Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

Volume 34 (2015), Number 4

Consistent Scene Editing by Progressive Difference Images

Tobias Günther[1] and Thorsten Grosch[2]

[1] Visual Computing Group, University of Magdeburg
[2] Computational Visualistics Group, University of Magdeburg

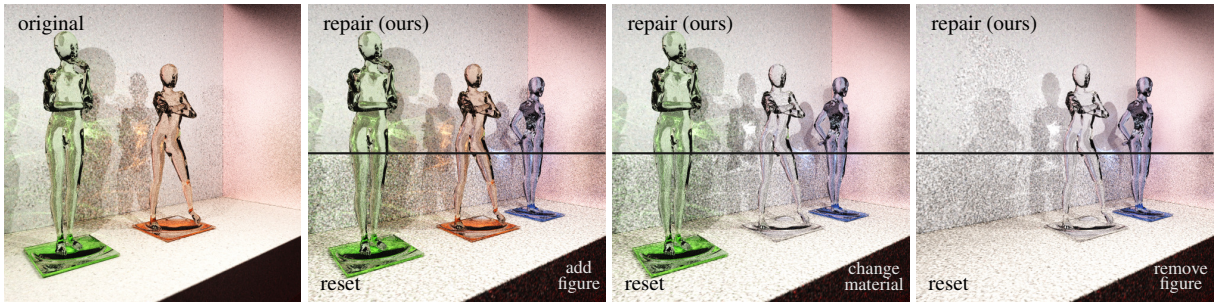


Figure 1: *Interactive scene editing with SPPM: Object adding, material change and object removal. Our approach (top row) preserves details and introduces noise only in affected regions. In contrast, a simple restart (bottom row) is noisy everywhere.*

Abstract

Even though much research was dedicated to the acceleration of consistent, progressive light transport simulations, the computation of fully converged images is still very time-consuming. This is problematic, as for the practical use in production pipelines, the rapid editing of lighting effects is important. While previous approaches restart the simulation with every scene manipulation, we make use of the coherence between frames before and after a modification in order to accelerate convergence of the context that remained similar. This is especially beneficial if a scene is edited that has already been converging for a long time, because much of the previous result can be reused, e.g., sharp caustics cast or received by the unedited scene parts. In its essence, our method performs the scene modification stochastically by predicting and accounting for the difference image. In addition, we employ two heuristics to handle cases in which stochastic removal is likely to lead to strong noise. Typical scene interactions can be broken down into object adding and removal, material substitution, camera movement and light editing, which we all examine in a number of test scenes both qualitatively and quantitatively. As we focus on caustics, we chose stochastic progressive photon mapping as the underlying light transport algorithm. Further, we show preliminary results of bidirectional path tracing and vertex connection and merging.

1. Introduction

Physically-based lighting and consistent global illumination gained increasing importance over the past years. Especially the modeling of light became an important step in nowadays visual effects and digital animation production pipelines. Therein, lighting is not only essential to set the mood of a scene, but also to guide attention and achieve depth and realism. In existing light manipulation methods, such as path-space manipulation [SNM*13], the light transport simulation starts anew every time an element in the scene changes. While much work pushed consistent, progressive light transport for faster convergence [GKDS12, HPJ12, KD13], individual effects such as sharp caustics may still take a long time to converge. This is problematic during scene editing, since fast feedback is essential to an artist [SPN*14]. Previously, light editing approaches started the simulation anew,

whenever a scene element changed. This is improvable, as frequently there is much coherence to exploit between the frames before and after a modification. For this, there are in general two orthogonal problems to consider: (a) fast radiance update of the object that is edited, and (b) fast radiance update of the unedited context, i.e., changes in indirect illumination. Problem (a) heavily depends on the object's BRDF and might require additional information on the incident and exitant light distribution. As we will show, problem (b) can be approached without requiring deep changes in the underlying light transport algorithm. In fact, our goal is to provide a drop-in solution for an existing light transport simulation that utilizes frame coherence to improve the convergence in the unedited scene parts. Thus, during object editing, our method allows to sooner see how the edited object integrates into the whole scene, as demonstrated in Fig. 1. Another use case is the editing of simulations that have been running for a while, since undesired details might be noticed late due to slowly converging effects such as caustics. In this case, a correction should not set back the entire simulation.

Our method adds a coarse progressively computed difference image to the previous frame in order to estimate the radiance in the slightly varied new frame. Doing so allows to preserve details in regions that have not been affected by the scene modification. However, such stochastic correction is subject to noise artifacts in regions where the modification had large impact. Even though progressive light transport simulations are guaranteed to converge in the limit anyway, they will in practice never recover from a strong bias. For this reason, we additionally apply two heuristics that reset a pixel if either the modified object is directly seen by the view ray, i.e., we reset in case of problem (a), or if the impact on the radiance is too large, i.e., strong indirect illumination. Our method supports the adding and removal of objects and lights, material substitutions and camera movement.

The correction procedure is easily integrated into existing light manipulation tools, because it only needs a black-box that progressively delivers radiance estimates. For this reason, our method is orthogonal to all the advances on acceleration [GKDS12, HPJ12, KD13] and specialized lighting effects, such as volumetric lighting [KGH*14]. The method is most beneficial for effects that take long to converge, e.g., caustics. Thus, we implemented stochastic progressive photon mapping [HJ09], which excels in caustics, on the GPU as in [HJ10] using OptiX [PBD*10] and used the progressive photon statistic updates derived by Knaus and Zwicker [KZ11]. Later in the paper, we also show preliminary results with bi-directional path tracing [LW93] and vertex connection and merging [GKDS12].

2. Related Work

Much work has been done in the fields of light transport and scene editing to tailor simulations and content creation to the requirements arising in modern production environments.

Consistent light transport simulations strive for an exact radiance computation. Famous examples are the unbiased bi-directional path tracing [LW93] and Metropolis light transport [VG97]. The biased photon mapping by Jensen [Jen96] was made consistent in a progressive extension of Hachisuka et al. [HOJ08], leading up to stochastic progressive photon mapping (SPPM) [HJ09]. SPPM alternately traces photons and eye rays, progressively reduces the query radius of the photon density estimation and maintains pixel statistics that eventually converge to the desired radiance. Using stochastic spatial hashing, it was brought later to the GPU [HJ10]. Knaus and Zwicker [KZ11] suggested a probabilistic reformulation of SPPM that led to the insight that the query radius can be decreased independent of the photon density.

More recent work pushed light transport simulations for faster convergence and generality. A thorough overview of GPU-based light transport was compiled by Davidovič et al. [DKHS14]. An improved sampling of the path space was achieved by vertex connection and merging [GKDS12] and a related path space extension by Hachisuka et al. [HPJ12]. In recent work, SPPM was brought out-of-core [GG14] and convergence rates were improved by adaptive progressive photon mapping of Kaplanyan and Dachsbacher [KD13]. Jarosz et al. [JZJ08, JNSJ11] accelerated volumetric radiance estimates, which were further generalized by unifying points, beams and paths in volumetric light transport [KGH*14]. Light transport of pre-defined animations was accelerated by Weiss and Grosch [WG12] for SPPM.

In interactive light editing, fast feedback is essential to the artist. Several fast approximations exist in animation rendering [MTAS01] that, however, exhibit a small perceptual error. A correct solution for interactive photon mapping with moving objects can be obtained using selective photon tracing [DBMS02]. Here, all photon paths are sorted by using Halton random numbers, which allows to quickly detect the photon paths that were affected by object motion. Correction is then performed by generating positive and negative photon paths that add/remove indirect radiance and shadows. Further early work on interactive global illumination in dynamic scenes includes the render cache system of Walter et al. [WDP99, WDG02], which caches shading points temporally. A separation and object-space caching of shading values, their interpolation and sampling were addressed in Tole et al. [TPWG02]. Another angle at the image synthesis from expensive sampled-based rendering was taken by Bala et al. [BWG03] in their edge-and-point renderer. All these aforementioned methods have great potential to be an accelerator of our difference image computation.

Recently, Schmidt et al. [SPN*14] compiled a comprehensive overview of artistic editing of appearance, lighting and material. One way to deliver fast updates is to relight a static scene. Several hardware-accelerated relighting engines have been successfully deployed, including Lpics by Pellacini et al. [PVL*05], the linear wavelet-compressed direct-

to-indirect transfer by Hašan et al. [HPB06] and the light-speed system by Ragan-Kelley et al. [RKKS*07]. Numerous techniques allow a (possibly) non-physical manipulation of light, which can be controlled directly as in the Bendy-Lights method of Kerr et al. [KPD10] or indirectly as in Nowrouzezahrai et al. [NJS*11] for artistic volumetric lighting. Schmidt et al. [SNM*13] combined both direct and indirect interactions, allowing manipulation of light paths and retargeting of paths according to edits in the scene. In the field of material editing, recent strides have been made by Hašan and Ramamoorthi [HR13], who built a material designer to interactively set the local (single-scattering) albedo coefficients and by Klehm et al. [KISE14], who enabled stylized property and lighting manipulation of static volumes.

Much research was dedicated to improving design methods, such as light, material and appearance editing, and on computing fast approximations for previews. In this paper, we propose a method that is compatible to these design methods, as it operates on a very abstract level, i.e., it operates on a progressive drop-in lighting simulation. In contrast to previous work, we specifically target fast updates in *consistent* light transport simulations. Moreover, our method allows to reuse already converged radiance results.

3. Scene Editing with Progressive Difference Images

Given is a scene $\mathcal{A}$ and a modified version of it $\mathcal{B}$, which was obtained by a scene editing operation, e.g., by modifying an object, substituting a material, changing a light or moving the camera. Let further be given a function $\widehat{L}_i^{\mathcal{S}}$ that delivers a radiance estimate for a scene $\mathcal{S} \in \{\mathcal{A}, \mathcal{B}\}$ in a given iteration i. Passing iteration i to this function allows to deterministically define the rays to generate. The progressive series of radiance estimates shall be converging to the correct radiance, i.e., we have a consistent progressive light transport (PLT) method at hand, e.g., SPPM [HJ09], BPT [LW93] or VCM [GKDS12]. While the theoretical background is general, we focus on SPPM in our experiments.

We distinguish between the pre-editing radiance L_{pre} (of scene $\mathcal{A}$), and the post-editing radiance L_{post} (of scene $\mathcal{B}$):

$$L_{pre} = \frac{1}{I}\sum_{i=1}^{I} \widehat{L}_i^{\mathcal{A}} \;, \quad L_{post} = \frac{1}{K}\sum_{k=1}^{K} \widehat{L}_{I+k}^{\mathcal{B}} \tag{1}$$

computed from I or K iterations of PLT, respectively. Once a scene modification occured, the trivial approach is to restart the entire lighting simulation, thereby rejecting all previous radiance estimates L_{pre}. This wasteful rejection was common in related literature, which focused on a different problem: finding intuitive interaction methods for light editing [SNM*13]. However, the scene modification might have had only small impact on most parts of the scene, causing an inherent, so far unexploited coherence between the frames. Thus instead, we want to adapt L_{pre} to the new scene, for which we approximate the radiance difference L_{diff}. For this, we alternately render the scenes $\mathcal{A}$ and $\mathcal{B}$, resulting in two estimates L_{old} and L_{new} from which we compute the progressively improving difference estimate L_{diff} by an interactively chosen number of correction iterations J:

$$L_{diff} = \frac{1}{J}\sum_{j=1}^{J}\left(\widehat{L}_j^{\mathcal{B}} - \widehat{L}_j^{\mathcal{A}}\right) = L_{new} - L_{old} \tag{2}$$

$$L_{new} = \frac{1}{J}\sum_{j=1}^{J}\widehat{L}_j^{\mathcal{B}} \;, \quad L_{old} = \frac{1}{J}\sum_{j=1}^{J}\widehat{L}_j^{\mathcal{A}} \tag{3}$$

A small number of correction iterations is desirable during interactive editing, which, however, limits the quality of the radiance difference approximation. We found $J = 50$ to be a good compromise, which we used in our tests. The choice of J is discussed later in Section 4.3. If we use the same random seeds in both scenes for ray generation, the images are precisely the same except for paths that were affected by the scene modification. Note that then, the noise in L_{diff} is only in the order of the radiance difference. Thus, the error of calculating an estimate of the post-editing radiance by $L_{pre} + L_{diff}$ is low in regions in which the scene modification had small impact. We use a heuristic to determine per pixel whether the pre-editing radiance can be corrected this way, i.e., if changes are small. The result of the heuristic is a binary mask M that switches between the options *correction* and *reset*. The heuristic is explained later in Section 3.2. The resulting radiance L_{repair} after I pre-editing iterations, J correction iterations and K post-editing iterations is:

$$L_{repair} = (1-M)\cdot\Big(\underbrace{\frac{I}{I+K}L_{pre}}_{\text{PLT in }\mathcal{A}} + \underbrace{\frac{I}{I+K}L_{diff}}_{\text{difference image}} + \underbrace{\frac{K}{I+K}L_{post}}_{\text{PLT in }\mathcal{B}}\Big) + M\cdot\frac{1}{J+K}\sum_{l=1}^{J+K}\widehat{L}_l^{\mathcal{B}} \tag{4}$$

The first line contains the *correction* case ($M = 0$) and consists of three terms: the pre-editing radiance (I iterations of PLT in $\mathcal{A}$), a difference image (J iterations of PLT), and the post-editing radiance (K iterations of PLT in $\mathcal{B}$). The sum of L_{pre} and L_{diff} approximates the radiance in the new scene after I iterations, which is weighted accordingly with L_{post} (K iterations). The second line contains the *reset* case ($M = 1$), i.e., standard PLT in scene $\mathcal{B}$ starting after the editing operation. Note that for $I = J$, the difference image completely removes all radiance from scene $\mathcal{A}$, since the same rays were generated. For $J < I$, the difference image is approximated. We will demonstrate that a coarse approximation is sufficient in regions where scene modifications had low impact.

3.1. Correction Iteration

While the progressive correction is in progress, we still have $K = 0$, thus Eq. (4) simplifies to:

$$L_{repair} = \left(L_{pre} + L_{diff}\right)\cdot(1-M) + L_{new}\cdot M \tag{5}$$

Fig. 2 (left) lists the correction procedure that we apply, whenever a scene modification occurs. During progressive

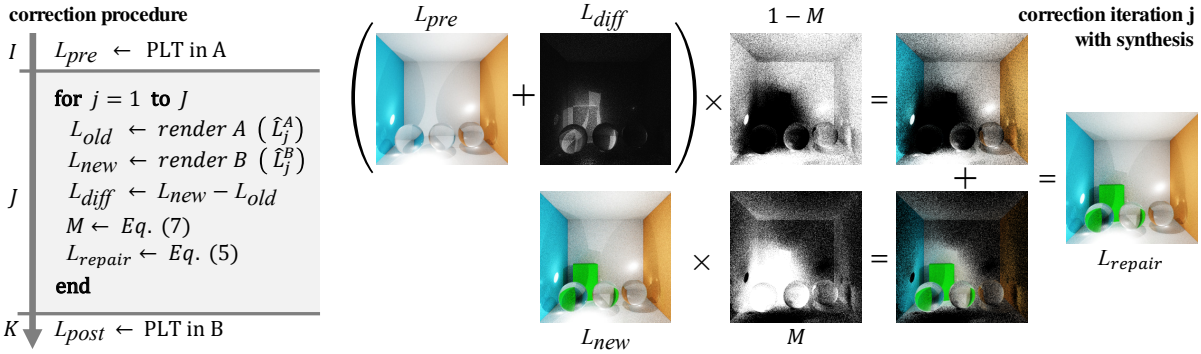


Figure 2: *Left: Pseudo code of correction prodedure. Right: Preview image synthesis in each correction iteration, here, for insertion of a green box. The top row shows the correction case and the bottom row depicts the reset case.*

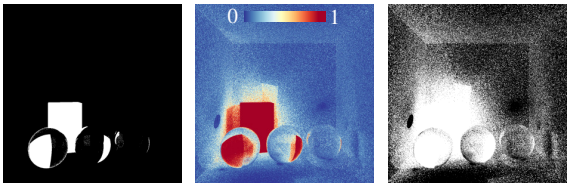
correction, each of the $j = 1 \ldots J$ correction iterations first renders scenes $\mathcal{A}$ and $\mathcal{B}$ to improve their estimates L_{old} and L_{new}. Based on the radiance difference L_{diff} the reset heuristic is updated (Section 3.2). Afterwards, radiance L_{repair} of the new scene is estimated, based on Eq. (5), as illustrated in Fig. 2 (right). This intermediate result L_{repair} is displayed for interactive feedback. After a total of J correction iterations, the synthesized image L_{repair} overwrites the previous result L_{pre}, and standard PLT carries on in the post-edited scene. Before J correction iterations are reached, the correction can be cancelled and we can simply fall back to L_{pre} at any time.

3.2. Heuristics to Reset Pixels

Progressive light transport methods are known to converge even if we just continue with the old pixel statistics in the new scene, because there was only a finite number of iterations in the old scene. (Consider the limit of Eq. (4): $\lim_{K\to\infty} L_{repair} = L_{post}$ for $M = 0$.) In practice, however, it cannot recover from the bias fast enough, as shown in Section 4.3. Thus, we estimate and account for the difference image between the frames. However, in regions where the modification has large impact on the radiance, e.g., in an added caustic, our form of stochastic editing is still costly, due to the high number of correction iterations required to sufficiently reduce the noise in L_{diff}. In the following, we describe two heuristics to identify pixels for which we refrain from a costly difference image computation, and instead reset the pixel. Since pixels reset individually, we store *statistical values* for each of them, comprising iteration number, accumulated radiance and for SPPM the query radius.

Reset on Sight

If an eye ray hits the edited object, we conservatively reset the statistics of the entire pixel, as the radiance is likely to be very different and should thus be rejected. Note that this might happen after a number of indirections, for instance if the object is seen through glass. An exemplary illustration of the affected pixels is shown in Figure 3(a). The reset on sight mask M_{sight} can be computed on-the-fly, because the object



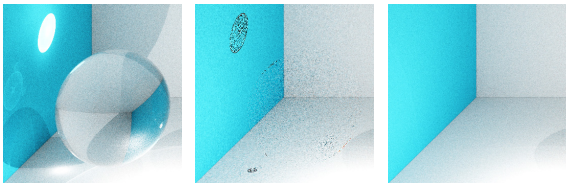
(a) Reset on sight (b) Error ratio ε (c) Threshold $\varepsilon > \tau$

Figure 3: *In this image the reset masks are visualized, showing where the heuristics reset the pixel statistics (white).*

is either seen in L_{old} (object was removed), L_{new} (object was added) or in both (material has changed). If a light or the camera have changed this mask remains empty. Thus, during the computation of L_{old} and L_{new}, we mark a pixel if its view ray hits the objects to edit. The number of correction iterations J affects the chance to see the edited object through glossy reflections, as demonstrated later in Section 4.3.

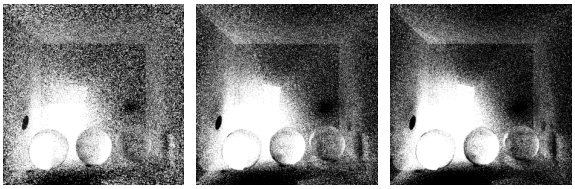
Error Threshold

As motivated earlier, a stochastic editing is subject to noise artifacts. For example, consider a glass object casting a bright caustic onto a diffuse surface, as in Fig. 4(a). If the glass object is removed, the caustic should vanish as well. When stochastically adding or removing radiance from a scene, the radiance remains as noise, as shown in Fig. 4(b). PLT is known to reduce this bias over time. However, the



(a) before removal (b) after, w/o heuristic (c) after, w/ heuristic

Figure 4: *This figure demonstrates the effect of the reset heuristics. (a) shows a caustic, which is removed in (b) without reset heuristics and in (c) with reset heuristics. The caustic clearly remains as visible noise in (b).*



(a) $J = 10$ (b) $J = 50$ (c) $J = 100$

Figure 5: *Increasing the number of correction iterations J reduces noise in error ratio ε, thus improves the reset mask.*

magnitude of the noise is in the order of the radiance to add or remove. Thus, we use a heuristic to determine the pixels that probably take too long to recover from the stochastic editing noise. We define the error ratio ε between the radiance difference L_{diff} and an estimate of either the previous scene's radiance L_{old} or the new scene's radiance L_{new}, all computed using a small number of J correction iterations:

$$\varepsilon = \begin{cases} \frac{L_{diff}}{L_{diff}+L_{old}} & \text{if } L_{diff} \geq 0 \\ \frac{L_{diff}}{L_{diff}-L_{new}} & \text{else} \end{cases} \tag{6}$$

For no difference $L_{diff} = 0$, this error ratio becomes $\varepsilon = 0$. The error ratio approaches $\varepsilon = 1$ if the radiance difference is high compared to the radiance of the previous scene L_{old} (when radiance was added) or compared to the radiance of the new scene L_{new} (when radiance was removed). The error ratio ε is visualized in Fig. 3(b).

If the error ratio ε is above a certain threshold τ, we reset the pixel statistics. We call this threshold *error threshold* τ and recommend a default value of $\tau = 0.1$. The resulting mask $M_{error} = \varepsilon > \tau$ is shown in Fig. 3(c) and its effect on the removal of editing noise is demonstrated in Fig. 4(c). In Section 4.3, different choices for τ are discussed in more detail. The combined reset mask is obtained by a logical or:

$$M = M_{sight} \vee M_{error} \tag{7}$$

While the J correction iterations are carried out, the noise in L_{old}, L_{new}, and consequentially in error ratio ε reduces. Due to possibly strong noise in error ratio ε in the first few correction iterations, more pixels would reset than necessary, as demonstrated in Fig. 5. For this reason, we synthesize radiance L_{repair} according to Eq. (5) only as a *preview* and do not actually reject the former statistics as soon as a pixel is marked for reset. Only the very last correction iteration merges L_{repair} into the final radiance statistics. In addition, this allows to cancel corrections at any time, since pixel statistics are not altered before the final correction iteration.

3.3. Synthesis during Interactive Scene Editing

So far, we explained the necessary steps to carry out an operation such as changing a material, or removing or adding an object. Another common interaction is to move objects. This can be achieved by concatenating two correction procedures. First, the object is removed, which requires to compute the reset mask and the respective difference image. The synthesized radiance L_{repair} serves as L_{pre} in the subsequent adding of the object at the new position. Both procedures take about twice the time of a usual iteration, which accumulates to four times. The remove procedure begins when the object starts moving and carries on unless the modification is canceled. The add procedure, on the other hand, restarts whenever the position of the object changes. All former changes due to adding at an intermediate position are thereby undone. This means, if an object is dragged across the entire viewport the pixels it swept over remain unaffected, as shown in Fig. 6.

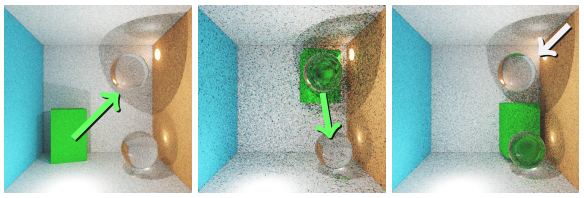


Figure 6: *In an interactive session, a green box moves along the green arrows in two steps. The adding of the box in the second image (first step) is canceled when the box moves further down (second step). Thus, the glass sphere on the top right can use the previous radiance and does not reset.*

3.4. Reprojection during Camera Movement

Reusing pixel statistics of the previous scene requires a correspondence between the pixels of the old scene's frame and the new scene's frame. During camera movement, we establish this correspondence by reverse reprojection [NSL*07]. For each pixel of the new frame, we trace a ray through the pixel center into the new scene and project its first intersection into the old camera's view. If the intersection point is visible by the old camera, i.e., it is not occluded and inside the old viewport, the corresponding pixel is found. Otherwise, we reset the pixel. For glass or glossy BRDFs the reprojected radiance might be inaccurate as radiance was observed from a different direction. For those, error ratio ε becomes high and our heuristic decides whether to reset. Note that the pair of corresponding pixels averaged radiance over different scene parts, thus error ratio ε is typically larger than for static cameras. An example follows later in Fig. 12.

4. Results and Evaluation

In this section, we study the convergence behavior, discuss parameters and test other light transport algorithms. All ground truth images are computed by $100k$ SPPM iterations.

4.1. Convergence

In the following, we conduct a qualitative and quantitative evaluation. Both convergence rate (quantitative) and visual result (qualitative) are important to study, because an early,

visually pleasing result might have a bad long-term convergence, whereas a result with a good convergence behavior might have noticeable artifacts. Concerning the convergence rate, we are interested in two factors: the improvement by our method right after the editing operation and the long-term convergence. This means, in a convergence plot, the curve of our method should stay well below the curve of the trivial reset approach and should ideally never cross it. In the following, all images are shown for $K = 100$ PLT iterations.

Using the Sponza scene in Fig. 7, we study the convergence in a large scene with glossy BRDFs. Especially the background benefits from our method, resulting in less noise (see arrow). A close-up on the glossy floor is shown later.

In Fig. 8, two rubber ducks and a light are added into a box filled with water. The actions have a different impact on noise and convergence rate, as visible in the RMSE plot.

The smaller the impact of a scene modification, the more coherence can be exploited. To study the worst case, i.e., a large modification, we remove a glass dragon in Fig. 9, which is a difficult task due to the high number of caustics. For comparison, we first add a small sphere to make the difference in the convergence plot apparent.

In Fig. 10, we show a scenario in which small objects are added and removed from a scene. For modifications that strongly affect the radiance in the entire scene, such as the adding of a light in the second column, there is not much to gain. This is apparent in the convergence plot by the short

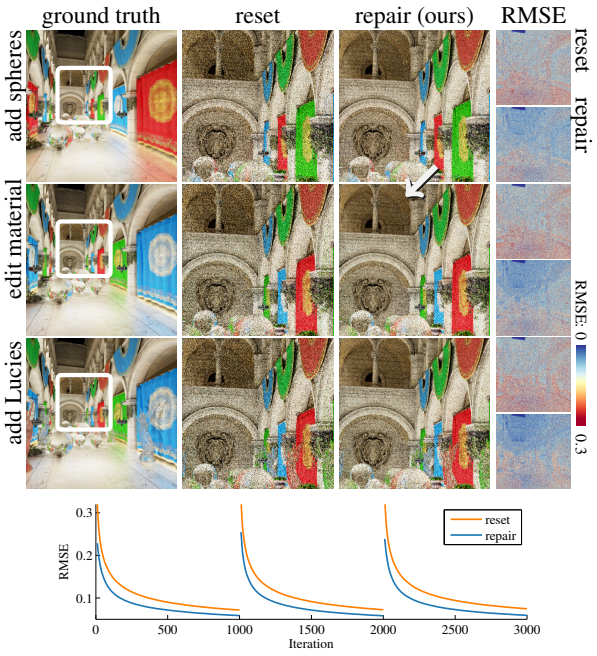


Figure 7: *In the Sponza scene spheres are added, curtains are recolored and Lucies are added. Our method reduces noise on both the glossy floor and in the diffuse background.*

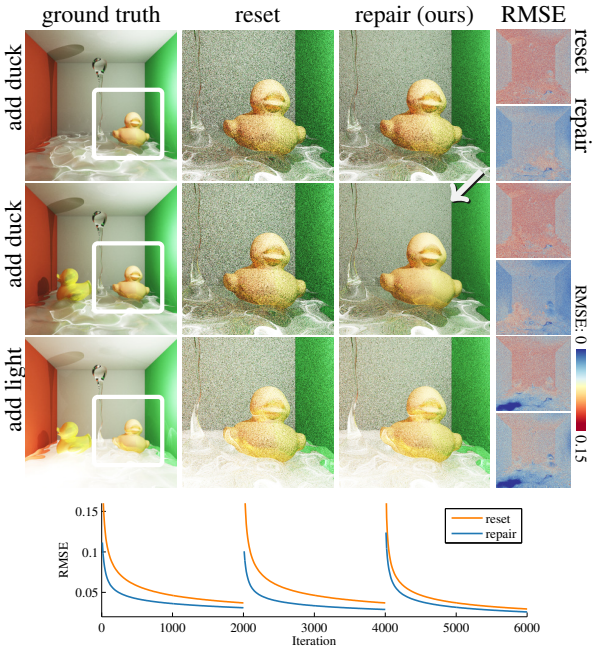


Figure 8: *In this scene, we add two rubber ducks and turn on a light. It can again be seen that less improvement is gained when the modification greatly affects the entire scene.*

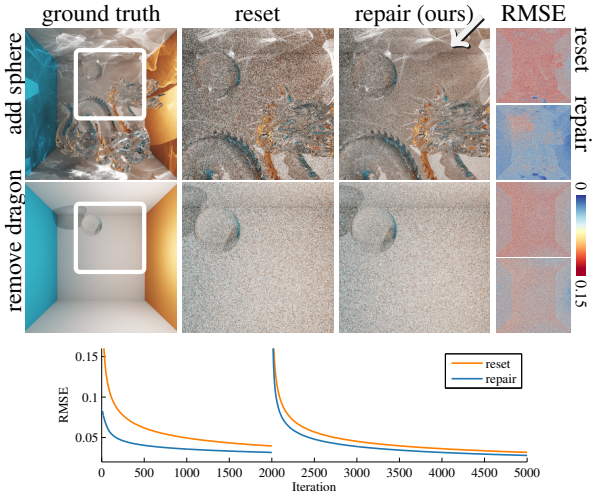


Figure 9: *Worst case study: change that affects the radiance in the entire scene. We first add a sphere (small modification) and afterwards remove the dragon. In the latter, our method delivers only minor improvement over the reset approach.*

distance between the curves. The faster the blue curve (our method) descends, the better the visual improvement.

In Fig. 11, we add and remove a number of glass objects. Here, our method excels in preserving converged caustics and diffuse background regions, as indicated by arrows and apparent in the RMSE visualizations.

Finally, Fig. 12 shows a camera movement. As corre-

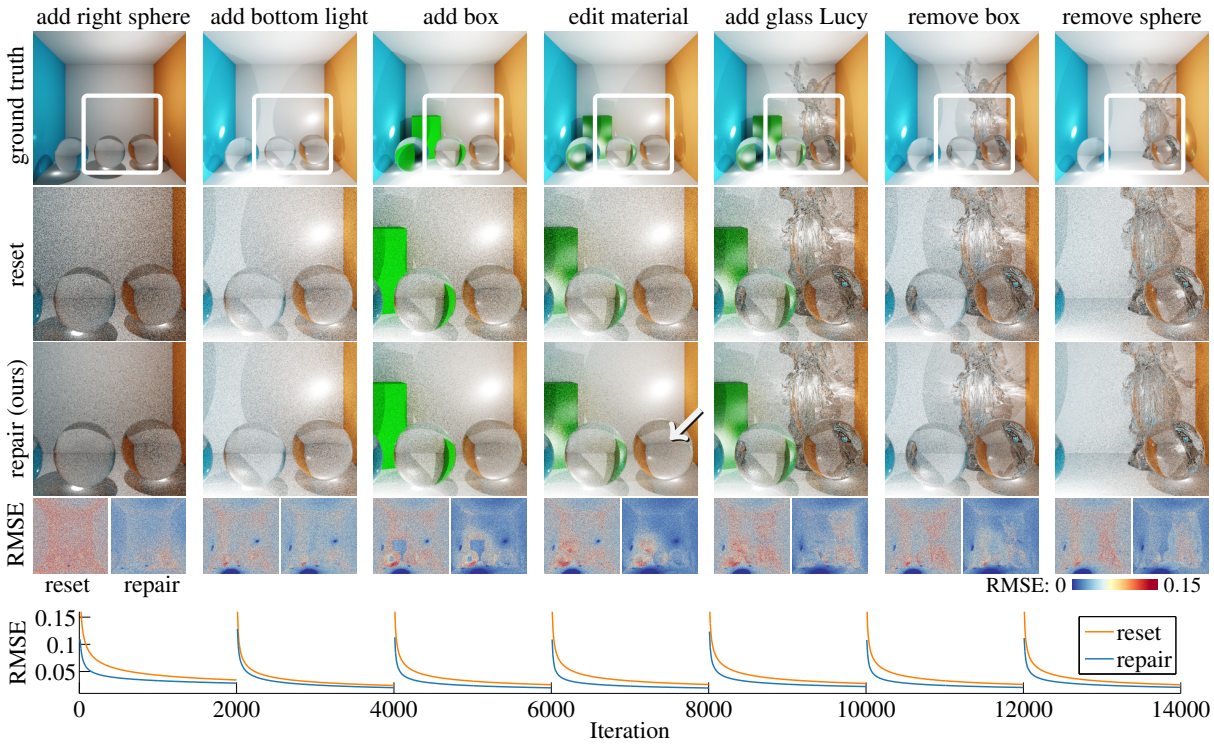


Figure 10: *Radiance after scene modifications. Below, the RMSE and convergence plots are shown for the trivial reset approach and our method (repair). Note that noise is reduced especially in areas that were not affected by the modification.*

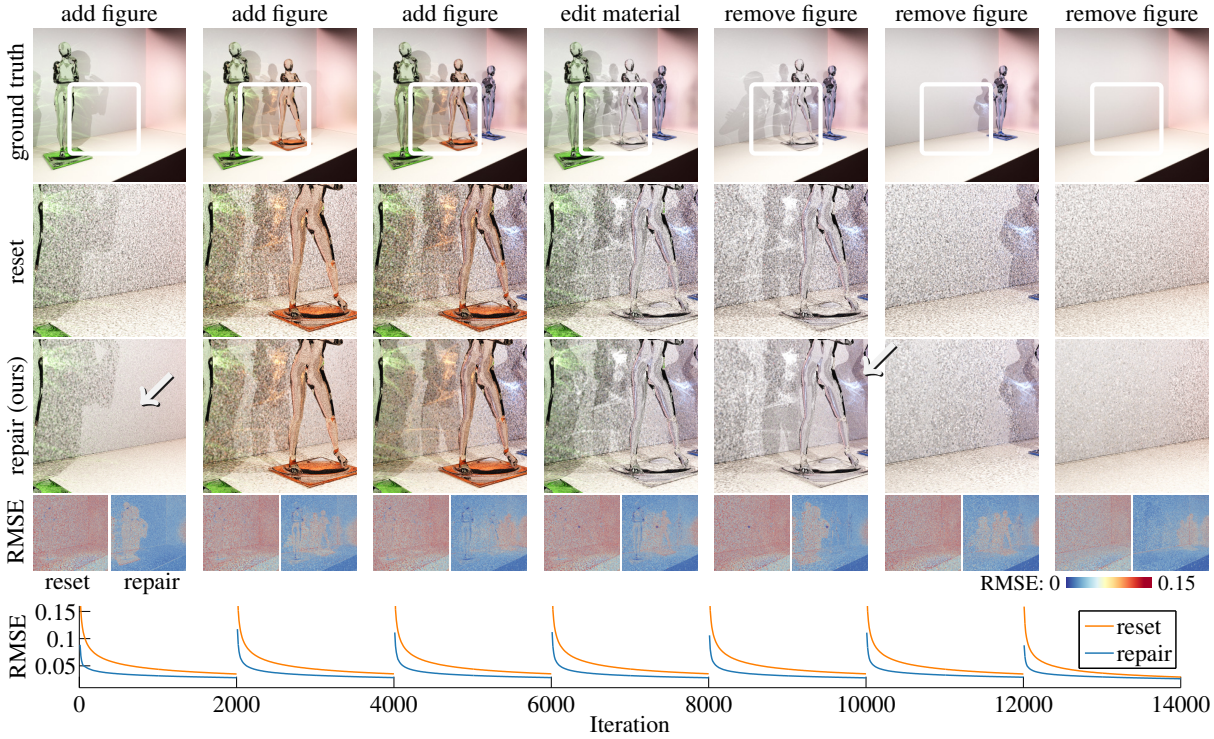


Figure 11: *In this figure, we add three glass figures into a scene, change the material of the middle one and eventually remove the figures. The radiance of the background and the unaffected caustics are very well preserved, as indicated by the arrows.*

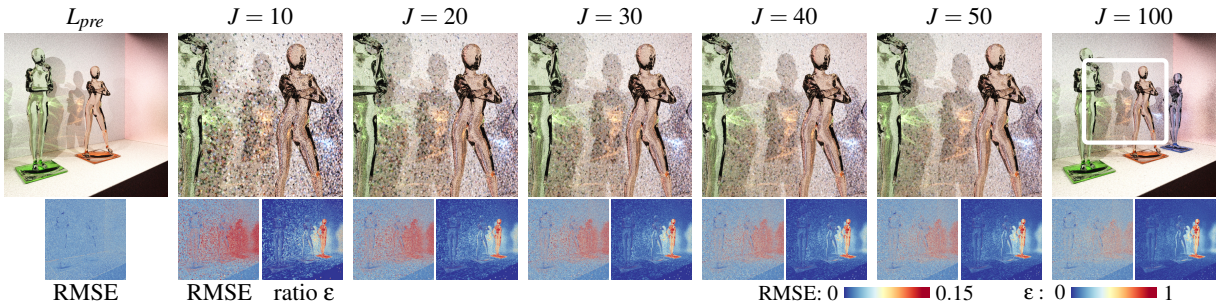


Figure 13: *Convergence series right after adding a blue glass figure. Shown are the synthesized radiance, RMSE and error ratio ε. As the error ratio ε becomes less noisy over time, the noise in the synthesized image reduces as well.*

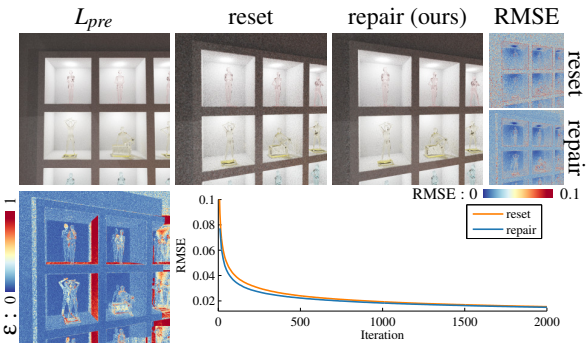


Figure 12: *Example of camera movement for $J = 100$. Radiance of visible scene parts can be preserved. Bottom row shows error ratio ε and the RMSE plot.*

sponding pixels average radiance over different scene parts, the RMSE improvement is smaller than for static scenes. Visual improvements can, however, be noticed in the top row in regions that could be seen in L_{pre}. Error ratio ε (bottom left) shows where reprojection errors are small.

4.2. Behavior right after Editing

While previous examples demonstrated the long-term convergence behavior, Fig. 13 focuses on the *intermediate* results right after a scene modification, which are presented as feedback to the user. It further depicts error ratio ε, showing where the scene modification has largest impact, which is where the progressively reducing noise is strongest. Next to it, a visualization of the RMSE depicts where coherence was exploited, resulting early in small errors.

4.3. Parameter Study

Number of Pre-Editing Iterations I

The first parameter is the number of PLT iterations *before* the editing operation. Fig. 14 shows how well previous results can be reused, depending on I. The images were traced up to $K = 200$ iterations after changing the radio's color from gray to blue. The plot shows that convergence is reached faster if more work was put into the simulation, i.e., for higher I.

Number of Correction Iterations J

The number of correction iterations can be specified in advance or be chosen interactively at runtime. In Fig. 15, we applied a varying number of correction iterations and afterwards continued the tracing up to $I + K = 2000$ iterations to study the impact of J on the long-term convergence. For $J = 0$ we simply continue with the previous pixel statistics in the new scene, causing a very slow blend, here, from a red curtain to a blue one. If J is chosen too small, glossy reflections are noisy (as shown for $J = 1$) and the noise in L_{diff} adds too much bias to the radiance, causing the curves to possibly cross the reset curve. The higher J, the better the long-term convergence and the visual quality, but the more overhead arises due to computing L_{diff}. In the bottom conver-

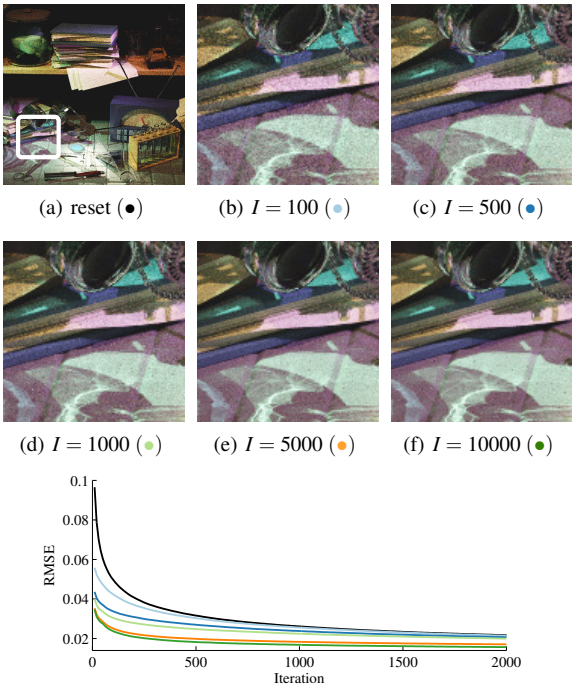


Figure 14: *Varying number of pre-editing iterations I. All lines start right after the editing operation (only showing K). Note that for higher I, convergence is achieved faster.*

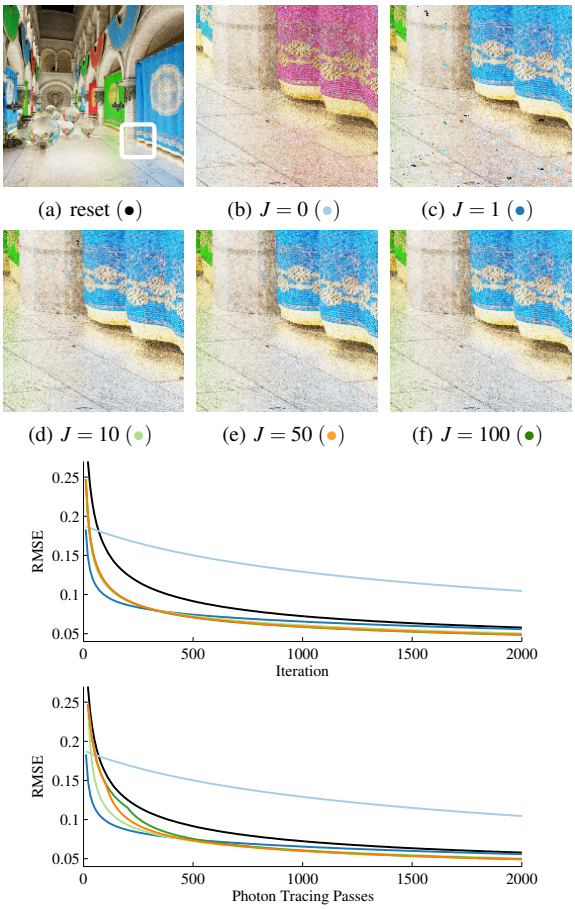


Figure 15: *Impact of the number of correction iterations J on the convergence. The bottom plot shows the number of photon tracing passes vs. RMSE. A cusp appears when we switch from correction iterations (two photon passes) to usual SPPM iterations (one photon pass). This results in a slower error reduction while corrections are taking place.*

gence plot, this overhead is taken into account, showing the number of photon tracing passes vs. RMSE, which better reflects the evolution of the error over time. The plots suggest a trade-off at $J = 50$ iterations, which is what we used.

Error threshold τ

The error threshold τ steers the overall quality of the achieved results, as demonstrated in Fig. 16. Setting $\tau = 0$ equals the trivial reset approach, i.e., every pixel resets. A low τ results in more noise in the early images and good long-term convergence. A value around $\tau = 0.5$ leads to rather local resets near the object to edit and thus weaker long-term convergence. When increasing τ, indirect illumination received from the modified object tends to become noisy, due to the noise in L_{diff}. As demonstrated earlier in Fig. 4(b), setting $\tau = 1$ leads to noticeable noise in case of strong radiance changes, e.g., at a caustic. Note that errors

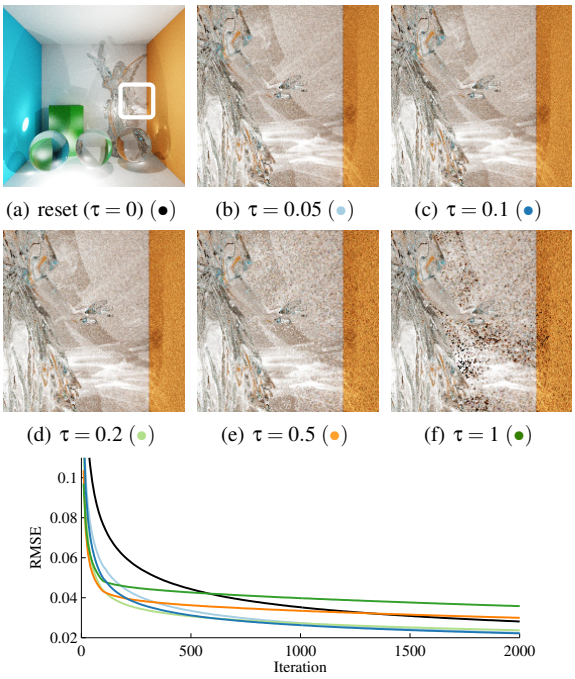


Figure 16: *Results and convergence plots for a close-up on a caustic, when using a different error threshold τ. The images show the radiance for $K = 1000$ after adding a glass Lucy. Note that for $\tau = 1.0$ the caustic remains as noise.*

might propagate from one scene modification to the next. Thus, a conservative choice for τ is recommendable. Our experiments showed that $\tau = 0.1 \ldots 0.2$ is advisable. For all tests, we used $\tau = 0.1$, which favors long-term convergence.

4.4. Light Transport Simulation Algorithm

Since our method operates on progressive radiance estimates, we can substitute the underlying light transport simulation. In Fig. 17, we applied our method to bidirectional path tracing (BPT) [LW93] and vertex connection and merging (VCM) [GKDS12]. The images depict the synthesized radiance after $J = 10$ correction iterations to show the noise behavior shortly after the modification. The RMSE visualizations show a significant improvement. While our method was primarily tested on SPPM, the results for BPT and VCM in these small scenes are promising. A comprehensive study of benefits with these techniques is a topic for future work.

4.5. Performance and Discussion

Our test system is equipped with an Intel Core i7-2600K CPU with 3.4 GHz, 24 GB RAM and an Nvidia GeForce GTX 560 Ti GPU with 2 GB VRAM. We measure timings for a viewport resolution of 600×600 pixels and emit $20k$ photons per photon tracing pass. Table 1 lists the timings of a usual SPPM iteration (Reset it.), a correction iteration (Repair it.) and the cost arising when interactively moving an

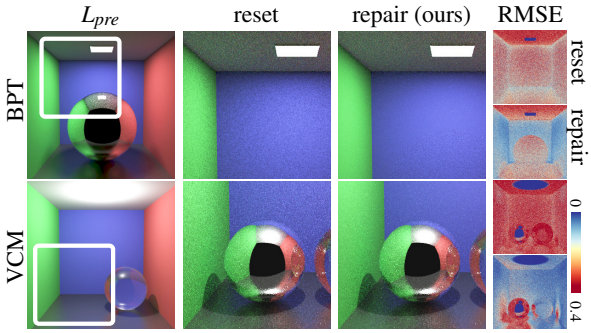
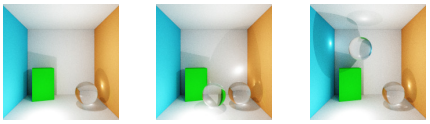


Figure 17: *Using our method to remove a sphere with bidirectional path tracing (BPT) and to add a sphere with vertex connection and merging (VCM). The radiance L_{pre} was traced in $I = 100$ iterations and for L_{diff} we used $J = 10$.*



Pass	Reset it.	Repair it.	Move it.	
1^{st}	33.6	33.6 (L_{old})	43.7 (L_{old})	(a), (b)
2^{nd}	—	43.7 (L_{new})	33.6 (L_{new})	
3^{rd}	—	—	32.0 (L_{old})	(c)
4^{th}	—	—	38.8 (L_{new})	
Total	33.6	77.3	148.1	

Table 1: *Timings in ms of the rendering passes.*

object (Move it.). Thereby, (a) refers to the adding of a glass sphere, which comprises two photon tracing passes; one for computing L_{old} and another one for computing L_{new}. In the right column the glass sphere is moved, consisting of a removal of the sphere (b) and an insertion at the new position (c). Thus, a correction iteration (add/remove object, change material, add/remove light) has twice as many, and an object movement four times as many photon tracing passes than a usual SPPM iteration, which has to be considered when designing an interactive system.

It is imaginable to store the radiance of the J correction iterations, i.e., L_{new} after a scene modification, so that L_{old} does not have to be recomputed at the next scene modification. This greatly accelerates correction iterations, as only one rendering pass is required, making it almost as expensive as a usual PLT iteration. The performance gain, however, is bought by a large memory consumption. For $J = 100$ at a resolution of 1000×1000 pixels more than 1 GB of memory is required to store L_{new}. Also, the number of correction iterations J must be known in advance. For this reason, we refrain from saving the radiance and instead recompute it.

Another angle to approach is to compute the difference radiance L_{diff} on-the-fly by the generation of positive and negative photons that reflect accordingly to add indirect light and shadows, whenever the object to edit is hit by a photon. This, however, would require deeper changes in the implementation of the underlying light transport simulation.

4.6. Limitations

Our system depends on heuristics, and as with all heuristics, there are failure cases, the one being here that slow gradual material or camera parameter changes might go unnoticed by the reset heuristic. If a user slowly varies a parameter, errors in indirect illumination or ghosting accumulate without triggering a reset. This problem, however, only occurs, if the user does not continuously change a parameter, but instead waits after each small step until the correction procedure has finished. If it has not finished, the former correction procedure would cancel and differences would be compared to the original image, where changes could be noticed.

5. Conclusions and Future Work

In this paper, we presented a method to accelerate convergence during interactive scene editing in progressive light transport simulations. We made use of the coherence between frames before and after a scene modification to reuse much of the previous results, which allows for faster previews of how an object integrates into the whole scene. Coherence is best exploited in unedited scene parts and most beneficial for effects that take long to converge, e.g., caustics. Our method is easy to integrate into existing pipelines, as it only requires a drop-in light transport simulation to create a visibility mask of the object to edit and a way to continue from a given intermediate solution, which is usually the case if the simulation can be paused and continued later.

For the future, we intend to focus on the BRDF of the edited object to accelerate convergence in regions in which strong changes occured. Nehab et al. [NSL*07] proposed amortized sampling, which smoothly blends older shading-samples into the most recent pixel statistics. This method and its extensions [YNS*09] provide an opportunity to display more temporally coherent results during interaction. Further, variation of camera parameters and motion blur could be studied to see how motion vectors and a priori knowledge on the camera lense can improve the reprojection. A scene-dependent and optimal automatic selection of parameters that achieves early noise-free previews and has good long-term convergence is also an avenue for future work. Further, more elaborate experiments with bidirectional path tracing and vertex connection and merging seem worthwhile topics.

Acknowledgments

We thank Frank Meinl for the Crytek Sponza scene and Tomáš Davidovič for his SmallVCM code (BPT and VCM). Dan Konieczka and Giorgio Luciano modeled the lab scene. Glass figures are by Dosch Design, and Dragon and Lucy are from the Stanford 3D Scanning Repository. This work was partially supported by the DFG grant GR 3833/3-1.

References

[BWG03] BALA K., WALTER B., GREENBERG D. P.: Combining edges and points for interactive high-quality rendering. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 631–640. 2

[DBMS02] DMITRIEV K., BRABEC S., MYSZKOWSKI K., SEIDEL H.-P.: Interactive global illumination using selective photon tracing. In *Rendering Techniques* (2002), vol. 28, Eurographics Association, pp. 25–36. 2

[DKHS14] DAVIDOVIČ T., KŘIVÁNEK J., HAŠAN M., SLUSALLEK P.: Progressive light transport simulation on the GPU: Survey and improvements. *ACM Trans. Graph. 33*, 3 (June 2014), 29:1–29:19. 2

[GG14] GÜNTHER T., GROSCH T.: Distributed out-of-core stochastic progressive photon mapping. *Computer Graphics Forum 33*, 6 (2014), 154–166. 2

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (Nov. 2012), 192:1–192:10. 1, 2, 3, 9

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 28*, 5 (Dec. 2009), 141:1–141:8. 2, 3

[HJ10] HACHISUKA T., JENSEN H. W.: Parallel progressive photon mapping on GPUs. In *SIGGRAPH Asia Sketches* (2010), ACM, pp. 54:1–54:1. 2

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27*, 5 (Dec. 2008), 130:1–130:8. 2

[HPB06] HAŠAN M., PELLACINI F., BALA K.: Direct-to-indirect transfer for cinematic relighting. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2006), 1089–1097. 3

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A path space extension for robust light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6 (2012), 191:1–191:10. 1, 2

[HR13] HAŠAN M., RAMAMOORTHI R.: Interactive albedo editing in path-traced volumetric materials. *ACM Trans. Graph. 32*, 2 (Apr. 2013), 11:1–11:11. 3

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Proc. Eurographics Workshop on Rendering Techniques* (1996), Springer-Verlag, pp. 21–30. 2

[JNSJ11] JAROSZ W., NOWROUZEZAHRAI D., SADEGHI I., JENSEN H. W.: A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph. 30*, 1 (Feb. 2011), 5:1–5:19. 2

[JZJ08] JAROSZ W., ZWICKER M., JENSEN H. W.: The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. Eurographics) 27*, 2 (2008), 557–566. 2

[KD13] KAPLANYAN A. S., DACHSBACHER C.: Adaptive progressive photon mapping. *ACM Trans. Graph. 32*, 2 (2013), 16:1–16:13. 1, 2

[KGH*14] KŘIVÁNEK J., GEORGIEV I., HACHISUKA T., VÉVODA P., ŠIK M., NOWROUZEZAHRAI D., JAROSZ W.: Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4 (2014), 103:1–103:13. 2

[KISE14] KLEHM O., IHRKE I., SEIDEL H.-P., EISEMANN E.: Property and lighting manipulations for static volume stylization using a painting metaphor. *IEEE Transactions on Visualization and Computer Graphics 20*, 7 (July 2014), 983–995. 3

[KPD10] KERR W. B., PELLACINI F., DENNING J. D.: Bendylights: Artistic control of direct illumination by curving light rays. In *Proc. Eurographics Symposium on Rendering* (2010), EGSR, pp. 1451–1459. 3

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph. 30*, 3 (May 2011), 25:1–25:13. 2

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. *Proc. Compugraphics* (1993), 145–153. 2, 3, 9

[MTAS01] MYSZKOWSKI K., TAWARA T., AKAMINE H., SEIDEL H.-P.: Perception-guided global illumination solution for animation rendering. In *Proc. Computer Graphics and Interactive Techniques* (2001), SIGGRAPH '01, ACM, pp. 221–230. 2

[NJS*11] NOWROUZEZAHRAI D., JOHNSON J., SELLE A., LACEWELL D., KASCHALK M., JAROSZ W.: A programmable system for artistic volumetric lighting. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (July 2011), 29:1–29:8. 3

[NSL*07] NEHAB D., SANDER P. V., LAWRENCE J., TATARCHUK N., ISIDORO J. R.: Accelerating real-time shading with reverse reprojection caching. In *Proc. Symposium on Graphics Hardware* (2007), pp. 25–35. 5, 10

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: a general purpose ray tracing engine. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4 (July 2010), 66:1–66:13. 2

[PVL*05] PELLACINI F., VIDIMČE K., LEFOHN A., MOHR A., LEONE M., WARREN J.: Lpics: A hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (July 2005), 464–470. 2

[RKKS*07] RAGAN-KELLEY J., KILPATRICK C., SMITH B. W., EPPS D., GREEN P., HERY C., DURAND F.: The lightspeed automatic interactive lighting preview system. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (July 2007), 25:1–25:11. 3

[SNM*13] SCHMIDT T.-W., NOVÁK J., MENG J., KAPLANYAN A. S., REINER T., NOWROUZEZAHRAI D., DACHSBACHER C.: Path-space manipulation of physically-based light transport. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4 (July 2013), 129:1–129:11. 1, 3

[SPN*14] SCHMIDT T.-W., PELLACINI F., NOWROUZEZAHRAI D., JAROSZ W., DACHSBACHER C.: State of the art in artistic editing of appearance, lighting, and material. In *Eurographics 2014 - State of the Art Reports* (Apr. 2014), Eurographics. 1, 2

[TPWG02] TOLE P., PELLACINI F., WALTER B., GREENBERG D. P.: Interactive global illumination in dynamic scenes. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 3 (2002), 537–546. 2

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proc. Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, ACM, pp. 65–76. 2

[WDG02] WALTER B., DRETTAKIS G., GREENBERG D. P.: Enhancing and optimizing the render cache. In *Proc. Eurographics Workshop on Rendering* (2002), pp. 37–42. 2

[WDP99] WALTER B., DRETTAKIS G., PARKER S.: Interactive rendering using the render cache. In *Proc. Eurographics Conference on Rendering* (1999), pp. 19–30. 2

[WG12] WEISS M., GROSCH T.: Stochastic progressive photon mapping for dynamic scenes. *Computer Graphics Forum (Proc. Eurographics) 31*, 2 (2012), 719–726. 2

[YNS*09] YANG L., NEHAB D., SANDER P. V., SITTHI-AMORN P., LAWRENCE J., HOPPE H.: Amortized supersampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 28*, 5 (2009), 135:1–135:12. 10