LightOpt: Lights Optimization for Real-time Rendering

TUO CHEN*, Tsinghua University, China
LUYAN CAO†, LIGHTSPEED, China
KUI WU, LIGHTSPEED, USA
SHIMIN HU, Tsinghua University, China

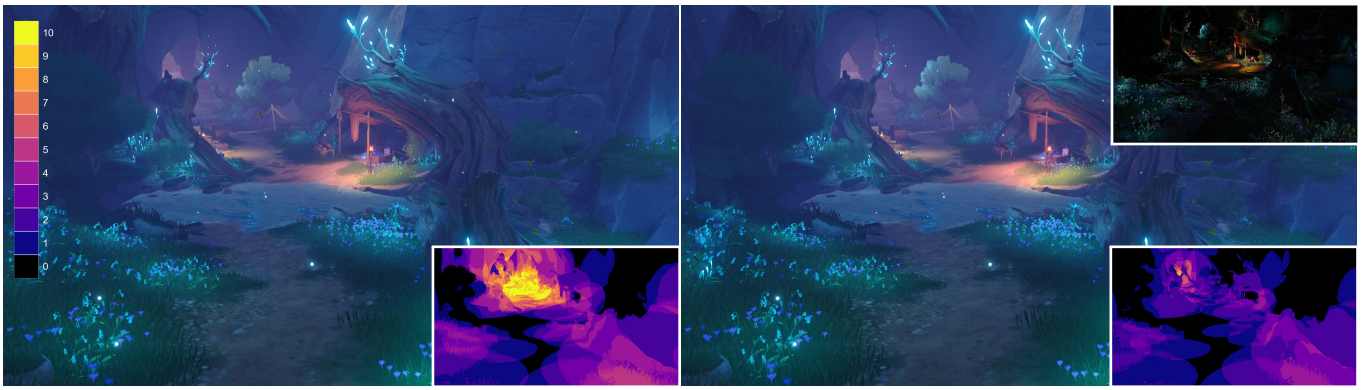

Fig. 1. *Given a game scene with 121 lights (left), our method automatically reduces the number of lights while preserving the original appearance in the optimized result (right), through a differentiable rendering pipeline equipped with novel loss functions. Our method reduces the number of lights to 57 and achieves speedups of* 1.3× *to* 2.1× *throughout the scene. Top-right: 5× magnified per-pixel difference relative to the original (RMSE=0.0552). Bottom-right: light overdraw heatmap (yellow = high, black = low). The average light overdraw (ALO) is reduced from 2.4 to 1.7.*

Modern game scenes often require hundreds of lights to achieve rich visual detail, yet evaluating many lights per pixel remains a major performance bottleneck in real-time rendering, especially on low-end platforms. We present a differentiable lighting optimization pipeline that automatically reduces the number of active lights while preserving the final rendered appearance. Our method jointly optimizes light parameters and dynamically performs light removal, merging, and insertion during optimization. We introduce two loss functions to minimize redundant light overlap and prevent under-illumination, and extend our approach to support Time-of-Day lighting. Experiments on production game scenes show that our method reduces lighting cost by 18–52% while consistently outperforming artist-authored light reductions in visual fidelity.

CCS Concepts: • **Computing methodologies** → **Rendering**.

ACM Reference Format:
Tuo Chen, Luyan Cao, Kui Wu, and Shimin Hu. 2026. LightOpt: Lights Optimization for Real-time Rendering. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3799902.3811072

*This work was done when Tuo Chen was an intern at LIGHTSPEED.
†Corresponding author

SIGGRAPH Conference Papers '26, July 19–23, 2026, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07.
https://doi.org/10.1145/3799902.3811072

1 INTRODUCTION

Lighting plays a central role in game scenes by shaping visual readability, depth perception, and emotional response. Modern game scenes often require a large number of light sources to faithfully represent complex environments. Multiple lights enable designers to shape the mood, emphasize points of interest, and dynamically respond to gameplay events across different regions of the scene. As a result, it is not uncommon for a single scene to contain hundreds of lights, with 10–50 lights overlapping in visually dense areas. In real-time rendering, however, lighting cost scales with the number of lights evaluated per visible pixel. Additional lights lead to substantial overhead due to the computational complexity of BRDF evaluation, pixel divergence caused by spatially varying light influence, and reduced cache coherence, particularly on low-end GPUs. Although numerous techniques have been proposed to mitigate this cost, including tile-based light culling [Andersson 2009; Lauritzen 2010; Olsson and Assarsson 2011], cluster-based light culling [Olsson et al. 2012], and tiled-Z culling [Drobot 2017], efficiently rendering scenes with a large number of lights remains a persistent challenge.

This challenge is further amplified in cross-platform game development, where comparable lighting quality is expected across PC, console, and mobile devices. Due to limited computational budgets, low-end platforms can allow only a small number of active light sources at interactive frame rates. Maintaining separate lighting configurations, one with many lights for high-end platforms and another with fewer lights but similar visual appearance for low-end, is labor-intensive and difficult to maintain, placing a significant burden on lighting artists and the production pipeline.

In this work, we present the first differentiable lighting optimization pipeline that reduces the number of active lights while jointly optimizing light parameters, such as position, intensity, color, and direction, to preserve the final rendered appearance of the scene. Our framework supports dynamic operations on the light set during optimization, including removal, merging, and addition of lights. To guide this process, we introduce two novel loss functions that respectively minimize redundant light overlap and prevent under-illumination. Furthermore, we extend our pipeline to handle Time-of-Day (ToD) lights, a common game feature for simulating dynamic natural illumination throughout the day. We evaluate our pipeline across several real game scenes and demonstrate that it can significantly reduce the number of active lights by 20–53%, lowering overall lighting costs by 18–52%. Compared to manually optimized light reductions performed by artists, our method requires only a few minutes to optimize and consistently outperforms the view-dependent industrial common practice in terms of deferred shading time and RMSE of the final rendered results. Cross-platform performance evaluations also validate the effectiveness of our approach.

2 RELATED WORK

Many lights problem. In real-time contexts, many-light direct lighting alone is a challenging problem with a very limited budget compared to offline rendering, restricting the number of lights (typically hundreds to thousands) [Dachsbacher et al. 2014]. Light culling methods are the most widely used approach for handling multiple light sources in real-time rasterization. They subdivide the view frustum into sub-frustums, maintain a light list for each sub-frustum by recording the lights that affect it, and shade each pixel using the light list of the corresponding sub-frustum. Tile-based light culling [Andersson 2009; Lauritzen 2010; Olsson and Assarsson 2011] divides the screen into regular 2D tiles, each tile corresponding to a sub-frustum. Cluster-based light culling [Olsson et al. 2012; Persson 2013] further extends the 2D tiling into a 3D grid of clusters by introducing a depth axis, achieving tighter culling with a light list for each 3D cluster (sub-frustum). Tiled-Z culling [Drobot 2017] preserves the simplicity of 2D tiles while achieving much of the depth discrimination by introducing Z-bins.

In ray-tracing pipelines, the dominant cost arises from light sampling rather than per-light evaluation. Light hierarchies enable importance sampling over many lights but incur nontrivial overhead to build and maintain [Conty Estevez and Kulla 2018; Lin and Yuksel 2020; Moreau et al. 2022; Yuksel 2019]. Adaptive direct-lighting PDFs and path-guiding methods [Dahm and Keller 2017; Fan et al. 2023, 2025; Herholz et al. 2016; Hey and Purgathofer 2002; Jensen 1995; Lafortune and Willems 1995; Li et al. 2022; Liang et al. 2024; Müller et al. 2017; Vorba et al. 2019, 2014] typically require high per-pixel sample counts, while more recent variants achieve real-time performance using additional data structures or simplified distributions [Dittebrandt et al. 2023; Lu et al. 2024; Zeng et al. 2025]. ReSTIR [Bitterli et al. 2020] improves efficiency via resampled importance sampling but introduces a large constant overhead [Wyman and Panteleev 2022]. In practice, systems such as Unreal Engine MegaLights [Narkowicz and Costa 2025] build per-tile visible-light lists from empirically traced visibility and use them as sampling priors for stochastic direct lighting. Overall, these approaches address the many-light problem at the algorithmic level by improving sampling or culling, without altering the underlying set of light sources used for rendering. The another line of works [Atanasov and Koylazov 2025; Velázquez-Armendáriz et al. 2015; Zhu et al. 2021] concentrate on the representation and sampling of complex luminaries, whereas our method optimizes the collective behavior of multiple simple lights at the scene level.

LightCuts. LightCuts and its variants are closely related to our work in that they share the high-level goal of reducing the cost of many-light rendering while preserving the appearance of the original illumination through a compact approximation of the light set. Walter et al. [2006, 2005, 2012] construct a hierarchical representation of the original lights and, for each pixel, select a subset of lights (a lightcut) that approximates the full lighting contribution. MRCS [Hašan et al. 2007, 2008] computes a unified cut shared across all pixels, while LightSlice [Ou and Pellacini 2011] extends this idea to tile-based cuts. Because these approaches rely on view-dependent cut evaluation, they are costly to recompute as the viewport changes in real-time rendering. Conceptually, these methods exploit redundancy among lights by selecting representative subsets from a fixed light set at runtime. In contrast, our framework performs simplification at the level of the light set itself: rather than selecting local representatives during rendering, we directly optimize and restructure the scene-wide lighting configuration to produce a new compact set of lights. Our method is orthogonal to runtime light-selection algorithms and can be used as a preprocessing step to reduce both the total number of light sources and per-sub-frustum light lists.

Differentiable rendering applications. Differentiable rendering has been widely used for inverse graphics problems, including geometry reconstruction, physics simulation, material and texture estimation, and lighting optimization, where scene parameters are optimized by optimizing variance [Yan et al. 2024]. A variety of methods leverage differentiable rendering [Li et al. 2018; Nimier-David et al. 2019] to recover 3D geometry, lighting, and material properties from 2D images [Hasselgren et al. 2022; Ling and Zhang 2023; Munkberg et al. 2022; Xie et al. 2024], as well as to simplify mesh [Hasselgren et al. 2021], optimize hair card geometry [Zheng et al. 2025] or jointly refine UVs and textures [Knodt et al. 2023], typically driven by screen-space reconstruction metrics. In contrast to methods that focus on final image reconstruction, we formulate a multi-objective optimization that directly accounts for surface irradiance sampled across the geometry. Lipp et al. [2024] addresses target-driven lighting design by continuously optimizing lighting configurations via adjoint light tracing, whereas our method focuses on real-time rendering and discrete light-set restructuring, including light removal, merging, and addition, during optimization.

3 DIFFERENTIABLE LIGHT OPTIMIZATION

In this section, we first present the problem statement, then provide an overview, and finally detail each component.

3.1 Problem Statement & Method Overview

In real-time rendering, lighting cost scales with the number of light sources evaluated per visible pixel. Each additional light incurs overhead from BRDF evaluation, increased pixel divergence due to spatially varying light influence, and reduced cache coherence, effects that are especially pronounced on low-end GPUs. Hence, *we solve a multi-objective optimization problem under a rendering pipeline to trade off visual fidelity against light complexity.*

Irradiance field. Ideally, the number of lights should be reduced while preserving the original visual appearance. However, the rendering equation embeds the BRDF within a hemispherical integral, leaving little room for direct optimization. Instead, we use the irradiance field as the target to approximate the appearance. In particular, the surface irradiance at a point on a scene surface is defined as the total incoming radiant power per unit area, integrated over all incident directions in the hemisphere above the surface. Let $\mathbf{x}$ be a point on a surface with unit normal $\mathbf{n}$. The irradiance $E(\mathbf{x})$ from a set of light sources $\mathcal{L}$ is given by

$$E(\mathbf{x}) = \sum_{l_i \in \mathcal{L}} \int_{\Omega^+} L_i(\mathbf{x}, \boldsymbol{\omega})\ \mathbf{n} \cdot \boldsymbol{\omega}\ d\boldsymbol{\omega} \ = \sum_{l_i \in \mathcal{L}} E_i(\mathbf{x})\ , \tag{1}$$

where Ω^+ denotes the hemisphere centered around $\mathbf{n}$, and L_i and E_i denote the incoming radiance and irradiance from the i-th light l_i.

Differentiable light optimization framework. Given a game scene containing multiple lights, our framework first uniformly samples the scene surfaces and computes the resulting irradiance field, which serves as the optimization target. The original lights are optimized using an iterative scheme that alternates between two substeps. In the first substep, assuming a fixed number of lights, we optimize the light parameters to preserve the original irradiance field by solving

$$\underset{\{\mathbf{c}_i, I_i, \mathbf{s}_i, \mathbf{d}_i, \dots\}}{\arg\min} \int_{\mathrm{S}} (\hat{E}(\mathbf{x}) - E(\mathbf{x}))^2 ds + a^{\mathrm{o}} l^{\mathrm{o}} + a^{\mathrm{a}} l^{\mathrm{a}}\ , \tag{2}$$

where S is the visible surface over the scene, $\{\mathbf{c}_i, I_i, \mathbf{s}_i, \mathbf{d}_i, \dots\}$ indicate light configurations (detailed in *Remark*), $E(\mathbf{x})$ and $\hat{E}(\mathbf{x})$ are the original and optimized surface irradiance, and $a^\star$ and $l^\star$ denote the corresponding regularization weights and loss terms, which are described in the following sections. We solve this problem using an irradiance-driven optimization framework with differentiable rendering. In the second substep, we update the number of light sources by removing unimportant lights, merging redundant lights, and introducing new ones to compensate for under-illuminated regions. The pseudo-code of our pipeline is shown in Alg. 1.

Algorithm 1: Optimization pipeline

```
Construct sample set S_all (Sec. 3.2)
Compute the original irradiance E(x) for each sample
for each iteration do
    Minimize Eq. 2 using differentiable rendering until convergence
      (Sec. 3.3 and 3.4)
    Remove unimportant light sources (Sec. 3.5)
    Merge lights (Sec. 3.6)
    Add new lights (Sec. 3.7)
end
```

Remark. While our framework is not inherently limited to specific light types, we focus on point lights and spotlights, as they are the most commonly used in the game industry. In practice, spotlight implementations vary across engines; our framework follows the Unity lighting model [Unity 2023], first introduced in Frostbite [Lagarde and De Rousiers 2014] and widely adopted in real-time games. The incident irradiance arriving at $\mathbf{x}$ from the i-th light source is modeled as

$$E_i(\mathbf{x}) = V_i(\mathbf{x})\ \mathbf{c}_i I_i D_i(\mathbf{x})\ A_i(\boldsymbol{\omega}_i)(\mathbf{n} \cdot \boldsymbol{\omega}_i)\ , \tag{3}$$

where $V_i(\mathbf{x})$ is the visibility term and is assumed to be 1, $\boldsymbol{\omega}_i$ is the normalized direction from $\mathbf{x}$ to the light position $\mathbf{s}_i$, and $\mathbf{c}_i$ and I_i denote the light color and intensity, respectively. The terms D_i and A_i represent distance-based and angular attenuation, defined as

$$D_i(\mathbf{x}) = \frac{a_i}{a_i + ||\mathbf{s}_i - \mathbf{x}||^2} \max(0.0, 1.0 - \frac{||\mathbf{s}_i - \mathbf{x}||^2}{r_i^2})^2\ , \tag{4}$$

$$A_i(\boldsymbol{\omega}) = \begin{cases} \mathrm{clamp}(\frac{(-\boldsymbol{\omega}) \cdot \mathbf{d}_i - \cos\theta_i^{\mathrm{o}}}{\cos\theta_i^{\mathrm{i}} - \cos\theta_i^{\mathrm{o}}}, 0.0, 1.0)^2, & \text{if spotlight} \\ 1.0, & \text{otherwise} \end{cases} \tag{5}$$

where a_i is an attenuation smoothness parameter, r_i is the light range, $\mathbf{d}_i$ denotes the spotlight direction, and θ_i^{o} and θ_i^{i} are the outer and inner cone angles of the spotlight. For point lights, the angular attenuation $A_i(\boldsymbol{\omega})$ is identically one.

3.2 Discretization

To efficiently evaluate the irradiance field defined in Eq. 1 as well as other regularizations, we discretize the scene surface into samples, as shown in Fig. 2. We first uniformly sample the scene surface using Poisson disk sampling [Yuksel 2015], expand each light range by a factor of two, and mark all reachable samples as $\mathcal{S}_{\mathrm{all}}$, which is composed of illuminated samples and non-illuminated ones. Our objective is twofold: to accurately preserve the irradiance of illuminated samples, while ensuring that non-illuminated samples remain dark. Furthermore, because global illumination from multiple bounces is typically prebaked in standard game development, we restrict our formulation to direct illumination. Hence, the first term in Eq. 2 can be rewritten as follows:

$$\int_{\mathrm{S}} (\hat{E}(\mathbf{x}) - E(\mathbf{x}))^2 ds \approx \Delta s \sum_{\mathbf{x} \in \mathcal{S}_{\mathrm{all}}} \left(\hat{E}(\mathbf{x}) - E(\mathbf{x})\right)^2\ , \tag{6}$$

where $\hat{E}(\mathbf{x}) = \sum_{l_i \in \hat{\mathcal{L}}} E_i(\mathbf{x})$, $\hat{\mathcal{L}}$ is the optimized light set, and we only evaluate on the sample from set $\mathcal{S}_{\mathrm{all}}$, $\Delta s = \pi r_0^2$ denotes the sample area, and r_0 is the Poisson sample radius.

3.3 Overlap Loss

In practical real-time rendering pipelines, such as tiled [Olsson and Assarsson 2011], tiled-Z [Drobot 2017], and clustered [Olsson et al. 2012] rendering, the computational cost per tile is approximately proportional to the number of pixels per tile multiplied by the length of the associated light list. Because directly optimizing per-view, per-tile light lists is impractical, we introduce an approximate overlap loss l^{o} that penalizes excessive light overlap at surface samples. In practice, this loss effectively reduces the average light list length per tile, thereby improving rendering efficiency.

Fig. 2. *Example of samples over the surface.*

At each optimization iteration, we first identify the set of light sources that can influence each surface sample $\mathbf{x}$, denoted by $\mathcal{L}_\mathbf{x}$. We then define a global overlap penalty over all samples:

$$l^{\mathrm{o}} = \sum_{\mathbf{x}\in\mathcal{S}_{\mathrm{all}}} \left[\alpha(|\mathcal{L}_\mathbf{x}|) \sum_{l_i\in\mathcal{L}_\mathbf{x}} w_i(\mathbf{x})\, d_i(\mathbf{x})\right], \tag{7}$$

where $|\mathcal{L}_\mathbf{x}|$ denotes the size of set $\mathcal{L}_\mathbf{x}$ and the scaling function

$$\alpha(n) = (\max(0, n - n^{\max}))^2 \tag{8}$$

penalizes samples influenced by more than a user-specified maximum allowed overlap threshold $n^{\max}$. This encourages sparsity in light coverage while preserving sufficient illumination.

To remove lights with weaker contributions while retaining dominant ones, we define a soft weighting function

$$w_i(\mathbf{x}) = \frac{\exp\left(-\beta_w\, E_i(\mathbf{x})\right)}{\sum_{l_j\in\mathcal{L}_\mathbf{x}} \exp\left(-\beta_w\, E_j(\mathbf{x})\right)}, \tag{9}$$

where β_w controls the sharpness of the weighting. It is inversely related to $E_i(\mathbf{x})$, the irradiance from the i-th light. Note that the light influence is evaluated using a doubled light range, as shown in Fig. 3, ensuring differentiability when lights move across influence boundaries.

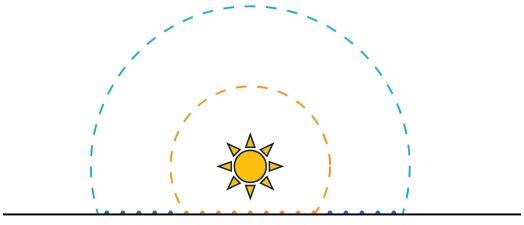

Fig. 3. *Extending the light source range by a factor of two ensures differentiability. During optimization, both illuminated samples (orange) and non-illuminated samples (blue) are taken into account.*

For point lights, the geometric penalty term (a smooth approximation of the interior distance-to-boundary) is defined as

$$d_i^{\mathrm{point}}(\mathbf{x}) = \mathrm{softplus}(r_i - \|\mathbf{x} - \mathbf{s}_i\|)\,. \tag{10}$$

For spotlights, we additionally account for the conical boundary by defining the distance term as the minimum of the radial and angular distances as

$$d_i^{\mathrm{spot}}(\mathbf{x}) = \mathrm{softmin}\big(r_i - \|\mathbf{x} - \mathbf{s}_i\|, \|\mathbf{s}_i - \mathbf{x}\|\sin(\theta_i^{\mathrm{o}} - \arccos(\boldsymbol{\omega}_i \cdot \mathbf{d}_i))\big)\,, \tag{11}$$

where the smooth approximations are defined as

$$\mathrm{softplus}(a) = \frac{1}{\beta_p}\ln(1 + \exp(\beta_p a))\,, \tag{12}$$

$$\mathrm{softmin}(a, b) = \frac{a\exp(-\beta_m a) + b\exp(-\beta_m b)}{\exp(-\beta_m a) + \exp(-\beta_m b)}\,, \tag{13}$$

where β_p and β_m are user-defined smoothness parameters. In all experiments, we set $\beta_w = \beta_p = \beta_m = 1$.

3.4 Attraction Loss

While the overlap loss effectively discourages redundant illumination by pushing lights away from densely lit regions, it can inadvertently lead to under-illuminated areas. The first term in Eq. 2 alone is inadequate for under-lit regions: if a sample falls outside the extent of all existing lights, it yields little effective gradient to attract a distant light toward that region. To address this issue, we introduce a geometric attraction loss that encourages nearby lights to move toward or expand coverage of under-lit regions. The set of under-lit samples is defined as

$$\mathcal{S}_{\text{under-lit}} = \{\mathbf{x} \mid \hat{E}(\mathbf{x}) \le \epsilon_{\mathrm{u}} E(\mathbf{x})\}\,, \tag{14}$$

where $\epsilon_{\mathrm{u}} = 0.1$ is a user-controlled threshold.

For each under-lit sample $\mathbf{x}$, we consider a set of candidate lights that are both geometrically valid and sufficiently close:

$$\mathcal{L}_{\mathbf{x},\text{attraction}} = \{l_i | (\mathbf{s}_i - \mathbf{x})\cdot\mathbf{n} > 0 \text{ and } \|\mathbf{s}_i - \mathbf{x}\| < \eta_i\}, \tag{15}$$

where the first condition ensures the light l_i is on the front-facing side of the surface. The adaptive search radius is defined based on the light's effective range as

$$\eta_i = \left(\frac{3v}{4\pi} + r_i^3\right)^{1/3} - r_i, \tag{16}$$

with v denoting a uniform volume threshold. This formulation grants lights with smaller ranges a proportionally larger extended search region, allowing them to compensate more aggressively for under-lit samples. Finally, we define the attraction loss as a simple geometric penalty

$$l^{\mathrm{a}} = \sum_{\mathbf{x}\in\mathcal{S}_{\text{under-lit}}} \sum_{l_i\in\mathcal{L}_{\mathbf{x},\text{attraction}}} d_i(\mathbf{x})\,. \tag{17}$$

where the distance function $d_i(\mathbf{x})$ is defined as Eq. 10 and 12 for point and spot light, respectively. Intuitively, this loss pulls nearby lights toward under-lit regions or extends their effective range, compensating for illumination deficits by minimizing overlap.

Remark. Both the overlap loss and the attraction loss penalize distances to light boundaries, but they operate on different sample sets. For the overlap loss, samples are drawn from regions covered by multiple lights, and minimizing the boundary-distance penalty encourages lights to reduce redundant overlap. In contrast, for the attraction loss, samples are drawn from regions outside all light

boundaries, and minimizing the distance encourages lights to move toward under-illuminated regions.

3.5 Light Removal

For each light i, we quantify its importance by the total irradiance it contributes to the scene irradiance field,

$$\gamma_i = \sum_{\mathbf{x}_j \in \mathcal{S}_i} E_i(\mathbf{x}_j) := \gamma(\mathcal{S}_i) \,, \tag{18}$$

where $\mathcal{S}_i$ denotes the set of samples illuminated by light i, $\mathbf{x}_j$ is the position of sample j, and $\gamma(\mathcal{S}_i)$ is further defined as the importance of the sample cluster lighted by light i. Prior to optimization, we prune lights whose importance γ_i falls below a user-defined importance threshold ϵ_i, as they have negligible impact on the overall irradiance field.

3.6 Light Merging

Light merging aims to reduce the total number of light sources by replacing a pair of lights with a single representative light.

Candidate pair selection. Lights with similar spatial influence should be merged to minimize visual deviation. We therefore measure the similarity between two lights using the Chamfer distance between the corresponding sets of surface samples. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ denote the illuminated sample sets of two lights; their distance is defined as

$$d(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{|\mathcal{S}_1|} \sum_{\mathbf{x} \in \mathcal{S}_1} \min_{\mathbf{p} \in \mathcal{S}_2} \|\mathbf{p} - \mathbf{x}\|^2 + \frac{1}{|\mathcal{S}_2|} \sum_{\mathbf{x} \in \mathcal{S}_2} \min_{\mathbf{p} \in \mathcal{S}_1} \|\mathbf{p} - \mathbf{x}\|^2$$

The merging cost between lights i and j is defined as

$$M(i, j) = \gamma_i \, \gamma_j \, d(\mathcal{S}_i, \mathcal{S}_j) \,, \tag{19}$$

where γ_i and γ_j denote the relative importance of the two lights.

Initialize new light. Based on the score, we select the lowest 10% pairs and merge them if the new light reduces the overlap loss and more than 80% samples are front-facing to the new light. After merging two lights, we initialize the parameters of the resulting light using importance-weighted averages of the original lights' color, position, and attenuation smoothness. The weighted position is subsequently used to determine the range and the spotlight cone, ensuring that the merged light fully covers both illumination point sets. If the required spotlight angle exceeds a predefined threshold A_{point}, the merged light is converted into a point light; otherwise, it is retained as a spotlight. This procedure yields a reliable initialization for the merged light, while deferring finer adjustments to the following optimization. Note that although the spotlight angle is defined over the range $[0°, 180°]$, we set $A_{\text{point}} = 130°$. In practice, light artists rarely use spotlight angles larger than $130°$, and we follow the same rule of thumb.

3.7 Light Addition

Light removal, merging, and optimization can introduce underexposed regions that are previously illuminated by the original lighting configuration but are no longer sufficiently covered by the optimized lights. While continuous light optimization can mitigate mild underexposure, it cannot recover regions that lack a nearby light source. Hence, our framework also adds new light during the optimization.

We identify under-lit samples as mentioned in Sec. 3.4. The under-lit samples are then grouped into clusters using DBSCAN [Ester et al. 1996], which uses Euclidean distance on sample positions. For each cluster, if its accumulated importance exceeds the threshold ϵ_i mentioned in Sec. 3.5, we spawn a new light source to compensate for the missing illumination. The parameters of the new light are initialized from the cluster statistics: its initial color, intensity, and direction are set based on the cluster's average target irradiance $\bar{E}$ and normal $\bar{\mathbf{n}}$. The final light position is determined by searching from the averaged position $\bar{\mathbf{x}}$ along $\bar{\mathbf{n}}$ to minimize the irradiance difference over all samples within the cluster. The light type (point light or spotlight) is then selected according to the same criterion used in the light merging.

3.8 Time-of-Day Lighting

In practical real-time rendering pipelines, lighting configurations are often time-dependent, as in Time-of-Day (ToD) setups where light intensity, color, and activation states vary continuously over time. This temporal variation further complicates the optimization problem. In production systems, each ToD light i is typically specified by a sparse set of keyframes $\mathcal{T}_i = \{t_1, \ldots, t_{K_i}\}$ with light parameters interpolated piecewise linearly between keyframes. To support ToD lighting, we define a global time set $\mathcal{T} = \bigcup_{i=1} \mathcal{T}_i$ which aggregates all keyframes across lights. We then extend the optimization in Eq. 2 by introducing an explicit time dimension and applying numerical quadrature over these keyframes:

$$\underset{\{\mathbf{c}_i, I_i, \mathbf{s}_i, \mathbf{d}_i, \ldots\}}{\arg\min} \sum_{k=1}^{|\mathcal{T}|} w_k \Big[\Delta s \sum_{\mathbf{x} \in \mathcal{S}_{\text{all}}} \big(\hat{E}_k(\mathbf{x}) - E_k(\mathbf{x}) \big)^2 + a^{\text{o}} I_k^{\text{o}} + a^{\text{a}} I_k^{\text{a}} \Big] \,, \tag{20}$$

where $w_k \geq 0$ are quadrature weights satisfying $\sum_{k=1}^{K} w_k = 1$. We compute these weights using the trapezoidal rule with non-uniform spacing to account for irregular keyframe timing. This formulation aligns with standard game-engine pipelines, in which ToD lighting parameters are evaluated at discrete keyframes and continuously interpolated in between.

4 RESULTS

LightOpt can be integrated seamlessly into existing tile-based rendering pipelines by preprocessing light sources without modifying the tile-based rendering algorithm. We evaluate our method on five real game scenes in Unity [Unity 2023] with tiled-Z culling in a deferred shading pipeline [Olsson and Assarsson 2011]. All results are rendered in real time at 1920×1080 on an Intel i9-12900KF with an NVIDIA GeForce RTX 3090.

Testing scenes. To evaluate our method, we conduct a comprehensive set of comparisons, evaluations, and ablation studies on five real game scenes, spanning a wide range of indoor and outdoor lighting scenarios commonly found in video games, including *Tavern*, *Valley*, *Square* (lighted by ToD lights), *Cave*, and *Street*, ranging from 80 to over 300 light sources placed by professional lighting artists. The training process takes about 10 minutes per scene. The scenes used in our experiments are already open-world environments (approximately 16km × 16km). Lights are optimized in spatial batches according to the scene partitioning.

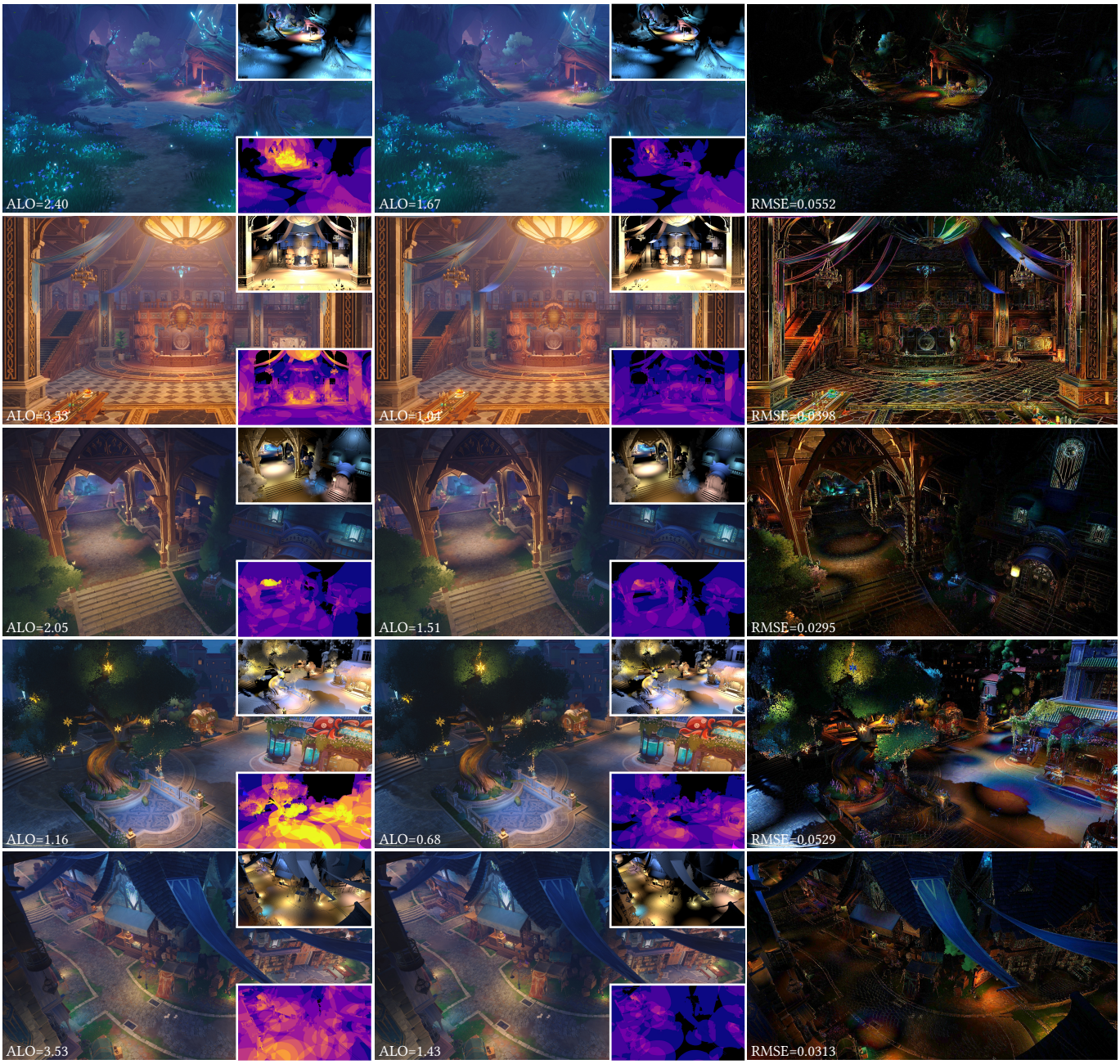

Fig. 4. *Rendering quality comparison in Valley, Tavern, Cave, Square, and Street. From left to right: original lighting, optimized lighting generated by ours, and the 5× magnified absolute per-pixel error between them. The top-right inset renders the scene with a white material (albedo removed) to highlight illumination differences, and the bottom-right inset shows per-pixel light overlap.*

Evaluation metrics. For quantitative evaluation, we use the root-mean-square error (RMSE) to quantify illumination differences. We further introduce a new metric, average light overdraw (ALO) ($|\mathcal{L}_\mathbf{x}|$ for points corresponding to each pixel; see Sec. 3.3) to quantify per-pixel light overlap in screen space. Performance is evaluated by measuring the GPU execution time of the deferred shading pass using Nsight Graphics. For each scene, we select up to eight representative camera poses and report both the average and maximum RMSE, ALO, and deferred shading time across these views.

Rendering quality. We compare the original and optimized lighting in Fig. 4. In addition to the full-material renderings, we include white-model renderings to better visualize the irradiance fields produced by both methods. Even under aggressive light reduction,

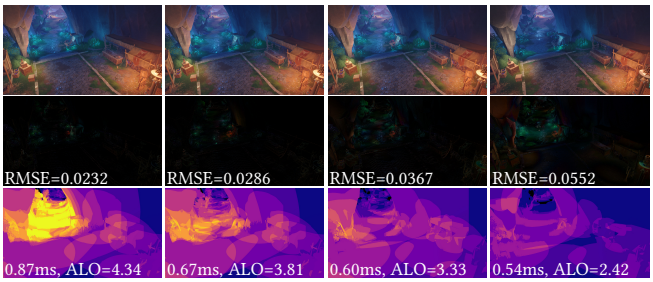

Fig. 5. *Ablation on overlap loss. From left to right:* $a^o = 0$, 0.001, 0.1, *and* 5. *From top to bottom: final renderings, per-pixel errors, and visualizations of per-pixel light overlaps, evaluated on an RTX 3090.*

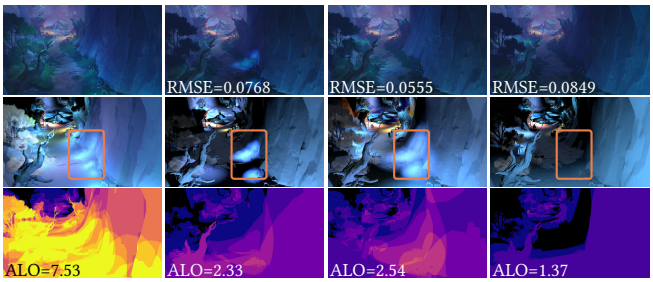

Fig. 6. *Ablation on attraction volume* v. *From left to right: input,* $v = 0$, $v = 15$, *and* $v = 100$. *From top to bottom: final renderings, white model, and visualizations of per-pixel light overlaps. Orange rectangles highlight that, without the attraction loss (*$v = 0$*), newly added lights may be pushed away from highly overlapping regions during optimization, leading to under-lit areas.*

no large-scale under- or over-illumination artifacts are observed. This indicates that the optimized light parameters robustly approximate the original appearance, with only minor deviations appearing in localized regions. The absolute-difference images indicate that LightOpt closely preserves appearance, whereas the light-overlap visualization shows a clear reduction in per-pixel light overlap (and thus the expected shading cost), with ALO decreasing by up to 50%.

Ablation studies. To evaluate the contribution of each component in our framework, we conduct ablation studies by disabling each individual module while keeping all other settings unchanged. In particular, we fix $a^{\mathrm{a}} = 0.2, r_0 = 0.25$, and choose $a^{\mathrm{o}} = 0.1, v = 15, \epsilon_i = 4$ as the default settings for all scenes. We further set $n^{\max} = 3$ to support cross-platform deployment.

Overlap loss. The overlap loss penalizes excessive light overlaps, thus reducing the number of lights contributing to each pixel and lowering the per-pixel lighting cost. Fig. 5 demonstrates a clear quality–efficiency trade-off when sweeping the overlap-loss weight a^o: a larger a^o decreases the light overlaps more radically but allows more illumination errors. Therefore, a larger a^o yields lower ALO, but increases illumination discrepancies and thus RMSE.

Attraction loss. The attraction loss encourages nearby lights to move toward under-illuminated regions. To isolate its contribution, we disable attraction loss by setting the attraction volume to $v = 0$,

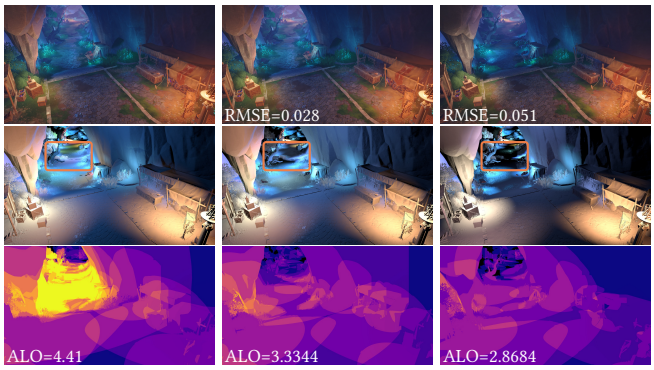

Fig. 7. *Ablation on dynamic light adjustment. From left to right: input, ours, and ours w/o dynamic light adjustment. From top to bottom: final renderings, white model, and visualizations of per-pixel light overlaps. Orange rectangles highlight dynamic adjustment that reduces under-lit artifacts while maintaining appropriate overlap.*

effectively removing the attraction region and preventing lights from compensating for illumination deficits. As shown in Fig. 6, removing the attraction loss results in pronounced under-illuminated areas and rigid boundaries between light and dark. This problem can be alleviated by introducing an attraction loss to enable the lights to detect nearby under-illuminated areas; however, overly large attraction volumes may cause the lights to expand their affected volume too aggressively and allocate illumination to regions that should not be lit, thereby reducing fine-scale lighting details.

Dynamic light adjustment. Dynamic removal, merging, and addition of lights enable the optimizer to adapt the number of lights during training, improving the final illumination. As shown in Fig. 7, without dynamic light adjustment, the highlighted regions exhibit similar levels of light overlap yet produce noticeably different illumination: our method without dynamic adjustment tends to leave these areas under-illuminated. In contrast, enabling dynamic control, together with the attraction loss, largely eliminates under-illumination artifacts. We evaluate different importance thresholds ϵ_i for light removal and addition. As shown in Fig. 8, in the highlighted region, a large ϵ_i over-prunes short-range (low-importance) lights, leading to local under-illumination and loss of fine rendering details. Moreover, an overly large threshold prunes candidate lights before merging, restricting the merge operation to the remaining lights and often producing merged lights with an excessively large range, which may lead to a larger ALO. The attraction loss and dynamic addition of lights act together to deal with under-lit samples. Table 1 reports mean and maximum RMSEs for five scenes. The attraction loss is effective when nearby lights already exist and can plausibly cover missing regions. In contrast, light addition is necessary when no suitable nearby light exists, and a new degree of freedom must be introduced.

Compared with industrial common practice and manual reduction. A common industrial practice for light reduction is to collect all lights within the view frustum at runtime, sort them by their contribution at the camera focal point (and thus in a view-dependent

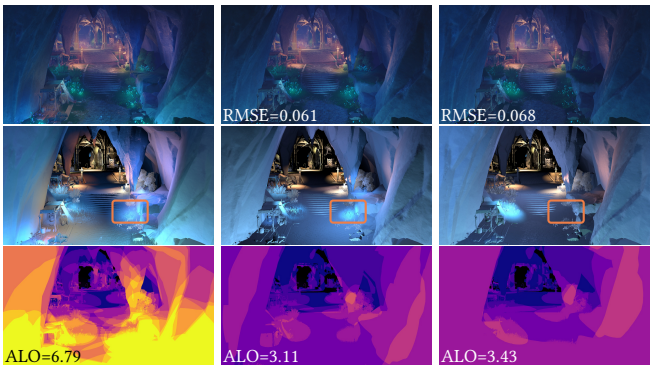

Fig. 8. *Ablation on the importance threshold ϵ_i. From left to right: input, $\epsilon_i = 4$, and $\epsilon_i = 8$. From top to bottom: final renderings, white model renderings, and visualizations of per-pixel light overlaps.*

Table 1. *Comparisons on, w/o the attraction loss (attr.), w/o light addition (add.), and w/ both, with mean/max RMSE for illumination differences.*

Method	Valley		Tavern		Square		Cave		Street	
	mean	max	mean	max	mean	max	mean	max	mean	max
w/o attr.	0.066	0.106	0.074	0.089	0.098	0.110	0.078	0.103	0.080	0.087
w/o add.	0.053	0.089	0.067	0.082	0.096	0.109	0.039	0.055	0.066	0.073
w/ both	**0.045**	**0.068**	**0.041**	**0.061**	**0.043**	**0.053**	**0.037**	**0.049**	**0.033**	**0.034**

manner), and then prune the set to a fixed budget before submitting it to the rendering pipeline. Following this setting, we strictly cap the maximum number of lights per frustum at 32 at runtime. Fig. 9 shows that the industrial baseline additionally relies on camera pose as input and tends to discard distant lights while retaining nearby ones, resulting in only marginal speedups while noticeably degrading distant illumination details, while artist-authored results aggressively remove illumination to meet real-time constraints on low-end devices, but with substantially higher RMSE.

Performance. Across all tested scenes, our method consistently reduces ALO, resulting in performance gains in the deferred shading pass, as summarized in Table 2. The industrial method uses camera poses to prune unimportant lights, but loses more detail and gives less speedup than ours. In these scenes, LightOpt gives an average speedup ($T_{\text{Input}}/T_{\text{Ours}}$) of 1.67× (Valley), 1.37× (Tavern), 1.48× (Square), 1.44× (Cave), and 1.52× (Street), and reaches a maximum speedup of 2.07× for complex light settings (Fig. 9).

ToD. We evaluate LightOpt on the Square scene with ToD lights. Fig. 10 shows that LightOpt effectively optimizes ToD lighting. The camera roaming video of Square also shows the ToD transition.

Cross-platform performance. To evaluate the effectiveness of LightOpt across different hardware platforms, we conduct additional performance tests on medium-end and low-end devices, in addition to the high-end GPU used in previous experiments. The medium-end platform is equipped with an Intel i9-9900 CPU and an NVIDIA GeForce RTX 2080 GPU, while the low-end platform uses an integrated GPU, Intel(R) UHD Graphics 630. For high-end and medium-end GPUs, we measure the GPU time of the deferred shading pass using NVIDIA Nsight Graphics. Since Nsight Graphics does not support integrated GPUs, we use RenderDoc to estimate the deferred shading time on the low-end platform. Although the profiling tools differ, both measurements consistently reflect the relative cost of per-pixel lighting evaluation.

Table 3 reports the mean and maximum deferred shading time for the test scenes. LightOpt consistently reduces shading cost, and the relative speedup remains comparable between high-end, medium-end, and low-end devices. Notably, the absolute reduction in shading time is most significant on low-end hardware, highlighting the practical benefit of LightOpt for performance-constrained platforms. For example, in the Valley scene, LightOpt reduces the average deferred shading time by up to 39.7 ms on the integrated GPU.

Table 2. *Quantitative comparisons against input lights, industrial solution, and ours in terms of total light number as well as mean/max ALO, deferred shading time in ms, and RMSE. * denotes the industrial method that only keeps a fixed number, 32, of lights in rendering, and the lights it keeps vary in different viewports. † denotes scenes with ToD lighting. ‡ denotes artist manual authored result.*

Scene	Method	Light #	ALO		Shading (ms)		RMSE	
			mean	max	mean	max	mean	max
Valley	Input	121	5.78	7.86	0.95	1.16	-	-
	Industrial	32*	4.46	7.05	0.75	1.10	0.055	0.076
	Manual‡	83	**2.12**	**2.60**	**0.54**	**0.59**	0.073	0.107
	Ours	57	2.54	3.76	0.57	0.63	**0.046**	**0.068**
Tavern	Input	210	3.35	4.06	0.71	0.80	-	-
	Industrial	32*	2.51	3.09	0.58	0.60	0.071	0.104
	Ours	126	**1.72**	**1.97**	**0.52**	**0.55**	**0.041**	**0.061**
Square†	Input	314	3.28	5.43	0.83	1.16	-	-
	Industrial	32*	2.00	4.17	0.60	0.93	0.073	0.086
	Ours	249	**1.57**	**2.13**	**0.56**	**0.68**	**0.043**	**0.053**
Cave	Input	80	3.43	6.13	0.71	1.05	-	-
	Industrial	32*	2.73	5.35	0.60	0.88	0.041	0.056
	Ours	64	**2.05**	**3.30**	**0.49**	**0.60**	**0.037**	**0.049**
Street	Input	135	3.83	4.82	0.67	0.72	-	-
	Industrial	32*	3.24	3.79	0.55	0.62	0.037	0.045
	Ours	95	**1.68**	**2.23**	**0.45**	**0.49**	**0.033**	**0.034**

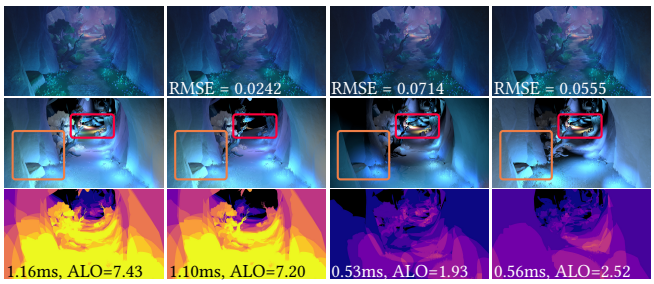

Fig. 9. *Comparison between different methods. From left to right: input, industrial, manual, and ours. From top to bottom: final renderings, white model rendering, and visualizations of per-pixel light overlaps.*

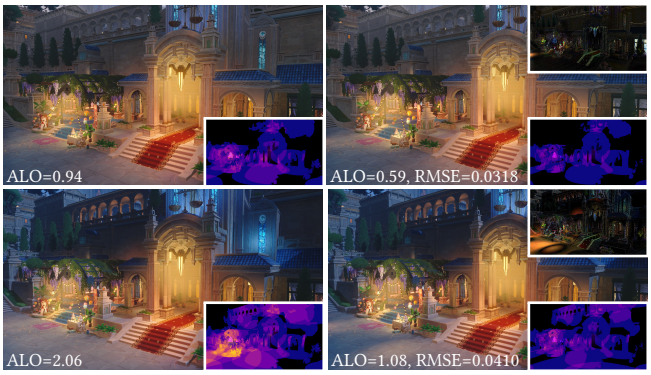

Fig. 10. *Scene "Square" with ToD lights. From left to right: input and optimized by ours. The bottom-right visualizes per-pixel light overlap, and the top-right is 5× magnified absolute per-pixel error.*

Table 3. *Cross-platform performance evaluation of the deferred shading pass. We report the mean/max GPU time (in ms) for the original lighting configuration (Input) and our optimized result (Ours), together with the corresponding speedup ratios (T_{Input}/T_{Ours}). High, medium, and low platforms correspond to NVIDIA RTX 3090, NVIDIA RTX 2080, and Intel(R) UHD Graphics 630, respectively. † denotes w/ ToD lighting.*

Scene	Platform	Input (ms)		Ours (ms)		Speedup (×)	
		mean	max	mean	max	mean	max
Valley	High	0.95	1.16	0.57	0.63	1.67	1.84
	Medium	3.00	3.87	1.76	1.96	1.70	1.97
	Low	117.22	146.40	77.50	91.79	1.51	1.59
Tavern	High	0.71	0.80	0.52	0.55	1.37	1.45
	Medium	2.21	2.39	1.64	1.69	1.35	1.41
	Low	110.89	113.41	84.55	87.84	1.31	1.29
Square†	High	0.83	1.16	0.56	0.68	1.48	1.71
	Medium	2.01	2.05	1.55	1.57	1.30	1.31
	Low	84.73	93.30	69.90	72.27	1.21	1.29
Cave	High	0.71	1.05	0.49	0.60	1.45	1.75
	Medium	2.17	3.15	1.62	2.01	1.34	1.57
	Low	91.20	127.05	75.19	92.11	1.21	1.38
Street	High	0.67	0.72	0.45	0.49	1.49	1.47
	Medium	2.05	2.16	1.42	1.49	1.44	1.45
	Low	98.96	103.58	70.41	73.15	1.41	1.42

Challenging cases. It is worth noting that scenes containing foliage, thin geometry, or regions with strong light overlap can be challenging for accurately preserving irradiance. Fig. 11 shows an example with foliage and strong light overlap that becomes under-lit after optimization. Although our method does not perfectly preserve irradiance in such challenging cases, it still produces globally reasonable results across the entire scene.

5 CONCLUSION

We presented a differentiable lighting optimization pipeline that automatically reduces the number of active lights while preserving visual fidelity in real-time game scenes. By jointly optimizing light parameters and dynamically restructuring the light set, our method significantly lowers lighting cost without requiring manual intervention. The proposed loss functions effectively balance redundancy reduction and coverage preservation, and the extension to Time-of-Day lighting enables consistent results under dynamic illumination. Our approach reduces artist workload, improves cross-platform scalability, and provides a practical solution for efficient lighting in modern real-time rendering pipelines.

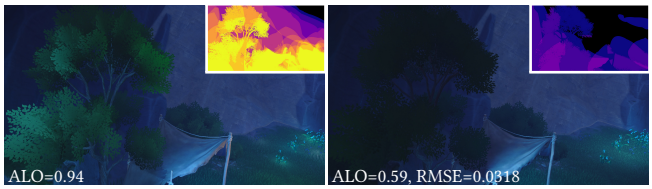

Fig. 11. *A challenging case containing foliage with strong light overlap. Left, input, and right, our optimized lighting. The top-right insets visualize per-pixel light overlap.*

Limitations. Our optimization is designed to preserve the irradiance field rather than the final shaded image. Although this approximation performs well in practice and generally preserves visual fidelity under deferred shading, it does not explicitly model view-dependent effects, such as specular highlights. As a result, minor discrepancies may appear in regions with highly glossy materials or under sharp lighting transitions. Incorporating perceptually motivated or BRDF-aware objectives could further improve accuracy in such cases. In addition, our dynamic light adjustment includes discrete removal, merging, and insertion operations. Making these steps fully differentiable is an interesting direction for future work. We also currently assume unit visibility for the optimized lights, since in practical game pipelines, only a small number of lights cast shadows due to performance constraints. Extending our framework to optimize shadow-casting lights is, therefore, another important avenue for future work, as it would require handling discontinuous visibility within differentiable optimization. Finally, our current framework does not explicitly support area lights, which would require more sophisticated differentiable treatment than point or spot lights. That said, there is no fundamental limitation preventing such an extension. In practice, however, area lights are rarely used in many real-time games. Environmental lighting is also outside the scope of this work, as it is typically authored or pre-baked by lighting artists as part of the global illumination setup.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (project No. 62220106003, 62495060), the Research Grant of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

Johan Andersson. 2009. Parallel graphics in frostbite-current & future. In *ACM SIGGRAPH 2009 Courses*. Association for Computing Machinery, New York, NY, USA.
A. Atanasov and V. Koylazov. 2025. Wavelet Representation and Sampling of Complex Luminaires. *Computer Graphics Forum* 44, 4, e70163.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (TOG)* 39, 4, Article 148 (Aug. 2020), 17 pages.

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 1–17.

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable Realistic Rendering with Many-Light Methods. *Computer Graphics Forum* 33, 1 (Feb. 2014), 88–104.

Ken Dahm and Alexander Keller. 2017. Learning light transport the reinforced way. In *ACM SIGGRAPH 2017 Talks* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 73, 2 pages.

Addis Dittebrandt, Vincent Schubler, Johannes Hanika, Sebastian Herholz, and Carsten Dachsbacher. 2023. Markov Chain Mixture Models for Real-Time Direct Illumination. *Computer Graphics Forum* 42, 4 (2023), e14881.

Michal Drobot. 2017. Improved Culling for Tiled and Clustered Rendering. In *ACM SIGGRAPH 2017 Courses*. Association for Computing Machinery, NY, USA.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Vol. 96. 226–231.

Zhimin Fan, Pengpei Hong, Jie Guo, Changqing Zou, Yanwen Guo, and Ling-Qi Yan. 2023. Manifold path guiding for importance sampling specular chains. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–14.

Zhimin Fan, Yiming Wang, Chenxi Zhou, Ling-Qi Yan, Yanwen Guo, and Jie Guo. 2025. Multiple Importance Reweighting for Path Guiding. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–11.

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, light, and material decomposition from images using monte carlo rendering and denoising. *Advances in Neural Information Processing Systems* 35 (2022), 22856–22869.

Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. 2021. Appearance-Driven Automatic 3D Model Simplification. *EGSR (DL)* 29 (2021), 85–97.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2007. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 26–es.

Miloš Hašan, Edgar Velázquez-Armendariz, Fabio Pellacini, and Kavita Bala. 2008. Tensor clustering for rendering many-light animations. In *Proceedings of the Nineteenth Eurographics Conference on Rendering* (Sarajevo, Bosnia and Herzegovina) *(EGSR '08)*. Eurographics Association, Goslar, DEU, 1105–1114.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum* 35, 4 (2016), 67–77.

Heinrich Hey and Werner Purgathofer. 2002. Importance sampling with hemispherical particle footprints. In *Proceedings of the 18th Spring Conference on Computer Graphics (SCCG '02)*. Association for Computing Machinery, New York, NY, USA, 107–114.

Henrik Wann Jensen. 1995. Importance Driven Path Tracing using the Photon Map. In *Rendering Techniques '95*, Patrick M. Hanrahan and Werner Purgathofer (Eds.). Springer Vienna, Vienna, 326–335.

Julian Knodt, Zherong Pan, Kui Wu, and Xifeng Gao. 2023. Joint UV Optimization and Texture Baking. *ACM Transactions on Graphics (TOG)* 43, 1 (2023), 20 pages.

Eric P. Lafortune and Yves D. Willems. 1995. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In *Rendering Techniques '95*, Patrick M. Hanrahan and Werner Purgathofer (Eds.). Springer Vienna, Vienna, 11–20.

Sebastien Lagarde and Charles De Rousiers. 2014. Moving frostbite to physically based rendering 3.0. *SIGGRAPH Course* 3 (2014).

Andrew Lauritzen. 2010. Deferred rendering for current and future rendering pipelines. In *ACM SIGGRAPH 2010 Courses*. Association for Computing Machinery, New York, NY, USA, 1–34.

He Li, Beibei Wang, Changhe Tu, Kun Xu, Nicolas Holzschuch, and Ling-Qi Yan. 2022. Unbiased Caustics Rendering Guided by Representative Specular Paths. In *SIGGRAPH Asia 2022 Conference Papers*. Association for Computing Machinery, New York, NY, USA, Article 41, 8 pages.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

Yuzhi Liang, Tao Liu, Yuchi Huo, Rui Wang, and Hujun Bao. 2024. Adaptive sampling and reconstruction for gradient-domain rendering. *Computational Visual Media* 10, 5 (2024), 885–902.

Daqi Lin and Cem Yuksel. 2020. Real-time stochastic lightcuts. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 1 (2020), 1–18.

Fei Ling and Frei Zhang. 2023. Machine Learning Summit: Differentiable Rendering for Scalable Asset Pipeline in 'Honor of Kings'. GDC Vault, Game Developers Conference 2023, Tools Summit presentation. Tencent Technology Inc..

Lukas Lipp, David Hahn, Pierre Ecormier-Nocca, Florian Rist, and Michael Wimmer. 2024. View-Independent Adjoint Light Tracing for Lighting Design Optimization. *ACM Trans. Graph.* 43, 3, Article 35 (May 2024), 16 pages.

Haolin Lu, Wesley Chang, Trevor Hedstrom, and Tzu-Mao Li. 2024. Real-Time Path Guiding Using Bounding Voxel Sampling. *ACM Transactions on Graphics (TOG)* 43, 4, Article 125 (July 2024), 14 pages.

P. Moreau, M. Pharr, and P. Clarberg. 2022. Dynamic many-light sampling for real-time ray tracing. In *Proceedings of the Conference on High-Performance Graphics* (Strasbourg, France) *(HPG '19)*. Eurographics Association, Goslar, DEU, 21–26.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum* 36, 4 (July 2017), 91–100.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8280–8290.

Krzysztof Narkowicz and Tiago Costa. 2025. MegaLights: Stochastic Direct Lighting in Unreal Engine 5. SIGGRAPH 2025 Course: Advances in Real-Time Rendering in Games. Course presentation.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (ToG)* 38, 6 (2019), 1–17.

Ola Olsson and Ulf Assarsson. 2011. Tiled shading. *Journal of Graphics, GPU, and Game Tools* 15, 4 (2011), 235–251.

Ola Olsson, Markus Billeter, and Ulf Assarsson. 2012. Clustered deferred and forward shading. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. Eurographics Association, Goslar, DEU, 87–96.

Jiawei Ou and Fabio Pellacini. 2011. LightSlice: matrix slice sampling for the many-lights problem. *ACM Transactions on Graphics (TOG)* 30, 6 (2011), 179.

Emil Persson. 2013. Practical clustered shading. *SIGGRAPH Course: Advances in Real-Time Rendering in Games* (2013).

Unity. 2023. Unity Technologies. https://unity.com/

Edgar Velázquez-Armendáriz, Zhao Dong, Bruce Walter, and Donald P Greenberg. 2015. Complex luminaires: Illumination and appearance rendering. *ACM Transactions on Graphics (TOG)* 34, 3 (2015), 1–15.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path guiding in production. In *ACM SIGGRAPH 2019 Courses* (Los Angeles, California) *(SIGGRAPH '19)*. Association for Computing Machinery, New York, NY, USA, Article 18, 77 pages.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. 2006. Multidimensional lightcuts. *ACM Transactions on Graphics (TOG)* 25, 3 (July 2006), 1081–1088.

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics (TOG)* 24, 3 (July 2005), 1098–1107.

Bruce Walter, Pramook Khungurn, and Kavita Bala. 2012. Bidirectional lightcuts. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Chris Wyman and Alexey Panteleev. 2022. Rearchitecting spatiotemporal resampling for production. In *Proceedings of the Conference on High-Performance Graphics (HPG '21)*. Eurographics Association, Goslar, DEU, 23–41.

Xueguang Xie, Yang Gao, Fei Hou, Aimin Hao, and Hong Qin. 2024. Dynamic ocean inverse modeling based on differentiable rendering. *Computational Visual Media* 10, 2 (2024), 279–294.

Kai Yan, Vincent Pegoraro, Marc Droske, Jiří Vorba, and Shuang Zhao. 2024. Differentiating Variance for Variance-Aware Inverse Rendering. In *SIGGRAPH Asia 2024 Conference Papers*. Association for Computing Machinery, New York, NY, USA, Article 117, 10 pages.

Cem Yuksel. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum* 34, 2 (2015), 25–32.

Cem Yuksel. 2019. Stochastic lightcuts. In *High-Performance Graphics (HPG 2019)*. The Eurographics Association, Strasbourg, France, 27–32.

Zheng Zeng, Markus Kettunen, Chris Wyman, Lifan Wu, Ravi Ramamoorthi, Ling-Qi Yan, and Daqi Lin. 2025. ReSTIR PG: Path Guiding with Spatiotemporally Resampled Paths. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*. Association for Computing Machinery, New York, NY, USA, Article 12, 11 pages.

Zhongtian Zheng, Tao Huang, Haozhe Su, Xueqi Ma, Yuefan Shen, Tongtong Wang, Yin Yang, Xifeng Gao, Zherong Pan, and Kui Wu. 2025. Auto Hair Card Extraction for Smooth Hair with Differentiable Rendering. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–13.

Junqiu Zhu, Yaoyi Bai, Zilin Xu, Steve Bako, Edgar Velázquez-Armendáriz, Lu Wang, Pradeep Sen, Milos Hasan, and Ling-Qi Yan. 2021. Neural complex luminaires: representation and rendering. *ACM Trans. Graph.* 40, 4 (2021), 57–1.