SN Computer Science (2025) 6:353
https://doi.org/10.1007/s42979-025-03863-z

ORIGINAL RESEARCH

Automated Prioritization for Context-Aware Re-rendering in Editing

Annalena Ulschmid[1] · Katharina Krösl[1,2] · Michael Wimmer[1] · Bernhard Kerbl[1]

Received: 28 November 2024 / Accepted: 6 March 2025

Abstract
Real-time Monte Carlo path tracing has become a feasible option for interactive 3D scene editing due to recent advancements in GPU ray tracing performance, as well as (AI-accelerated) denoising techniques. While it is thus gaining increasing support in popular modeling software, even minor edits such as adjusting materials or moving small objects typically require current solutions to discard previous samples and restart the image formation process from scratch. A recent solution introduced two adaptive, priority-based re-rendering techniques implementing incremental updates while focusing first on reconstructing regions of high importance and gradually addressing less critical areas. An extensive user study compared these prioritized renderings with conventional same-time re-rendering to evaluate their effectiveness for interactive scene editing. Results indicate a significant preference for incremental rendering techniques for editing small objects over traditional full-screen re-rendering with denoising, even with basic priority policies. Building upon these results, we revisit the underlying design choices and derive more sophisticated priority policies that respect global illumination effects (shadows and reflections) as well as employing attention-based techniques (based either on eye tracking to prioritize areas in the user's gaze or, alternatively, using the cursor position).

Keywords Path tracing · Scene editing · Adaptive rendering · Empirical studies

Introduction

To visualize complex scenes with effects such as global illumination, reflections, soft shadows, and caustics, Monte Carlo (MC) path tracing [9, 20] is a widely adopted technique in physically-based rendering. It naturally incorporates such effects without any special treatments, in contrast to, e.g., rasterization-based pipelines. In the movie industry and production rendering in particular, offline path tracing has become a fundamental tool. For each pixel in a rendered image, a path originating at that pixel is stochastically sampled and traced into the scene. Subsequently, multiple of these paths are computed and accumulated over time. The sampling quality is typically directly defined by the number of spp invested in the process. While path tracing is unbiased due to its stochastic nature, achieving a noise-free, low-variance image requires a large number of samples. Consequently, even under optimal conditions, several seconds are needed to reach an acceptable image quality.

Given the high computational cost of path tracing, many applications that demand quick feedback loops (such as interactive scene editing previews) often rely on less realistic, real-time approximation methods (e.g., via rasterization). Interactive feedback ultimately remains a persistent challenge in artistic editing as highlighted by [27]. While these approximations aim to closely mimic the results of converged path-traced renders, they fall short at capturing complex light transport effects accurately. For more precise previews, interactive path tracing during scene editing is desirable, to provide a seamless and productive workflow for digital artists that current approximation methods cannot offer.

In support of interactive ray-based rendering, modern graphics hardware has advanced to support ray-tracing pipelines alongside traditional rasterization. However, in

✉ Annalena Ulschmid
a.ulschmid@cg.tuwien.ac.at

✉ Bernhard Kerbl
kerbl@cg.tuwien.ac.at

Katharina Krösl
kkroesl@cg.tuwien.ac.at

Michael Wimmer
wimmer@cg.tuwien.ac.at

1 TU Wien, Vienna, Austria

2 VRVis Forschungs-GmbH, Vienna, Austria

Published online: 05 April 2025

real-time settings with 30–60 frames per second, modern GPU path-traced outputs are usually limited to approximately 1 spp per frame. To complement these noisy results, multiple powerful noise-reduction techniques have been developed, enabling the refinement of grainy, low-quality images [2, 6, 11, 17, 26, 29, 34].

With these tools, it is possible to obtain satisfactory rendering quality by accumulating a few dozen samples over the course of several seconds. However, most interactive solutions still discard all previously accumulated samples whenever the scene changes, e.g., through camera movements, object modifications, or material edits. This all-or-nothing behavior preserves correct appearance in response to any scene edits, but also resets the image quality and can cause artifacts, such as flickering or blurriness, during the scene editing process. Global image updates are justified when the entire scene changes, but are unnecessarily disruptive for minor modifications affecting localized regions: Artists must retain their vision of the desired change while observing an immediate regression from converged images to grainy or blurry visuals during the initial moments of image formation after an edit. While some production renderers allow manually specifying a static region of interest for high-quality previews, this approach cannot dynamically adapt to significant scene changes involving large object movements or complex global illumination effects. We hypothesize that focused, prioritized re-rendering updates, designed with an artist's workflow in mind, would enable them to better follow the impact of small edits and obtain accurate previews faster, thus enhancing the interactive process.

[31] propose to identify areas of high priority and re-render these first at higher quality when a scene edit occurs. They employ an importance policy, which ideally identifies all image areas where changes have a high impact. Updates are then scheduled gradually and in an incremental fashion to re-render the image in the order of importance. This way, the entire image is eventually updated and physical correctness is maintained, but an artist is allowed to first focus on their modification.

Additionally, Ulschmid et al. conducted a detailed user study comparing two prioritization methods [31] for real-time scene editing with a global approach [2], to verify the usability of such incremental re-rendering methods in an interactive editing setting. Based on their findings, this paper extends their method, incorporating two new incremental re-rendering flavors with increased automation and better user apprehension in mind. While the original method computes the Chebyshev distance of pixels from the modified object's position on the screen to determine the importance, we propose using more refined priority metrics taking global illumination effects and the users' actual focus point into consideration. Some participants of the original study also reported symptoms akin to "simulator sickness", either due to the flickering of global methods or the spiral-like spreading of incremental re-rendering; We thus added a version with a softer transition after the scene edits, which is initially biased but converges to an unbiased solution. A comparison of different global and incremental techniques can be seen in Fig. 1.

In addition to revisiting the key insights from the original study on incremental re-rendering with and without denoising, this paper provides further experiments for verifying our two new re-rendering methods. Early evidence strongly indicates user preference for the two re-rendering methods introduced in this paper in all cases. Our main contributions are thus:

- Development of a practical, user-friendly, incremental real-time path tracing pipeline. Scheduled tiles are repeated multiple times in the buffer input of a path tracer. This enables replacing conventional, global re-rendering with same-time, adaptive re-rendering, requiring only minimal changes to underlying rendering backends.
- Multiple implementations of automated, prioritized re-rendering policies for editing: we include both noisy and denoised variants, supporting different priority measures and transition modes.
- Validation using a detailed user study that investigates basic incremental re-rendering methods among experienced artists. Analysis of the results encouraged further, more refined re-rendering techniques and experiments presented in this paper.

Related Work

The foundations of MC path tracing were originally provided in ground-breaking work by [9]. [20] more thoroughly describe their implementation of an offline path tracing framework. Hardware-accelerated ray-tracing was enabled by the recently introduced Turin architecture [16], facilitating real-time path tracing at a low sample count.

Adaptive Sampling and Denoising

For a thorough overview of recent literature on adaptive sampling and denoising techniques we refer the reader to the original publication by [31]. Most adaptive sampling techniques either respect a-priory measures such as the gradients based on local 3D geometry and material models, or a-posteriori statistics such as the variance or error in certain image regions [35]. In contrast to standard adaptive sampling techniques [5, 13, 18, 32, 35], we exploit the priors of the editing process by using the information where changes occurred to steer the sample distribution.

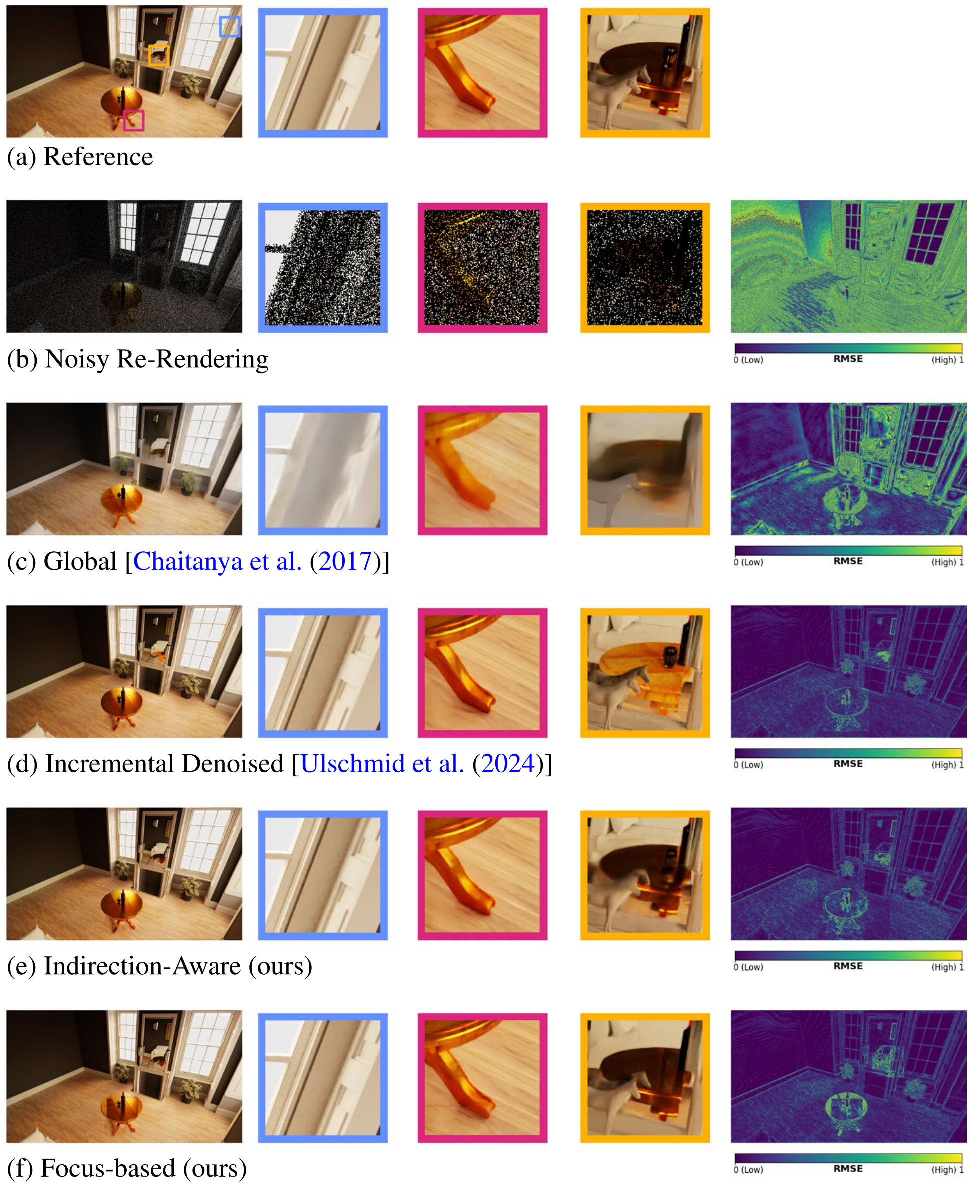


Fig. 1 **a**–**e** Comparison of different re-rendering methods with 1 samples per pixel (spp) sample budget to the ground truth (**a**) of an interactively edited scene, after increasing the glossiness of the table material. In **b** accumulated samples before the edit are discarded during the computation of the new visualization and the image is updated globally, resulting in disruptive quality degradation everywhere. **c** Employs denoising to reconstruct high-fidelity images from (**b**). **d**–**f** Use a tile-based approach with incremental updates, concentrating available samples on a subset of screen-space tiles in order of importance to avoid both noise and overblurring in the entire scene. **d** is measuring the priority by computing the Chebyshev distance from the projected position of the edit on screen. **e** Uses information gathered during the path tracing process to determine the tiles' priority, based on direct and indirect lighting effects, and can thus better capture changes in reflections and shadows. **f** Uses both the edit's screen position and a user defined focus point, which can also be selected via eye-tracking. Both **d** and **f** sample the priority tiles with 16 spp

Denoising techniques often rely on spatial and historical, temporal information, as well as variance measures. One prime example of a traditional denoiser is Spatiotemporal Variance- Guided Filtering (SVGF), which strives to output a temporally coherent image sequence by accumulating multiple samples, incorporating sparsely sampled gradients in the decision of whether to include a previous sample in a pixels history buffer Schied et al. [26]. Recently, deep learning-based denoisers have been introduced such as Intel's Open Image Denoise (OIDN) Intel [8] used for example by Unreal Engine or Nvidia's OptiX denoiser based on the research of Chaitanya et al. [2]. The latter adopts a recurrent autoencoder additionally using auxiliary buffers to include for example depth information and normals. For some optical effects such as specular materials, these auxiliary buffers however fail to correctly guide the denoising as they contain no information about the reflected scene. Deep learning-based denoisers thus often tend to hallucinate and lack temporal consistency. A recent work by [25] introduces an approach purely based on statistical reasoning about the per-pixel sample distribution, thus avoiding expensive pre-training and data dependencies, compared to neural denoisers. In a dynamic context however, standard denoising techniques [2, 6, 11, 17, 26, 29, 34] discard all previous samples to compute the current frame whenever something in the scene changes and only reuse them via temporal reprojection during the post-processing denoising step. Instead of these global updates, we propose a re-rendering approach with incremental updates tailored to artists, using render-time information to adaptively guide the sample distribution.

Scene Editing and Perception

[12] present an incremental ray-tracing scheme, monitoring changes by employing a spatial voxel partition and hash indices. To ensure consistent scene editing, [3] suggest utilizing progressively computed difference images to determine areas where modifications significantly impact the scene. Their approach relies on stochastic progressive photon mapping, specifically targeting slowly converging effects such as caustics, which are particularly adversely affected by a global reset. Building up on this, [23] adapt the previous approach to fit into a general control-variate integration scheme, investigating combinations of estimators based on covariance for offline editing in static scenes with gradient-domain rendering. Although they provide a comprehensive qualitative analysis of their technique, they do not assess the user benefits of such adaptive methods, particularly in interactive editing contexts. Moreover, the aforementioned approaches utilize uniformly distributed rather than prioritized, adaptive screen-space samples. In contrast, we propose and rigorously evaluate re-rendering techniques that automatically identify and prioritize regions of interest. Our solutions are specifically crafted for seamless integration into real-time pipelines, ensuring immediate visual feedback.

[33] recently introduced a residual path integral, which allows for importance sampling of light paths influenced by changes between frames, focusing on both scene editing and animation. Additionally, they investigate path mapping methods between two frames, generalizing from existing gradient-domain mapping approaches. Their evaluation also includes light, material and transformational edits with same-time renderings, however they only analyze the visual quality and not the editing experience for an artist. Similar to our findings, they also report their method to work best for small scene changes, as the path mapping struggles with more dramatic changes between frames. Xu et al. however do not consider the artist's focus, whereas we additionally support attention-based importance metrics through eye or mouse pointer tracking.

To investigate how different noise levels are perceived in images rendered with MC path tracing, [15] conducted several studies. They found that participants primarily relied on their most central vision, as well as non-textured and brightly lit areas to detect noise. Foveated rendering for path tracing in Virtual Reality (VR) utilizes eye-tracking technology to identify regions of greater interest to the viewer, allowing for a higher sample count to be allocated to those areas. This fundamental concept aligns well with our approach, as similarly to foveated rendering, our method takes advantage of the fact that unbiased MC rendering enables straightforward averaging of partial results, resulting in higher quality at specific, localized image areas. Additionally, we estimate regions of focused attention by also utilizing eye-tracking or alternatively the mouse position.

Prioritized Re-rendering

Building upon [31], we first describe two fundamental variants for incremental re-rendering techniques with prioritized screen updates according to Chebyshev distance (Sects. Incremental Updates and Denoised Incremental Updates). Next, we introduce two refined priority metrics that additionally consider changes due to global illumination effects and the user's focus point on the screen (Sects. Indirection-Aware Priority Policy and Focus-Based Re-Rendering). We implemented our approaches in Nvidia's Falcor framework for rapid prototyping of real-time techniques [10] (version 5.1.). All code is available online Ulschmid et al. [30].

Falcor utilizes render graphs constructed from individual render passes. Our implementation mainly changes the Megakernel Path Tracer and Accumulation Pass, while adding new passes to introduce basic editing features, as Falcor originally is only able to render pre-scripted animations. We thus enable three modes of operation: global updates, continuous refinement, and incremental updates, depicted in Fig. 2. In the global update mode, all previously accumulated samples are discarded, and the new frame is rendered uniformly at 1 spp. The continuous refinement mode then accumulates per-pixel samples for each tile using a moving average. Traditional re-rendering exclusively relies on these two modes, defaulting to the global update whenever there is some camera movement or the scene changes. By introducing an additional mode with incremental updates, we handle scene modifications separately. During both global updates and continuous refinement, we use Nvidia's OptiX [2] framework for noise reduction.

All methods show the same runtime complexity as conventional re-rendering (aside from a small delta for tile merging and denoising in ours, as well as constructing the tile priority queue, see below) and achieve real-time performance during editing (30 frames per second at 1080p resolution) on state-of-the-art hardware (Nvidia RTX 3060).

Incremental Updates

To support incremental updates, [31] modified Falcor's Megakernel Path Tracer to operate on tiles representing separate parts of the scene in image space, rather than the entire viewport. By repeatedly selecting specific tiles in the input buffer of the unbiased path tracer, they can sample these regions at higher quality without altering the underlying implementation itself. The number of repetitions determines the sample count within a tile, which we refer to as the

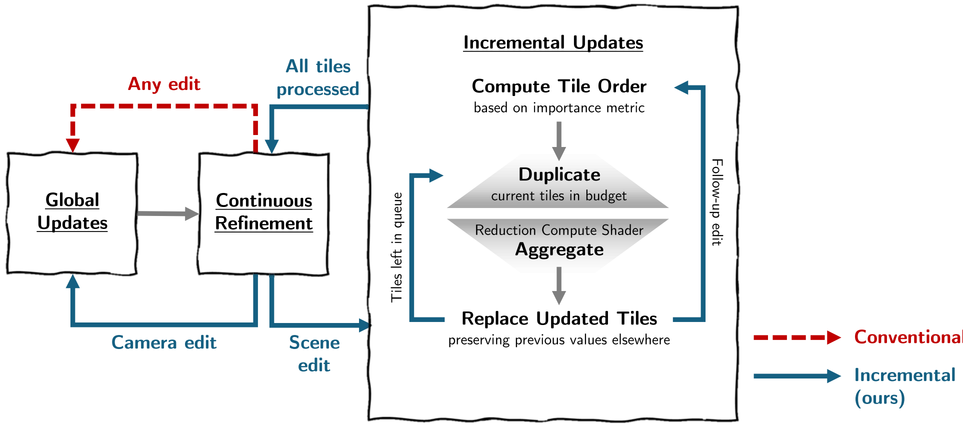


Fig. 2 Diagram of our suggested re-rendering process containing incremental updates. Image adapted from [31]

tile quality. This approach enforces a tile budget, allowing to redirect the original GPU workload from full-screen updates to just a few tiles.

After the standard path tracing procedure, a compute shader aggregates the duplicated tiles' content using a tree-based averaging reduction. Thus, a few high-quality screen tiles are computed in the same time it would take to render an entire scene at a lower sample rate, disregarding variations in scene complexity. The buffer layout is shown in Fig. 3.

Falcor's path tracing pipeline includes a sample accumulation stage, which aggregates color values over time as long as the scene remains unchanged, resetting them otherwise. Our approaches incrementally replace only a subset of all the tiles that need to be re-rendered, preserving values in the other tiles. This updates the most affected image regions at high quality quickly, while less influenced regions are handled later. If no further scene modifications occur, we switch to full-screen accumulation mode after one iteration of incremental updates. For continuous modifications (e.g., carefully moving an object), the incremental updates are restarted from the modified object's most recent screen position.

Within a given budget, higher target tile quality means fewer tiles can be processed per update, resulting in smaller re-rendered image regions and lower update speed to re-render the entire image after an edit. Artifacts can therefore occur when a user interrupts the incremental update procedure by continuously changing the object position with very high tile quality settings. If the resulting tile size is smaller than the object, moving it may result in ghosting artifacts, as parts of the object in its old position persist. To allow for variations in user preference (consistency vs. quality vs. responsiveness), tile size and quality can be modified manually, but we also later introduce techniques adding automatic parameter selection based on refined importance metrics.

During global updates for camera movement and continuous sample accumulation, tiles are processed top to bottom and left to right. For incremental updates triggered by scene modifications, Ulschmid et al. suggest a tile order which is computed by constructing a spiral starting at the object's projected position in image space (see Fig. 4a). This importance metric thus defines the priority P of a tile t with center (x_t, y_t) in image space using the Chebyshev distance from the selected object o with center (x_o, y_o) as follows:

$$P(t) = max(|x_o - x_t|, |y_o - y_t|) \tag{1}$$

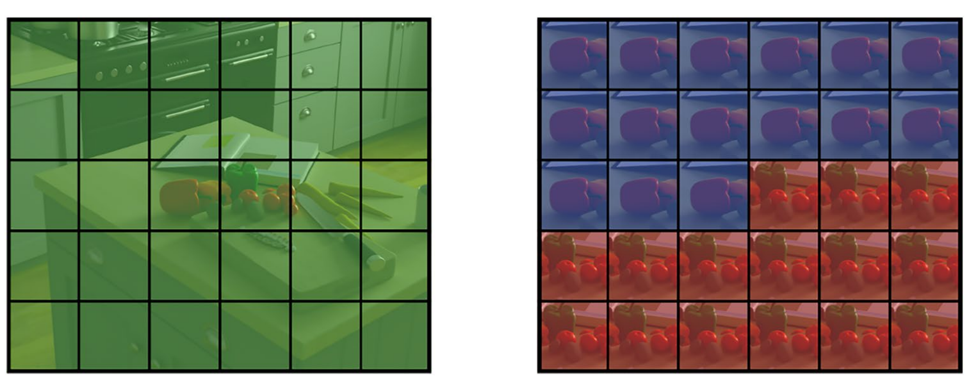


Fig. 3 Tiled buffer layout. Left: Standard approach of sampling each tile in the entire image once (green). Right: Our approach of repeatedly sampling only a few selected tiles (in the depicted case two in red and blue) during the incremental updates, while the remaining tiles remain unchanged. Note that the actual tiles are much smaller than depicted here. Image taken from [31]

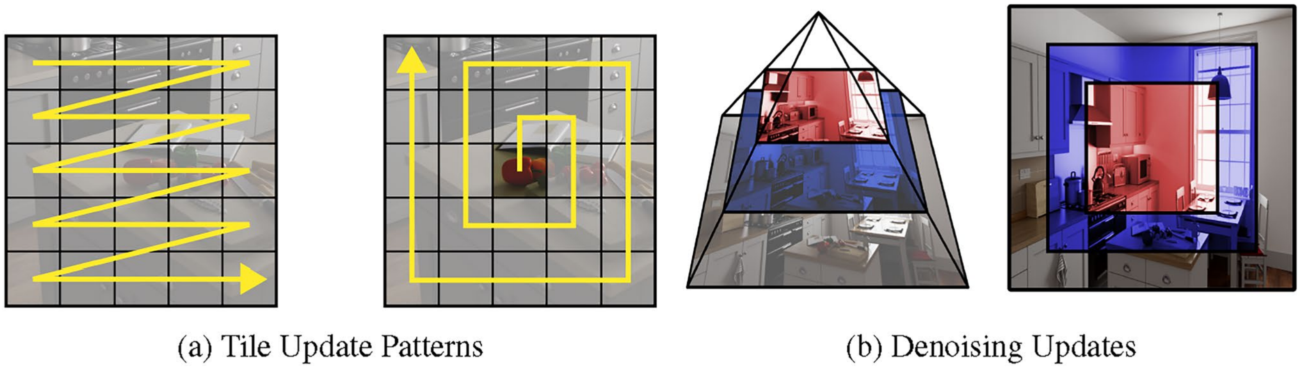


Fig. 4 **a** Left: tiles are updated simultaneously in a resting scene. Right: a spiral determines the order of tile updates after an edit. Image taken from [31]. **b** Combining incremental updates with a denoiser. The denoised region expands when a certain threshold is reached, with the oldest result being displayed on top to ensure temporal stability

Denoised Incremental Updates

The method described above is combined with a noise reduction technique. In our implementation, the OptiX denoiser is applied for continuous refinement and during incremental updates. Following [31], we resize its domain to the space of already re-rendered tiles based on a certain threshold, as resizing each frame can be costly. To increase visual consistency, we deal with temporal stability artifacts by always keeping the earlier, smaller denoising result in front, placing later results behind it whenever a certain size threshold is reached (Fig. 4b).

Higher threshold values can result in more noticeable edges between old and new updates and block-like visuals, but improve performance without significantly affecting perceived responsiveness during editing. Our default threshold value is set to 200.

Indirection-Aware Priority Policy

The above methods, while automatically identifying the object's center as the region of interest, require manual setting of the tile quality, which influences the size of the re-rendering area. Wrongly selecting the tile quality can also lead to ghosting artifacts. Additionally, the Chebyshev distance always results in a symmetric, evenly-sized quadratic region during scene updates. We thus developed a more sophisticated priority metric, which can adapt to arbitrary object shapes and re-render them at the highest possible resolution, which is also automatically selected. As the selected tiles contain the entire object, the ghosting artifacts, which previous methods suffered from, are removed.

To achieve this, we use information gathered during the path tracing process to determine the tiles' priority, based on both direct and indirect lighting effects. Importantly, this allows us to react to, e.g., changes in the image due to reflections and cast shadows. To this end, we store a weighted sum of direct and indirect ray intersections for the pixels in each tile with the modified object. Storing these values per tile conveniently results in a smaller storage overhead than computing a pixel-wise priority and also integrates easily with the underlying tile-based architecture. Direct hits are weighted the highest, yielding the largest priority values. Indirect hits are categorized into different priorities, with reflected rays being weighted higher than shadow rays, as can be seen in Fig. 5.

The higher-priority indirect hits for reflections are combined with the direct hits to determine the number of tiles needed to be rendered during an ongoing scene edit. Based on this, the tile quality is automatically determined such that as many tiles as possible fit into the tile buffer, while still adhering to the same-time rendering budget established above.

To save computation time, this priority is computed only once when the object is selected, as well as after a scene edit when all tiles have been processed and we return to the continuous refinement mode. It thus still contains noise, which could be solved by accumulating the indirect hits over multiple timesteps. However, we found that this policy suffices to reliably detect reflections and shadows in most cases.

During a translational scene edit, the screen space motion vector of the object is used to determine which new tiles to cover. In case a tile's center shifted by the motion vector falls outside the tile's borders, the new tile the shifted center falls in, is added, while the old tile is removed from the priority queue. As tiles are added and removed due to object movement, the tile quality is updated as well to adapt to the tile count.

As neural denoisers heavily rely on additional information in the G-buffers, the re-rendering of tiles with indirect

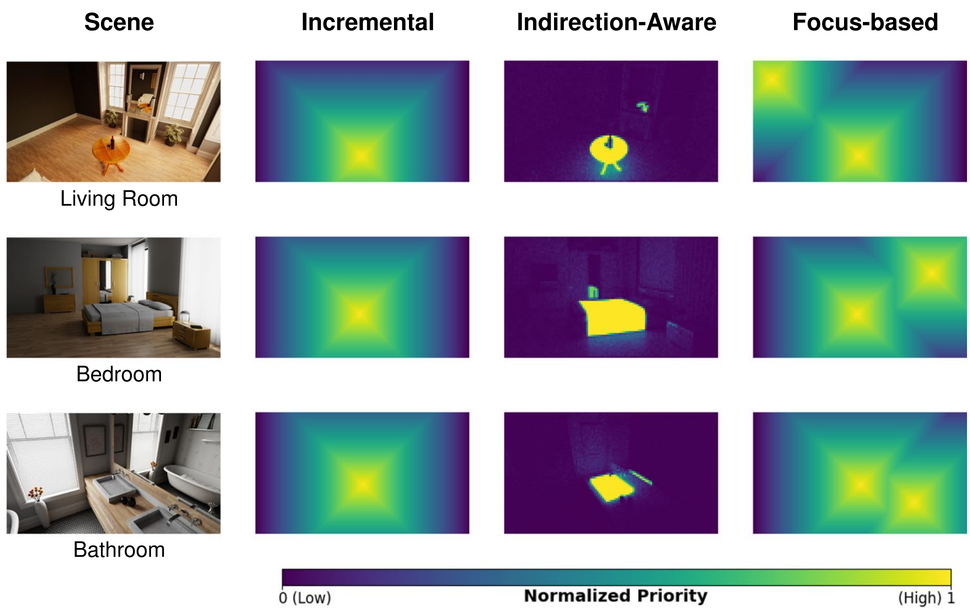


Fig. 5 Visualizations of different tile-wise priority policies in three different scenes, where each selected object (table, blanket, sink) is slightly moved along the y-axis. Global approaches result in a constant priority throughout the entire image. The incremental approach described by [31] assumes the object's screen-space position as the center of attention and sorts the tiles based on the Chebyshev distance to this center. Our indirection-aware approach gathers information during the path tracing process to determine direct and indirect hits of the selected object during the path tracing process before a scene edit. This way, it can assign tiles covering a reflection of the object or the object's shadow a higher importance. Lastly, the focus-based priority policy is constructed using the minimal Chebyshev distance to either the object's screen space center or the user-defined attention

reflections experiences more noise-reduction artifacts than tiles that directly hit an object. To combine this policy with a noise reduction technique, we denoise the rectangular image region spanned by all tiles selected during priority computation.

This new importance metric can be combined with the priority established by the Chebyshev distance to correctly re-render all tiles in the image eventually. According to [31] however, some artists reported "simulator sickness" from the spiral-like spread. We thus propose a smoother transition mode inspired by the work of [14]. After computing high-quality version of the tiles most affected by the edit, the other tiles eventually receive corrected sample values, which are accumulated together with the existing, now wrong samples. This way, the other samples values are initially biased towards the old results, but will eventually converge to the correct solution.

Focus-Based Re-rendering

Apart from localized modifications, several scene edits (e.g., light changes) can have a uniform, global influence on the rendered image even if the camera remains stationary. This makes it difficult to ascribe priorities to affected tiles: we thus added an importance measure for this scenario, where an artist can directly influence the priority influenced by their focus. In addition to the object's screen-space center, used for the incremental update described by [31], we add either the user's screen-space gaze position when using eye tracking or a manually selected position as a second center of attention.

The priority is then computed as the minimum of the Chebyshev distances to both the object and gaze position (visualized in Fig. 5). In our implementation, this is handled by simultaneously constructing a spiral starting from both centers and alternately adding tiles along their path to our priority queue. This way, when selecting the same tile quality as for the naive incremental update method described in Sections "Incremental Updates" and "Denoised Incremental Updates", a lower area of the selected object is covered however, as part of the tile budget is spend on rendering the focus region (see Fig. 1).

Evaluation

Especially for smaller edits, reusing tiles from previous, converged images for re-rendering can significantly increase quality as measured by metrics, e.g., PSNR and PSNR-HVS-M, [19, 21]. Figure 6 illustrates this for an edit in the *Bathroom* scene, where one of the cups on the sink is translated along the y-axis. Starting from an almost converged

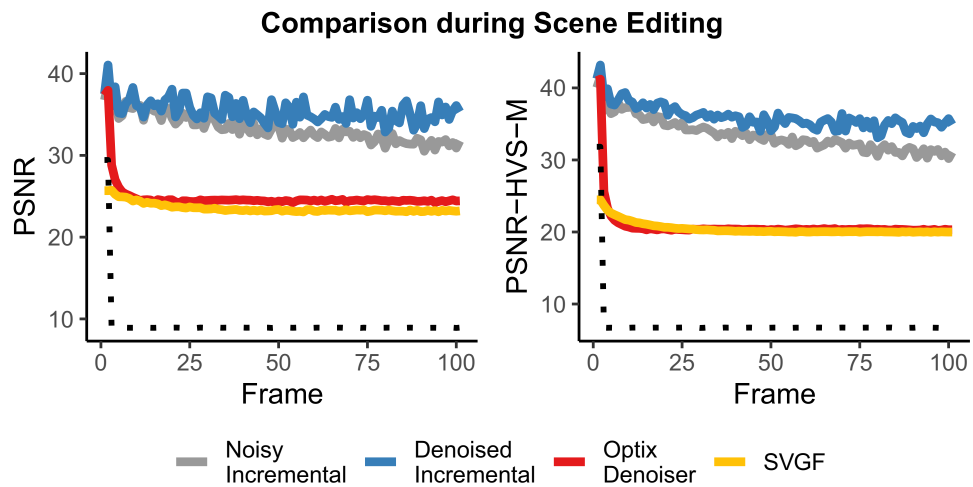


Fig. 6 Comparison of different global (Optix [2], SVGF [26]) and incremental re-rendering methods [31] during an edit in the Bathroom scene, starting at 1024 spp. The dotted line represents 1 spp baseline without denoising. As our method replaces converged tiles before the edit with new ones, image accuracy degrades gracefully while moving the cup. Image taken from [31]

rendering with high quality expressed by high PSNR and PSNR-HVS-M values, the cup is uniformly moved over a duration of 100 frames. For each frame the chosen metrics are evaluated in comparison to a converged rendering of the transformed cup. Both metrics demonstrate that the image quality degenerates slower during the edit when using an incremental method as the converged result is kept for most parts of the image, where the edit is less influential. However, our main focus in this work lies on verifying whether such policies also result in an improved usability and interactive editing experience. [31] conducted an extensive user study to compare conventional and priority-based re-rendering, covering the methods discussed in Sections "Incremental Update" and "Denoised Incremental Updates". Following evaluation and given participants' feedback, the adapted methods (Sections "Indirection-Aware Priority Policy" and "Focus-Based Re-rendering)" were evaluated in a series of custom experiments.

Prioritization vs. Global Methods

We first revisit the study design and key insights of [31]. The authors investigated if artists prefer an incremental or global rendering method in the context of scene editing and if their preference depends on a specific editing scenario. They measure preference w.r.t. perceived workload and focusability, and considered three scenarios:

1. Small edits (modify localized geometry).
2. Large edits (modify large scene objects).
3. Modify a scene's light sources.

They started from the initial premise that edits on small objects primarily affect a local region of the scene and it might thus be obstructive for a fluid workflow to re-render the entire image. On the other hand, light source edits mostly have a global influence and are more likely to require a full image update to convey a complete impression of their effect. As a representative of conventional, global re-rendering with denoising, they use the AI-accelerated Nvidia OptiX denoiser [2].

[31] use a modified NASA Task Load Index (NASA-TLX) [7] questionnaire to extract a participant-wise score for each of the three scenarios and each method. They test their hypotheses via ANOVA and pairwise t-tests (see Fig. 7a), as well as analyse central tendencies and effect sizes to identify trends in the pairwise differences between the scores. Additionally, they compute a favored method based on the aforementioned score via majority vote, next to directly asking participants to state their favorite method (see Fig. 7b). Moreover, they asked artists to rate the techniques based on whether they would buy them as extensions in an existing scene editing framework, to construct a metric similar to a Net Promoter Score (NPS) [22]. They also explore the reasons for not using path tracing during scene editing, by conducting structured interviews.

For the task-based evaluation, following the original formulation of Likert-typed scales [28, 31] aggregated the extended NASA-TLX questions by averaging the individual Likert items. The aggregation is computed for each combination of scenario and method on the participant level, resulting in nine scores for each participant. This procedure is also referred to as Raw TLX [4]. As the same participant ranked all of the combinations during the study and the individual ratings are thus dependent on the subject, they use paired or repeated measure methods.

The 21 participating artists ($m = 15$, $f = 4$, $d = 2$) were recruited from both industry and universities (10 professionals, 11 students). Their ages ranged from 21 to 51 (Mean *M*: 30.48, standard deviation *SD*: 7.79) and participants' experience measured in hours spent using 3D

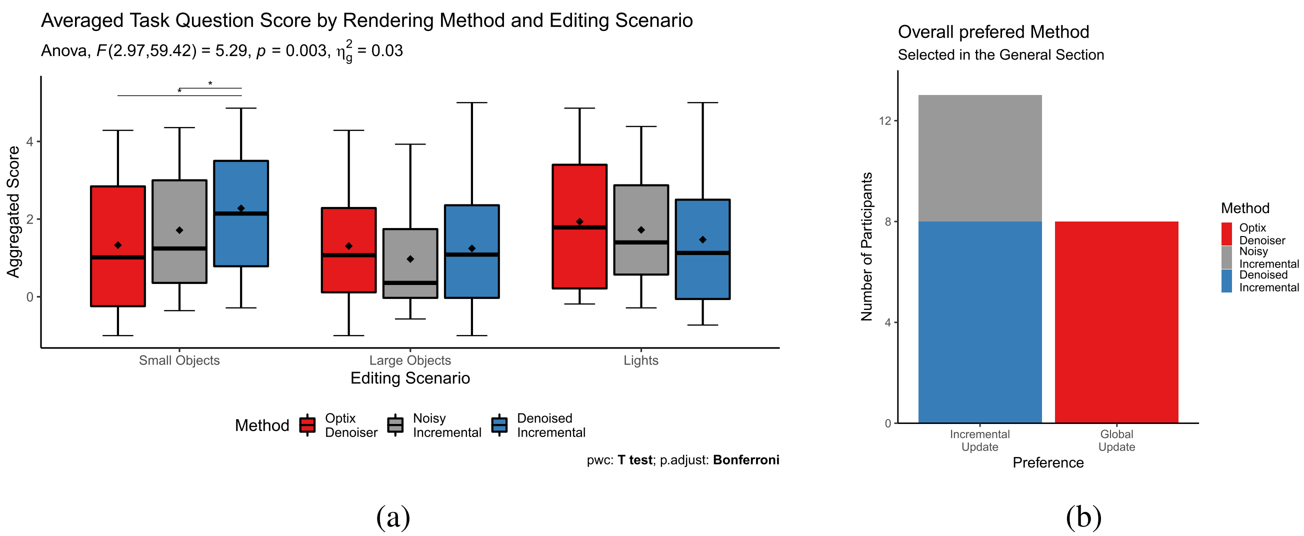


Fig. 7 **a** Box plot of the aggregated score of the task-based evaluation of the study comparing prioritized and global re-rendering methods (⧫: mean, *: $\alpha < 0.05$). **b** Bar plot showing the favorite rendering approach selected at the end of the questionnaire. Both plots taken from [31]

rendering software ranged from '50–99' to '10,000+' hours (Median *Mdn*: '1000–4999', interquartile range *IQR*: ['100–499', '5000–9999']).

ANOVA and t-Test Results: After through assumption testing and, if applicable, adaption of the data, [31] analyze the interaction between the rendering method and editing scenario on the score aggregated by participants. They performed a two-way repeated measures ANOVA with the scenario as the focal variable and the used method as the moderator variable. As the sphericity assumption is violated, they applied the conservative Greenhouse-Geisser correction. They found a statistically significant interaction between the rendering approach and editing scenario on the aggregated value ($F(2.97, 59.42) = 5.29$, $p = 0.003$, $\eta_g^2 = 0.03$).

Therefore, they analyzed the effect of the used method for each scenario, reporting p-values adjusted using the Bonferroni multiple testing correction method. Using one-way ANOVAs, the simple main effect of the rendering approach was found to be only significant for small objects ($p = 0.03$, $\eta_g^2 = 0.06$). Pairwise comparisons using paired t-tests show that for small objects, the mean aggregated value was significantly different between using the denoiser and the denoised incremental method ($p = 0.026$, $d = 0.63$) and between the noisy and denoised incremental method ($p = 0.05$, $d = 0.57$; with outlier removed $p = 0.025$, $d = 0.66$). There is no significant difference for light edits between using the denoiser and the noisy ($p = 1$, $d = 0.15$) or denoised ($p = 0.69$, $d = 0.27$; with outlier removed $p = 0.079$, $d = 0.54$) incremental rendering technique (see Fig. 7a). While the study is slightly underpowered with respect to the a-priory power analysis for pairwise t-tests (34 participants required for a power of $1 - \beta = 0.8$ with a medium effect of $d = 0.5$), we observed larger effect sizes than assumed.

Further Analysis: From the data collected by [31], we can analyze the pairwise differences between the rendering methods at the per-participant level for each scenario and compute a metric similar to an NPS based on the question if artists would buy the respective rendering approach as a feature in 3D software. Moreover, we both computed which rendering method artists preferred overall according to the task-based evaluation, and followed up with the participating artists at the end of the study for their preferred method (see Fig. 7b). Both the NPS score and preferences suggest that most participants preferred incremental over global updates. Lastly, we analyzed the optimal parameter settings for the tile size and quality.

Regression Analysis: Building on the analysis from [31], we additionally built a linear mixed-effect regression model to assess the influence of the artist's experience measured in hours and status (student or professional) on the aggregated value. As there is a significant relation between experience and status (Fisher's exact test (FET), $p = 0.014$) we model them as a fully crossed effect and use the participants' ID as a random effect. Similarly, we investigated the effect of how artists participated (parsec, own hardware, or in person) and in which order they encountered the rendering methods on the aggregated value using separate models. However, for all three models, no significant relation was found.

Audio Recordings: When analyzing the transcribed voice recordings, we detected one participant reporting minor motion sickness when working with the noisy incremental method. Another participant noted getting a slight headache from using the denoiser. Four artists recognized the denoiser as being AI-enhanced. When encountering the denoised incremental method for the first time, five participants voiced that it is the best or most seamless method so far. Three reported the same about the OptiX denoiser. Two people commented that the incremental methods felt slower, whereas one person felt that the denoiser converged slower.

Nine participants assessed the overall differences between all methods as minor or felt that none perfectly caters to their artistic needs. Most participants suggested having both globally and incrementally updating methods depending on the task at hand. One participant explicitly requested having full control over the rendering parameters additionally to an automatic system, whereas most of the other artists would have found it more comfortable if the regions being re-rendered were adjusted fully automatically.

A frequently stated reason for choosing a certain preference for the rendering approach was that the person is used to the respective method and the noise or blurriness resulting from it. Regarding the future of path tracing, the majority of artists assessed a combined approach of ray-tracing and real-time approximations like Unreal Engines Lumen or light-baking as more relevant due to the long convergence rate of path tracing. Denoisers were regarded as being useful to reduce the spp count and thus render time for the final results, but not sufficient for scene editing.

Enhanced Prioritization Methods

Building upon [31] results, we investigate the impact of our new, more sophisticated prioritization techniques for re-rendering and if user preference depends on a specific editing scenario. To this end, we compare the manual, denoised incremental re-rendering method with the automated indirection-aware and focus-based prioritizations. For the indirection-aware technique we evaluated the initially biased, but smooth transition without the spiral effect. The following three scenarios are considered:

1. Material edits (modify static geometry).
2. Translational edits (moving objects).
3. Light edits.

To evaluate the users' preferences, we instructed them to execute one task with manual and automated incremental re-rendering methods under each of the scenarios listed above. Afterwards, we let the participants rate the experienced combination of method and scenario to extract a participant-wise score. After each scenario, we also let them directly state their favorite method. While [31] conducted an extensive study with many participants, we focused on only a few experts in the field of rendering and modelling. Regardless, this setup enables us to research the preferences of users in an exploratory fashion and confirm the relative improvement provided by the new prioritization techniques.

Technical Setup and Participation

The survey was implemented as a self-hosted Drupal Webform and the full questionnaire, as well as the anonymized collected data and R code used for the evaluation, can be found online [30]. We use three different scenes from the Bitterli Rendering Resources [1]: The *Contemporary Bathroom*, the *Country Kitchen*, and *The Grey & White Room*. We refer to them simply as *Bathroom*, *Kitchen*, and *Living Room*.

The artists were asked to voice their thoughts during the study. While [31] offered hyprid participation and extensively analysed recorded audio and video using OpenAI's Whisper [24], we asked professionals to participate in person. Throughout the entire questionnaire as well as in the Falcor framework we obfuscated the names of the rendering methods using letters. We also hid all user interface (UI) elements except the needed transformational, material, and rendering parameters.

User Study Design

Before beginning the experiment, participants were asked to sign a consent form. The study itself consists of three parts: an introductory tutorial, the task-based evaluation of the three methods (denoised incremental, indirection-aware, focus-based) in three different editing scenarios (material edits, translational edits, lights), and general and demographic questions.

Tutorial: The tutorial explained the necessary elements of the UI to the participants (see Section "Prioritized Re-rendering") and allowed them to practice by executing four exemplary tasks in the *Kitchen* scene using the basic 1 spp rendering without any additional features enabled. The users were also instructed to use predefined camera viewports but were allowed to move the camera if they really needed it to not interrupt their artistic workflow. However, we ensured that they would always stay at the same distance from the object. The tutorial also provided a short text explaining the current limitations of real-time path tracing and showed two prerecorded videos of a cup moving in the *Bathroom* scene, one with a 1 spp noisy re-rendering and one with 4096 spp plus denoising with OptiX. The latter was supposed to serve as a reference for an imaginary approach that instantly converges.

Task-Based Evaluation: During the task-based evaluation, participants performed one editing task in the *Kitchen*, *Bathroom* and *Living Room* scene for each of the three editing scenarios and the three rendering methods, yielding nine combinations in total. After each condition, the users were

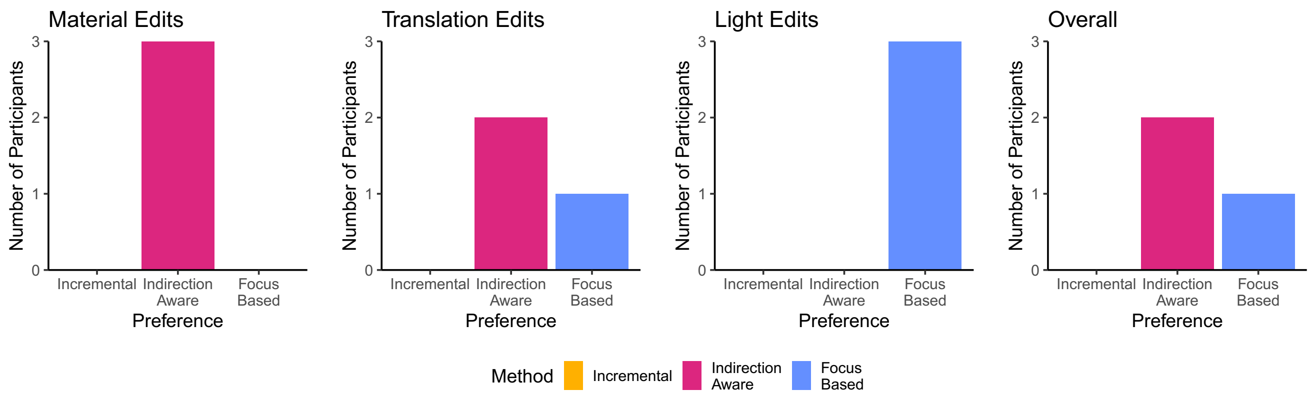


Fig. 8 User preference stated after completing each editing scenario, as well as overall

asked to rate the experienced method, and after each editing scenario, they selected their favorite out of the three methods.

To rate the experienced combination of scenario and method, we used the same questionnaire as [31]. It is based on the standardized NASA-TLX [7], which measures the perceived workload of tasks, but adds two questions to assess focus and distraction.

General and Demographic Questions: We asked participants to select an overall preferred rendering method, as well as how likely they would buy the respective techniques as a feature in a rendering tool on a scale between zero and ten to compute a metric similar to a NPS [22].

Scale Design: We adapt the same scale design (Table 1) as [31] following the construction guidlines in [28]. It is supposed to offer a better point of comparison, by using the conventional 1 spp first encountered in the tutorial, as well as an imaginary converged solution as reference points.

Results

We recruited three professional experts (m = 3, f = 0), with ages ranged from 36 to 38 (Mean *M*: 37, standard deviation *SD*: 1).

Qualitative Evaluation: The results for the explicitly stated preferences can be seen in Fig. 8. While indirection-aware is strongly preferred for material edits, focus-based prioritization is exclusively chosen for light edits. For translational edits, as well as overall, indirection-aware seems to slightly be preferred.

These results are in line with the aggregated NASA-TLX scores depicted in Fig. 9b. For material edits, indirection-aware achieves higher scores on average, while the focus-based technique is rated lowest for this scenario. For translational edits, there is no clear trend visible, whereas for light edits there seems to be a preference towards automated methods, with focus-based prioritization yielding the highest scores on average.

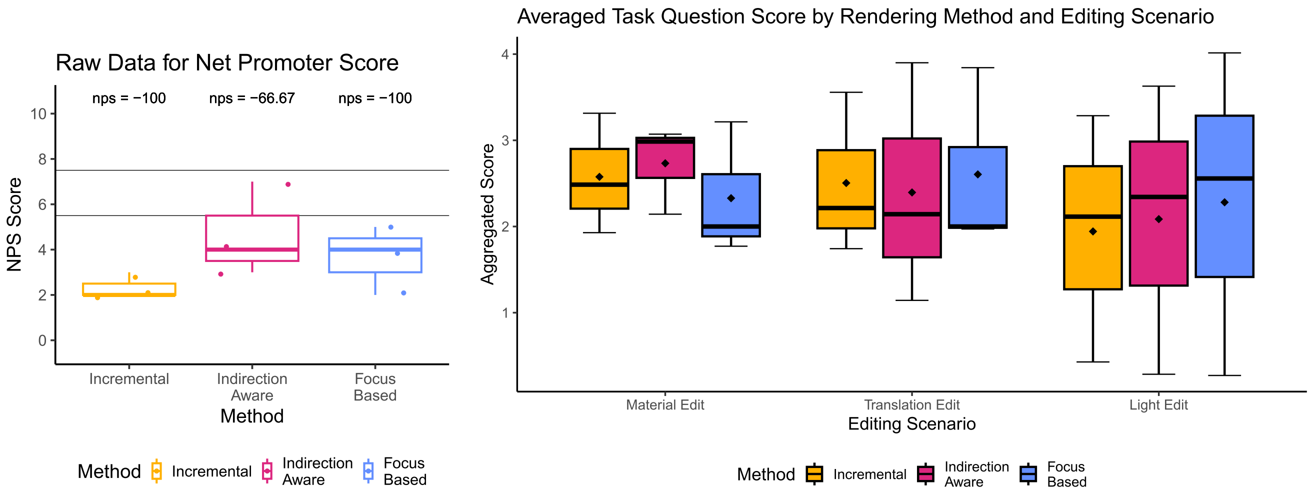
Extracted NPS scores (Fig. 9a) are all negative. While indirection-aware achieves the highest score, on average it is on par with focus-based prioritization.

Free-Form Feedback: The incremental re-rendering methods was noted to be lagging behind with longer waiting times until reflections are updated, which participants found distracting. One participant described it as “deterministic”, while all participants commented on the visible threshold borders, which could be mistaken for shadows or features, as well as the ghosting artifacts, when selecting a too high tile quality.

The indirection-aware prioritization was received as less obviously distracting and more adaptive, while it allows to focus on the edited area more and obtain relevant information faster. While overall participants praised it for the lesser delay to the final result, some noted the slightly higher delay for areas only slightly influenced due to the initially

Table 1 Format of the Likert items for the adapted scale of the extended NASA-TLX questions and their numerical mapping applied in evaluation [31]

Worse than 1spp	Same as 1spp	Slightly better than 1spp	Better, but still flawed	Mostly like converged	Almost equivalent	Same as converged
−1	0	1	2	3	4	5



(a) (b)

Fig. 9 **a** Box plot showing the distribution of ratings when asking users whether they would buy the respective rendering method as a feature in a 3D software. Based on these ratings, we computed a metric similar to a NPS, which is stated on top of the box plots. **b** Box plot of the aggregated task-wise NASA-TLX score comparing manual and automated prioritization techniques (⧫: mean)

biased, smooth transition. One participant specifically stated they preferred not having the spiral. Another participant described it as "better for real-time feedback as it allows faster iterations when testing different parameter settings".

For light edits, multiple participants noted their preference for the focus-based method, as it allows them to determine regions of interest themselves. It was described as predictable, exhibiting the least tearing, and better for focusing on shadows, which were not always detected by the indirection-aware method.

Overall, participants suggested a combination of the focus-based and indirection-aware methods or extending the focus-based technique by allowing to set three points of interest. One suggested, manually specifying whether the focus point should be static (i.e., for shadows) or move along with the object (i.e., for reflections). Two participants wished for a smoother blending of the outline of the spiral.

Discussion and Conclusion

In this work, we explored multiple adaptive, priority-based re-rendering approaches designed for interactively editing a 3D scene with MC path tracing. Using a tile-based approach introduced in [31], we can efficiently update image regions with higher quality in an incremental fashion, applying various importance metrics based on an artist's preference and use case scenario.

[31] suggest incremental updates are preferred for scene editing compared to global approaches, in particular for small objects. On the other hand, artists mentioned during a structured interview that none of the methods perfectly caters to their needs so far and they would thus still mainly fall back to less ideal real-time approximations during the editing process. However, accurate, physically-based methods to produce realistic images remain highly important and continuously gain relevance in both offline and high-performance real-time applications.

Based on these results, we introduced refined priority policies to their implementation. We replaced our initial, naive prioritization with an indirection-aware priority computation, considering both direct and indirect lighting. Thus global effects, such as shadows and reflections, are respected during the editing process. Adapting to an arbitrary object shape and size, the quality of the re-rendering is also automatically chosen such that all tiles are re-rendered at the highest possible resolution. Ghosting artifacts from the original importance metric, occurring when the user selected excessively high quality settings, resulting in a small re-rendering area, are averted as well.

Moreover, we added support for capturing the artist's focus position in the importance metric. In particular, global editing scenarios such as light edits will benefit from this addition, as these changes affect the rendering rather uniformly. Using either eye tracking or the cursor position, we can determine which particular area the artist is interested in during the edit and focus our rendering budget on this part of the image, allowing for a user-guided re-rendering. To counteract motion sickness problems, we also added a smoother transition after the most important scene edits, which updates areas of lower importance in the background.

All methods suffer from errors resulting from the used denoiser. For example, there is less error in uniformly lit, large, planar regions, but higher error in shadowed or reflective regions, where the G-buffer information is less helpful in guiding the denoiser. However, as can be seen in Fig. 1, the incremental methods reduce the overall error global methods suffer from. While the naive priority metric still results in higher errors in reflective areas, the indirection-aware and focus-based policies manage to reduce this error – at the cost of slightly less quality in the changed image regions, as a larger number of tiles has to be re-rendered within the same budget.

Future work could extend the building blocks of the incremental methods by combining them with higher-quality denoisers, such as Intel's OIDN [8]. The priority policy could be even further refined by integrating gradient-based techniques and difference images [23]. Different priority metrics could be combined and indirectly hit or non-textured areas sampled with an even higher number of samples, to counteract the dependence of neural denoisers on G-buffer information. Overall, our evaluation provides a strong motivation to pursue further methods for automatic, priority-based, and interactive re-rendering techniques for creative processes.

Acknowledgements We thank our students Pascal Hann and Nina Semmelrath for their contributions.

Funding Open access funding provided by TU Wien (TUW). This research was funded by FWF project F77 (SFB Advanced Computational Design SP4) and by WWTF project ICT22-055 (Instant Visualization and Interaction for Large Point Clouds).

Data availability The data used for the evaluation of surveys is available at [30].

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

References

1. Bitterli B. Rendering resources; 2016. https://benedikt-bitterli.me/resources/. Accessed 1Aug 2024
2. Chaitanya CRA, Kaplanyan AS, Schied C, Salvi M, Lefohn A, Nowrouzezahrai D, Aila T. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. ACM Trans Graph. 2017. https://doi.org/10.1145/3072959.3073601.
3. Günther T, Grosch T. Consistent scene editing by progressive difference images. Comput Graph Forum. 2015. https://doi.org/10.1111/cgf.12677.
4. Hart S. Nasa-task load index (nasa-tlx); 20 years later, vol. 50. 2006. https://doi.org/10.1177/154193120605000909.
5. Hachisuka T, Jarosz W, Weistroffer RP, Dale K, Humphreys G, Zwicker M, Jensen HW. Multidimensional adaptive sampling and reconstruction for ray tracing. ACM Trans Graph. 2008;27(3):1–10. https://doi.org/10.1145/1360612.1360632.
6. Hasselgren J, Munkberg J, Salvi M, Patney A, Lefohn A. Neural temporal adaptive sampling and denoising. Comput Graph Forum. 2020;39:147–55. https://doi.org/10.1111/cgf.13919.
7. Hart SG, Staveland LE. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In: Human Mental Workload. Advances in Psychology, 1988;52: 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9.
8. Intel: Open Image Denoise; 2019. https://www.openimagedenoise.org/. Accessed 1 Aug 2024
9. Kajiya JT. The rendering equation. SIGGRAPH Comput Graph. 1986;20(4):143–50. https://doi.org/10.1145/15886.15902.
10. Kallweit S, Clarberg P, Kolb C, Davidovič T, Yao K-H, Foley T, He Y, Wu L, Chen L, Akenine-Möller T, Wyman C, Crassin C, Benty N. The Falcor Rendering Framework; 2022. https://github.com/NVIDIAGameWorks/Falcor. Accessed 1 Aug 2024
11. Kuznetsov A, Kalantari NK, Ramamoorthi R. Deep adaptive sampling for low sample count rendering. Comput Graph Forum. 2018;37:35–44.
12. Murakami K, Hirota K. In: Bouatouch, K., Bouville, C. (eds.) Incremental ray tracing, pp. 1992:17–32 . https://doi.org/10.1007/978-3-662-09287-3_2.
13. Mitchell DP. Generating antialiased images at low sampling densities. SIGGRAPH Comput Graph. 1987;21(4):65–72. https://doi.org/10.1145/37402.37410.
14. Misso Z, Li YK, Burley B, Teece D, Jarosz W. Progressive null-tracking for volumetric rendering. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23. Association for Computing Machinery, New York, NY, USA; 2023. https://doi.org/10.1145/3588432.3591557.
15. Myrodia V. Psychophysical studies on Monte Carlo rendering-noise visual perception. PhD thesis (2021). https://theses.hal.science/tel-03716128. Accessed 1 Aug 2024
16. Nvidia: NVIDIA TURING GPU ARCHITECTURE. Technical report (2018)
17. NVIDIA: Nvidia Real-Time Denoisers. 2021. https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers. Accessed 1 Aug 2024
18. Overbeck RS, Donner C, Ramamoorthi R. Adaptive wavelet rendering. ACM Trans Graph. 2009;28(5):1–12. https://doi.org/10.1145/1618452.1618486.
19. Ponomarenko N, Ieremeiev O, Lukin V, Egiazarian K, Carli M. Modified image visual quality metrics for contrast change and mean shift accounting. In: The Experience of Designing and Application of CAD Systems in Microelectronics, 2011:305–311
20. Pharr M, Jakob W, Humphreys G. Physically based rendering: from theory to implementation, 3rd edn., 2016:1266.
21. Ponomarenko N, Silvestri F, Egiazarian K, Carli M, Astola J, Lukin V. On between-coefficient contrast masking of dct basis

functions. Proc. of the 3rd Int. Workshop on Video Processing and Quality Metrics for Consumer Electronics; 2007.
22. Reichheld F. The one number you need to grow. Harv Bus Rev. 2003;81:46–54.
23. Rousselle F, Jarosz W, Novák J. Image-space control variates for rendering. ACM Trans. Graph. 2016;35(6), https://doi.org/10.1145/2980179.2982443.
24. Radford A, Kim JW, Xu T, Brockman G, McLeavey C, Sutskever I. Robust speech recognition via large-scale weak supervision. arXiv (2022). https://doi.org/10.48550/ARXIV.2212.04356.
25. Sakai H, Freude C, Auzinger T, Hahn D, Wimmer M. A statistical approach to monte carlo denoising. In: SA '24: SIGGRAPH Asia 2024 Conference Papers, p. 11. Association for Computing Machinery, New York, NY, USA, 2024. https://doi.org/10.1145/3680528.3687591. https://www.cg.tuwien.ac.at/research/publications/2024/sakai-2024-asa/. Accessed 1 Aug 2024
26. Schied C, Peters C, Dachsbacher C. Gradient estimation for real-time adaptive temporal filtering. Proc ACM Comput Graph Interact Tech. 2018. https://doi.org/10.1145/3233301.
27. Schmidt T-W, Pellacini F, Nowrouzezahrai D, Jarosz W, Dachsbacher C. State of the art in artistic editing of appearance, lighting, and material. Comput Graph Forum. 2014. https://doi.org/10.1111/cgf.12721.
28. South L, Saffo D, Vitek O, Dunne C, Borkin MA. Effective use of likert scales in visualization evaluations: a systematic review. CGF. 2022. https://doi.org/10.1111/cgf.14521.
29. Thomas MM, Liktor G, Peters C, Kim S, Vaidyanathan K, Forbes AG. Temporally stable real-time joint neural denoising and supersampling. Proc ACM Comput Graph Interact Tech. 2022. https://doi.org/10.1145/3543870.
30. Ulschmid A, Kerbl B, Krösl K, Wimmer M. https://github.com/cg-tuwien/Prioritized-ReRendering, 2023.
31. Ulschmid A, Kerbl B, Krösl K, Wimmer M. Real-time editing of path-traced scenes with prioritized re-rendering. In: Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - GRAPP, 2024:46–57. https://doi.org/10.5220/0012324600003660. INSTICC
32. Wyman C, Panteleev A. Rearchitecting spatiotemporal resampling for production. In: High-Performance Graphics - Symposium Papers. The Eurographics Association, Goslar, DEU; 2021. https://doi.org/10.2312/hpg.20211281.
33. Xu B, Li T, Georgiev I, Hedstrom T, Ramamoorthi R. Residual path integrals for re-rendering. Comput Graph Forum. 2024. https://doi.org/10.1111/cgf.15152.
34. Zhdan D. Reblur: A hierarchical recurrent denoiser. In: Ray Tracing Gems II, 2021:823–844. https://doi.org/10.1007/978-1-4842-7185-8_49.
35. Zwicker M, Jarosz W, Lehtinen J, Moon B, Ramamoorthi R, Rousselle F, Sen P, Soler C, Yoon S-E. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. Comput Graph Forum. 2015;34(2):667–81.