and real-world headlight control.
This material-specific approach ensures optimal consideration of surface reflection characteristics, minimizing glare and energy consumption while maximizing object visibility [Mül+24c; Mül+25d]. MBL has superior precision compared to OBL by accounting for multi-material object composition and prioritizing surface reflection properties over object classification. Fig. 9.4 presents an implementation example of MBL applied to the Friedrichschule in Lippstadt, using three SSL|HD headlight modules. Here different materials like the crosswalk stripes are illuminated with adjusted $I_v$ for optimal visibility.

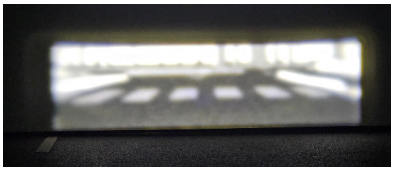
(a) Virtual target environment.



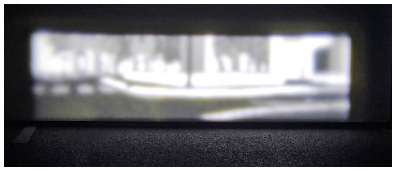
(b) MBL illumination in the light tunnel.



(c) Illumination pattern generated by the left module.



(d) Illumination pattern generated by the central module.



(e) Illumination pattern generated by the right module.

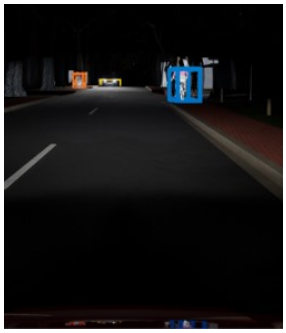
Figure 9.4.: MBL by three SSL|HD headlight modules [Mül+24d; Wal+25b; Wal+25a]. Fig. 9.4a shows the target scene at daytime and Fig. 9.4b shows the superposed MBL illumination in the real light tunnel by three modules. Fig. 9.4c, 9.4d and 9.4e display the real illuminations projected onto a screen by the left, central and right modules.

## 9.3. Energy-Efficient Object-Based Lighting

Using SSC in SOL allows investigations and optimization of various dynamic lighting functions by using the high-level beam pattern definitions via primitives (see Ch. 4). SSC allows investigating OBL methodologies for automated driving [Bar+15] by configuration of the primitives. By adjusting the primitives, SOL enables comparative analysis of diverse OBL strategies, including evaluation of homogeneous versus structured LIDs onto the detected objects [Cla+21] and assessment of GFHB for both vehicle operators and pedestrians.

Fig. 9.5 shows a practical implementation of energy-efficient OBL in a multi-object scenario involving two cyclists and a vehicle. SSC realizes the object-specific high illumination through strategically positioned cuboid primitives generating homogeneous illuminations. These primitives are positioned to avoid direct illumination of human head regions by the matrix headlight, maintaining optimal visibility while minimizing glare effects.

(a) Virtual world.



(b) Cuboids of primitives.

(c) Left grayscale pixel utilization matrix.

(d) Right grayscale pixel utilization matrix.

Figure 9.5.: Energy-efficient OBL by adaptive high illumination of multiple traffic objects. The environment is only slightly illuminated to ensure that objects can be detected [Wal+22b]. The pixel utilization matrices of both matrix headlights are equally grayscaled

A novel OBL function is energy-efficient OBL [Wal+22a; Wal+22b], which reduces the standard ambient illumination by the matrix headlight to the minimum threshold required to maintain reliable object detection with acceptable confidence values, even in scenarios with potential misclassification. If any objects are detected, the system generates high $I_v$ local LIDs corresponding to the bounding boxes of the detected objects and their immediate surroundings, increasing classification accuracy and improving the network's confidence.
A control algorithm incrementally increases object-specific illumination until a plateau or decrease in network confidence is observed for an energy-efficient illumination of objects. The illumination strategy incorporates a spatially-varying LID adapted to object classification confidence levels. This method requires robust object-tracking capabilities to account for dynamic variations in bounding box parameters and object classifications. Fig. 9.6 presents an energy-efficient OBL architecture, illustrating the closed-loop system that dynamically positions primitives coincident with detected objects while continuously optimizing their respective target luminance levels.
Almehio et al. [Alm+21] presents an alternative OBL strategy that uses object class predictions categorizing entities as road users, ground markings, or traffic signs to determine appropriate LIDs within object-specific zones. This classification-dependent OBL typically requires higher baseline illumination levels than the energy-efficient OBL, as it relies on high-confidence class assignments for optimal performance.

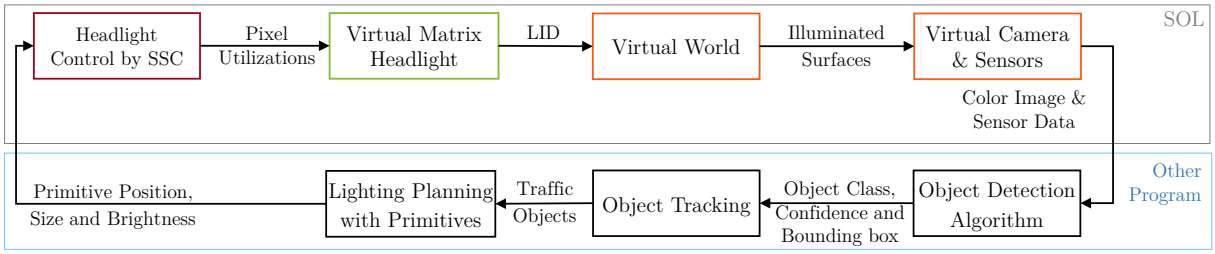
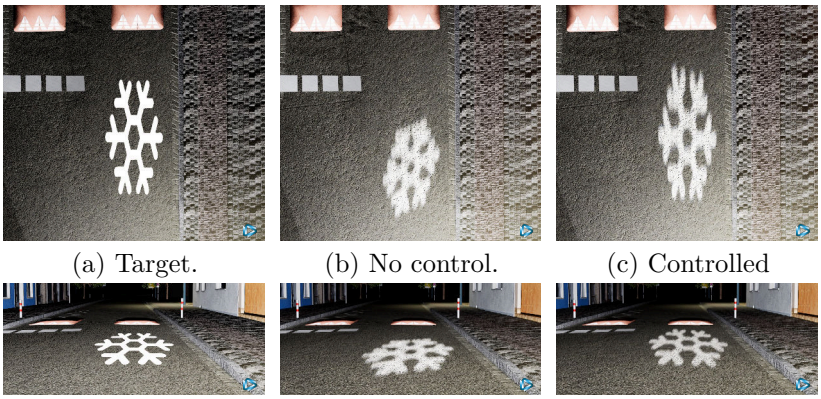



Figure 9.6.: Energy-efficient OBL headlight controller with SSC. The controller optimizes LIDs for detected traffic objects. Color coding corresponds to Fig. 4.2.

## 9.4. Feedback Control of Symbol Projection[‡t]

HD matrix headlights can project symbols onto the road surface, facilitating communication between traffic participants (see Sec. 3.5.2). However, the efficacy of the communication can be compromised when symbols are projected onto non-planar surfaces, resulting in geometric distortions that can impair symbol recognition and interpretation [Wal+24b; Wal+24c]. Fig. 9.7 shows this phenomenon by projecting a snowflake, where surface irregularities induce geometric distortions that can be mitigated through closed-loop control intervention.
To improve the comprehensibility of the symbol projection, an innovative feedback control system is developed that uses SSC and enables continuous, differentiable symbol transformations to compensate for unevenness without requiring a priori knowledge of the road surface topology through 3D height estimation [Wal+24b; Wal+24c].

Existing approaches to symbol projection compensation [Riv+15; Yar23b] necessitate high-fidelity 3D surface models of the approaching road segment for open-loop control.



(a) Target. (b) No control. (c) Controlled

(d) Target. (e) No control. (f) Controlled.

Figure 9.7.: Snowflake projection onto an uneven surface without and with projection control. The upper images are the Birdseye view and the lower is the driver's perspective [Wal+24b].

Yargeldi [Yar23a] uses structured light patterns for real-time surface topology estimation, subsequently combining this information with vehicle kinematic data to predict symbol distortions and implement feedforward compensation by shifting the symbol in the beam pattern against the estimated distortion.

Alternatively, Rivero et al. [Riv+15] and Yargeldi [Yar23a] use computationally expensive inverse ray tracing techniques applied to pre-existing 3D road models to compute optimal projections. A limitation of these methodologies is their reliance on open-loop compensation without direct feedback of the projected symbol, making them susceptible to surface modeling and estimation errors.

The novel feedback symbol controller eliminates the dependency on explicit surface modeling through real-time camera-based feedback of the projected symbols, enabling dynamic adaptation to arbitrary surface geometries [Wal+24b; Wal+24c]. The control error is continuously computed through computer vision analysis of the projected symbol's appearance in the camera image, with corresponding adjustments to the beam pattern through a cascaded control structure. The fundamental innovation is the controller's ability to achieve distortion compensation without explicit surface height estimates, circumventing road surface mapping systems' complexity and potential unreliability.

### 9.4.1. Symbol Projection Closed-Loop Control System

The closed-loop controller incorporates real-time visual feedback from an onboard vehicle camera to dynamically regulate symbol projection. The control system processes captured images of the projected symbol through a symbol extraction algorithm that isolates the symbol from background illumination and environmental noise. The extraction methodology uses differential imaging, where an image captured without symbol projection is subtracted from the current capture to isolate the symbol contribution.

The extracted symbol characteristics are compared against reference parameters to compute two position and two shape control errors [Wal+24c]. The error calculation uses an imaginary symbol appearance plane, a virtual plane surface representing the apparent projection plane of the symbol independent of the physical road surface.

The controller aims to minimize the geometric disparity between the estimated current appearance plane and the ideal reference appearance plane. These disparities manifest as positional and geometric deviations in the symbol's projection. A cascaded two-tier control then calculates the pixel utilizations to minimize the absolute of these four errors. An outer feedback controller manages the target symbol projection plane geometry, which is symbol projection primitive for SSC and an inner loop incorporating a feedforward SSC controller for pixel utilization control. The outer loop controller optimizes the position and orientation of the appearance plane relative to the reference configuration, using dedicated Proportional Integral Derivative (PID) controllers for independent regulation of planar position and geometric shape parameters. The PID controller controls the position and rotation of the target symbol projection primitive of SSC (see Sec. 7.1 and 7.2.1), which serves as an idealized flat surface target for SSC.
The primitive is the input to the inner loop SSC controller, which determines the optimal pixel utilizations. Fig. 9.8 illustrates the complete symbol projection control loop.

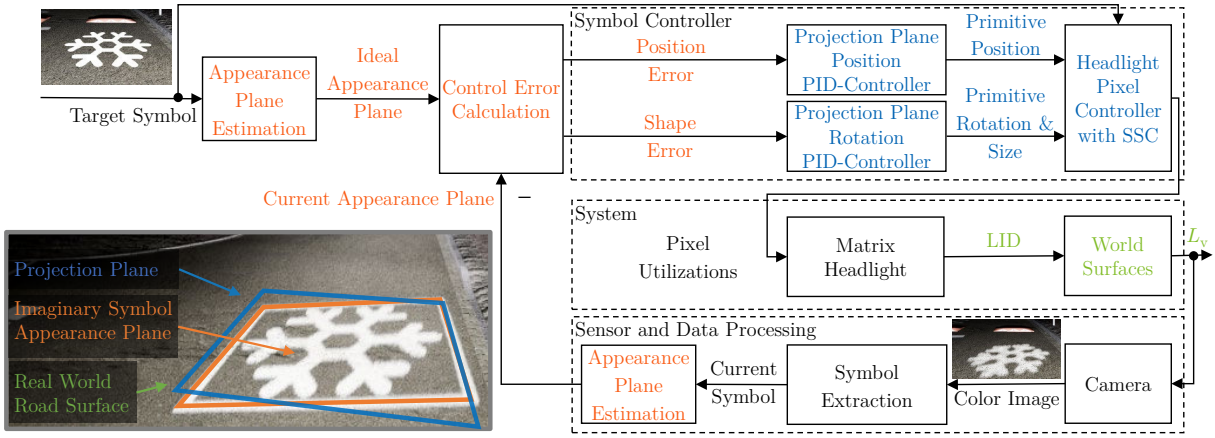



Figure 9.8.: Closed-loop feedback control system for symbol projection with SSC. At the bottom-left is a visualization of the appearance and projection plane. The color scheme shows the major connection of system components to the planes [Wal+24c].

## 9.4.2. Error Definition for Symbol Projection

The symbol projection error is based on analyzing the symbol's appearance in the camera-captured image, incorporating both positional displacement and geometric distortion components.
A symbol projection plane, which is a SSC primitive, has in general eight DoF, which are three positional DoF corresponding to Cartesian coordinates ($x$, $y$, $z$), three orientational DoF representing rotational transformations (roll, pitch, yaw) and 2D DoF defining the projection plane's geometric extent (height and width).
However, due to the symbol projection from a punctiform light source, certain DoF exhibit linked behaviors that create equivalent visual manifestations in the camera image. Specifically, variations in the projection plane's dimensional parameters (height and width) produce visual effects equivalent to certain orientational transformations. Similarly, modifications to the $x$ coordinate generate indistinguishable visual changes from those produced by variations in the $z$ coordinate.

These redundancies allow dimensional reduction in the control space by fixing specific parameters. The $z$ coordinate is constrained to the assumed road height, the yaw rotation is nullified and the plane dimensions are maintained at predetermined operational values. Consequently, the original eight DoF are reduced to four DoF for the controller.

The symbol projection control error in the camera image can be decomposed into four components representing position and shape deviations. The four errors correspond to the four retained DoF of the target projection plane. This connection is advantageous as each error component's magnitude and sign directly correlate with the required corrective adjustments.
Each of the four PID-controllers of the outer control loop minimizes the absolute of one error through continuous adjustment of the symbol projection primitive of SSC for maintaining precise symbol projection. With constant geometric relationships, e.g., headlamp pitch, camera and real-world surface, the errors are independent because the DoFs of the target projection plane and appearance plane are not linked.

Following symbol extraction, the current appearance plane from the processed image data is estimated and the control errors are formulated using image centroids. The appearance plane position and orientation errors differ between the current and target appearance plane centroids. To calculate the centroids, the extracted symbol from the camera image is stored in a scalar-valued (grayscale) image $\boldsymbol{I} \in \mathbb{R}_{\geq 0}^{n_{\text{row}} \times n_{\text{col}}}$. In the ideal case, this extraction process results in a binary representation where the symbol appears as white pixels on a uniform black background. For robust appearance analysis, the controller uses image centroids computed using raw image moments [Hu62]. Using centroids is selected due to its inherent robustness against both symmetric image distortions and uniformly distributed additive noise around the ideal position.
To calculate the centroid or Center of Gravity (COG) of an image the function $I(v, u)$ returns the scalar value at image coordinates $(u, v)$, so a raw image moment $m_{p,q}$ of order $(p, q)$ is defined as

$$m_{p,q} = \sum_{u=0}^{n_{\text{col}}-1} \sum_{v=0}^{n_{\text{row}}-1} u^p v^q I(v, u). \tag{9.4.1}$$

The central image centroid position $p_{\text{cog,cen}} \in \mathbb{R}^2$ is then computed using first-order moments normalized by the zeroth-order moment as

$$p_{\text{cog,cen}} = \begin{bmatrix} \frac{m_{1,0}}{m_{0,0}} & \frac{m_{0,1}}{m_{0,0}} \end{bmatrix}^\top = \begin{bmatrix} p_{\text{cog,cen}}(1) & p_{\text{cog,cen}}(2) \end{bmatrix}^\top, \tag{9.4.2}$$

where $p_{\text{cog,cen}}(k)$ is an accessor function for the $k$-th $p_{\text{cog,cen}}$ coordinate component.

The positional errors are computed as vector differences between the current and target centroid coordinates. Specifically, the horizontal error $e_{\text{sy,x}} \in \mathbb{R}$ is $e_{\text{sy,x}} = p_{\text{cog,cen}}(1) - p_{\text{cog,cen,s}}(1)$, while the vertical error $e_{\text{sy,y}} \in \mathbb{R}$ is $e_{\text{sy,y}} = p_{\text{cog,cen}}(2) - p_{\text{cog,cen,s}}(2)$, where $p_{\text{cog,cen,s}}$ is the target centroid position with the $k$-th component accessor $p_{\text{cog,cen,s}}(k)$.

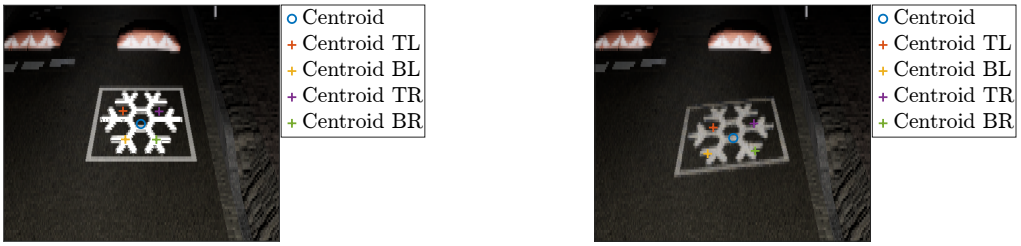
The rotational error is based on analyzing four auxiliary centroids around the central centroid to estimate the shape difference. This estimation is accomplished by partitioning $\boldsymbol{I}$ at $p_{\text{cog,cen}}$ into four rectangles as sub-images, computing individual centroids $p_{\text{cog,tl}}, p_{\text{cog,tr}}, p_{\text{cog,bl}}, p_{\text{cog,br}} \in \mathbb{R}^2$ for the Top-Left (tl), Top-Right (tr), Bottom-Left (bl) and Bottom-Right (br) regions respectively. The same decomposition is applied to the target

symbol, generating corresponding centroids $p_{\text{cog,tl,s}}, p_{\text{cog,tr,s}}, p_{\text{cog,bl,s}}, p_{\text{cog,br,s}} \in \mathbb{R}^2$. Fig. 9.9 shows the centroid distribution for both current and target symbols.
The roll error $e_{\text{sy,ro}} \in \mathbb{R}$ of rotations around the longitudinal $x$-axis of the appearance plane is computed using the vertical coordinates of the quadrant centroids as

$$\begin{aligned} e_{\text{sy,ro}} = &\frac{(p_{\text{cog,tr}}(2) - p_{\text{cog,tl}}(2) + p_{\text{cog,br}}(2) - p_{\text{cog,bl}}(2))}{2} + \\ &- \frac{(p_{\text{cog,tr,s}}(2) - p_{\text{cog,tl,s}}(2) + p_{\text{cog,br,s}}(2) - p_{\text{cog,bl,s}}(2))}{2}. \end{aligned} \tag{9.4.3}$$

Similarly, the pitch error $e_{\text{sy,pi}} \in \mathbb{R}$ of rotations around the transverse $y$-axis is

$$\begin{aligned} e_{\text{sy,pi}} = &\frac{(p_{\text{cog,tr}}(2) + p_{\text{cog,tl}}(2))}{2} - \frac{(p_{\text{cog,br}}(2) + p_{\text{cog,bl}}(2))}{2} + \\ &- \frac{(p_{\text{cog,tr,s}}(2) + p_{\text{cog,tl,s}}(2))}{2} + \frac{(p_{\text{cog,br,s}}(2) + p_{\text{cog,bl,s}}(2))}{2}. \end{aligned} \tag{9.4.4}$$




(a) Five centroids of the target symbol. (b) Five centroids of the current symbol.

Figure 9.9.: Position of one center and four sub-image centroids [Wal+24c].

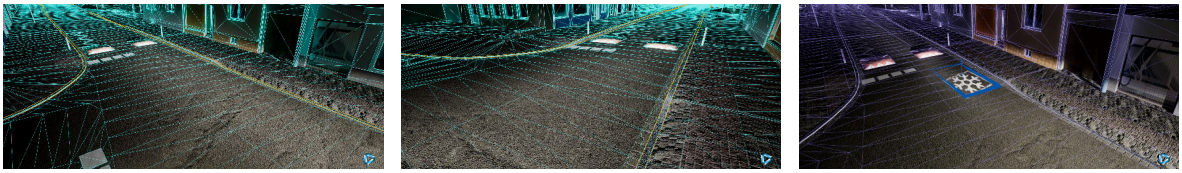
### 9.4.3. Evaluation of the Feedback Controller

The feedback controller is first evaluated against the feedforward approach using inverse ray tracing [Riv+15; Yar23a]. When the 3D environmental model exhibits perfect fidelity, the novel feedback and conventional methods theoretically converge to equivalent solutions as they operate on opposing terminals of identical light ray paths of the headlamp.
The inverse ray tracing transforms the symbol into the headlamp's LID, computing here the control error. Conversely, the feedback controller captures the emitted symbol from the headlamp via the camera and performs error computation within the camera image. However, the crucial distinction arises when considering 3D model imperfections. The open-loop conventional method can not compensate for transformation errors because it cannot incorporate information about the system output.

The second part of the evaluation involves the stability and robustness of the PID controllers. The four symbol appearance plane errors $e_{\text{sy,x}}$, $e_{\text{sy,y}}$, $e_{\text{sy,ro}}$ and $e_{\text{sy,pi}}$ are used by the PID-controllers for adjustment of the symbol projection primitive of SSC.
The controlled system is the illumination of the environment anterior to the ego-vehicle. This system has minimal temporal dependencies or delays when the computational latency of headlamp data processing can be considered negligible. However, the system exhibits spatial dependencies related to the target projection plane location, resulting in abrupt system parameter variations based on road surface irregularities for each plane position. For example, when projecting a symbol across a curb with constant primitive height,

the centroid positions experience discontinuous transitions at the boundary, creating a position-dependent nonlinearity in the control system.
Furthermore, the accuracy of centroid calculations is linked to the robustness of symbol extraction under varying environmental illumination conditions, which may fluctuate rapidly due to external factors such as opposing vehicle headlights.
These system characteristics challenge theoretical stability analysis over the entire operating range, including all possible surface profiles, ambient light conditions and headlights. Therefore, a practical measure to ensure safety is the introduction of constraints, like the primitive rotation angles, to ensure reliable calculation of all five centroids. When symbol extraction and centroid calculation fail, the controllers should be re-initialized with a flat road assumption.

The experimental evaluation of the symbol projection controller is conducted in a virtual environment using SOL, explicitly modeling the Bastionstraße 4 in Lippstadt, Germany, created by 3D Mapping Solutions GmbH [3D 24]. Fig. 9.10 illustrates the uneven road surface by showing the simulation model's mesh edges and the resulting illumination using a default flat road assuming primitive.

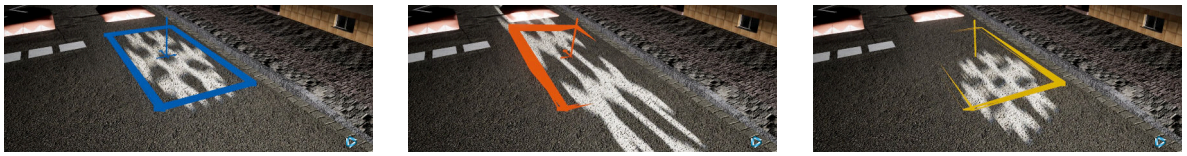


(a) Left view. (b) Right view. (c) Flat road assuming primitive.

Figure 9.10.: Unevenness of the road Bastionstraße 4 of Lippstadt. The edges of the simulation model's mesh are shown in cyan/purple to make the bumps visible. The ego-vehicle is at the bottom of the right-hand road and lights up the speed limiter [Wal+24b; Wal+24c].

The feedback controller works on the current virtual SSL|HD headlights, which cannot be shown due to confidentiality limitations. Therefore, the evaluation uses an idealized matrix headlight configuration featuring $332 \times 1{,}024 = 339{,}968$ pixels with minimal inter-pixel overlap characteristics, a horizontal beam angle coverage of $\pm 10°$ and vertical coverage ranging from $1.5°$ (upper limit) to $-3.5°$ (lower limit).
The PID controller parameters are empirically optimized through successive approximation. The position control uses PID gain factors for proportional, integral and derivative terms of 0.75, 0.5 and 0.26 and rotation control uses PID gains of 0.5, 0.1 and 0.1.
The projection parameters specify symbol projection from the left headlight at a distance of 17 m anterior to the ego-vehicle, with dimensional constraints of 2 m width and 4 m height, maintaining zero yaw rotation.
The evaluation encompassed four distinct initial positions situated at the peripheral boundaries of the headlamp's illumination zone. The initial orientation parameters are standardized across all test cases, with the projection plane configured at a roll angle of $5°$ and a pitch angle of $2°$. These initializations are selected to examine the controller's performance under extreme operating conditions.
The simulation experimental assessment uses a distance warning symbol, a sequence of chevron arrows, a triangular warning sign and a snowflake symbol. For every initial position, a different symbol is projected. Each symbol is evaluated separately.

Fig. 9.11 illustrates the difficulty of projecting symbols onto uneven surfaces, comparing the quality of the projection with a symbol plane adapted to the real surface, a non-matching plane and the default assumption of an ideal flat surface.



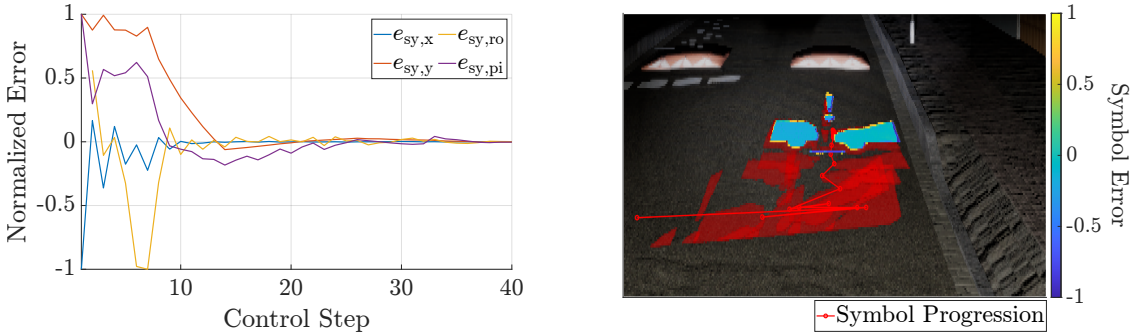
(a) Adjusted plane to match the road surface.

(b) Wrong adjusted plane with too much rotation.

(c) Flat surface assuming plane. Center point is below the road.

Figure 9.11.: Illumination differences with a target symbol projection plane adapted to the surface, a non-matching plane and the assumption of an ideal flat surface. The SSC target primitives with their center point and the normal are shown in blue, red and yellow [Wal+24b].

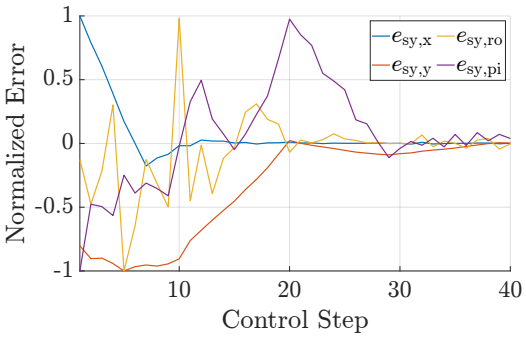
The temporal behavior of the control process and the associated error metrics for different symbols are visualized in Fig. 9.12, 9.13, 9.14 and 9.15. Each figure presents the progression of quantitative control error, qualitative symbol trajectory and changing shape per symbol. The simulation experimental results demonstrate the controller's ability to successfully guide each projected symbol to its designated target position while simultaneously correcting geometric distortions. Due to inherent irregularities in the road surface topology, a small residual error between the target and actual symbol position persists, which cannot be eliminated due to physical constraints of the system by using a flat primitive for SSC. The four start positions and symbols show that the controller can handle different symbols and road conditions. Fig. 9.12 and 9.14 show the controller's ability to cope with scenarios where symbols are partially truncated due to the finite illumination area of the headlamp.



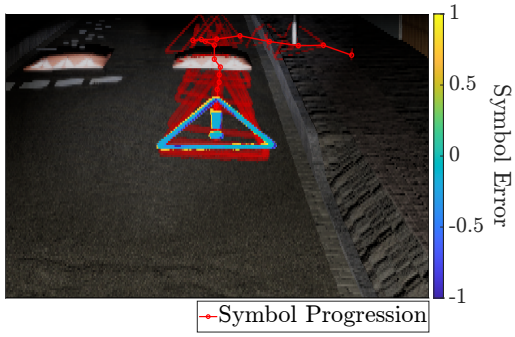

(a) Control error progression.

(b) Symbol progression in the image.

Figure 9.12.: Control error and symbol progression for the distance warning symbol. The control errors on the left are normalized to $[-1, 1]$ from their absolute maximum. The symbol projection error on the right is normalized to $[-1, 1]$ from the symbol $I_v$ setpoint. Positive values mean the projected symbol is too bright. The right image shows the progression of the symbol as a red line and past symbols as red transparent images.

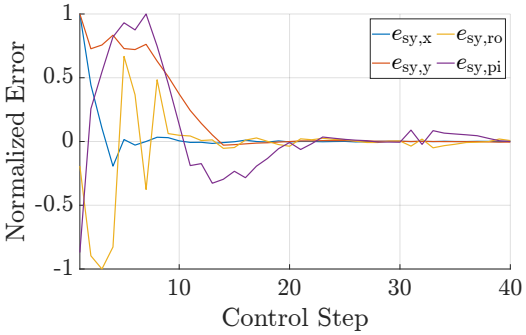



(a) Control error progression.

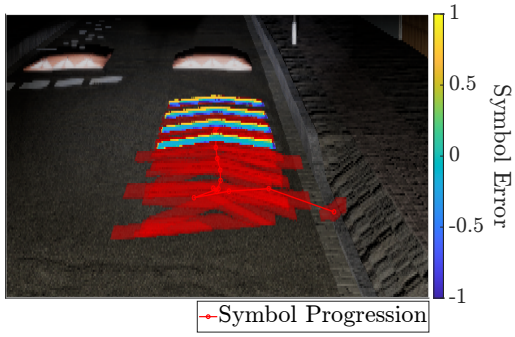



(b) Symbol progression in the image.

Figure 9.13.: Control error and symbol progression for the attention symbol [Wal+24c]. The control and symbol projection errors are normalized to $[-1, 1]$ from their absolute maximum. The red line and red transparent images indicate past symbols.

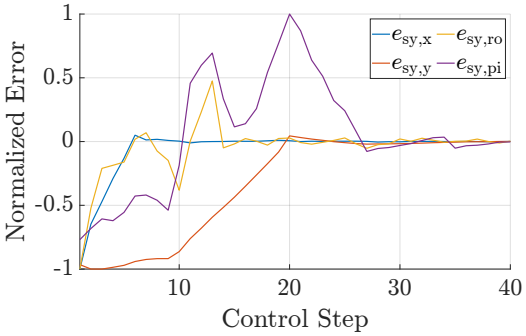



(a) Control error progression.

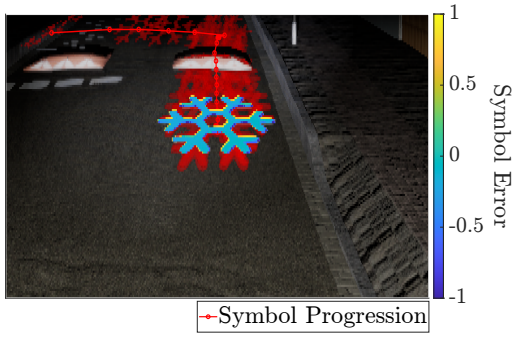



(b) Symbol progression in the image.

Figure 9.14.: Control error and symbol progression for the chevron arrows symbol [Wal+24c]. The control and symbol projection errors are normalized to $[-1, 1]$ from their absolute maximum. The red line and red transparent images indicate past symbols.



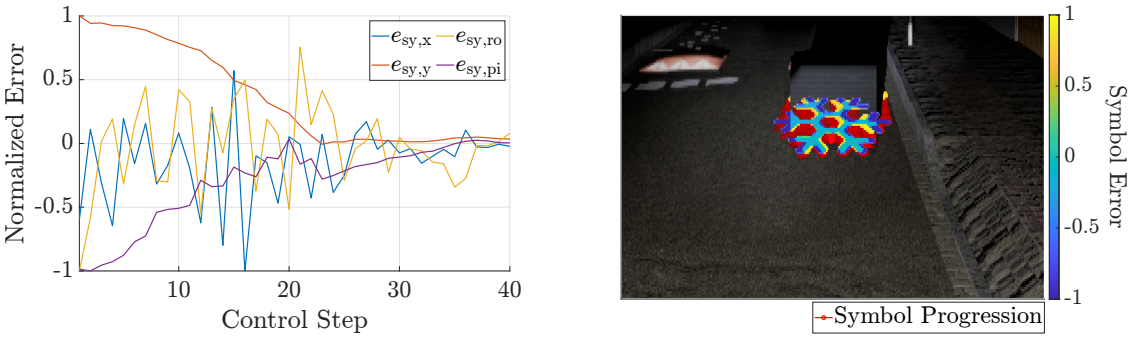

(a) Control error progression.




(b) Symbol progression in the image.

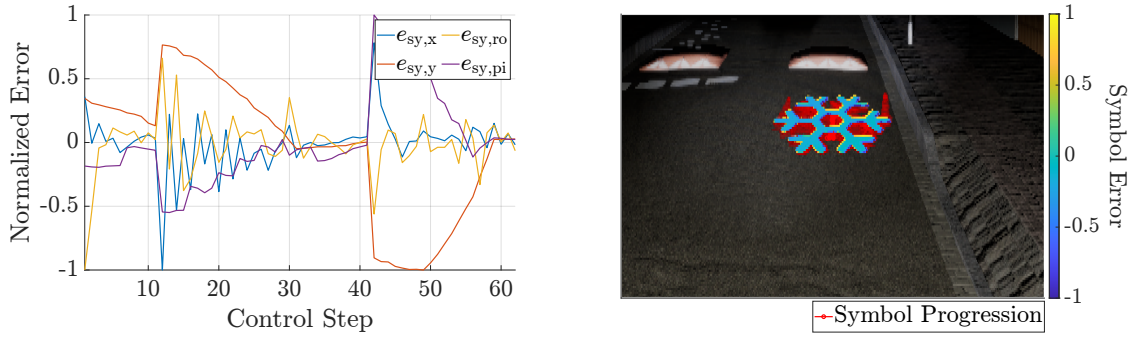
Figure 9.15.: Control error and symbol progression for the snowflake symbol [Wal+24c]. The control and symbol projection errors are normalized to $[-1, 1]$ from their absolute maximum. The red line and red transparent images indicate past symbols.

To evaluate the controller's robustness against structural disturbances, additional simulation experiments are conducted introducing controlled optical occlusions of the target area. Fig. 9.16 and 9.17 present the controller's response to static and dynamic black box occlusions in the target projection zone.
The results indicate that the controller actively compensates for these disturbances primarily through adaptive adjustments of the projection plane's pitch angle, demonstrating its capability to maintain symbol visibility and positioning under adverse conditions.




(a) Control error progression. (b) Symbol progression in the image.

Figure 9.16.: Control error and symbol progression for a black box in the symbol target area [Wal+24c]. The control and symbol projection errors are normalized to $[-1, 1]$ from their absolute maximum. The red line and red transparent images indicate past symbols.




(a) Control error progression. (b) Symbol progression in the image.

Figure 9.17.: Control error and symbol progression for a dynamic black box in the symbol target area. The black box appears at control step 11 and disappears at step 41 [Wal+24c]. The control and symbol projection errors are normalized to $[-1, 1]$ from their absolute maximum. The red line and red transparent images indicate past symbols.

The feedback controller empirically demonstrates robust capability in compensating for irregular road surfaces and structural perturbations, maintaining precise symbol positioning and geometric integrity [Wal+24b; Wal+24c]. Empirical results indicate reliable controller performance across diverse environmental conditions and initialization parameters, although formal stability proofs remain an open research direction. The controller exhibits versatility in accommodating various matrix headlights, symbol geometries and road conditions. An advantage of the feedback approach is its elimination of the requirement for real-time road profile estimation, a significant limitation in previous methodologies [Riv+15; Yar23a]. The simulation experimental evaluation presented here proves the controller's practical viability. However, additional real-world testing using the test vehicle and the SOL communication module is recommended to validate the controller under broader operational conditions.

# 10

# Conclusion and Outlook‡t

This thesis has presented novel approaches for real-time simulation and control of HD matrix headlights, making several significant contributions to the efficient development, verification and validation of automotive lighting systems. The work addresses the two fundamental challenges of achieving efficient real-time simulation of arbitrary matrix headlights in commercial rendering engines and developing real-time rapid prototyping methods for dynamic arbitrary lighting functions.

The first significant contribution is developing a novel, physically accurate, real-time simulation approach that combines standard rendering engine spotlights with SpMV. This novel combination enables efficient simulation of HD matrix headlights by reformulating the linear superposition of individual pixellight illuminations as a SpMV operation of a sparse headlight matrix and a pixel utilization vector.

This SpMV description of a matrix headlight is independent of lighting technology as long as the pixellight photometric data can be described in an IES-file-like format.

SpMV uses well-established sparse matrix formats like CSR and parallel computing capabilities of GPUs for real-time capability.

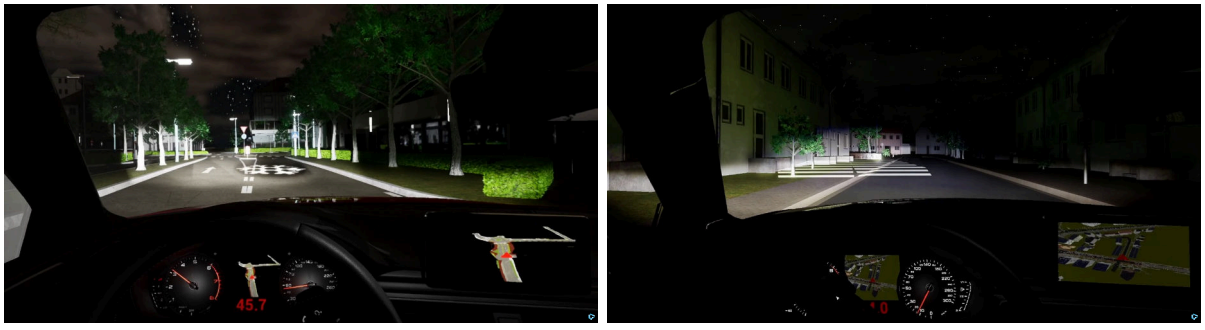
A key innovation is using default rendering engine spotlights as the virtual light source, allowing the simulation method to be used in the Unity and Unreal rendering engines (see Fig. 6.3 and Fig. 6.4). Using a default spotlight allows using established engine features like volumetric fog, real-time ray tracing and dynamic shadows for headlight simulation without additional effort.

Experimental verification demonstrated that the method could simulate a matrix headlight with $1.51 \cdot 10^6$ pixels at a mean computation time of 0.47 ms, proving its real-time capability. A notable advancement is introducing an incremental update method that optimizes performance by only recomputing changed illumination regions of beam patterns.

The second significant contribution is the SSC algorithm for real-time matrix headlight control for arbitrary lighting functions and pixels. SSC combines ray tracing, SSAA and the MapReduce parallel processing paradigm into an intuitive yet computationally efficient approach for rapidly prototyping arbitrary dynamic lighting functions with arbitrary pixel shapes. This flexibility is achieved by SSC with a unique combination of high-level lighting definition with rectangular 3D target objects, so-called primitives and parallel pixel utilization optimization.

SSC enables the implementation of arbitrary lighting functions through dynamic, intuitive manipulation of setpoint LIDs by adjusting the primitives. The user places, orientates and parametrizes the primitives in front of the ego-vehicle to define the setpoint LID in the 3D space as a collection of desired projected symbols.
SSC's computational efficiency and real-time capability stems from its parallel processing architecture, with experimental evaluation showing that SSC can control a matrix headlight with $1.05 \cdot 10^6$ pixels at a median computation time of 1.79 ms while maintaining high illumination quality.
Although SSC cannot achieve the optimal illumination of non-real-time JPO approaches in all test cases, it provides comparable subjective illumination quality with reduced computational complexity and real-time capability. In many applications for HD matrix headlights, the difference in lighting quality between SSC and JPO is subjectively difficult for human vision to recognize and SSC and JPO appears to be subjective similar (e.g., Fig. 8.1).
As SSC can mimic specialized headlight control algorithms with its flexibility, it can be used to identify optimal control strategies for specific lighting functions and pixels through SSC parameter optimization.

The third significant contribution is the development of the SOL real-time matrix headlight simulation in Unreal 5, which integrates the proposed real-time simulation and control methods into a rapid prototyping and development tool. The developed methods enable efficient virtual development of advanced lighting functions while maintaining real-time performance constraints. Fig. 10.1 shows a virtual matrix headlight test drive with SOL. SOL allows mirroring the headlamp illumination in simulation and real-world scenarios, enabling simulation parameter adjustment, verification and validation. Therefore, SOL enables hybrid development approaches by supporting both virtual prototyping and HiL testing with real matrix headlights.
SOL's architecture allows efficient development and evaluation of novel lighting functions for automated driving. Practical applications of SOL are energy-efficient OBL, MBL for environment-optimal illumination and observer-specific distortion-free symbol projection with closed-loop feedback control. The examples prove that SOL has been validated through testing with real matrix headlights, demonstrating its practical applicability in automotive lighting development.



(a) Snowflake projection and bend lighting [Kau+24]. (b) Structured light projection [Kau+24].

Figure 10.1.: Virtual matrix headlight test drive with SOL [Kau+24].

Several promising directions for future research emerge from this work.
As mentioned in Sec. 5.4 for the simulation domain, investigating adaptive sparse matrix formats tailored to specific headlight architectures and incremental update strategies could improve computational efficiency.
SSC should be extended to support more advanced lighting functions through secondary ray analysis like cooperative inter-vehicle symbol projection or dynamic adaptation to street lighting conditions (see Sec. 7.5.1).
Additional future work should focus on improving mapping and reduction functions or adding additional SSC processing steps (see Sec. 7.5). The development of adaptive strategies optimized for different setpoint LID scenarios and headlight configurations could improve the handling of edge cases and non-uniform pixellight LID distributions. SSC could realize flexible, adaptable control strategies by dynamically adjusting its operational parameters based on real-time analysis of the driving environment and setpoint LID requirements. For instance, projecting symbolic information using peripheral pixels may require different optimal control parameters than projections using central pixels.

Integrating SSC into SOL creates promising research directions for matrix headlights for automated driving [Mül+25a; Wal+25c; Gla+25]. Future research applications of SOL are energy-consumption optimization of environment illumination and substantial lighting strategies like MBL. SOL could be enhanced to analyze power consumption patterns and optimize lighting functions for reduced energy usage while maintaining safety and functionality requirements.
Machine learning applications present significant opportunities to use SOL capabilities. The system could generate training data for neural networks used in lighting control systems, helping to improve the robustness of adaptive lighting functions. Additionally, machine learning techniques could be applied to optimize beam patterns for specific driving conditions and scenarios and for handling environmental variations and system degradation. This includes investigating adaptive algorithms that can compensate for aging effects in light elements like reduced brightness or defects, lens contamination and varying atmospheric conditions [Jan+19].
Future work could also explore how matrix headlight control can be optimized for both human and machine vision systems simultaneously, potentially improving the robustness of autonomous vehicle perception systems under challenging lighting conditions. By simulating illumination and sensor responses with SOL, developers could optimize beam patterns to enhance the performance of automated driving sensors like cameras while maintaining good visibility for human drivers. This dual-objective optimization problem requires careful consideration of the different human and machine vision requirements, particularly in contrast, dynamic range and temporal response characteristics.

In conclusion, this thesis has substantially contributed to the real-time simulation and control of matrix headlights, providing a foundation for future research in automotive lighting. The proposed methods enable the development, verification and validation of advanced lighting functions while maintaining real-time performance constraints, representing a significant step forward in the efficient development of next-generation automotive lighting systems for automated driving.

# Bibliography


[3D 24] **3D Mapping Solutions GmbH (Ed.):** *Road surface models.* https://www.3d-mapping.de/en/our-service/driving-dynamics/ Last accessed on 15.12.2024. 2024.

[AF14] **I. Agricola and T. Friedrich:** *Elementargeometrie: Fachwissen für Studium und Mathematikunterricht.* Springer, 2014.

[Ake+18] **T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire:** *Real-time rendering.* AK Peters/crc Press, 2018.

[Ake03] **T. Akenine-Möller:** "An extremely inexpensive multisampling scheme". In: *Chalmers University of Technology, Technical Report No. 03-14* (2003).

[Ale81] **G. Alefeld:** "On the convergence of Halley's Method". In: *The American Mathematical Monthly* 88.7 (1981).

[Alm+21] **Y. Almehio, M. Mimoun, R. Mezari, and I. H. El:** "Method for controlling a lighting system of a motor vehicle". US20240233301A9. 2021.

[Ans23] **Ansys,Inc (Ed.):** *Ansys AVxcelerate Headlamp – Vehicle Headlight Simulation Software.* https://www.ansys.com/de-de/products/av-simulation/ansys-avxcelerate-headlamp Last accessed on 31.03.2023. 2023.

[Aud17] **Audi AG (Ed.):** *Lighting.* https://www.audi-mediacenter.com/en/audi-technology-lexicon-7180/lighting-7187 Last accessed on 17.01.2025. 2017.

[Aud22a] **Audi AG (Ed.):** *Completely digitized light in the Audi A8.* https://www.audi-mediacenter.com/en/audimediatv/video/completely-digitized-light-in-the-audi-a8-6082 Last accessed on 31.03.2023. Cropped input colors above 180 and rescale to make lighting function better visible. 2022.

[Aud22b] **Audi AG (Ed.):** *Digital Matrix LED headlights: Lane light, including orientation light (on highways).* https://www.audi-mediacenter.com/en/photos/detail/techday-digital-light-109729 Last accessed on 18.10.2023. 2022.

[Aus+23] **A. Austerschulte, S. Bauer, and S. Schildmann:** "Evolution of HD-Headlamps: What's coming next?" In: *15th International Symposium on Automotive Lighting (ISAL).* 2023.

[AVS22a] **AVSimulation (Ed.):** *SCANeR Headlights Pack*. `https://www.avsimulation.com/pack-headlights/` and `https://www.avsimulation.com/headlight/` Last accessed on 31.03.2023. 2022.

[AVS22b] **AVSimulation (Ed.):** *Use Case: Pixel Lighting in SCANeR*. `https://www.avsimulation.com/use-case-pixel-lighting-in-scaner/` Last accessed on 31.03.2023. 2022.

[AW21] **P. S. Adam Marrs and I. Wald:** *Ray Tracing Gems II*. Apress, 2021.

[Bar+15] **F. Barmeyer, C. Funk, T. Armbruster, J. Reim, and S. Omerbegovic:** "System and method for object-based control of a high-resolution headlight for a motor vehicle". DE102015016375A1. 2015.

[Bau+21] **M. Baumann, K. Trampert, C. Neumann, R. Danov, and E. Thiessen:** "Verfahren zur Validierung von Symbolprojektionen mit heterogenen Pixelsätzen am Beispiel eines HD-Scheinwerfers". In: *LICHT 2021 – 24. Europäischer Lichtkongress*. 2021.

[Bau+22] **M. Baumann, M. Helmer, C. Neumann, D. Blömer, Y. Won, G. Han, and H. LEE:** "Symbol Projection for Pedestrians". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[BB09] **M. M. Baskaran and R. Bordawekar:** "Optimizing sparse matrix-vector multiplication on GPUs". In: *IBM research report RC24704* W0812–047 (2009).

[Ben22] **L. Bennes:** *Win the Race for New Adaptive Driving Beam (ADB) Territories*. `https://www.ansys.com/resource-center/webinar/win-the-race-for-adaptive-driving-beam-territories` Last accessed on 31.03.2023. 2022.

[Ber+03] **J. Berssenbrügge, J. Gausemeier, M. Grafe, C. Matysczok, and K. Pöhland:** "Real-Time Representation of Complex Lighting Data in a Nightdrive Simulation". In: *Eurographics Workshop on Virtual Environments*. 2003.

[Ber+14] **S. Berlitz, C. Funk, S. Omerbegovic, and T. Armbruster:** "Informing a road user about an autopilot-controlled journey". DE102014011811B4. 2014.

[Ber05] **J. Berssenbrügge:** "Virtual Nightdrive – Ein Verfahren zur Darstellung der komplexen Lichtverteilungen moderner Scheinwerfersysteme im Rahmen einer virtuellen Nachtfahrt". Doctoral thesis. Fakultät für Maschinenbau, Universität Paderborn, 2005.

[BG08] **N. Bell and M. Garland:** *Efficient sparse matrix-vector multiplication on CUDA*. Tech. rep. Nvidia Technical Report NVR-2008-004, Nvidia Corporation, 2008.

[BG09] **N. Bell and M. Garland:** "Implementing sparse matrix-vector multiplication on throughput-oriented processors". In: *Proceedings of the conference on high performance computing networking, storage and analysis*. 2009.

[BK17] **D. Brunne and F.-J. Kalze:** "Outlook on high resolution pixel light". In: *12th International Symposium on Automotive Lighting (ISAL)*. 2017.

[BL14] **L. Barba and S. Langerman:** "Optimal detection of intersections between convex polyhedra". In: *Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2014.

[BMW23] **BMW Group (Ed.):** *BMW Group puts new light channel for testing headlights and exterior lighting into operation.* `https://www.press.bmwgroup.com/canada/article/detail/T0418122EN/bmw-group-puts-new-light-channel-for-testing-headlights-and-exterior-lighting-into-operation` Last accessed on 15.12.2024. 2023.

[Böh19] **G. Böhm:** "Future of Automotive Headlamps – Light for Sensors". In: *13th International Symposium on Automotive Lighting (ISAL)*. 2019.

[Bor+22] **M. Borowski, D. Velkin, S. Finn, P. Kober, and U. Kostanzer:** "A Unique High-Definition Night Driving Simulator for Development and Testing of New Generation Lighting Systems at Mercedes-Benz". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[Bro92] **L. G. Brown:** "A survey of image registration techniques". In: *ACM computing surveys (CSUR)* 24.4 (1992).

[Bur12] **B. Burley:** "Physically-Based Shading at Disney". In: *Acm Siggraph*. Vol. 2012. 2012.

[Bus03] **S. R. Buss:** *3D computer graphics: a mathematical introduction with OpenGL*. Cambridge University Press, 2003.

[Cla+21] **S. Cladé, G. Planche, K. BEEV, R. Mezari, and M. Mimoun:** "High Definition Lighting solutions to complement camera and improve AEB Effectiveness at Night". In: *SIA VISION*. 2021.

[Com06] **P. Comninos:** *Mathematical and computer programming techniques for computer graphics*. Springer, 2006.

[Cro77] **F. C. Crow:** "The aliasing problem in computer-generated shaded images". In: *Communications of the ACM* 20.11 (1977), pp. 799–805.

[Dal+15] **S. Dalton, L. Olson, and N. Bell:** "Optimizing sparse matrix—matrix multiplication for the gpu". In: *ACM Transactions on Mathematical Software (TOMS)* 41.4 (2015).

[Dan+21] **R. Danov, E. Thiessen, K. Trampert, and M. Helmer:** "Method and device for generating a light object". DE102021202584A1. 2021.

[De 06] **B. De Greve:** *Reflections and refractions in ray tracing.* https://graphics.stanford.edu/courses/cs148-10-summer/docs/2006--degreve--reflection_refraction.pdf Last accessed on 10.10.2024. 2006.

[Det17] **A. Deter:** *Hightech im Hella Lichtkanal Lippstadt.* https://www.topagrar.com/mediathek/fotos/hightech-im-hella-lichtkanal-lippstadt-9708068.html Last accessed on 15.12.2024. 2017.

[DF17] **D. Duhme and B. Fischer:** "Next Generation LCD Module". In: *12th International Symposium on Automotive Lighting (ISAL).* 2017.

[DG04a] **J. Dean and S. Ghemawat:** "MapReduce: Simplified Data Processing on Large Clusters". In: *OSDI'04: Sixth Symposium on Operating System Design and Implementation* (2004).

[DG04b] **J. Dean and S. Ghemawat:** "System and method for efficient large-scale data processing". US7650331B1. 2004.

[Doe61] **R. Doerfling:** *Mathematik für Ingenieure und Techniker: Ein Lehrbuch.* De Gruyter, 1961.

[Dos+17] **A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun:** "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning.* 2017, pp. 1–16.

[DY11] **R. Dosselmann and X. D. Yang:** "A comprehensive assessment of the structural similarity index". In: *Signal, Image and Video Processing* 5 (2011).

[Epi23a] **Epic Games (Ed.):** *Spotlight – Unreal Engine.* https://docs.unrealengine.com/en-US/spot-lights-in-unreal-engine/ Last accessed on 31.03.2023. 2023.

[Epi23b] **Epic Games (Ed.):** *Unreal Engine 5.2.* https://www.unrealengine.com Last accessed on 31.03.2023. 2023.

[Erk+22] **A. Erkan, D. Hoffmann, T. Singer, and I. T. Q. Khanh:** "Field Test Validation of the Headlamp Safety Performance Rating (HSPR)". In: *14th International Symposium on Automotive Lighting (ISAL).* 2022.

[Erk+24] **A. Erkan, M. Kruppa, C. Hinterwälder, and C. Funke:** "Digital Matrix Light – How $\mu$LED arrays are pushing the limits of Adaptive Digital Light Distribution (ADLD) design". In: *SIA VISION 2024.* 2024.

[ET 21] **ET Auto (Ed.):** *Presentation on Enabling Autonomous Driving & ADAS with Simulation & Testing.* https://www.youtube.com/watch?v=IhnyRr-FoXQ Last accessed on 31.03.2023. 2021.

[Eve01] **C. Everitt:** "Projective texture mapping". In: *White paper, NVidia Corporation* 4.3 (2001).

[Evr+16] **L. Evrard, B. Reiss, J. Ripperger, and D. Cabanne:** "Modular design for LED glare free High Beam". In: *Auto Tech Review* 5.6 (2016), pp. 26–31.

[Far01] **G. Farin:** *Curves and surfaces for CAGD: a practical guide*. Elsevier, 2001.

[Fec60] **G. T. Fechner:** *Elemente der psychophysik*. Vol. 2. Breitkopf u. Härtel, 1860.

[Fil+17] **S. Filippone, V. Cardellini, D. Barbieri, and A. Fanfarillo:** "Sparse matrix-vector multiplication on GPGPUs". In: *ACM Transactions on Mathematical Software (TOMS)* 43.4 (2017).

[FM99] **M. D. Flouris and E. P. Markatos:** "The network RamDisk: Using remote memory on heterogeneous NOWs". In: *Cluster computing* 2 (1999), pp. 281–293.

[Fol+97] **J. D. Foley, A. v. Dam, S. K. Feiner, J. F. Hughes, and M. P. Carter:** "Computer graphics: Principles and practice". In: *Color Research and Application* 22.1 (1997).

[For20] **Forvia-Hella GmbH (Ed.):** *The HELLA light tunnel*. `https://www.youtube.com/watch?v=-D70AESWdpA` Last accessed on 15.12.2024. 2020.

[For21] **Ford Europe (Ed.):** *Ford's New Night Driving Headlights are Ahead of the Curve*. `https://www.youtube.com/watch?v=PN5lvwjf1VY&t=105s` Last accessed on 15.11.2023. 2021.

[For23a] **Forvia-Hella GmbH (Ed.):** *HELLA Lighting*. `https://www.hella.com/lighting/en` Last accessed on 31.03.2023. 2023.

[For23b] **Forvia-Hella GmbH (Ed.):** *Lighting technology at the highest level: HELLA and Porsche launch world's first SSL | HD matrix headlamp*. `https://www.hella.com/en/Newsroom/Press-Releases/2023-07-18-Lighting-technology-at-the-highest-level-HELLA-and-Porsche-launch-world-s-first-SSL-HD-matrix-1749/` Last accessed on 27.11.2024. 2023.

[For24] **Forvia-Hella GmbH (Ed.):** *SSL | HD Matrix Led Lamps: Interesting Facts, Repair Instructions And Practical Tips*. `https://www.hella.com/techworld/us/Technical/Automotive-lighting/SSL-HD-Matrix-LED-lamps-95887/` Last accessed on 25.06.2024. 2024.

[FR22] **F. Freytag and E. Rosenhahn:** "Headlamp Safety Performance Rating (HSPR): Method Improvements triggered by GTB Experts". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[Fun+16] **C. Funk, F. Barmeyer, T. Armbruster, and S. Omerbegovic:** "Driver assistance system for controlling a light emission of a vehicle-side headlamp device". DE102016001692A1. 2016.

[Fun23] **C. Funke:** "Digital Matrix Light for Increased Road Safety". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Gan85] **W. Gander:** "On Halley's iteration method". In: *The American Mathematical Monthly* 92.2 (1985).

[GD21] **N. V. L. Giang and N. T. Dinh:** "Research the Anti-dazzle Headlight System by Leds Matrix with Image Processing Method". In: *2021 International Conference on System Science and Engineering (ICSSE)*. IEEE. 2021, pp. 77–82.

[GI19] **W. Gonçalves and A. Issoufou:** "Optimized ADB Symbol Projection". In: *13th International Symposium on Automotive Lighting (ISAL)*. 2019.

[Gla+25] **F. Glatzel, N. Müller, M. Waldner, and T. Bertram:** "Energy-Efficient Pedestrian Marker Light Based on Pose Probability for Computer Vision". In: *16th International Symposium on Automotive Lighting (ISAL)*. Best Poster Award. 2025.

[Gla89] **A. S. Glassner:** *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[Goc13] **T. Gocke:** "Objektivierte Homogenitätsbewertung des Erscheinungsbildes automobiler Signalleuchten". Doctoral thesis. Technische Universität Ilmenau, 2013.

[Göt+07] **M. Götz, S. Hagedorn, C. Hüster, A. Kornek, W. Menk, R. Jacek, and R. Giebl:** "System for generating a light beam in the area in front of a motor vehicle". WO2009027221A1. 2007.

[Gra97] **J. Gravesen:** "Adaptive subdivision and the length and energy of Bézier curves". In: *Computational Geometry* 8.1 (1997).

[GRE19] **GRE (Ed.):** "Comparison of ADB provisions between current R48 and R53 proposal". In: *Informal document GRE* (2019). `https://unece.org/DAM/trans/doc/2019/wp29gre/GRE-82-18e.pdf` Last accessed on 12.10.2023.

[Gro+18] **S. Groetsch, J. Reill, M. Schwind, S. Haneder, and U. Hiller:** "Illumination vs. Visualization in Headlamps: Way towards HD light source requirements". In: *SIA VISION*. 2018.

[GTB23] **GTB (Ed.):** *Headlight Safety Performance Rating (HSPR) – GTB Recommended Practice*. `https://www.gtb-lighting.org/wp-content/uploads/2023/10/2023-09_GTB-Recommended-Practice-for-HSPR_SEP2023.pdf` Last accessed on 20.12.2024. 2023.

[GW17] **R. Gonzalez and R. Woods:** *Digital Image Processing Global Edition*. Pearson Deutschland, 2017.

[GW24] **G. Gräfe and M. Waldner:** *Umgebungsmodell zur Scheinwerfersimulation für das automatisierte Fahren*. VDI Mechatroniktagung. Keynote. 2024.

[HA01] **K. Hormann and A. Agathos:** "The point in polygon problem for arbitrary polygons". In: *Computational geometry* 20.3 (2001).

[HA19] **E. Haines and T. Akenine-Möller:** *Ray Tracing Gems*. Apress, 2019.

[Ham+22] **M. Hamm, J. Kobbert, and C. Hinterwälder:** "Digitial Projections: Distraction Potential for Other Traffic Participants". In: *SIA VISION*. 2022.

[Har13a] **M. Harris:** *How to Access Global Memory Efficiently in CUDA C/C++ Kernels*. `https://developer.nvidia.com/blog/how-access-global-memory-efficiently-cuda-c-kernels/` Last accessed on 15.11.2023. 2013.

[Har13b] **M. Harris:** *Using Shared Memory in CUDA C/C++*. `https://developer.nvidia.com/blog/using-shared-memory-cuda-cc/` Last accessed on 15.11.2023. 2013.

[Hes15] **H. Hesse:** "BMBF-Project VOLIFA 2020 – High resolution light distribution by using a LCD". In: *11th International Symposium on Automotive Lighting (ISAL)*. 2015.

[HG23] **D. Hýnar and E. Guzejová:** "New field of headlight optical design – imaging systems and their usecases". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[HNI19] **HNI (Ed.):** *Kooperationsprojekt „Smart Headlamp Technology" – Virtuelle Entwicklung hochaufgelöster Scheinwerfer mit Echtzeitmodellen*. `https://www.hni.uni-paderborn.de/aktuelles-veranstaltungen/details/?tx_ttnews[tt_news]=1173&cHash=6fa9e36a8fa07ba0508abea20a3e03f4` Last accessed on 15.11.2023. 2019.

[Hof09] **I. Hoffmann:** "Method and device for controlling the light emission of a front headlamp of a vehicle". EP2508391B1. 2009.

[HS18] **B. Hummel and S. Saralajew:** "Method for operating a lighting device of a motor vehicle". DE102018103262B4. 2018.

[Hu62] **M.-K. Hu:** "Visual pattern recognition by moment invariants". In: *IRE transactions on information theory* 8.2 (1962).

[Hum+18] **B. Hummel, S. Saralajew, S. Söhner, and G. Benderman:** "Method for operating a lighting device of a motor vehicle". DE102018103487B4. 2018.

[Hum10] **B. Hummel:** "Blendfreies LED-Fernlicht". Doctoral thesis. 2010.

[Hum18] **B. Hummel:** "Method for controlling a light distribution of a plurality of luminous elements". DE102018102667A1. 2018.

[Hüs+13] **C. Hüster, J. Roslak, U. Venker, and C. Wilks:** "Method for controlling a matrix beam headlamp with adaptive light functions". DE102013104277A1. 2013.

[HZ10] **A. Hore and D. Ziou:** "Image quality metrics: PSNR vs. SSIM". In: *2010 20th international conference on pattern recognition*. IEEE. 2010.

[IES20] **IES (Ed.):** "IES Standard File Format for the Electronic Transfer of Photometric Data and Related Information". In: *ANSI/IES LM-63-19* (2020). https://store.ies.org/product/lm-63-19-approved-method-ies-standard-file-format-for-the-electronic-transfer-of-photometric-data-and-related-information/?v=3a52f3c22ed6 Last accessed on 23.11.2023.

[Im00] **E.-J. Im:** "Optimizing the performance of sparse matrix-vector multiplication". PhD thesis. University of California, Berkeley, 2000.

[IM11] **R. Isermann and M. Münchhof:** *Identification of dynamic systems: an introduction with applications*. Vol. 85. Springer, 2011.

[Ins24] **Instrument Systems GmbH (Ed.):** *AMS Screen Imaging System*. https://www.instrumentsystems.com/en/systems/ams-screen-imaging-system Last accessed on 02.07.2024. 2024.

[Jac12] **N. Jacobson:** *Basic algebra I*. Courier Corporation, 2012.

[Jan+19] **P. Janke, C. Lankeit, M. Krämer, and M. Waldner:** "Intelligenz im Scheinwerfer: Künftige LED-Systeme heilen sich selbst". In: *Automobil-Elektronik 04-05/2019* (2019).

[Jia+20] **M. Jian Min, W. Dong, J. Shuqi, C. Qi, X. Wang, L. Baohua, and W. Hao:** "Optimization control method for office lighting". In: *2020 Chinese Automation Congress (CAC)*. IEEE. 2020.

[Jun23] **F. Jung:** "Lighting real-world driving can be reduced by up to 50%". In: *ATZ worldwide* 125.10 (2023), pp. 24–27.

[Kaj86] **J. T. Kajiya:** "The rendering equation". In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 1986.

[Kal+16] **F.-J. Kalze, U.-C. Knaack, and B. Böke:** "Multi-row Headlamp System in the Mercedes-Benz E-Class". In: *ATZ worldwide* 118.2 (2016).

[Kan+22] **A. Kanj, C. Prat, M. Drezet, and H. E. Idrissi:** "Method for controlling a light pattern and automotive lighting device". WO2022207581A1. 2022.

[Kat+22] **M. Katona, K. Trampert, C. Schwanengel, U. Krüger, and C. Neumann:** "Geometric system analysis of ILMD-based LID measurement systems using Monte-Carlo simulation". In: *Journal of Physics: Conference Series*. Vol. 2149. IOP Publishing. 2022.

[Kau+21a] **R. Kauschke, N. Rüddenklau, F. Ernst, and M. Waldner:** "Effizientere Entwicklung von Scheinwerfersystemen". In: *ATZ – Automobiltechnische Zeitschrift* 123.2 (2021).

[Kau+21b] **R. Kauschke, N. Rüddenklau, F. Ernst, and M. Waldner:** "More Efficient Development of Headlamp Systems". In: *ATZ worldwide* 123.2 (2021).

[Kau+22] **R. Kauschke, M. Waldner, N. Müller, T. Bertram, and M. Grünke:** "Ist automatisiertes Fahren mit kamera-optimierten Lichtfunktionen besser möglich? Top-Down-Entwicklung optimierter Lichtverteilungen für das automatisierte Fahren". In: *9. VDI-Tagung Optische Technologien in der Fahrzeugtechnik*. 2022.

[Kau+23] **R. Kauschke, B. Kubitza, C. Wilks, M. Waldner, N. Müller, T. Bertram, and M. Grünke:** "AHEAD – Innovative Lighting for Automated Driving – Lighting Strategies for a Better Camera Detection". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Kau+24] **R. Kauschke, B. Kubitza, C. Wilks, M. Waldner, N. Müller, T. Bertram, M. Grünke, and A. Maroke:** "AHEAD-Light Distributions for camera-based automated driving – in simulation and validated in real world scenarios". In: *SIA VISION*. 2024.

[KE95] **J. Kennedy and R. Eberhart:** "Particle swarm optimization". In: *Proceedings of ICNN'95 – International Conference on Neural Networks*. IEEE. 1995.

[Kie12] **H. Kiel:** "Bewertung von Kraftfahrzeugscheinwerfern mit Lichtsimulation". Doctoral thesis. Technische Universität Ilmenau, 2012.

[Kin+89] **D. R. Kincaid, T. C. Oppe, and D. M. Young:** *ITPACKV 2D user's guide*. Tech. rep. Texas Univ., Austin, TX (USA). Center for Numerical Analysis, 1989.

[Klä+18] **R. Klädtke, S. Hauptmann, and G. Böhm:** "Lighting technology for communication in autonomous driving". In: *ATZ worldwide* 120.2 (2018), pp. 28–33.

[Kle+19] **M. Kleinkes, I. Moellers, W. Pohlmann, and C. Wilks:** "Lighting device for a motor vehicle, in particular high-resolution headlight for a motor vehicle". US20220290827A1. 2019.

[Kle+22] **M. Kleinkes, W. Pohlmann, and C. Wilks:** "SSL|HD – High Tech Light for new safety & comfort functions". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[Kle03] **M. Kleinkes:** "Objektivierte Bewertung des Gütemerkmals Homogenität für Scheinwerfer-Lichtverteilungen". Doctoral thesis. Universität Bielefeld, 2003.

[Knö+19] **M. Knöchelmann, M. P. Held, G. Kloppenburg, and R. Lachmayer:** "High-resolution headlamps – technology analysis and system design". In: *Advanced Optical Technologies* 8.1 (2019), pp. 33–46.

[Kob19] **J. Kobbert:** "Optimization of Automotive Light Distributions for Different Real Life Traffic Situations". en. Doctoral thesis. Technische Universität, 2019.

[Kos+20] **K. Kosmas, J. Kobbert, and T. Q. Khanh:** "Requirements for dynamic levelling devices to prevent headlamp glare blinding oncoming road users". In: *GRE-83-03* (2020).

[KP18] **A. Kanj and C. Prat:** "Method for controlling a light pattern using a matrix of light sources responsive to steering angle". US11505112B2. 2018.

[KR23] **S. Köhler and P. Rausch:** "Benchmarking color fringes with focus on high resolution headlamps". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Kre+14] **M. Kreutzer, G. Hager, G. Wellein, H. Fehske, and A. R. Bishop:** "A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units". In: *SIAM Journal on Scientific Computing* 36.5 (2014).

[Kri+23] **F. Krieft, C. Neumann, and M. Niedling:** "Headlamp sensor: Visibility sensor based on structured light". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Kri24] **F. Krieft:** *Erfassung der Sichtweite bei Nebel durch die Kombination aus hochauflösendem Scheinwerfer und einem Kamerasystem. Spektrum der Lichttechnik / Karlsruher Institut für Technologie (KIT), Lichttechnisches Institut*. Vol. 32. KIT Scientific Publishing, 2024.

[Kro+23] **M. Kropac, A. Freiding, and D. Ceyhun:** "Application potential and challenges of HD front lighting". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Kub+14] **B. Kubitza, C. Wilks, and S. Schäfer:** "Automatische Justage eines hochauflösenden Scheinwerfers". In: *6. VDI-Fachtagung Optische Technologien in der Fahrzeugtechnik*. 2014.

[Kub+17] **B. Kubitza, U. Venker, S. Knoop, and C. Hüster:** "Method for controlling a light module of a lighting unit of a vehicle, lighting unit, computer program product and computer-readable medium". DE102017120358A1. 2017.

[KW18] **B. Kubitza and C. Wilks:** "Digital light as support for the driver". In: *ATZ worldwide* 120.4 (2018).

[Lec+99] **P. Lecocq, J. Kelada, and A. Kemeny:** "Interactive headlight simulation". In: *Proceedings of the DSC*. Vol. 99. 1999.

[Lef+16] **N. Lefaudeux, A. D. Lamberterie, G. Thin, S. Mbata, T. Canonne, V.-T. Hoang, V. DuBois, and F.-X. Amiel:** "Pixelise light beam control". FR3055981B1. 2016.

[Leh+23] **K. Lehn, M. Gotzes, and F. Klawonn:** *Introduction to Computer Graphics: Using OpenGL and Java*. Springer, 2023.

[Ler+24] **J. Lerch, D. Hoffmann, T. Q. Khanh, M. Hofmann, and A. Guenther:** "From ADB with high-resolution micro LED pixel emitter arrays to camera-controlled adaptive light distributions for different situations in nighttime traffic – a data driven approach". In: *SIA VISION 2024*. 2024.

[LG18] **Y. Lin and V. Grover:** *Using CUDA Warp-Level Primitives*. `https://developer.nvidia.com/blog/using-cuda-warp-level-primitives/` Last accessed on 15.11.2023. 2018.

[Li+20] **M. Li, Y. Ao, and C. Yang:** "Adaptive SpMV/SpMSpV on GPUs for input vectors of varied sparsity". In: *IEEE Transactions on Parallel and Distributed Systems* 32.7 (2020).

[Liu+07] **Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma:** "A survey of content-based image retrieval with high-level semantics". In: *Pattern recognition* 40.1 (2007).

[Loz+06] **A. Loza, L. Mihaylova, N. Canagarajah, and D. Bull:** "Structural similarity-based object tracking in video sequences". In: *2006 9th International Conference on Information Fusion*. IEEE. 2006, pp. 1–6.

[LS02] **J. P. Löwenau and M. H. Strobl:** "Advanced lighting simulation (ALS) for the evaluation of the BMW system adaptive light control (ALC)". In: *SAE Transactions* (2002).

[LS15] **Y. Liu and B. Schmidt:** "LightSpMV: Faster CSR-based sparse matrix-vector multiplication on CUDA-enabled GPUs". In: *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE. 2015.

[LS18] **Y. Liu and B. Schmidt:** "Lightspmv: faster cuda-compatible sparse matrix-vector multiplication using compressed sparse rows". In: *Journal of Signal Processing Systems* 90 (2018).

[Mai+23] **M. Maier, F. Priegler, A. Müller, and C. Weiss:** "DIGITAL LIGHT – A Résumé after 4 Years of Series Production". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Mar+09] **S. Marschner, P. Shirley, and M. Ashikhmin:** *Fundamentals of computer graphics*. AK Peters/CRC Press, 2009.

[Mar24] **Marmoset LLC (Ed.):** *Physically-Based Rendering and You Can Too!* https://marmoset.co/posts/physically-based-rendering-and-you-can-too/ Last accessed on 13.11.2024. 2024.

[Mat24a] **MathWorks, Inc. (Ed.):** *Choosing the Algorithm*. https://de.mathworks.com/help/optim/ug/choosing-the-algorithm.html Last accessed on 2.10.2024. 2024.

[Mat24b] **MathWorks, Inc. (Ed.):** *fmincon Find minimum of constrained nonlinear multivariable function*. https://de.mathworks.com/help/optim/ug/fmincon.html Last accessed on 2.10.2024. 2024.

[Mat24c] **MathWorks, Inc. (Ed.):** *lsqlin Solve constrained linear least-squares problems*. https://de.mathworks.com/help/optim/ug/lsqlin.html Last accessed on 30.08.2024. 2024.

[Mat24d] **MathWorks, Inc. (Ed.):** *particleswarm Particle swarm optimization*. https://de.mathworks.com/help/gads/particleswarm.html Last accessed on 30.08.2024. 2024.

[MB19] **R. Mezari and F. Benamar:** "Adaptation of a motor vehicle high-beam function". WO2020008062A1. 2019.

[MB24] **M. Musheghyan and A. Bieler:** "Investigation of Technologies for Fully Dynamic Ground Projections: Performance Criteria and Use Cases". In: *SIA VISION*. 2024.

[MC11] **E. Mezura-Montes and C. A. C. Coello:** "Constraint-handling in nature-inspired numerical optimization: past, present and future". In: *Swarm and Evolutionary Computation* 1.4 (2011).

[McC04] **S. McConnell:** *Code complete*. Pearson Education, 2004.

[Mei+21] **D. Meister, S. Ogaki, C. Benthin, M. J. Doyle, M. Guthe, and J. Bittner:** "A survey on bounding volume hierarchies for ray tracing". In: *Computer Graphics Forum*. Vol. 40. 2. Wiley Online Library. 2021.

[Mer16] **Mercedes-Benz AG (Ed.):** *Headlamps in the new Mercedes-Benz E-Class: MULTIBEAM LED*. https://www.youtube.com/watch?v=0OJjvYPV3oc Last accessed on 18.10.2023. Cropped input colors above 180 and rescale to make lighting function better visible. 2016.

[Mer18] **Mercedes-Benz AG (Ed.):** *DIGITAL LIGHT The light of the future hits the road*. https://group.mercedes-benz.com/innovation/specials/geneva-2018/digital-light.html Last accessed on 18.10.2023. Cropped input colors above 180 and rescale to make lighting function better visible. 2018.

[Mer21] **Mercedes-Benz AG (Ed.):** *TV-FOOTAGE C-Class Digital Light HD PAL.* `https://media.mercedes-benz.com/(lightbox:video/0edd12e1-1215-4367-b63a-1b255d1fd65e)` Last accessed on 18.10.2023. 2021.

[MH16] **W. J. Matt Pharr and G. Humphreys:** *Physically Based Rendering From Theory to Implementation.* Elsevier, 2016.

[Mic+13] **S. Michenfelder, M. Neumeyer, and C. Neumann:** "Konvertierungsalgorithmus für automobilen Forschungsscheinwerfer". In: *11. Internationales Forum für den lichttechnischen Nachwuchs (Lux Junior).* 2013.

[Mic14] **S. Michenfelder:** "Konzeption, Realisierung und Verifikation eines automobilen Forschungsscheinwerfers auf Basis von Digitalprojektoren". Doctoral thesis. 2014.

[Mil+94] **P. B. Miltersen, S. Subramanian, J. S. Vitter, and R. Tamassia:** "Complexity models for incremental computation". In: *Theoretical Computer Science* 130.1 (1994), pp. 203–236.

[Mim19] **M. Mimoun:** "Method for controlling a lighting device for emitting a non-dazzling beam for lighting the road". US11981250B2. 2019.

[MN14] **S. Michenfelder and C. Neumann:** "Propix (Projektor-Pixellicht) – Aktueller Entwicklungsstand und praktische Erprobung mittels Probandenstudien". In: *6. VDI-Fachtagung Optische Technologien in der Fahrzeugtechnik.* 2014.

[Moi15] **J. Moisel:** "Requirements for Future High Resolution Adb Modules". In: *11th International Symposium on Automotive Lighting (ISAL).* 2015.

[Mül+22] **N. Müller, M. Waldner, and T. Bertram:** "Virtual Development and Optimization of High-Definition Headlights". In: *IEEE International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME).* 2022.

[Mül+23a] **N. Müller, F. Glatzel, M. Waldner, and T. Bertram:** "Virtual Optimization of Matrix Headlights for Improved Automated Object Detection and Energy Efficiency". In: *15th International Symposium on Automotive Lighting (ISAL).* 2023.

[Mül+23b] **N. Müller, M. Waldner, and T. Bertram:** "Usage of Material Properties of 3D Objects for an Improved Illumination by High-Definition Matrix Headlights". In: *Lux junior: 16. Internationales Forum für den lichttechnischen Nachwuchs.* 2023.

[Mül+24a] **N. Müller, M. Waldner, and T. Bertram:** "Concept of a Matrix Headlight Control Loop for Enhanced Object Detection in Automated Driving". In: *Lux junior: 17. Internationales Forum für den lichttechnischen Nachwuchs.* 2024.

[Mül+24b] **N. Müller, M. Waldner, and T. Bertram:** "Improving Computer Vision by Virtual Optimization of Matrix Headlights Using Surface Properties". In: *IEEE International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. 2024.

[Mül+24c] **N. Müller, M. Waldner, and T. Bertram:** "Material-Based Illumination Optimization for Computer Vision in Automated Driving". In: *IEEE International Conference on Applied System Innovation (ICASI)*. 2024.

[Mül+24d] **N. Müller, M. Waldner, and T. Bertram:** "Matrix Headlights and Surface Properties: A New Way to Boost Computer Vision and Save Energy for Automated Vehicles". In: *SIA VISION*. 2024.

[Mül+24e] **N. Müller, M. Waldner, and T. Bertram:** "Verbesserung der Objektdetektion durch Berücksichtigung von Materialeigenschaften bei der Ausleuchtung durch hochauflösende Matrix-Scheinwerfer". In: *VDI Mechatroniktagung*. 2024.

[Mül+25a] **N. Müller, M. Waldner, and T. Bertram:** "Comparison of Different Adaptive Light Distributions for Automated Driving". In: *16th International Symposium on Automotive Lighting (ISAL)*. 2025.

[Mül+25b] **N. Müller, M. Waldner, and T. Bertram:** "Dynamic Matrix Headlight Illumination Strategy for Robust Object Detection in Automated Vehicles". In: *IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*. 2025.

[Mül+25c] **N. Müller, M. Waldner, and T. Bertram:** "Dynamic Sampling-Based Matrix Headlight Control Strategy to Improve Object Detection in Automated Driving". In: *IEEE International Conference on Applied System Innovation (ICASI)*. 2025.

[Mül+25d] **N. Müller, M. Waldner, and T. Bertram:** "Material-based Illumination: Enhanced Object Detection for Automated Driving in Adverse Weather Conditions". In: *Licht: 26. Europäischer Lichtkongress der Lichtgesellschaften Deutschlands, der Niederlande, Österreichs und der Schweiz*. 2025.

[NA16] **J. Neukam and P. Ansorg:** "Environment-dependent calibration of a projection, in particular using a high-resolution vehicle headlight". DE102021202584A1. 2016.

[Neu16] **J. Neukam:** "Lighting device for a motor vehicle for increasing the detectability of an obstacle". EP3465534B1. 2016.

[Nic+08] **J. Nickolls, I. Buck, M. Garland, and K. Skadron:** "Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for?" In: *Queue* 6.2 (2008).

[Nvi01] **Nvidia (Ed.):** "HRAA: High-resolution Antialiasing through Multisampling". In: *Technical Brief* (2001). URL: https://www.evga.com/articles/images/11HRAA.pdf.

[Nvi23a] **Nvidia (Ed.):** *CUDA C++ Best Practices Guide*. https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html Last accessed on 15.11.2023. 2023.

[Nvi23b] **Nvidia (Ed.):** *CUDA C++ Programming Guide*. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html Last accessed on 15.11.2023. 2023.

[Nvi23c] **Nvidia (Ed.):** *CUDA Toolkit Documentation*. https://docs.nvidia.com/cuda/index.html Last accessed on 15.11.2023. 2023.

[Nvi23d] **Nvidia (Ed.):** *cuSPARSE – CUDA Sparse Matrix Library*. https://docs.nvidia.com/cuda/cusparse/ Last accessed on 15.11.2023. 2023.

[Nvi24] **Nvidia (Ed.):** *GeForce RTX 4090 – Specs*. https://www.nvidia.com/de-de/geforce/graphics-cards/40-series/rtx-4090/ Last accessed on 13.11.2024. 2024.

[Pal17] **G. Palubinskas:** "Image similarity/distance measures: what is really behind MSE and SSIM?" In: *International Journal of Image and Data Fusion* 8.1 (2017).

[Pan90] **A. T. Pang:** "Line-drawing algorithms for parallel machines". In: *IEEE Computer graphics and Applications* 10.5 (1990).

[Ped10] **M. E. H. Pedersen:** "Good parameters for particle swarm optimization". In: *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001* (2010).

[Pen55] **R. Penrose:** "A generalized inverse for matrices". In: *Mathematical proceedings of the Cambridge philosophical society*. Vol. 51. 3. Cambridge University Press. 1955.

[Per+19] **A. Perrotin, F. Evanno, J. Doha, M. Hermitte, and Y. Gromfeld:** "Technical & Industrial Strategy for High Efficiency Front Lighting Modules". In: *13th International Symposium on Automotive Lighting (ISAL)*. 2019.

[PK18] **C. Prat and A. Kanj:** "Method for controlling a light pattern and automotive lighting device". EP3672369B1. 2018.

[Por22] **Porsche AG (Ed.):** *Performance leap in light technology*. https://newsroom.porsche.com/en/2022/innovation/porsche-led-main-headlights-with-hd-matrix-beam-light-technology-30770.html Last accessed on 30.03.2023. 2022.

[Por23a] **Porsche (Ed.):** *A Shining Light*. https://newsroom.porsche.com/christophorus/en/2023/407/technique-cayenne-headlights.html Last accessed on 14.06.2024. 2023.

[Por23b] **Porsche AG (Ed.):** *Cayenne: HD-Matrix LED Headlights*. https://press.porsche.co.uk/prod/presse_pag/PressResources.nsf/Content?ReadForm&languageversionid=1442325&hl=modelle-cayenne_e3ll-cayenne_e-hybrid_coupe Last accessed on 20.12.2024. Cropped top and bottom gray background. Added orange arrows and text from [Por23a]. 2023.

[Pot+18] **D. E. Potter, C. Mazuir, R. da Silveira Cabral, P. Furgale, C. F. Cull, D. J. Reetz, and M. J. Milovich:** "System and method for light and image projection". US10558866B2. 2018.

[RA00] **R. H. Simons and A. R. Bean:** *Lighting Engineering: Applied Calculations*. Elsevier Science & Technology Books, 2000.

[Ran+07] **C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis:** "Evaluating mapreduce for multi-core and multiprocessor systems". In: *2007 IEEE 13th International Symposium on High Performance Computer Architecture*. 2007.

[RB85] **J. R. Rice and R. F. Boisvert:** *Solving elliptic problems using ELLPACK*. Vol. 2. Springer, 1985.

[Rei11] **K. Reif:** *Bosch Autoelektrik und Autoelektronik*. Springer, 2011.

[Rez+19] **H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese:** "Generalized Intersection over Union". In: (June 2019).

[Rfp22a] **Rfpro (Ed.):** *rFpro headlight development in simulation*. https://www.youtube.com/watch?v=tey_R6vY85g Last accessed on 15.11.2023. 2022.

[Rfp22b] **Rfpro (Ed.):** *ZKW uses rFpro to develop next-generation headlights*. https://rfpro.com/zkw-uses-rfpro-to-develop-next-generation-headlights/ Last accessed on 31.03.2023. 2022.

[Riv+15] **J. R. V. Rivero, C. Gut, and J. Reim:** "Projection of a Pre-Definable Light Pattern". US20190016256A1. 2015.

[RL18] **J. Roth and S. Lampe:** "System and method for projecting informational content for a motor vehicle". DE102018218038A1. 2018.

[Roe18] **P. Roeckl:** "Optical systems for HD lighting: new paradigms, new methods". In: *SIA VISION*. 2018.

[Roe22] **P. Roeckl:** "Challenges for DRIVER ASSISTANCE PROJECTIONS". In: *SIA VISION*. 2022.

[Ros+09] **R. J. Rost, B. Licea-Kane, D. Ginsburg, J. Kessenich, B. Lichtenbelt, H. Malan, and M. Weiblen:** *OpenGL shading language*. Pearson Education, 2009.

[Ros+18] **E. O. Rosenhahn, H. Seibold, J. Geywitz-Senn, and I. Rutkiewicz:** "Digital light millions of pixels on the road". In: *ATZ worldwide* 120.11 (2018).

[Ros18] **E.-O. Rosenhahn:** "New Systems for Safety and Comfort Improvement by High Resolution Flexibility". In: *SIA VISION*. 2018.

[RR14] **K. Reif and K. Reif:** *Automobilelektronik: Eine Einführung für Ingenieure*. Springer, 2014.

[RR93] **G. Ramalingam and T. Reps:** "A categorized bibliography on incremental computation". In: *Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 1993, pp. 502–510.

[RT21] **N. Rüddenklau and A. Trächtler:** "Method for controlling a pixel headlight system of a motor vehicle". DE102021133055A1. 2021.

[Rüd+19a] **N. Rüddenklau, P. Biemelt, S. Mertin, S. Gausemeier, and A. Trächtler:** "Real-Time Lighting of High-Definition Headlamps for Night Driving Simulation". In: *IARIA SysMea*. Vol. 12. Changed luminous intensity symbol $I$ to $I_\mathrm{v}$ in Fig. 4.1. 2019.

[Rüd+19b] **N. Rüddenklau, P. Biemelt, S. Mertin, S. Gausemeier, and A. Trächtler:** "Simulation-Based Lighting Function Development of High-Definition Headlamps". In: *13th International Symposium on Automotive Lighting – ISAL 2019*. 2019.

[Rüd+22] **N. Rüddenklau, S. Gausemeier, and A. Trächtler:** "Simulative development of Object-Based Lighting". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[Rüd20] **N. Rüddenklau:** "Simulation method for a pixel headlamp system". WO2021224004A1. 2020.

[Rüd22] **N. Rüddenklau:** "Hardware-in-the-Loop-Simulation von HD-Scheinwerfer-Steuergeräten zur Entwicklung von Lichtfunktionen in virtuellen Nachtfahrten". Doctoral thesis. Fakultät für Maschinenbau, Universität Paderborn, 2022.

[RW20] **N. Rüddenklau and M. Weise:** "Method for determining a total light distribution of a pixel spotlight". WO2021214115A1. 2020.

[Rye97] **A. Ryer:** *Light Measurement Handbook*. International Light Technologies, 1997.

[SA18] **U. Schlöder and K. F. Albrecht:** "Headlamp DMD-Technology with High Resolution on the Road". In: *SIA VISION*. 2018.

[Saa03] **Y. Saad:** *Iterative methods for sparse linear systems*. SIAM, 2003.

[Sar+16] **S. Saralajew, S. Mates, K. Stefaniak, and H. Zimmermann:** "Rapid Prototyping of headlamp light distributions". In: *7. VDI-Fachtagung Optische Technologien in der Fahrzeugtechnik*. 2016.

[Sar92] **R. G. Sargent:** "Validation and verification of simulation models". In: *Proceedings of the 24th conference on Winter Simulation*. 1992.

[SB16] **S. Saralajew and G. Benderman:** "Matrix-Beam Algorithm: How fast is possible?" In: *7. VDI-Fachtagung Optische Technologien in der Fahrzeugtechnik*. 2016.

[Sch+09] **C. Schmidt, F. Kalze, and T. Irmscher:** "Illumination Strategies for Dynamic Headlamp Functions like Adaptive and Vertical Cut-Off-Line". In: *9th International Symposium on Automotive Lighting (ISAL)*. 2009.

[Sch+23] **J. Schleusner, H. Blume, and S. Lampe:** "Dynamic Model-Based Safety Margins for High-Density Matrix Headlight Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 24.7 (2023).

[Sch11] **D. Schneider:** "Markierungslicht – eine Scheinwerferlichtverteilung zur Aufmerksamkeitssteuerung und Wahrnehmungssteigerung von Fahrzeugführern". Doctoral thesis. 2011.

[Seg+92] **M. Segal, C. Korobkin, R. Van Widenfelt, J. Foran, and P. Haeberli:** "Fast shadows and lighting effects using texture mapping". In: *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 1992.

[Sei+21] **S. Seidt, T. Winz, R. Sondershaus, and I. Rutkiewicz:** "Soft- and Hardware for High-Definition Automotive Lighting Systems". In: *ATZelectronics worldwide* 16.4 (2021).

[Sei20] **B. Seiser:** *Improving Visibility with DLP Headlights*. `https://www.ti.com/lit/wp/slyy201/slyy201.pdf` Last accessed on 18.10.2023. 2020.

[Sha+23] **R. Shamey et al.:** *Encyclopedia of color science and technology*. Springer, 2023.

[She+08] **A. Sherrod, C. L. (Firm), and I. Course Technology:** *Game Graphics Programming*. Course Technology PTR game development series. Course Technology/Charles River Media/Cengage Learning, 2008.

[Sin+22] **T. Singer, A. Erkan, J. Willmann, D. Hoffmann, and T. Q. Khanh:** "Photometric Characterization and Evaluation of Head-Mounted-Displays for Virtual Night Driving". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[SN18] **S. Strebel and C. Neumann:** "Durchgängige simulative Evaluierung von Lichtfuntionen". In: *8. VDI-Fachtagung Optische Technologien in der Fahrzeugtechnik*. 2018.

[ST95] **T. Scavo and J. Thoo:** "On the geometry of Halley's method". In: *The American mathematical monthly* 102.5 (1995).

[Ste+16] **M. Steinberger, A. Derlery, R. Zayer, and H.-P. Seidel:** "How naive is naive SpMV on the GPU?" In: *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE. 2016.

[Str22] **S. Strebel:** "Ein Beitrag zum simulationsbasierten Test von Lichtfunktionen". Doctoral thesis. Karlsruher Institut für Technologie (KIT), 2022.

[Sul17] **N. A. Suleiman:** "An Algorithm for Computing the Shortest Distance between a Point and Quadratic Bezier Curve". In: *American Journal of Computer Science and Engineering Survey* (2017).

[Sun24] **M. C. Sundermeier:** "Mensch-Maschine-Sehen als Grundlage der Entwicklung optomechatronischer Systeme für kamerabasierte Objekterkennung". Doctoral thesis. 2024.

[Syn21a] **Synopsys,Inc (Ed.):** *LucidDrive Product Introduction | Synopsys*. `https://www.youtube.com/watch?v=IJf3tD-xSeQ` Last accessed on 31.03.2023. 2021.

[Syn21b] **Synopsys,Inc (Ed.):** *Product Update: LucidDrive v2021.12*. `https://www.synopsys.com/optical-solutions/e-news/optical-solutions-enews/osg-enews-dec2021.html` Last accessed on 31.03.2023. 2021.

[Syn23] **Synopsys,Inc (Ed.):** *LucidDrive: Night Driving Simulation*. `https://www.synopsys.com/optical-solutions/lucidshape/luciddrive.html` Last accessed on 31.03.2023. 2023.

[Tam+14] **R. Tamburo, E. Nurvitadhi, A. Chugh, M. Chen, A. Rowe, T. Kanade, and S. G. Narasimhan:** "Programmable automotive headlights". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*. Springer. 2014, pp. 750–765.

[Tec24] **TechnoTeam Bildverarbeitung GmbH (Ed.):** *Screen Photometry*. `https://www.technoteam.de/products/screen_photometry/index_eng.html` Last accessed on 02.07.2024. 2024.

[Ten+22] **T. Tentrup, P. Schuler, S. Schuler, T. Auer, and F. Mai:** "Functional chain tests of dynamic automated resp. autonomous security relevant vehicle driving functions in the frame of periodic technical inspection". In: *Fachtagung Mechatronik 2022*. 2022.

[Thy22] **D. Thy:** *Lighting simulation, a major challenge in the automotive domain.* https://www.avsimulation.com/lighting-simulation/ Last accessed on 31.03.2023. 2022.

[Tka12] **E. R. Tkaczyk:** "Vectorial laws of refraction and reflection using the cross product and dot product". In: *Optics letters* 37.5 (2012), pp. 972–974.

[TM83] **M. Takeda and K. Mutoh:** "Fourier transform profilometry for the automatic measurement of 3-D object shapes". In: *Applied optics* 22.24 (1983).

[Tro+19] **J. Trommer, T. Feil, and M. Wild:** "LED modules for high-resolution matrix headlamps". In: *ATZ worldwide* 121.2 (2019), pp. 26–31.

[Übl21] **R. Übler:** "The head and taillamp concepts of the new BMW 5 Series". In: *SIA VISION*. 2021.

[Ult23] **Ultralytics (Ed.):** *Ultralytics YOLOv8.* https://github.com/ultralytics/ultralytics,https://docs.ultralytics.com/ Last accessed on 11.12.2023. 2023.

[UNE23a] **UNECE (Ed.):** "UN Regulation No. 112 – Headlamps emitting an asymmetrical passing-beam". In: *UN Regulations (1958 Agreement)* (2023). https://unece.org/transport/vehicle-regulations-wp29/standards/addenda-1958-agreement-regulations-101-120 Last accessed on 12.10.2023.

[UNE23b] **UNECE (Ed.):** "UN Regulation No. 123 – Adaptive front-lighting systems (AFS)". In: *UN Regulations (1958 Agreement)* (2023). https://unece.org/transport/vehicle-regulations-wp29/standards/addenda-1958-agreement-regulations-121-140 Last accessed on 12.10.2023.

[UNE23c] **UNECE (Ed.):** "UN Regulation No. 149 – Road Illumination Devices (RID)". In: *UN Regulations (1958 Agreement)* (2023). https://unece.org/transport/vehicle-regulations-wp29/standards/addenda-1958-agreement-regulations-141-160 Last accessed on 12.10.2023.

[UNE23d] **UNECE (Ed.):** "UN Regulation No. 48 – Installation of lighting and light-signalling devices". In: *UN Regulations (1958 Agreement)* (2023). https://unece.org/transport/vehicle-regulations-wp29/standards/addenda-1958-agreement-regulations-41-60 Last accessed on 12.10.2023.

[Uni23] **Unity Technologies (Ed.):** *Unity.* https://unity.com/.com Last accessed on 31.03.2023. 2023.

[Uni24] **Unity Technologies (Ed.):** *Light component reference – HDRP 17 Unity.* https://docs.unity3d.com/Packages/com.unity.render-

pipelines.high-definition@17.0/manual/reference-light-component.html Last accessed on 21.06.2024. 2024.

[Ups89] **S. Upstill:** *RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[Url+21] **M. Urlaub, B. Spinger, H. B. R. Bonne, D. Vanderhaeghen, and N. Lesch:** "Micro-LED and Matrix-LED, a hybrid light source architecture for high resolution headlighting". In: *SIA VISION*. 2021.

[Usó+99] **C. Usón, V. Eclairage, and P. Bouchon:** "A physically exact real time simulator for car headlight". In: *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*. IEEE. 1999, p. 76.

[Val21] **Valeo (Ed.):** *PictureBeam Monolithic, Adaptative headlights*. https://www.valeo.com/en/picturebeam-monolithic-adaptive-headlights/ Last accessed on 25.06.2024. 2021.

[VIR23] **VIRES Simulationstechnologie GmbH (Ed.):** *Headlighting Simulation – VTD*. http://vires.mscsoftware.com/applications/headlighting Last accessed on 31.03.2023. 2023.

[Vog23] **S. Vogel:** "Adaptive Scheinwerferlichtverteilungen für unterschiedliche Straßenbeleuchtungsszenarien". Doctoral thesis. Technische Universität Berlin, 2023.

[Vol18] **Volkswagen AG (Ed.):** *IQ.Light – LED Matrix Headlights*. https://www.volkswagen-newsroom.com/en/images/detail/volkswagen-touareg-25173 Last accessed on 25.06.2024. 2018.

[Wal+07] **I. Wald, S. Boulos, and P. Shirley:** "Ray tracing deformable scenes using dynamic bounding volume hierarchies". In: *ACM Transactions on Graphics (TOG)* 26.1 (2007).

[Wal+19a] **M. Waldner, M. Krämer, and T. Bertram:** "Hardware-In-The-Loop-Simulation of the Light Distribution of Automotive Matrix-LED-Headlights". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2019.

[Wal+19b] **M. Waldner, M. Krämer, and T. Bertram:** "Optimierung des Entwicklungsprozesses von Scheinwerfersystemen durch mechanische Nachbildung der Fahrdynamik". In: *VDI Mechatroniktagung*. 2019.

[Wal+20] **M. Waldner, M. Krämer, and T. Bertram:** "Digitization of Matrix-Headlights That Move as in the Real Test Drive". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2020.

[Wal+22a] **M. Waldner, N. Müller, and T. Bertram:** *Energy-Efficient Illumination by Matrix Headlamps for Nighttime Automated Object Detection*.

IEEE International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). Presentation. 2022.

[Wal+22b] **M. Waldner, N. Müller, and T. Bertram:** "Energy-Efficient Illumination by Matrix Headlamps for Nighttime Automated Object Detection". In: *International Journal of Electrical and Computer Engineering Research* 2.3 (2022).

[Wal+22c] **M. Waldner, N. Müller, and T. Bertram:** "Flexible Modeling of High-Definition Matrix Headlights". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2022.

[Wal+23a] **M. Waldner, N. Müller, and T. Bertram:** "Different Approaches to Optimize High-Definition Matrix Headlights to Improve Computer Vision". In: *Lux junior: 16. Internationales Forum für den lichttechnischen Nachwuchs*. 2023.

[Wal+23b] **M. Waldner, N. Müller, and T. Bertram:** "Lessons Learned in the Development of a High-Definition Matrix Headlight Real-Time Simulator". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Wal+24a] **M. Waldner, N. Müller, and T. Bertram:** "Efficient Real-Time Control of Matrix Headlights". In: *IEEE International Conference on Applied System Innovation (ICASI)*. 2024.

[Wal+24b] **M. Waldner, N. Müller, and T. Bertram:** "Feedback Control of the Projected Symbol of Matrix Headlights on Uneven Surfaces". In: *IEEE International Conference on Control and Robotics Engineering (ICCRE)*. 2024.

[Wal+24c] **M. Waldner, N. Müller, and T. Bertram:** "Matrix Headlight Control Loop for Undistorted Symbol Projection". In: *Lux junior: 17. Internationales Forum für den lichttechnischen Nachwuchs*. 2024.

[Wal+25a] **M. Waldner, N. Müller, and T. Bertram:** "Hybrid Rapid Prototyping of Lighting Functions for Humans and Machines". In: *ATZ worldwide* 2-3 (2025).

[Wal+25b] **M. Waldner, N. Müller, and T. Bertram:** "Hybrides Rapid Prototyping von Lichtfunktionen für Mensch und Maschine". In: *ATZ – Automobiltechnische Zeitschrift* 2-3 (2025).

[Wal+25c] **M. Waldner, N. Müller, and T. Bertram:** "Zone-Based Matrix Headlight Feedback Control for Enhanced Object Detection in Automated Driving". In: *16th International Symposium on Automotive Lighting (ISAL)*. Equal contribution of the first two authors. 2025.

[Wan+04] **Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli:** "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[WB19a] **M. Waldner and T. Bertram:** "Digitalisierung von Lichtverteilungen zur Hardware-in-the-LoopVerifikation und Validierung von Matrixscheinwerfern". In: *9. VDI/VDE-Fachtagung Autoreg.* 2019.

[WB19b] **M. Waldner and T. Bertram:** "Evaluation of the Light Distribution of a Matrix-Headlight with a Hardware-in-the-Loop-Simulation". In: *13th International Symposium on Automotive Lighting (ISAL).* 2019.

[WB20] **M. Waldner and T. Bertram:** "Simulation of High-Definition Pixel-Headlights". In: *15th International Symposium on Visual Computing (ISVC).* 2020.

[WB21a] **M. Waldner and T. Bertram:** "Optimal Real-time Digitization of Matrix-Headlights". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).* 2021.

[WB21b] **M. Waldner and T. Bertram:** "Parallelisierte Steuerung hochauflösender Matrix-Scheinwerfer". In: *7. IFToMM D-A-CH.* 2021.

[WB21c] **M. Waldner and T. Bertram:** "Virtual Evaluation of High-Definition Lighting Functions for Various Beam Patterns for Rapid-Prototyping". In: *Lux junior: 15. Internationales Forum für den lichttechnischen Nachwuchs.* 2021.

[WB22] **M. Waldner and T. Bertram:** "Feedforward Control of HD-Headlights for Automated Driving". In: *14th International Symposium on Automotive Lighting (ISAL).* 2022.

[WH20] **D. Walter and B. Hummel:** "Method for operating a matrix headlight system of a vehicle and matrix headlight system". DE102020119673B3. 2020.

[Wik24] **Wikipedia (Ed.):** *Supersampling.* `https://en.wikipedia.org/wiki/Supersampling` Last accessed on 25.06.2024. 2024.

[Win+15] **H. Winner, S. Hakuli, and G. Wolf:** *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort.* Springer, 2015.

[Win20] **T. Winz:** "Light Distribution by Software". In: *ATZelectronics worldwide* 15.10 (2020).

[Wol+23] **H. Wolfgang, D. Baccarin, and U. Schubert:** "Advanced Stereo Night Vision with Headlamps". In: *15th International Symposium on Automotive Lighting (ISAL).* 2023.

[Won+24] **Y. Won, M. Kim, K. Park, J. Lim, and H. Jeong:** "A Study on the improvement of ADB function for HD Micro LED Headlamp". In: *SIA VISION 2024.* 2024.

[Wör+07] **B. Wördenweber, J. Wallaschek, P. Boyce, and Hoffman, D.D.:** *Automotive Lighting and Human Vision.* Springer, 2007.

[WW00] **G. Wyszecki and W. S. Stiles:** *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley Series in Pure and Applied Optics. Wiley, 2000.

[Xia+23] **G. Xiao, C. Yin, T. Zhou, X. Li, Y. Chen, and K. Li:** "A survey of accelerating parallel sparse linear algebra". In: *ACM Computing Surveys* 56.1 (2023).

[Yar23a] **S. Yargeldi:** "Compensation of Vehicle Movement and Surface Roughness for Headlamp Road Projections". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Yar23b] **S. Yargeldi:** "Umfeldbezogene Stabilisierung von Fahrzeugscheinwerferprojektionen". Doctoral thesis. Eberhard Karls Universität, 2023.

[Yeo21] **R. Yeo:** *The Science of Goniophotometry*. `https://sphereoptics.de/en/wp-content/uploads/sites/3/2021/07/Technical-Note-The-Science-of-Goniophotometry-Pro-Lite-June-2021.pdf` Last accessed on 02.07.2024. 2021.

[Zac30] **M. Zacharias:** *Elementargeometrie der Ebene und des Raumes*. 1930.

[Zim07] **C. Zimmermann:** "Method for a headlight with dynamic cornering light and adaptive behavior". DE102007012834B4. 2007.

[ZP09] **J. Zhao and J. Pjesivac-Grbovic:** *MapReduce: The programming model and practice*. `https://research.google.com/archive/papers/mapreduce-sigmetrics09-tutorial.pdf` Last accessed on 20.12.2024. 2009.

## Own Publications

[Gla+25] **F. Glatzel, N. Müller, M. Waldner, and T. Bertram:** "Energy-Efficient Pedestrian Marker Light Based on Pose Probability for Computer Vision". In: *16th International Symposium on Automotive Lighting (ISAL)*. Best Poster Award. 2025.

[Mül+25a] **N. Müller, M. Waldner, and T. Bertram:** "Comparison of Different Adaptive Light Distributions for Automated Driving". In: *16th International Symposium on Automotive Lighting (ISAL)*. 2025.

[Mül+25b] **N. Müller, M. Waldner, and T. Bertram:** "Dynamic Matrix Headlight Illumination Strategy for Robust Object Detection in Automated Vehicles". In: *IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*. 2025.

[Mül+25c] **N. Müller, M. Waldner, and T. Bertram:** "Dynamic Sampling-Based Matrix Headlight Control Strategy to Improve Object Detection in Automated Driving". In: *IEEE International Conference on Applied System Innovation (ICASI)*. 2025.

[Mül+25d] **N. Müller, M. Waldner, and T. Bertram:** "Material-based Illumination: Enhanced Object Detection for Automated Driving in Adverse Weather Conditions". In: *Licht: 26. Europäischer Lichtkongress der Lichtgesellschaften Deutschlands, der Niederlande, Österreichs und der Schweiz*. 2025.

[Wal+25a] **M. Waldner, N. Müller, and T. Bertram:** "Hybrid Rapid Prototyping of Lighting Functions for Humans and Machines". In: *ATZ worldwide* 2-3 (2025).

[Wal+25b] **M. Waldner, N. Müller, and T. Bertram:** "Hybrides Rapid Prototyping von Lichtfunktionen für Mensch und Maschine". In: *ATZ – Automobiltechnische Zeitschrift* 2-3 (2025).

[Wal+25c] **M. Waldner, N. Müller, and T. Bertram:** "Zone-Based Matrix Headlight Feedback Control for Enhanced Object Detection in Automated Driving". In: *16th International Symposium on Automotive Lighting (ISAL)*. Equal contribution of the first two authors. 2025.

[Kau+24] **R. Kauschke, B. Kubitza, C. Wilks, M. Waldner, N. Müller, T. Bertram, M. Grünke, and A. Maroke:** "AHEAD-Light Distributions for camera-based automated driving – in simulation and validated in real world scenarios". In: *SIA VISION*. 2024.

[Mül+24b] **N. Müller, M. Waldner, and T. Bertram:** "Concept of a Matrix Headlight Control Loop for Enhanced Object Detection in Automated Driving". In: *Lux junior: 17. Internationales Forum für den lichttechnischen Nachwuchs*. 2024.

[Mül+24c] **N. Müller, M. Waldner, and T. Bertram:** "Improving Computer Vision by Virtual Optimization of Matrix Headlights Using Surface Properties". In: *IEEE International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. 2024.

[Mül+24d] **N. Müller, M. Waldner, and T. Bertram:** "Material-Based Illumination Optimization for Computer Vision in Automated Driving". In: *IEEE International Conference on Applied System Innovation (ICASI)*. 2024.

[Mül+24e] **N. Müller, M. Waldner, and T. Bertram:** "Matrix Headlights and Surface Properties: A New Way to Boost Computer Vision and Save Energy for Automated Vehicles". In: *SIA VISION*. 2024.

[Mül+24f] **N. Müller, M. Waldner, and T. Bertram:** "Verbesserung der Objektdetektion durch Berücksichtigung von Materialeigenschaften bei der Ausleuchtung durch hochauflösende Matrix-Scheinwerfer". In: *VDI Mechatroniktagung*. 2024.

[Wal+24a] **M. Waldner, N. Müller, and T. Bertram:** "Efficient Real-Time Control of Matrix Headlights". In: *IEEE International Conference on Applied System Innovation (ICASI)*. 2024.

[Wal+24b] **M. Waldner, N. Müller, and T. Bertram:** "Feedback Control of the Projected Symbol of Matrix Headlights on Uneven Surfaces". In: *IEEE International Conference on Control and Robotics Engineering (ICCRE)*. 2024.

[Wal+24d] **M. Waldner, N. Müller, and T. Bertram:** "Matrix Headlight Control Loop for Undistorted Symbol Projection". In: *Lux junior: 17. Internationales Forum für den lichttechnischen Nachwuchs*. 2024.

[Kau+23] **R. Kauschke, B. Kubitza, C. Wilks, M. Waldner, N. Müller, T. Bertram, and M. Grünke:** "AHEAD – Innovative Lighting for Automated Driving – Lighting Strategies for a Better Camera Detection". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Mül+23a] **N. Müller, F. Glatzel, M. Waldner, and T. Bertram:** "Virtual Optimization of Matrix Headlights for Improved Automated Object Detection and Energy Efficiency". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Mül+23c] **N. Müller, M. Waldner, and T. Bertram:** "Usage of Material Properties of 3D Objects for an Improved Illumination by High-Definition Matrix Headlights". In: *Lux junior: 16. Internationales Forum für den lichttechnischen Nachwuchs*. 2023.

[Wal+23a] **M. Waldner, N. Müller, and T. Bertram:** "Different Approaches to Optimize High-Definition Matrix Headlights to Improve Computer Vision". In: *Lux junior: 16. Internationales Forum für den lichttechnischen Nachwuchs*. 2023.

[Wal+23b] **M. Waldner, N. Müller, and T. Bertram:** "Lessons Learned in the Development of a High-Definition Matrix Headlight Real-Time Simulator". In: *15th International Symposium on Automotive Lighting (ISAL)*. 2023.

[Kau+22] **R. Kauschke, M. Waldner, N. Müller, T. Bertram, and M. Grünke:** "Ist automatisiertes Fahren mit kamera-optimierten Lichtfunktionen besser möglich? Top-Down-Entwicklung optimierter Lichtverteilungen für das automatisierte Fahren". In: *9. VDI-Tagung Optische Technologien in der Fahrzeugtechnik*. 2022.

[Mül+22] **N. Müller, M. Waldner, and T. Bertram:** "Virtual Development and Optimization of High-Definition Headlights". In: *IEEE International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 2022.

[WB22] **M. Waldner and T. Bertram:** "Feedforward Control of HD-Headlights for Automated Driving". In: *14th International Symposium on Automotive Lighting (ISAL)*. 2022.

[Wal+22c] **M. Waldner, N. Müller, and T. Bertram:** "Energy-Efficient Illumination by Matrix Headlamps for Nighttime Automated Object Detection". In: *International Journal of Electrical and Computer Engineering Research* 2.3 (2022).

[Wal+22d] **M. Waldner, N. Müller, and T. Bertram:** "Flexible Modeling of High-Definition Matrix Headlights". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2022.

[Kau+21a] **R. Kauschke, N. Rüddenklau, F. Ernst, and M. Waldner:** "Effizientere Entwicklung von Scheinwerfersystemen". In: *ATZ – Automobiltechnische Zeitschrift* 123.2 (2021).

[Kau+21b] **R. Kauschke, N. Rüddenklau, F. Ernst, and M. Waldner:** "More Efficient Development of Headlamp Systems". In: *ATZ worldwide* 123.2 (2021).

[Sta+21] **N. Stannartz, L. Jui-Lin, M. Waldner, and T. Bertram:** "Semantic Landmark-based HD Map Localization Using Sliding Window Max-Mixture Factor Graphs". In: *IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021.

[WB21a] **M. Waldner and T. Bertram:** "Optimal Real-time Digitization of Matrix-Headlights". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2021.

[WB21b] **M. Waldner and T. Bertram:** "Parallelisierte Steuerung hochauflösender Matrix-Scheinwerfer". In: *7. IFToMM D-A-CH*. 2021.

[WB21c] **M. Waldner and T. Bertram:** "Virtual Evaluation of High-Definition Lighting Functions for Various Beam Patterns for Rapid-Prototyping". In:

*Lux junior: 15. Internationales Forum für den lichttechnischen Nachwuchs.* 2021.

[WB20] **M. Waldner and T. Bertram:** "Simulation of High-Definition Pixel-Headlights". In: *15th International Symposium on Visual Computing (ISVC).* 2020.

[Wal+20a] **M. Waldner, M. Krämer, and T. Bertram:** "Digitization of Matrix-Headlights That Move as in the Real Test Drive". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).* 2020.

[Jan+19] **P. Janke, C. Lankeit, M. Krämer, and M. Waldner:** "Intelligenz im Scheinwerfer: Künftige LED-Systeme heilen sich selbst". In: *Automobil-Elektronik 04-05/2019* (2019).

[WB19a] **M. Waldner and T. Bertram:** "Digitalisierung von Lichtverteilungen zur Hardware-in-the-LoopVerifikation und Validierung von Matrixscheinwerfern". In: *9. VDI/VDE-Fachtagung Autoreg.* 2019.

[WB19b] **M. Waldner and T. Bertram:** "Evaluation of the Light Distribution of a Matrix-Headlight with a Hardware-in-the-Loop-Simulation". In: *13th International Symposium on Automotive Lighting (ISAL).* 2019.

[Wal+19a] **M. Waldner, M. Krämer, and T. Bertram:** "Hardware-In-The-Loop-Simulation of the Light Distribution of Automotive Matrix-LED-Headlights". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).* 2019.

[Wal+19b] **M. Waldner, M. Krämer, and T. Bertram:** "Optimierung des Entwicklungsprozesses von Scheinwerfersystemen durch mechanische Nachbildung der Fahrdynamik". In: *VDI Mechatroniktagung.* 2019.

## Additional Presentations and Poster Contributions

[GW24] **G. Gräfe and M. Waldner:** *Umgebungsmodell zur Scheinwerfersimulation für das automatisierte Fahren.* VDI Mechatroniktagung. Keynote. 2024.

[Mül+24a] **N. Müller, M. Waldner, and T. Bertram:** *Virtual Optimization of the Illumination of Matrix Headlamps for Automated Driving.* 19. DortmunderAutoTag (DAT). Poster. 2024.

[Mül+23b] **N. Müller, M. Waldner, and T. Bertram:** *Virtual Optimization of the Illumination of Matrix Headlamps for Automated Driving.* 18. DortmunderAutoTag (DAT). Poster. 2023.

[Wal+22a] **M. Waldner, N. Müller, and T. Bertram:** *Energy-Efficient Illumination by Matrix Headlamps for Nighttime Automated Object Detection*. IEEE International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). Presentation. 2022.

[Wal+22b] **M. Waldner, N. Müller, and T. Bertram:** *Erforschung innovativer Lichtfunktionen hochauflösender Matrix-Scheinwerfer für das (teil-)automatisierte Fahren*. 17. DortmunderAutoTag (DAT). Poster. 2022.

[Wal+20b] **M. Waldner, M. Krämer, and T. Bertram:** *Echtzeit Digitalisierung der adaptiven Ausleuchtung von Matrix-Scheinwerfern*. 15. DortmunderAutoTag (DAT). Presentation. 2020.

[Wal+19c] **M. Waldner, M. Krämer, and T. Bertram:** "Hardware-in-the-Loop-Prüfstand zur Evaluation der Lichtverteilung hochauflösender Matrixscheinwerfer". In: *14. DortmunderAutoTag (DAT)*. 2019.

[Wal+18] **M. Waldner, M. Krämer, and T. Bertram:** "Entwicklungsmethoden für Matrix-Scheinwerfer–Konzept eines Hardware-in-the-Loop-Prüfstands zur Validierung von Lichtfunktionen". In: *13. DortmunderAutoTag (DAT)*. 2018.

## Award

- **M. Waldner, N. Müller, and T. Bertram:** *Feedback Control of the Projected Symbol of Matrix Headlights on Uneven Surfaces*. Best Presentation Award: IEEE International Conference on Control and Robotics Engineering (ICCRE). 2024.

## Supervised Theses

- *Automated rating of the light distributions of automotive headlights.* Master's thesis. 2019.
- *Simulation and Control of the Symbol Projection of High-resolution Headlights.* Master's thesis. 2019.
- *Digitale Leuchtweitenregulierung von Pixescheinwerfern.* Bachelor's thesis. 2020.
- *Virtuelle Entwicklung und Evaluation hochauflösender Lichtfunktionen.* Master's thesis. 2022.
- *Evaluierung des Einflusses von Beleuchtungs- und Trainingsparametern auf die Objekterkennung von künstlichen neuronalen Netzen bei virtuellen Nachtfahrten.* Bachelor's thesis. 2024.
- *Evaluierung des Einflusses von Beleuchtungs- und Trainingsparametern auf die Objekterkennung von künstlichen neuronalen Netzen bei virtuellen Nachtfahrten.* Bachelor's thesis. 2024.
- *Verbesserung der Objektdetektion bei Nacht durch Deep Learning.* Scientific project work. 2024.
- *Derivation of Objective Metrics for Human Perception of High-Definition Matrix Headlight Distributions for Automated Driving.* Master's thesis. 2025.

## Supervised Student Project Groups

- *Robust Object Detection at Night for Automated Driving.* Project group. 2024.

# A

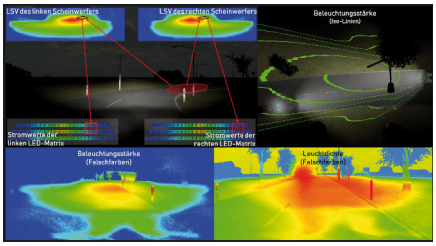
# Appendix: Matrix Headlight Simulation

## A.1. Missing Shadows in Headlamp Simulators


(a) Unreal 5.2. SOL with headlight shadows.

(b) Ansys VRXPERIENCE [For21].

(c) AVSimulation SCANeR[AVS22b].



(d) Heinz Nixdorf Institute [HNI19].

(e) Synopsys Luciddrive [Syn21a].

(f) rFPro [Rfp22a].

(g) Hexagon VTD [ET 21]

Figure A.1.: Many real-time headlight simulators do not simulate the headlight shadows. The missing cast shadows create a homogeneous illumination behind delineators and road signs.

## A.2. Complete System Architecture of the Matrix Headlight Simulator[‡t]

Sec. A.2 and Fig. 4.1 and 4.2 provide only a simplified overview of the SOL architecture. The system encompasses additional components and interconnections, resulting in a more complex system. Fig. A.2 offers a more detailed representation of the system's critical components and their interactions.

The data flow within SOL begins and ends with the user (human or machine algorithm) and monitors. The user commands are input, or predefined instructions are executed. For instance, a driver might use the steering wheel or other controls to signal a turn, which the system must account for when adjusting the headlight direction and intensity.

The scenario manager, embedded within the 3D rendering engine, orchestrates the overall simulation. Its duties include initializing the virtual world, managing traffic objects, passing inputs to other modules and ensuring the correct execution of lighting functions. This manager coordinates the virtual evaluation scenario and various system components. The 3D rendering engine is pivotal in simulating and rendering the virtual world, including traffic objects, the ego-vehicle (i.e., the vehicle equipped with the matrix headlight) and various environment lighting scenarios. The ego-vehicle, which traverses the virtual world, comprises numerous virtual spotlights, each representing a module of the matrix headlight. The ego-vehicle's position, speed and orientation are continuously updated based on input from the scenario manager.
A virtual sensor system captures data on the surrounding environment, including other vehicles, obstacles and road conditions. This data is then fed into the matrix headlight controller, which is used to determine the optimal beam pattern.

At the system's core, the matrix headlight simulation and control module creates, manages and optimizes the headlamp's dynamic LIDs. The controller consists of two parts: the higher-level lighting planer defines the optimal lighting for the current environment and the lower-level pixel utilization controller adjusts the pixel utilization to match the plan with SSC.
The pixel utilization matrix is sent to the headlight simulation module, which uses SpMV to compute and create the desired LID. This LID is then loaded into the 3D rendering engine as a light texture, projected into the virtual world by a spotlight.

The headlamp hardware communication module is a crucial aspect of the system, which translates the calculated pixel utilization and other control data into commands for the physical headlight modules. These commands include adjustments to the beam angle, maximal luminous intensity and beam pattern, ensuring that the headlights adapt dynamically to changing driving conditions.

The real matrix headlight test stand could incorporate a reflection scene and image processing capabilities. A real camera at the test stand captures these reflections. It feeds the data back into the system as a virtual headlight, allowing for further refinement of the beam patterns. This feedback loop enables continuous improvement and verification of the simulated beam patterns against real-world performance.

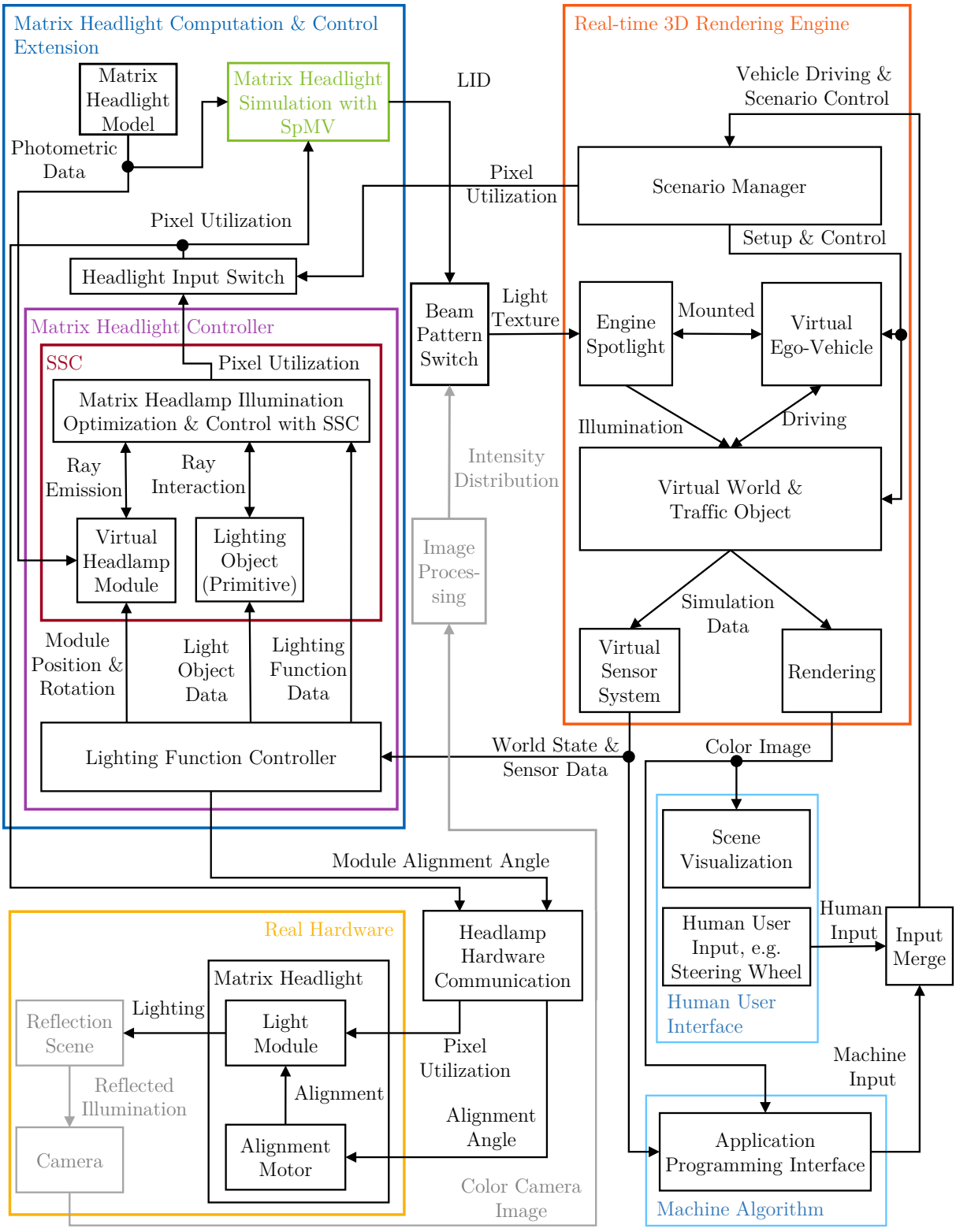



Figure A.2.: Detailed schematic representation of the SOL architecture. Colored blocks in this figure correspond to the system modules depicted in Fig. 4.1 and 4.2, illustrating the integration of various components within SOL.

## A.3. Computing Performance of the Sparse Matrix-Vector Multiplication[‡t]

The evaluation of computational time of SpMV for automotive headlight simulations is done with the following empirical investigation [WB20]. The performance of CSR format for headlight simulation is evaluated for different texture resolutions. The subsequent analysis, including quantitative measurements, tabulated results and graphical representations, is published in Waldner and Bertram [WB20].

The experimental methodology uses a virtual headlamp configuration with symmetrical opening angles of $\pm 8°$ along the horizontal axis and $\pm 4°$ in the vertical direction. To maintain geometric consistency across multiple headlight configurations, the investigation implements a fixed aspect ratio of 2:1 between columns and rows and an overlap of approximately 25 % between adjacent pixels. The light texture is configured as a square matrix, with its virtual spotlight characterized by a symmetrical opening angle of $\pm 8.1°$. The study encompasses five distinct headlight resolutions, spanning a pixel number $n_\mathrm{p}$ range from $45 \times 90 = 4.05 \cdot 10^3$ pixels to $870 \times 1{,}740 = 1{,}513.8 \cdot 10^3$ pixels.
The SpMV computations are executed within a dedicated CUDA 10.1 C++ module, operating independently of Unity or Unreal environments to isolate pure computational performance. The experimental hardware configuration consists of an Intel i9-9980XE CPU coupled with a Nvidia 2080 Ti GPU.
The investigation focuses exclusively on texture generation processes to ensure precise performance measurement of the SpMV operations, deliberately excluding auxiliary 3D visualization overhead. The experimental protocol involves stochastic random generation of utilization values, with explicit CUDA stream synchronization implemented before and after each measurement interval to ensure temporal accuracy.

The initial experimental phase maintains a constant texture resolution $n_\mathrm{t}$ of $2{,}048^2$ texels, ensuring complete illumination of at least one texel per pixel. This configuration enables assessment of SpMV computational efficiency as a function of increasing headlight pixel number and matrix columns while maintaining constant output dimensionality. Table A.1 presents a statistical analysis derived from 500 iterations across each headlight configuration and shows the extrema, the mean, Standard Deviation (StD), median, 5 % quantile (quan.) and 95 % quantile (quan.) of the measured computation times.
The empirical results demonstrate stability in median computation time across the entire spectrum of headlight resolutions, exhibiting a confined range from 429.7 µs to 480.3 µs. This computational stability provides strong evidence for the efficient scaling of SpMV under increasing headlight resolution while maintaining constant output texture dimensions.

The second experimental phase implements an adaptive texture resolution strategy, where the texture dimensions are dynamically adjusted according to the number of pixels $n_\mathrm{p}$ of each headlamp. The adaptive resolution is determined through the mathematical formulation $n_\mathrm{t} = 64\lceil \lfloor 2.2 n_\mathrm{p} \rceil / 64 \rceil$, ensuring consistent discretization quality across all pixel configurations. Table A.2 presents the results of this adaptive resolution methodology.
The adaptive resolution results reveal a distinct correlation between texture resolution and computational complexity. As $n_\mathrm{t}$ increases to accommodate higher $n_\mathrm{p}$, the median

Table A.1.: Computing times in µs over 500 iterations with constant $n_t$ [WB20].

| $n_p$ | $n_t$ | Min | Max | Mean | StD | Median | 5 % Quan. | 95 % Quan. |
|---|---|---|---|---|---|---|---|---|
| $4.05 \cdot 10^3$ | $2{,}048^2$ | 400.1 | 866.5 | 447 | 49.4 | 429.7 | 410.4 | 531.6 |
| $31.25 \cdot 10^3$ | $2{,}048^2$ | 403.6 | 1,074.1 | 460 | 80.1 | 432.9 | 411.5 | 556.1 |
| $304.2 \cdot 10^3$ | $2{,}048^2$ | 457.0 | 782.2 | 491 | 37.1 | 480.3 | 465.1 | 579.3 |
| $897.8 \cdot 10^3$ | $2{,}048^2$ | 420.2 | 875.3 | 472 | 57.5 | 454.9 | 433.1 | 561.9 |
| $1{,}513.8 \cdot 10^3$ | $2{,}048^2$ | 434.1 | 761.2 | 467 | 36.4 | 454.9 | 439.3 | 554.5 |

computation time exhibits a monotonic increase from 98.7 µs for the minimal resolution to 1,646.3 µs for the maximum resolution, demonstrating a clear scaling relationship between computational resources and problem size.

Table A.2.: Computing times in µs over 500 iterations with adaptive texture resolution [WB20].

| Pixels | $n_t$ | Min | Max | Mean | StD | Median | 5 % Quan. | 95 % Quan. |
|---|---|---|---|---|---|---|---|---|
| $4.05 \cdot 10^3$ | $256^2$ | 73.7 | 261.4 | 112 | 37.5 | 98.7 | 81.2 | 197.2 |
| $31.25 \cdot 10^3$ | $576^2$ | 104.4 | 265.0 | 145 | 33.3 | 135.1 | 115.4 | 224.7 |
| $304.2 \cdot 10^3$ | $1{,}728^2$ | 372.6 | 706.4 | 408 | 35.9 | 397.8 | 381.5 | 493.1 |
| $897.8 \cdot 10^3$ | $3{,}008^2$ | 820.6 | 1,445.5 | 857 | 44.0 | 845.9 | 828.2 | 941.2 |
| $1{,}513.8 \cdot 10^3$ | $3{,}840^2$ | 1,616.6 | 2,719.1 | 1,665 | 97.7 | 1,646.3 | 1,630.0 | 1,744.7 |

A normalized performance metric facilitates quantitative comparison between constant and adaptive texture resolution strategies. This dimensionless metric is derived by normalizing the median computation time concerning a baseline time and optional to the number of texels. Fig. A.3 provides a graphical representation of the normalized performance characteristics for both constant and adaptive texture resolutions across the spectrum of headlight configurations.

The observation of nearly constant computation time across increasing $n_p$ provides strong empirical evidence for the efficiency of the SpMV operation under conditions of fixed output dimensions.

The approximately linear scaling relationship between computation time and number of texel aligns with the finite number of parallel GPU threads. This linear dependency suggests that the computational complexity exhibits direct proportionality concerning the output texture dimensions.

A notable computational overhead is observed for configurations using minimal texture resolutions, which can be attributed to the fixed costs associated with GPU thread initialization and management. This initialization overhead becomes progressively less significant as the problem size increases.

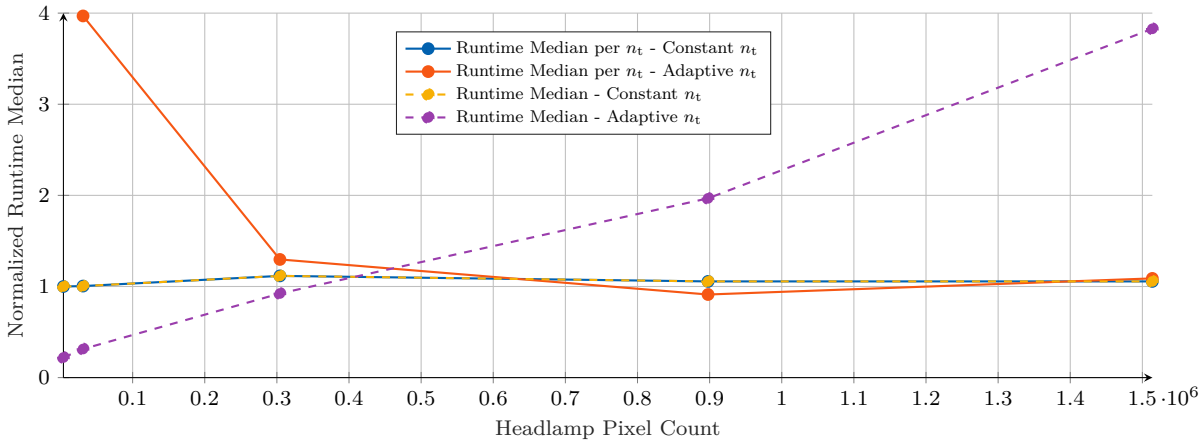



Figure A.3.: Normalized performance comparison with constant and adaptive texture resolutions. The normalization value is 430 µs for constant texture size and 430 µs/2,048$^2$ for adaptive size [WB20].

## A.4. Source Code of Incremental Update

```
// CUDA Kernel to compute pixel intensity differences from current to last frame
__global__ void ComputeDiffs(int* pixIdx, int numPix, float* diffBuf, float* curPow, float* prevPow){
int tid = blockDim.x * blockIdx.x + threadIdx.x;
if (tid >= numPix) return;

diffBuf[tid] = abs(curPow[tid] - prevPow[tid]);
prevPow[tid] = curPow[tid];
}

// CUDA Kernel to update simulation flags based on pixel differences
__global__ void UpdateSimFlags(int* pixIdx, int numPix, float* diffBuf, bool* simFlag, const int* pixOff, const int* pixData){
int tid = blockDim.x * blockIdx.x + threadIdx.x;
if (tid >= numPix) return;
int pIdx = pixIdx[tid];

if (diffBuf[pIdx] > 0.001f) {
  for (int j = pixOff[pIdx]; j < pixOff[pIdx + 1]; j++) {
    simFlag[pixData[j]] = true;
}}}

// CUDA Kernel to selectively include light indices in the next simulation step
__global__ void FilterLights(const bool* simFlag, const int* lightIdx, int* filtLightIdx, int numLights){
int tid = blockDim.x * blockIdx.x + threadIdx.x;
if (tid >= numLights) return;

if (simFlag[tid]) {
  filtLightIdx[tid] = lightIdx[tid];
} else {
  filtLightIdx[tid] = numLights + 1; // Mark as invalid
```

```
}}

void MxBeam::Create2(float* rms, float* rms2) {
using namespace std::chrono;

// Initialize CUDA grid and block sizes for full update
int minGridSize = 1;
int count = numLightIdx;
blockSize[0] = minf((float)CudaProp()->maxThreadsPerBlock, count);
if (blockSize[0] < 10) blockSize[0] = CudaProp()->maxThreadsPerBlock;
gridSize[0] = (count + blockSize[0] - 1) / blockSize[0];
count = numPix;
blockSize[2] = CudaProp()->maxThreadsPerBlock;
gridSize[2] = (count + blockSize[2] - 1) / blockSize[2];

// Random number generation setup
std::random_device rndDev;
std::mt19937 engine{ rndDev() };
std::uniform_real_distribution<float> dist{ 0.01, 0.99 };
auto randGen = [&dist, &engine]() {return dist(engine);};

// Generate random values for the power buffer
std::vector<float> powerBuf(numPix);
std::generate(begin(powerBuf), end(powerBuf), randGen);

float fixVal = 0.9; // 10% of pixels are random
std::fill(powerBuf.begin(), powerBuf.begin() + std::floor(numPix * (
    fixVal / 2)), 1.0f);
std::fill(powerBuf.begin() + std::floor(numPix * (1 - fixVal / 2)),
    powerBuf.end(), 1.0f);

// Shuffle power buffer
unsigned seed = std::chrono::high_resolution_clock::now().
    time_since_epoch().count();
std::shuffle(powerBuf.begin(), powerBuf.end(), std::
    default_random_engine(seed));
cudaMemcpy(devicePowBuf, powerBuf.data(), numPix * sizeof(float),
    cudaMemcpyHostToDevice);

// Clear simulation flags and set the old power buffer
cudaMemset(simFlagBuf, 0, numLightIdx * sizeof(bool));
std::fill(powerBuf.begin(), powerBuf.end(), 1.0f);
cudaMemcpy(devicePrevPowBuf, powerBuf.data(), numPix * sizeof(float),
    cudaMemcpyHostToDevice);

// Random first or last variant setup
std::srand(std::time(nullptr));
int randDecision = std::rand() % 100;
std::vector<int> simOrder = randDecision > 50 ? std::vector<int>{ 1, 2
    } : std::vector<int>{ 2, 1 };

// Allocate memory for filtered indices
int* filtIdx;
cudaMalloc(&filtIdx, numLightIdx * sizeof(int));
```

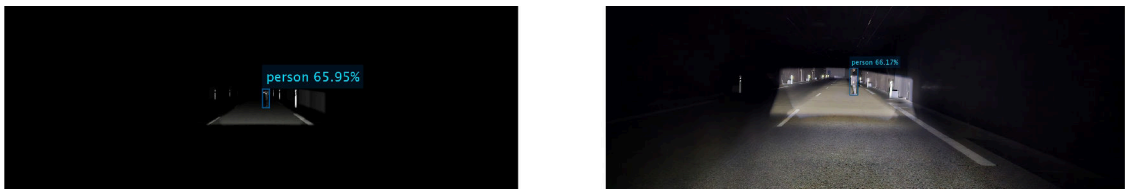
```
77
78  auto startTime = high_resolution_clock::now();
79  for (int i = 0; i < simOrder.size(); i++) {
80    if (simOrder[i] == 1) {  // First simulation strategy
81      cudaStreamSynchronize(stream);
82      startTime = high_resolution_clock::now();
83
84      // Run illumination simulation with CSR with all light indices
85      SimIllum<<<gridSize[0], blockSize[0], 0, stream>>>(numLightIdx,
            lightIdx, lightCount, pixIdx, red, green, blue,lightOff,
            devicePowBuf, lightDataOut);
86
87      cudaStreamSynchronize(stream);
88      *rms = ((float)duration_cast<nanoseconds>(high_resolution_clock::now
            () - startTime).count()) / (1e6f);
89    }
90
91    if (simOrder[i] == 2) {  // Second simulation strategy
92      // Compute differences between power buffers
93      ComputeDiffs<<<gridSize[2], blockSize[2], 0, stream>>>(pixIdxBuf,
            numPix, powDiffBuf, devicePowBuf, devicePrevPowBuf );
94
95      cudaStreamSynchronize(stream);
96      startTime = high_resolution_clock::now();
97
98      // Update simulation flags
99      UpdateSimFlags<<<gridSize[2], blockSize[2], 0, stream>>>(pixIdxBuf,
            numPix, powDiffBuf, simFlagBuf, pixOffBuf, pixDataBuf);
100
101     // Filter lights based on simulation flags
102     FilterLights<<<gridSize[0], blockSize[0], 0, stream>>>(simFlagBuf,
            lightIdx, filtIdx, numLightIdx);
103     cudaStreamSynchronize(stream);
104
105     // Use Thrust to remove duplicates
106     int* newEnd = thrust::unique(thrust::device, filtIdx, filtIdx +
            numLightIdx);
107
108     // Calculate number blocks and grids for valid indices
109     int validCount = newEnd - filtIdx;
110     float reductionRatio = (float)validCount / numLightIdx;
111     int blockSize0 = minf((float)CudaProp()->maxThreadsPerBlock,
            validCount);
112     int gridSize0 = (validCount + blockSize0 - 1) / blockSize0;
113
114     // Run illumination simulation with CSR with filtered indices
115     SimIllum<<<gridSize0, blockSize0, 0, stream>>>(validCount, filtIdx,
            lightCount, pixIdx, red, green, blue,lightOff, devicePowBuf,
            lightDataOut);
116
117     cudaStreamSynchronize(stream);
118   * rms2 = ((float)duration_cast<nanoseconds>(high_resolution_clock::
        now() - startTime).count()) / (1e6f);
119 }}
```

```
120
121 // Free allocated memory
122 cudaFree(filtIdx);
123 }
```

Listing A.1: CUDA C++ Code for computation time measurement of delta to normal update reworked by ChatGPT-4o to increase readability ‡t

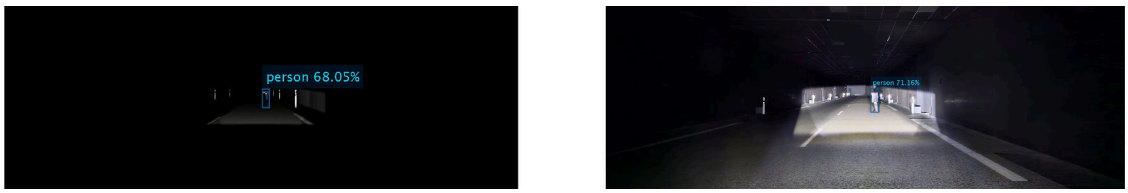
## A.5. Verification of the Visual Quality of Headlight Simulations

The evaluation of rendering fidelity is done through parallel assessment of simulated and real-world illumination scenarios conducted in the Forvia-Hella light tunnel. The virtual camera image was rendered by Unreal 5.2, and the Samsung S23 Ultra smartphone took real photos using photo optimization. The parameters of the PBR material model and virtual camera model were adjusted manually by empirical suggestive approximation and the simulated SSL|HD headlight has a non-realistic uniform white color.
Fig. A.4, A.5, A.6, A.7, A.8 and A.9 show the comparison of simulation and reality with YOLOv8 extra large 8.0.81 [Ult23] person detection. The predicted bounding box, the object class and the confidence score of the neural network are visualized in the images. The YOLOv8 network is used default-trained on the COCO dataset without domain-specific fine-tuning. The COCO dataset consists mostly of daytime images.



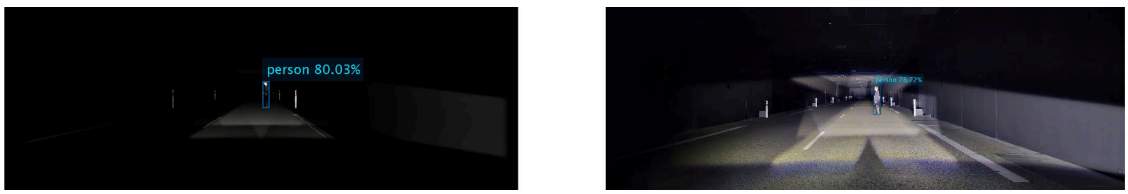

(a) Simulation. (b) Reality

Figure A.4.: Comparison of simulation and reality with the blue person detection.



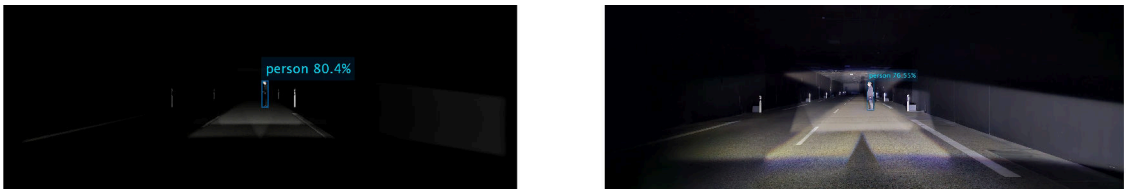

(a) Simulation. (b) Reality.

Figure A.5.: Comparison of simulation and reality with the blue person detection.



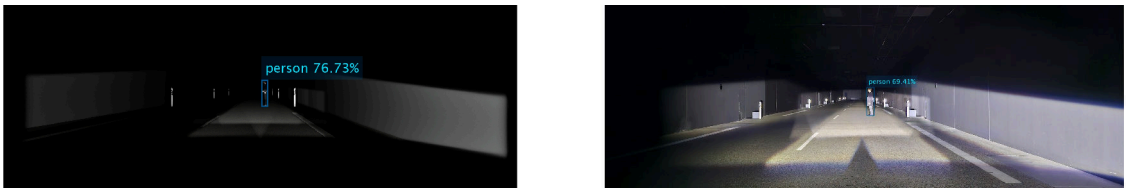

(a) Simulation. (b) Reality.

Figure A.6.: Comparison of simulation and reality with the blue person detection.




(a) Simulation. (b) Reality.

Figure A.7.: Comparison of simulation and reality with the blue person detection.




(a) Simulation. (b) Reality.

Figure A.8.: Comparison of simulation and reality with the blue person detection.


(a) Simulation. (b) Reality.

Figure A.9.: Comparison of simulation and reality with the blue person detection.

# B

# Appendix: Matrix Headlight Control

## B.1. Moore-Penrose Inverse Minimizes the Mean Squared Error‡c

That the Moore-Penrose inverse [Pen55] of (3.5.2) minimizes the MSE can be elucidated following the approach outlined by Isermann and Münchhof [IM11]. First, the quadratic error $e_{\text{sse}} \in \mathbb{R}_{\geq 0}$ is defined as the sum of squared differences between the desired illumination as $\boldsymbol{i}_{\text{v,h,s}}$ and the achievable illumination $\boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}}$ in matrix format as

$$e_{\text{sse}} = (\boldsymbol{i}_{\text{v,h,s}} - \boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}})^{\top}\,(\boldsymbol{i}_{\text{v,h,s}} - \boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}}) \tag{B.1.1}$$

$$= \boldsymbol{i}_{\text{v,h,s}}^{\top}\boldsymbol{i}_{\text{v,h,s}} - \boldsymbol{u}_{\text{p}}^{\top}\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{i}_{\text{v,h,s}} - \boldsymbol{i}_{\text{v,h,s}}^{\top}\boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}} + \boldsymbol{u}_{\text{p}}^{\top}\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}} \tag{B.1.2}$$

$$= \boldsymbol{i}_{\text{v,h,s}}^{\top}\boldsymbol{i}_{\text{v,h,s}} - \boldsymbol{u}_{\text{p}}^{\top}\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{i}_{\text{v,h,s}} - \left(\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{i}_{\text{v,h,s}}\right)^{\top}\boldsymbol{u}_{\text{p}} + \boldsymbol{u}_{\text{p}}^{\top}\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}} \tag{B.1.3}$$

To find the minimum of $e_{\text{sse}}$, the derivative of (B.1.3) with respect to $\boldsymbol{u}_{\text{p}}$ is calculated and set it to zero as

$$2\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}} - 2\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{i}_{\text{v,h,s}} \stackrel{!}{=} \boldsymbol{0} \Leftrightarrow \boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{A}_{\text{h}}\,\boldsymbol{u}_{\text{p}} = \boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{i}_{\text{v,h,s}}. \tag{B.1.4}$$

When $\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{A}_{\text{h}}$ is invertible with a $\det(\boldsymbol{A}_{\text{h}}^{\top}\boldsymbol{A}_{\text{h}}) \neq 0$, (B.1.4) can be solved for $\boldsymbol{u}_{\text{p}}$, yielding equation (3.5.2) and the Moore-Penrose inverse.

## B.2. Real-Time path projection by Bézier curves

Real-time path projection is analogous to curved line rendering in computer graphics [Fol+97], using the primitive symbol image as a planar drawing surface. Path definition uses Bézier curves [Far01], selected for their intuitive control point manipulation.

A set of start, end and shape defining control points $\boldsymbol{p}_{\text{bez},i} \in \mathbb{R}^2$ defines a 2D Bézier curve and a non-normalized curve parameter $l_{\text{b}} \in [0, 1]$. As the Bézier curve is drawn on the planar symbol image of a primitive, the points and the curve have two dimensions. For a $n_{\text{bez}}$-th degree Bézier curve $\boldsymbol{b}(l_{\text{b}})$, the mathematical formulation is

$$\boldsymbol{b}(l_{\text{b}}) = \sum_{i=0}^{n_{\text{bez}}} \boldsymbol{p}_{\text{bez},i} \binom{n_{\text{bez}}}{i} (1 - l_{\text{b}})^{n_{\text{bez}}-i} l_{\text{b}}^{i} \text{ with } l_{\text{b}} \in [0, 1], \tag{B.2.1}$$

where $\binom{n_{\text{bez}}}{i}$ denotes the binomial coefficient $\frac{n_{\text{bez}}!}{i!(n_{\text{bez}}-i)!}$.

When a texel of $\boldsymbol{I}_\mathrm{s}$ belongs to the Bézier curve, its distance to the curve must be below the width of the curve $d_\mathrm{cu} \in \mathbb{R}_{\geq 0}$. For texture point membership in the Bézier curve, the condition must be satisfied

$$\arg\min_{l_\mathrm{b} \in [0,1]} \left( ||\boldsymbol{b}(l_\mathrm{b}) - \boldsymbol{p}_{\mathrm{t},i}||^2 \right) \leq d_\mathrm{cu}^2, \tag{B.2.2}$$

where $\boldsymbol{p}_{\mathrm{t},i} \in N^2$ is the 2D position of a texel in texture coordinates. When the drawing is done during the ray casting, $\boldsymbol{p}_{\mathrm{t},i}$ is the position of the ray's hit with a primitive. When at a preprocessing, then $\boldsymbol{p}_{\mathrm{t},i}$ is an image coordinate. All $\boldsymbol{p}_{\mathrm{t},i}$ of a symbol image are evaluated for the inside condition of (B.2.2) in parallel [Pan90].

While quadratic curves ($n_\mathrm{bez} = 2$) permit algebraic solutions of the minimal distance of $\boldsymbol{p}_{\mathrm{t},i}$ to $\boldsymbol{b}(l_\mathrm{b})$ [Sul17], higher-degree curves necessitate approximation techniques. The derivation of the squared distance of the Bézier curve to a point

$$f_\mathrm{bd}(l_\mathrm{b}) = ||\boldsymbol{b}(l_\mathrm{b}) - \boldsymbol{p}_{\mathrm{t},i}||^2, \tag{B.2.3}$$

results for cubic curves ($n_\mathrm{bez} = 3$) in a fifth-degree polynomial equation requiring an approximation solution because for some polynomial equations of degree five or higher exists according to the Abel–Ruffini theorem, no solution in radicals [Jac12]. A solution in radicals means a solution only with the four basic arithmetic operations and the extraction of roots. Another reason for using a root-finding algorithm is the limited $l_\mathrm{b}$ interval of the Bézier curve interval as $[0, 1]$.

To find the minimum of $f_\mathrm{bd}(l_\mathrm{b})$ with a root-finding algorithm, the derivative of $f_\mathrm{bd}(l_\mathrm{b})$ to $l_\mathrm{b}$ is set to zero and the resulting equation is solved with a root-finding algorithm like Halley's method [ST95; Gan85; Ale81]. As Halley's method only finds local minima, the initial length $l_{\mathrm{b},0}$ should be chosen in the interval of the Bézier curve from a set of evaluated $l_\mathrm{b}$ so that the start distance to the curve is minimal. The iteration sequence of Halley's method at step $k$ and the length $l_{\mathrm{b},k}$ of the $k$-th step is

$$l_{\mathrm{b},k+1} = l_{\mathrm{b},k} - \frac{2 f'_\mathrm{bd}(l_{\mathrm{b},k}) f''_\mathrm{bd}(l_{\mathrm{b},k})}{2 f''_\mathrm{bd}(l_{\mathrm{b},k})^2 - f'_\mathrm{bd}(l_{\mathrm{b},k}) f'''_\mathrm{bd}(l_{\mathrm{b},k})}. \tag{B.2.4}$$

The iteration stops when the difference of two consecutive iterations is below a threshold or leaves the Bézier curve interval $[0, 1]$.
The resulting $l_\mathrm{b}$ is used to check the inside condition of (B.2.2) and calculate the distance as $\sqrt{f_\mathrm{bd}(l_\mathrm{b})}$. When the point is a member of the curve, the distance and $l_\mathrm{b}$ can be used as $u, v$ texture coordinates to bend a desired symbol along the Bézier curve path as seen in Fig. 7.6a. As $l_\mathrm{b}$ can deviate from the curve arc length, an additional arc length estimation [Gra97] may be required.

## B.3. Computational Performance Analysis of the Matrix Headlight Control[‡†t]

This section evaluates the computational performance of SSC [WB21b; WB21c; WB22]. The SSC implementation uses CUDA C/C++ 11.3. The evaluation was conducted on an Intel i9-9980XE CPU with 64 GB RAM and a Nvidia 2080 Ti GPU with 11 GB memory.

Throughout the evaluation process, resource utilization remained below 90 % for CPU and GPU, indicating adequate computational headroom.
The analysis focuses on the relationship between computation time, headlight pixel number and the corresponding number of rays cast for lighting computation. The result tables show the mean, StD, median, 5 % quantile (quan.) and 95 % quantile (quan.) of the measured computation times.

### B.3.1. Initial Performance Characterization

Table B.1 presents a statistical analysis of computational performance across three headlights with 84, 30,000 and 100,000 pixels, evaluated within the scenario depicted in Fig. 7.5 [WB21b]. The computational time distribution exhibits non-Gaussian characteristics. A notable observation is the relative stability of median computation time between 84 and 30,000 pixels, followed by a significant increase of approximately 34.4 % when scaling to 100,000 pixels. This non-linear scaling behavior can be attributed to architectural constraints of the GPU hardware, precisely the finite number of concurrent threads and memory access bandwidth limitations.
Despite these constraints, the algorithm has robust real-time performance capabilities, maintaining sub-millisecond average computation times across all configurations.

Table B.1.: Computation times in µs based on 800 measurement samples [WB21b].

| Pixels | Rays Cast | Mean | StD | Median | 5 % Quan. | 95 % Quan. |
|---|---|---|---|---|---|---|
| 84 | 45,826 | 262.1 | 88.3 | 257 | 183.5 | 306.0 |
| 30,000 | 140,146 | 251.2 | 46.5 | 256 | 176.0 | 299.5 |
| 100,000 | 371,392 | 336.8 | 55.6 | 344 | 250.0 | 386.0 |

### B.3.2. Extended Performance Analysis

A second evaluation was conducted over 2,000 computational cycles, examining the same pixel configurations as Table B.1 within a scenario illustrated in Fig. B.1 [WB21c]. The relationship between pixel count and ray casting exhibits a fundamental coupling, as the lighting database resolution scales proportionally with pixel number. Higher pixel numbers require increased ray count to preserve illumination fidelity across different pixel densities.

Tables B.2 and B.3 present performance metrics for the right headlight. The left headlight, not shown in the tables, exhibits similar performance characteristics. Minor variations exist due to differences in optimization effectiveness and intersection computation priorities. Notably, the distribution of measured computation times does not conform to a normal distribution.
Given that a GPU is constrained by a finite number of parallel threads and limited memory bandwidth, the increase in computation times is observed to be less than proportional to the increase in pixels and rays, as evidenced by Tab. B.3.

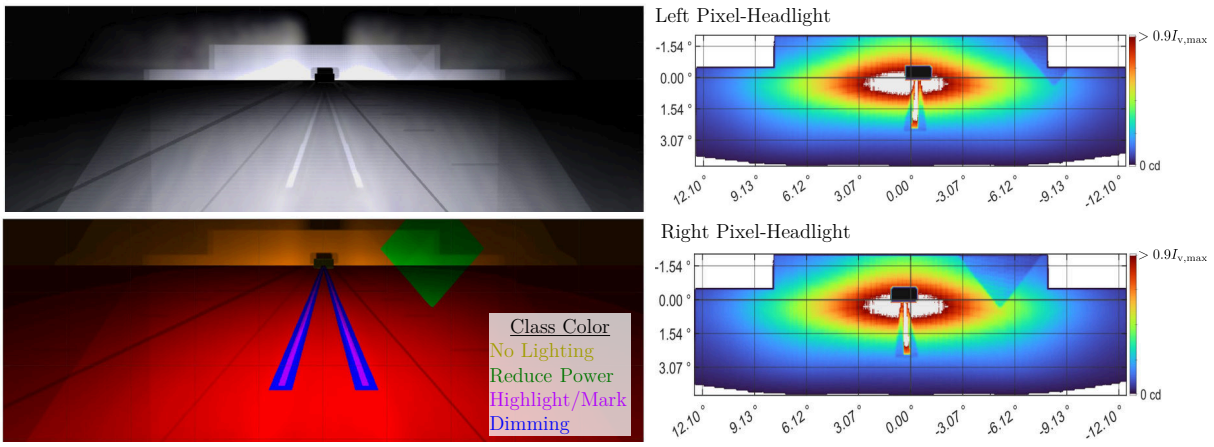



Figure B.1.: Simulated illumination, primitives and beam patterns for a 30,000-pixel headlamp without overlap effects [WB21c].

Table B.2.: Detailed computation time analysis in ms across 2,000 measurement cycles for the right matrix headlamp [WB21c].

| Pixels | Rays Cast | Mean | StD | Median | 5 % Quan. | 95 % Quan. |
|---|---|---|---|---|---|---|
| 84 | 45,846 | 0.184 | 0.085 | 0.173 | 0.157 | 0.221 |
| 30k | 140,146 | 0.206 | 0.104 | 0.189 | 0.178 | 0.241 |
| 100k | 371,392 | 0.340 | 0.185 | 0.308 | 0.287 | 0.436 |

Table B.3.: Comparative scaling analysis of computation metrics normalized to the 84-pixel baseline configuration [WB21c].

| Pixels | Rays Cast | Mean | StD | Median | 5 % Quan. | 95 % Quan. |
|---|---|---|---|---|---|---|
| 84 | 100 % | 100 % | 100 % | 100 % | 100 % | 100 % |
| 30k | 305.6 % | 111.9 % | 122.3 % | 109.2 % | 113.3 % | 109.0 % |
| 100k | 810.0 % | 184.7 % | 217.6 % | 178.0 % | 182.8 % | 197.2 % |

### B.3.3. High-Resolution Performance

The geometric primitive generation process has consistent performance with a median computation time of 0.28 ms across 2,000 cycles for the evaluation scenarios presented in Fig. 7.6a, 7.6b and 7.7 [WB22].
A dataset of 5,000 samples, equally distributed between left and right headlight configurations, was collected and analyzed [WB22]. Table B.4 presents both absolute performance metrics for the 1,000-pixel baseline configuration and relative scaling characteristics for higher-resolution variants up to 1.05M pixels. The distribution of computation times does not conform to a normal distribution.
SSC demonstrates scalability, achieving control of a 1.05M pixels headlight configuration with a median computation time of 1.79 ms (calculated as 0.185 ms · 9.659).
The percentage increase in computation time is less pronounced than the corresponding increase in pixel count and cast rays.
The superior performance of the 30,000-pixel configuration compared to the 1,000-pixel baseline is attributed to reduced pixel illumination areas, resulting in fewer rays per pixel. This optimization effect is particularly pronounced in the reduce phase of SSC, where weighted averaging of ray setpoints results in fewer global memory transactions per pixel at higher resolutions.

The dynamic Bézier curve path projection for the 1.05M pixel configuration yielded median computation times of 1.515 ms (95 % quantile: 2.021 ms).

Table B.4.: Performance analysis across pixel configurations, based on 5,000 samples per headlight assembly. Metrics for configurations above 1,000 pixels are presented as percentages relative to the baseline [WB22].

| Pixels | Rays Cast | Mean | StD | Median | 5 % Quan. | 95 % Quan. |
|---|---|---|---|---|---|---|
| 1k | 130.1k | 205 µs | 107 µs | 185 µs | 176 µs | 279 µs |
| 30k | 107.7 % | 93.7 % | 98.1 % | 94.6 % | 94.3 % | 94.3 % |
| 100k | 285.4 % | 166.3 % | 82.2 % | 178.9 % | 175.6 % | 149.5 % |
| 1,050k | 2,360.5 % | 882.4 % | 198.1 % | 965.9 % | 938.1 % | 800.4 % |

## B.4. Distribution of Optimal Matrix Headlight Control Parameters

The optimal parameters for the SSC method are determined for the HD84 and SSL|HD matrix headlights for the set of setpoint LID of Sec. 8. Fig. B.2, B.3 and B.4 show optimal values of $a_{\mathrm{m,ssc}}$, $a_{\mathrm{x,ssc}}$ and $a_{\mathrm{r,ssc}}$ for the HD84 matrix headlight and Fig. B.5, B.6 and B.7 for the SSL|HD matrix headlight.

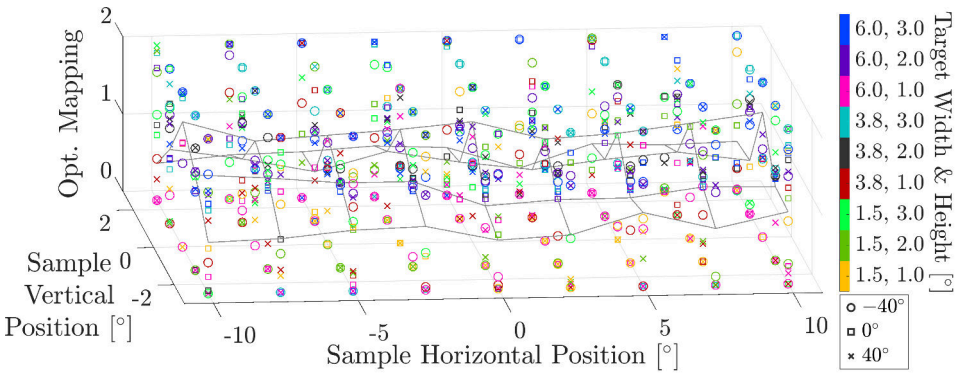



Figure B.2.: Optimal $a_{\mathrm{m,ssc}}$ of SSC for the HD84 matrix headlight (mean 0.91, median 0.99).

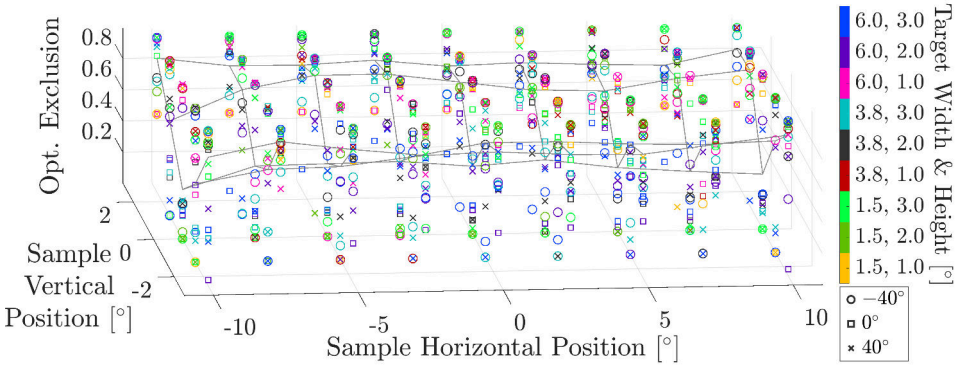



Figure B.3.: Optimal $a_{\mathrm{x,ssc}}$ of SSC for the HD84 matrix headlight (mean 0.70, median 0.87).

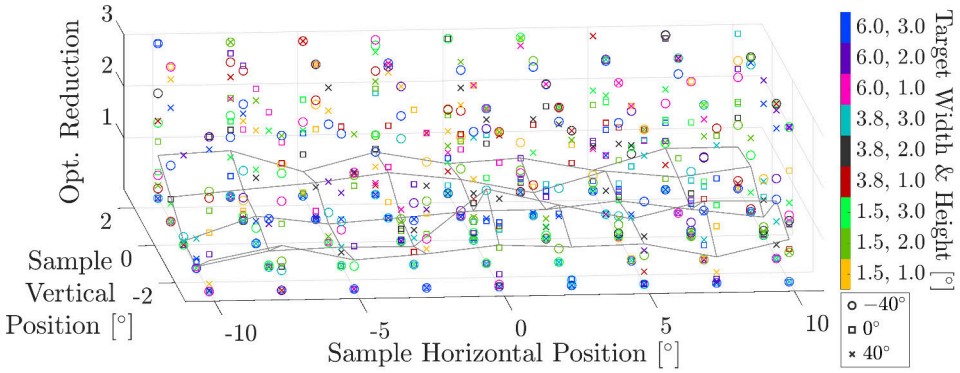



Figure B.4.: Optimal $a_{\mathrm{r,ssc}}$ of SSC for the HD84 matrix headlight (mean 0.65, median 0.001).

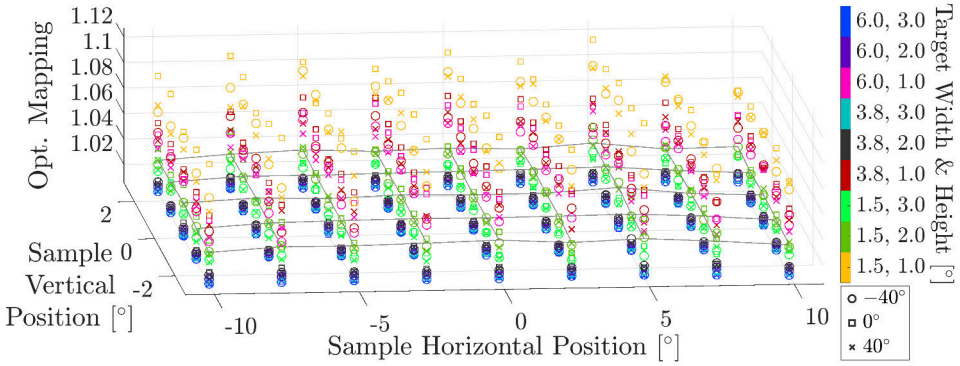



Figure B.5.: Optimal $a_{\mathrm{m,ssc}}$ of SSC for the SSL|HD matrix headlight (mean 1.035, median 1.025).

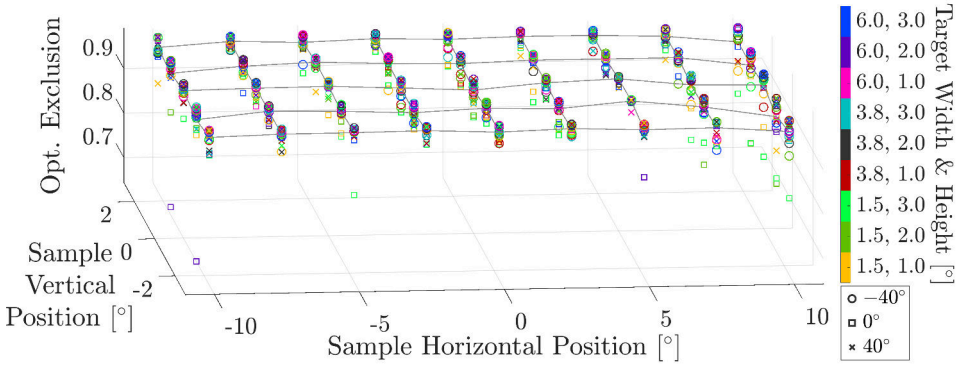



Figure B.6.: Optimal $a_{\mathrm{x,ssc}}$ of SSC for the SSL|HD matrix headlight (mean 0.97, median 0.98).

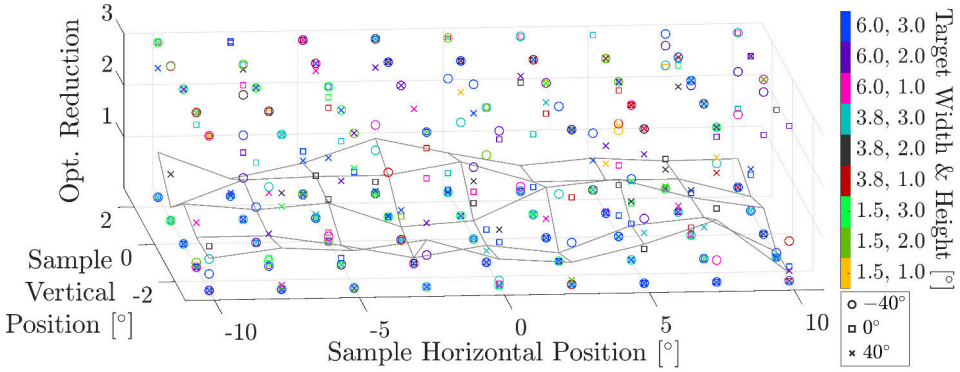
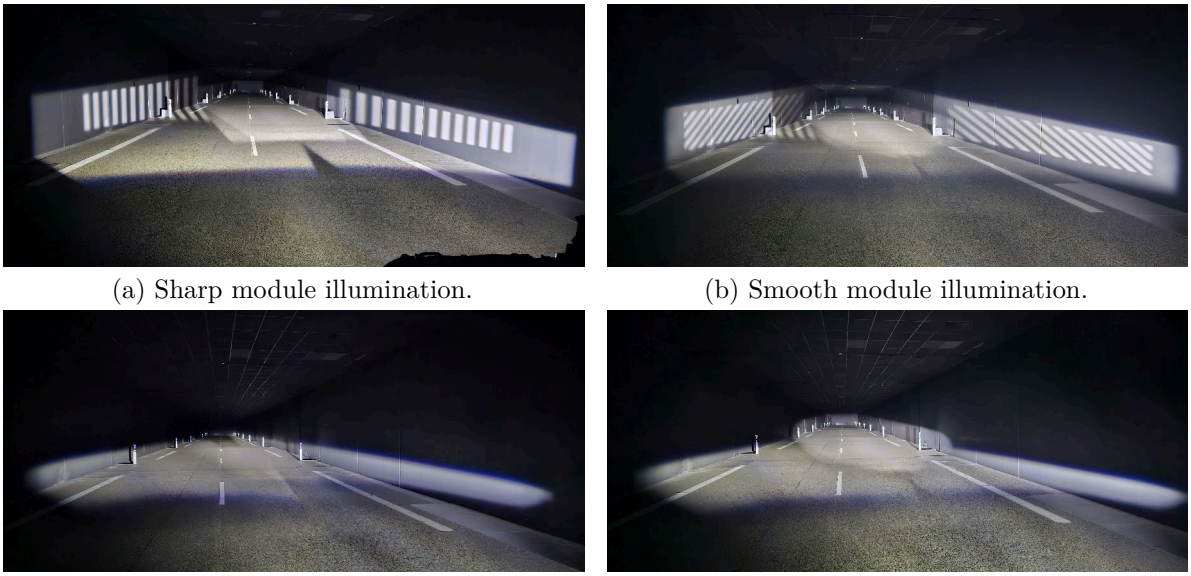



Figure B.7.: Optimal $a_{\mathrm{r,ssc}}$ of SSC for the SSL|HD matrix headlight. (mean 0.61, median 0.0010).

## B.5. Real Illuminations of six Matrix Headlights

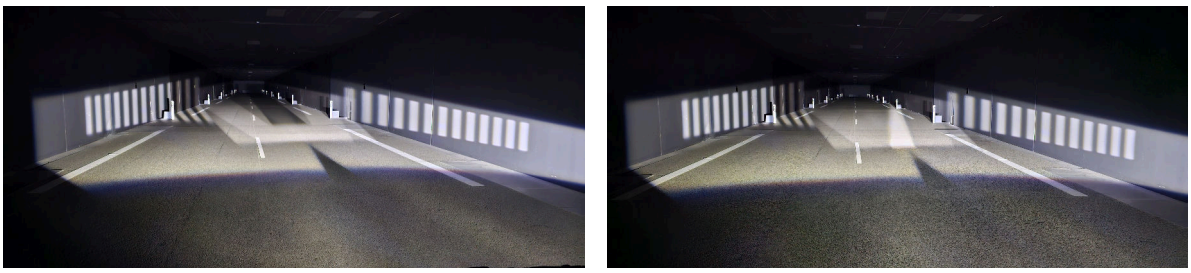
Evaluation of the capabilities of SSC in the Forvia-Hella light tunnel. The SSC algorithm simultaneously controls six SSL|HD matrix headlights and realizes various lighting functions and beam patterns. The following figures B.8, B.9, B.11, B.12 and B.10 show the real illuminations of the six SSL|HD modules.



(a) Sharp module illumination. (b) Smooth module illumination.

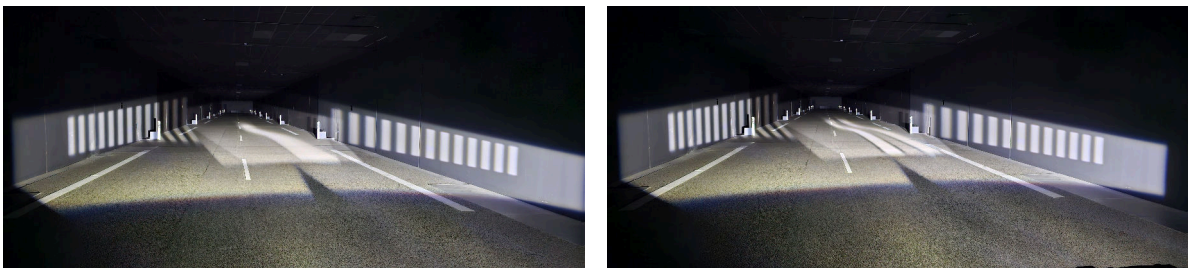
(c) Low beam illumination. (d) High beam illumination.

Figure B.8.: Base lighting of six SSL|HD modules.



(a) GFHB. (b) Marker light.

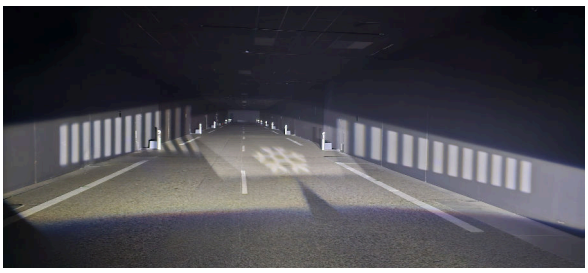
Figure B.9.: Classic lighting functions.



(a) Homogeneous (b) With symbol

Figure B.10.: Path projection.

(a) Arrow.



(b) Snowflake.

(c) Collision warning.

(d) Warning sign.

Figure B.11.: Symbol projection.

(a) Desired observer perspective.

(b) Wrong perspective.

(c) Desired observer perspective.

(d) Wrong perspective.

Figure B.12.: Symbol projection for a desired observer.

# Usage of Generative AI Models

Indication of the use of ChatGPT, Claude and comparable generative AI (GenAI) tools in the context of this document, unless the GenAI is the subject of the scientific investigation. The symbol ‡ indicates the usage of GenAI for the marked section and all subsections in the thesis. In this document, ChatGPT, Claude or another GenAI was used as follows:

- ☐ Not at all.
- ☐ Finding ideas.
- ☐ Creating the outline and structure.
- ☒ Content generation: Generating entire text and media sections.
- ☒ Media generation: Creating entire passages from given content.
- ☒ Media optimization: Correction, optimization, or restructuring of the entire written work without explicitly marking individual passages or sections (text and media optimization). The entire text is optimized with the GenAI Claude 3.5 Sonnet.
- ☐ Value generation in code: Generating the entire relevant software source code.
- ☐ Code generation: Creating entire relevant software functions from a detailed function description.
- ☒ Code optimization: Correction, optimization, or restructuring of relevant software functions. GitHub Copilot was active during programming.
- ☒ Other: The LaTeX document is written in Visual Studio Code with active GitHub Copilot and Grammarly.