Lux junior 2021

15. Internationales Forum für den lichttechnischen Nachwuchs
Ilmenau, 04. – 06. Juni 2021
DOI (proceedings): 10.22032/dbt.48427

DOI: 10.22032/dbt.49337

Virtual Evaluation of High-Definition Lighting Functions for Various Beam Patterns for Rapid-Prototyping

Mirko Waldner and Torsten Bertram

TU Dortmund University, Institute of Control Theory and Systems Engineering (RST)
mirko.waldner@tu-dortmund.de

Abstract

This contribution presents a concept for early virtual evaluation of high-definition (HD) lighting functions for different beam patterns without requiring major manual changes to the headlight control algorithm. This is achieved by designing the headlight control algorithm to automatically adapt to the current beam pattern and realize various lighting functions of generalized targets. This paper delivers the general concept of the algorithm for the case of relative target utilization of the light sources and presents the modifications to the approach required to control the illumination of a spot using a low-resolution matrix-headlamp.

Index Terms: Digital Light, Pixel-Headlights, Rapid-Prototyping

1 Introduction

Modern (matrix-)headlights use a combination of different light modules, e.g. high beam module and cornering light module, with a different number of individual light sources called pixels to illuminate the road. The illumination in front of the ego vehicle is created by superimposing the illuminations from the light modules. Fig. 1 shows a simulated illumination produced by a pair of matrix-headlamps with two light modules. One is a low-definition (LD) matrix-headlight with 84 pixels and the other is a high-definition (HD) pixel-headlight with 30k pixels. The HD light module is more focused on the road to make better use of the pixels, such as for more precise symbol projection. One of the first challenges in system development of such a front lighting system is finding the optimal target values such as the number of pixels and maximum beam angles for product development. This challenge can be solved by evaluating promising system designs in simulated scenarios as virtual agile rapid prototyping of design ideas.

To accelerate the rapid prototyping process, the contribution at hand presents a real-time control approach that can handle different beam patterns and headlamp resolutions (i.e. pixel count) without major manual adjustments. The algorithm uses

paralyzed raycasting and the known approach to real-time headlamp simulation [1], which accesses only important data at runtime to achieve real-time capability on a graphics processing unit (GPU). This enables the real-time evaluation of various HD lighting functions like projecting lines, signs and so on. The presented algorithm is called Super-Sampling-Control (SSC) because it works in principle in the same way as the well-known super-sampling anti-aliasing techniques in computer graphics [2]. The user defines SSC's targets as 3D objects in the world, such as a cuboid for a vehicle or a rectangle for a projected line, that represent the ideal quantity and shape of an illumination function. Possible object classes are reduced, normal, and strong illumination. SSC attempts to optimally replicate the shape and brightness of the targets with the illumination of the current matrix-headlamp by casting rays against the world objects and evaluates the objects it hits.

The next section introduces SSC for relative target utilizations of the pixels to the current Adaptive Front Lighting System´s (AFS) base light distribution and shows possible connections with different development methods [3]. The following section discusses the use of SSC for low-resolution headlamps using the example of marker light. The evaluation section gives an overview of the performance and visual quality of SSC and the paper concludes with a summary and outlook.

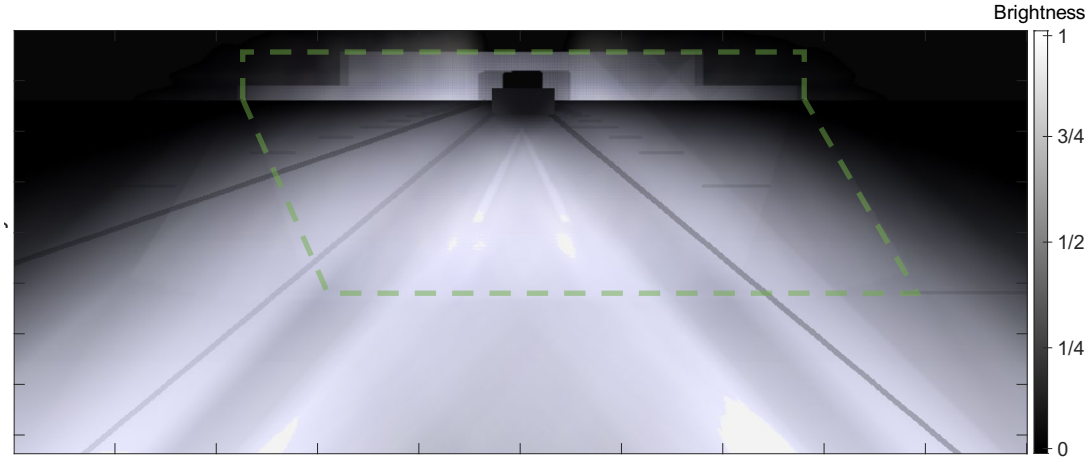

Fig. 1: Visualization of illumination of two matrix-headlamps with a low- and a high-definition light module. The illumination area of the right HD light module is highlighted in green. This visualization includes the simulation of a volumetric lighting to visualize the beam propagation in three-dimensional space. This makes visible the "dark tunnel", which are created by turning off pixels to reduce for example the dazzle of other traffic participants.

2 Super-Sampling-Control (SSC) of Matrix-Headlights

In general, at an arbitrary position $(x, y, z)^{\mathrm{T}} \in \mathbb{R}^3$ in a three-dimensional world the illuminance $E_{\mathrm{v}}(x, y, z) \in \mathbb{R}^{\geq 0}$ is the superposition of the maximum intensities $I_{\mathrm{v},i,\mathrm{Max}} \in \mathbb{R}^{\geq 0}$ of all n pixels multiplied by the individual utilization $p_i \in \{ x \in \mathbb{R} \,|0 \leq x \leq 1\}$ of the pixel and an attenuation factor $f_{\mathrm{a}}(\mathcal{G}_i, \mathcal{V}_i, \mathcal{B}_i) \in \mathbb{R}^{\geq 0}$. This quantity describes the influence of the i-th pixel to the level of $E_{\mathrm{v}}(x, y, z)$, which depends on a set of geometric

properties $\mathcal{G}_i$, such as the distance and relative position of the light origin, the set of visibility or shading properties $\mathcal{V}_i$,e.g. information on whether other objects block the light, and a set the individual beam pattern properties $\mathcal{B}_i$, e.g., the intensity distribution and the amount and direction of the maximum intensity, of the pixel. The superposition of the lights from n pixels for an arbitrary position is

$$E_{\mathrm{v}}(x,y,z) = \sum_{i=0}^{n-1} f_{\mathrm{a}}(\mathcal{G}_i, \mathcal{V}_i, \mathcal{B}_i)\, I_{\mathrm{v},i,\mathrm{Max}}\, p_i. \tag{1}$$

The goal of every algorithm for headlight control is to find the best utilization p_i for all pixels so that the illumination is optimal at all relevant world positions. Due to a possible large number of pixels, which can be ≥ 1M pixels per module for a digital headlight, and discrete world points, optimization and control is time consuming without parallelization and assumptions or simplifications.

The SSC algorithm casts for each pixel m_i parallel individual rays for the calculation of the best utilization $p_{i,j}$ for the direction and illumination of the j-th ray. In a second step, the target values from all the m_i rays are collected and from all $p_{i,j}$ the optimum p_i for the i-th pixel is generated with a weighted average. This is also done in parallel. SSC is a two-step approach with a raycasting step and a pixel evaluation step. This method is similar in principle to the MapReduce programming model [4] and super-sampling anti-aliasing in computer graphics [2]. In a realistic headlamp the ray patterns of the pixels overlap so that the target quantity, e.g. E_{v}, at a position must be divided into the optimal individual illuminations of all pixels for this position, which is a kind of inversion of (1) for all positions.

To find the optimal individual illumination, SSC casts m_i individual rays from a single point origin of the entire headlamp, or at least for one light module. This point source is the origin of all controlled pixels of this headlamp. More precisely all radiation characteristics must be referenced to a single common origin. For each discrete propagation segment, i.e. discrete angular steps in spherical headlamp data, a ray is cast in parallel. SSC uses this ray by testing it against the objects of the world, e.g. a cuboid for a vehicle, to determine the sets of $\mathcal{G}_j$ and $\mathcal{V}_j$ for the j-th ray and the target value by evaluating the class of the object. If the user specifies the target values for SSC as relative workloads $p_{i,\mathrm{Rel}} \in \mathbb{R}^{\geq 0}$ then the utilization of the pixel is calculated in the last step by multiplying the found $p_{i,\mathrm{Rel}}$ with the utilization from the current AFS class $p_{i,\mathrm{AFS}}$. This contribution restricts the description of SSC from now on to the case of relative target utilizations. The founded $p_{j,\mathrm{Rel}}$, which is determined by the class of the hit world object, is transferred to the individual pixels (more precisely, the value is added to the set of values of all rays of that pixel) that illuminate the area of the ray. When multiple objects are hit, SSC considers the object with the shortest distance to the origin, i.e., the visible object, except for critical light features, e.g., Glare-Free-High-Beam (GFHB), which are always considered even if they are obscured by other targets.

The transfer of target utilization to the pixels uses the same data structure as the real-time headlight simulation [1], which is basically a compressed collection of the relevant illuminations of all pixels. For each ray there exists an individual set of the indices of only the pixels that illuminate with an intensity $I_v > 0$ of the direction of the ray. This set is evaluated for the transmission of the target utilizations.

One possible strategy is to calculate the relative utilization of the pixel using the weighted average of the utilizations $p_{j,\mathrm{Rel}}$ from all m_i rays. With the weight $w_j \in \mathbb{R}^{\geq 1}$, which is also determined by the class of the object hit (or a total failure), the utilization p_i of the i-th pixel is

$$p_i = \frac{p_{i,\mathrm{AFS}}}{\sum_{j=0}^{m_i-1} w_j} \sum_{j=0}^{m_i-1} w_j \, p_{j,\mathrm{Rel}}. \tag{2}$$

Fig. 2 shows a simplified example of the described process without weights.

For HD headlights, the difference of weights for different classes of objects is a tuning parameter for the sharpness of edges and the visibility of the individual beam pattern of pixels, or in other words for the weakness of anti-aliasing and edge smoothing. The transition of illuminations of objects with similar weights is smoothed by using (2) to calculate the utilization, as p_i is then set by the average of the utilization factors weighted by the individual illuminated area of the objects.

For the realization of GFHB equation (2) is not used because it does not guarantee the reduction of p_i to ≈ 0 if the weights are poorly chosen. Since GFHB as a safety-critical illumination function has a high priority compared to other functions, it is safer and more run-time efficient to set $p_i = 0$ as soon as a ray detects a target with $p_{j,Rel} = 0$ and then terminate the control process. To achieve a transcend effect at the edges of the "dark tunnel", a vehicle can be represented by an inner and outer cuboid. The inner

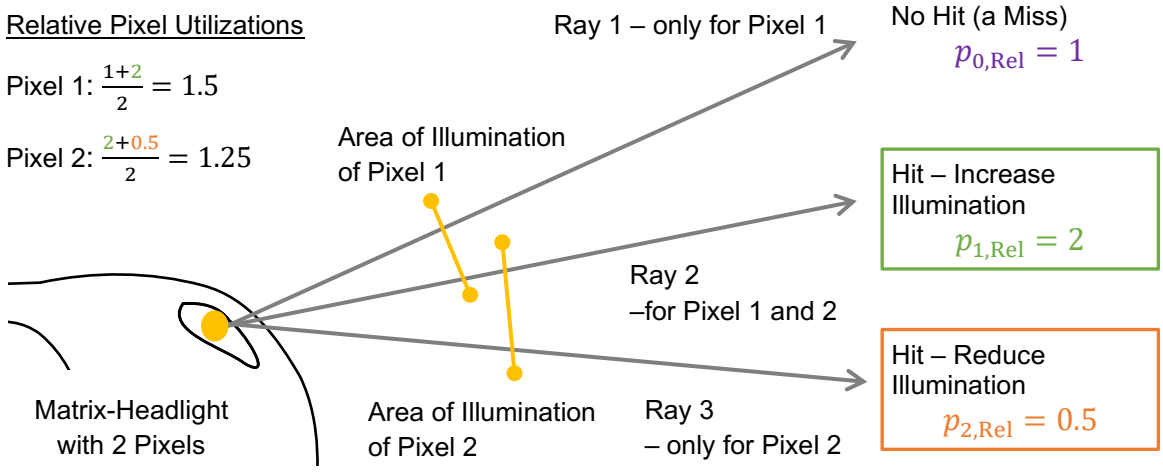

Fig. 2: Example of SSC for a headlight with $n = 2$ pixels and $m_i = 2$ casted rays without weights. The pixels overlap only in the area of the second ray, so $p_{1,Rel}$, which was founded by the second ray, is transferred to both pixels. The relative pixel utilization is calculated by the mean of the two founded values.

smaller object has the target value $p_{i,Rel} = 0$ to provide glare protection and the outer larger cuboid has $0 < p_{i,Rel} < 1$ for the blending effect.

SSC is designed to run on a GPU and for early rapid prototyping, but can be combined or linked with other headlight development methods [3]. SSC acts as an abstract prototype of the headlight control software that is not intended to be directly converted into production-ready code, e.g. C code on a CPU. The approach relies on the availability and exploitation of parallel activity threads to achieve real-time capability, so if there are few threads available, the computational performance is not comparable. Second, SSC does not use look-up tables, which are precomputed utilizations of pixels for projecting symbols in different positions, orientations, or situations. This is an advantage for prototyping because the illumination is always optimally created for the current situation and no decision or design process of the look-up table is required. By not using look-up tables, SSC is not computationally resource efficient and should not be used when the number of options is limited. Then it is more efficient to use look-up tables to reduce computation times and minimize the use of parallel threads, especially on microcontrollers.

Since the output values of SSC are the utilizations of pixels, the approach can control simulated headlights in early development steps or real headlights in the later process. This allows SSC to be used as a prototyping control method and later replaced by the production-ready approach such as the C implementation. Combined with real-time digitization [5] of the matrix headlight, the real device can be tested hardware-in-the-loop as soon as it can be built.

3 Highlight a Spot with Low-Definition Matrix-Headlights

The described algorithm for SSC does not consider the properties of the pixel-individual beam patterns $\mathcal{B}_i$ like the shape of the illumination, but only whether an area is illuminated by a pixel or not. In general, using the method for relative target utilizations may be accurate, since $\mathcal{B}_i$ is considered by scaling the standard utilization $p_{i,\mathrm{AFS}}$ from the current AFS class. The default light distribution for an AFS class includes the optimal consideration of $\mathcal{B}_i$ as an assumption, so that $\mathcal{B}_i$ is indirectly taken into account.

This approach may lead to unsatisfactory results for matrix-headlights with relatively large overlaps of the illumination areas of the pixels. If the size of the target object e.g. a line is similar to the size of the pixel beam pattern, then the target shape will be blurred and spread over a larger area by affecting more pixels than perhaps necessary. This is shown in fig. 3 with an example of marker light for a matrix-headlight with 84 pixels. The target is a line with a linewidth like the horizontal size of a pixel illumination, but the overlaps transfer the target utilization to more than the (perhaps) minimum number of pixels needed. This increases the highlighted area (fig. 3, right) and making the target point harder to locate.

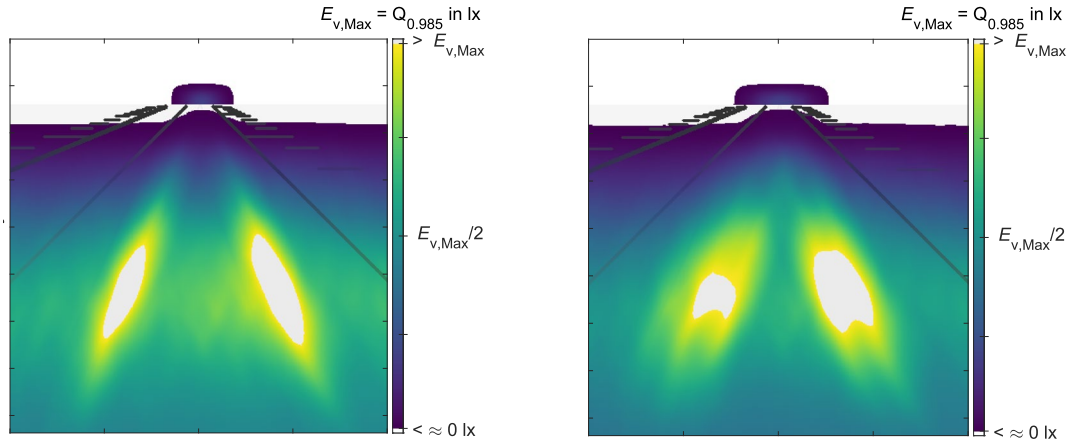

Fig. 3: Illuminance E_v of a matrix-headlight with 84 pixels in false colors. The headlight should highlight a distant point with a line. On the left SSC use one horizontal pixel per headlight for creating the line and on the right two. The illuminance $E_{\mathrm{v,Max}}$ for the false color visualization is clamped to the 98,5 % quantile $Q_{0.985}$ of all simulated illuminances E_v in the screen. Further than the point there is an invisible wall.

One solution is defining an adaptive transferring of $p_{j,Rel}$ which depends on the beam pattern properties $\mathcal{B}_i$ of all $\mathrm{n_j}$ pixels, which illuminate in the direction of the j-th ray. All $\mathcal{B}_i$ are used to determine the importance of the i-th pixel on the j-th ray $w_{j,i} \in (0,1]$. The weight is modified by a weight function $f_\mathrm{w}(w_{j,i}) \in [0,1]$ to calculate the utilization $p_{j,i,\mathrm{Rel}} \in \mathbb{R}^{\geq 0}$, which is transferred from the j-th ray to the i-th pixel

$$p_{j,i,\mathrm{Rel}} = 1 + f_\mathrm{w}(w_{j,i}) \cdot (p_{j,\mathrm{Rel}} - 1). \tag{3}$$

Choosing the right $f_\mathrm{w}(w_{j,i})$ is a lighting function design decision. A relative utilization of $p_{j,i,\mathrm{Rel}} = 1$ does not lead to a change in the AFS base utilization, i.e. the utilization of pixels with little influence on a spot is not changed. The weight $w_{j,i}$ is therefore calculated by dividing the maximal possible intensity $I_{\mathrm{v},i,j,\mathrm{Max}}$, which is a part of the set $\mathcal{B}_i$ that the pixel can emit in the ray direction, by the maximum of all the $\mathrm{n_j}$ intensities in ray direction

$$w_{j,i} = \frac{I_{\mathrm{v},i,j,\mathrm{Max}}}{\max\left(\left\{I_{\mathrm{v},0,j,\mathrm{Max}}, I_{\mathrm{v},1,j,\mathrm{Max}}, \dots, I_{\mathrm{v},\mathrm{n_j}-1,j,\mathrm{Max}}\right\}\right)}. \tag{4}$$

This means that always the pixel with the highest intensity gets $p_{j,\mathrm{Rel}}$ as the target value. Applying (3) and (4) produces the result on the left side of fig. 3 and reduces the size of the projected line.

For moving objects the movement between the areas of pixels becomes more visible because the change of pixel intensity with (3) and (4) starts as soon as the object enters the overlapping area of pixels and not when the object leaves the illumination area of a pixel. When an object enters the beam pattern of the complete headlamp the

modifications have no benefit because the outer areas are usually illuminated by only one pixel. For the case of marker illumination by LD matrix-headlights (see fig. 3), the line and spot are more visible, because the overlap area of the two lines is smaller. Only one pixel is activated in the horizontal direction when the target line is close to the illumination center of a pixel and the utilization of the surrounding pixels does not change. Without the described modification this is not the case, because the overlaps would lead to an activation of at least two pixels in horizontal direction.

4 Evaluation

After the introduction of the algorithm this section shows at first results of an implementation of SSC from the aspects of visual quality and target distribution matching. Afterwards the absolute running times of SSC for different headlamps resolutions are shown and compared.

SSC and the real-time headlight simulation [1] are implemented in CUDA C/C++ 11.3 [6] and operate therefore primary on a GPU. The generated data and visualization is post-processed with OpenCV 4.5 [7], e.g. tone mapping or false colors. The computer has an Intel i9-9980XE CPU with 64 GB of RAM and an Nvidia RTX 2080 Ti GPU with 11 GB of dedicated GDD6 memory. Neither the CPU nor the GPU are utilized above 90% during the evaluation.

The overall visual quality of the system already was shown in fig. 1 and 3. Therefore fig. 4 gives a more detailed insight to the functionality of SSC. The target light distribution is set by placing objects, the label cuboids, in the world. The line that should illuminate a point at a distance of 100 m is created by two objects per line. The small line at the top (height in 3D space) increases the illumination and the large line at the

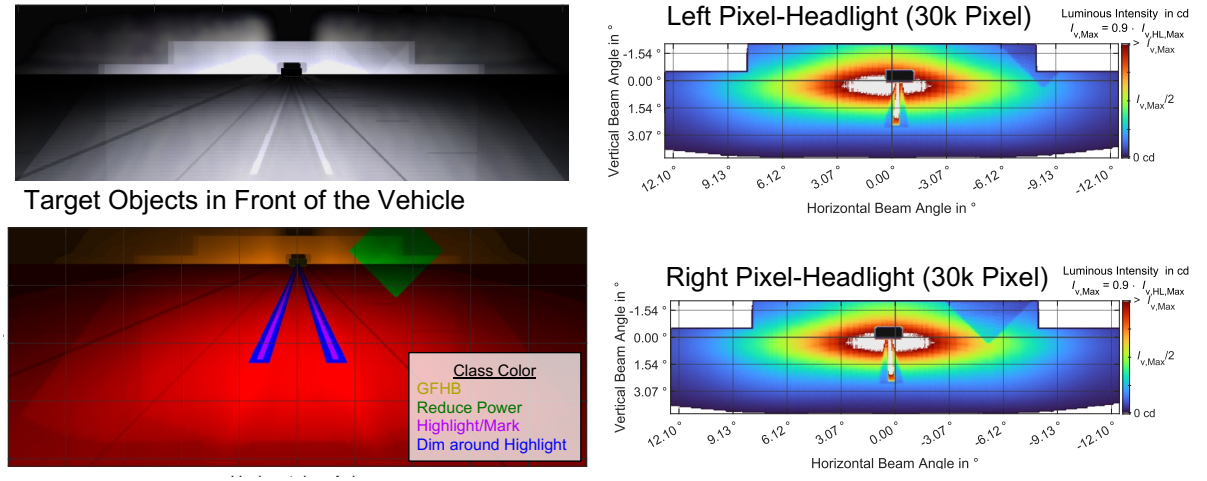

Fig. 4: Visualization of the SSC controlled illumination of a pair of headlamps with a LD light module and a HD light module. The target objects (down left) for different lighting functions, e.g. golden cuboid for GFHB, are analyzed by SSC. SSC adapts the intensity I_v distribution to match the target objects. The cast intensities of the left and right HD light modules are shown in the right. The false color visualization is clamped to 90 % of the headlight maximum intensity $I_{v,HL,Max}$. The resulting illumination is shown in the top left.

bottom decreases the illumination to make the line better visible when the surrounding illumination is similar. Fig. 4 also shows the cast light distribution by the two HD pixel-headlights (30k pixels) and the resulting illumination in front of the ego vehicle.

The effectiveness of the adjustment of the target relative utilization with (3) and (4) for marker light with an LD matrix-headlight was shown in false colors in fig. 3 (left with and right without). With the adjustment the line becomes smaller and better visible and traceable in motion, which unfortunately cannot be shown in the written paper.

After the evaluation of the visual quality the computational performance is presented. For this purpose 2000 computation runtimes for different headlamp resolutions (84, 30k and 100k pixels) are measured for one scenario. The number of cast rays changes for different resolutions as the number of data points in the headlight database is adjusted to the size of the pixel illumination. For similar aperture angles, the number of rays increases with resolution to preserve the detail of the smaller pixel illuminations. The measurement scenario is shown in fig.4. The implantation in CUDA uses shared memory within a thread block to reduce the amount of slow memory accesses to global GPU memory. Furthermore the algorithm terminates threads as quickly as possible, e.g. when no object has been hit by a ray, to minimize hardware utilization. Table 1 and table 1.a show the results for the right headlight. The runtimes for the left and right headlights are not identical but similar, which is caused by optimizations and priorities in testing the rays against primitives. The measured runtimes are not normally distributed.

Table 1: Runtimes in ms for 2000 measured cycles for the right matrix-headlamp.

Resolution	Rays Cast	Mean	Standard Deviation	Median	5 % Quantil	95 % Quantil
84	45,846	0.184	0.085	0.173	0.157	0.221
30 k	140,146	0.206	0.104	0.189	0.178	0.241
100 k	371,392	0.340	0.185	0.308	0.287	0.436

Table 1.a: Change of the quantities against the 84 pixels headlamp to visualize scaling at different resolutions.

Resolution	Rays Cast	Mean	Standard Deviation	Median	5 % Quantil	95 % Quantil
84	100 %	100 %	100 %	100 %	100 %	100 %
30 k	305.6 %	111.9 %	122.3 %	109.2 %	113.3 %	109.0 %
100 k	810.0 %	184.7 %	217.6 %	178.0 %	182.8 %	197.2 %

A GPU can only execute a limited number of threads and executions truly in parallel. Also the memory bandwidth is limited. As seen in table 1.a, runtimes increase at a flatter percentage than the number of pixels and rays.

5 Conclusion & Outlook

The contribution presents the SSC approach for feedforward control of LD and HD matrix-headlights for the case of relative target utilizations. As the evaluation shows the approach can control a headlight with 100 k pixels in real time. In addition, the necessary modifications in the SSC algorithm to control LD matrix-headlights for projecting simple symbols are shown. This makes it possible to use SSC for LD and HD headlights without manual adjustments and it enables rapid prototyping of illumination functions for different beam patterns.

The next steps in the development of SSC are the integration of absolute target quantities such as the illuminance at a specific position. Compared to relative target quantities this will give lighting designers another way to specify their desires, since the target illuminance will be set independently of the default light distribution. The ability to switch between relative and absolute target values per object will expand SSC's functionality in the future.

6 References

[1] M. Waldner and T. Bertram, “Simulation of High-Definition Pixel-Headlights”, 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020

[2] L. Yang, S. Liu and M. Salvi, “A Survey of Temporal Antialiasing Techniques”, Computer Graphics Forum, Wiley Online, p 607-621, May 2020.

[3] R. Kauschke, et al. "More Efficient Development of Headlamp Systems." ATZ worldwide 123.2, p 14-19, 2021

[4] J. Dean and S. Ghemawat, ”MapReduce: Simplified Data Processing on Large Clusters”, OSDI'04: Sixth Symposium on Operating System Design and Implementation, p 137—150, 2004

[5] M. Waldner and T. Bertram, “Optimal Real-time Digitization of Matrix-Headlights”, IEEE/ASME International Conference on Advanced Intelligent Mechatronics 2021 (AIM 2021), July 2021, *(Accepted and finally submitted)*

[6] NVIDIA (ed.), CUDA C++ 11.3, April 15, 2021, https://developer.nvidia.com /cuda-toolkit, https://docs.nvidia.com/cuda/

[7] G. Bradski., “The OpenCV Library”, Dr. Dobb's Journal of Software Tools, 2000, Source Code Ver. 4.5 https://github.com/opencv/opencv (2021)