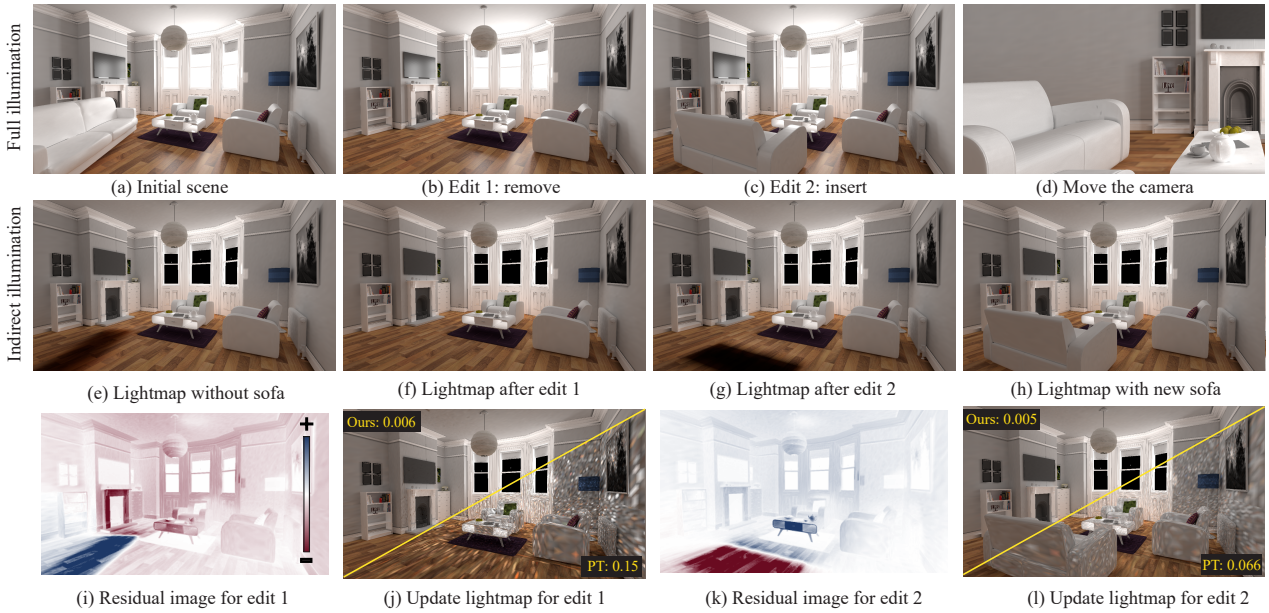
# Iterative Lightmap Updates for Scene Editing

G. Lu, C. Peters, P. Kellnhofer, E. Eisemann

Delft University of Technology




(a) Initial scene (b) Edit 1: remove (c) Edit 2: insert (d) Move the camera

(e) Lightmap without sofa (f) Lightmap after edit 1 (g) Lightmap after edit 2 (h) Lightmap with new sofa

(i) Residual image for edit 1 (j) Update lightmap for edit 1 (k) Residual image for edit 2 (l) Update lightmap for edit 2

**Figure 1:** *We update a lightmap for global illumination of a scene (a) after object removal (b) or insertion (c) while allowing for viewpoint changes (d). The scene edits affect the indirect illumination (e–h) as shown in the residual images (i,k). Our algorithm computes these residuals faster than standard path tracing, which leads to a lower RMSE in equal-time comparisons (j,l).*

**Abstract**

*Lightmaps are popular for precomputed global illumination, but require costly recomputation when the scene changes. We present the theory and an iterative algorithm to update lightmaps efficiently, when objects are inserted or removed. Our method is based on path tracing, but focuses on updates to those paths that are affected by the scene change. Using an importance sampling scheme, our solution substantially accelerates convergence. Our GPU implementation is well-suited for interactive scene editing scenarios.*

**CCS Concepts**

• ***Computing methodologies*** → ***Ray tracing;***

## 1. Introduction

Although real-time path tracing is gaining traction, precomputed global illumination (GI) is still widely used, especially on low- and mid-end hardware. For diffuse surfaces, lightmaps are the most common approach. However, lengthy precomputation results in long turnaround times, which is detrimental to artist productivity. If each scene change requires full recomputation of light maps, which may take hours in production environments, that becomes a serious problem [BB17].

We propose a method that updates lightmaps incrementally after the insertion or removal of arbitrary objects. The goal is to compute the difference between the previous lightmap and the new



DOI: 10.2312/pg.20251293

lightmap. The previous lightmap serves as a control variate. We use an operator-based formulation of light transport, with dedicated operators modeling the change in radiance due to the object. To apply these object operators, we use importance sampling of the solid angle of the object. The object operators are combined with standard path tracing and applied iteratively to obtain the updated lightmap. Unlike learning-based methods for GI [RWG*13, GMX22, ZHM*23, RHP*24], the only bias in our algorithm stems from the finite resolution of the lightmap and bounds on the maximal path length and iteration count. As such, it can be controlled by adjusting these parameters.

We implement our method on the GPU and find that it reduces the time required for an update of global illumination after a scene change by an order of magnitude when compared to GPU-accelerated, full-scene path tracing. Therefore, it is an attractive solution for interactive scene editing scenarios. The lightmap provides real-time rendering with pre-computed global illumination during camera movement. At the same time, our method facilitates quick lightmap updates after a scene edit, where noise reaches acceptable levels within seconds, rather than minutes.

In summary, this paper presents the following contributions:

- An efficient iterative method to update lightmaps after insertion or removal of objects;
- An importance-sampling strategy to accelerate this method;
- An experimental evaluation and analysis of the design choices.

In the following, we discuss related work (Sec. 2), provide a formal derivation of our algorithm (Sec. 3), experimentally validate its efficiency (Sec. 4), and discuss strengths and limitations (Sec. 5).

## 2. Related Work

Our method uses iterative updating of precomputed GI suitable for interactive scene editing. Here, we discuss prior work addressing three related GI challenges: Fast computation, storage and update.

**Interactive Global illumination** GI aims to accurately simulate the complex transport of light in a scene, as formulated by the rendering equation [Kaj86]. One common strategy to accelerate GI is to trade accuracy for speed. Voxel cone tracing [CNS*11] discretizes the scene into sparse voxel octrees and uses queries at diminishing resolutions for cone tracing, which estimates indirect incoming irradiance. Voxel-based GI [CAPP25] stores direction-dependent visibility per voxel, while keeping static and dynamic geometry separated and gathers indirect illumination per visible voxel. Dynamic diffuse global illumination [MGNM19, MMSM21] uses adaptively placed irradiance probes, which get updated using ray tracing and interpolated with heuristics that avoid light leaking. These real-time methods operate at a lower spatial resolution than lightmaps and some of them [CNS*11, CAPP25] only support one bounce of indirect light in their current implementation. In contrast, our method enables high-fidelity lightmap updates suitable for interactive scene editing.

More recently, neural networks have been used to approximate global light transport with greater flexibility. Object-oriented [ZHM*23] and light-oriented [RHP*24] light transfer methods leverage neural representations to enable modular updates for scenes with dynamic objects, lighting, and materials. However, these methods typically provide only coarse-scale or view-dependent illumination, limiting their use in scenarios requiring consistent, full-scene updates.

There has been a significant focus on visibility changes of dynamic objects in the context of interactive GI. Techniques such as *antiradiance* with hierarchical finite elements [DSDD07], implicit indirect shadow [Fra14], and residual path intregrals [XLG*24] address occlusion changes by introducing negative light-transport contributions. These methods target computations around the changes in the scene, enabling efficient updates to GI without the need to recompute the full scene. For a broader overview of global illumination techniques, we refer the reader to two surveys [Ram09, RDGK12].

**Precomputed lighting** Given the limited time budget in real-time applications, much work has addressed precomputing lighting and storing it in data structures that can be efficiently queried at runtime. Most modeling and rendering tools provide utilities to bake such lighting into texture atlases or vertex attributes [SSS*20]. Despite their efficiency, baked lighting approaches come with significant limitations [Iwa13, BB17]: Because precomputation assumes static geometry and lighting, any edits, such as object movement or insertion, have no effect on the baked illumination, requiring costly recomputation to maintain visual consistency. We address this limitation in our method by iterative updates explicitly targeting the lightmap changes due to object insertion or removal.

**Scene editing** Fast and consistent feedback is essential for interactive scene editing [SPN*16]. Re-rendering methods aim to update lighting or images efficiently after scene changes without full recomputation. Techniques using control variates and correlated sampling [RJN16] reduce variance by reusing samples. Recently, Xu et al. [XLG*24] compute radiance differences in path space, integrating only residual contributions via seven different path sampling strategies. However, such methods often operate in image space or require path mapping, limiting interactivity and generalization. Our method instead solves for lightmaps covering the entire scene, hence enabling low-cost rendering of novel viewpoints.

## 3. Our Method

We now introduce our method to update lightmaps after scene changes. First, we focus on the insertion of objects. Then, we extend the method to object removal (Sec. 3.6). In combination, these operations enable arbitrary scene changes. Our method involves light transport operators and their difference between scenes (Sec. 3.1 and 3.2). We update the scene's lightmaps by iterating the operators (Sec. 3.3 and 3.4), while the lightmap of the inserted object is handled differently (Sec. 3.5).

### 3.1. Light Transport Operator

Let $S \subset \mathbb{R}^3$ be the surface of our scene. According to the rendering equation, the outgoing radiance from a surface point $x \in S$ into

direction $\omega_o \in \mathbb{S}^2$ (where $\mathbb{S}^2 \subset \mathbb{R}^3$ denotes the unit sphere) is

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{\Omega(x)} L_i(x,\omega_i) f_r(x,\omega_i,\omega_o)\,\mathrm{d}\omega_i^{\perp},$$

where $L_i$ is the incoming radiance, $L_e$ is the emitted radiance, $f_r$ is the bidirectional reflectance distribution function (BRDF), $\mathrm{d}\omega_i^{\perp} := n(x)\cdot\omega_i\,\mathrm{d}\omega_i$ is the projected solid angle measure, $n(x) \in \mathbb{S}^2$ is the surface normal and $\Omega(x)$ is the hemisphere with $n(x)\cdot\omega_i \geq 0$.

We assume scenes without participating media. Let $\mathrm{ray}_S(x,\omega_i) \in S$ denote the closest hit of the ray $x + t\omega_i$ with the scene $S$ for $t > 0$. If a ray actually hits the scene, the incoming radiance is

$$L_i(x,\omega_i) = L_o(\mathrm{ray}_S(x,\omega_i), -\omega_i).$$

If not, we assume that $L_o$ provides the background radiance.

Combining these two equations, we arrive at a light transport operator that applies a single bounce of light transport to a radiance field $L(x,\omega_o)$ (Fig. 2(a)):

$$T_S[L](x,\omega_o) := \int_{\Omega(x)} L(\mathrm{ray}_S(x,\omega_i), -\omega_i) f_r(x,\omega_i,\omega_o)\,\mathrm{d}\omega_i^{\perp}. \quad (1)$$

The rendering equation becomes simply $L = L_e + T_S[L]$, which can be solved using a Neumann series:

$$L = \sum_{j=0}^{\infty} T_S^j[L_e] =: T_S^{\star}[L_e].$$

The operator $T_S^{\star}$ performs light transport with full global illumination. Alternatively, we can interpret this as a system of linear equations $L = (I - T_S)^{-1}[L_e]$, where $I$ denotes the identity operator.

With lightmaps for diffuse surfaces, we have a discretized representation of $L(x,\omega_o) = L(x)$. Per surface point $x \in S$, the lightmaps store outgoing radiance divided by diffuse albedo (to avoid loss of texture detail). That is the same as incoming irradiance times $\frac{1}{\pi}$. Our implementation uses a simplified triangle mesh color representation [Yuk17] with vertex colors stored in a linear buffer. That avoids the need for UV unwrapping, but if a texture atlas for lightmaps were available, it could also be used for our method. The operator $T_S$ can be thought of as a large matrix that is multiplied onto a vector of vertex colors. However, we never construct this matrix. Instead, we directly evaluate $T_S[L]$ (or similar operators) as needed, using Monte Carlo integration.

### 3.2. Difference Operators

We now consider an inserted object $O \subset \mathbb{R}^3$. We obtain the transport operator $T_{S\cup O}$ for the modified scene. To update our lightmaps, we are interested in the object difference operator $\mathcal{O} := T_{S\cup O} - T_S$. Upon inserting the object, the integral in Eq. 1 only changes for rays that intersect $O$ before they intersect $S$. Therefore, we let $\Omega_O(x) \subseteq \Omega(x)$ denote the visible solid angle of the object as observed from $x \in S \cup O$, i.e., the set of direction vectors for such rays. Then the object difference operator is $\mathcal{O} = \mathcal{O}_+ + \mathcal{O}_-$, where

$$\mathcal{O}_+[L](x,\omega_o) := +\int_{\Omega_O(x)} L(\mathrm{ray}_O(x,\omega_i), -\omega_i) f_r(x,\omega_i,\omega_o)\,\mathrm{d}\omega_i^{\perp},$$

$$\mathcal{O}_-[L](x,\omega_o) := -\int_{\Omega_O(x)} L(\mathrm{ray}_S(x,\omega_i), -\omega_i) f_r(x,\omega_i,\omega_o)\,\mathrm{d}\omega_i^{\perp}.$$

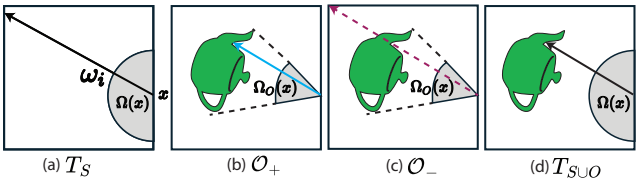



**Figure 2:** ***Illustration of single-bounce light transport operators.*** *(a) illustrates the static scene light transport operator $T_S$ at point $x$, integrating incoming radiance over the hemisphere $\Omega(x)$. (b,c) depict the object difference operators $\mathcal{O}_+$ and $\mathcal{O}_-$, capturing the radiance reflected and antiradiance occluded by the inserted object over the solid angle domain $\Omega_O(x)$, respectively. (d) shows the combined effect of all three, which yields $T_{S\cup O}$, the transport operator for the scene with the object inserted.*

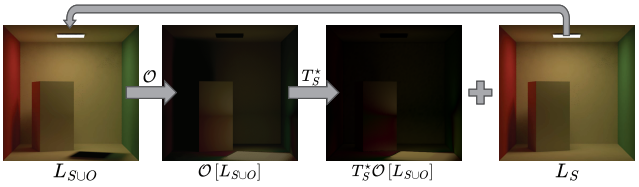



**Figure 3:** *We iteratively apply the object operator $\mathcal{O} = \mathcal{O}_- + \mathcal{O}_+$ and path tracing in the scene $S$ to the original lightmap to update it after insertion of an object $O$ (in this case, a second box).*

As shown in Fig. 2, the operator $\mathcal{O}_+$ accounts for radiance reflected by the object $O$, whereas the operator $\mathcal{O}_-$ accounts for the radiance that previously reached $x$ from the scene parts that are now occluded by the object $O$; the antiradiance [DSDD07].

To evaluate these operators, we employ importance sampling: We use a bounding sphere around the object $O$ and sample its solid angle uniformly [Wan92]. If $x$ is inside the bounding sphere, we revert to projected solid angle sampling of $\Omega(x)$. For sampled directions $\omega_i$ where $\mathrm{ray}_{S\cup O}(x,\omega_i) \notin O$, the contribution is set to zero.

### 3.3. Updating Lightmaps with Full Global Illumination

We assume that we already know the lightmap $L_S$ for the scene $S$ and seek the updated lightmap $L_{S\cup O}$ for $S \cup O$, where

$$L_S := (I - T_S)^{-1}[L_e], \qquad L_{S\cup O} := (I - T_{S\cup O})^{-1}[L_e]. \quad (2)$$

To reason about their difference, we artificially introduce factors of $I = (I - T_S)^{-1}(I - T_S)$ and $I = (I - T_{S\cup O})(I - T_{S\cup O})^{-1}$, which gives us:

$$\begin{aligned} L_{S\cup O} - L_S &= (I - T_S)^{-1}((I - T_S) - (I - T_{S\cup O}))(I - T_{S\cup O})^{-1}[L_e] \\ &= (I - T_S)^{-1}(T_{S\cup O} - T_S)(I - T_{S\cup O})^{-1}[L_e] \\ &= (I - T_S)^{-1}\mathcal{O}[L_{S\cup O}]. \end{aligned}$$

We now add $L_S$ on both sides and solve using a Neumann series:

$$L_{S\cup O} = L_S + T_S^\star \mathcal{O}\left[L_{S\cup O}\right] \tag{3}$$
$$\Leftrightarrow L_{S\cup O} = \sum_{j=0}^{\infty} \left(T_S^\star \mathcal{O}\right)^j \left[L_S\right]$$

That means we can update the lightmap by applying the object operator $\mathcal{O}$ (Fig. 4(a)) and the full light transport operator for the static scene $T_S^\star$ repeatedly. Our iterative solution is $L_{S\cup O} = \sum_{n=0}^{\infty} L_{n,S\cup O}$, where

$$L_{0,S\cup O} := L_S, \qquad L_{n+1,S\cup O} := T_S^\star \left[\mathcal{O}\left[L_{n,S\cup O}\right]\right].$$

In each step, we apply the object operator $\mathcal{O}$ to the lightmap once, followed by standard path tracing to apply $T_S^\star$. Fig. 3 illustrates this procedure. In practice, we limit the length of these path-tracing steps to $M = 3$. Overall, this still accounts for longer paths, but the length of paths between two interactions with object $O$ is limited.

Compared to from-scratch path tracing of the lightmap $L_{S\cup O}$, our method only computes the difference and utilizes dedicated importance sampling strategies to its advantage. We evaluate a control variate based on the readily available lightmap $L_S$. Additionally, the cost of ray tracing is better amortized, since we cache intermediate results for shorter paths into intermediate lightmaps $L_{n,S\cup O}$.

### 3.4. Updating Lightmaps with Indirect Illumination

Direct illumination often contains high-frequency details, especially due to sharp shadow boundaries, whereas indirect illumination is smoother. Therefore, direct illumination is more prone to undersampling artifacts. We choose to use standard unidirectional path tracing with paths generated per pixel on the screen for direct illumination, whilst using the lightmap for indirect illumination only.

We assume that the indirect lightmap for the scene $S$ is already available: $\tilde{L}_S := L_S - L_e - T_S\left[L_e\right]$. We use it to compute the indirect lightmap for the updated scene $S \cup O$:

$$\begin{aligned}
\tilde{L}_{S\cup O} :=& L_{S\cup O} - L_e - T_S\left[L_e\right] - \mathcal{O}\left[L_e\right] \\
\overset{(3)}{=}& L_S - L_e - T_S\left[L_e\right] - \mathcal{O}\left[L_e\right] + T_S^\star \mathcal{O}\left[L_{S\cup O}\right] \\
=& \tilde{L}_S - \mathcal{O}\left[L_e\right] + T_S^\star \mathcal{O}\left[L_{S\cup O}\right] \\
=& \underbrace{\tilde{L}_S - \mathcal{O}\left[L_e\right] + T_S^\star \mathcal{O}\left[L_e + T_{S\cup O}\left[L_e\right]\right]}_{=:\tilde{L}_{0,S\cup O}} + T_S^\star \mathcal{O}\left[\tilde{L}_{S\cup O}\right]
\end{aligned}$$

To solve this equation, we use the same kind of iteration as above:

$$\tilde{L}_{n+1,S\cup O} := T_S^\star \left[\mathcal{O}\left[\tilde{L}_{n,S\cup O}\right]\right], \qquad \tilde{L}_{S\cup O} = \sum_{n=0}^{\infty} \tilde{L}_{n,S\cup O}. \tag{4}$$

In practice, we limit the number of iterations according to a user-defined iteration count $N \in \mathbb{N}$.

When the object $O$ does not emit light, $\mathcal{O}_+\left[L_e\right] = 0$. Then $\mathcal{O}\left[L_e\right] = \mathcal{O}_-\left[L_e\right]$ describes direct antiradiance, i.e., shadows cast by the newly-inserted object $O$. Standard spherical cap sampling is a bad importance sampling strategy for $\mathcal{O}_-\left[L_e\right]$ in the presence of small light sources. Instead, we employ area sampling of light sources. Rays that hit $O$ and reach the light without encountering $S$ contribute according to the radiance emitted by the light source, others have zero contribution. We apply the same area sampling to $\mathcal{O}\left[T_{S\cup O}\left[L_e\right]\right]$ when computing $\tilde{L}_{0,S\cup O}$ (Fig. 4(b)).

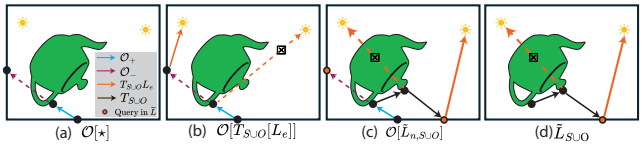



**Figure 4:** ***Overview of iterative lightmap updates.*** (a) *The general difference operator $\mathcal{O}$ combines $\mathcal{O}_+$ and $\mathcal{O}_-$ and applies to both direct (b) and indirect (c) lighting cases.* (b) *The operator $\mathcal{O}$ is applied to direct light via light sampling ($T_{S\cup O}[L_e]$).* (c) *The operator $\mathcal{O}$ is applied to the indirect lightmap $\tilde{L}_{n,S\cup O}$, using a random walk to simulate self-reflection and estimate indirect lighting on the object surface.* (d) *Computation of the object lightmap from the scene lightmap. ⊠ denotes zero-contribution rays due to occlusion.*

### 3.5. The Object Lightmap

The described method is also suitable for generating the parts of the lightmap that pertain to points on the inserted object $x \in O$. However, these parts have to be generated from scratch, which calls for more samples. Having access to the updated lightmap of the scene $S$ makes it easy to achieve high-quality results at a low cost. Per entry of the object lightmap, we initiate path tracing. Once a ray hits anything other than the object $O$ (i.e. the scene $S$ or the background), we query the indirect radiance based on the known lightmap, use light sampling to estimate direct radiance, and terminate the path (Fig. 4(d)). Unless the inserted object is extremely concave, these paths will be short.

With this approach, computation of the object lightmap is the last step. Therefore, it is unavailable when we evaluate the object reflection operator $\mathcal{O}_+$. Instead of directly querying the lightmap at the hit point on the object $O$, we initiate path tracing from the hit point until we hit a point not on the object (black rays in Fig. 4(c)). At this point, we query the lightmap and sample lights. That results in a modified object reflection operator $\mathcal{O}_+$, which models light transport between different points on the scene surface $S$ via one or more interactions with the object $O$. However, Eq. 2 still holds with this modified operator, so the derivations of our iterative algorithms remain intact.

### 3.6. Removal of Objects

Previously, we focused on inserting objects and computed $L_{S\cup O}$ from $L_S$. We now go the other way to support object removal, which in turn enables arbitrary scene changes (even if our method does benefit from localized changes). We see that all of our equations relating these two lightmaps remain valid, making the derivation of object removal for full global illumination surprisingly simple. We modify Eq. 3:

$$L_{S\cup O} = L_S + T_S^\star \mathcal{O}\left[L_{S\cup O}\right]$$
$$\Leftrightarrow L_S = L_{S\cup O} - T_S^\star \left[\mathcal{O}\left[L_{S\cup O}\right]\right]$$

Similarly, for indirect illumination

$$\tilde{L}_S = \tilde{L}_{S\cup O} + \mathcal{O}\left[L_e\right] - T_S^\star\left[\mathcal{O}\left[L_e + T_{S\cup O}\left[L_e\right] + \tilde{L}_{S\cup O}\right]\right].$$

In both of these equations, the unknown lightmap only occurs on the left-hand side, so we can explicitly solve for the lightmap; we just apply the relevant light transport operators once and do not need to update lightmaps iteratively.

## 4. Evaluation

In this section, we experimentally validate our contributions. First, we briefly discuss details of our implementation. Next, we compare our iterative lightmap update for object insertion and removal to standard path tracing of the full scene. Finally, we conduct an ablation study to validate our design.

### 4.1. Implementation

Our C++/CUDA GPU implementations of both our method and the path-tracing baseline are based on PBRT [PJH23] (specifically the path tracer described in Chapter 13.4 and the wavefront rendering in Chapter 15). In our tests, we avoid updating the scene's bounding volume hierarchy (BVH) by maintaining a separate BVH for each potential object. Our experiments were carried out on a Windows 10 system equipped with an Intel Core i9-12900KF 16-core processor running at 4 GHz with 64 GiB of RAM and an NVIDIA RTX 3090 GPU with 24 GiB of VRAM. We use the root mean square error (RMSE) as a metric to compare both the lightmaps and the rendered images. For perceptual assessment of the latter, we use *FLIP* [ANAM*20].

### 4.2. Baseline comparison

We now assess our method for insertion of a new object into a static scene with precomputed global illumination stored in our lightmap representation.

**Setup** We compare our method (*Ours*) with the pbrt-v4 path tracer (*PT*), which renders the lightmap for the modified scene from scratch. We tested three scenes curated by Bitterli [Bit16]. We subdivided their meshes to ensure sufficient vertex density which leads to lightmaps with $230k$ – $1.2M$ elements. We used *PT* with 4,096 samples per pixel (spp) to precompute static lightmaps as input to our algorithm, as well as lightmaps after the scene change as a *Reference*. We vary the spp for lightmap updates in our experiments and use the same value for both the $T_S^\star$ and $\mathcal{O}$ operators. Both operators are included in the reported timings.

**Results** In Fig. 5, we show qualitative indirect-illumination comparisons for an object-insertion scenario with equal-time assumption. Since *PT* starts from scratch, it must integrate many more paths, which is costly. Therefore, we reduce its spp values to match the computation time of our method. We observe that this leads to severe noise in the *PT* solution, especially in the *Living Room* scene due to its large and bright walls. Correlating our signed difference maps with the *Reference* shows that our method successfully computes the illumination change in the affected areas of the scene.

**Table 1:** *An equal-time comparison for editing operations: Object insertion (+, above bar) and removal (−, below bar). The percentages denote the object's size relative to the scene and the errors are the lightmap RMSE values (lower is better). The relative improvement over* PT *is shown in parentheses in the right-most column.*

| Scene | Editing | PT | Ours |
|---|---|---|---|
| Living Room | + Sofa (4%) | 0.354 | 0.010 (35x) |
| Dining Room | + Dragon (42%) | 0.117 | 0.038 (3x) |
| Staircase | + Chair (2%) | 0.332 | 0.008 (43x) |
| Living Room | − Sofa (8%) | 0.823 | 0.034 (24x) |
| Dining Room | − Carafa (2%) | 0.009 | 0.001 (9x) |

The insets and *FLIP* metric maps illustrate the strong resemblance to *Reference* renderings compared to *PT*.

We further demonstrate the benefits quantitatively in a lightmap comparison (Table 1). We observe the maximum error reduction (up to 43×, RMSE) when inserting smaller objects. Larger objects (e.g., *Dragon* in the *Dining Room* scene) intersect more lightpaths and, hence, are computationally more expensive. Even then, we still achieve a factor of 3 in error reduction. Fig. 6 highlights the changes in lightmaps by showing close-up views of regions around the object while making the object itself invisible.

Finally, in Fig. 7, we show a cost-performance analysis of our method and *PT*, achieved by sweeping the spp values. Our method is optimal across a large range of parameters and maintains near-linear convergence in terms of variance until bias due to the lightmap discretization and the limited path length $M$ and iteration count $N$ dominates. Moreover, our method requires order-of-magnitude lower computation times to achieve an RMSE equivalent to *PT* (from 3.7× in *The Dining Room* up to 179× in *The Staircase*). The relatively lower separation of our performance curve in the *Dining Room* scene once again reflects the large relative impact of the inserted *Dragon*. The original static lightmap is biased but free of noise, which explains why its RMSE is sometimes lower than that of noisy lightmaps resulting from updates with extremely low spp values.

**Object removal** We further demonstrate a lightmap update for object removal. This operation is cheaper than insertion because it does not require iterations for the lightmap update nor relighting of the object itself. We provide a qualitative comparison in Fig. 8 and a quantitative assessment in Table 1 (below the bar). We observe similar trends as for object insertion.

### 4.3. Ablation

In this section, we analyze the impact of our two hyper-parameters: The overall update iteration count $N$ and the maximum scene-to-scene path length $M$ in the operator $T_S^\star$.

**Iteration count** Our lightmap-update algorithm in Eq. 4 is iterative. Here, we analyze how the iteration count $N$ affects the rendering error. In Fig. 9, we see that in the *Living room* scene, the


**Figure 5:** ***An equal-time comparison for object insertion achieved by adjustment of spp values.*** *Only the indirect illumination is displayed.* Rows: *Three different scenes.* Columns: *First, an unmodified input scene above the* PT *solution. Second (in purple boxes), the reference solution and its difference to the input above our solution and its difference to the input. Finally, three zoom-in crops (rows) for the input,* PT, Ours *and* Reference *along with* FLIP *error maps w.r.t.* Reference. *The yellow numbers show the RMSE.*

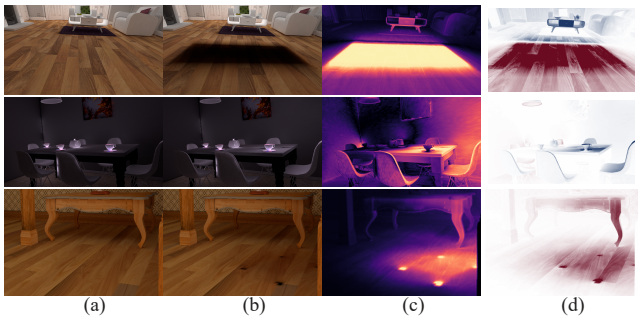



**Figure 6:** *Camera views of surfaces close to inserted objects in our three object-insertion scenarios: (a) Rendering with the lightmap prior to editing, (b) rendering after applying our lightmap update with the object made invisible, (c) FLIP difference visualization, and (d) residual image highlighting the lighting changes.*

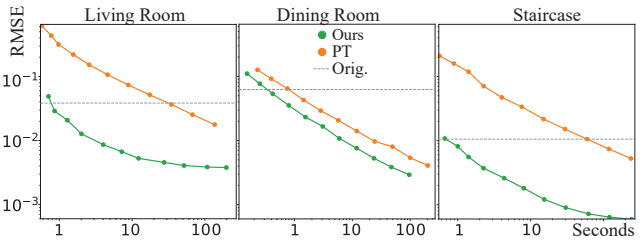



**Figure 7:** *A cost-performance comparison based on rendering RMSE measured in our three object insertion scenarios for different spp values. The horizontal line denotes the original static lightmap.*

algorithm converges after approximately only four iterations, despite the bright and large walls. This motivates our default choice of $N = 4$.

**Path length** Furthermore, we limit the scene-to-scene transport operator $T_S^\star$ to path lengths of $M$ as we observe that longer paths have negligible contribution to the solution. This is empirically analyzed in Table 2, where we vary $M$ from 0 to 5 while inserting an object into our three scenes as in Fig. 5. The experiment is conducted at a high value of 1024 spp to reduce influence of sampling noise. We observe that the RMSE of the rendered images is nearly constant for $M \geq 2$, with only a modest computation time increase. Hence, we choose $M = 3$ as a practical trade-off between accuracy

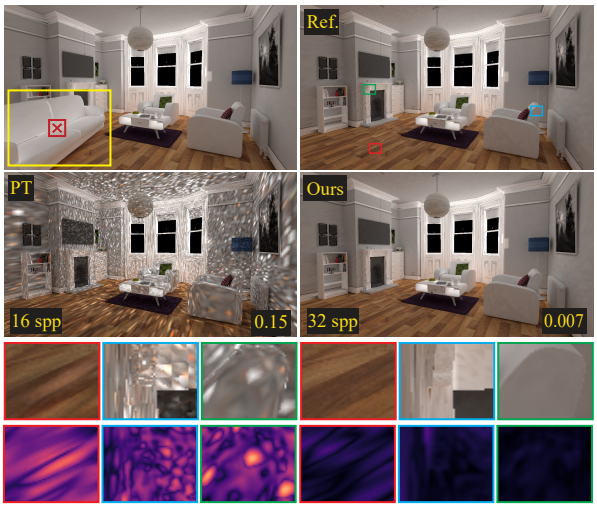



**Figure 8:** ***An equal-time comparison for removal of the sofa from the* Living Room *scene.*** *Only the indirect illumination is displayed.* First row: *The input scene and the* Reference *solution.* Second row: *The solutions from* PT *(16 spp) and from our method (32 spp) with RMSE w.r.t. the* Reference. Third and fourth row: *Zoomed-in crops and the corresponding* FLIP *error maps.*

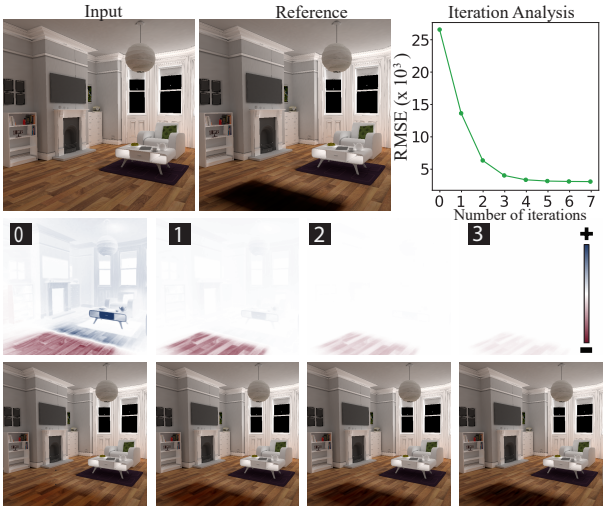



**Figure 9:** *Insertion of a* sofa *into the* Living Room. First row: *The input and the* Reference. *The* sofa *participates in the light transport but it is hidden for visualization clarity.* Second row: *Our residual image for 1–3 update iterations.* Third row: *Our solution for 1–3 update iterations.* Top right: *A cost-performance analysis for rendering RMSE number of iterations N.*

and performance, while longer paths can still be accounted for as a compounding effect of our iterative algorithm.

**Table 2:** *Analysis of path length M: Rendering RMSE ×1000 (top, black) and computation time in seconds (bottom, gray) for different path length limits* $M \in \{0,1,2,3,4,5\}$ *measured during an object insertion for a high setting of 1024 spp.*

| | *M* | | | | | |
|---|---|---|---|---|---|---|
| **Scene** | 0 | 1 | 2 | 3 | 4 | 5 |
| Living Room | 5.36 | 3.97 | 3.77 | 3.75 | 3.79 | 3.80 |
| | 134 | 159 | 177 | 187 | 192 | 199 |
| Dining Room | 5.47 | 3.36 | 2.98 | 2.94 | 2.93 | 2.95 |
| | 87 | 90 | 95 | 96 | 97 | 100 |
| Staircase | 1.38 | 0.87 | 0.59 | 0.56 | 0.56 | 0.55 |
| | 208 | 232 | 246 | 259 | 263 | 270 |

## 5. Limitations and Future Work

Our experiments demonstrate a superior equal-time rendering quality of our lightmaps updated for small scene edits involving one object at a time. This trivially extends to multiple objects by a sequential application. Despite the linear cost-scaling, our measured speed-up supports feasibility of such an approach and further improvements are possible by treating clustered objects as a unit. Moreover, a dynamic scene can similarly be modeled as a temporal sequence, although explicit consideration of temporal coherency could lead to an additional speed-up.

We inherit standard lightmap limitations. Our separation of indirect illumination allows for view-dependent direct illumination effects, but we are limited to Lambertian diffuse surfaces for the indirect component. This could be remedied with higher-dimensional representations such as spherical harmonics [O'D18], leveraging the smoothness of indirect illumination, or by incorporating specular path displacement through glossy probe reprojection [RLP*20], inspired by similar principles from ray tracing. Second, the lightmap's discretization introduces bias. However, this can be controlled by adjusting the resolution as our update rule itself is unbiased in the hyper-parameter limit for $M \to \infty$, $N \to \infty$.

## 6. Conclusion

In this work, we introduced an efficient approach for scene editing that allows artists to perform object insertions and removals with reduced computational overhead. By modifying a precomputed lightmap through an iterative update method, our technique provides order-of-magnitude speedups over traditional path tracing, while maintaining a high accuracy, as demonstrated by the *RMSE* and *FLIP* metrics. Finally, we show that the tuning of our hyper-parameters leads to a favorable cost-performance tradeoff enabling interactive editing and navigation in 3D design.

## Acknowledgments

The work is part of VR-Renovate (project number 403.19.222), financed by the Dutch Research Council (NWO). The Living room, Dining room, Staircase, and Dragon in Figs. 1, 5, 8 and 9 are obtained from Blend Swap and created by Jay-Artist, Wig42, and Delatronic. The Cornell box in Fig. 3 is created by Benedikt Bitterli.

## References


[ANAM*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: Flip: A difference evaluator for alternating images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 3*, 2 (2020), 1–23. doi:10.1145/3406183. 5

[BB17] BARRÉ-BRISEBOIS C.: Open problems in real-time rendering: A certain slant of light: Past, present and future challenges of global illumination in games. In *ACM SIGGRAPH 2017 Courses* (2017), Tatarchuk N., Lefohn A., (Eds.). doi:10.1145/3084873.3127940. 1, 2

[Bit16] BITTERLI B.: Rendering resources, 2016. URL: https://benedikt-bitterli.me/resources/. 5

[CAPP25] COSIN AYERBE A., POULIN P., PATOW G.: Dynamic voxel-based global illumination. *Computer Graphics Forum 44*, 1 (2025). doi:10.1111/cgf.15262. 2

[CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum 30*, 7 (2011). doi:10.1111/j.1467-8659.2011.02063.x. 2

[DSDD07] DACHSBACHER C., STAMMINGER M., DRETTAKIS G., DURAND F.: Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics (TOG) 26*, 3 (2007). doi:10.1145/1275808.1276453. 2, 3

[Fra14] FRANKE T. A.: Delta voxel cone tracing. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (2014), IEEE, pp. 39–44. doi:10.1109/ISMAR.2014.6948407. 2

[GMX22] GAO D., MU H., XU K.: Neural global illumination: interactive indirect illumination prediction under dynamic area lights. *IEEE Transactions on Visualization and Computer Graphics* (2022). doi:10.1109/TVCG.2022.3209963. 2

[Iwa13] IWANICKI M.: Lighting technology of the last of us. In *ACM SIGGRAPH 2013 Talks*. 2013. doi:10.1145/2504459.2504484. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (1986). doi:10.1145/15922.15902. 2

[MGNM19] MAJERCIK Z., GUERTIN J.-P., NOWROUZEZAHRAI D., MCGUIRE M.: Dynamic diffuse global illumination with ray-traced irradiance fields. *Journal of Computer Graphics Techniques (JCGT) 8*, 2 (2019). URL: https://jcgt.org/published/0008/02/01/. 2

[MMSM21] MAJERCIK Z., MARRS A., SPJUT J., MCGUIRE M.: Scaling probe-based real-time dynamic global illumination for production. *Journal of Computer Graphics Techniques (JCGT) 10*, 2 (2021). URL: http://jcgt.org/published/0010/02/01/. 2

[O'D18] O'DONNELL Y.: Precomputed global illumination in Frostbite. In *Game Developers Conference* (2018), vol. 1. 7

[PJH23] PHARR M., JAKOB W., HUMPHREYS G.: *Physically based rendering: From theory to implementation*. MIT Press, 2023. 5

[Ram09] RAMAMOORTHI R.: Precomputation-based rendering. *Foundations and Trends in Computer Graphics and Vision 3*, 4 (2009). doi:10.1561/0600000021. 2

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Computer Graphics Forum 31*, 1 (2012). doi:10.1111/j.1467-8659.2012.02093.x. 2

[RHP*24] REN H., HUO Y., PENG Y., SHENG H., HUANG H., XUE W., LAN J., WANG R., BAO H.: Lightformer: Light-oriented global neural rendering in dynamic scene. *ACM Transactions on Graphics 43*, 4 (2024). doi:10.1145/3658229. 2

[RJN16] ROUSSELLE F., JAROSZ W., NOVÁK J.: Image-space control variates for rendering. *ACM Transactions on Graphics (TOG) 35*, 6 (2016). doi:10.1145/2980179.2982443. 2

[RLP*20] RODRIGUEZ S., LEIMKÜHLER T., PRAKASH S., WYMAN C., SHIRLEY P., DRETTAKIS G.: Glossy probe reprojection for interactive global illumination. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–16. doi:10.1145/3414685.3417823. 7

[RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Transactions on Graphics 32*, 4 (7 2013). doi:10.1145/2461912.2462009. 2

[SPN*16] SCHMIDT T.-W., PELLACINI F., NOWROUZEZAHRAI D., JAROSZ W., DACHSBACHER C.: State of the art in artistic editing of appearance, lighting and material. *Computer Graphics Forum 35*, 1 (2016). doi:10.1111/cgf.12721. 2

[SSS*20] SEYB D., SLOAN P.-P., SILVENNOINEN A., IWANICKI M., JAROSZ W.: The design and evolution of the UberBake light baking system. *ACM Transactions on Graphics (TOG) 39*, 4 (2020). doi:10.1145/3386569.3392394. 2

[Wan92] WANG C.: Physically correct direct lighting for distribution ray tracing. In *Graphics Gems III*. Academic Press Professional, 1992, pp. 307–313. 3

[XLG*24] XU B., LI T.-M., GEORGIEV I., HEDSTROM T., RAMAMOORTHI R.: Residual path integrals for re-rendering. *Computer Graphics Forum 43*, 4 (2024). doi:10.1111/cgf.15152. 2

[Yuk17] YUKSEL C.: Mesh color textures. In *Proceedings of High Performance Graphics*. ACM, 2017. doi:10.1145/3105762.3105780. 3

[ZHM*23] ZHENG C., HUO Y., MO S., ZHONG Z., WU Z., HUA W., WANG R., BAO H.: NeLT: Object-oriented neural light transfer. *ACM Transactions on Graphics 42*, 5 (2023). doi:10.1145/3596491. 2